

Enhanced Unsatisfiable Cores for QBF: Weakening Universal to Existential Quantifiers

(extended version with all appendices; git: 3849414; compiled: 2020-06-18 15:23:57+02:00)

Viktor Schuppan

Email: Viktor.Schuppan@gmx.de, URL: <http://schuppan.de/viktor/>

Abstract—We propose an enhanced notion of unsatisfiable cores for QBF in prenex CNF that weakens universal to existential quantifiers in addition to the traditional removal of clauses. We can thus obtain unsatisfiable cores that are semantically different from those obtained by the traditional notion; this gives rise to explanations — and, via hitting set duality, diagnoses — of unsatisfiability that are not provided by traditional unsatisfiable cores. We use a source-to-source transformation on QBF that reduces the weakening of universal to existential quantifiers to the removal of clauses. This enables any tool or method that can compute unsatisfiable cores of the traditional notion to also compute unsatisfiable cores of our enhanced notion. We implement our approach in the QBF solver `DepQBF`, and we experimentally evaluate it on a subset of `QBFLIB`. Several case studies illustrate that interesting information can be learned from our enhanced notion of unsatisfiable cores.

Index Terms—QBF, unsatisfiable cores, quantifier weakening

I. INTRODUCTION

a) Motivation and Contributions: Many important problems can be naturally encoded as quantified Boolean formulas (QBF), e.g., two-player games (e.g., [GR03,AGS05]), variants of planning (e.g., [Rin99,Tur02]), satisfiability of modal logic K [PV03], and several problems in knowledge representation (e.g., [EETW00]) and formal methods (e.g., [AB00,SB01]); for a more extensive list see [GMN09]. Unsatisfiable cores have been established as a fundamental concept in applied logic with significant applications in AI and formal methods. For example, unsatisfiable cores are commonly taken to represent causes of and serve as explanations of unsatisfiability in various logics (e.g., [CD91,BS01,SC03,SSJ+03,YM05,Sch12]), and they are used as building blocks to obtain advanced explanations of unsatisfiability (e.g., [KLM06]) and to diagnose (e.g., [Rei87]) and repair (e.g., [Sch05]) unsatisfiability. Existing work on unsatisfiable cores for QBF in prenex conjunctive normal form (PCNF) weakens formulas by removing clauses [YM05,KZ06,IJM13,LE15].

In this paper we propose an enhanced notion of unsatisfiable cores for QBF in PCNF that, in addition to removing clauses, weakens universal to existential quantifiers (Section III). Our enhanced notion of unsatisfiable cores can represent causes and lead to explanations of unsatisfiability that are different from any one that can be obtained from an unsatisfiable core of the traditional notion (Section IV). Moreover, via the well-known hitting set duality (e.g., [Rei87]; for a generic formulation see [Sla14]), this induces diagnoses and repairs for unsatisfiability that cannot be obtained when using the

traditional notion of unsatisfiable cores. On a less rigorous, but nevertheless practically relevant note, an unsatisfiable core, in which the set of quantifiers that has been weakened from universal to existential has some unexpected characteristics, may provide the initial “hunch” to the user that something may not be quite right in the QBF under consideration. In Section V we show that if in an unsatisfiable QBF in PCNF no clause can be removed without making the result satisfiable, then also no universal quantifier can be weakened to an existential one without losing unsatisfiability. Then we extend the PSPACE-completeness result for minimally unsatisfiable cores of the traditional notion [KZ06] to our enhanced notion (Section VI). We describe a transformation of QBF in PCNF such that weakening of universal to existential quantifiers can be performed by removing clauses in the transformed formula (Sections VII,VIII). That allows to obtain unsatisfiable cores in our enhanced notion by first applying the transformation, then using existing tools and methods to compute an unsatisfiable core by removing clauses, and finally mapping back the result to an unsatisfiable core in the enhanced notion. Next we provide some hints on how to interpret unsatisfiable cores (Section IX), and we classify universal quantifications into non-trivially, trivially, and not \forall -to- \exists reducible (Section X). We implement our approach in `DepQBF` [LE17] (Section XI), and we experimentally evaluate it on a subset of `QBFLIB` [GNPT] (Section XIII). Using a number of case studies including two-player games [GR03], conformant planning [Rin07], and satisfiability of modal logic K [PV03] we illustrate that interesting information can be learned from our enhanced notion of unsatisfiable cores (Section XII). Our experiments show that on instances from `QBFLIB` unsatisfiable cores of our enhanced notion can be computed and that indeed universal quantifications are weakened to existential ones.

b) Related Work: Previous work on unsatisfiable cores for QBF in PCNF uses the traditional notion of removal of clauses from the matrix [YM05,KZ06,IJM13,LE15].

Weakening universal to existential quantifiers has been called “quantifier abstraction” in a work on failed literal detection for QBF [LB11] and “existential abstraction” in the context of generalizing Q-resolution [LES16].

`QBFD` [BLB10,qbfd] allows quantifier manipulations when minimizing failure-inducing input.

[RSMB14], which is probably the most closely related work, introduces the concept of soft variables, which are variables that may be placed at different positions of the prefix

of a QBF subject to a preference function. The authors then define the optimization problem of finding a placement for the soft variables that maximizes the preference function while maintaining satisfiability of the resulting QBF. They use a transformation, which can be seen as a generalized version of our transformation in Section VII, to reduce their problem to a weighted partial MaxQBF problem (e.g., [CFLS93]). (We discovered our transformation independently.) They implement their approach in `quantom` [RPSB12]. Our work differs from [RSMB14] as follows. When seen as a specification language for sets of prefixes of QBF the notion of soft variables in Definition 1–3 of [RSMB14] is more powerful than our notion of cores in Definition 1. [RSMB14] searches for a still satisfiable result, while we search for a still unsatisfiable result. While the two are related via hitting set duality, the approaches are complementary, and often one is used as part of a method to obtain the other ([IJM13] is an example). In [RSMB14] the authors make no connection to unsatisfiable cores. [RSMB14] uses a MaxSAT-based algorithm [ZSM03]; we use a standard algorithm to obtain (optionally) minimal clausal unsatisfiable cores (e.g., [Mar12]). [RSMB14] finds a maximum solution, while we (optionally) find a minimal solution. [RSMB14] leaves the matrix unchanged, whereas we (optionally) also weaken the matrix. As a minor practical point, our approach does not require to enhance a QBF in PCNF with additional information, thus making a large set of benchmarks directly available.

When debugging unsatisfiable Alloy models Shlyakhter et al. [SSJ⁺03] point out which values of bound variables are irrelevant to the unsatisfiability. For a Boolean variable p in some formula $\forall p. f[p]$ this corresponds to weakening $f[\perp/p] \wedge f[\top/p]$ to $f[\perp/p]$ or to $f[\top/p]$ — which can be achieved by removing clauses with occurrences of p of the suitable polarity and, hence, by the traditional notion —, whereas we can additionally weaken to $f[\perp/p] \vee f[\top/p]$.

Finally, our work is in the spirit of efforts investigating the aspect of granularity in various notions including: unsatisfiable cores for propositional logic (e.g., [KLM06, Sch16b]), temporal logic (e.g., [Sch12, Sch16a]), and constraint programming (e.g., [GMP07]); equivalent formulas (e.g., [GW11]); unrealizable cores (e.g., [Sch12]); vacuity (e.g., [AFF⁺03, GC04]); justifications (e.g., [KPG06, LPSV06, HPS08]); diagnoses (e.g., [PQ13]); and repair (e.g., [KPSG06, DQF14]). For a uniform treatment of some such notions and their relationships see the work on minimal sets over monotone predicates [MJ14].

II. PRELIMINARIES

We consider QBF in PCNF (e.g., [KB09, GMN09]); any QBF can be transformed into an equivalent QBF in PCNF (e.g., [KB09]).

We assume a set of variables V ; variables are denoted by the letter p . The Boolean constants are \perp (false) and \top (true). Literals are variables, Boolean constants, or their negations, denoted \neg ; we write literals as the letter l . A clause $(l_1 \vee \dots \vee l_n)$ is a disjunction of literals, denoted by the letter c . In clauses we use implication \rightarrow as an abbreviation as usual.

A conjunctive normal form (CNF) formula $c_1 \wedge \dots \wedge c_n$ is a conjunction of clauses; CNF formulas are denoted by the letter C . When convenient we view clauses as sets of literals and CNF formulas as sets of clauses. A variable p is *pure* in a CNF formula C , if it occurs only non-negated or only negated in C . $\mathbb{B} = \{0, 1\}$ is the set of Booleans. An assignment v for C is a mapping from V to \mathbb{B} . A literal l evaluates to 1 under v iff $l = \top$, $l = \neg \perp$, $l = p$ and $v(p) = 1$, or $l = \neg p$ and $v(p) = 0$. A clause c evaluates to 1 under v iff one or more of its literals evaluate to 1 under v . The empty clause evaluates to 0. A CNF formula C evaluates to 1 under v iff all of its clauses evaluate to 1 under v . The empty CNF formula evaluates to 1. A CNF formula C is *satisfiable* if there exists an assignment v such that C evaluates to 1 under v ; otherwise, it is unsatisfiable.

\forall and \exists denote *universal* and *existential quantifiers*, respectively. We use the letter Q to represent quantifiers. Let $Q_1, \dots, Q_n \in \{\forall, \exists\}$ be quantifiers, let $p_1, \dots, p_n \in V$ be pairwise different variables, and let C be a CNF formula whose variables are contained in p_1, \dots, p_n . Then $Q_1 p_1 \dots Q_n p_n. C$ is a QBF in PCNF with prefix $Q_1 p_1 \dots Q_n p_n$ and matrix C . Prefixes are written as the letter Π . The *alternation depth* of a QBF in PCNF is one plus the number of alternations between \forall and \exists in the prefix. If $\Pi.C$ is a QBF in PCNF and $p \in V$, then $(\Pi.C)[\perp/p]$ (resp. $(\Pi.C)[\top/p]$) denotes the QBF in PCNF that is obtained from $\Pi.C$ by replacing every occurrence of p in C with \perp (resp. \top). Satisfiability of a QBF in PCNF is then defined as follows. $\forall p \Pi.C$ is satisfiable iff $(\Pi.C)[\perp/p]$ and $(\Pi.C)[\top/p]$ are satisfiable. $\exists p \Pi.C$ is satisfiable iff $(\Pi.C)[\perp/p]$ or $(\Pi.C)[\top/p]$ are satisfiable. Deciding the satisfiability of a QBF in PCNF is PSPACE-complete [SM73]; the satisfiability problems for QBF in PCNF with alternation depth at most $i \in \mathbb{N}$ and either \forall or \exists as the first quantifier yield complete problems for the i -th level of the polynomial hierarchy Π_i^P and Σ_i^P , respectively [Sto76, Wra76].

III. ENHANCED UNSATISFIABLE CORES FOR QBF

In this section we add to the traditional notion of cores for QBF in PCNF (henceforth called *c*-cores), which are obtained by removing clauses, the notions of *q*-cores, which are obtained by weakening universal to existential quantifiers, and of *qc*-cores, which combine *c*-cores and *q*-cores. In Definition 1 we characterize *c*-, *q*-, and *qc*-cores. In Definitions 2 and 3 we state natural extensions of proper cores and unsatisfiable cores to *q*- and *qc*-cores. In Definition 4, we introduce quantifier-minimally unsatisfiable cores in addition to the traditional clause-minimally unsatisfiable cores. Let $\Pi.C$ be a QBF in PCNF.

Definition 1 (Core):

- 1) Let $C' \subseteq C$. Then $\Pi.C'$ is a *c-core* of $\Pi.C$.
- 2) Let $\Pi = Q_1 p_1 \dots Q_n p_n$, $\Pi' = Q'_1 p_1 \dots Q'_n p_n$ be prefixes such that, $\forall 1 \leq i \leq n$: if Q_i is \exists , then Q'_i is \exists ; otherwise, $Q'_i \in \{\forall, \exists\}$. Then $\Pi'.C$ is a *q-core* of $\Pi.C$.
- 3) Let $\Pi.C'$ be a *c-core* of $\Pi.C$, and let $\Pi'.C'$ be a *q-core* of $\Pi.C'$. Then $\Pi'.C'$ is a *qc-core* of $\Pi.C$.

Some authors (e.g., [LE15]) remove quantifications from the prefix of a c-core if the quantified variables cease to occur in the matrix of the c-core. In our implementation we do this as a generic postprocessing step and, therefore, we opt to keep our exposition simple and omit this step from Definition 1.

Definition 2 (Proper Core): Let $\Pi'.C'$ be a qc-core (resp. c-core, q-core) of $\Pi.C$ such that $\Pi' \neq \Pi$ or $C' \neq C$. Then $\Pi'.C'$ is a *proper qc-core* (resp. proper c-core, q-core) of $\Pi.C$.

Definition 3 (Unsatisfiable Core): Let $\Pi'.C'$ be a qc-core (resp. c-core, q-core) of $\Pi.C$ such that $\Pi'.C'$ is unsatisfiable. Then $\Pi'.C'$ is an *unsatisfiable qc-core* (resp. unsatisfiable c-core, q-core) of $\Pi.C$.

If $\Pi'.C'$ is an unsatisfiable c-core, q-core, or qc-core of $\Pi.C$, then $\Pi.C$ is unsatisfiable.

Definition 4 (Minimal Unsatisfiability): Let $\Pi.C$ be unsatisfiable such that there is no proper unsatisfiable c-core (resp. q-core) of $\Pi.C$. Then $\Pi.C$ is *c-minimally unsatisfiable* (resp. q-minimally unsatisfiable).

Example 1: As an example consider $\Pi.C = \forall p.(p) \wedge (\neg p)$. Clearly, $\Pi.C$ is unsatisfiable. $\Pi.C$ has four c-cores $\Pi.C$, $\forall p.(p)$, $\forall p.(\neg p)$, and $\forall p.\top$. The first three are unsatisfiable c-cores, the last three are proper c-cores, and the second and third are both q- and c-minimally unsatisfiable.

$\Pi.C$ has two q-cores $\Pi.C$ and $\exists p.(p) \wedge (\neg p)$, both of which are unsatisfiable. Only $\exists p.(p) \wedge (\neg p)$ is a proper q-core and q-minimally unsatisfiable; it is also c-minimally unsatisfiable.

Any c-core or q-core is also a qc-core. $\exists p.(p)$, $\exists p.(\neg p)$, and $\exists p.\top$ are the only qc-cores of $\Pi.C$ that are both proper c-cores and proper q-cores of $\Pi.C$. However, none of them is unsatisfiable. \square

IV. QC-CORES CAN BE DIFFERENT FROM C-CORES

Unsatisfiable cores are commonly taken to be causes and/or explanations of unsatisfiability (e.g., [CD91,BS01,SC03,SSJ⁺03,YM05,Sch12]). Some authors prefer minimally or minimum cardinality unsatisfiable cores (e.g., [CD91,LS04,SC03,TCJ08]), and some authors use unsatisfiable cores as building blocks of more advanced explanations (e.g., [KLM06]). In this paper we take the view that a minimally unsatisfiable core represents a cause of unsatisfiability and gives rise to an explanation of unsatisfiability. We now argue that our enhanced notion of unsatisfiable qc-cores for QBF in PCNF can identify additional causes of unsatisfiability (giving rise to additional explanations of unsatisfiability) that are indeed different from the ones identified by the traditional notion of unsatisfiable c-cores.

We consider $\forall p.(p) \wedge (\neg p)$ from Example 1 with q- and c-minimally unsatisfiable qc-cores $\forall p.(p)$, $\forall p.(\neg p)$, and $\exists p.(p) \wedge (\neg p)$. Clearly, the q-core $\exists p.(p) \wedge (\neg p)$ is syntactically different from the c-cores $\forall p.(p)$ and $\forall p.(\neg p)$. However, in general, syntactic differences may carry little meaning; we therefore proceed to discuss differences based on semantics.

One semantics for unsatisfiable QBF is given by tree refutations [Gel12,CFL⁺06]. A tree refutation for an unsatisfiable QBF $\Pi.C$ is a tree such that

- 1) its non-leaf nodes are labeled with variables in Π (the labeling of leaf nodes is irrelevant);
- 2) its edges are labeled with Booleans (representing assignments to the variables that are labeling their source nodes);
- 3) every node labeled with a universally quantified variable has one outgoing edge labeled with either 0 or 1;
- 4) every node labeled with an existentially quantified variable has two outgoing edges labeled with 0 and 1, respectively;
- 5) on every path from the root to a leaf node the sequence of labels on the non-leaf nodes is identical to the sequence of variables given by the prefix Π ; and
- 6) on every path from the root to a leaf node the induced assignment to the variables in Π falsifies C .

Intuitively, a tree refutation shows how to choose the assignment to the universally quantified variables in order to falsify $\Pi.C$.

$\forall p.(p)$ has one tree refutation with the root node labeled p and its single outgoing edge labeled 0. $\exists p.(p) \wedge (\neg p)$ has one tree refutation with the root node labeled p and two outgoing edges labeled 0 and 1. Clearly, the tree refutation for $\exists p.(p) \wedge (\neg p)$ differs from the one for $\forall p.(p)$. The two tree refutations correspond to different ways to explain the unsatisfiability of $\forall p.(p) \wedge (\neg p)$: for $\forall p.(p)$ assigning 0 to p falsifies (p) ; for $\exists p.(p) \wedge (\neg p)$ each assignment to p falsifies one of the clauses (p) and $(\neg p)$. The case of $\forall p.(\neg p)$ is analogous.

Let C_1, C_2 be two different matrices that have the same sets of satisfying assignments. For any prefix Π such that $\Pi.C_1$ and $\Pi.C_2$ are unsatisfiable, the sets of tree refutations for $\Pi.C_1$ and $\Pi.C_2$ are identical. I.e., tree refutations are not always sufficient to distinguish unsatisfiable cores. In that case we may turn to proof-theoretic semantics, which can be more discriminating [Fra14]. We can, for example, assign to unsatisfiable QBFs in PCNF the sets of their Q-resolution proofs of unsatisfiability [KKF95] and then compare unsatisfiable cores in terms of their sets of Q-resolution proofs. Q-resolution essentially allows for two operations (we omit some details and assume a working knowledge of resolution): (i) resolve two clauses on an existentially quantified literal; and (ii) remove a universally quantified literal l from a clause c if there is no existentially quantified literal in c that occurs to the right of the variable of l in the prefix. Then a QBF in PCNF is unsatisfiable iff the empty clause can be derived via Q-resolution [KKF95].

$\forall p.(p)$ (resp. $\forall p.(\neg p)$) is proved unsatisfiable by removing p (resp. $\neg p$) from (p) (resp. $(\neg p)$), whereas $\exists p.(p) \wedge (\neg p)$ is proved unsatisfiable by resolving (p) with $(\neg p)$. Again, the set of proofs for $\exists p.(p) \wedge (\neg p)$ is different from the sets of proofs for $\forall p.(p)$ and $\forall p.(\neg p)$.

V. C-MINIMAL UNSATISFIABILITY IMPLIES Q-MINIMAL UNSATISFIABILITY

In this section we show that any c-minimally unsatisfiable core is also q-minimally unsatisfiable.

Theorem 1: Let $\Pi.C$ be a c-minimally unsatisfiable QBF in PCNF such that every universally quantified variable in Π occurs in some clause in C . Then $\Pi.C$ is also q-minimally unsatisfiable. The converse is not true.

Proof: The first part is an immediate consequence of the following Lemma 1. An example that the converse is not true is $\Pi.C = \exists p \exists p' . (p) \wedge (\neg p) \wedge (p')$. $\Pi.C$ is clearly q-minimally unsatisfiable; however, removing (p') from C results in a proper c-core of $\Pi.C$, i.e., $\Pi.C$ is not c-minimally unsatisfiable. This concludes the proof. ■

Lemma 1: Let

$$\Pi.C = Q_1 p_1 \dots Q_{l-1} p_{l-1} \forall p_l Q_{l+1} p_{l+1} \dots Q_n p_n . \\ c_1 \wedge \dots \wedge c_{i-1} \wedge c_i \wedge c_{i+1} \wedge \dots \wedge c_m$$

be a QBF in PCNF such that p_l occurs in c_i , let $\Pi'.C'$ be obtained from $\Pi.C$ by changing $\forall p_l$ to $\exists p_l$ in Π , and let $\Pi''.C''$ be obtained from $\Pi.C$ by removing c_i from C . If $\Pi'.C'$ is unsatisfiable, then so is $\Pi''.C''$.

Proof: By induction over l . For the base case let $l-1 = 0$. By assumption

$$\Pi'.C' = \exists p_l Q_{l+1} p_{l+1} \dots Q_n p_n . \\ c_1 \wedge \dots \wedge c_{i-1} \wedge c_i \wedge c_{i+1} \wedge \dots \wedge c_m$$

is unsatisfiable. Expanding $\exists p_l$ gives unsatisfiability of both

$$(Q_{l+1} p_{l+1} \dots Q_n p_n . c_1 \wedge \dots \wedge c_{i-1} \wedge c_i \wedge c_{i+1} \wedge \dots \wedge c_m) [\perp / p_l],$$

and

$$(Q_{l+1} p_{l+1} \dots Q_n p_n . c_1 \wedge \dots \wedge c_{i-1} \wedge c_i \wedge c_{i+1} \wedge \dots \wedge c_m) [\top / p_l].$$

Without limitation of generality let p_l occur non-negated in c_i . Hence,

$$(Q_{l+1} p_{l+1} \dots Q_n p_n . c_1 \wedge \dots \wedge c_{i-1} \wedge c_{i+1} \wedge \dots \wedge c_m) [\top / p_l]$$

is also unsatisfiable. Finally, by the definition of \forall ,

$$\forall p_l Q_{l+1} p_{l+1} \dots Q_n p_n . c_1 \wedge \dots \wedge c_{i-1} \wedge c_{i+1} \wedge \dots \wedge c_m$$

is unsatisfiable as desired.

For the inductive case let $l-1 > 0$. First let $Q_1 = \exists$. By assumption

$$\Pi'.C' = \exists p_1 Q_2 p_2 \dots Q_{l-1} p_{l-1} \exists p_l Q_{l+1} p_{l+1} \dots Q_n p_n . \\ c_1 \wedge \dots \wedge c_{i-1} \wedge c_i \wedge c_{i+1} \wedge \dots \wedge c_m$$

is unsatisfiable. Expanding $\exists p_1$ gives unsatisfiability of both

$$(Q_2 p_2 \dots Q_{l-1} p_{l-1} \exists p_l Q_{l+1} p_{l+1} \dots Q_n p_n . \\ c_1 \wedge \dots \wedge c_{i-1} \wedge c_i \wedge c_{i+1} \wedge \dots \wedge c_m) [\perp / p_1],$$

and

$$(Q_2 p_2 \dots Q_{l-1} p_{l-1} \exists p_l Q_{l+1} p_{l+1} \dots Q_n p_n . \\ c_1 \wedge \dots \wedge c_{i-1} \wedge c_i \wedge c_{i+1} \wedge \dots \wedge c_m) [\top / p_1].$$

With the inductive assumption both

$$(Q_2 p_2 \dots Q_{l-1} p_{l-1} \forall p_l Q_{l+1} p_{l+1} \dots Q_n p_n . \\ c_1 \wedge \dots \wedge c_{i-1} \wedge c_{i+1} \wedge \dots \wedge c_m) [\perp / p_1],$$

and

$$(Q_2 p_2 \dots Q_{l-1} p_{l-1} \forall p_l Q_{l+1} p_{l+1} \dots Q_n p_n . \\ c_1 \wedge \dots \wedge c_{i-1} \wedge c_{i+1} \wedge \dots \wedge c_m) [\top / p_1]$$

are unsatisfiable as well. Finally, by the definition of \exists ,

$$\exists p_1 Q_2 p_2 \dots Q_{l-1} p_{l-1} \forall p_l Q_{l+1} p_{l+1} \dots Q_n p_n . \\ c_1 \wedge \dots \wedge c_{i-1} \wedge c_{i+1} \wedge \dots \wedge c_m$$

is unsatisfiable as desired.

Now let $Q_1 = \forall$. By assumption

$$\Pi'.C' = \forall p_1 Q_2 p_2 \dots Q_{l-1} p_{l-1} \exists p_l Q_{l+1} p_{l+1} \dots Q_n p_n . \\ c_1 \wedge \dots \wedge c_{i-1} \wedge c_i \wedge c_{i+1} \wedge \dots \wedge c_m$$

is unsatisfiable. Expanding $\forall p_1$ gives unsatisfiability of

$$(Q_2 p_2 \dots Q_{l-1} p_{l-1} \exists p_l Q_{l+1} p_{l+1} \dots Q_n p_n . \\ c_1 \wedge \dots \wedge c_{i-1} \wedge c_i \wedge c_{i+1} \wedge \dots \wedge c_m) [\perp / p_1]$$

or

$$(Q_2 p_2 \dots Q_{l-1} p_{l-1} \exists p_l Q_{l+1} p_{l+1} \dots Q_n p_n . \\ c_1 \wedge \dots \wedge c_{i-1} \wedge c_i \wedge c_{i+1} \wedge \dots \wedge c_m) [\top / p_1].$$

Without limitation of generality let the first part \perp / p_1 be unsatisfiable. Hence, with the inductive assumption,

$$(Q_2 p_2 \dots Q_{l-1} p_{l-1} \forall p_l Q_{l+1} p_{l+1} \dots Q_n p_n . \\ c_1 \wedge \dots \wedge c_{i-1} \wedge c_{i+1} \wedge \dots \wedge c_m) [\perp / p_1]$$

is unsatisfiable. Finally, by the definition of \forall ,

$$\forall p_1 Q_2 p_2 \dots Q_{l-1} p_{l-1} \forall p_l Q_{l+1} p_{l+1} \dots Q_n p_n . \\ c_1 \wedge \dots \wedge c_{i-1} \wedge c_{i+1} \wedge \dots \wedge c_m$$

is unsatisfiable as desired. This concludes the proof. ■

One might think that Theorem 1 would cast doubt on the usefulness of q- or qc-cores, as it shows that essentially any c-minimally unsatisfiable c-core is also a q-minimally unsatisfiable c-core (and qc-core). However, as shown in Section IV, qc-cores can represent causes of unsatisfiability of a formula (and give rise to corresponding explanations) that none of the c-cores represents.

VI. COMPLEXITY

Let CMF, QMF, and QCMF denote the sets of c-minimally unsatisfiable QBF in PCNF, q-minimally unsatisfiable QBF in PCNF, and $\text{CMF} \cap \text{QMF}$, respectively. CMF has been shown to be PSPACE-complete in [KZ06]. In this section we extend this result to QMF and QCMF.

Theorem 2: QMF and QCMF are PSPACE-complete.

Proof: Membership of QMF and QCMF in PSPACE is obvious. For PSPACE-hardness of QMF let

$$\Pi.C = Q_1 p_1 \dots Q_m p_m . c_1 \wedge \dots \wedge c_n$$

be a QBF in PCNF. Let $\Pi'.C$ be obtained from $\Pi.C$ by removing those universal quantifications from Π whose variables do not occur in any clause of C . Let

$$\Pi''.C'' = \Pi' \forall p'_1 \dots \forall p'_n . (c_1 \vee p'_1) \wedge \dots \wedge (c_n \vee p'_n)$$

with $p'_1 \dots p'_n$ fresh. Clearly, the size of $\Pi''.C''$ is linear in the size of $\Pi.C$. We show that $\Pi.C$ is in CMF iff $\Pi''.C''$ is in QMF. First assume that $\Pi.C$ is in CMF. Then $\Pi'.C$ is also in CMF and, by Theorem 1, in QMF. Hence, $\Pi''.C''$ is in QMF as well. Now assume that $\Pi.C$ is not in CMF. If $\Pi.C$ is satisfiable, then so is $\Pi''.C''$; thus, $\Pi''.C'' \notin \text{QMF}$. Let $\Pi.C$ be unsatisfiable. Clearly, $\Pi'.C$ is also not in CMF. For some $0 \leq i \leq n$ let c_i be a clause that can be removed from C without making the resulting QBF satisfiable. Then

$$\Pi' \forall p'_1 \dots \forall p'_{i-1} \exists p'_i \forall p'_{i+1} \dots \forall p'_n. (c_1 \vee p'_1) \wedge \dots \wedge (c_n \vee p'_n),$$

which is a proper q-core of $\Pi''.C''$, is unsatisfiable. Hence, $\Pi''.C''$ is not in QMF. Thus, QMF is PSPACE-hard. The proof for PSPACE-hardness of QCMF is similar. This concludes the proof. ■

VII. A2AECC: Q- AND QC-CORES AS C-CORES

We now describe a source-to-source transformation on QBF in PCNF that allows to cast q- and qc-cores as c-cores. Let $\Pi.C$ be a QBF in PCNF. For each universally quantified variable p_i in $\Pi.C$ the transformation replaces the quantification $\forall p_i$ in the prefix Π with $\forall p'_i \exists p_i$, where p'_i is a fresh variable, and conjoins the matrix C with two clauses $(p_i \rightarrow p'_i)$ and $(p'_i \rightarrow p_i)$. Hence, the acronym A2AECC. This is formalized in Definition 5.

Definition 5 (A2AECC): Let $\Pi.C = Q_1 p_1 \dots Q_n p_n. C$. Let p'_1, \dots, p'_n be fresh. Let, for all $1 \leq i \leq n$,

$$a2ae(Q_i p_i) = \begin{cases} \forall p'_i \exists p_i & \text{if } Q_i = \forall \\ \exists p_i & \text{otherwise,} \end{cases}$$

and

$$a2cc(Q_i p_i) = \begin{cases} (p_i \rightarrow p'_i) \wedge (p'_i \rightarrow p_i) & \text{if } Q_i = \forall \\ \top & \text{otherwise.} \end{cases}$$

Then

$$a2aecc(\Pi.C) = a2ae(Q_1 p_1) \dots a2ae(Q_n p_n). \\ \left(\bigwedge_{1 \leq i \leq n} a2cc(Q_i p_i) \right) \wedge C.$$

Let $\Pi.C$ be an unsatisfiable QBF in PCNF. Definition 5 allows to compute an unsatisfiable q- or qc-core $\Pi'.C'$ of $\Pi.C$ by computing an unsatisfiable c-core of $a2aecc(\Pi.C)$ as follows.

- 1) Let $\Pi_{a2aecc}.C_{a2aecc} = a2aecc(\Pi.C)$.
- 2) Compute an unsatisfiable c-core $\Pi_{a2aecc}.C'_{a2aecc}$ of $\Pi_{a2aecc}.C_{a2aecc}$.
- 3) Let $C' = C$ if a q-core is desired, and let $C' = C \cap C'_{a2aecc}$ if a qc-core is desired.
- 4) Obtain Π' from Π by replacing each quantification $Q_i p_i$ in Π with $Q'_i p_i$ where

$$Q'_i = \begin{cases} \exists & \text{if } (Q_i = \exists) \text{ or } (Q_i = \forall \text{ and } C'_{a2aecc} \cap \{(p_i \rightarrow p'_i), (p'_i \rightarrow p_i)\} = \emptyset), \\ \forall & \text{otherwise.} \end{cases}$$

The correctness of this procedure is established in Theorem 3 below. Its proof uses the following Lemma 2, which is immediate by the semantics of QBF.

Lemma 2: Let

$$\Pi.C = Q_1 p_1 \dots Q_{l-1} p_{l-1} \forall p_l Q_{l+1} p_{l+1} \dots Q_m p_m. \\ c_1 \wedge \dots \wedge c_n$$

be a QBF in PCNF. Let p'_l be fresh. Let

$$\Pi'.C' = Q_1 p_1 \dots Q_{l-1} p_{l-1} \forall p'_l \exists p_l Q_{l+1} p_{l+1} \dots Q_m p_m. \\ (p_l \rightarrow p'_l) \wedge (p'_l \rightarrow p_l) \wedge c_1 \wedge \dots \wedge c_n.$$

Then $\Pi.C$ is satisfiable iff $\Pi'.C'$ is satisfiable.

Theorem 3: Let $\Pi.C$ be a QBF in PCNF. Let P be a subset of the universally quantified variables in Π . Let Π' be obtained from Π by weakening $\forall p$ to $\exists p$ for all $p \in P$. Let

$$\Pi_{a2aecc}.C_{a2aecc} = a2aecc(\Pi.C)$$

and let

$$C'_{a2aecc} = C_{a2aecc} \setminus \bigcup_{p \in P} \{(p \rightarrow p'), (p' \rightarrow p)\}.$$

Then

- 1) $\Pi'.C$ is a q-core of $\Pi.C$.
- 2) $\Pi_{a2aecc}.C'_{a2aecc}$ is a c-core of $\Pi_{a2aecc}.C_{a2aecc}$.
- 3) $\Pi'.C$ is satisfiable iff $\Pi_{a2aecc}.C'_{a2aecc}$ is satisfiable.

Proof: Claims 1, 2 follow directly from Definition 1. We prove claim 3 by induction on the cardinality of P . The base case $|P| = 0$ follows by repeated application of Lemma 2. For the inductive case assume that the claim holds for any P with $|P| = n$. Now let $P = \{p_1, \dots, p_{n+1}\}$. Let Π'' be obtained from Π by weakening $\forall p_{n+1}$ in Π to $\exists p_{n+1}$. Let $\Pi''_{a2aecc}.C''_{a2aecc} = a2aecc(\Pi''.C)$. By inductive assumption $\Pi'.C$ is satisfiable iff

$$\Pi''_{a2aecc}.C''_{a2aecc} \setminus \bigcup_{p \in \{p_1, \dots, p_n\}} \{(p \rightarrow p'), (p' \rightarrow p)\}$$

is satisfiable. By construction of C'_{a2aecc} and C''_{a2aecc} we have

$$C'_{a2aecc} = C''_{a2aecc} \setminus \bigcup_{p \in \{p_1, \dots, p_n\}} \{(p \rightarrow p'), (p' \rightarrow p)\}.$$

Hence, $\Pi'.C$ is satisfiable iff $\Pi''_{a2aecc}.C'_{a2aecc}$ is satisfiable. Notice that Π_{a2aecc} only differs from Π''_{a2aecc} by having $\forall p'_{n+1} \exists p_{n+1}$ in place of $\exists p_{n+1}$. Hence, as p'_{n+1} does not occur in C'_{a2aecc} , $\Pi_{a2aecc}.C'_{a2aecc}$ is satisfiable iff $\Pi_{a2aecc}.C'_{a2aecc}$ is satisfiable. Thus, by transitivity $\Pi'.C$ is satisfiable iff $\Pi_{a2aecc}.C'_{a2aecc}$ is satisfiable as desired. This concludes the proof. ■

If a prefix Π has m universal quantifiers, then the alternation depth of $a2aecc(\Pi.C)$ is either $2m$ or $2m + 1$. In the next Section VIII we present a variant of the transformation that does not affect alternation depth but has different semantics.

If a universally quantified variable p is pure in a matrix C , then either $(p' \rightarrow p)$ (if p occurs only non-negated in C) or $(p \rightarrow p')$ (if p occurs only negated in C) is a

quantified blocked clause [BLS11] in $a2aecc(\Pi.C)$ and can be eliminated.

If a solver for QBF in PCNF supports grouping of clauses when extracting c-cores (e.g., [NRS14, Nad10, LS08]), as does DepQBF [LE15], then a clause group for each pair of clauses $(p_i \rightarrow p'_i), (p'_i \rightarrow p_i)$ introduced by Definition 5 can be used to ensure that either none or both of $(p_i \rightarrow p'_i), (p'_i \rightarrow p_i)$ are present in a c-core of $a2aecc(\Pi.C)$.

Example 2: As an example we revisit $\Pi.C = \forall p.(p) \wedge (\neg p)$ from Example 1. We have

$$a2aecc(\Pi.C) = \forall p' \exists p. (p \rightarrow p') \wedge (p' \rightarrow p) \wedge (p) \wedge (\neg p).$$

The unsatisfiable c-cores

$$\Pi'.C'_1 = \forall p' \exists p. (p \rightarrow p') \wedge (p' \rightarrow p) \wedge (p)$$

and

$$\Pi'.C'_2 = \forall p' \exists p. (p \rightarrow p') \wedge (p' \rightarrow p) \wedge (\neg p)$$

of $a2aecc(\Pi.C)$ correspond to the unsatisfiable c-cores $\forall p.(p)$ and $\forall p.(\neg p)$ of $\Pi.C$. The unsatisfiable c-core

$$\Pi'.C'_3 = \forall p' \exists p. (p) \wedge (\neg p)$$

of $a2aecc(\Pi.C)$ corresponds to the unsatisfiable q-core $\exists p.(p) \wedge (\neg p)$ of $\Pi.C$. Contrary to $\Pi'.C'_3$, neither $\Pi'.C'_1$ nor $\Pi'.C'_2$ is c-minimally unsatisfiable; however, when treating $(p \rightarrow p'), (p' \rightarrow p)$ as a clause group as mentioned above, then $\Pi'.C'_1$ and $\Pi'.C'_2$ are c-minimally unsatisfiable under a suitable definition of c-minimality that takes clause groups into account. \square

The transformation in Definition 5, Theorem 3 is also of theoretical interest. For example, it can be used to extend the hitting set-based relationship [Sla14] between unsatisfiable subsets of clauses and co-satisfiable subsets of clauses (complements of satisfiable subsets [Sla14], i.e., diagnoses [Rei87] and repairs [Sch05]) to a relationship between unsatisfiable q- or qc-cores and suitably defined “co-satisfiable q- or qc-cores” of QBF in PCNF. The latter induce an enhanced notion of diagnosis and repair for QBF in PCNF that diagnoses and repairs unsatisfiable QBF not only by removal of clauses but also by weakening of universal to existential quantifiers.

For a second example consider the following relationship between Definition 6 (see Section X) and [KLM06]. In [KLM06] Kullmann et al. classify the clauses of a CNF as necessary if they are contained in all minimal unsatisfiable cores, as only potentially necessary if they are contained in some but not all minimal unsatisfiable cores, and as never necessary if they are not contained in any minimal unsatisfiable core. It is easy to see that $\forall p$ is not \forall -to- \exists reducible in $\Pi.C$ iff $\{(p \rightarrow p'), (p' \rightarrow p)\}$ has a non-empty intersection with all c-minimally unsatisfiable c-cores of $a2aecc(\Pi.C)$ (cf. “necessary” in [KLM06]) and trivially \forall -to- \exists reducible in $\Pi.C$ iff $\{(p \rightarrow p'), (p' \rightarrow p)\}$ has an empty intersection with all c-minimally unsatisfiable c-cores of $a2aecc(\Pi.C)$ (cf. “never necessary” in [KLM06]).

VIII. A VARIANT OF A2AECC: REDUCING ALTERNATION DEPTH BY REDUCING PRECISION

In this section we discuss a variant of the A2AECC transformation that avoids the increase in alternation depth when going from $\Pi.C$ to $a2aecc(\Pi.C)$, but underapproximates the set of universal quantifiers that can be weakened to existential ones in an unsatisfiable q- or qc-core of $\Pi.C$.

Let $\Pi.C$ be a QBF in PCNF with n universal quantifiers and alternation depth m . Let $\forall p_{i,1} \dots \forall p_{i,n_i}$ be a maximal sequence, called a *block*, of universal quantifications in $\Pi.C$. Definition 5 turns this block into $\forall p'_{i,1} \exists p_{i,1} \dots \forall p'_{i,n_i} \exists p_{i,n_i}$. Overall, this increases the alternation depth of $a2aecc(\Pi.C)$ compared to $\Pi.C$ by $2 \cdot n - m$ (+1, if Π starts with \exists).

Consider a variant of Definition 5, denoted $a2aecc'$, that instead turns each block of universal quantifications $\forall p_{i,1} \dots \forall p_{i,n_i}$ into $\forall p'_{i,1} \dots \forall p'_{i,n_i} \exists p_{i,1} \dots \exists p_{i,n_i}$. Now the increase in alternation depth from $\Pi.C$ to $a2aecc'(\Pi.C)$ is at most 1. Moreover, by considering the respective tree refutations (see Section IV), it is easy to see that $a2aecc(\Pi.C)$ is unsatisfiable iff $a2aecc'(\Pi.C)$ is unsatisfiable. As shown in Theorem 3, removing $(p_{i,i'} \rightarrow p'_{i,i'}) \wedge (p'_{i,i'} \rightarrow p_{i,i'})$ from $a2aecc(\Pi.C)$ corresponds to weakening $\forall p_{i,1} \dots \forall p_{i,i'-1} \forall p_{i,i'} \forall p_{i,i'+1} \dots \forall p_{i,n_i}$ to $\forall p_{i,1} \dots \forall p_{i,i'-1} \exists p_{i,i'} \forall p_{i,i'+1} \dots \forall p_{i,n_i}$ in $\Pi.C$. In contrast, it is straightforward to prove that removing $(p_{i,i'} \rightarrow p'_{i,i'}) \wedge (p'_{i,i'} \rightarrow p_{i,i'})$ from $a2aecc'(\Pi.C)$ corresponds to weakening $\forall p_{i,1} \dots \forall p_{i,i'-1} \forall p_{i,i'} \forall p_{i,i'+1} \dots \forall p_{i,n_i}$ to $\forall p_{i,1} \dots \forall p_{i,i'-1} \forall p_{i,i'+1} \dots \forall p_{i,n_i} \exists p_{i,i'}$ in $\Pi.C$.

By the semantics of QBF the unsatisfiability of a c-core of $a2aecc'(\Pi.C)$ implies the unsatisfiability of the corresponding c-core of $a2aecc(\Pi.C)$. For an example that the converse is not true consider $\Pi.C = \forall p_1 \forall p_2. (p_1 \rightarrow p_2) \wedge (p_2 \rightarrow p_1)$. Weakening $\forall p_1$ to $\exists p_1$ in $\Pi.C$ results in $\exists p_1 \forall p_2. (p_1 \rightarrow p_2) \wedge (p_2 \rightarrow p_1)$, which is unsatisfiable. Correspondingly, in line with Theorem 3, removing $(p_1 \rightarrow p'_1) \wedge (p'_1 \rightarrow p_1)$ from $a2aecc(\Pi.C)$ yields the unsatisfiable

$$\forall p'_1 \exists p_1 \forall p'_2 \exists p_2. (p_2 \rightarrow p'_2) \wedge (p'_2 \rightarrow p_2) \wedge (p_1 \rightarrow p_2) \wedge (p_2 \rightarrow p_1).$$

On the other hand, removing $(p_1 \rightarrow p'_1) \wedge (p'_1 \rightarrow p_1)$ from $a2aecc'(\Pi.C)$ leads to

$$\forall p'_1 \forall p'_2 \exists p_1 \exists p_2. (p_2 \rightarrow p'_2) \wedge (p'_2 \rightarrow p_2) \wedge (p_1 \rightarrow p_2) \wedge (p_2 \rightarrow p_1),$$

which is satisfiable, as is $\forall p_2 \exists p_1. (p_1 \rightarrow p_2) \wedge (p_2 \rightarrow p_1)$.

We finally discuss a different perspective on the semantics of $a2aecc'$. $a2aecc$ considers the positions of quantifications within a quantifier block as fixed, i.e., a block of universal quantifications is treated as a *list* of quantifications. However, the semantics of QBF allows to arbitrarily shuffle the quantifications within a quantifier block without affecting the satisfiability of the resulting QBF. Hence, a quantifier block can also be seen as a *set* of quantifications. In the light of that, $a2aecc'$ can be interpreted as employing the set semantics of a quantifier block and push the universal quantifications that have been weakened to existential ones to the right of their quantifier block (i.e., towards the inside of

the QBF). We call the semantics obtained when using *a2aecc* *list semantics* and the semantics obtained when using *a2aecc'* *set-inner semantics*. List semantics takes a very conservative approach in that it assigns maximal meaning to the order of the quantifications in a quantifier block, whereas set-inner semantics is very relaxed and assigns no meaning to the order of quantifications in a quantifier block at all. Keep in mind that, while — as mentioned above — shuffling quantifications inside a quantifier block is a satisfiability-preserving operation, as shown by the example in the previous paragraph weakening universal quantifications to existential ones is not the same in list and in set-inner semantics.

IX. INTERPRETING UNSATISFIABLE Q- AND QC-CORES

We now explain that the weakening of a universal to an existential quantifier in an unsatisfiable core may have different reasons and that it is easier to judge the significance of a weakening in an unsatisfiable core if the core is c-minimal.

Let $\Pi.C$ be an unsatisfiable QBF in PCNF and consider an unsatisfiable q- or qc-core $\Pi'.C'$ of $\Pi.C$. Assume that some $\forall p$ in Π has been weakened to $\exists p$ in Π' . Let C'' be a subset of C' such that $\Pi'.C''$ is c-minimally unsatisfiable (such C'' obviously exists). Distinguish two cases. First, assume that p occurs in some clause c in C'' . Then there is a cause of the unsatisfiability of $\Pi.C$ that requires c , including its occurrence of p , but needs p to be only existentially quantified (as it is in Π') rather than universally quantified (as it is in Π). Second, assume that there is no such clause. Then the weakening of $\forall p$ to $\exists p$ in Π' is due to the fact that the unsatisfiability of $\Pi.C$ does not require any clause that contains p or $\neg p$.

Notice that in a q- or qc-core that is unsatisfiable but not c-minimal both cases may occur simultaneously for different choices of C'' . Hence, the fact that $\forall p$ has been weakened to $\exists p$ in a non-c-minimally unsatisfiable q- or qc-core $\Pi'.C'$ of $\Pi.C$ should be interpreted with some care. Moreover, if $\forall p$ has been weakened to $\exists p$ in a c-minimally unsatisfiable q- or qc-core $\Pi'.C'$, then it should be checked whether C' contains p or not (if not, our implementation removes $\exists p$ from Π' during postprocessing).

Example 3: As an example consider

$$\Pi.C = \forall p_1 \forall p_2 \forall p_3 \exists p_4. (p_1 \rightarrow p_2) \wedge (p_2 \rightarrow p_1) \wedge (p_3 \rightarrow p_4)$$

and a (non-c-minimally) unsatisfiable qc-core of $\Pi.C$

$$\Pi'.C' = \exists p_1 \forall p_2 \exists p_3 \exists p_4. (p_1 \rightarrow p_2) \wedge (p_2 \rightarrow p_1) \wedge (p_3 \rightarrow p_4).$$

Inspection of $\Pi'.C'$ shows that its unsatisfiability is caused by $\exists p_1 \forall p_2. (p_1 \rightarrow p_2) \wedge (p_2 \rightarrow p_1)$, and that for its unsatisfiability it is sufficient for p_1 to be existentially quantified. Hence, the weakening of $\forall p_1$ to $\exists p_1$ in $\Pi'.C'$ provides useful additional information about the unsatisfiability of $\Pi.C$. On the other hand, $\exists p_3 \exists p_4. (p_3 \rightarrow p_4)$ does not contribute to the unsatisfiability of $\Pi'.C'$. Hence, the fact that p_3 is existentially quantified in $\Pi'.C'$ provides little information about the unsatisfiability of $\Pi.C$. $\Pi'.C'$ has a single c-minimally unsatisfiable c-core $\Pi''.C'' = \exists p_1 \forall p_2 \exists p_3 \exists p_4. (p_1 \rightarrow p_2) \wedge (p_2 \rightarrow p_1)$. Remember that in a c-minimally unsatisfiable core every

clause is required for unsatisfiability. As we can see, p_1 occurs in the matrix C'' , while p_3 does not. \square

X. \forall -TO- \exists REDUCIBILITY

We now lift the discussion of the previous section from a single unsatisfiable core to the original formula $\Pi.C$ by partitioning the set of universally quantified variables in Π into three sets as follows. The first set contains those universally quantified variables p of Π for which a c-minimal qc-core $\Pi'.C'$ of $\Pi.C$ exists such that p is existentially quantified in Π' and occurs in C' ; these are the variables that are actually still relevant for the unsatisfiability of $\Pi.C$ when weakened from universally to existentially quantified. The second set contains those universally quantified variables of Π that can be weakened to existentially quantified variables without making the result satisfiable, but for which no c-minimal qc-core $\Pi'.C'$ of $\Pi.C$ exists in which they are existentially quantified in Π' and occur in C' ; these are the variables that are not relevant for the unsatisfiability of $\Pi.C$. Finally, the third set contains those universally quantified variables of Π that cannot be weakened to existentially quantified variables without making the result satisfiable.

Definition 6 (\forall -to- \exists Reducibility): Let $\Pi.C$ be unsatisfiable, and let $\forall p$ occur in Π .

- 1) If there exists a c-minimally unsatisfiable qc-core $\Pi'.C'$ of $\Pi.C$ such that $\forall p$ in Π has been weakened to $\exists p$ in Π' and such that p occurs in C' , then $\forall p$ is *non-trivially \forall -to- \exists reducible* in $\Pi.C$.
- 2) If $\forall p$ is not non-trivially \forall -to- \exists reducible in $\Pi.C$ but there exists an unsatisfiable q-core $\Pi'.C$ of $\Pi.C$ such that $\forall p$ in Π has been weakened to $\exists p$ in Π' , then $\forall p$ is *trivially \forall -to- \exists reducible* in $\Pi.C$.
- 3) If there exists no unsatisfiable q-core $\Pi'.C$ of $\Pi.C$ in which $\forall p$ has been weakened to $\exists p$, then $\forall p$ is *not \forall -to- \exists reducible* in $\Pi.C$.

If a universally quantified variable p is pure in C , then — because of the pure literal rule for existentially quantified variables (e.g., [GMN09]) — $\forall p$ is either trivially or not \forall -to- \exists reducible in $\Pi.C$.

Example 4: We continue Example 3. In

$$\Pi.C = \forall p_1 \forall p_2 \forall p_3 \exists p_4. (p_1 \rightarrow p_2) \wedge (p_2 \rightarrow p_1) \wedge (p_3 \rightarrow p_4)$$

p_1 is non-trivially \forall -to- \exists reducible, p_2 is not \forall -to- \exists reducible, and p_3 is trivially \forall -to- \exists reducible. \square

To better understand the potential for weakening \forall to \exists we are interested in computing which variables in an unsatisfiable QBF $\Pi.C$ are non-trivially \forall -to- \exists reducible. A precise result might require finding all c-minimally unsatisfiable qc-cores of $\Pi.C$. We suggest two methods to underapproximate the set of non-trivially \forall -to- \exists reducible variables. We start with the second method. For each $\forall p$ in Π it performs the following steps.

- 1) $\Pi'_{(i)}.C'_{(i)}$ is obtained from $\Pi.C$ by weakening $\forall p$ to $\exists p$.
- 2) If $\Pi'_{(i)}.C'_{(i)}$ is satisfiable, then $\forall p$ is not \forall -to- \exists reducible in $\Pi.C$, and the method moves on to the next universal quantification in Π .

- 3) $\Pi'_{(iii)}.C'_{(iii)}$ is obtained from $\Pi'_{(i)}.C'_{(i)}$ by weakening a maximal set of universal quantifiers to existential ones in $\Pi'_{(i)}$ and by removing a maximal set of clauses without occurrences of p from $C_{(i)}$ such that the result is still unsatisfiable.
- 4) $\Pi'_{(iv)}.C'_{(iv)}$ is obtained from $\Pi'_{(iii)}.C'_{(iii)}$ by removing all clauses with occurrences of p from $C'_{(iii)}$.
- 5) If $\Pi'_{(iv)}.C'_{(iv)}$ is satisfiable, then $\forall p$ is non-trivially \forall -to- \exists reducible in $\Pi.C$; otherwise, $\forall p$ is trivially or non-trivially \forall -to- \exists reducible in $\Pi.C$.

The first method, which is cheaper but reports “trivially or non-trivially” \forall -to- \exists reducible more often, omits step 3.

XI. IMPLEMENTATION

We implemented our ideas in DepQBF [LE17] version 6.03; we call our version DepQBF-a2aecc. DepQBF-a2aecc takes a QBF in PCNF $\Pi.C$ as input. DepQBF-a2aecc can either be used as a preprocessor to obtain $a2aecc(\Pi.C)$, or it can compute — optionally q- and c-minimally — unsatisfiable c-cores, q-cores, or qc-cores of $\Pi.C$. DepQBF allows to declare clause groups and, if a formula is found unsatisfiable, to obtain the clause groups used to establish unsatisfiability [LE15]. We use this to obtain an initial unsatisfiable c-core $\Pi'.C'$ of $\Pi.C$ (for c-cores) or of $a2aecc(\Pi.C)$ (for q- and qc-cores). For c-cores $\Pi'.C'$ can be output directly. For q-cores and qc-cores Theorem 3 is applied to translate $\Pi'.C'$ back into a q- or qc-core of $\Pi.C$. If minimality is desired, then C' is minimized using a deletion-based algorithm (e.g., [Mar12]) with clause set refinement (CSR) (e.g., [BLM12]), where in the repeated checks for satisfiability the DepQBF API [LE15] is used to disable or enable clause groups as needed. Because of Theorem 1 minimization is first applied to the clauses introduced by Definition 5 and then to the clauses from C ; optionally, CSR can also be restricted to be applied to the clauses introduced by Definition 5 during the first phase of minimization. The variant of the A2AECC transformation presented in Sec. VIII is available as an option.

XII. CASE STUDIES

In this section we discuss some case studies, which we encountered during our experimental evaluation, that illustrate how the weakening of universal to existential quantifiers in unsatisfiable cores can trigger improved understanding of unsatisfiable QBFs. The examples are taken from QBF LIB [GNPT].

a) Winning Strategies in Two-Player Games: The Gent-Rowley suite models variants of the well-known Connect-4 game that are parameterized by the length of a winning line and the width and height of the game board [GR03]. Some instances model whether player 1 can enforce a draw. For some of these instances, with winning lines of length 2 on boards with at least two rows and two columns, there exists an unsatisfiable core in which all universal quantifiers have been turned into existential ones. I.e., even if player 1 had full control over the moves of player 2, she could not enforce a draw. This is clear, because eventually there must

be a winning line of length 2 for one of the two players, which is confirmed by the corresponding unsatisfiable core.

Moreover, for instances with longer winning lines and on larger boards, we obtained unsatisfiable cores with only a single universal quantifier left, which seemed odd (the number of universal quantifiers in the input formula grows with the maximal number of moves, i.e., the board size). Upon inspection of the unsatisfiable cores it turned out that the game is modeled in such a way that player 2 can spoil a draw by playing an illegal move at her first turn, thus forcing a win of player 1. This seems to be a fact that a user of the model in [GR03] should be aware of.

Finally, other instances model whether player 2 can enforce a win. Again, we obtained an unsatisfiable core with only one universal quantifier left. The core showed that the unsatisfiability was caused by player 1 playing an illegal first move, which should imply a win for player 2; this, however, is forbidden by Eqn. 12. in [GR03]. This seems to warrant an investigation of whether this way of modeling the game is indeed as intended.

b) Conformant Planning: The Rintanen/Sorting_networks family encodes a set of problems such that an instance with parameters d and l is satisfiable iff there exists a sorting network of depth d that, for all input sequences of length l , produces a sorted output sequence [Rin07,BG00]. The instance with $d = 3$, $l = 6$ is unsatisfiable. It yields an unsatisfiable core in which the universal quantification over the first number of the input sequence has been weakened to an existential one. I.e., even if the “planner” were allowed to freely choose the first number of the input sequence, there would be no sorting network. This is an interesting information in itself; it additionally implies that there is also no sorting network of depth 3 for input sequences of length 5.

c) Satisfiability of Modal Logic K: The Pan suite of examples encodes the satisfiability of formulas in the modal logic K as QBF [PV03,BHS00]. In the QBF encoding universal quantification runs over the values of an index variable, where each value of the index variable activates a part of the encoding that corresponds to a different \Diamond -subformula from the original K formula. This is done to avoid repeating certain subformulas in the resulting QBF, which keeps the complexity of the translation from K to QBF polynomial rather than exponential [PV03]. We obtained an unsatisfiable core for the instance k_branch_p-2 in which a universal quantifier had been weakened to an existential one. This showed that either one of two \Diamond -subformulas in the input formula is sufficient to obtain unsatisfiability.

d) Answer Set Programming: The Faber-Leone-Maratea-Ricca/Strategic_Companies family of examples encodes the question of whether two fixed companies out of a set of companies are strategic [FLMR07,LPF⁺06,CEG97]. Instance $x25.17$ is unsatisfiable, which indicates that the companies under consideration are indeed strategic. In the unsatisfiable core the universal quantification over the variable for a third company has been weakened to an existential one, signaling that that company, too, is strategic.

XIII. EXPERIMENTAL EVALUATION

a) Setup and Benchmarks: We used one machine with a Xeon E3-1245v5 CPU and 32 GB RAM, utilizing 3 out of 4 physical cores for our experiments. The operating system was Ubuntu 16.04. Run time and memory limits were 300 s and 8 GB. The experiments took about 2.5 months on our machine.

We selected 5342 instances from QBFLIB [GNPT]. Instances were chosen randomly such that equally many instances were taken from each benchmark suite (subject to availability) and, recursively within benchmark suites, equally many instances from each subfamily (for the selection algorithm see Appendix A). I.e., from benchmark suites with fewer than 193 available instances all instances were included, and from each of the remaining suites at least 193 instances were used. We did not use any other selection criteria. Table I shows the resulting number of instances per benchmark suite; “solved” means solved by any solver in any of our experiments. Table II shows the minimum, first quartile, medium, third quartile, maximum, and mean values of the number of universal quantifiers, the number of existential quantifiers, the alternation depth, the number of clauses, and the maximum variable index for the set of instances. As we were interested in determining the potential for weakening universal to existential quantifiers in the examples, we did not use a preprocessor such as bloqqer [BLS11].

For our implementation and experimental data see <http://schuppan.de/viktor/ictai18/>.

In the tables and plots below “n.s.” stands for not solved. In plots red diagonal crosses are unsatisfiable and green horizontal-vertical crosses are satisfiable instances. Scatter plots such as Figure 2 (a) potentially suffer from overplotting, i.e., several benchmark instances resulting in the same x- and y-coordinates cannot be distinguished in the plot. In our case the effect tends to be most pronounced in the corners of the plot. We therefore replace the crosses *in the corners* by the numbers of instances exhibiting the corresponding x- and y-coordinates. When two values are given, then the red, upper value is for unsatisfiable and the green, lower value for satisfiable instances. For example, in Figure 2 (a) there are 804 instances that remained unsolved by both methods.

b) Extracting Unsatisfiable Cores: In our first set of experiments we used DepQBF-a2aecc to extract unsatisfiable cores from the 2528 instances that were found to be unsatisfiable. In Section XII we already described some of the unsatisfiable qc-cores that we obtained in more detail.

In the upper two sections of Table III we show how many universal quantifiers could be weakened to existential ones relative to the number of universal quantifiers in the original formula. Column 1 states which kind of unsatisfiable cores was extracted. “q” (resp. “qc”) refers to q-cores (resp. qc-cores), “min” to q-minimality for q-cores and to q,c-minimality for qc-cores, and “minsepcsr” to minimality with separate CSR. “list” refers to list semantics and “set-inner” refers to set-inner semantics in the A2AECC-transformation (see Section VIII). Column 2 states the number of solved instances (this

TABLE I: Number of instances per benchmark suite.

	all	solved unsatisfiable	solved satisfiable
Akshay-Chakraborty-John-Shah-Rabe	20	2	16
Amendola-Ricca-Truszczyński	112	9	7
Ansotegui	38	12	11
Ayari	71	48	23
Basler	193	75	118
Biere	194	25	159
Cashmore-Fox-Giunchiglia	150	110	40
Castellini	169	112	57
Chen-Interian	194	16	0
Diptarama-Jordan-Shinohara	14	11	2
Egly-Seidl-Tompits-Woltran-Zolda	194	97	78
Faber-Leone-Maratea-Ricca	194	128	7
Gent-Rowley	193	142	10
Herbsttritt	194	157	27
Interian	193	24	69
Jordan-Kaiser	194	83	87
Katz	20	8	8
Klieber	30	15	6
Kontchakov	136	70	66
Kronegger-Pfandler-Pichler	194	133	44
Lahiri-Seshia	3	1	2
Lee-Jiang	5	2	3
Letombe	194	85	107
Letz	14	9	5
Ling	8	3	5
Mangassarian-Veneris	170	60	71
MayerEichberger-Saffidine	113	3	35
Messinger	63	0	9
Miller-Marin	194	189	5
Miller-Scholl-Becker	194	160	18
Mneimneh-Sakallah	180	44	123
Narizzano	193	78	115
Palacios	24	9	14
Pan	194	89	98
Peitl	10	10	0
Preusser	12	0	9
qbfeval12	17	8	9
Rabe	14	3	0
Rintanen	131	55	71
Sauer-Reimer	193	42	142
Scholl-Becker	64	30	25
Seidl	194	194	0
Tacchella	193	122	70
Tentrup	74	17	29
Wintersteiger	194	38	96
sum	5342	2528	1896

TABLE II: Statistics of structural properties of the set of instances.

	min.	1st quart.	median	3rd quart.	max.	mean
all (n = 5342)						
number of \forall	0	19.25	90	213	55,022	325.8
number of \exists	1	477.5	2,239	7,215	2,202,774	18,980.3
alternation depth	1	2	3	6	1,141	17.7
num. of clauses	1	2,000	9,126.5	29,861.75	5,534,890	80,410.1
max. var. index	1	558.25	2,556.5	8,556.75	2,202,778	33,383.3
solved unsatisfiable (n = 2528)						
number of \forall	0	20	81.5	189	55,022	313.8
number of \exists	1	622	2,993	8,332.5	2,202,774	23,311.2
alternation depth	1	3	3	12	781	25.9
num. of clauses	5	2,274	11,207.5	35,299	5,534,890	104,313.6
max. var. index	5	764.25	3,287.5	9,880	2,202,778	42,507.8
solved satisfiable (n = 1896)						
number of \forall	0	12	63	232	10,404	314.8
number of \exists	1	408.75	1,338.5	4,707	1,112,278	7,802.7
alternation depth	1	2	3	4	133	6.3
num. of clauses	1	1,525.25	5,007	18,777	2,812,458	35,569.7
max. var. index	1	458	1,592.5	5,605.5	1,112,282	15,112.6

is the sum of the remaining columns). (For reference, the corresponding numbers for c-cores and c-minimal c-cores are 1830 and 1682, respectively.) Column 3 lists the number of solved instances that had no universal quantifiers. The

TABLE III: Number of instances whose number of weakened \forall in the core (resp. \forall -to- \exists reducible \forall) divided by the number of \forall in the original formula is in a range. For reference, c-cores were obtained for 1830 instances and c-minimal c-cores for 1682 instances, respectively.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	1649	21	465	1	4	4	13	46	11	8	95	159	305	96	291	130
q set-inner	1516	21	432	2	6	5	20	25	22	25	128	140	183	85	290	132
q min list	1139	21	195				5	5	6		37	82	250	96	266	176
q min set-inner	995	21	207				2	1	6		32	78	146	79	245	178
qc list	1551	21	1528	1			1									
qc set-inner	1414	21	1392						1							
qc min list	1441	21	1356	5	3	37	10	5	2	1	1					
qc min set-inner	1305	21	1266	1	1	3	5	3	3	2						
qc minsepcsr list	927	21	580	1	3	9	42	101	37	9	20	44	27	22	11	
qc minsepcsr set-inner	854	21	659		2	1	27	33	12	5	14	37	20	20	3	
enuma2e1 list	986	21	831		1		5	9	8	1	21	15	4	10	7	53
enuma2e1 set-inner	930	21	825		1		1	5			5	9	5	4	1	53
enuma2e2 list	657	21	385	1	2	9	29	38	28	22	36	19	7	10	5	45
enuma2e2 set-inner	645	21	465	2		6	11	25	15	10	15	21	4	6		44

remaining columns state how many instances exhibited q- or qc-cores whose share of weakened universal quantifiers falls in the range from the first row; as during postprocessing our implementation removes quantifications from the prefix whose variables have no occurrences in the matrix, the numerator of this fraction includes only weakened universal quantifications whose variables still occur in some clause of the matrix of the core. For example, for q-,c-minimal qc-cores with separate CSR, there were 22 instances such that the number of weakened universal quantifiers in the unsatisfiable core divided by the number of universal quantifiers in the original formula is in the interval $[0.6, 0.8[$. A number of instances exhibited q-cores in which quite a large share of universal quantifiers was weakened to existential ones; in the light of Section IX note, though, that these cores need not be c-minimal. Finding a qc-core in which a significant share of universal quantifiers is weakened to existential ones seems to require enabling minimization with separate CSR. Then also here instances in which a fairly large share of universal quantifiers is weakened to existential ones can be found; these cores are c-minimal. Unsurprisingly, Figure 1 shows that for q-cores, q-minimal q-cores, and q-,c-minimal qc-cores with separate CSR higher numbers of weakened universal quantifiers tend to be obtained from original instances with higher numbers of universal quantifiers.

In the lower section of Table III we show how many universal quantifiers were found to be non-trivially \forall -to- \exists reducible relative to the number of universal quantifiers in the original formula, where “enuma2e1” refers to the first and “enuma2e2” to the second method from Section X. Inspection of our data show that, as expected, the second method finds more non-trivially \forall -to- \exists reducible quantifiers than the first method.

In Figure 2 (a) we compare the sizes of q-,c-minimally unsatisfiable qc-cores obtained with separate CSR with the corresponding c-minimally unsatisfiable c-cores in terms of number of clauses. We find that the qc-cores obtained with CSR can be significantly larger than the corresponding c-cores. This is not surprising: weakening a universal to an existential quantifier corresponds to weakening a conjunction

to a disjunction, and proving unsatisfiability of a disjunction requires both disjuncts, while proving unsatisfiability of a conjunction requires only one conjunct. Remember (see Section XII) that already the fact that a certain universal quantifier has been weakened to an existential one may convey valuable information, irrespective of the remainder of the unsatisfiable core under consideration. Figure 2 (b) shows that large increases in core size tend to coincide with large numbers of weakened universal quantifiers, which is expected.

In Figure 3 (a)–(f) we show the run time overhead that is incurred by each step when going from no core extraction via c-core extraction to c-minimal c-core extraction and from no core extraction via q-core extraction, qc-core extraction and q-,c-minimal qc-core extraction to q-,c-minimal qc-core extraction with separate CSR. C-core extraction incurs limited costs (a); minimization comes with a high overhead (b). The relation of the run times between no core extraction and q-core extraction is quite variable (c). While moving from q-cores to qc-cores incurs only a moderate overhead (d), adding minimization (e) and, on top of that, separate CSR (f) are quite costly. In Figure 3 (g)–(l) we show the corresponding plots for memory; memory usage turned out not to be a problem for `DepQBF-a2aecc`. Notice that (b) involves solving the original versus solving the A2AECC-transformed instance; although the increase in alternation depth of the transformed instance depends on twice the number of universal quantifiers minus the alternation depth in the original instance, we did not observe a clear corresponding dependence of the overhead in (b) (see Figure 4).

We also ran the experiments using set-inner instead of list semantics. As expected, when using set-inner semantics, often fewer universal quantifiers were weakened to existential ones. However, despite lower alternation depth of the transformed formula, we did not find an unambiguous performance advantage for set-inner semantics (see Figure 5).

c) Solving A2AECC-Transformed Versus Original Instances: Our method for extracting unsatisfiable q- and qc-cores as described in Section VII consists of a preprocessing step that applies the A2AECC transformation, extraction of unsatisfiable c-cores, and a postprocessing step that maps c-cores

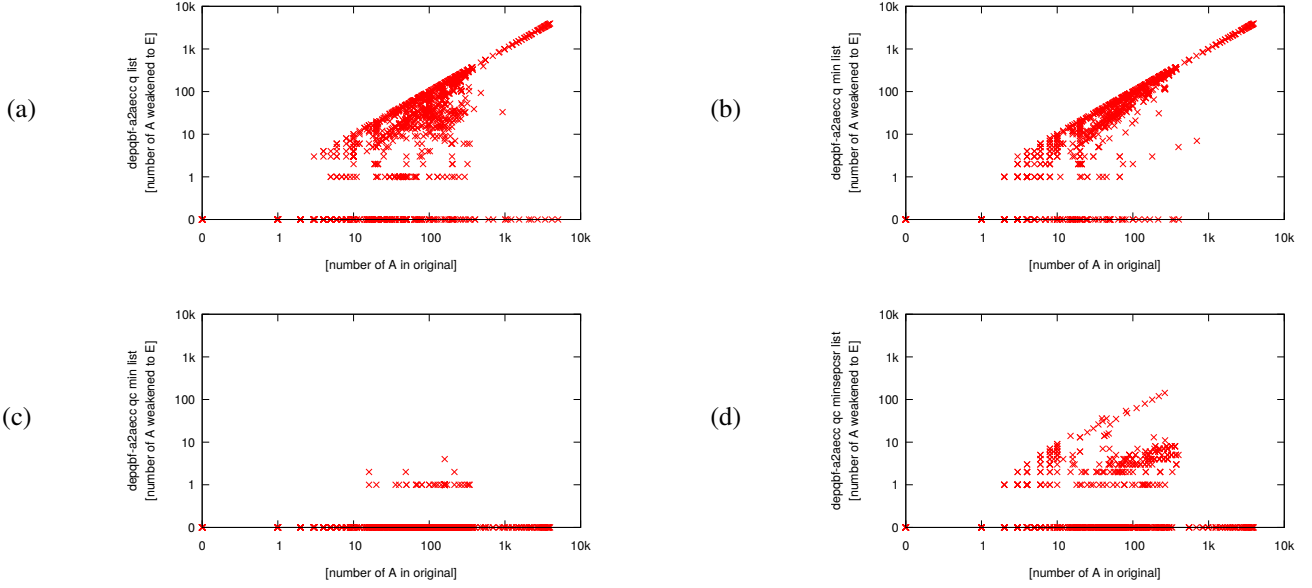


Fig. 1: Number of universal quantifiers weakened to existential ones depending on the number of universal quantifiers in the original formula: (a): q-cores; (b): q-minimal q-cores; (c): q-,c-minimal qc-cores; (d): q-,c-minimal qc-cores with separate CSR.

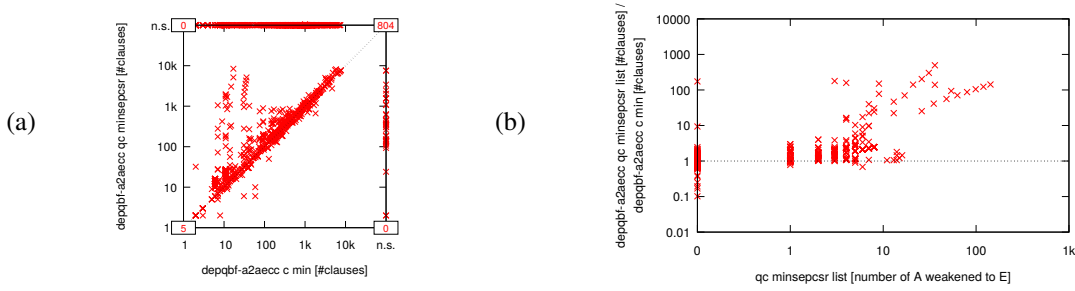


Fig. 2: (a) Comparing sizes of unsatisfiable cores [number of clauses]: x-axis: c-minimal c-cores, y-axis: q-,c-minimal qc-cores with separate CSR. (b) Ratio of core sizes [numbers of clauses] between q-,c-minimal qc-cores with separate CSR and c-minimal c-cores (y-axis) depending on the number of universal quantifications weakened to existential ones in the q-,c-minimal qc-core with separate CSR (x-axis); only pairs for which both unsatisfiable cores were obtained are included.

back to q- or qc-cores. This makes it possible to investigate the impact of the preprocessing step not only on `DepQBF-a2aecc` but also on other QBF solvers, thus allowing for a partial evaluation of our proposed methodology beyond `DepQBF-a2aecc`. Therefore, in our second set of experiments, we used `DepQBF-a2aecc` as a preprocessor and ran the following QBF solvers on the original and transformed instances: `DepQBF` v. 6.03 [LE17,depqbf], `AIGSolve` [PS10,aigsolve], `CAQE` v. qbfeval 2017 [Ten17,caqe], `GhostQ` v. 2017-07-26 [JKMC12,ghostq], `QESTO` v. 1.0 [JM15,qesto], and `RAREQS` v. 1.1 [JKMC12,rareqs]. Table IV shows the numbers of solved instances, and Figure 6 (a)–(f) compare the run times for solving the transformed versus the original instances for `DepQBF` (a), `AIGSolve` (b), `CAQE` (c), `GhostQ` (d), `QESTO` (e), and `RAREQS` (f). We observe that (i) the transformed instances can be solved in many cases, (ii) the overhead for solving the transformed instances depends on the solver, and (iii) some of the transformed instances are solved faster than

the original instances by some solvers. Only `AIGSolve` and `QESTO` ran into memory out on a larger number of instances; the number of memory outs reached up to 25 % of the number of time outs. For plots see Figure 6 (g)–(l). For `CAQE`, `QESTO`, and, to a lesser extent, `RAREQS` our data indicate a dependence of the overhead of solving the transformed versus the original instance on twice the number of universal quantifiers minus the alternation depth in the original instance (see Figure 7).

We also ran the experiments using set-inner instead of list semantics (see Figure 8). Only for `RAREQS` set-inner semantics resulted in a fairly unambiguous performance advantage. `AIGSolve` and `GhostQ` were affected comparatively little by the choice of transformation, while for the remaining solvers no clear picture arose.

d) quantum: Despite its differences quantum is the most closely related tool. In our last set of experiments we performed a preliminary comparison of both tools. We used quantum to obtain a minimum cardinality set of uni-

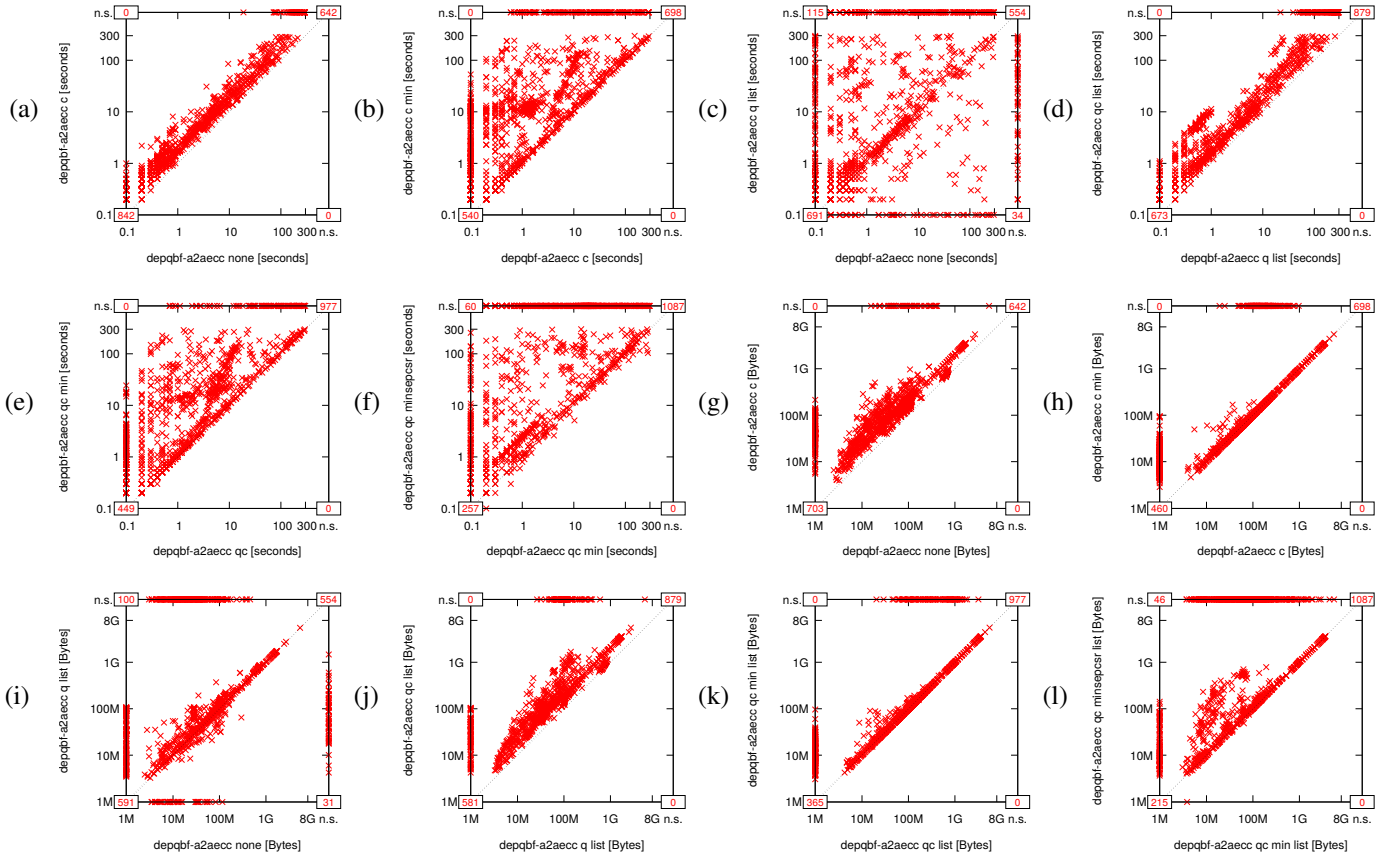


Fig. 3: (a)–(f) Comparing run times for extracting unsatisfiable cores in list semantics [seconds]: (a) x-axis: no cores, y-axis: c-cores; (b) x-axis: c-cores, y-axis: c-minimal c-cores; (c) x-axis: no cores, y-axis: q-cores; (d) x-axis: q-cores, y-axis: qc-cores; (e) x-axis: qc-cores, y-axis: q-,c-minimal qc-cores; (f) x-axis: q-,c-minimal qc-cores, y-axis: q-,c-minimal qc-cores with separate CSR. (g)–(l): as (a)–(f) but for memory [Bytes].

TABLE IV: Solving A2AECC-transformed versus original instances: number of (un)-solved instances. The best value per column is in **bold** font, and “vbs” is the virtual best solver.

	original			A2AECC list			A2AECC set-inner		
	unsat	sat	n.s.	unsat	sat	n.s.	unsat	sat	n.s.
DepQBF	1911	1293	2138	1556	980	2806	1589	974	2779
AIGSolve	1655	1445	2242	1663	1448	2231	1670	1447	2225
CAQE	2091	1154	2097	1666	921	2755	1413	670	3259
GhostQ	1831	1275	2236	1829	1280	2233	1818	1282	2242
QESTO	1793	995	2554	1387	826	3129	1149	528	3665
RAREQS	1900	942	2500	1106	519	3717	1201	519	3622
vbs	2496	1866	980	2374	1787	1181	2333	1789	1220

versal quantifiers that, when weakened to existential ones, make an unsatisfiable QBF satisfiable, and compared the performance with extracting q-minimally unsatisfiable q-cores with DepQBF-a2aecc (this compares minimum cardinality diagnoses with minimal unsatisfiable cores, which are quite different!). DepQBF-a2aecc (resp. quantum) was faster on 835 (resp. 81) instances, with some large differences both ways. For a scatter plot see Figure 9.

XIV. CONCLUSIONS

We proposed a notion of unsatisfiable q- and qc-cores for QBF in PCNF that weakens universal to existential quantifiers in addition to removing clauses, leading to unsatisfiable cores

and, thus, explanations and diagnoses of unsatisfiability that cannot be obtained from traditional c-cores. We used the A2AECC-transformation to reduce obtaining unsatisfiable q- and qc-cores to obtaining unsatisfiable c-cores. We illustrated with case studies that useful additional information can be obtained from unsatisfiable qc-cores, and we demonstrated in our experimental evaluation that our approach can successfully compute unsatisfiable q- and qc-cores on examples from QFBLIB. Potential future work includes analyzing how the A2AECC-transformation and its variant affect different solvers, obtaining unsatisfiable q- and qc-cores without using a transformation, e.g., directly from a run of the solver, and extending this work to logics with quantification beyond QBF.

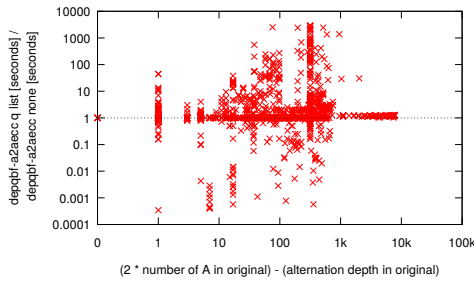


Fig. 4: Ratio of run times [seconds] between q-core extraction and no core extraction (y-axis) depending on the increase in alternation depth between the original and the A2AECC-transformed instances (x-axis); only pairs for which both instances were solved are included. Using `gbutils` [gbutils] we obtained Kendall’s rank correlation coefficient $\tau = 0.17$ at $p = 3.1 \cdot 10^{-25}$.

ACKNOWLEDGMENTS

I thank the authors of [RSMB14], especially Sven Reimer, for discussion of their work and for providing me with `quantom` [RPSB12]. I thank Alessandro Cimatti for mentioning that proofs can be used to compare formulas. I am grateful to the reviewers for their suggestions on how to improve the paper.

REFERENCES

- [AB00] A. Ayari and D. A. Basin. “Bounded Model Construction for Monadic Second-Order Logics”. In: *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*. Ed. by E. A. Emerson and A. P. Sistla. Vol. 1855. Lecture Notes in Computer Science. Springer, 2000, pp. 99–112. ISBN: 3-540-67770-4. DOI: [10.1007/10722167_11](https://doi.org/10.1007/10722167_11) (cit. on p. 1).
- [AFF⁺03] R. Armoni, L. Fix, A. Flaisher, et al. “Enhanced Vacuity Detection in Linear Temporal Logic”. In: *Computer Aided Verification, 15th International Conference, CAV 2003, Boulder, CO, USA, July 8-12, 2003, Proceedings*. Ed. by W. A. Hunt Jr. and F. Somenzi. Vol. 2725. Lecture Notes in Computer Science. Springer, 2003, pp. 368–380. ISBN: 3-540-40524-0. DOI: [10.1007/978-3-540-45069-6_35](https://doi.org/10.1007/978-3-540-45069-6_35) (cit. on p. 2).
- [AGS05] C. Ansótegui, C. P. Gomes, and B. Selman. “The Achilles’ Heel of QBF”. In: *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*. Ed. by M. M. Veloso and S. Kambhampati. AAAI Press / The MIT Press, 2005, pp. 275–281. ISBN: 1-57735-236-X. URL: <http://www.aaai.org/Library/AAAI/2005/aaai05-044.php> (cit. on p. 1).
- [aigsolve] http://abs.informatik.uni-freiburg.de/src/projects_view.php?projectID=19 (cit. on p. 11).
- [BG00] B. Bonet and H. Geffner. “Planning with Incomplete Information as Heuristic Search in Belief Space”. In: *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems, Breckenridge, CO, USA, April 14-17, 2000*. Ed. by S. A. Chien, S. Kambhampati, and C. A. Knoblock. AAAI, 2000, pp. 52–61. ISBN: 1-57735-111-8. URL: <http://www.aaai.org/Library/AIPS/2000/aips00-006.php> (cit. on p. 8).
- [BHS00] P. Balsiger, A. Heuerding, and S. Schwendimann. “A Benchmark Method for the Propositional Modal Logics K, KT, S4”. In: *J. Autom. Reasoning* 24.3 (2000), pp. 297–317. DOI: [10.1023/A:1006249507577](https://doi.org/10.1023/A:1006249507577) (cit. on p. 8).
- [BLB10] R. Brummayer, F. Lonsing, and A. Biere. “Automated Testing and Debugging of SAT and QBF Solvers”. In: *Theory and Applications of Satisfiability Testing - SAT 2010, 13th International Conference, SAT 2010, Edinburgh, UK, July 11-14, 2010. Proceedings*. Ed. by O. Strichman and S. Szeider. Vol. 6175. Lecture Notes in Computer Science. Springer, 2010, pp. 44–57. ISBN: 978-3-642-14185-0. DOI: [10.1007/978-3-642-14186-7_6](https://doi.org/10.1007/978-3-642-14186-7_6) (cit. on p. 1).
- [BLM12] A. Belov, I. Lynce, and J. Marques-Silva. “Towards efficient MUS extraction”. In: *AI Commun.* 25.2 (2012), pp. 97–116. DOI: [10.3233/AIC-2012-0523](https://doi.org/10.3233/AIC-2012-0523) (cit. on p. 8).
- [BLS11] A. Biere, F. Lonsing, and M. Seidl. “Blocked Clause Elimination for QBF”. In: *Automated Deduction - CADE-23 - 23rd International Conference on Automated Deduction, Wroclaw, Poland, July 31 - August 5, 2011. Proceedings*. Ed. by N. Bjørner and V. Sofronie-Stokkermans. Vol. 6803. Lecture Notes in Computer Science. Springer, 2011, pp. 101–115. ISBN: 978-3-642-22437-9. DOI: [10.1007/978-3-642-22438-6_10](https://doi.org/10.1007/978-3-642-22438-6_10) (cit. on pp. 6, 9).
- [BS01] R. Bruni and A. Sassano. “Restoring Satisfiability or Maintaining Unsatisfiability by finding small Unsatisfiable Subformulae”. In: *Electronic Notes in Discrete Mathematics* 9 (2001), pp. 162–173. DOI: [10.1016/S1571-0653\(04\)00320-8](https://doi.org/10.1016/S1571-0653(04)00320-8) (cit. on pp. 1, 3).
- [caqe] <https://www.react.uni-saarland.de/tools/caqe/> (cit. on p. 11).
- [CD91] J. W. Chinneck and E. W. Dravnieks. “Locating Minimal Infeasible Constraint Sets in Linear Programs”. In: *INFORMS Journal on Computing* 3.2 (1991), pp. 157–168. DOI: [10.1287/ijoc.3.2.157](https://doi.org/10.1287/ijoc.3.2.157) (cit. on pp. 1, 3).
- [CEG97] M. Cadoli, T. Eiter, and G. Gottlob. “Default Logic as a Query Language”. In: *IEEE Trans. Knowl. Data Eng.* 9.3 (1997), pp. 448–463. DOI: [10.1109/69.599933](https://doi.org/10.1109/69.599933) (cit. on p. 8).
- [CFL⁺06] S. Coste-Marquis, H. Fargier, J. Lang, et al. “Representing Policies for Quantified Boolean Formulae”. In: *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*. Ed. by P. Doherty, J. Mylopoulos, and C. A. Welty. AAAI Press, 2006, pp. 286–297. ISBN: 978-1-57735-271-6. URL: <http://www.aaai.org/Library/KR/2006/kr06-031.php> (cit. on p. 3).
- [CFLS93] A. Condon, J. Feigenbaum, C. Lund, and P. W. Shor. “Probabilistically checkable debate systems and approximation algorithms for PSPACE-hard functions”. In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*. Ed. by S. R. Kosaraju, D. S. Johnson, and A. Aggarwal. ACM, 1993, pp. 305–314. ISBN: 0-89791-591-7. DOI: [10.1145/167088.167190](https://doi.org/10.1145/167088.167190) (cit. on p. 2).
- [depqbf] <https://lonsing.github.io/depqbf/> (cit. on p. 11).
- [DQF14] J. Du, G. Qi, and X. Fu. “A Practical Fine-grained Approach to Resolving Incoherent OWL 2 DL Terminologies”. In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*. Ed. by J. Li, X. S. Wang, M. N. Garofalakis, et al. ACM, 2014, pp. 919–928.

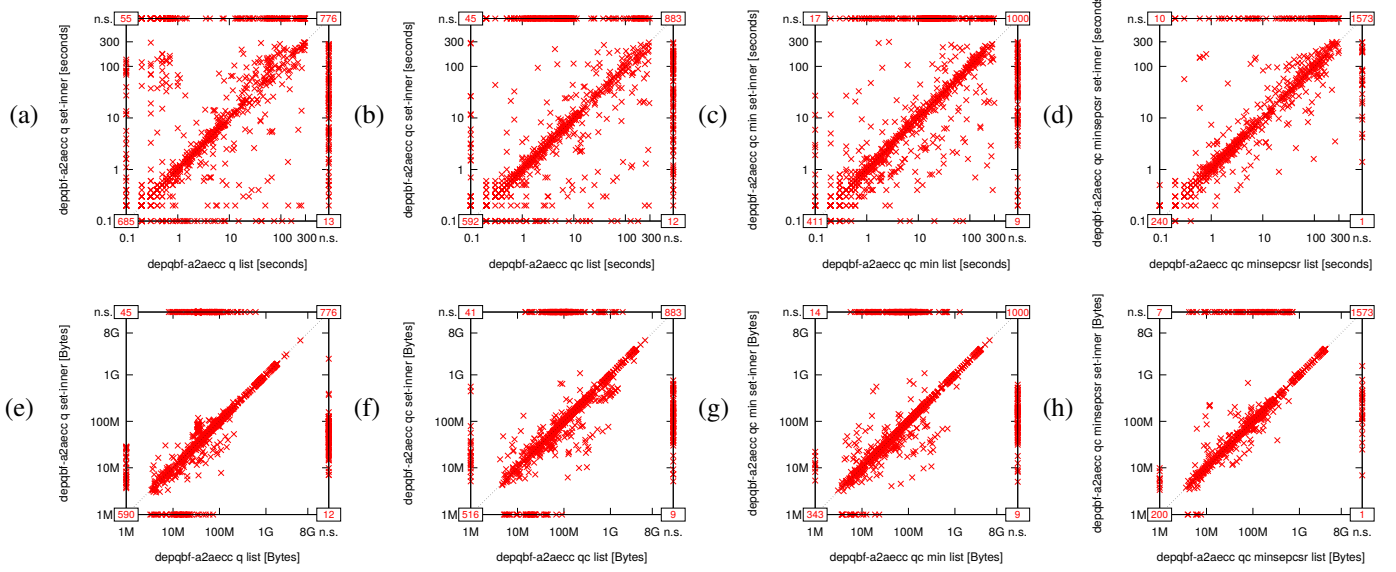


Fig. 5: (a)–(d) Comparing run times for extracting unsatisfiable cores in list semantics (x-axis) vs. set-inner semantics (y-axis) [seconds]: (a) q-cores; (b) qc-cores; (c) q-,c-minimal qc-cores; (d) q-,c-minimal qc-cores with separate CSR. (e)–(h): as (a)–(d) but for memory [Bytes].

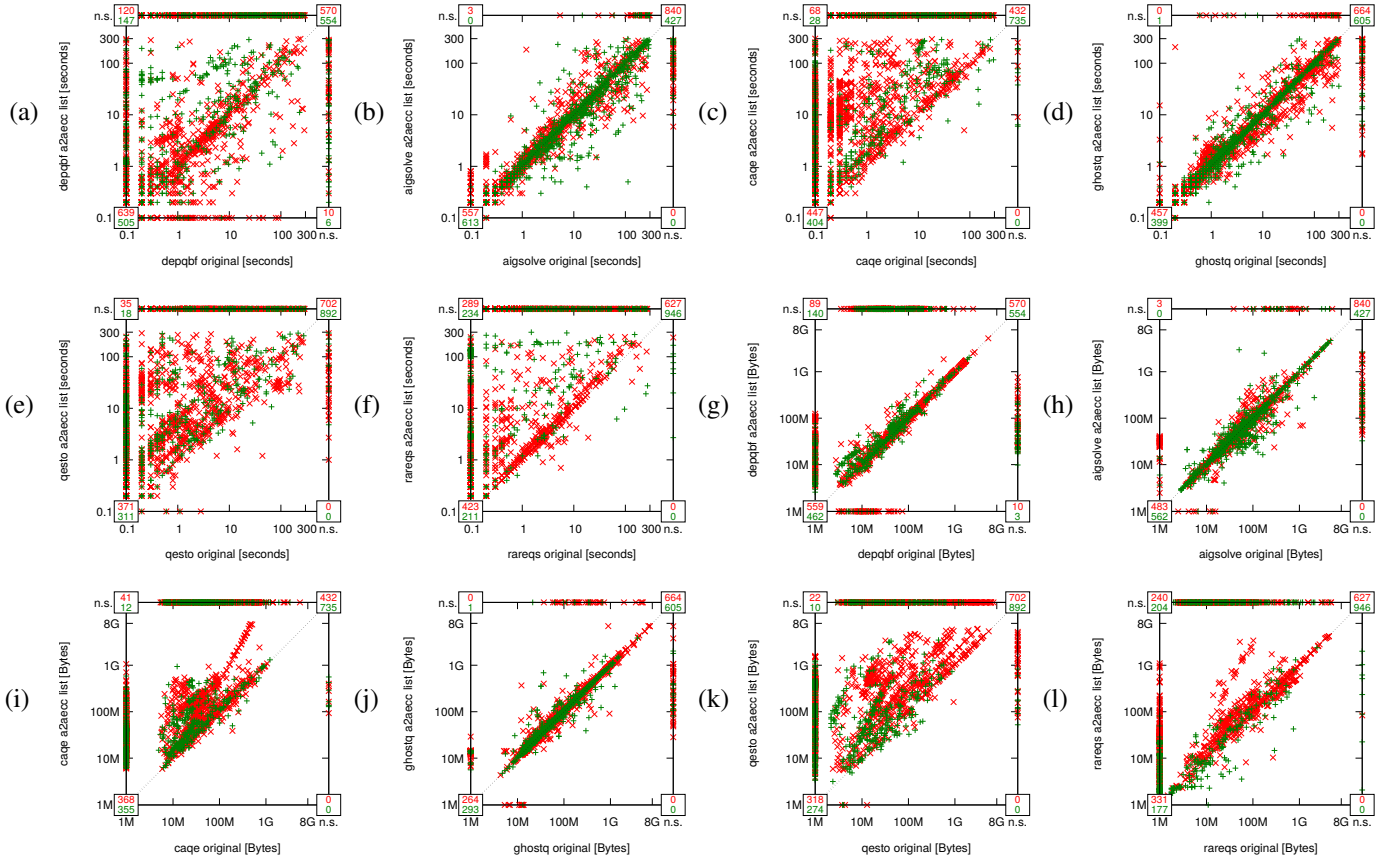


Fig. 6: (a)–(f): Comparing run times for solving original (x-axis) versus transformed (y-axis) instances [seconds]: (a) DepQBF; (b) AIGSolve; (c) CAQE; (d) GhostQ; (e) QESTO; (f) RAREQS. (g)–(l): as (a)–(f) but for memory [Bytes].

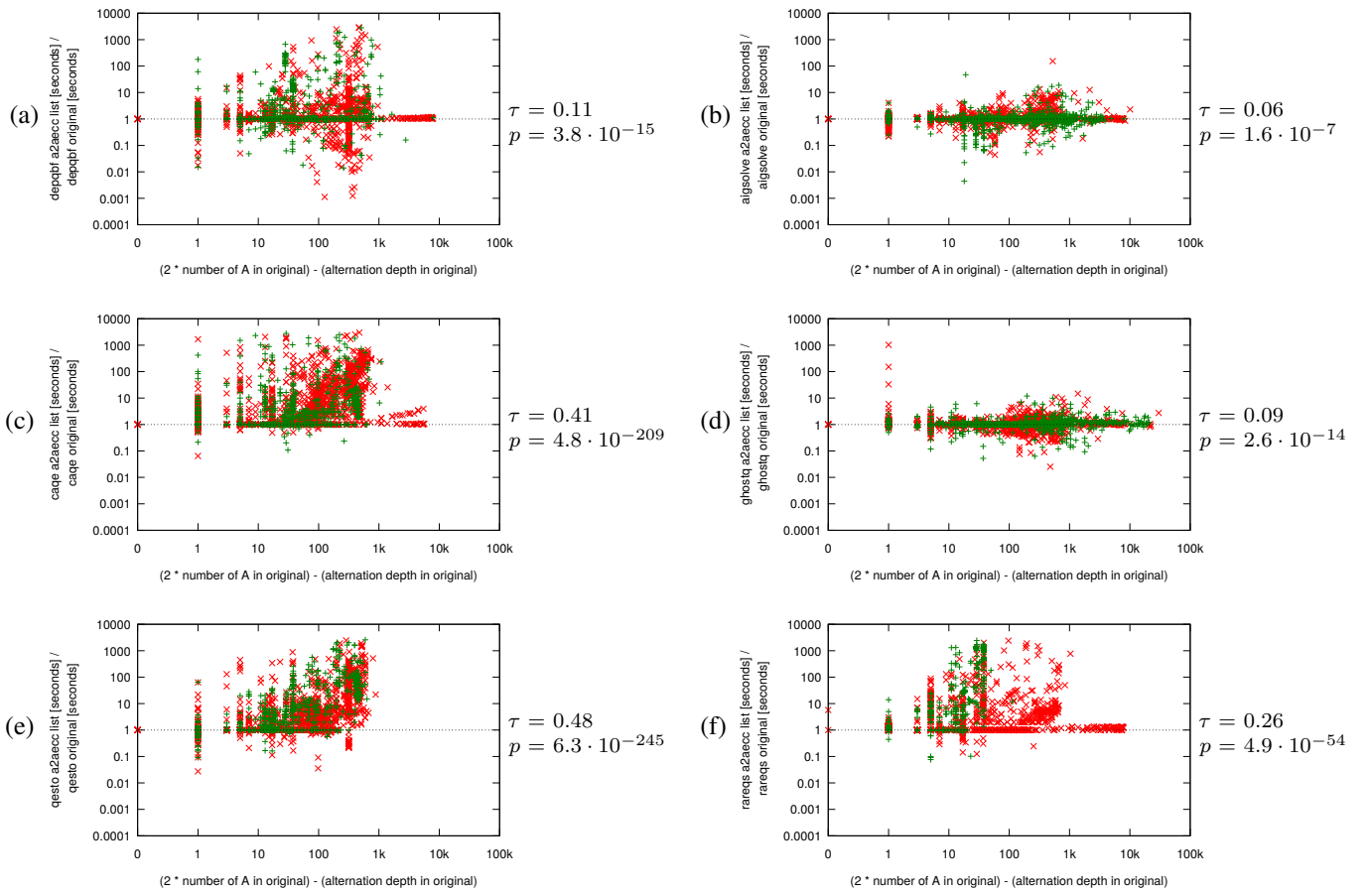


Fig. 7: Ratio of run times [seconds] between solving A2AECC-transformed and original instances (y-axis) depending on the increase in alternation depth between the original and the A2AECC-transformed instances (x-axis): (a) DepQBF; (b) AIGSolve; (c) CAQE; (d) GhostQ; (e) QESTO; (f) RAReQS. Only pairs for which both instances were solved are included. Using `gbutils` [gbutils] we obtained Kendall's rank correlation coefficient τ at p as shown to the right of the plots.

- ISBN: 978-1-4503-2598-1. DOI: [10.1145/2661829.2662046](https://doi.org/10.1145/2661829.2662046) (cit. on p. 2). [gbutils]
- [EETW00] U. Egly, T. Eiter, H. Tompits, and S. Woltran. “Solving Advanced Reasoning Tasks Using Quantified Boolean Formulas”. In: *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA*. Ed. by H. A. Kautz and B. W. Porter. AAAI Press / The MIT Press, 2000, pp. 417–422. ISBN: 0-262-51112-6. URL: <http://www.aaai.org/Library/AAAI/2000/aaai00-064.php> (cit. on p. 1). [GC04]
- [FLMR07] W. Faber, N. Leone, M. Maratea, and F. Ricca. “Looking Back in DLV: Experiments and Comparison to QBF Solvers”. In: *Answer Set Programming, 4th International Workshop, ASP 2007, Porto, Portugal, September 8 and 13, 2007, Proceedings*. 2007 (cit. on p. 8).
- [Fra14] N. Francez. “The Granularity of Meaning in Proof-Theoretic Semantics”. In: *Logical Aspects of Computational Linguistics - 8th International Conference, LACL 2014, Toulouse, France, June 18-20, 2014, Proceedings*. Ed. by N. Asher and S. Soloviev. Vol. 8535. Lecture Notes in Computer Science. Springer, 2014, pp. 96–106. ISBN: 978-3-662-43741-4. DOI: [10.1007/978-3-662-43742-1_8](https://doi.org/10.1007/978-3-662-43742-1_8) (cit. on p. 3). [Gel12]
- <http://cafim.sssup.it/~giulio/software/gbutils/> (cit. on p. 13, 15).
- A. Gurfinkel and M. Chechik. “How Vacuous Is Vacuous?” In: *Tools and Algorithms for the Construction and Analysis of Systems, 10th International Conference, TACAS 2004, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, March 29 - April 2, 2004, Proceedings*. Ed. by K. Jensen and A. Podelski. Vol. 2988. Lecture Notes in Computer Science. Springer, 2004, pp. 451–466. ISBN: 3-540-21299-X. DOI: [10.1007/978-3-540-24730-2_34](https://doi.org/10.1007/978-3-540-24730-2_34) (cit. on p. 2).
- A. V. Gelder. “Contributions to the Theory of Practical Quantified Boolean Formula Solving”. In: *Principles and Practice of Constraint Programming - 18th International Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012, Proceedings*. Ed. by M. Milano. Vol. 7514. Lecture Notes in Computer Science. Springer, 2012, pp. 647–663. ISBN: 978-3-642-33557-0. DOI: [10.1007/978-3-642-33558-7_47](https://doi.org/10.1007/978-3-642-33558-7_47) (cit. on p. 3). <https://www.wkliber.com/ghostq/> (cit. on p. 11). [ghostq]
- E. Giunchiglia, P. Marin, and M. Narizzano. “Reasoning with Quantified Boolean Formulas”. In: *Handbook of Satisfiability*. Ed. by A. Biere, M. Heule, H. van Maaren, and T. Walsh. Vol. 185. Frontiers in Artificial Intelligence and Applications. IOS Press, 2009, pp. 1–14. ISBN: 978-1-58603-929-2. DOI: [10.1007/978-1-58603-929-2_1](https://doi.org/10.1007/978-1-58603-929-2_1) (cit. on p. 1). [GMN09]

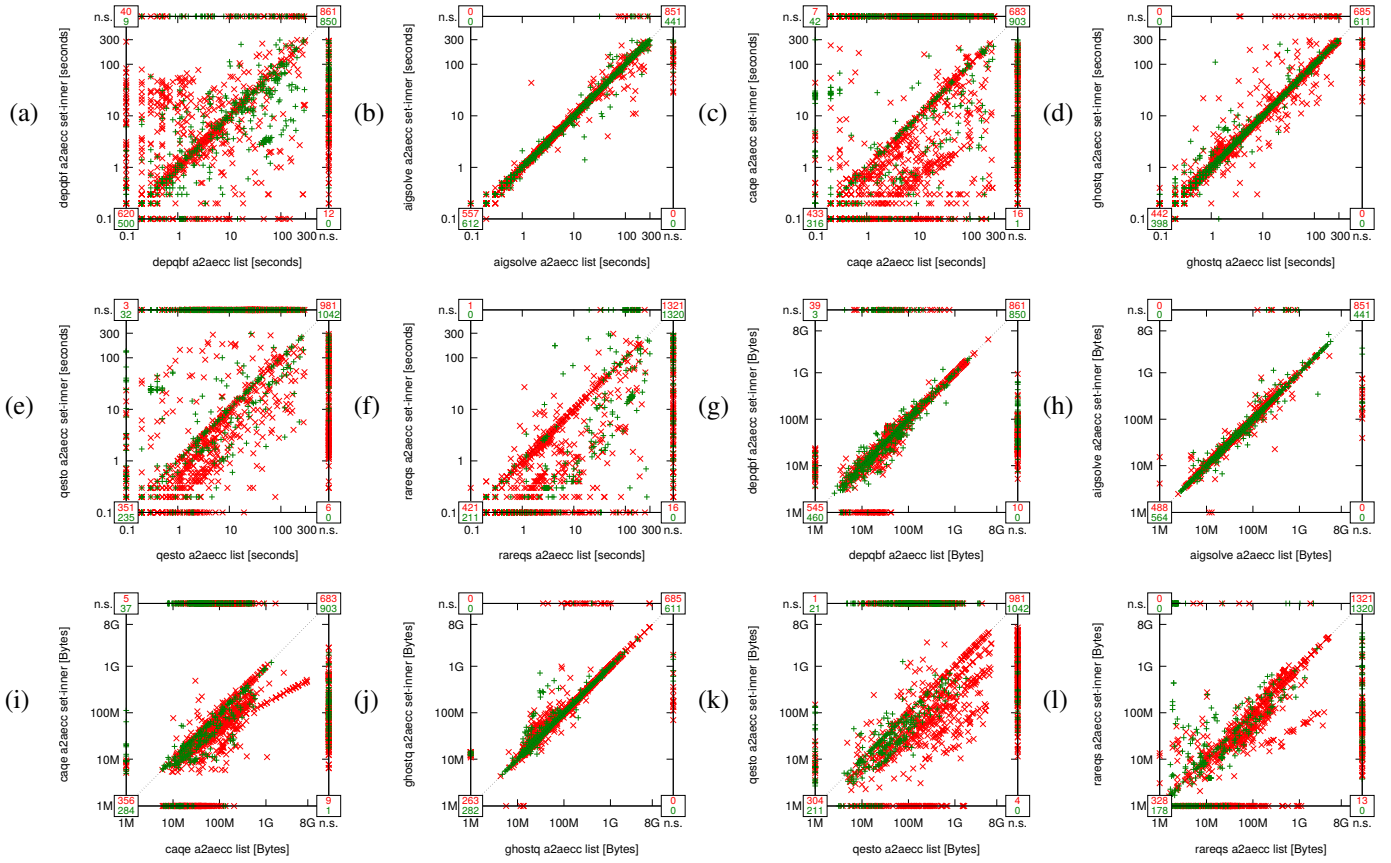


Fig. 8: (a)–(f) Comparing run times for solving transformed instances in list semantics (x-axis) vs. set-inner semantics (y-axis) [seconds]: (a) DepQBF; (b) AIGSolve; (c) CAQE; (d) GhostQ; (e) QESTO; (f) RAREQS. (g)–(l): as (a)–(f) but for memory [Bytes].

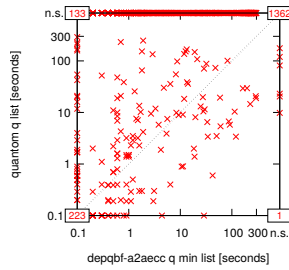


Fig. 9: Comparing run times for finding minimal unsatisfiable q-cores with DepQBF-a2aecc (x-axis) with finding minimum-cardinality sets of universal quantifiers whose weakening to existential quantifiers results in satisfiability (y-axis) [seconds].

- [GMP07] É. Grégoire, B. Mazure, and C. Piette. “MUST: Provide a Finer-Grained Explanation of Unsatisfiability”. In: *Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings*. Ed. by C. Bessière. Vol. 4741. Lecture Notes in Computer Science. Springer, 2007, pp. 317–331. ISBN: 978-3-540-74969-1. DOI: 10.1007/978-3-540-74970-7_24 (cit. on pp. 1, 2, 7).

- [GNPT] E. Giunchiglia, M. Narizzano, L. Pulina, and A. Tacchella. *Quantified Boolean Formulas satisfiability library (QBFLIB)*. <http://www.qbflib.org/> (cit. on pp. 1, 8, 9).
- [GR03] I. P. Gent and A. G. D. Rowley. “Encoding Connect-4 Using Quantified Boolean Formulae”. In: *Modelling and Reformulating Constraint Satisfaction Problems: Towards Systematisation and Automation, Second International Workshop, Kinsale Ireland, September 2003, Proceedings*. Ed. by A. M. Frisch. 2003, pp. 78–93. URL: <https://www-users.cs.york.ac.uk/frisch/Reformulation/03/proceedings.pdf#page=84> (cit. on pp. 1, 8).
- [GW11] S. Grimm and J. Wissmann. “Elimination of Redundancy in Ontologies”. In: *The Semantic Web: Research and Applications - 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29-June 2, 2011, Proceedings, Part I*. Ed. by G. Antoniou, M. Grobelnik, E. P. B. Simperl, et al. Vol. 6643. Lecture Notes in Computer Science. Springer, 2011, pp. 260–274. ISBN: 978-3-642-21033-4. DOI: 10.1007/978-3-642-21034-1_18 (cit. on p. 2).
- [HPS08] M. Horridge, B. Parsia, and U. Sattler. “Laconic and Precise Justifications in OWL”. In: *The Semantic Web - ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings*. Ed. by A. P. Sheth, S. Staab,

- M. Dean, et al. Vol. 5318. Lecture Notes in Computer Science. Springer, 2008, pp. 323–338. ISBN: 978-3-540-88563-4. DOI: [10.1007/978-3-540-88564-1_21](https://doi.org/10.1007/978-3-540-88564-1_21) (cit. on p. 2).
- [IJM13] A. Ignatiev, M. Janota, and J. Marques-Silva. “Quantified Maximum Satisfiability: A Core-Guided Approach”. In: *Theory and Applications of Satisfiability Testing - SAT 2013 - 16th International Conference, Helsinki, Finland, July 8-12, 2013. Proceedings*. Ed. by M. Järvisalo and A. V. Gelder. Vol. 7962. Lecture Notes in Computer Science. Springer, 2013, pp. 250–266. ISBN: 978-3-642-39070-8. DOI: [10.1007/978-3-642-39071-5_19](https://doi.org/10.1007/978-3-642-39071-5_19) (cit. on pp. 1, 2).
- [JKMC12] M. Janota, W. Klieber, J. Marques-Silva, and E. M. Clarke. “Solving QBF with Counterexample Guided Refinement”. In: *Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*. Ed. by A. Cimatti and R. Sebastiani. Vol. 7317. Lecture Notes in Computer Science. Springer, 2012, pp. 114–128. ISBN: 978-3-642-31611-1. DOI: [10.1007/978-3-642-31612-8_10](https://doi.org/10.1007/978-3-642-31612-8_10) (cit. on p. 11).
- [JM15] M. Janota and J. Marques-Silva. “Solving QBF by Clause Selection”. In: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. Ed. by Q. Yang and M. Wooldridge. AAAI Press, 2015, pp. 325–331. ISBN: 978-1-57735-738-4. URL: <http://ijcai.org/Abstract/15/052> (cit. on p. 11).
- [KB09] H. Kleine Büning and U. Bubeck. “Theory of Quantified Boolean Formulas”. In: *Handbook of Satisfiability*. Ed. by A. Biere, M. Heule, H. van Maaren, and T. Walsh. Vol. 185. Frontiers in Artificial Intelligence and Applications. IOS Press, 2009, pp. 735–760. ISBN: 978-1-58603-929-5. DOI: [10.3233/978-1-58603-929-5-735](https://doi.org/10.3233/978-1-58603-929-5-735) (cit. on p. 2).
- [KKF95] H. Kleine Büning, M. Karpinski, and A. Flögel. “Resolution for Quantified Boolean Formulas”. In: *Inf. Comput.* 117.1 (1995), pp. 12–18. DOI: [10.1006/inco.1995.1025](https://doi.org/10.1006/inco.1995.1025) (cit. on p. 3).
- [KLM06] O. Kullmann, I. Lynce, and J. Marques-Silva. “Categorisation of Clauses in Conjunctive Normal Forms: Minimally Unsatisfiable Sub-clause-sets and the Lean Kernel”. In: *Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*. Ed. by A. Biere and C. P. Gomes. Vol. 4121. Lecture Notes in Computer Science. Springer, 2006, pp. 22–35. ISBN: 3-540-37206-7. DOI: [10.1007/11814948_4](https://doi.org/10.1007/11814948_4) (cit. on pp. 1–3, 6).
- [KPG06] A. Kalyanpur, B. Parsia, and B. C. Grau. “Beyond Asserted Axioms: Fine-Grain Justifications for OWL-DL Entailments”. In: *Proceedings of the 2006 International Workshop on Description Logics (DL2006), Windermere, Lake District, UK, May 30 - June 1, 2006*. Ed. by B. Parsia, U. Sattler, and D. Toman. Vol. 189. CEUR Workshop Proceedings. CEUR-WS.org, 2006. URL: http://ceur-ws.org/Vol-189/submission_30.pdf (cit. on p. 2).
- [KPSG06] A. Kalyanpur, B. Parsia, E. Sirin, and B. C. Grau. “Repairing Unsatisfiable Concepts in OWL Ontologies”. In: *The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11-14, 2006, Proceedings*. Ed. by Y. Sure and J. Domingue. Vol. 4011. Lecture Notes in Computer Science. Springer, 2006, pp. 170–184. ISBN: 3-540-34544-2. DOI: [10.1007/11762256_15](https://doi.org/10.1007/11762256_15) (cit. on p. 2).
- [KZ06] H. Kleine Büning and X. Zhao. “Minimal False Quantified Boolean Formulas”. In: *Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*. Ed. by A. Biere and C. P. Gomes. Vol. 4121. Lecture Notes in Computer Science. Springer, 2006, pp. 339–352. ISBN: 3-540-37206-7. DOI: [10.1007/11814948_32](https://doi.org/10.1007/11814948_32) (cit. on pp. 1, 4).
- [LB11] F. Lonsing and A. Biere. “Failed Literal Detection for QBF”. In: *Theory and Applications of Satisfiability Testing - SAT 2011 - 14th International Conference, SAT 2011, Ann Arbor, MI, USA, June 19-22, 2011. Proceedings*. Ed. by K. A. Sakallah and L. Simon. Vol. 6695. Lecture Notes in Computer Science. Springer, 2011, pp. 259–272. ISBN: 978-3-642-21580-3. DOI: [10.1007/978-3-642-21581-0_21](https://doi.org/10.1007/978-3-642-21581-0_21) (cit. on p. 1).
- [LE15] F. Lonsing and U. Egly. “Incrementally Computing Minimal Unsatisfiable Cores of QBFs via a Clause Group Solver API”. In: *Theory and Applications of Satisfiability Testing - SAT 2015 - 18th International Conference, Austin, TX, USA, September 24-27, 2015, Proceedings*. Ed. by M. Heule and S. Weaver. Vol. 9340. Lecture Notes in Computer Science. Springer, 2015, pp. 191–198. ISBN: 978-3-319-24317-7. DOI: [10.1007/978-3-319-24318-4_14](https://doi.org/10.1007/978-3-319-24318-4_14) (cit. on pp. 1, 3, 6, 8).
- [LE17] F. Lonsing and U. Egly. “DepQBF 6.0: A Search-Based QBF Solver Beyond Traditional QCDCL”. In: *Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings*. Ed. by L. de Moura. Vol. 10395. Lecture Notes in Computer Science. Springer, 2017, pp. 371–384. ISBN: 978-3-319-63045-8. DOI: [10.1007/978-3-319-63046-5_23](https://doi.org/10.1007/978-3-319-63046-5_23) (cit. on pp. 1, 8, 11).
- [LES16] F. Lonsing, U. Egly, and M. Seidl. “Q-Resolution with Generalized Axioms”. In: *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*. Ed. by N. Creignou and D. L. Berre. Vol. 9710. Lecture Notes in Computer Science. Springer, 2016, pp. 435–452. ISBN: 978-3-319-40969-6. DOI: [10.1007/978-3-319-40970-2_27](https://doi.org/10.1007/978-3-319-40970-2_27) (cit. on p. 1).
- [LPF+06] N. Leone, G. Pfeifer, W. Faber, et al. “The DLV system for knowledge representation and reasoning”. In: *ACM Trans. Comput. Log.* 7.3 (2006), pp. 499–562. DOI: [10.1145/1149114.1149117](https://doi.org/10.1145/1149114.1149117) (cit. on p. 8).
- [LPSV06] S. C. Lam, J. Z. Pan, D. H. Sleeman, and W. W. Vasconcelos. “A Fine-Grained Approach to Resolving Unsatisfiable Ontologies”. In: *2006 IEEE / WIC / ACM International Conference on Web Intelligence (WI 2006), 18-22 December 2006, Hong Kong, China*. IEEE Computer Society, 2006, pp. 428–434. ISBN: 0-7695-2747-7. DOI: [10.1109/WI.2006.11](https://doi.org/10.1109/WI.2006.11) (cit. on p. 2).
- [LS04] I. Lynce and J. P. M. Silva. “On Computing Minimum Unsatisfiable Cores”. In: *SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada, Online Proceedings*. 2004. URL: <http://www.satisfiability.org/SAT04/programme/110.pdf> (cit. on p. 3).
- [LS08] M. H. Liffiton and K. A. Sakallah. “Algorithms for Computing Minimal Unsatisfiable Subsets of Constraints”. In: *J. Autom. Reasoning* 40.1 (2008), pp. 1–33. DOI: [10.1007/s10817-007-9084-z](https://doi.org/10.1007/s10817-007-9084-z) (cit. on p. 6).

- [Mar12] J. Marques-Silva. “Computing Minimally Unsatisfiable Subformulas: State of the Art and Future Directions”. In: *Multiple-Valued Logic and Soft Computing* 19.1-3 (2012), pp. 163–183. URL: <http://www.oldcitypublishing.com/MVLSC/MVLSCabstracts/MVLSC18.5-6abstracts/MVLSCv18n5-6p617-636Li.html> (cit. on pp. 2, 8).
- [MJ14] J. Marques-Silva and M. Janota. “Computing Minimal Sets on Propositional Formulae I: Problems & Reductions”. In: *CoRR* abs/1402.3011 (2014). URL: <http://arxiv.org/abs/1402.3011> (cit. on p. 2).
- [Nad10] A. Nadel. “Boosting minimal unsatisfiable core extraction”. In: *Proceedings of 10th International Conference on Formal Methods in Computer-Aided Design, FMCAD 2010, Lugano, Switzerland, October 20-23*. Ed. by R. Bloem and N. Sharygina. IEEE, 2010, pp. 221–229. ISBN: 978-1-4577-0734-6. URL: <http://ieeexplore.ieee.org/document/5770953/> (cit. on p. 6).
- [NRS14] A. Nadel, V. Ryvchin, and O. Strichman. “Accelerated Deletion-based Extraction of Minimal Unsatisfiable Cores”. In: *JSAT* 9 (2014), pp. 27–51. URL: <https://satassociation.org/jsat/index.php/jsat/article/view/116> (cit. on p. 6).
- [PQ13] I. Pill and T. Quaritsch. “Behavioral Diagnosis of LTL Specifications at Operator Level”. In: *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*. Ed. by F. Rossi. IJCAI/AAAI, 2013, pp. 1053–1059. ISBN: 978-1-57735-633-2. URL: <http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6595> (cit. on p. 2).
- [PS10] F. Pigorsch and C. Scholl. “An AIG-Based QBF-solver using SAT for preprocessing”. In: *Proceedings of the 47th Design Automation Conference, DAC 2010, Anaheim, California, USA, July 13-18, 2010*. Ed. by S. S. Sapatnekar. ACM, 2010, pp. 170–175. ISBN: 978-1-4503-0002-5. DOI: 10.1145/1837274.1837318 (cit. on p. 11).
- [PV03] G. Pan and M. Y. Vardi. “Optimizing a BDD-Based Modal Solver”. In: *Automated Deduction - CADE-19, 19th International Conference on Automated Deduction Miami Beach, FL, USA, July 28 - August 2, 2003, Proceedings*. Ed. by F. Baader. Vol. 2741. Lecture Notes in Computer Science. Springer, 2003, pp. 75–89. ISBN: 3-540-40559-3. DOI: 10.1007/978-3-540-45085-6_7 (cit. on pp. 1, 8).
- [qbfdd] <https://github.com/aniemetz/qbfdd> (cit. on p. 1).
- [qesto] <http://sat.inesc-id.pt/~mikolas/sw/qesto/> (cit. on p. 11).
- [rareqs] <http://sat.inesc-id.pt/~mikolas/sw/areqs/> (cit. on p. 11).
- [Rei87] R. Reiter. “A Theory of Diagnosis from First Principles”. In: *Artif. Intell.* 32.1 (1987), pp. 57–95. DOI: 10.1016/0004-3702(87)90062-2 (cit. on pp. 1, 6).
- [Rin07] J. Rintanen. “Asymptotically Optimal Encodings of Conformant Planning in QBF”. In: *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*. AAAI Press, 2007, pp. 1045–1050. ISBN: 978-1-57735-323-2. URL: <http://www.aaai.org/Library/AAAI/2007/aaai07-166.php> (cit. on pp. 1, 8).
- [Rin99] J. Rintanen. “Constructing Conditional Plans by a Theorem-Prover”. In: *J. Artif. Intell. Res.* 10 (1999), pp. 323–352. DOI: 10.1613/jair.591 (cit. on p. 1).
- [RPSB12] S. Reimer, F. Pigorsch, C. Scholl, and B. Becker. “Enhanced Integration of QBF Solving Techniques”. In: *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen (MBMV), Kaiserslautern, Germany, March 5-7, 2012*. Ed. by J. Brandt and K. Schneider. Verlag Dr. Kovac, 2012, pp. 133–143 (cit. on pp. 2, 13).
- [RSMB14] S. Reimer, M. Sauer, P. Marin, and B. Becker. “QBF with Soft Variables”. In: *ECEASST* 70 (2014). URL: <http://journal.ub.tu-berlin.de/eceasst/article/view/973> (cit. on pp. 1, 2, 13).
- [SB01] C. Scholl and B. Becker. “Checking Equivalence for Partial Implementations”. In: *Proceedings of the 38th Design Automation Conference, DAC 2001, Las Vegas, NV, USA, June 18-22, 2001*. ACM, 2001, pp. 238–243. ISBN: 1-58113-297-2. DOI: 10.1145/378239.378471 (cit. on p. 1).
- [SC03] S. Schlobach and R. Cornet. “Non-Standard Reasoning Services for the Debugging of Description Logic Terminologies”. In: *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*. Ed. by G. Gottlob and T. Walsh. Morgan Kaufmann, 2003, pp. 355–362. URL: <http://ijcai.org/Proceedings/03/Papers/053.pdf> (cit. on pp. 1, 3).
- [Sch05] S. Schlobach. “Diagnosing Terminologies”. In: *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*. Ed. by M. M. Veloso and S. Kambhampati. AAAI Press / The MIT Press, 2005, pp. 670–675. ISBN: 1-57735-236-X. URL: <http://www.aaai.org/Library/AAAI/2005/aaai05-105.php> (cit. on pp. 1, 6).
- [Sch12] V. Schuppan. “Towards a notion of unsatisfiable and unrealizable cores for LTL”. In: *Sci. Comput. Program.* 77.7-8 (2012), pp. 908–939. DOI: 10.1016/j.scico.2010.11.004 (cit. on pp. 1–3).
- [Sch16a] V. Schuppan. “Enhancing unsatisfiable cores for LTL with information on temporal relevance”. In: *Theor. Comput. Sci.* 655, Part B (2016), pp. 155–192. DOI: 10.1016/j.tcs.2016.01.014 (cit. on p. 2).
- [Sch16b] V. Schuppan. “Extracting unsatisfiable cores for LTL via temporal resolution”. In: *Acta Inf.* 53.3 (2016), pp. 247–299. DOI: 10.1007/s00236-015-0242-1 (cit. on p. 2).
- [Sla14] J. Slaney. “Set-theoretic duality: A fundamental feature of combinatorial optimisation”. In: *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*. Ed. by T. Schaub, G. Friedrich, and B. O’Sullivan. Vol. 263. Frontiers in Artificial Intelligence and Applications. IOS Press, 2014, pp. 843–848. ISBN: 978-1-61499-418-3. DOI: 10.3233/978-1-61499-419-0-843 (cit. on pp. 1, 6).
- [SM73] L. J. Stockmeyer and A. R. Meyer. “Word Problems Requiring Exponential Time: Preliminary Report”. In: *Proceedings of the 5th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1973, Austin, Texas, USA*. Ed. by A. V. Aho, A. Borodin, R. L. Constable, et al. ACM, 1973, pp. 1–9. DOI: 10.1145/800125.804029 (cit. on p. 2).
- [SSJ+03] I. Shlyakhter, R. Seater, D. Jackson, et al. “Debugging Overconstrained Declarative Models Using Unsatisfiable Cores”. In: *18th IEEE International Conference on Automated Software Engineering (ASE 2003), 6-10 October 2003, Montreal, Canada*. IEEE Computer Society, 2003, pp. 94–105. ISBN: 0-7695-2035-9. DOI: 10.1109/ASE.2003.1240298 (cit. on pp. 1–3).

- [Sto76] L. J. Stockmeyer. “The Polynomial-Time Hierarchy”. In: *Theor. Comput. Sci.* 3.1 (1976), pp. 1–22. DOI: [10.1016/0304-3975\(76\)90061-X](https://doi.org/10.1016/0304-3975(76)90061-X) (cit. on p. 2).
- [TCJ08] E. Torlak, F. S. Chang, and D. Jackson. “Finding Minimal Unsatisfiable Cores of Declarative Specifications”. In: *FM 2008: Formal Methods, 15th International Symposium on Formal Methods, Turku, Finland, May 26-30, 2008, Proceedings*. Ed. by J. Cuéllar, T. S. E. Maibaum, and K. Sere. Vol. 5014. Lecture Notes in Computer Science. Springer, 2008, pp. 326–341. ISBN: 978-3-540-68235-6. DOI: [10.1007/978-3-540-68237-0_23](https://doi.org/10.1007/978-3-540-68237-0_23) (cit. on p. 3).
- [Ten17] L. Tentrup. “On Expansion and Resolution in CEGAR Based QBF Solving”. In: *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II*. Ed. by R. Majumdar and V. Kuncak. Vol. 10427. Lecture Notes in Computer Science. Springer, 2017, pp. 475–494. ISBN: 978-3-319-63389-3. DOI: [10.1007/978-3-319-63390-9_25](https://doi.org/10.1007/978-3-319-63390-9_25) (cit. on p. 11).
- [Tur02] H. Turner. “Polynomial-Length Planning Spans the Polynomial Hierarchy”. In: *Logics in Artificial Intelligence, European Conference, JELIA 2002, Cosenza, Italy, September, 23-26, Proceedings*. Ed. by S. Flesca, S. Greco, N. Leone, and G. Ianni. Vol. 2424. Lecture Notes in Computer Science. Springer, 2002, pp. 111–124. ISBN: 3-540-44190-5. DOI: [10.1007/3-540-45757-7_10](https://doi.org/10.1007/3-540-45757-7_10) (cit. on p. 1).
- [Wra76] C. Wrathall. “Complete Sets and the Polynomial-Time Hierarchy”. In: *Theor. Comput. Sci.* 3.1 (1976), pp. 23–33. DOI: [10.1016/0304-3975\(76\)90062-1](https://doi.org/10.1016/0304-3975(76)90062-1) (cit. on p. 2).
- [YM05] Y. Yu and S. Malik. “Validating the result of a Quantified Boolean Formula (QBF) solver: theory and practice”. In: *Proceedings of the 2005 Conference on Asia South Pacific Design Automation, ASP-DAC 2005, Shanghai, China, January 18-21, 2005*. Ed. by T. Tang. ACM Press, 2005, pp. 1047–1051. ISBN: 0-7803-8737-6. DOI: [10.1145/1120725.1120821](https://doi.org/10.1145/1120725.1120821) (cit. on pp. 1, 3).
- [ZSM03] H. Zhang, H. Shen, and F. Manyà. “Exact Algorithms for MAX-SAT”. In: *Electr. Notes Theor. Comput. Sci.* 86.1 (2003), pp. 190–203. DOI: [10.1016/S1571-0661\(04\)80663-7](https://doi.org/10.1016/S1571-0661(04)80663-7) (cit. on p. 2).

APPENDIX A SELECTION OF BENCHMARKS

We briefly describe how we obtained our set of benchmarks instances. There were three main goals.

- 1) The set of benchmarks instances should be randomized at the lowest (leaf-) level of the benchmark family tree.
- 2) The set of benchmarks instances should give equal weight to the immediate subfamilies of each (inner) node of the benchmark family tree, as long as each subfamily has enough members.
- 3) It should be possible to continue supplying slices of benchmark instances to the server running the experiments until the time for running the experiments is up — without knowing in advance when that is or how many slices will be run until then.

To achieve these goals we created a queue of benchmark instances as described below that, when traversed from head to any point between head and tail, ensures the first two goals to a reasonable extent. We then split the benchmark queue into slices of size 25 and fed the slices in ascending order to the server running the experiments. We ran slices 1 through 214. Finally, we removed 8 benchmark instances in which at least one variable occurred both universally and existentially quantified in the prefix. For full details we refer to our experimental data, which includes the shell scripts used, and which is available from <http://schuppan.de/viktor/ictai18/>.

Figure 10 shows the algorithm used to generate the queue of benchmark instances. It is called with the root node of the benchmark family tree as argument. In that tree each node represents a (sub)family of benchmark instances; a leaf node holds a set of benchmark instances; and an inner node holds no benchmark instances but has a non-empty set of child nodes. The algorithm uses the following subroutines.

`is_leaf_node(node n)` returns true iff n is a leaf node.

`get_instances(node n)` returns the set of benchmark instances of leaf node n .

`number_of_subfamilies(node n)` returns the number of subfamilies of node n .

`get_subfamily(node n , natural i)` returns the node corresponding to the i -th subfamily of node n .

`empty_queue()` returns the empty queue.

`dequeue(queue q)` takes a non-empty queue q , dequeues its first element, and returns that element.

`enqueue(queue q , element e)` takes a queue q and an element e and enqueues e to q .

`enqueue_queue(queue q_1 , queue q_2)` takes two queues q_1 and q_2 and returns the concatenation of q_1 and q_2 .

`shuffle(queue q)` takes a queue of benchmark instances q and returns a random permutation of q .

```

1 Function benchmark_queue (node  $n$ ) : queue
2   if is_leaf_node ( $n$ ) then
3     return shuffle (get_instances ( $n$ ));
4   else
5      $k \leftarrow$  number_of_subfamilies ( $n$ );
6     for  $i \leftarrow 1$  to  $k$  do
7       queues_subfamilies[ $i$ ]  $\leftarrow$  benchmark_queue (get_subfamily ( $n, i$ ));
8     end
9     queue  $\leftarrow$  empty_queue ();
10    repeat
11      tmp  $\leftarrow$  empty_queue ();
12      for  $i \leftarrow 1$  to  $k$  do
13        if queues_subfamilies[ $i$ ]  $\neq$  empty_queue () then
14          enqueue (tmp, dequeue (queues_subfamilies[ $i$ ]));
15        end
16      end
17      enqueue_queue (queue, shuffle (tmp));
18    until tmp = empty_queue ();
19    return queue;
20  end
21 end

```

Fig. 10: Generating a queue of benchmark instances.

APPENDIX B

ADDITIONAL TABLES AND PLOTS — MODES OF UNSATISFIABLE CORE EXTRACTION AND OF APPROXIMATING NON-TRIVIALY \forall -TO- \exists REDUCIBLE QUANTIFICATIONS

In the following appendices the modes for unsatisfiable core extraction and for approximating non-trivially \forall -to- \exists reducible quantifications are referred to as follows:

original:	no unsatisfiable core extraction, original version of <code>DepQBF</code> without incremental solving;
none:	no unsatisfiable core extraction, <code>DepQBF-a2aecc</code> with incremental solving enabled;
c:	unsatisfiable c-cores;
q:	unsatisfiable q-cores;
qc:	unsatisfiable qc-cores;
min:	c-minimality for unsatisfiable c-cores, q-minimality for unsatisfiable q-cores, and c,q-minimality for unsatisfiable qc-cores;
minsepcsr:	use separate CSR for minimization;
enuma2e1:	the first method for approximating non-trivially \forall -to- \exists reducible quantifications from Section X;
enuma2e2:	the second method for approximating non-trivially \forall -to- \exists reducible quantifications from Section X;
list:	list semantics; and
set-inner:	set-inner semantics.

APPENDIX C
ADDITIONAL TABLES

In this appendix we show tables as Tab. III, but partitioned by benchmark suite (Tab. VI etc.). (Tab. VI is identical to Tab. III.) Moreover, we show corresponding tables with absolute rather than relative numbers (Tab. V, etc.) and with numbers relative to the number of universal and weakened quantifiers in the unsatisfiable core rather than in the original formula (Tab. VII, etc.). In this appendix the number of benchmarks in a subsection heading or table caption ($n = \dots$) is the number of all benchmarks relevant to the subsection or table that have been found to be unsatisfiable by any solver in our experiments.

A. All ($n = 2528$)

TABLE V: Suite All ($n = 2528$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	1649	486	68	39	66	110	154	157	321	135	52	10	14	37
q set-inner	1516	453	52	65	67	134	166	161	176	128	51	11	14	38
q min list	1139	216	48	62	70	72	92	117	236	115	50	10	14	37
q min set-inner	995	228	45	51	63	65	87	117	129	101	48	10	14	37
qc list	1551	1549	2											
qc set-inner	1414	1413	1											
qc min list	1441	1377	60	3	1									
qc min set-inner	1305	1287	18											
qc minsepcsr list	927	601	63	90	132	25	6	6	3	1				
qc minsepcsr set-inner	854	680	37	55	60	7	7	4	3	1				
enuma2e1 list	986	852	13	42	43	35		1						
enuma2e1 set-inner	930	846	9	25	19	31								
enuma2e2 list	657	406	44	106	58	39	4							
enuma2e2 set-inner	645	486	25	53	42	35	4							

TABLE VI: Suite All ($n = 2528$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	1649	21	465	1	4	4	13	46	11	8	95	159	305	96	291	130
q set-inner	1516	21	432	2	6	5	20	25	22	25	128	140	183	85	290	132
q min list	1139	21	195				5	5	6		37	82	250	96	266	176
q min set-inner	995	21	207				2	1	6		32	78	146	79	245	178
qc list	1551	21	1528	1			1									
qc set-inner	1414	21	1392						1							
qc min list	1441	21	1356	5	3	37	10	5	2	1	1					
qc min set-inner	1305	21	1266	1	1	3	5	3	3	2						
qc minsepcsr list	927	21	580	1	3	9	42	101	37	9	20	44	27	22	11	
qc minsepcsr set-inner	854	21	659		2	1	27	33	12	5	14	37	20	20	3	
enuma2e1 list	986	21	831		1		5	9	8	1	21	15	4	10	7	53
enuma2e1 set-inner	930	21	825		1		1	5			5	9	5	4	1	53
enuma2e2 list	657	21	385	1	2	9	29	38	28	22	36	19	7	10	5	45
enuma2e2 set-inner	645	21	465	2		6	11	25	15	10	15	21	4	6		44

TABLE VII: Suite All ($n = 2528$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	1551	21	124	1404			1	1									
qc set-inner	1414	21	123	1269						1							
qc min list	1441	21	116	1240	3	2	2	46	7	2		2					
qc min set-inner	1305	21	116	1150	1	1	2	6	2	4	2						
qc minsepcsr list	927	21	132	448			1	9	15	21	22	26	38	27	49	98	20
qc minsepcsr set-inner	854	21	133	526				2	7		2	9	35	15	43	42	19

B. Akshay-Chakraborty-John-Shah-Rabe ($n = 2$)

TABLE VIII: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	0													
q set-inner	2											1		1
q min list	0													
q min set-inner	0													
qc list	0													
qc set-inner	2	2												
qc min list	0													
qc min set-inner	2	2												
qc minsepcsr list	0													
qc minsepcsr set-inner	0													
enuma2e1 list	0													
enuma2e1 set-inner	0													
enuma2e2 list	0													
enuma2e2 set-inner	0													

TABLE IX: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	0															
q set-inner	2												1	1		
q min list	0															
q min set-inner	0															
qc list	0															
qc set-inner	2		2													
qc min list	0															
qc min set-inner	2		2													
qc minsepcsr list	0															
qc minsepcsr set-inner	0															
enuma2e1 list	0															
enuma2e1 set-inner	0															
enuma2e2 list	0															
enuma2e2 set-inner	0															

TABLE X: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	0																
qc set-inner	2			2													
qc min list	0																
qc min set-inner	2			2													
qc minsepcsr list	0																
qc minsepcsr set-inner	0																

C. Amendola-Ricca-Truszczyński ($n = 9$)

TABLE XI: Suite Amendola-Ricca-Truszczyński ($n = 9$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	3	3												
q set-inner	4	4												
q min list	0													
q min set-inner	0													
qc list	2	2												
qc set-inner	3	3												
qc min list	2	2												
qc min set-inner	3	3												
qc minsepcsr list	0													
qc minsepcsr set-inner	0													
enuma2e1 list	0													
enuma2e1 set-inner	0													
enuma2e2 list	0													
enuma2e2 set-inner	0													

TABLE XII: Suite Amendola-Ricca-Truszczyński ($n = 9$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	3		3													
q set-inner	4		4													
q min list	0															
q min set-inner	0															
qc list	2		2													
qc set-inner	3		3													
qc min list	2		2													
qc min set-inner	3		3													
qc minsepcsr list	0															
qc minsepcsr set-inner	0															
enuma2e1 list	0															
enuma2e1 set-inner	0															
enuma2e2 list	0															
enuma2e2 set-inner	0															

TABLE XIII: Suite Amendola-Ricca-Truszczyński ($n = 9$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	2			2													
qc set-inner	3			3													
qc min list	2			2													
qc min set-inner	3			3													
qc minsepcsr list	0																
qc minsepcsr set-inner	0																

D. Ansotegui ($n = 12$)

TABLE XIV: Suite Ansotegui ($n = 12$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	9	9												
q set-inner	9	9												
q min list	3	2		1										
q min set-inner	3	2		1										
qc list	7	7												
qc set-inner	7	7												
qc min list	5	5												
qc min set-inner	5	5												
qc minsepcsr list	2	1		1										
qc minsepcsr set-inner	2	1	1											
enuma2e1 list	2	1			1									
enuma2e1 set-inner	2	2												
enuma2e2 list	1	1												
enuma2e2 set-inner	1	1												

TABLE XV: Suite Ansotegui ($n = 12$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	sol- ved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	9		9													
q set-inner	9		9													
q min list	3		2								1					
q min set-inner	3		2								1					
qc list	7		7													
qc set-inner	7		7													
qc min list	5		5													
qc min set-inner	5		5													
qc minsepcsr list	2		1								1					
qc minsepcsr set-inner	2		1							1						
enuma2e1 list	2		1									1				
enuma2e1 set-inner	2		2													
enuma2e2 list	1		1													
enuma2e2 set-inner	1		1													

TABLE XVI: Suite Ansotegui ($n = 12$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	sol- ved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	7			7													
qc set-inner	7			7													
qc min list	5			5													
qc min set-inner	5			5													
qc minsepcsr list	2			1								1					
qc minsepcsr set-inner	2			1							1						

E. Ayari ($n = 48$)

TABLE XVII: Suite Ayari ($n = 48$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						—	—	—	—	—	—	—	—	—
						15	31	63	127	255	511	1023	2047	4096
q list	35	15				1	3	7	2	3	4			
q set-inner	24	4				1	3	7	2	3	4			
q min list	27	1	1	3	2	1	3	7	2	3	4			
q min set-inner	23	2		1		1	3	7	2	3	4			
qc list	34	34												
qc set-inner	24	24												
qc min list	30	29	1											
qc min set-inner	24	24												
qc minsepcsr list	20	2	1	3	1	1	4	4	3	1				
qc minsepcsr set-inner	16	2		1		1	4	4	3	1				
enuma2e1 list	22	16		3		2		1						
enuma2e1 set-inner	18	16		2										
enuma2e2 list	13	8	1	2		2								
enuma2e2 set-inner	10	2	7	1										

TABLE XVIII: Suite Ayari ($n = 48$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	sol- ved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	35		15												19	1
q set-inner	24		4												19	1
q min list	27		1				3	2			1					20
q min set-inner	23		2								1					20
qc list	34		34													
qc set-inner	24		24													
qc min list	30		29				1									
qc min set-inner	24		24													
qc minsepcsr list	20		2				2	2			1		7	5	1	
qc minsepcsr set-inner	16		2								1		7	5	1	
enuma2e1 list	22		16					1		1	4					
enuma2e1 set-inner	18		16					1			1					
enuma2e2 list	13		8				1			1	3					
enuma2e2 set-inner	10		2				1	4	2		1					

TABLE XIX: Suite Ayari ($n = 48$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	sol- ved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	34		1	33													
qc set-inner	24		1	23													
qc min list	30		1	28					1								
qc min set-inner	24		1	23													
qc minsepcsr list	20		1	1				2	2				1				13
qc minsepcsr set-inner	16		1	1									1				13

F. Basler ($n = 75$)

TABLE XX: Suite Basler ($n = 75$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	45							1	26	18				
q set-inner	22							4	11	7				
q min list	28								18	10				
q min set-inner	11								6	5				
qc list	43	43												
qc set-inner	15	15												
qc min list	43	43												
qc min set-inner	15	15												
qc minsepcsr list	27	19	8											
qc minsepcsr set-inner	10	8	2											
enuma2e1 list	6	3		3										
enuma2e1 set-inner	1	1												
enuma2e2 list	2	2												
enuma2e2 set-inner	1	1												

TABLE XXI: Suite Basler ($n = 75$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	45												25	20		
q set-inner	22												14	8		
q min list	28													24	4	
q min set-inner	11													6	5	
qc list	43	43														
qc set-inner	15	15														
qc min list	43	43														
qc min set-inner	15	15														
qc minsepcsr list	27	19			2	6										
qc minsepcsr set-inner	10	8			2											
enuma2e1 list	6	3					1	2								
enuma2e1 set-inner	1	1														
enuma2e2 list	2	2														
enuma2e2 set-inner	1	1														

TABLE XXII: Suite Basler ($n = 75$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	43		43														
qc set-inner	15		15														
qc min list	43		43														
qc min set-inner	15		15														
qc minsepcsr list	27		19					6	2								
qc minsepcsr set-inner	10		8						2								

G. Biere ($n = 25$)

TABLE XXIII: Suite Biere ($n = 25$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8 –	16 –	32 –	64 –	128 –	256 –	512 –	1024 –	2048 –
						15	31	63	127	255	511	1023	2047	4096
q list	13	1			1	1	1	4	3	1	1			
q set-inner	11	1					1	4	3	1	1			
q min list	13					2	2	4	3	1	1			
q min set-inner	11					1	1	4	3	1	1			
qc list	13	13												
qc set-inner	11	11												
qc min list	13	12	1											
qc min set-inner	11	10	1											
qc minsepcsr list	12	11	1											
qc minsepcsr set-inner	10	10												
enuma2e1 list	12	11			1									
enuma2e1 set-inner	10	10												
enuma2e2 list	4	4												
enuma2e2 set-inner	4	4												

TABLE XXIV: Suite Biere ($n = 25$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	sol- ved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	13		1							1						10
q set-inner	11		1													10
q min list	13										1			1		10
q min set-inner	11											1				10
qc list	13		13													
qc set-inner	11		11													
qc min list	13		12				1									
qc min set-inner	11		10					1								
qc minsepcsr list	12		11					1								
qc minsepcsr set-inner	10		10													
enuma2e1 list	12		11								1					
enuma2e1 set-inner	10		10													
enuma2e2 list	4		4													
enuma2e2 set-inner	4		4													

TABLE XXV: Suite Biere ($n = 25$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	sol- ved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	13		10	3													
qc set-inner	11		10	1													
qc min list	13		10	2				1									
qc min set-inner	11		10							1							
qc minsepcsr list	12		10	1						1							
qc minsepcsr set-inner	10		10														

H. Cashmore-Fox-Giunchiglia ($n = 110$)

TABLE XXVI: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	17	17												
q set-inner	17	17												
q min list	14	14												
q min set-inner	15	15												
qc list	16	16												
qc set-inner	16	16												
qc min list	15	15												
qc min set-inner	15	15												
qc minsepcsr list	12	12												
qc minsepcsr set-inner	12	12												
enuma2e1 list	14	14												
enuma2e1 set-inner	14	14												
enuma2e2 list	14	14												
enuma2e2 set-inner	14	14												

TABLE XXVII: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	17		17													
q set-inner	17		17													
q min list	14		14													
q min set-inner	15		15													
qc list	16		16													
qc set-inner	16		16													
qc min list	15		15													
qc min set-inner	15		15													
qc minsepcsr list	12		12													
qc minsepcsr set-inner	12		12													
enuma2e1 list	14		14													
enuma2e1 set-inner	14		14													
enuma2e2 list	14		14													
enuma2e2 set-inner	14		14													

TABLE XXVIII: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	16			16													
qc set-inner	16			16													
qc min list	15			15													
qc min set-inner	15			15													
qc minsepcsr list	12			12													
qc minsepcsr set-inner	12			12													

I. Castellini ($n = 112$)

TABLE XXIX: Suite Castellini ($n = 112$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	106	106												
q set-inner	104	104												
q min list	105	34	28	17	22	4								
q min set-inner	104	43	24	17	18	2								
qc list	105	105												
qc set-inner	104	104												
qc min list	102	102												
qc min set-inner	100	100												
qc minsepcsr list	98	30	27	16	21	4								
qc minsepcsr set-inner	95	39	24	15	15	2								
enuma2e1 list	102	33	6	16	19	28								
enuma2e1 set-inner	103	43	4	14	14	28								
enuma2e2 list	93	33	6	16	18	20								
enuma2e2 set-inner	95	43	4	14	14	20								

TABLE XXX: Suite Castellini ($n = 112$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	sol- ved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	106	106														
q set-inner	104	104														
q min list	105	34									6	27	16	14	8	
q min set-inner	104	43									4	29	14	12	2	
qc list	105	105														
qc set-inner	104	104														
qc min list	102	102														
qc min set-inner	100	100														
qc minsepcsr list	98	30									5	26	15	14	8	
qc minsepcsr set-inner	95	39									4	27	11	12	2	
enuma2e1 list	102	33										4	2	6	5	52
enuma2e1 set-inner	103	43										4		4		52
enuma2e2 list	93	33										4	2	6	4	44
enuma2e2 set-inner	95	43										4		4		44

TABLE XXXI: Suite Castellini ($n = 112$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	sol- ved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	105		105														
qc set-inner	104		104														
qc min list	102		102														
qc min set-inner	100		100														
qc minsepcsr list	98		30									5	26	15	14	8	
qc minsepcsr set-inner	95		39									4	27	11	12	2	

TABLE XXXII: Suite Chen-Interian ($n = 16$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	0													
q set-inner	0													
q min list	0													
q min set-inner	0													
qc list	0													
qc set-inner	0													
qc min list	0													
qc min set-inner	0													
qc minsepcsr list	0													
qc minsepcsr set-inner	0													
enuma2e1 list	0													
enuma2e1 set-inner	0													
enuma2e2 list	0													
enuma2e2 set-inner	0													

TABLE XXXIII: Suite Chen-Interian ($n = 16$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	0															
q set-inner	0															
q min list	0															
q min set-inner	0															
qc list	0															
qc set-inner	0															
qc min list	0															
qc min set-inner	0															
qc minsepcsr list	0															
qc minsepcsr set-inner	0															
enuma2e1 list	0															
enuma2e1 set-inner	0															
enuma2e2 list	0															
enuma2e2 set-inner	0															

TABLE XXXIV: Suite Chen-Interian ($n = 16$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	0																
qc set-inner	0																
qc min list	0																
qc min set-inner	0																
qc minsepcsr list	0																
qc minsepcsr set-inner	0																

K. Diptarama-Jordan-Shinohara ($n = 11$)

TABLE XXXV: Suite Diptarama-Jordan-Shinohara ($n = 11$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	11	6				2		3						
q set-inner	11	6				2		3						
q min list	9	1	2			1	1	3	1					
q min set-inner	8	3				1	1	2	1					
qc list	11	11												
qc set-inner	9	9												
qc min list	10	10												
qc min set-inner	9	9												
qc minsepcsr list	7	3	3	1										
qc minsepcsr set-inner	7	7												
enuma2e1 list	7	3		2	1	1								
enuma2e1 set-inner	7	7												
enuma2e2 list	2	1		1										
enuma2e2 set-inner	4	4												

TABLE XXXVI: Suite Diptarama-Jordan-Shinohara ($n = 11$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	11		6									2	2	1		
q set-inner	11		6									1	3	1		
q min list	9		1					2					4	2		
q min set-inner	8		3										3	2		
qc list	11		11													
qc set-inner	9		9													
qc min list	10		10													
qc min set-inner	9		9													
qc minsepcsr list	7		3					3	1							
qc minsepcsr set-inner	7		7													
enuma2e1 list	7		3						1		2	1				
enuma2e1 set-inner	7		7													
enuma2e2 list	2		1								1					
enuma2e2 set-inner	4		4													

TABLE XXXVII: Suite Diptarama-Jordan-Shinohara ($n = 11$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	11		11														
qc set-inner	9		9														
qc min list	10		10														
qc min set-inner	9		9														
qc minsepcsr list	7		3						2			2					
qc minsepcsr set-inner	7		7														

TABLE XXXVIII: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						15	31	63	127	255	511	1023	2047	4096
q list	36	6	2	1		3	5	7	12					
q set-inner	44	9	3	3		3	6	9	11					
q min list	22		2	3		3	1	3	10					
q min set-inner	23		2	3	1	3	2	4	8					
qc list	36	35	1											
qc set-inner	42	42												
qc min list	36	32	4											
qc min set-inner	42	39	3											
qc minsepcsr list	19	11	2	5		1								
qc minsepcsr set-inner	21	14	1	6										
enuma2e1 list	17	10		6	1									
enuma2e1 set-inner	20	16		3	1									
enuma2e2 list	12	3	2	6		1								
enuma2e2 set-inner	13	8	2	2		1								

TABLE XXXIX: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	36		6				2	1			3	18	6			
q set-inner	44		9		3		1	2			3	21	5			
q min list	22						2	1	2		1	4	12			
q min set-inner	23						2	1	2		2	6	10			
qc list	36		35	1												
qc set-inner	42		42													
qc min list	36		32	1			3									
qc min set-inner	42		39	1			2									
qc minsepcsr list	19		11	1		1	2	3	1							
qc minsepcsr set-inner	21		14			1	2	4								
enuma2e1 list	17		10				3	4								
enuma2e1 set-inner	20		16				1	3								
enuma2e2 list	12		3			1	2	4	1		1					
enuma2e2 set-inner	13		8			1		2	1		1					

TABLE XL: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	36		35				1										
qc set-inner	42		42														
qc min list	36		32			1	3										
qc min set-inner	42		39	1			2										
qc minsepcsr list	19		11			1	1	4	1	1							
qc minsepcsr set-inner	21		14				2	5									

TABLE XLI: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	15	15												
q set-inner	15	15												
q min list	13	12	1											
q min set-inner	13	13												
qc list	15	15												
qc set-inner	15	15												
qc min list	15	15												
qc min set-inner	15	15												
qc minsepcsr list	11	10	1											
qc minsepcsr set-inner	9	9												
enuma2e1 list	10	9		1										
enuma2e1 set-inner	9	9												
enuma2e2 list	9	9												
enuma2e2 set-inner	9	9												

TABLE XLII: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	15		15													
q set-inner	15		15													
q min list	13		12						1							
q min set-inner	13		13													
qc list	15		15													
qc set-inner	15		15													
qc min list	15		15													
qc min set-inner	15		15													
qc minsepcsr list	11		10						1							
qc minsepcsr set-inner	9		9													
enuma2e1 list	10		9								1					
enuma2e1 set-inner	9		9													
enuma2e2 list	9		9													
enuma2e2 set-inner	9		9													

TABLE XLIII: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	15			15													
qc set-inner	15			15													
qc min list	15			15													
qc min set-inner	15			15													
qc minsepcsr list	11			10						1							
qc minsepcsr set-inner	9			9													

TABLE XLIV: Suite Gent-Rowley ($n = 142$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	141	1			3	5	4	15	33	47	33			
q set-inner	141	1			3	4	5	15	33	47	33			
q min list	140			1	2	5	3	15	35	47	32			
q min set-inner	137				3	5	3	15	34	46	31			
qc list	141	141												
qc set-inner	141	141												
qc min list	141	141												
qc min set-inner	141	141												
qc minsepcsr list	135	4	1	35	80	13		2						
qc minsepcsr set-inner	133	56	2	30	43		2							
enuma2e1 list	127	124	2	1										
enuma2e1 set-inner	126	123	2	1										
enuma2e2 list	67	22	4	41										
enuma2e2 set-inner	48	31		14	3									

TABLE XLV: Suite Gent-Rowley ($n = 142$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	141		1									1		2	137	
q set-inner	141		1									1		2	137	
q min list	140										1		1		126	12
q min set-inner	137										1	1			123	12
qc list	141	141														
qc set-inner	141	141														
qc min list	141	141														
qc min set-inner	141	141														
qc minsepcsr list	135	4					24	64	27	6	5	1	1	2	1	
qc minsepcsr set-inner	133	56					25	29	12	4	2	1	2	2		
enuma2e1 list	127	124									3					
enuma2e1 set-inner	126	123									3					
enuma2e2 list	67	22					11	16	7	3	5	3				
enuma2e2 set-inner	48	31							5	4	4	4				

TABLE XLVI: Suite Gent-Rowley ($n = 142$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	141		141														
qc set-inner	141		141														
qc min list	141		141														
qc min set-inner	141		141														
qc minsepcsr list	135	2	2							1					34	89	7
qc minsepcsr set-inner	133	3	53								1				30	40	6

TABLE XLVII: Suite Herbstritt ($n = 157$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	140	19	6		3	22	45	41	4					
q set-inner	141	19	7		3	22	45	41	4					
q min list	53	19				1	11	18	4					
q min set-inner	53	19				1	11	18	4					
qc list	135	135												
qc set-inner	136	136												
qc min list	124	114	9	1										
qc min set-inner	126	123	3											
qc minsepcsr list	53	47	2			3	1							
qc minsepcsr set-inner	53	49				3	1							
enuma2e1 list	50	47			2	1								
enuma2e1 set-inner	50	49				1								
enuma2e2 list	50	41	2	3	3	1								
enuma2e2 set-inner	50	43	2	3	1	1								

TABLE XLVIII: Suite Herbstritt ($n = 157$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	sol- ved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	140	19		1	3	1	3	1		3	51	31			8	19
q set-inner	141	19		2	3	1	3	1		3	51	31			8	19
q min list	53	19									2	1	1	3	8	19
q min set-inner	53	19									2	1	1	3	8	19
qc list	135	19	116													
qc set-inner	136	19	117													
qc min list	124	19	95	3	3	2	2									
qc min set-inner	126	19	104		1	1	1									
qc minsepcsr list	53	19	28				2									
qc minsepcsr set-inner	53	19	30									4				
enuma2e1 list	50	19	28					1	1			1				
enuma2e1 set-inner	50	19	30									1				
enuma2e2 list	50	19	22				2	2	2	1	1	1				
enuma2e2 set-inner	50	19	24				2	1	1	1	1	1				

TABLE XLIX: Suite Herbstritt ($n = 157$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	sol- ved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	135	19	20	96													
qc set-inner	136	19	20	97													
qc min list	124	19	20	75	3	2	1	4									
qc min set-inner	126	19	20	84		1		2									
qc minsepcsr list	53	19	20	8								2		4			
qc minsepcsr set-inner	53	19	20	10										4			

P. Interian ($n = 24$)

TABLE L: Suite Interian ($n = 24$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	16	16												
q set-inner	16	15	1											
q min list	6			3	3									
q min set-inner	1			1										
qc list	16	16												
qc set-inner	16	16												
qc min list	16	16												
qc min set-inner	16	16												
qc minsepcsr list	0													
qc minsepcsr set-inner	0													
enuma2e1 list	0													
enuma2e1 set-inner	0													
enuma2e2 list	0													
enuma2e2 set-inner	0													

TABLE LI: Suite Interian ($n = 24$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	16		16													
q set-inner	16		15						1							
q min list	6											3				
q min set-inner	1										1					
qc list	16		16													
qc set-inner	16		16													
qc min list	16		16													
qc min set-inner	16		16													
qc minsepcsr list	0															
qc minsepcsr set-inner	0															
enuma2e1 list	0															
enuma2e1 set-inner	0															
enuma2e2 list	0															
enuma2e2 set-inner	0															

TABLE LII: Suite Interian ($n = 24$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	16			16													
qc set-inner	16			16													
qc min list	16			16													
qc min set-inner	16			16													
qc minsepcsr list	0																
qc minsepcsr set-inner	0																

TABLE LIII: Suite Jordan-Kaiser ($n = 83$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						—	—	—	—	—	—	—	—	—
						15	31	63	127	255	511	1023	2047	4096
q list	49	29		2	3	3	1	9	2					
q set-inner	49	28		2	3	3	2	9	2					
q min list	38	10	1	6	3	6	1	8	3					
q min set-inner	43	17	2	2	6	4	1	8	3					
qc list	46	46												
qc set-inner	47	47												
qc min list	35	32	2	1										
qc min set-inner	34	32	2											
qc minsepcsr list	34	23	5	3	3									
qc minsepcsr set-inner	33	29	2	2										
enuma2e1 list	38	21		5	11	1								
enuma2e1 set-inner	41	36		2	2	1								
enuma2e2 list	34	21		4	9									
enuma2e2 set-inner	39	36		2	1									

TABLE LIV: Suite Jordan-Kaiser ($n = 83$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	sol- ved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	49	29										2	1	15	2	
q set-inner	49	28										2	3	13	3	
q min list	38	10									1	6	2	15	4	
q min set-inner	43	17									2	5	2	13	4	
qc list	46	46														
qc set-inner	47	47														
qc min list	35	32							1	1	1					
qc min set-inner	34	32								2						
qc minsepcsr list	34	23				4	1				1	5				
qc minsepcsr set-inner	33	29									4					
enuma2e1 list	38	21						1	5		3	6	1	1		
enuma2e1 set-inner	41	36										4	1			
enuma2e2 list	34	21						1	5		3	3	1			
enuma2e2 set-inner	39	36										3				

TABLE LV: Suite Jordan-Kaiser ($n = 83$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	sol- ved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	46	46															
qc set-inner	47	47															
qc min list	35	32								1		2					
qc min set-inner	34	32									2						
qc minsepcsr list	34	23						1	1			4	2	3			
qc minsepcsr set-inner	33	29										2	2				

R. Katz ($n = 8$)

TABLE LVI: Suite Katz ($n = 8$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	4	4												
q set-inner	2	2												
q min list	4	4												
q min set-inner	2	2												
qc list	4	4												
qc set-inner	2	2												
qc min list	2	2												
qc min set-inner	2	2												
qc minsepcsr list	2	2												
qc minsepcsr set-inner	2	2												
enuma2e1 list	4	4												
enuma2e1 set-inner	2	2												
enuma2e2 list	4	4												
enuma2e2 set-inner	2	2												

TABLE LVII: Suite Katz ($n = 8$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	4		4													
q set-inner	2		2													
q min list	4		4													
q min set-inner	2		2													
qc list	4		4													
qc set-inner	2		2													
qc min list	2		2													
qc min set-inner	2		2													
qc minsepcsr list	2		2													
qc minsepcsr set-inner	2		2													
enuma2e1 list	4		4													
enuma2e1 set-inner	2		2													
enuma2e2 list	4		4													
enuma2e2 set-inner	2		2													

TABLE LVIII: Suite Katz ($n = 8$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	4			4													
qc set-inner	2			2													
qc min list	2			2													
qc min set-inner	2			2													
qc minsepcsr list	2			2													
qc minsepcsr set-inner	2			2													

S. Klieber ($n = 15$)

TABLE LIX: Suite Klieber ($n = 15$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8 –	16 –	32 –	64 –	128 –	256 –	512 –	1024 –	2048 –
q list	0					15	31	63	127	255	511	1023	2047	4096
q set-inner	13	9	2	2										
q min list	0													
q min set-inner	0													
qc list	0													
qc set-inner	10	9	1											
qc min list	0													
qc min set-inner	8	6	2											
qc minsepcsr list	0													
qc minsepcsr set-inner	0													
enuma2e1 list	0													
enuma2e1 set-inner	0													
enuma2e2 list	0													
enuma2e2 set-inner	0													

TABLE LX: Suite Klieber ($n = 15$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	sol- ved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	0															
q set-inner	13		9					1	1		2					
q min list	0															
q min set-inner	0															
qc list	0															
qc set-inner	10		9						1							
qc min list	0															
qc min set-inner	8		6						2							
qc minsepcsr list	0															
qc minsepcsr set-inner	0															
enuma2e1 list	0															
enuma2e1 set-inner	0															
enuma2e2 list	0															
enuma2e2 set-inner	0															

TABLE LXI: Suite Klieber ($n = 15$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	sol- ved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	0																
qc set-inner	10			9						1							
qc min list	0																
qc min set-inner	8			6						2							
qc minsepcsr list	0																
qc minsepcsr set-inner	0																

TABLE LXII: Suite Kontchakov ($n = 70$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	18	9	1	3	4	1								
q set-inner	17	2		12	2	1								
q min list	0													
q min set-inner	0													
qc list	17	17												
qc set-inner	16	16												
qc min list	17	17												
qc min set-inner	16	16												
qc minsepcsr list	0													
qc minsepcsr set-inner	0													
enuma2e1 list	0													
enuma2e1 set-inner	0													
enuma2e2 list	0													
enuma2e2 set-inner	0													

TABLE LXIII: Suite Kontchakov ($n = 70$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	18		9		1		3	4			1					
q set-inner	17		2				12	2	1							
q min list	0															
q min set-inner	0															
qc list	17		17													
qc set-inner	16		16													
qc min list	17		17													
qc min set-inner	16		16													
qc minsepcsr list	0															
qc minsepcsr set-inner	0															
enuma2e1 list	0															
enuma2e1 set-inner	0															
enuma2e2 list	0															
enuma2e2 set-inner	0															

TABLE LXIV: Suite Kontchakov ($n = 70$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	17			17													
qc set-inner	16			16													
qc min list	17			17													
qc min set-inner	16			16													
qc minsepcsr list	0																
qc minsepcsr set-inner	0																

TABLE LXV: Suite Kronegger-Pfandler-Pichler ($n = 133$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	86	28	3	16	19	13	7							
q set-inner	87	29	2	16	19	14	7							
q min list	47	5	2	11	10	10	9							
q min set-inner	49	5	3	10	10	12	9							
qc list	83	83												
qc set-inner	84	84												
qc min list	81	81												
qc min set-inner	81	81												
qc minsepcsr list	44	39	1	4										
qc minsepcsr set-inner	43	43												
enuma2e1 list	46	46												
enuma2e1 set-inner	47	47												
enuma2e2 list	30	28	1	1										
enuma2e2 set-inner	30	30												

TABLE LXVI: Suite Kronegger-Pfandler-Pichler ($n = 133$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	86	28									1	18	20	7	11	1
q set-inner	87	29										18	22	6	11	1
q min list	47	5										4	10	13	14	1
q min set-inner	49	5										5	10	14	14	1
qc list	83	83														
qc set-inner	84	84														
qc min list	81	81														
qc min set-inner	81	81														
qc minsepcsr list	44	39										3	2			
qc minsepcsr set-inner	43	43														
enuma2e1 list	46	46														
enuma2e1 set-inner	47	47														
enuma2e2 list	30	28										2				
enuma2e2 set-inner	30	30														

TABLE LXVII: Suite Kronegger-Pfandler-Pichler ($n = 133$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	83	2	81														
qc set-inner	84	2	82														
qc min list	81	2	79														
qc min set-inner	81	2	79														
qc minsepcsr list	44	2	37										2	3			
qc minsepcsr set-inner	43	2	41														

TABLE LXVIII: Suite Lahiri-Seshia ($n = 1$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	0													
q set-inner	0													
q min list	0													
q min set-inner	0													
qc list	0													
qc set-inner	0													
qc min list	0													
qc min set-inner	0													
qc minsepcsr list	0													
qc minsepcsr set-inner	0													
enuma2e1 list	0													
enuma2e1 set-inner	0													
enuma2e2 list	0													
enuma2e2 set-inner	0													

TABLE LXIX: Suite Lahiri-Seshia ($n = 1$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	0															
q set-inner	0															
q min list	0															
q min set-inner	0															
qc list	0															
qc set-inner	0															
qc min list	0															
qc min set-inner	0															
qc minsepcsr list	0															
qc minsepcsr set-inner	0															
enuma2e1 list	0															
enuma2e1 set-inner	0															
enuma2e2 list	0															
enuma2e2 set-inner	0															

TABLE LXX: Suite Lahiri-Seshia ($n = 1$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	0																
qc set-inner	0																
qc min list	0																
qc min set-inner	0																
qc minsepcsr list	0																
qc minsepcsr set-inner	0																

TABLE LXXI: Suite Lee-Jiang ($n = 2$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	0													
q set-inner	0													
q min list	0													
q min set-inner	0													
qc list	0													
qc set-inner	0													
qc min list	0													
qc min set-inner	0													
qc minsepcsr list	0													
qc minsepcsr set-inner	0													
enuma2e1 list	0													
enuma2e1 set-inner	0													
enuma2e2 list	0													
enuma2e2 set-inner	0													

TABLE LXXII: Suite Lee-Jiang ($n = 2$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	sol- ved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	0															
q set-inner	0															
q min list	0															
q min set-inner	0															
qc list	0															
qc set-inner	0															
qc min list	0															
qc min set-inner	0															
qc minsepcsr list	0															
qc minsepcsr set-inner	0															
enuma2e1 list	0															
enuma2e1 set-inner	0															
enuma2e2 list	0															
enuma2e2 set-inner	0															

TABLE LXXIII: Suite Lee-Jiang ($n = 2$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	sol- ved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	0																
qc set-inner	0																
qc min list	0																
qc min set-inner	0																
qc minsepcsr list	0																
qc minsepcsr set-inner	0																

X. Letombe ($n = 85$)

TABLE LXXIV: Suite Letombe ($n = 85$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	30	16								14				
q set-inner	22	10								12				
q min list	23	12								11				
q min set-inner	9	5								4				
qc list	28	28												
qc set-inner	22	22												
qc min list	28	28												
qc min set-inner	22	22												
qc minsepcsr list	20	20												
qc minsepcsr set-inner	9	9												
enuma2e1 list	10	10												
enuma2e1 set-inner	5	5												
enuma2e2 list	10	10												
enuma2e2 set-inner	5	5												

TABLE LXXV: Suite Letombe ($n = 85$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	sol- ved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	30		16											1	13	
q set-inner	22		10												12	
q min list	23		12												11	
q min set-inner	9		5												4	
qc list	28		28													
qc set-inner	22		22													
qc min list	28		28													
qc min set-inner	22		22													
qc minsepcsr list	20		20													
qc minsepcsr set-inner	9		9													
enuma2e1 list	10		10													
enuma2e1 set-inner	5		5													
enuma2e2 list	10		10													
enuma2e2 set-inner	5		5													

TABLE LXXVI: Suite Letombe ($n = 85$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	sol- ved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	28			28													
qc set-inner	22			22													
qc min list	28			28													
qc min set-inner	22			22													
qc minsepcsr list	20			20													
qc minsepcsr set-inner	9			9													

TABLE LXXVII: Suite Letz ($n = 9$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	9	9												
q set-inner	9	9												
q min list	9	9												
q min set-inner	9	9												
qc list	9	9												
qc set-inner	9	9												
qc min list	9	9												
qc min set-inner	9	9												
qc minsepcsr list	9	9												
qc minsepcsr set-inner	9	9												
enuma2e1 list	9	9												
enuma2e1 set-inner	9	9												
enuma2e2 list	9	9												
enuma2e2 set-inner	9	9												

TABLE LXXVIII: Suite Letz ($n = 9$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	9		9													
q set-inner	9		9													
q min list	9		9													
q min set-inner	9		9													
qc list	9		9													
qc set-inner	9		9													
qc min list	9		9													
qc min set-inner	9		9													
qc minsepcsr list	9		9													
qc minsepcsr set-inner	9		9													
enuma2e1 list	9		9													
enuma2e1 set-inner	9		9													
enuma2e2 list	9		9													
enuma2e2 set-inner	9		9													

TABLE LXXIX: Suite Letz ($n = 9$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	9			9													
qc set-inner	9			9													
qc min list	9			9													
qc min set-inner	9			9													
qc minsepcsr list	9			9													
qc minsepcsr set-inner	9			9													

TABLE LXXX: Suite Ling ($n = 3$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	3	3												
q set-inner	3	3												
q min list	2	2												
q min set-inner	3	3												
qc list	3	3												
qc set-inner	3	3												
qc min list	2	2												
qc min set-inner	1	1												
qc minsepcsr list	1	1												
qc minsepcsr set-inner	1	1												
enuma2e1 list	2	2												
enuma2e1 set-inner	3	3												
enuma2e2 list	2	2												
enuma2e2 set-inner	3	3												

TABLE LXXXI: Suite Ling ($n = 3$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	3		3													
q set-inner	3		3													
q min list	2		2													
q min set-inner	3		3													
qc list	3		3													
qc set-inner	3		3													
qc min list	2		2													
qc min set-inner	1		1													
qc minsepcsr list	1		1													
qc minsepcsr set-inner	1		1													
enuma2e1 list	2		2													
enuma2e1 set-inner	3		3													
enuma2e2 list	2		2													
enuma2e2 set-inner	3		3													

TABLE LXXXII: Suite Ling ($n = 3$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	3			3													
qc set-inner	3			3													
qc min list	2			2													
qc min set-inner	1			1													
qc minsepcsr list	1			1													
qc minsepcsr set-inner	1			1													

TABLE LXXXIII: Suite Mangassarian-Veneris ($n = 60$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	49	49												
q set-inner	48	48												
q min list	27	27												
q min set-inner	33	33												
qc list	48	48												
qc set-inner	47	47												
qc min list	48	48												
qc min set-inner	47	47												
qc minsepcsr list	25	25												
qc minsepcsr set-inner	32	32												
enuma2e1 list	27	27												
enuma2e1 set-inner	33	33												
enuma2e2 list	27	27												
enuma2e2 set-inner	33	33												

TABLE LXXXIV: Suite Mangassarian-Veneris ($n = 60$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	49		49													
q set-inner	48		48													
q min list	27		27													
q min set-inner	33		33													
qc list	48		48													
qc set-inner	47		47													
qc min list	48		48													
qc min set-inner	47		47													
qc minsepcsr list	25		25													
qc minsepcsr set-inner	32		32													
enuma2e1 list	27		27													
enuma2e1 set-inner	33		33													
enuma2e2 list	27		27													
enuma2e2 set-inner	33		33													

TABLE LXXXV: Suite Mangassarian-Veneris ($n = 60$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	48			48													
qc set-inner	47			47													
qc min list	48			48													
qc min set-inner	47			47													
qc minsepcsr list	25			25													
qc minsepcsr set-inner	32			32													

AB. MayerEichberger-Saffidine ($n = 3$)

TABLE LXXXVI: Suite MayerEichberger-Saffidine ($n = 3$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	1		1											
q set-inner	2		2											
q min list	0													
q min set-inner	0													
qc list	1	1												
qc set-inner	1	1												
qc min list	1	1												
qc min set-inner	1	1												
qc minsepcsr list	0													
qc minsepcsr set-inner	0													
enuma2e1 list	0													
enuma2e1 set-inner	0													
enuma2e2 list	0													
enuma2e2 set-inner	0													

TABLE LXXXVII: Suite MayerEichberger-Saffidine ($n = 3$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	1						1									
q set-inner	2						2									
q min list	0															
q min set-inner	0															
qc list	1		1													
qc set-inner	1		1													
qc min list	1		1													
qc min set-inner	1		1													
qc minsepcsr list	0															
qc minsepcsr set-inner	0															
enuma2e1 list	0															
enuma2e1 set-inner	0															
enuma2e2 list	0															
enuma2e2 set-inner	0															

TABLE LXXXVIII: Suite MayerEichberger-Saffidine ($n = 3$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	1			1													
qc set-inner	1			1													
qc min list	1			1													
qc min set-inner	1			1													
qc minsepcsr list	0																
qc minsepcsr set-inner	0																

AC. *Messinger* ($n = 0$)

TABLE LXXXIX: Suite *Messinger* ($n = 0$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	0													
q set-inner	0													
q min list	0													
q min set-inner	0													
qc list	0													
qc set-inner	0													
qc min list	0													
qc min set-inner	0													
qc minsepcsr list	0													
qc minsepcsr set-inner	0													
enuma2e1 list	0													
enuma2e1 set-inner	0													
enuma2e2 list	0													
enuma2e2 set-inner	0													

TABLE XC: Suite *Messinger* ($n = 0$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	sol- ved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	0															
q set-inner	0															
q min list	0															
q min set-inner	0															
qc list	0															
qc set-inner	0															
qc min list	0															
qc min set-inner	0															
qc minsepcsr list	0															
qc minsepcsr set-inner	0															
enuma2e1 list	0															
enuma2e1 set-inner	0															
enuma2e2 list	0															
enuma2e2 set-inner	0															

TABLE XCI: Suite *Messinger* ($n = 0$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	sol- ved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	0																
qc set-inner	0																
qc min list	0																
qc min set-inner	0																
qc minsepcsr list	0																
qc minsepcsr set-inner	0																

TABLE XCII: Suite Miller-Marin ($n = 189$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	158				5	7	13	21	21	26	4	10	14	37
q set-inner	160				5	7	13	21	21	28	4	10	14	37
q min list	158				5	7	13	20	22	26	4	10	14	37
q min set-inner	160				5	7	13	20	22	28	4	10	14	37
qc list	150	150												
qc set-inner	149	149												
qc min list	126	126												
qc min set-inner	127	127												
qc minsepcsr list	123	123												
qc minsepcsr set-inner	122	122												
enuma2e1 list	86	86												
enuma2e1 set-inner	77	77												
enuma2e2 list	14	14												
enuma2e2 set-inner	15	15												

TABLE XCIII: Suite Miller-Marin ($n = 189$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	sol- ved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	158														62	96
q set-inner	160														62	98
q min list	158														48	110
q min set-inner	160														48	112
qc list	150	150														
qc set-inner	149	149														
qc min list	126	126														
qc min set-inner	127	127														
qc minsepcsr list	123	123														
qc minsepcsr set-inner	122	122														
enuma2e1 list	86	86														
enuma2e1 set-inner	77	77														
enuma2e2 list	14	14														
enuma2e2 set-inner	15	15														

TABLE XCIV: Suite Miller-Marin ($n = 189$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	sol- ved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	150		88	62													
qc set-inner	149		87	62													
qc min list	126		80	46													
qc min set-inner	127		80	47													
qc minsepcsr list	123		93	30													
qc minsepcsr set-inner	122		93	29													

TABLE XCV: Suite Miller-Scholl-Becker ($n = 160$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	134			2	11	43	53	24	1					
q set-inner	160				8	63	61	27	1					
q min list	92				8	24	38	22						
q min set-inner	97				8	22	36	28	3					
qc list	131	131												
qc set-inner	160	160												
qc min list	130	129	1											
qc min set-inner	160	160												
qc minsepcsr list	85	68	2	11	4									
qc minsepcsr set-inner	87	87												
enuma2e1 list	113	110			3									
enuma2e1 set-inner	116	116												
enuma2e2 list	74	35	7	13	17	2								
enuma2e2 set-inner	72	72												

TABLE XCVI: Suite Miller-Scholl-Becker ($n = 160$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	134							2	4	3	11	67	47			
q set-inner	160							1	5	12	32	50	60			
q min list	92											25	65	2		
q min set-inner	97											23	67	7		
qc list	131	131														
qc set-inner	160	160														
qc min list	130	129					1									
qc min set-inner	160	160														
qc minsepcsr list	85	68					1	9	5	1	1					
qc minsepcsr set-inner	87	87														
enuma2e1 list	113	110									3					
enuma2e1 set-inner	116	116														
enuma2e2 list	74	35					4	4	5	12	12	2				
enuma2e2 set-inner	72	72														

TABLE XCVII: Suite Miller-Scholl-Becker ($n = 160$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	131		131														
qc set-inner	160		160														
qc min list	130		129					1									
qc min set-inner	160		160														
qc minsepcsr list	85		68						1	4	6	6					
qc minsepcsr set-inner	87		87														

TABLE XCVIII: Suite Mneimneh-Sakallah ($n = 44$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	3	3												
q set-inner	3	3												
q min list	3			1	1	1								
q min set-inner	3		1	1	1									
qc list	3	3												
qc set-inner	3	3												
qc min list	3	1	1	1										
qc min set-inner	3	3												
qc minsepcsr list	3			2	1									
qc minsepcsr set-inner	3	3												
enuma2e1 list	2			1	1									
enuma2e1 set-inner	2	2												
enuma2e2 list	2			1	1									
enuma2e2 set-inner	2	2												

TABLE XCIX: Suite Mneimneh-Sakallah ($n = 44$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	3		3													
q set-inner	3		3													
q min list	3															
q min set-inner	3								2		3	1				
qc list	3		3													
qc set-inner	3		3													
qc min list	3		1					1	1							
qc min set-inner	3		3													
qc minsepcsr list	3										3					
qc minsepcsr set-inner	3		3													
enuma2e1 list	2										2					
enuma2e1 set-inner	2		2													
enuma2e2 list	2										2					
enuma2e2 set-inner	2		2													

TABLE C: Suite Mneimneh-Sakallah ($n = 44$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	3			3													
qc set-inner	3			3													
qc min list	3			1					1	1							
qc min set-inner	3			3													
qc minsepcsr list	3											3					
qc minsepcsr set-inner	3			3													

TABLE CI: Suite Narizzano ($n = 78$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8 – 15	16 – 31	32 – 63	64 – 127	128 – 255	256 – 511	512 – 1023	1024 – 2047	2048 – 4096
q list	50	16	31				1	2						
q set-inner	48	12	11	16	6		1	2						
q min list	8		1	2	2		1	2						
q min set-inner	3						1	2						
qc list	36	36												
qc set-inner	42	42												
qc min list	3	3												
qc min set-inner	3	3												
qc minsepcsr list	3	3												
qc minsepcsr set-inner	3	3												
enuma2e1 list	3	3												
enuma2e1 set-inner	3	3												
enuma2e2 list	3	3												
enuma2e2 set-inner	3	3												

TABLE CII: Suite Narizzano ($n = 78$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	50		16					31								3
q set-inner	48		12					11	9	7	6					3
q min list	8								1		4					3
q min set-inner	3															3
qc list	36		36													
qc set-inner	42		42													
qc min list	3		3													
qc min set-inner	3		3													
qc minsepcsr list	3		3													
qc minsepcsr set-inner	3		3													
enuma2e1 list	3		3													
enuma2e1 set-inner	3		3													
enuma2e2 list	3		3													
enuma2e2 set-inner	3		3													

TABLE CIII: Suite Narizzano ($n = 78$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	36		3	33													
qc set-inner	42		3	39													
qc min list	3		3														
qc min set-inner	3		3														
qc minsepcsr list	3		3														
qc minsepcsr set-inner	3		3														

TABLE CIV: Suite Palacios ($n = 9$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	2	2												
q set-inner	2	2												
q min list	1				1									
q min set-inner	1			1										
qc list	2	2												
qc set-inner	2	2												
qc min list	1	1												
qc min set-inner	1	1												
qc minsepcsr list	1				1									
qc minsepcsr set-inner	1			1										
enuma2e1 list	1					1								
enuma2e1 set-inner	1					1								
enuma2e2 list	1					1								
enuma2e2 set-inner	0													

TABLE CV: Suite Palacios ($n = 9$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	2		2													
q set-inner	2		2													
q min list	1													1		
q min set-inner	1										1					
qc list	2		2													
qc set-inner	2		2													
qc min list	1		1													
qc min set-inner	1		1													
qc minsepcsr list	1													1		
qc minsepcsr set-inner	1										1					
enuma2e1 list	1															1
enuma2e1 set-inner	1															1
enuma2e2 list	1															1
enuma2e2 set-inner	0															

TABLE CVI: Suite Palacios ($n = 9$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	2			2													
qc set-inner	2			2													
qc min list	1			1													
qc min set-inner	1			1													
qc minsepcsr list	1														1		
qc minsepcsr set-inner	1											1					

TABLE CVII: Suite Pan ($n = 89$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	61	25	12	11	12	1								
q set-inner	64	24	12	12	15	1								
q min list	36	14	3	9	9	1								
q min set-inner	37	15	3	10	9									
qc list	61	61												
qc set-inner	61	61												
qc min list	61	61												
qc min set-inner	61	61												
qc minsepcsr list	31	30	1											
qc minsepcsr set-inner	32	31	1											
enuma2e1 list	29	28	1											
enuma2e1 set-inner	31	30	1											
enuma2e2 list	26	25	1											
enuma2e2 set-inner	30	29	1											

TABLE CVIII: Suite Pan ($n = 89$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	sol- ved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	61		25				4	5	2		12	3	10			
q set-inner	64		24				4	5	2		16	3	10			
q min list	36		14						2		8	1	11			
q min set-inner	37		15						2		9		11			
qc list	61		61													
qc set-inner	61		61													
qc min list	61		61													
qc min set-inner	61		61													
qc minsepcsr list	31		30								1					
qc minsepcsr set-inner	32		31								1					
enuma2e1 list	29		28								1					
enuma2e1 set-inner	31		30								1					
enuma2e2 list	26		25								1					
enuma2e2 set-inner	30		29								1					

TABLE CIX: Suite Pan ($n = 89$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	sol- ved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	61		61														
qc set-inner	61		61														
qc min list	61		61														
qc min set-inner	61		61														
qc minsepcsr list	31		30									1					
qc minsepcsr set-inner	32		31									1					

TABLE CX: Suite Peitl ($n = 10$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	10	10												
q set-inner	10	10												
q min list	10	10												
q min set-inner	10	10												
qc list	10	10												
qc set-inner	10	10												
qc min list	10	10												
qc min set-inner	10	10												
qc minsepcsr list	10	10												
qc minsepcsr set-inner	10	10												
enuma2e1 list	10	10												
enuma2e1 set-inner	10	10												
enuma2e2 list	10	10												
enuma2e2 set-inner	10	10												

TABLE CXI: Suite Peitl ($n = 10$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	10		10													
q set-inner	10		10													
q min list	10		10													
q min set-inner	10		10													
qc list	10		10													
qc set-inner	10		10													
qc min list	10		10													
qc min set-inner	10		10													
qc minsepcsr list	10		10													
qc minsepcsr set-inner	10		10													
enuma2e1 list	10		10													
enuma2e1 set-inner	10		10													
enuma2e2 list	10		10													
enuma2e2 set-inner	10		10													

TABLE CXII: Suite Peitl ($n = 10$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	10			10													
qc set-inner	10			10													
qc min list	10			10													
qc min set-inner	10			10													
qc minsepcsr list	10			10													
qc minsepcsr set-inner	10			10													

TABLE CXIII: Suite Preusser ($n = 0$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	0													
q set-inner	0													
q min list	0													
q min set-inner	0													
qc list	0													
qc set-inner	0													
qc min list	0													
qc min set-inner	0													
qc minsepcsr list	0													
qc minsepcsr set-inner	0													
enuma2e1 list	0													
enuma2e1 set-inner	0													
enuma2e2 list	0													
enuma2e2 set-inner	0													

TABLE CXIV: Suite Preusser ($n = 0$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	0															
q set-inner	0															
q min list	0															
q min set-inner	0															
qc list	0															
qc set-inner	0															
qc min list	0															
qc min set-inner	0															
qc minsepcsr list	0															
qc minsepcsr set-inner	0															
enuma2e1 list	0															
enuma2e1 set-inner	0															
enuma2e2 list	0															
enuma2e2 set-inner	0															

TABLE CXV: Suite Preusser ($n = 0$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	0																
qc set-inner	0																
qc min list	0																
qc min set-inner	0																
qc minsepcsr list	0																
qc minsepcsr set-inner	0																

TABLE CXVI: Suite qbfeval12 ($n = 8$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	8						6		2					
q set-inner	8						6		2					
q min list	8						3	3	2					
q min set-inner	8						3	3	2					
qc list	7	7												
qc set-inner	7	7												
qc min list	7	7												
qc min set-inner	7	7												
qc minsepcsr list	7	7												
qc minsepcsr set-inner	7	7												
enuma2e1 list	6	6												
enuma2e1 set-inner	6	6												
enuma2e2 list	6	6												
enuma2e2 set-inner	6	6												

TABLE CXVII: Suite qbfeval12 ($n = 8$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	sol- ved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	8													4	4	
q set-inner	8													4	4	
q min list	8														7	1
q min set-inner	8														7	1
qc list	7		7													
qc set-inner	7		7													
qc min list	7		7													
qc min set-inner	7		7													
qc minsepcsr list	7		7													
qc minsepcsr set-inner	7		7													
enuma2e1 list	6		6													
enuma2e1 set-inner	6		6													
enuma2e2 list	6		6													
enuma2e2 set-inner	6		6													

TABLE CXVIII: Suite qbfeval12 ($n = 8$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	sol- ved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	7			7													
qc set-inner	7			7													
qc min list	7			7													
qc min set-inner	7			7													
qc minsepcsr list	7		1	6													
qc minsepcsr set-inner	7		1	6													

AM. Rabe ($n = 3$)

TABLE CXIX: Suite Rabe ($n = 3$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	0													
q set-inner	0													
q min list	0													
q min set-inner	0													
qc list	0													
qc set-inner	0													
qc min list	0													
qc min set-inner	0													
qc minsepcsr list	0													
qc minsepcsr set-inner	0													
enuma2e1 list	0													
enuma2e1 set-inner	0													
enuma2e2 list	0													
enuma2e2 set-inner	0													

TABLE CXX: Suite Rabe ($n = 3$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	0															
q set-inner	0															
q min list	0															
q min set-inner	0															
qc list	0															
qc set-inner	0															
qc min list	0															
qc min set-inner	0															
qc minsepcsr list	0															
qc minsepcsr set-inner	0															
enuma2e1 list	0															
enuma2e1 set-inner	0															
enuma2e2 list	0															
enuma2e2 set-inner	0															

TABLE CXXI: Suite Rabe ($n = 3$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	0																
qc set-inner	0																
qc min list	0																
qc min set-inner	0																
qc minsepcsr list	0																
qc minsepcsr set-inner	0																

TABLE CXXII: Suite Rintanen ($n = 55$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	21	14	7											
q set-inner	21	14	7											
q min list	16	7	6	3										
q min set-inner	19	8	9	2										
qc list	21	21												
qc set-inner	21	21												
qc min list	17	17												
qc min set-inner	16	16												
qc minsepcsr list	13	8	5											
qc minsepcsr set-inner	15	11	4											
enuma2e1 list	16	9	2	3	2									
enuma2e1 set-inner	18	13		3	2									
enuma2e2 list	13	9	1	3										
enuma2e2 set-inner	15	12		3										

TABLE CXXIII: Suite Rintanen ($n = 55$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	sol- ved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	21	2	12								7					
q set-inner	21	2	12								7					
q min list	16	2	5								2	7				
q min set-inner	19	2	6								6	5				
qc list	21	2	19													
qc set-inner	21	2	19													
qc min list	17	2	15													
qc min set-inner	16	2	14													
qc minsepcsr list	13	2	6								1	4				
qc minsepcsr set-inner	15	2	9								1	3				
enuma2e1 list	16	2	7								1	1	1	3	1	
enuma2e1 set-inner	18	2	11										4		1	
enuma2e2 list	13	2	7									1		3		
enuma2e2 set-inner	15	2	10										3			

TABLE CXXIV: Suite Rintanen ($n = 55$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	sol- ved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	21	2		19													
qc set-inner	21	2		19													
qc min list	17	2		15													
qc min set-inner	16	2		14													
qc minsepcsr list	13	2		6								1	4				
qc minsepcsr set-inner	15	2		9								1	3				

TABLE CXXV: Suite Sauer-Reimer ($n = 42$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8 –	16 –	32 –	64 –	128 –	256 –	512 –	1024 –	2048 –
						15	31	63	127	255	511	1023	2047	4096
q list	40	1	1	1	2	1	1	8	10	7	8			
q set-inner	41	1	1	1		3	1	8	9	10	7			
q min list	38	1		2		3	1	4	13	7	7			
q min set-inner	38	1		2		3	1	4	13	7	7			
qc list	40	40												
qc set-inner	41	41												
qc min list	39	38	1											
qc min set-inner	41	41												
qc minsepcsr list	37	27	3	7										
qc minsepcsr set-inner	36	36												
enuma2e1 list	38	37			1									
enuma2e1 set-inner	39	39												
enuma2e2 list	9	7	1		1									
enuma2e2 set-inner	9	9												

TABLE CXXVI: Suite Sauer-Reimer ($n = 42$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	40		1						1			3	5	8	22	
q set-inner	41		1						1			3	4	11	21	
q min list	38		1								1		6	8	22	
q min set-inner	38		1								1		6	8	22	
qc list	40		40													
qc set-inner	41		41													
qc min list	39		38	1												
qc min set-inner	41		41													
qc minsepcsr list	37		27		1	2	6		1							
qc minsepcsr set-inner	36		36													
enuma2e1 list	38		37									1				
enuma2e1 set-inner	39		39													
enuma2e2 list	9		7						1			1				
enuma2e2 set-inner	9		9													

TABLE CXXVII: Suite Sauer-Reimer ($n = 42$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	40		40														
qc set-inner	41		41														
qc min list	39		38						1								
qc min set-inner	41		41														
qc minsepcsr list	37		27						2	2	6						
qc minsepcsr set-inner	36		36														

TABLE CXXVIII: Suite Scholl-Becker ($n = 30$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	18	17	1											
q set-inner	18	17	1											
q min list	18	17	1											
q min set-inner	18	17	1											
qc list	17	17												
qc set-inner	17	17												
qc min list	13	13												
qc min set-inner	13	13												
qc minsepcsr list	13	13												
qc minsepcsr set-inner	13	13												
enuma2e1 list	17	17												
enuma2e1 set-inner	17	17												
enuma2e2 list	16	16												
enuma2e2 set-inner	16	16												

TABLE CXXIX: Suite Scholl-Becker ($n = 30$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	18		17								1					
q set-inner	18		17								1					
q min list	18		17								1					
q min set-inner	18		17								1					
qc list	17		17													
qc set-inner	17		17													
qc min list	13		13													
qc min set-inner	13		13													
qc minsepcsr list	13		13													
qc minsepcsr set-inner	13		13													
enuma2e1 list	17		17													
enuma2e1 set-inner	17		17													
enuma2e2 list	16		16													
enuma2e2 set-inner	16		16													

TABLE CXXX: Suite Scholl-Becker ($n = 30$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	17			17													
qc set-inner	17			17													
qc min list	13			13													
qc min set-inner	13			13													
qc minsepcsr list	13			13													
qc minsepcsr set-inner	13			13													

TABLE CXXXI: Suite Seidl ($n = 194$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	177								177					
q set-inner	50								50					
q min list	113								113					
q min set-inner	19								19					
qc list	131	131												
qc set-inner	1	1												
qc min list	130	95	35											
qc min set-inner	1	1												
qc minsepcsr list	19			1	18									
qc minsepcsr set-inner	0													
enuma2e1 list	48	48												
enuma2e1 set-inner	0													
enuma2e2 list	0													
enuma2e2 set-inner	0													

TABLE CXXXII: Suite Seidl ($n = 194$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	177															177
q set-inner	50															50
q min list	113															113
q min set-inner	19															19
qc list	131		131													
qc set-inner	1		1													
qc min list	130		95			35										
qc min set-inner	1		1													
qc minsepcsr list	19						1	18								
qc minsepcsr set-inner	0															
enuma2e1 list	48		48													
enuma2e1 set-inner	0															
enuma2e2 list	0															
enuma2e2 set-inner	0															

TABLE CXXXIII: Suite Seidl ($n = 194$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	131			131													
qc set-inner	1			1													
qc min list	130			95				35									
qc min set-inner	1			1													
qc minsepcsr list	19								1	10	8						
qc minsepcsr set-inner	0																

TABLE CXXXIV: Suite Tacchella ($n = 122$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8 –	16 –	32 –	64 –	128 –	256 –	512 –	1024 –	2048 –
						15	31	63	127	255	511	1023	2047	4096
q list	103	17	3	3	3	6	12	13	26	18	2			
q set-inner	105	17	3	1	3	9	14	11	26	19	2			
q min list	33				2	1	4	7	8	9	2			
q min set-inner	22				2	2	1	2	8	6	1			
qc list	102	101	1											
qc set-inner	105	105												
qc min list	102	96	5		1									
qc min set-inner	105	98	7											
qc minsepcsr list	31	26		1	2	1	1							
qc minsepcsr set-inner	20	17			2	1								
enuma2e1 list	94	92	1	1										
enuma2e1 set-inner	101	99	2											
enuma2e2 list	84	28	18	15	8	11	4							
enuma2e2 set-inner	90	27	9	14	23	13	4							

TABLE CXXXV: Suite Tacchella ($n = 122$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	103		17			2	1	1	4	1	8	11	10	38	10	
q set-inner	105		17			2		2	2	3	10	10	10	39	10	
q min list	33										1	1	6	15	10	
q min set-inner	22										1	1	1	14	5	
qc list	102		101				1									
qc set-inner	105		105													
qc min list	102		96				2	4								
qc min set-inner	105		98			2	2	2	1							
qc minsepcsr list	31		26						1		1	1	1	1		
qc minsepcsr set-inner	20		17								1	1		1		
enuma2e1 list	94		92		1				1							
enuma2e1 set-inner	101		99		1			1								
enuma2e2 list	84		28	1	2	8	9	11	6	5	7	2	4	1		
enuma2e2 set-inner	90		27	2		5	8	18	6	5	7	9	1	2		

TABLE CXXXVI: Suite Tacchella ($n = 122$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	102		101					1									
qc set-inner	105		105														
qc min list	102		96					2	4								
qc min set-inner	105		98				2	2	2	1							
qc minsepcsr list	31		26									2	1	1	1		
qc minsepcsr set-inner	20		17									1	1		1		

TABLE CXXXVII: Suite Tentrup ($n = 17$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						15	31	63	127	255	511	1023	2047	4096
q list	7	3				1	1		1	1				
q set-inner	7	3				1	1		1	1				
q min list	7	2				2	1		1	1				
q min set-inner	7	3				1	1		1	1				
qc list	7	7												
qc set-inner	7	7												
qc min list	6	6												
qc min set-inner	6	6												
qc minsepcsr list	6	5				1								
qc minsepcsr set-inner	6	6												
enuma2e1 list	5	4				1								
enuma2e1 set-inner	7	7												
enuma2e2 list	3	2				1								
enuma2e2 set-inner	5	5												

TABLE CXXXVIII: Suite Tentrup ($n = 17$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	sol- ved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	7		3										1		3	
q set-inner	7		3										1		3	
q min list	7		2										1		4	
q min set-inner	7		3										1		3	
qc list	7		7													
qc set-inner	7		7													
qc min list	6		6													
qc min set-inner	6		6													
qc minsepcsr list	6		5													1
qc minsepcsr set-inner	6		6													
enuma2e1 list	5		4													1
enuma2e1 set-inner	7		7													
enuma2e2 list	3		2													1
enuma2e2 set-inner	5		5													

TABLE CXXXIX: Suite Tentrup ($n = 17$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	sol- ved	no \forall in input	no (weak- ened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	7		7														
qc set-inner	7		7														
qc min list	6		6														
qc min set-inner	6		6														
qc minsepcsr list	6		5														1
qc minsepcsr set-inner	6		6														

TABLE CXL: Suite Wintersteiger ($n = 38$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall is in a range.

	solved	0	1	2-3	4-7	8	16	32	64	128	256	512	1024	2048
						–	–	–	–	–	–	–	–	–
						15	31	63	127	255	511	1023	2047	4096
q list	21	17					1	2	1					
q set-inner	6	6												
q min list	15	13						1	1					
q min set-inner	3	3												
qc list	20	20												
qc set-inner	6	6												
qc min list	18	18												
qc min set-inner	6	6												
qc minsepcsr list	14	12			1	1								
qc minsepcsr set-inner	2	2												
enuma2e1 list	13	12	1											
enuma2e1 set-inner	2	2												
enuma2e2 list	13	12		1										
enuma2e2 set-inner	2	2												

TABLE CXLI: Suite Wintersteiger ($n = 38$): Extracting unsatisfiable cores and approximating non-trivially \forall -to- \exists reducible quantifications: number of instances whose number of weakened (resp. non-trivially \forall -to- \exists reducible) \forall divided by the number of \forall in the original formula is in a range.

	solved	no \forall in input	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
q list	21	17						1				3				
q set-inner	6	6														
q min list	15	13										2				
q min set-inner	3	3														
qc list	20	20														
qc set-inner	6	6														
qc min list	18	18														
qc min set-inner	6	6														
qc minsepcsr list	14	12								2						
qc minsepcsr set-inner	2	2														
enuma2e1 list	13	12					1									
enuma2e1 set-inner	2	2														
enuma2e2 list	13	12							1							
enuma2e2 set-inner	2	2														

TABLE CXLII: Suite Wintersteiger ($n = 38$): Extracting unsatisfiable cores: number of instances whose number of weakened \forall divided by the number of \forall in the unsatisfiable core plus the number of weakened \forall is in a range. The values for q-cores are as in the previous table except for universally quantified variables that do not appear in the matrix of the original formula and, therefore, not shown.

	solved	no \forall in input	no (weakened) \forall in UC	0	[0.002, 0.004[[0.004, 0.006[[0.006, 0.008[[0.008, 0.02[[0.02, 0.04[[0.04, 0.06[[0.06, 0.08[[0.08, 0.2[[0.2, 0.4[[0.4, 0.6[[0.6, 0.8[[0.8, 1[1
qc list	20		20														
qc set-inner	6		6														
qc min list	18		18														
qc min set-inner	6		6														
qc minsepcsr list	14		12								1	1					
qc minsepcsr set-inner	2		2														

APPENDIX D

ADDITIONAL PLOTS — COMPARISON OF SIZES OF UNSATISFIABLE CORES

In this appendix we show plots of the sizes of unsatisfiable cores that partition the set of benchmark instances by

- benchmark suite,
- number of \forall quantified variables,
- alternation depth,
- number of clauses,
- maximum variable index, and
- number of weakened \forall in mode qc minsepcsr list.

When we provide a number of benchmarks for a plot or a number of plots ($n = \dots$), then in this appendix (except for the last subsection) this is the number of all benchmarks relevant to the plot that have been found to be unsatisfiable by any solver in our experiments.

A. *Partitioned by Benchmark Suite*

In this subsection there are 7 figures for each benchmark suite with subfigures for different modes of unsatisfiable core extraction:

- 1) Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min.
- 2) As 1. but with set-inner semantics.
- 3) Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula.
- 4) As 3. but with set-inner semantics.
- 5) Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics.
- 6) As 5. but with set-inner semantics.
- 7) Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics.

1) All ($n = 2528$):

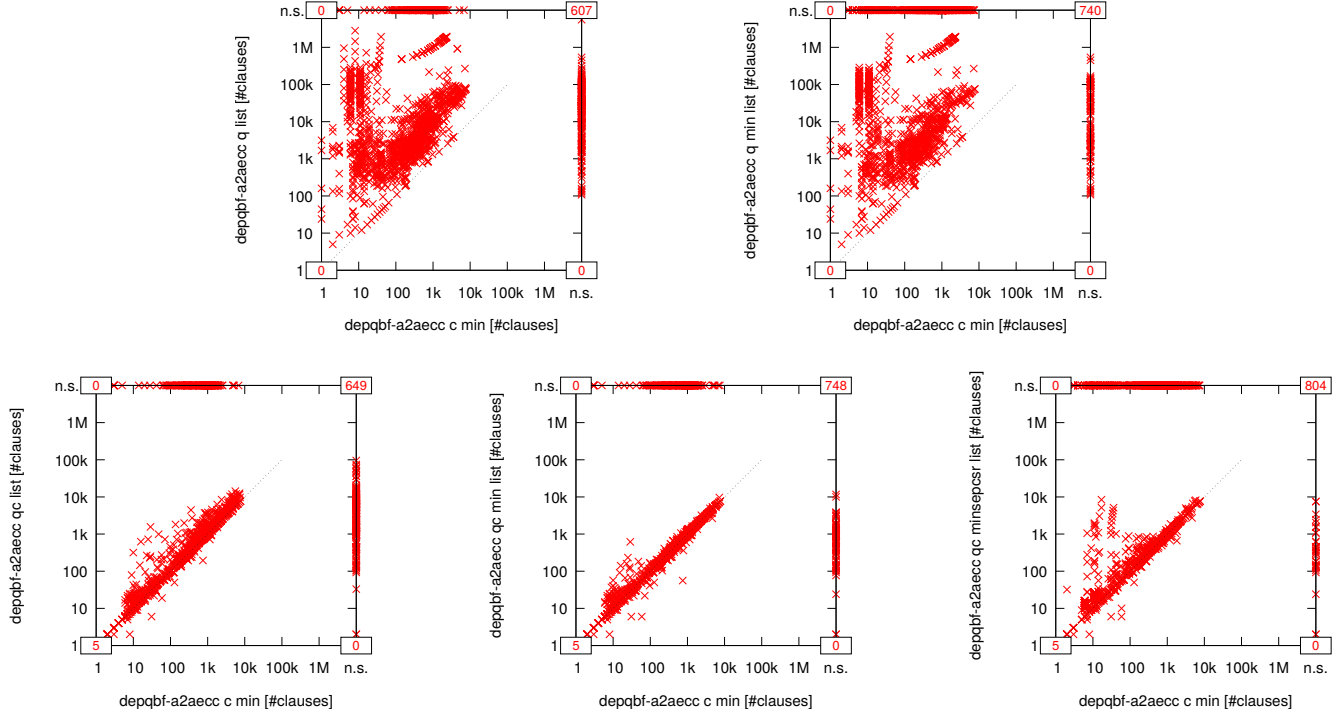


Fig. 11: Suite All ($n = 2528$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

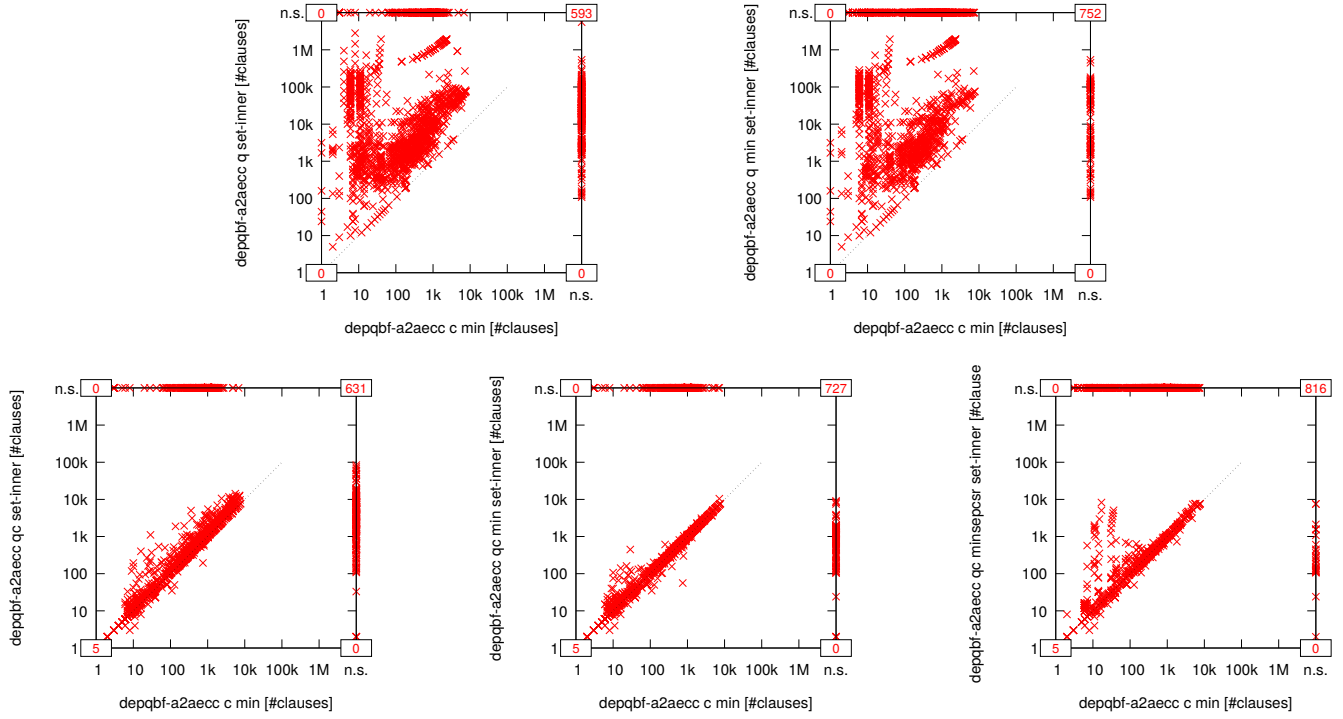


Fig. 12: Suite All ($n = 2528$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

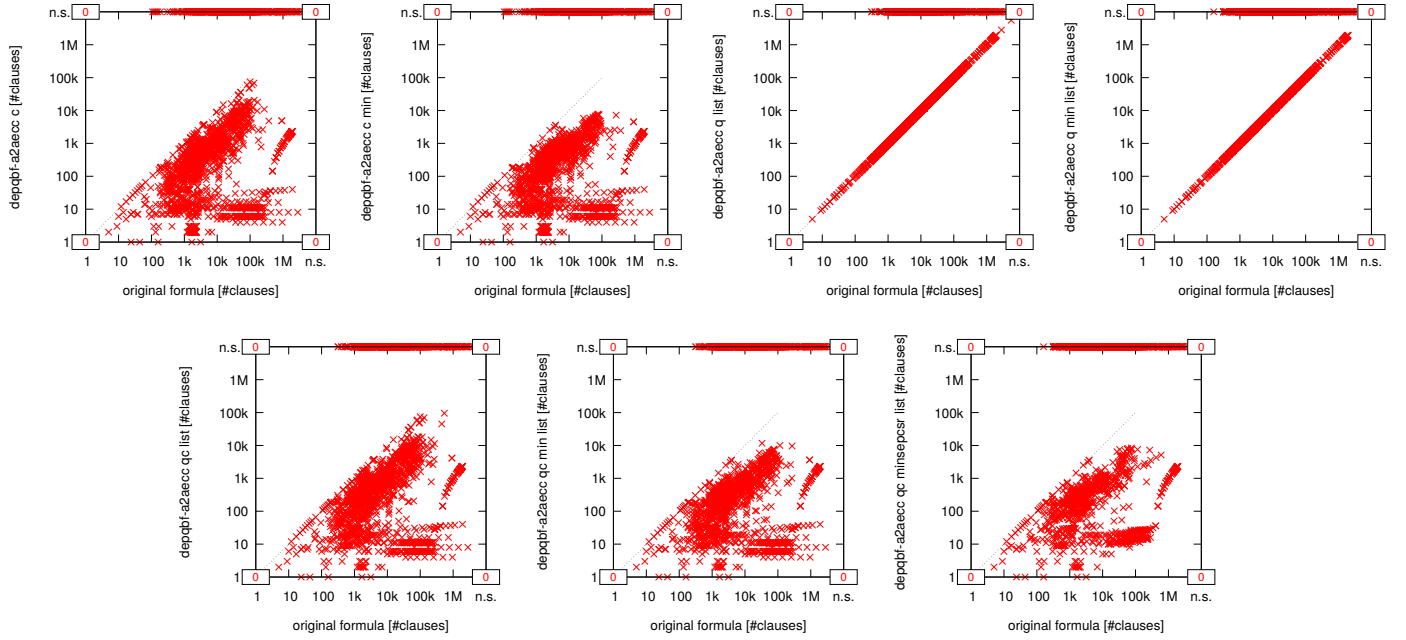


Fig. 13: Suite All ($n = 2528$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

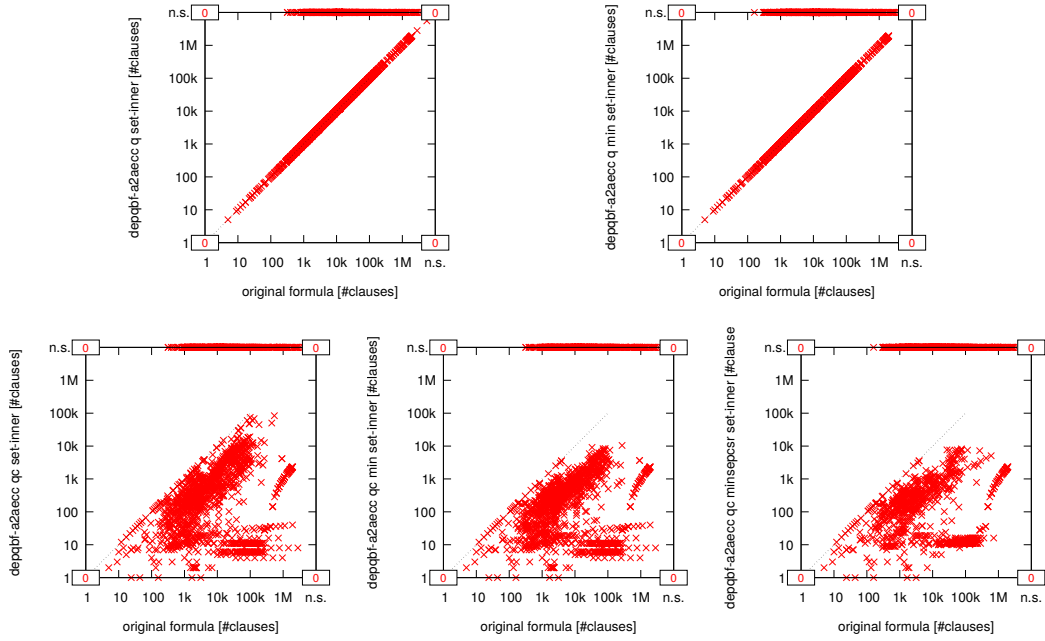


Fig. 14: Suite All ($n = 2528$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

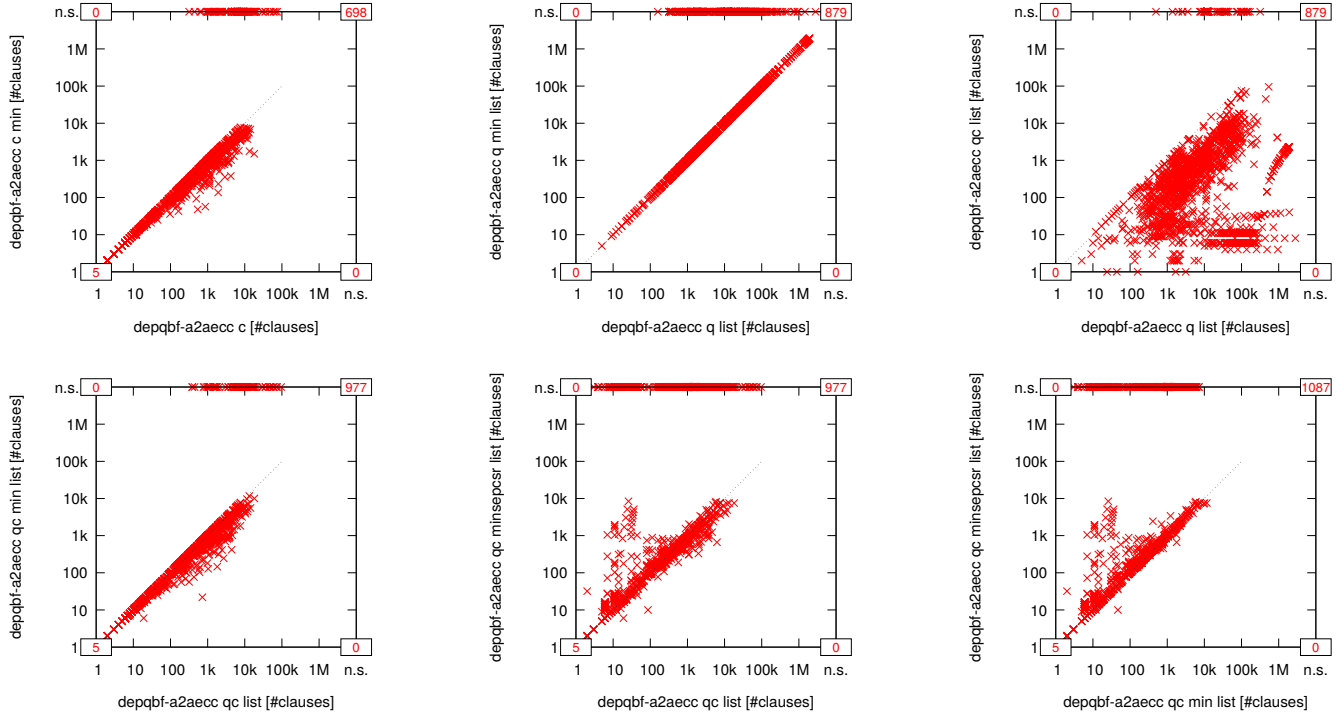


Fig. 15: Suite All ($n = 2528$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

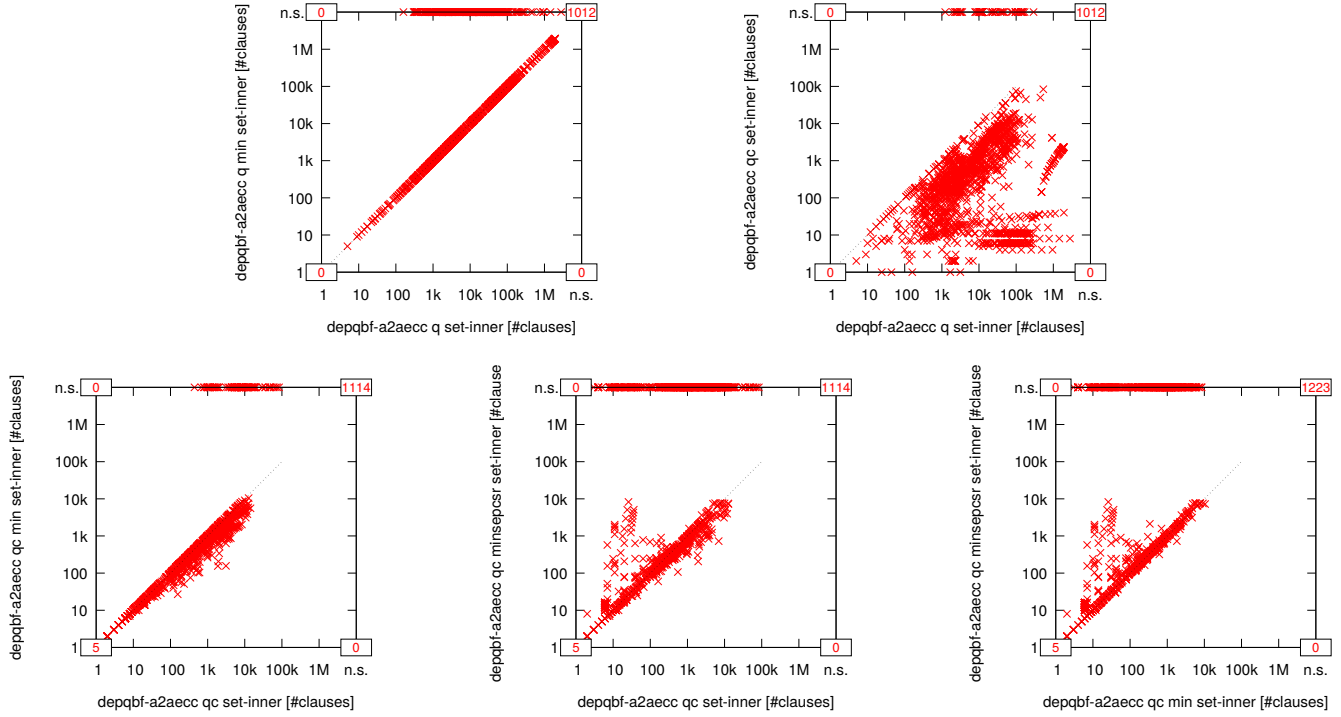


Fig. 16: Suite All ($n = 2528$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

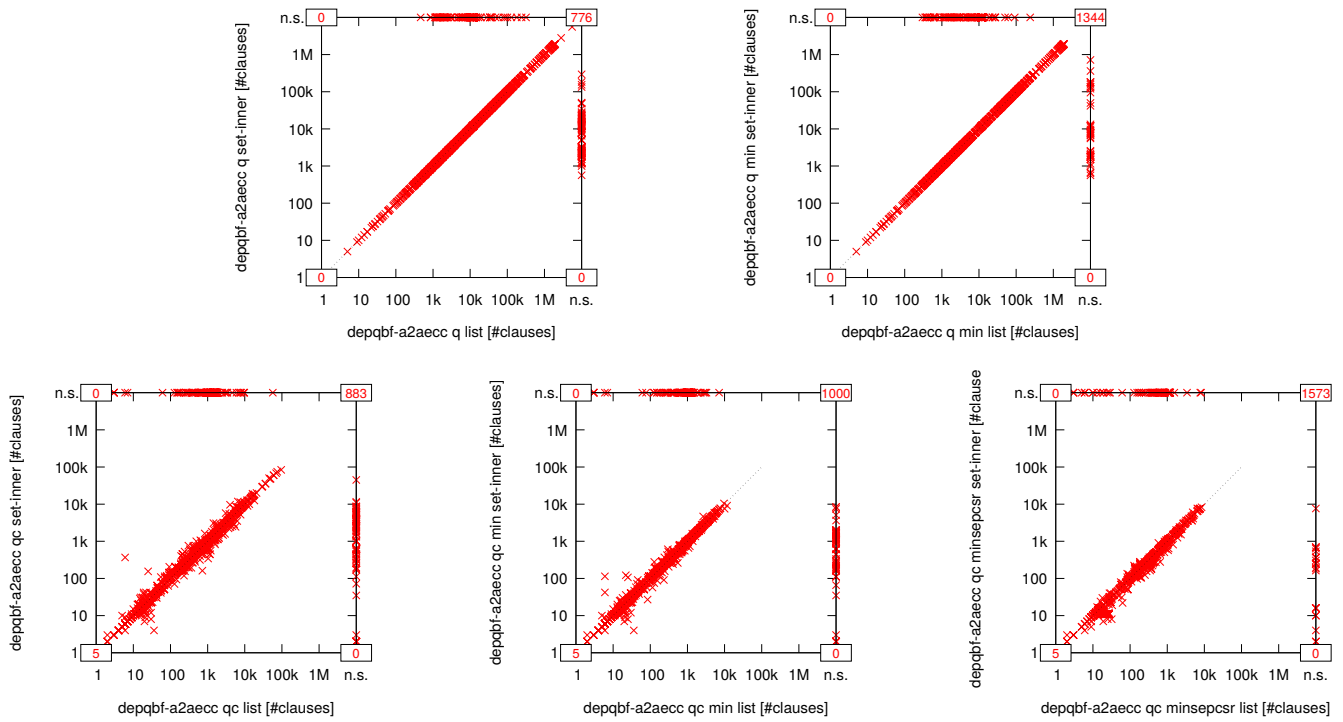


Fig. 17: Suite All ($n = 2528$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

2) Akshay-Chakraborty-John-Shah-Rabe ($n = 2$):

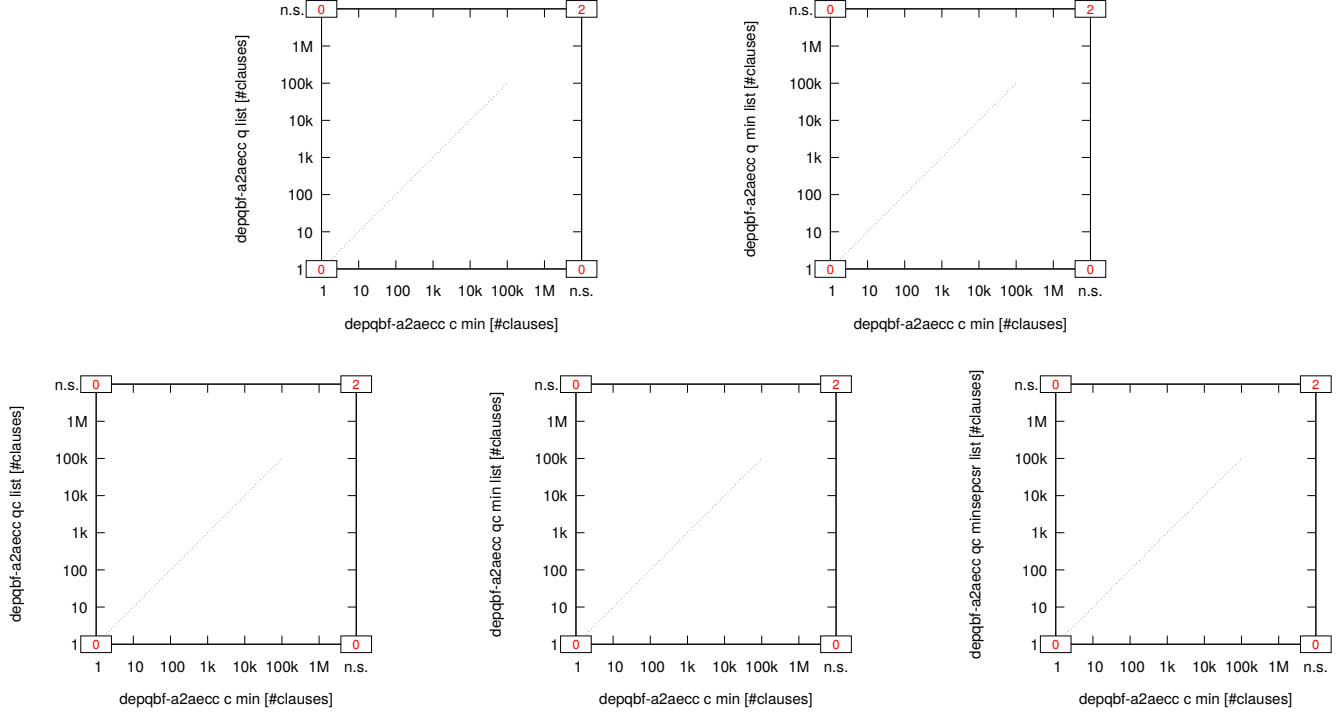


Fig. 18: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

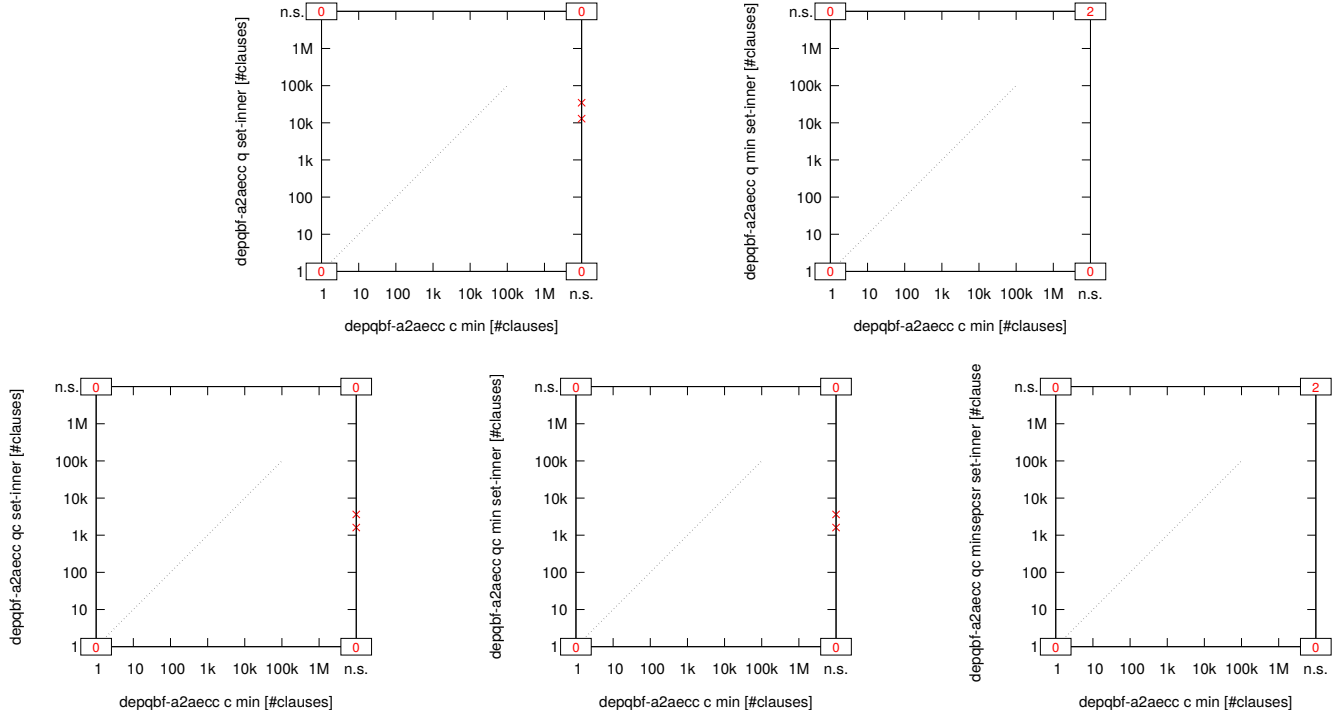


Fig. 19: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

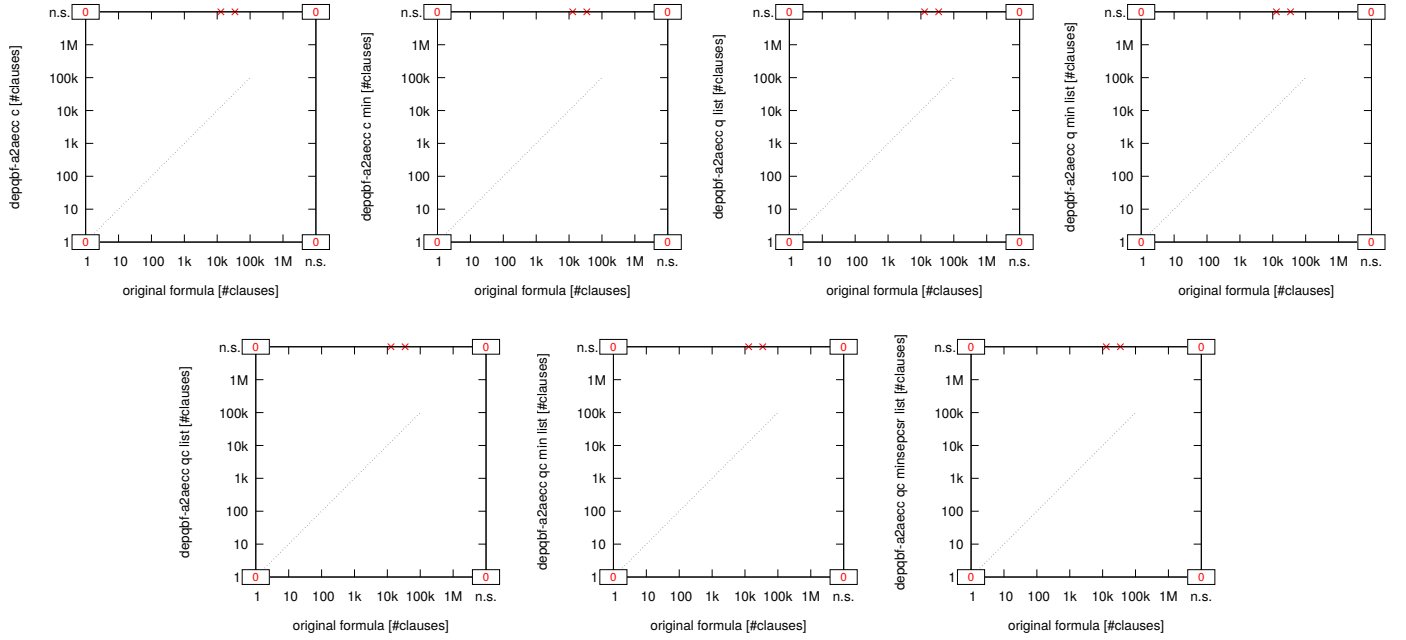


Fig. 20: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

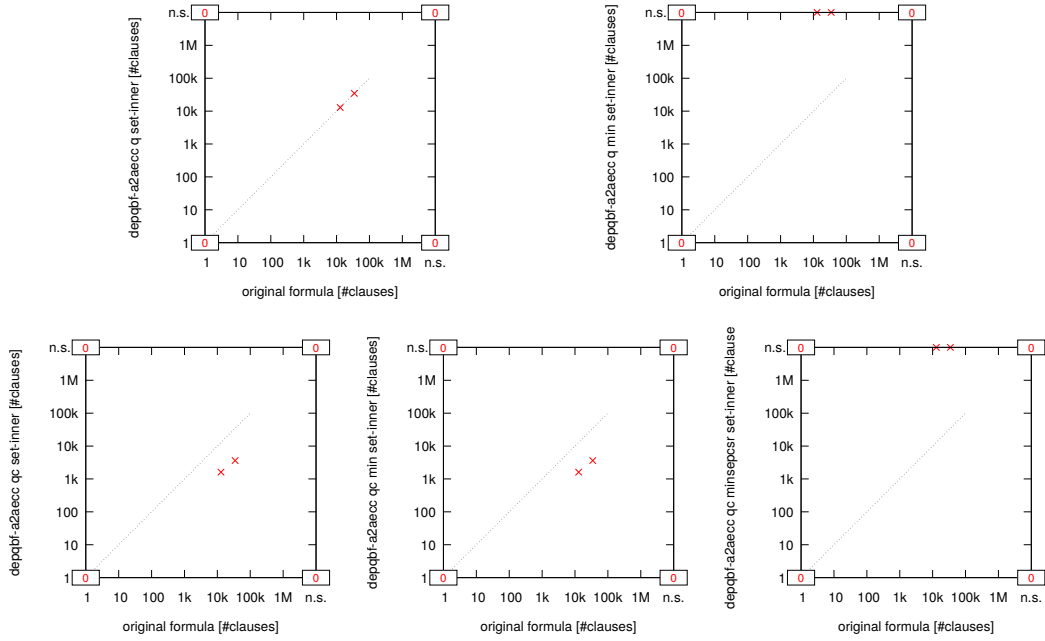


Fig. 21: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

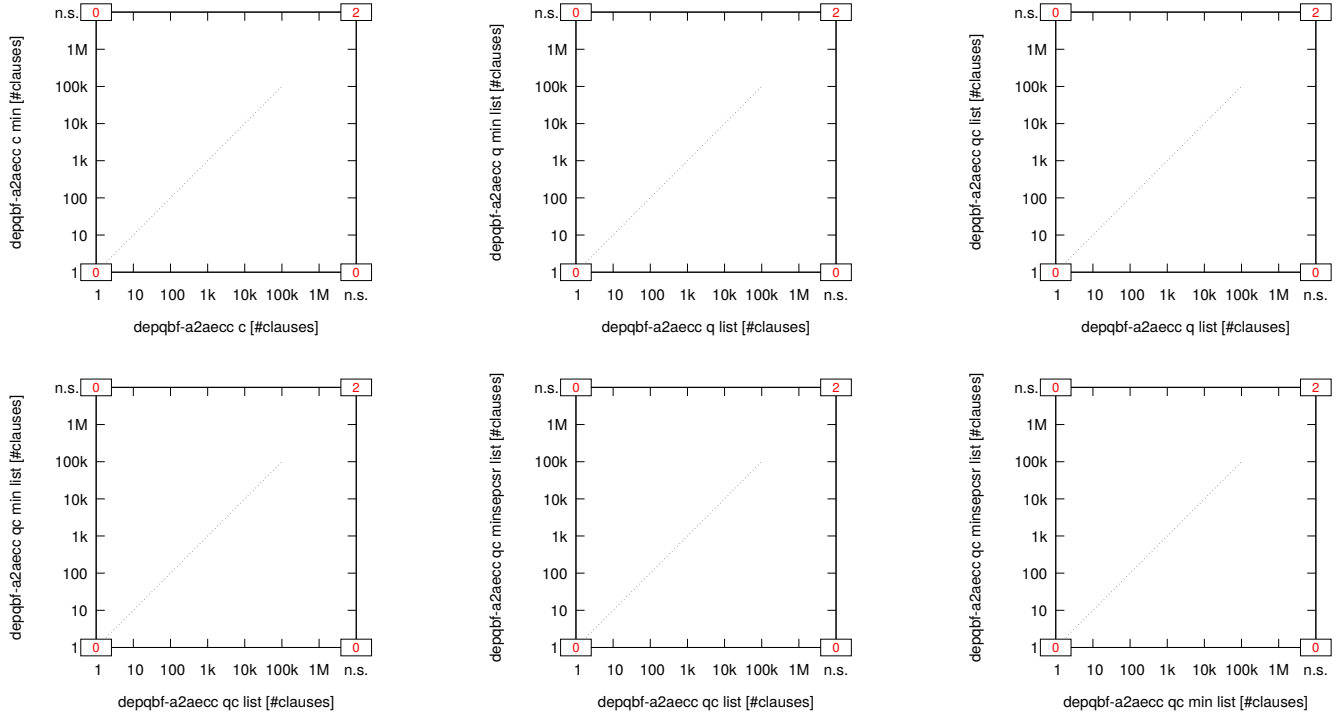


Fig. 22: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

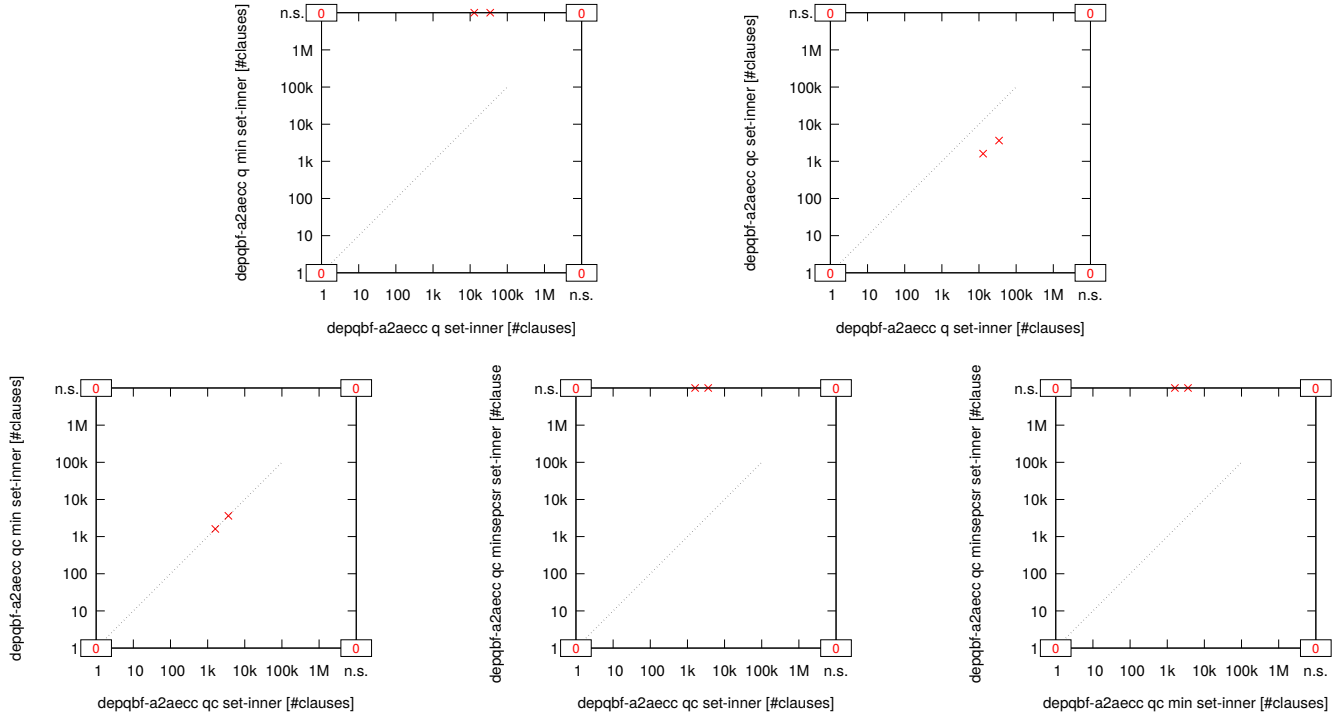


Fig. 23: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

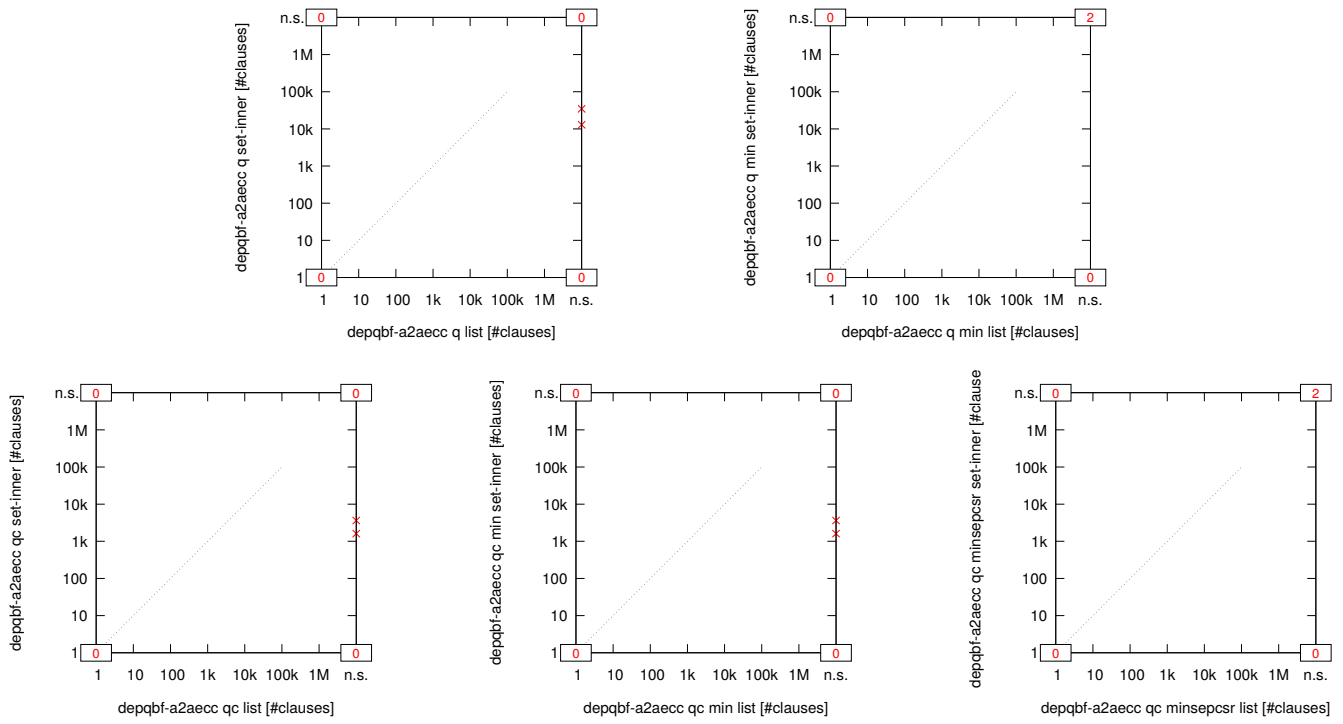


Fig. 24: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

3) Amendola-Ricca-Truszczyński ($n = 9$):

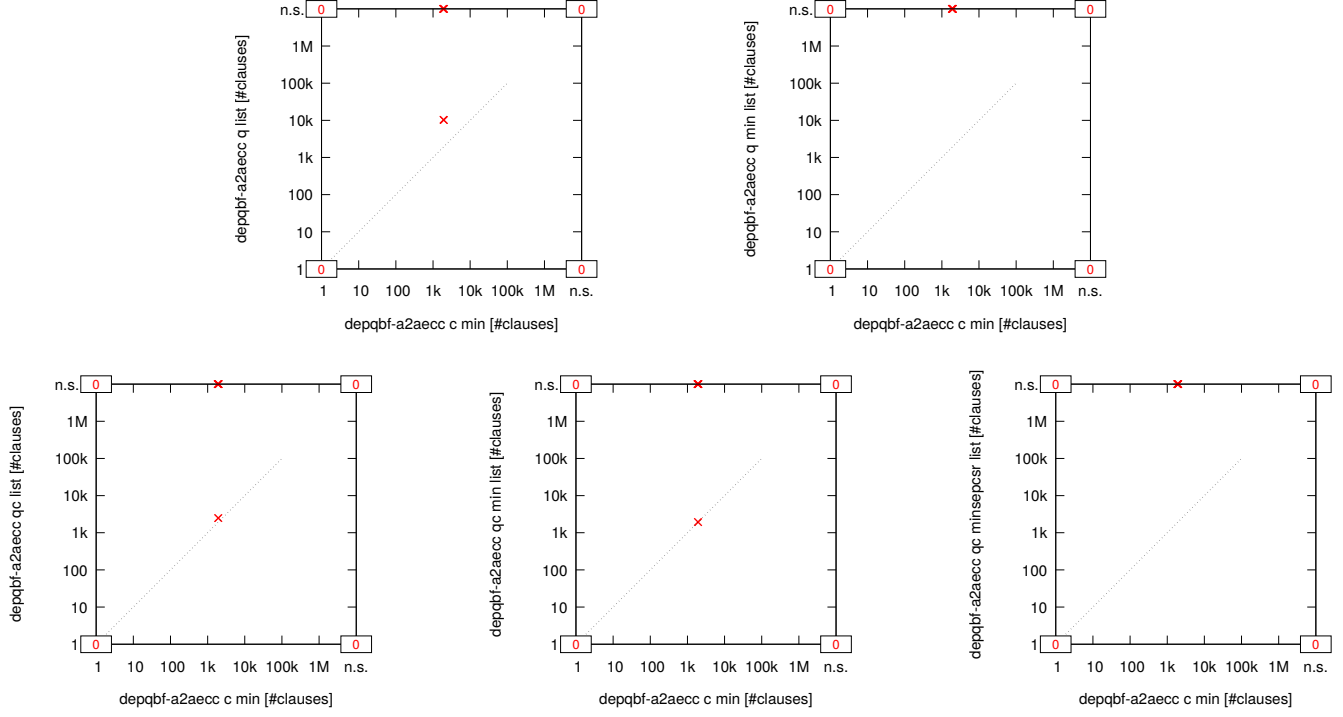


Fig. 25: Suite Amendola-Ricca-Truszczyński ($n = 9$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

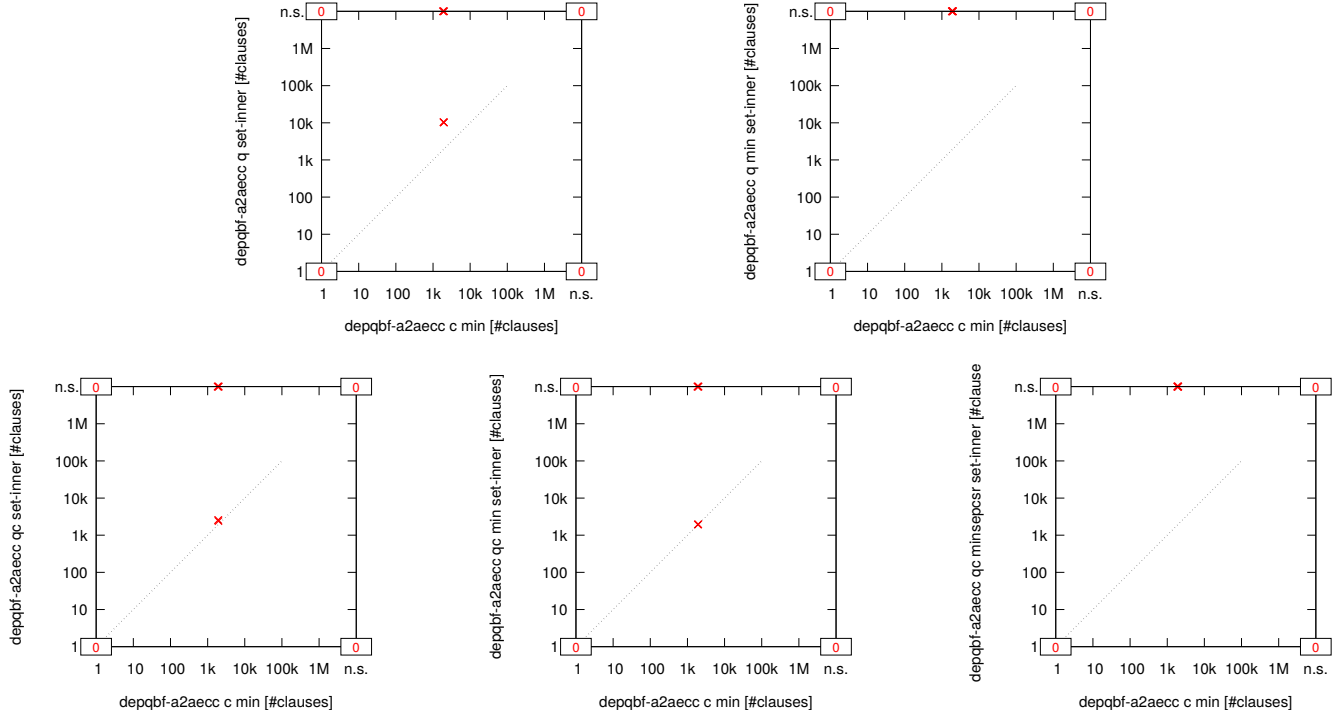


Fig. 26: Suite Amendola-Ricca-Truszczyński ($n = 9$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

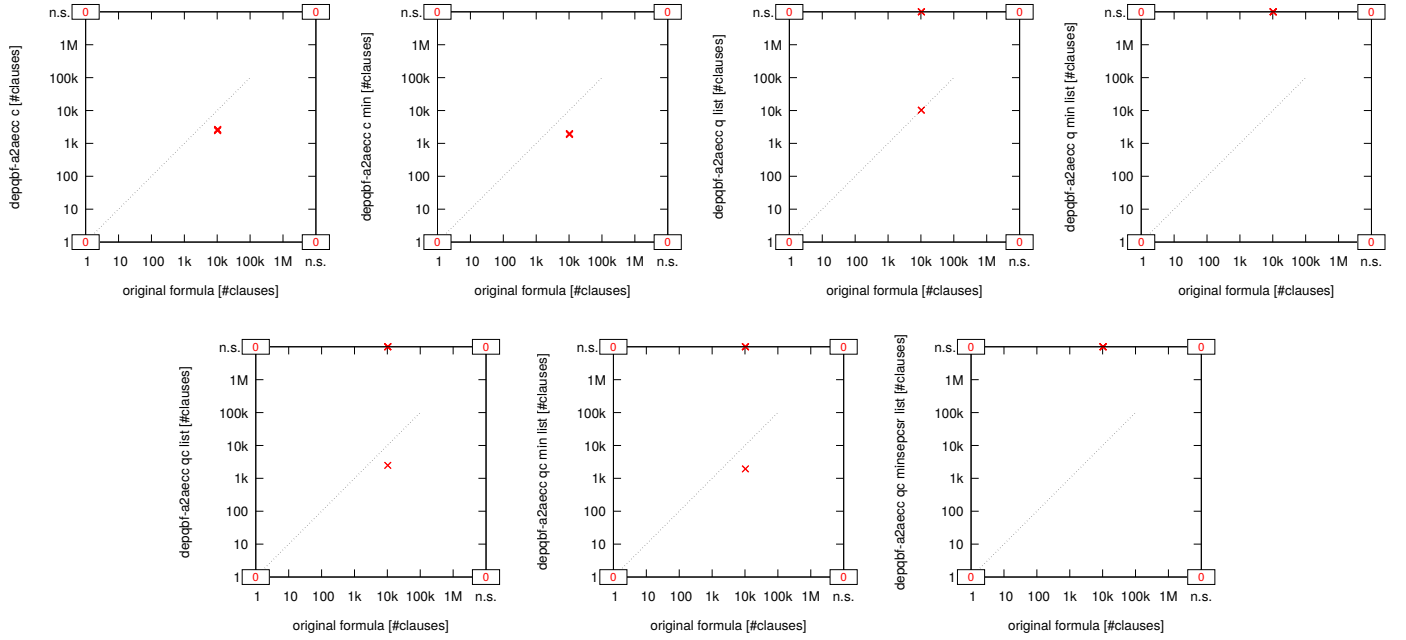


Fig. 27: Suite Amendola-Ricca-Truszczyński ($n = 9$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

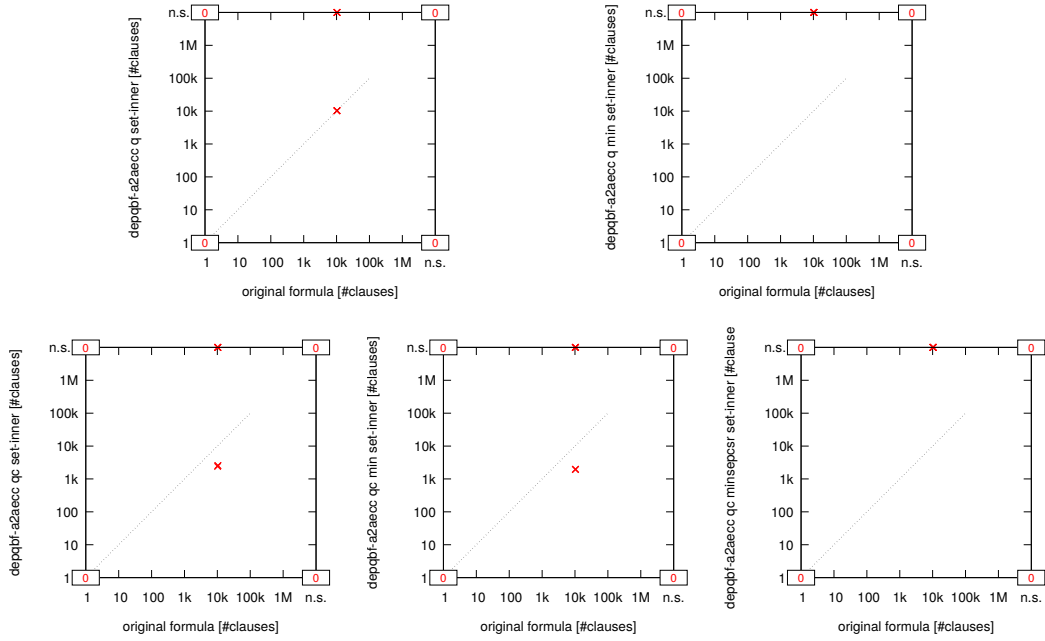


Fig. 28: Suite Amendola-Ricca-Truszczyński ($n = 9$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

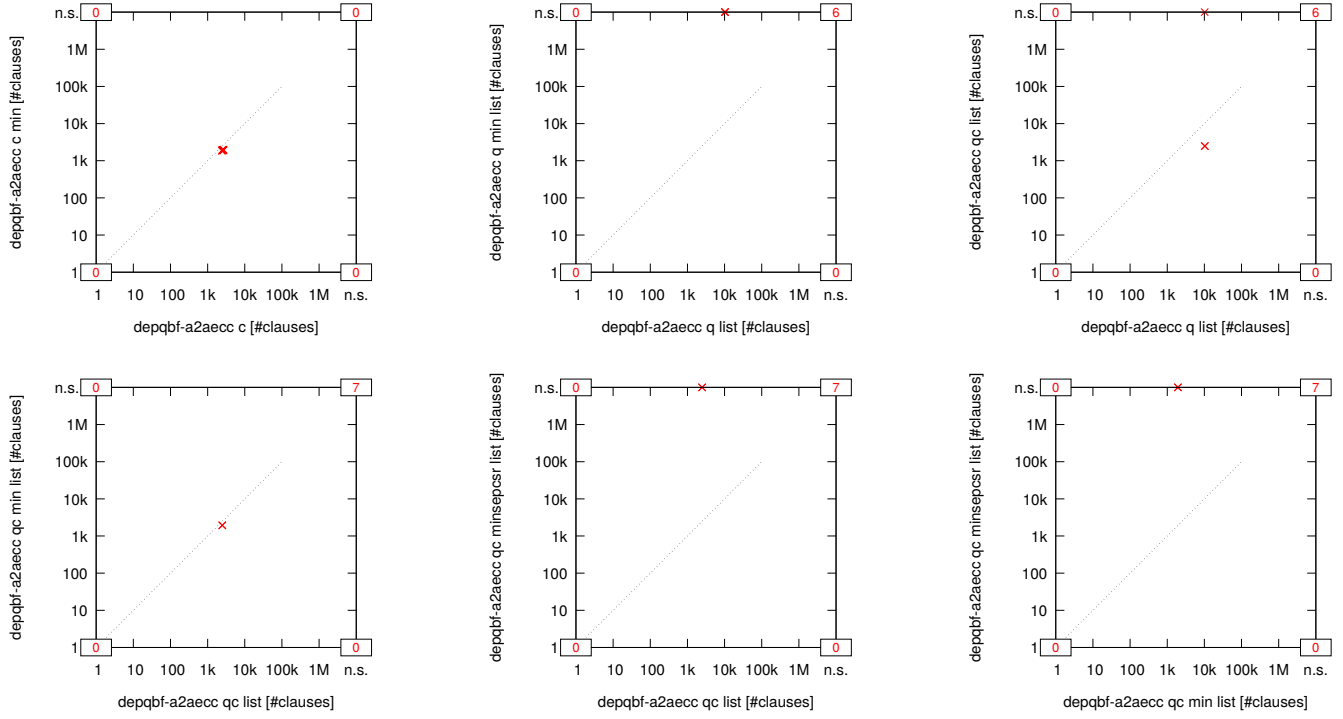


Fig. 29: Suite Amendola-Ricca-Truszczyński ($n = 9$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

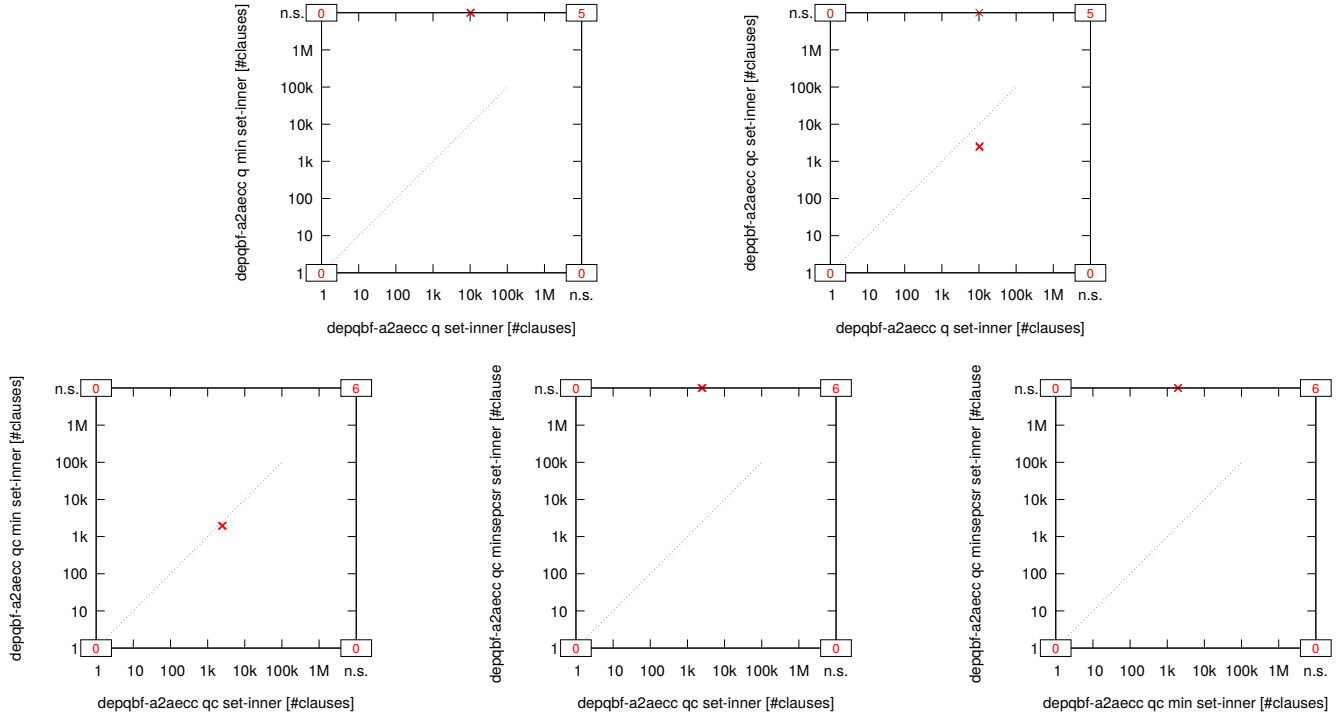


Fig. 30: Suite Amendola-Ricca-Truszczyński ($n = 9$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

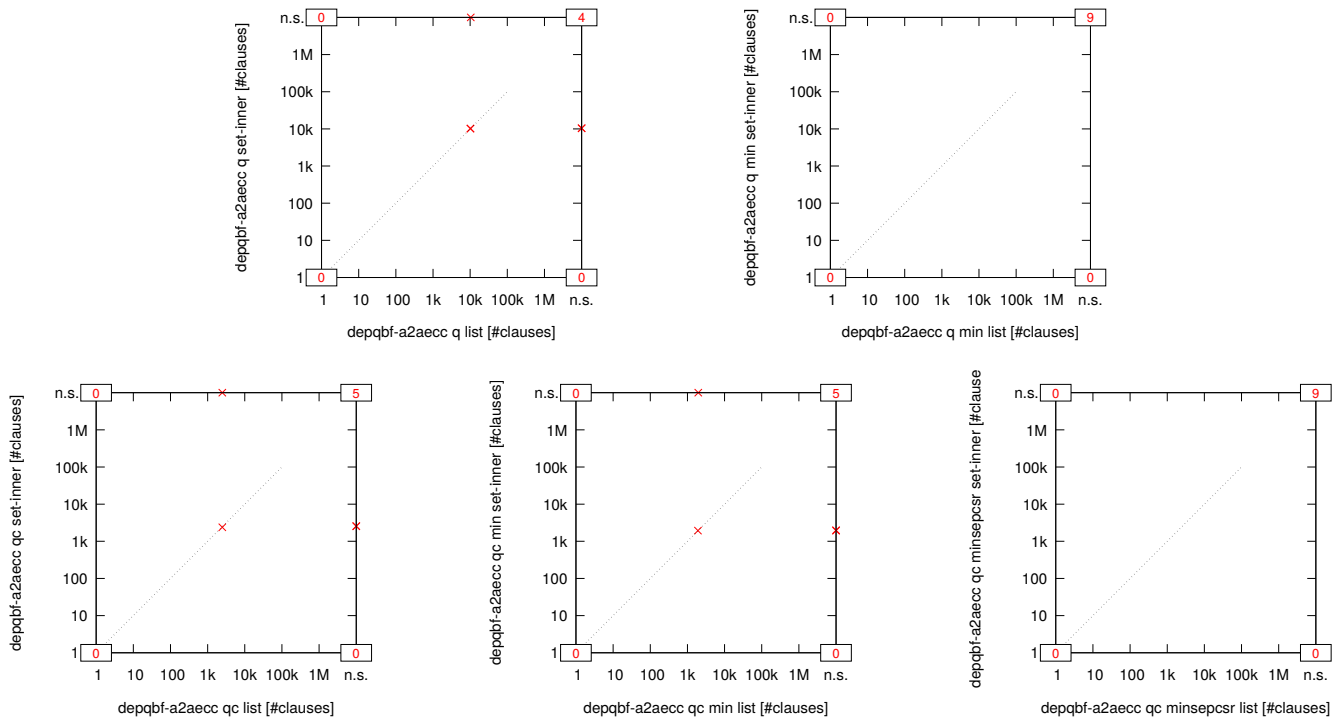


Fig. 31: Suite Amendola-Ricca-Truszczyński ($n = 9$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

4) Ansotegui ($n = 12$):

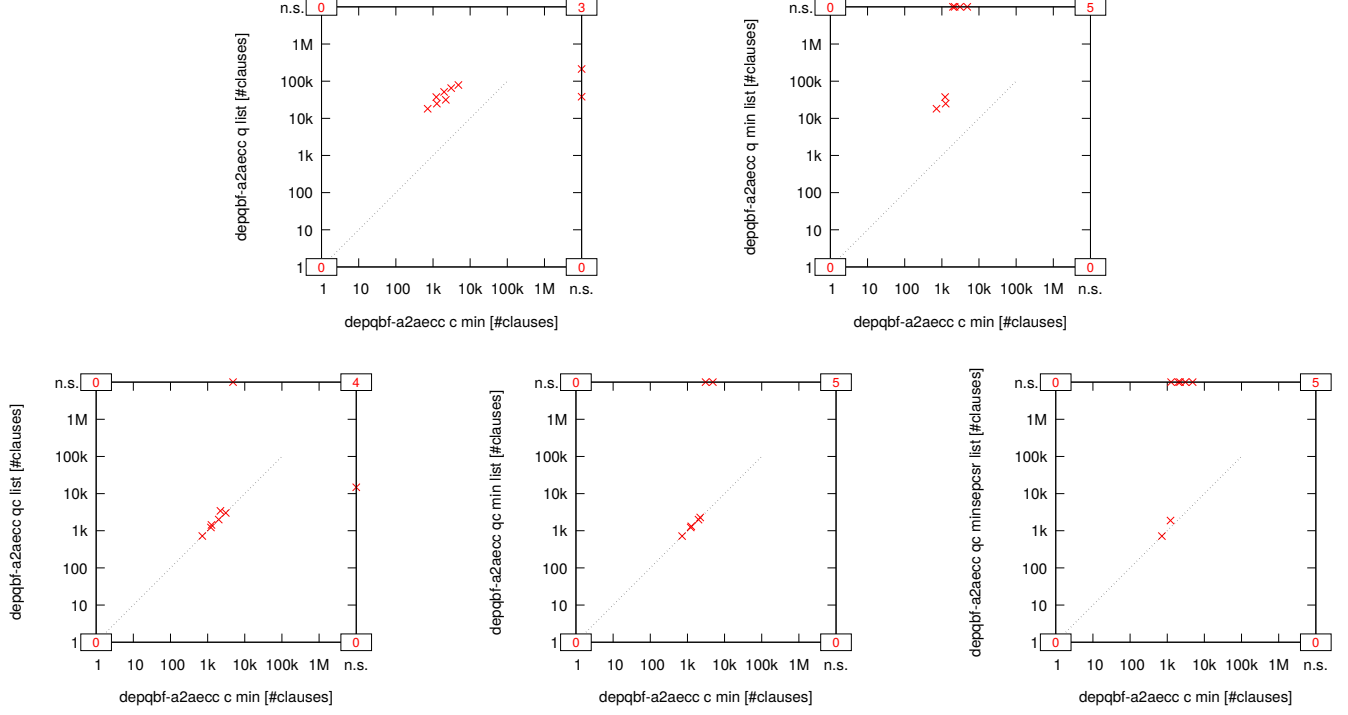


Fig. 32: Suite Ansotegui ($n = 12$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

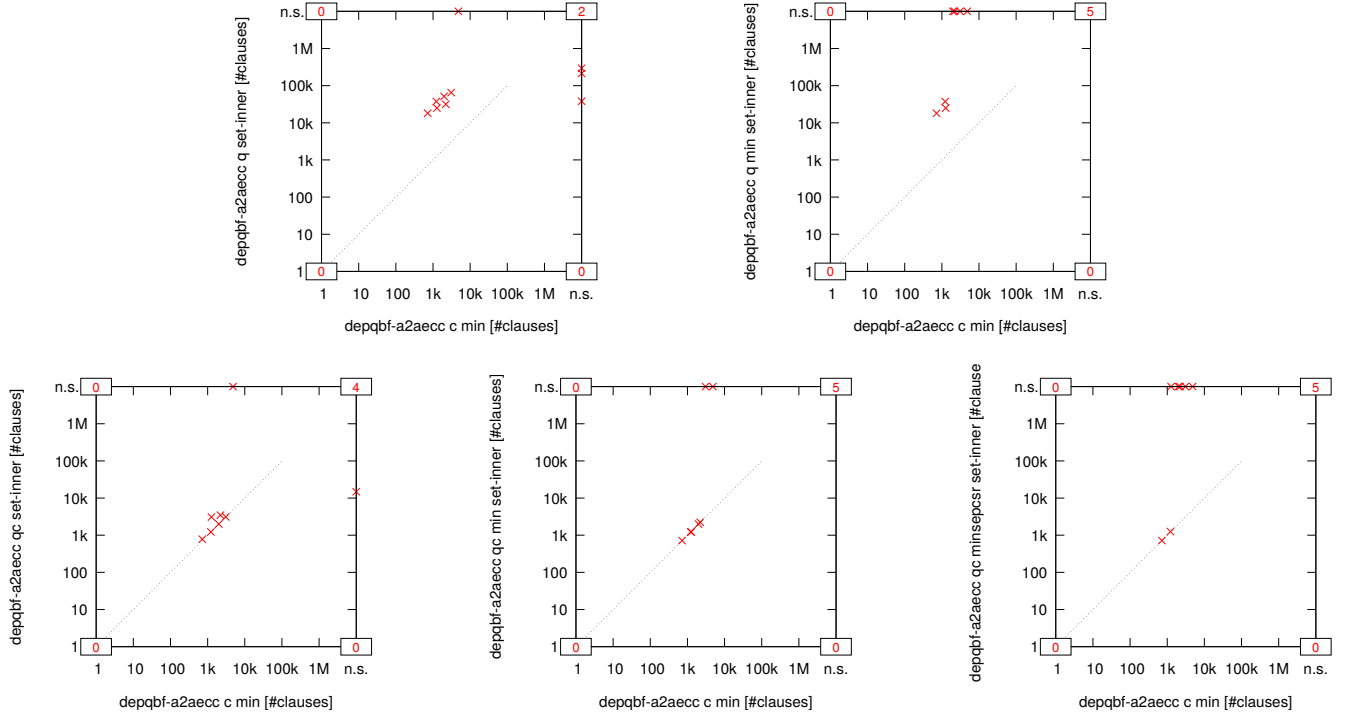


Fig. 33: Suite Ansotegui ($n = 12$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

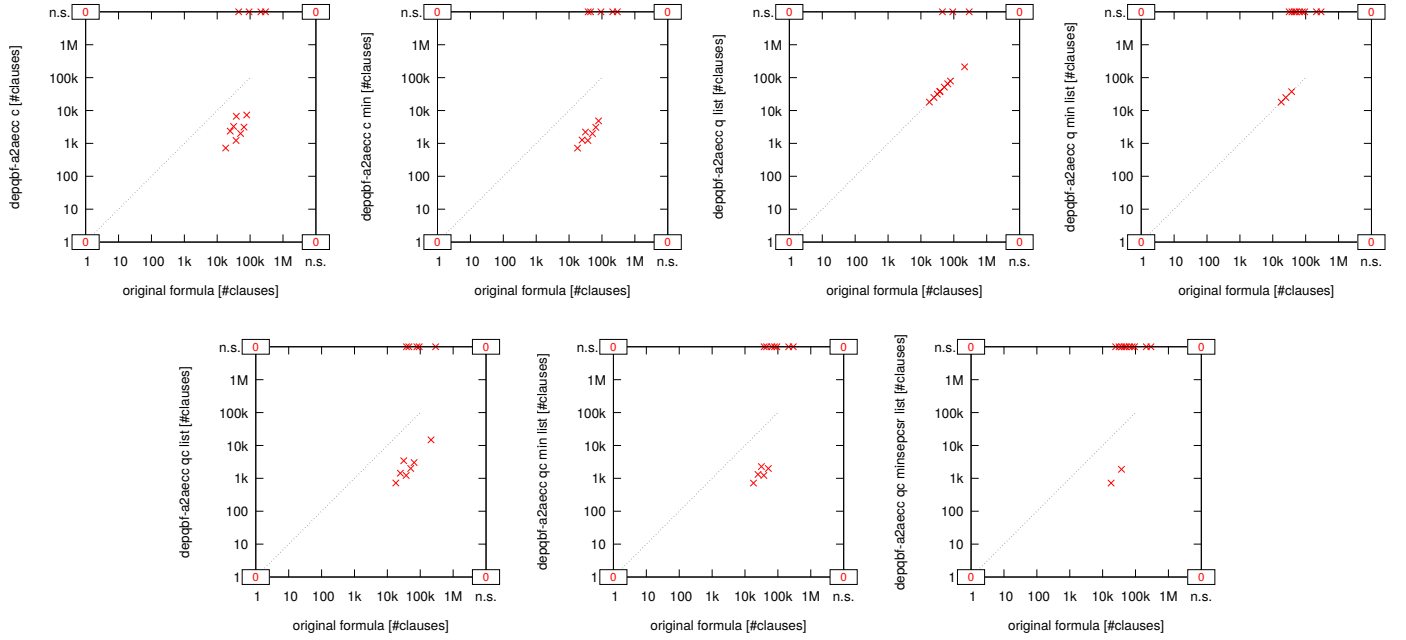


Fig. 34: Suite Ansotegui ($n = 12$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

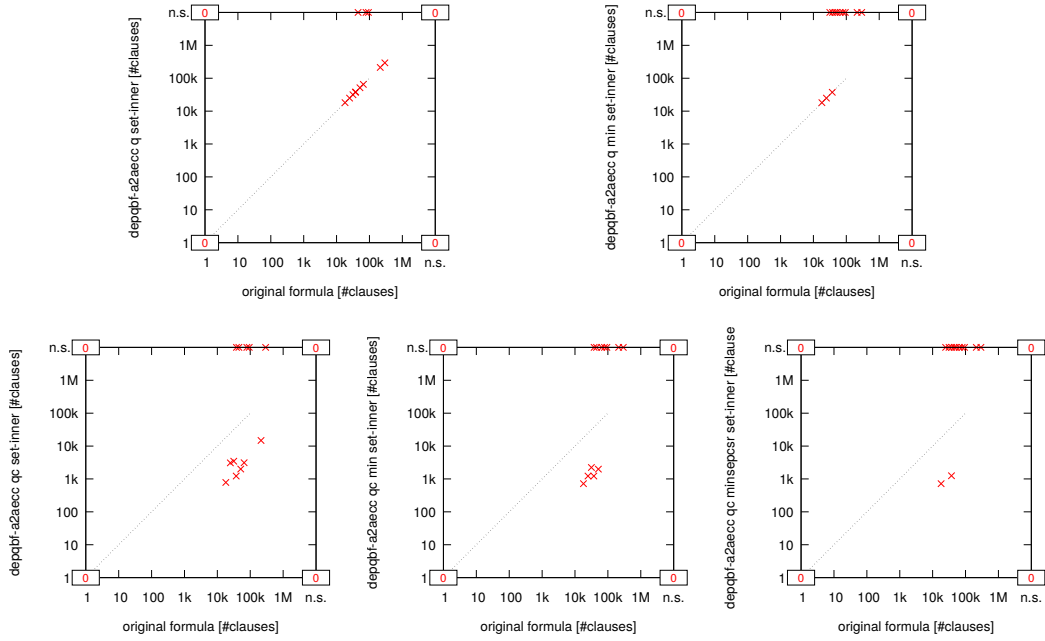


Fig. 35: Suite Ansotegui ($n = 12$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

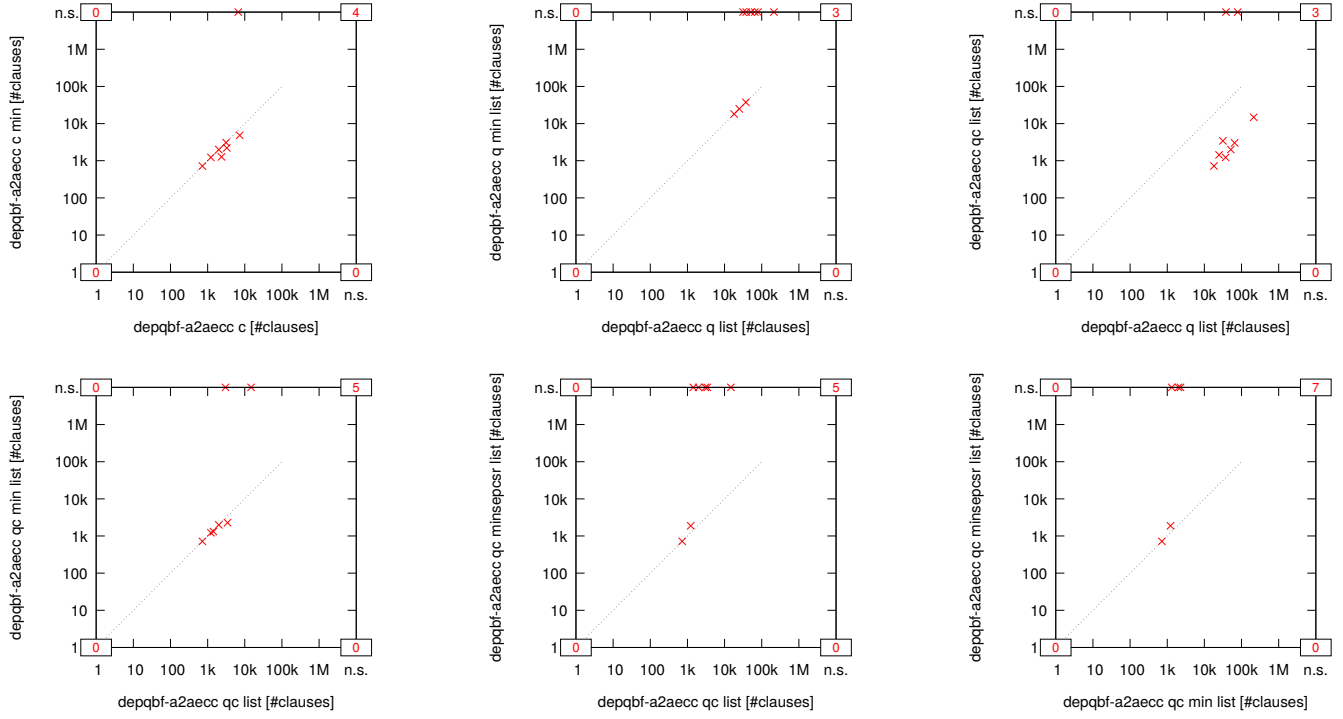


Fig. 36: Suite Ansotegui ($n = 12$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

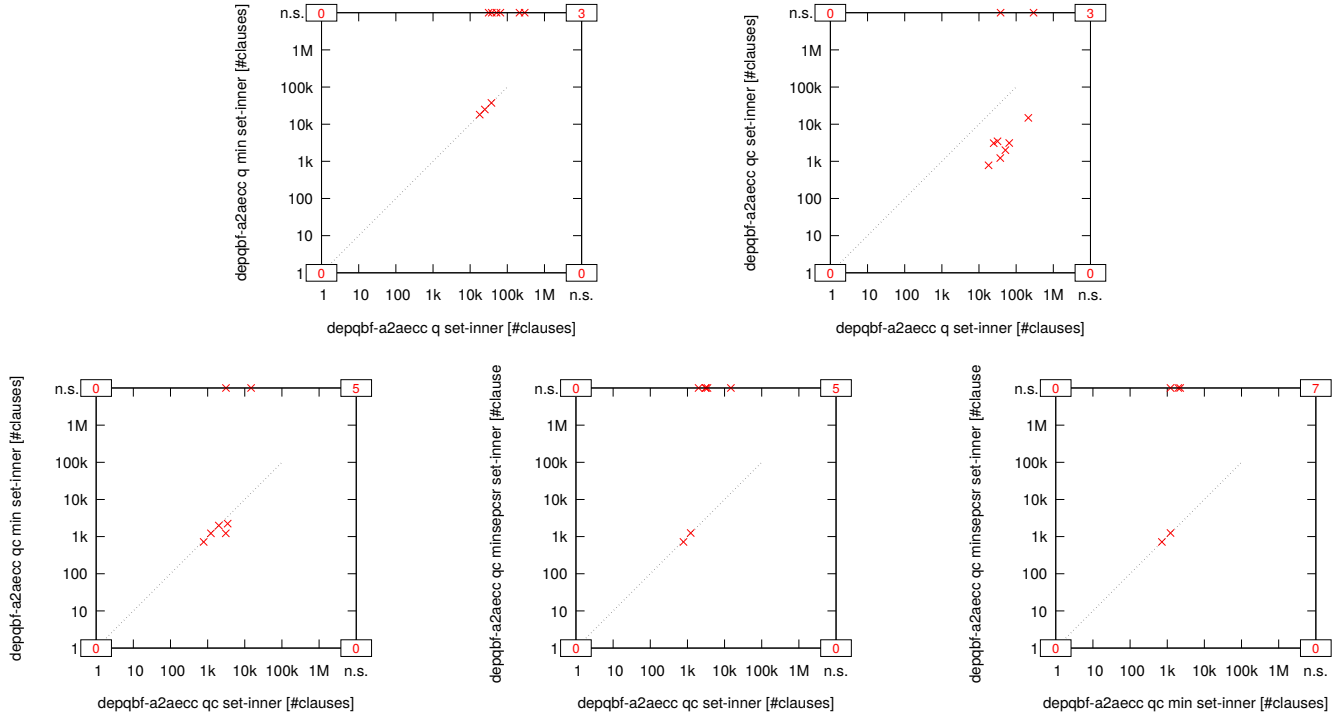


Fig. 37: Suite Ansotegui ($n = 12$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

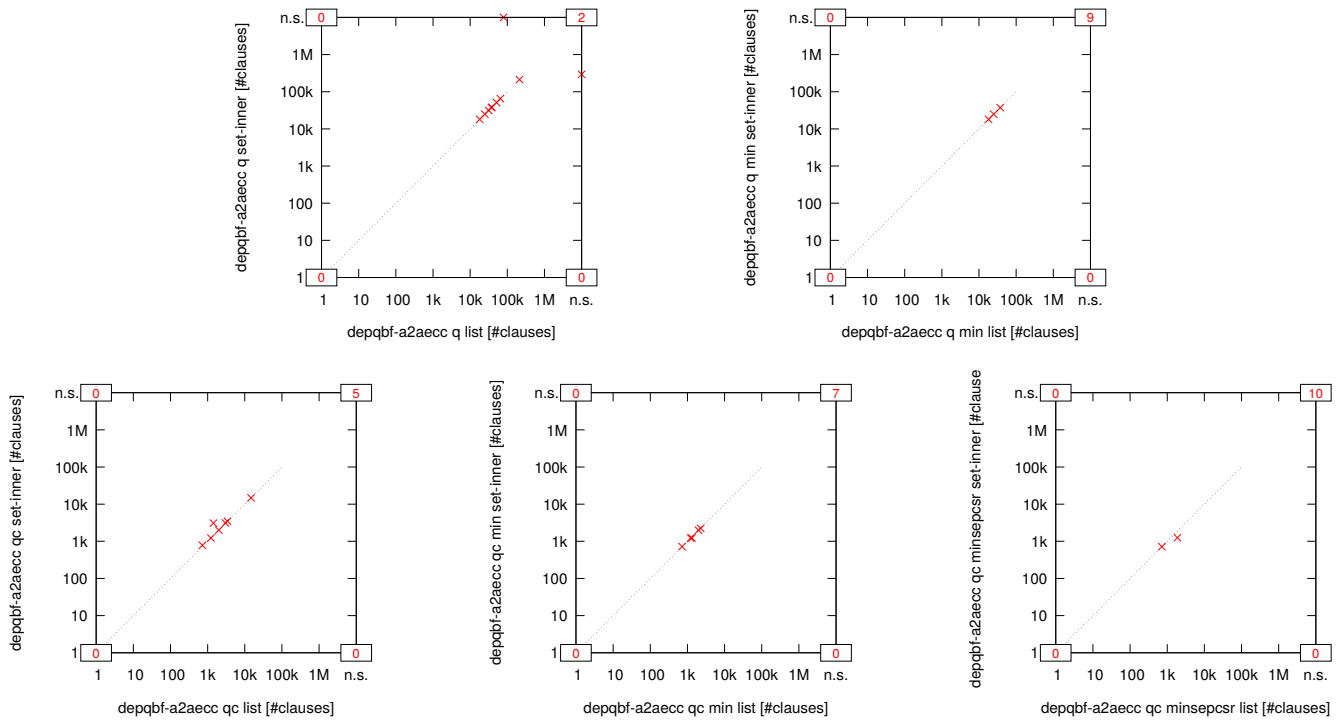


Fig. 38: Suite Ansotegui ($n = 12$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

5) *Ayari* ($n = 48$):

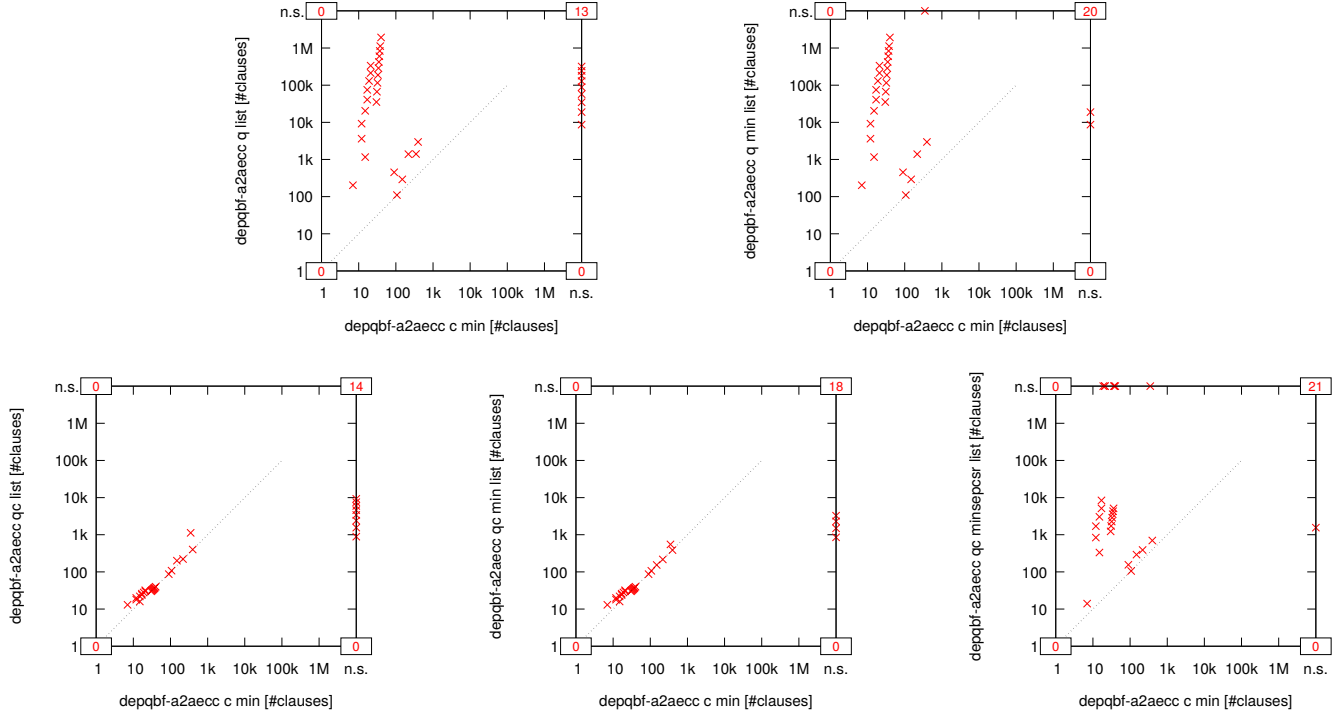


Fig. 39: Suite Ayari ($n = 48$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

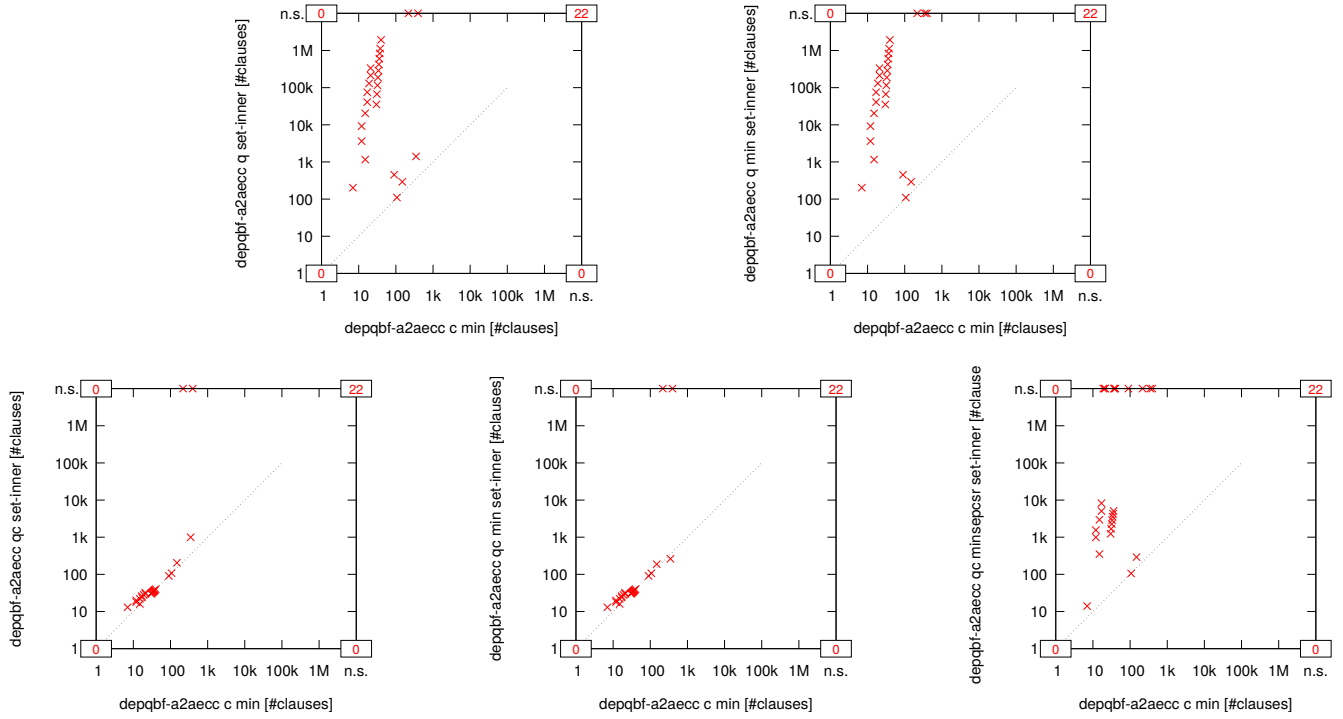


Fig. 40: Suite Ayari ($n = 48$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

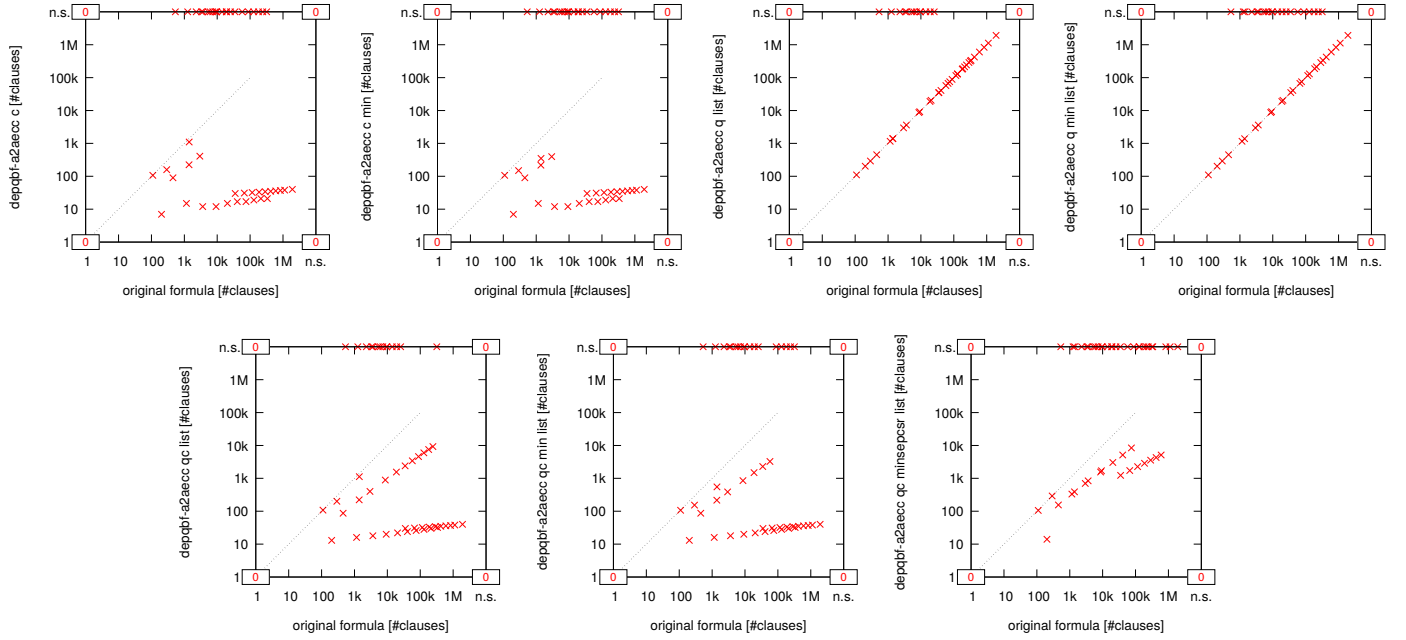


Fig. 41: Suite Ayari ($n = 48$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

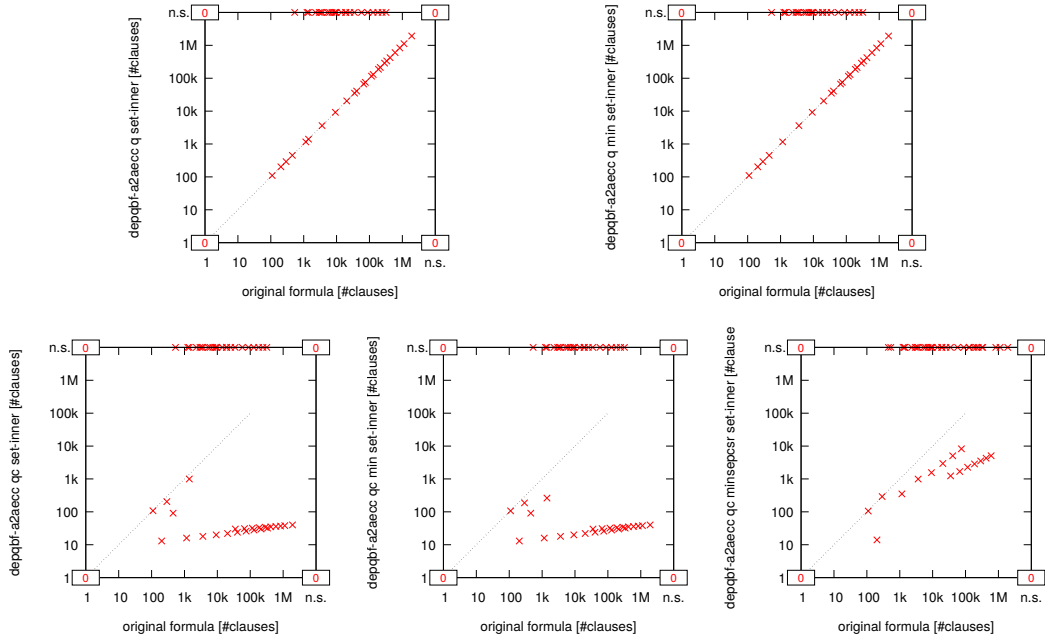


Fig. 42: Suite Ayari ($n = 48$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

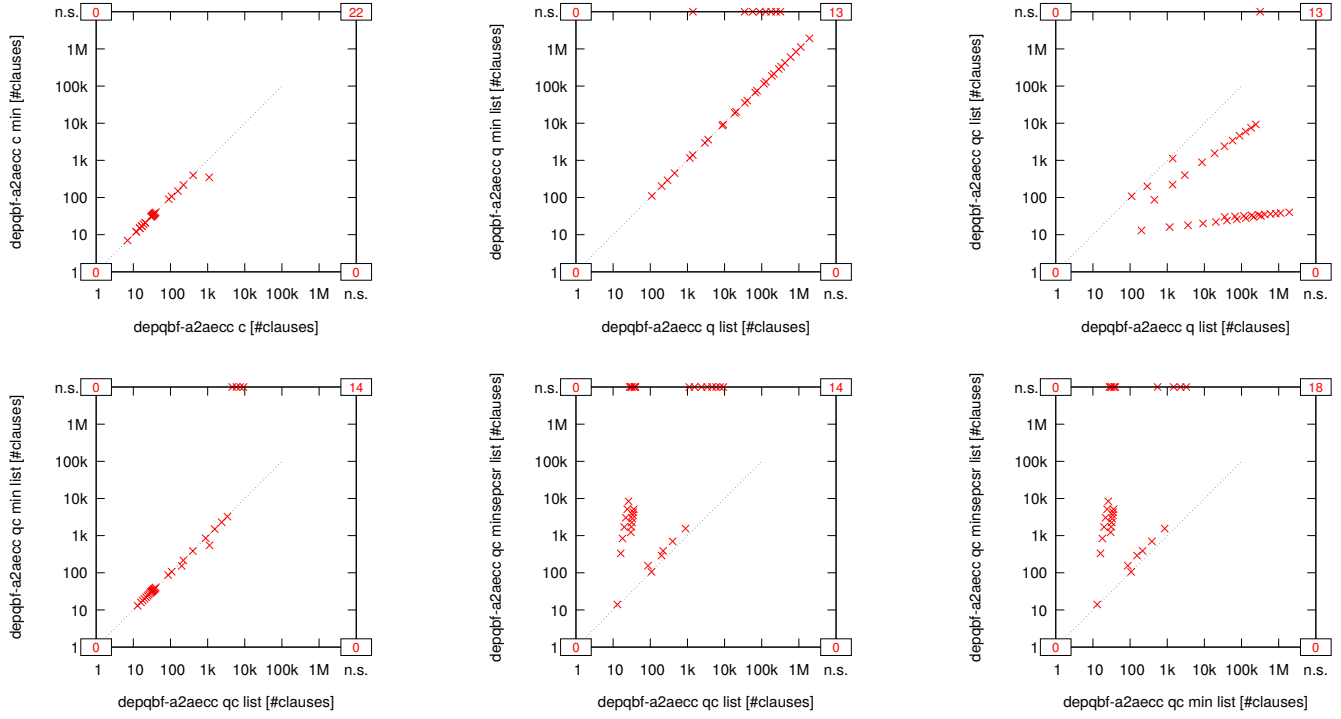


Fig. 43: Suite Ayari ($n = 48$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

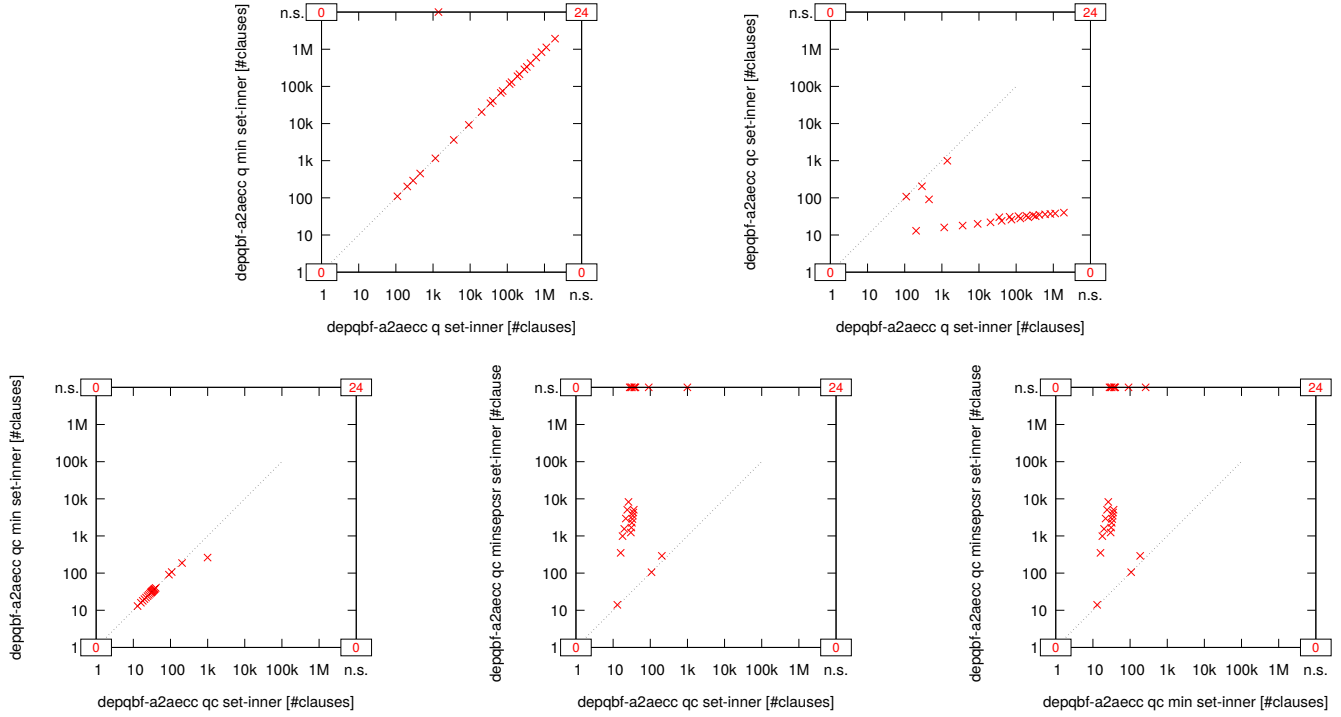


Fig. 44: Suite Ayari ($n = 48$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

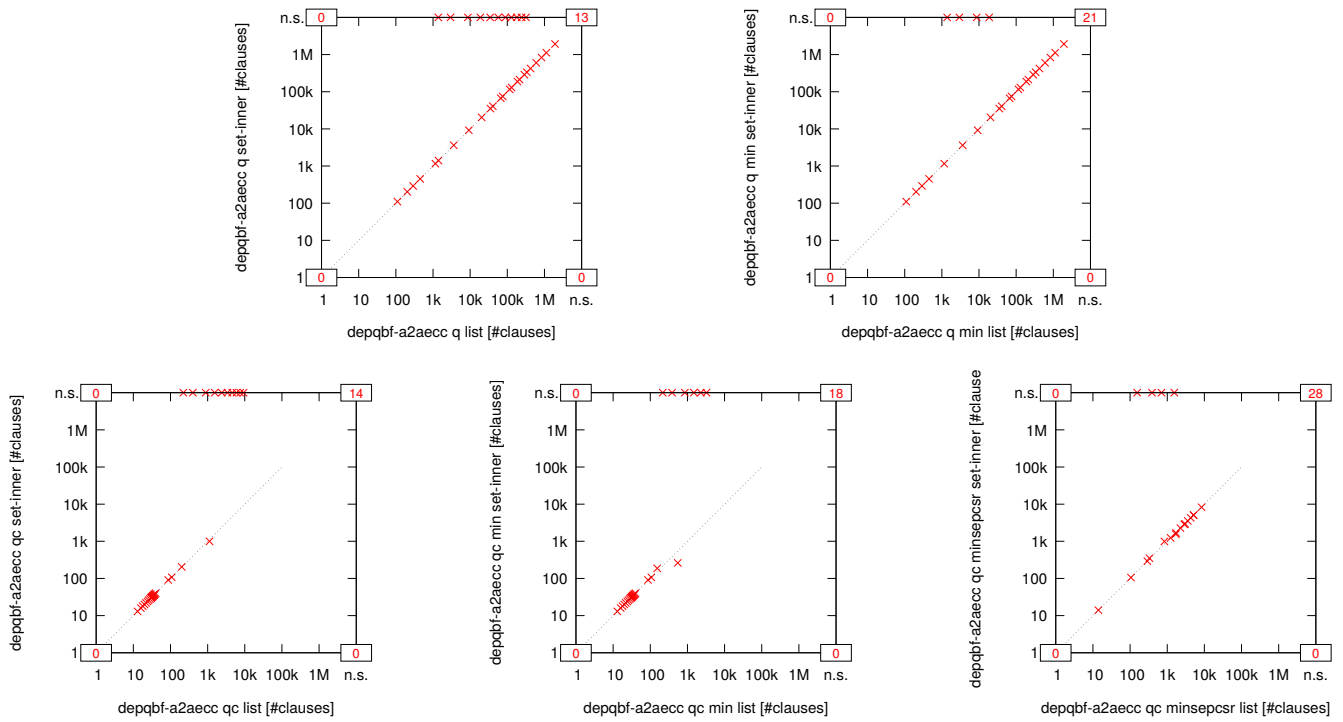


Fig. 45: Suite Ayari ($n = 48$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

6) Basler ($n = 75$):

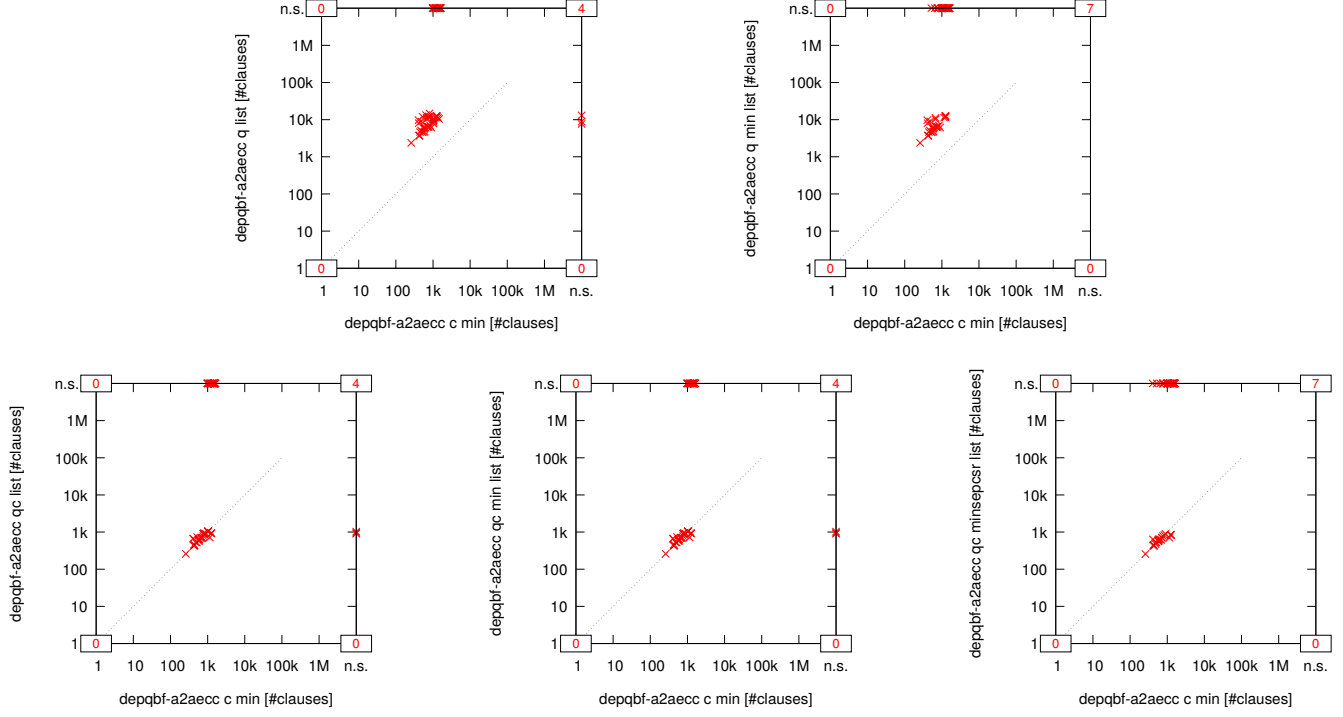


Fig. 46: Suite Basler ($n = 75$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

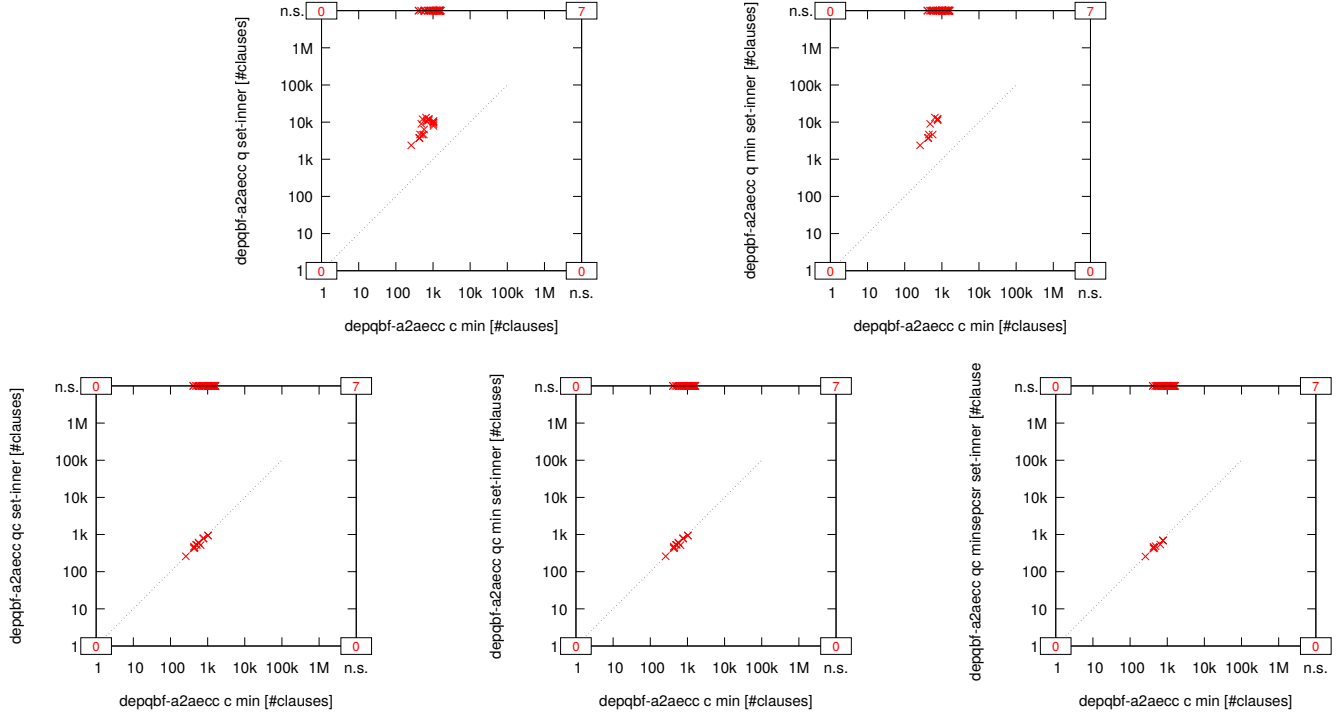


Fig. 47: Suite Basler ($n = 75$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

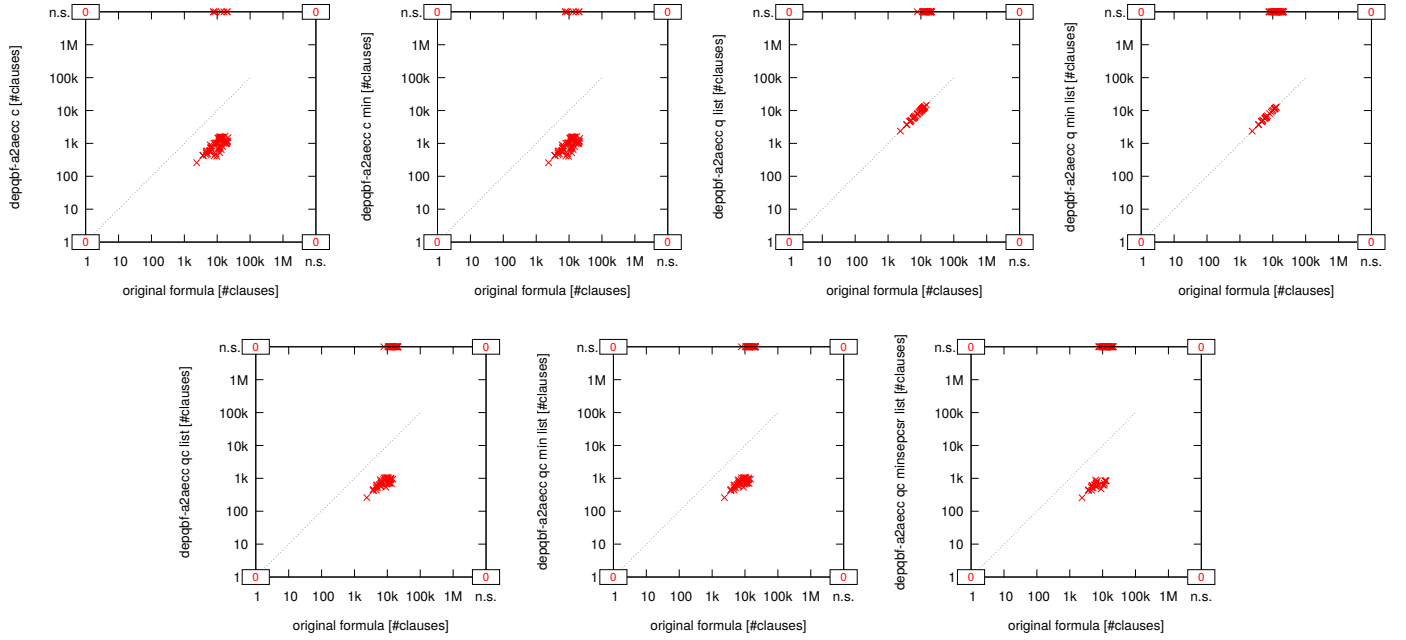


Fig. 48: Suite Basler ($n = 75$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

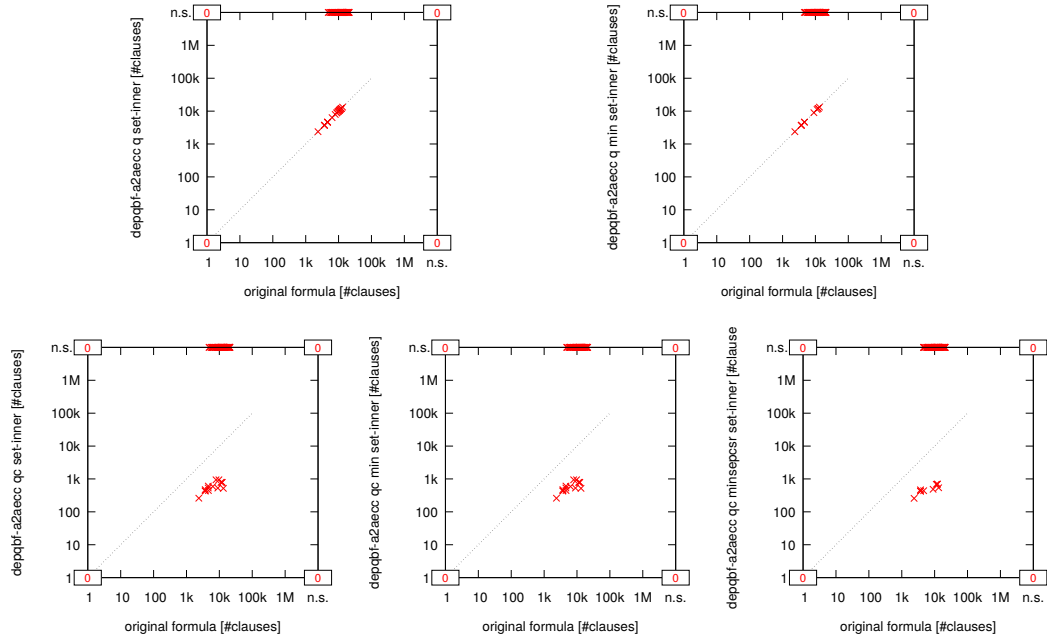


Fig. 49: Suite Basler ($n = 75$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

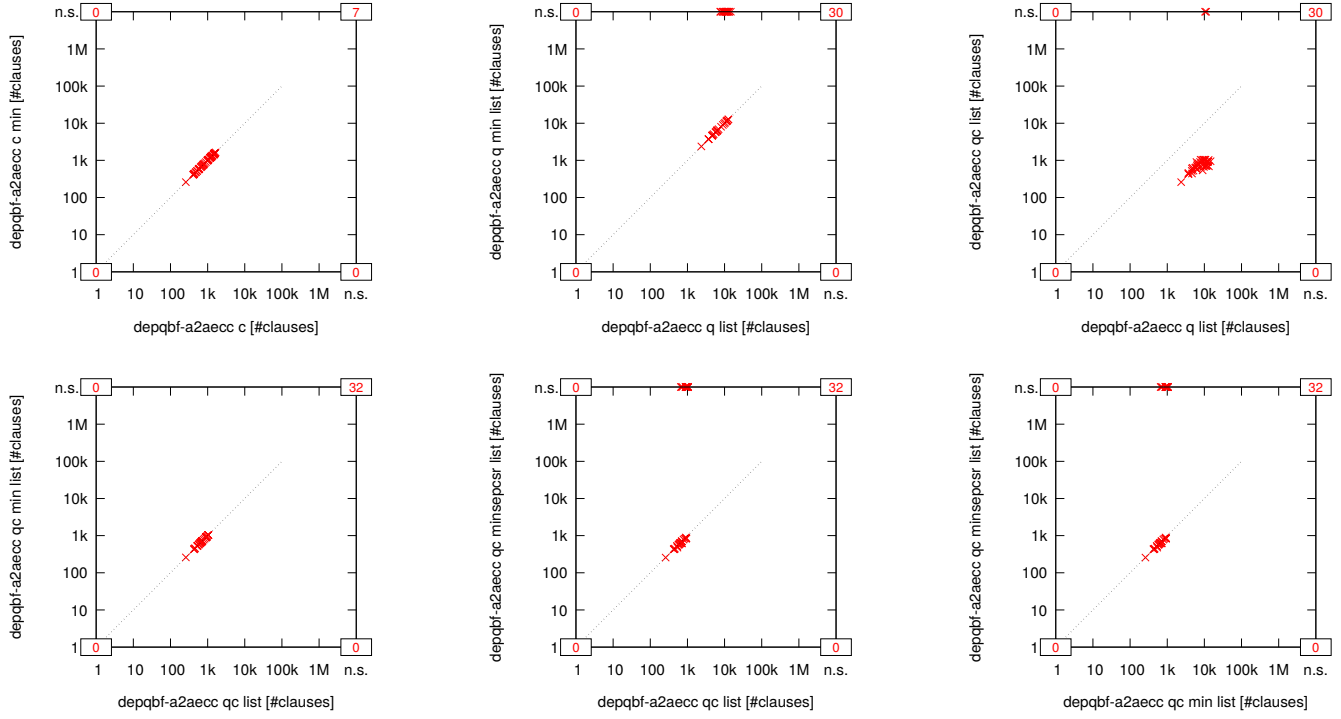


Fig. 50: Suite Basler ($n = 75$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

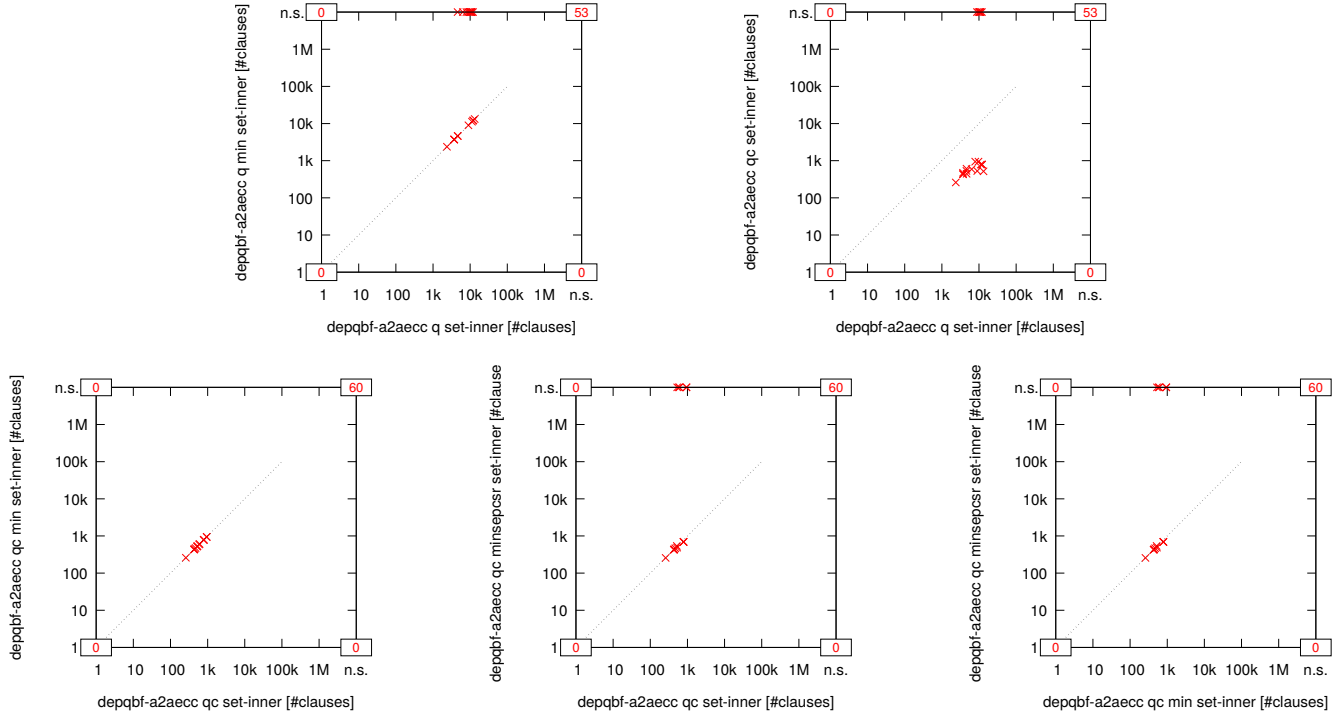


Fig. 51: Suite Basler ($n = 75$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

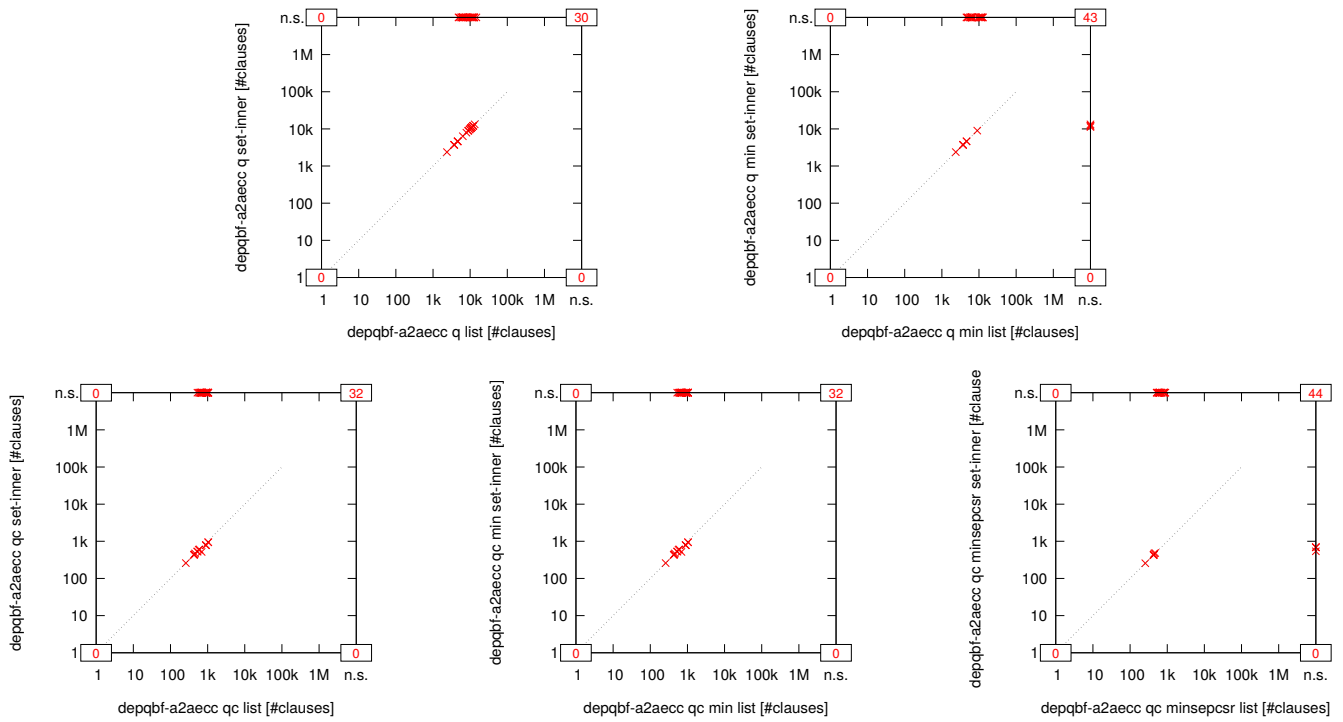


Fig. 52: Suite Basler ($n = 75$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

7) *Biere* ($n = 25$):

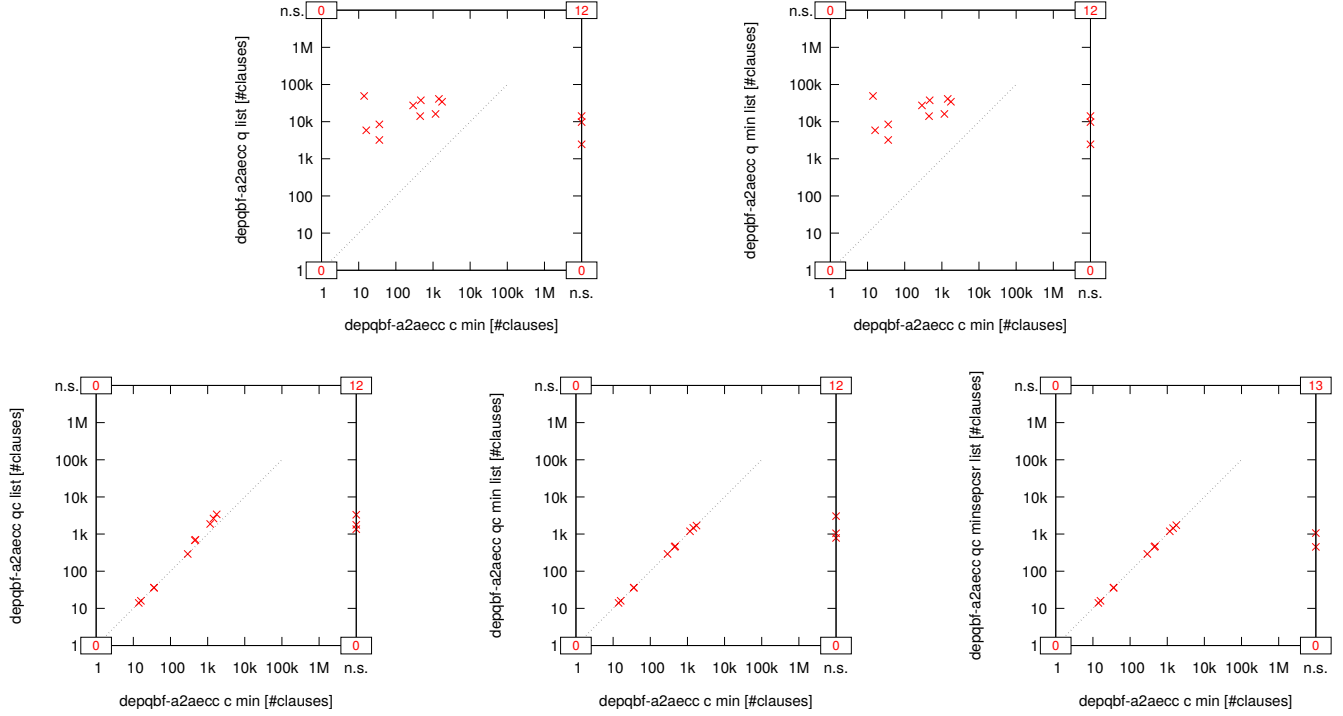


Fig. 53: Suite Biere ($n = 25$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

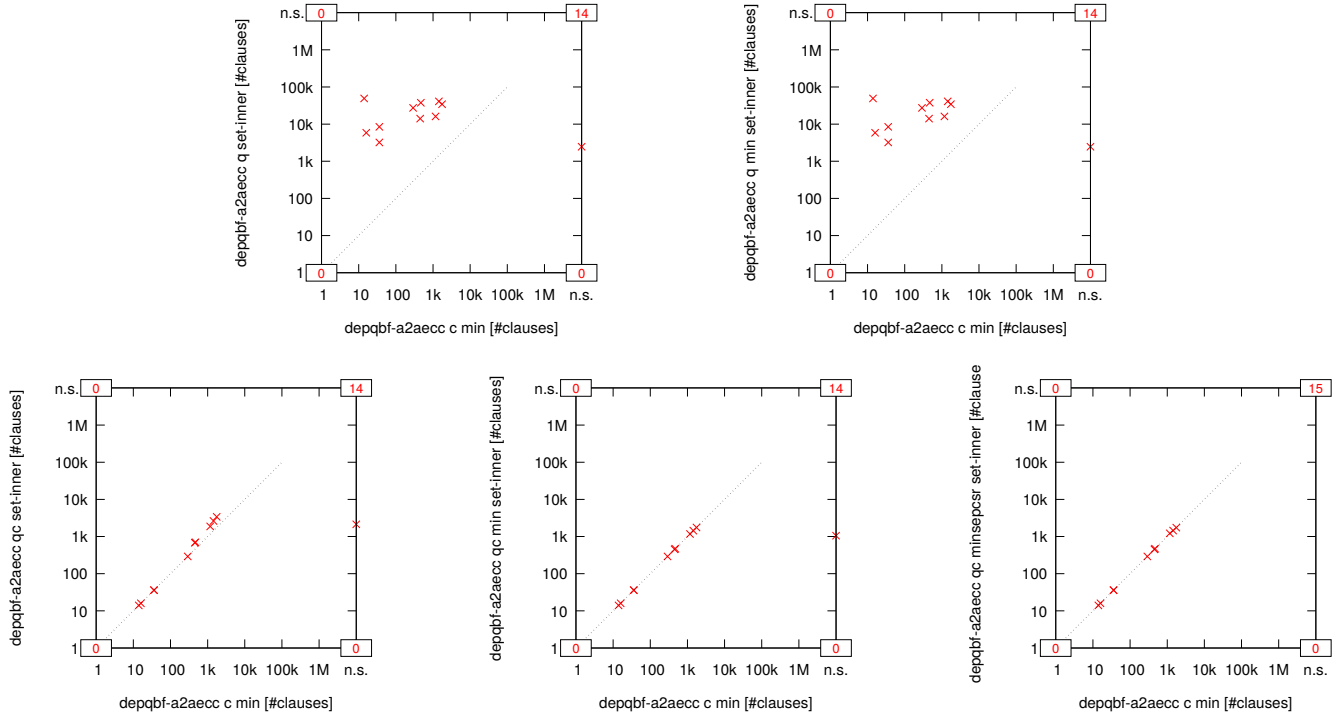


Fig. 54: Suite Biere ($n = 25$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

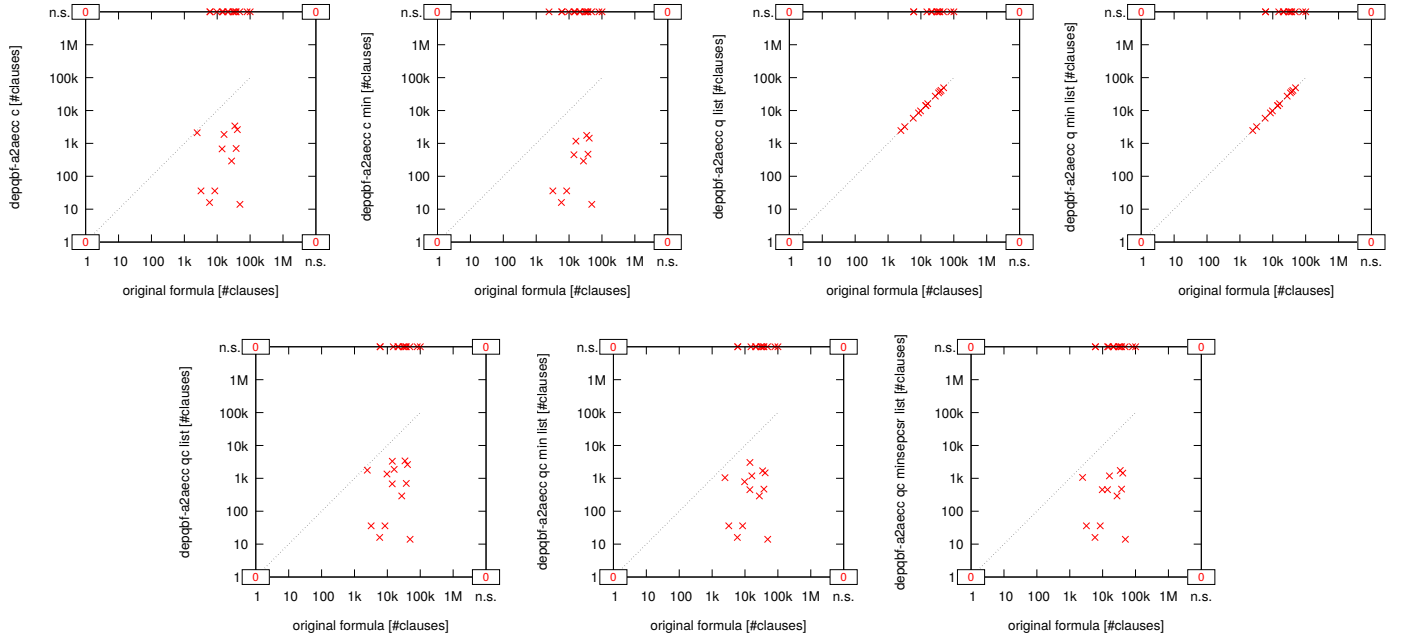


Fig. 55: Suite Biere ($n = 25$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

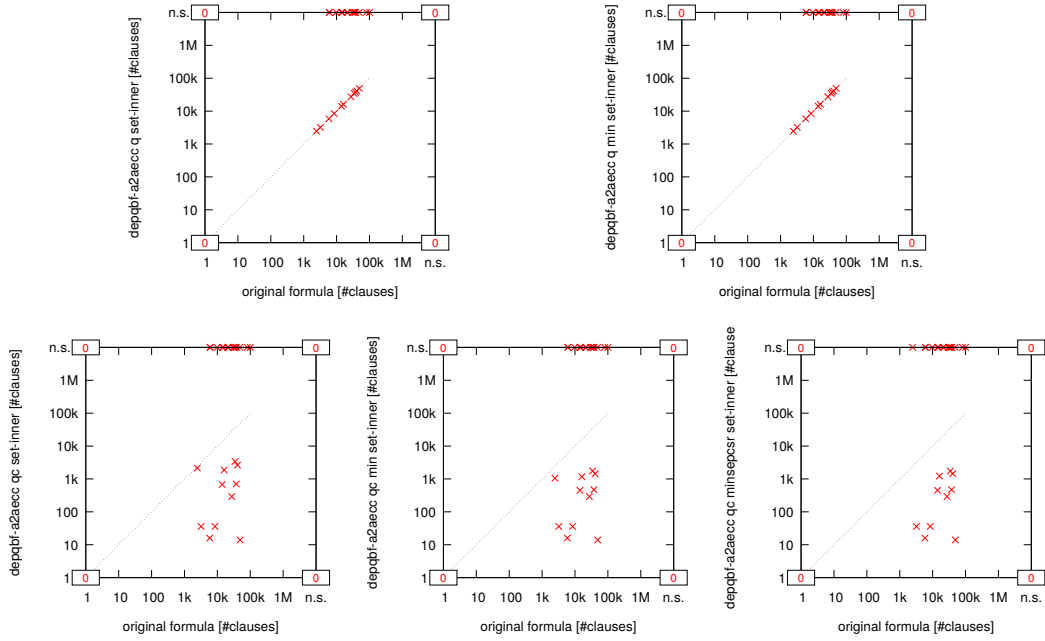


Fig. 56: Suite Biere ($n = 25$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

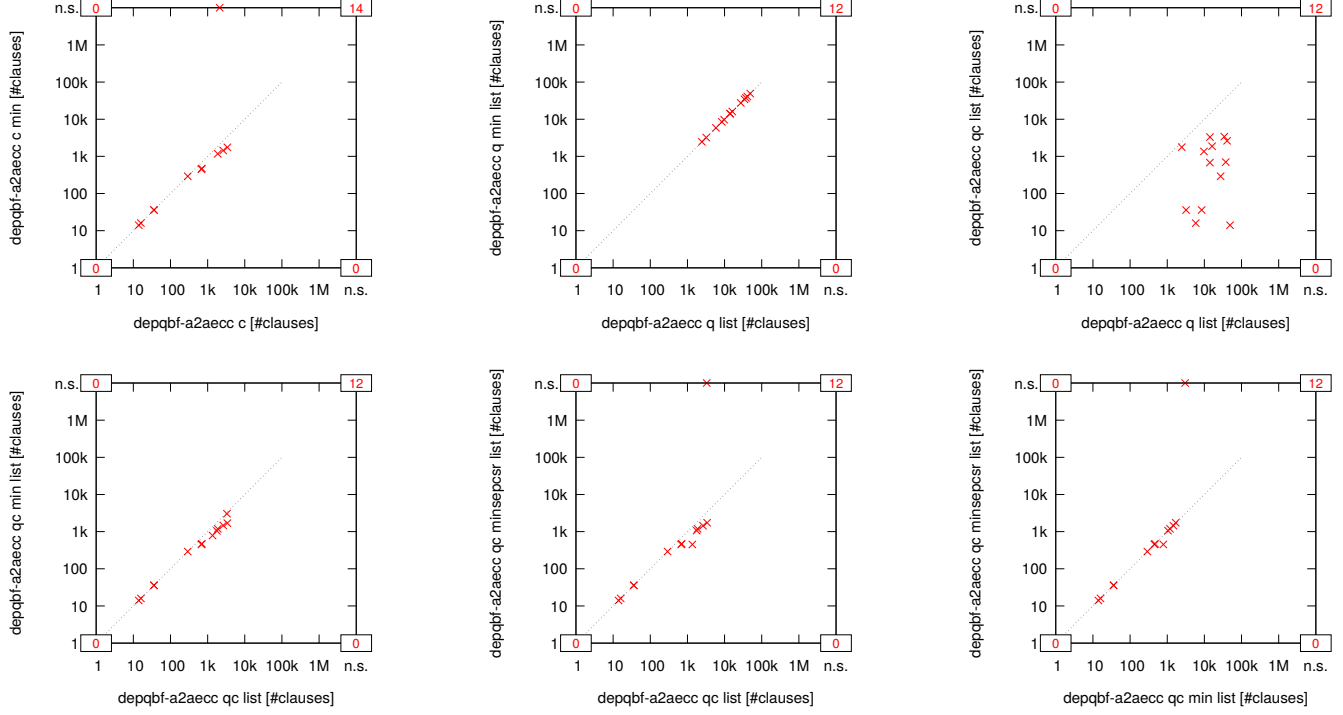


Fig. 57: Suite Biere ($n = 25$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

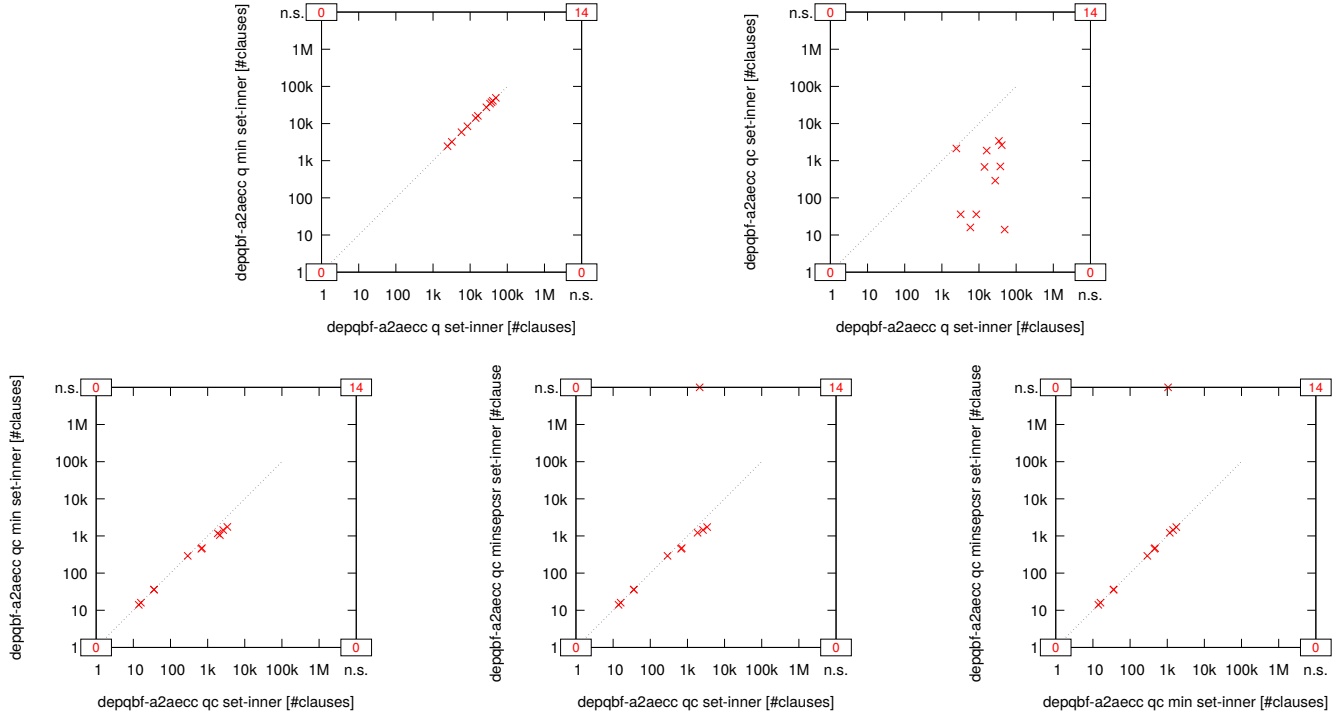


Fig. 58: Suite Biere ($n = 25$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

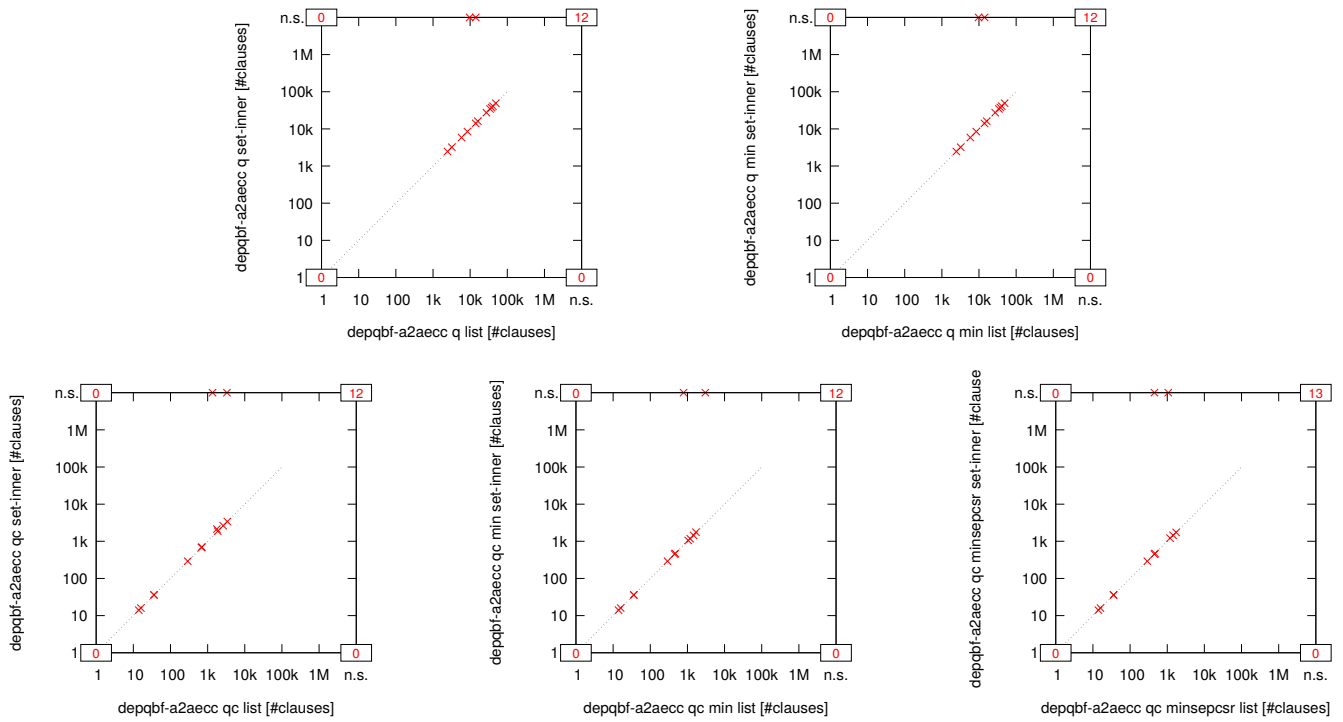


Fig. 59: Suite Biere ($n = 25$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

8) *Cashmore-Fox-Giunchiglia* ($n = 110$):

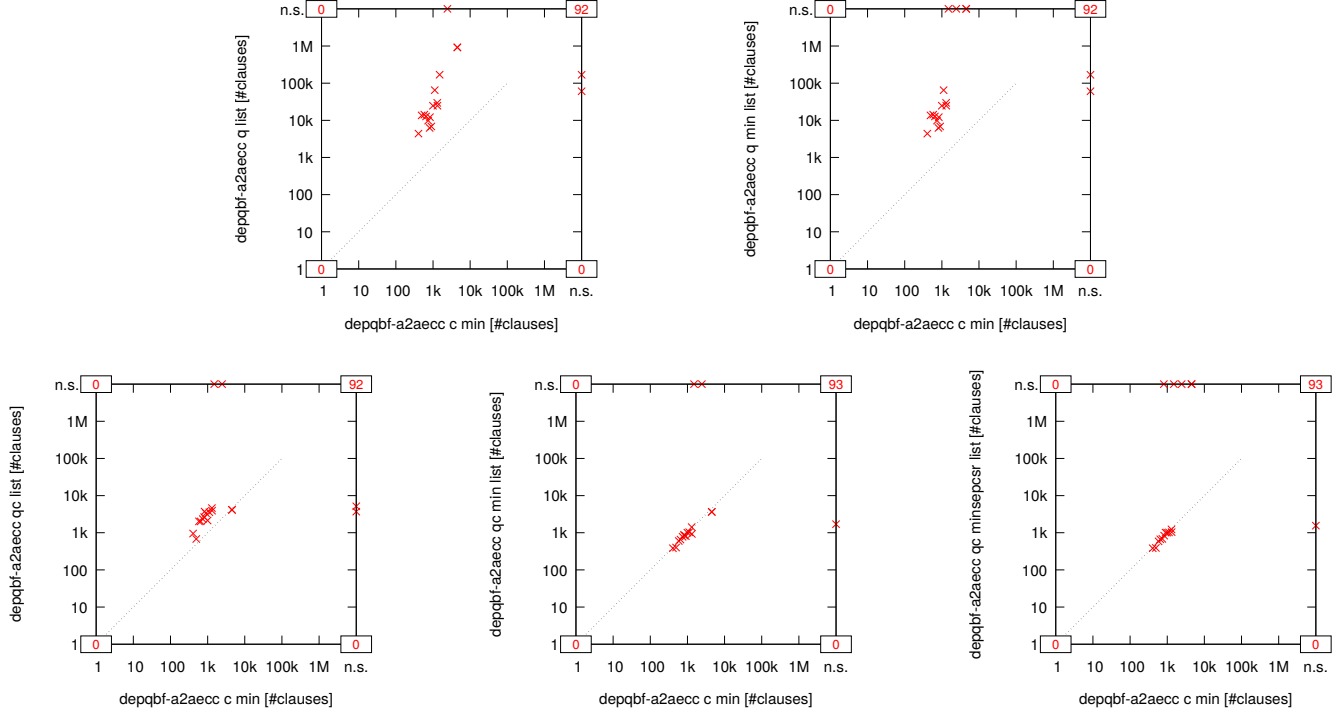


Fig. 60: Suite *Cashmore-Fox-Giunchiglia* ($n = 110$): Comparing sizes of unsatisfiable cores in *DepQBF-a2aecc* in list semantics versus mode *c min* (number of clauses).

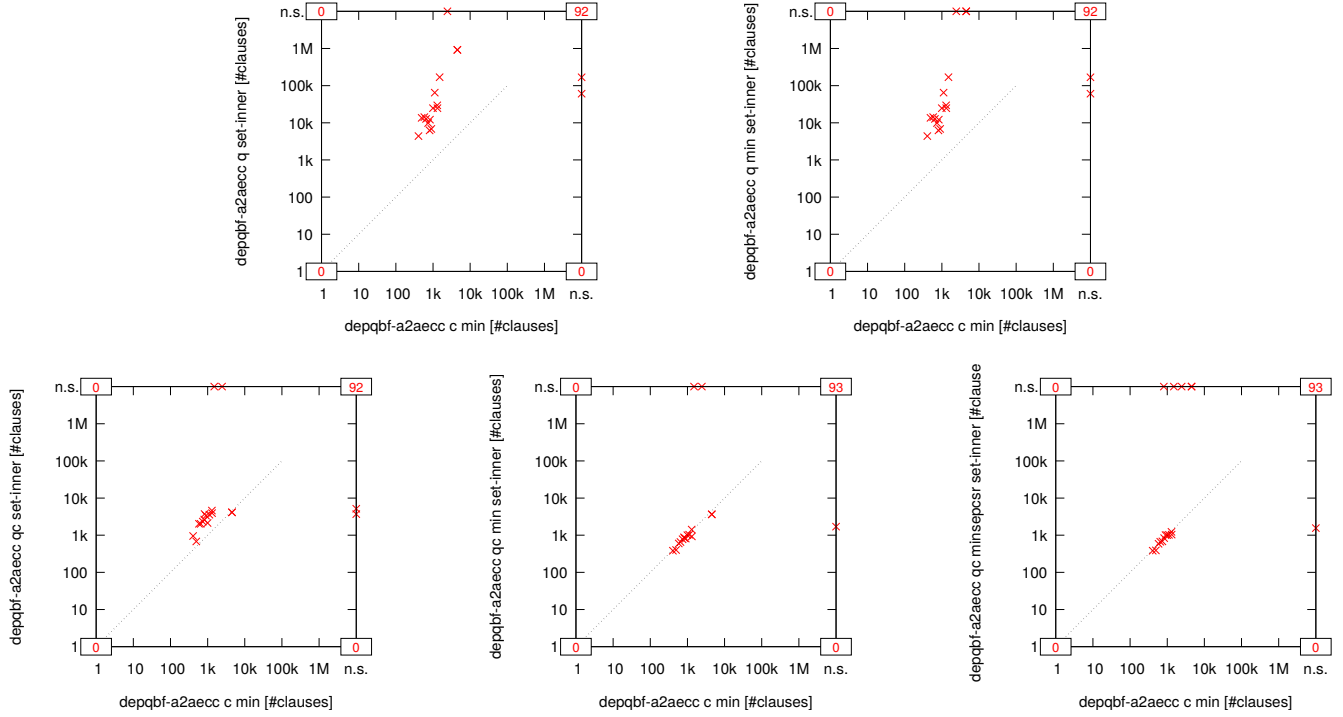


Fig. 61: Suite *Cashmore-Fox-Giunchiglia* ($n = 110$): Comparing sizes of unsatisfiable cores in *DepQBF-a2aecc* in set-inner semantics versus mode *c min* (number of clauses).

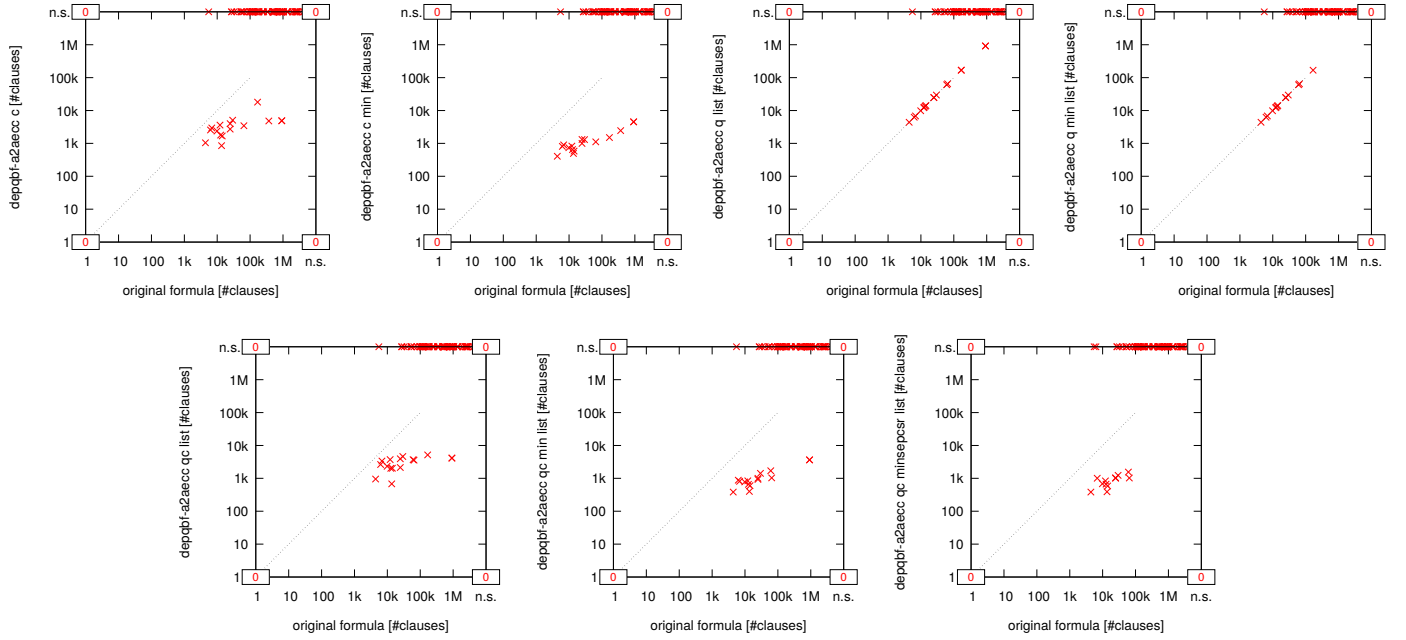


Fig. 62: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

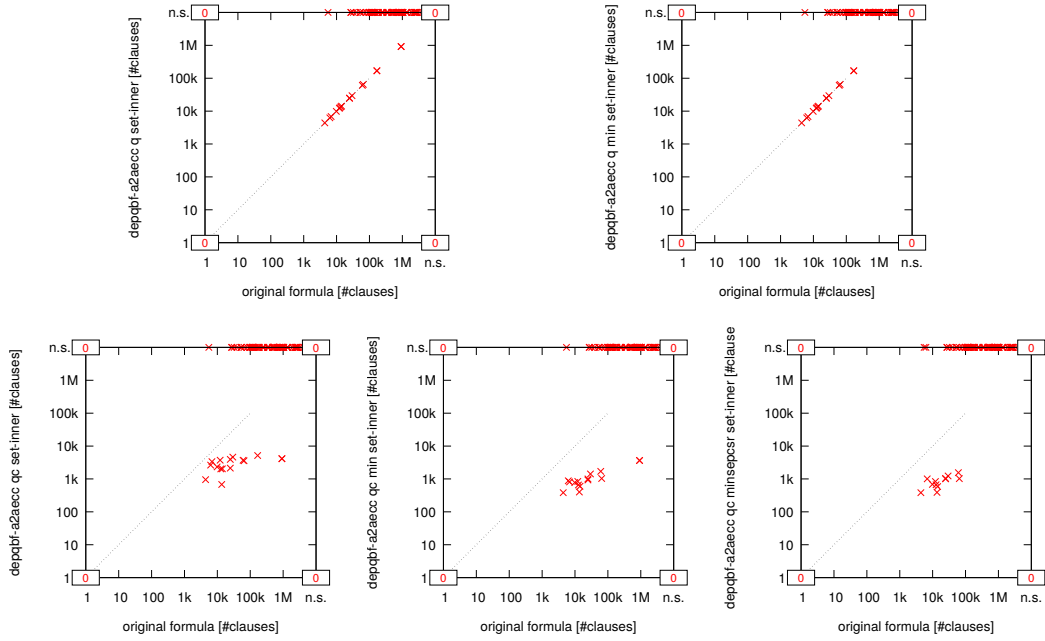


Fig. 63: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

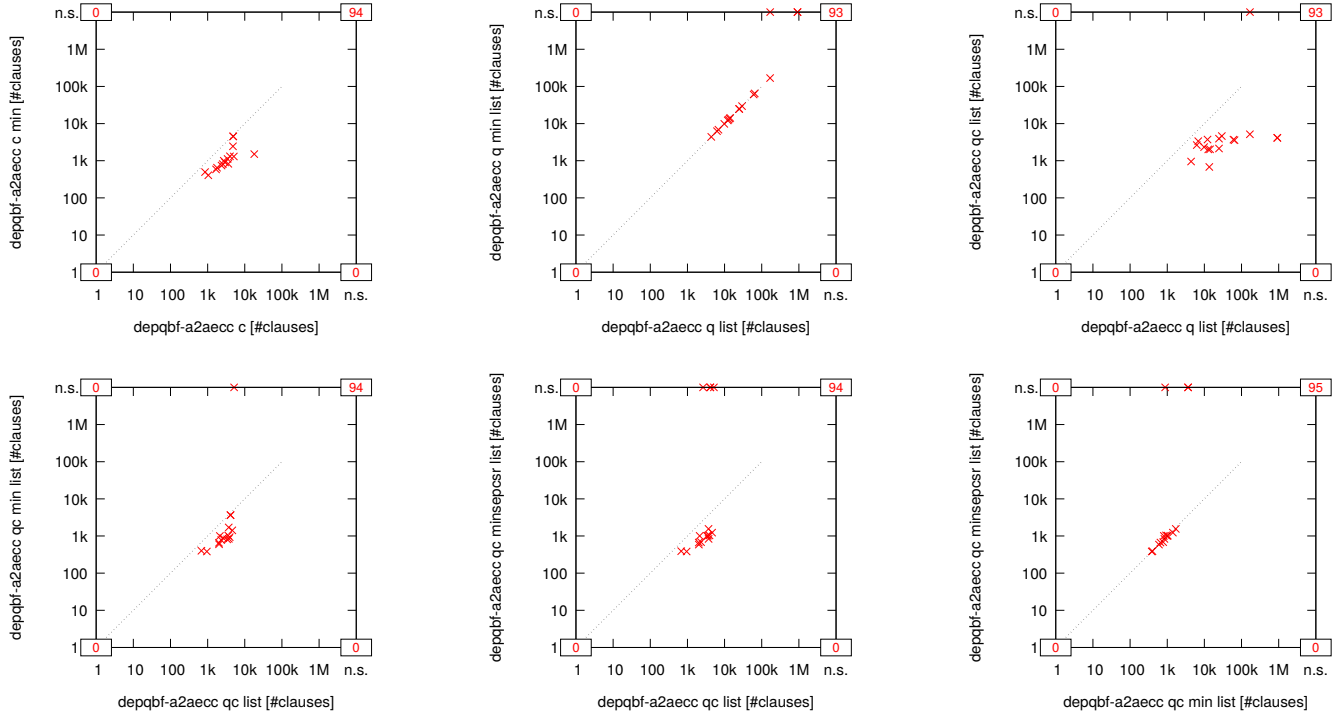


Fig. 64: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

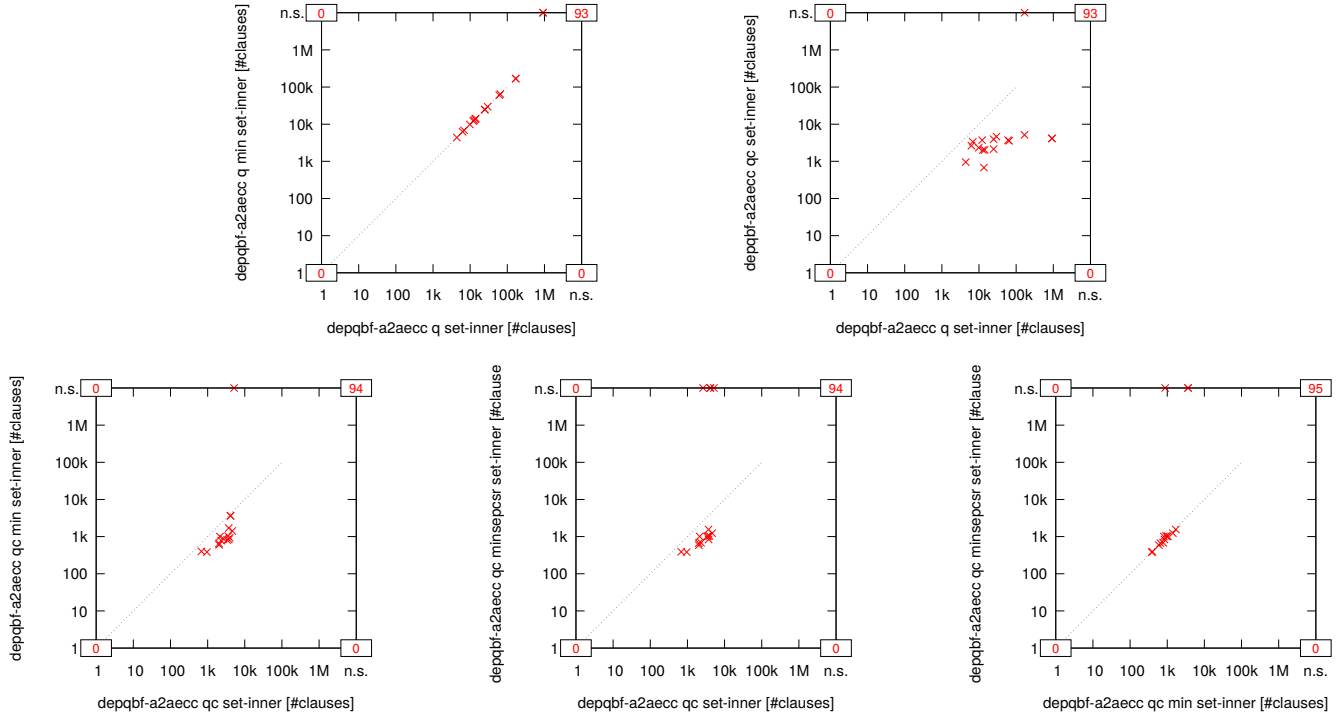


Fig. 65: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

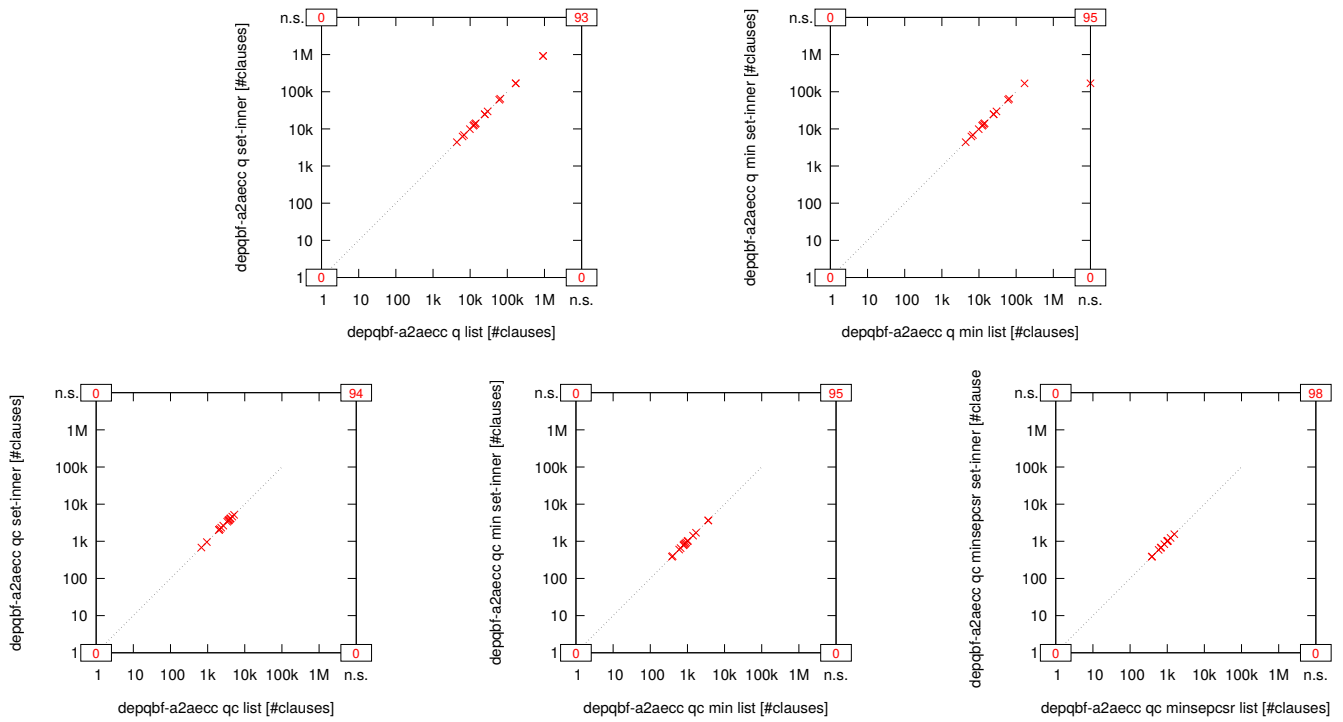


Fig. 66: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Comparing sizes of different unsatisfiable cores in `DepQBF-a2aecc` in list versus set-inner semantics (number of clauses).

9) Castellini ($n = 112$):

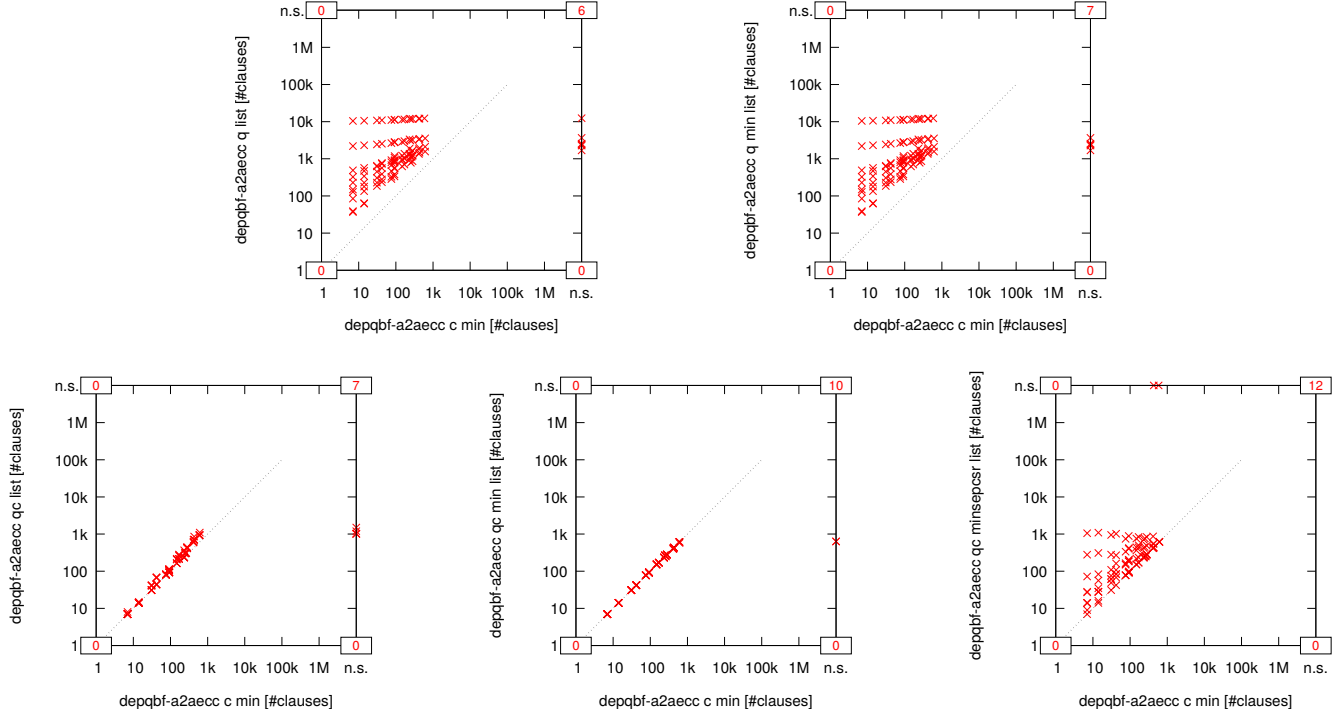


Fig. 67: Suite Castellini ($n = 112$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

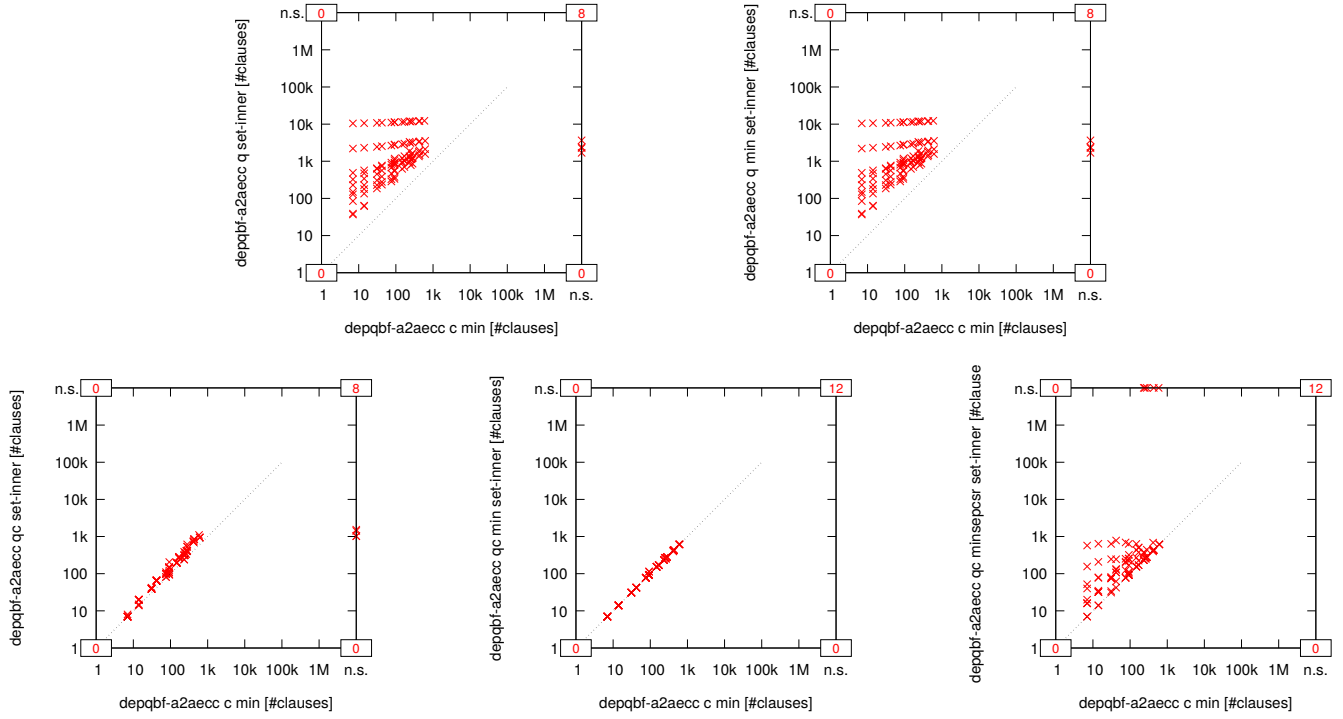


Fig. 68: Suite Castellini ($n = 112$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

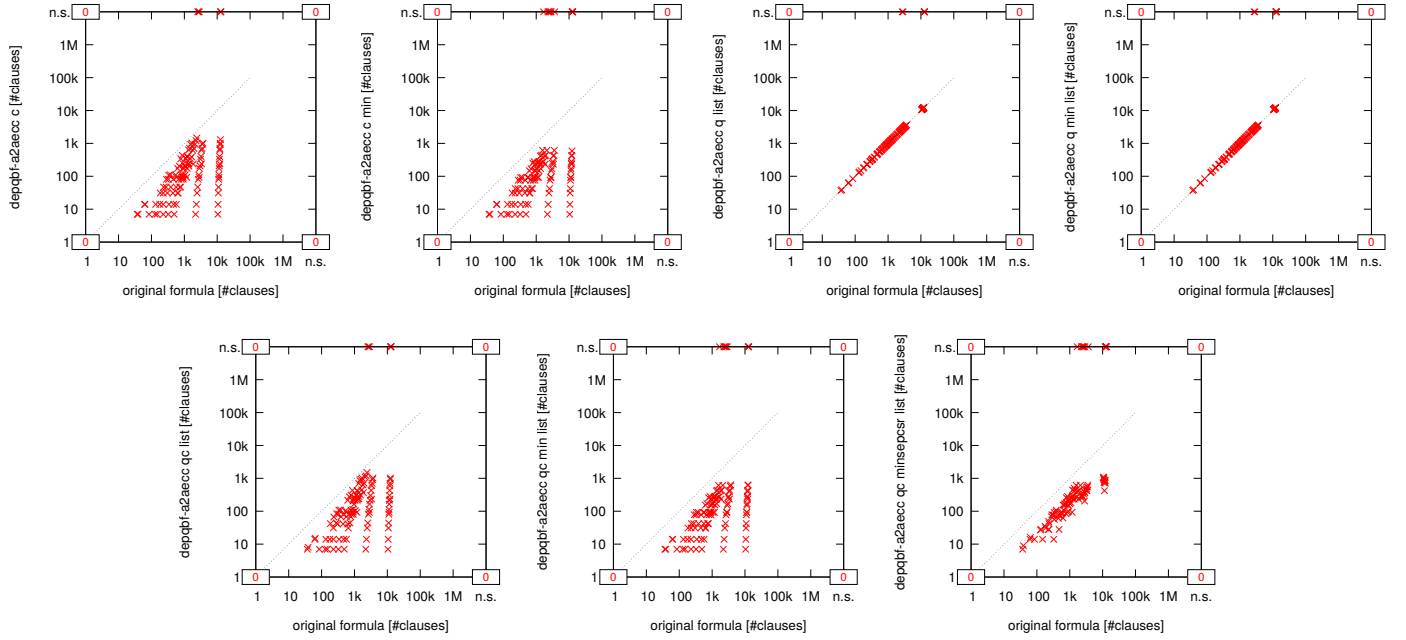


Fig. 69: Suite Castellini ($n = 112$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

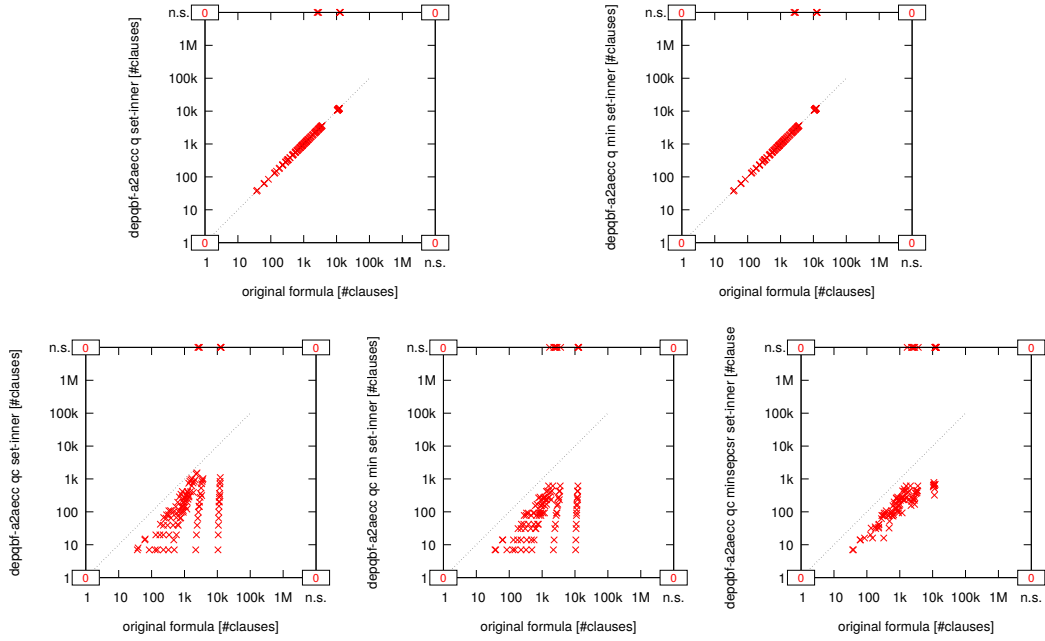


Fig. 70: Suite Castellini ($n = 112$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

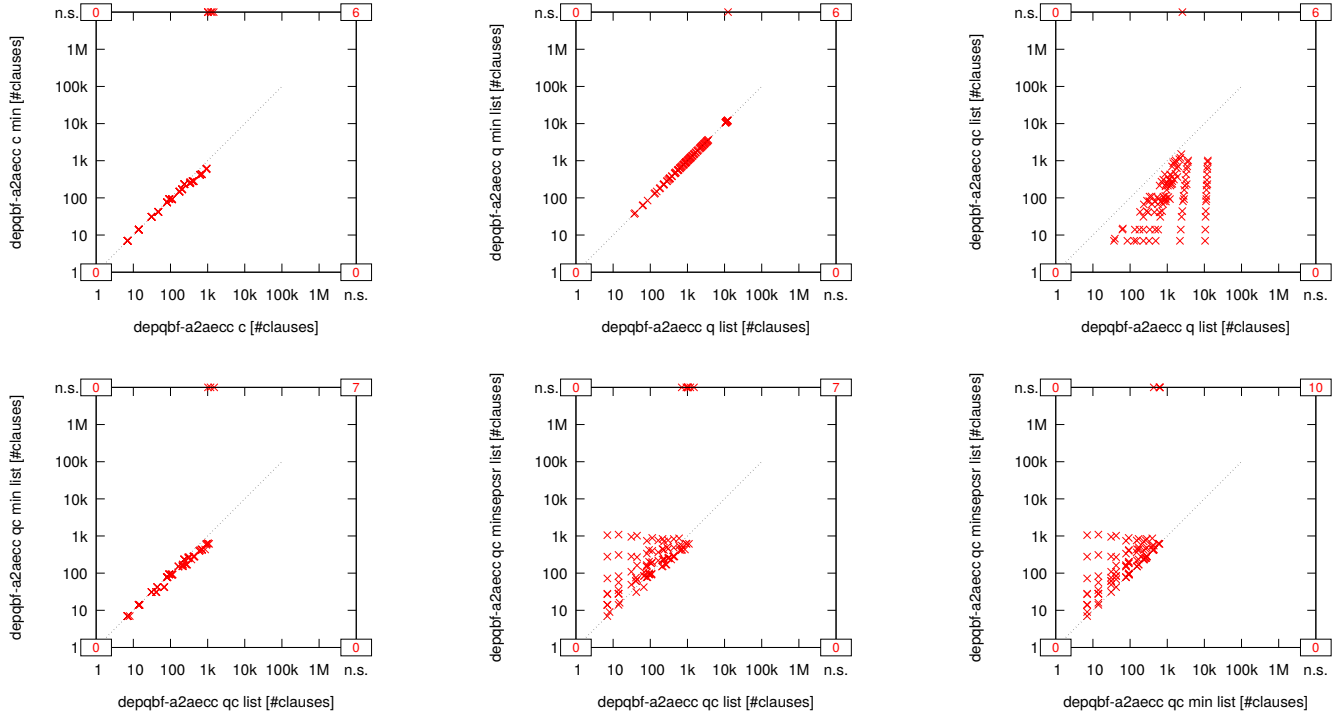


Fig. 71: Suite Castellini ($n = 112$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

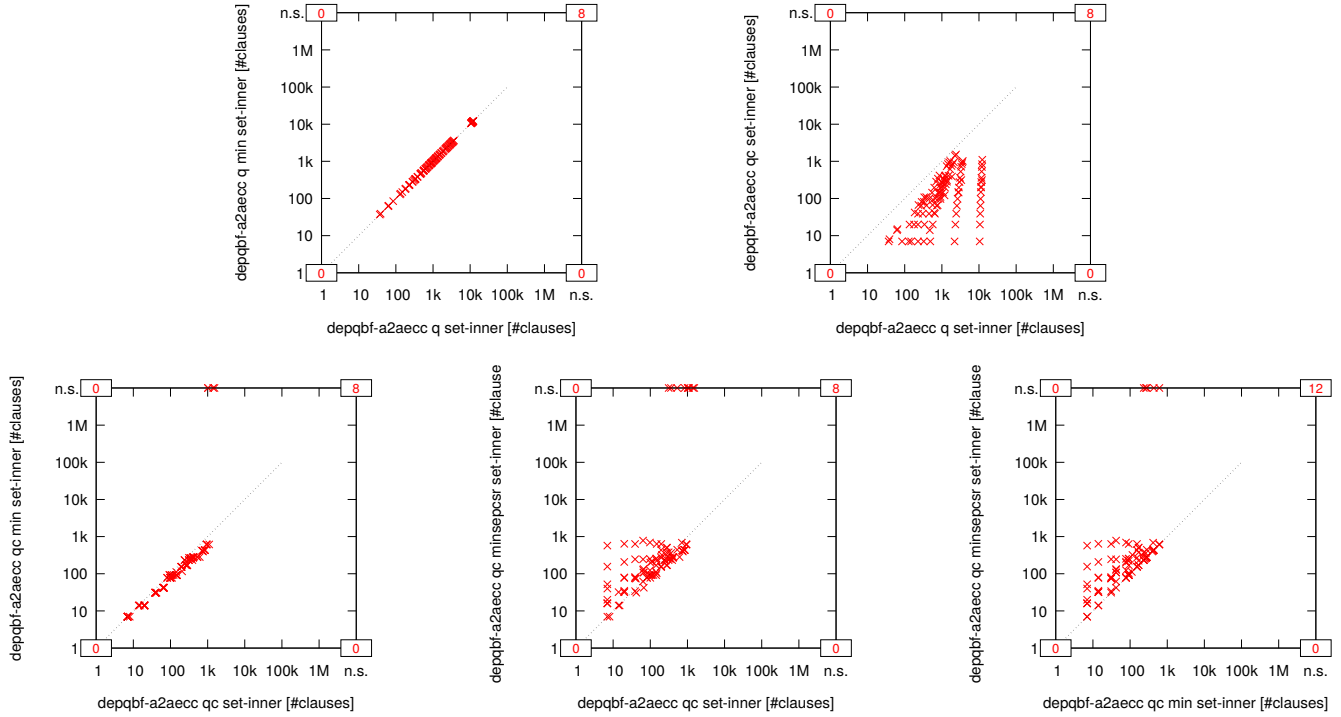


Fig. 72: Suite Castellini ($n = 112$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

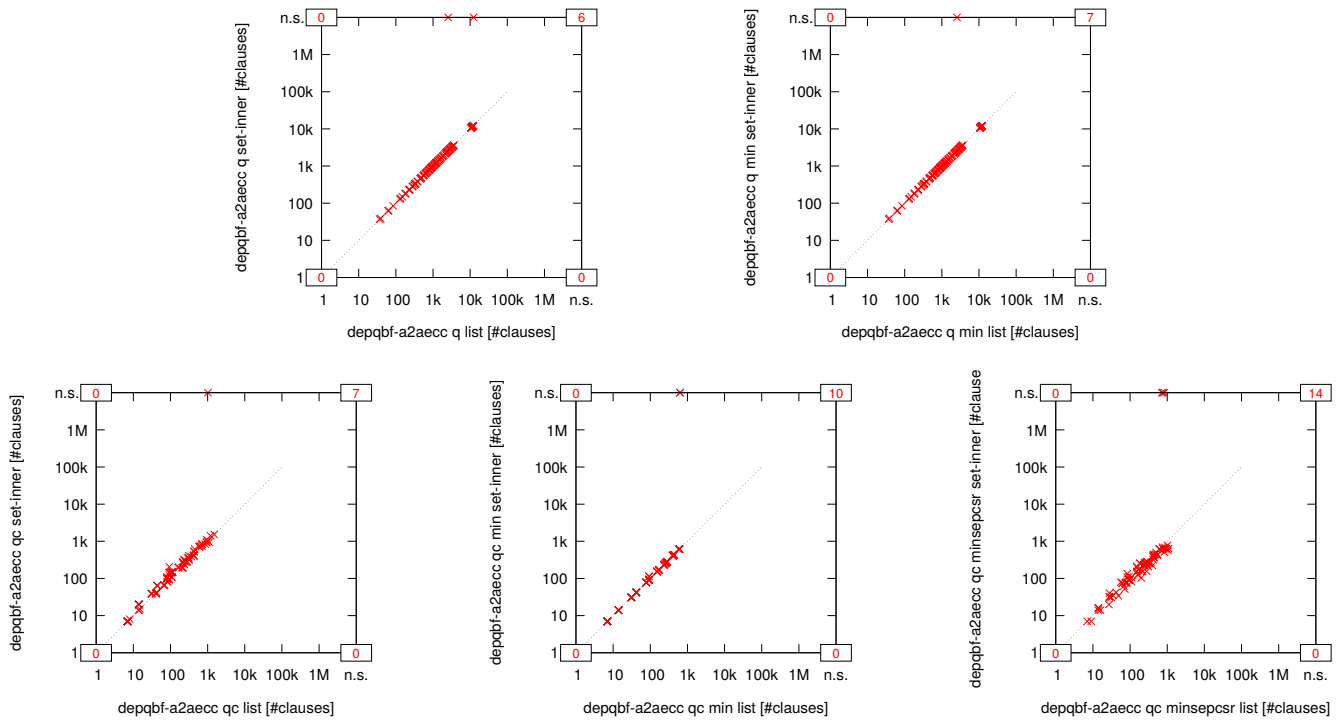


Fig. 73: Suite Castellini ($n = 112$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

10) *Chen-Interian* ($n = 16$):

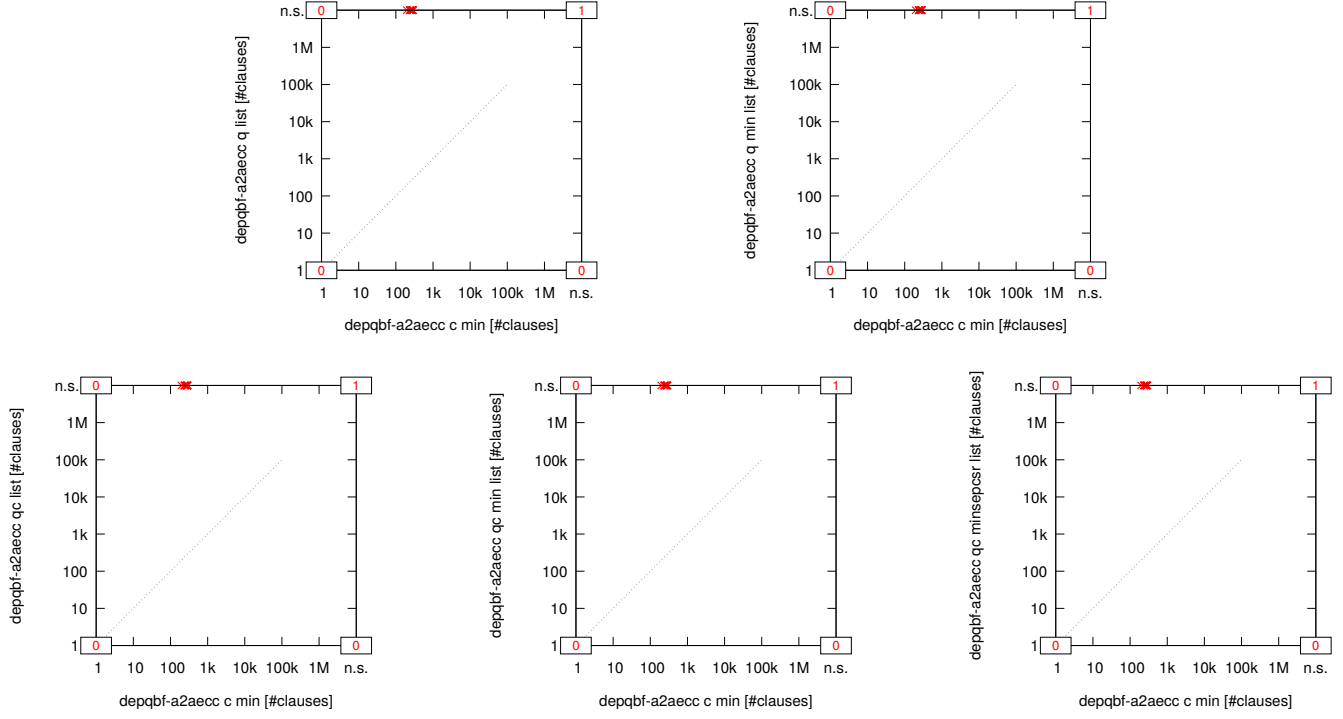


Fig. 74: Suite Chen-Interian ($n = 16$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

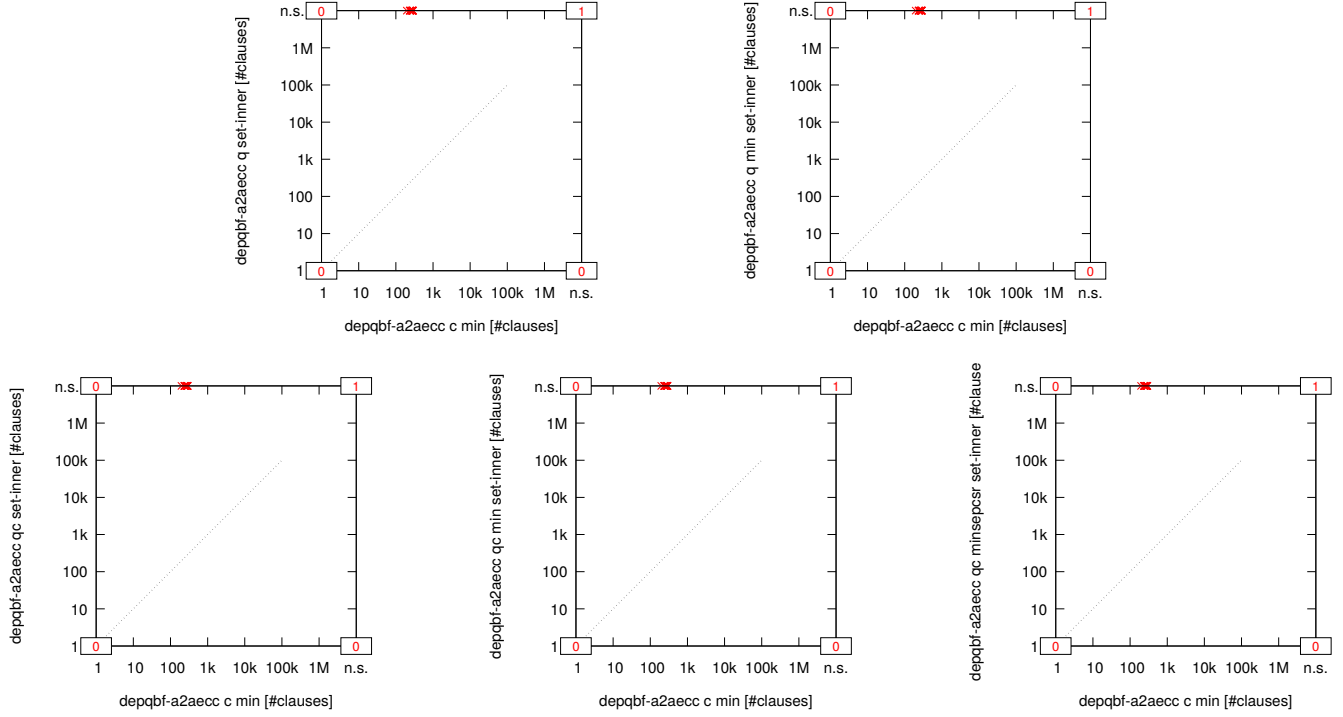


Fig. 75: Suite Chen-Interian ($n = 16$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

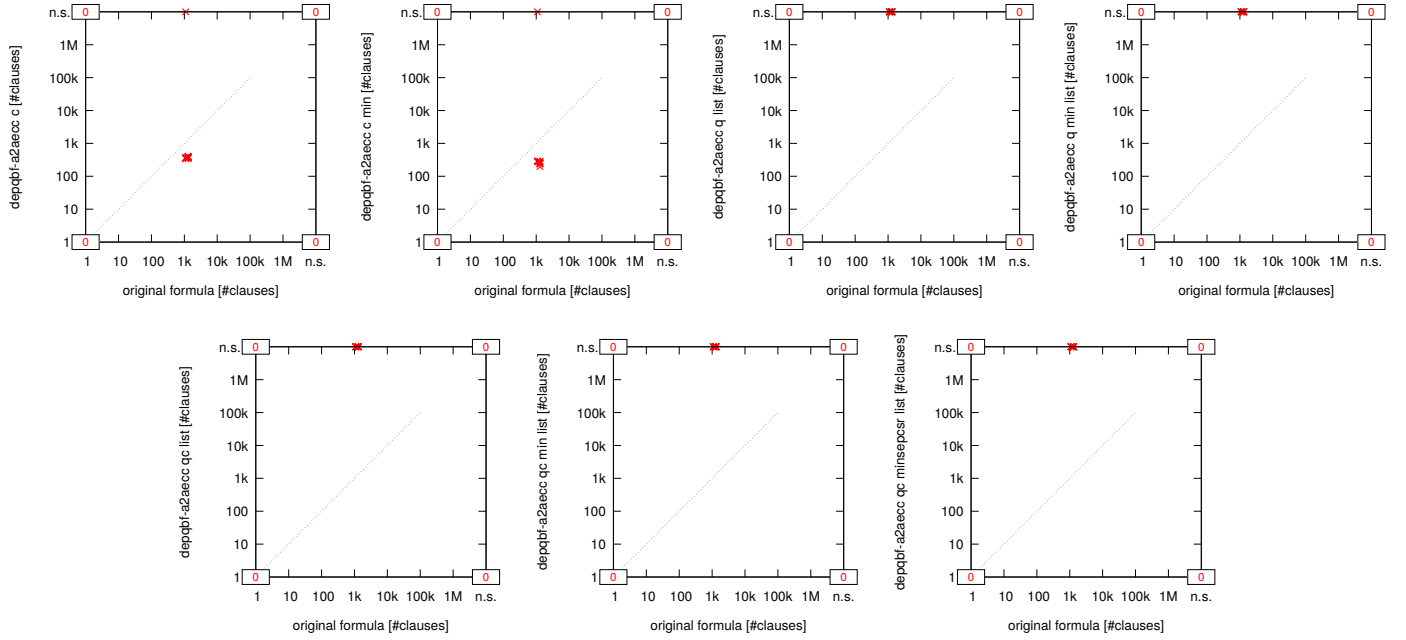


Fig. 76: Suite Chen-Interian ($n = 16$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

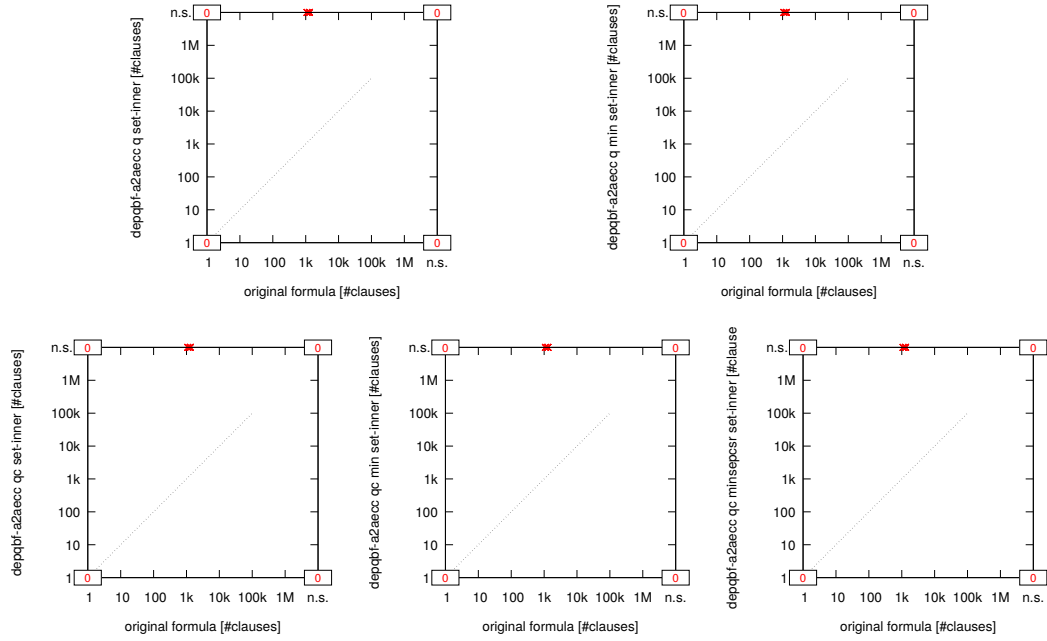


Fig. 77: Suite Chen-Interian ($n = 16$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

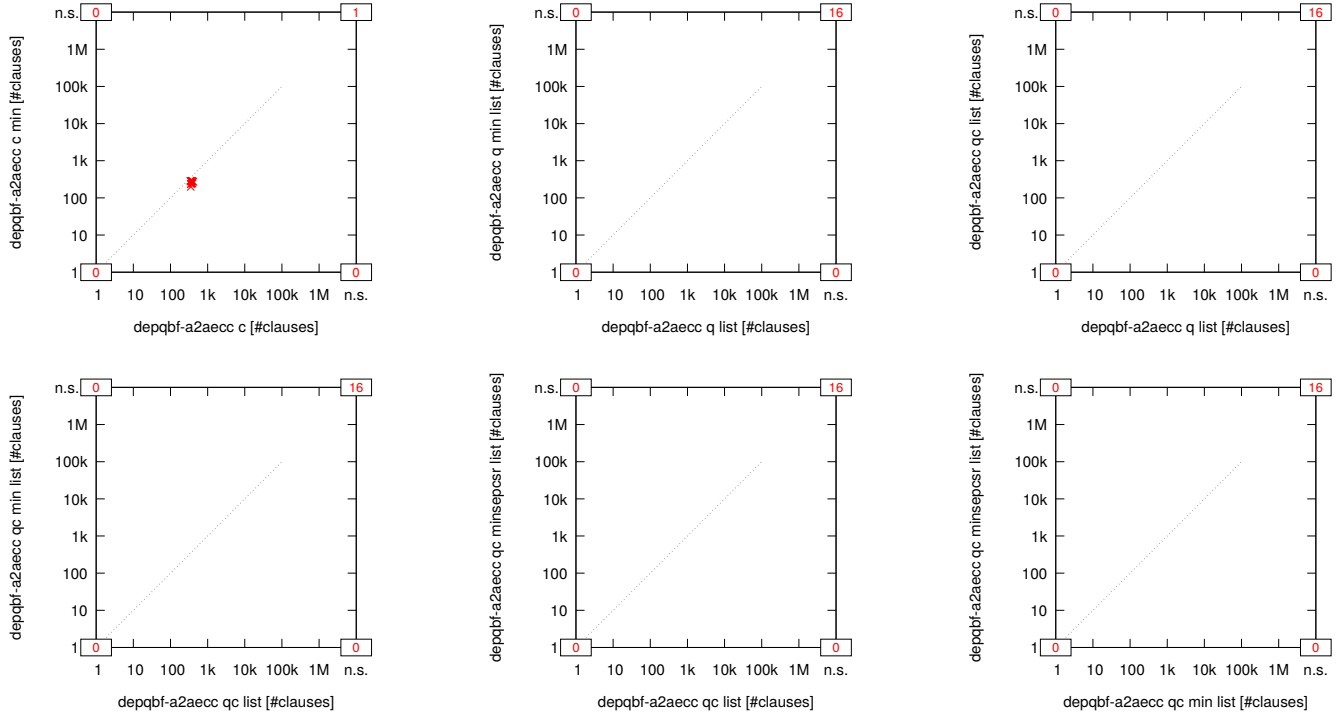


Fig. 78: Suite Chen-Interian ($n = 16$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

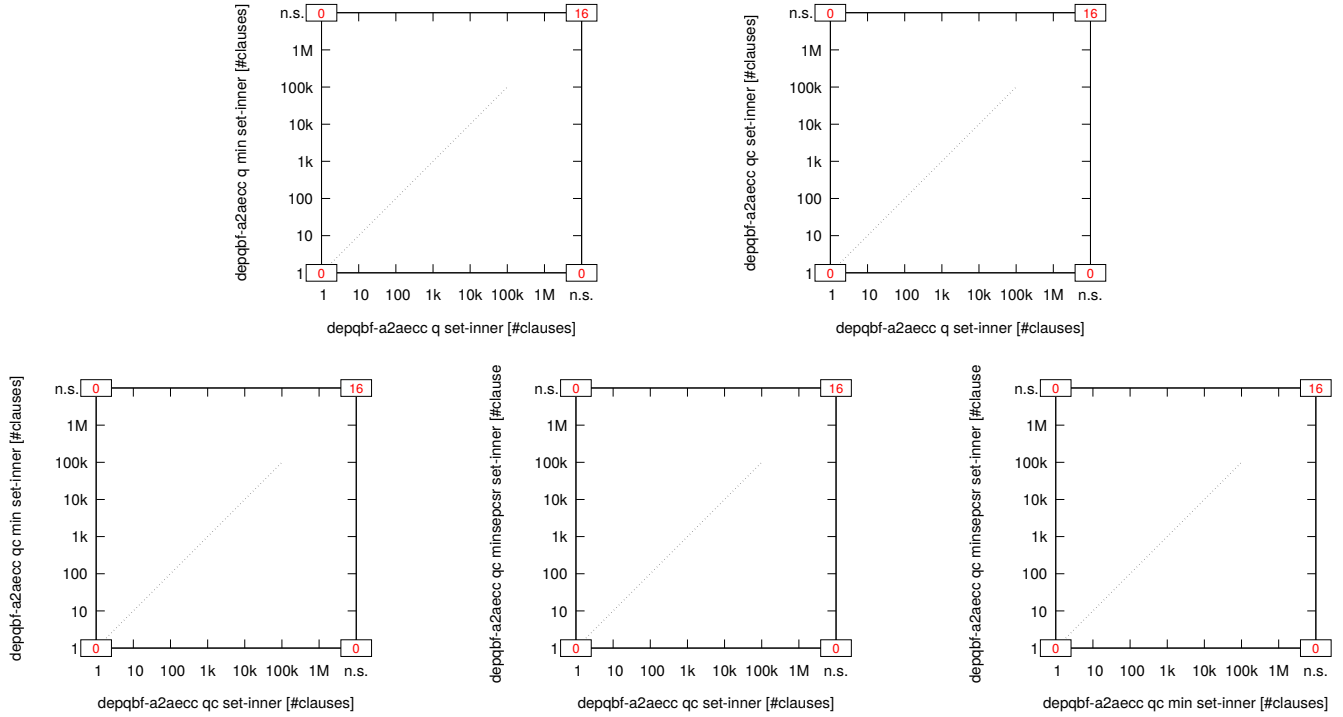


Fig. 79: Suite Chen-Interian ($n = 16$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

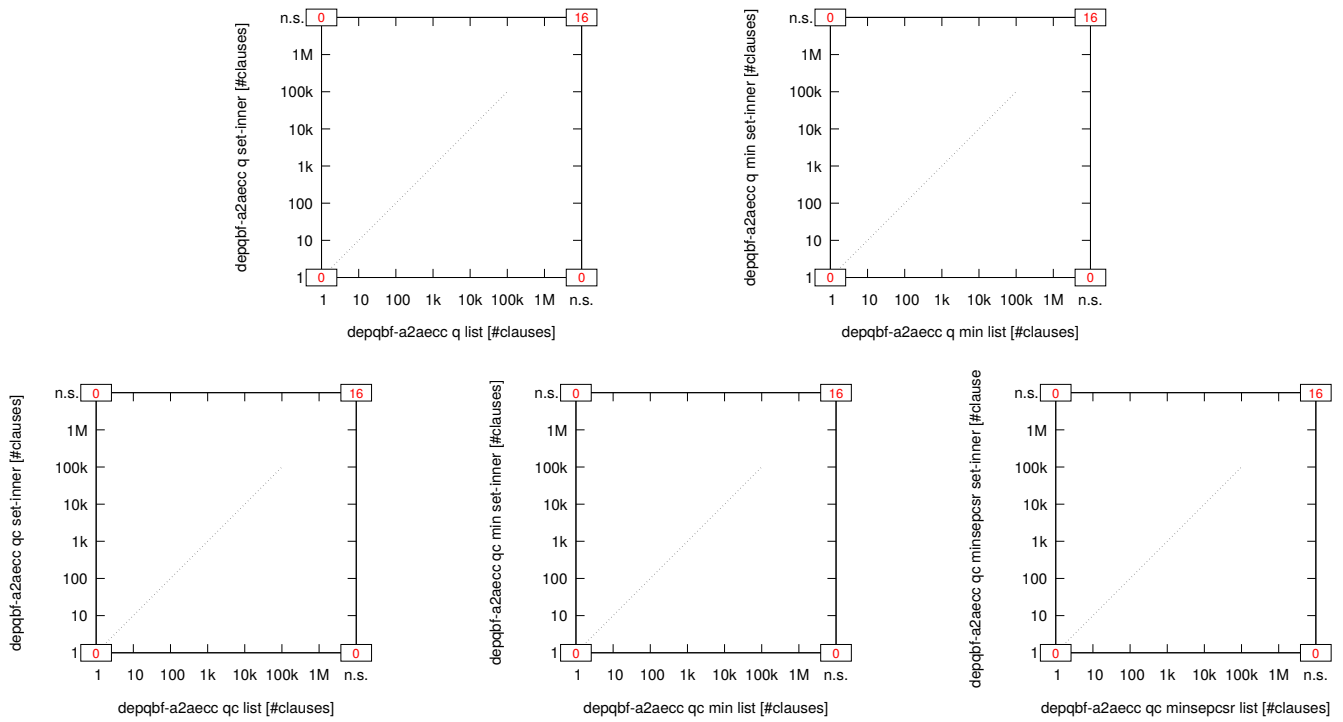


Fig. 80: Suite Chen-Interian ($n = 16$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

11) Diptarama-Jordan-Shinohara ($n = 11$):

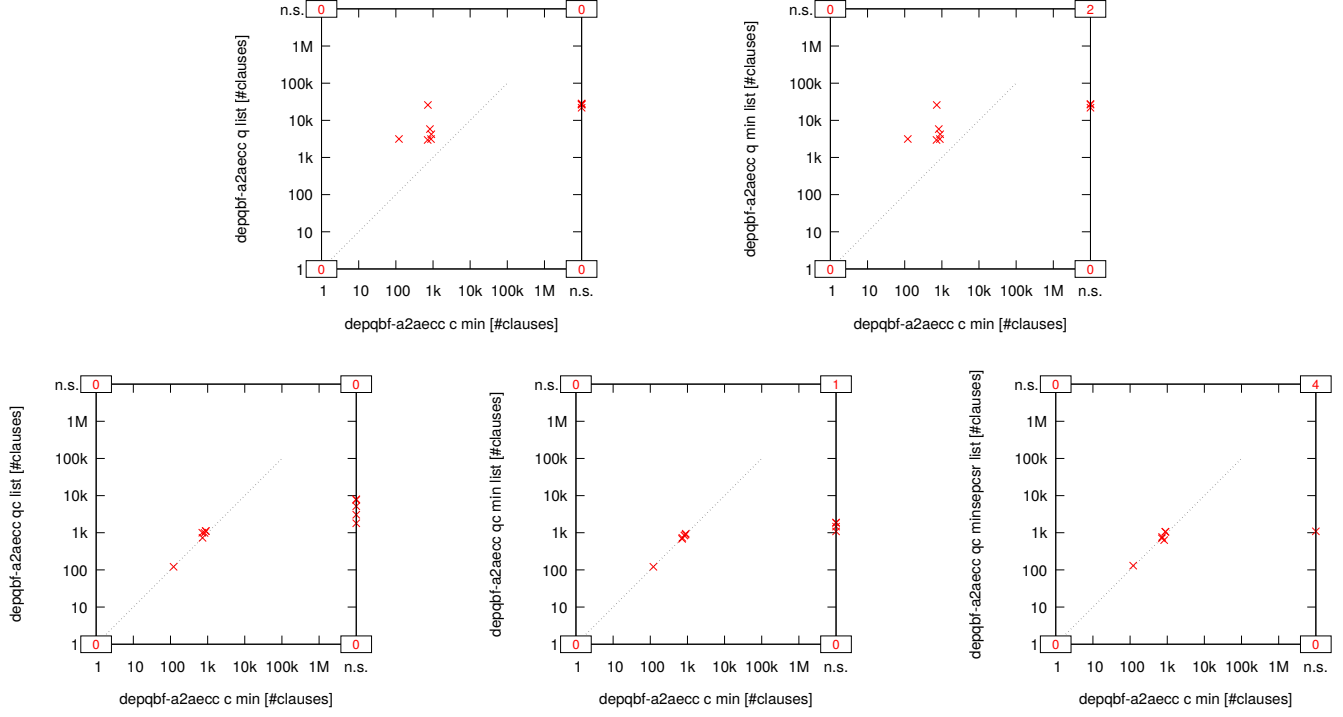


Fig. 81: Suite Diptarama-Jordan-Shinohara ($n = 11$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

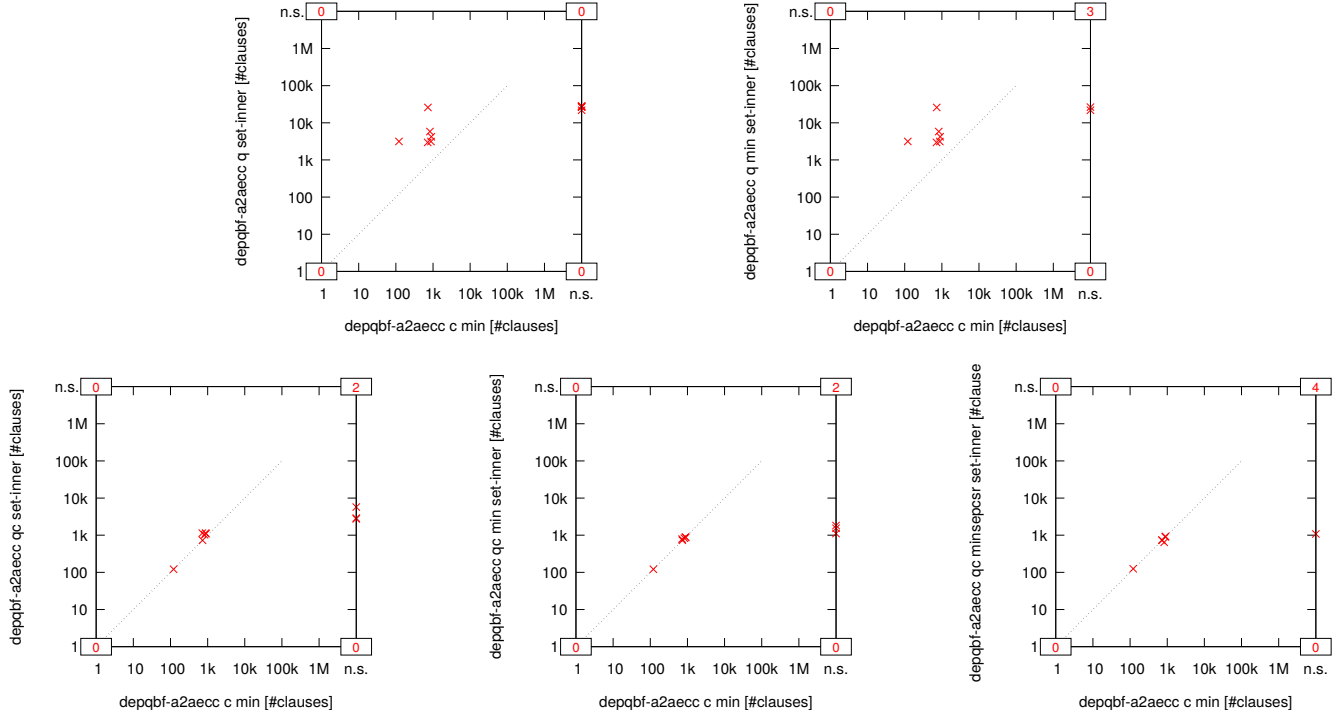


Fig. 82: Suite Diptarama-Jordan-Shinohara ($n = 11$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

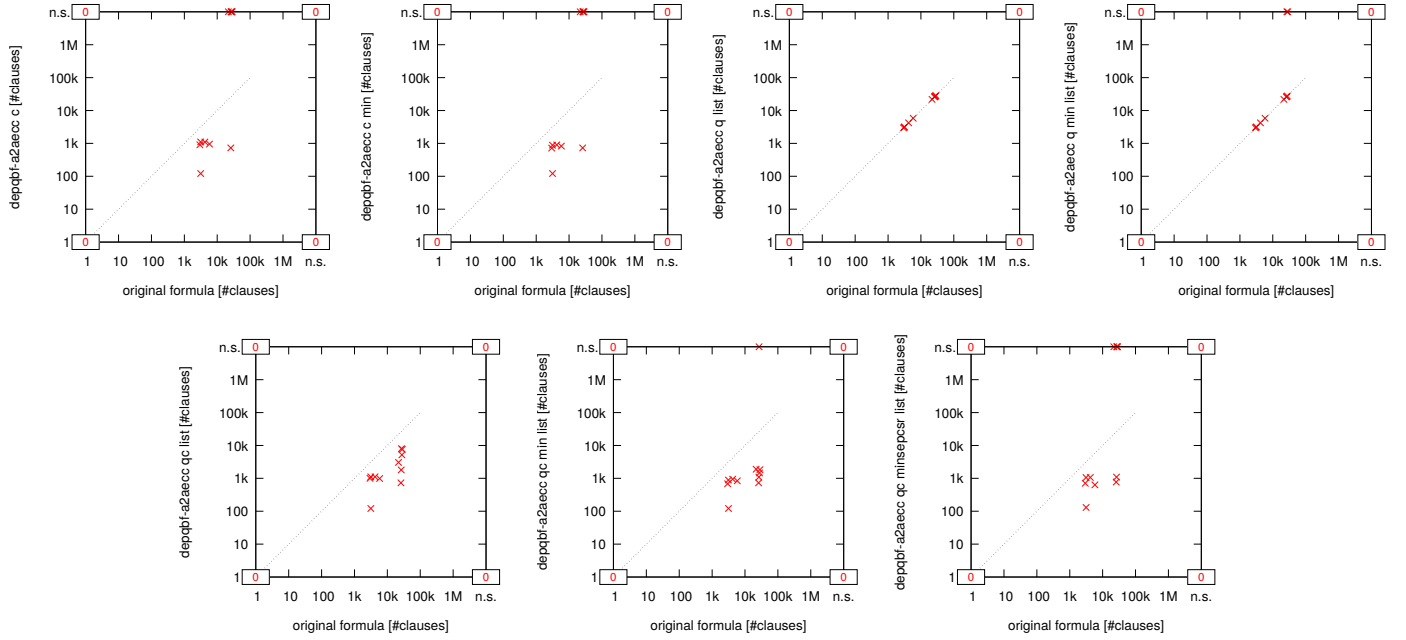


Fig. 83: Suite Diptarama-Jordan-Shinohara ($n = 11$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

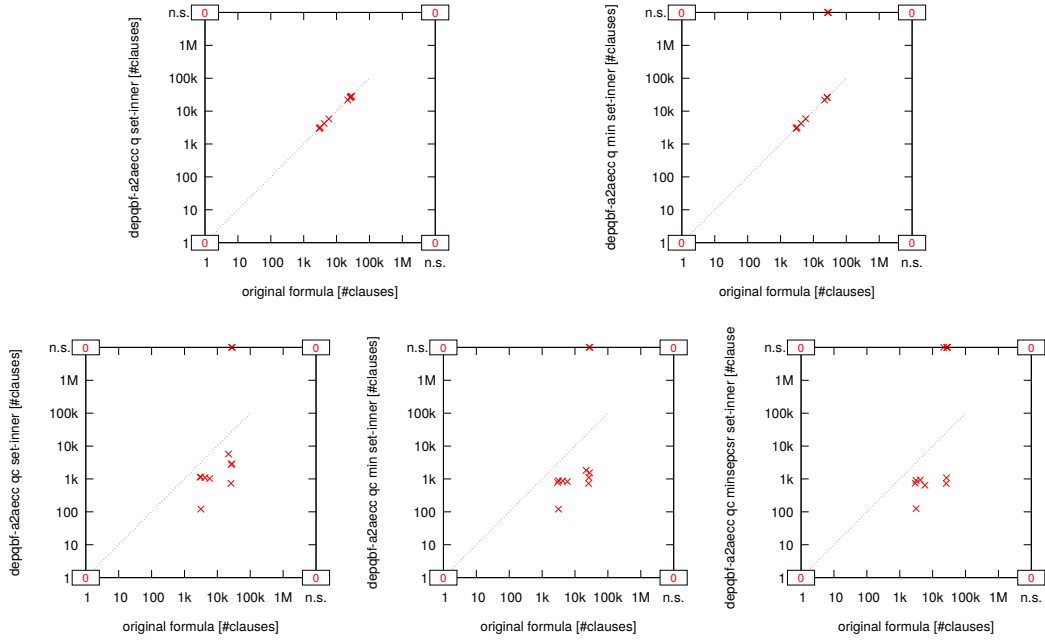


Fig. 84: Suite Diptarama-Jordan-Shinohara ($n = 11$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

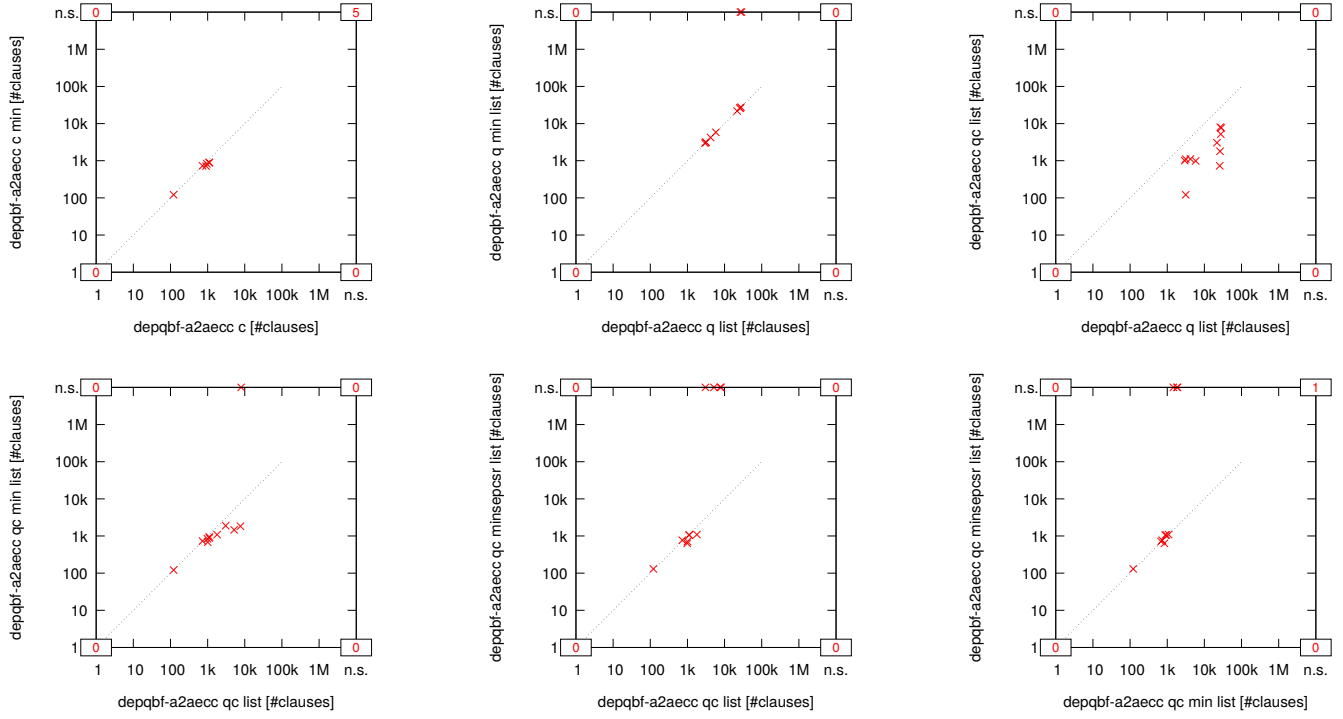


Fig. 85: Suite Diptarama-Jordan-Shinohara ($n = 11$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

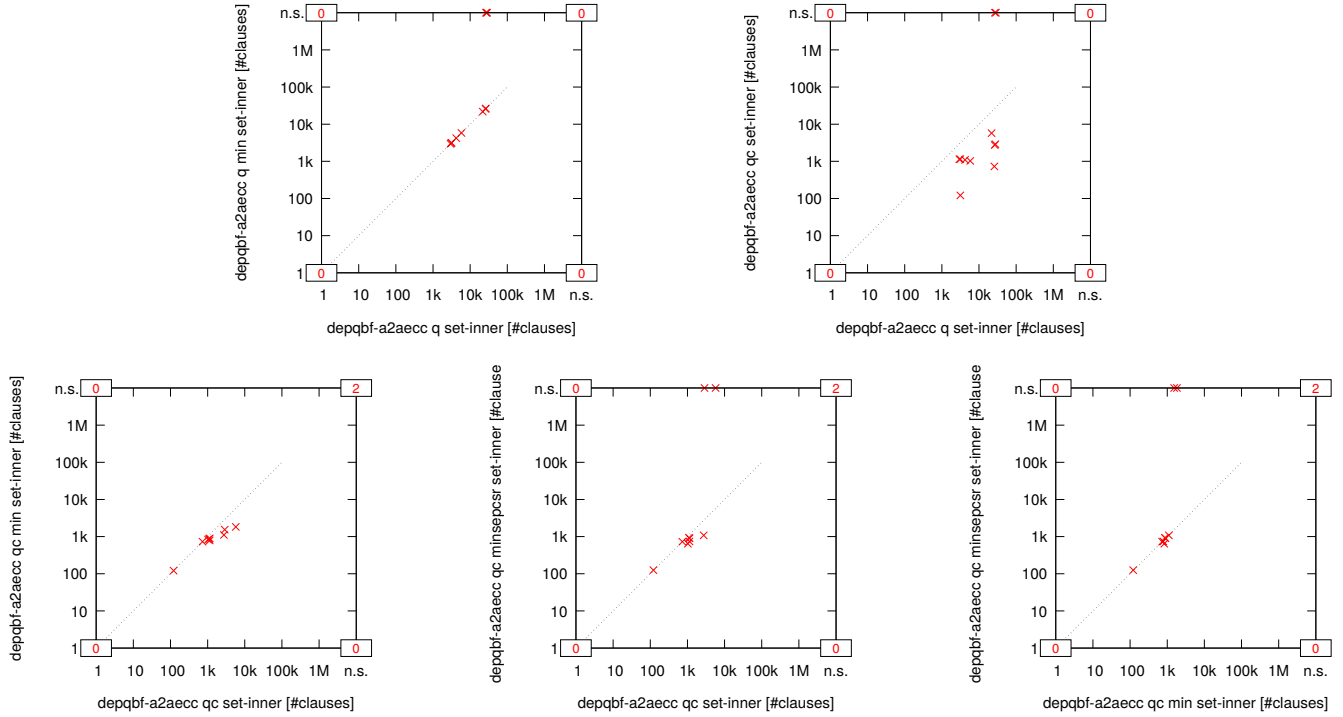


Fig. 86: Suite Diptarama-Jordan-Shinohara ($n = 11$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

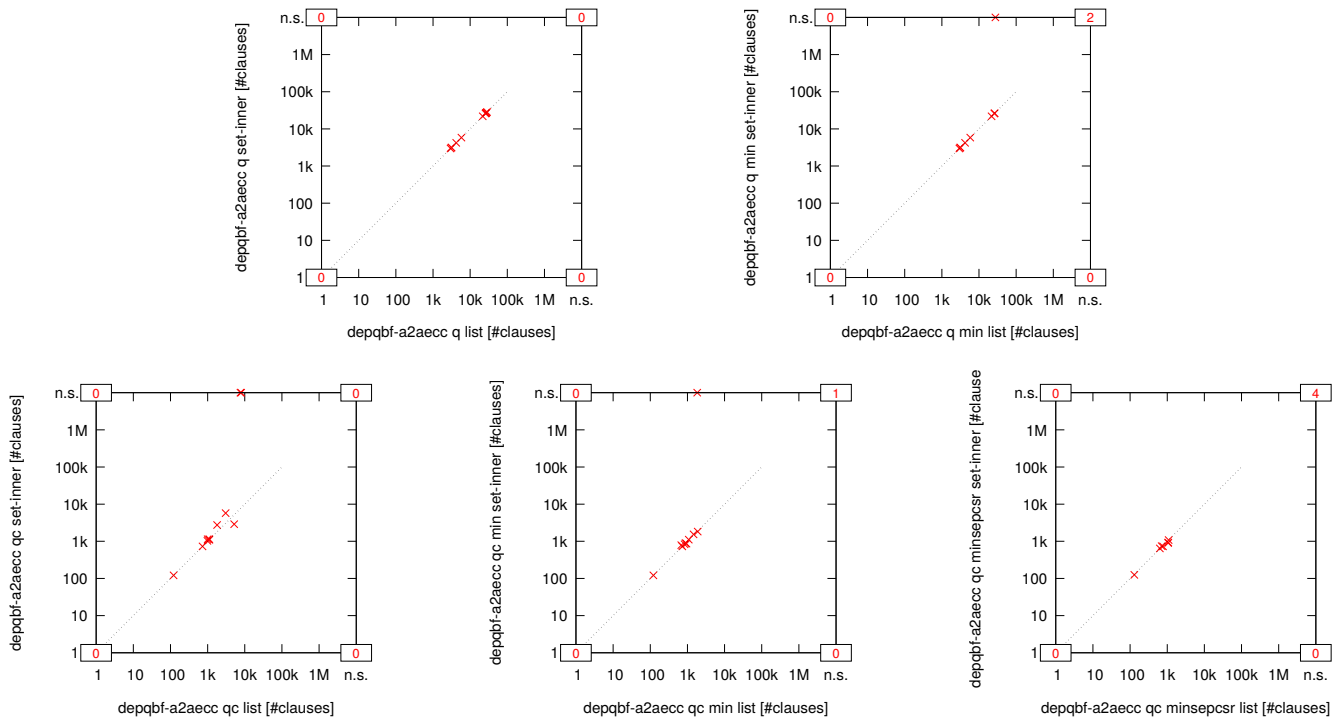


Fig. 87: Suite Diptarama-Jordan-Shinohara ($n = 11$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

12) Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$):

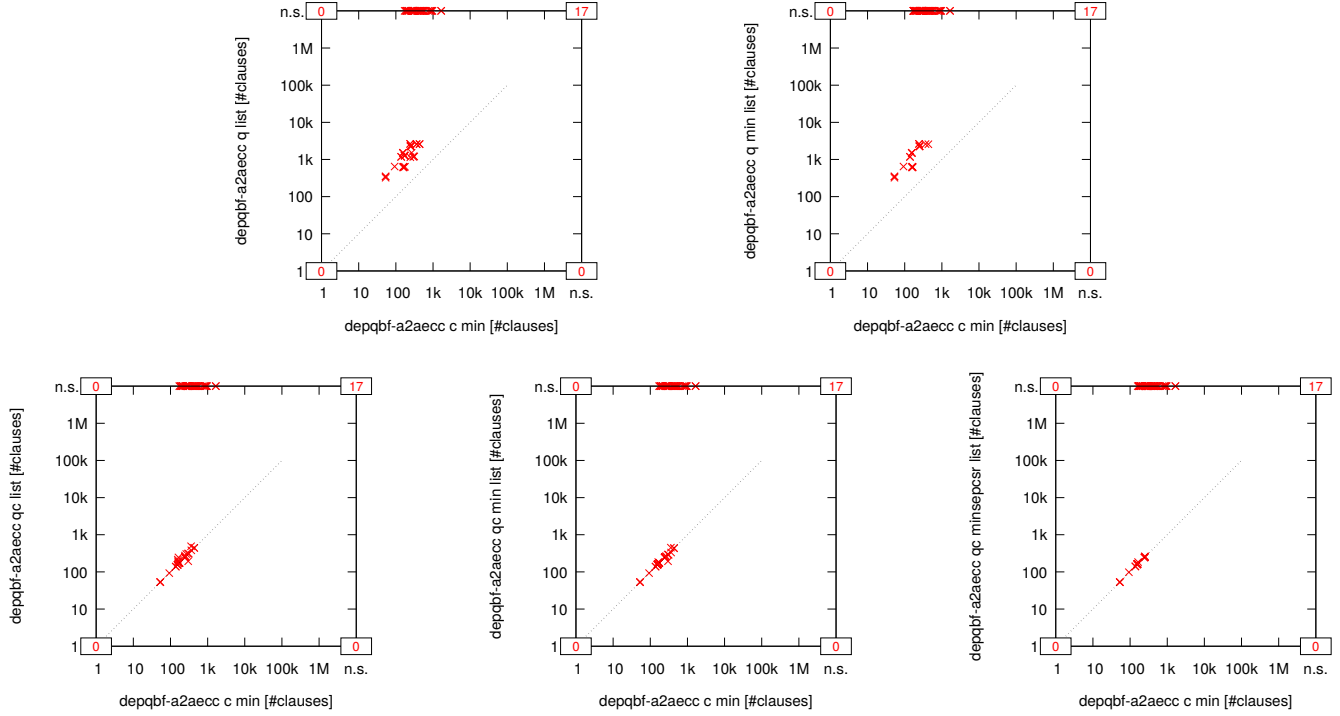


Fig. 88: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

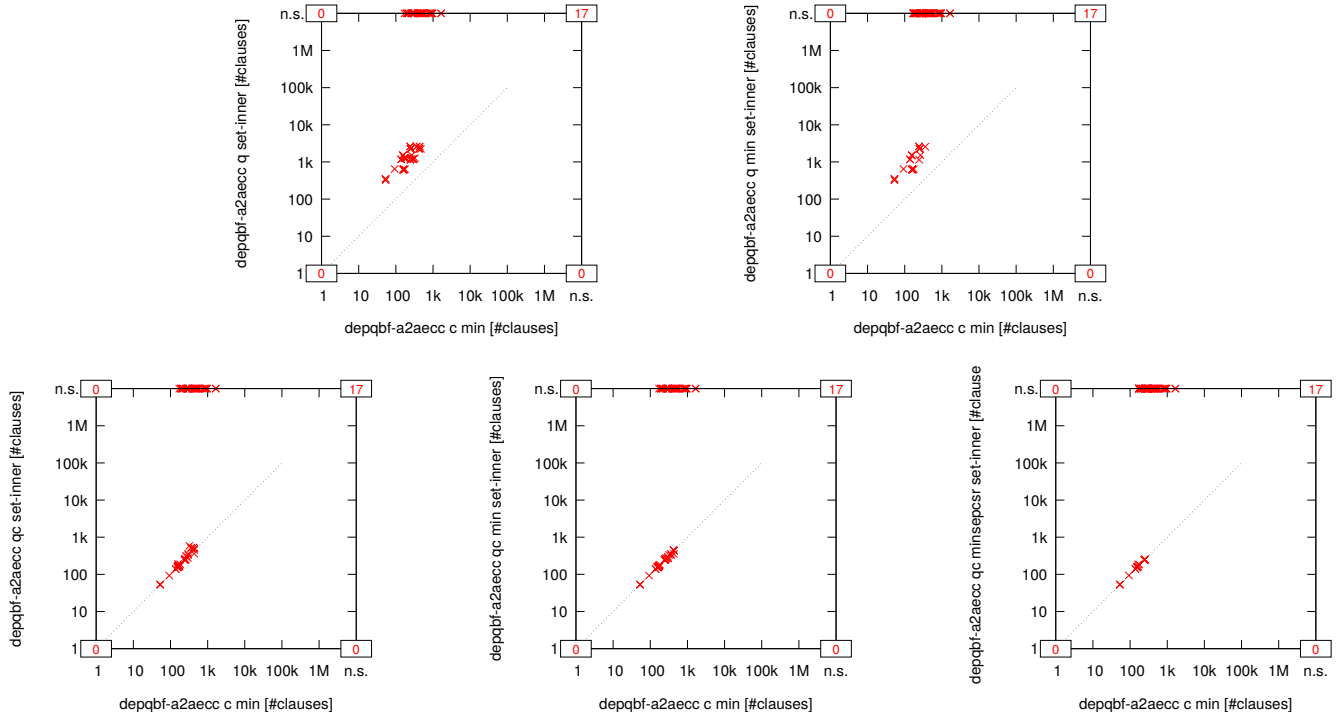


Fig. 89: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

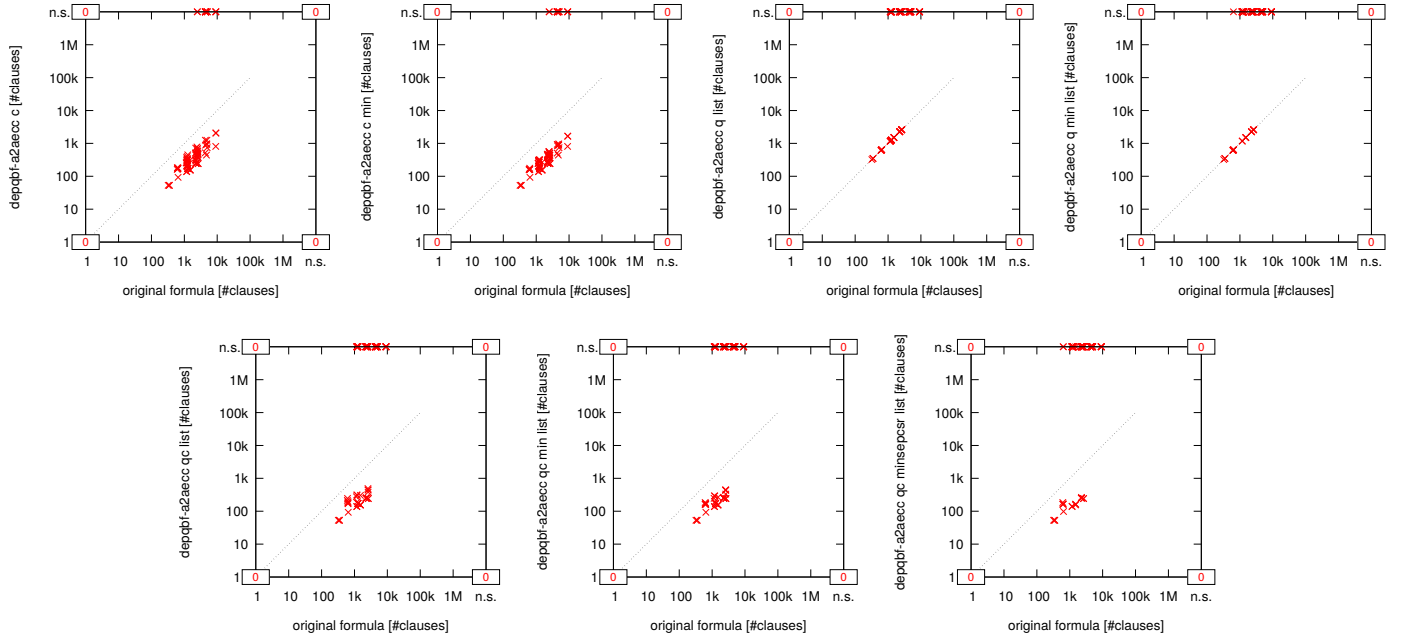


Fig. 90: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

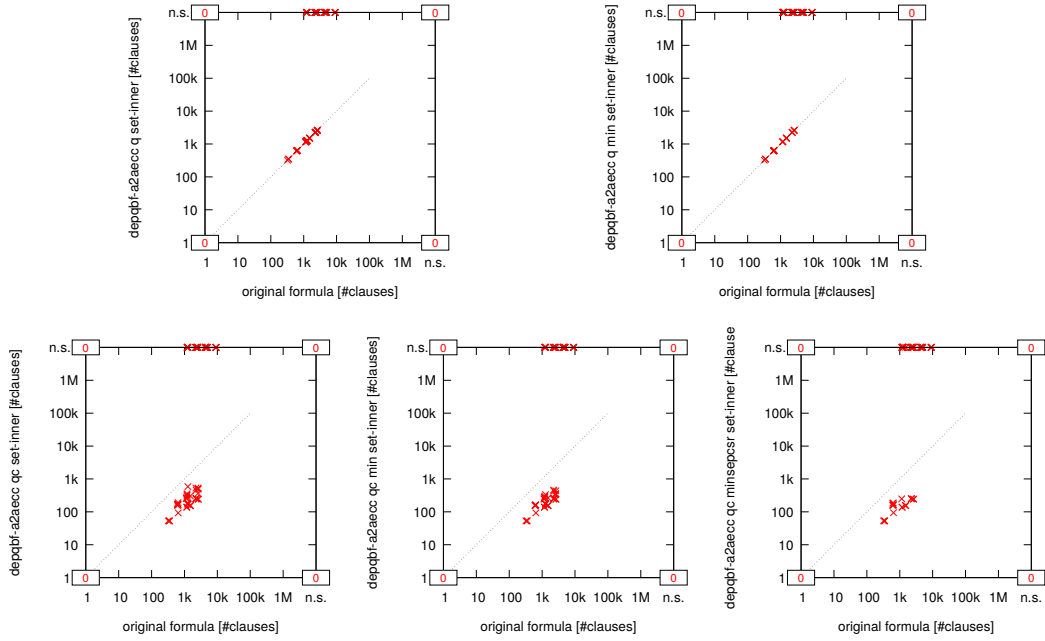


Fig. 91: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

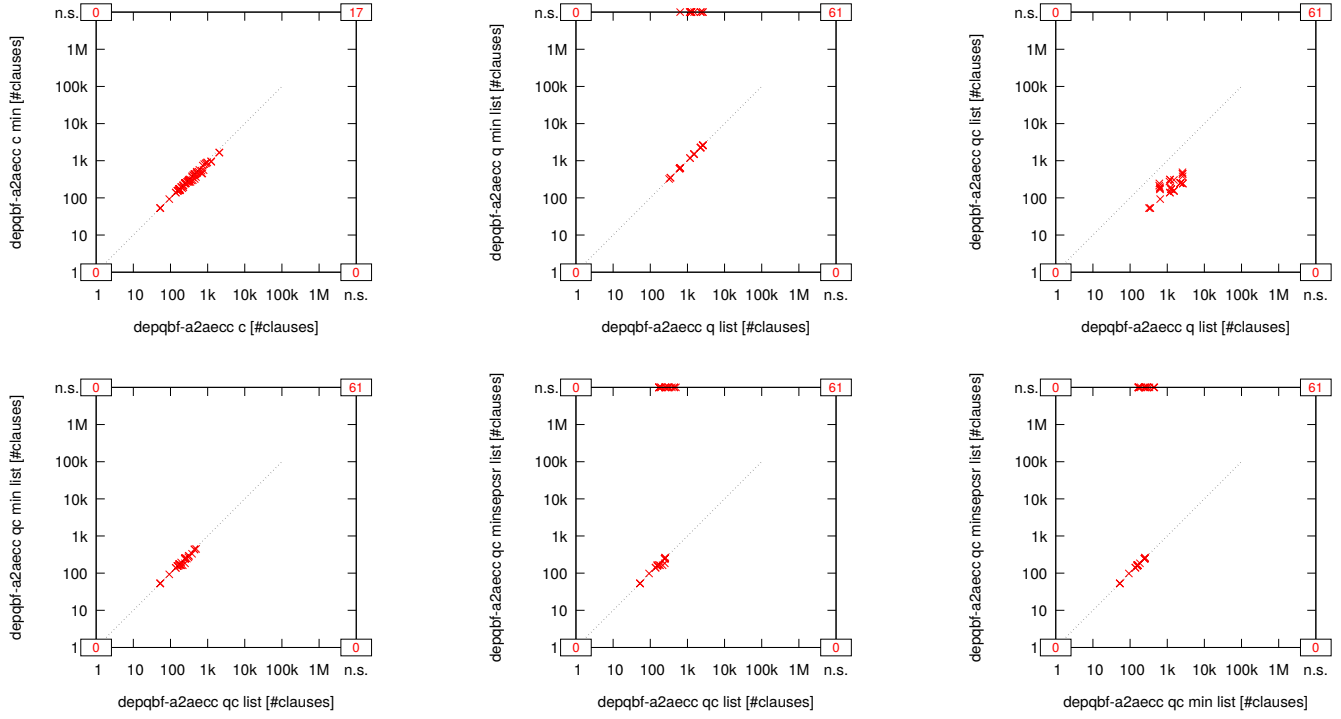


Fig. 92: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

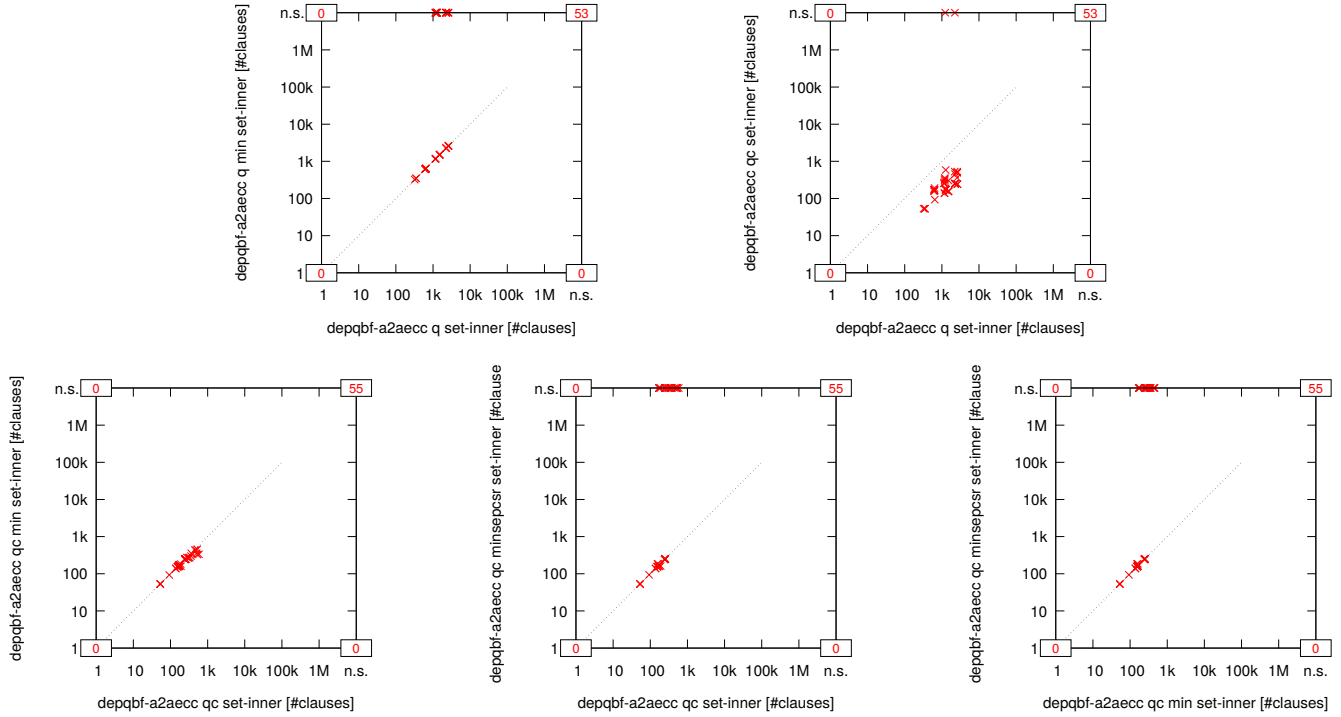


Fig. 93: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

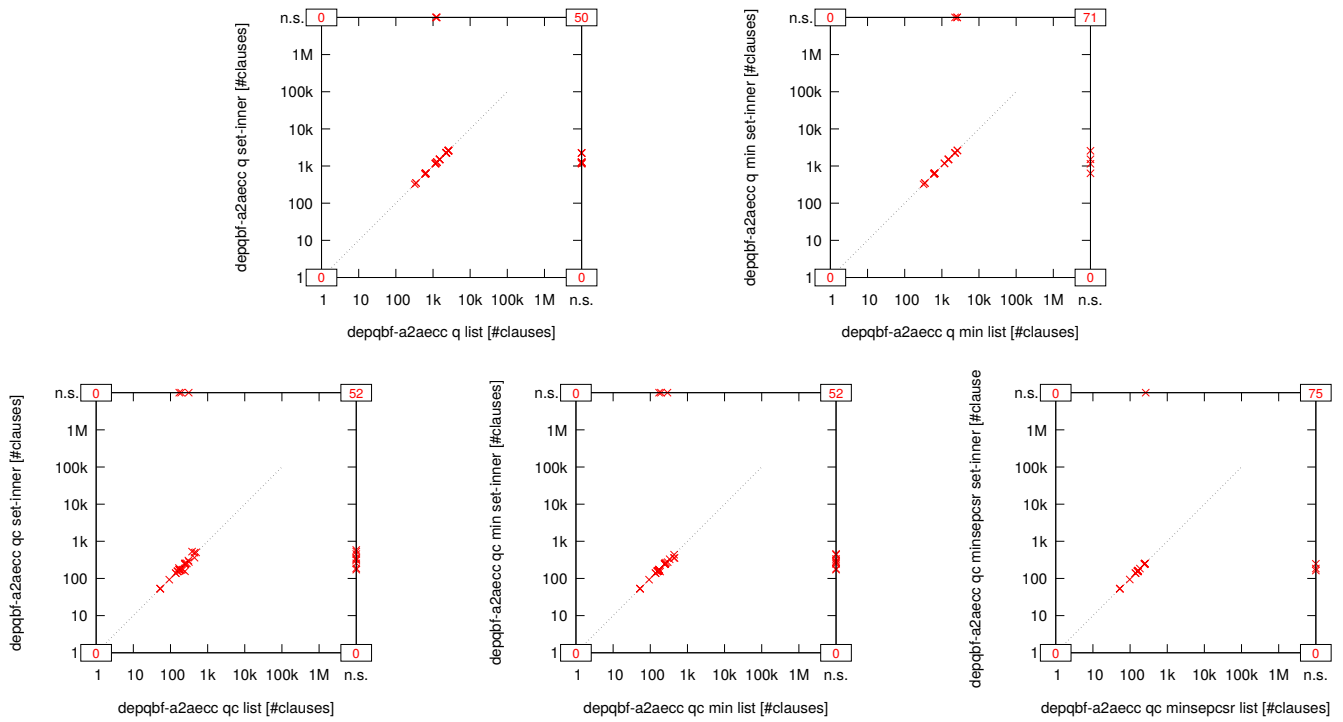


Fig. 94: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

13) *Faber-Leone-Maratea-Ricca* ($n = 128$):

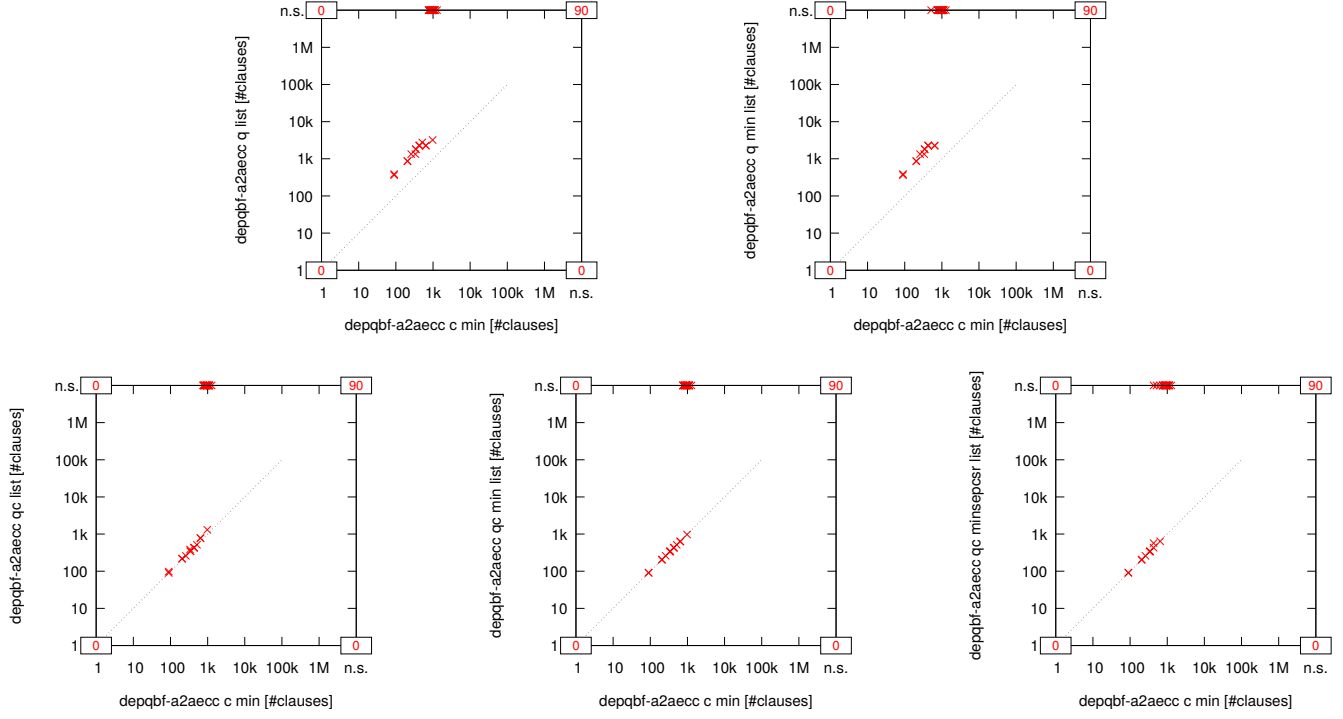


Fig. 95: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

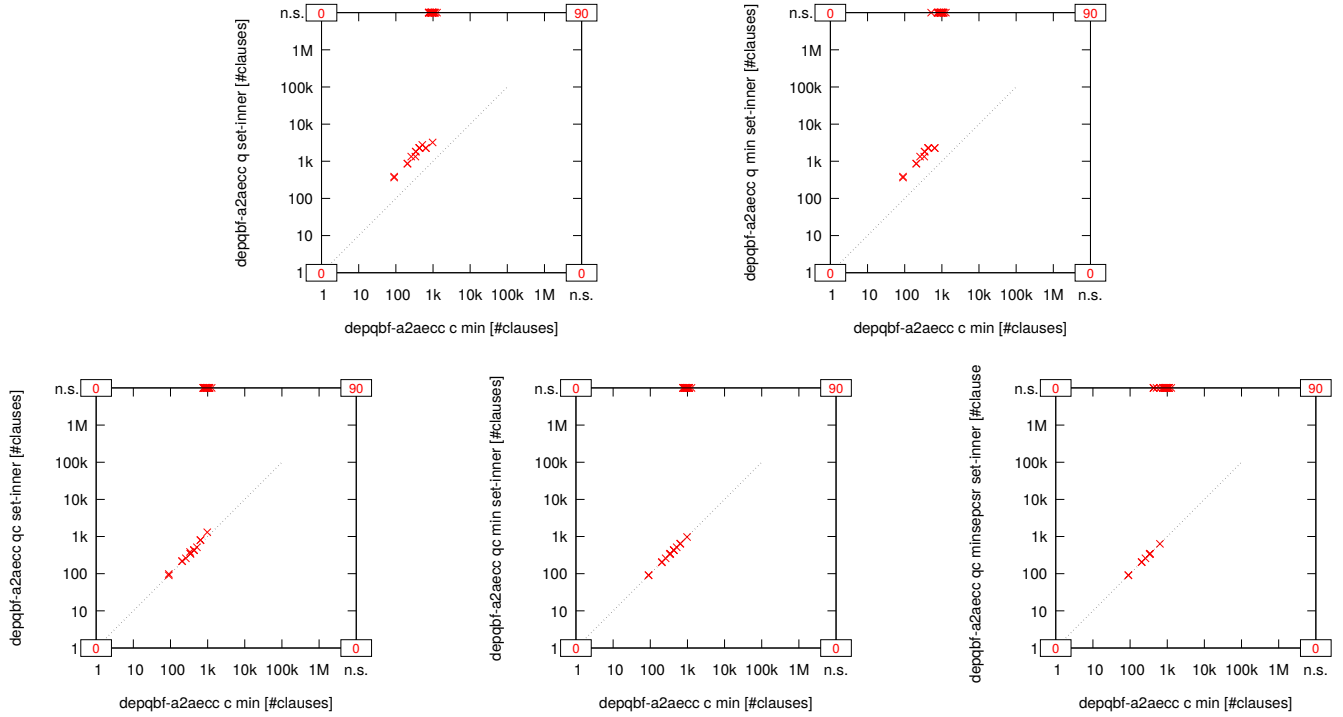


Fig. 96: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

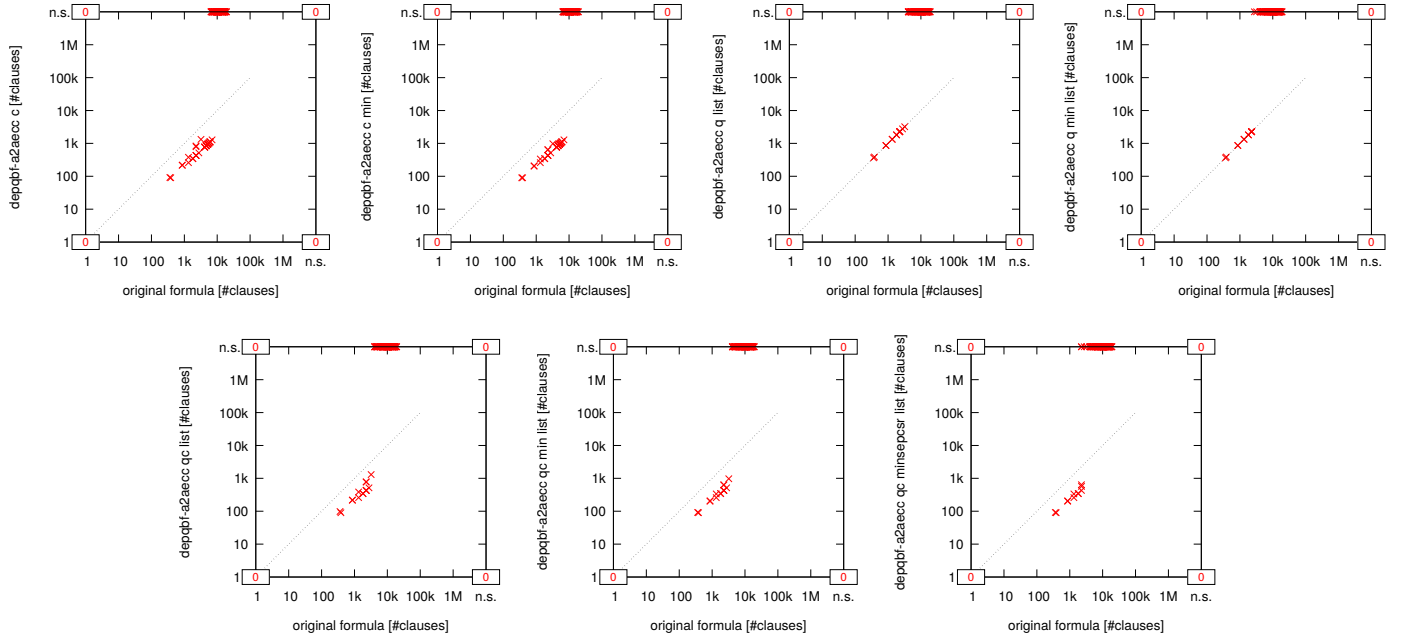


Fig. 97: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

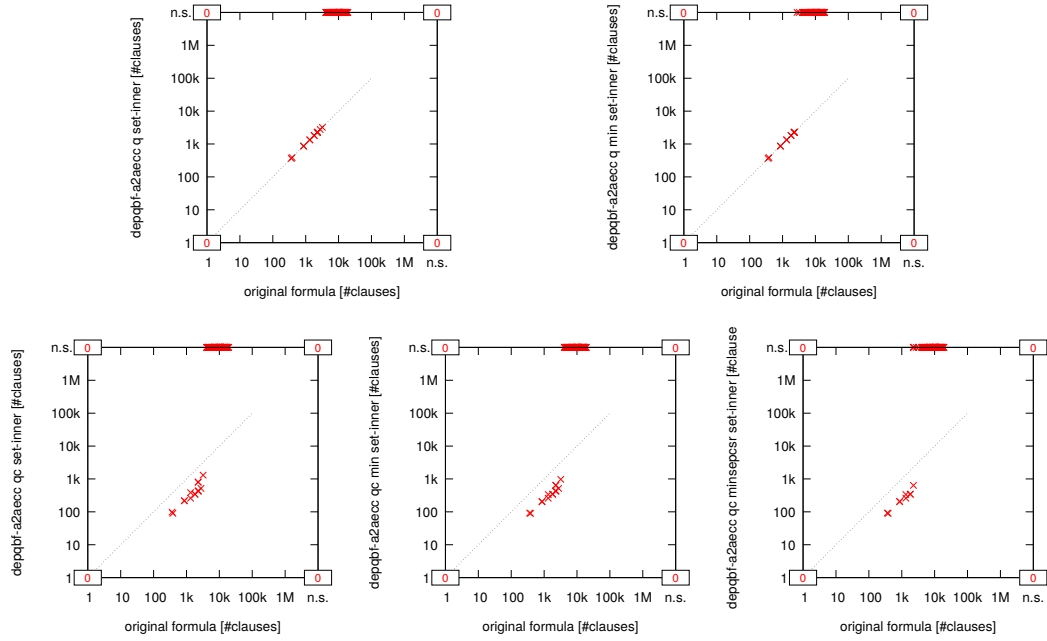


Fig. 98: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

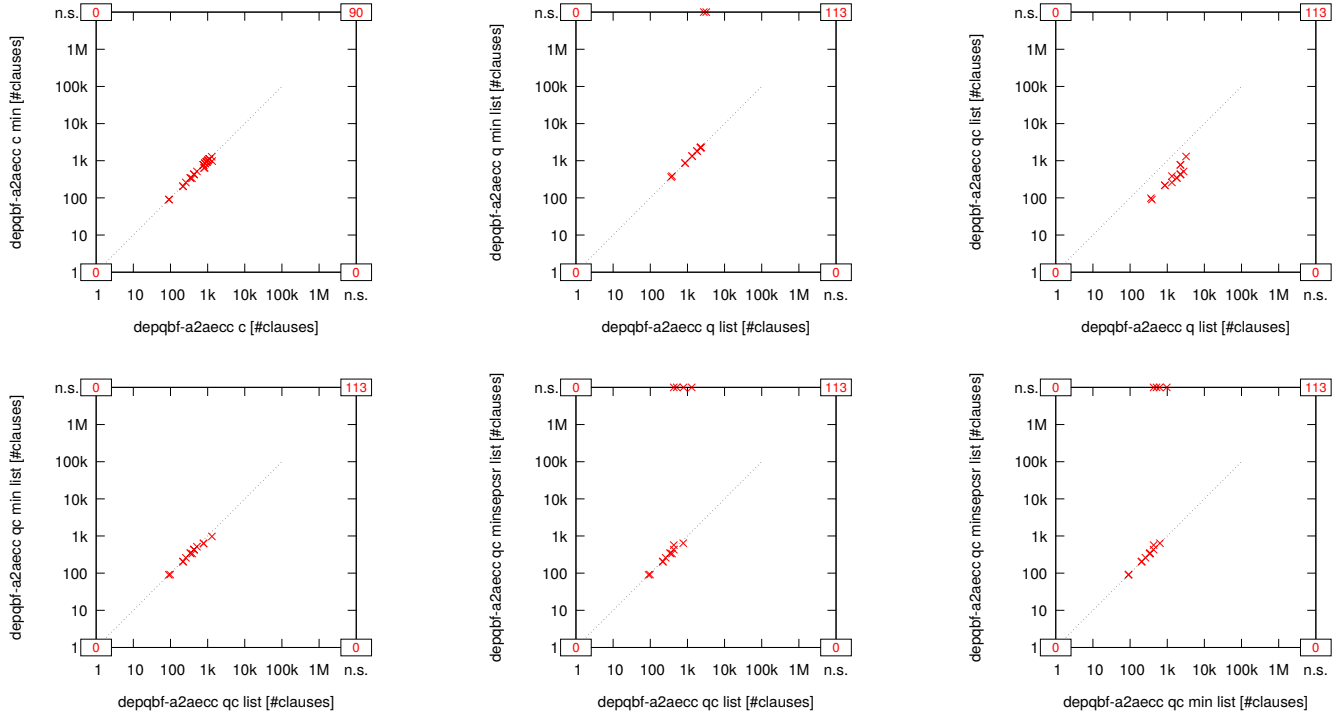


Fig. 99: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

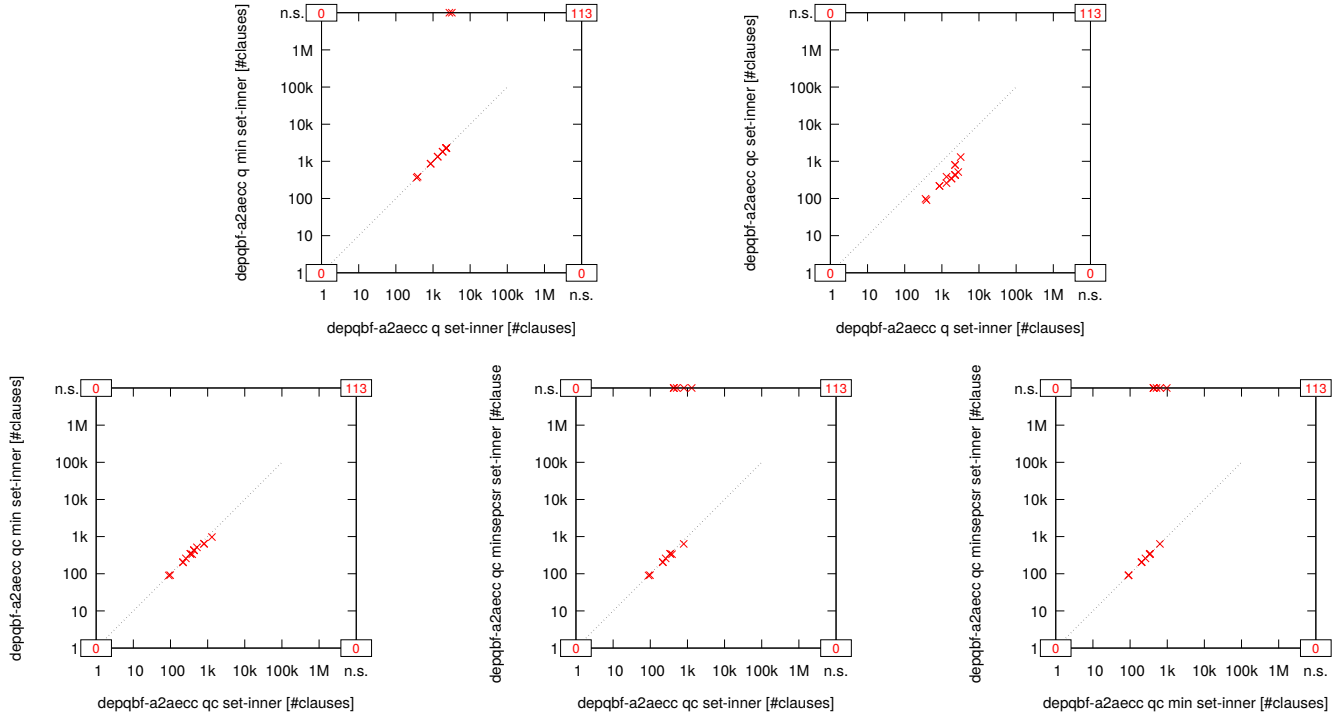


Fig. 100: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

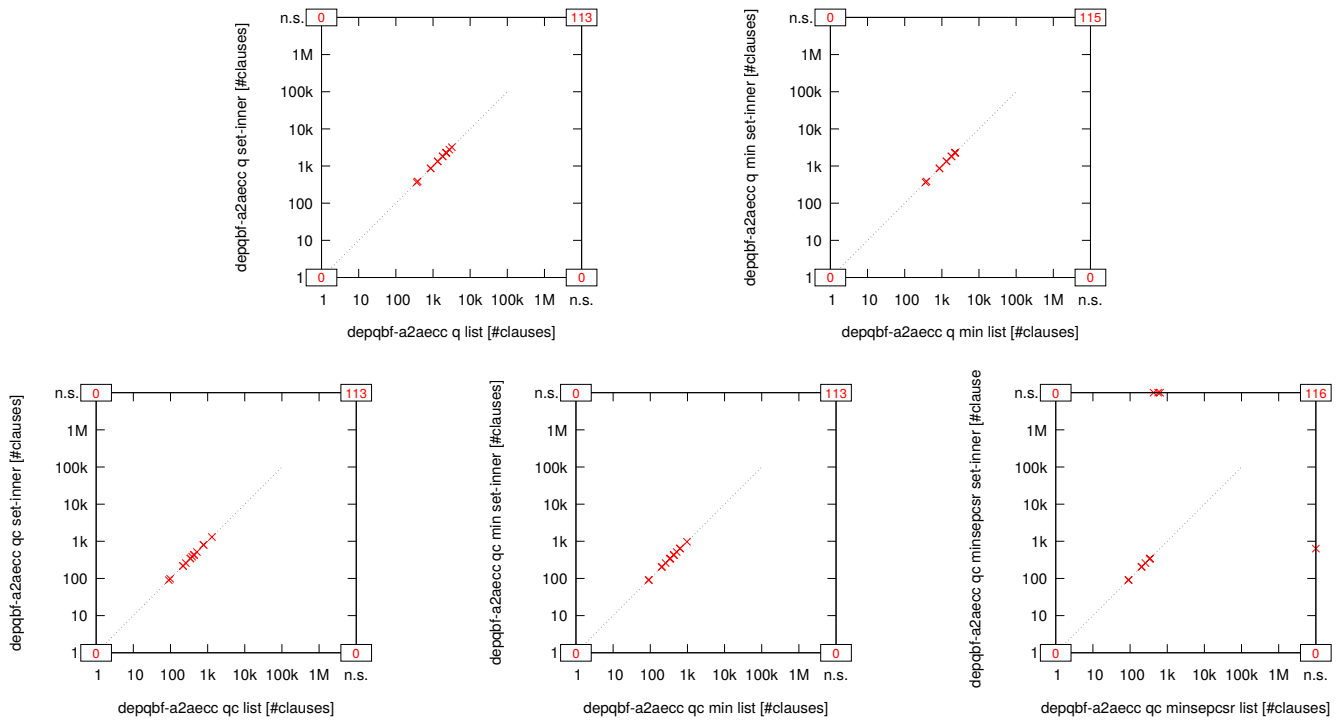


Fig. 101: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

14) Gent-Rowley ($n = 142$):

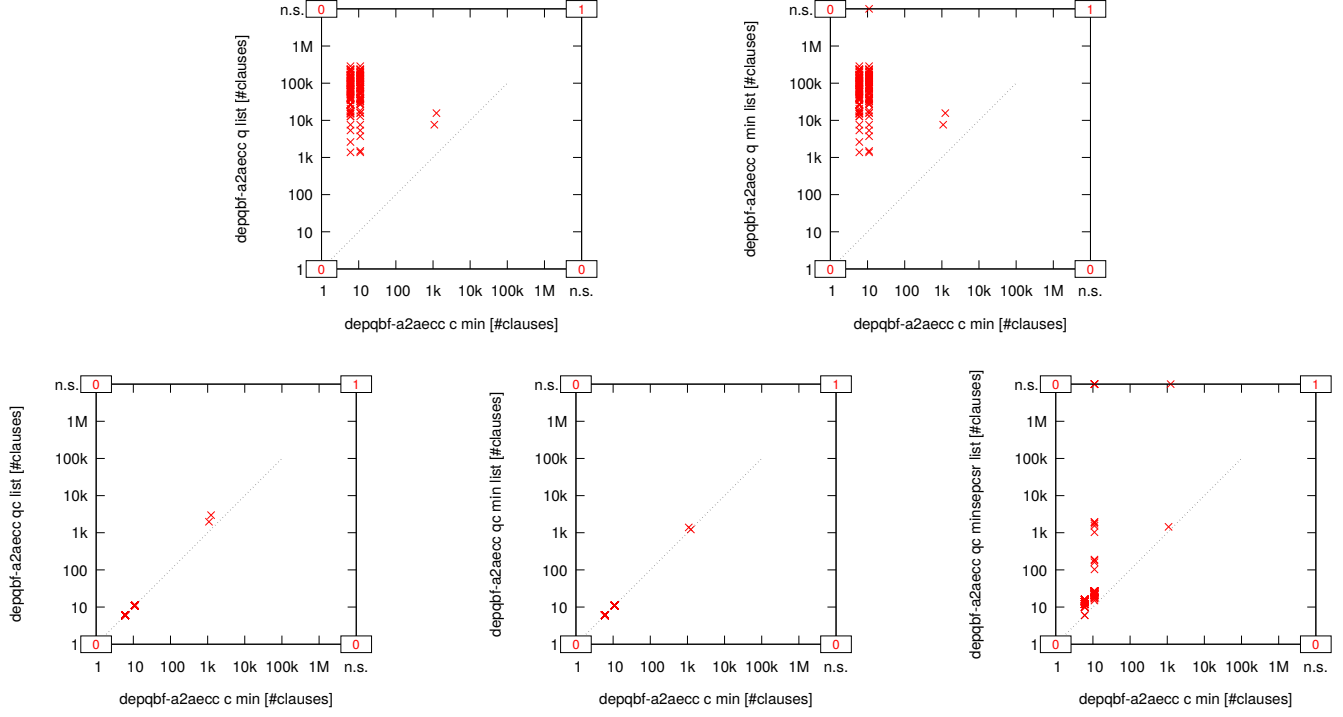


Fig. 102: Suite Gent-Rowley ($n = 142$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

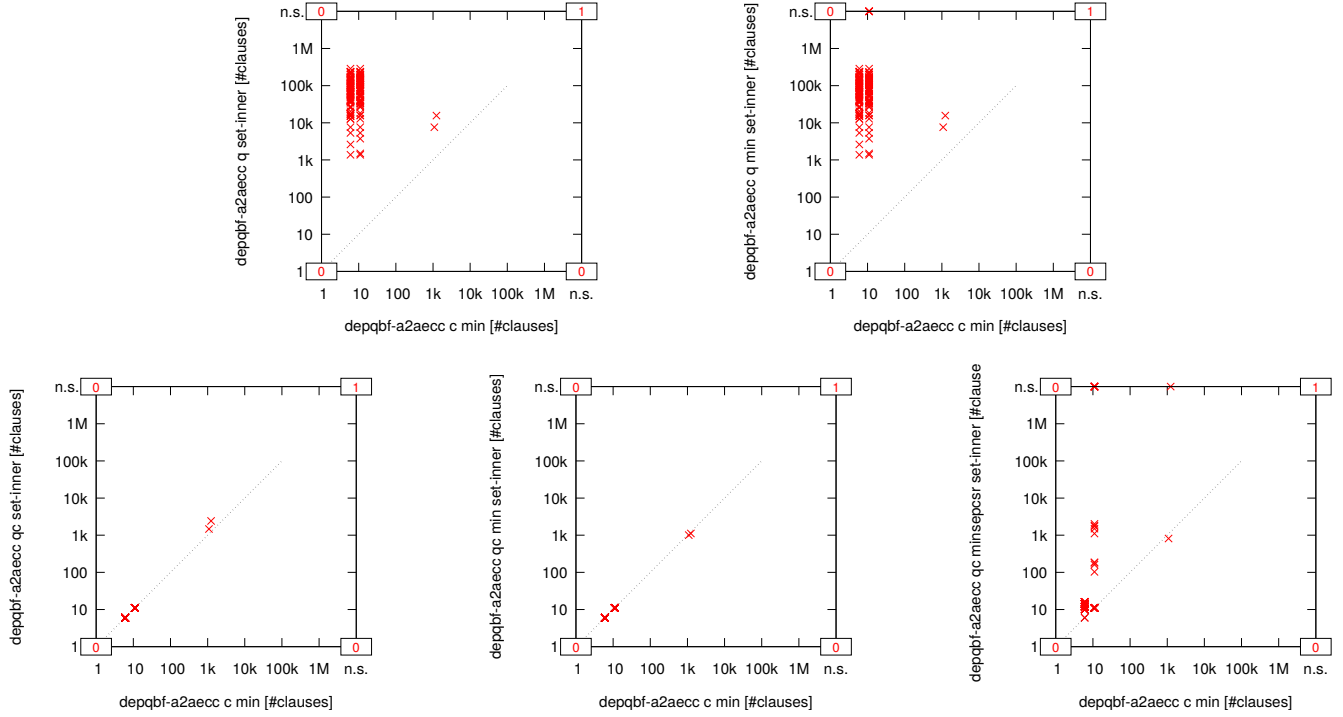


Fig. 103: Suite Gent-Rowley ($n = 142$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

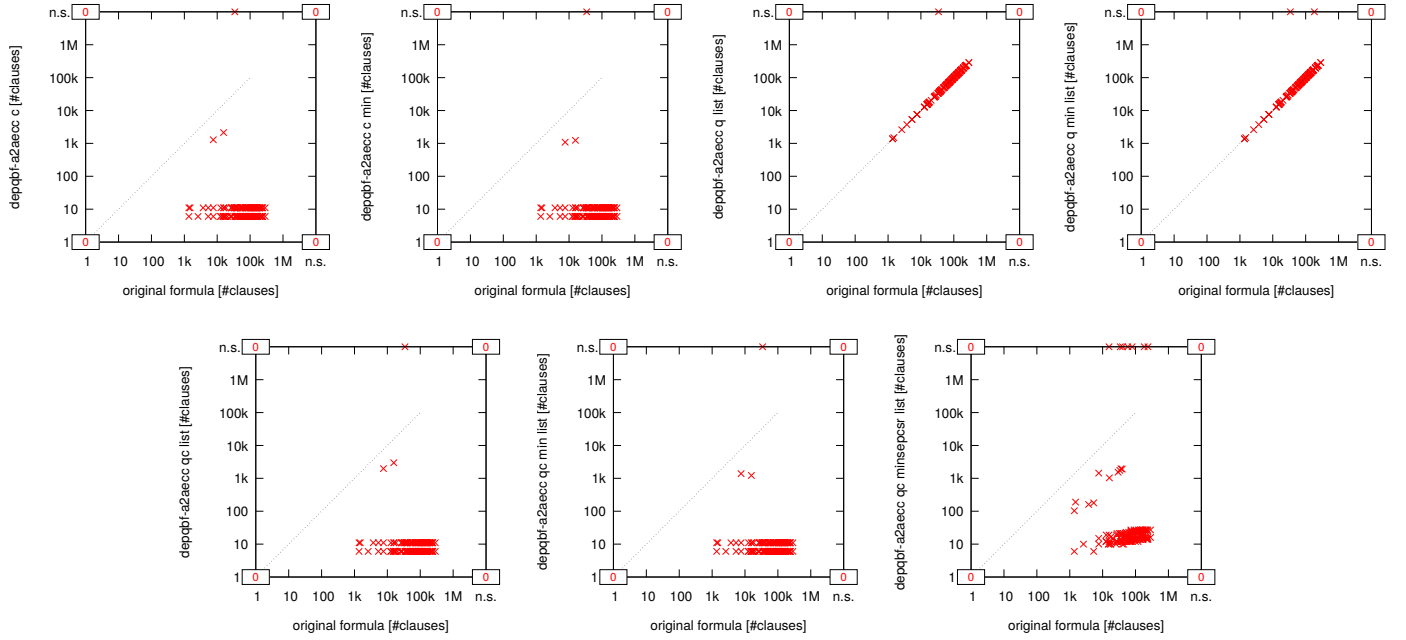


Fig. 104: Suite Gent-Rowley ($n = 142$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

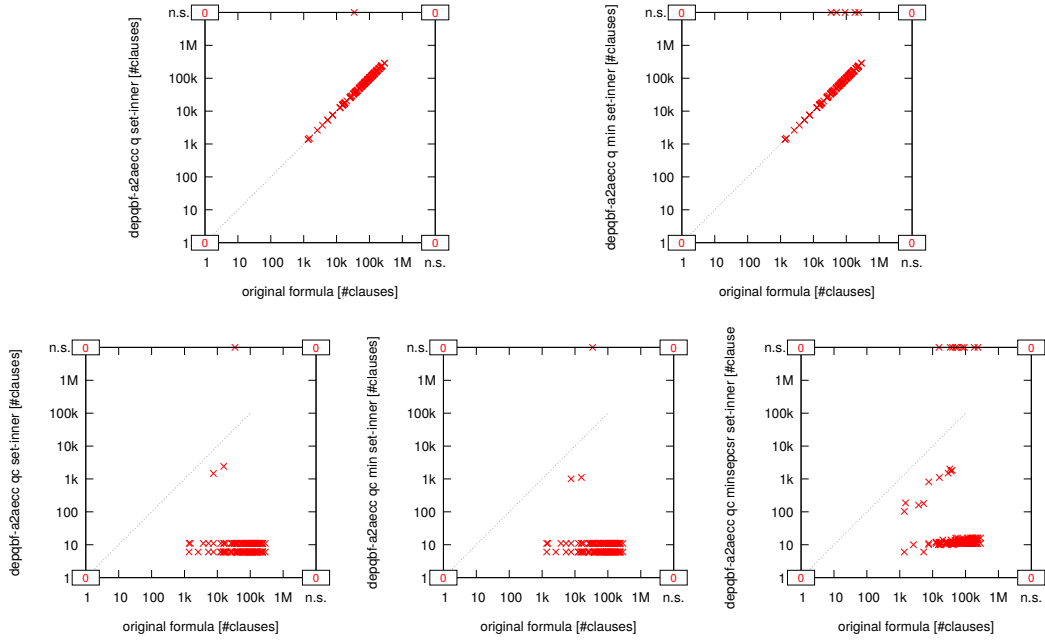


Fig. 105: Suite Gent-Rowley ($n = 142$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

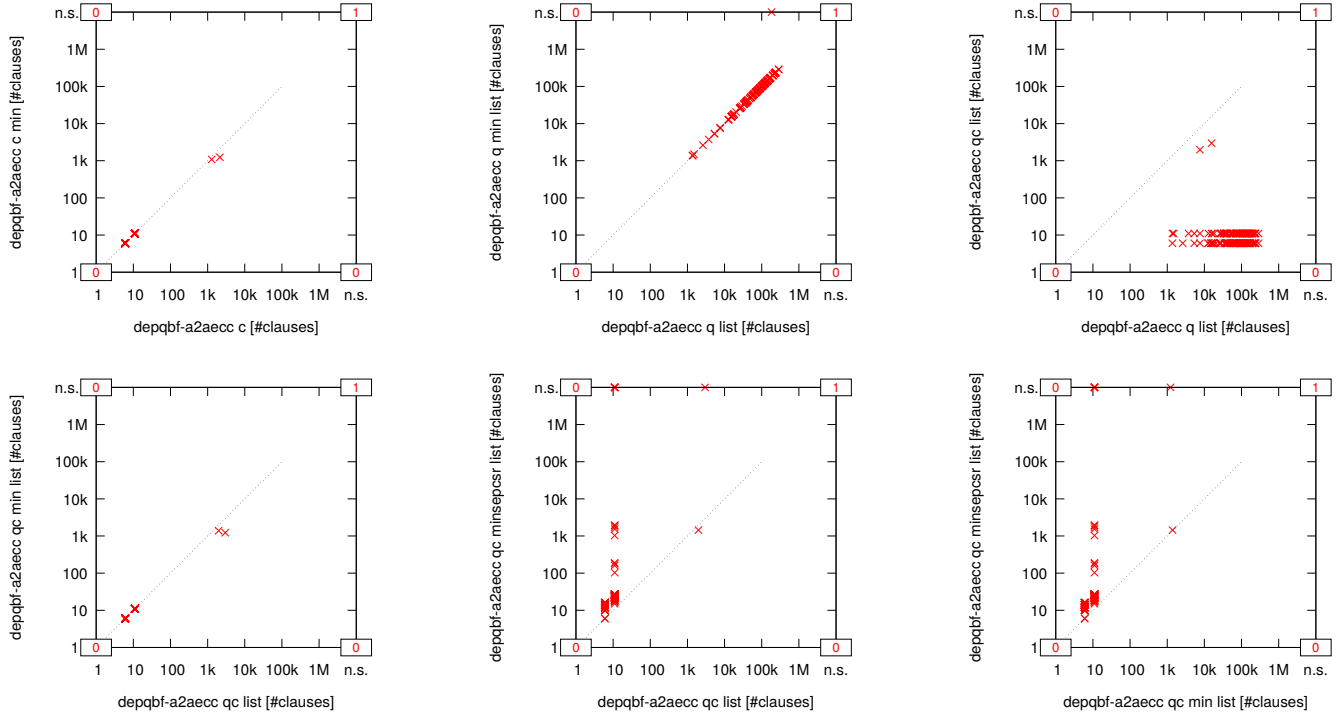


Fig. 106: Suite Gent-Rowley ($n = 142$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

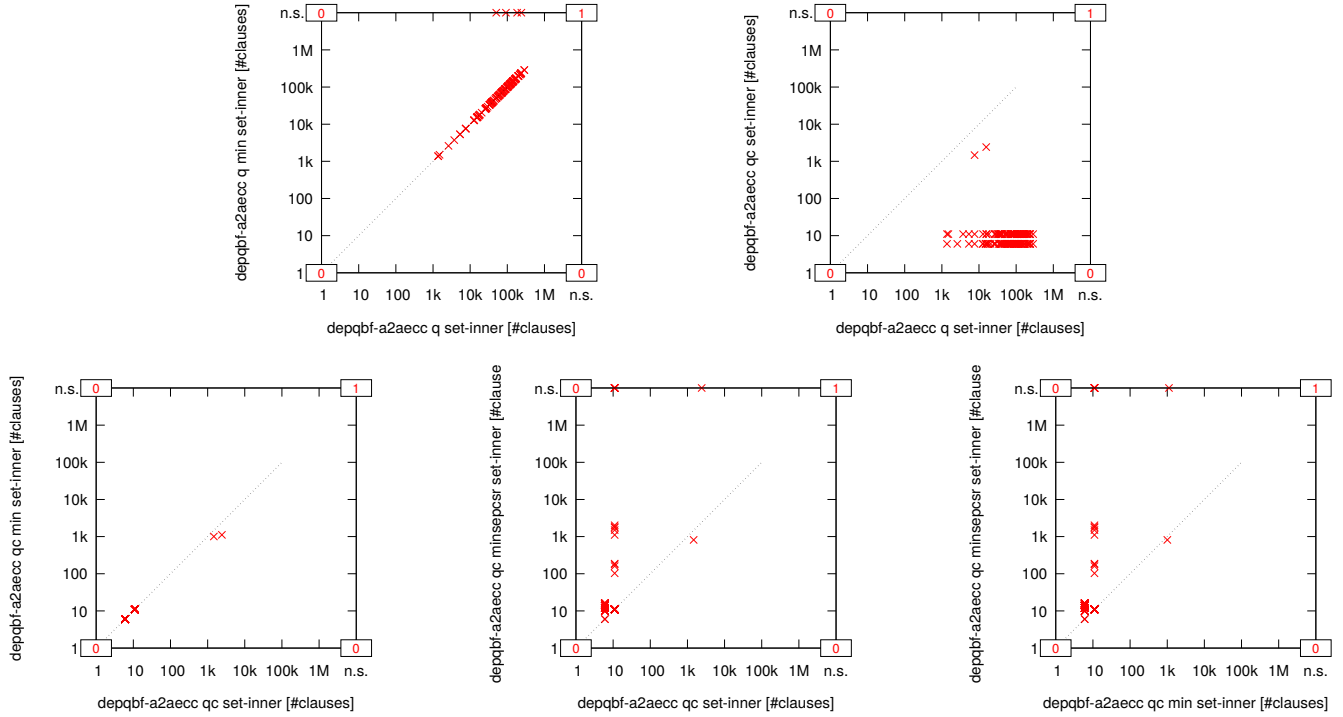


Fig. 107: Suite Gent-Rowley ($n = 142$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

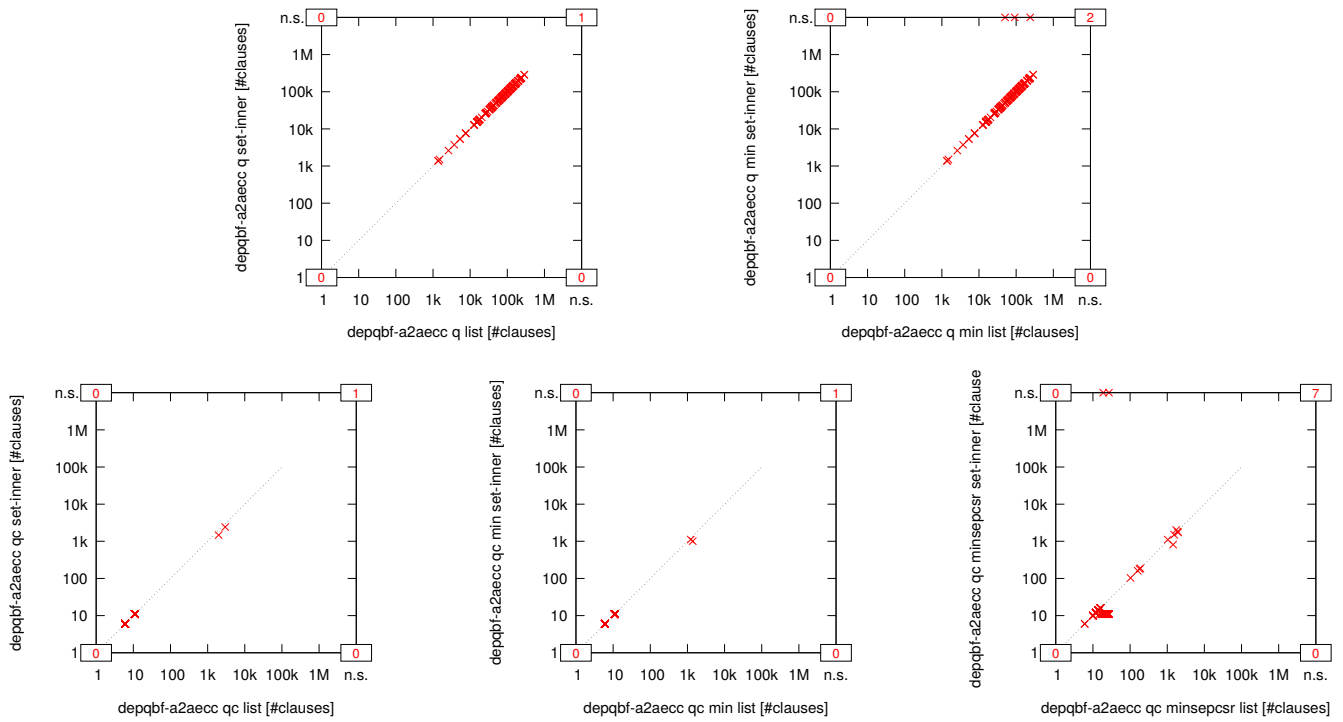


Fig. 108: Suite Gent-Rowley ($n = 142$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

15) *Herbstritt* ($n = 157$):

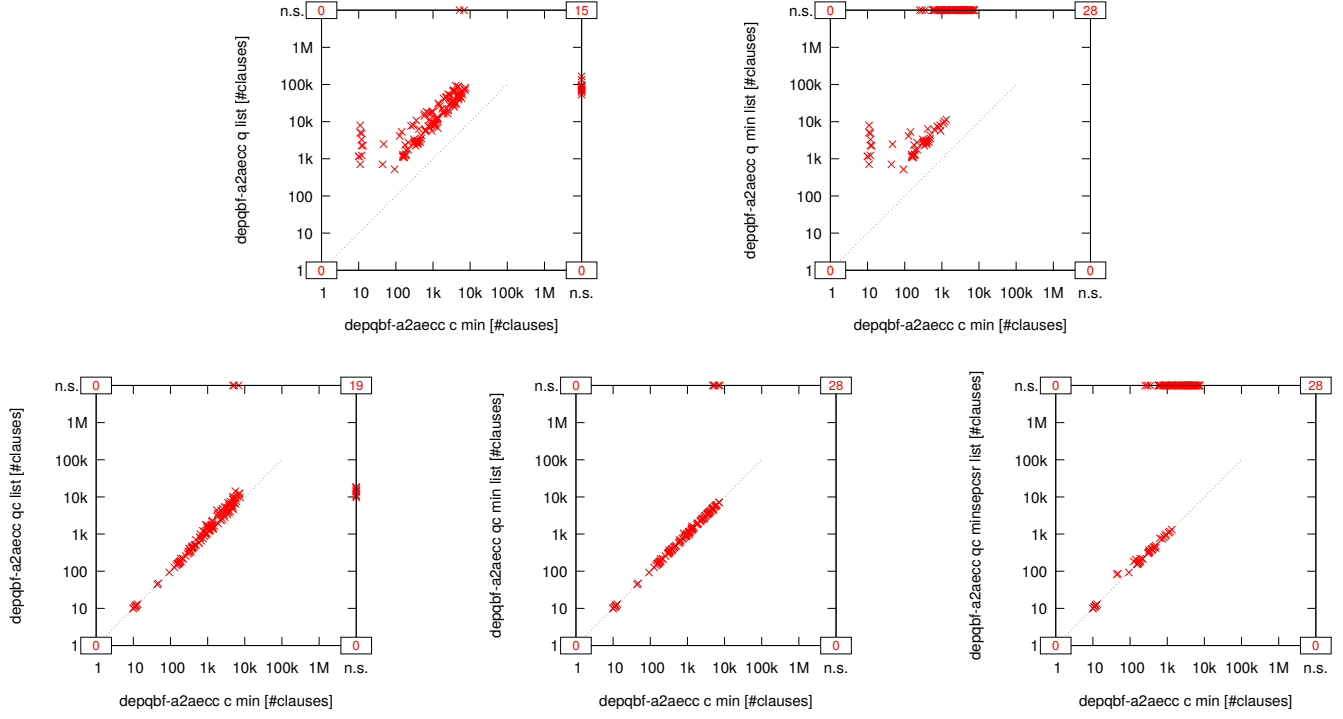


Fig. 109: Suite Herbstritt ($n = 157$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

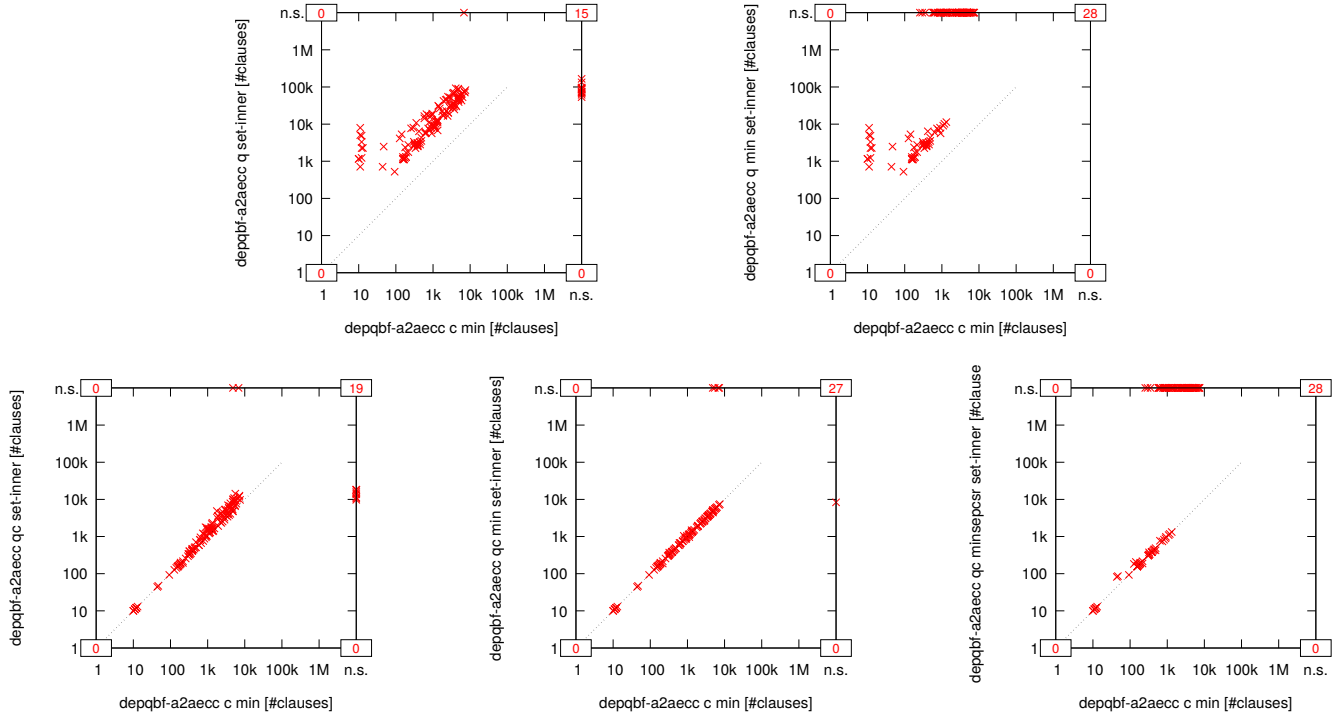


Fig. 110: Suite Herbstritt ($n = 157$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

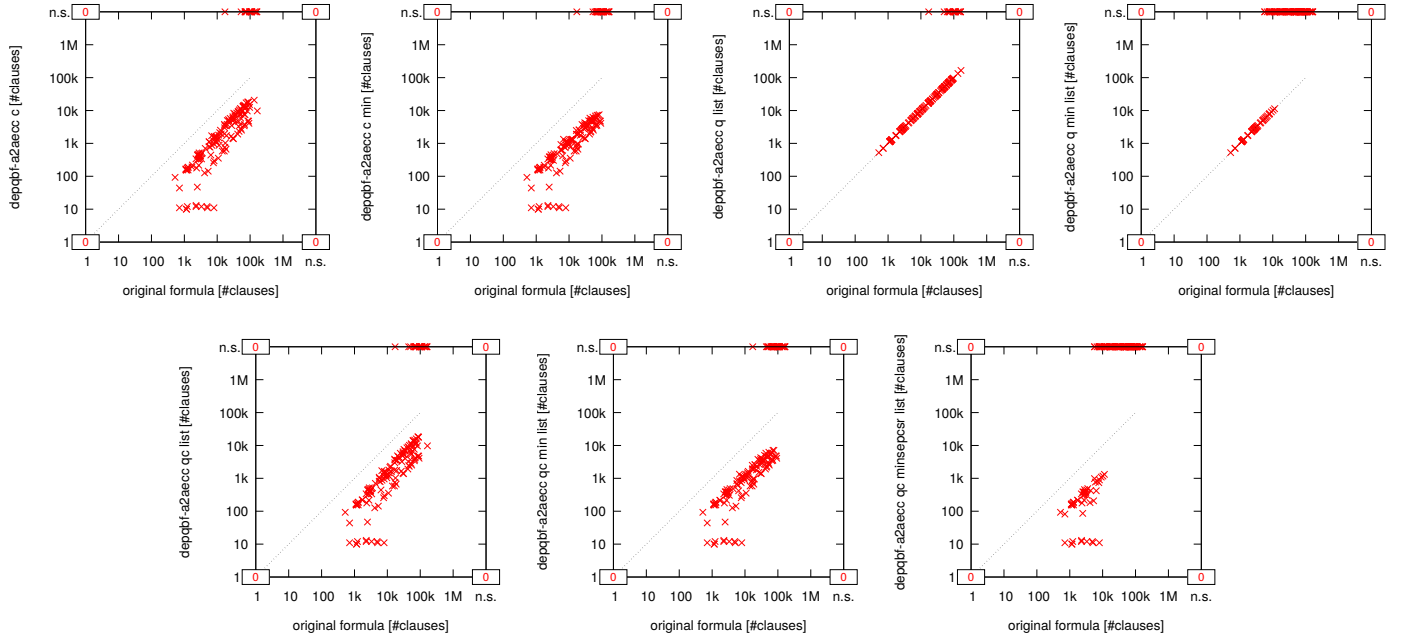


Fig. 111: Suite Herbstritt ($n = 157$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

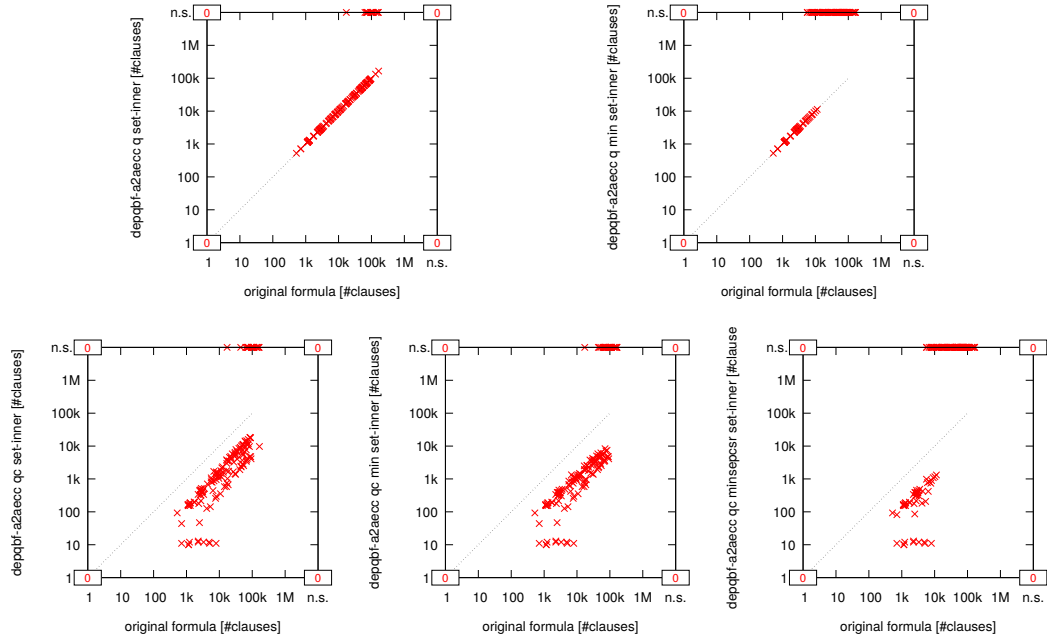


Fig. 112: Suite Herbstritt ($n = 157$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

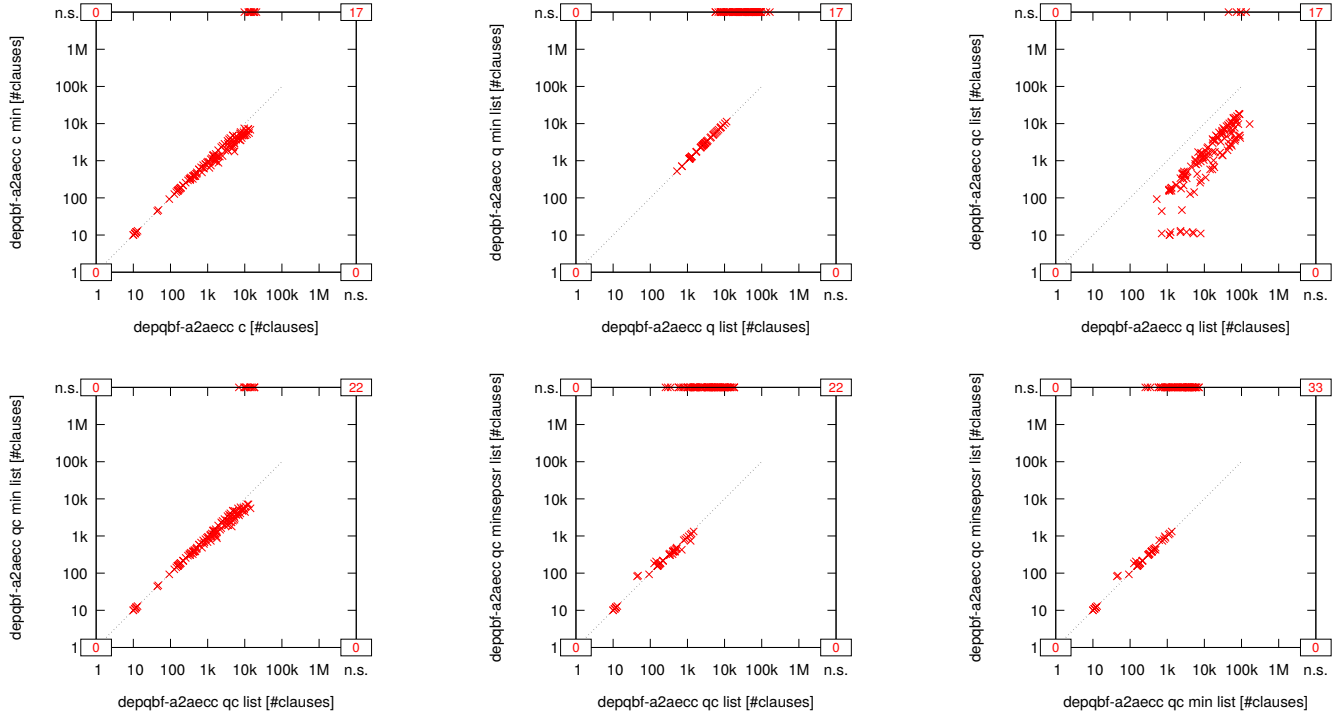


Fig. 113: Suite Herbstritt ($n = 157$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

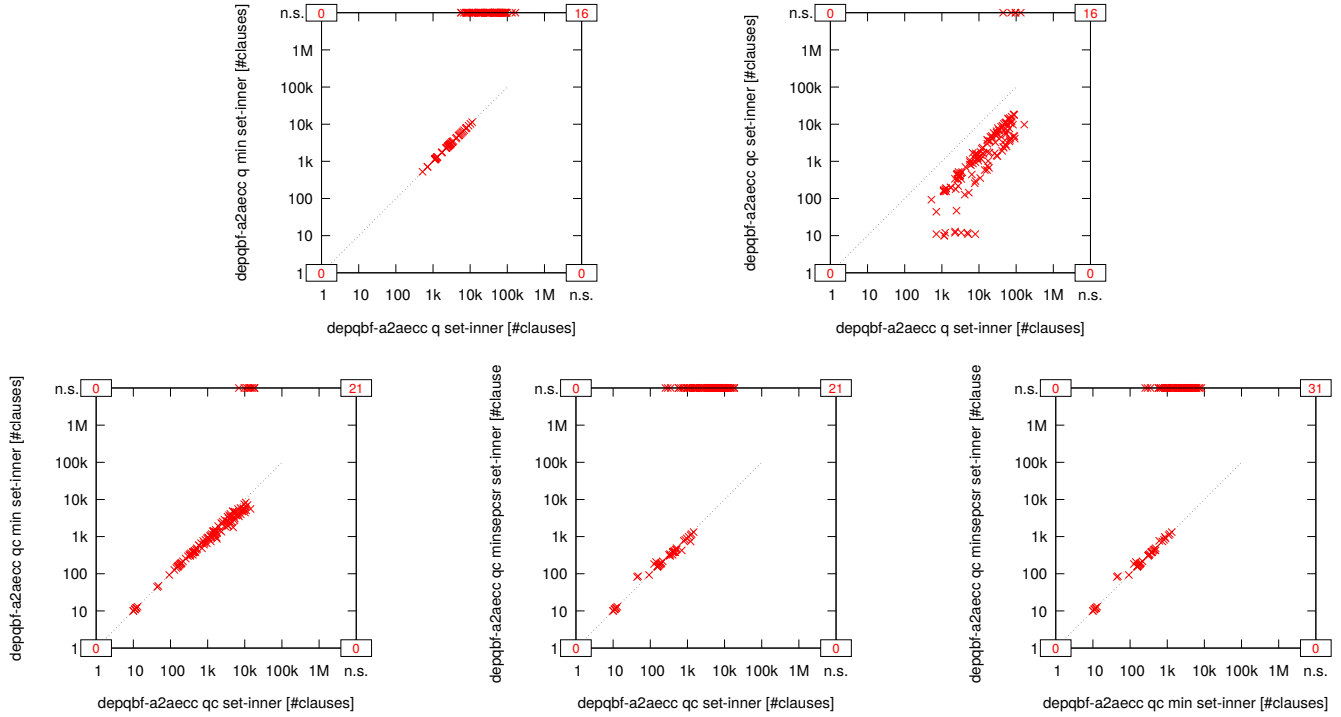


Fig. 114: Suite Herbstritt ($n = 157$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

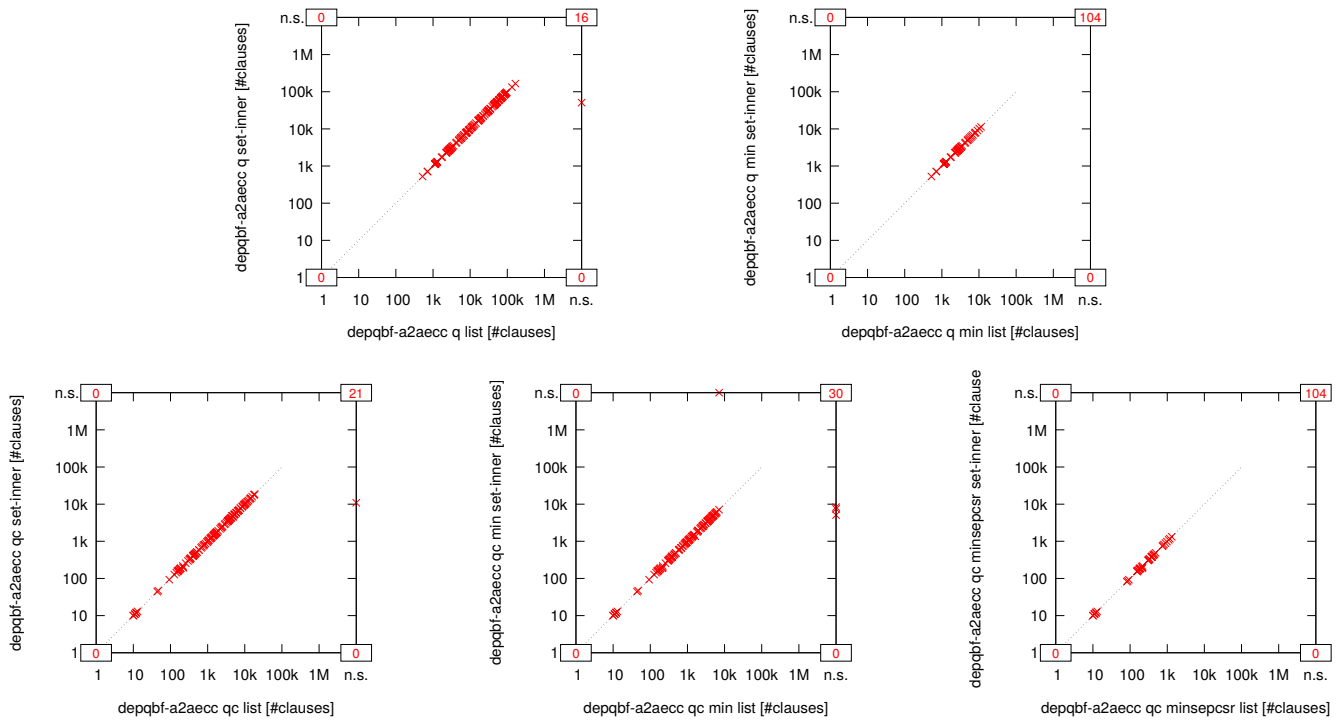


Fig. 115: Suite Herbristrit ($n = 157$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

16) Interian ($n = 24$):

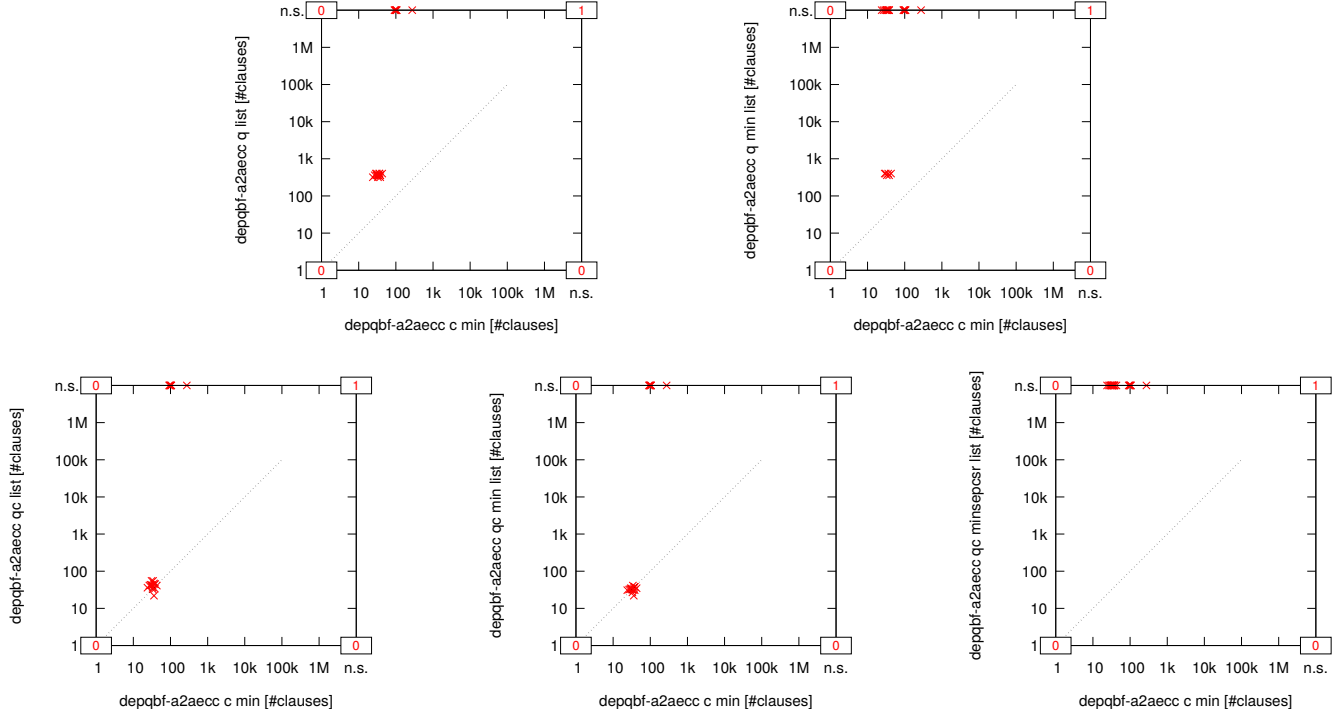


Fig. 116: Suite Interian ($n = 24$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

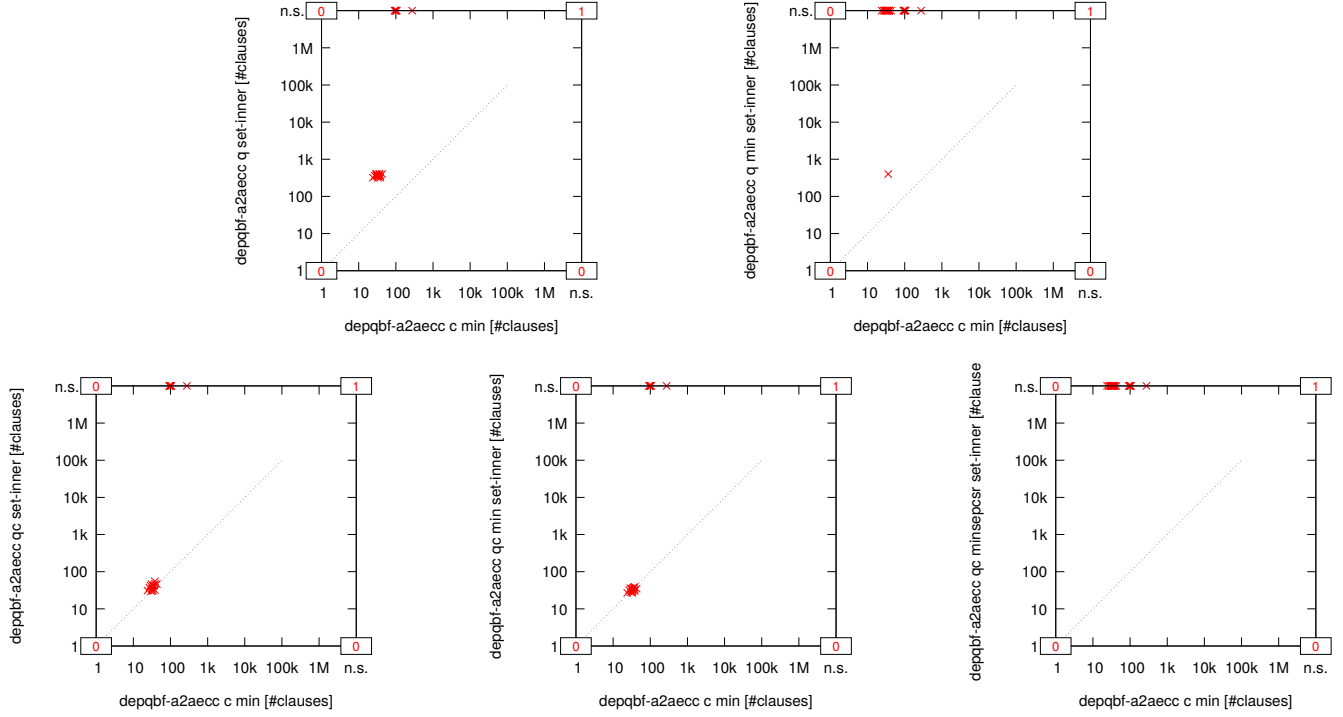


Fig. 117: Suite Interian ($n = 24$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

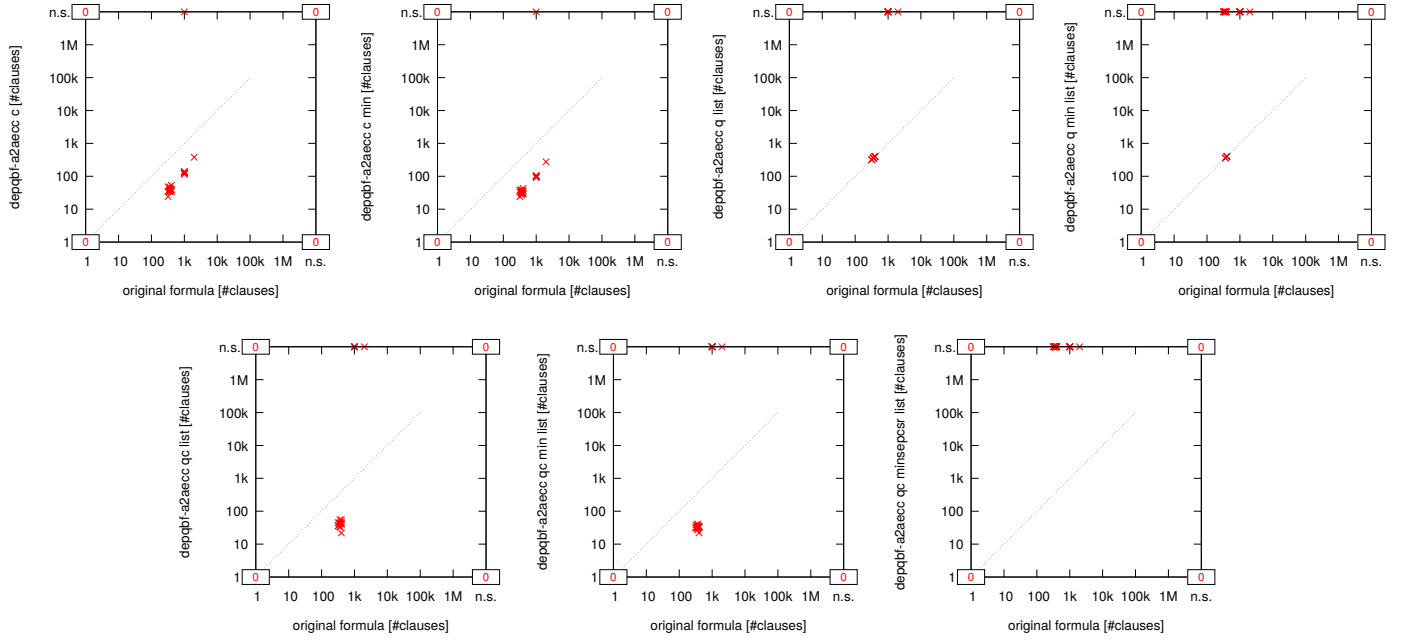


Fig. 118: Suite Interian ($n = 24$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

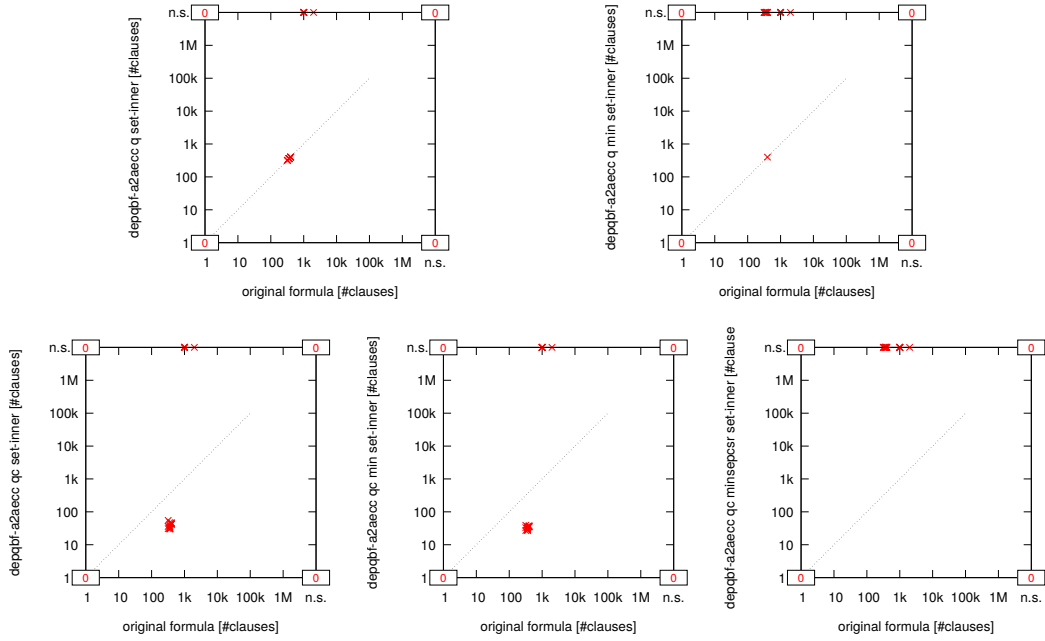


Fig. 119: Suite Interian ($n = 24$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

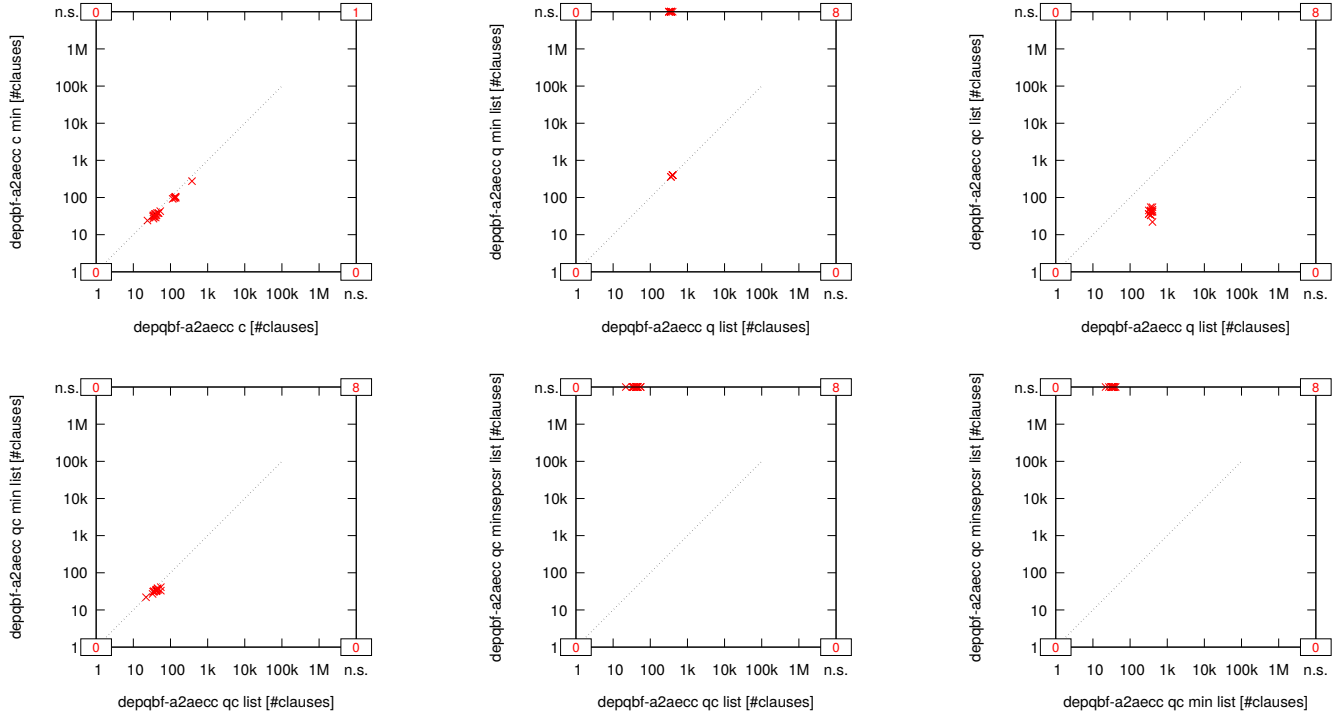


Fig. 120: Suite Interian ($n = 24$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

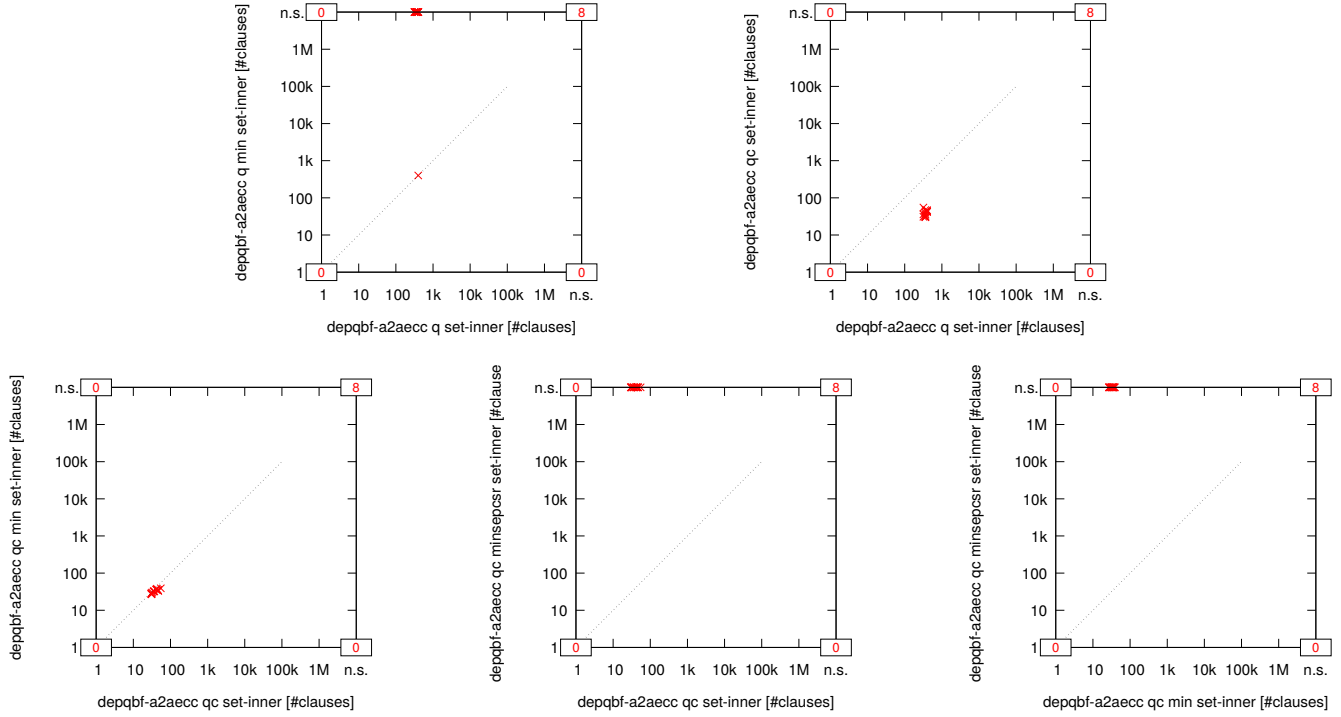


Fig. 121: Suite Interian ($n = 24$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

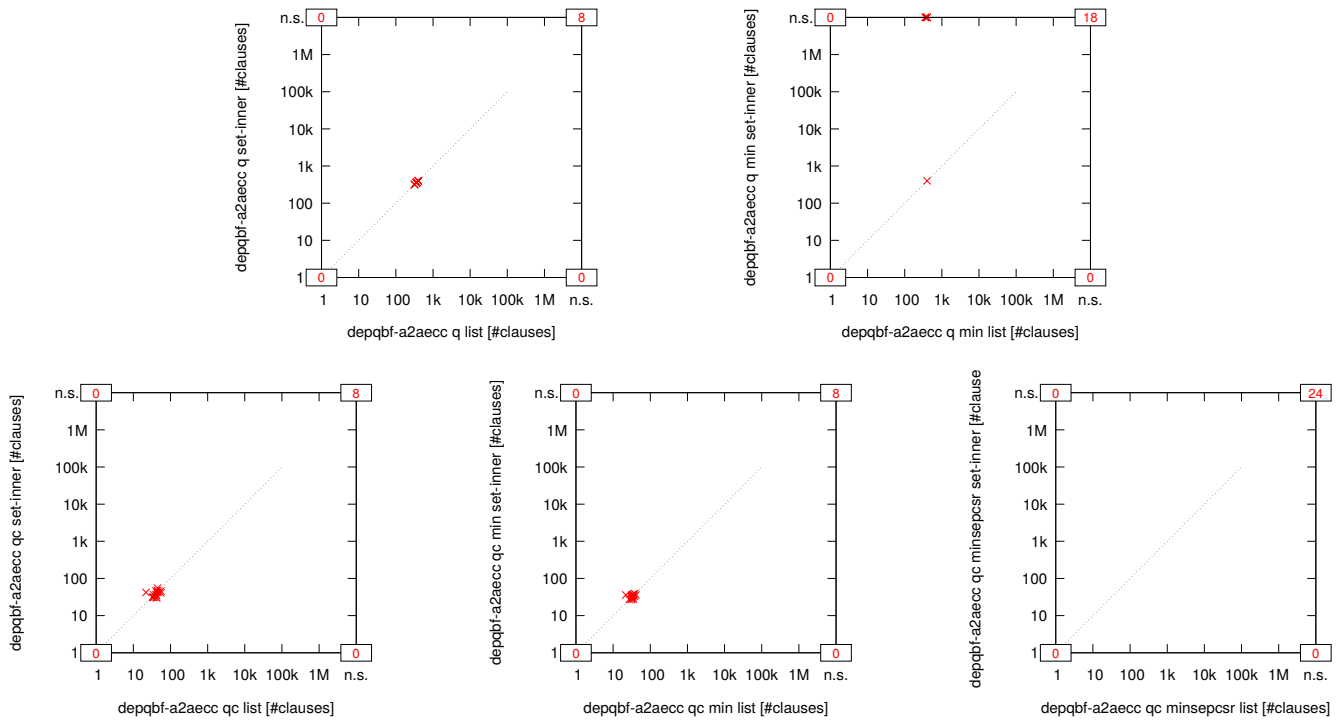


Fig. 122: Suite Interian ($n = 24$): Comparing sizes of different unsatisfiable cores in `DepQBF-a2aecc` in list versus set-inner semantics (number of clauses).

17) Jordan-Kaiser ($n = 83$):

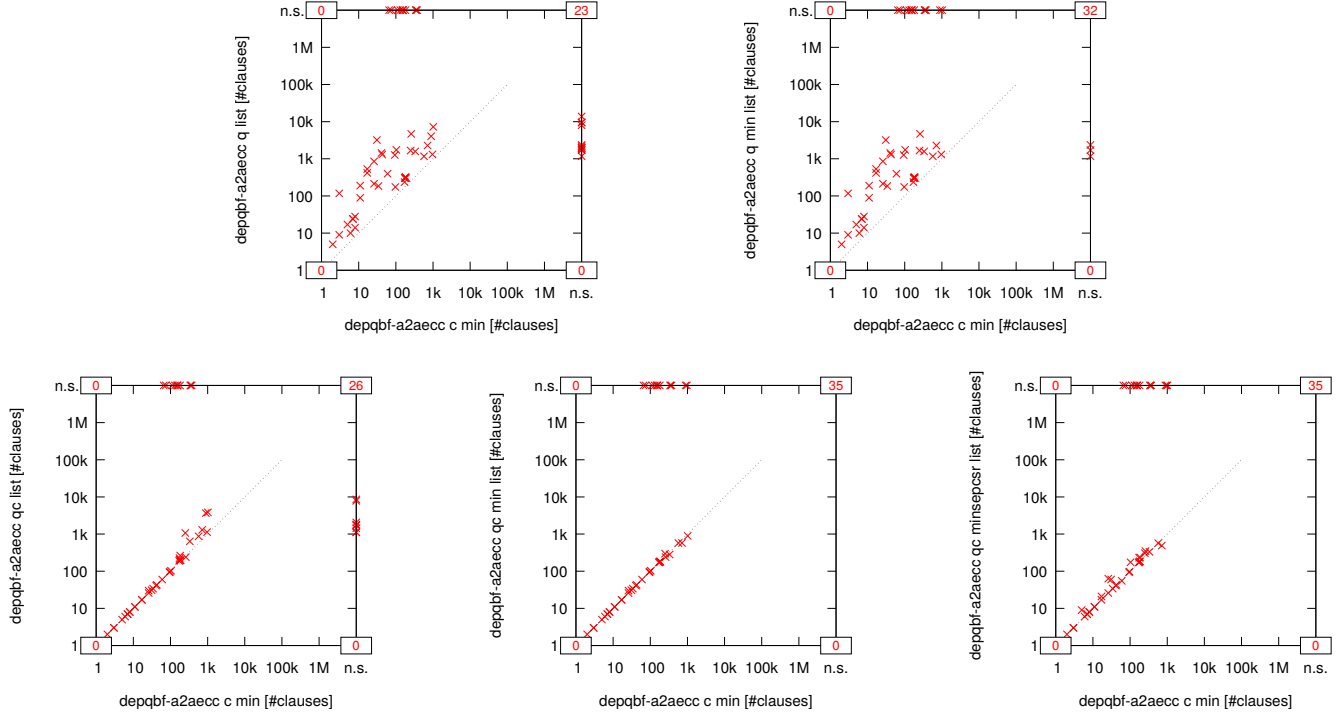


Fig. 123: Suite Jordan-Kaiser ($n = 83$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

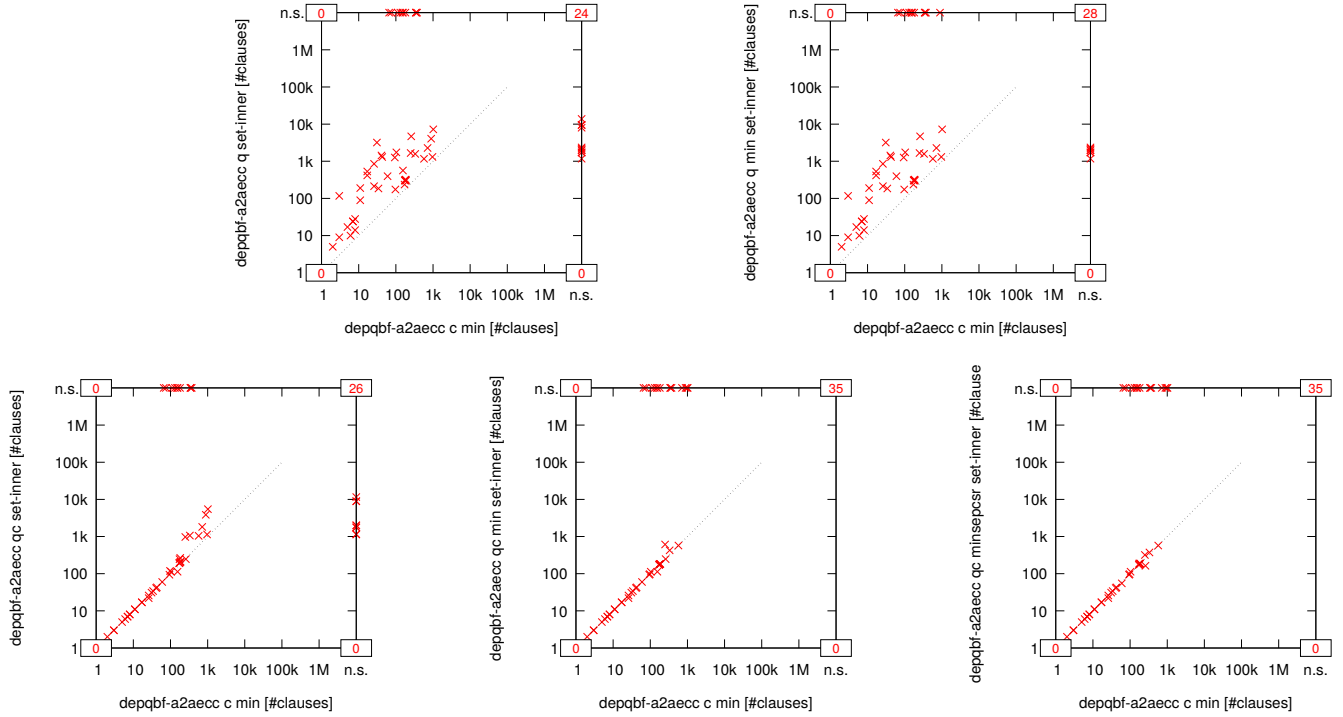


Fig. 124: Suite Jordan-Kaiser ($n = 83$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

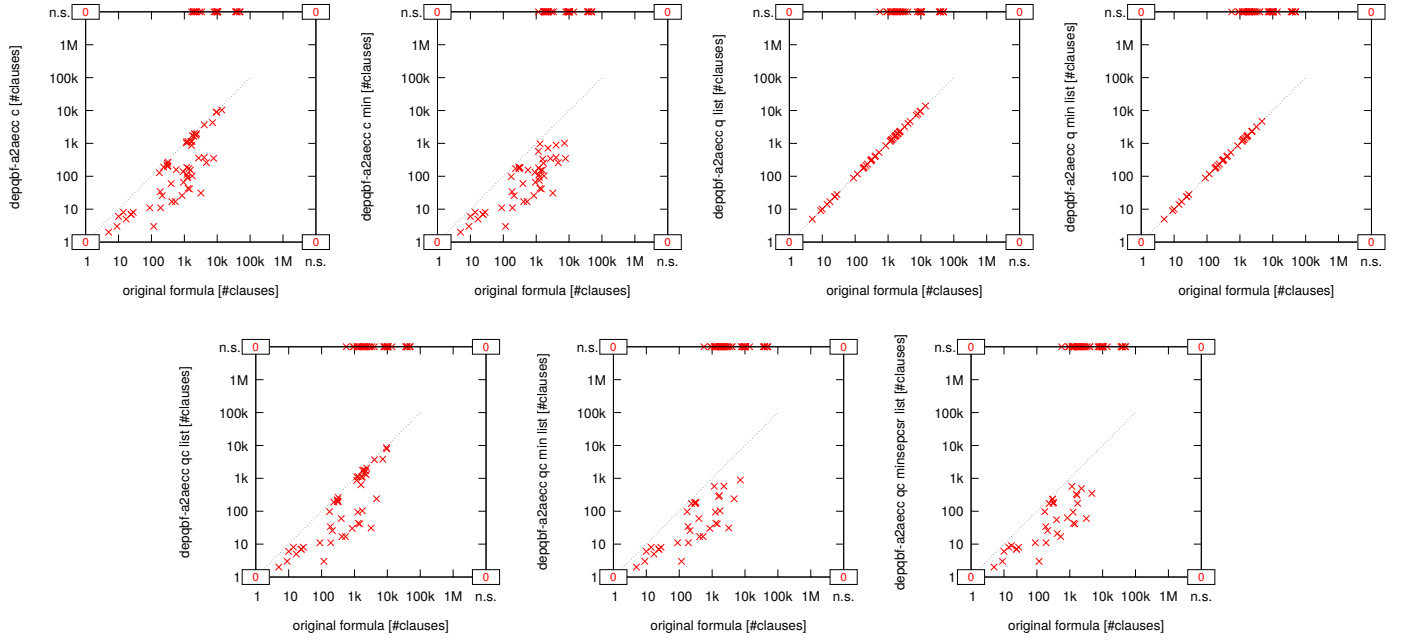


Fig. 125: Suite Jordan-Kaiser ($n = 83$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

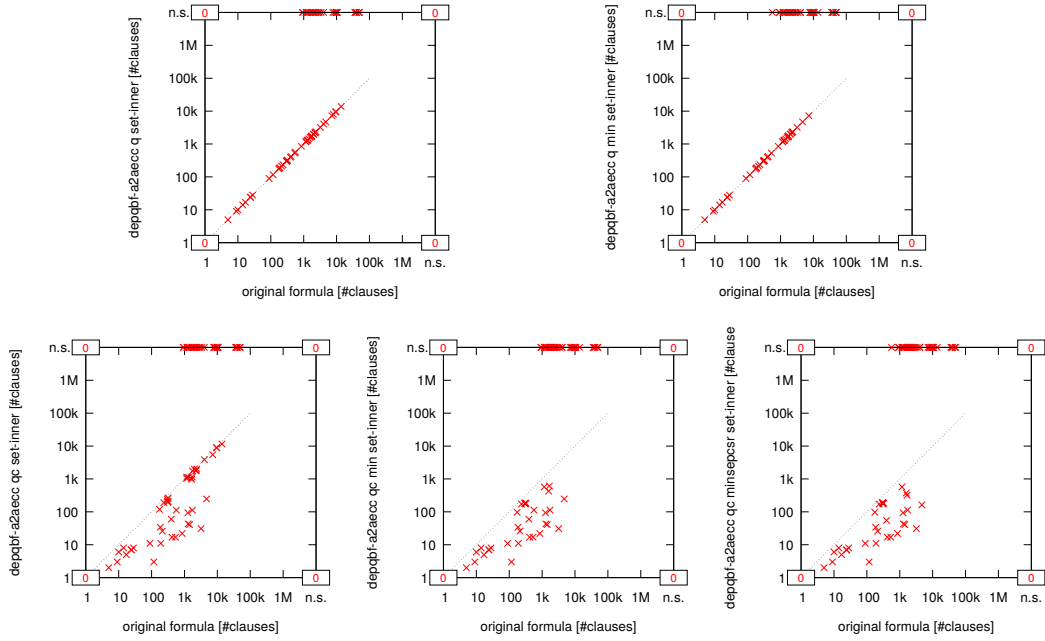


Fig. 126: Suite Jordan-Kaiser ($n = 83$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

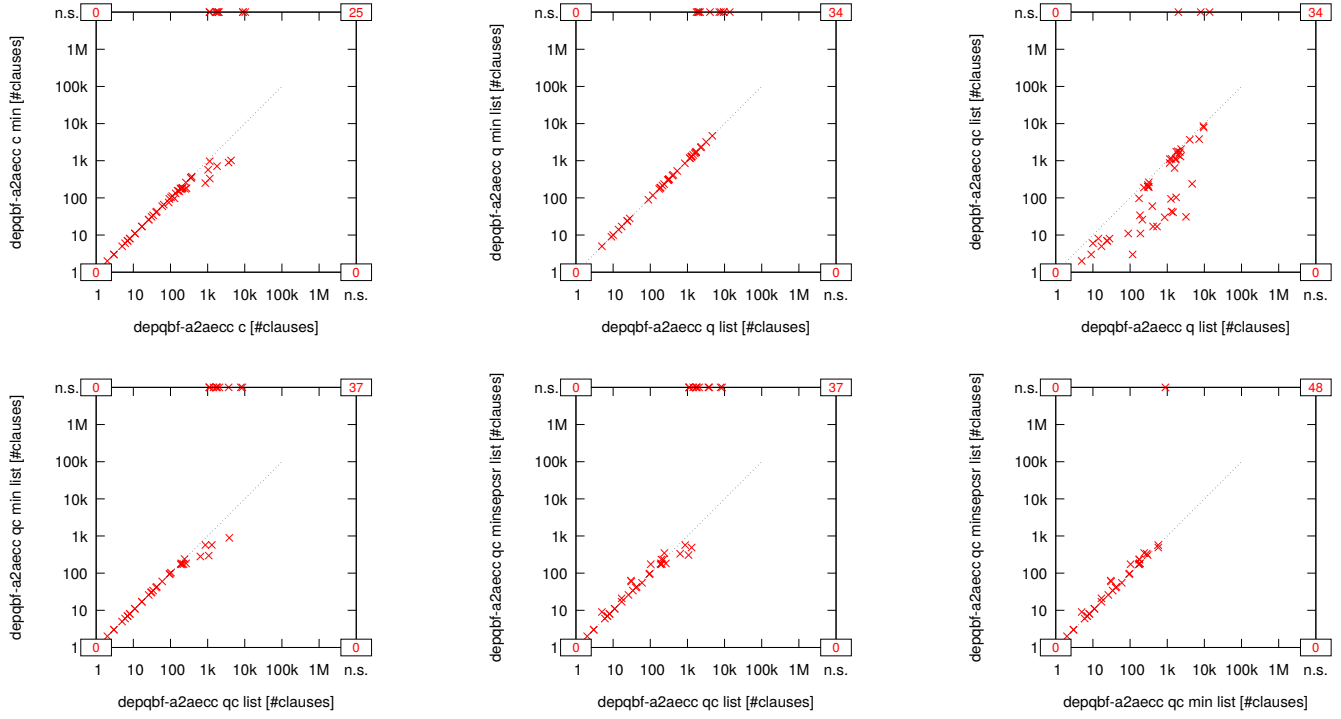


Fig. 127: Suite Jordan-Kaiser ($n = 83$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

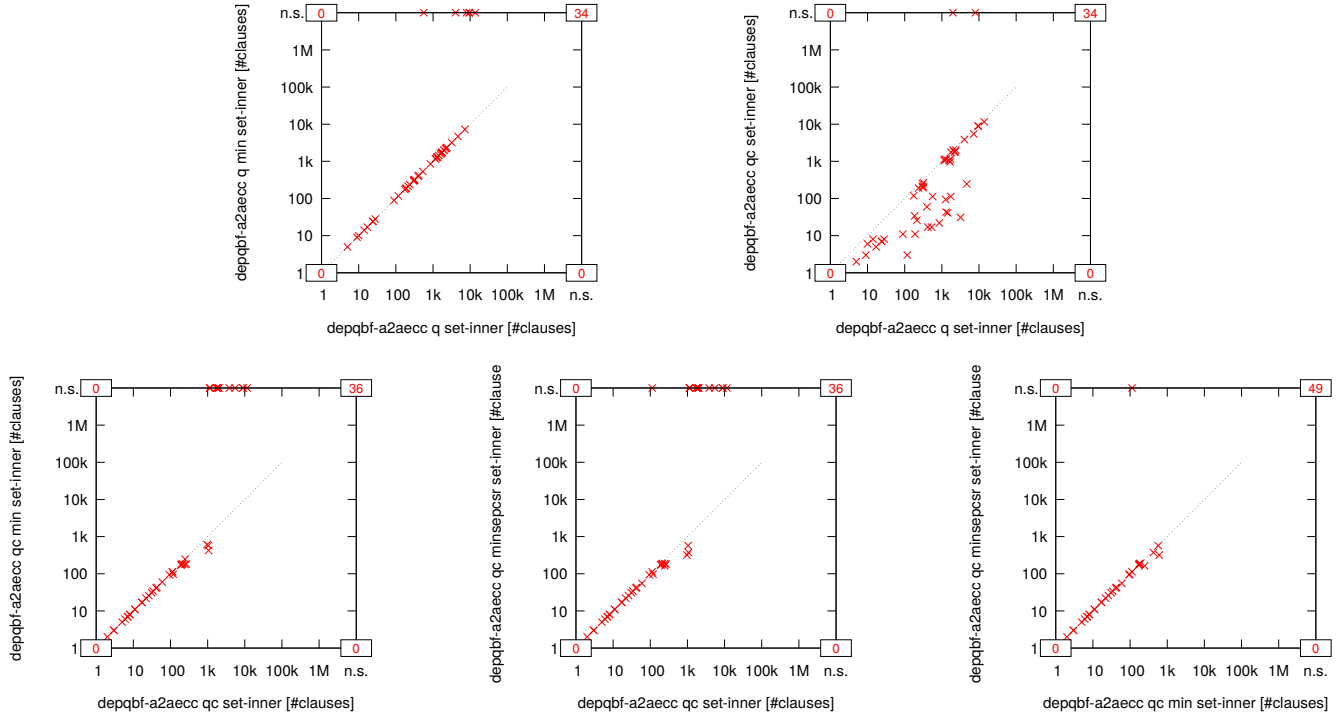


Fig. 128: Suite Jordan-Kaiser ($n = 83$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

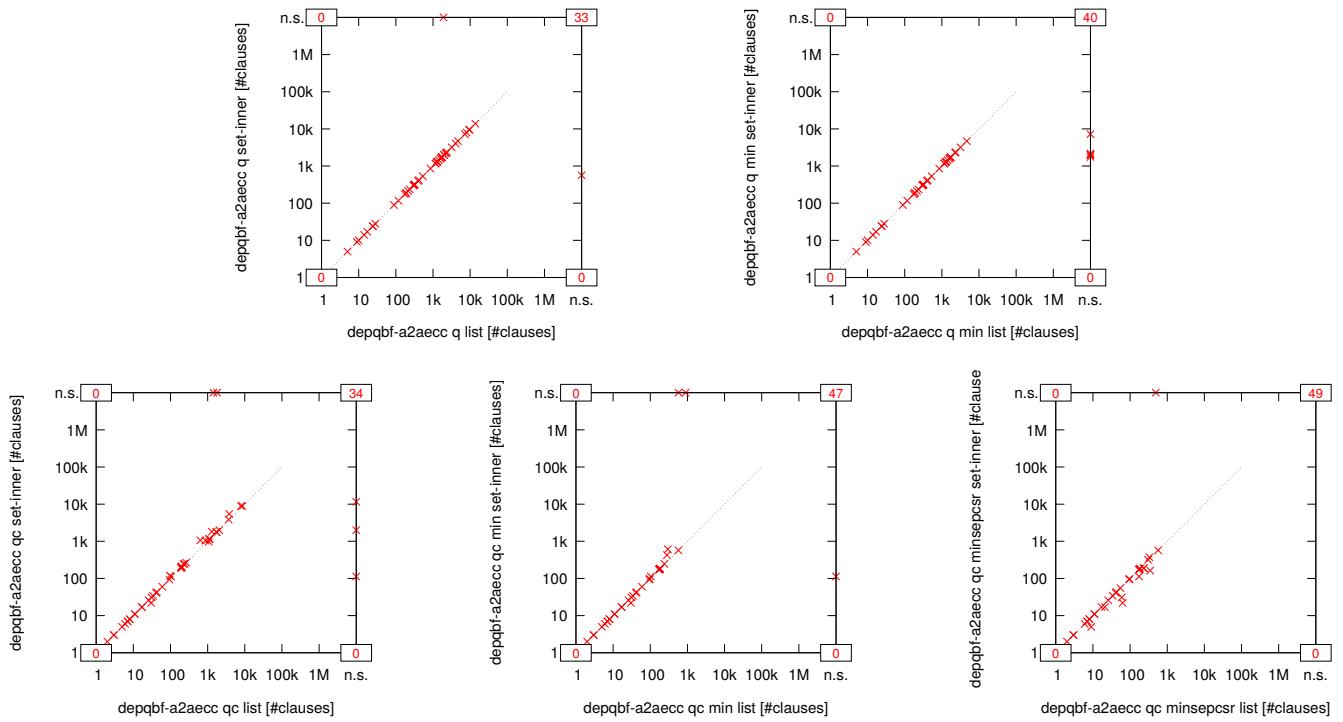


Fig. 129: Suite Jordan-Kaiser ($n = 83$): Comparing sizes of different unsatisfiable cores in `DepQBF-a2aecc` in list versus set-inner semantics (number of clauses).

18) Katz ($n = 8$):

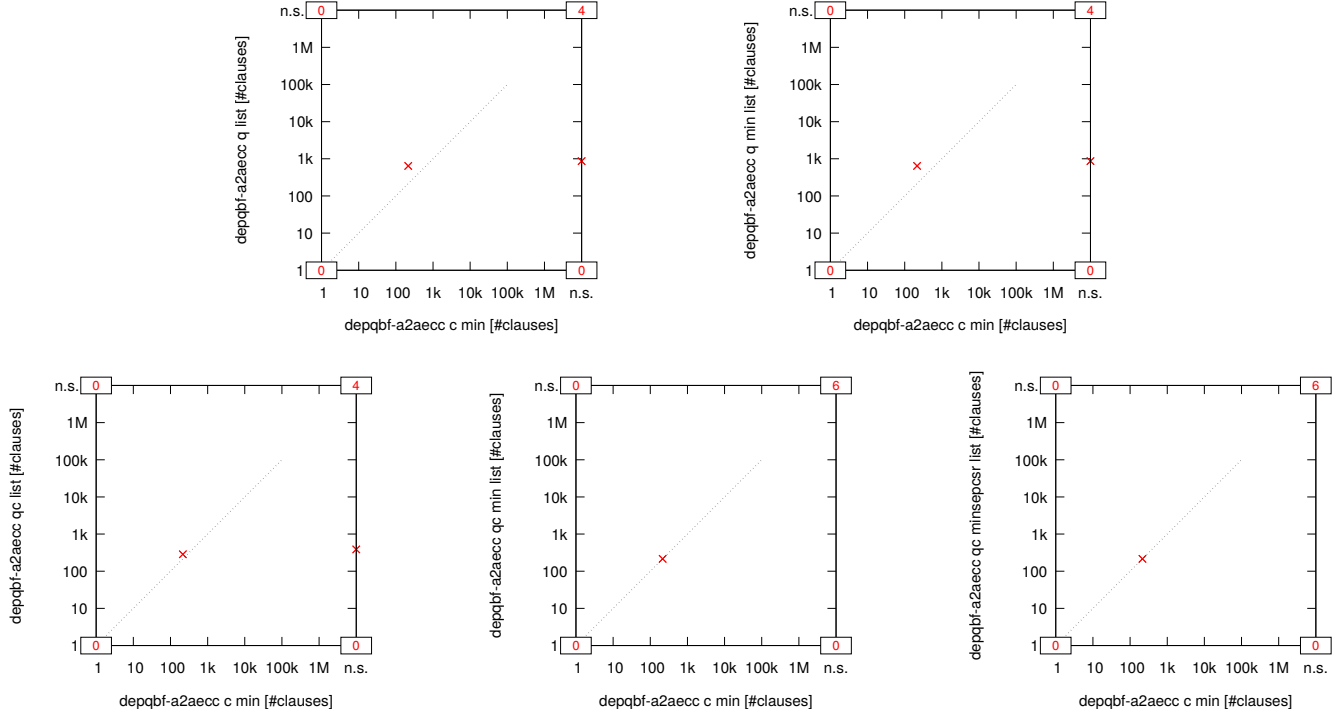


Fig. 130: Suite Katz ($n = 8$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

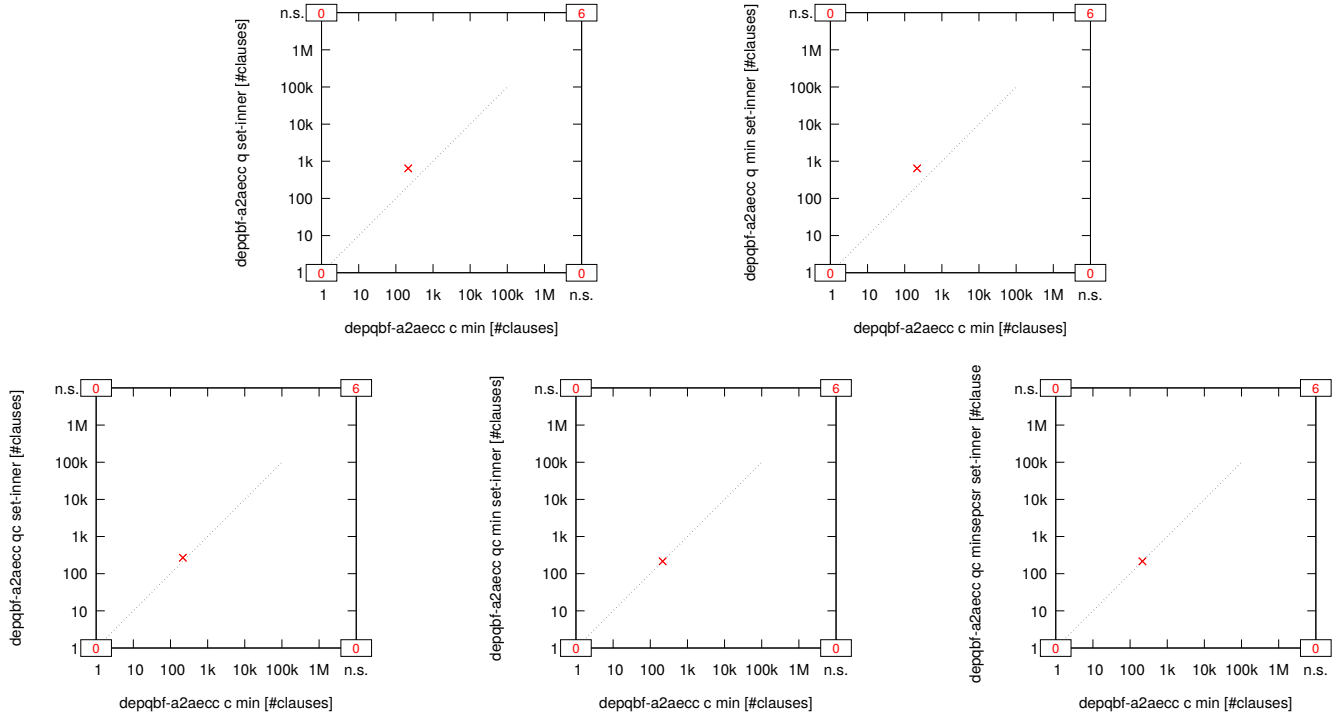


Fig. 131: Suite Katz ($n = 8$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

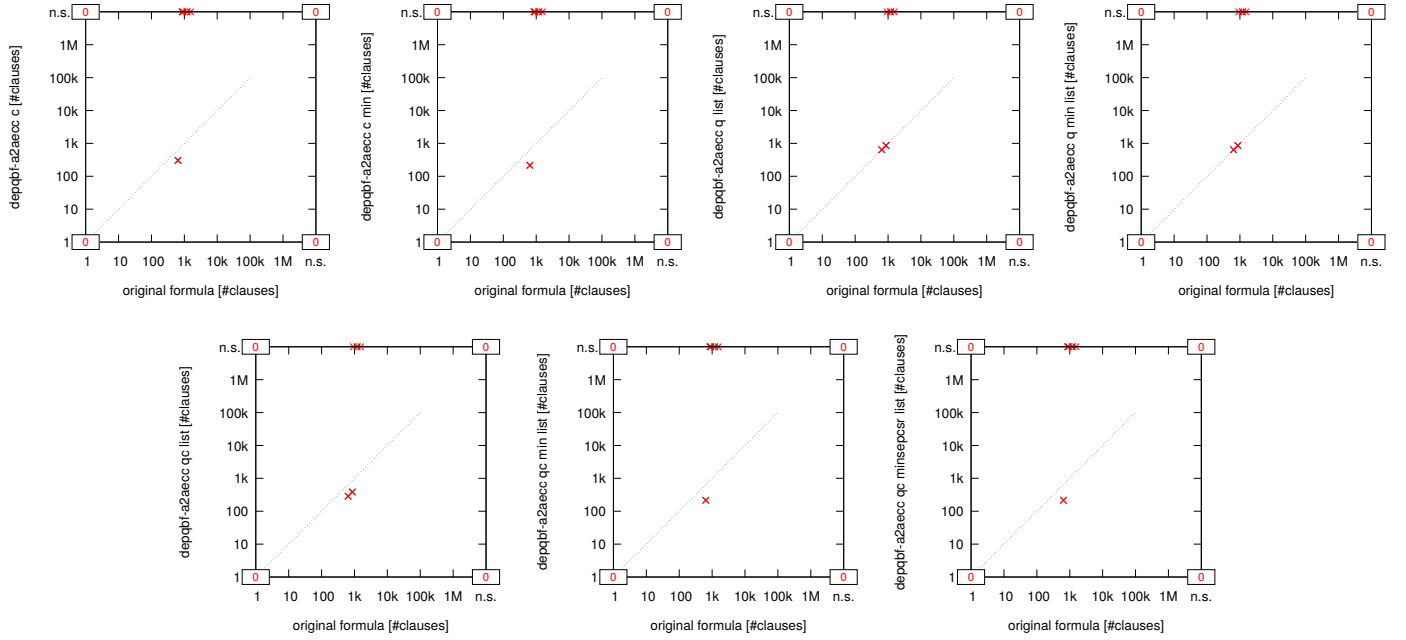


Fig. 132: Suite Katz ($n = 8$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

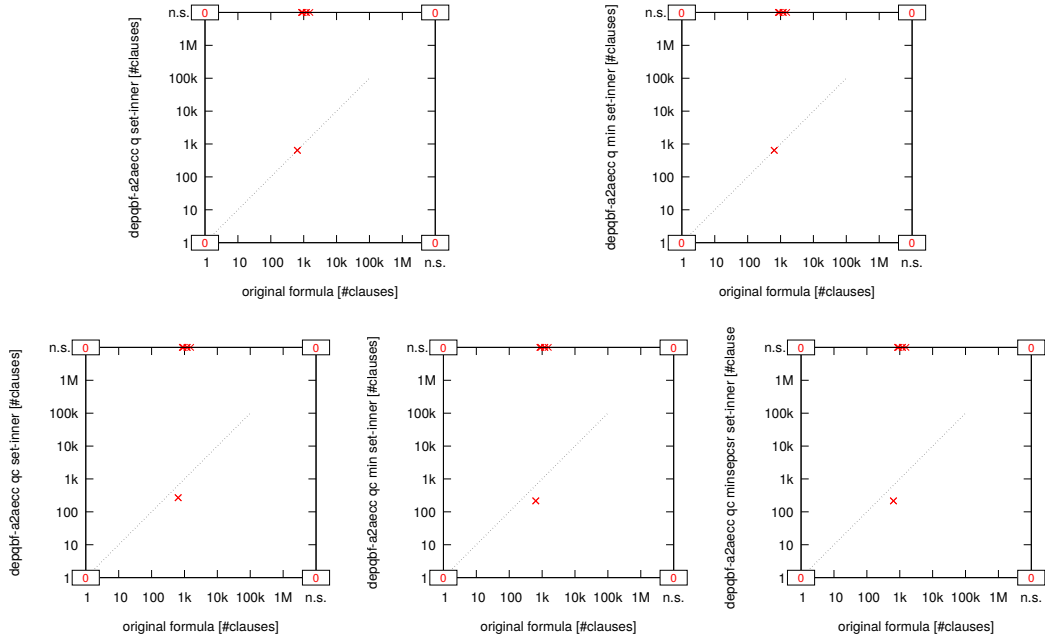


Fig. 133: Suite Katz ($n = 8$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

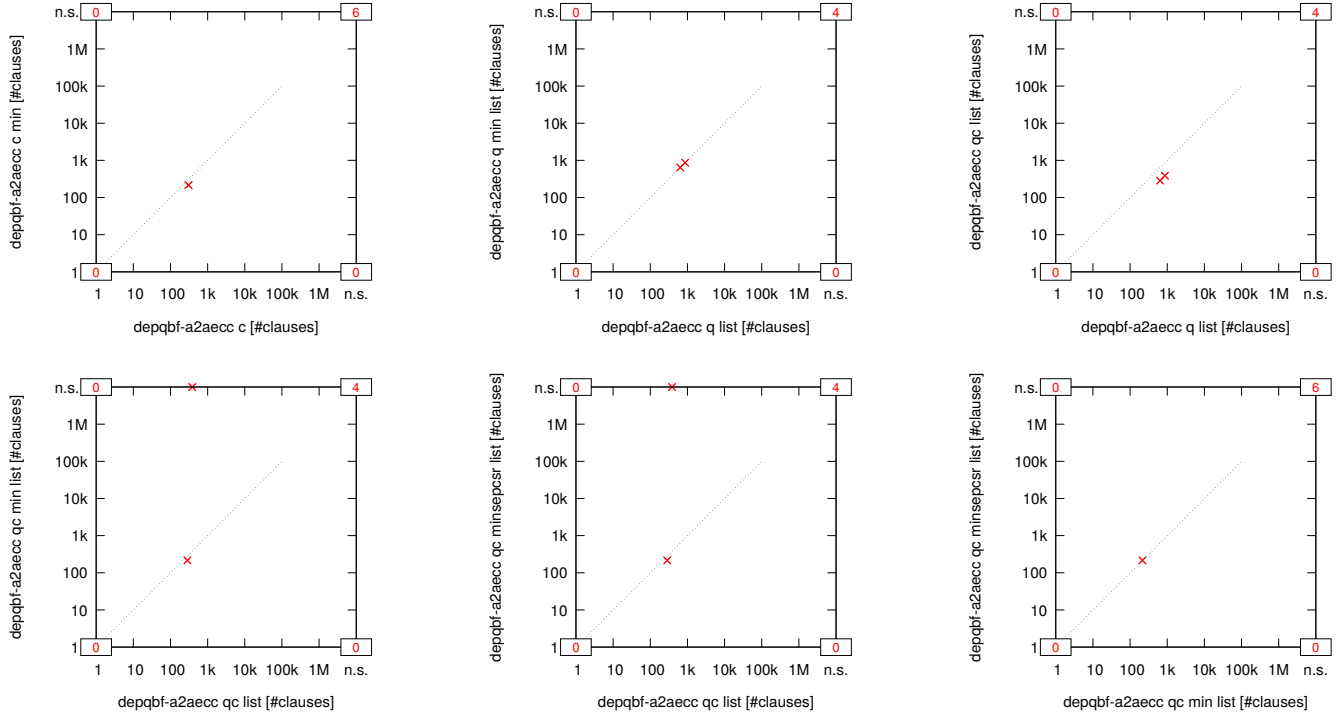


Fig. 134: Suite Katz ($n = 8$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

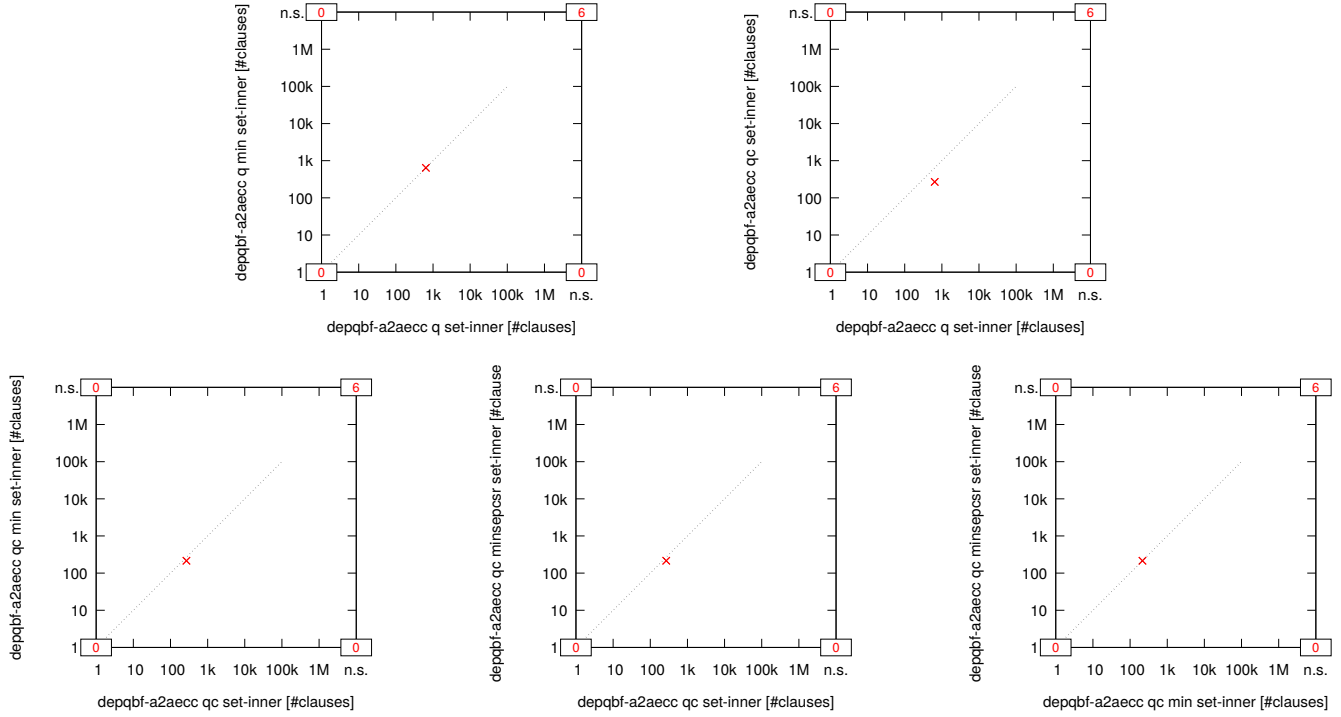


Fig. 135: Suite Katz ($n = 8$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

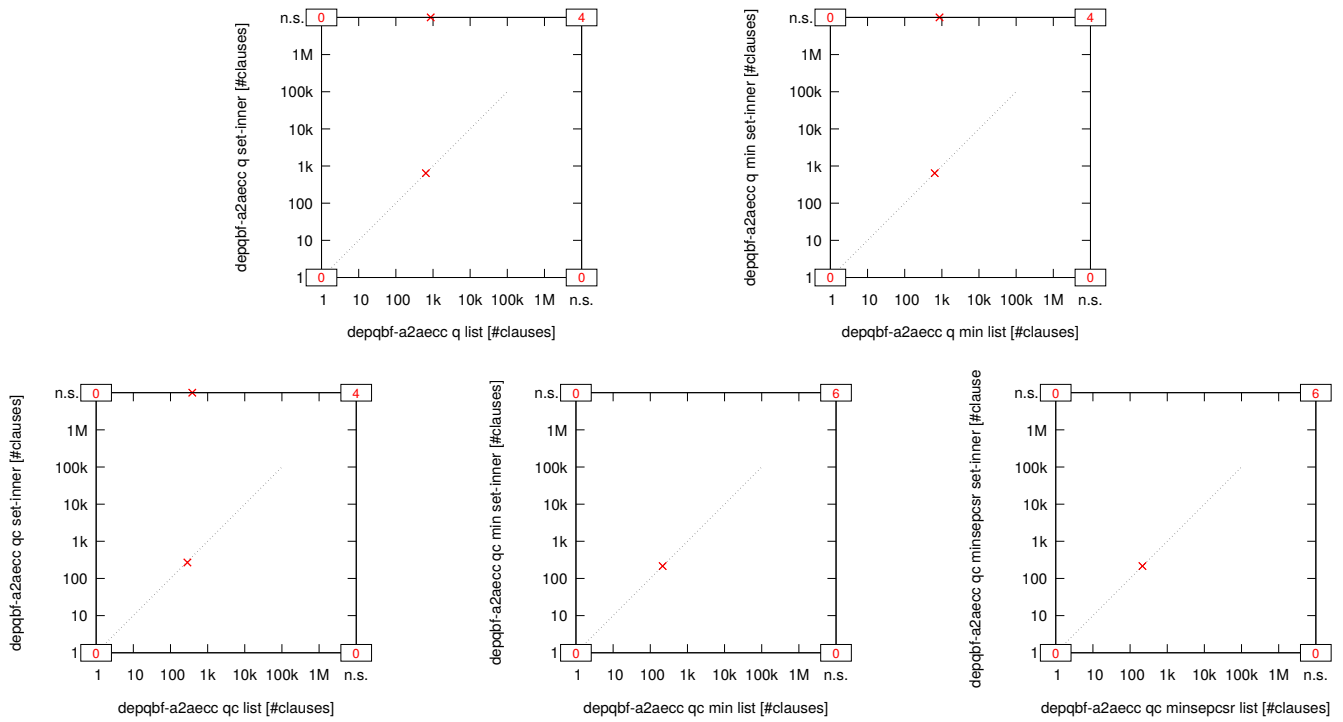


Fig. 136: Suite Katz ($n = 8$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

19) Klieber ($n = 15$):

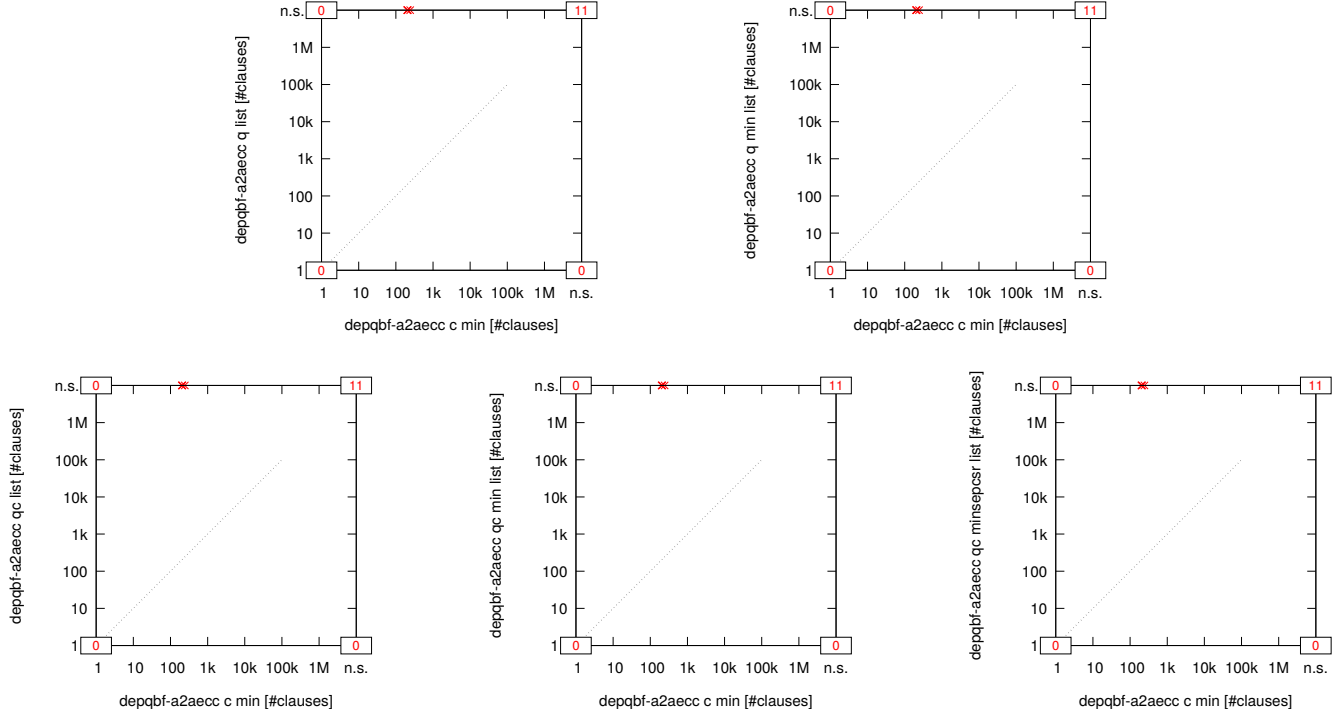


Fig. 137: Suite Klieber ($n = 15$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

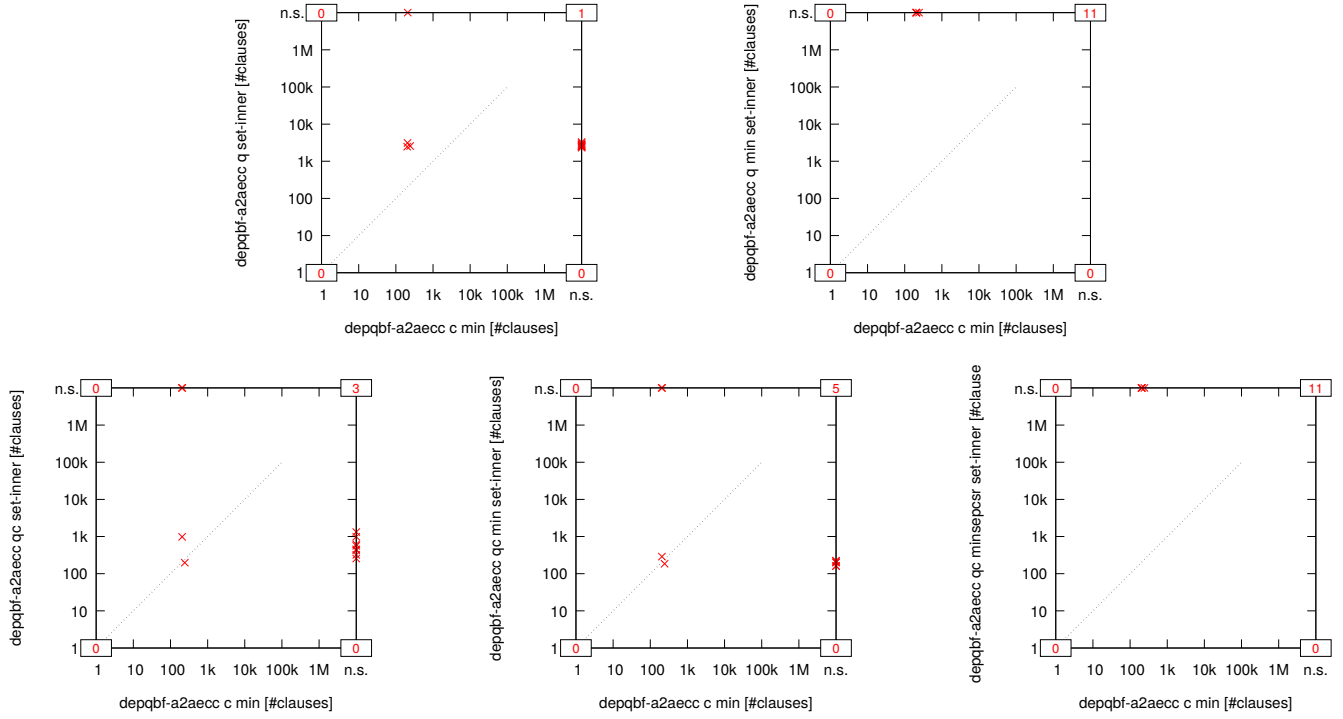


Fig. 138: Suite Klieber ($n = 15$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

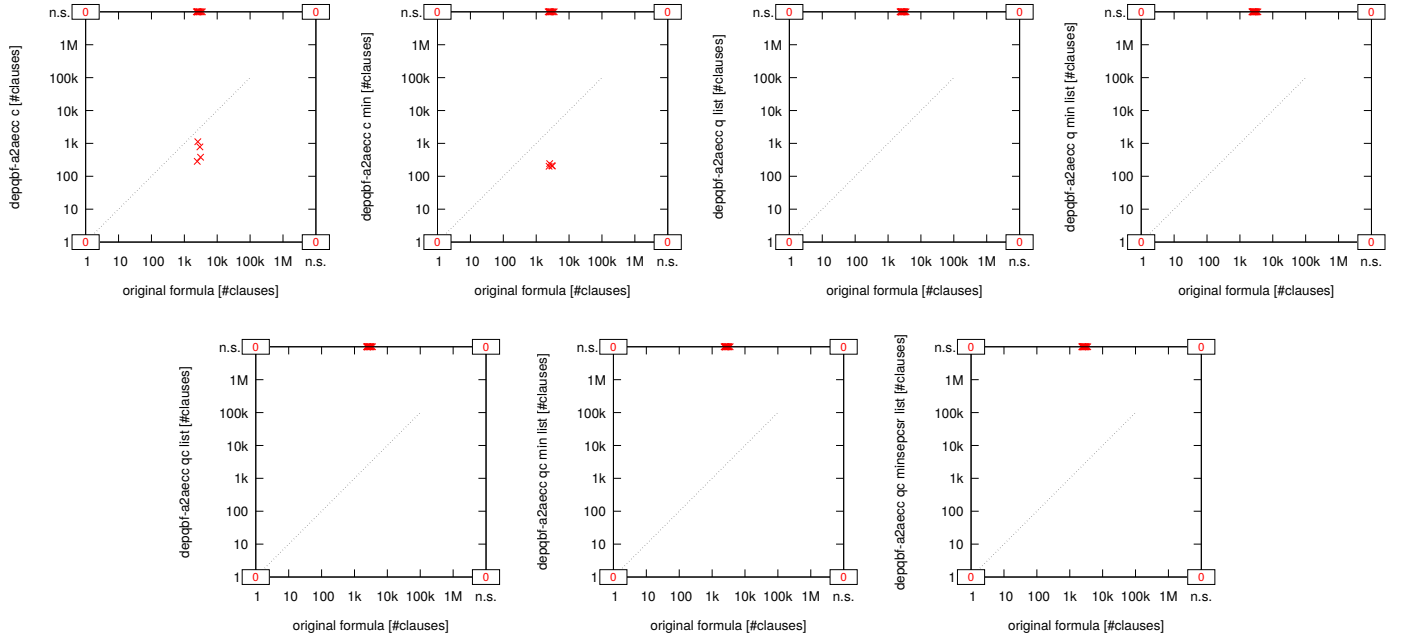


Fig. 139: Suite Klieber ($n = 15$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

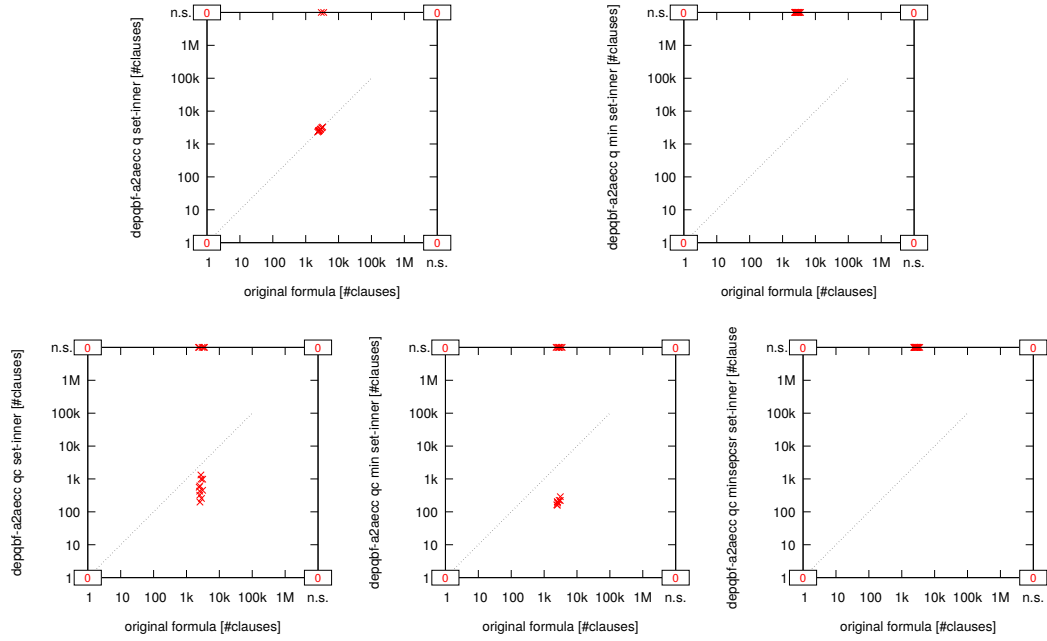


Fig. 140: Suite Klieber ($n = 15$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

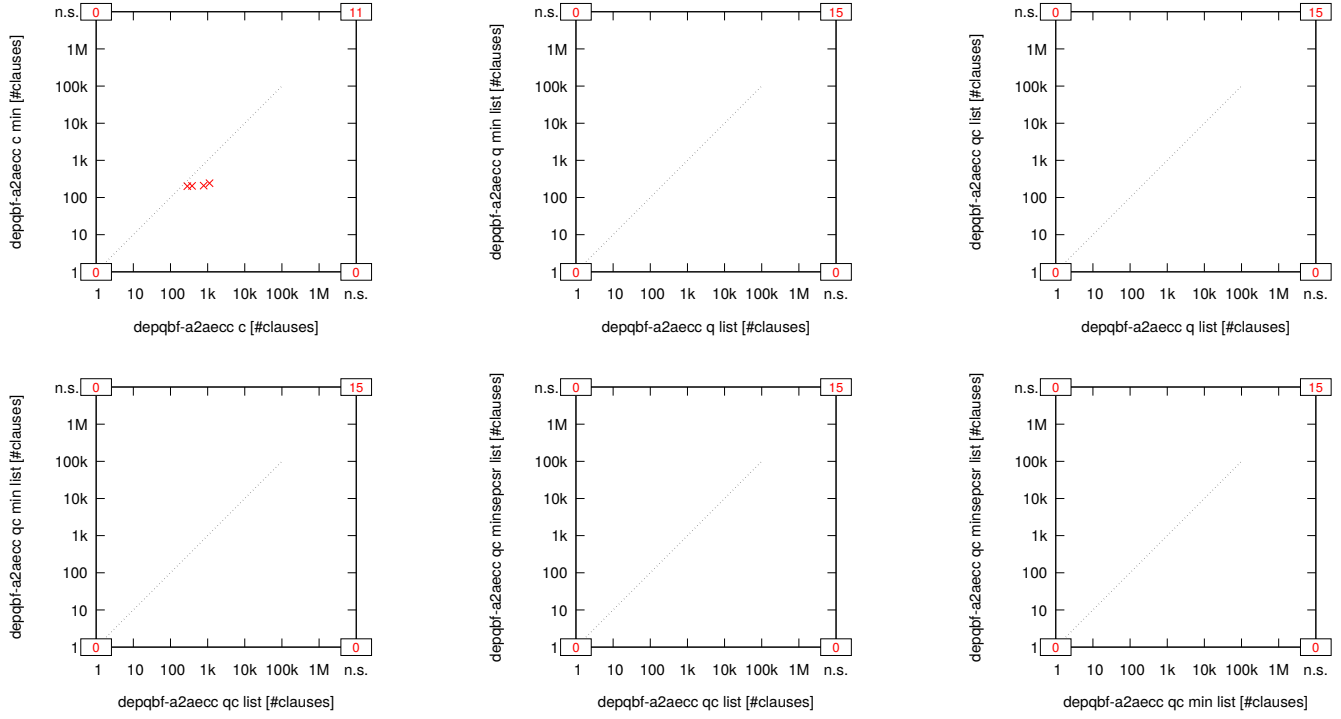


Fig. 141: Suite Klieber ($n = 15$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

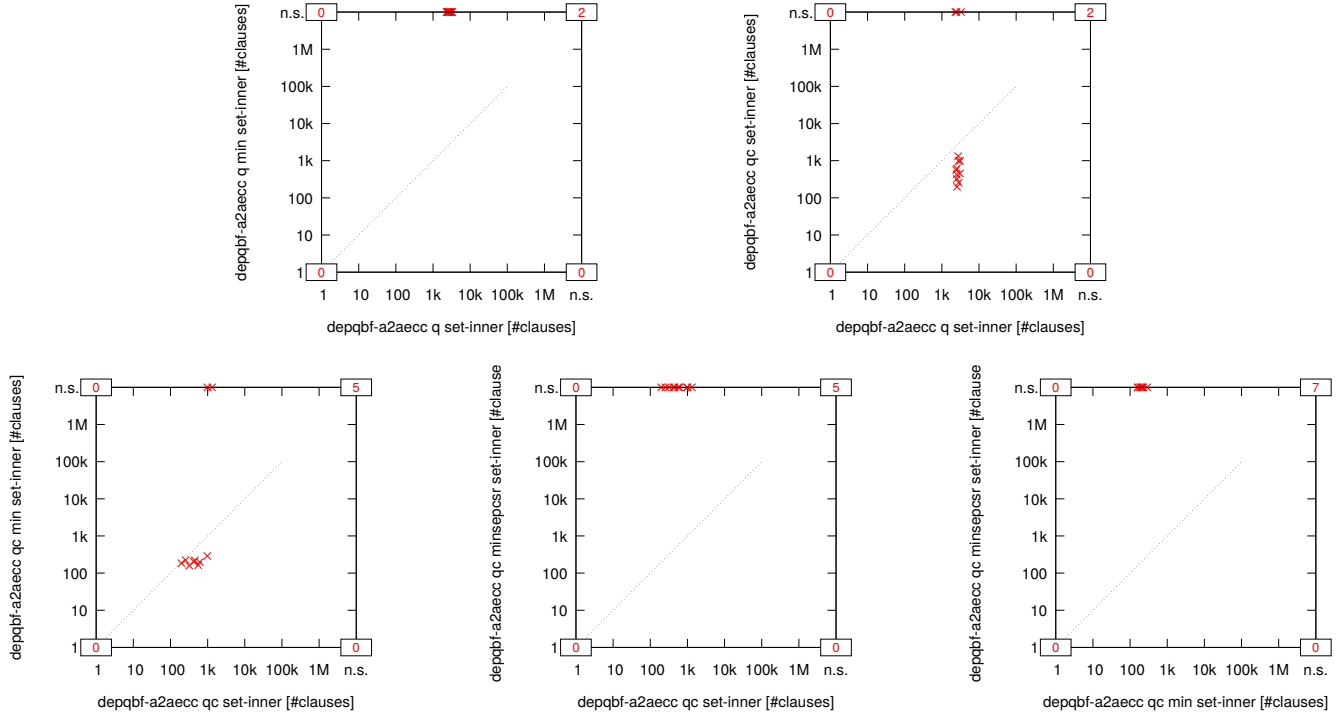


Fig. 142: Suite Klieber ($n = 15$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

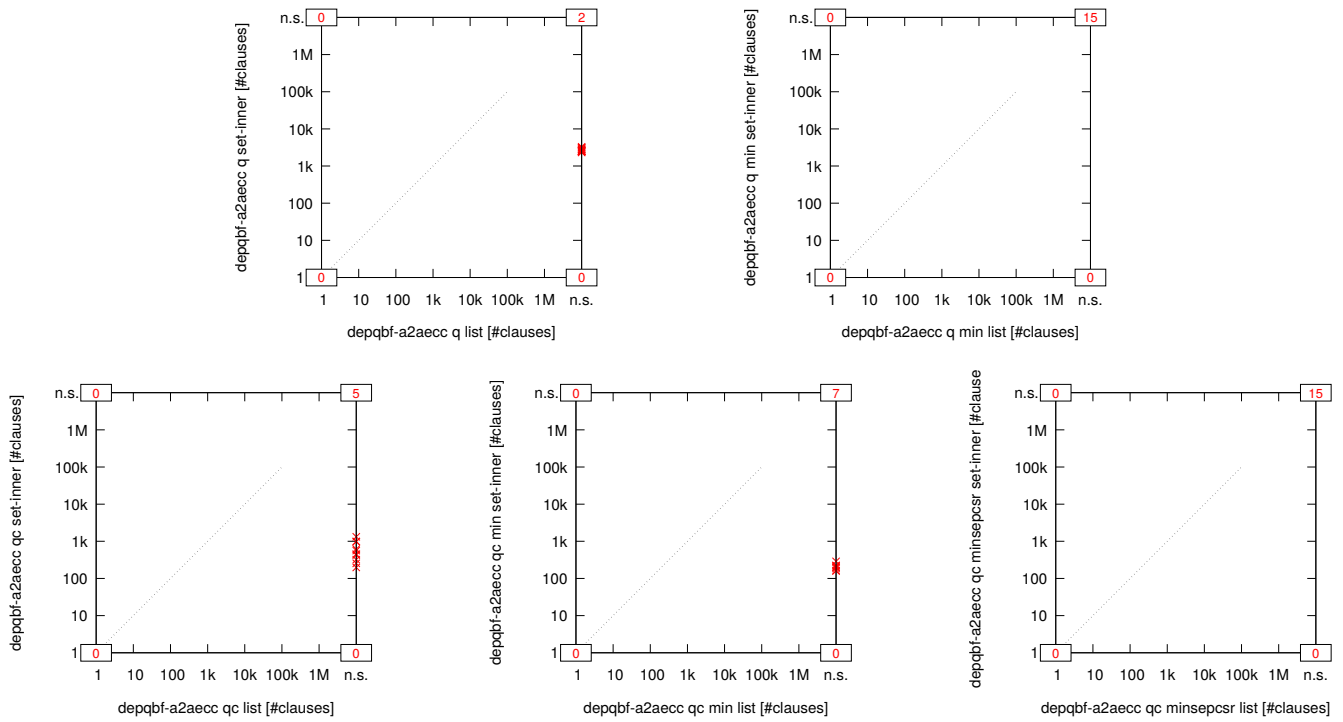


Fig. 143: Suite Klieber ($n = 15$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

20) *Kontchakov* ($n = 70$):

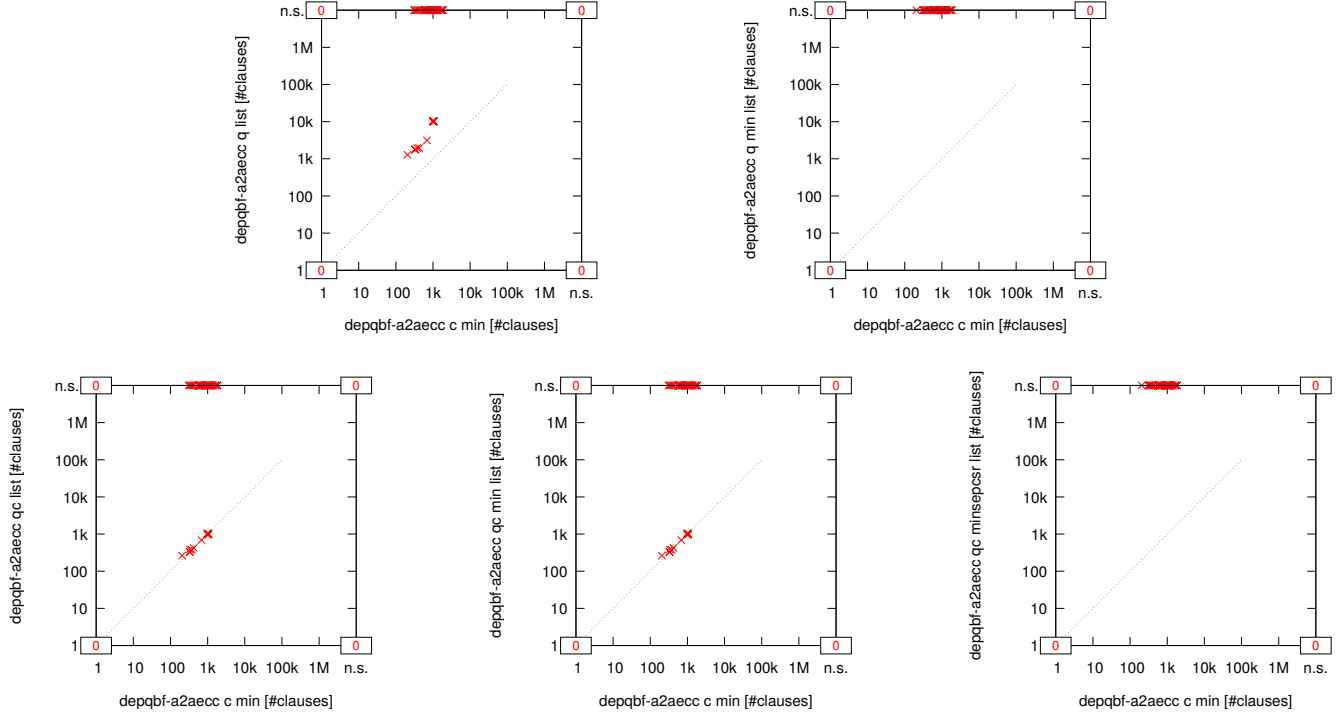


Fig. 144: Suite Kontchakov ($n = 70$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

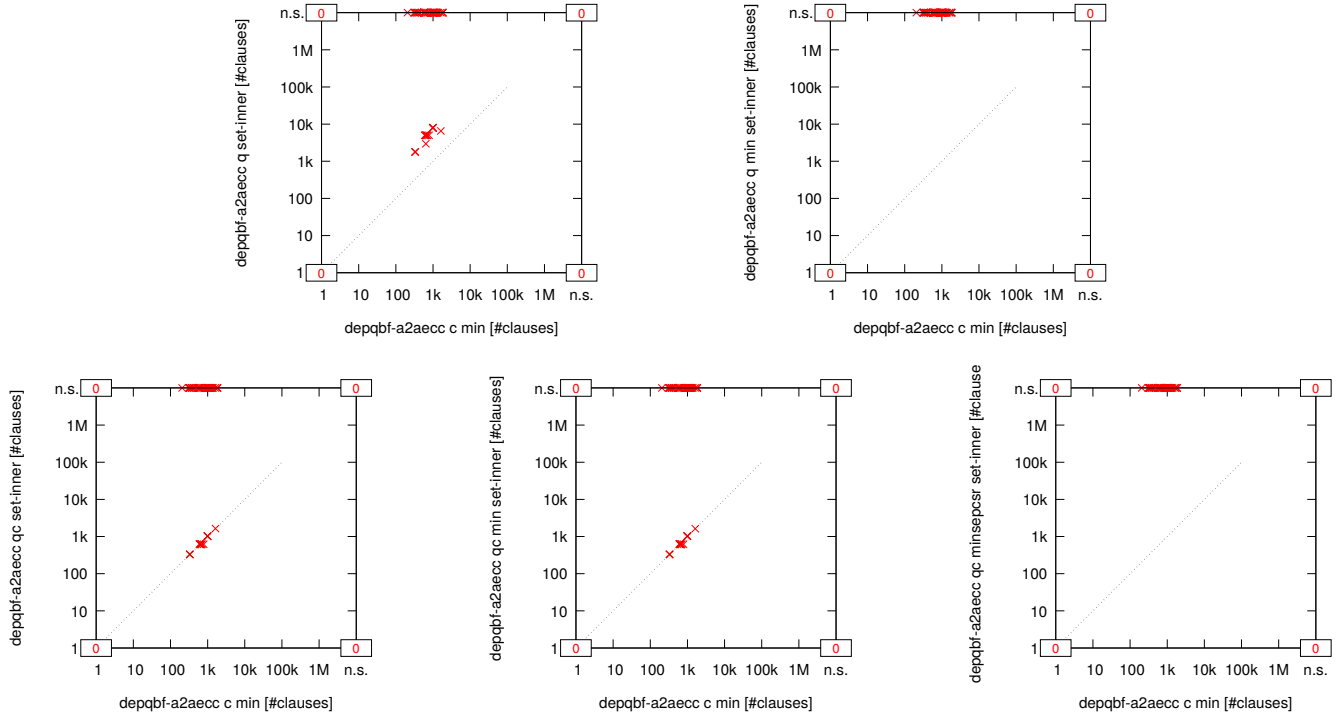


Fig. 145: Suite Kontchakov ($n = 70$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

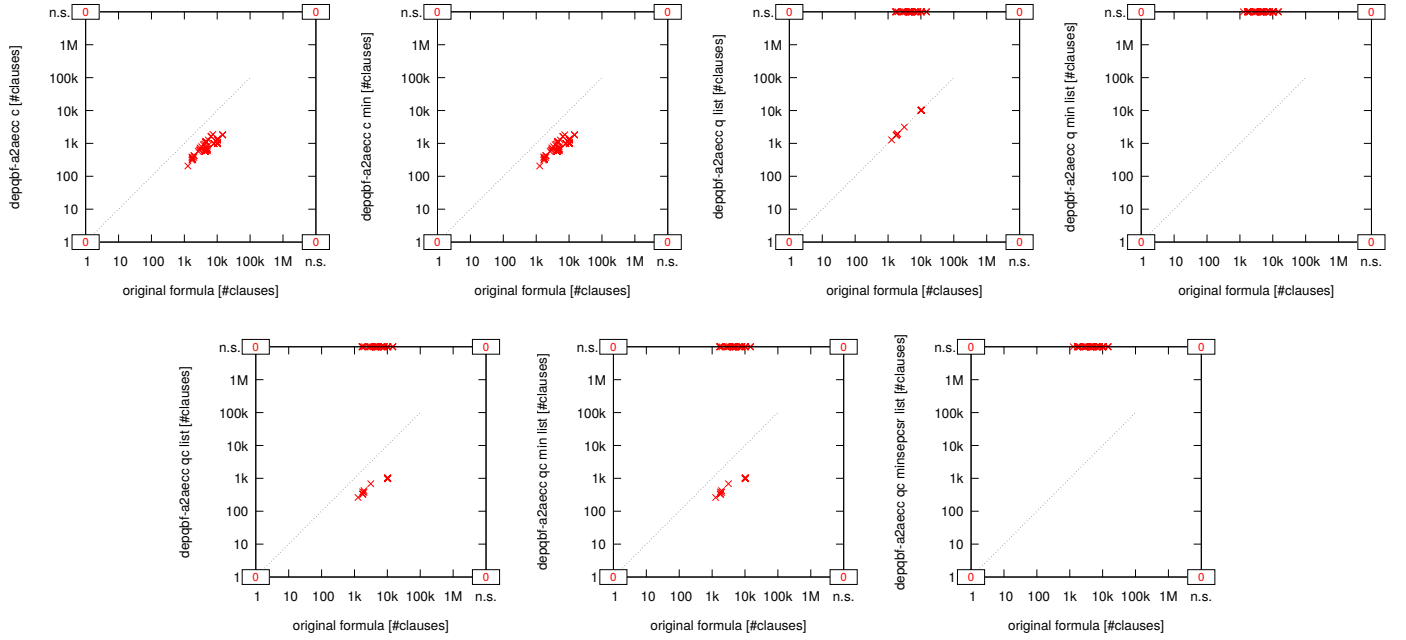


Fig. 146: Suite Kontchakov ($n = 70$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

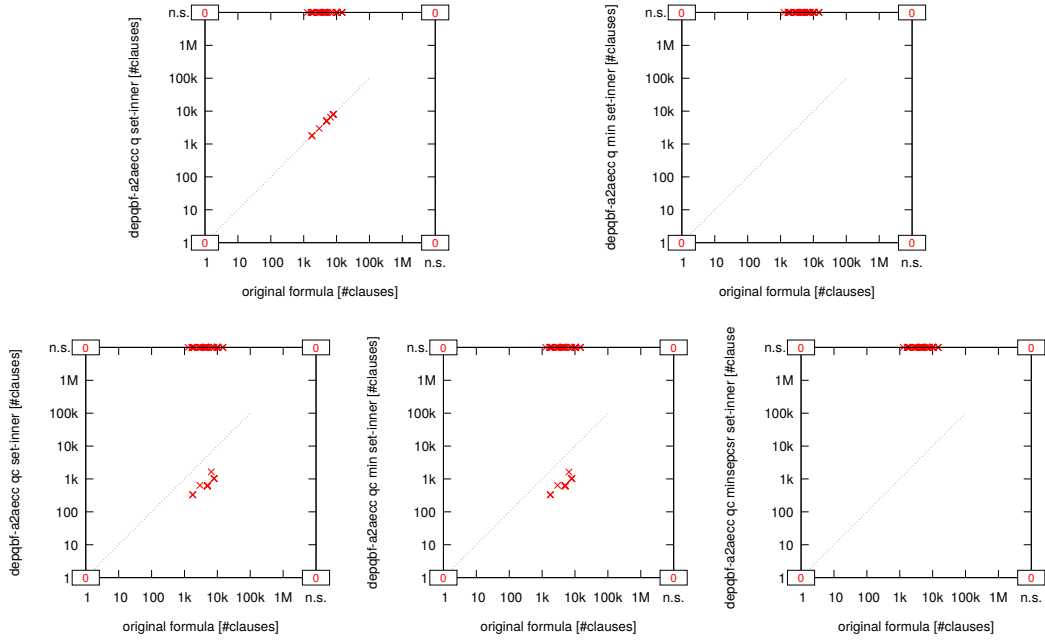


Fig. 147: Suite Kontchakov ($n = 70$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

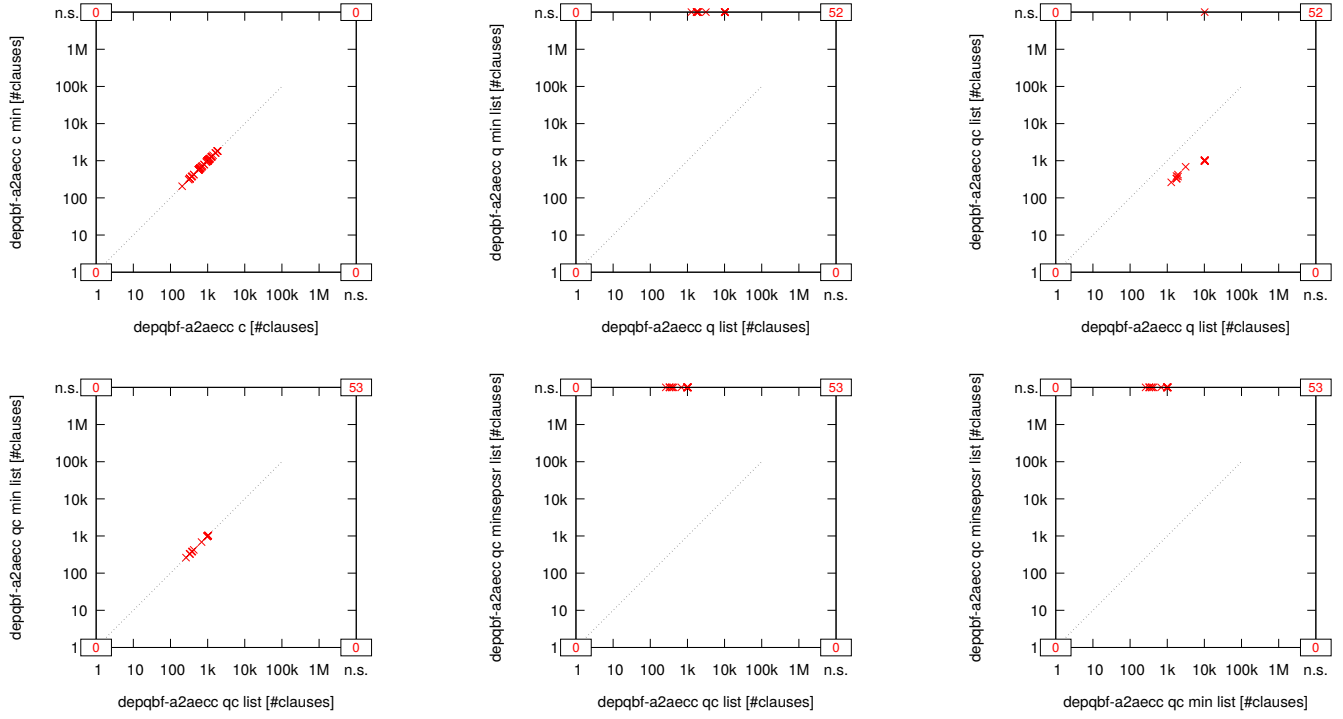


Fig. 148: Suite Kontchakov ($n = 70$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

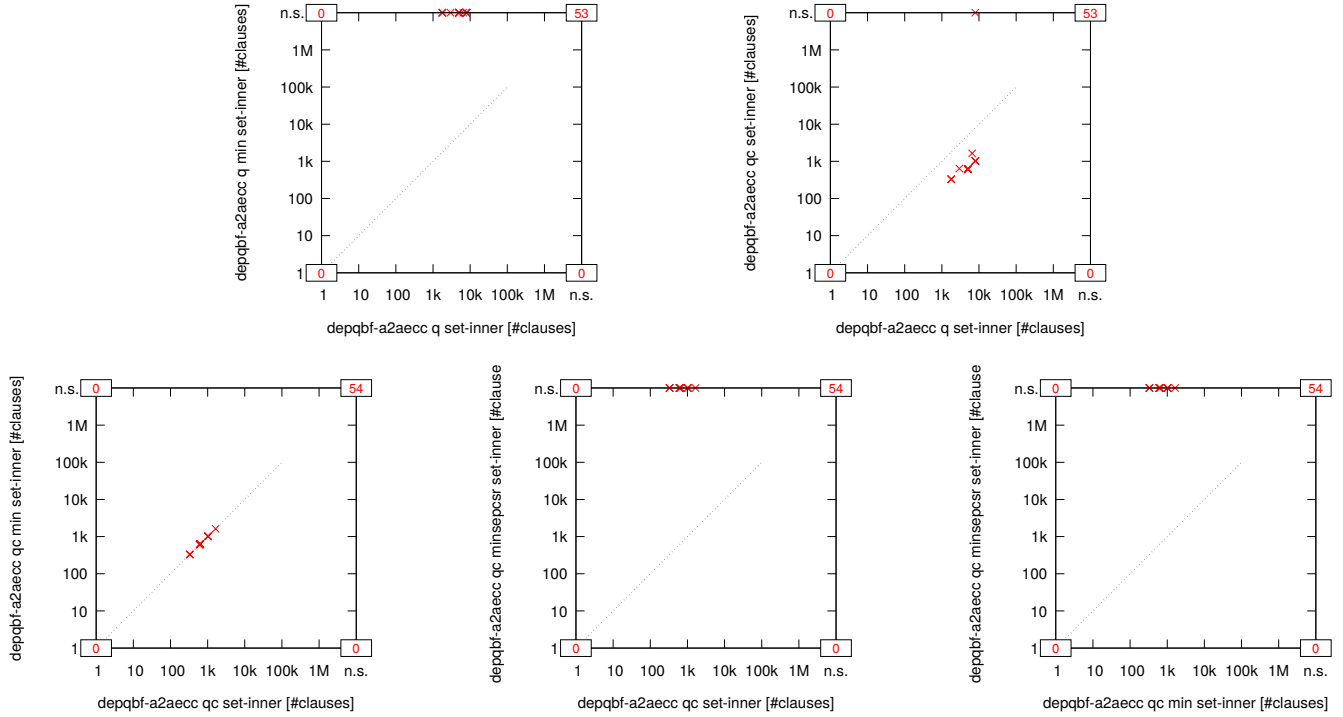


Fig. 149: Suite Kontchakov ($n = 70$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

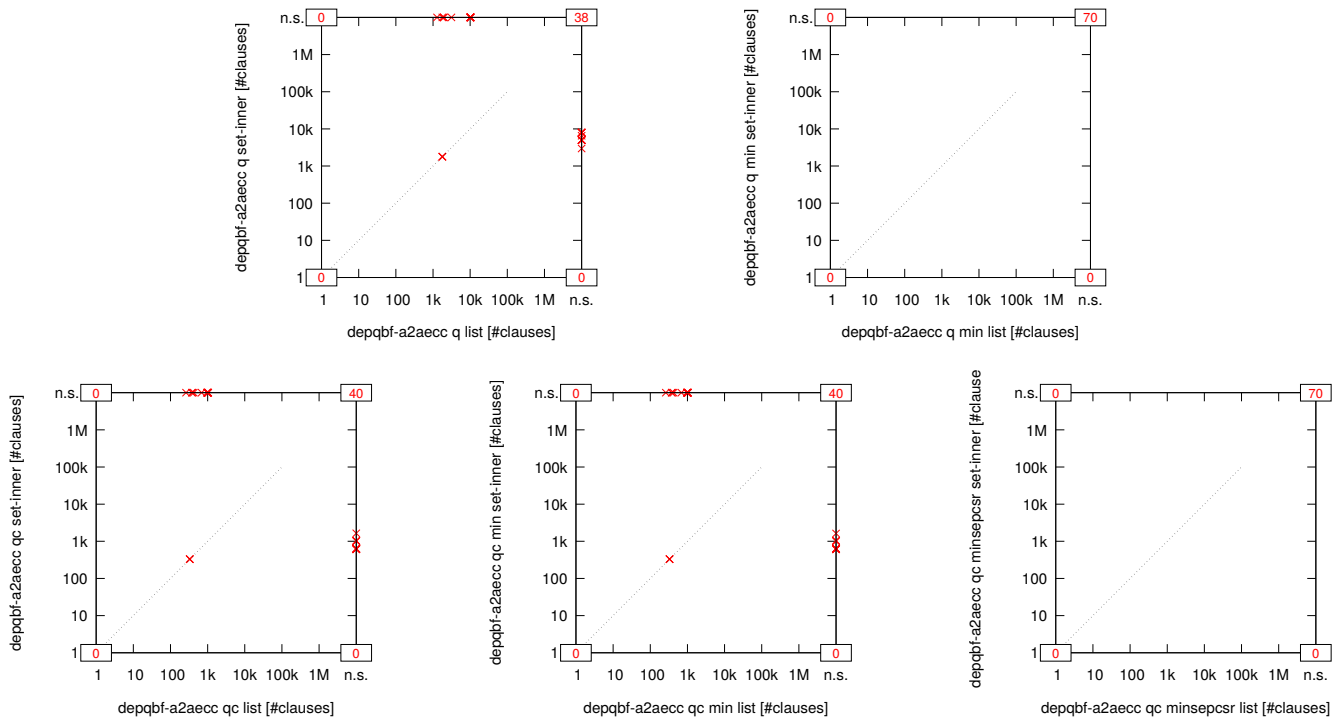


Fig. 150: Suite Kontchakov ($n = 70$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

21) Kronegger-Pfandler-Pichler ($n = 133$):

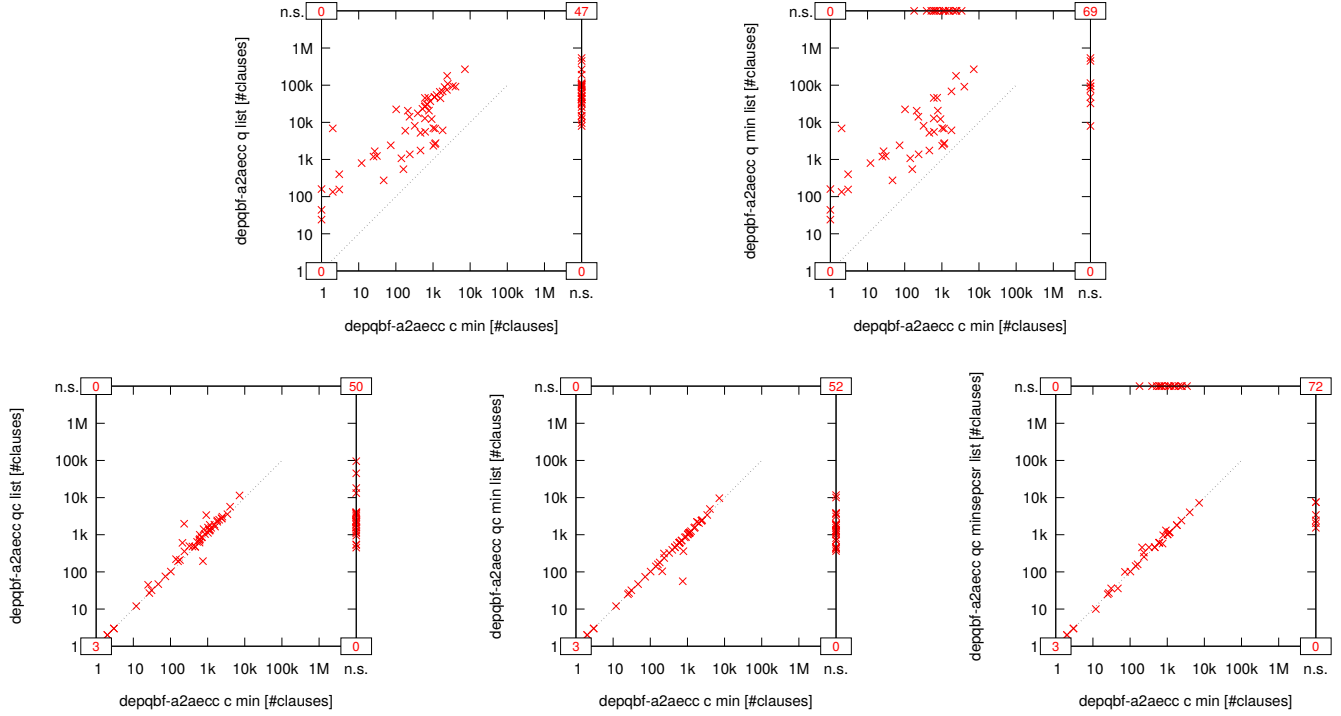


Fig. 151: Suite Kronegger-Pfandler-Pichler ($n = 133$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

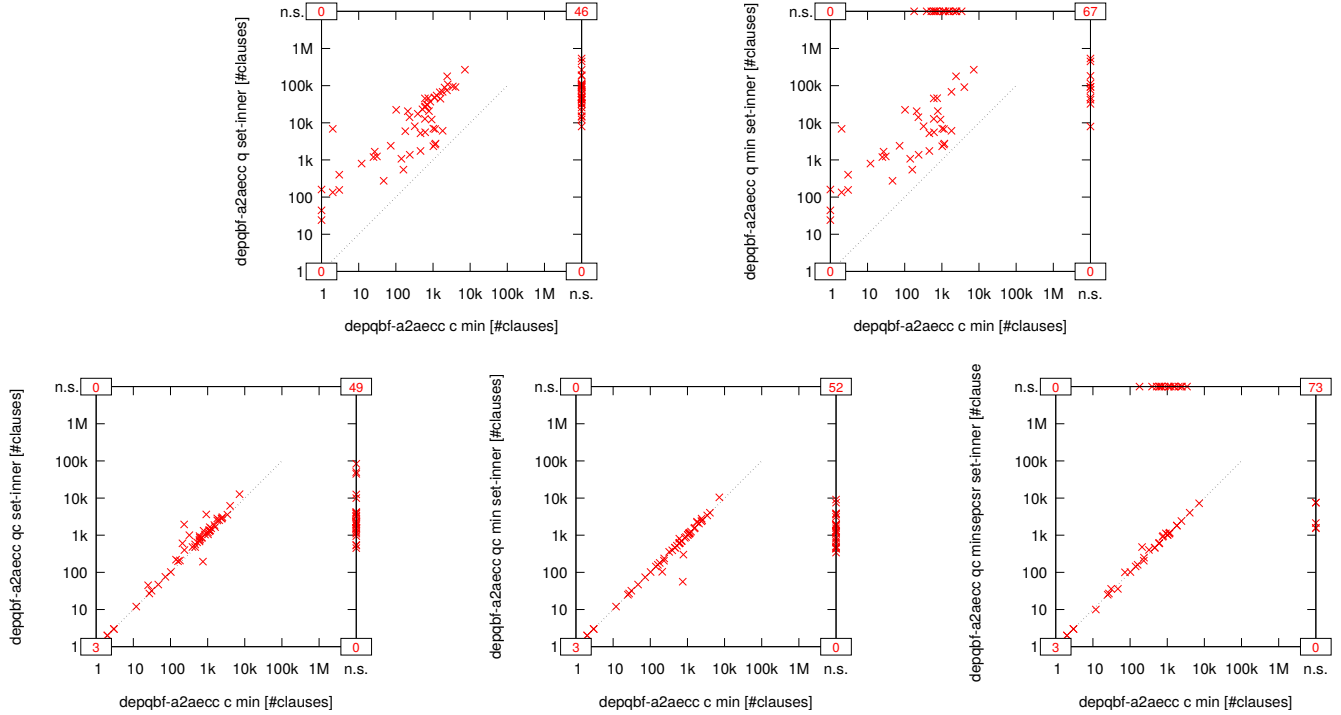


Fig. 152: Suite Kronegger-Pfandler-Pichler ($n = 133$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

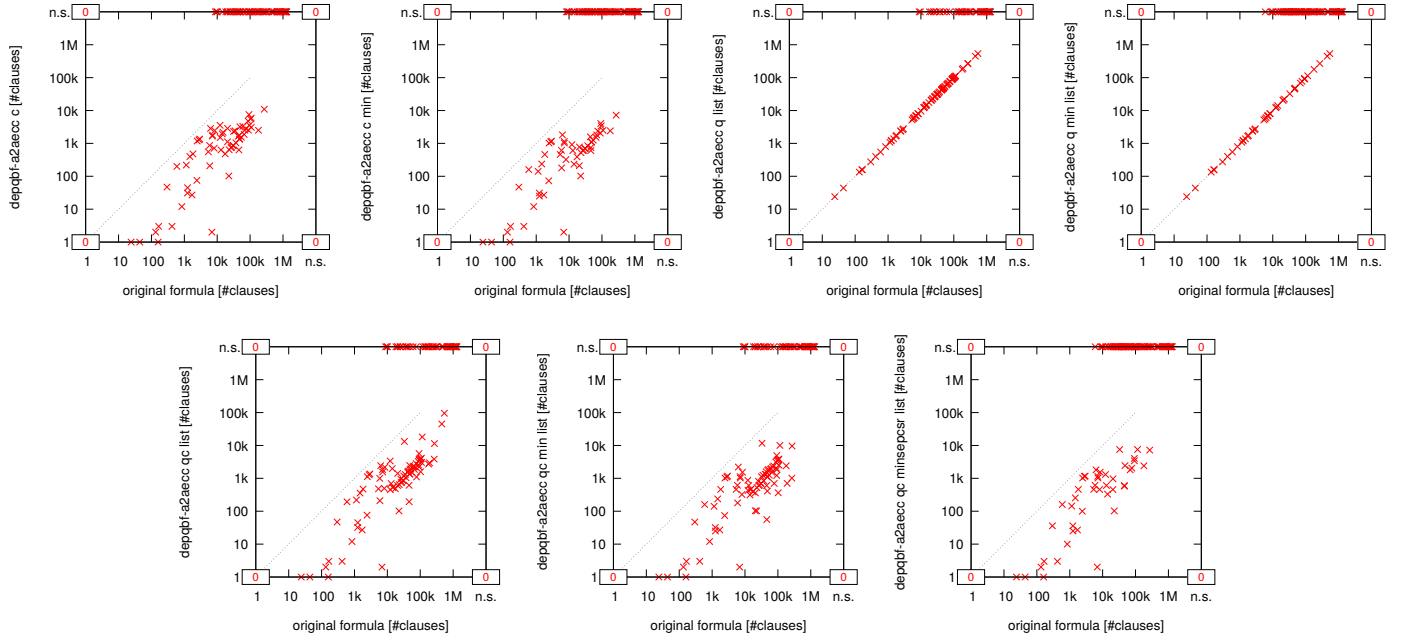


Fig. 153: Suite Kronegger-Pfandler-Pichler ($n = 133$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

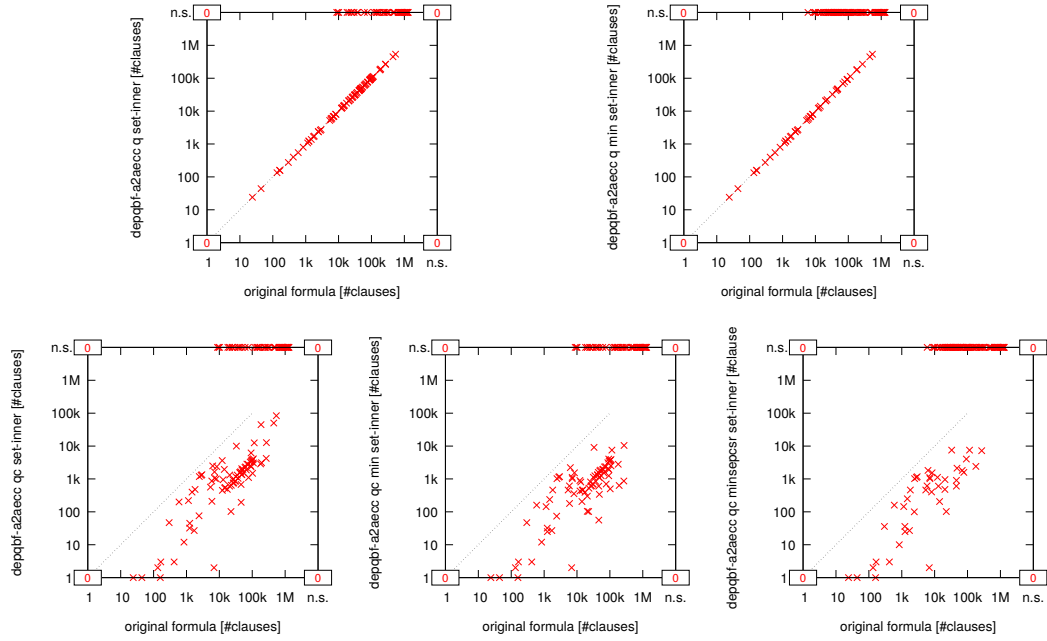


Fig. 154: Suite Kronegger-Pfandler-Pichler ($n = 133$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

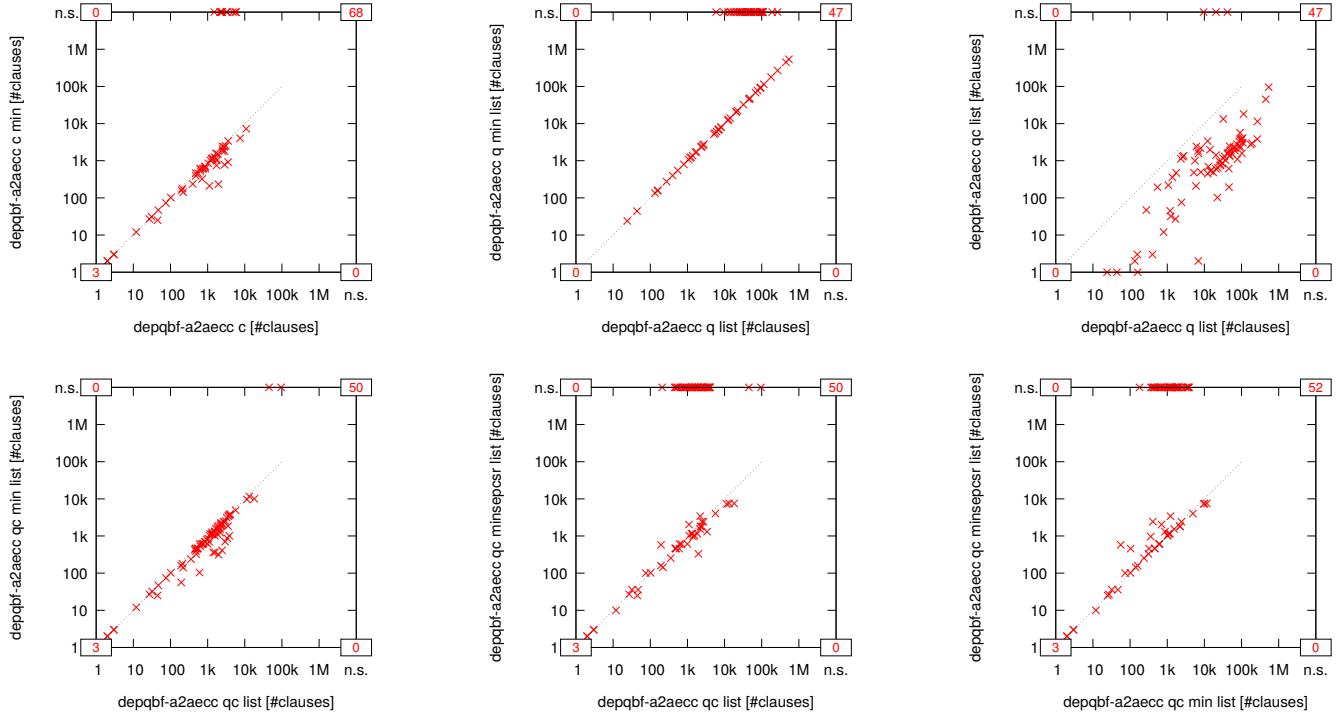


Fig. 155: Suite Kronegger-Pfandler-Pichler ($n = 133$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

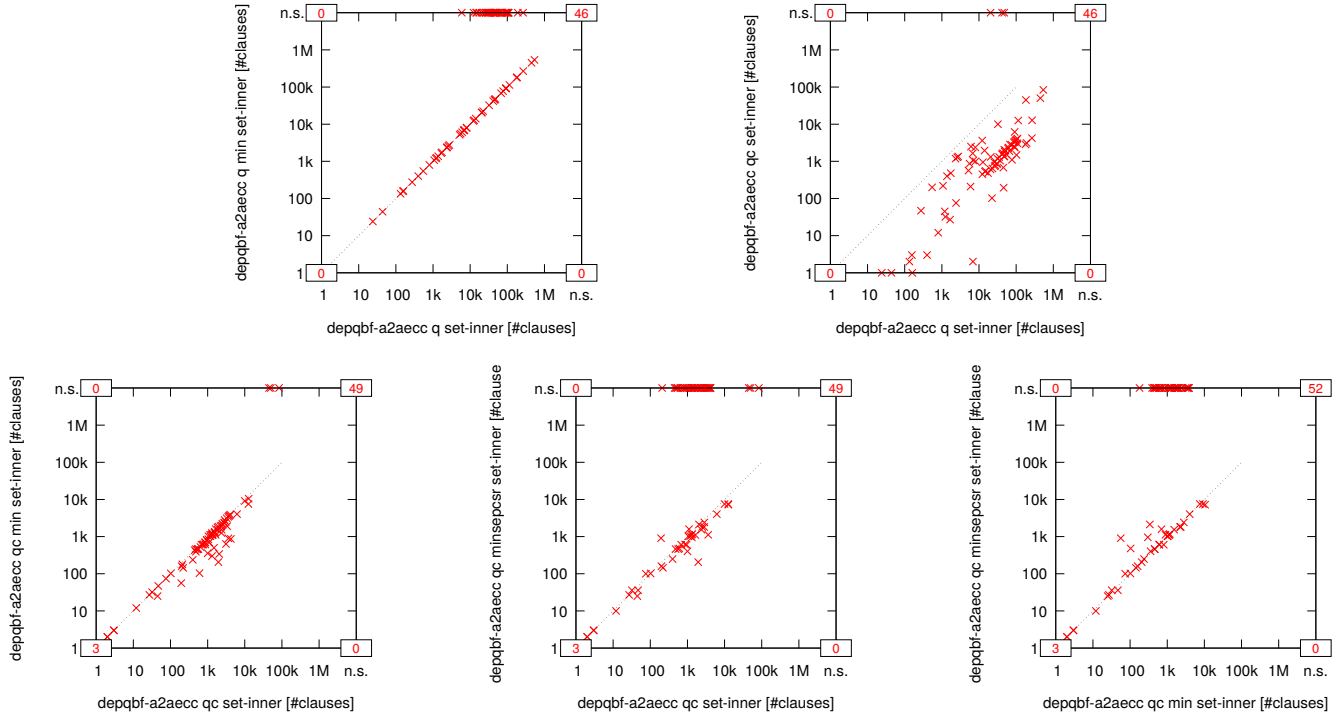


Fig. 156: Suite Kronegger-Pfandler-Pichler ($n = 133$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

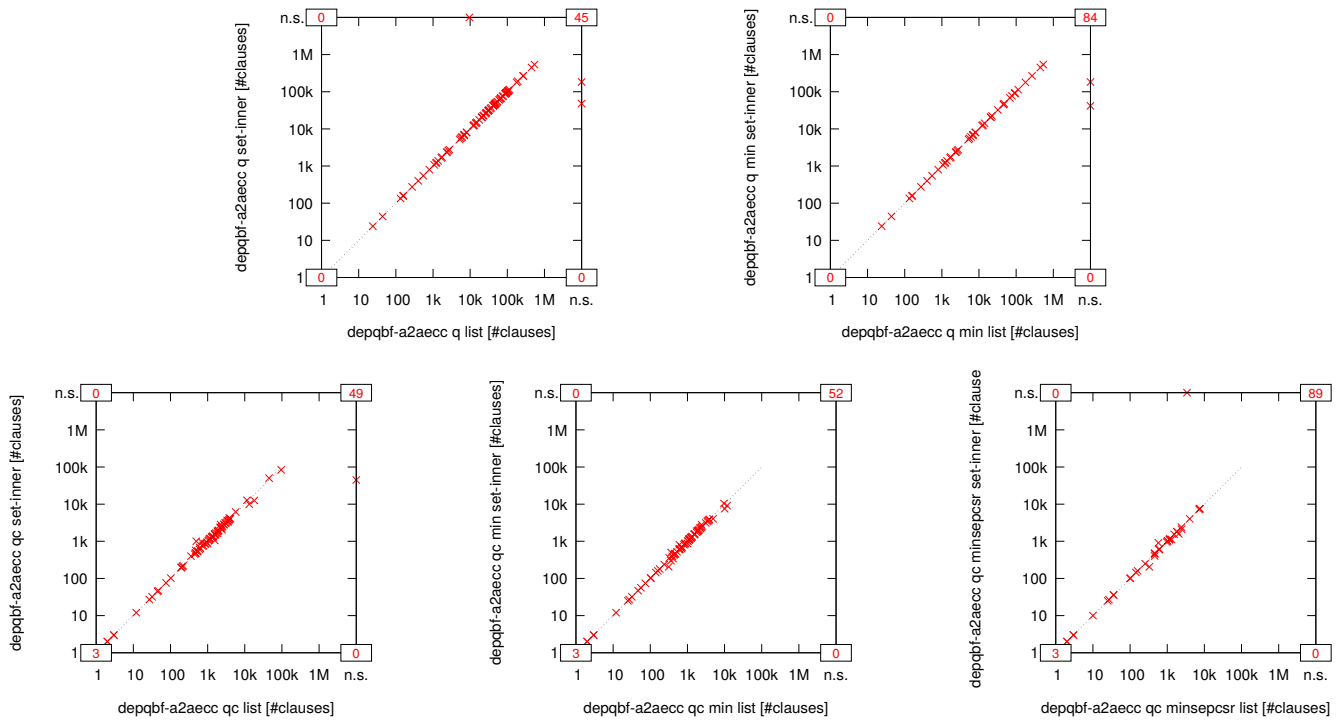


Fig. 157: Suite Kronegger-Pfandler-Pichler ($n = 133$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

22) Lahiri-Seshia ($n = 1$):

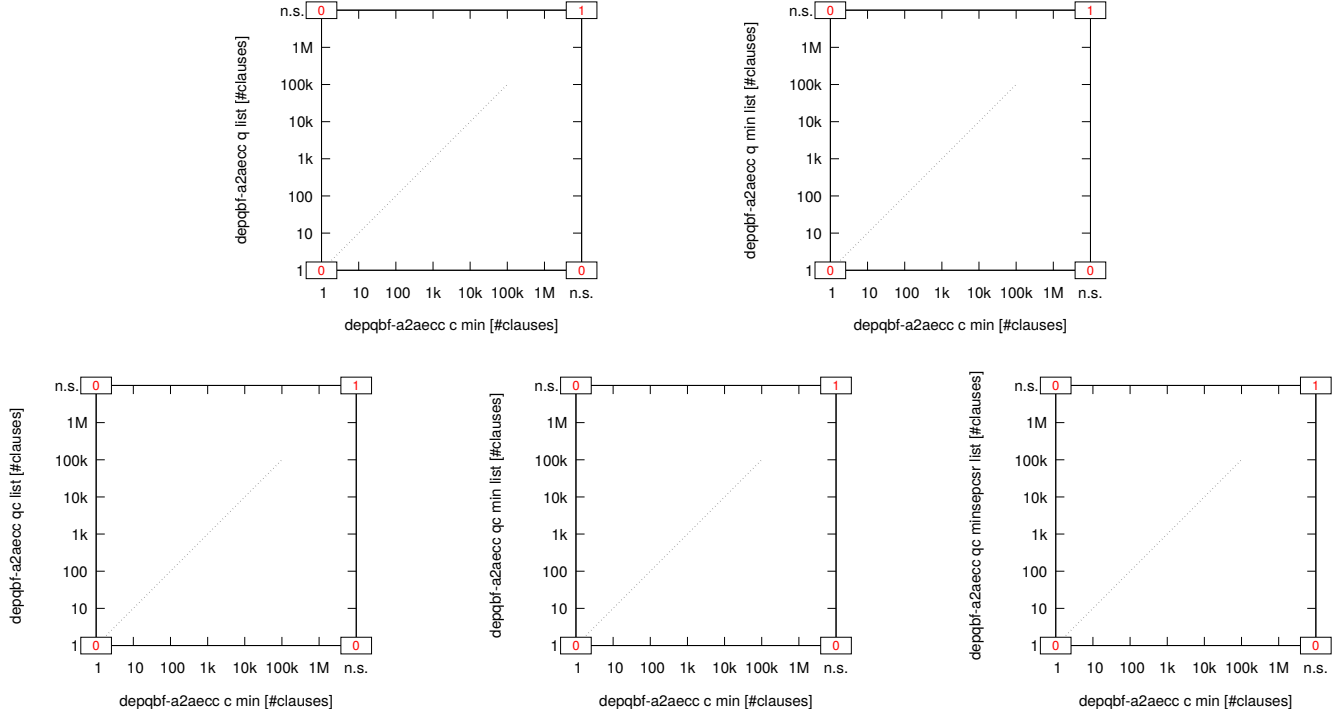


Fig. 158: Suite Lahiri-Seshia ($n = 1$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

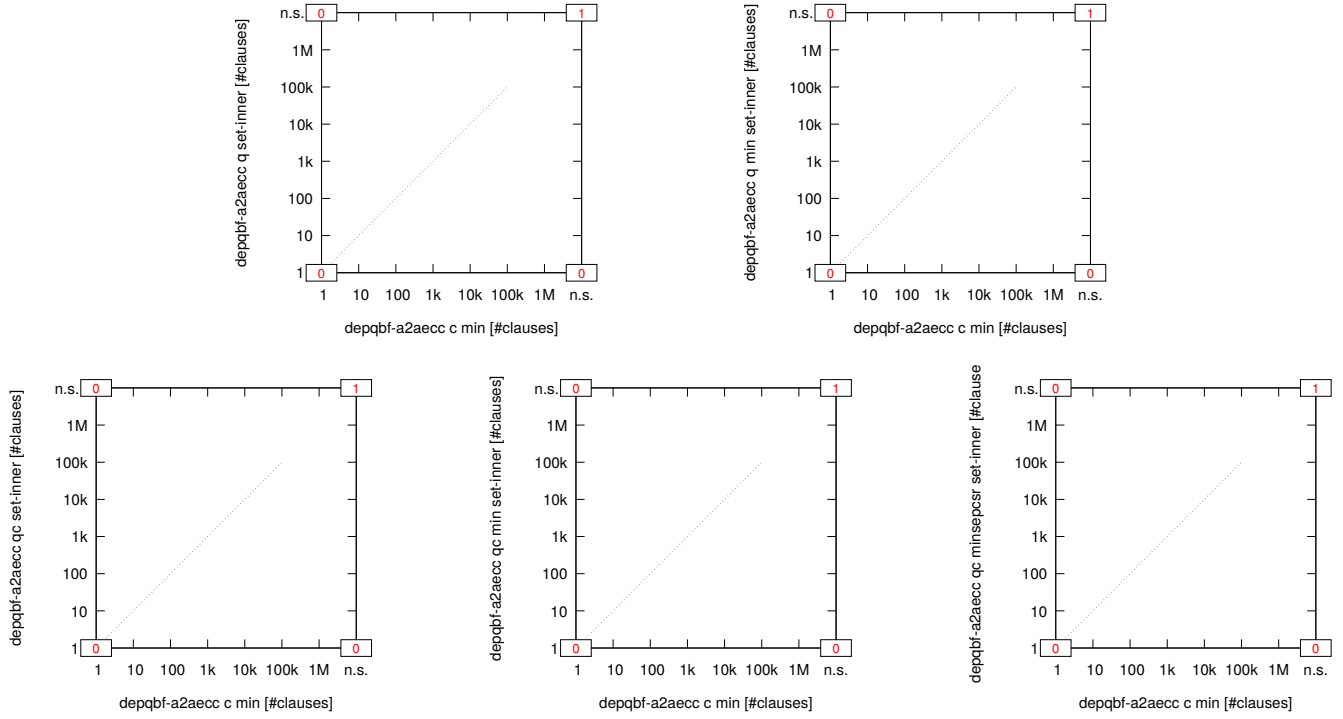


Fig. 159: Suite Lahiri-Seshia ($n = 1$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

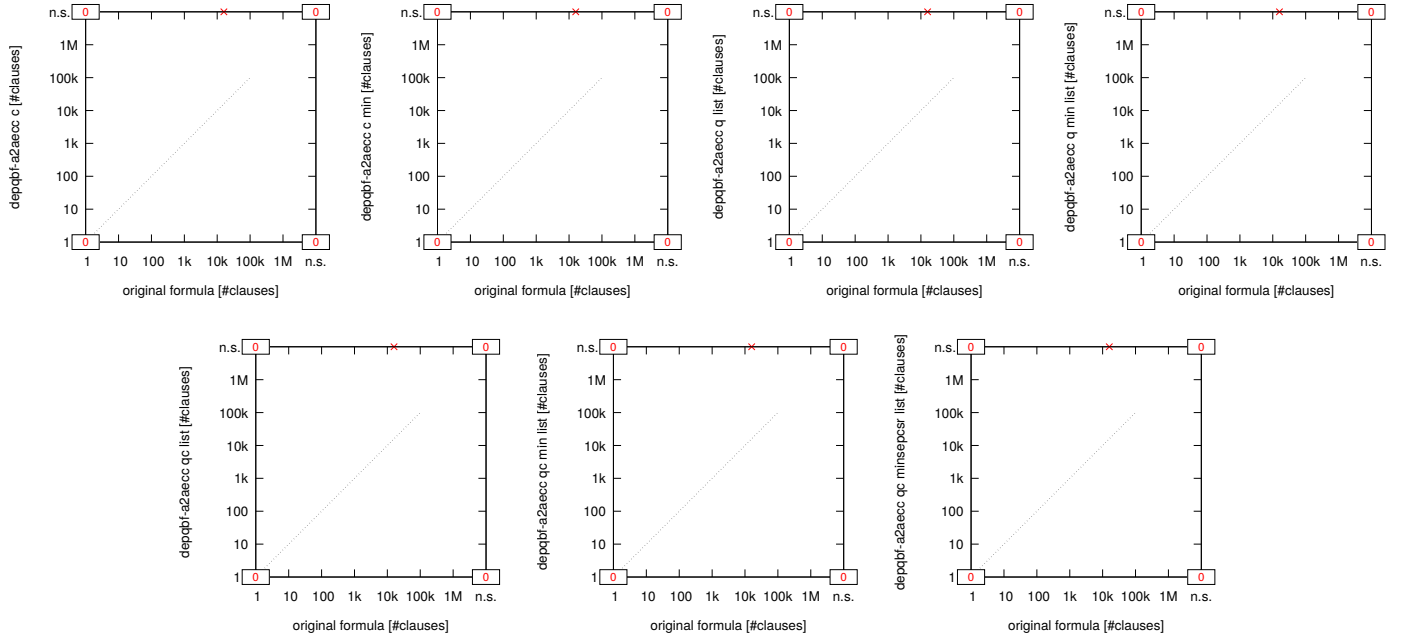


Fig. 160: Suite Lahiri-Seshia ($n = 1$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

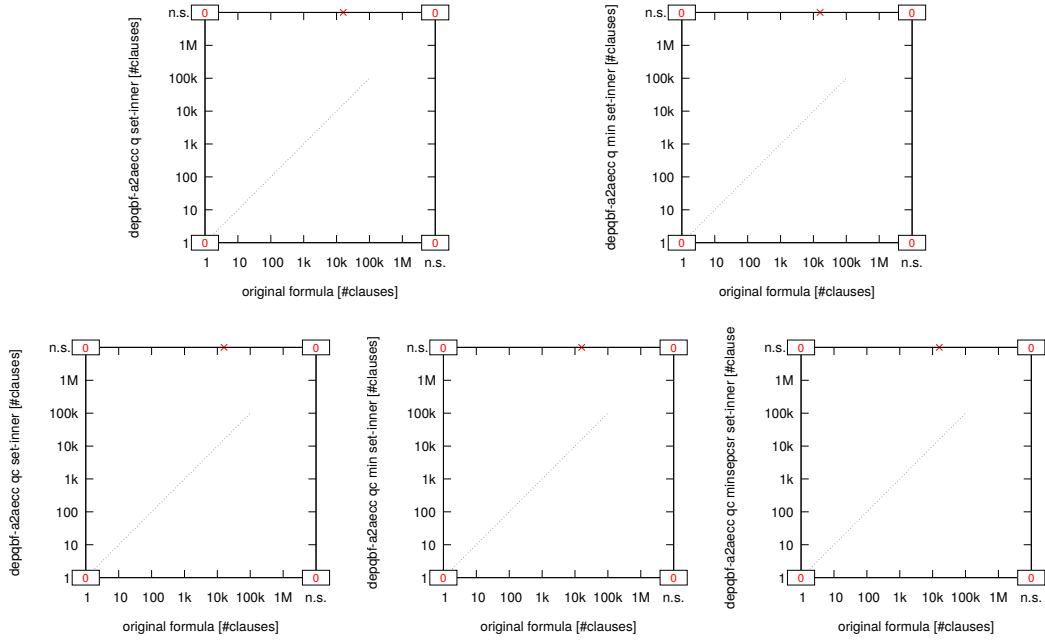


Fig. 161: Suite Lahiri-Seshia ($n = 1$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

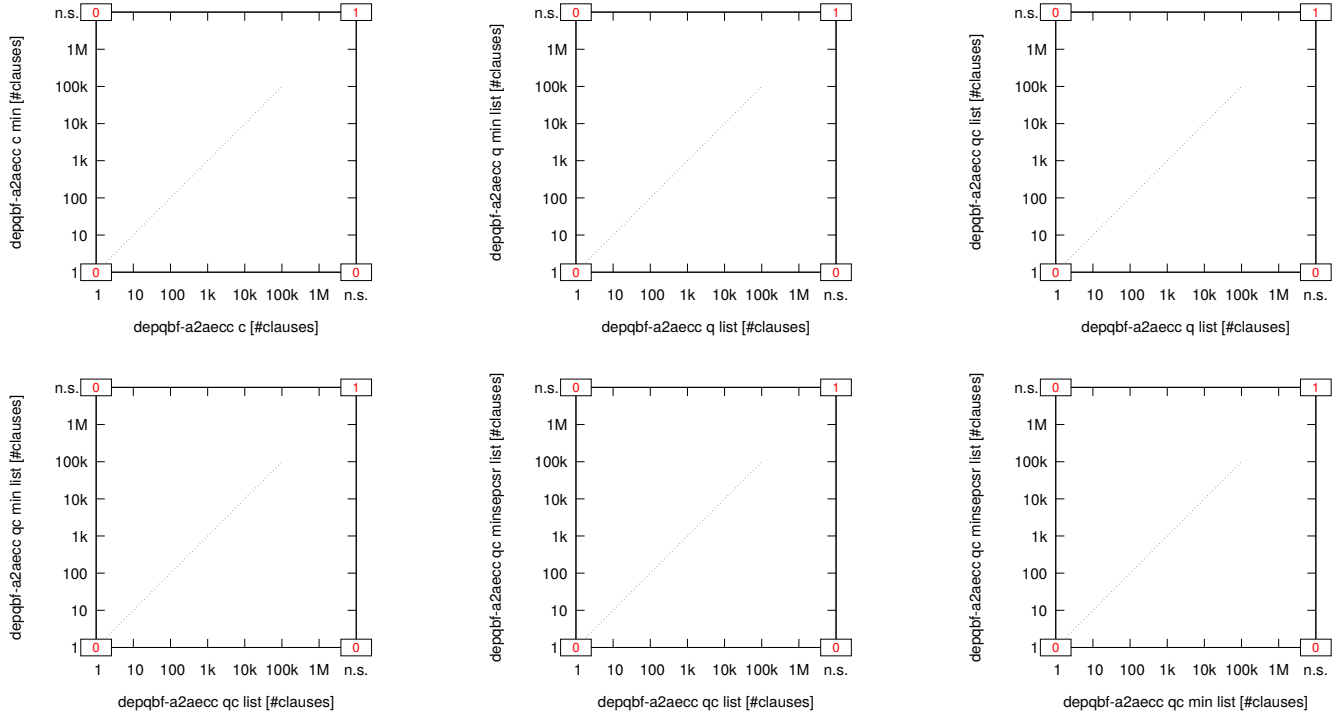


Fig. 162: Suite Lahiri-Seshia ($n = 1$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

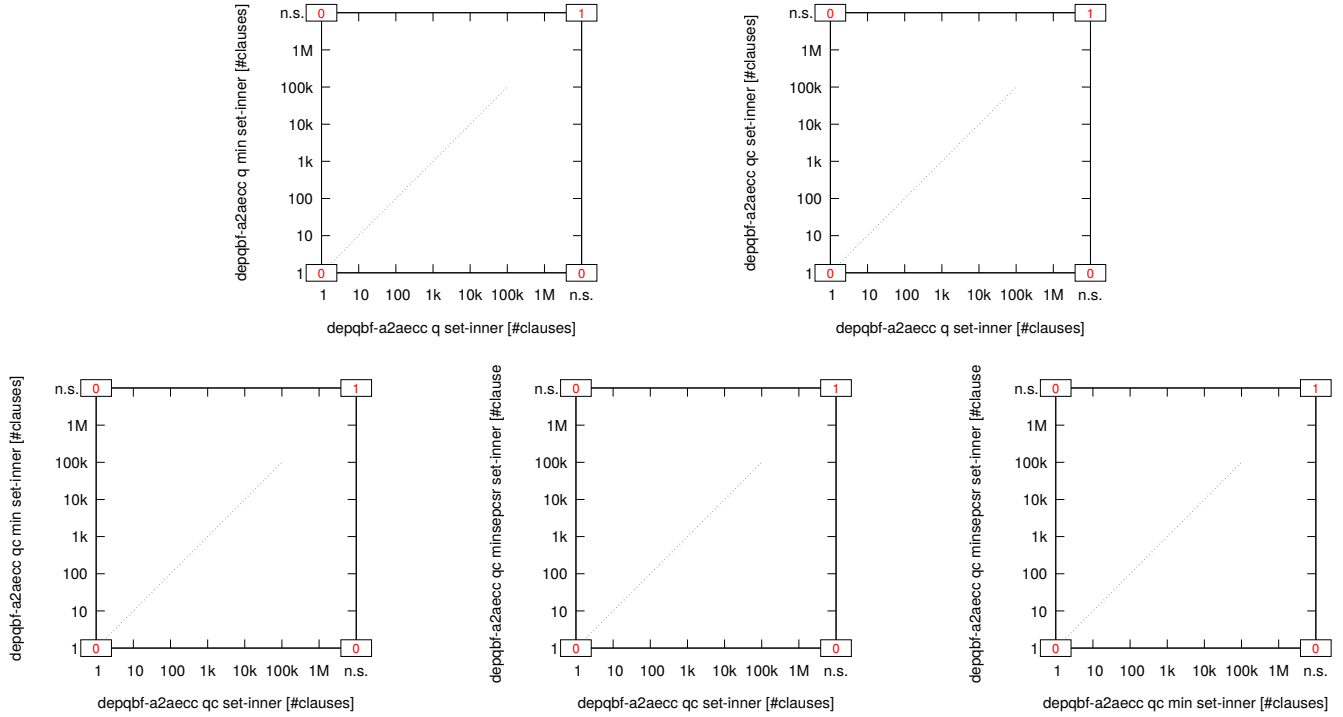


Fig. 163: Suite Lahiri-Seshia ($n = 1$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

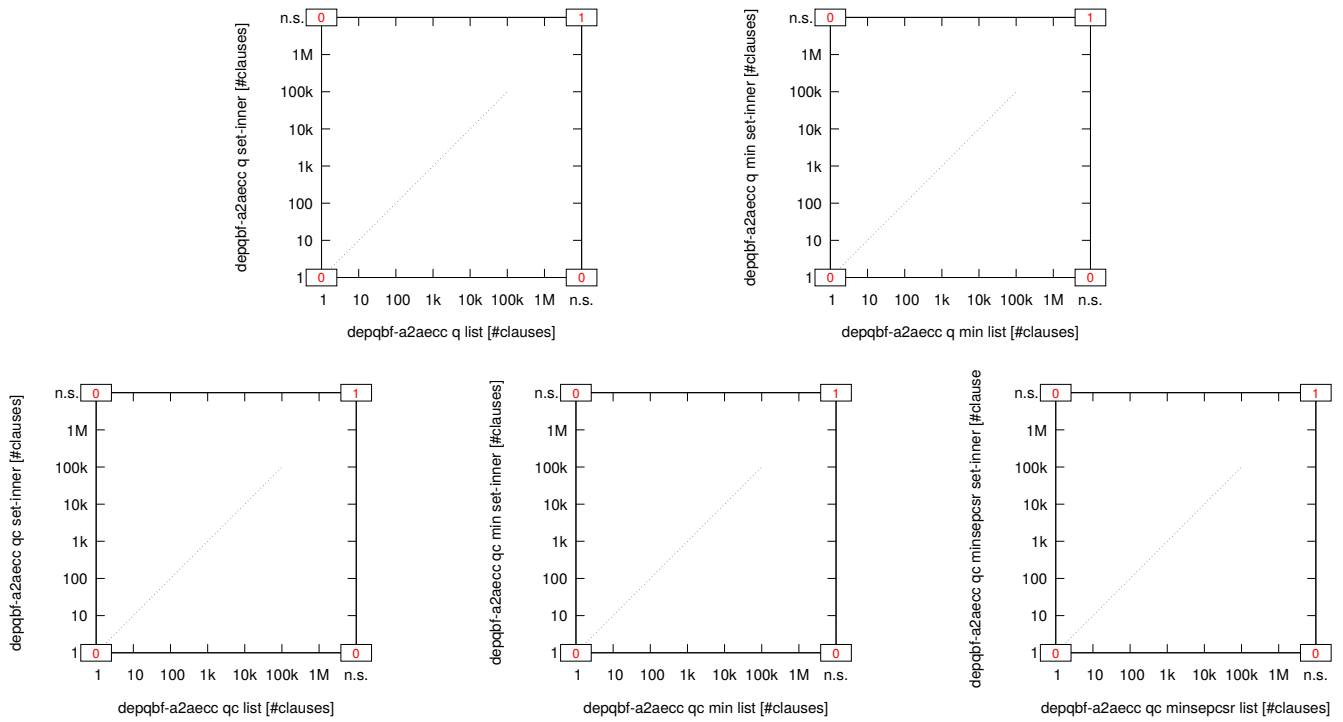


Fig. 164: Suite Lahiri-Seshia ($n = 1$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

23) *Lee-Jiang* ($n = 2$):

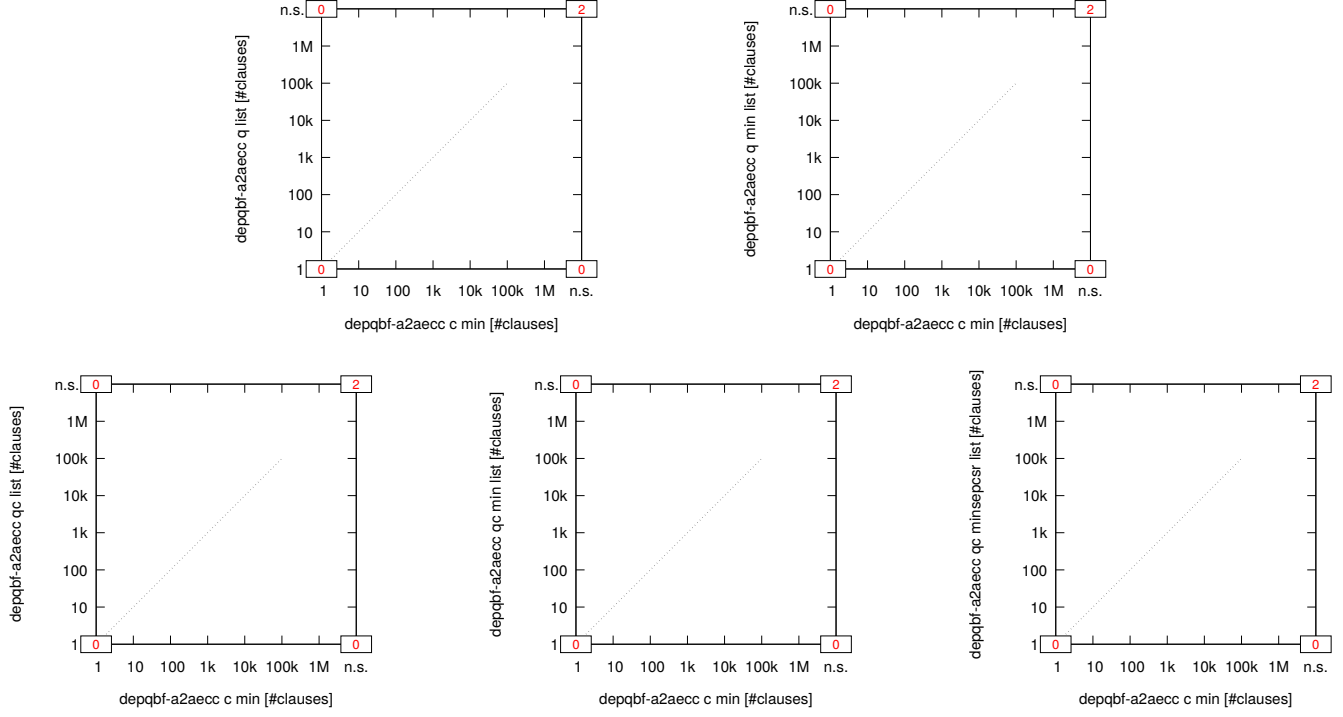


Fig. 165: Suite Lee-Jiang ($n = 2$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

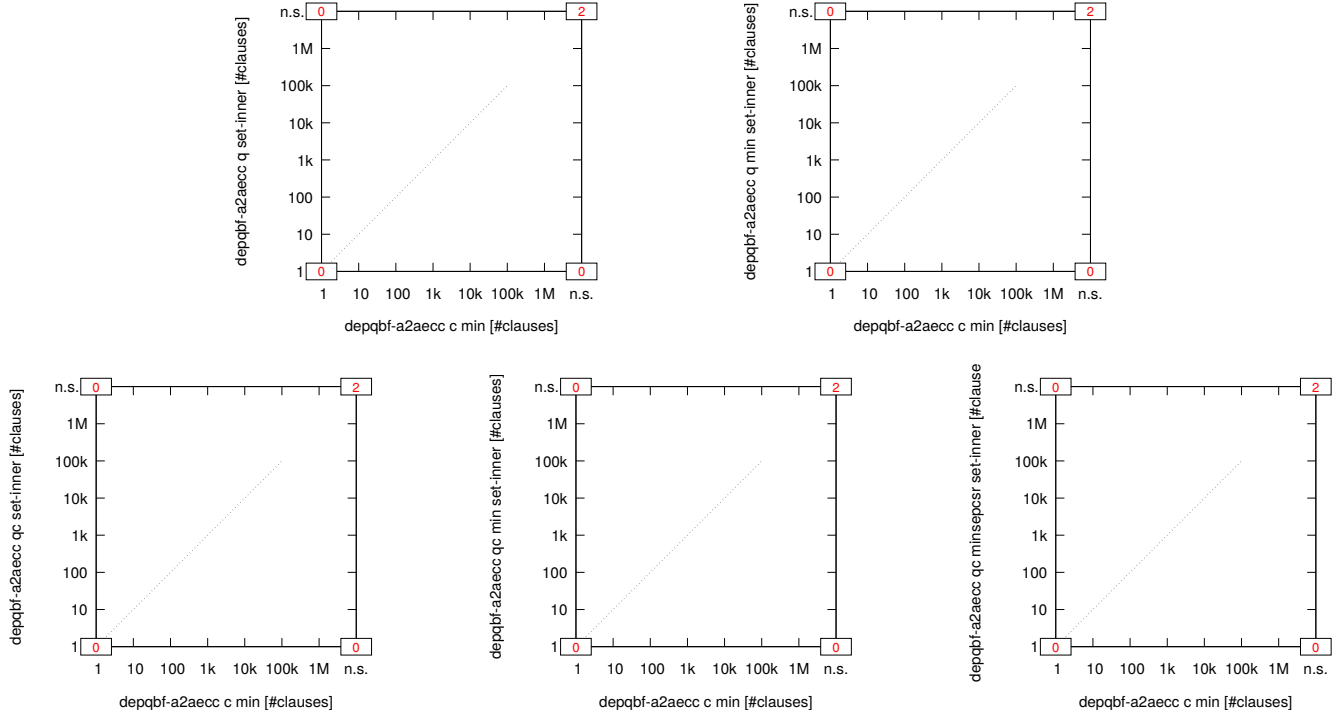


Fig. 166: Suite Lee-Jiang ($n = 2$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

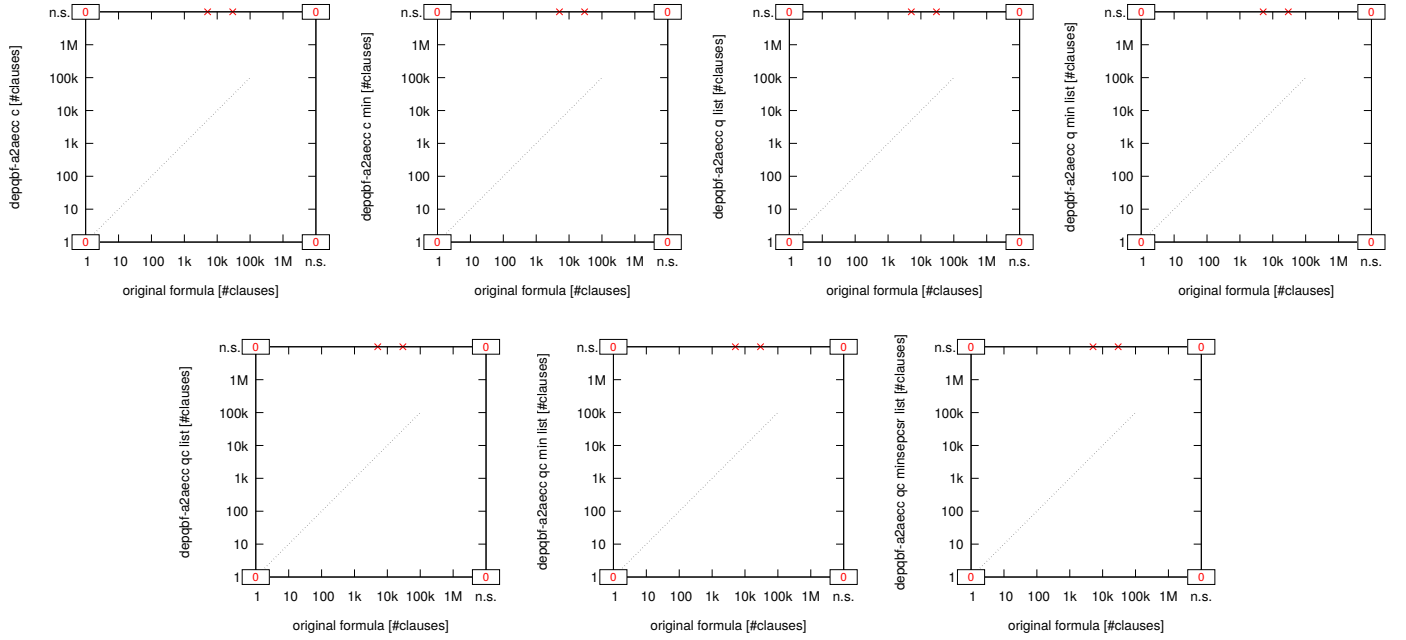


Fig. 167: Suite Lee-Jiang ($n = 2$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

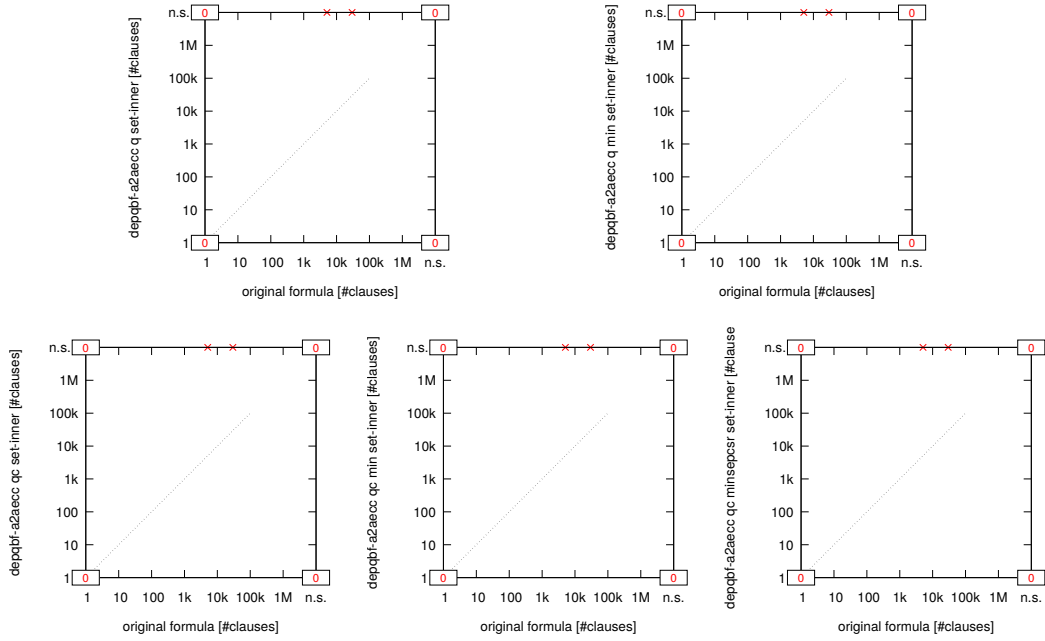


Fig. 168: Suite Lee-Jiang ($n = 2$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

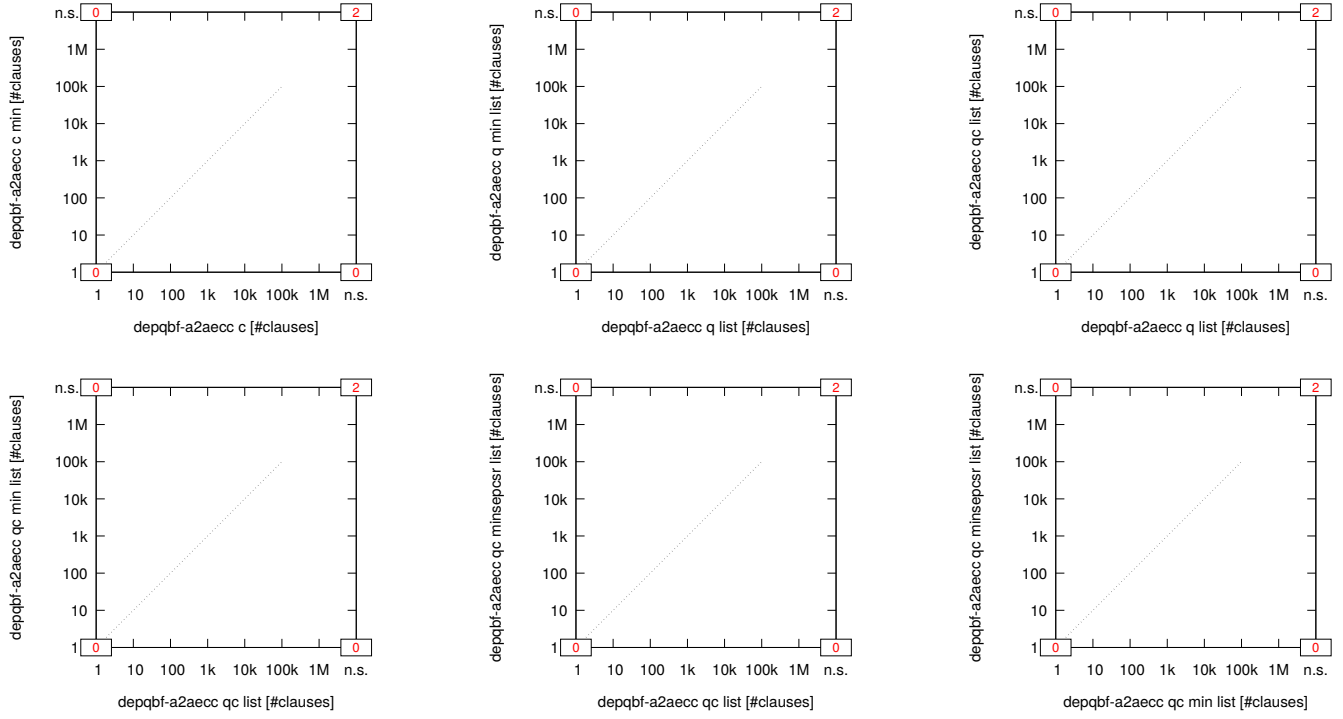


Fig. 169: Suite Lee-Jiang ($n = 2$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

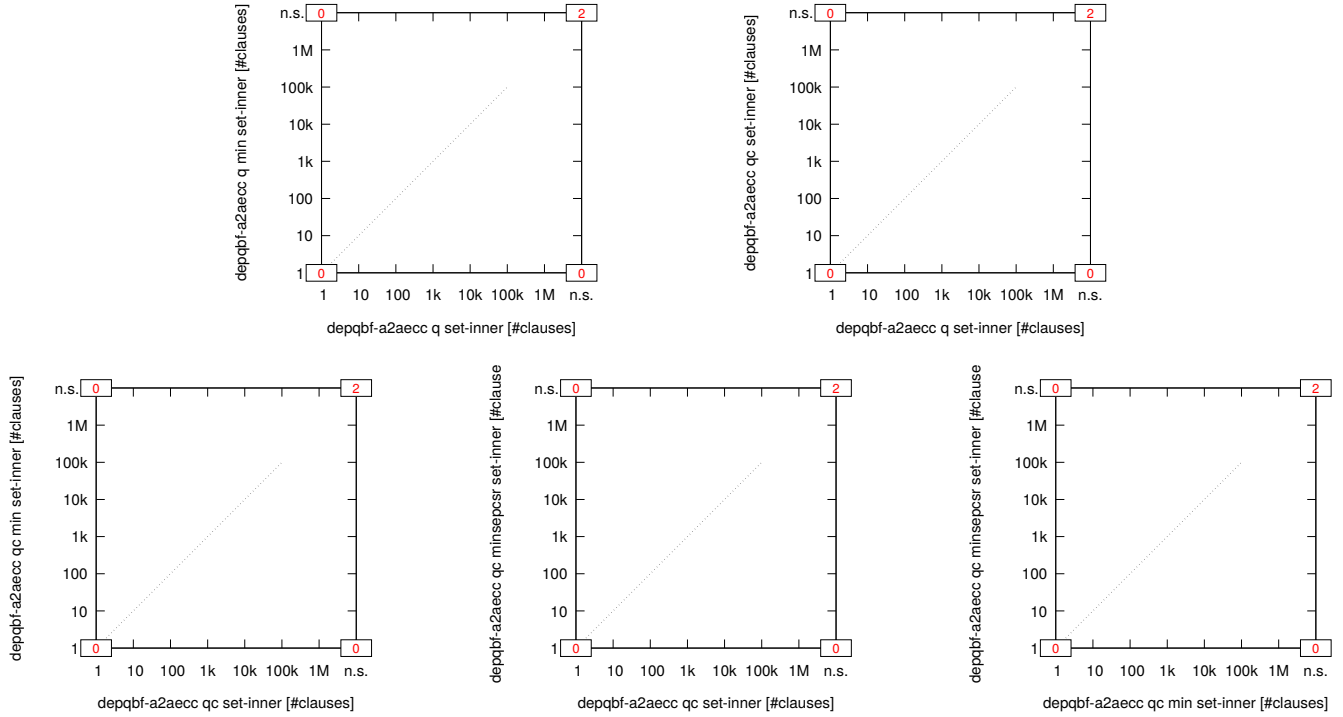


Fig. 170: Suite Lee-Jiang ($n = 2$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

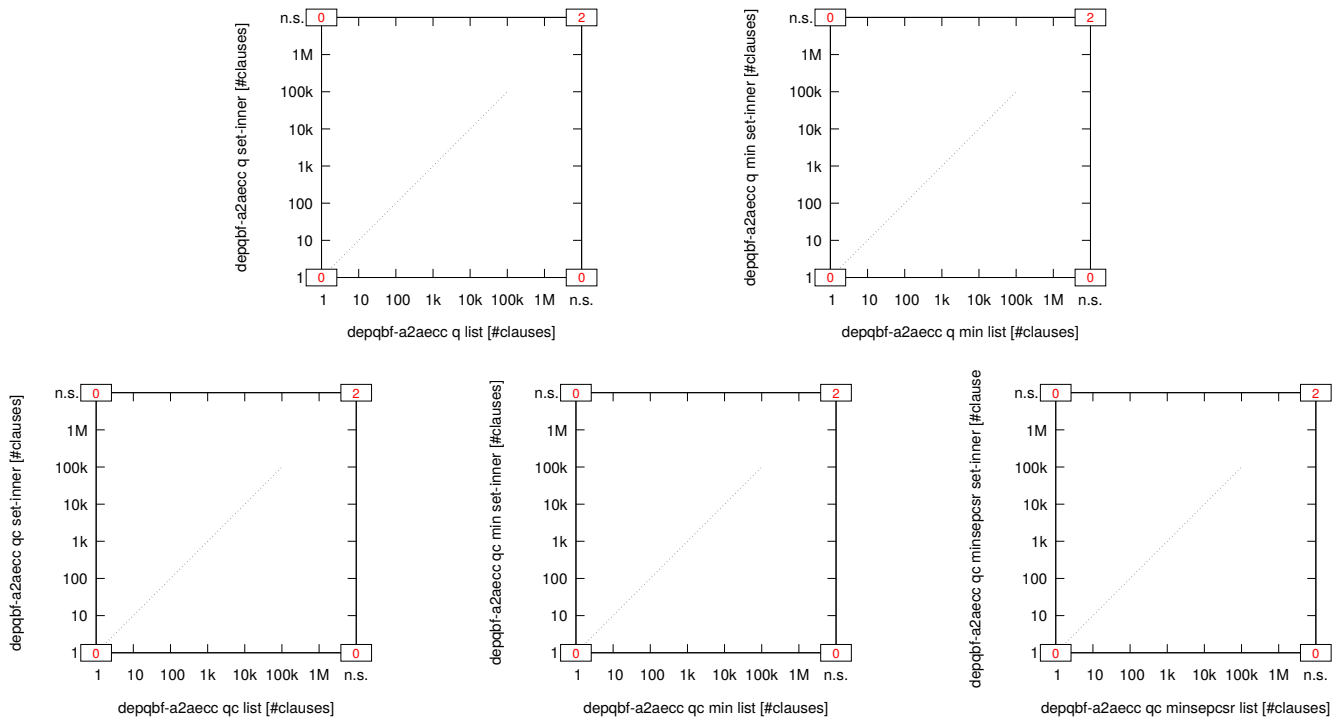


Fig. 171: Suite Lee-Jiang ($n = 2$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

24) *Letombe* ($n = 85$):

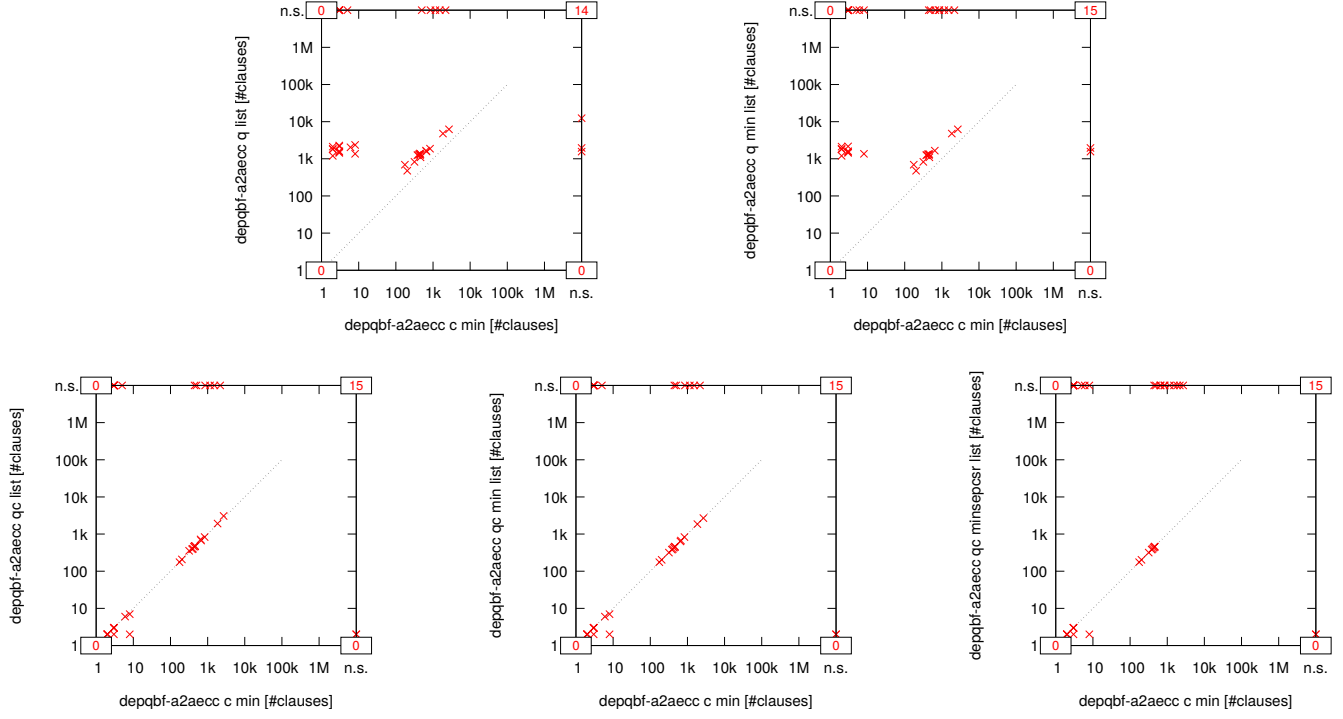


Fig. 172: Suite Letombe ($n = 85$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

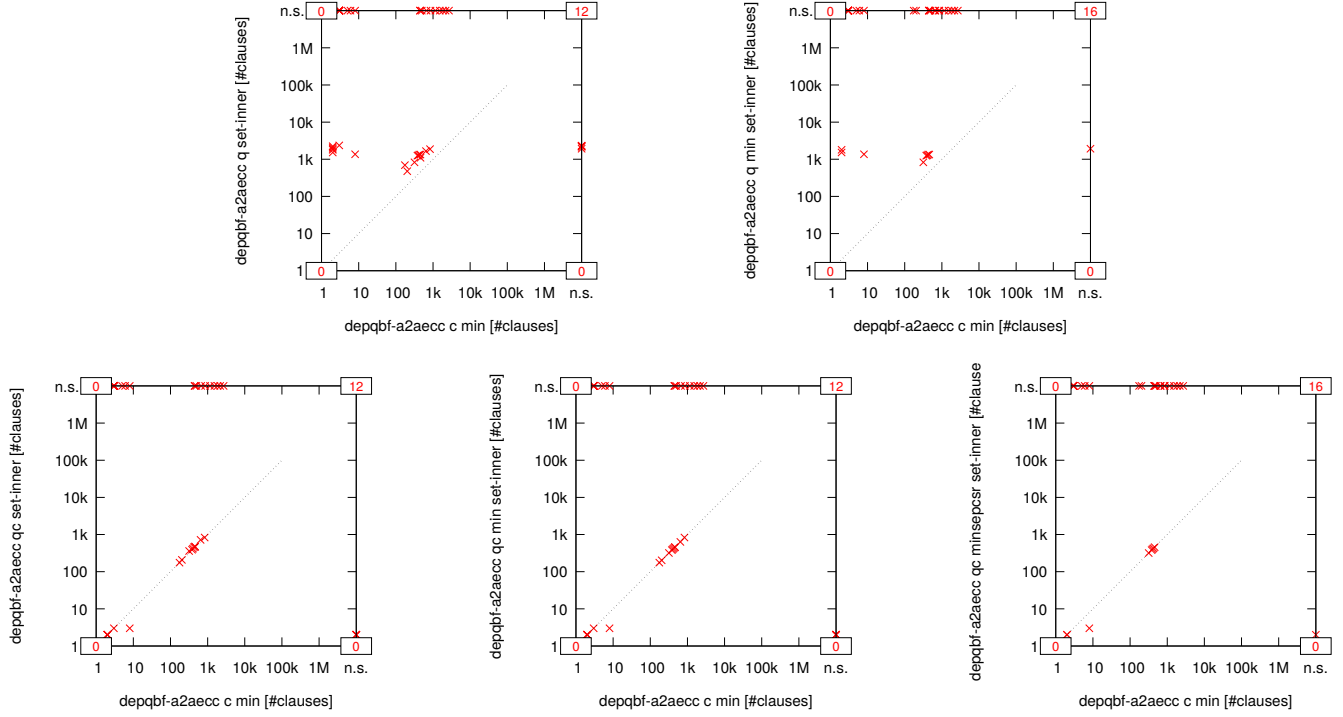


Fig. 173: Suite Letombe ($n = 85$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

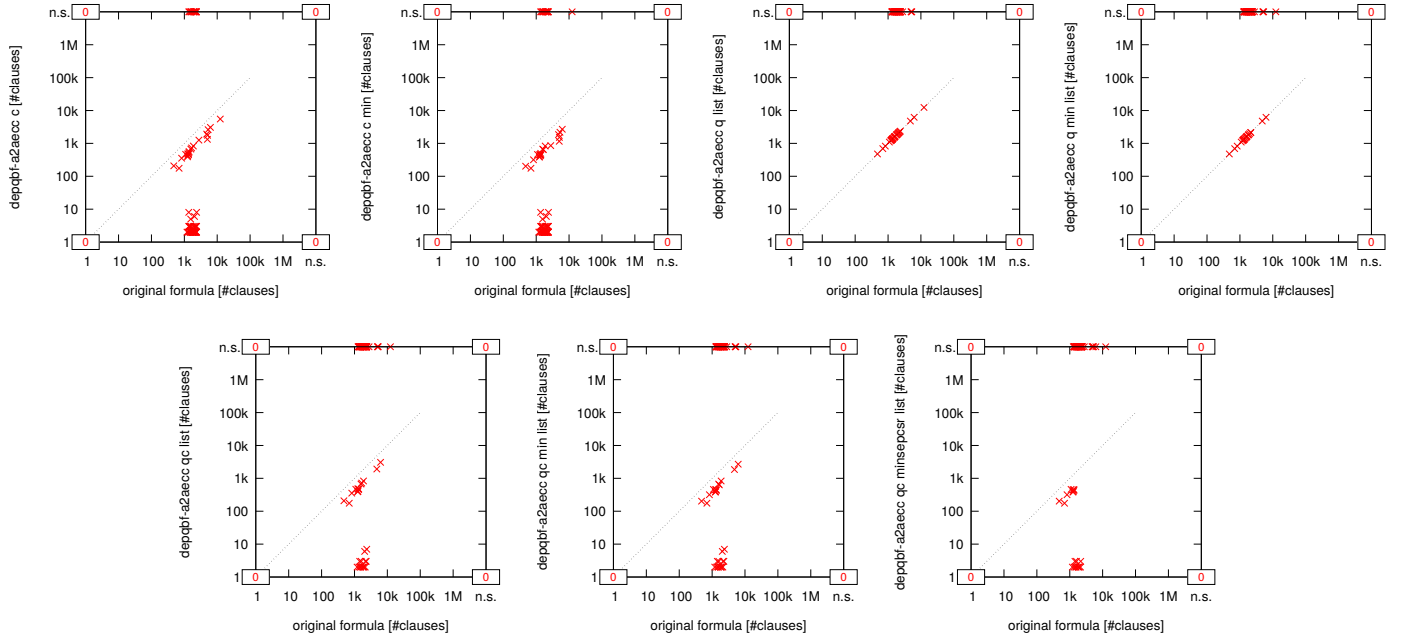


Fig. 174: Suite Letombe ($n = 85$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

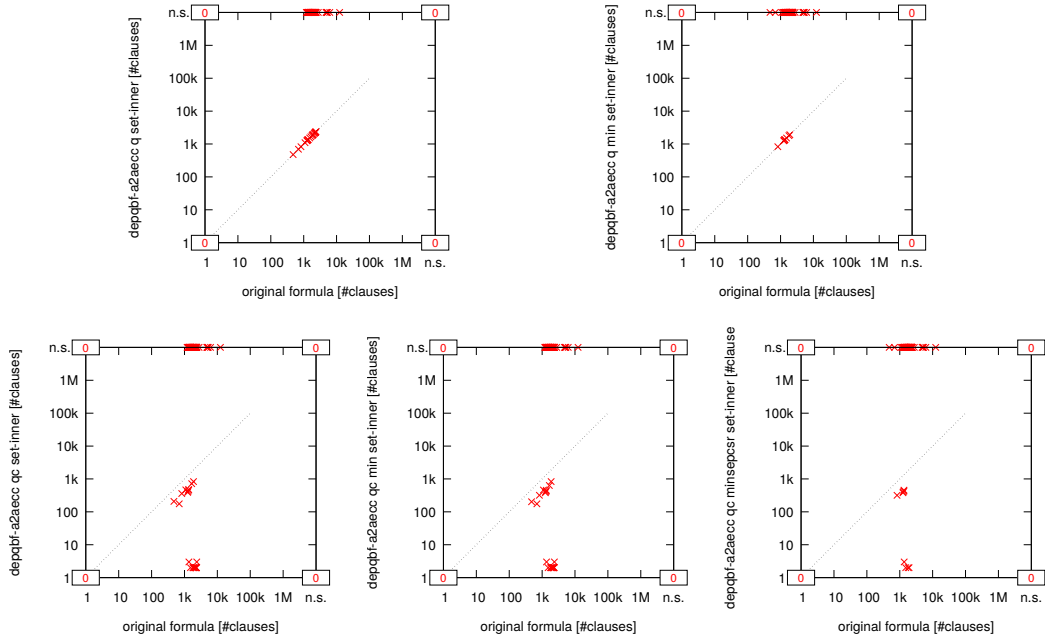


Fig. 175: Suite Letombe ($n = 85$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

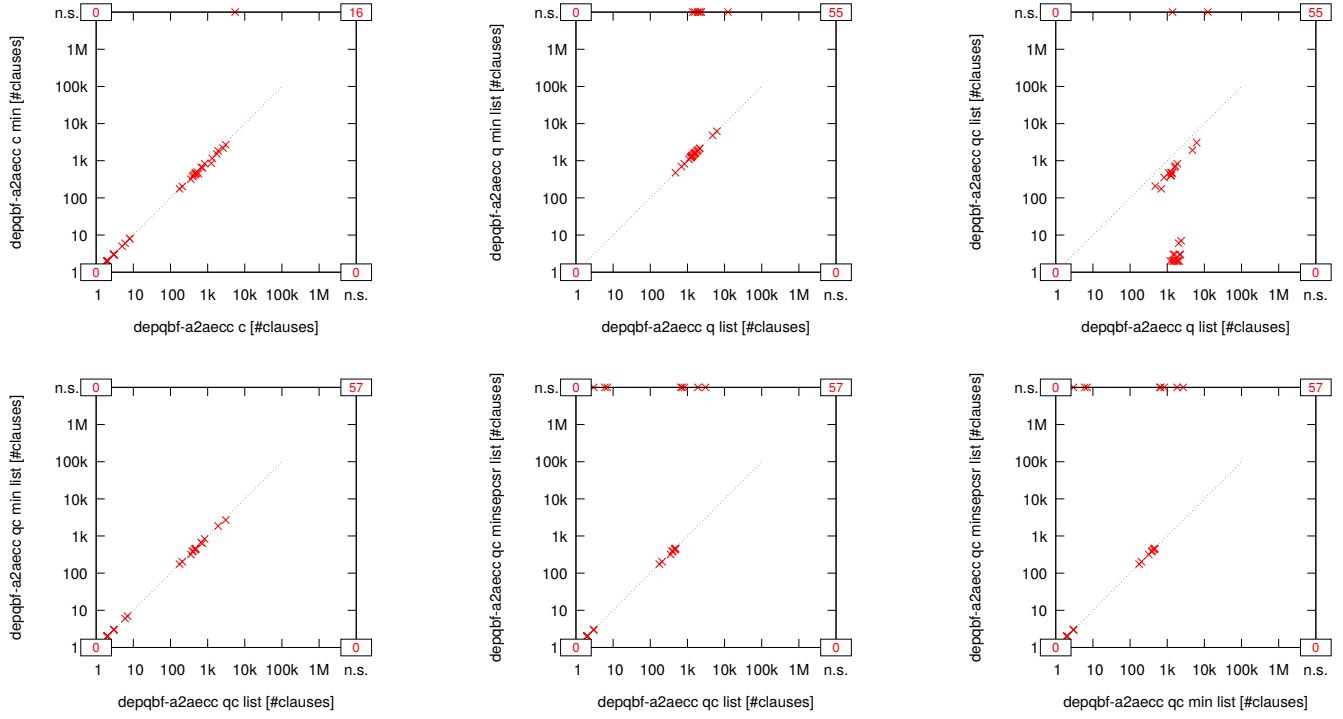


Fig. 176: Suite Letombe ($n = 85$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

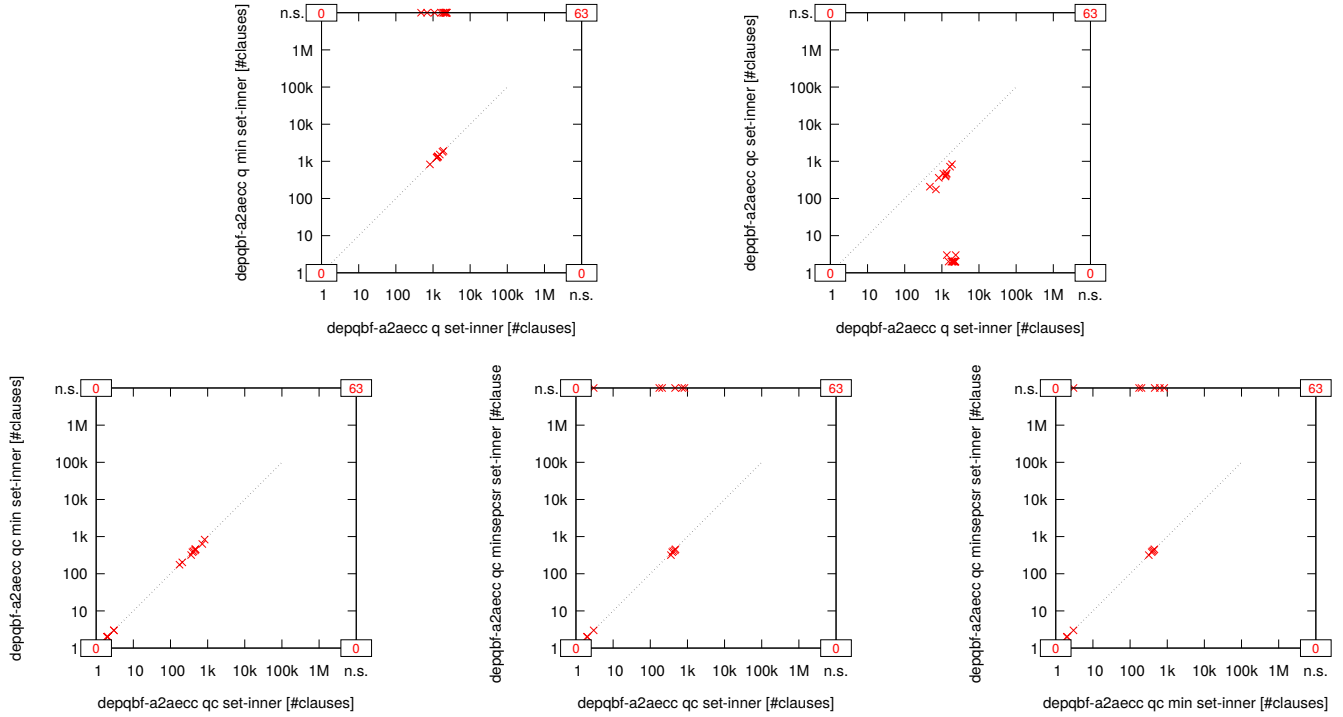


Fig. 177: Suite Letombe ($n = 85$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

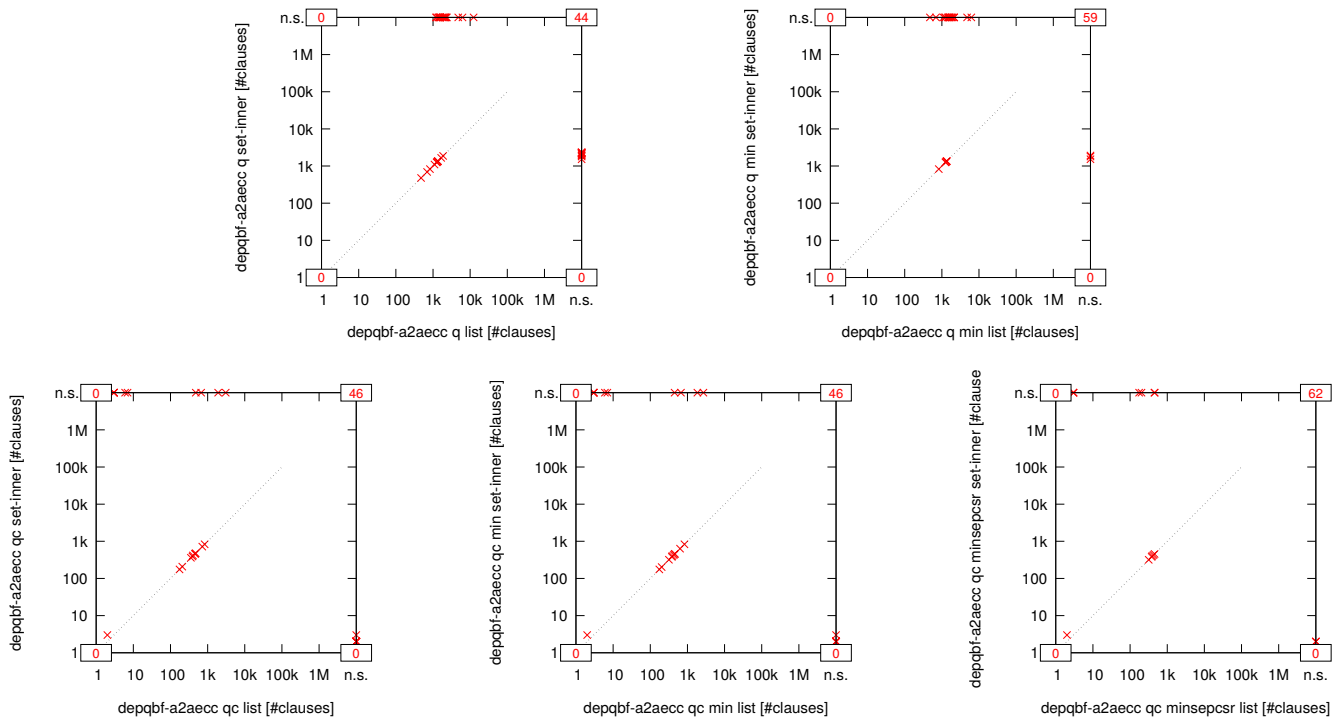


Fig. 178: Suite Letombe ($n = 85$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

25) Letz ($n = 9$):

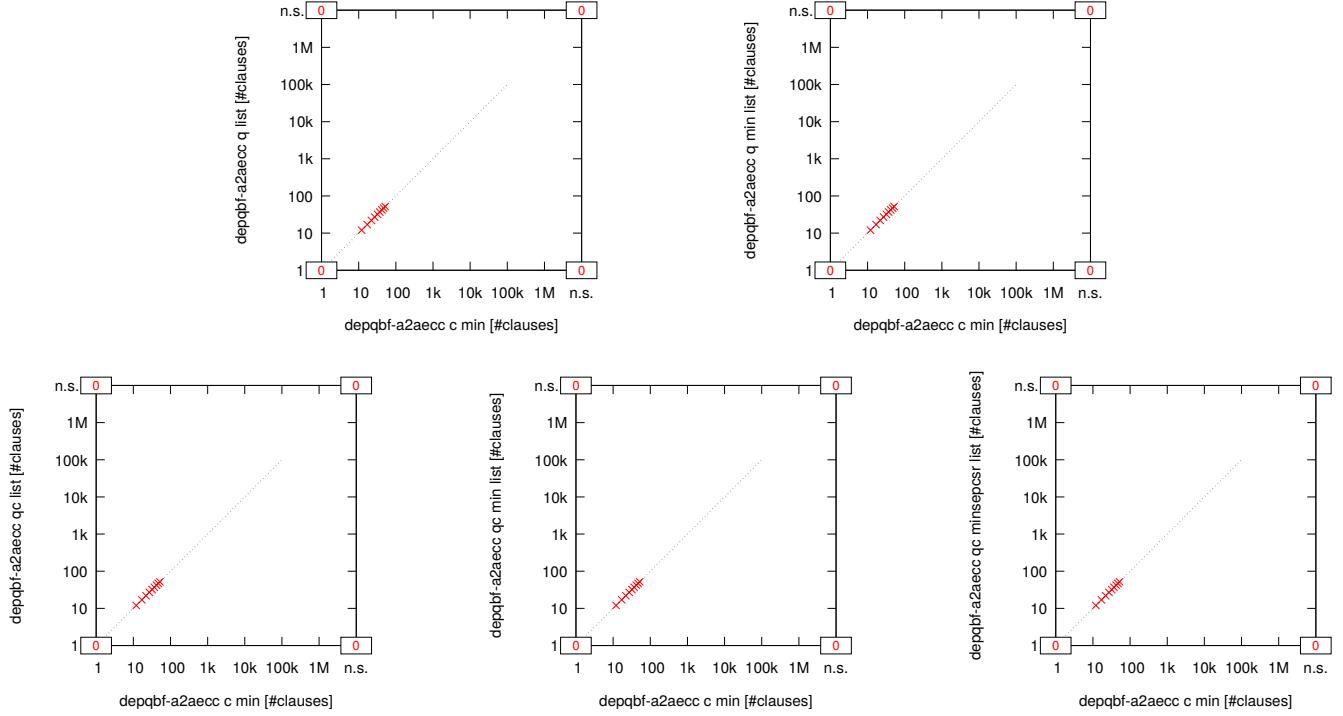


Fig. 179: Suite Letz ($n = 9$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

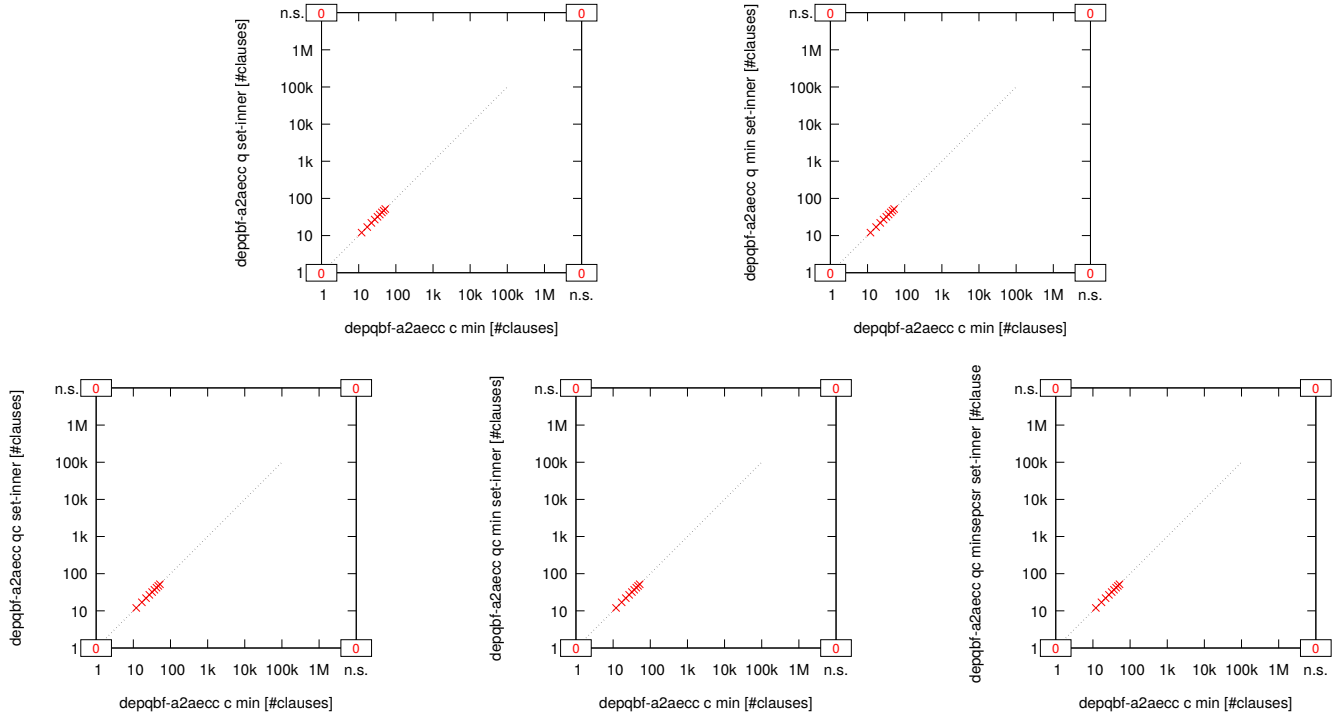


Fig. 180: Suite Letz ($n = 9$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

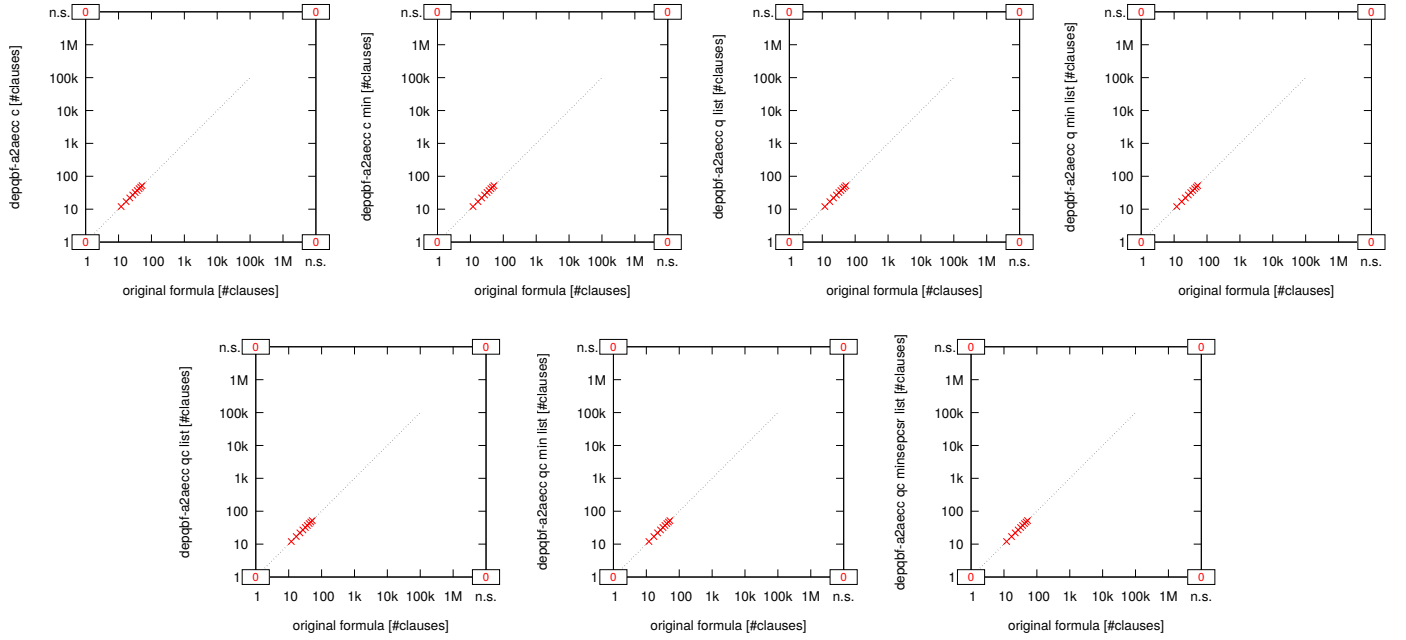


Fig. 181: Suite Letz ($n = 9$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

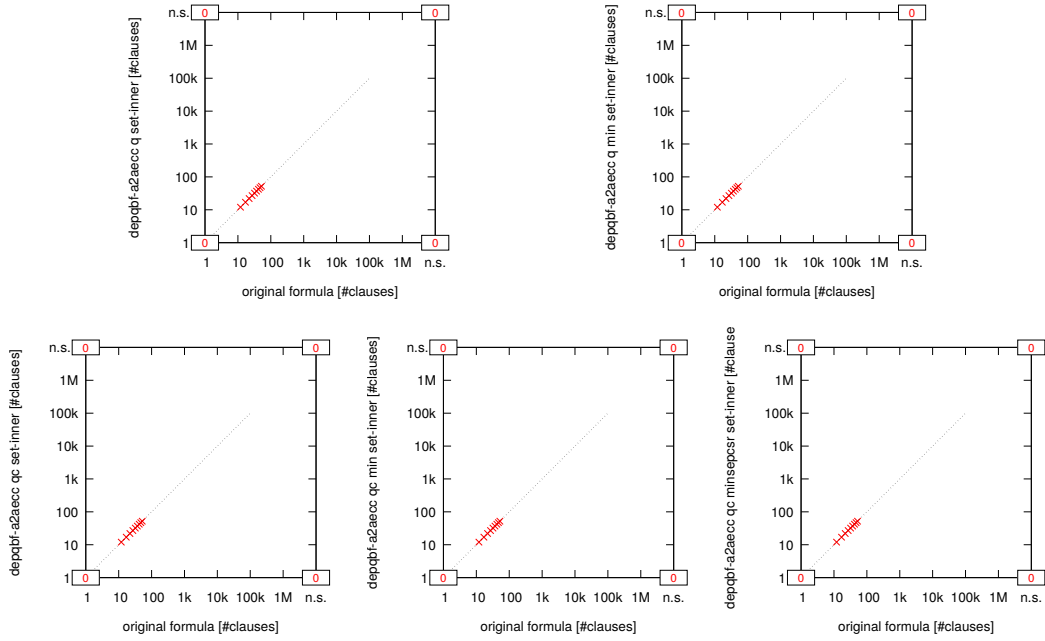


Fig. 182: Suite Letz ($n = 9$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

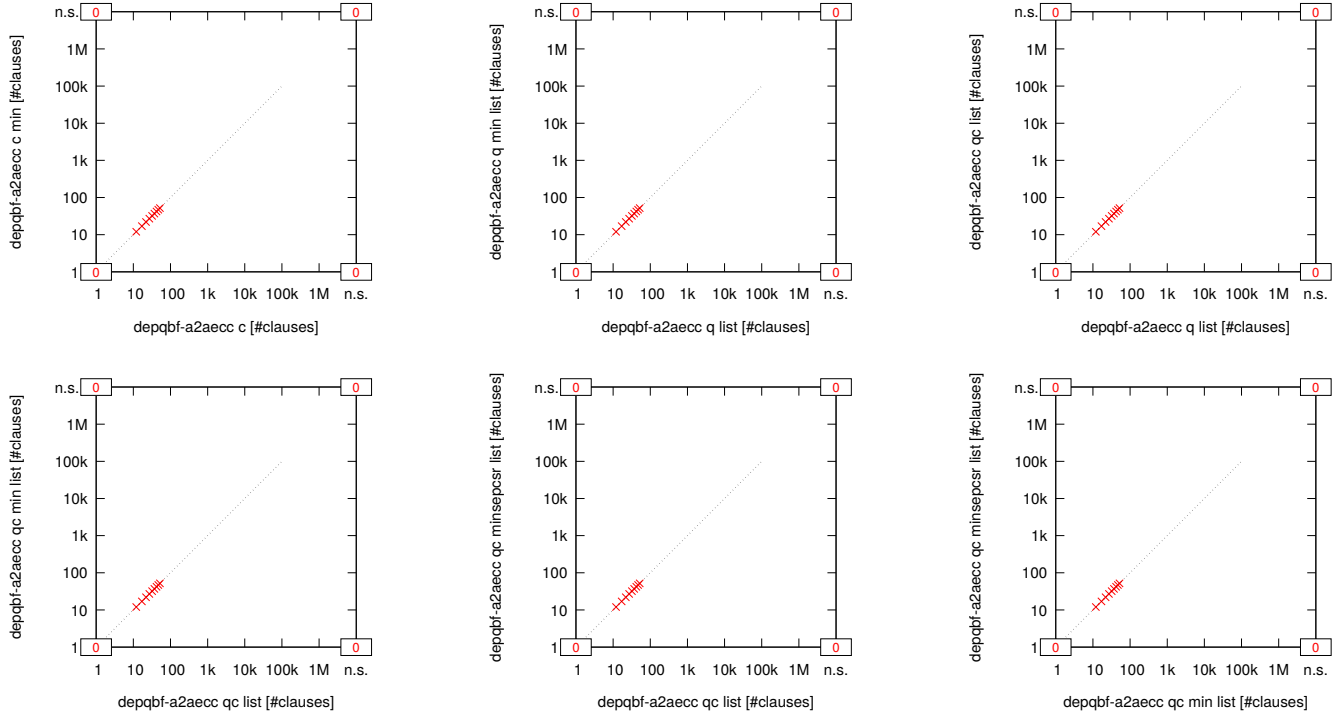


Fig. 183: Suite Letz ($n = 9$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

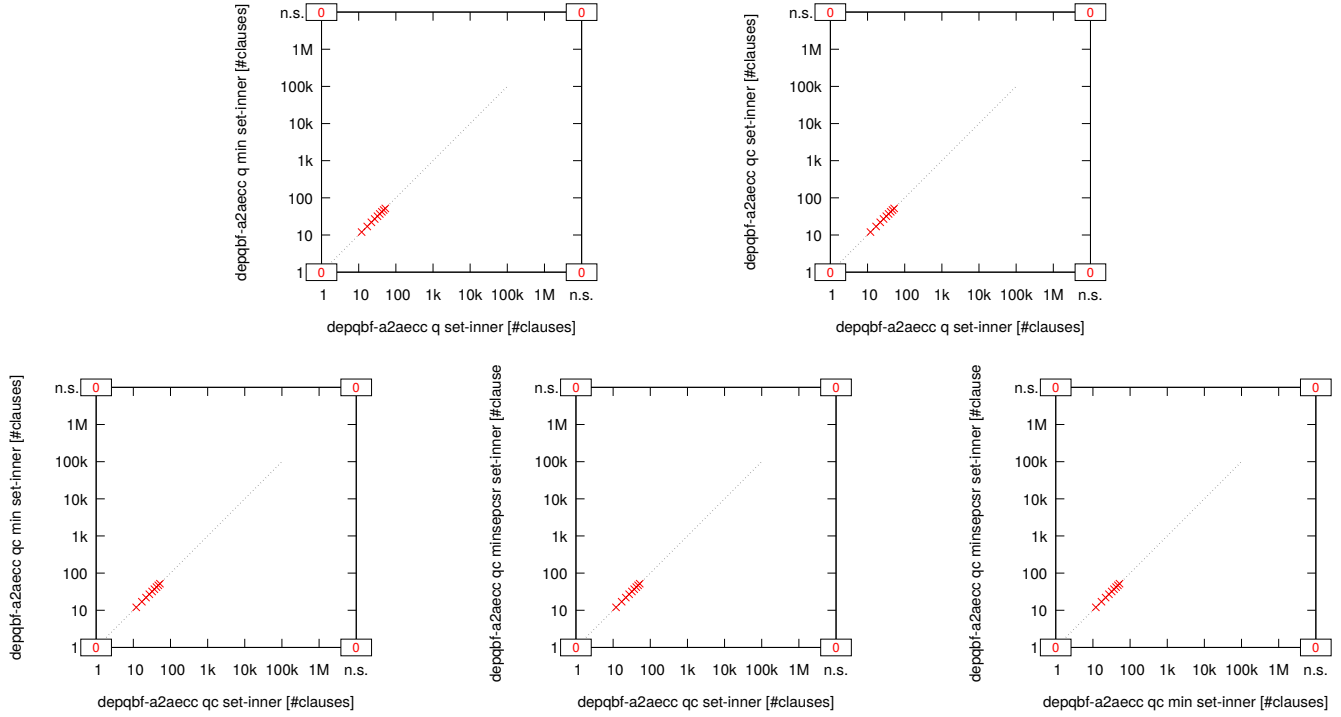


Fig. 184: Suite Letz ($n = 9$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

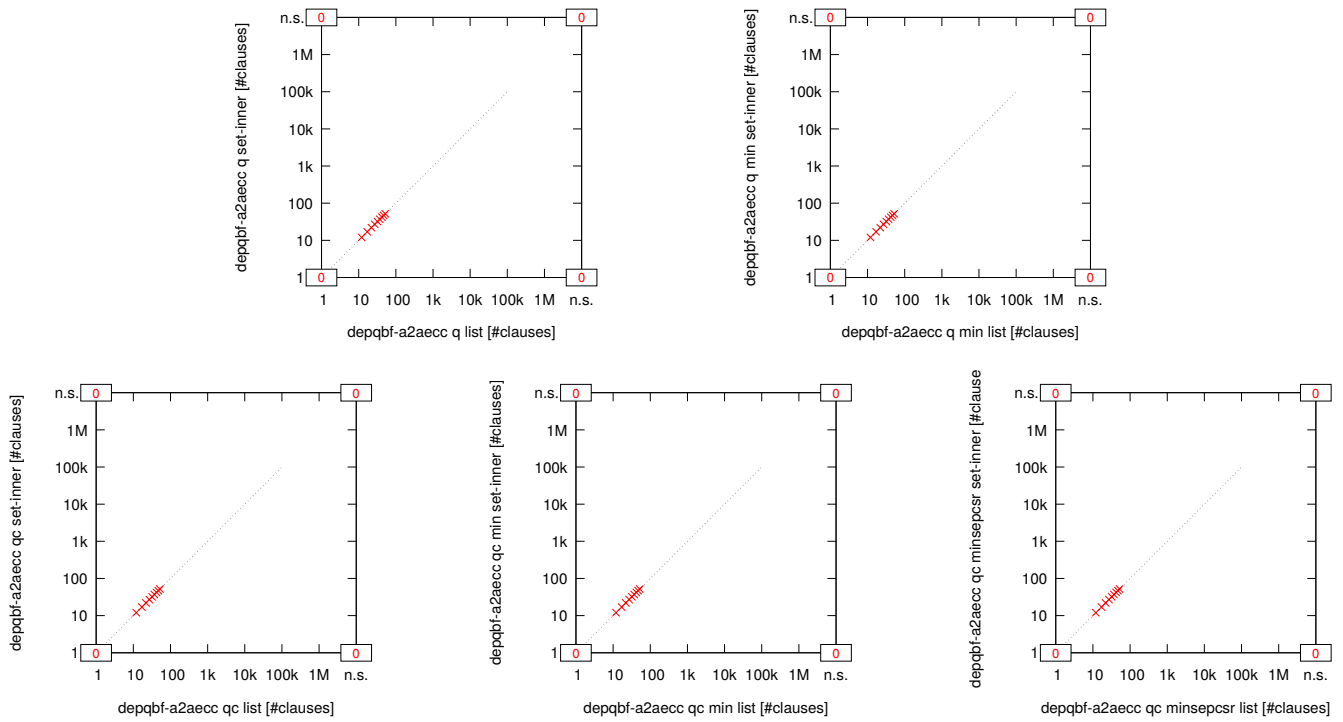


Fig. 185: Suite Letz ($n = 9$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

26) Ling ($n = 3$):

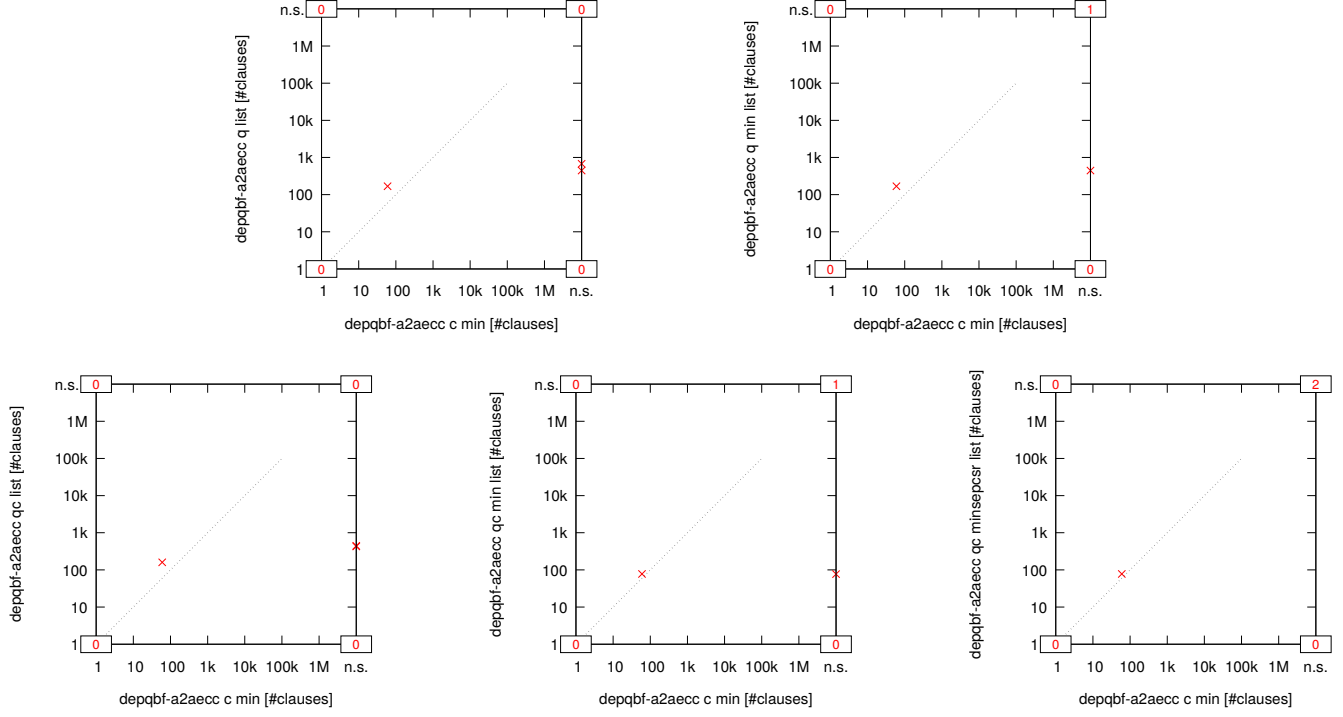


Fig. 186: Suite Ling ($n = 3$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

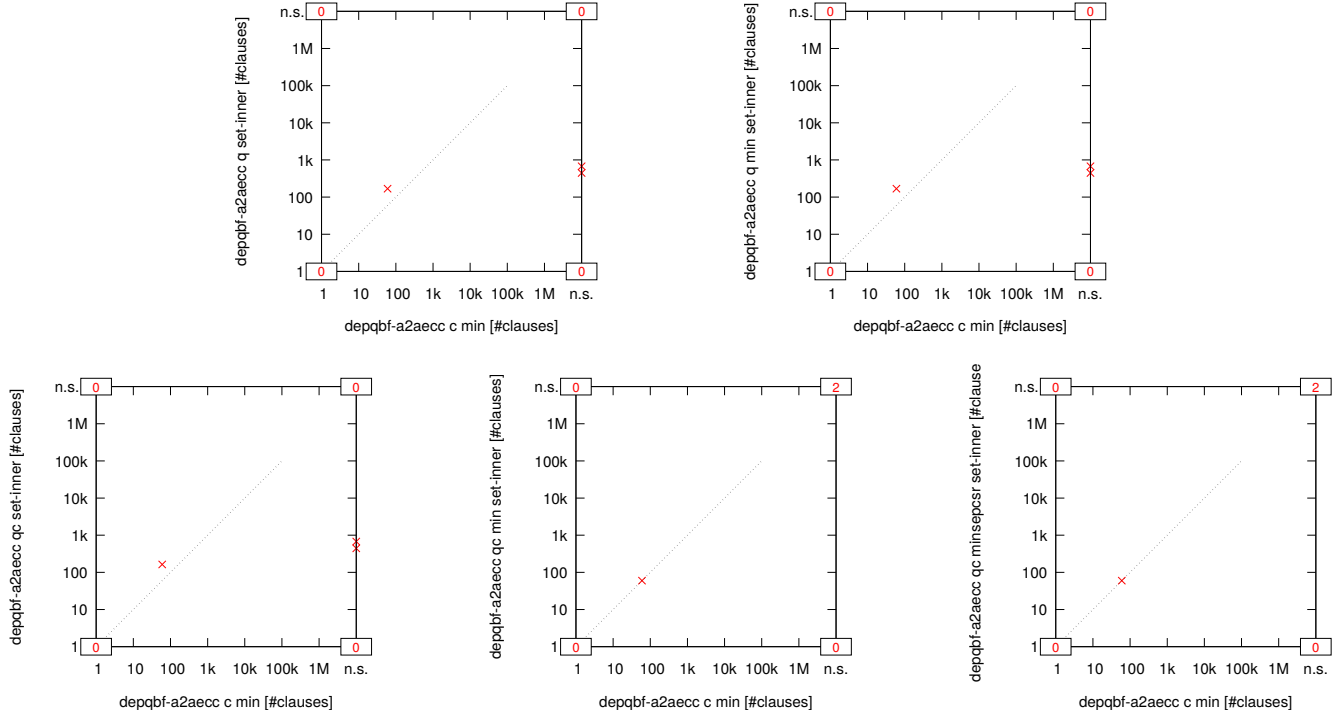


Fig. 187: Suite Ling ($n = 3$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

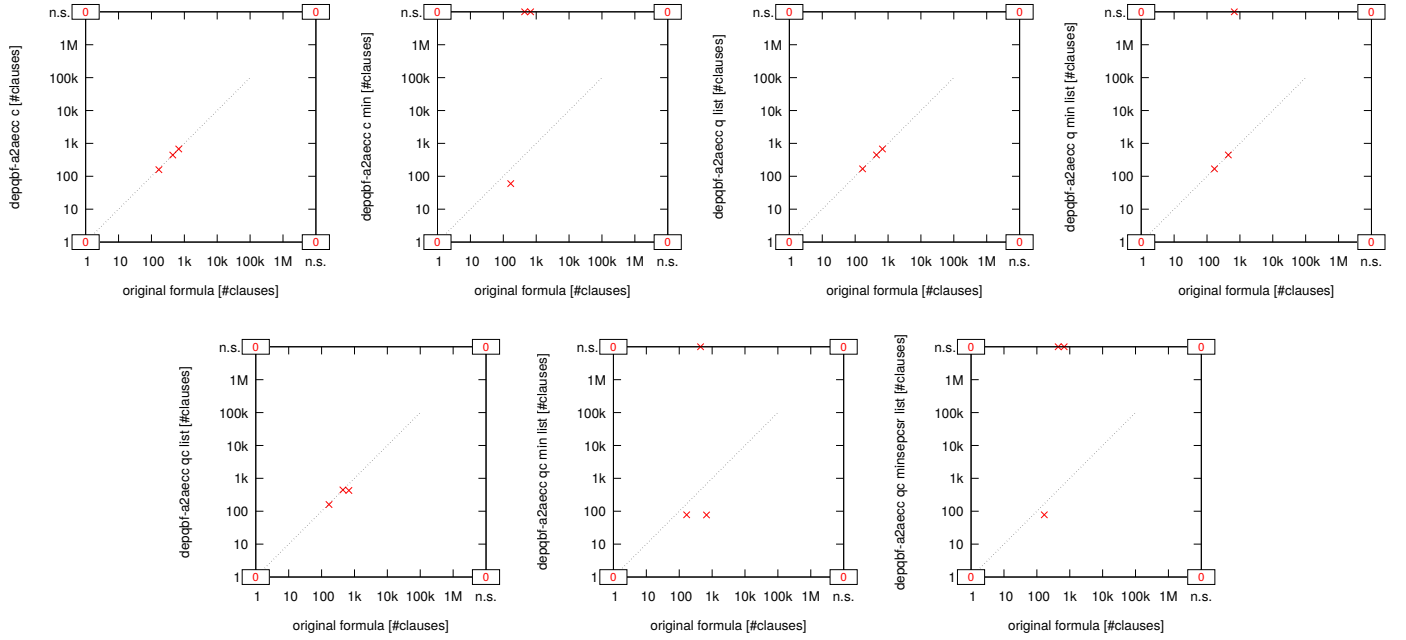


Fig. 188: Suite Ling ($n = 3$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

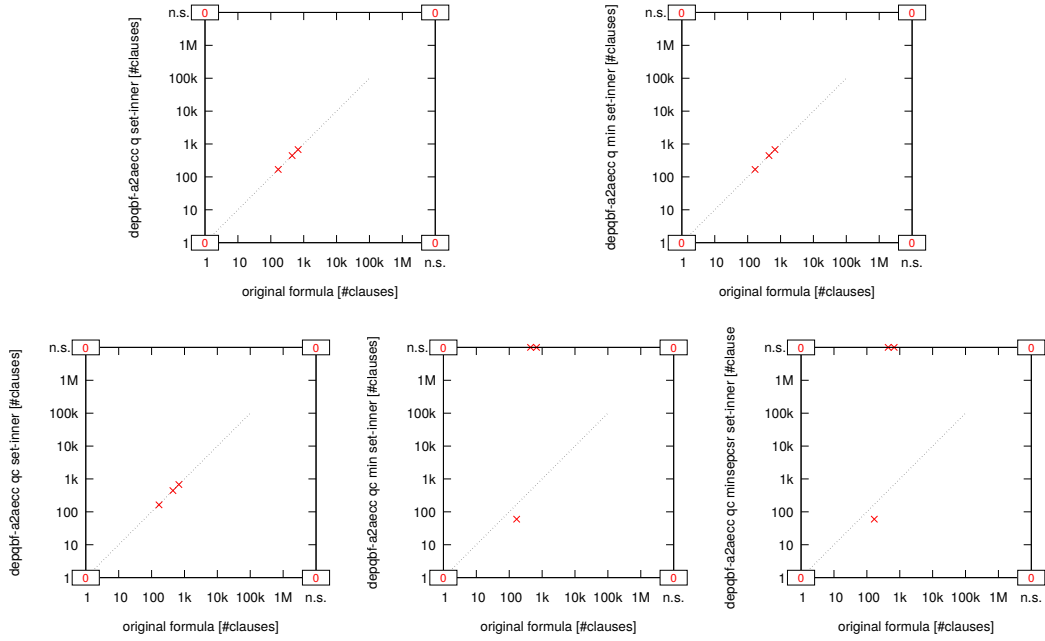


Fig. 189: Suite Ling ($n = 3$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

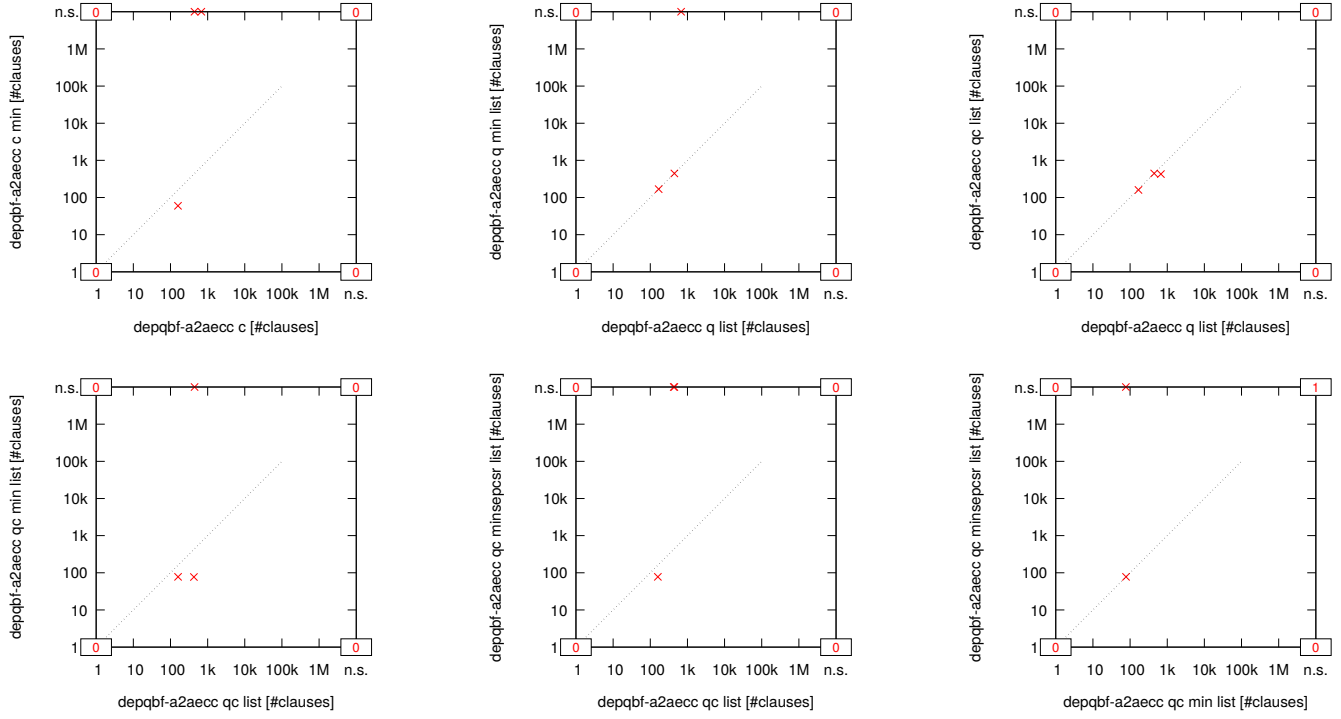


Fig. 190: Suite Ling ($n = 3$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

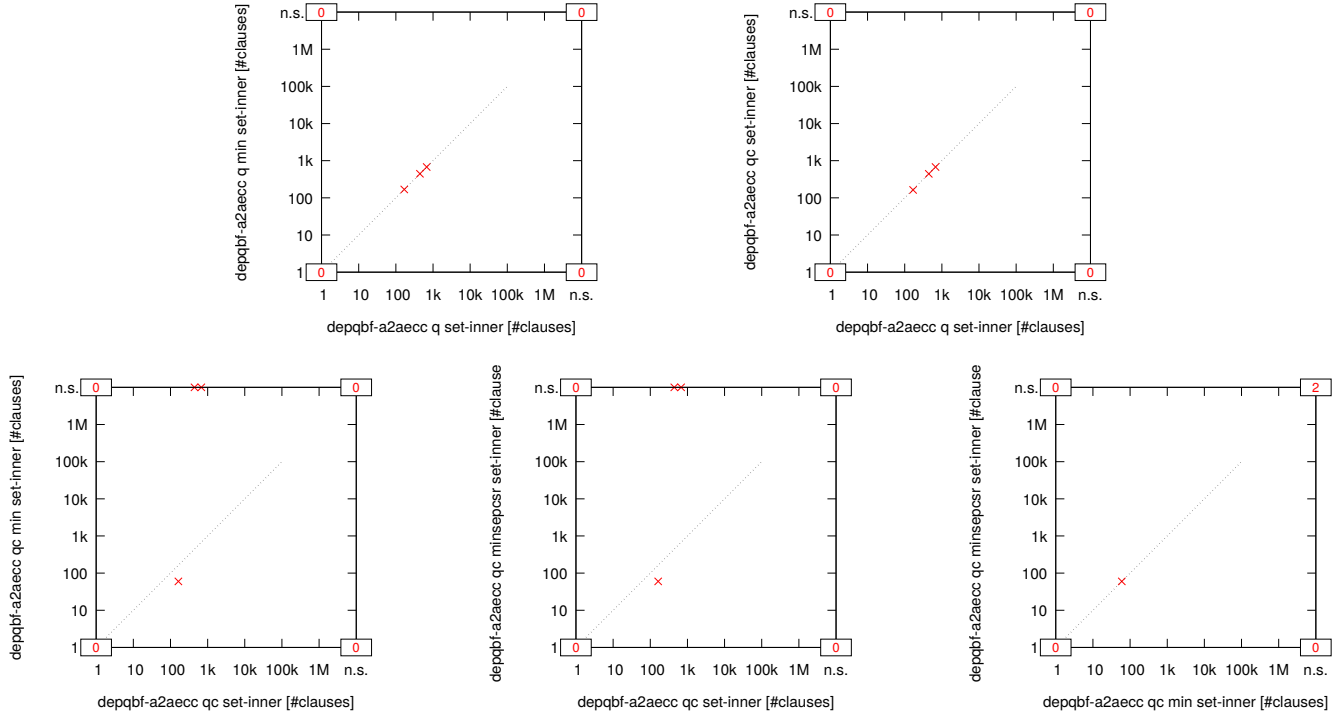


Fig. 191: Suite Ling ($n = 3$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

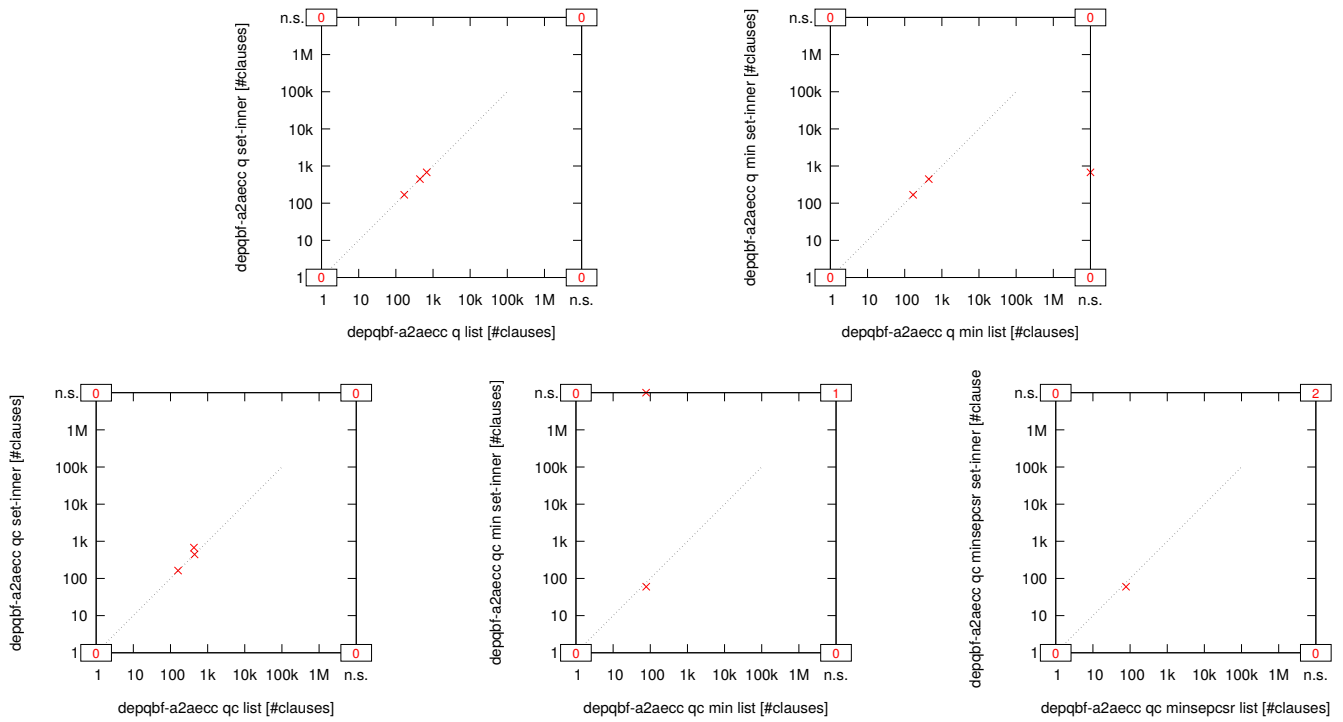


Fig. 192: Suite Ling ($n = 3$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

27) Mangassarian-Veneris ($n = 60$):

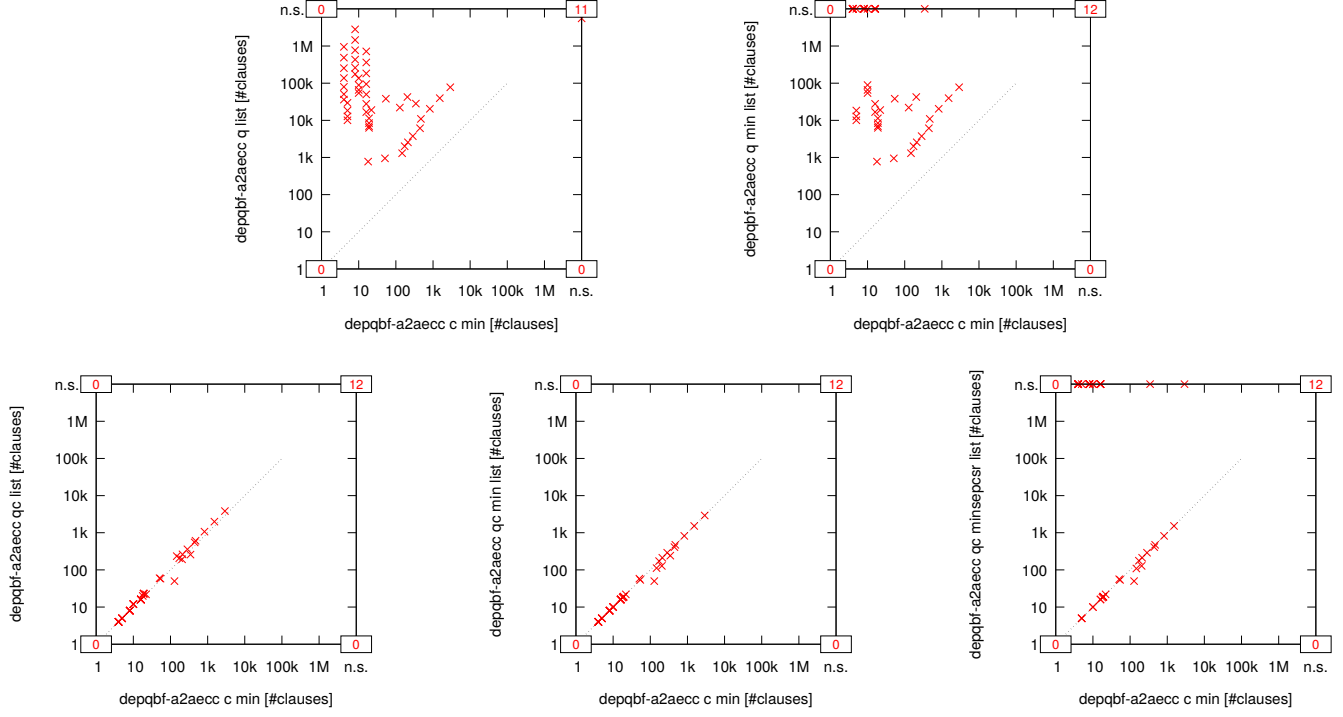


Fig. 193: Suite Mangassarian-Veneris ($n = 60$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

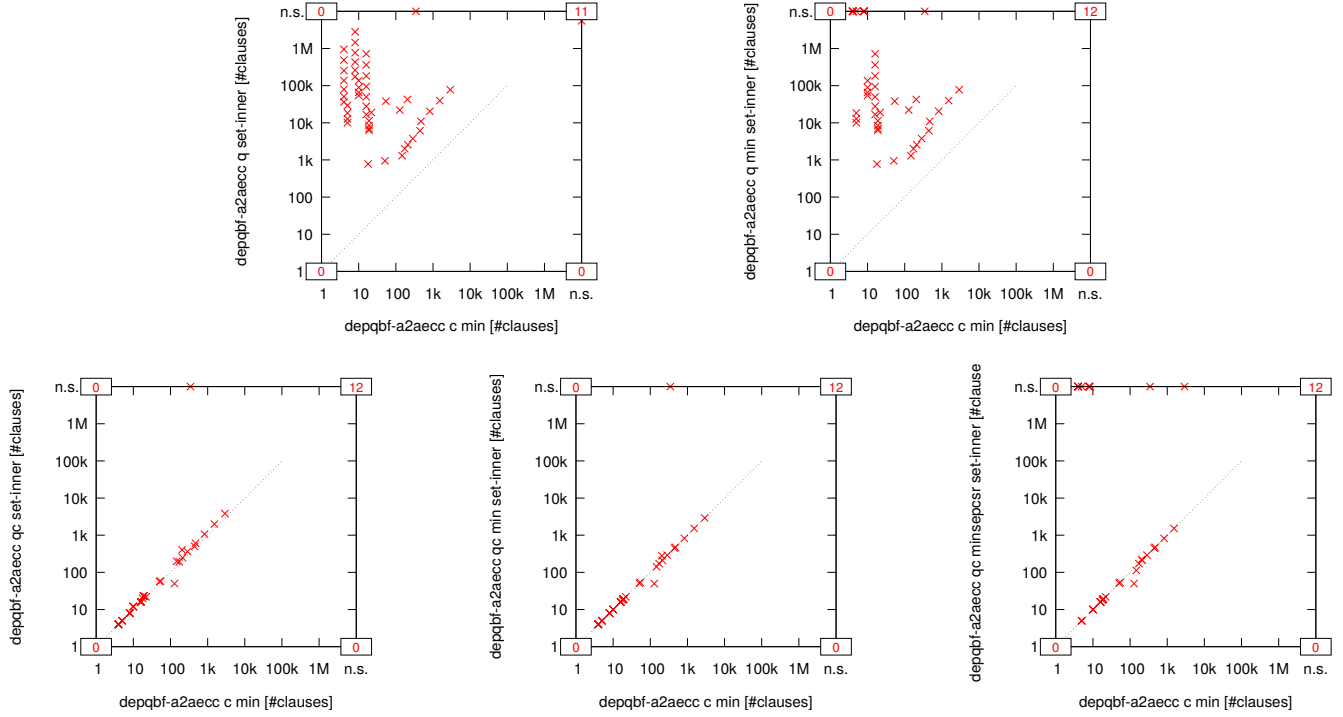


Fig. 194: Suite Mangassarian-Veneris ($n = 60$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

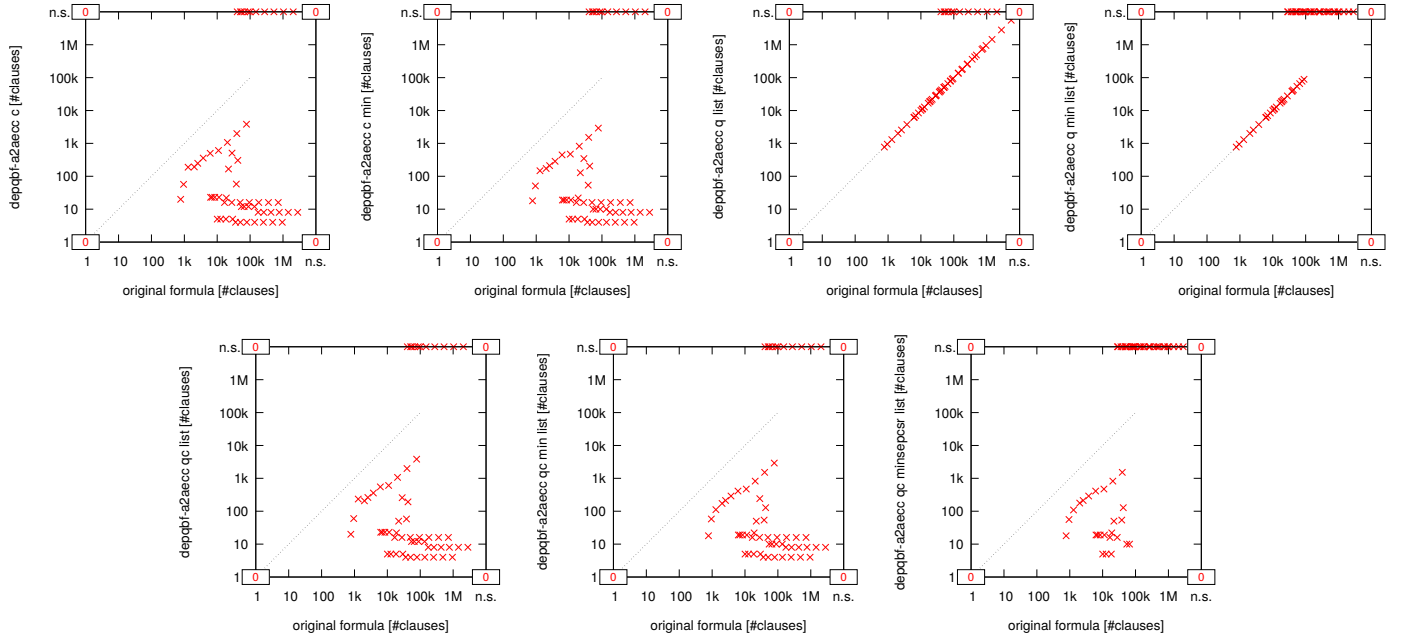


Fig. 195: Suite Mangassarian-Veneris ($n = 60$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

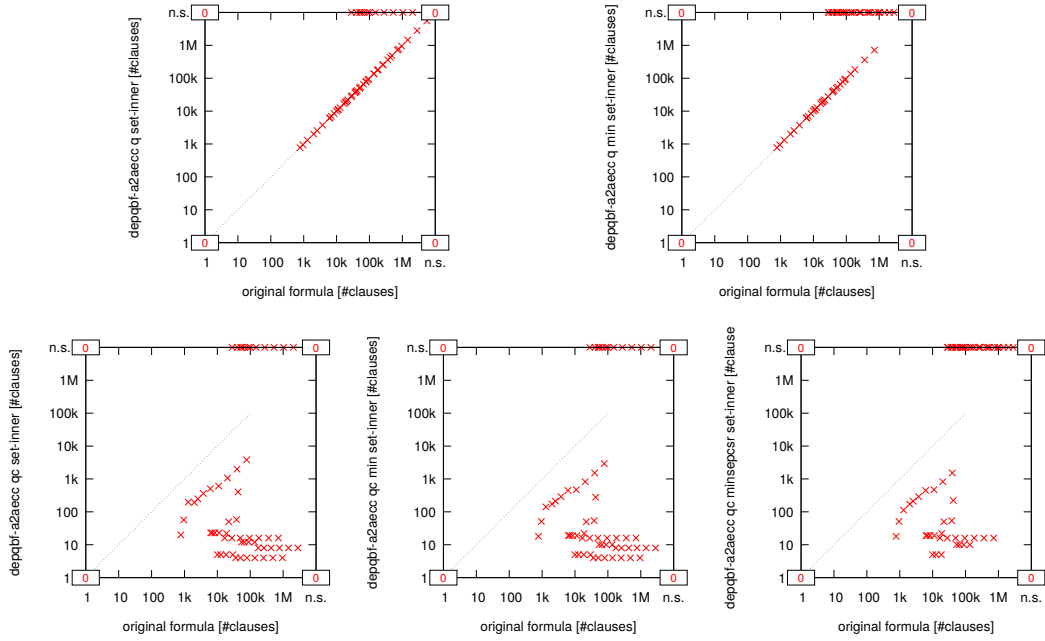


Fig. 196: Suite Mangassarian-Veneris ($n = 60$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

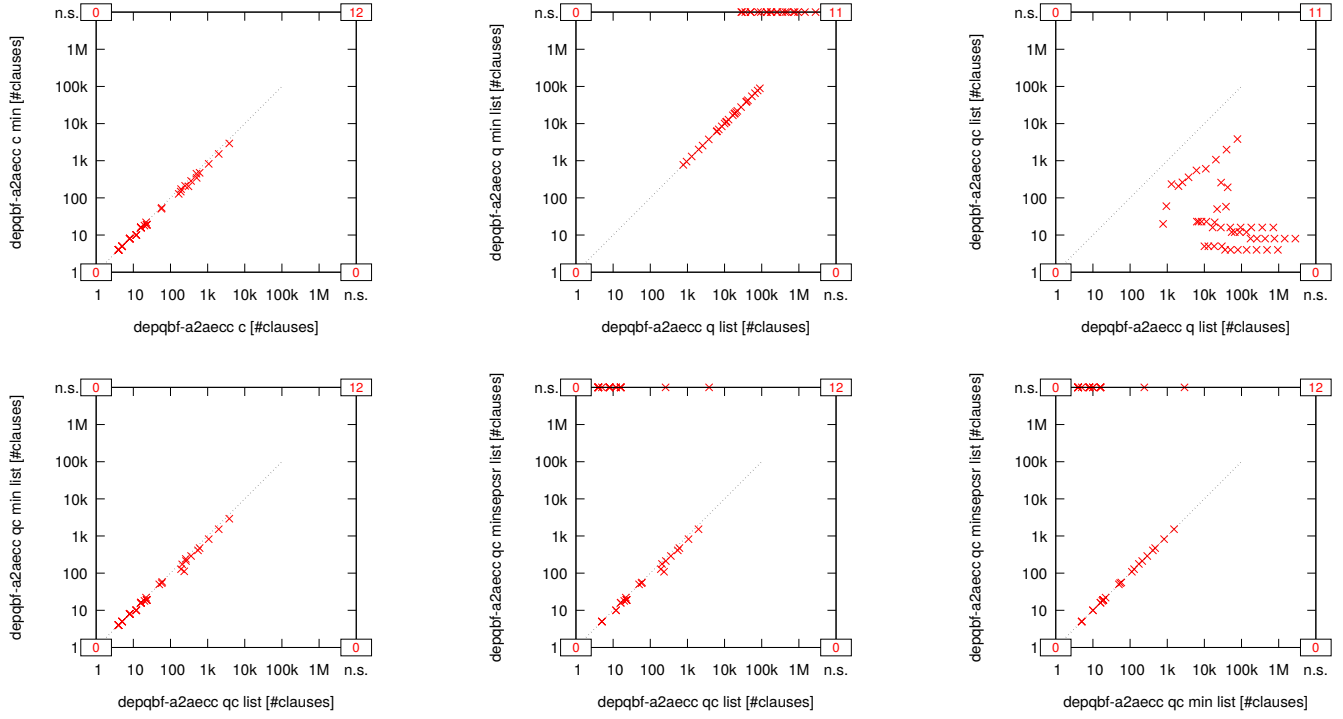


Fig. 197: Suite Mangassarian-Veneris ($n = 60$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

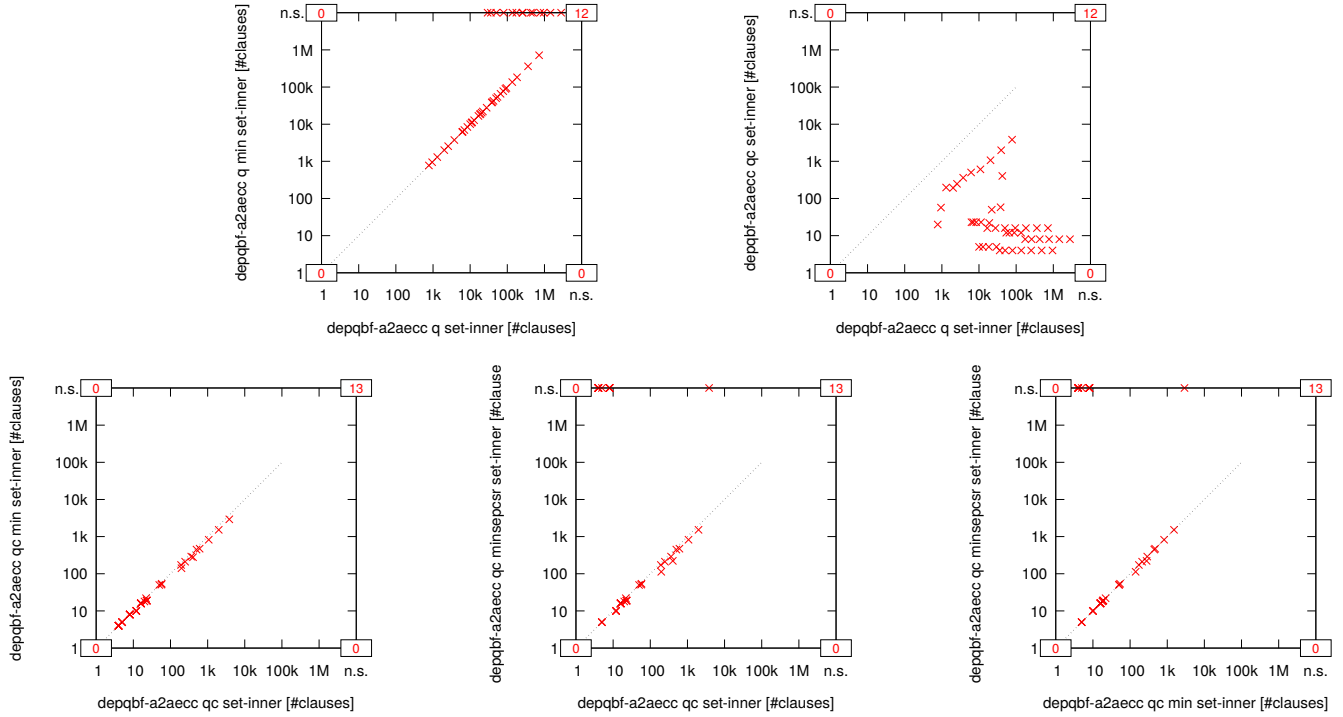


Fig. 198: Suite Mangassarian-Veneris ($n = 60$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

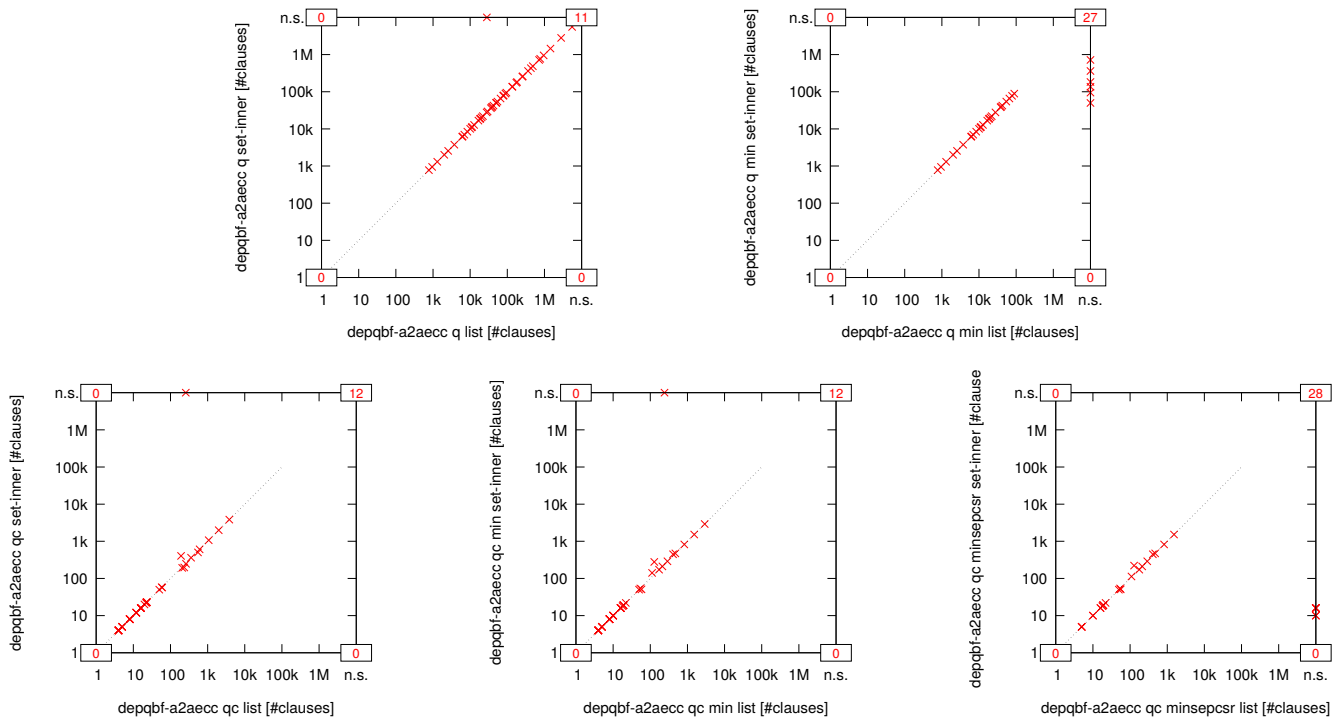


Fig. 199: Suite Mangassarian-Veneris ($n = 60$): Comparing sizes of different unsatisfiable cores in `DepQBF-a2aecc` in list versus set-inner semantics (number of clauses).

28) MayerEichberger-Saffidine ($n = 3$):

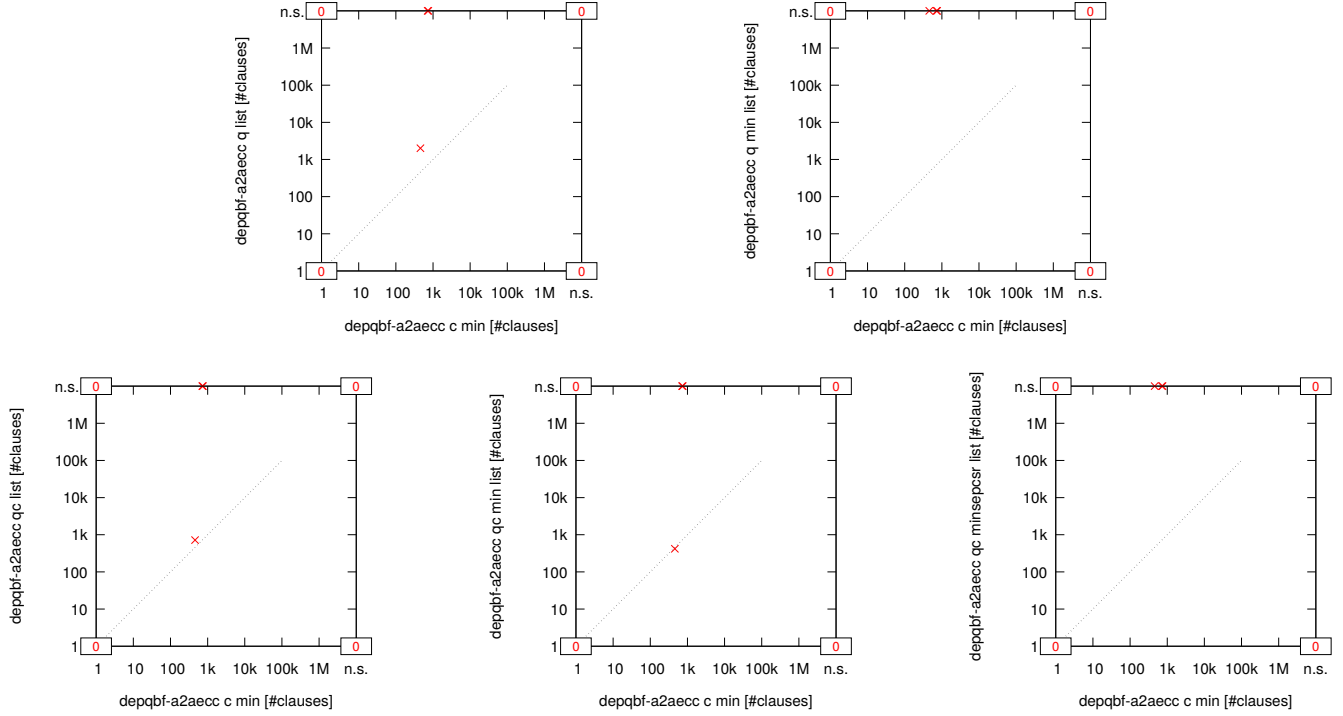


Fig. 200: Suite MayerEichberger-Saffidine ($n = 3$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

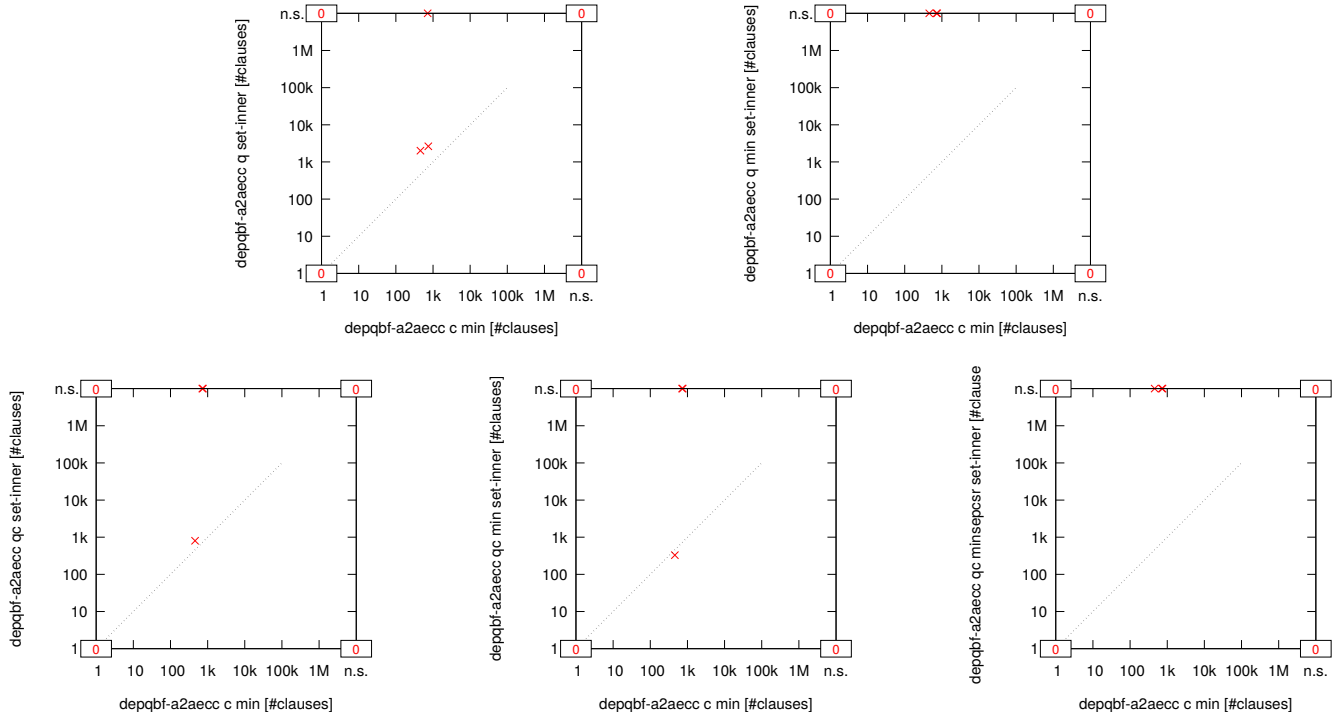


Fig. 201: Suite MayerEichberger-Saffidine ($n = 3$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

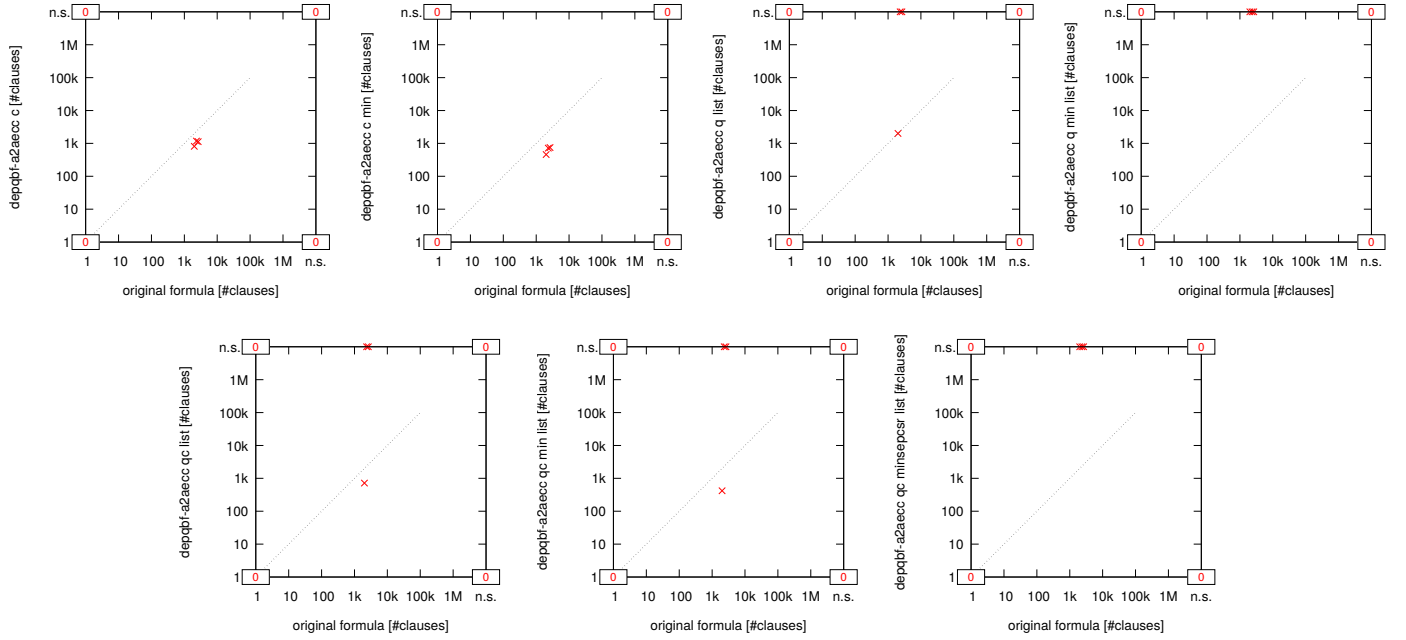


Fig. 202: Suite MayerEichberger-Saffidine ($n = 3$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

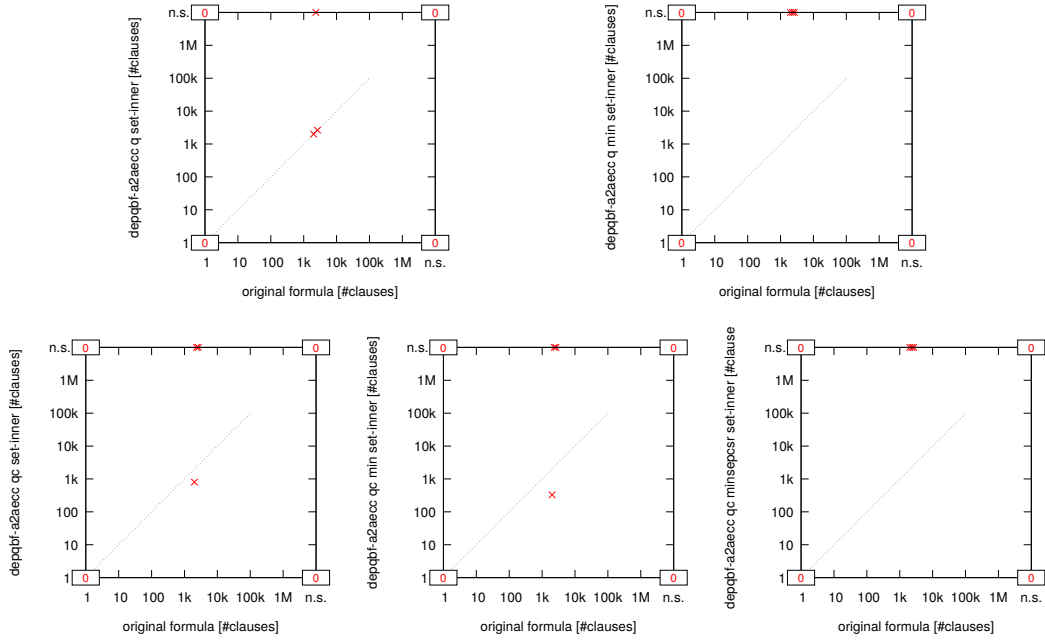


Fig. 203: Suite MayerEichberger-Saffidine ($n = 3$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

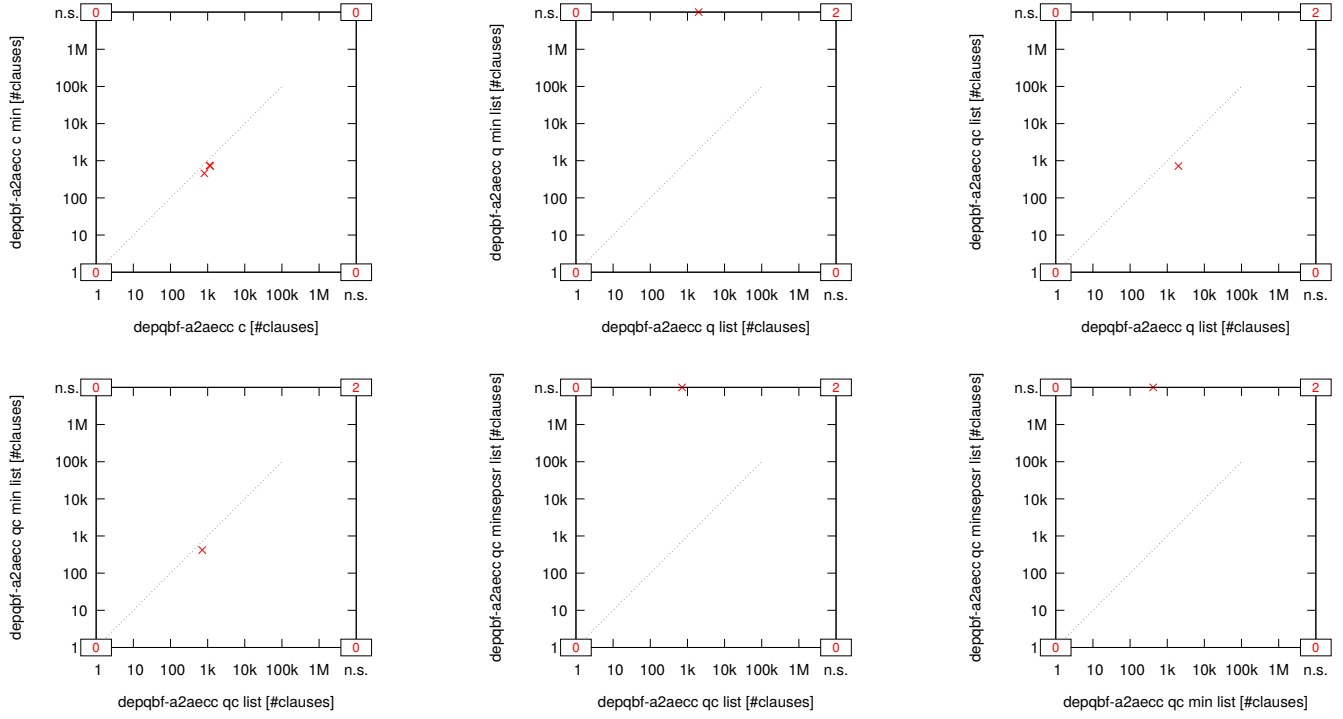


Fig. 204: Suite MayerEichberger-Saffidine ($n = 3$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

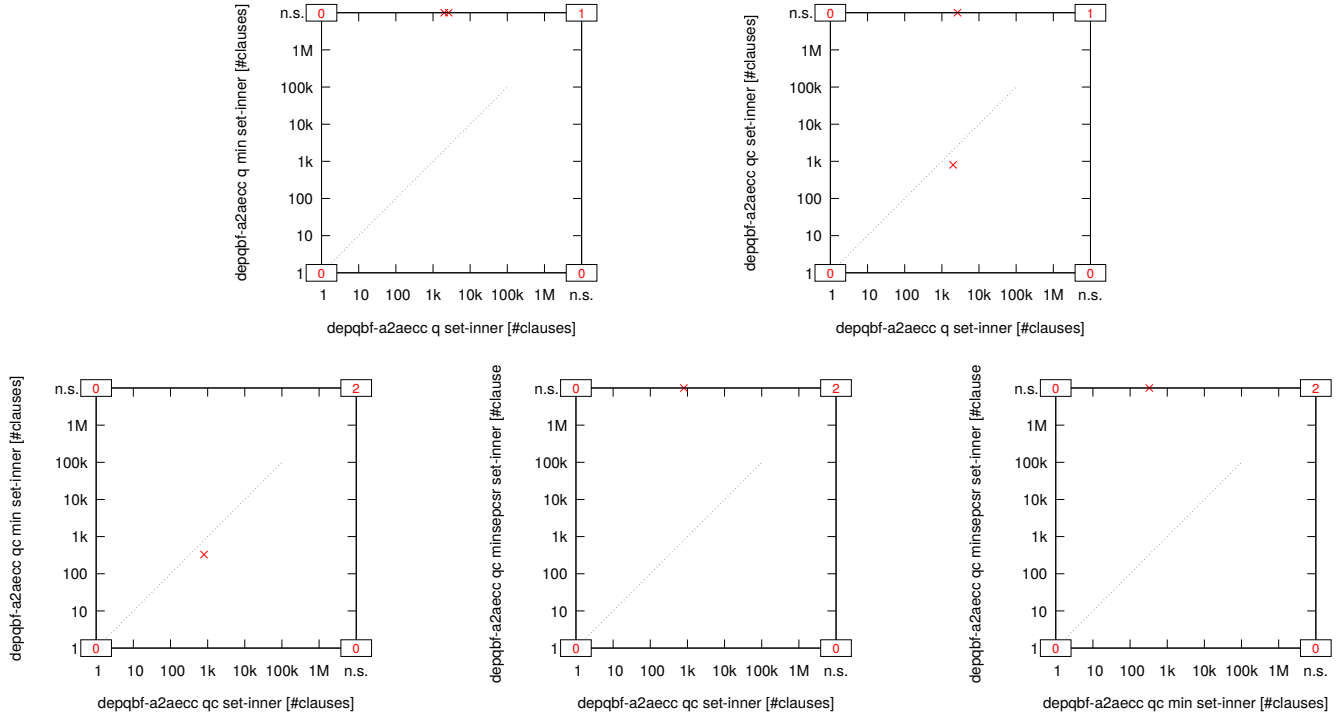


Fig. 205: Suite MayerEichberger-Saffidine ($n = 3$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

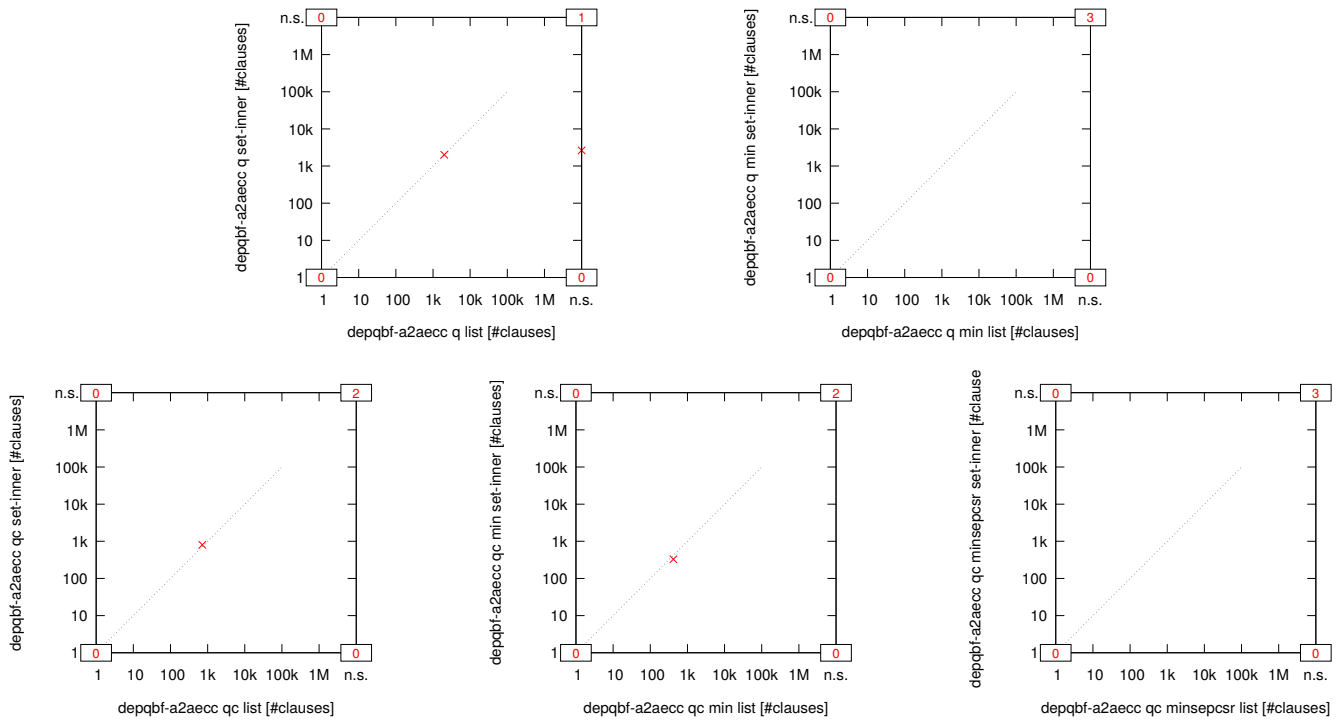


Fig. 206: Suite MayerEichberger-Saffidine ($n = 3$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

29) *Messinger* ($n = 0$):

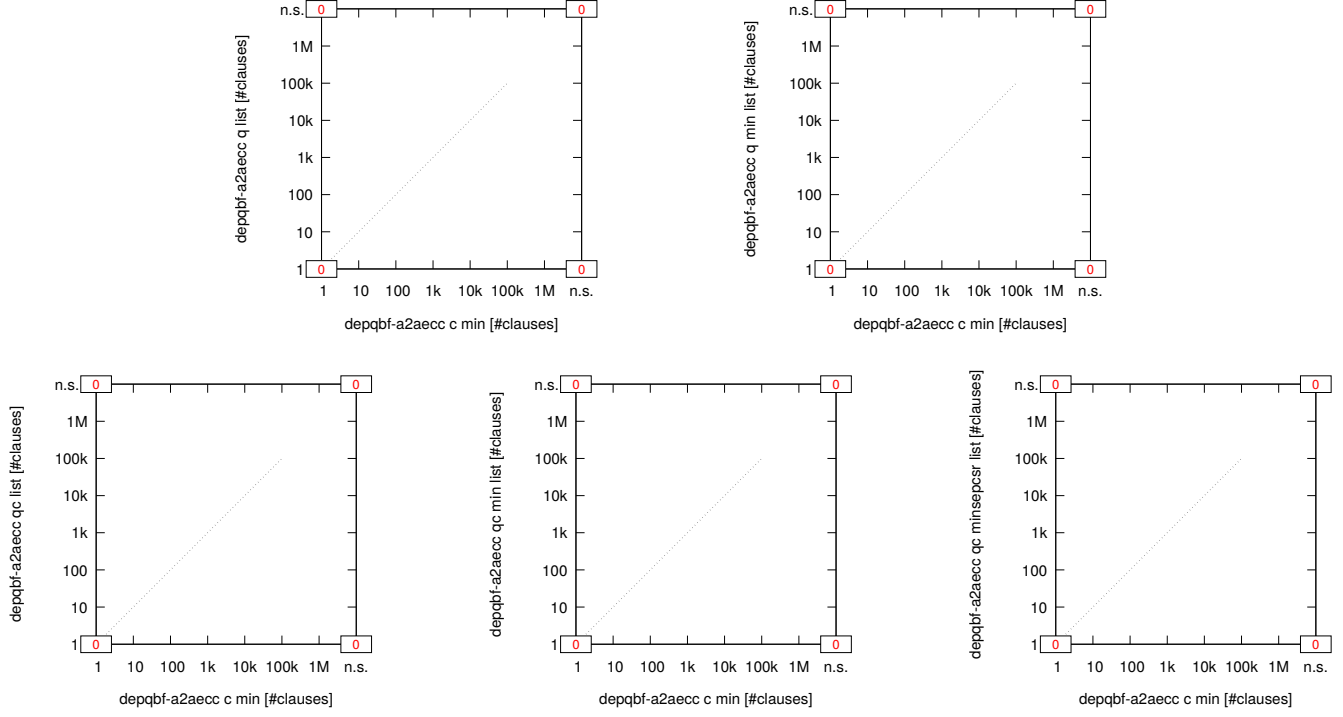


Fig. 207: Suite Messinger ($n = 0$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

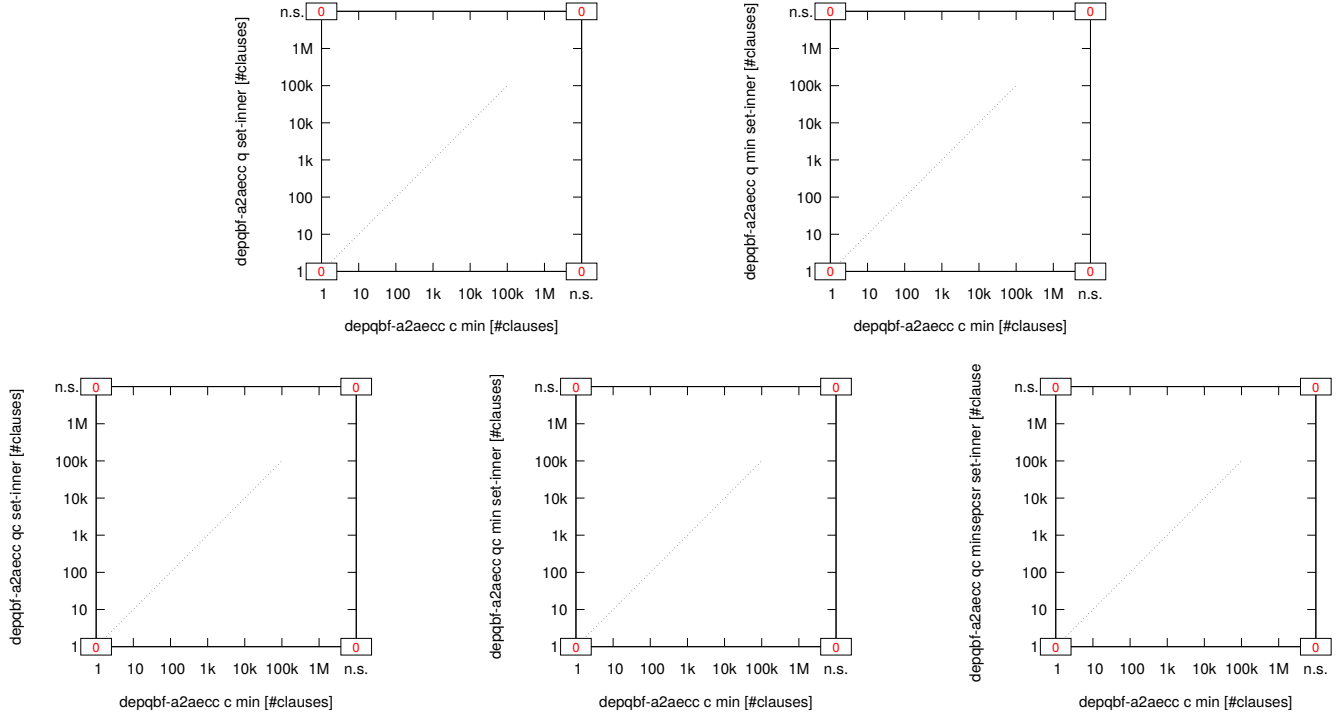


Fig. 208: Suite Messinger ($n = 0$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

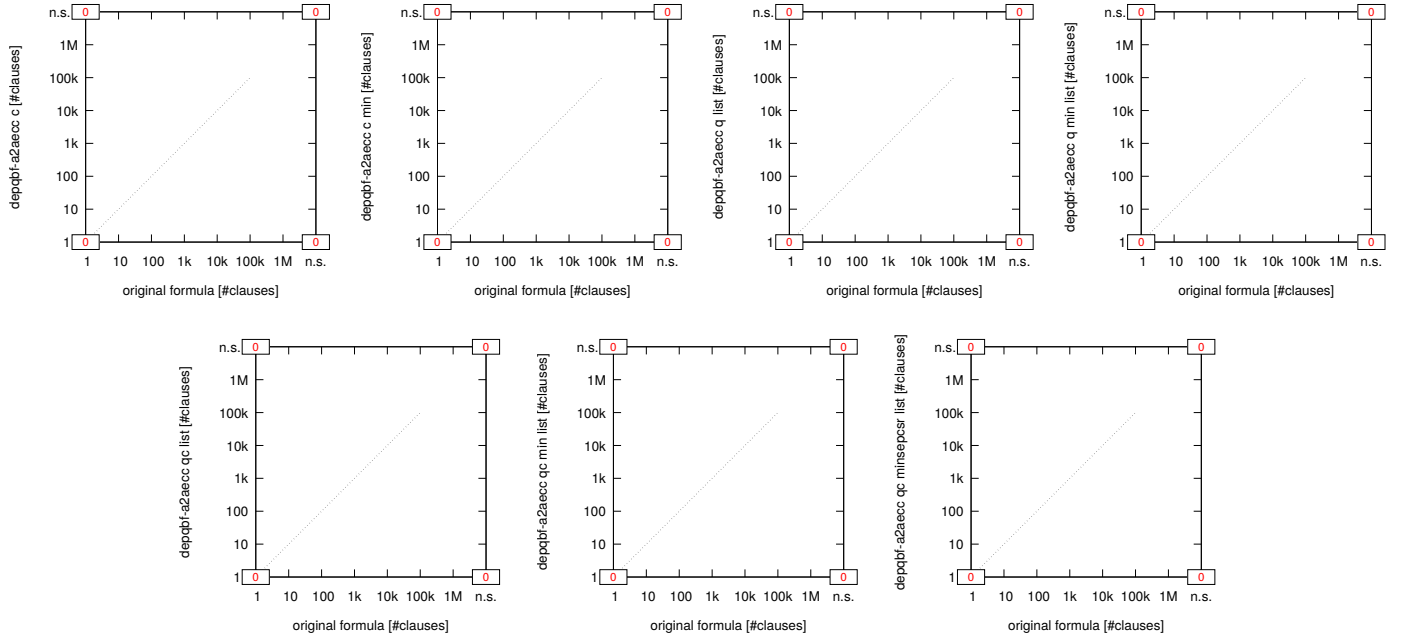


Fig. 209: Suite Messenger ($n = 0$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

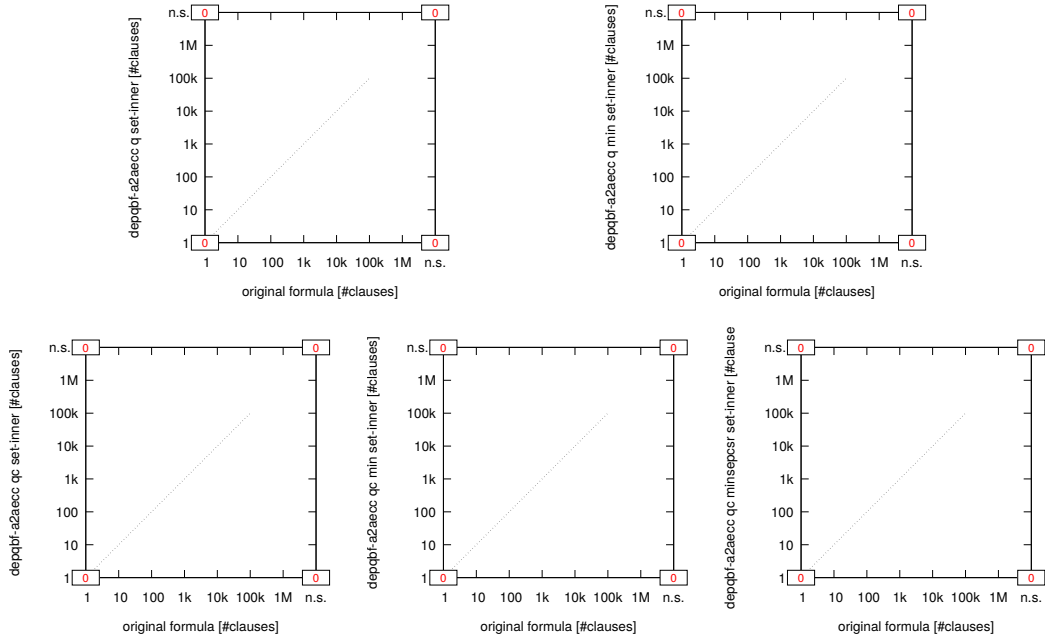


Fig. 210: Suite Messenger ($n = 0$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

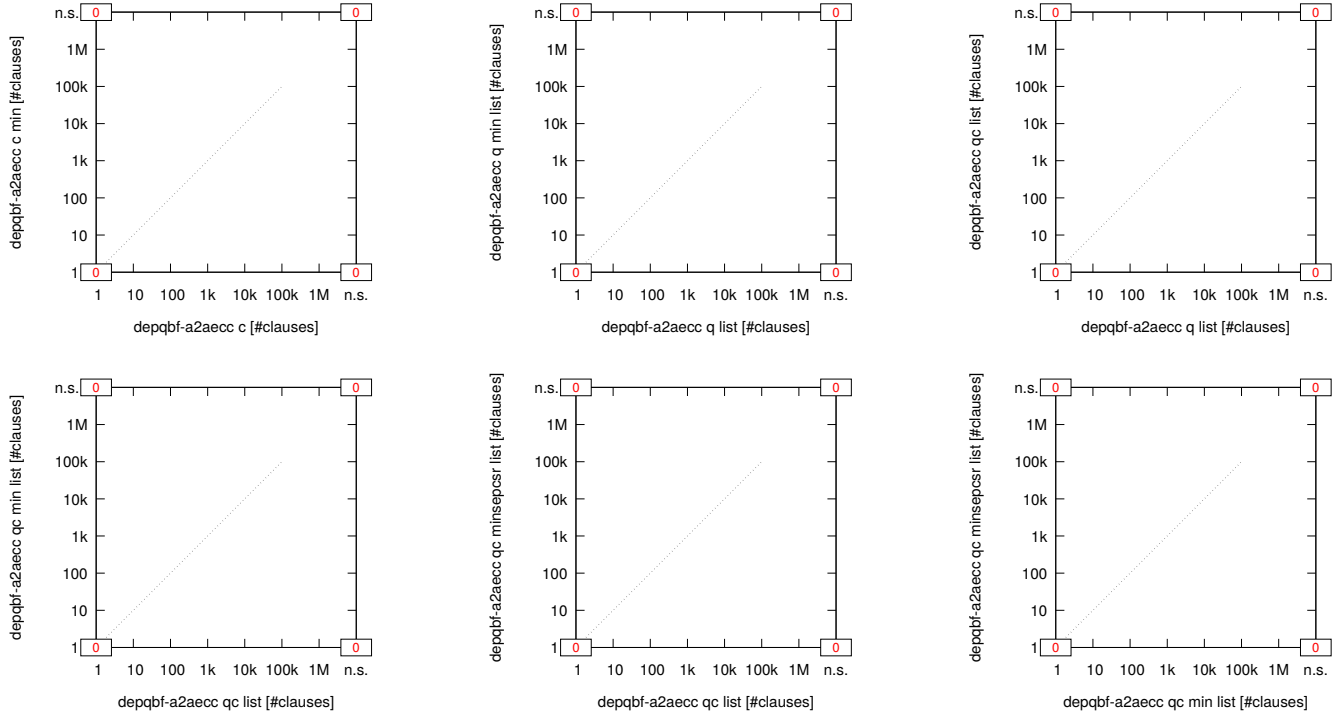


Fig. 211: Suite Messenger ($n = 0$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

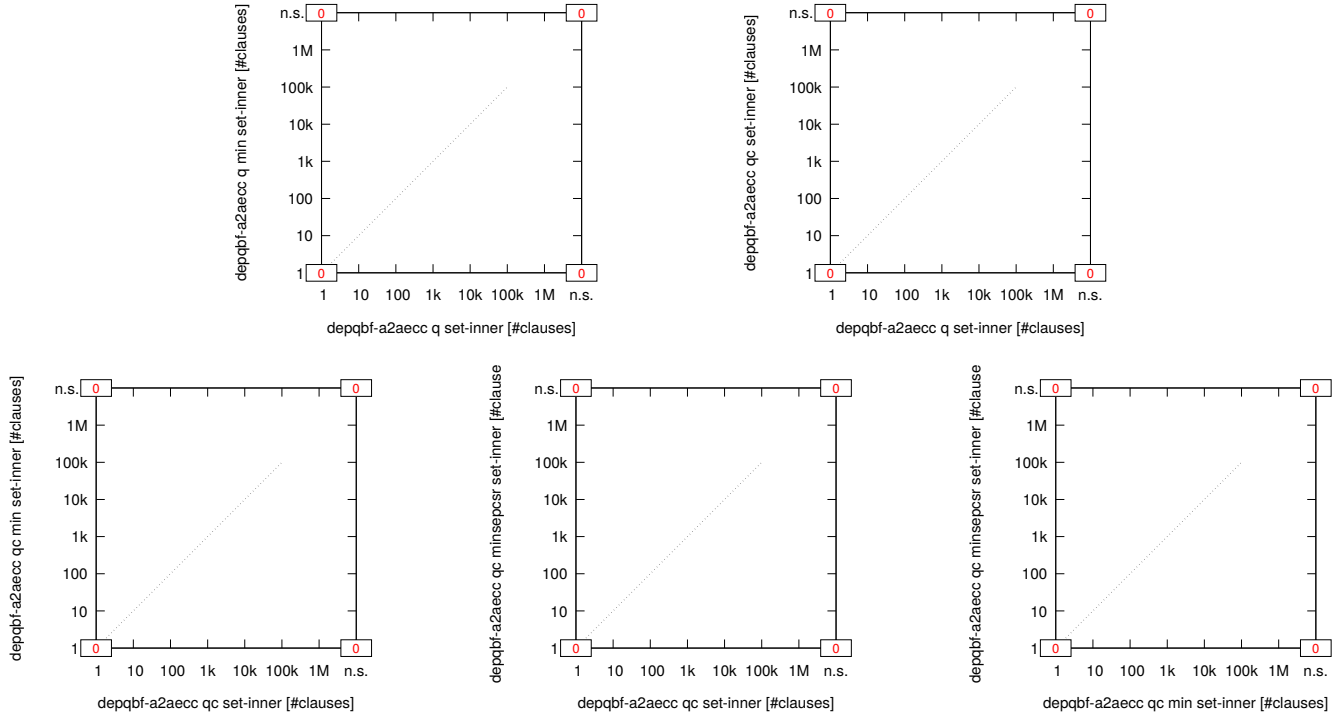


Fig. 212: Suite Messenger ($n = 0$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

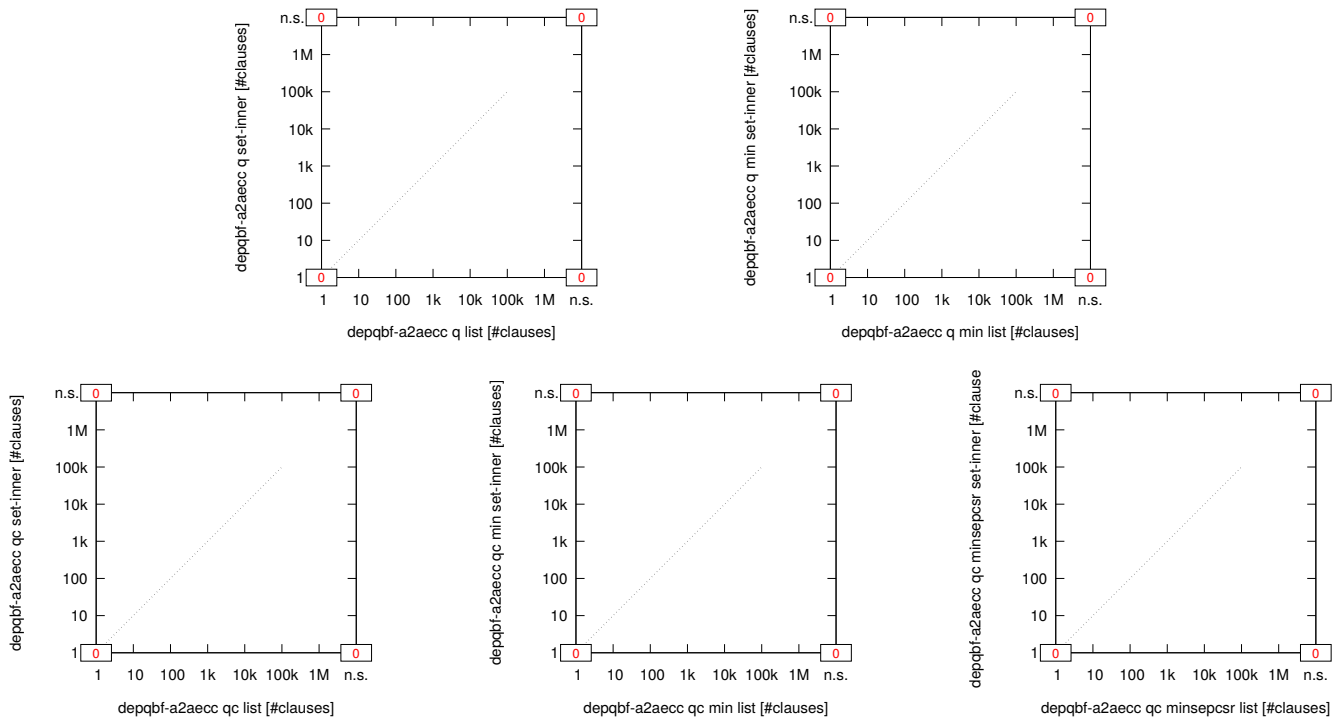


Fig. 213: Suite Messenger ($n = 0$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

30) Miller-Marin ($n = 189$):

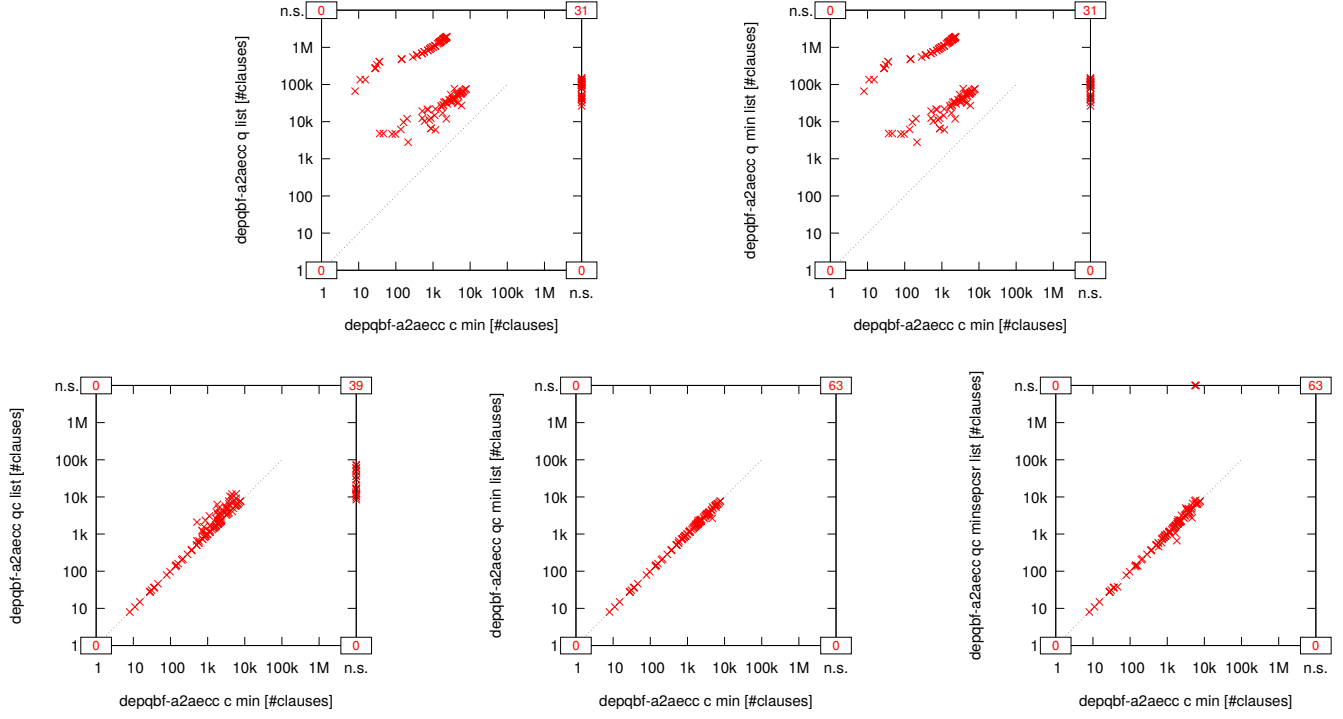


Fig. 214: Suite Miller-Marin ($n = 189$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

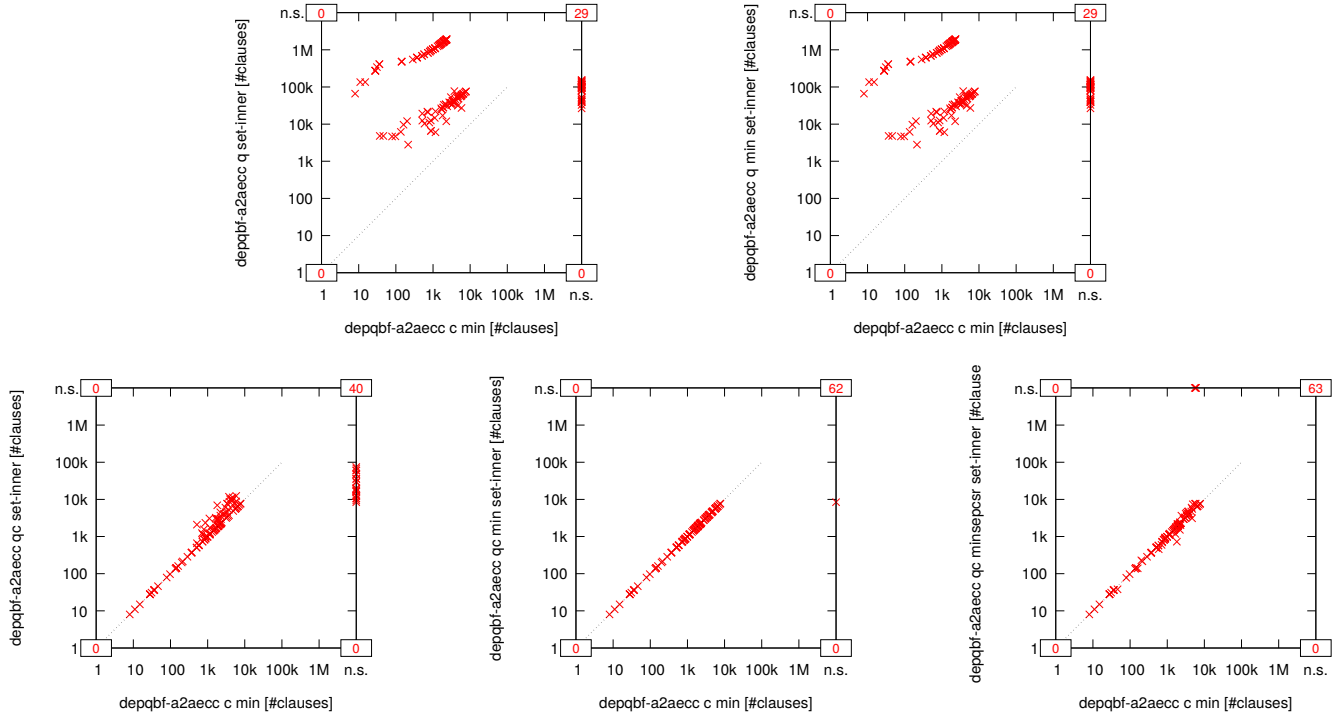


Fig. 215: Suite Miller-Marin ($n = 189$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

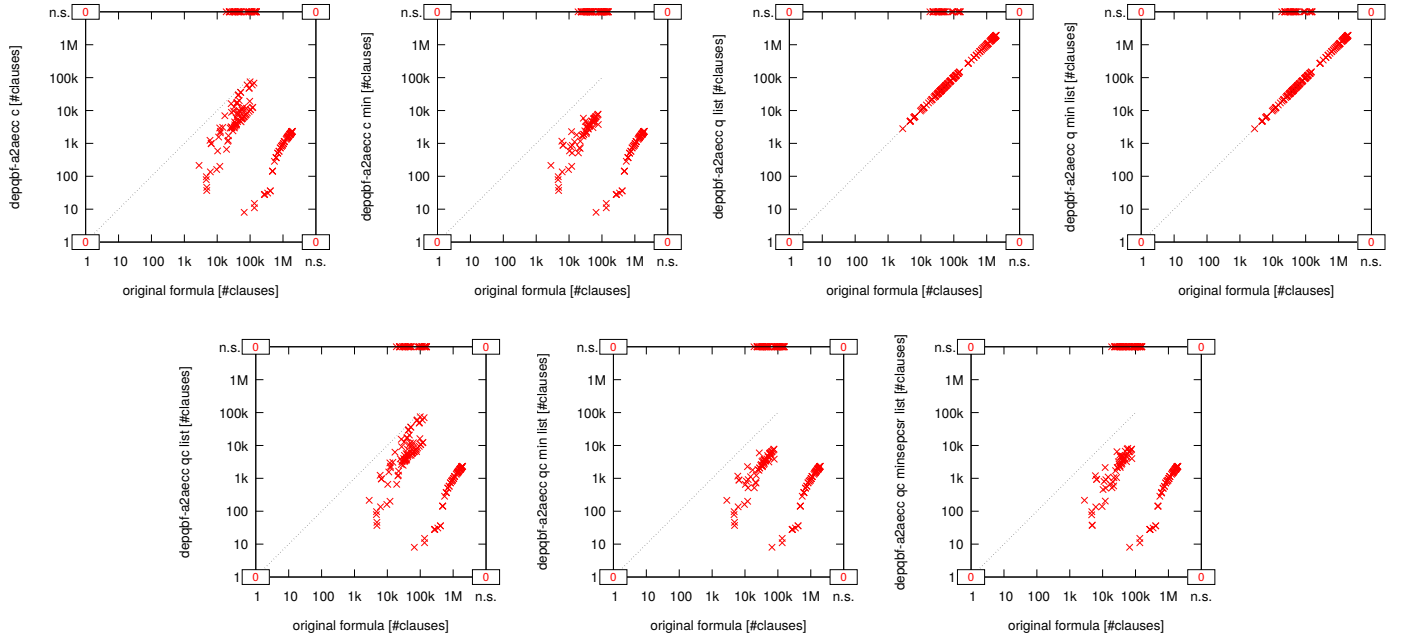


Fig. 216: Suite Miller-Marin ($n = 189$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

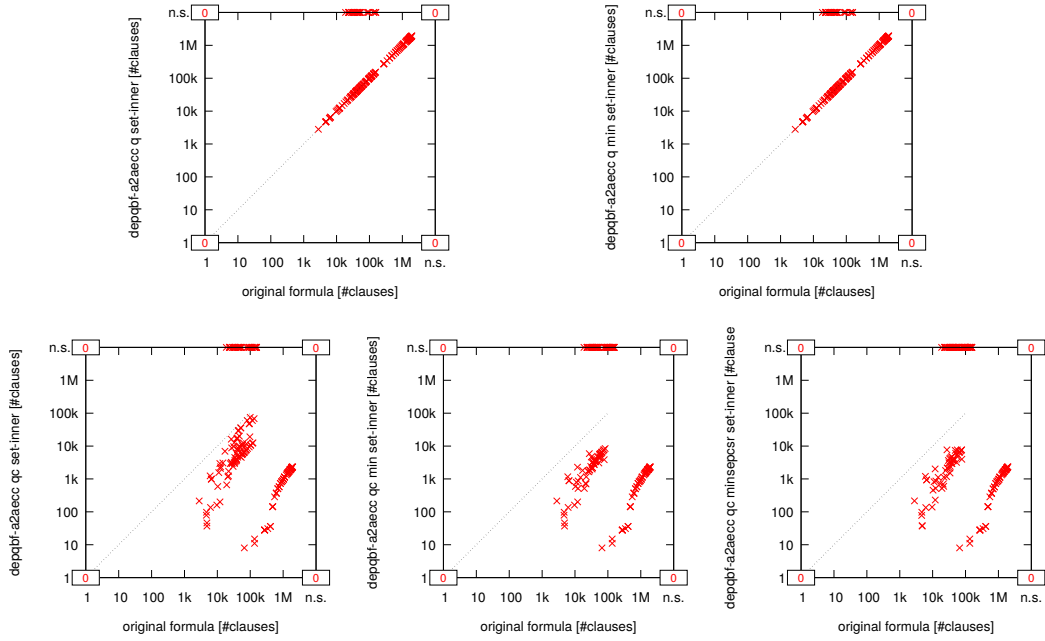


Fig. 217: Suite Miller-Marin ($n = 189$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

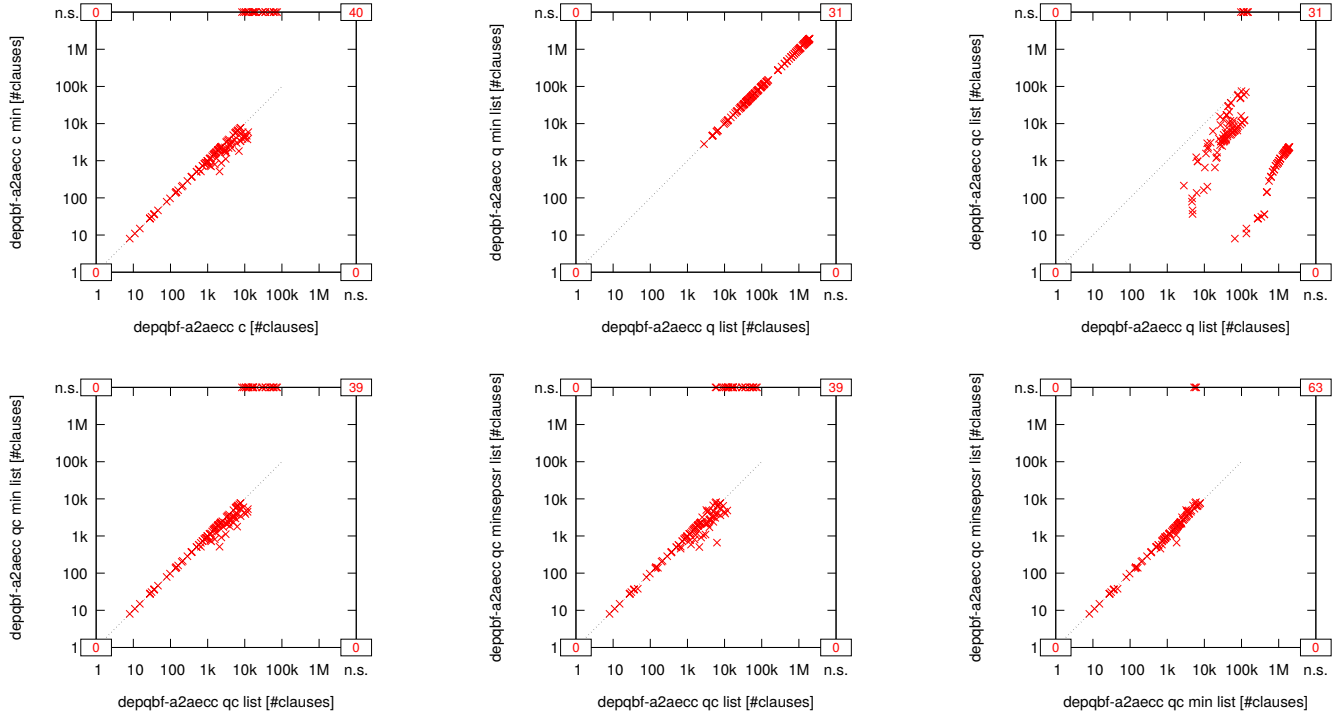


Fig. 218: Suite Miller-Marin ($n = 189$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

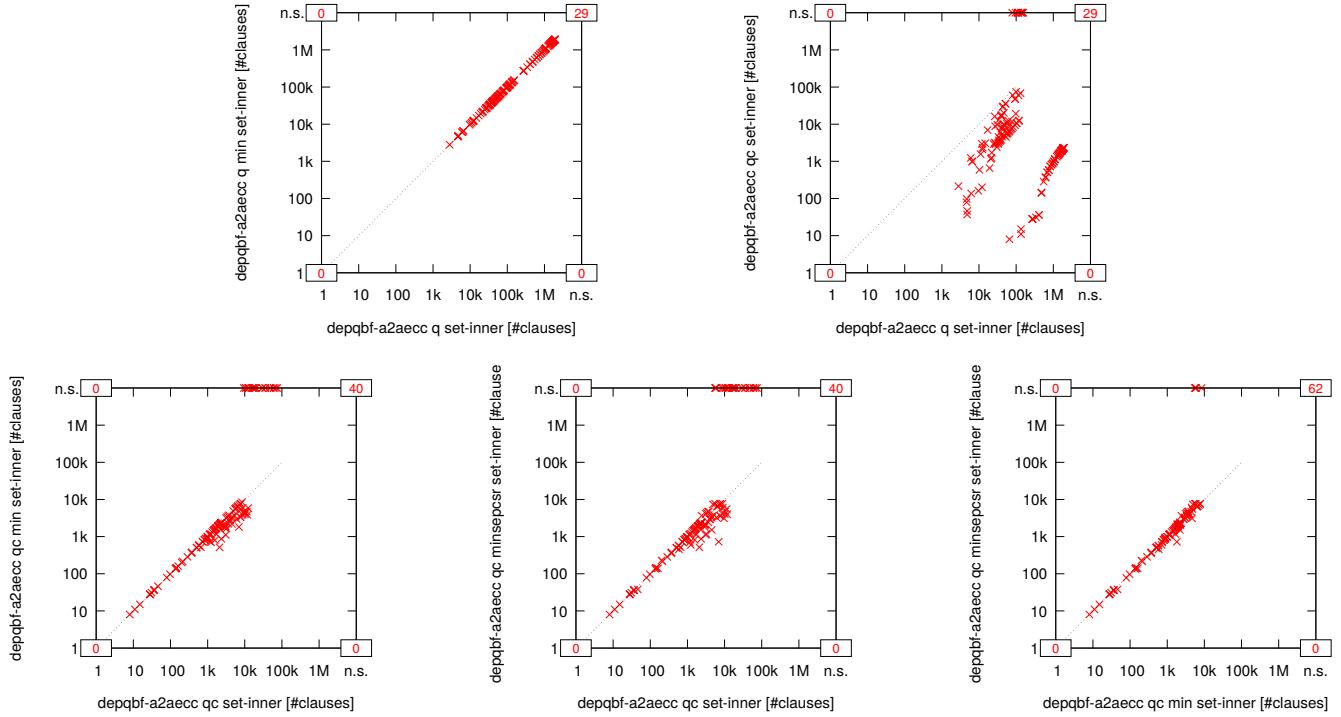


Fig. 219: Suite Miller-Marin ($n = 189$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

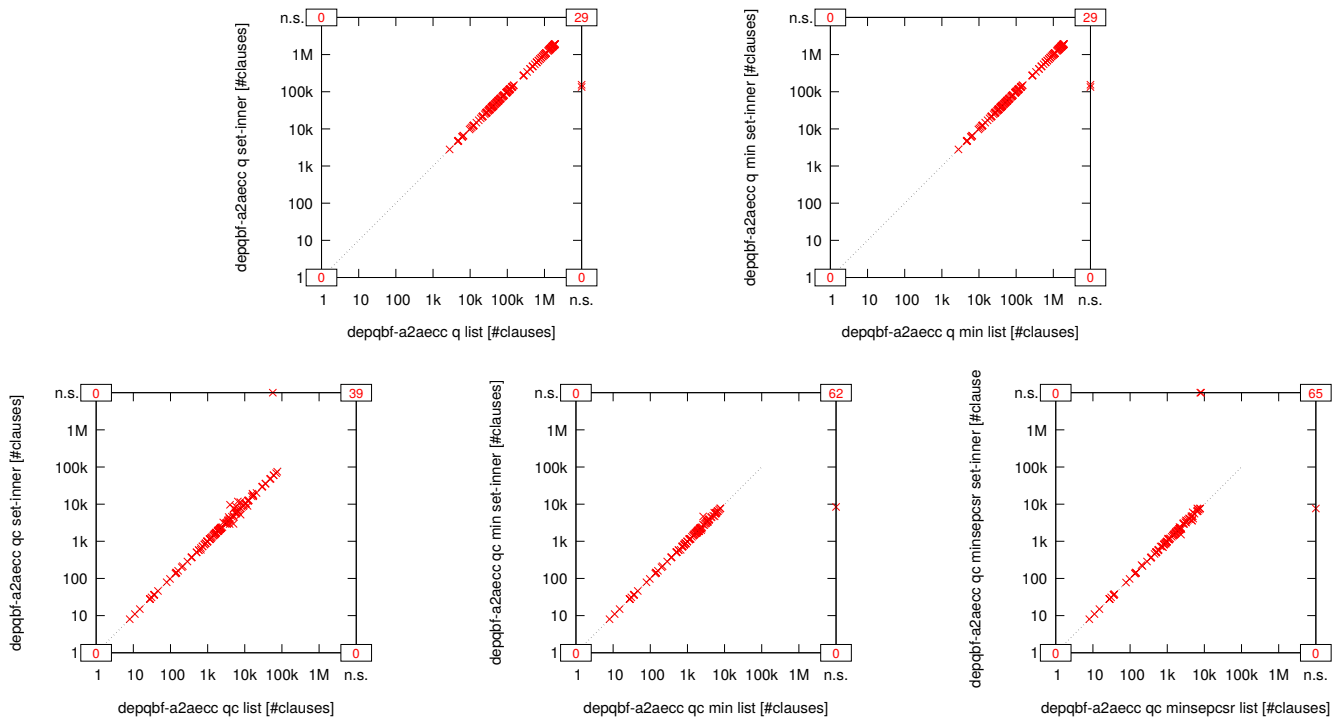


Fig. 220: Suite Miller-Marin ($n = 189$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

31) Miller-Scholl-Becker ($n = 160$):

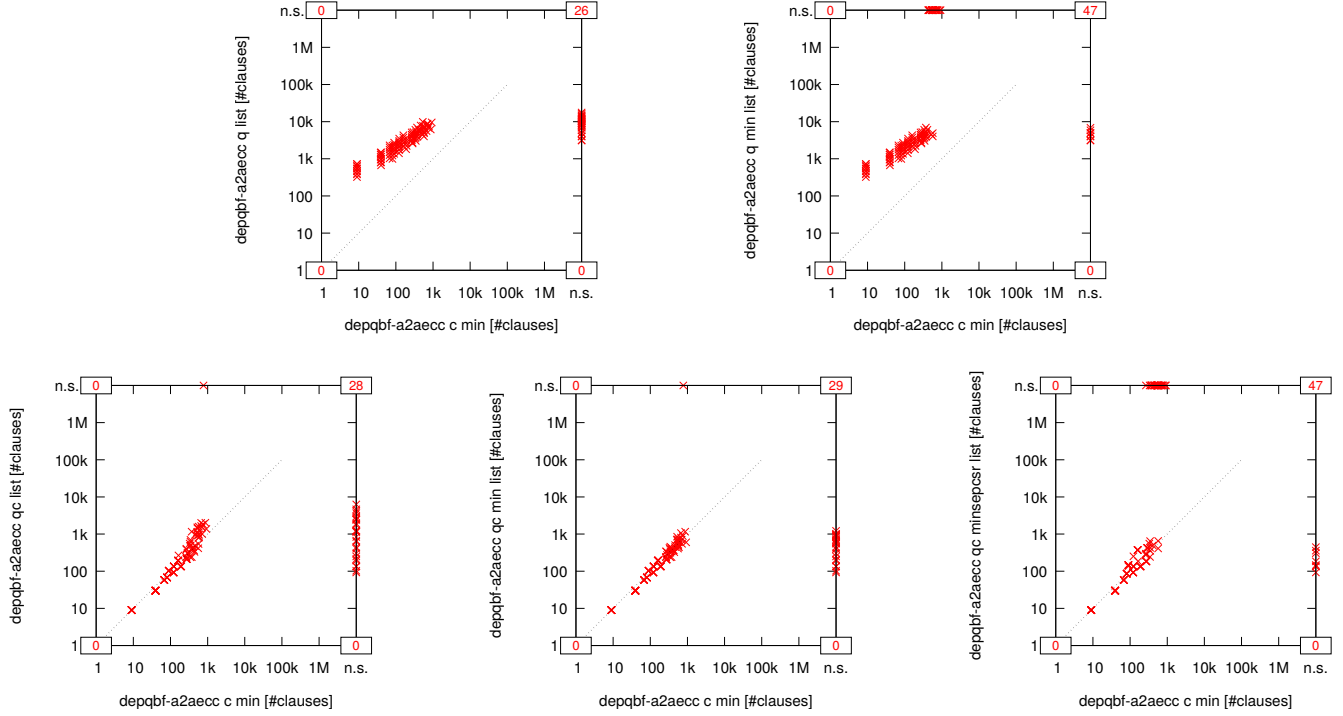


Fig. 221: Suite Miller-Scholl-Becker ($n = 160$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

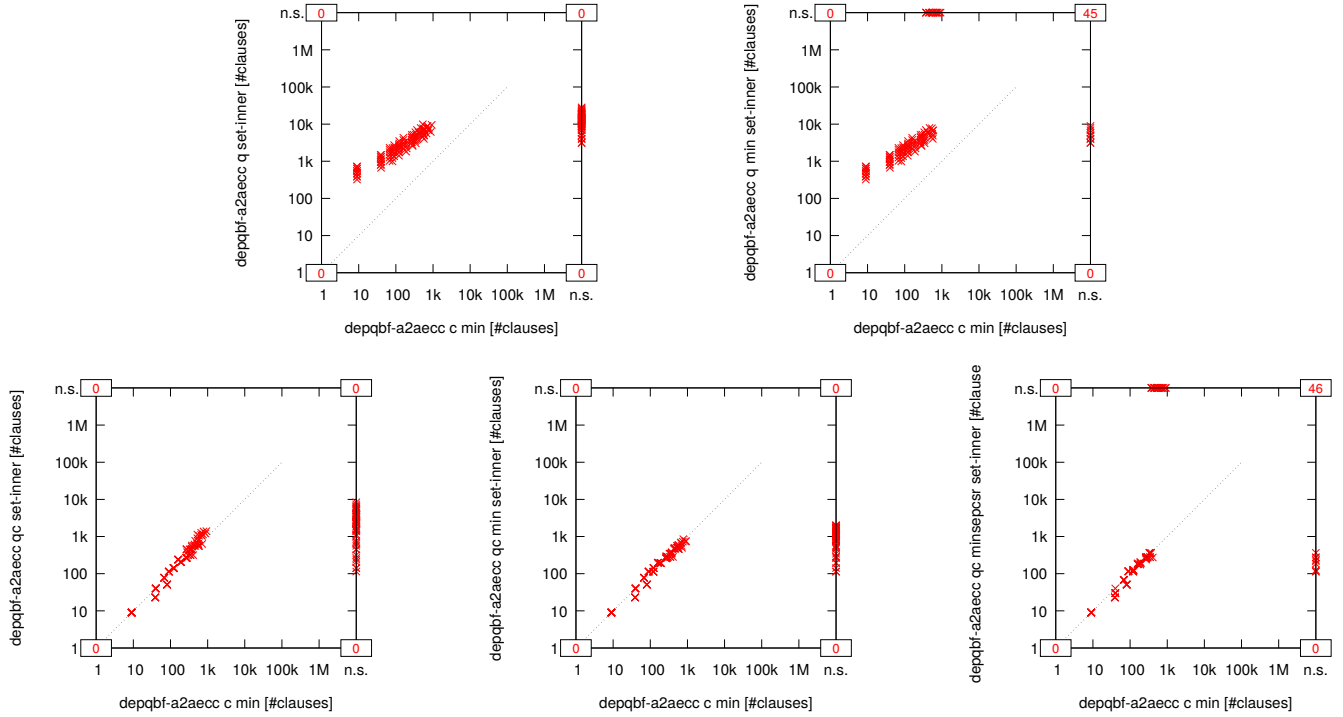


Fig. 222: Suite Miller-Scholl-Becker ($n = 160$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

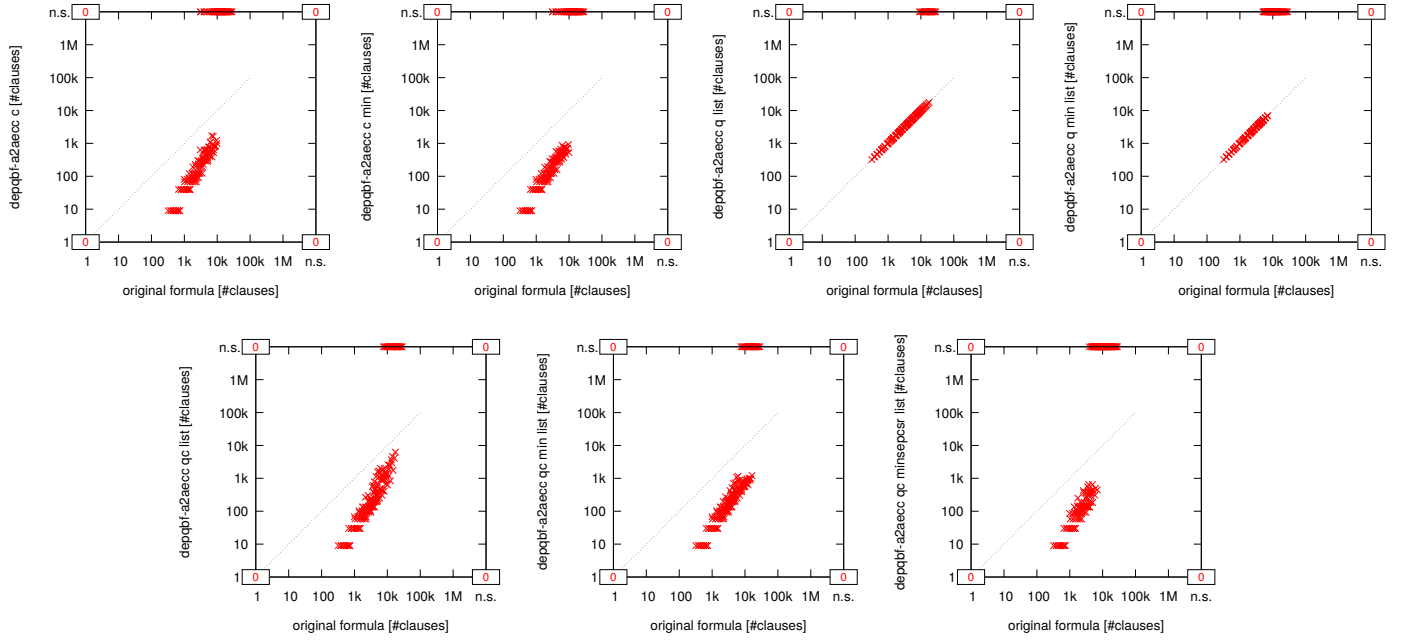


Fig. 223: Suite Miller-Scholl-Becker ($n = 160$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

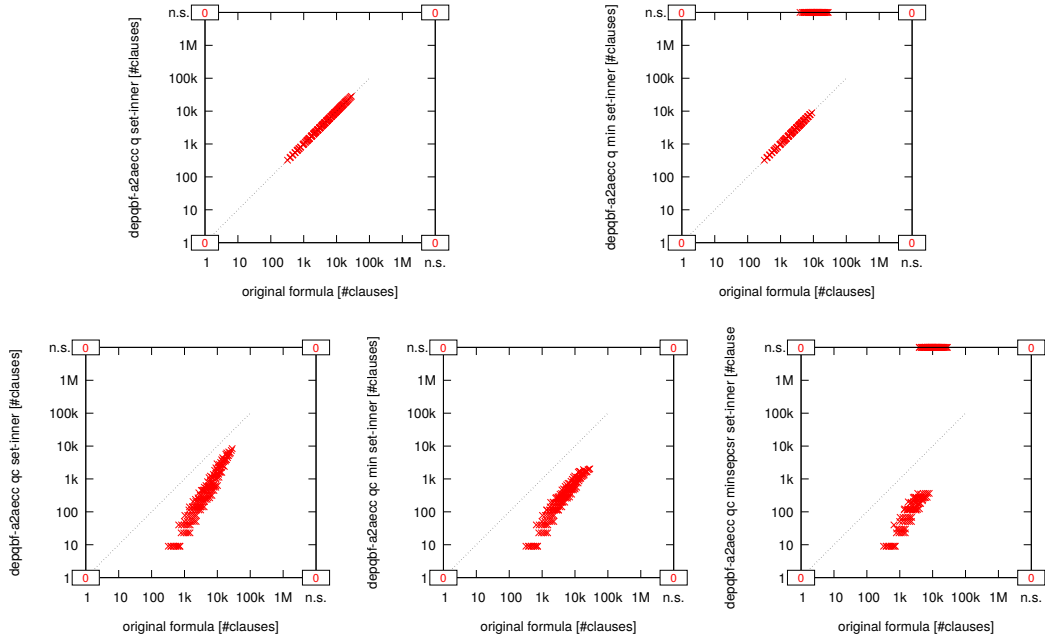


Fig. 224: Suite Miller-Scholl-Becker ($n = 160$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

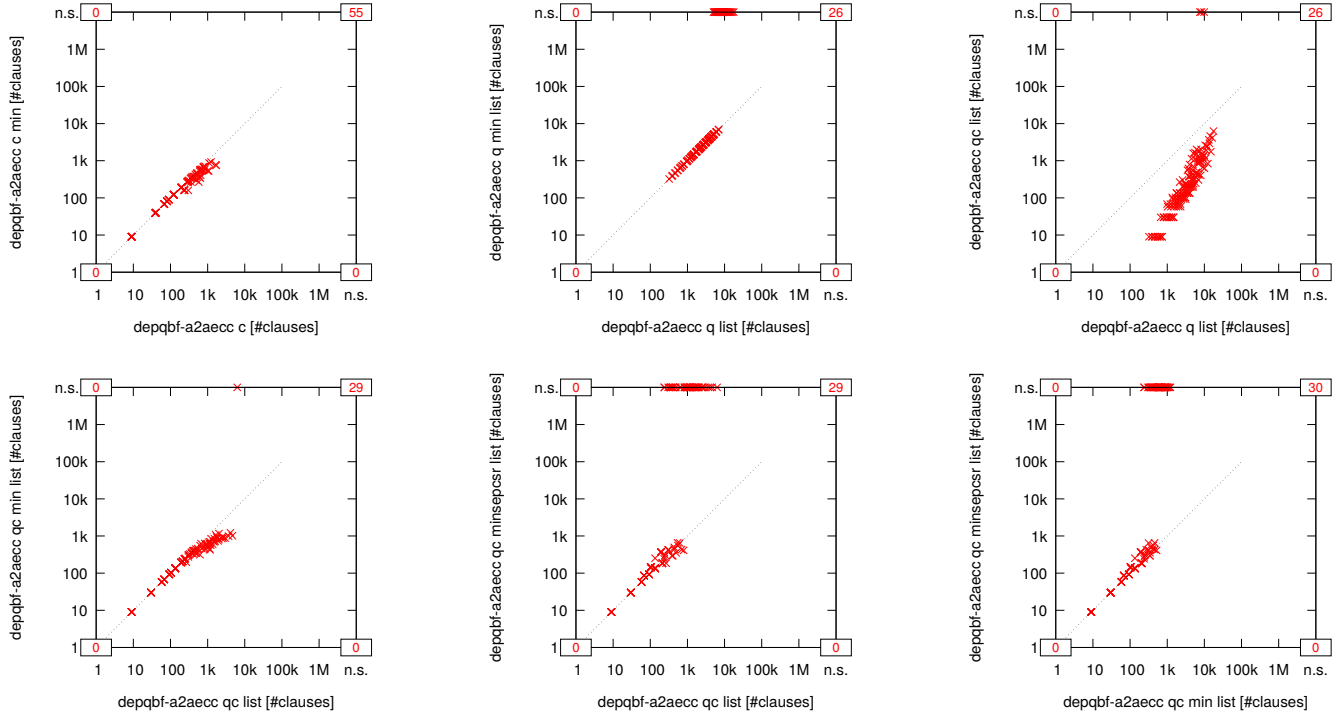


Fig. 225: Suite Miller-Scholl-Becker ($n = 160$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

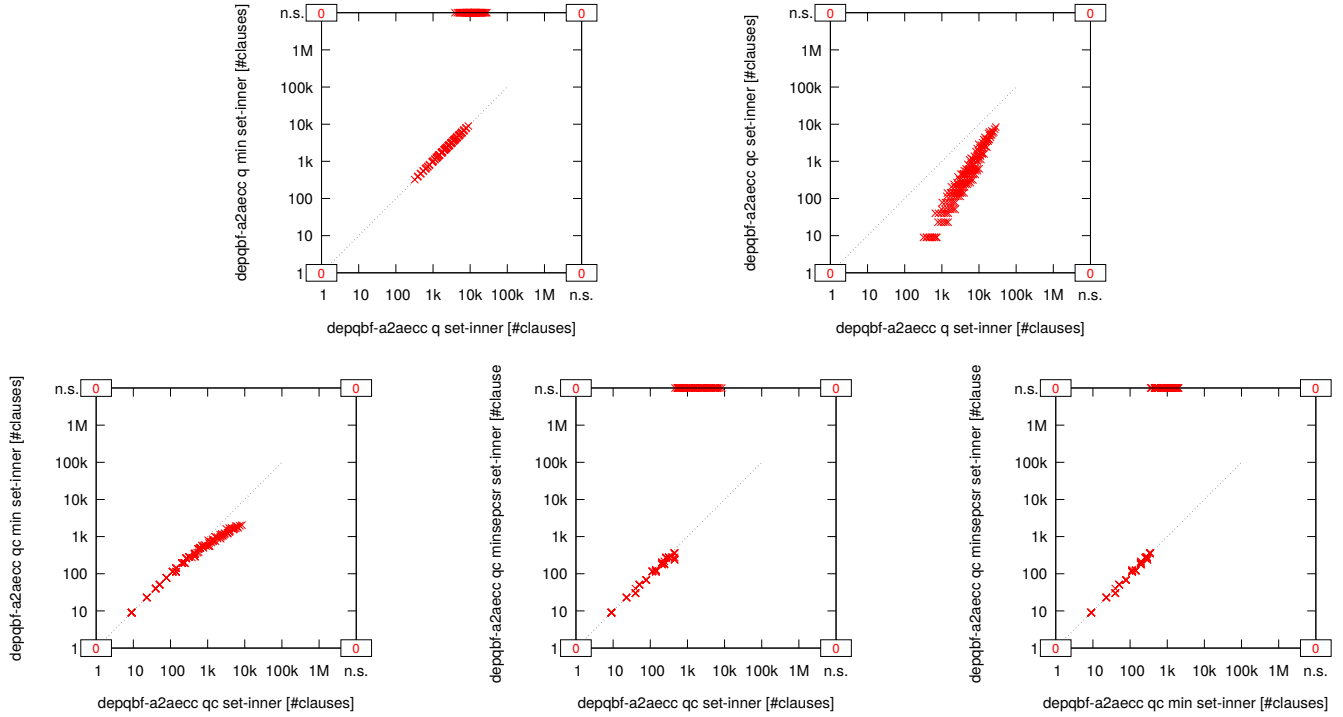


Fig. 226: Suite Miller-Scholl-Becker ($n = 160$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

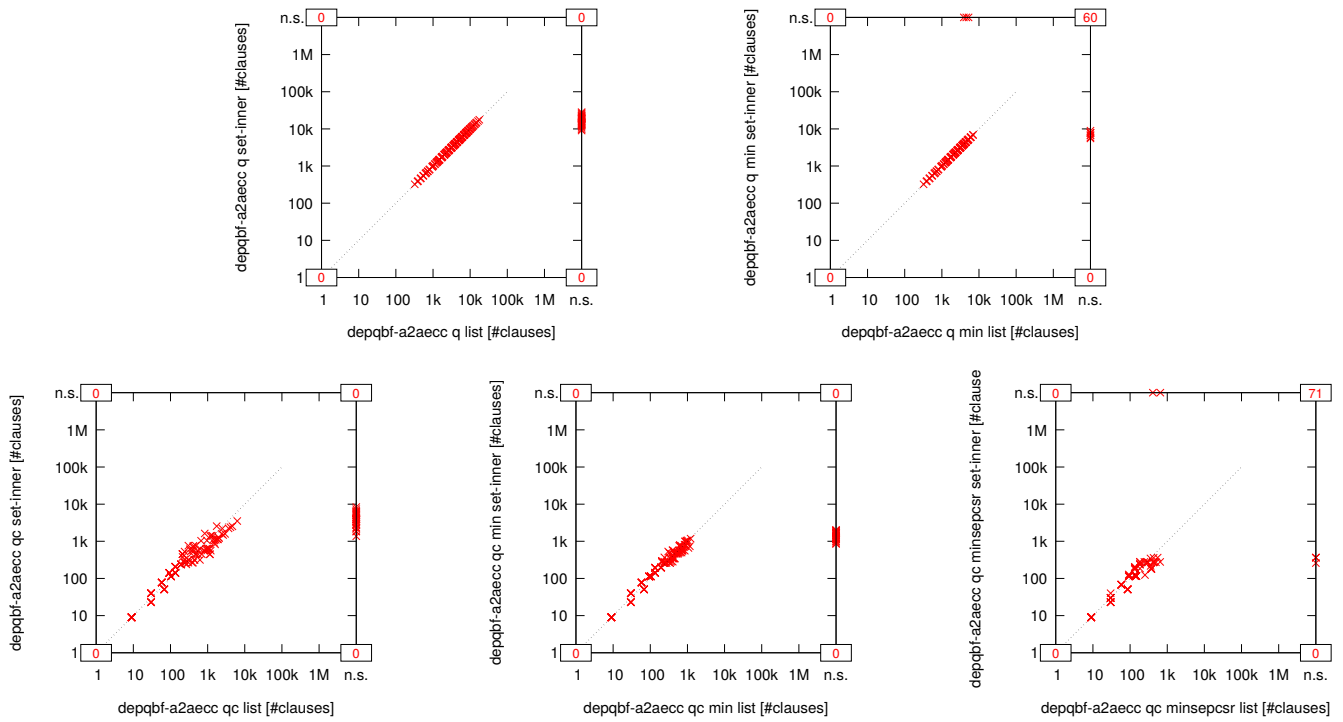


Fig. 227: Suite Miller-Scholl-Becker ($n = 160$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

32) Mneimneh-Sakallah ($n = 44$):

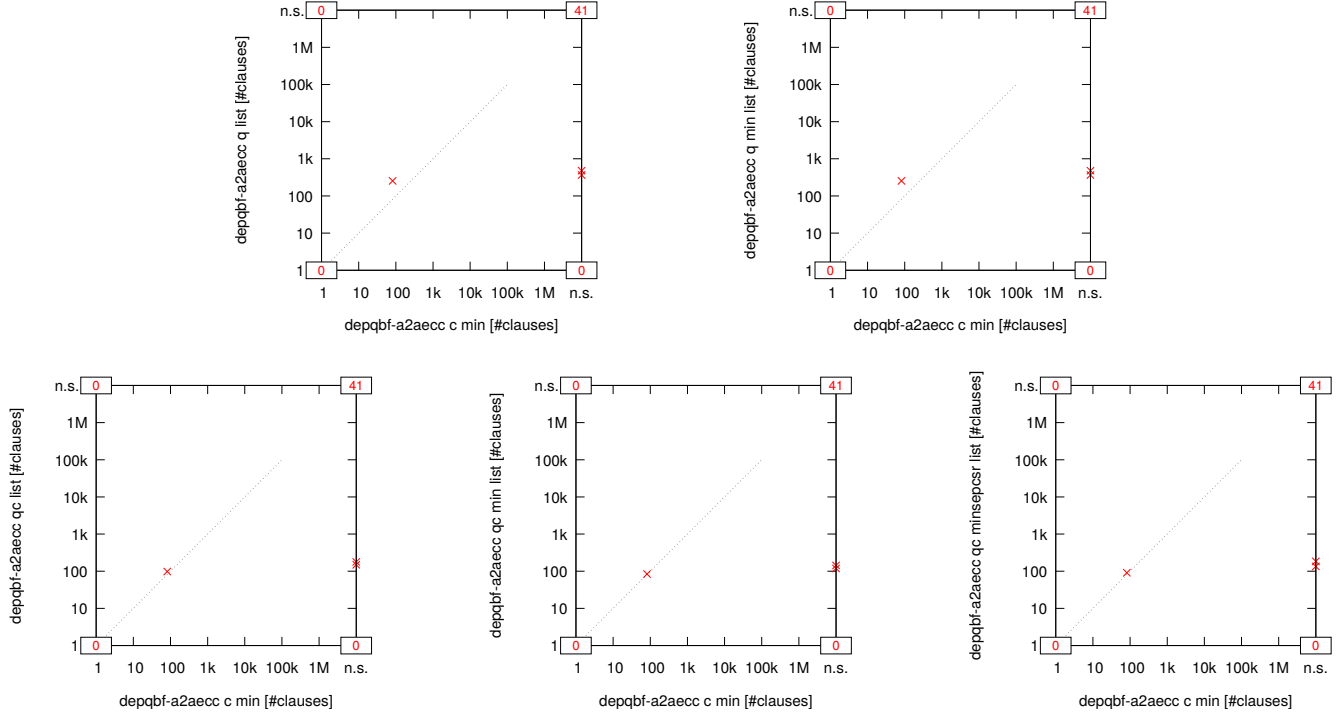


Fig. 228: Suite Mneimneh-Sakallah ($n = 44$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

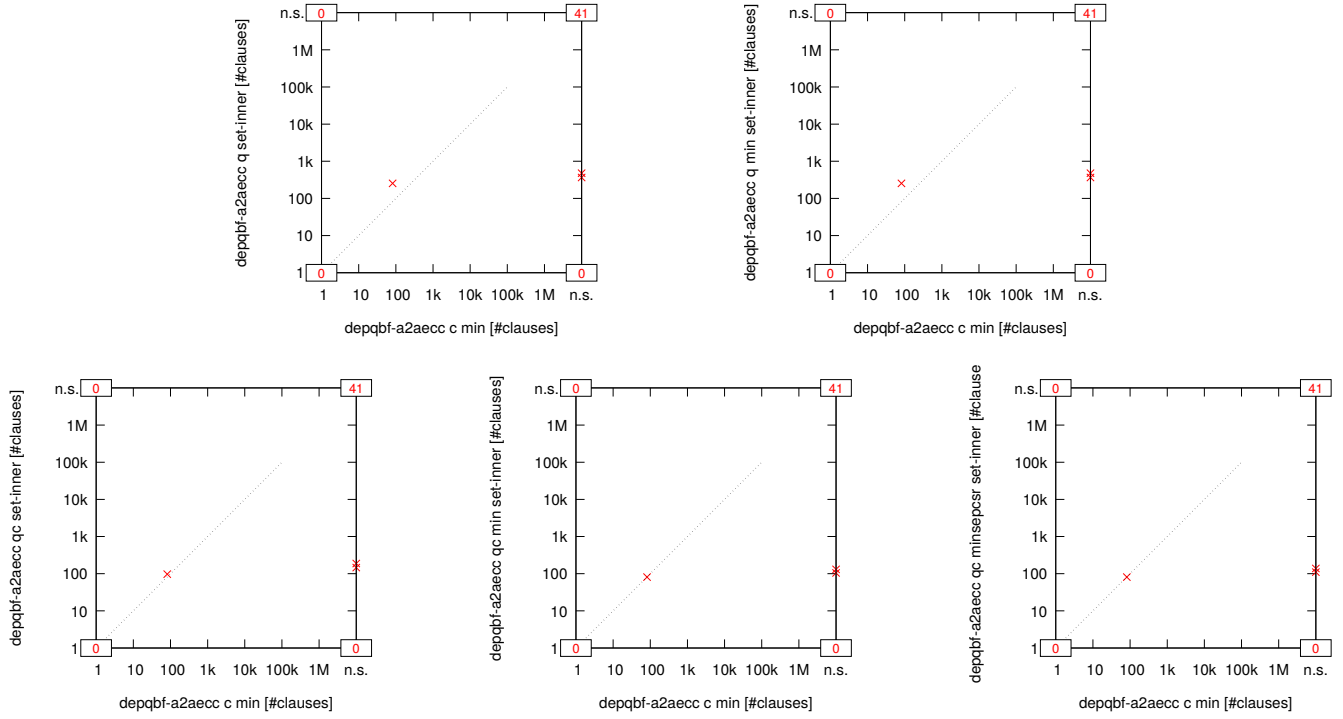


Fig. 229: Suite Mneimneh-Sakallah ($n = 44$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

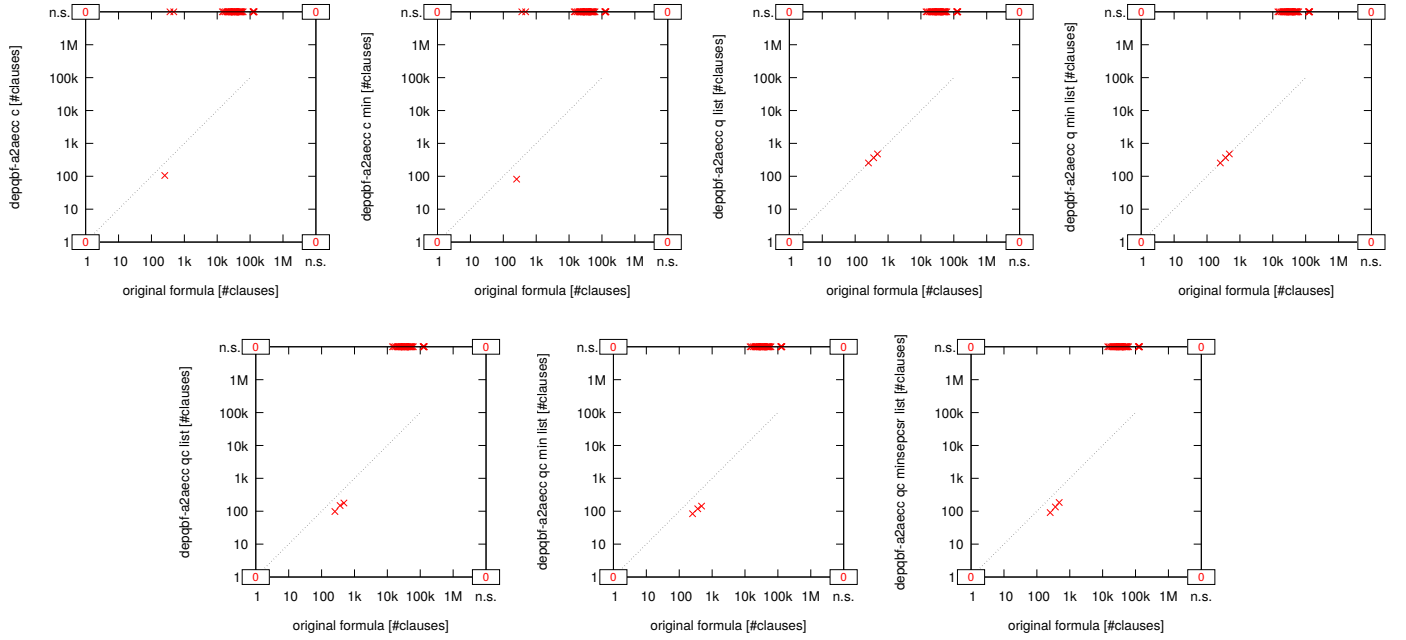


Fig. 230: Suite Mneimneh-Sakallah ($n = 44$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

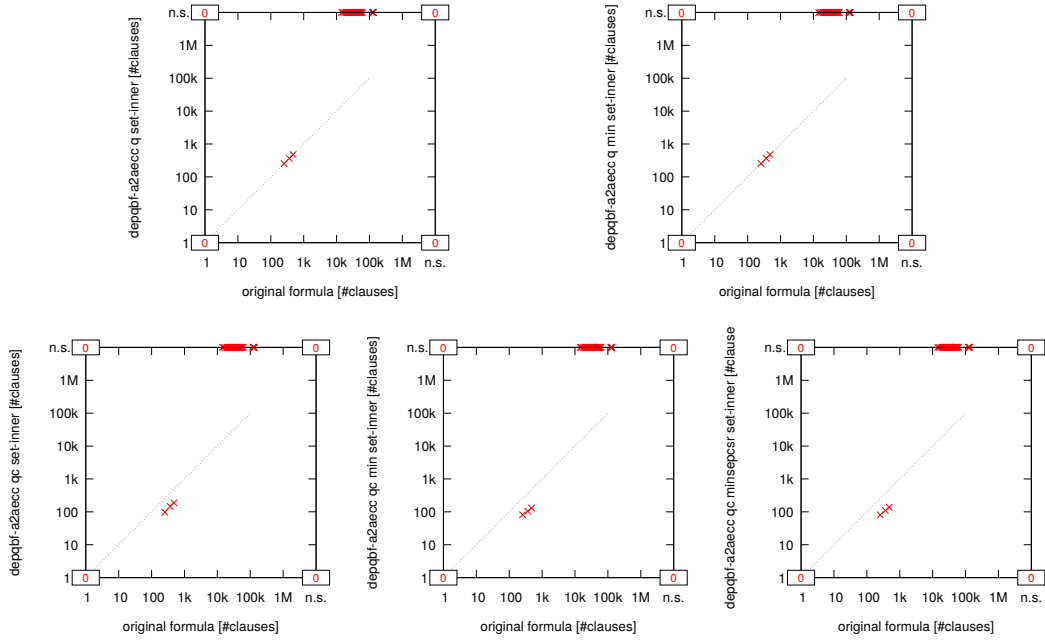


Fig. 231: Suite Mneimneh-Sakallah ($n = 44$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

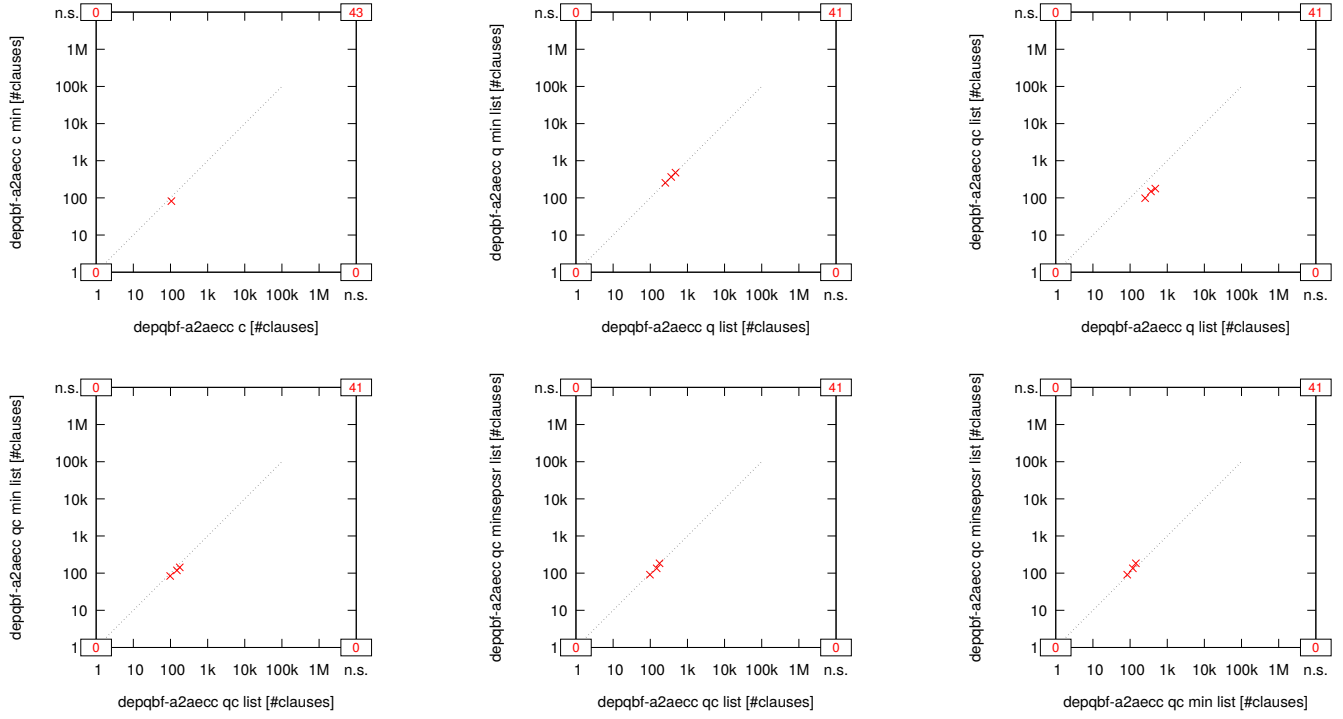


Fig. 232: Suite Mneimneh-Sakallah ($n = 44$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

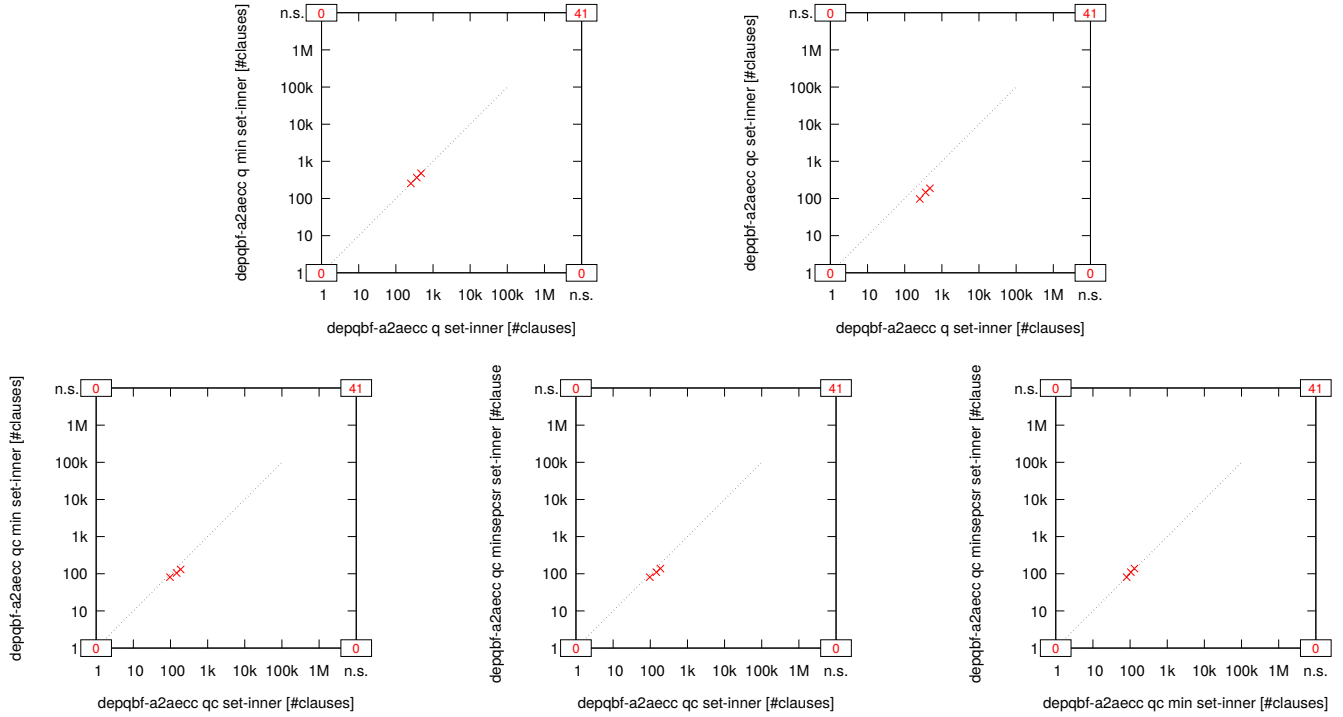


Fig. 233: Suite Mneimneh-Sakallah ($n = 44$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

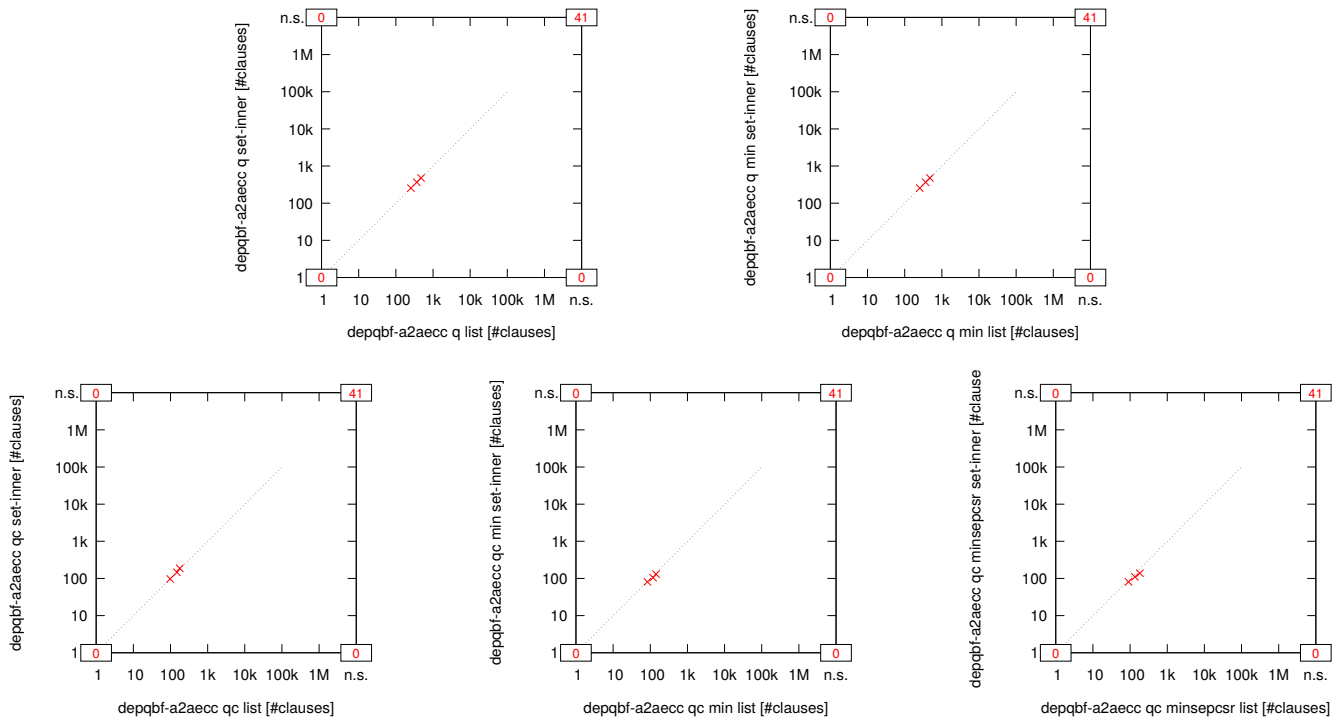


Fig. 234: Suite Mneimneh-Sakallah ($n = 44$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

33) Narizzano ($n = 78$):

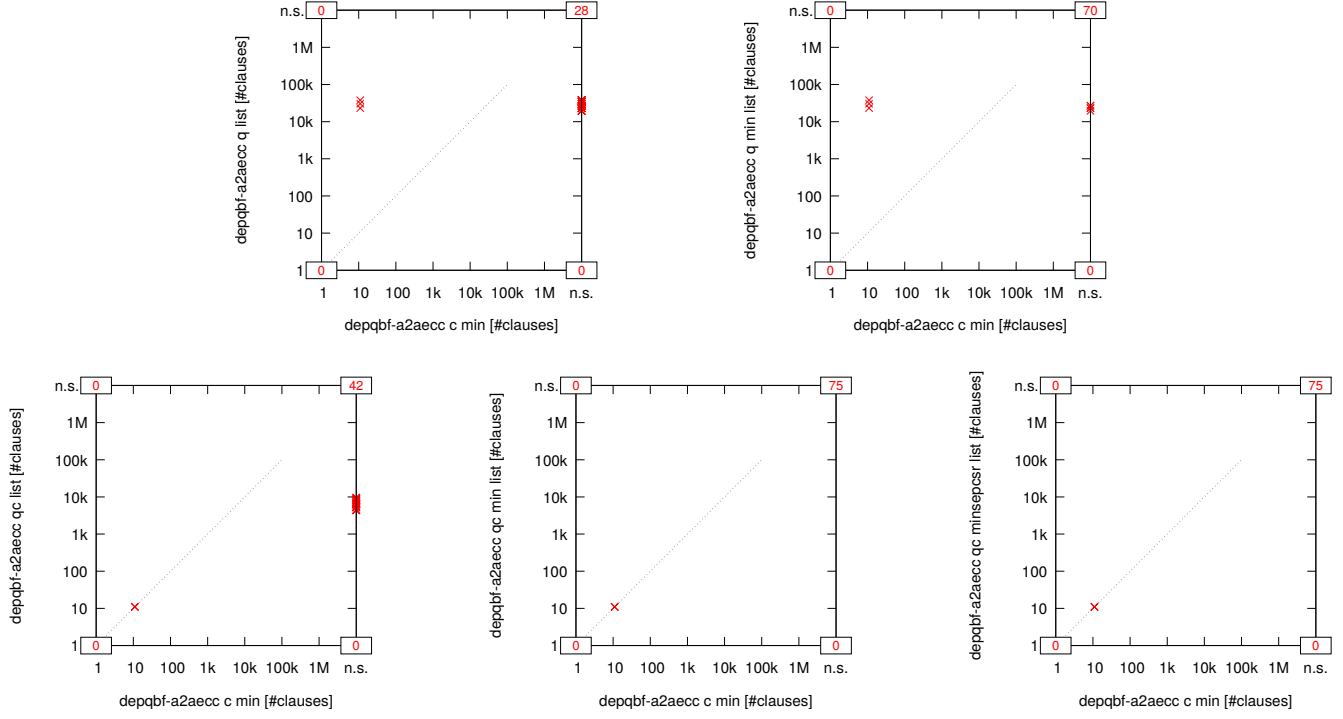


Fig. 235: Suite Narizzano ($n = 78$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

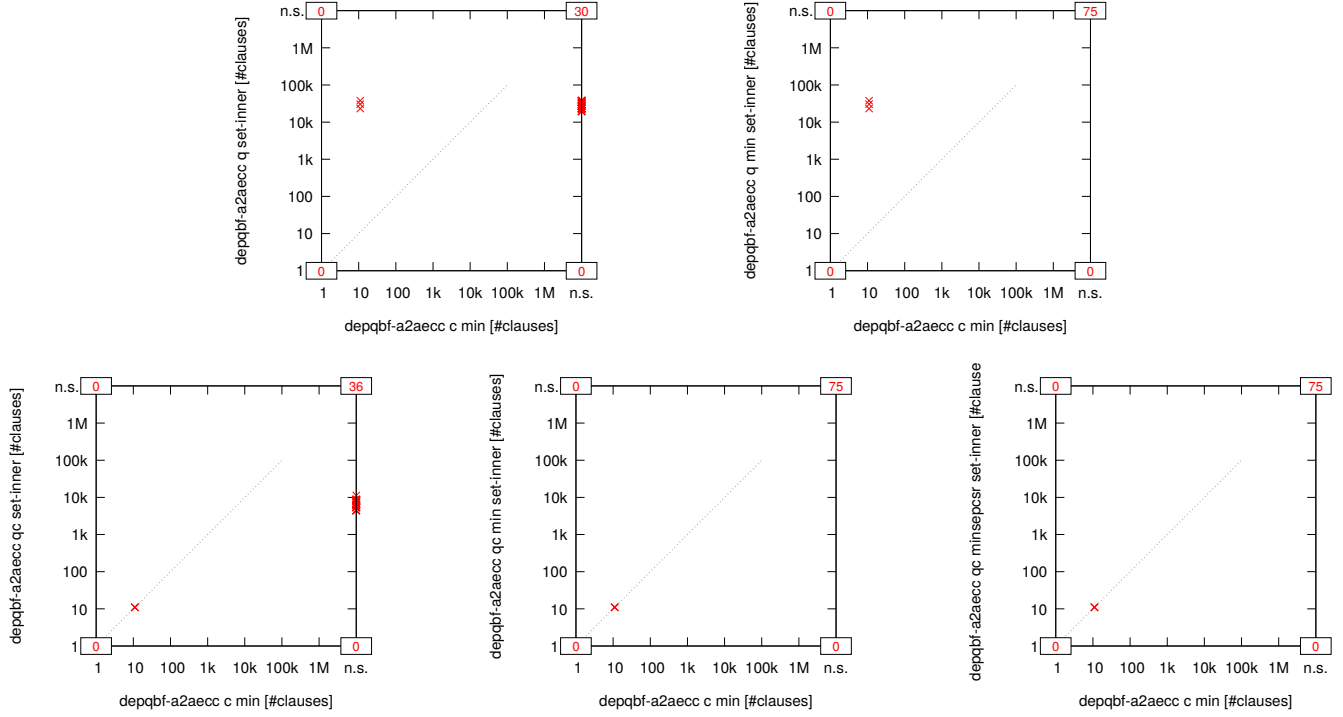


Fig. 236: Suite Narizzano ($n = 78$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

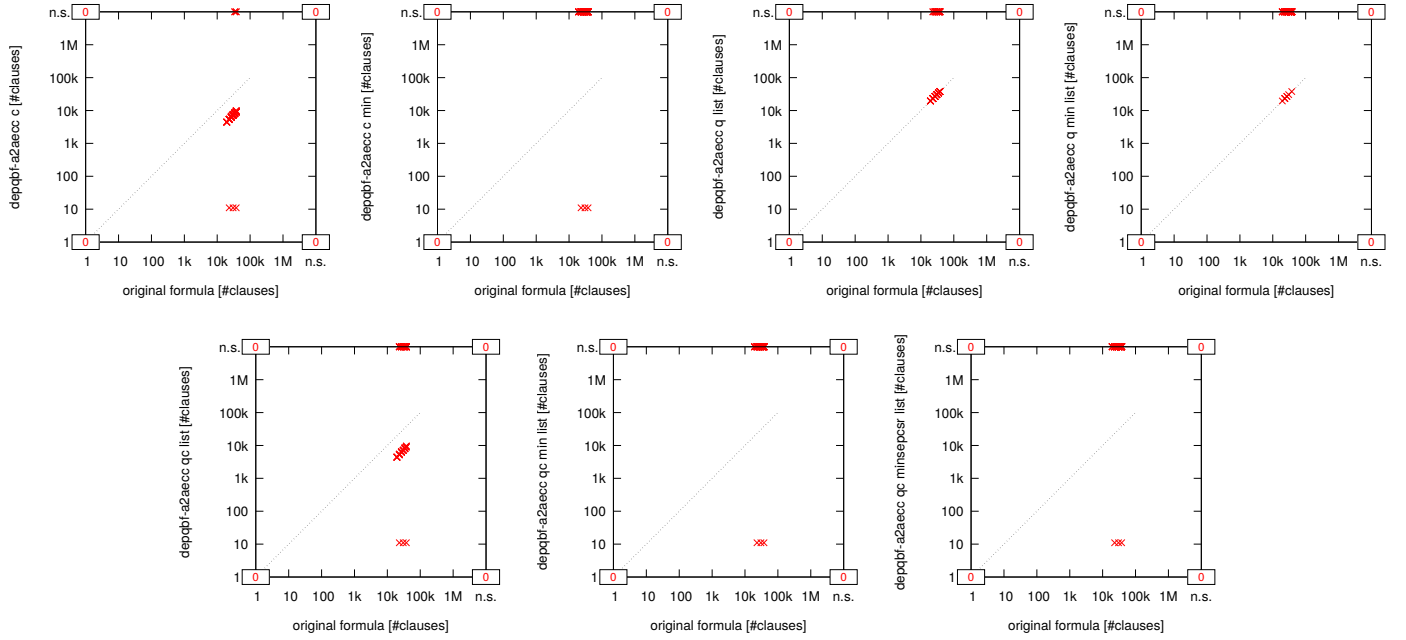


Fig. 237: Suite Narizzano ($n = 78$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

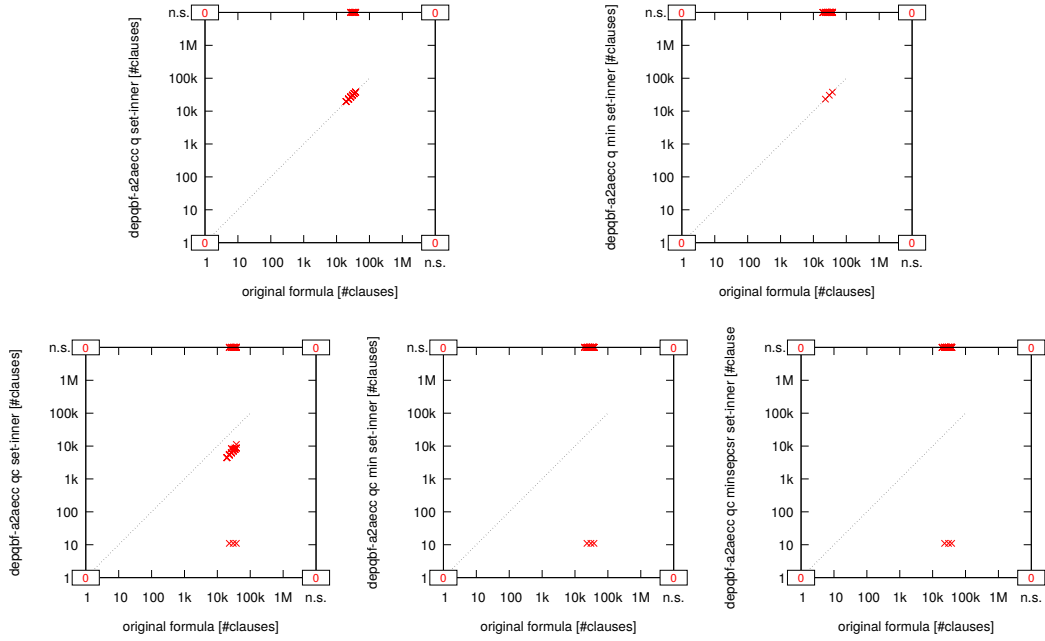


Fig. 238: Suite Narizzano ($n = 78$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

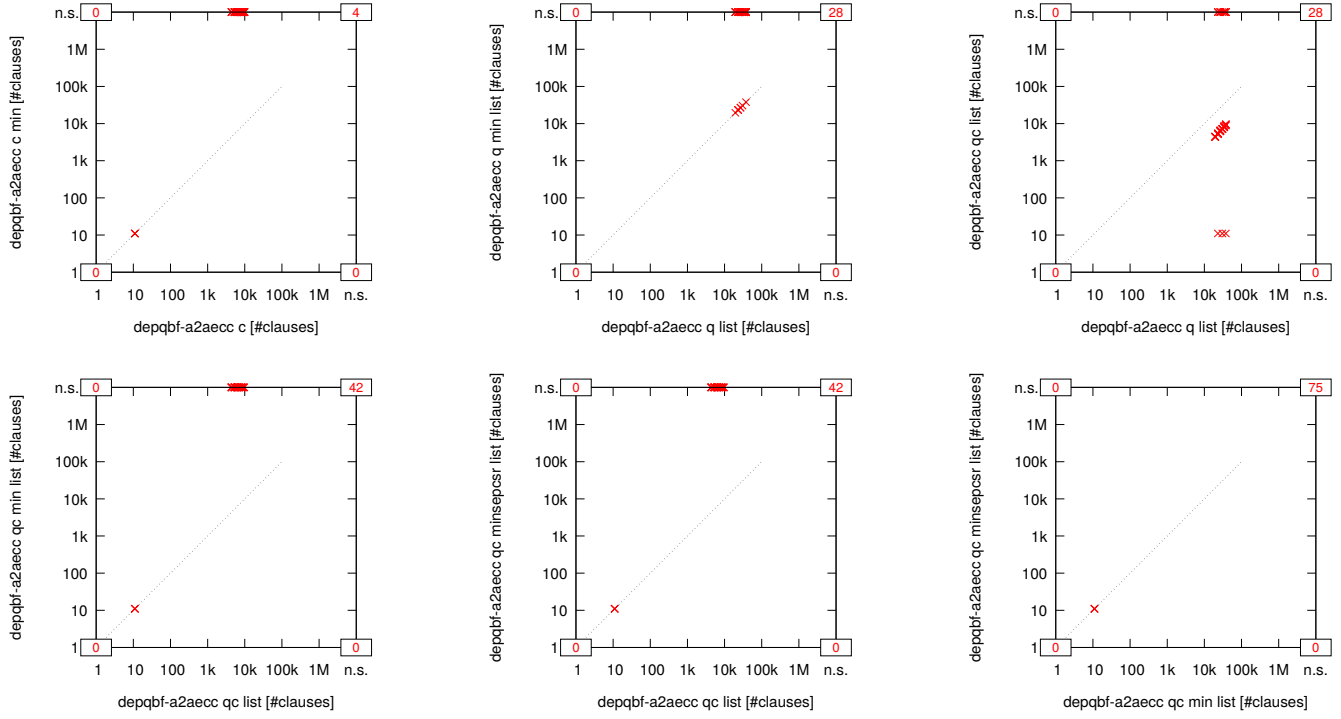


Fig. 239: Suite Narizzano ($n = 78$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

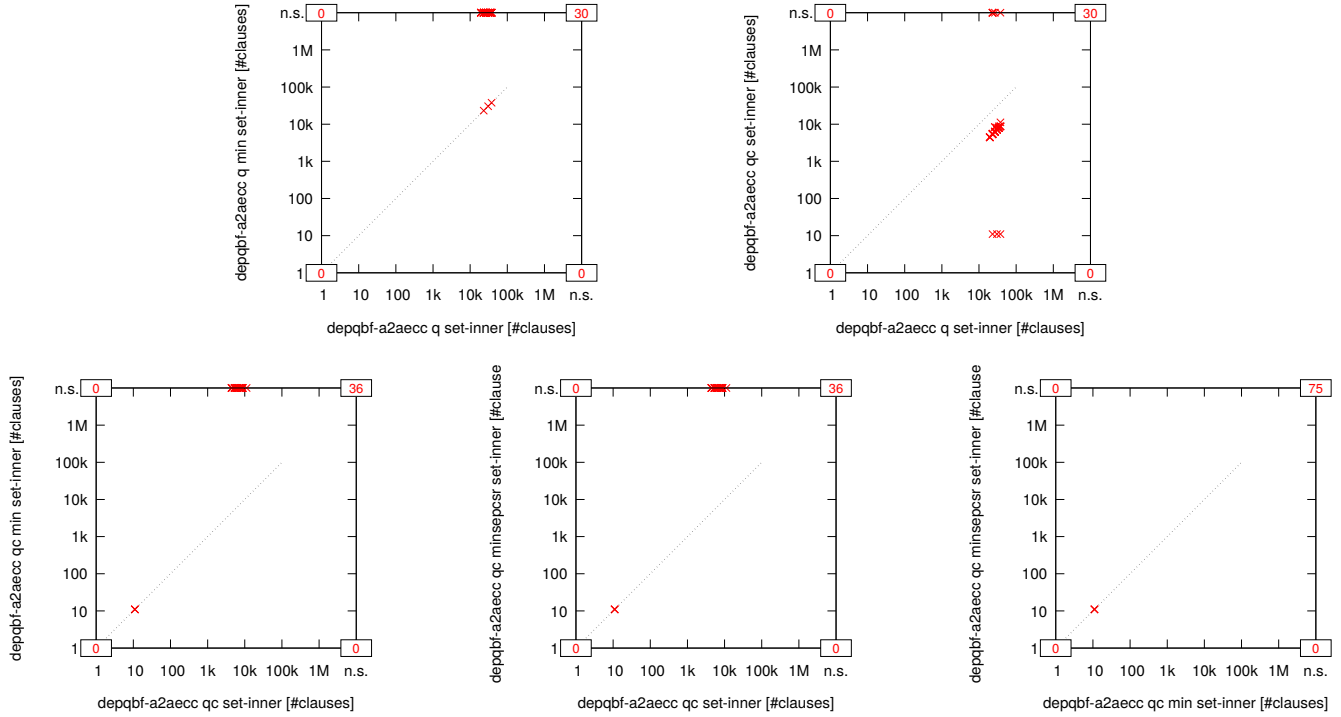


Fig. 240: Suite Narizzano ($n = 78$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

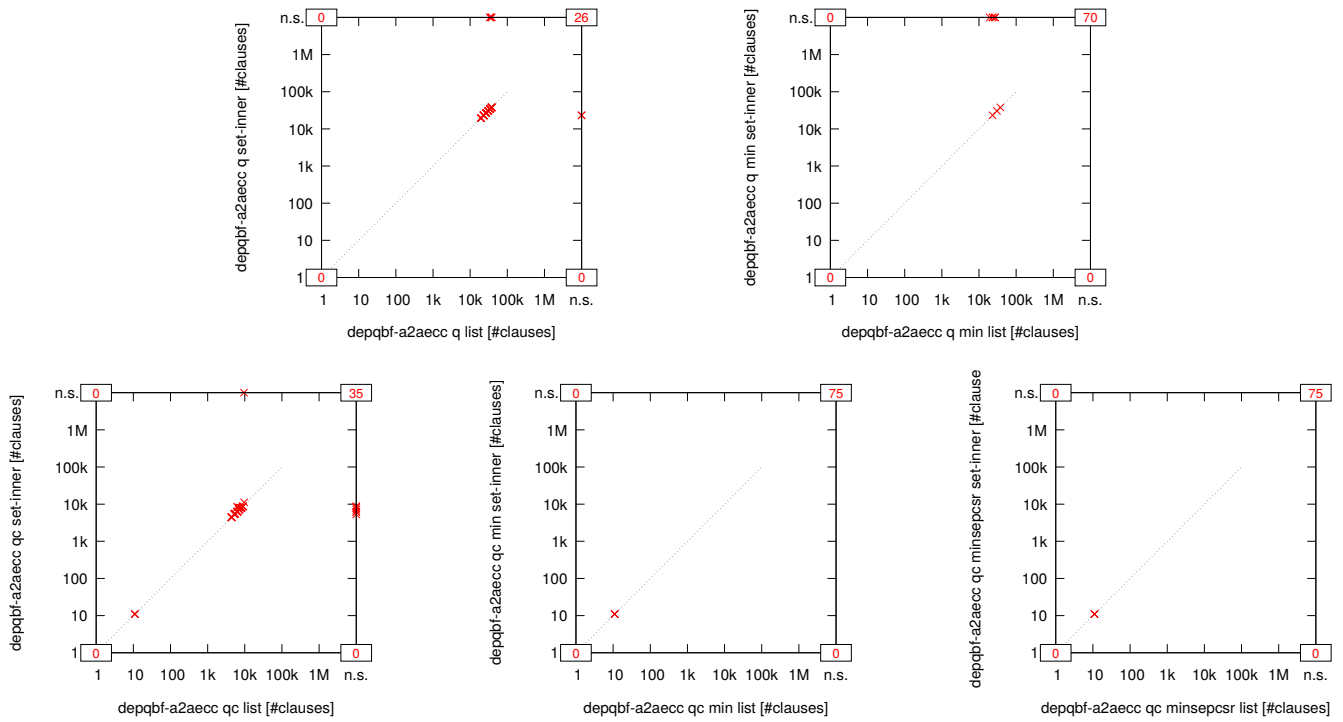


Fig. 241: Suite Narizzano ($n = 78$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

34) *Palacios* ($n = 9$):

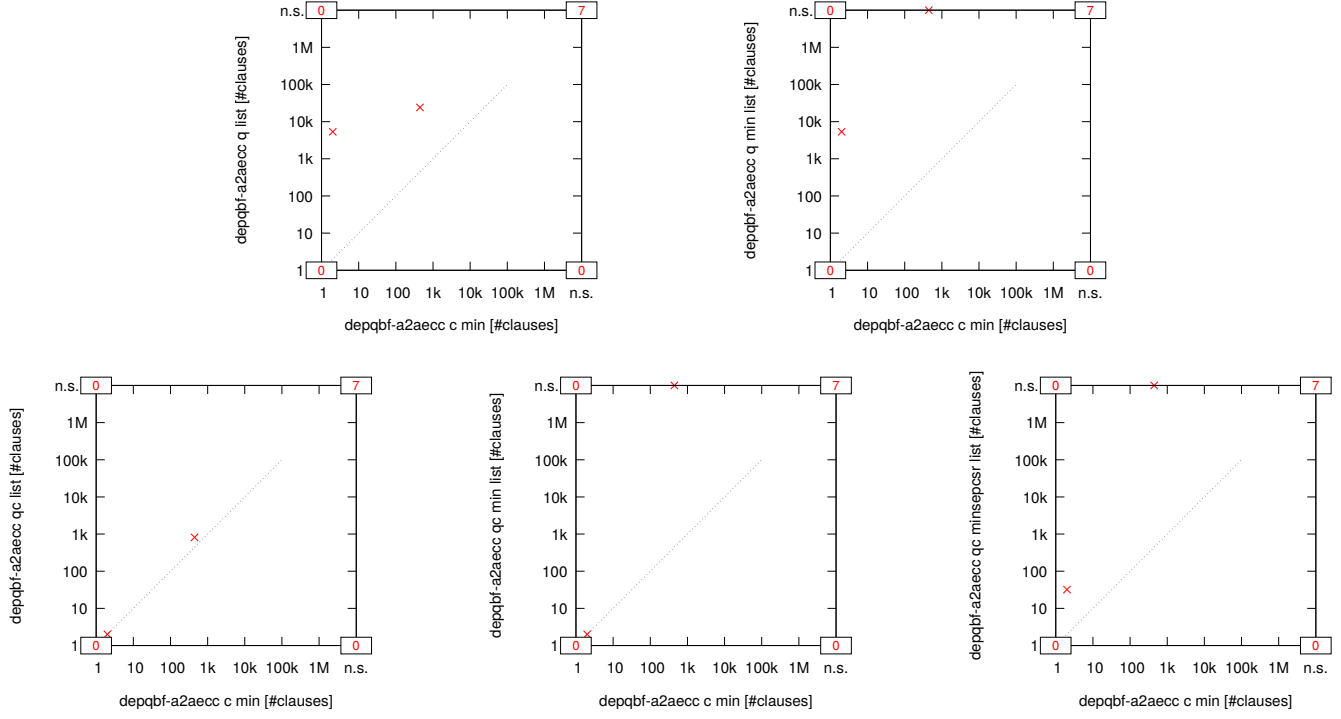


Fig. 242: Suite Palacios ($n = 9$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

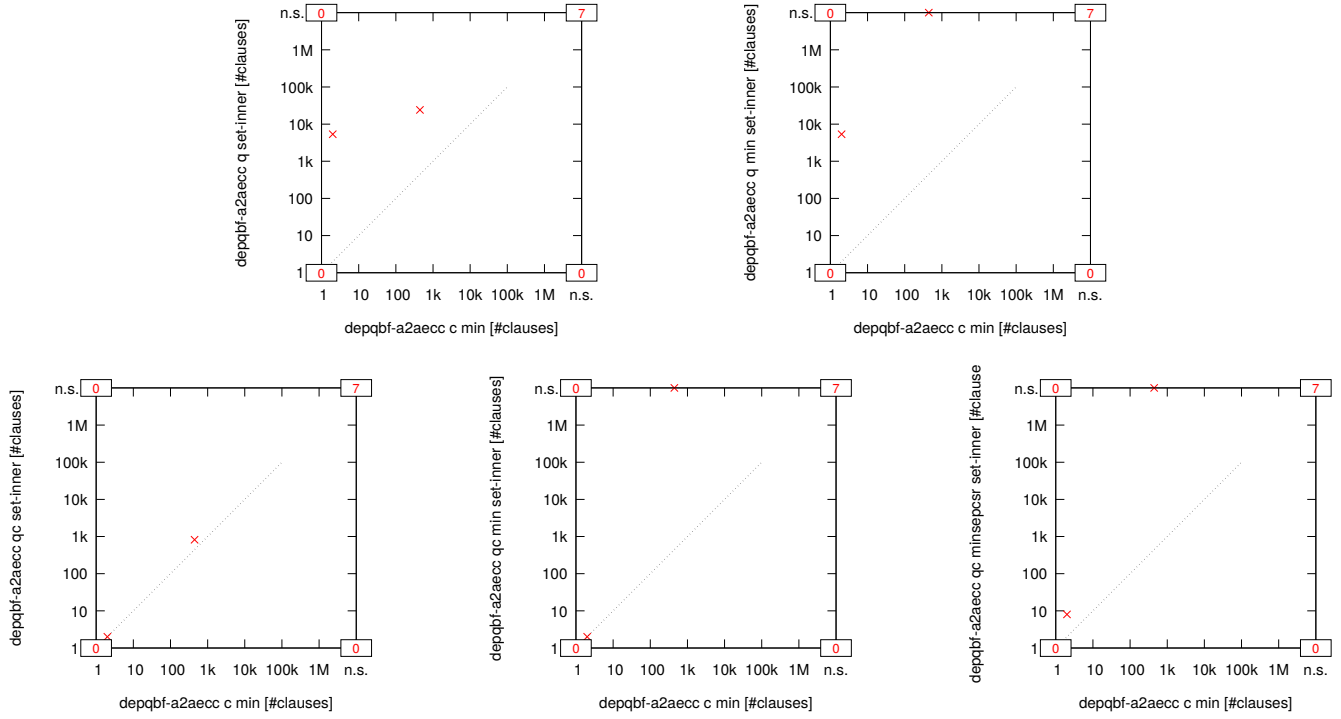


Fig. 243: Suite Palacios ($n = 9$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

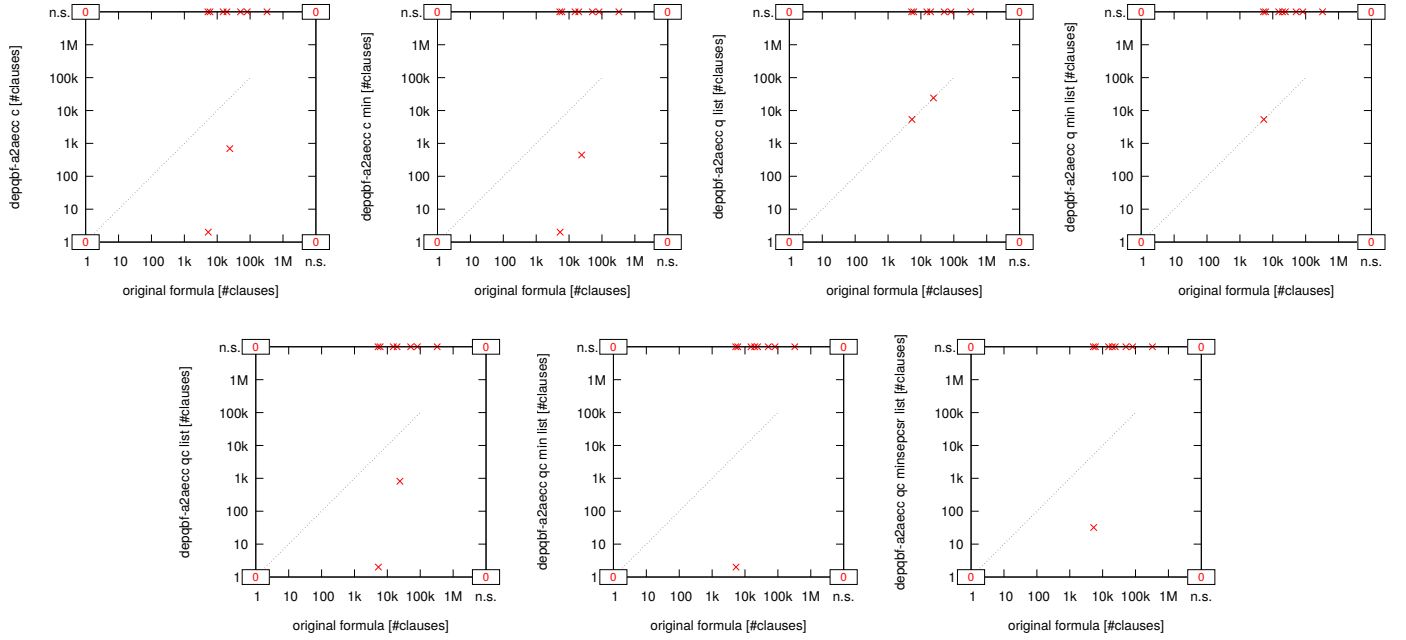


Fig. 244: Suite Palacios ($n = 9$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

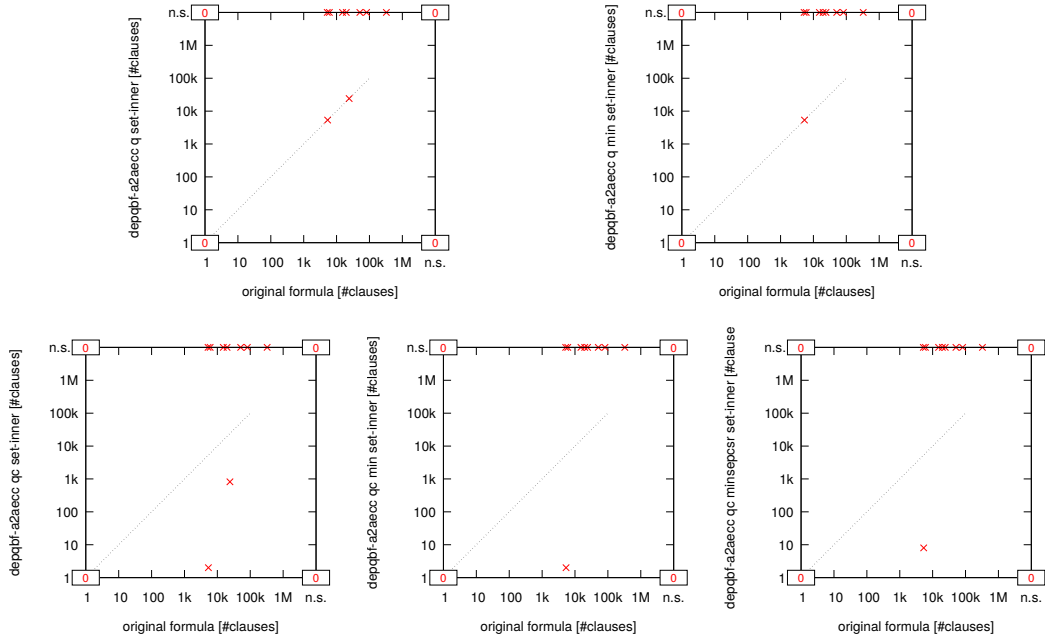


Fig. 245: Suite Palacios ($n = 9$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

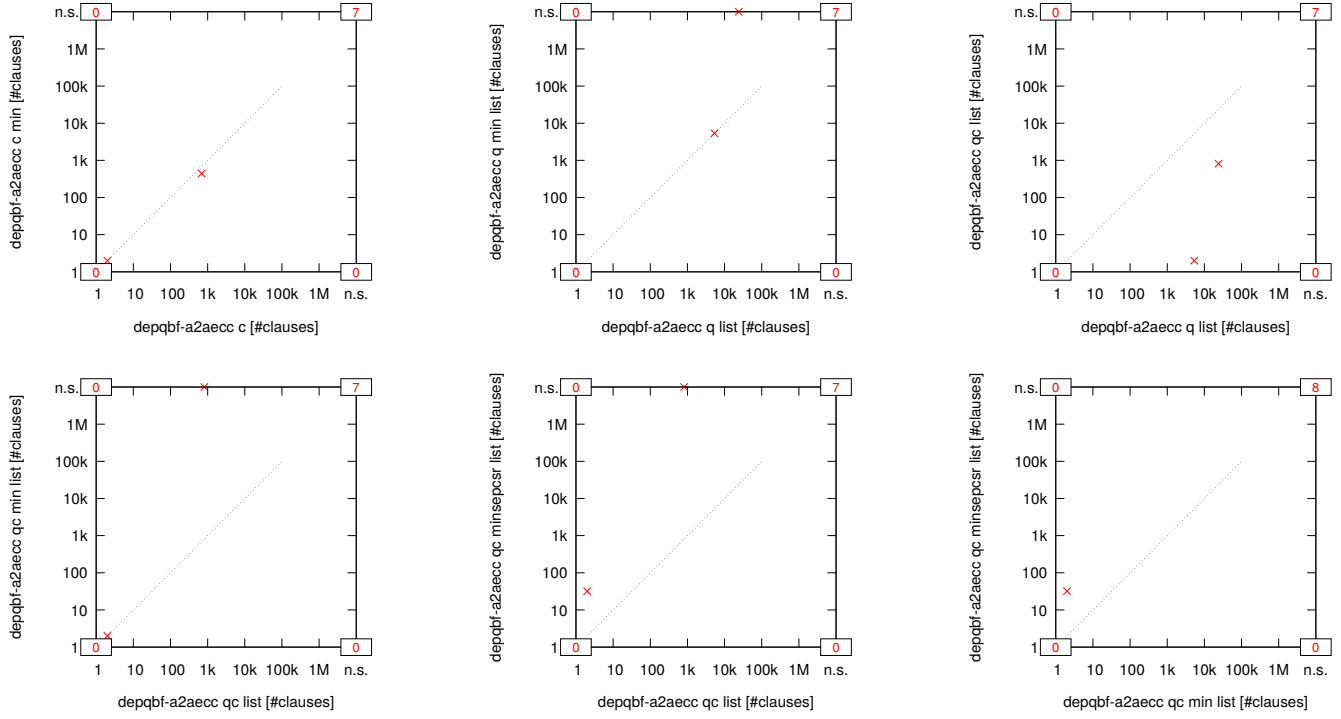


Fig. 246: Suite Palacios ($n = 9$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

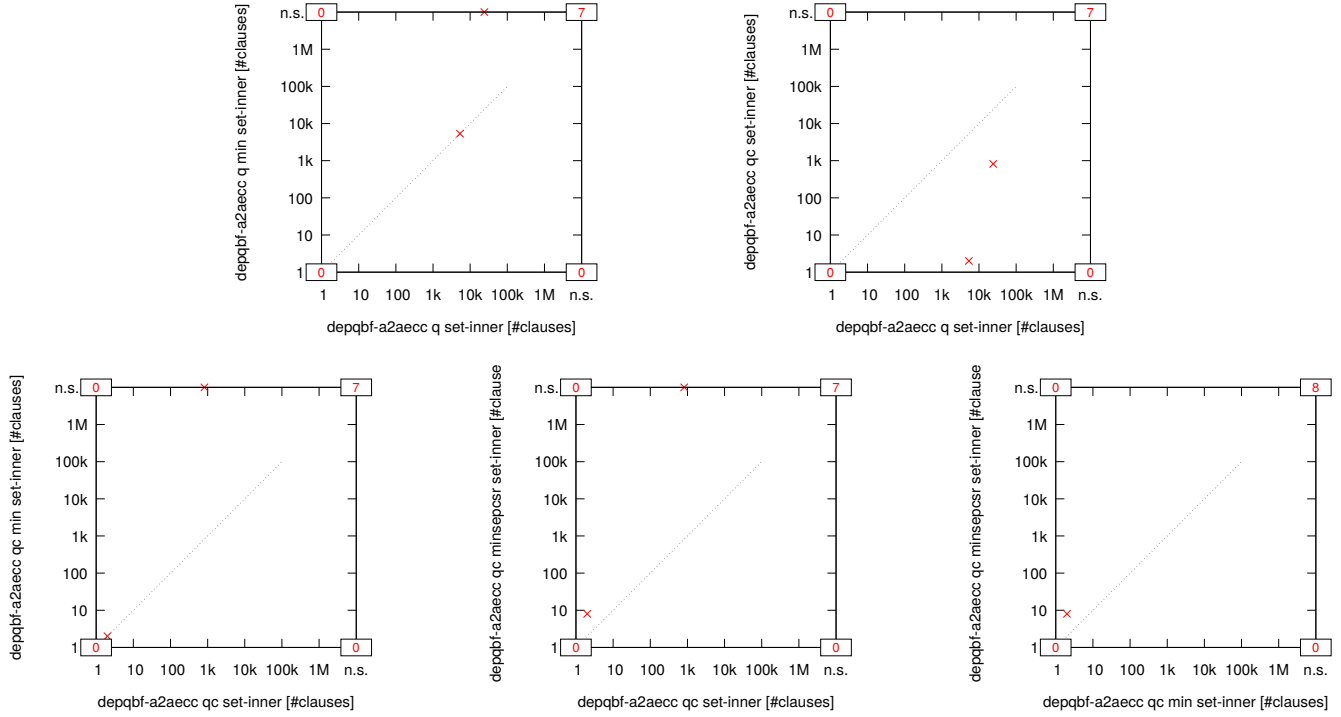


Fig. 247: Suite Palacios ($n = 9$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

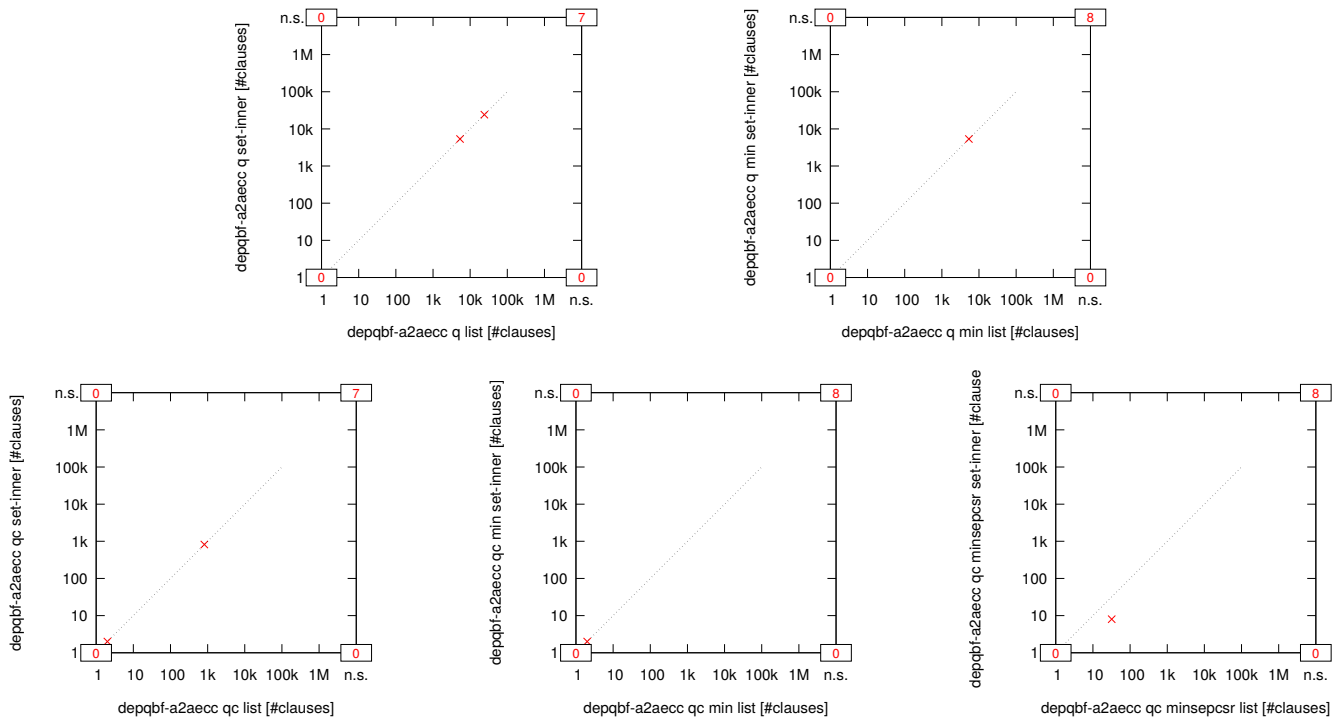


Fig. 248: Suite Palacios ($n = 9$): Comparing sizes of different unsatisfiable cores in `DepQBF-a2aecc` in list versus set-inner semantics (number of clauses).

35) *Pan* ($n = 89$):

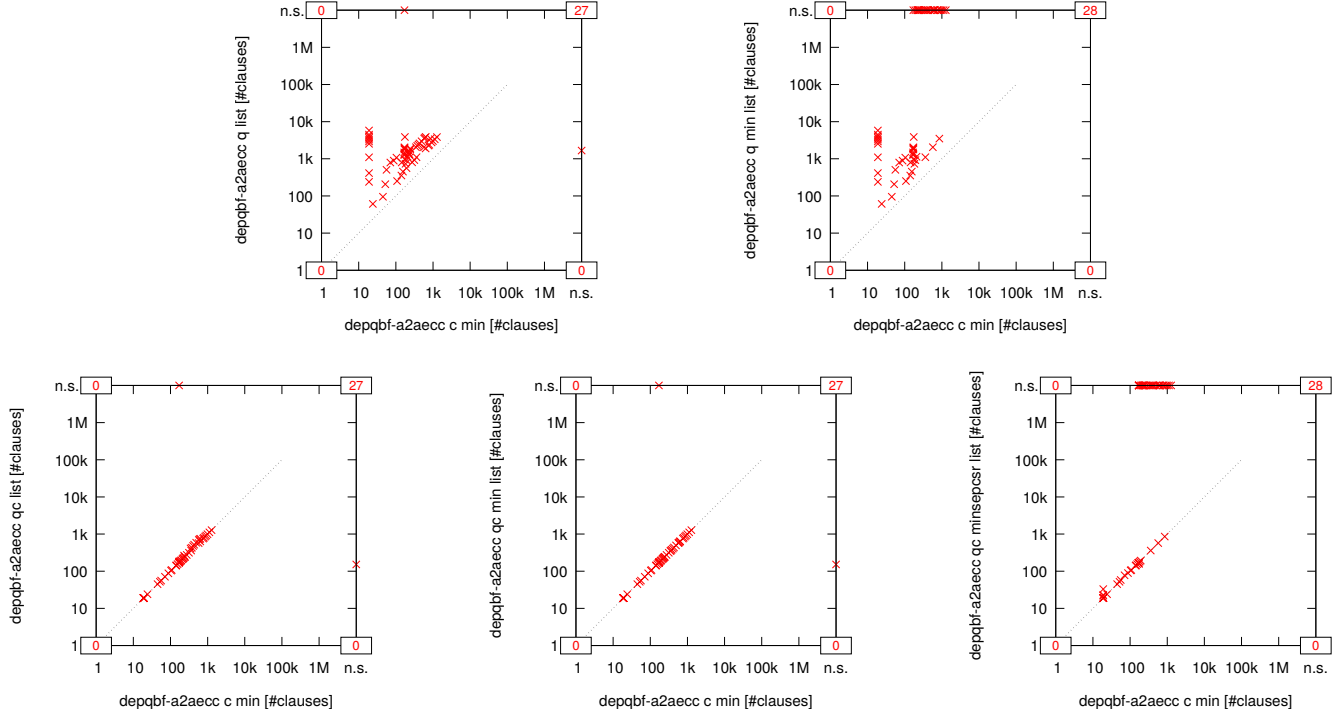


Fig. 249: Suite Pan ($n = 89$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

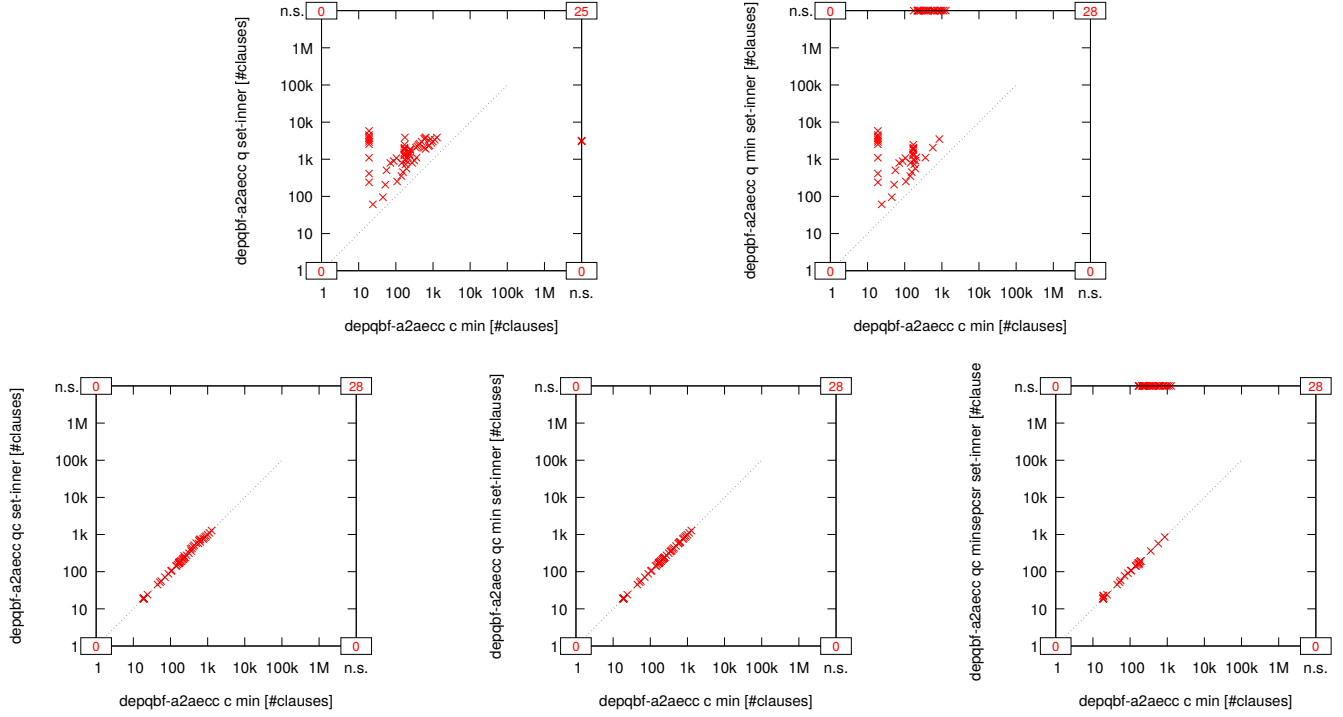


Fig. 250: Suite Pan ($n = 89$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

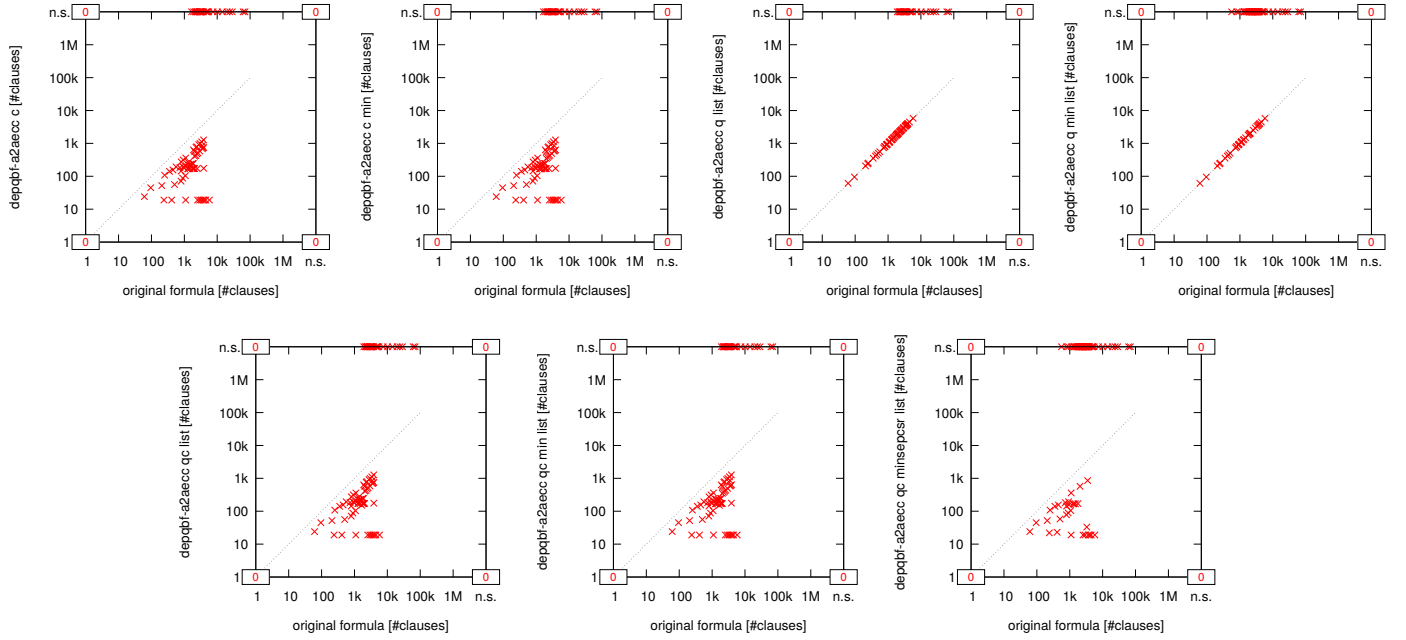


Fig. 251: Suite Pan ($n = 89$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

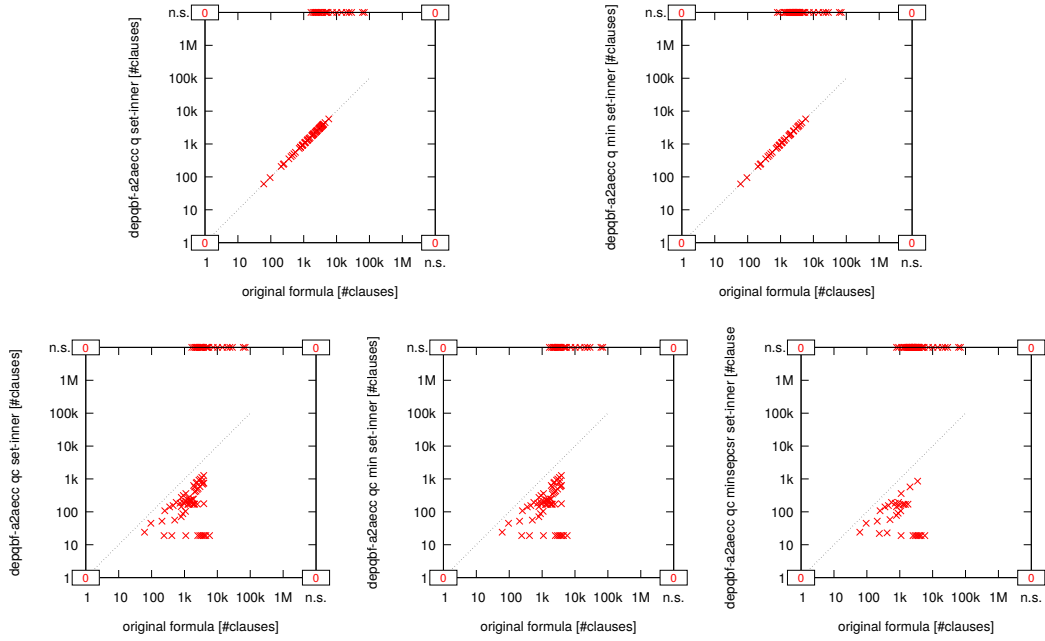


Fig. 252: Suite Pan ($n = 89$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

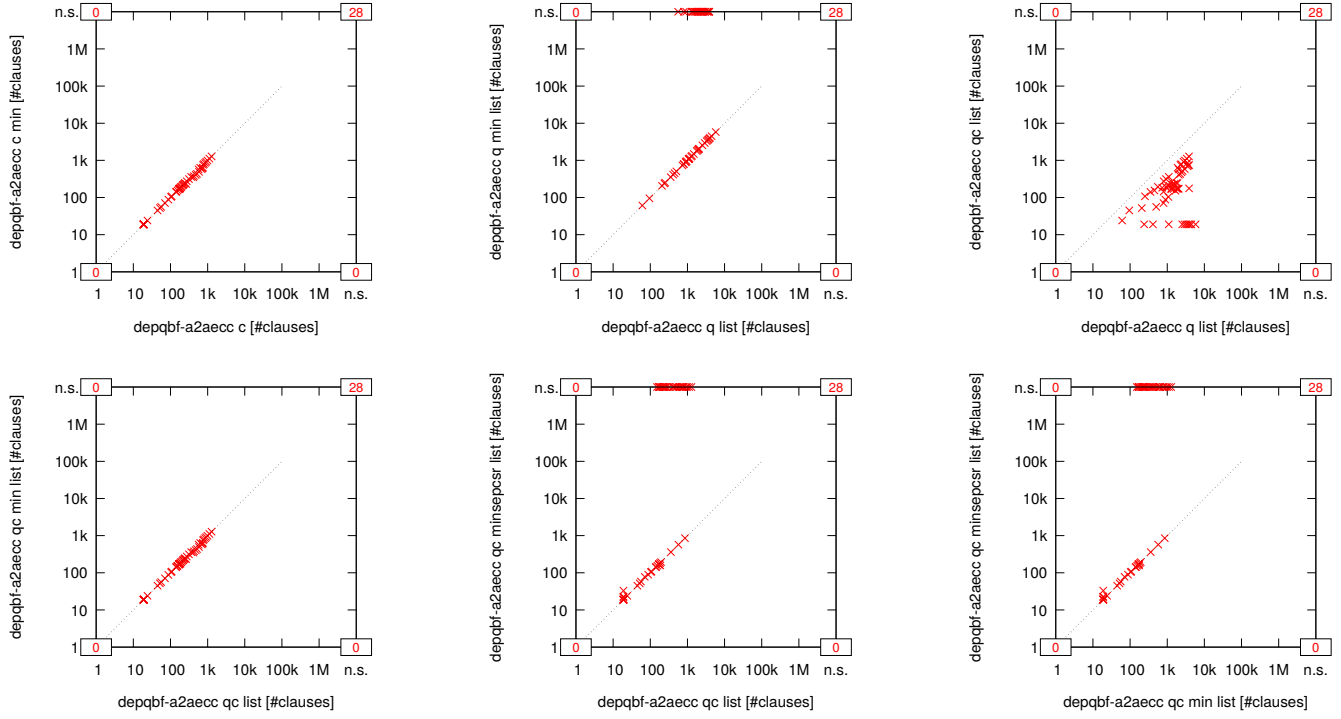


Fig. 253: Suite Pan ($n = 89$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

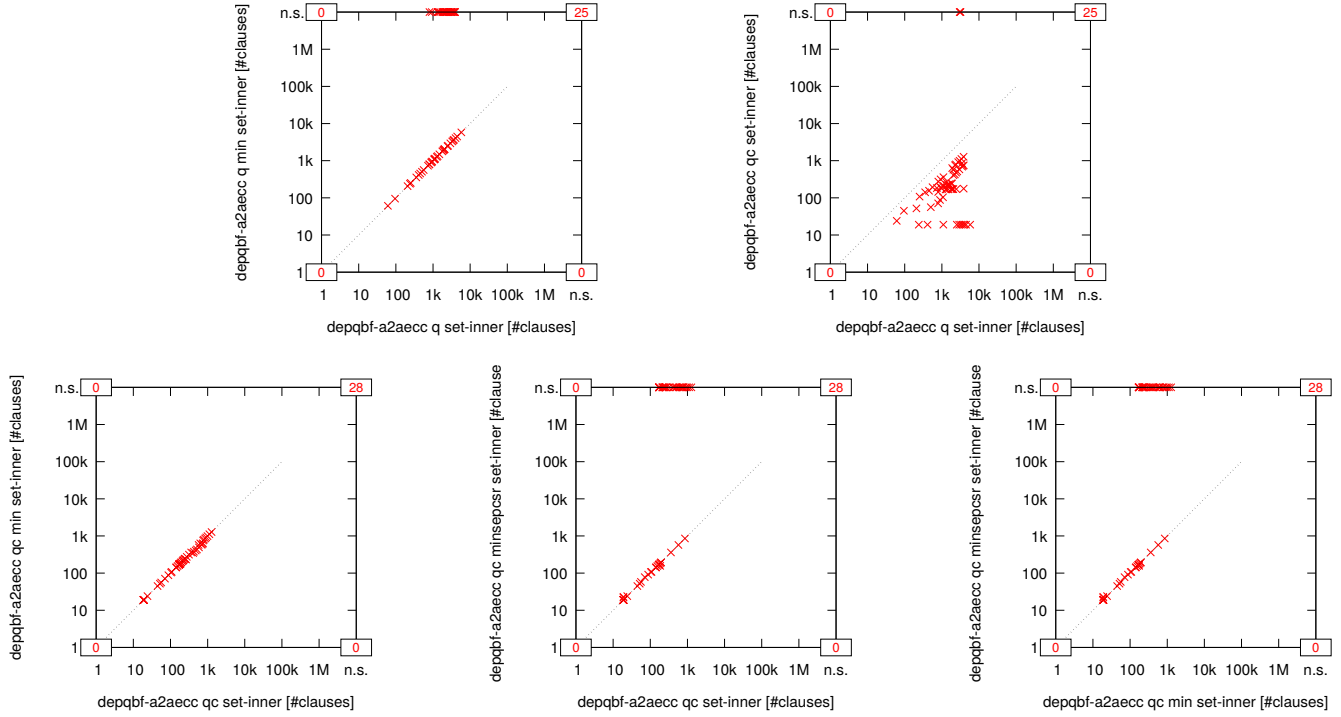


Fig. 254: Suite Pan ($n = 89$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

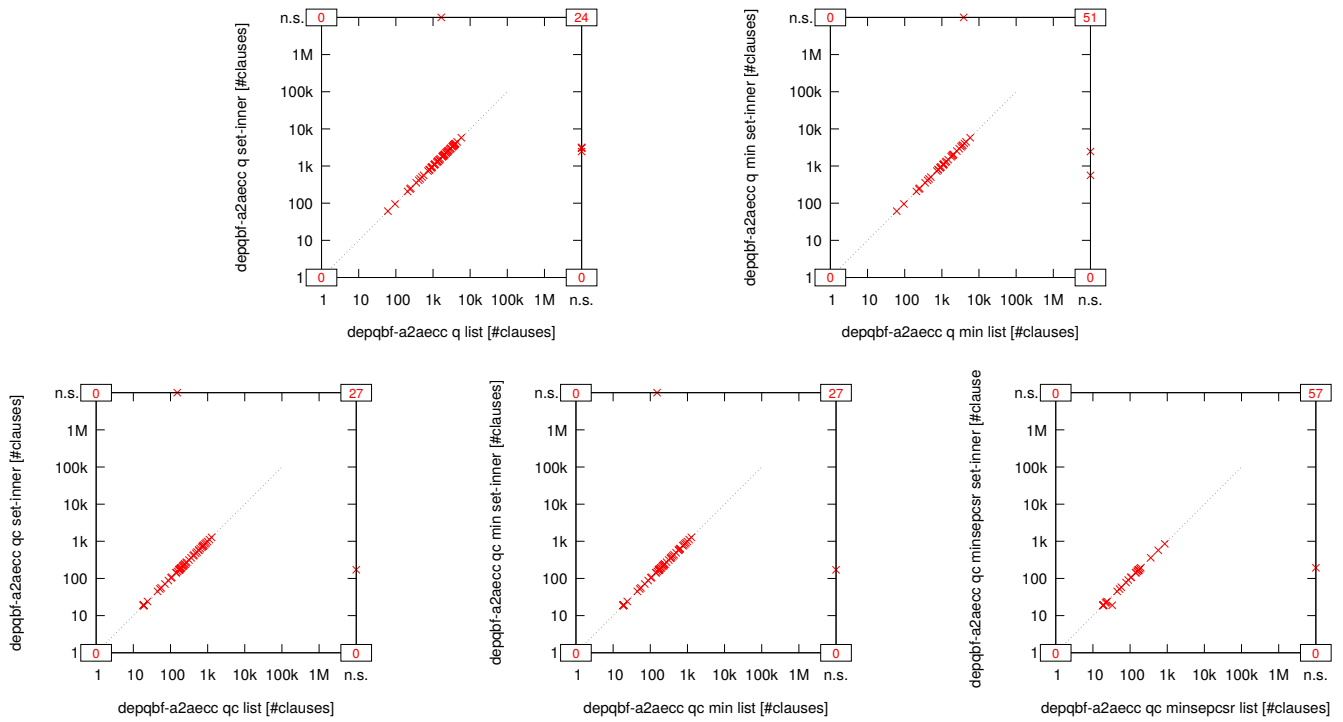


Fig. 255: Suite Pan ($n = 89$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

36) *Peitl* ($n = 10$):

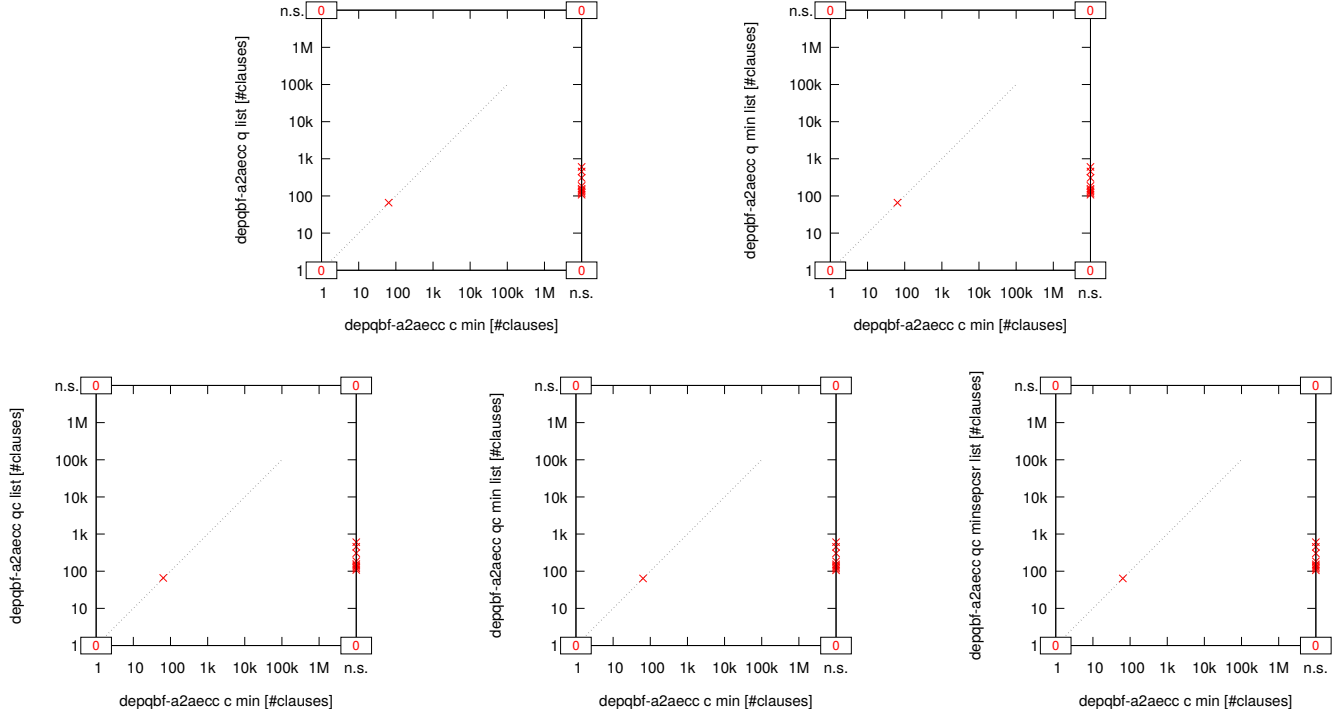


Fig. 256: Suite Peitl ($n = 10$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

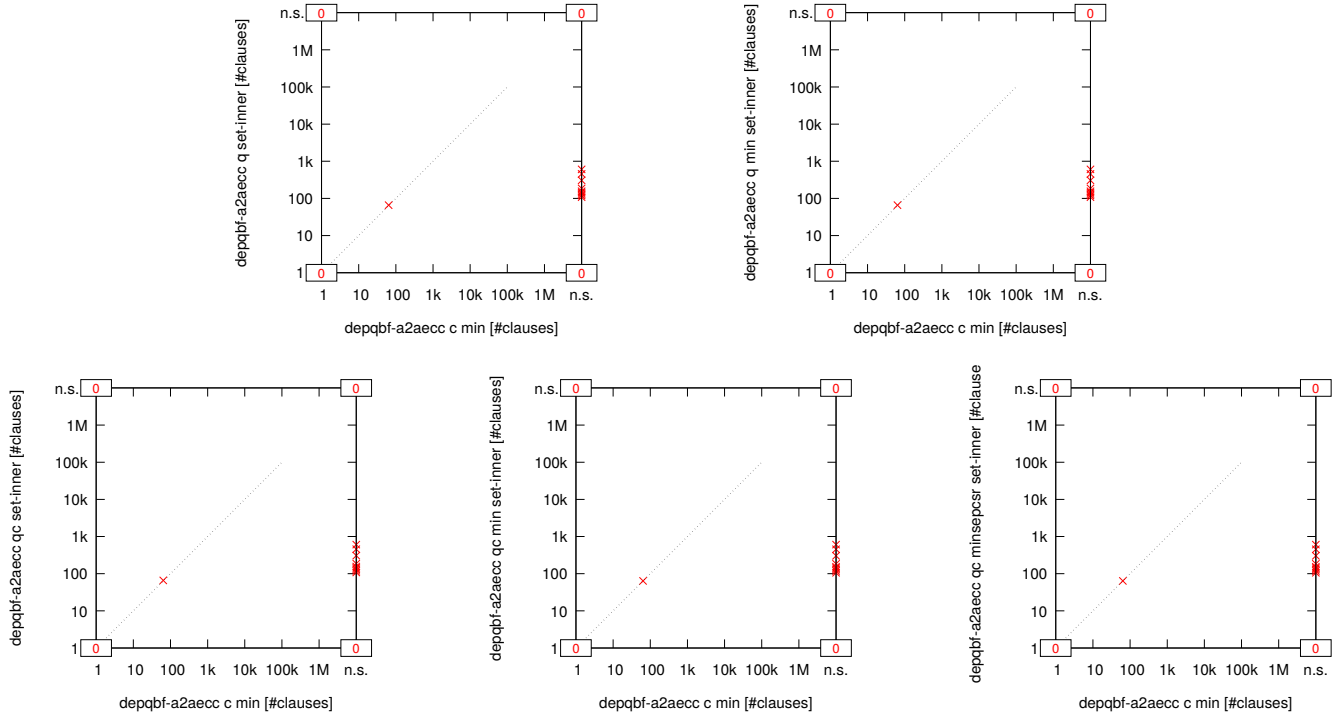


Fig. 257: Suite Peitl ($n = 10$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

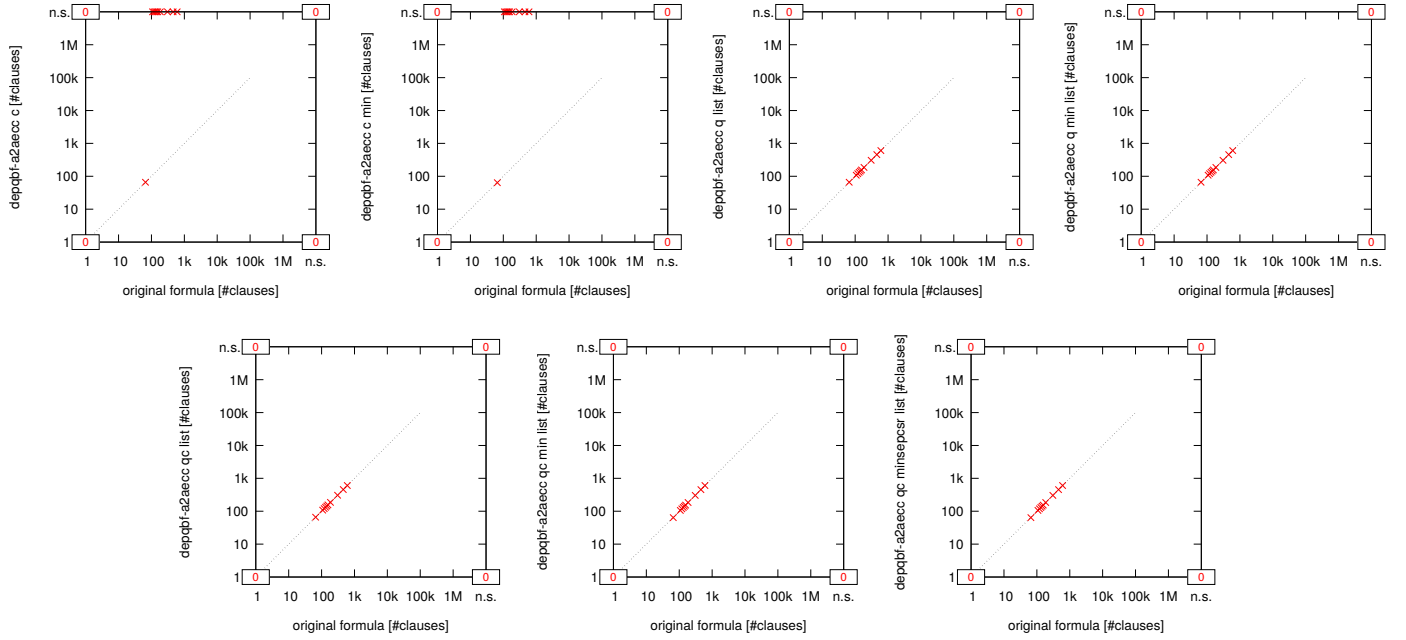


Fig. 258: Suite Peitl ($n = 10$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

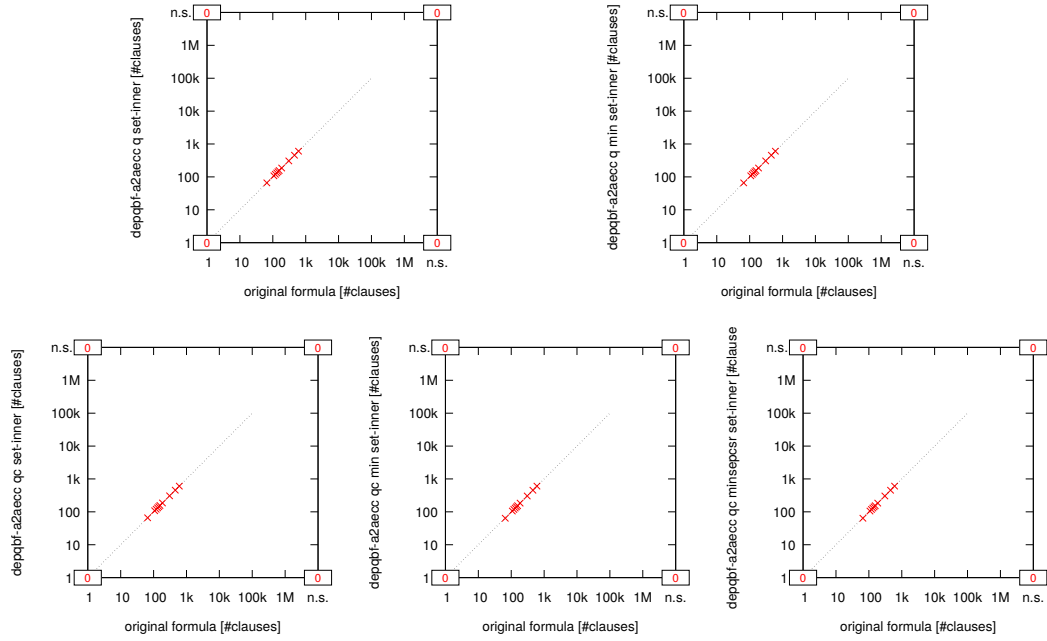


Fig. 259: Suite Peitl ($n = 10$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

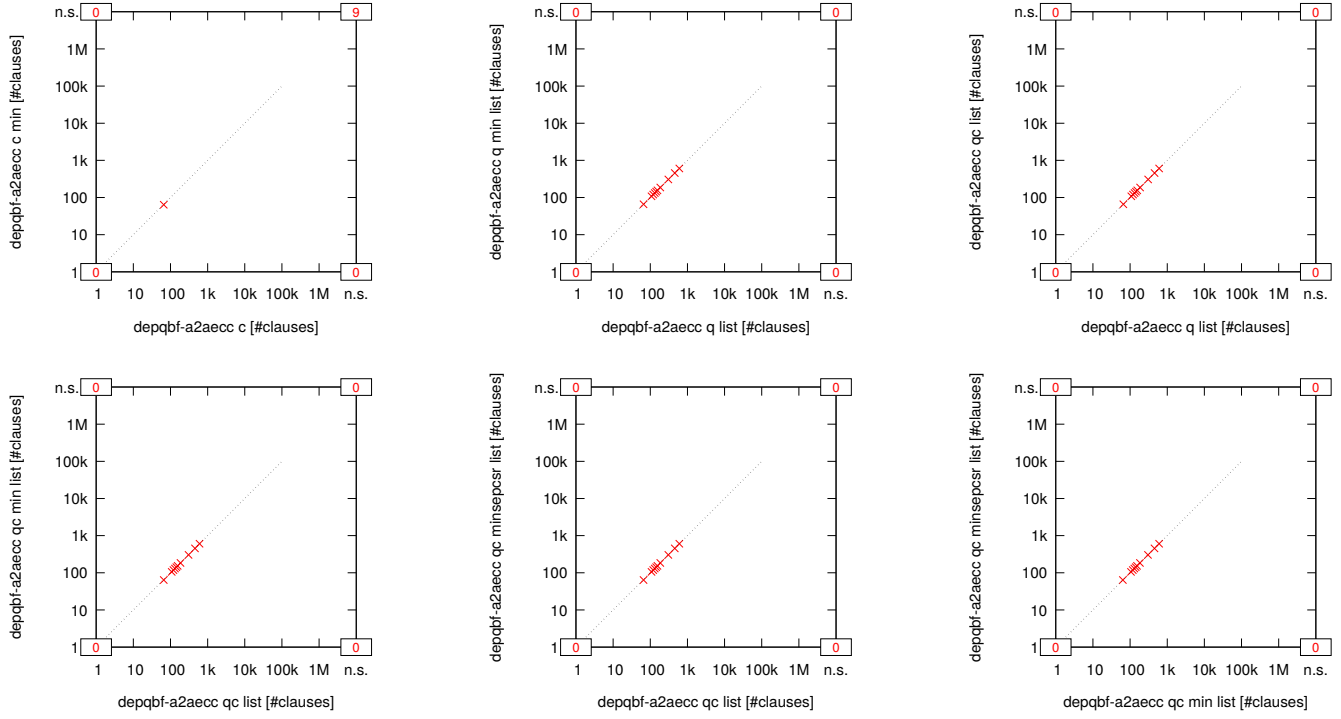


Fig. 260: Suite Peitl ($n = 10$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

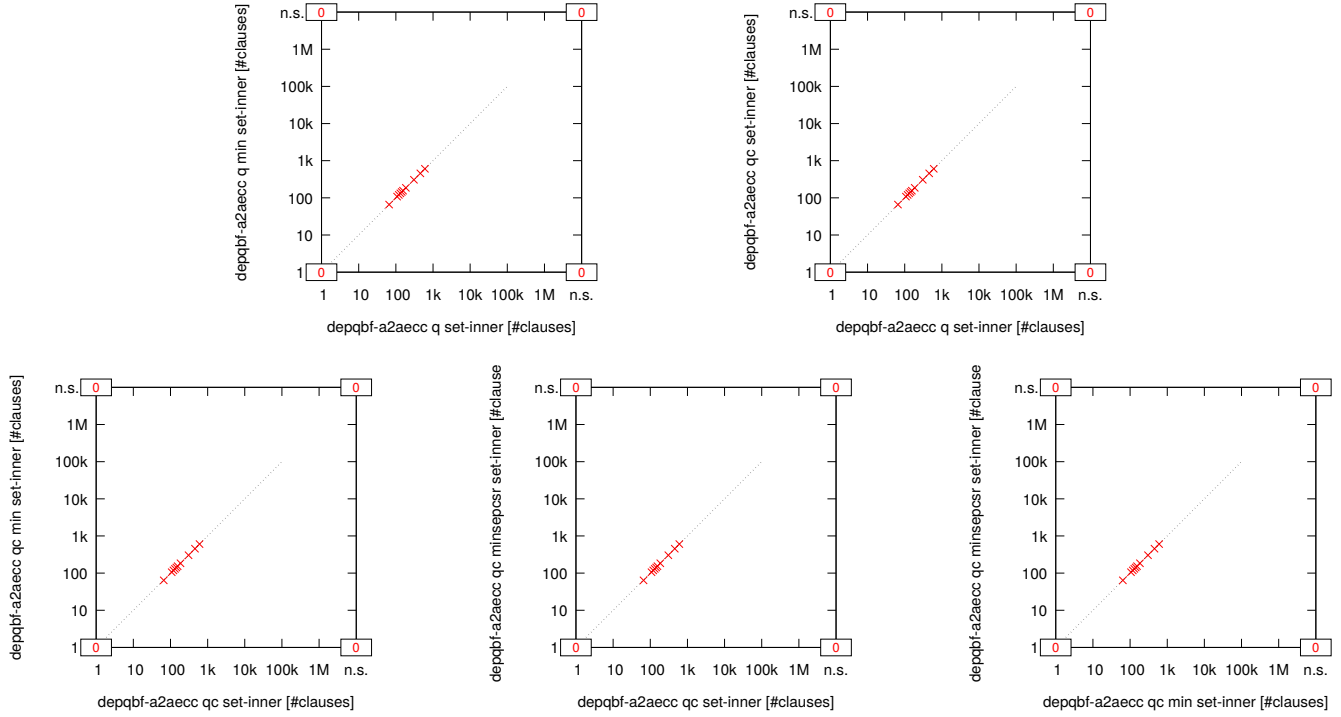


Fig. 261: Suite Peitl ($n = 10$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

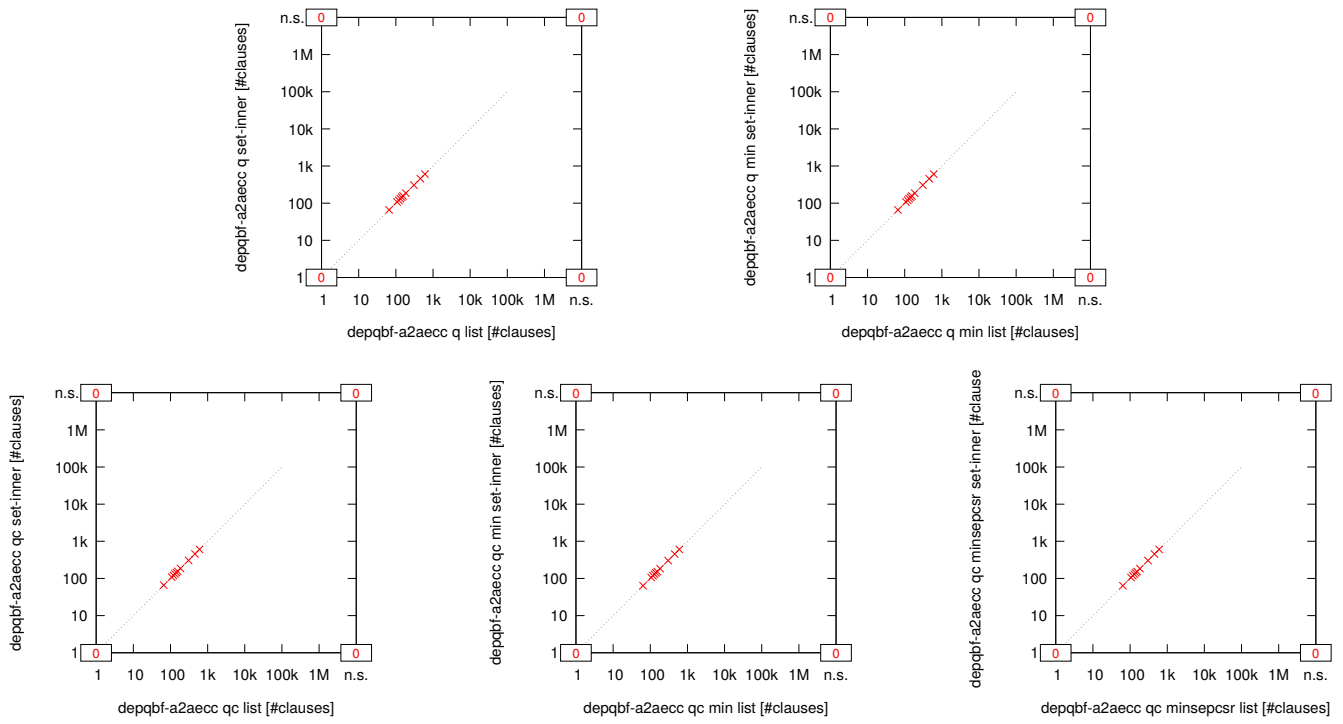


Fig. 262: Suite Peitl ($n = 10$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

37) Preusser ($n = 0$):

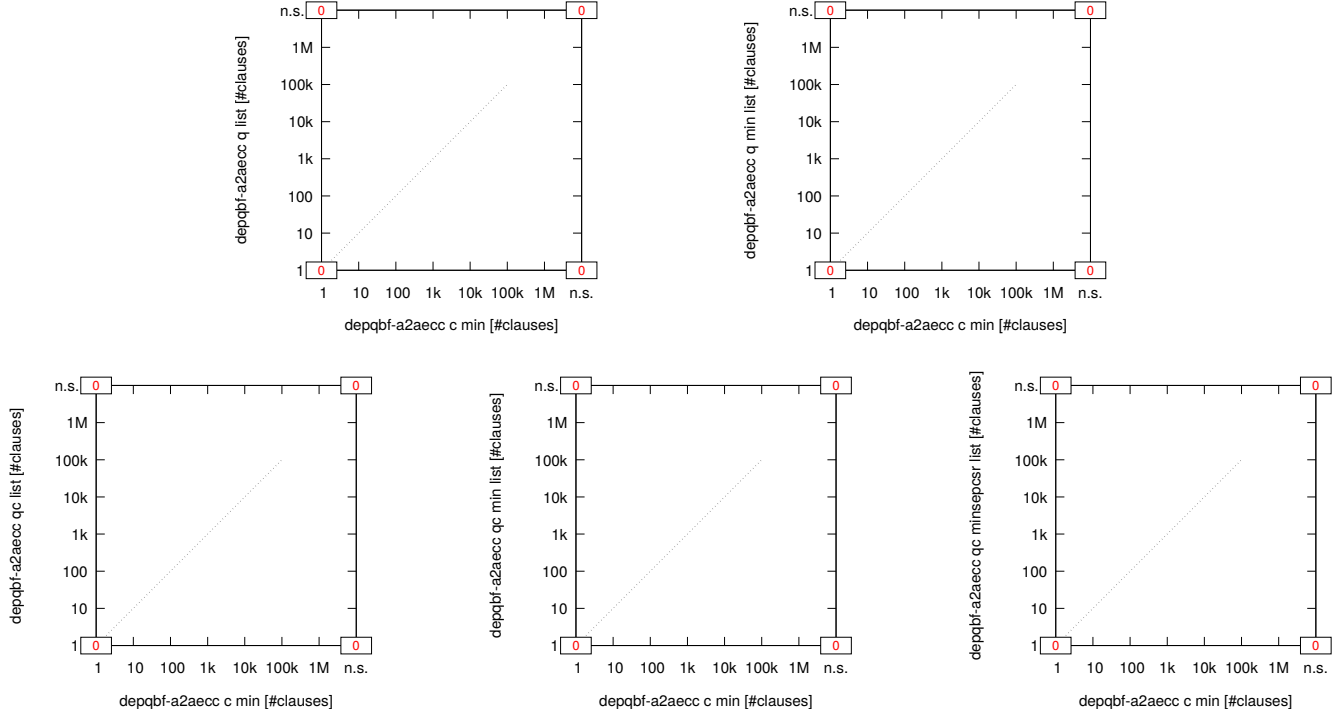


Fig. 263: Suite Preusser ($n = 0$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

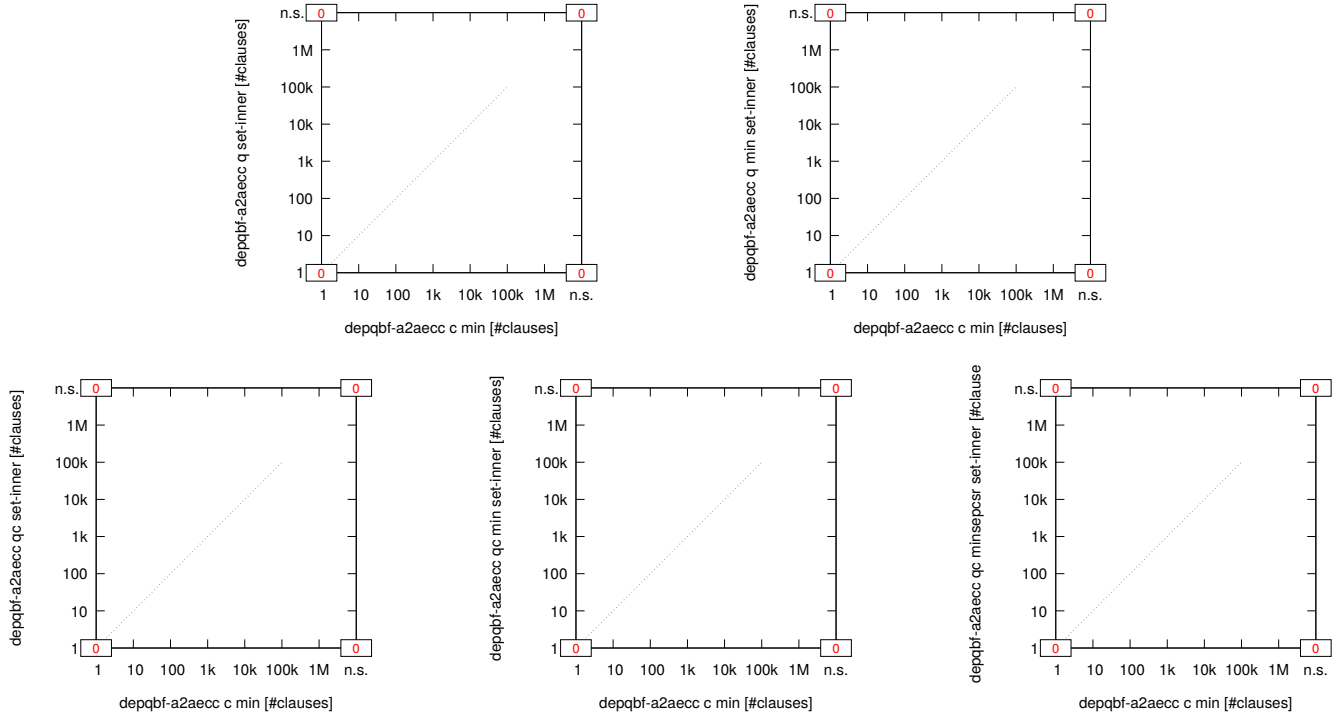


Fig. 264: Suite Preusser ($n = 0$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

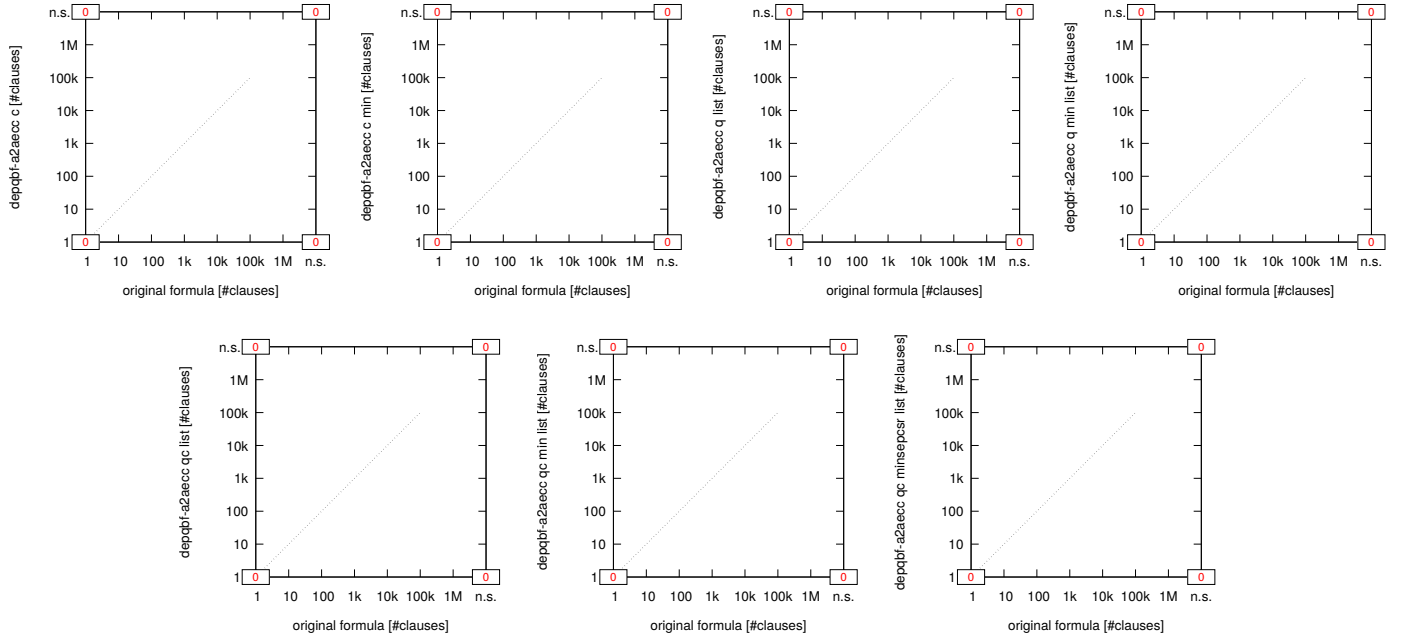


Fig. 265: Suite Preusser ($n = 0$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

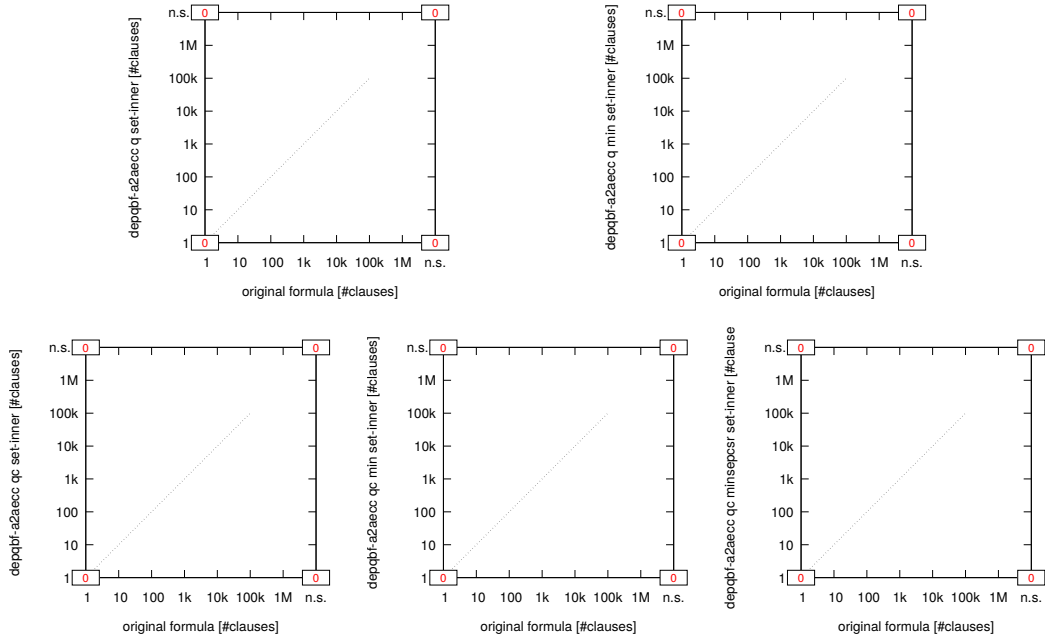


Fig. 266: Suite Preusser ($n = 0$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

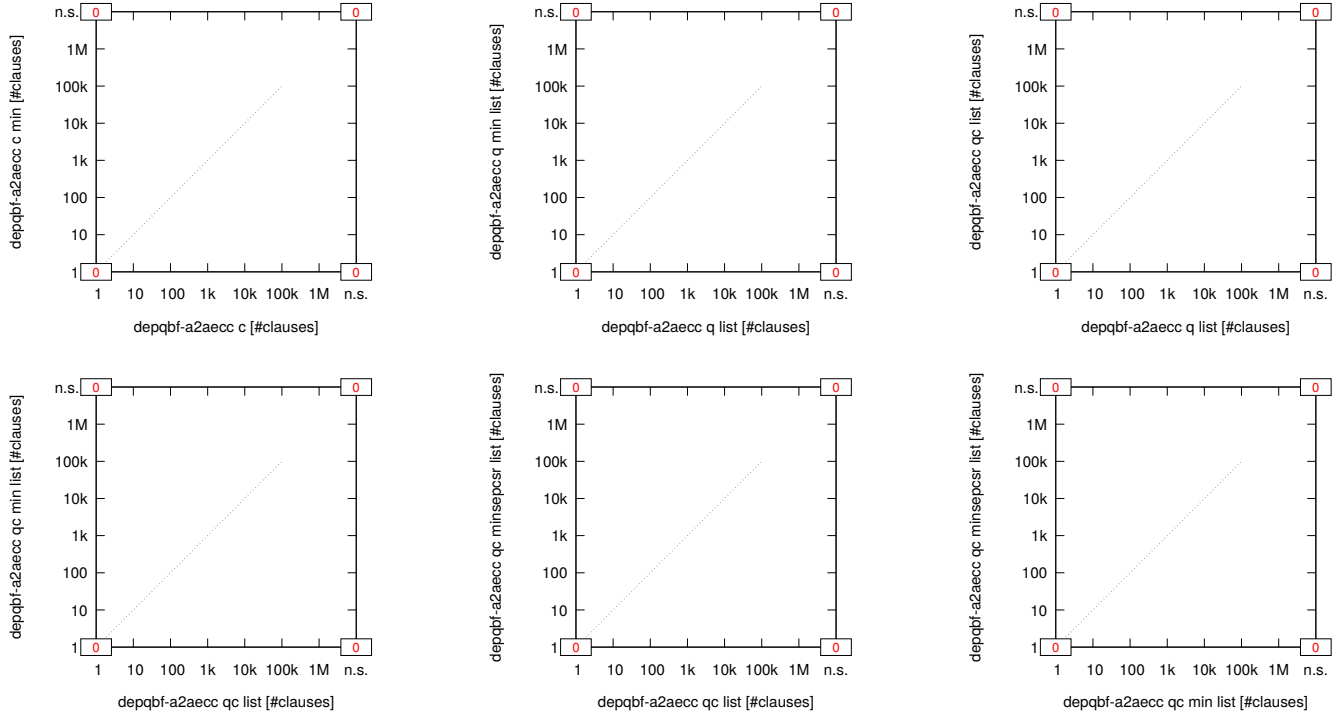


Fig. 267: Suite Preusser ($n = 0$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

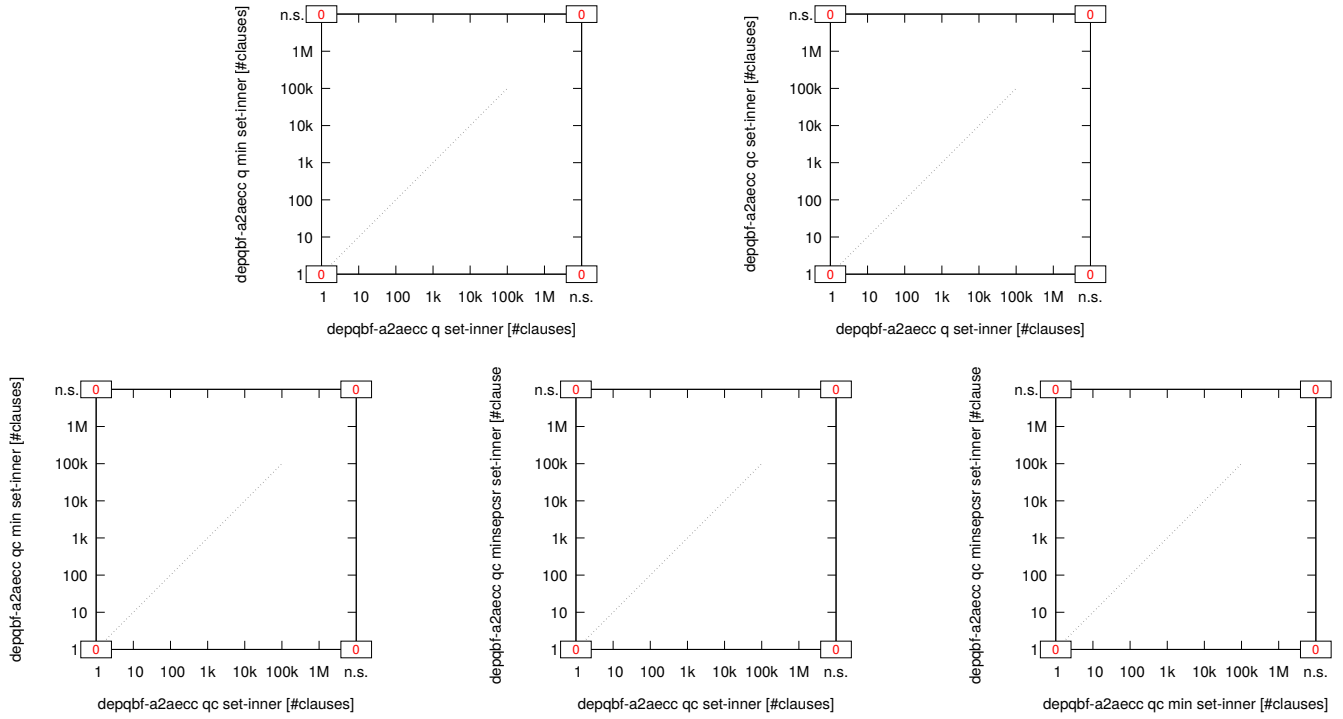


Fig. 268: Suite Preusser ($n = 0$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

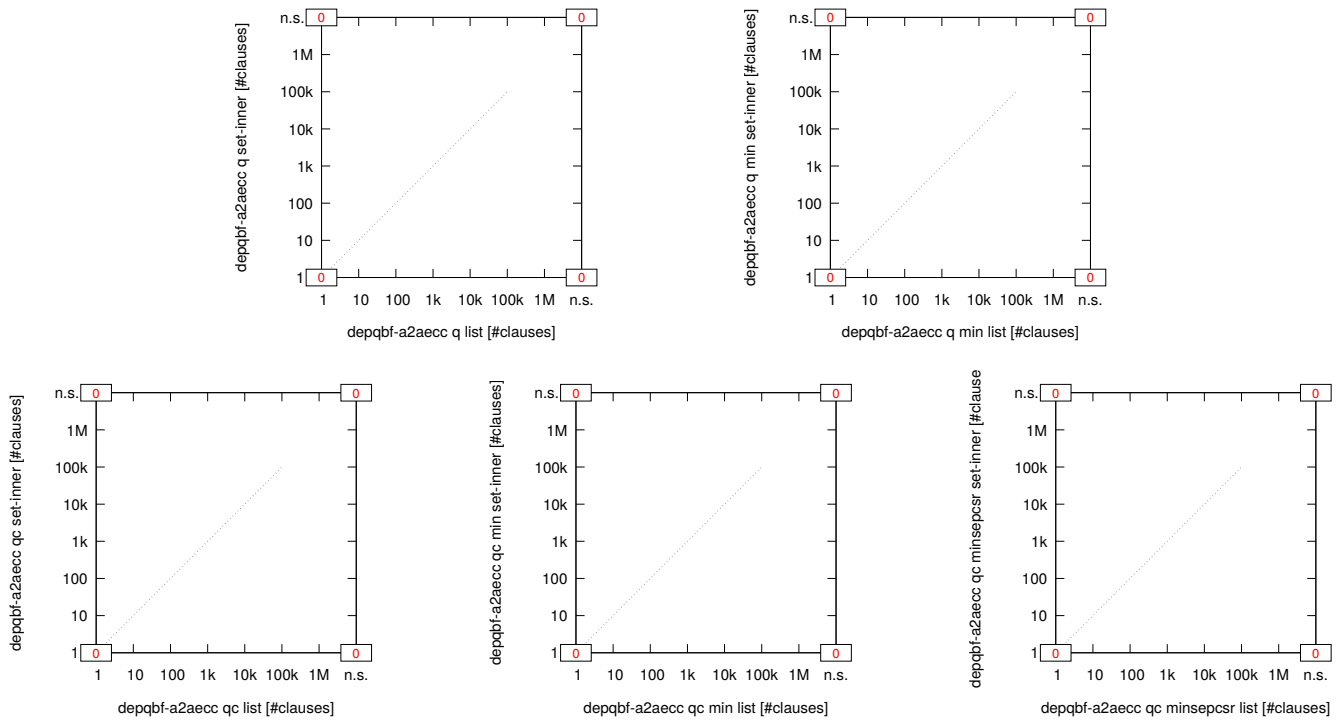


Fig. 269: Suite Preusser ($n = 0$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

38) *qbfeval12* ($n = 8$):

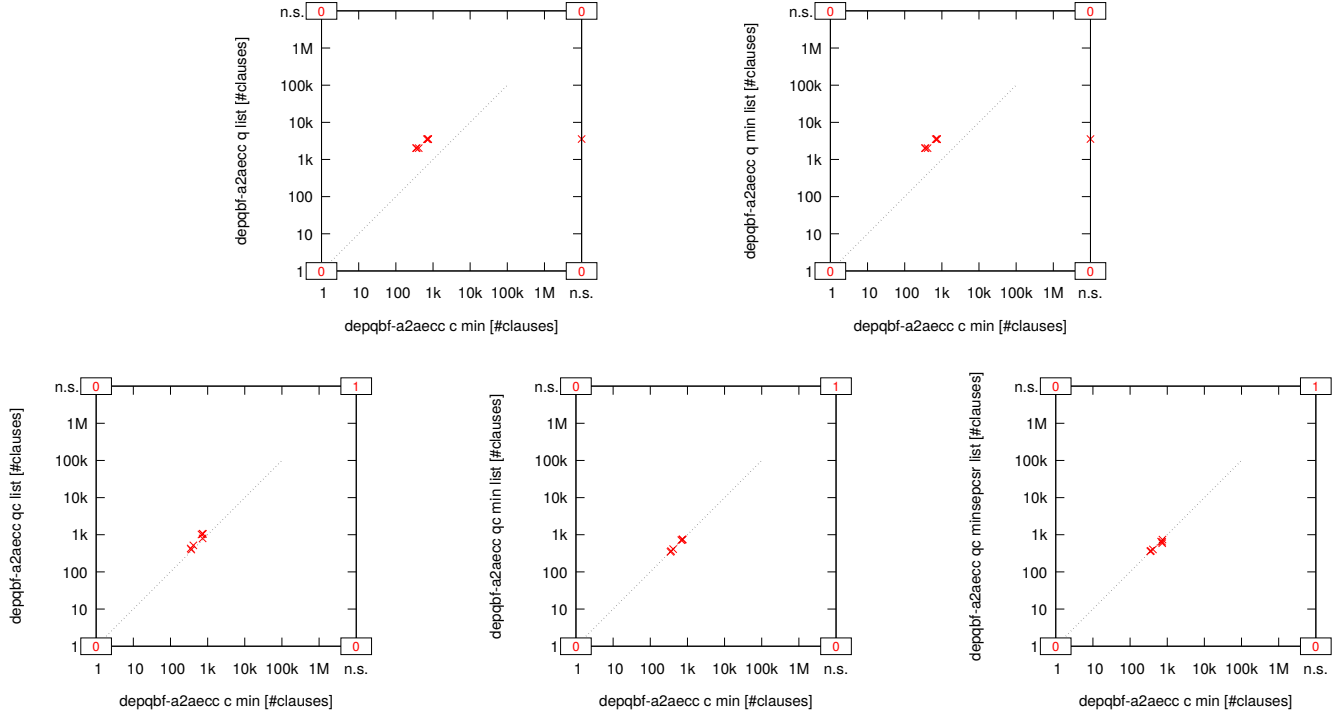


Fig. 270: Suite *qbfeval12* ($n = 8$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

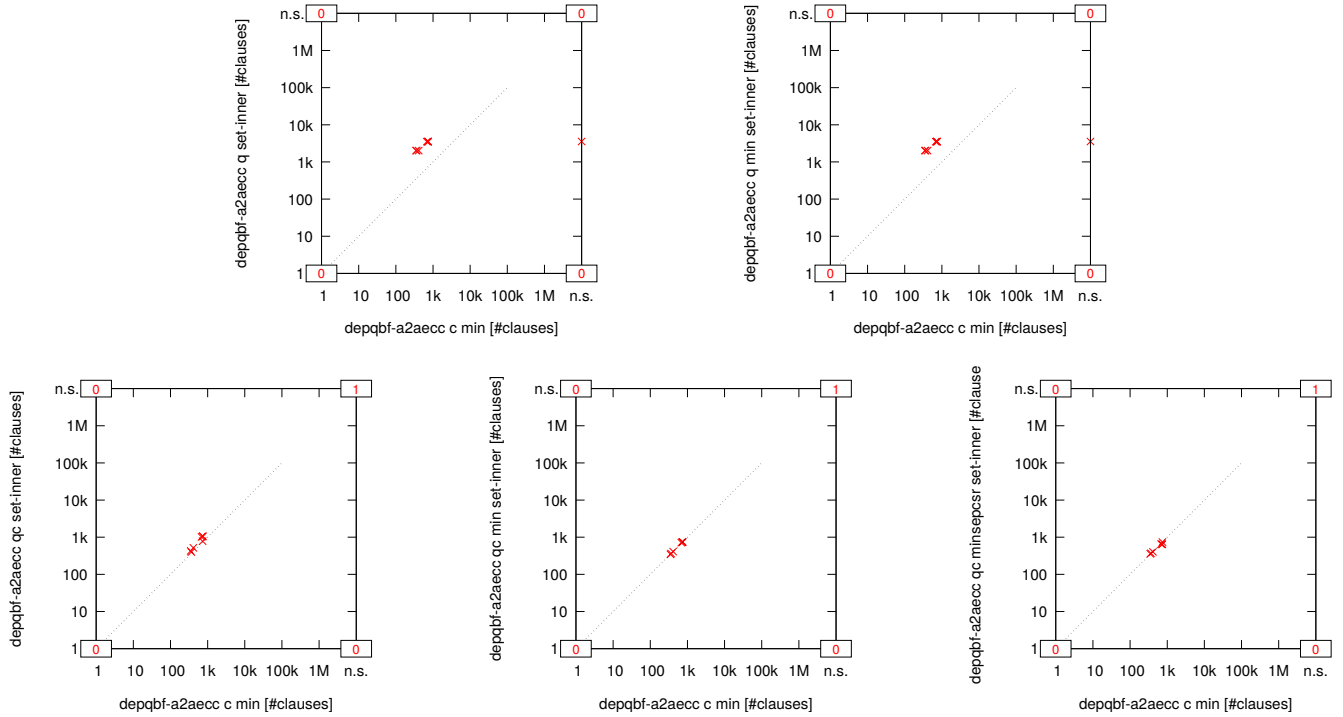


Fig. 271: Suite *qbfeval12* ($n = 8$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

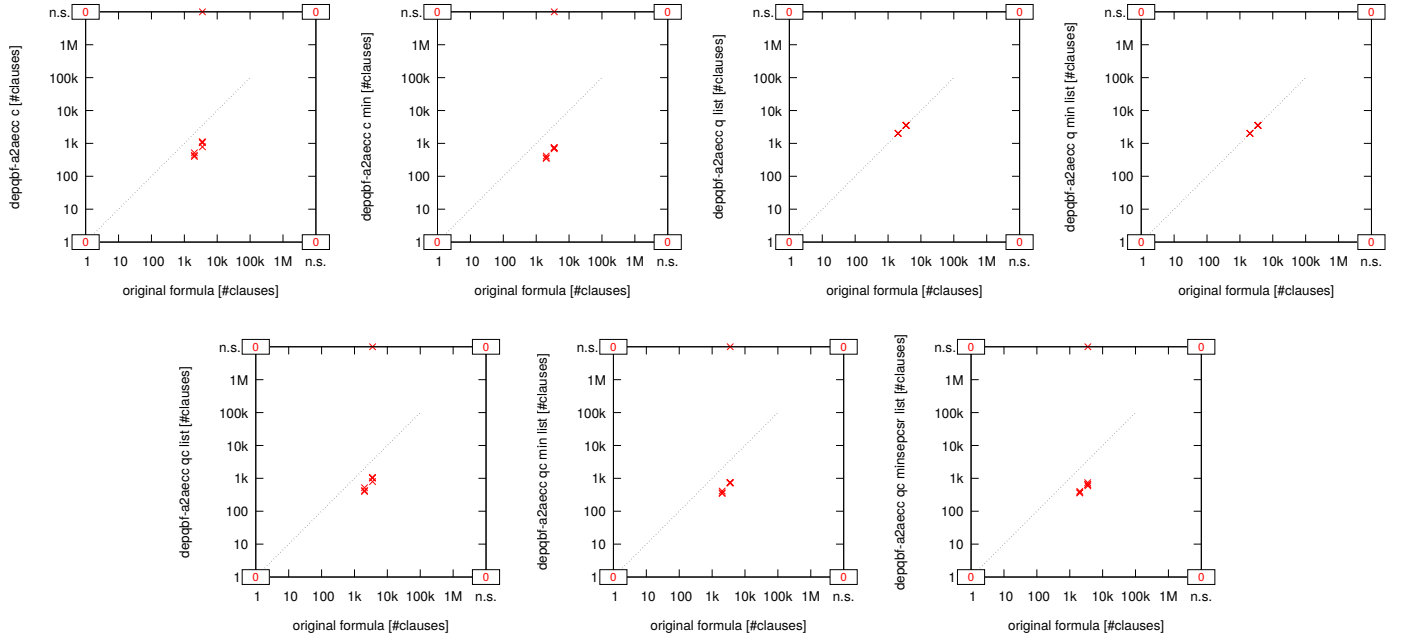


Fig. 272: Suite qbfeval12 ($n = 8$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

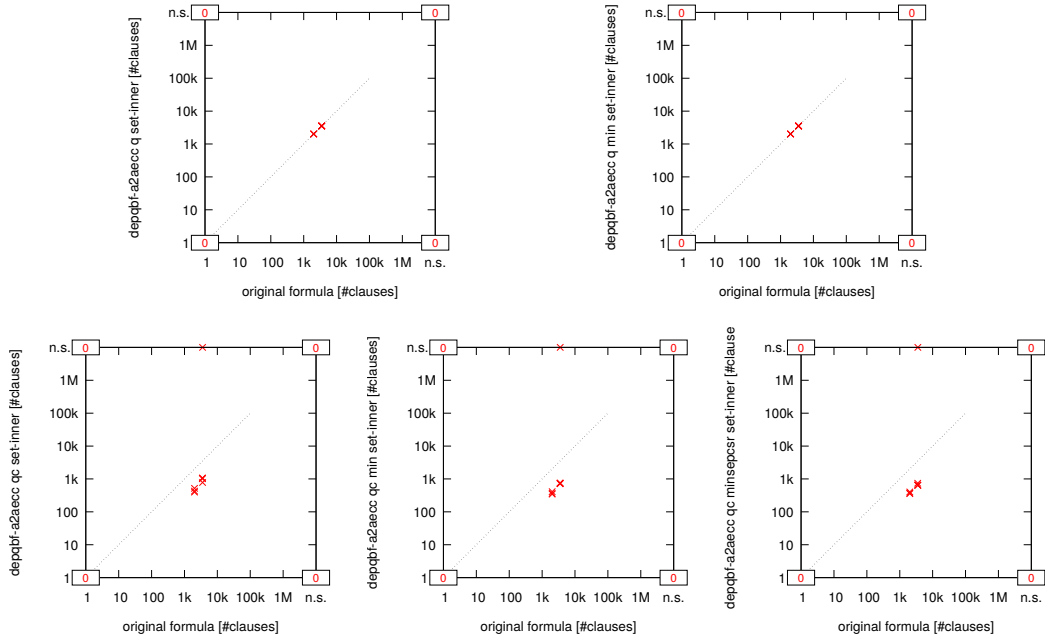


Fig. 273: Suite qbfeval12 ($n = 8$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

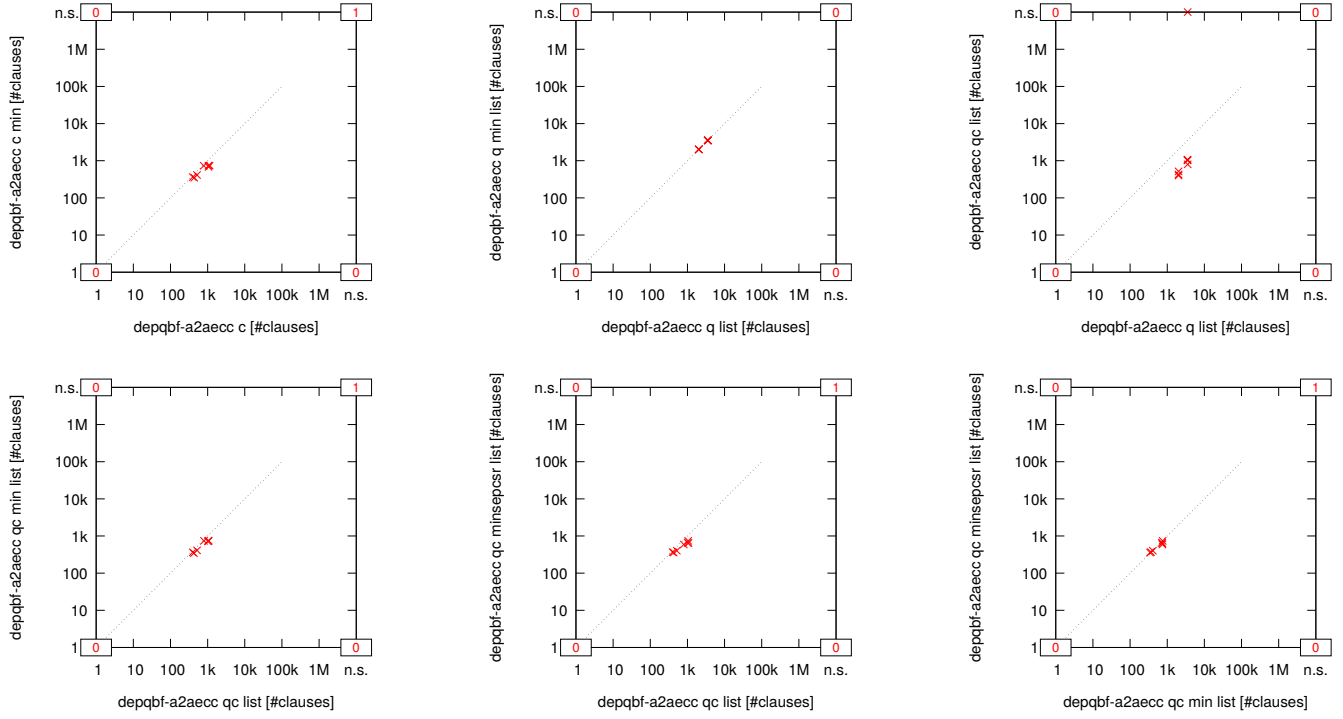


Fig. 274: Suite qbfeval12 ($n = 8$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

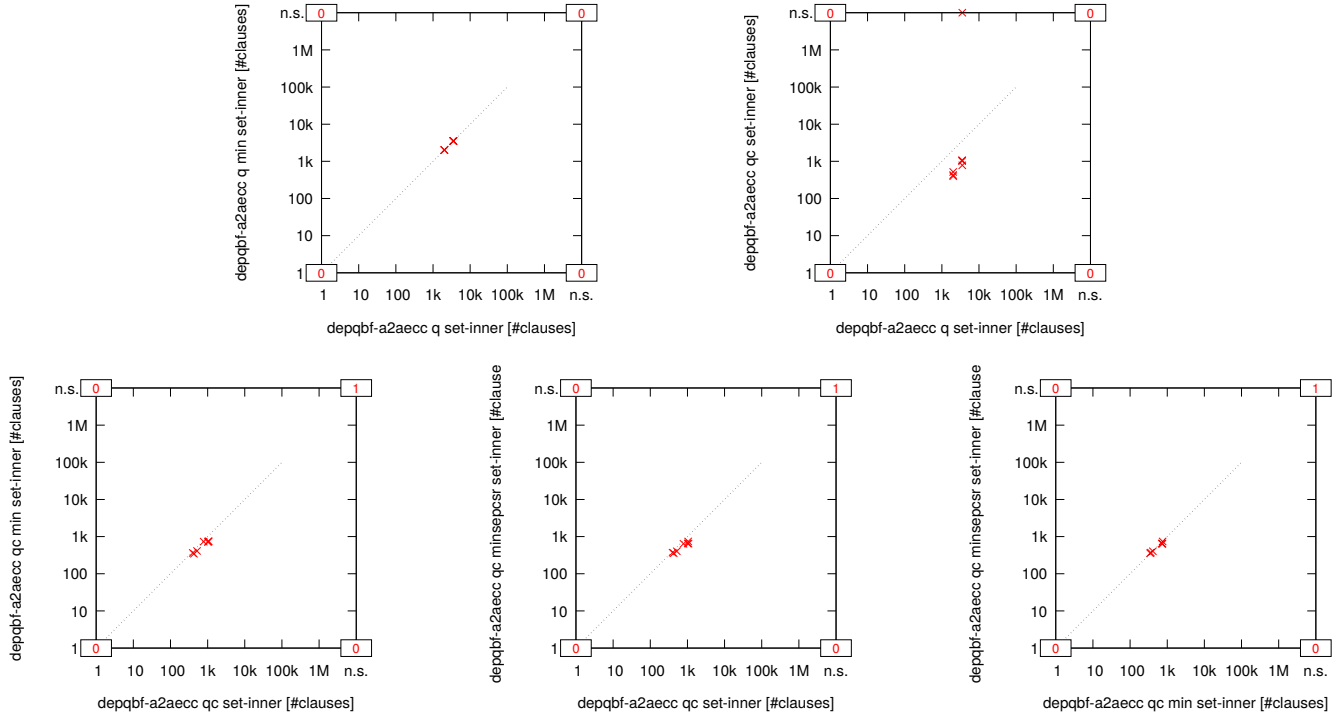


Fig. 275: Suite qbfeval12 ($n = 8$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

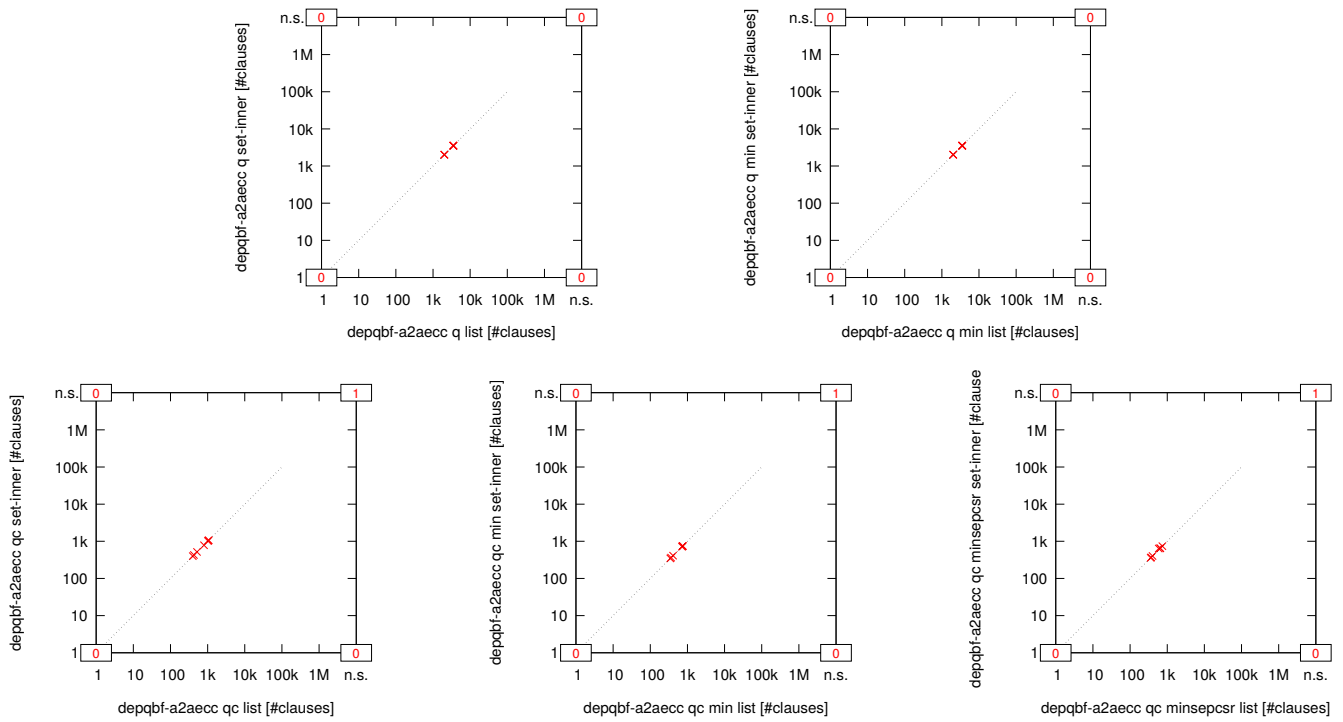


Fig. 276: Suite qbfeval12 ($n = 8$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

39) Rabe ($n = 3$):

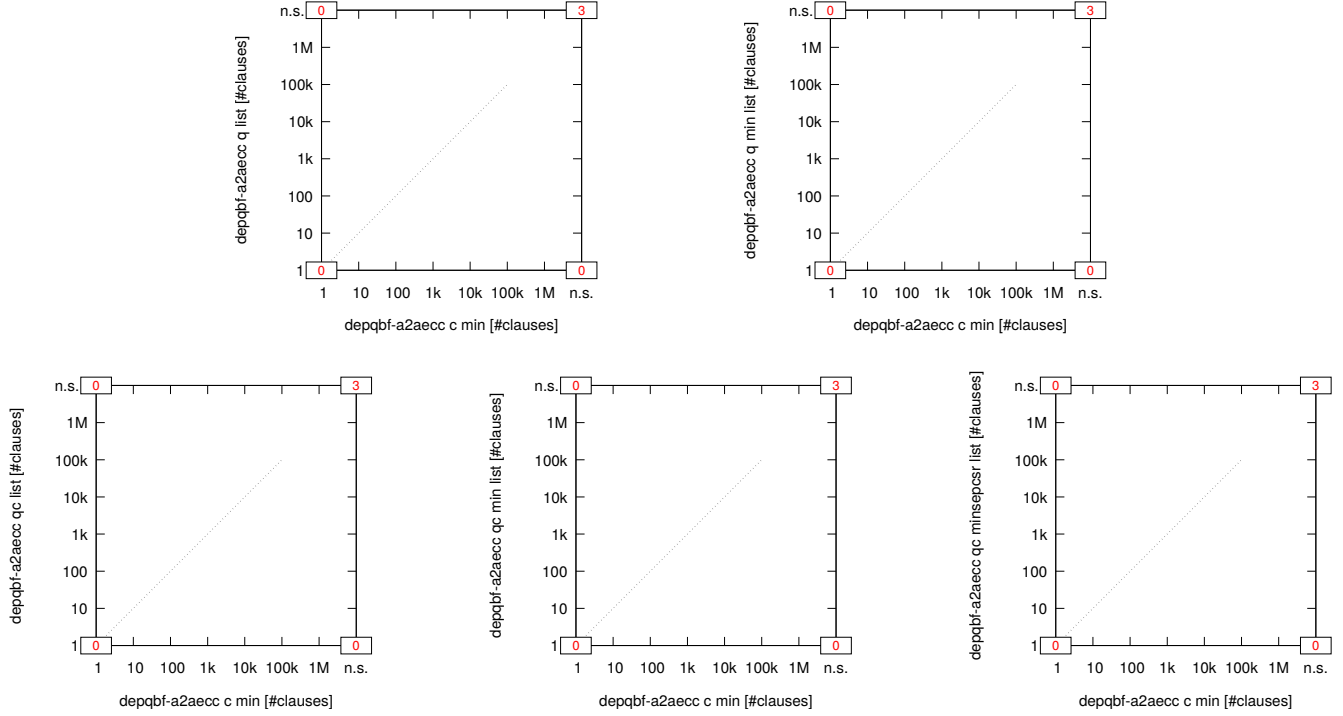


Fig. 277: Suite Rabe ($n = 3$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

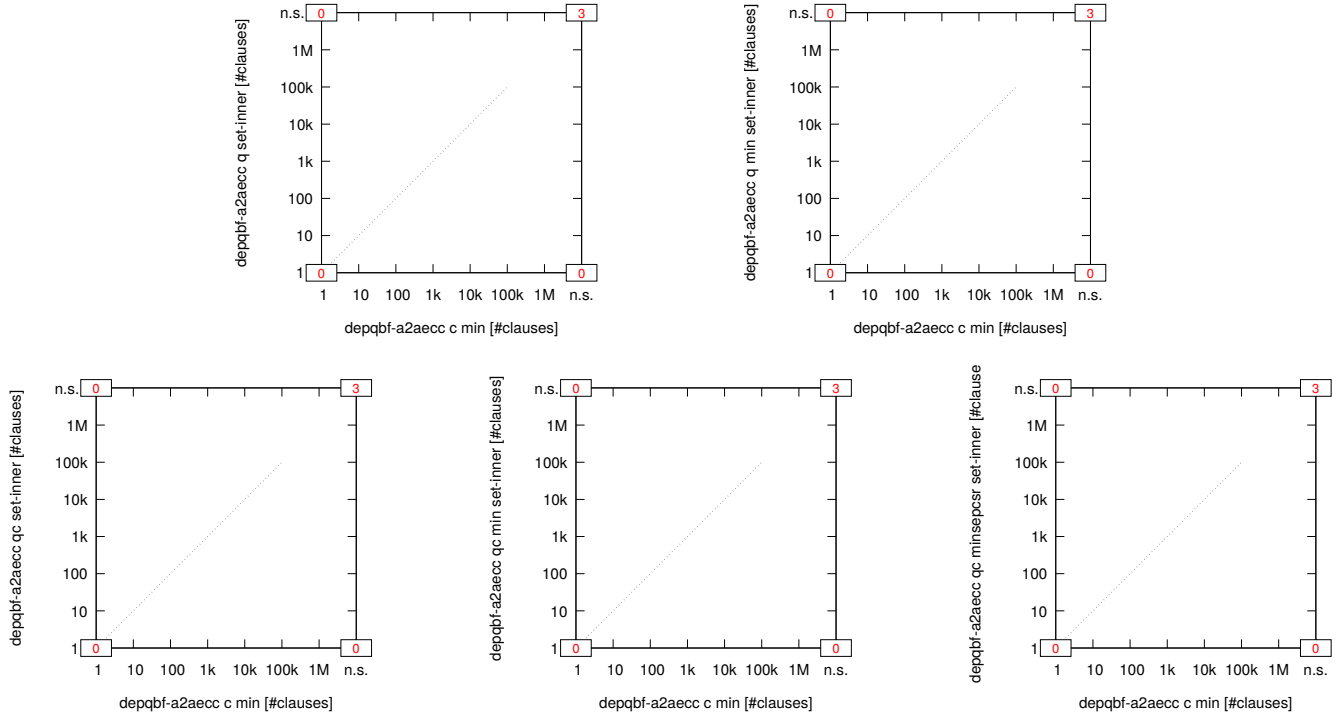


Fig. 278: Suite Rabe ($n = 3$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

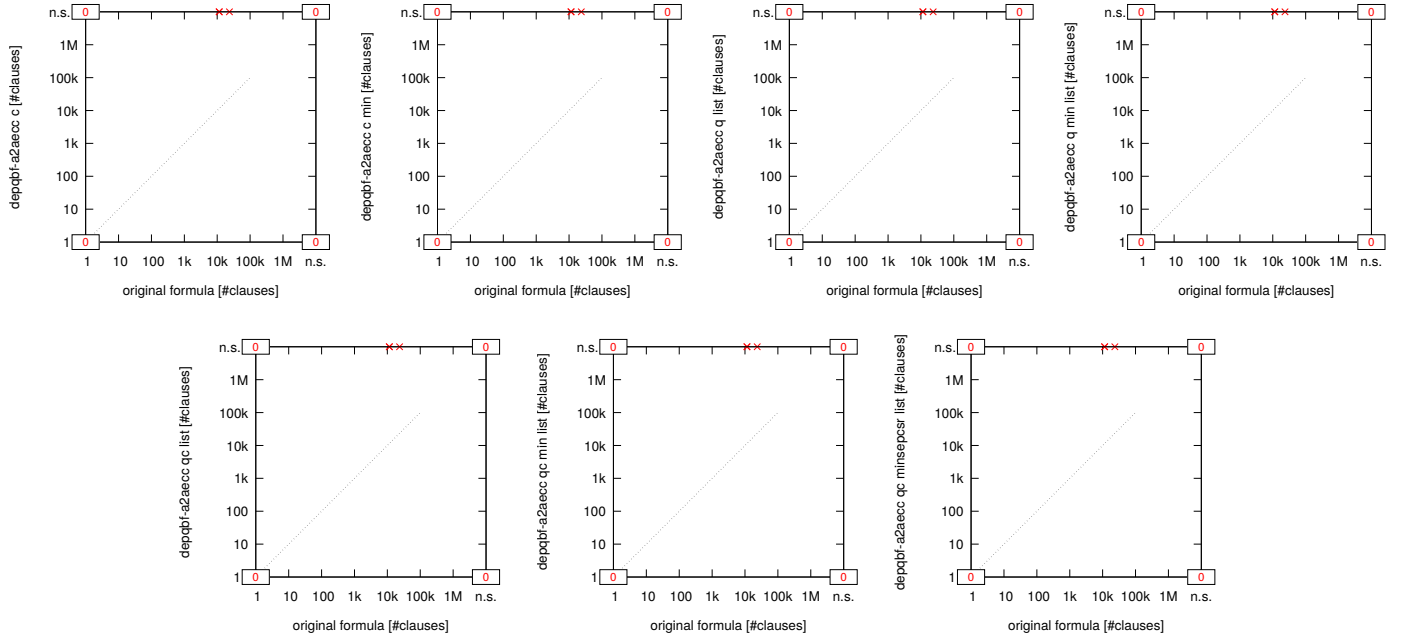


Fig. 279: Suite Rabe ($n = 3$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

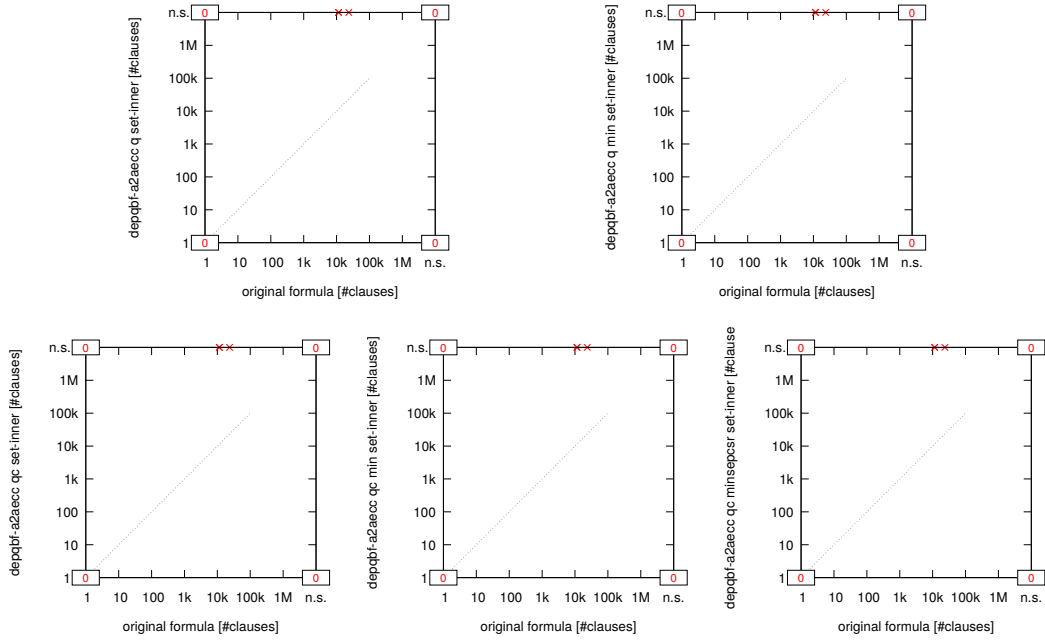


Fig. 280: Suite Rabe ($n = 3$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

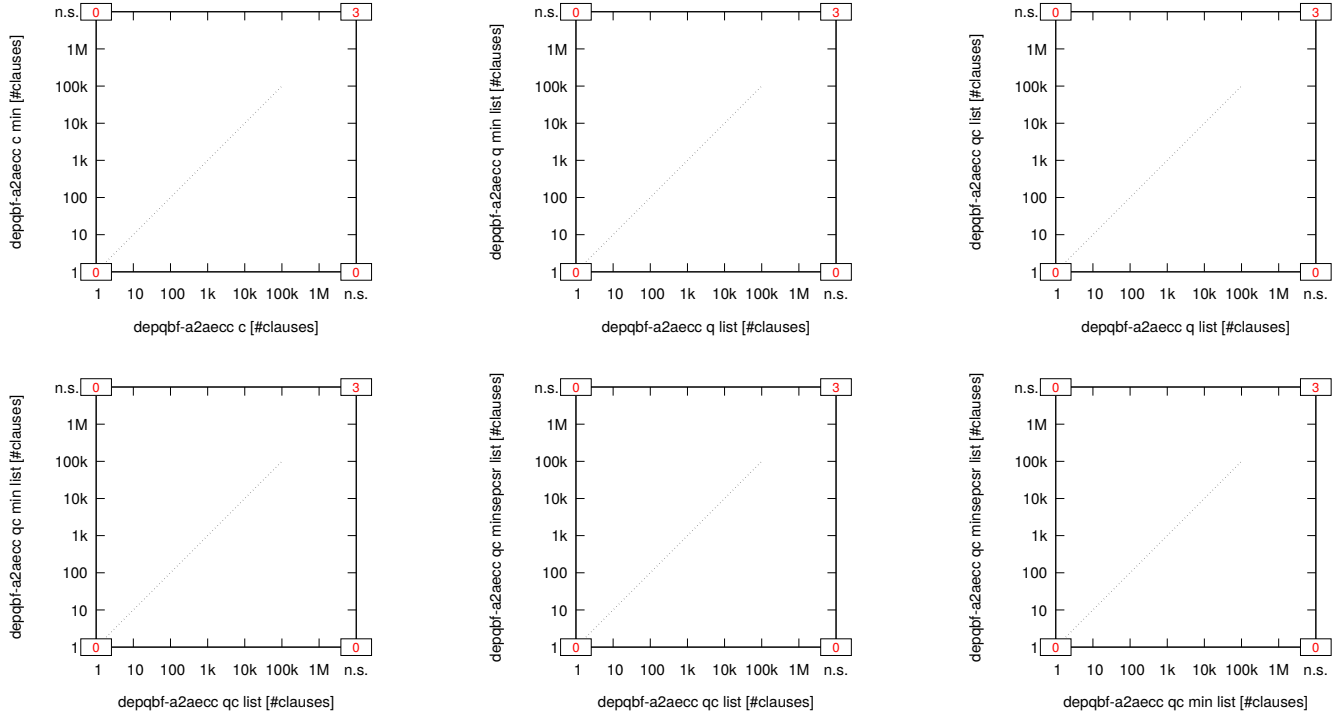


Fig. 281: Suite Rabe ($n = 3$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

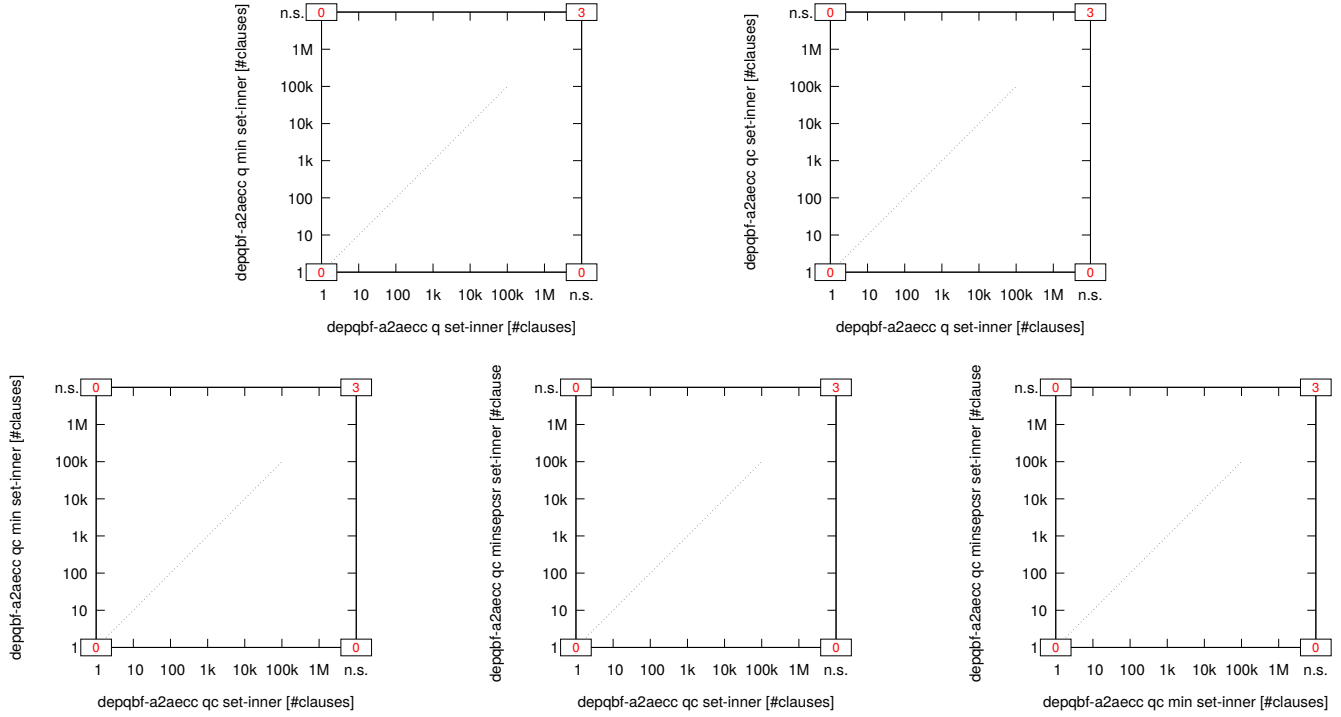


Fig. 282: Suite Rabe ($n = 3$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

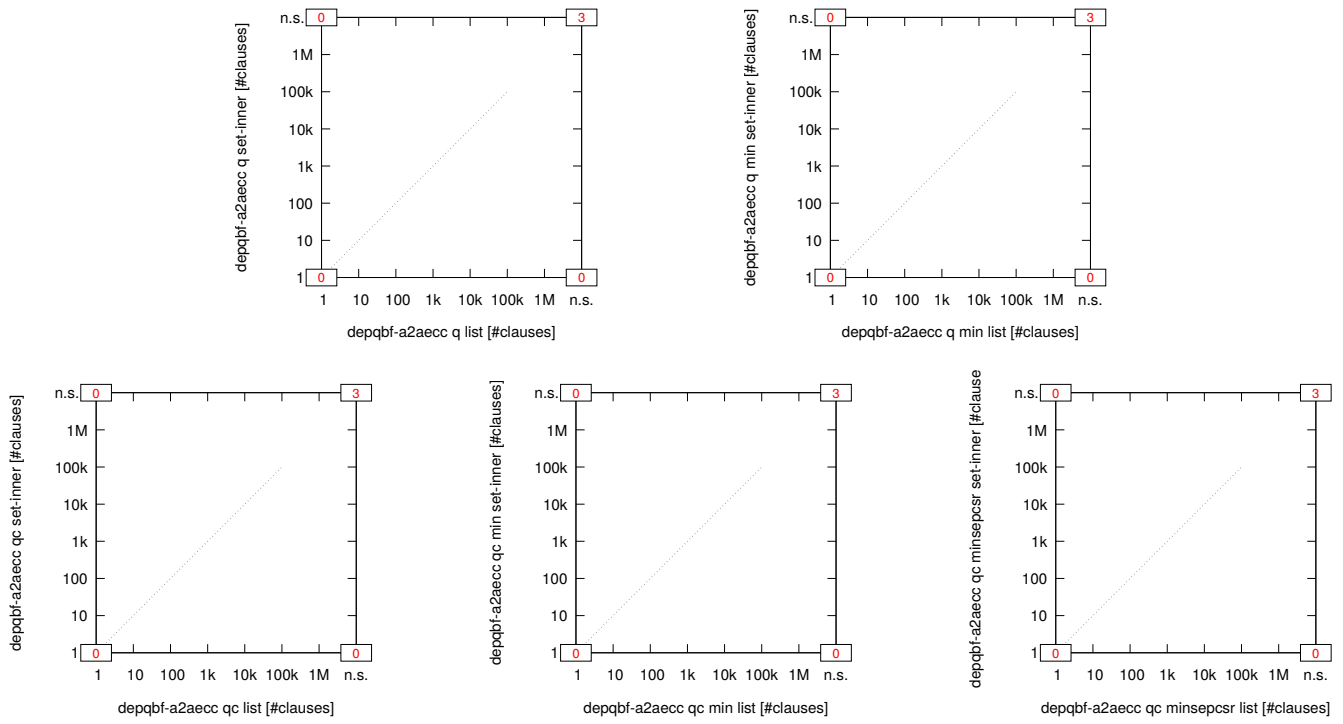


Fig. 283: Suite Rabe ($n = 3$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

40) Rintanen ($n = 55$):

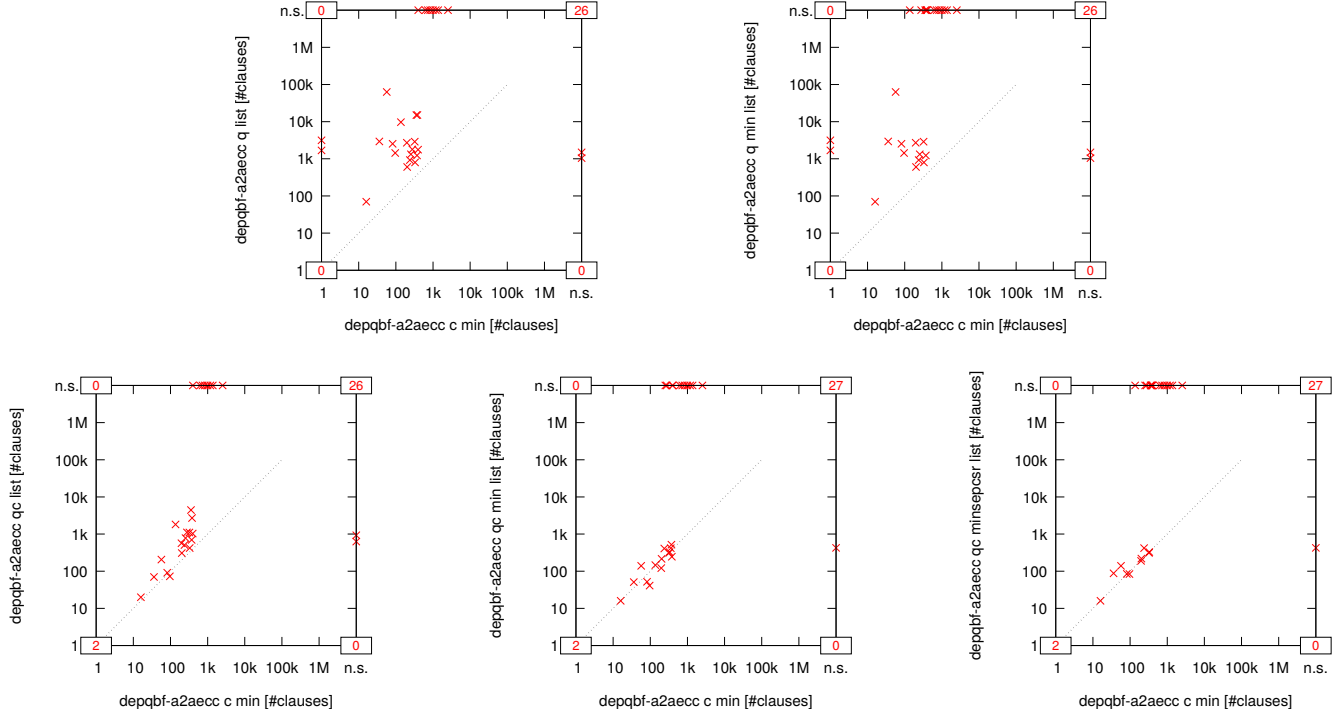


Fig. 284: Suite Rintanen ($n = 55$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

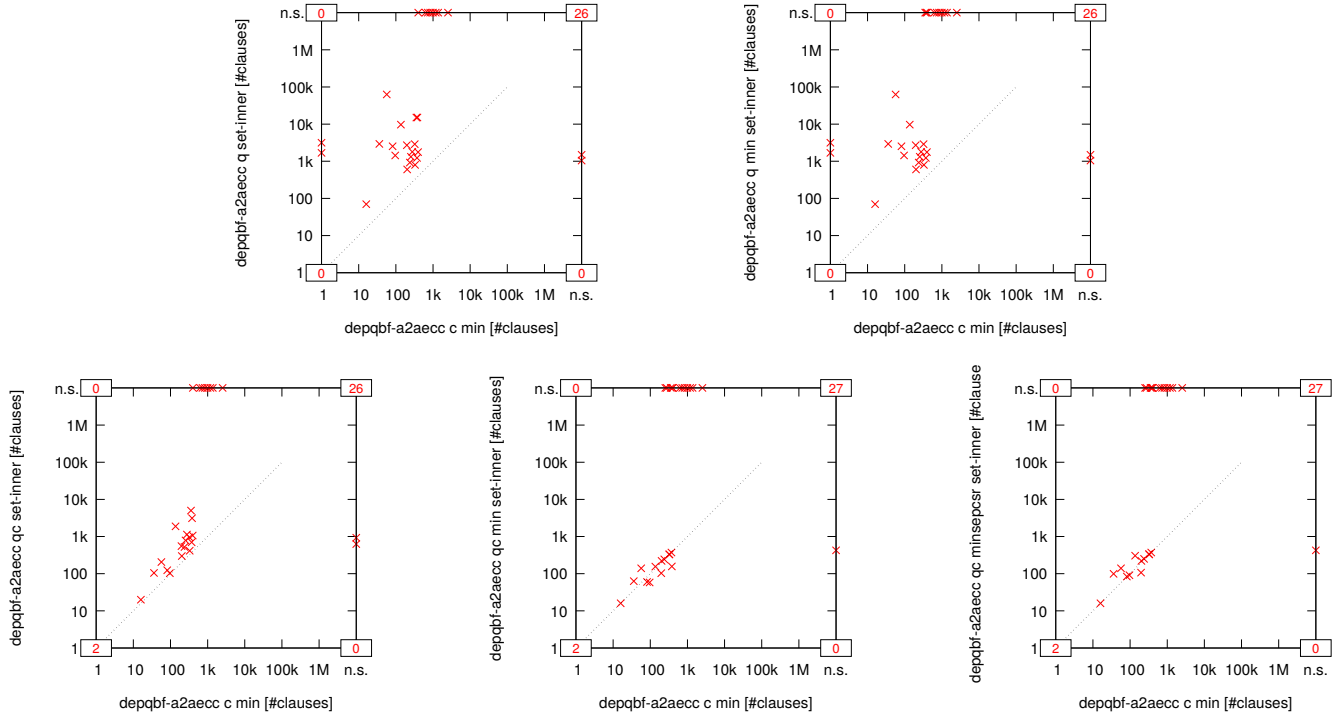


Fig. 285: Suite Rintanen ($n = 55$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

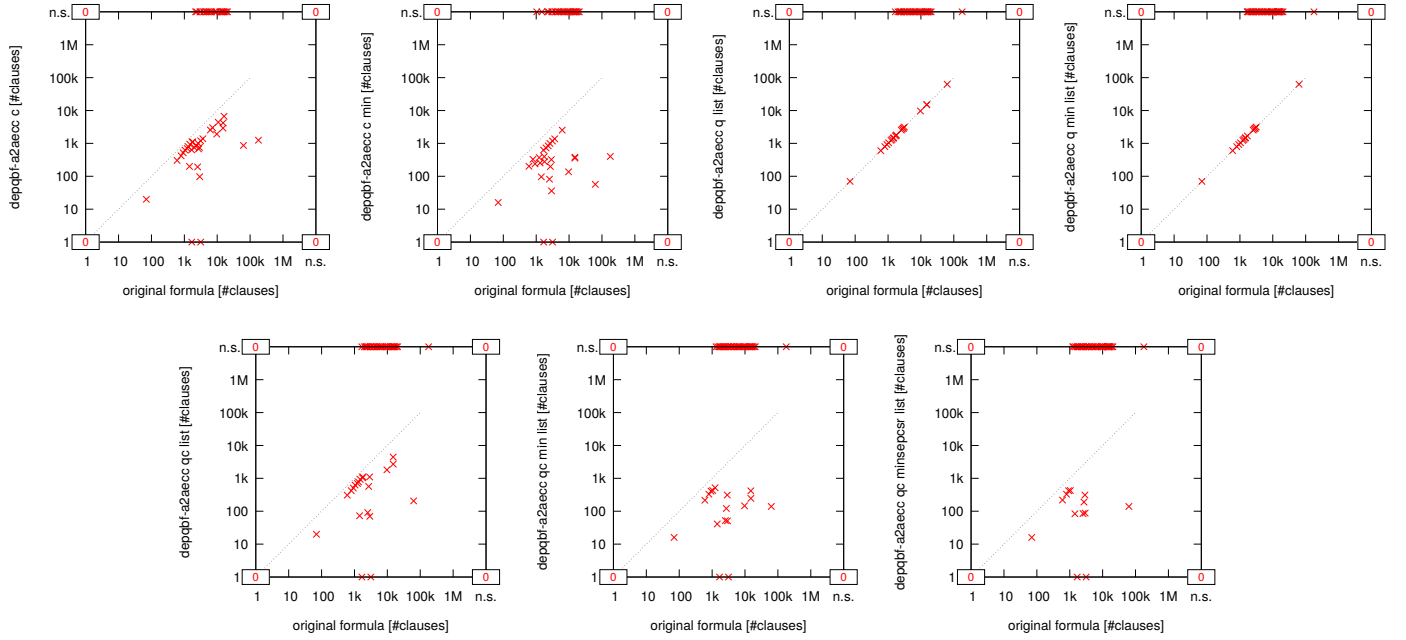


Fig. 286: Suite Rintanen ($n = 55$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

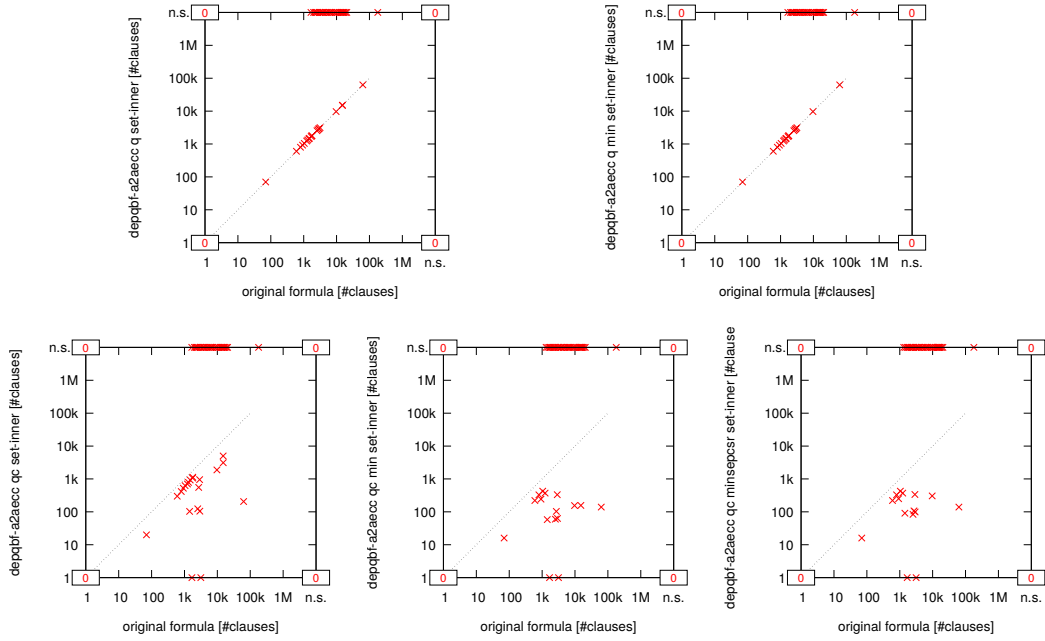


Fig. 287: Suite Rintanen ($n = 55$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

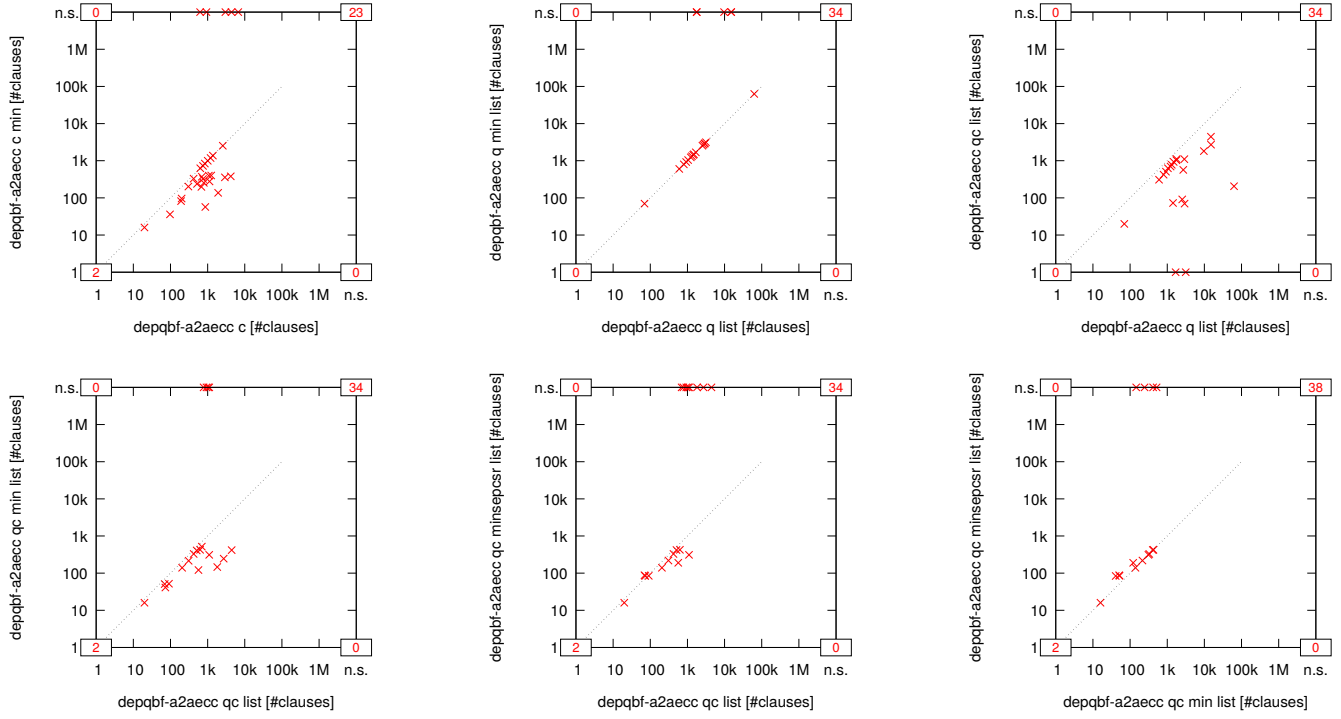


Fig. 288: Suite Rintanen ($n = 55$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

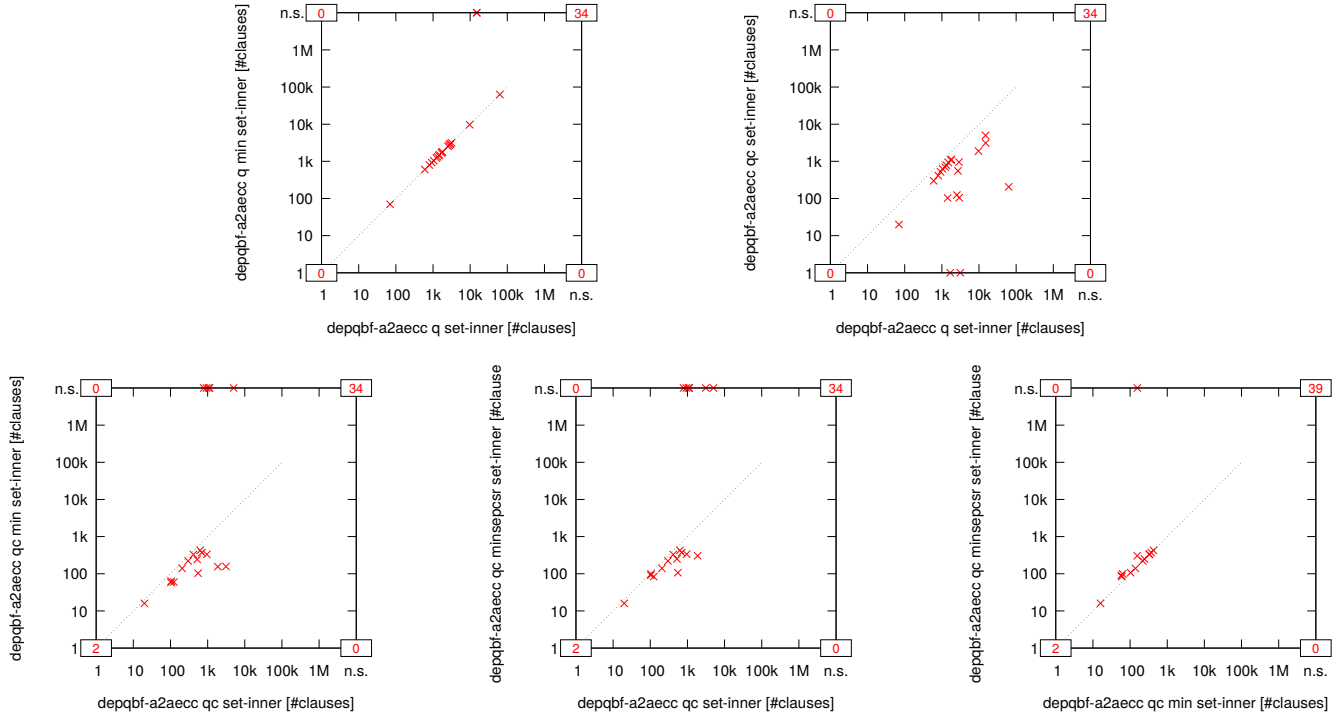


Fig. 289: Suite Rintanen ($n = 55$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

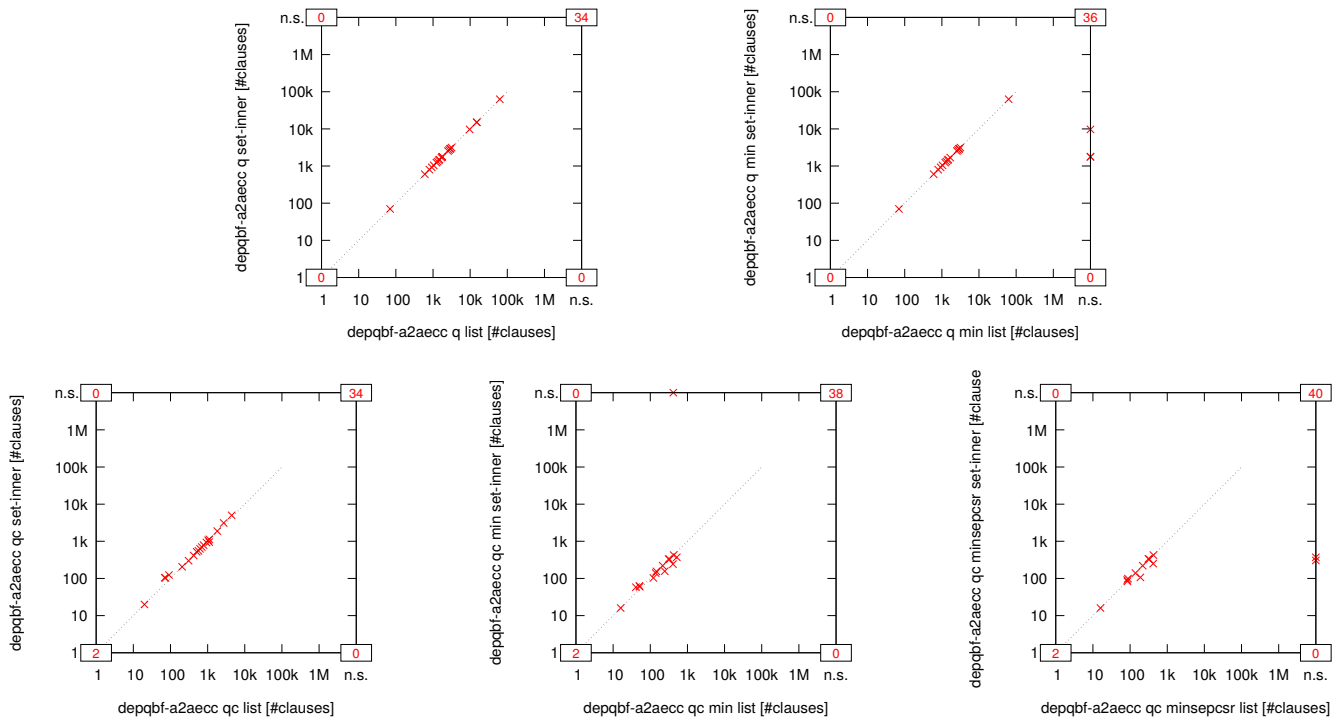


Fig. 290: Suite Rintanen ($n = 55$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

41) Sauer-Reimer ($n = 42$):

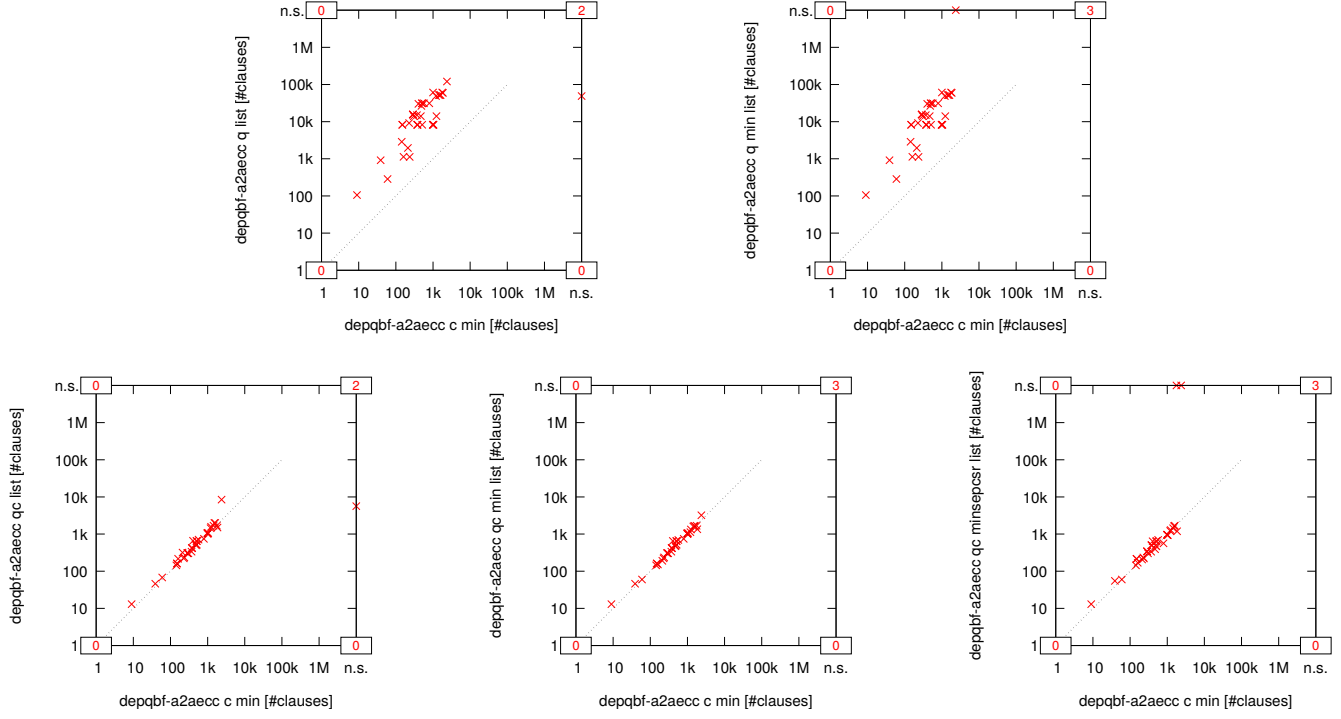


Fig. 291: Suite Sauer-Reimer ($n = 42$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

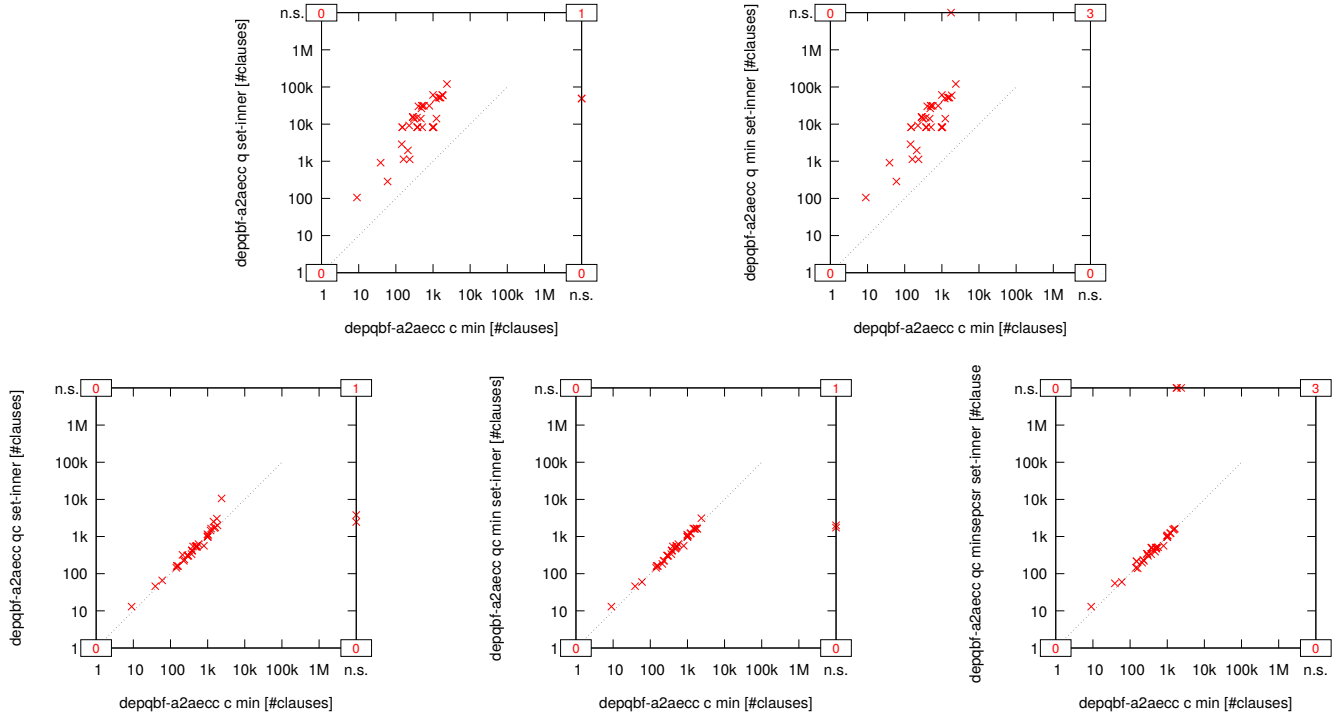


Fig. 292: Suite Sauer-Reimer ($n = 42$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

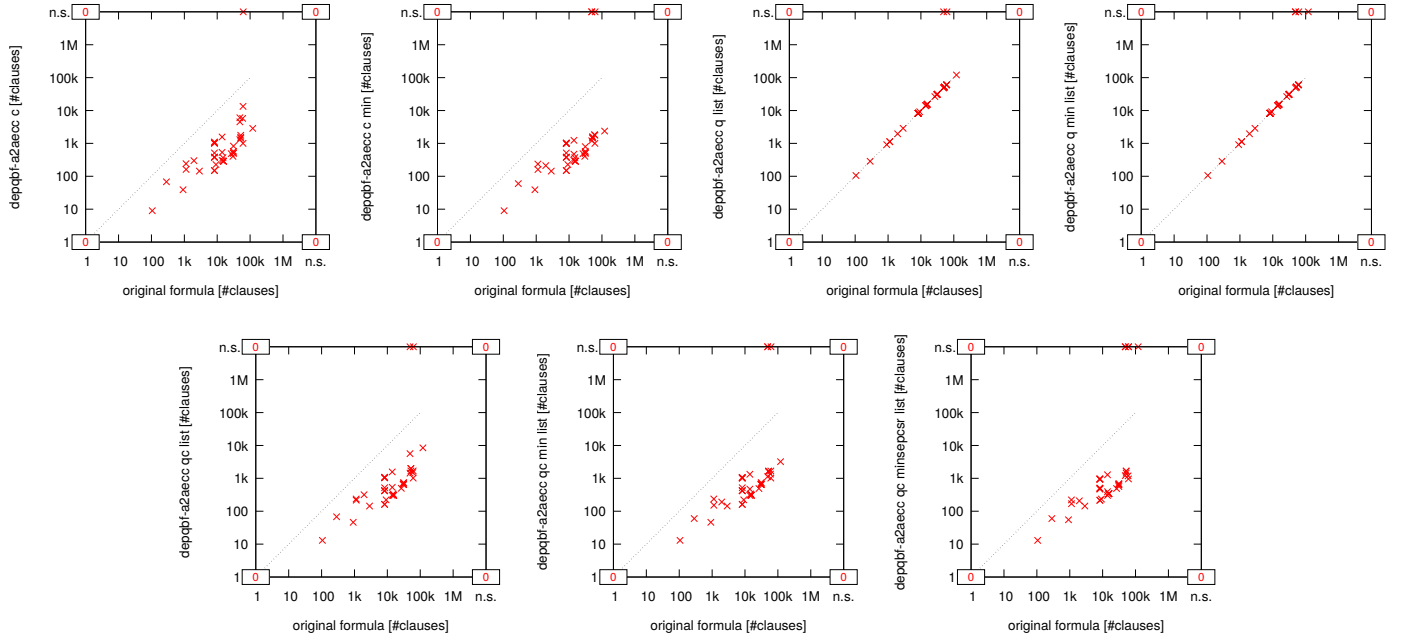


Fig. 293: Suite Sauer-Reimer ($n = 42$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

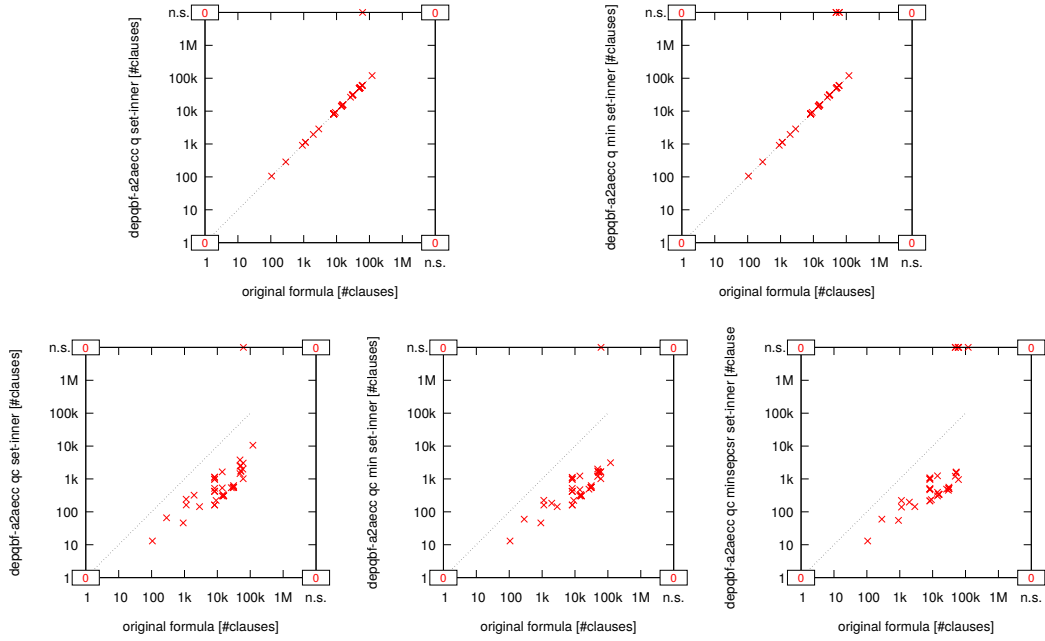


Fig. 294: Suite Sauer-Reimer ($n = 42$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

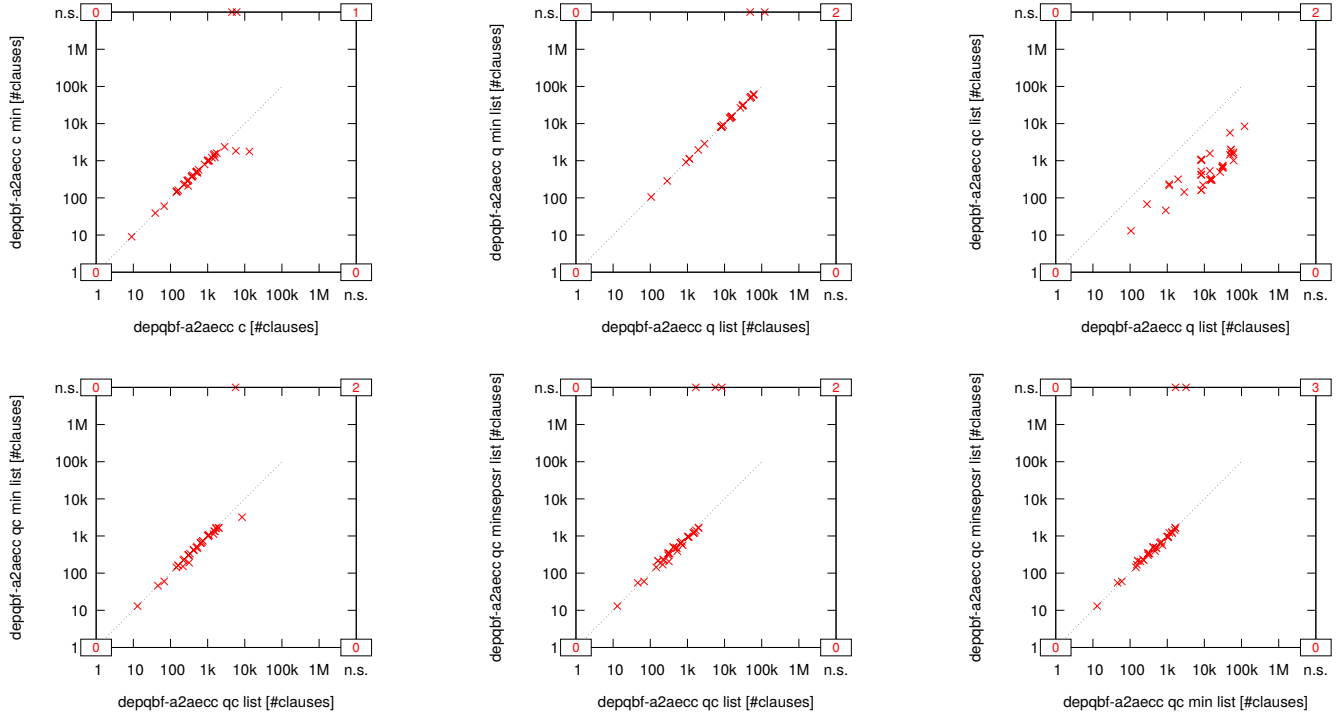


Fig. 295: Suite Sauer-Reimer ($n = 42$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

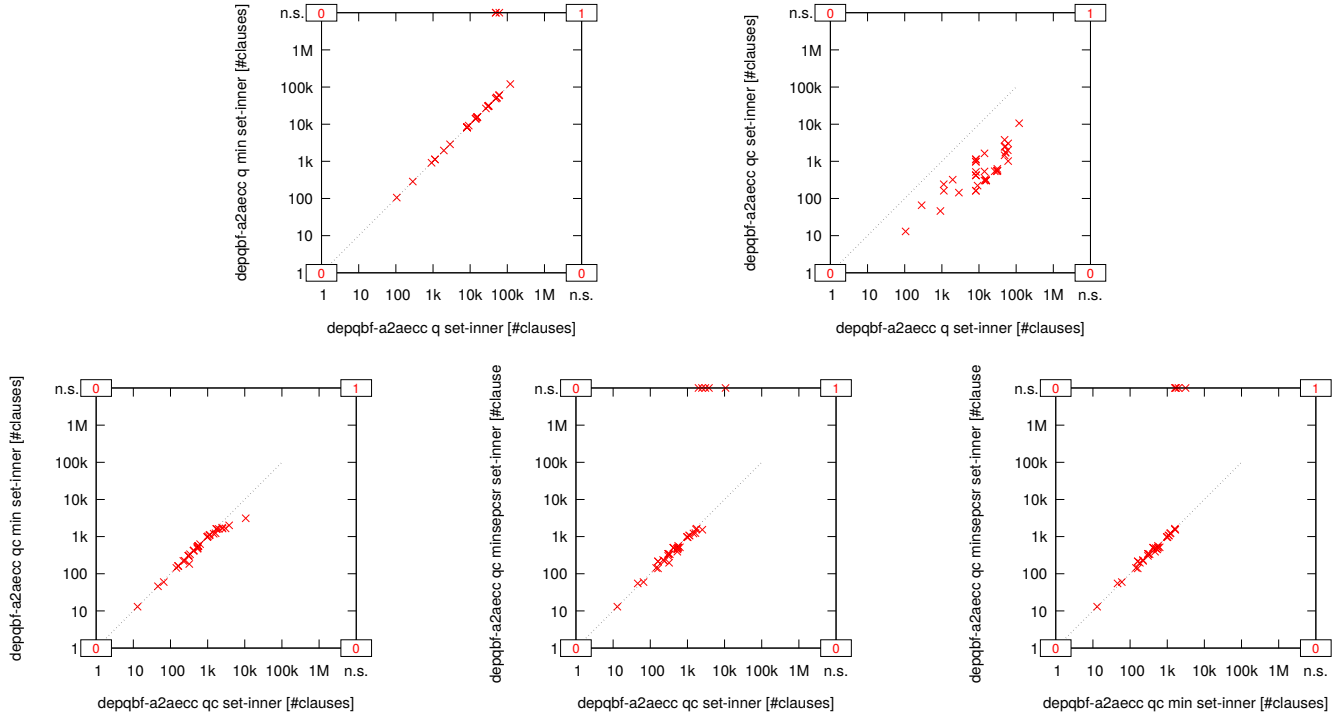


Fig. 296: Suite Sauer-Reimer ($n = 42$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

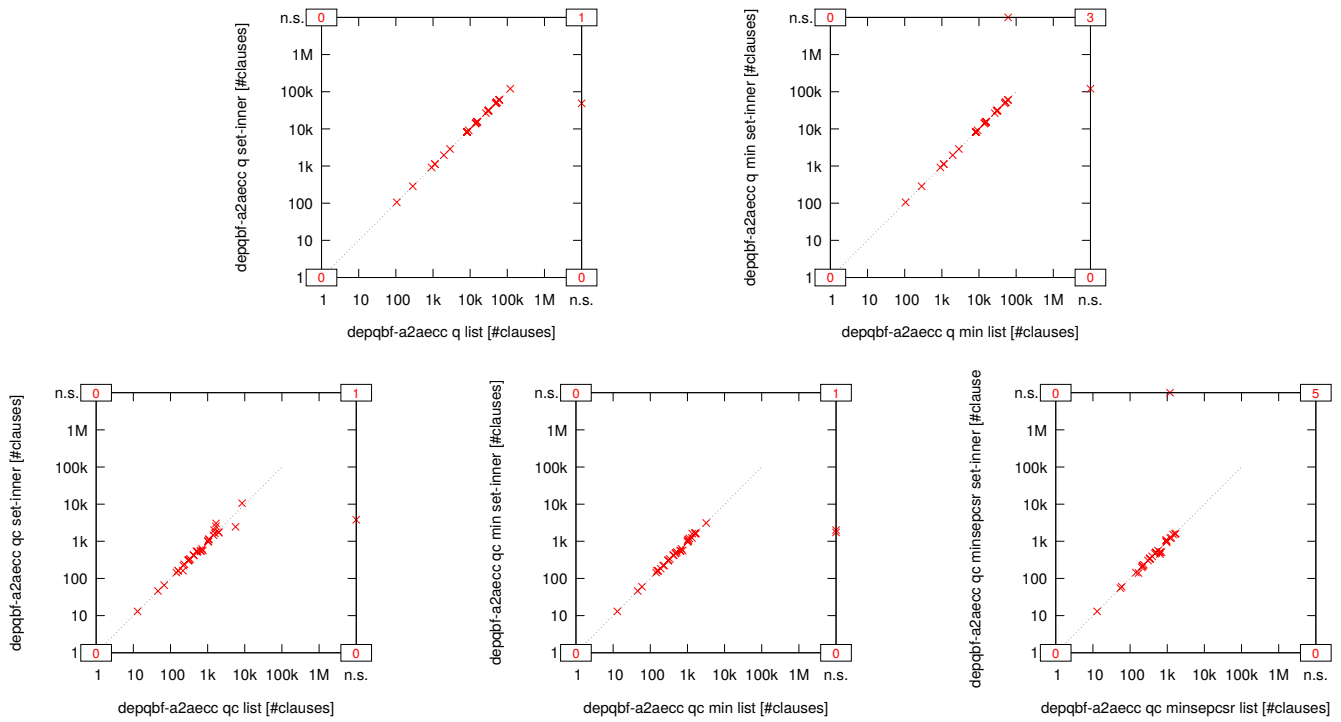


Fig. 297: Suite Sauer-Reimer ($n = 42$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

42) Scholl-Becker ($n = 30$):

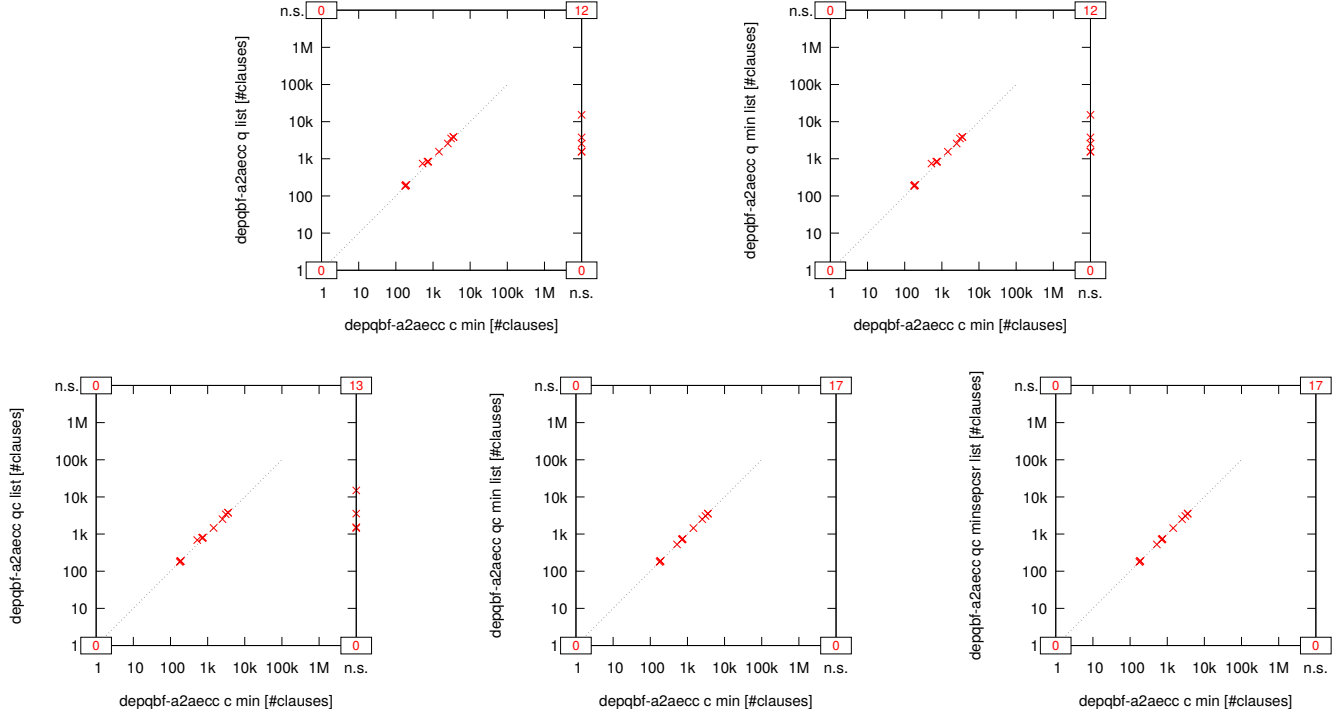


Fig. 298: Suite Scholl-Becker ($n = 30$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

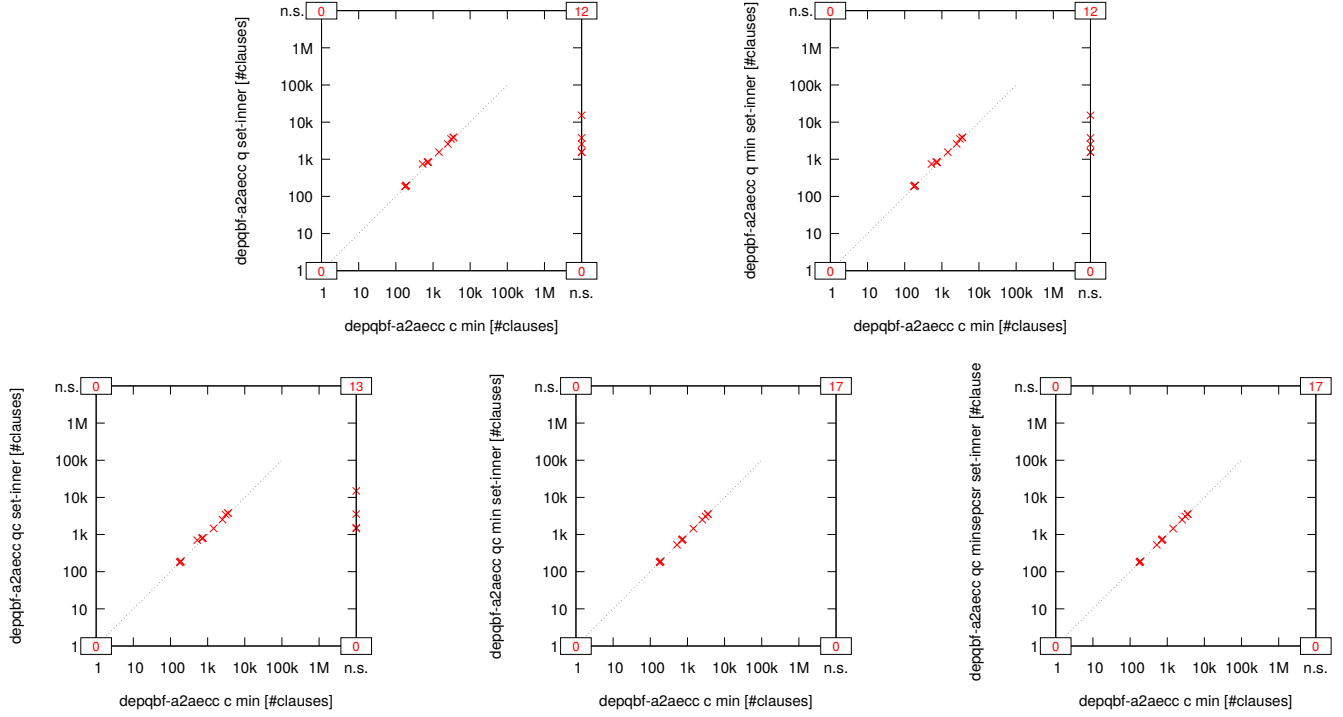


Fig. 299: Suite Scholl-Becker ($n = 30$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

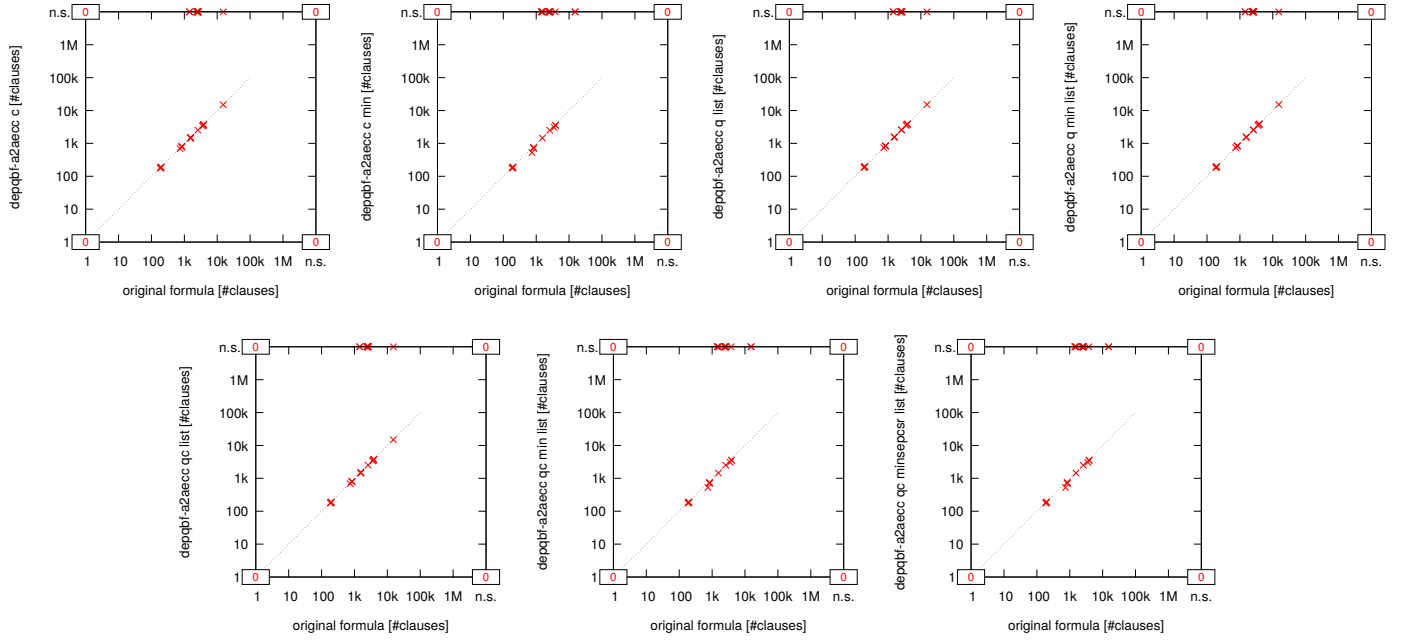


Fig. 300: Suite Scholl-Becker ($n = 30$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

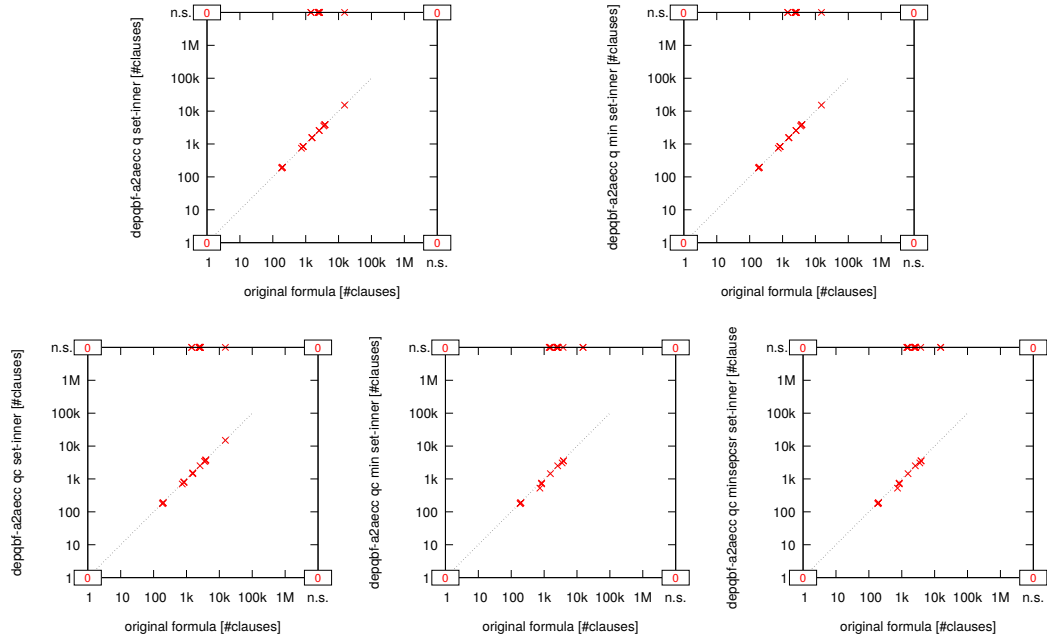


Fig. 301: Suite Scholl-Becker ($n = 30$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

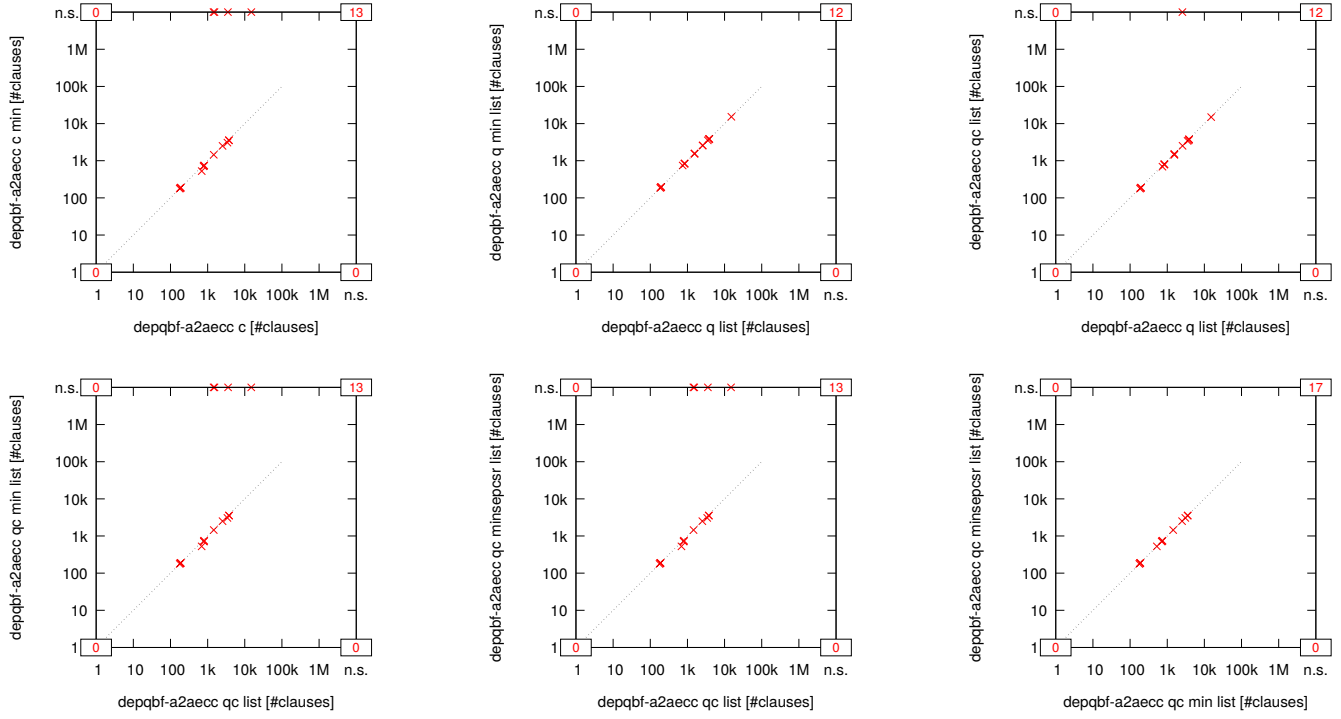


Fig. 302: Suite Scholl-Becker ($n = 30$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

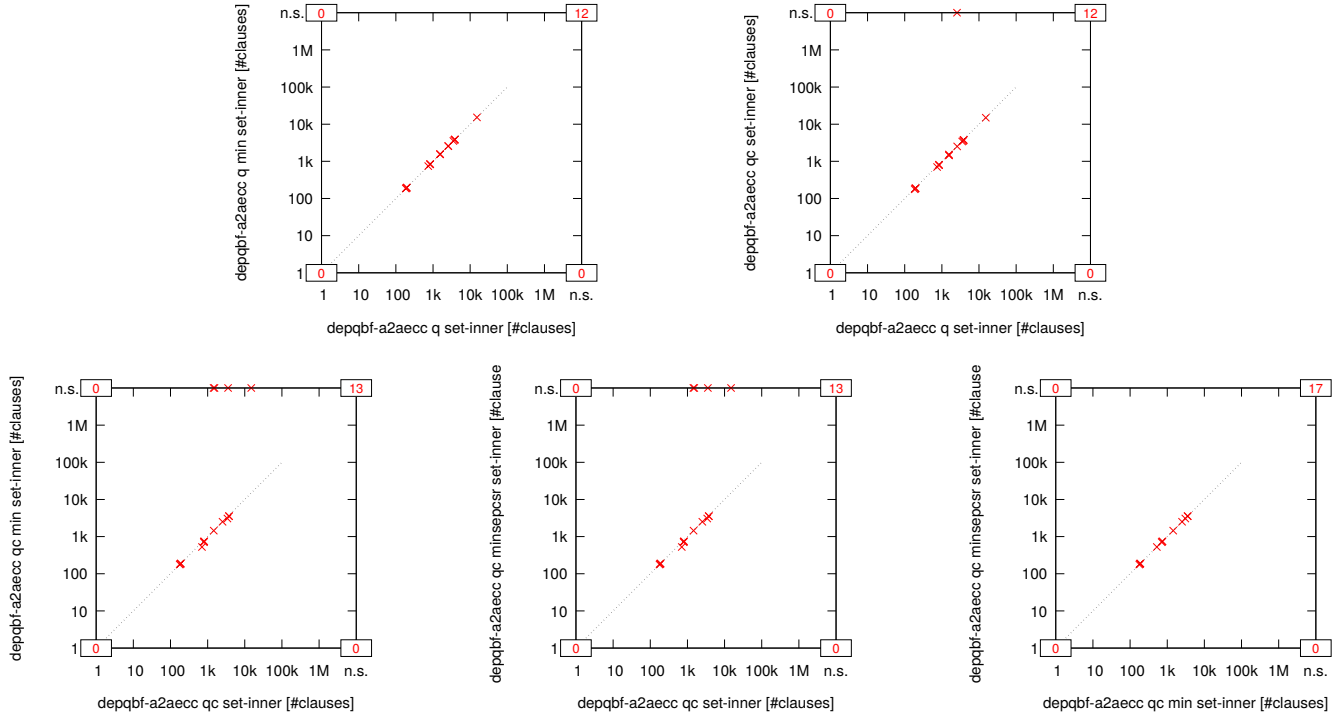


Fig. 303: Suite Scholl-Becker ($n = 30$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

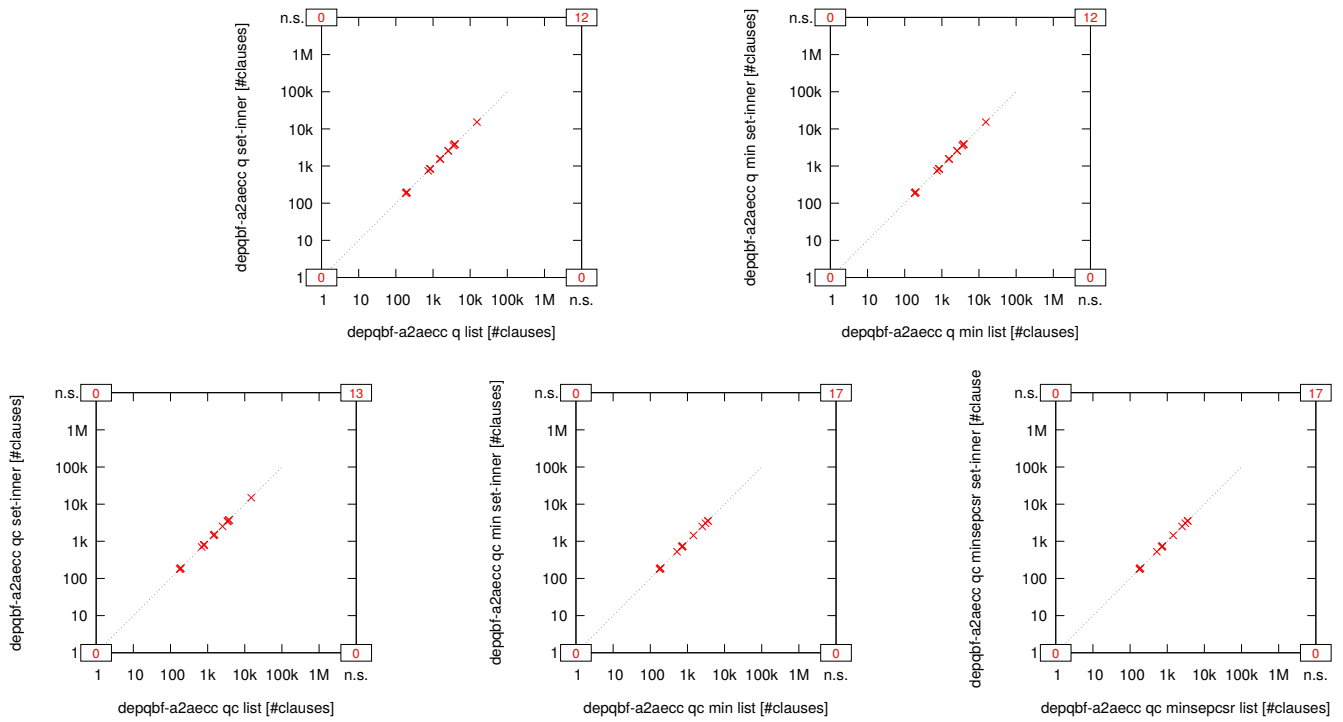


Fig. 304: Suite Scholl-Becker ($n = 30$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

43) Seidl ($n = 194$):

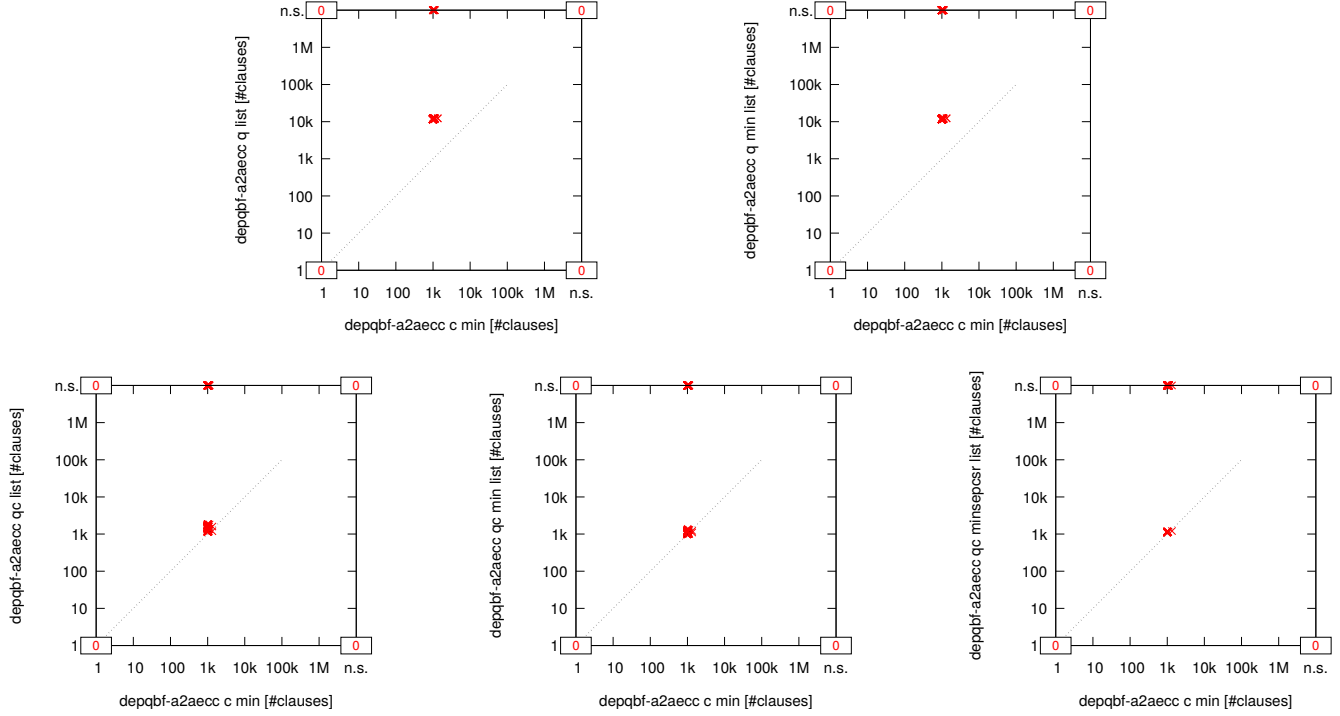


Fig. 305: Suite Seidl ($n = 194$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

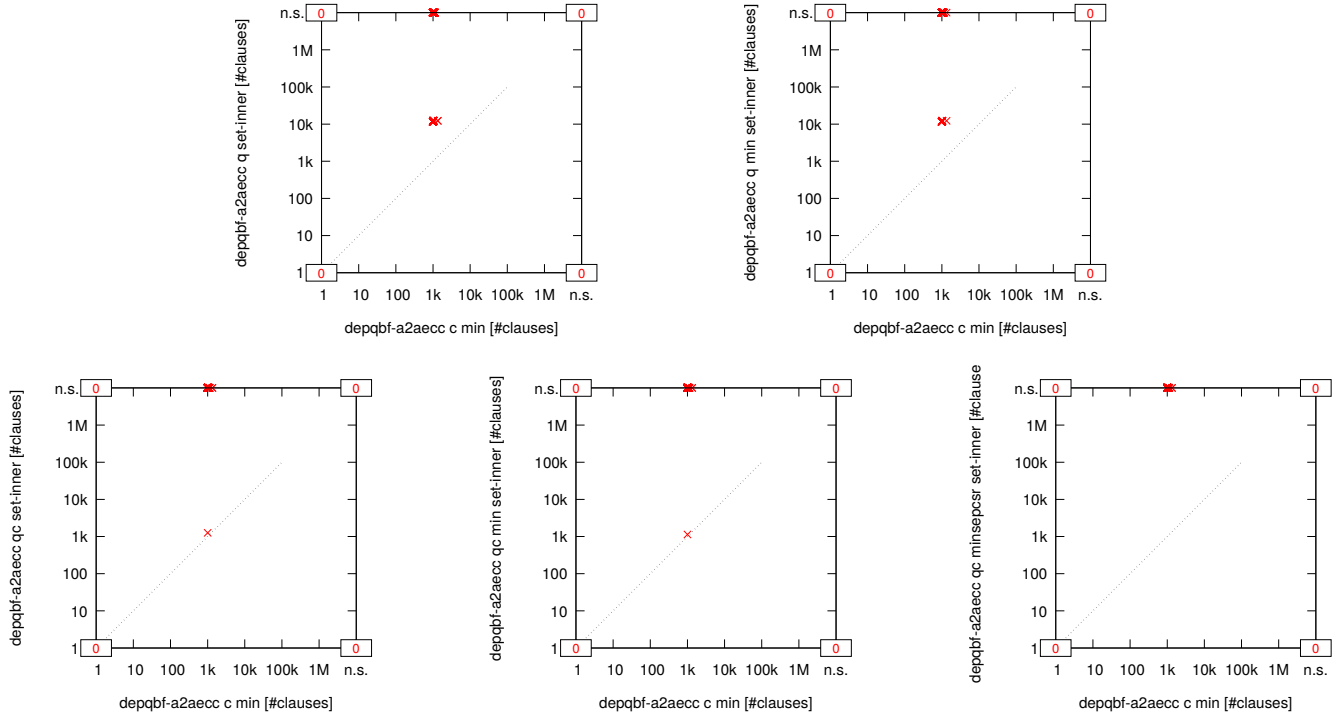


Fig. 306: Suite Seidl ($n = 194$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

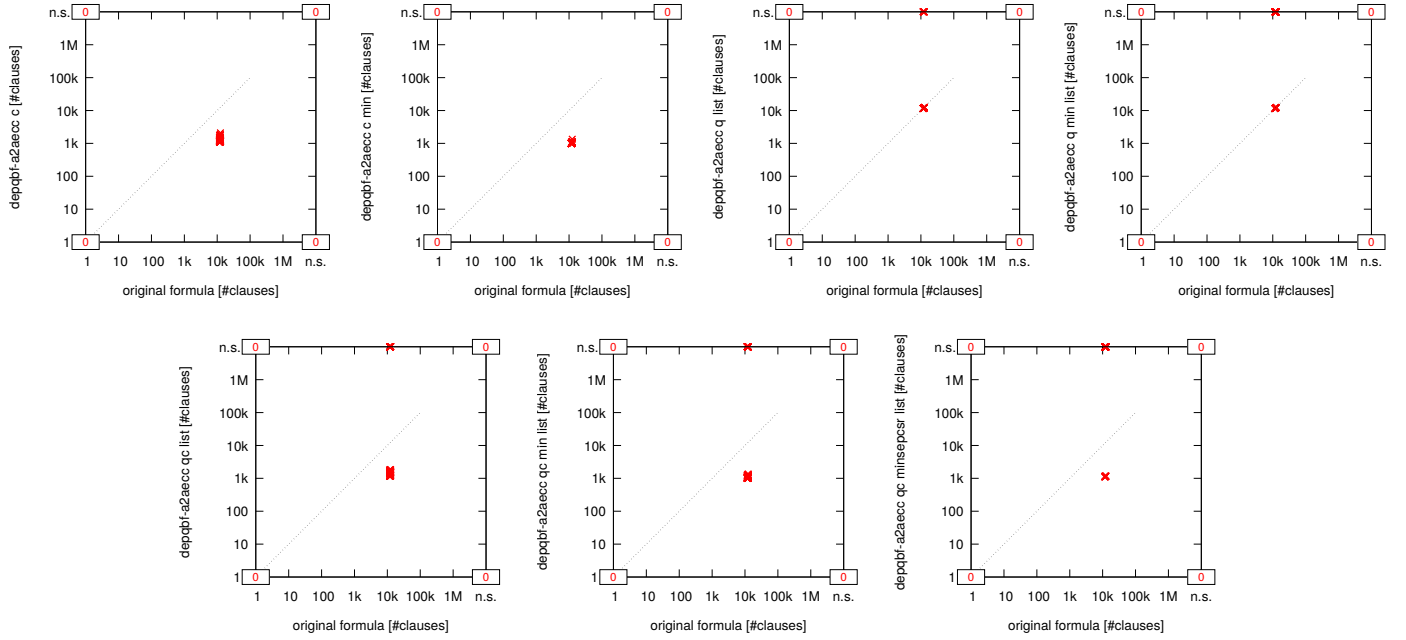


Fig. 307: Suite Seidl ($n = 194$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

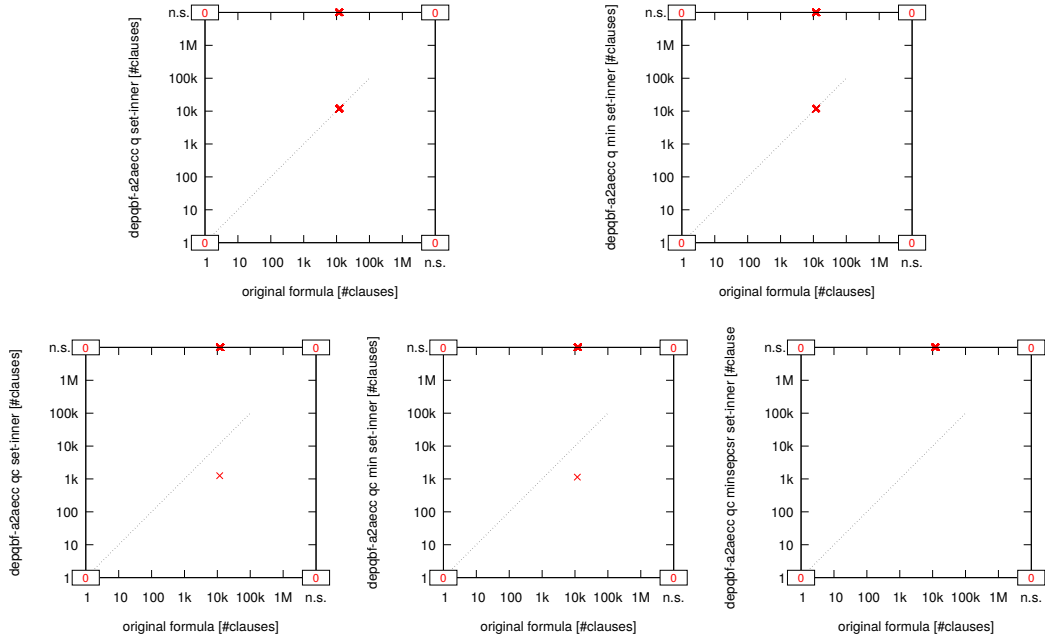


Fig. 308: Suite Seidl ($n = 194$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

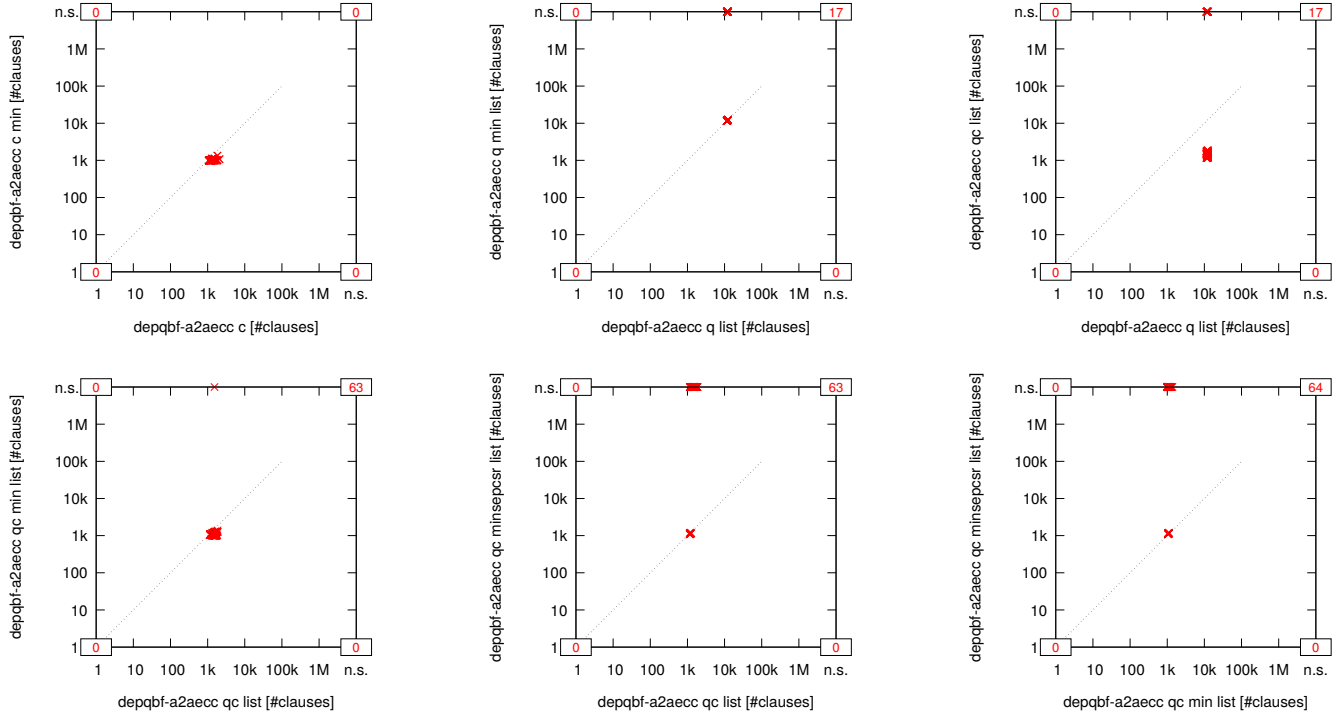


Fig. 309: Suite Seidl ($n = 194$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

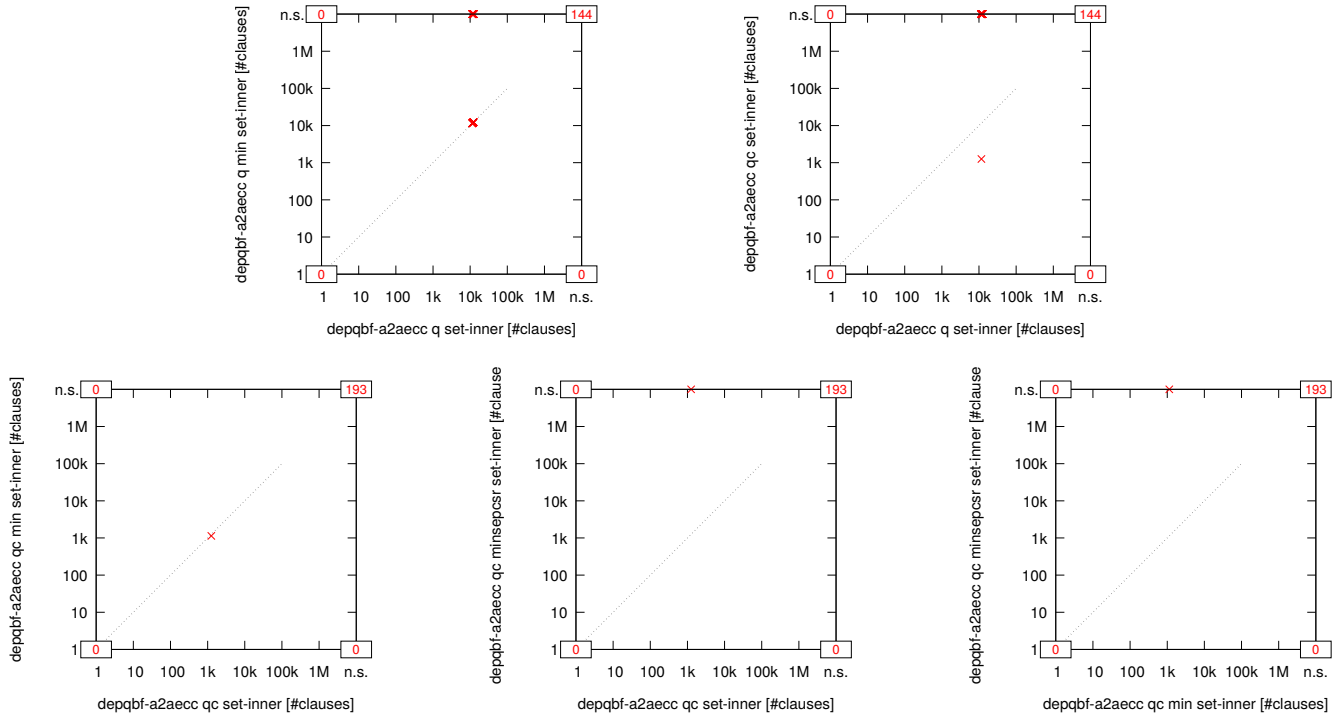


Fig. 310: Suite Seidl ($n = 194$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

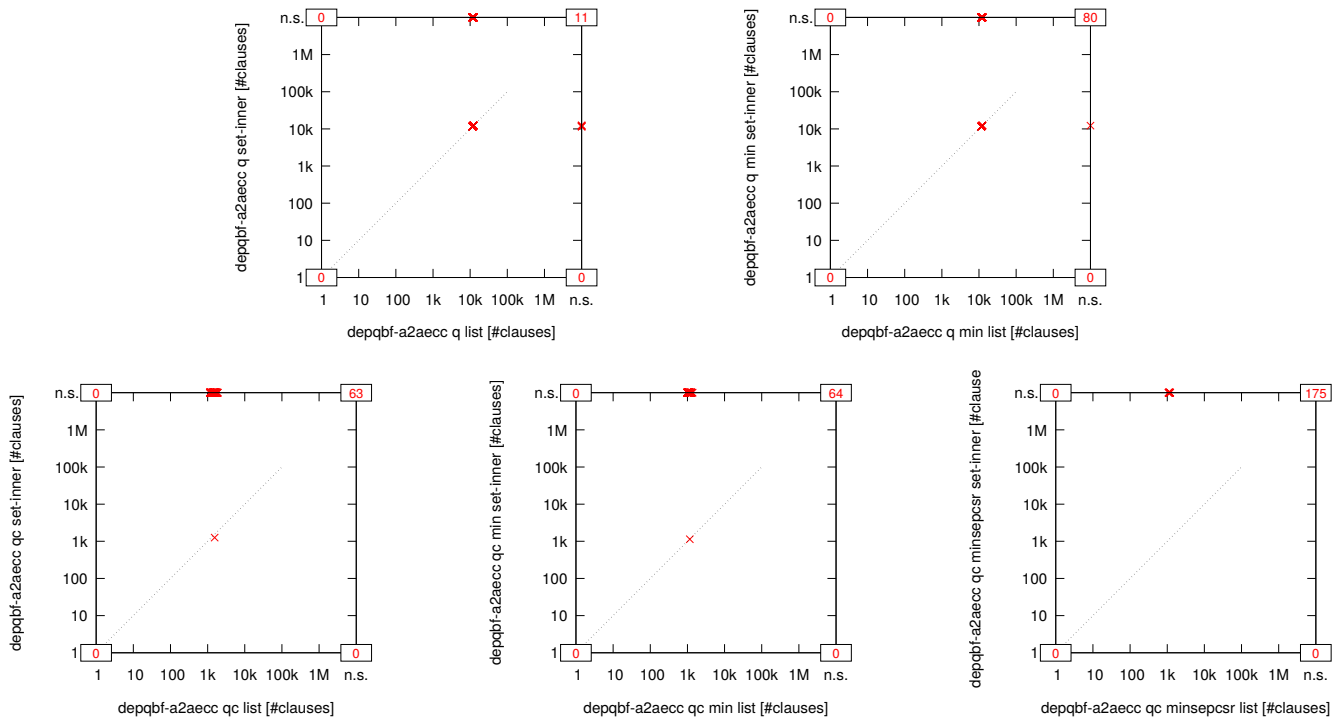


Fig. 311: Suite Seidl ($n = 194$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

44) Tacchella ($n = 122$):

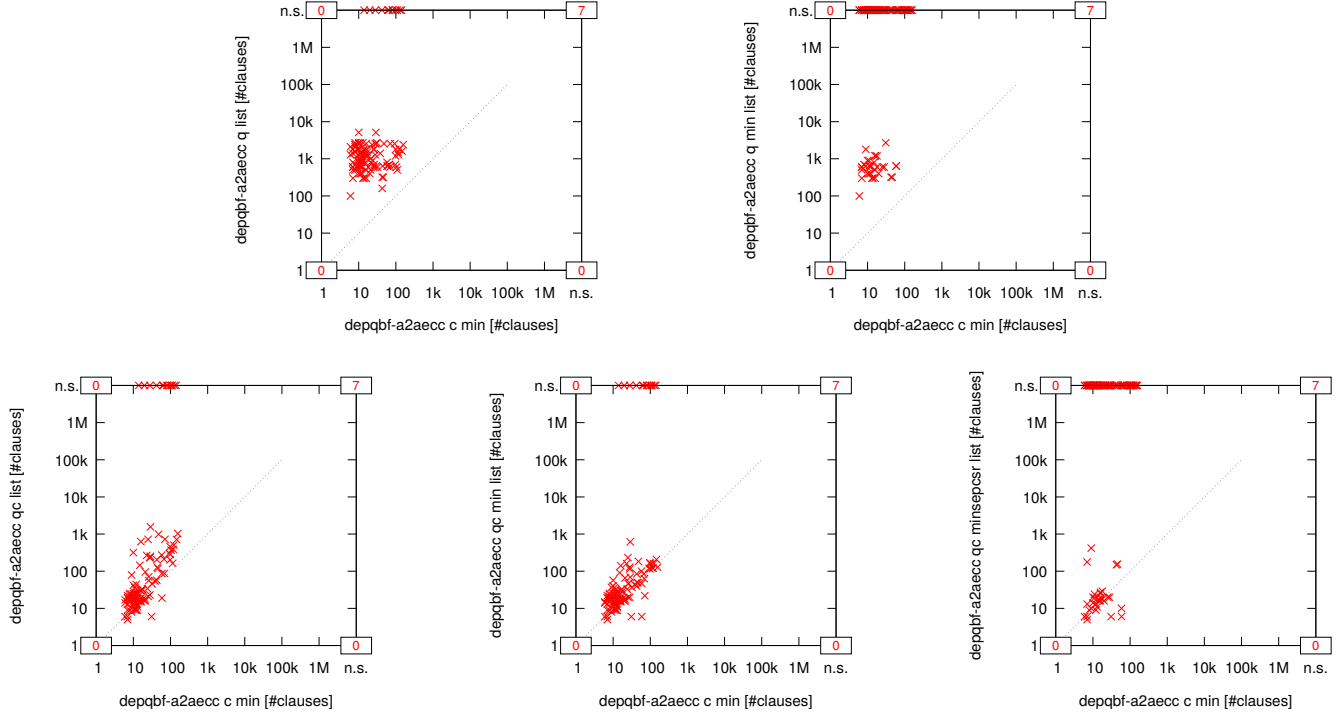


Fig. 312: Suite Tacchella ($n = 122$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

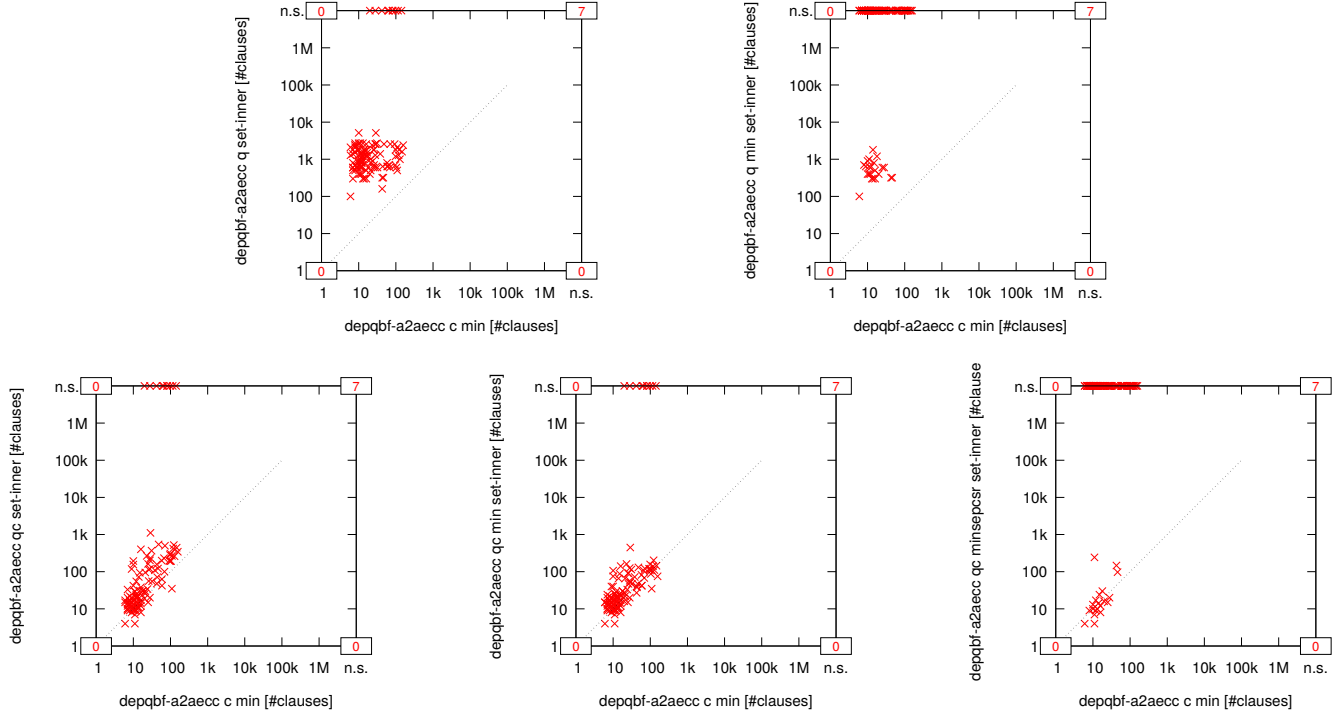


Fig. 313: Suite Tacchella ($n = 122$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

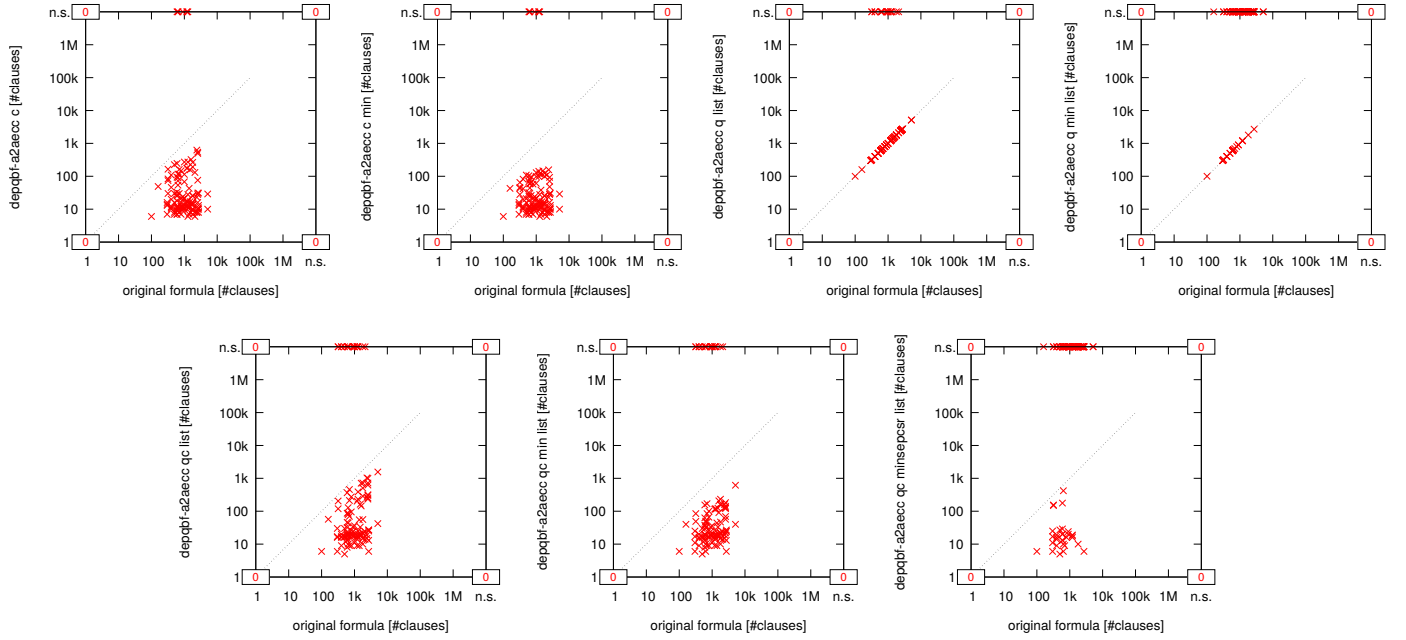


Fig. 314: Suite Tacchella ($n = 122$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

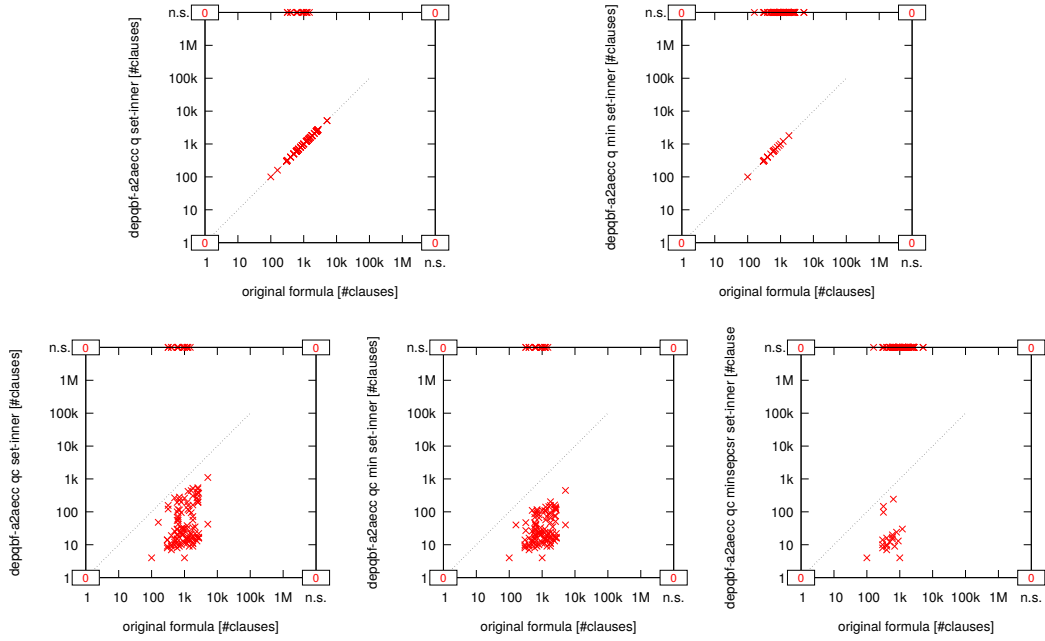


Fig. 315: Suite Tacchella ($n = 122$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

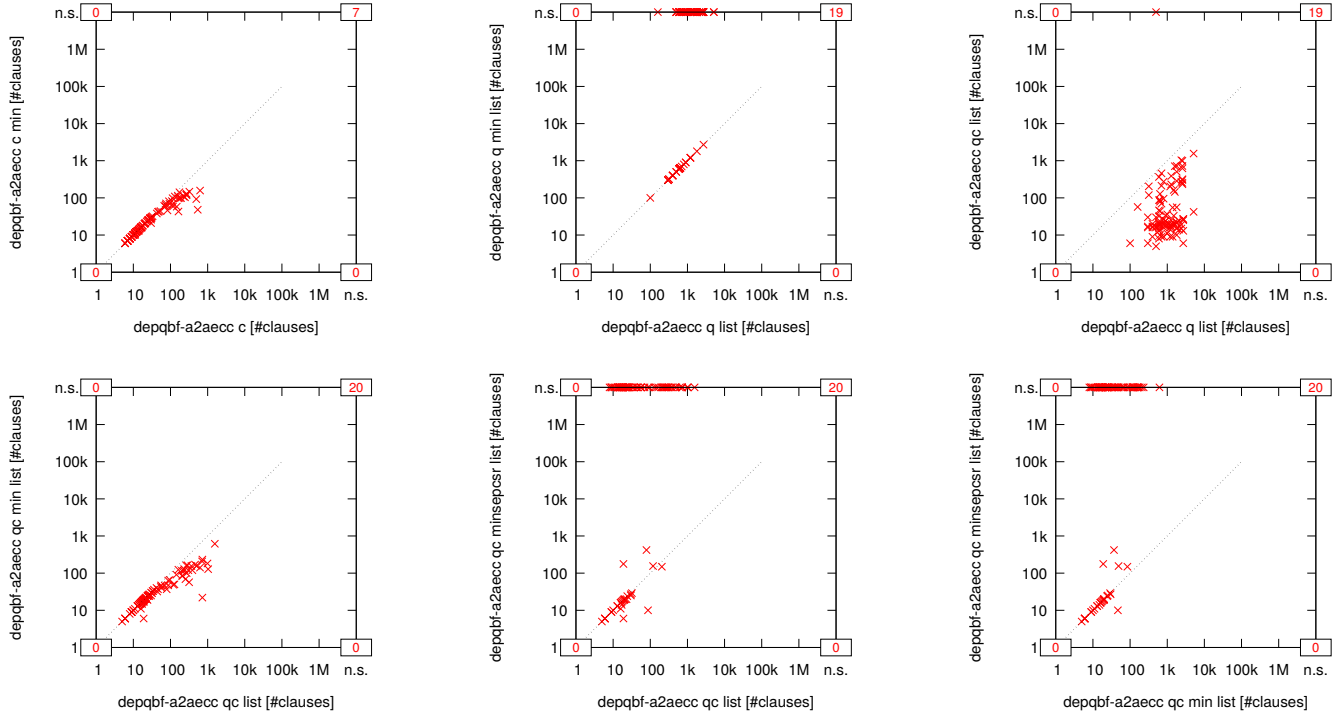


Fig. 316: Suite Tacchella ($n = 122$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

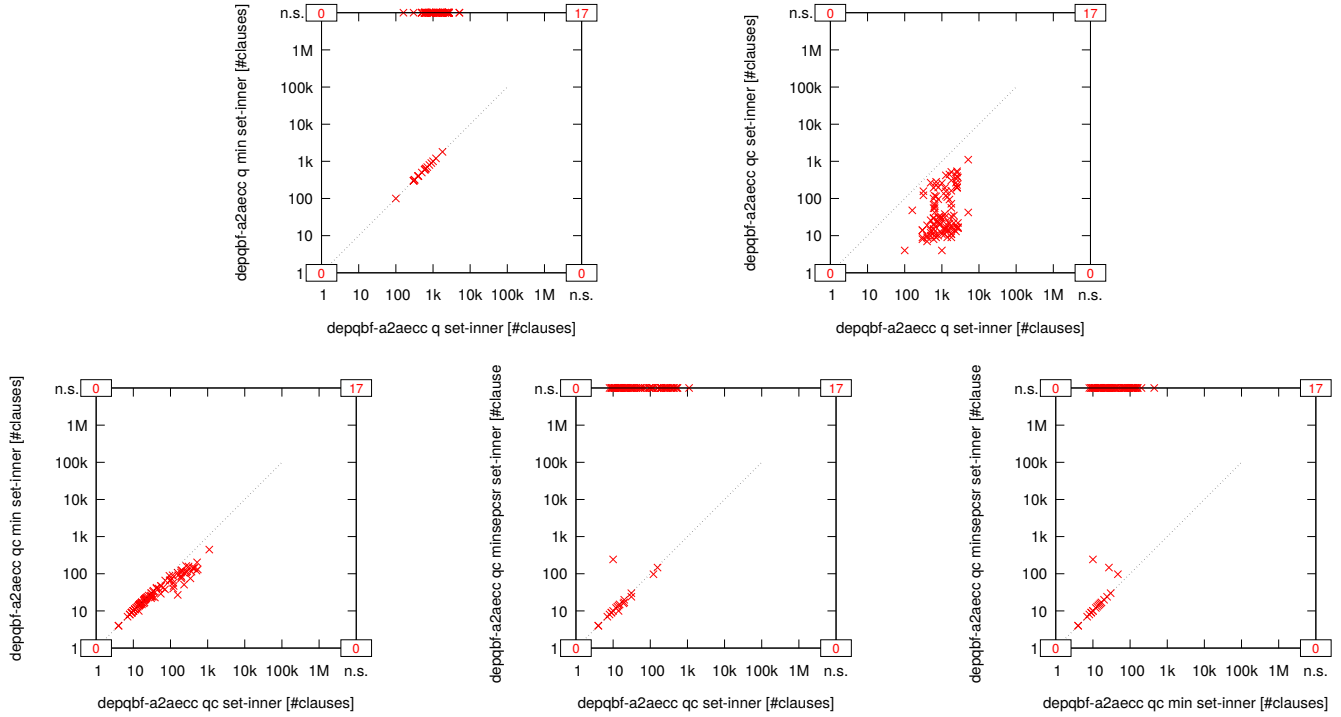


Fig. 317: Suite Tacchella ($n = 122$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

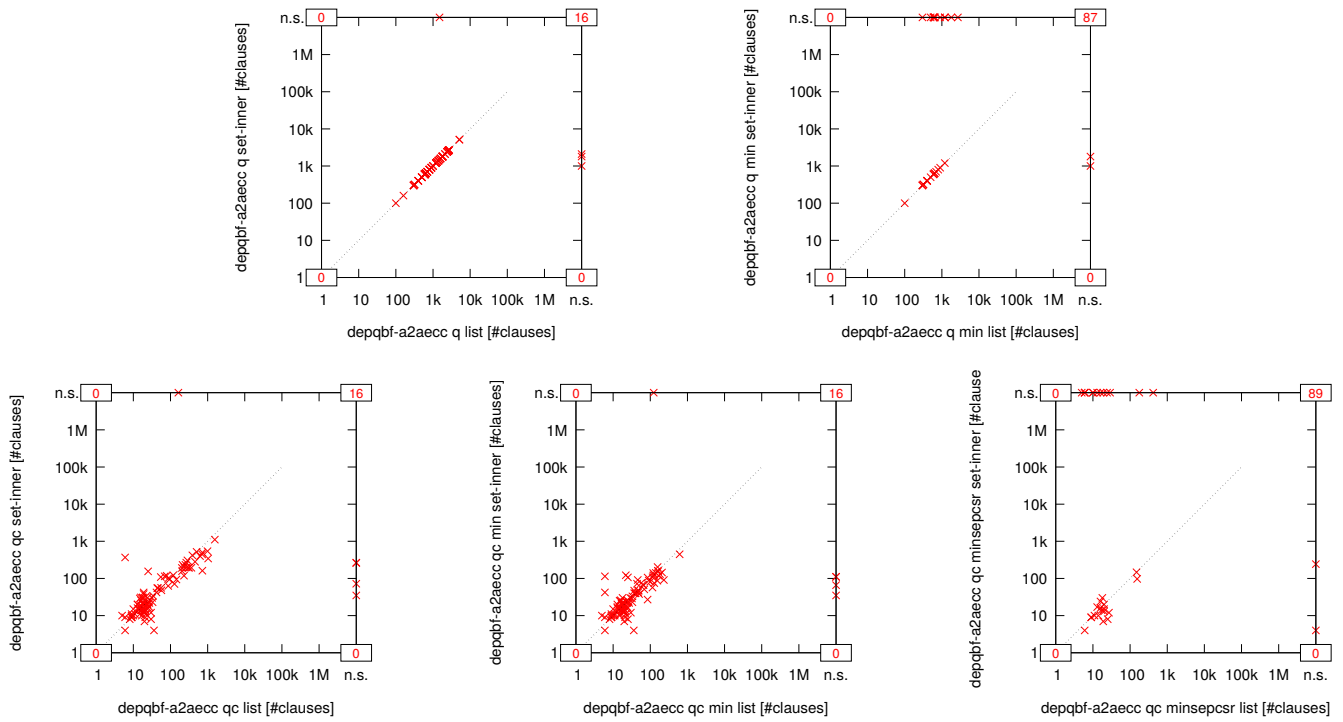


Fig. 318: Suite Tacchella ($n = 122$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

45) *Tentrup* ($n = 17$):

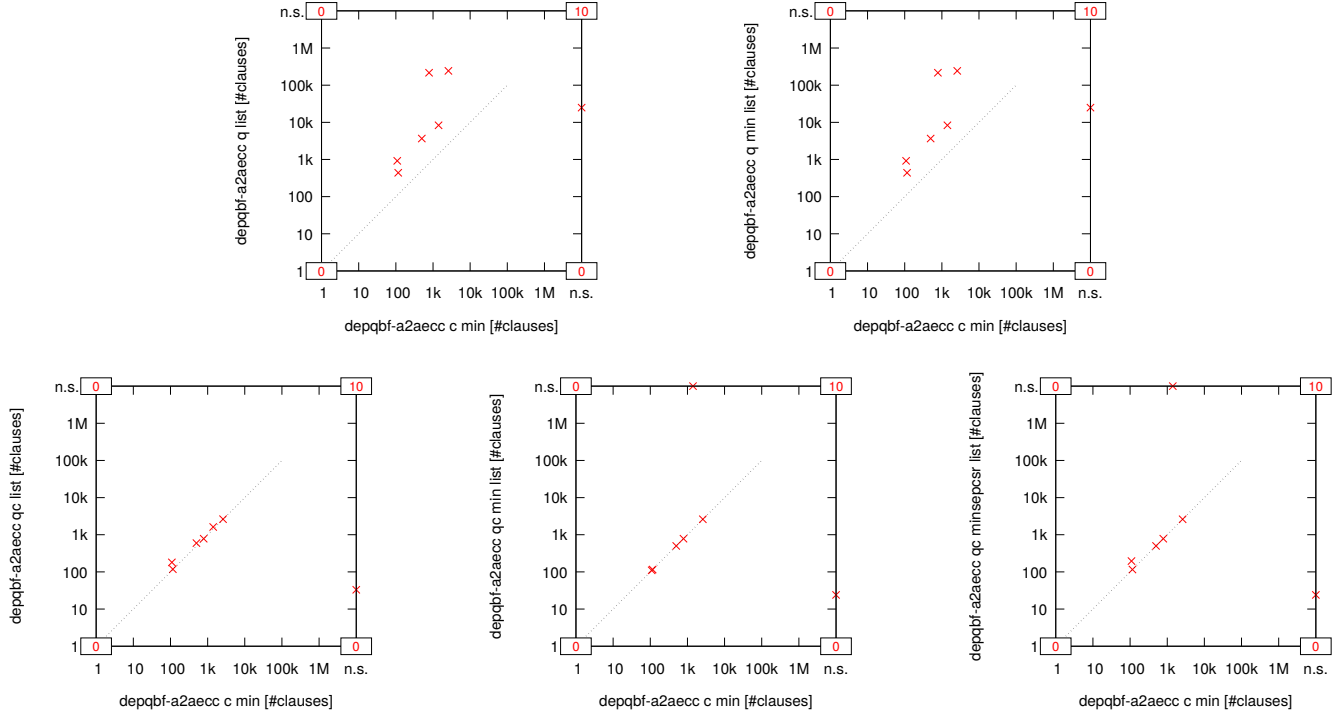


Fig. 319: Suite Tentrup ($n = 17$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

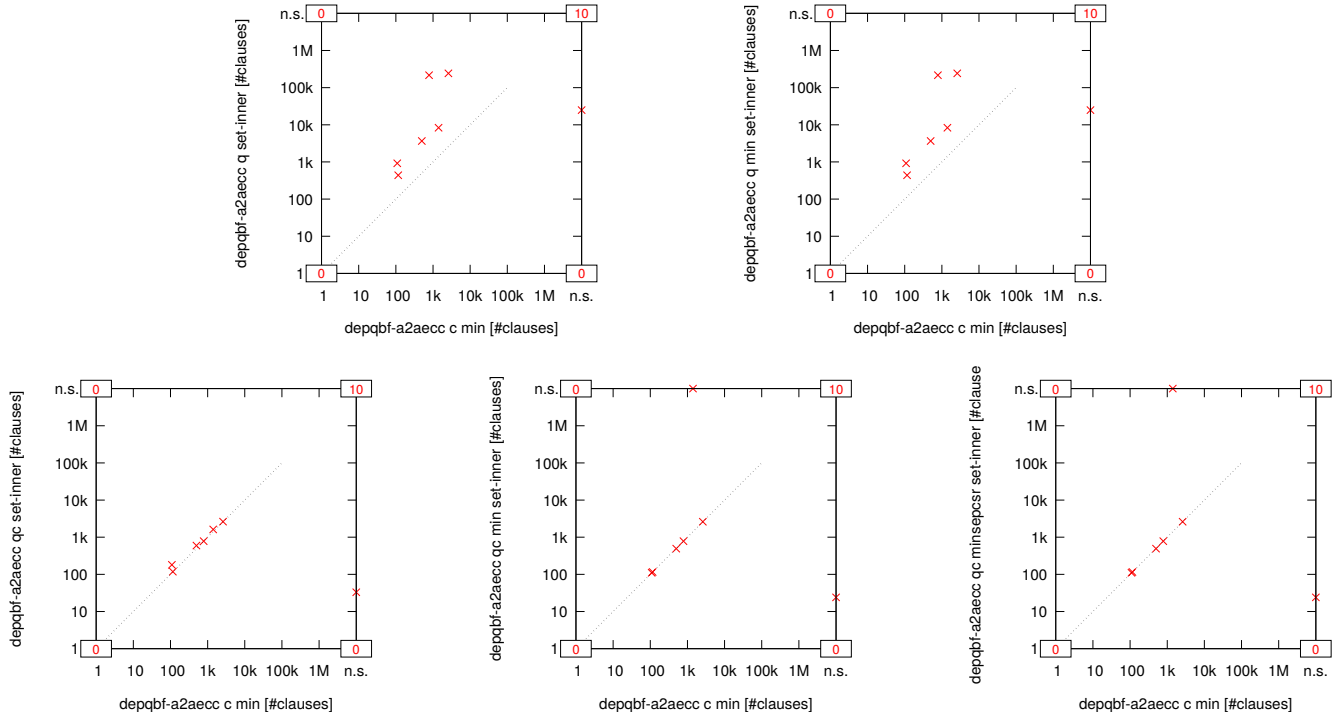


Fig. 320: Suite Tentrup ($n = 17$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

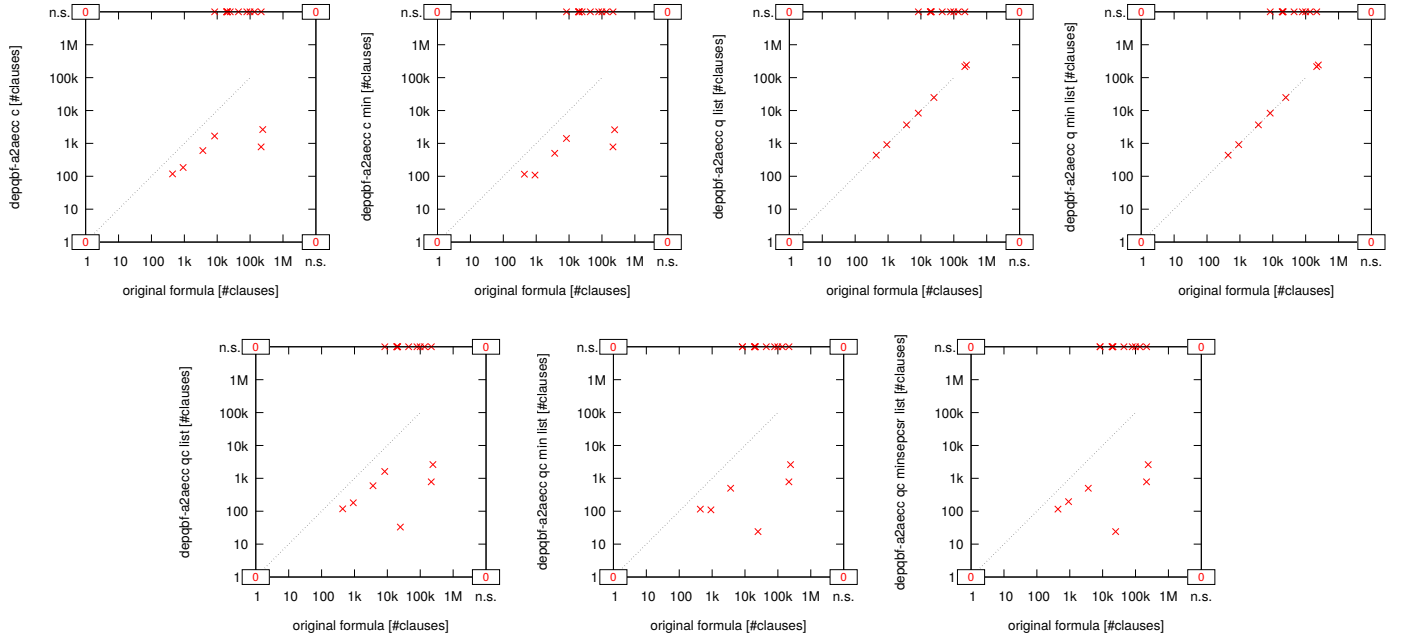


Fig. 321: Suite Tentrup ($n = 17$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

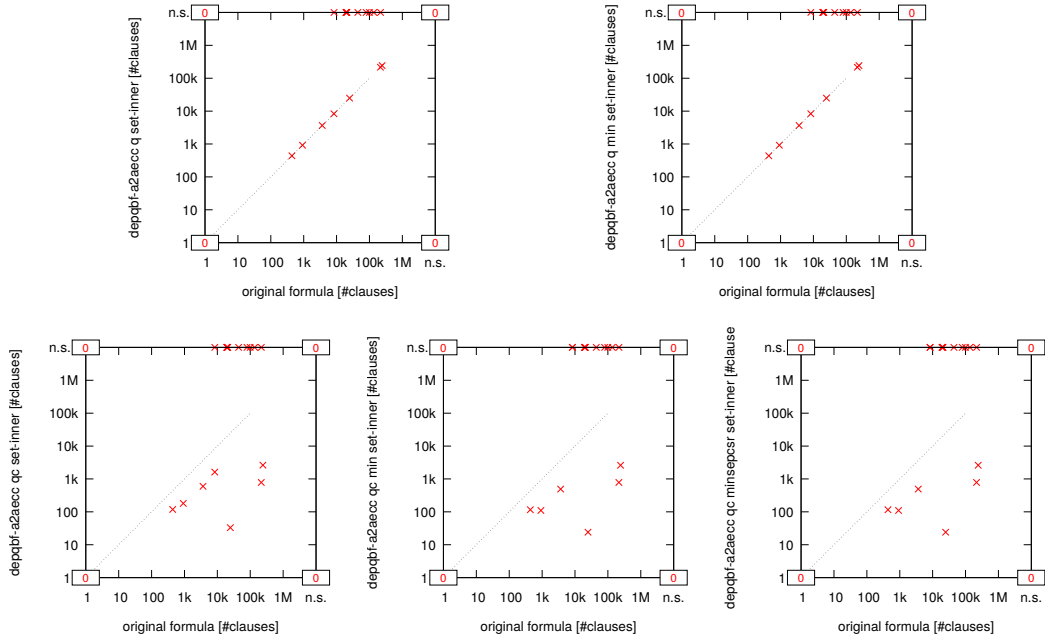


Fig. 322: Suite Tentrup ($n = 17$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

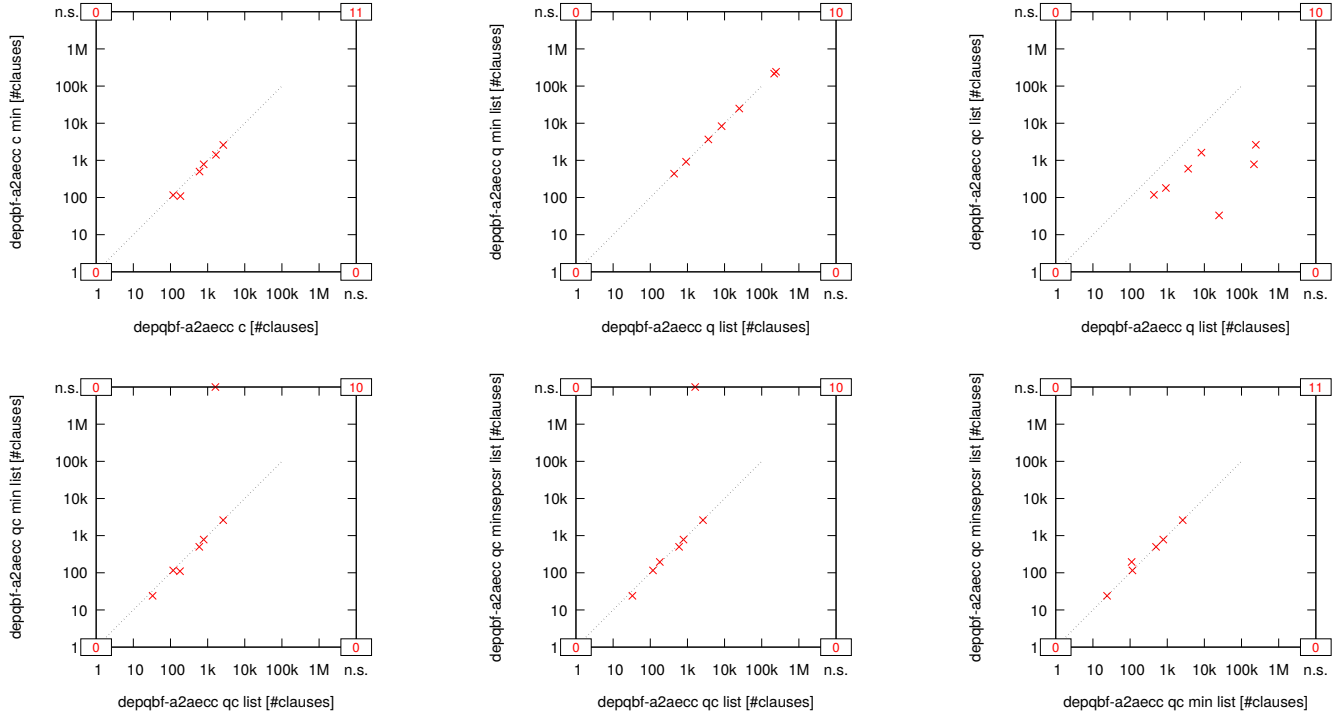


Fig. 323: Suite Tentrup ($n = 17$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

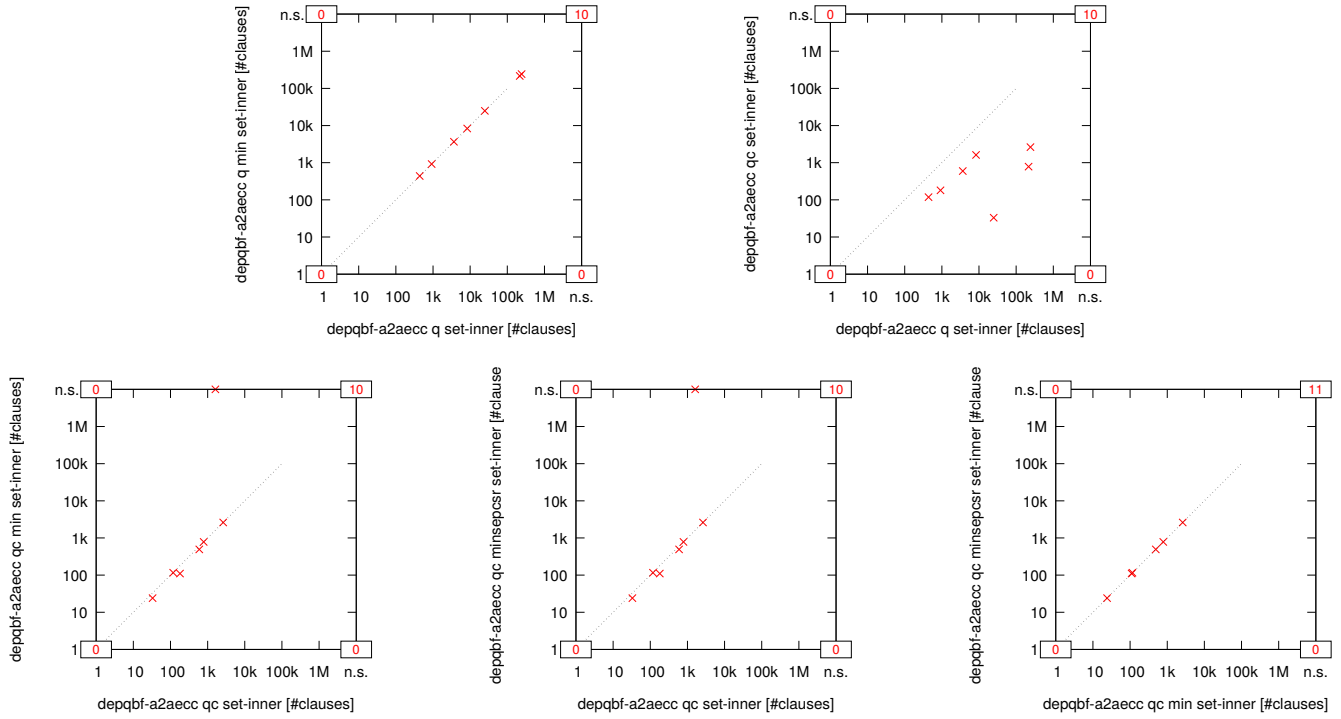


Fig. 324: Suite Tentrup ($n = 17$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

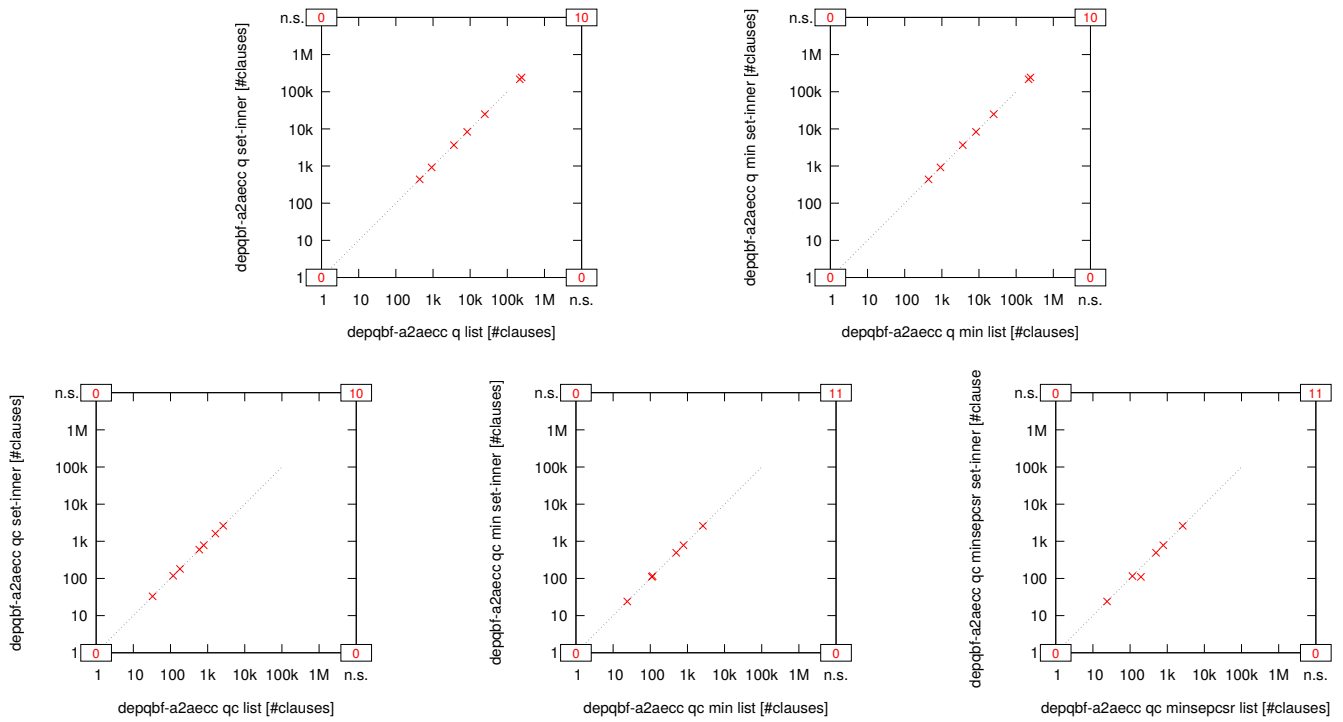


Fig. 325: Suite Tentrup ($n = 17$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

46) Wintersteiger ($n = 38$):

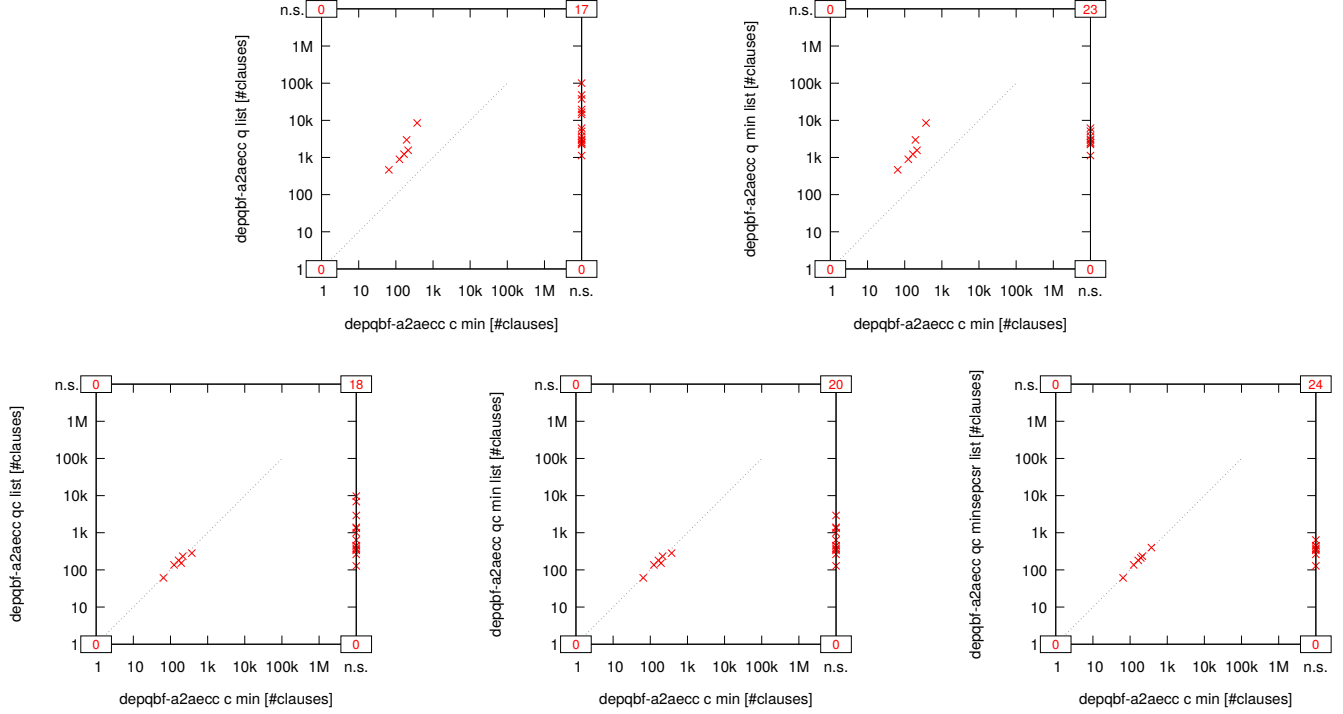


Fig. 326: Suite Wintersteiger ($n = 38$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus mode c min (number of clauses).

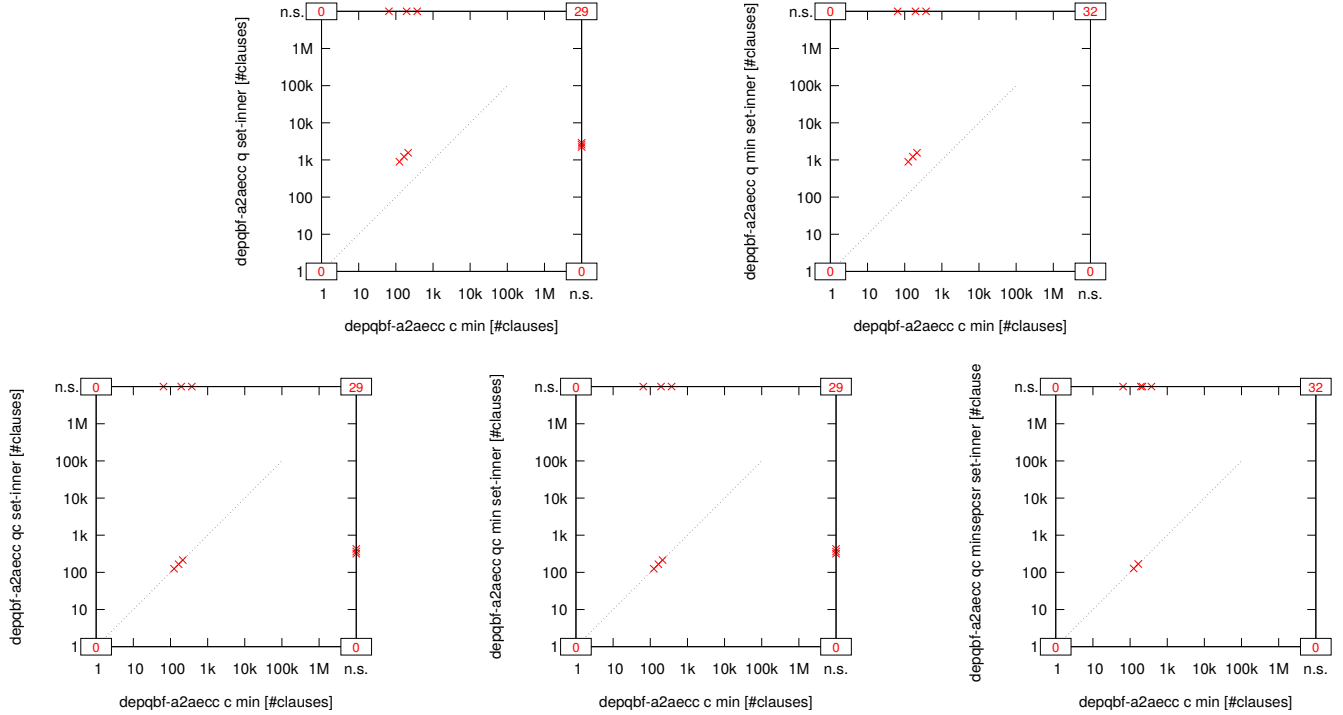


Fig. 327: Suite Wintersteiger ($n = 38$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus mode c min (number of clauses).

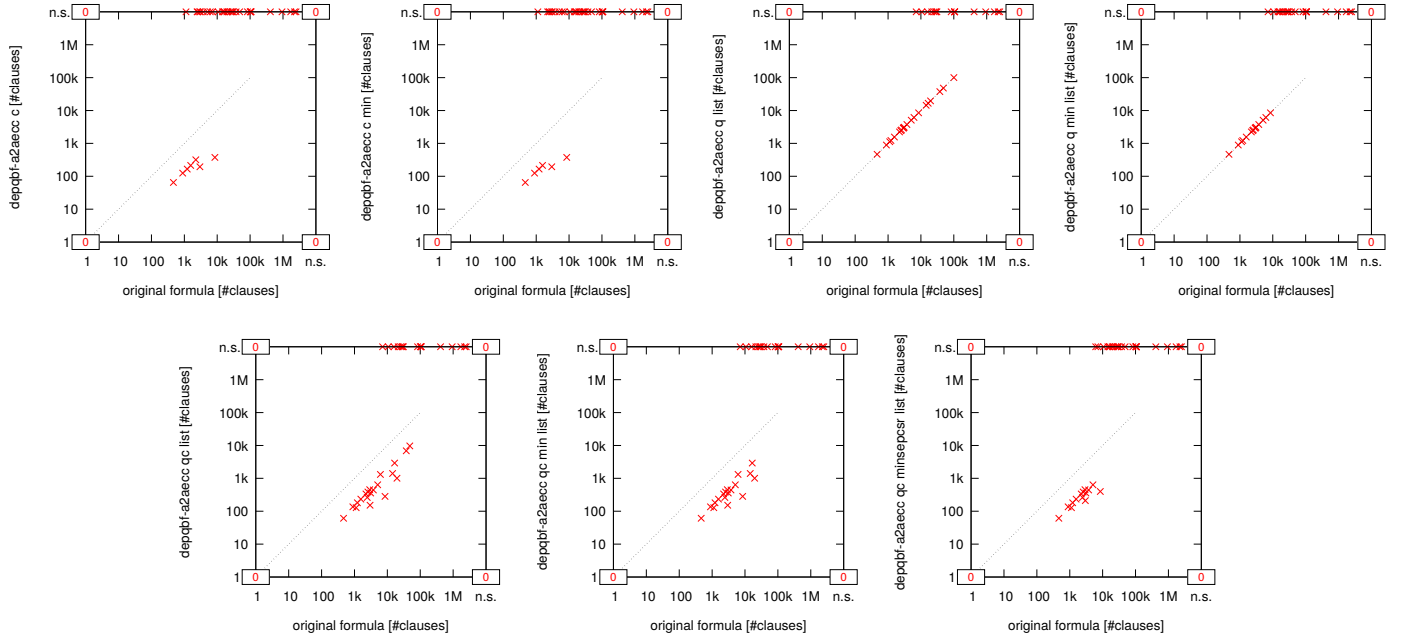


Fig. 328: Suite Wintersteiger ($n = 38$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in list semantics versus original formula (number of clauses).

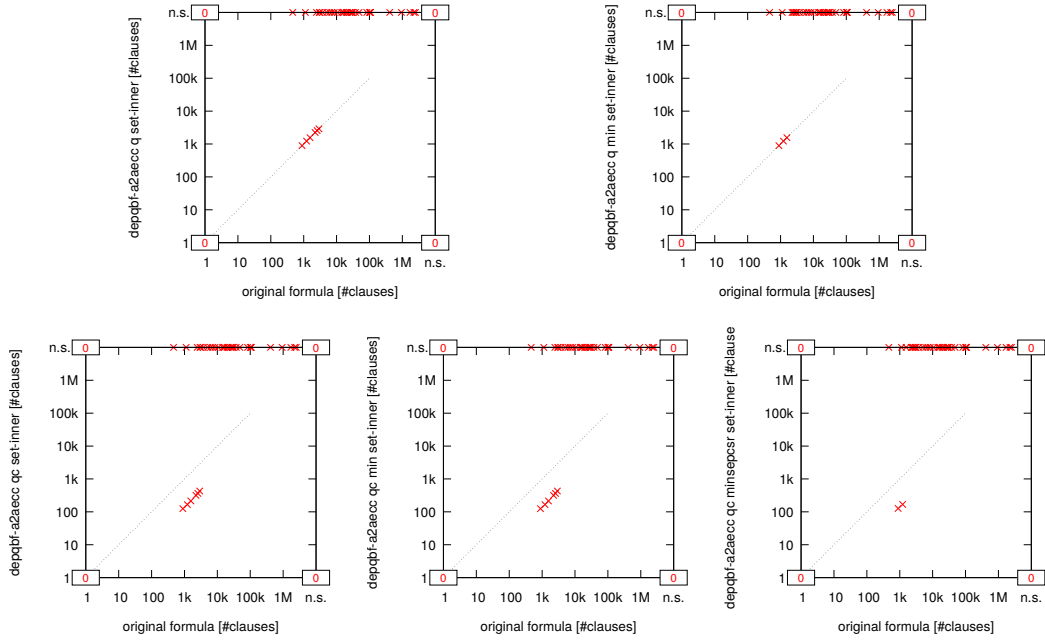


Fig. 329: Suite Wintersteiger ($n = 38$): Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in set-inner semantics versus original formula (number of clauses).

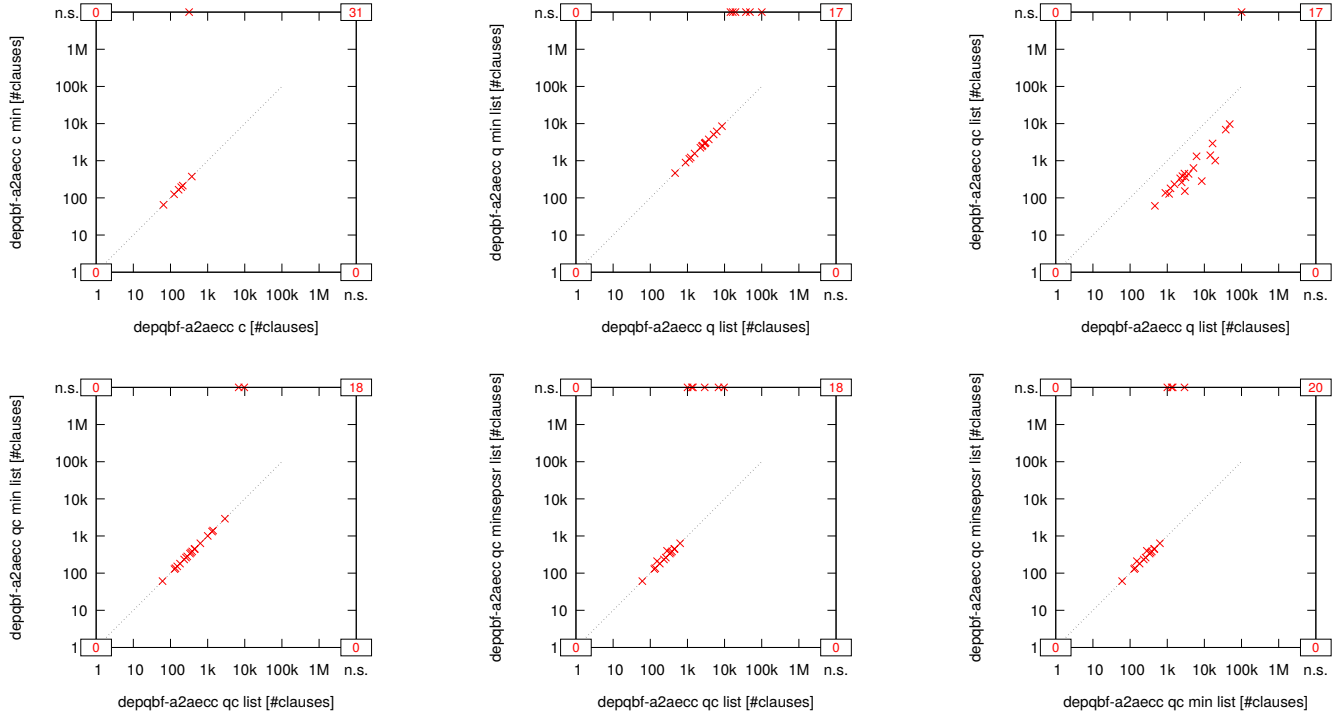


Fig. 330: Suite Wintersteiger ($n = 38$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list semantics (number of clauses).

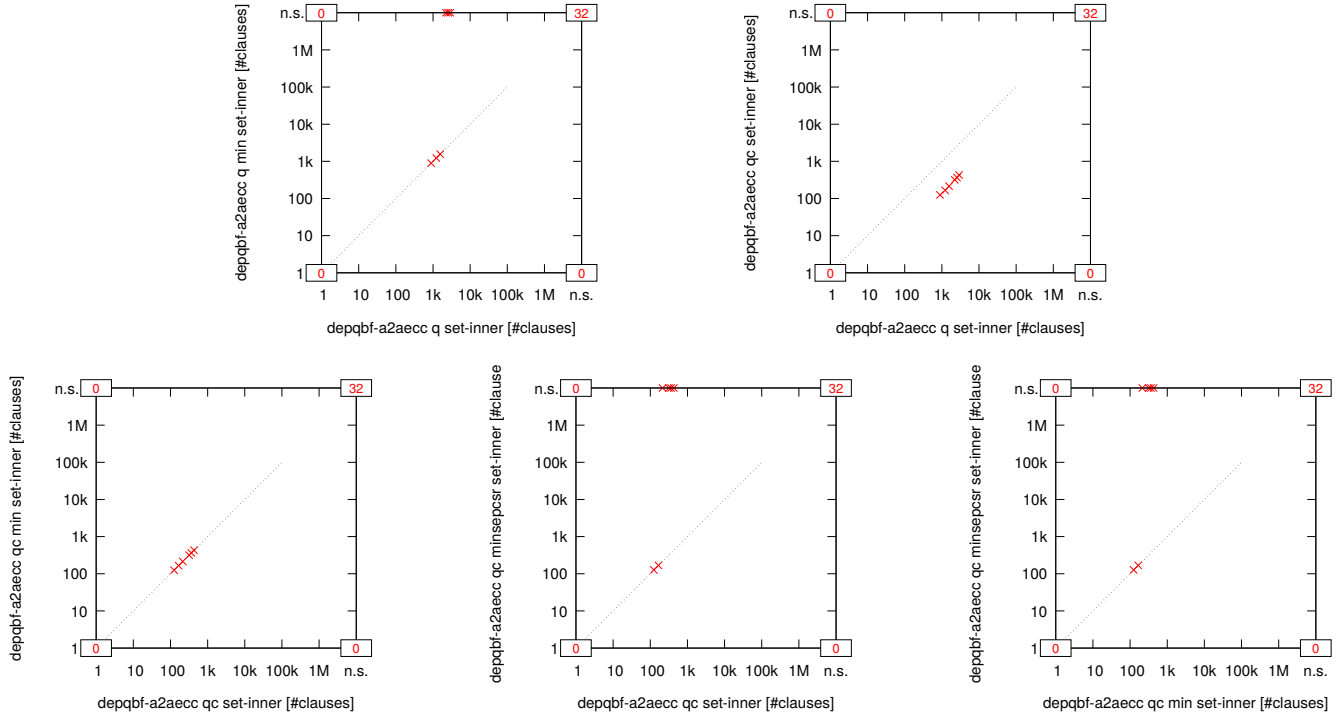


Fig. 331: Suite Wintersteiger ($n = 38$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in set-inner semantics (number of clauses).

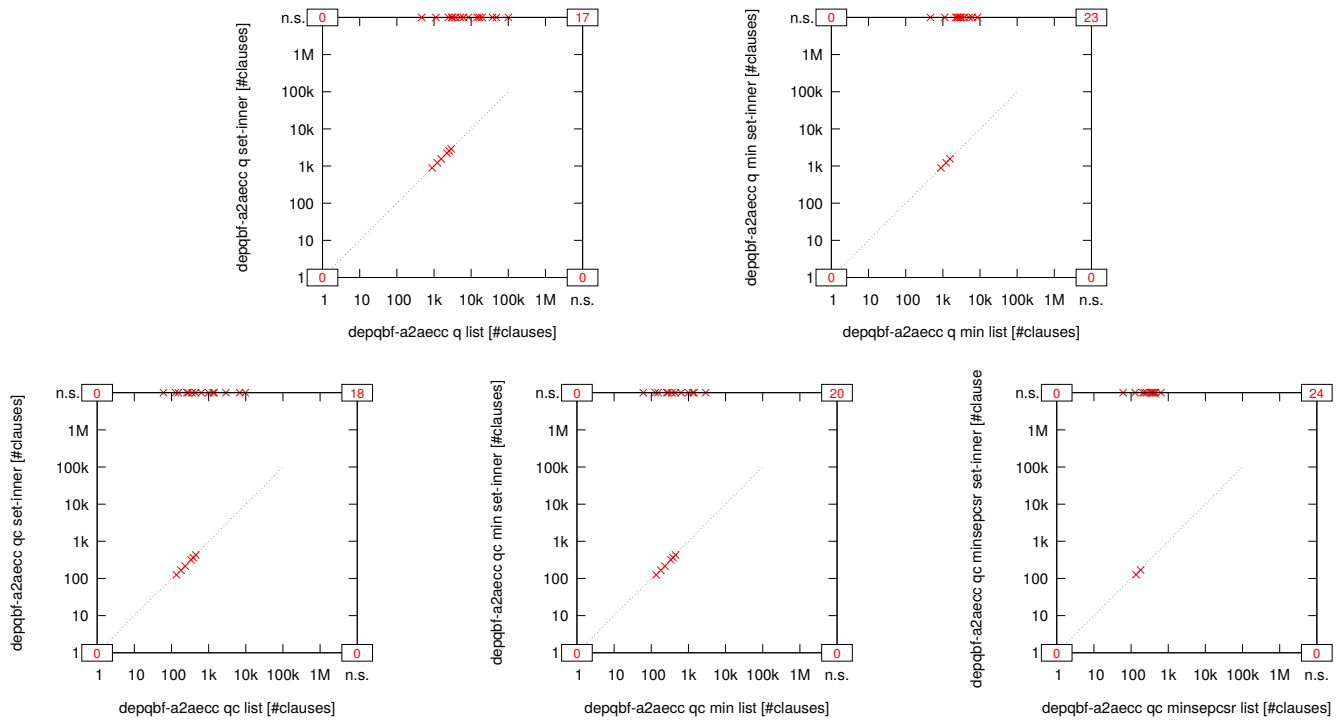


Fig. 332: Suite Wintersteiger ($n = 38$): Comparing sizes of different unsatisfiable cores in DepQBF-a2aecc in list versus set-inner semantics (number of clauses).

B. Partitioned by Number of \forall Quantified Variables

This subsection shows figures of the sizes of unsatisfiable cores with subfigures for partitions of the benchmarks according to the number of \forall quantified variables.

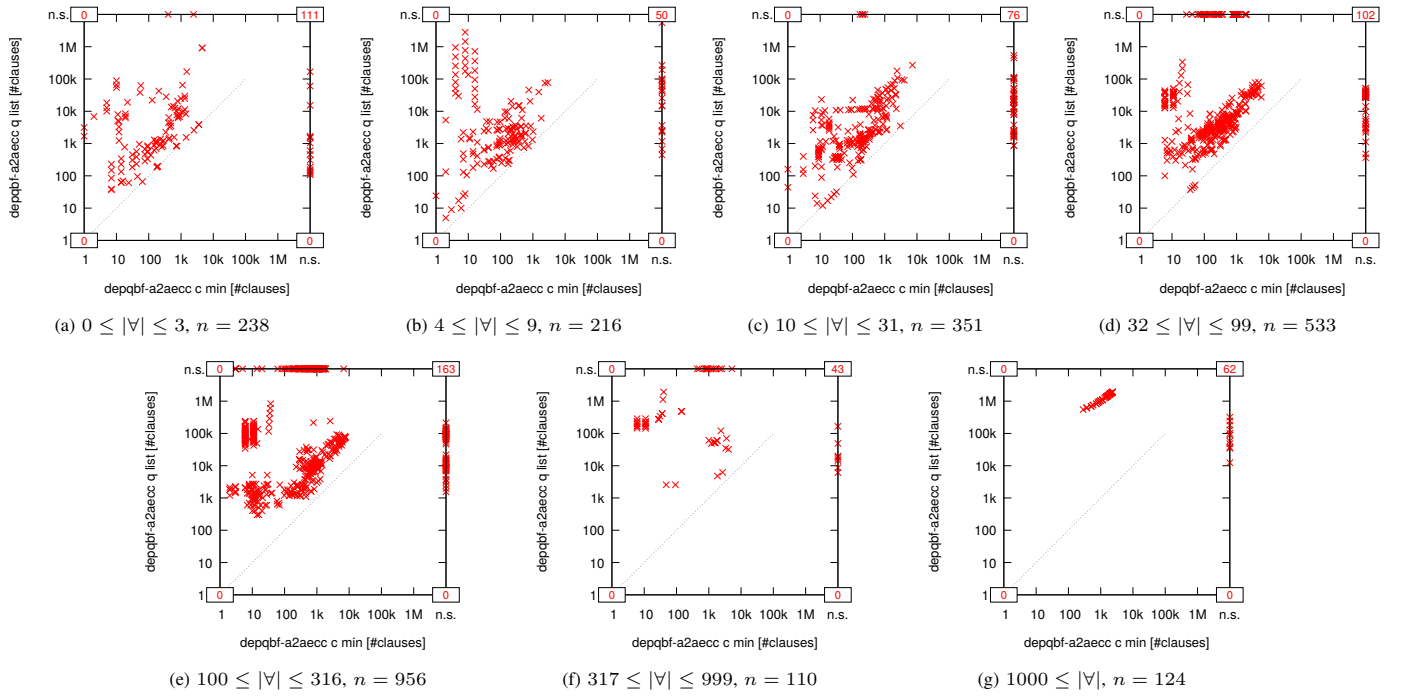


Fig. 333: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q list partitioned by number of \forall quantified variables (number of clauses).

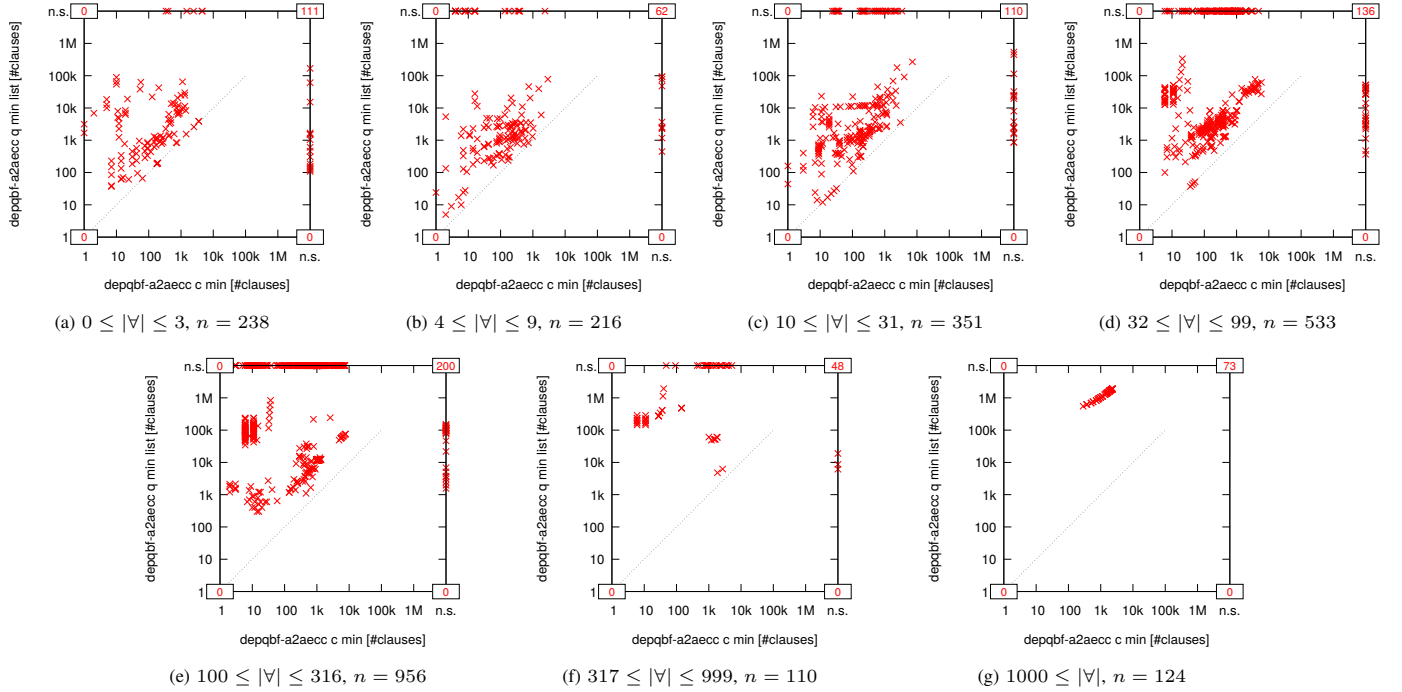


Fig. 334: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q min list partitioned by number of \forall quantified variables (number of clauses).

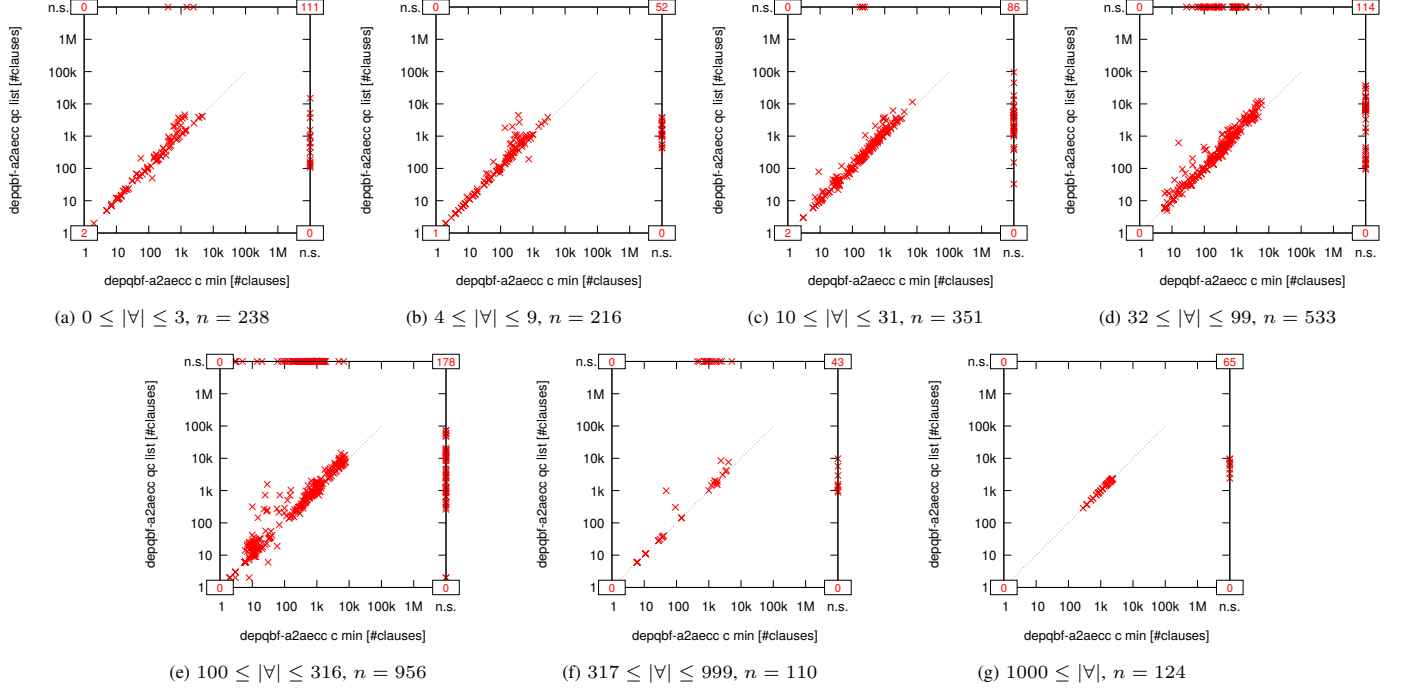


Fig. 335: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc list partitioned by number of \forall quantified variables (number of clauses).

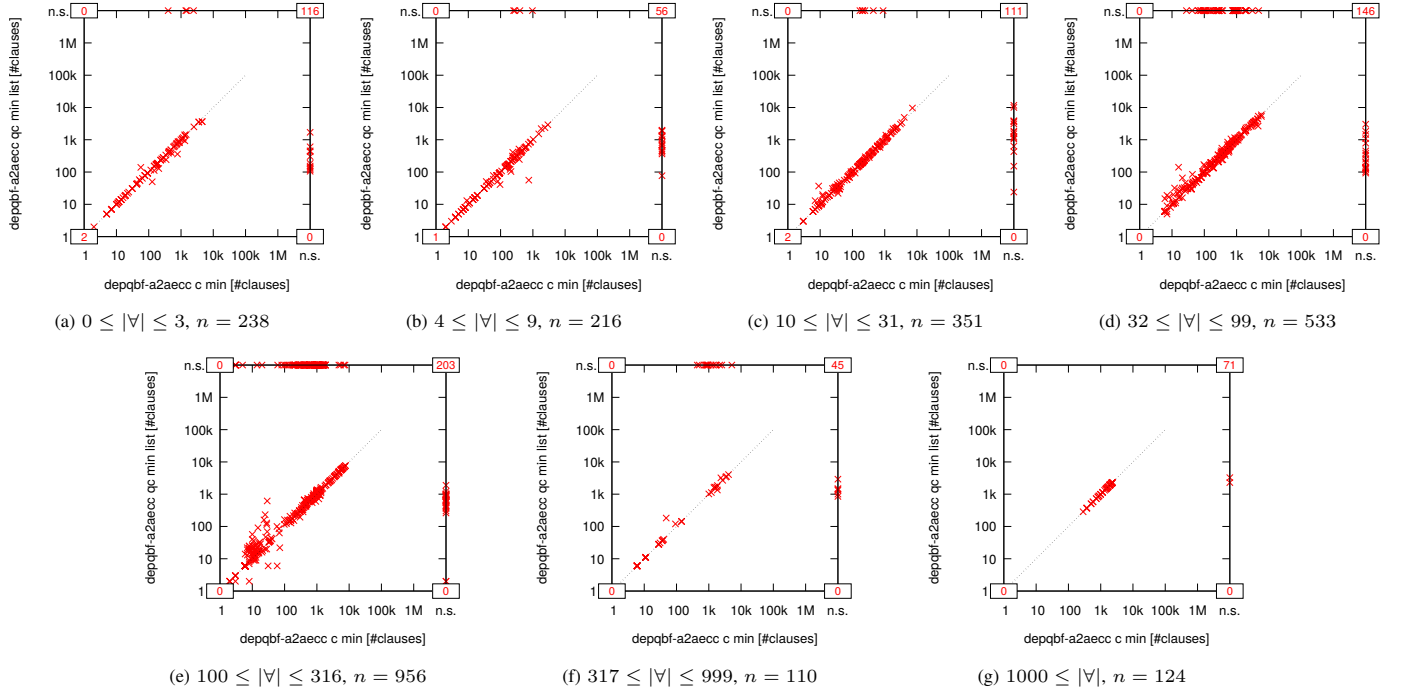


Fig. 336: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc min list partitioned by number of \forall quantified variables (number of clauses).

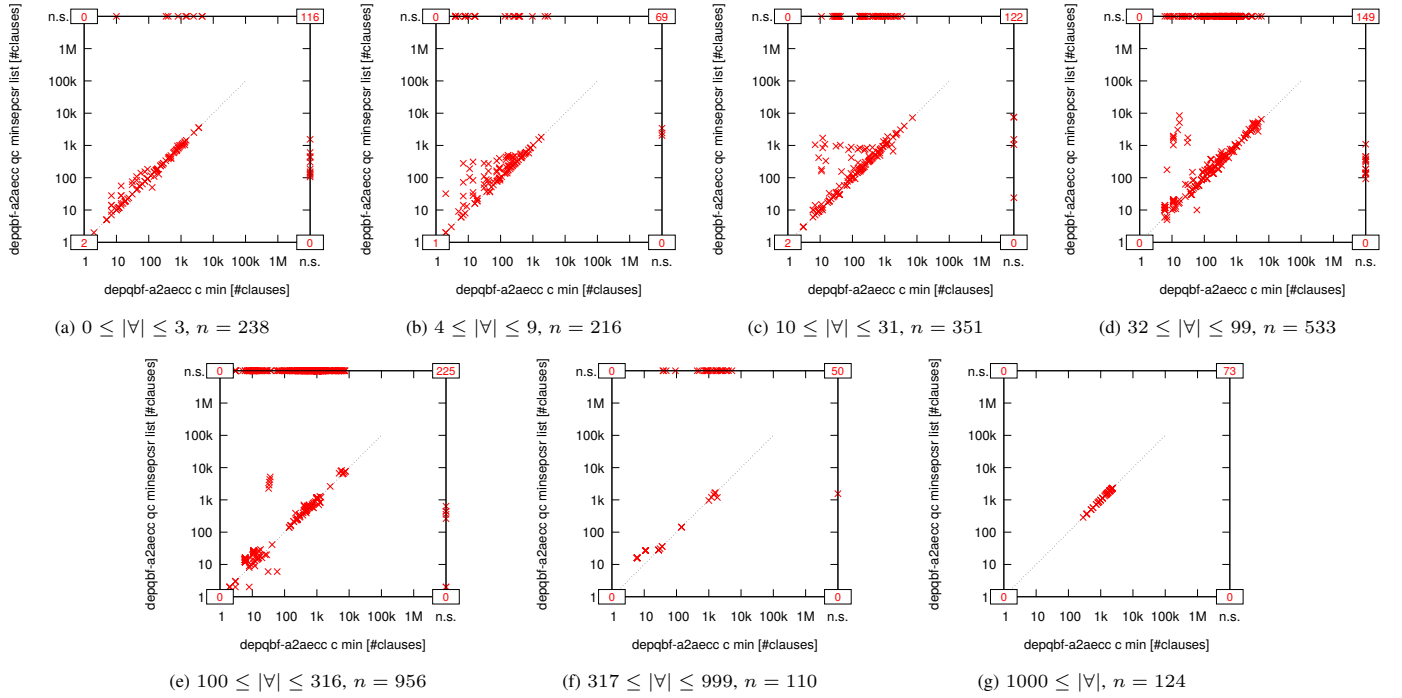


Fig. 337: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc minsepcsr list partitioned by number of \forall quantified variables (number of clauses).

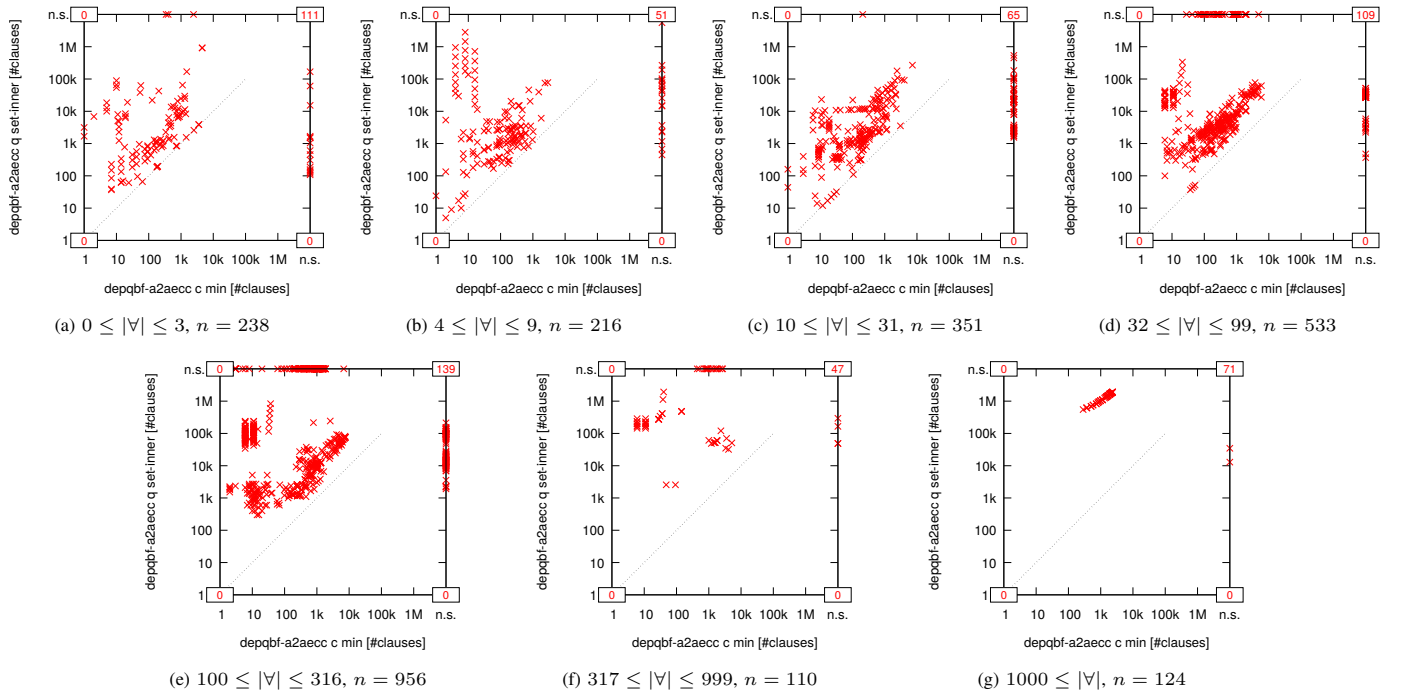


Fig. 338: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q set-inner partitioned by number of \forall quantified variables (number of clauses).

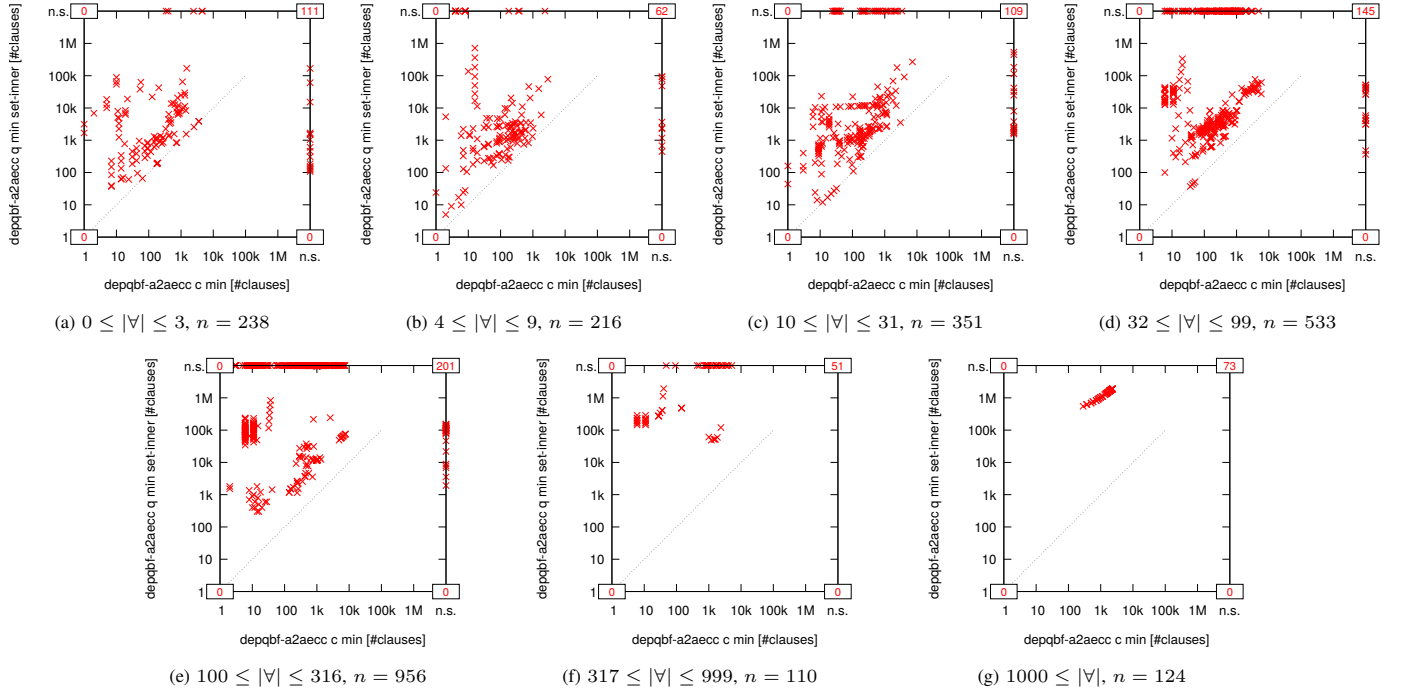


Fig. 339: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q min set-inner partitioned by number of \forall quantified variables (number of clauses).

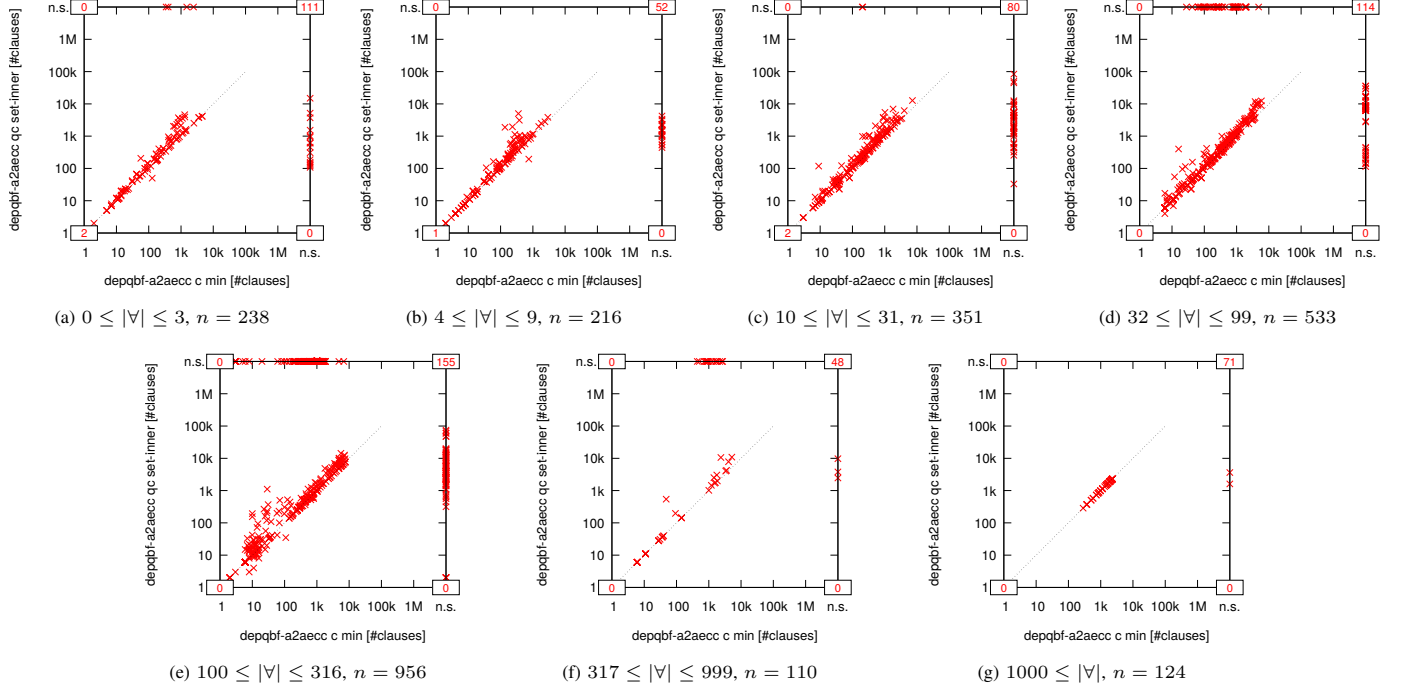


Fig. 340: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc set-inner partitioned by number of \forall quantified variables (number of clauses).

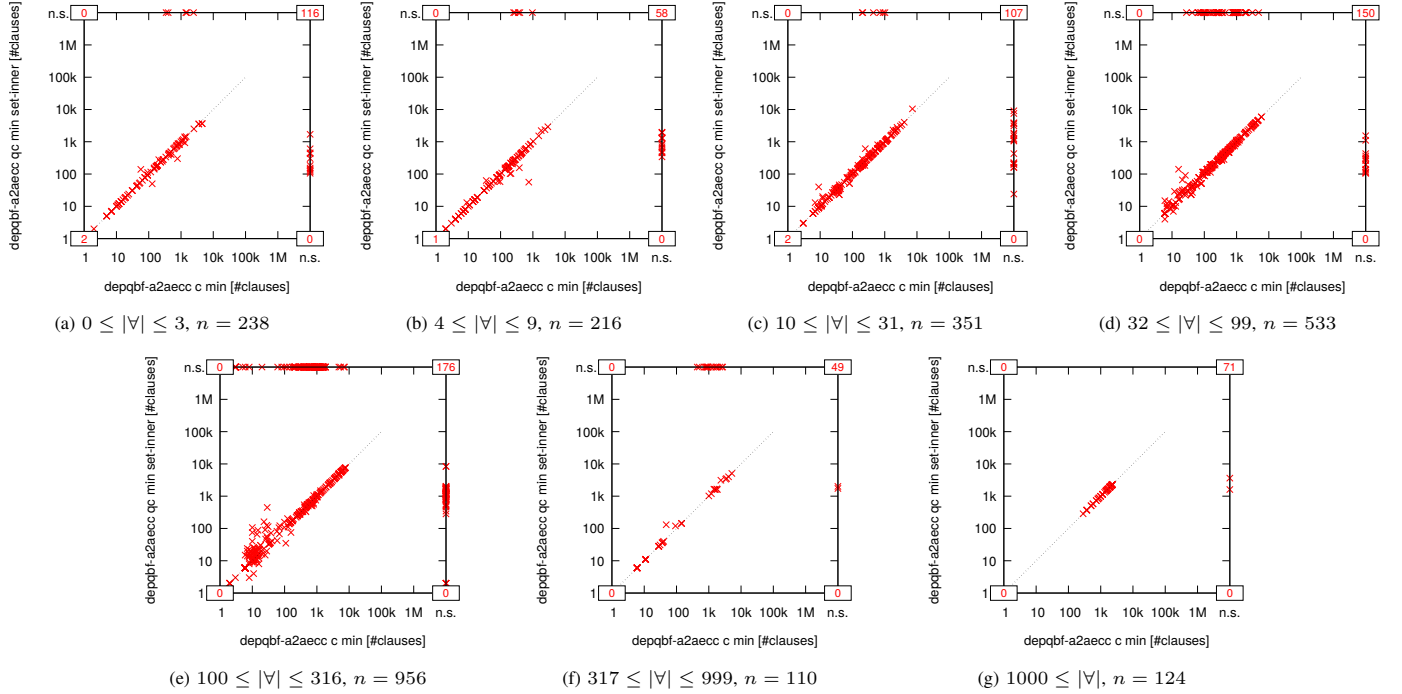


Fig. 341: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc min set-inner partitioned by number of \forall quantified variables (number of clauses).

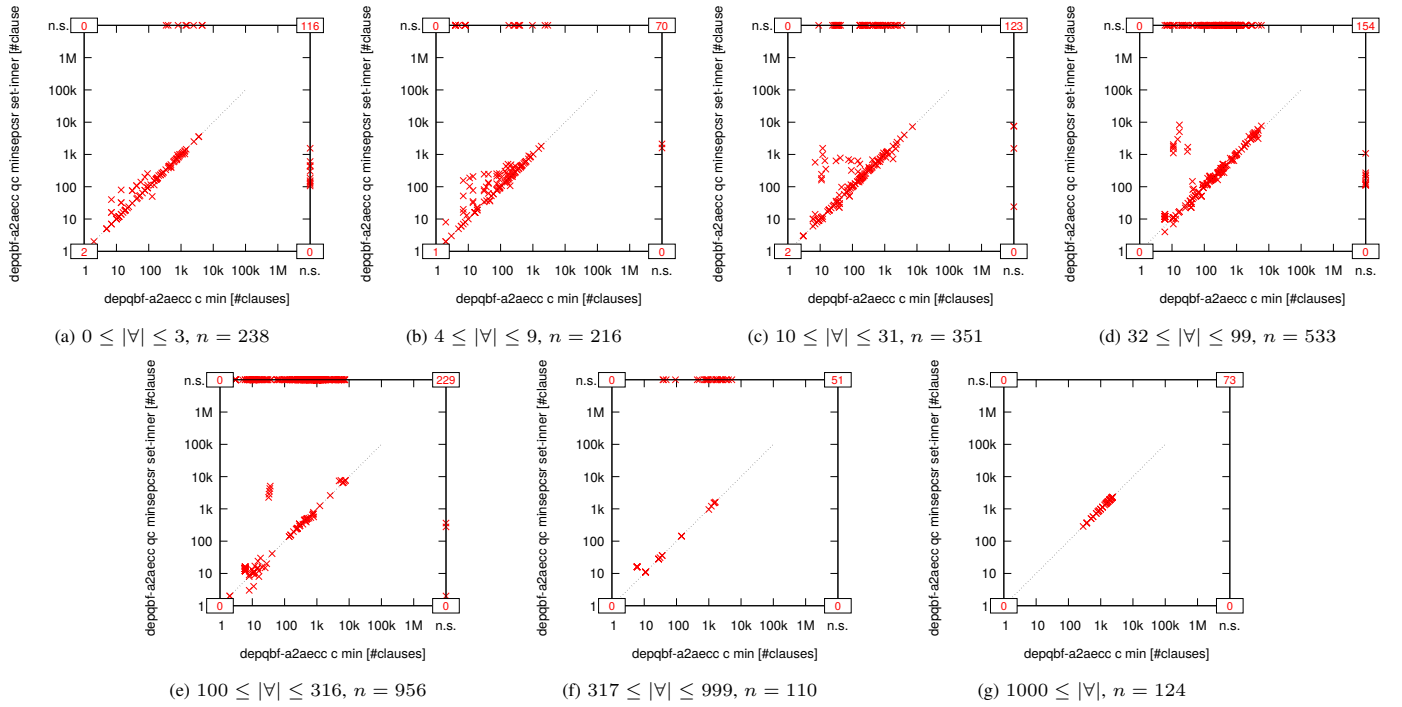


Fig. 342: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc minsepcsr set-inner partitioned by number of \forall quantified variables (number of clauses).

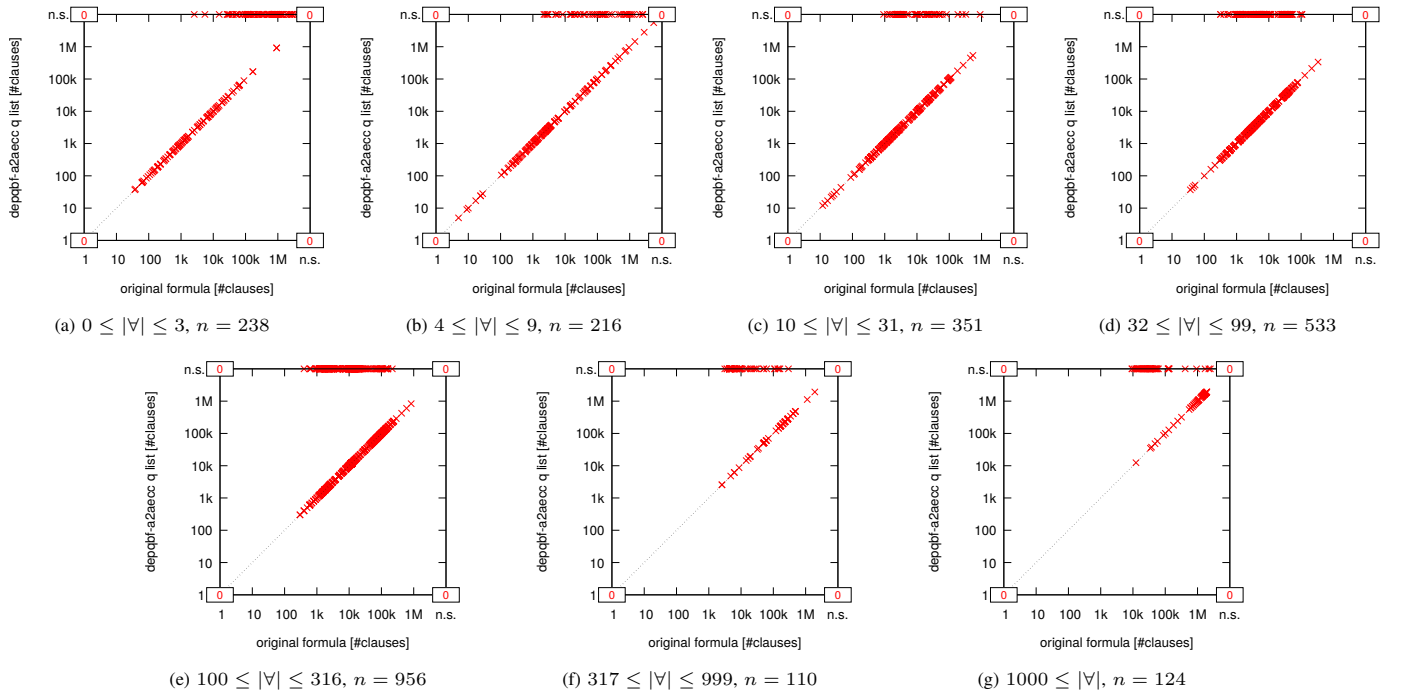


Fig. 343: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q list partitioned by number of \forall quantified variables (number of clauses).

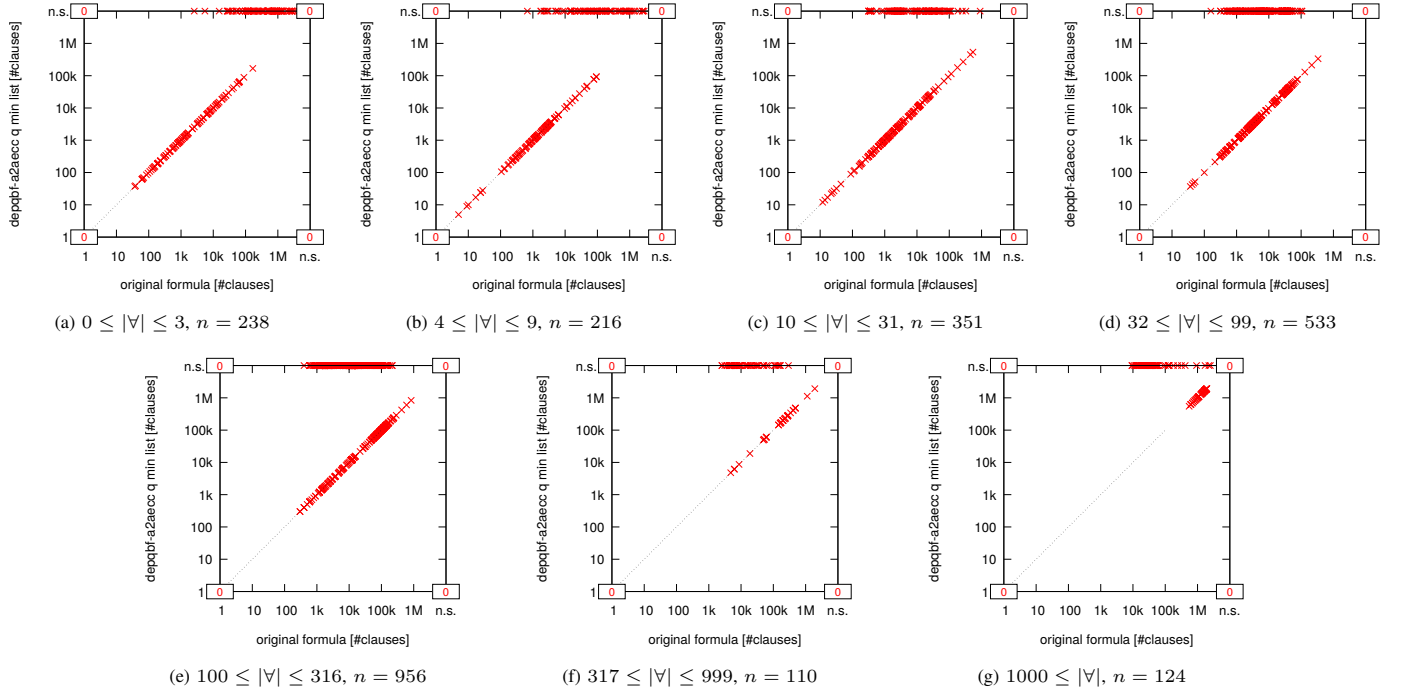


Fig. 344: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q min list partitioned by number of \forall quantified variables (number of clauses).

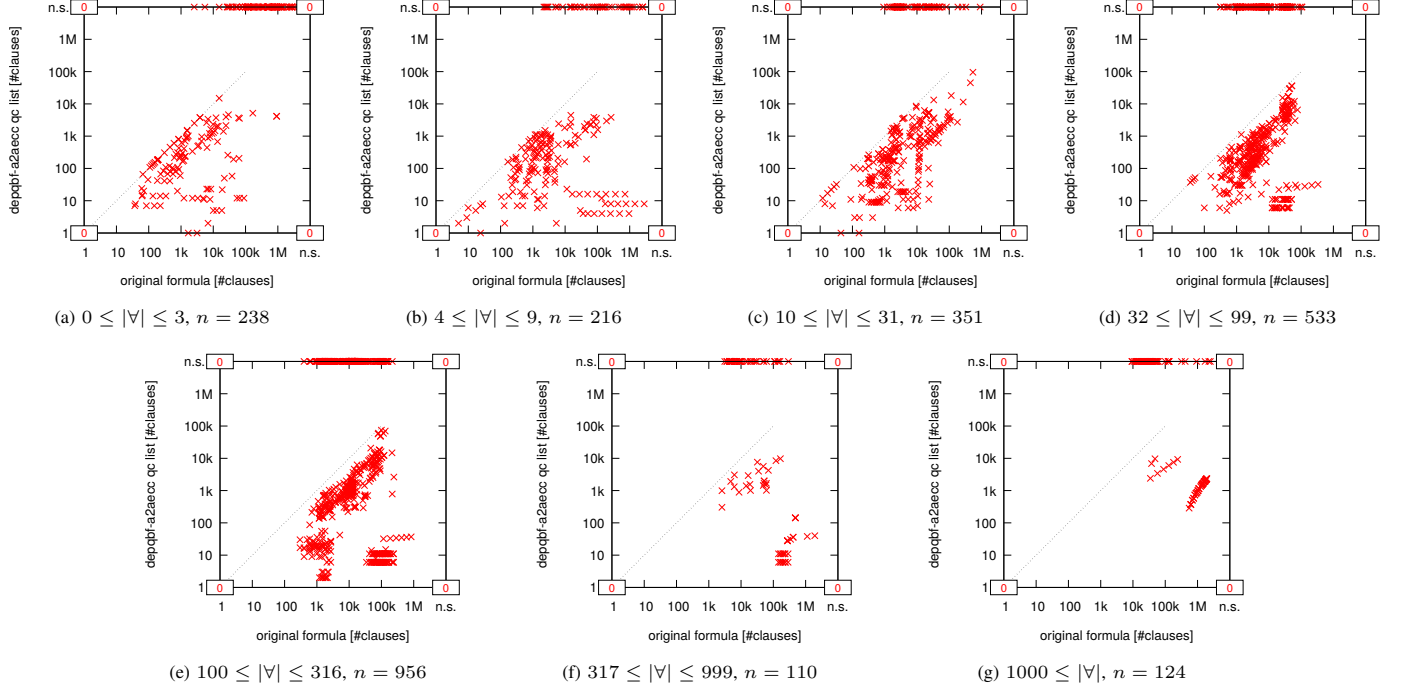


Fig. 345: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc list partitioned by number of \forall quantified variables (number of clauses).

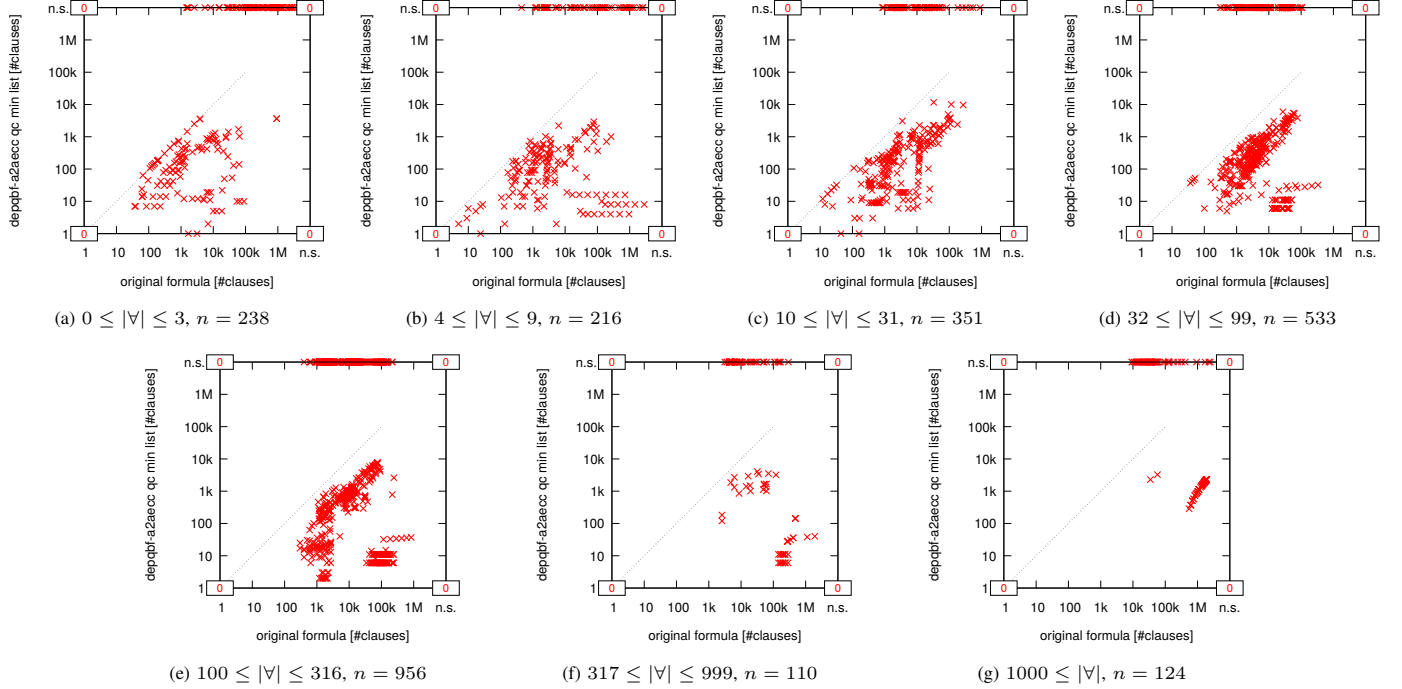


Fig. 346: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc min list partitioned by number of \forall quantified variables (number of clauses).

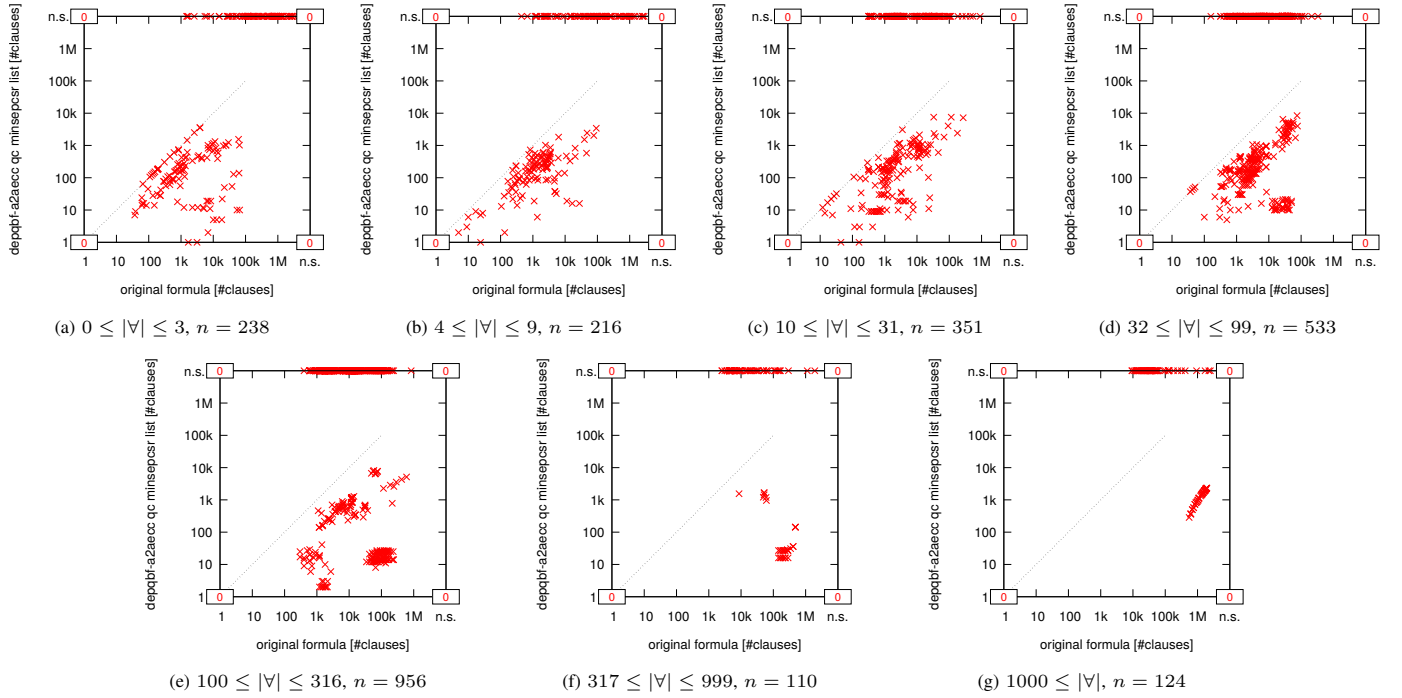


Fig. 347: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc minsepcsr list partitioned by number of \forall quantified variables (number of clauses).

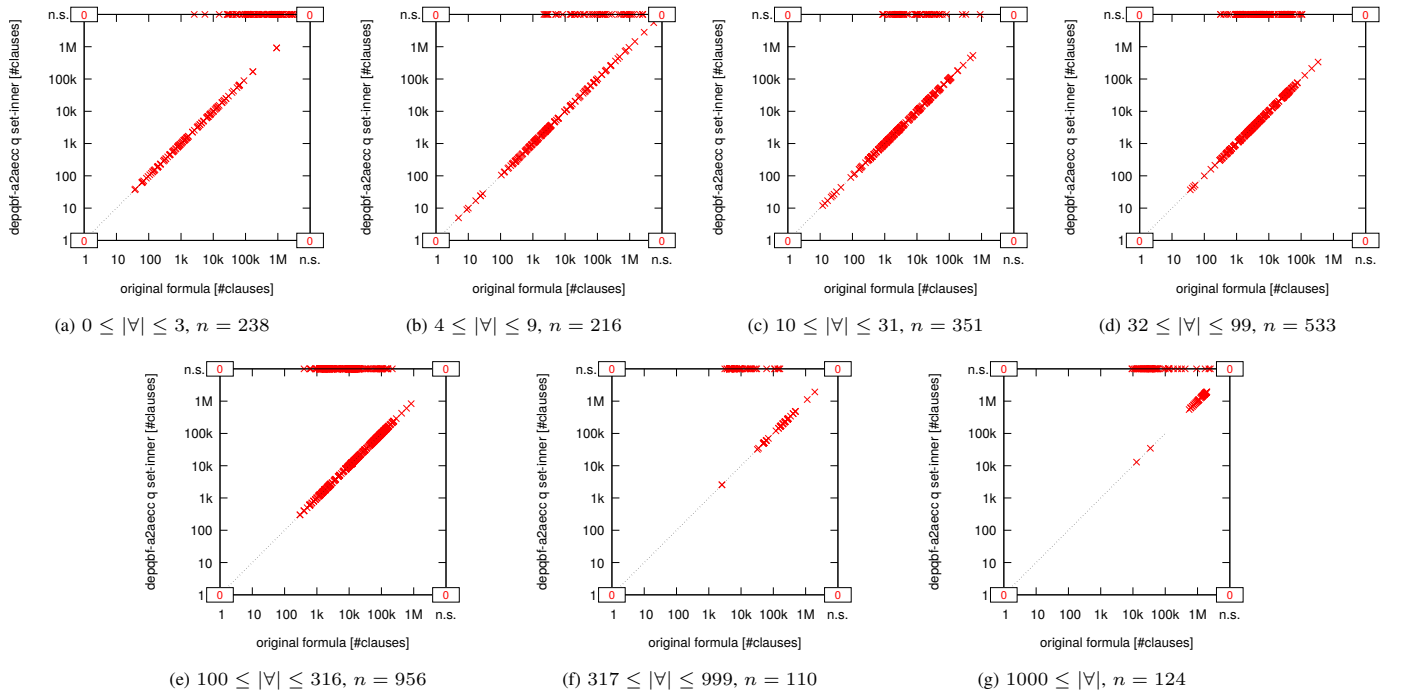


Fig. 348: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q set-inner partitioned by number of \forall quantified variables (number of clauses).

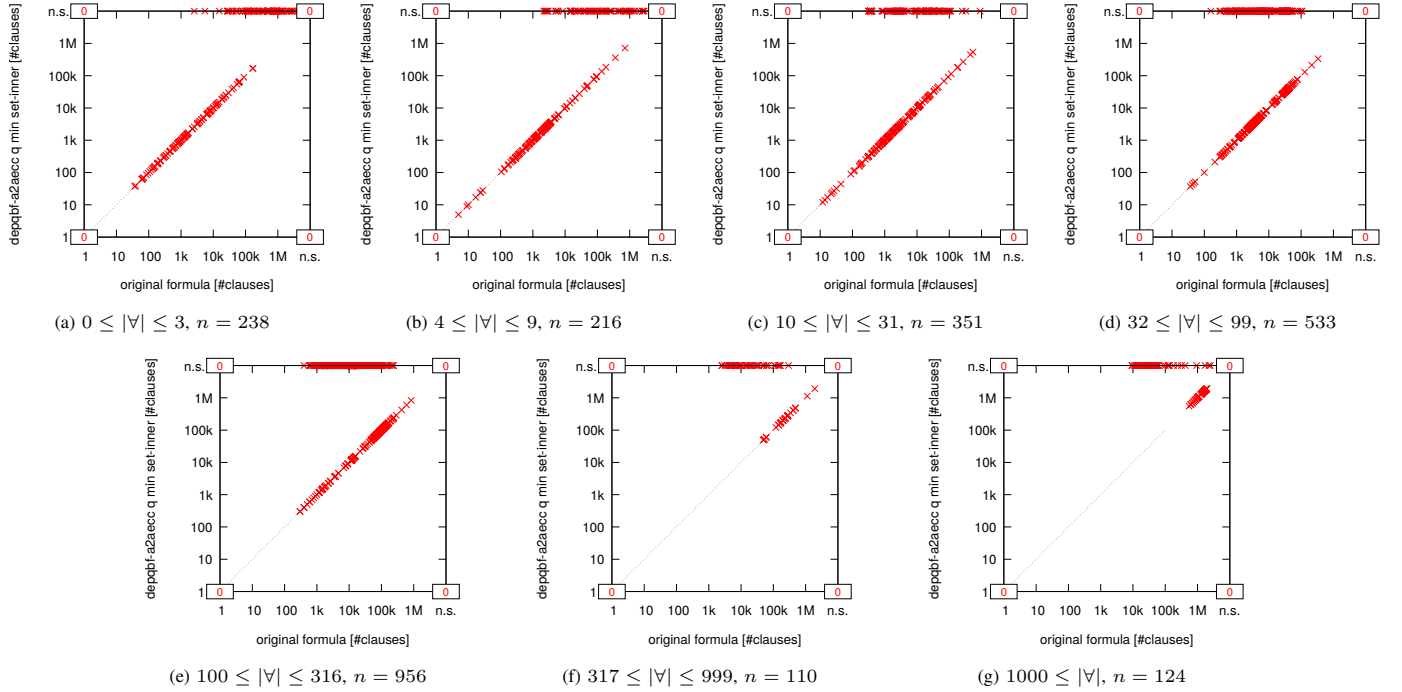


Fig. 349: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q min set-inner partitioned by number of \forall quantified variables (number of clauses).

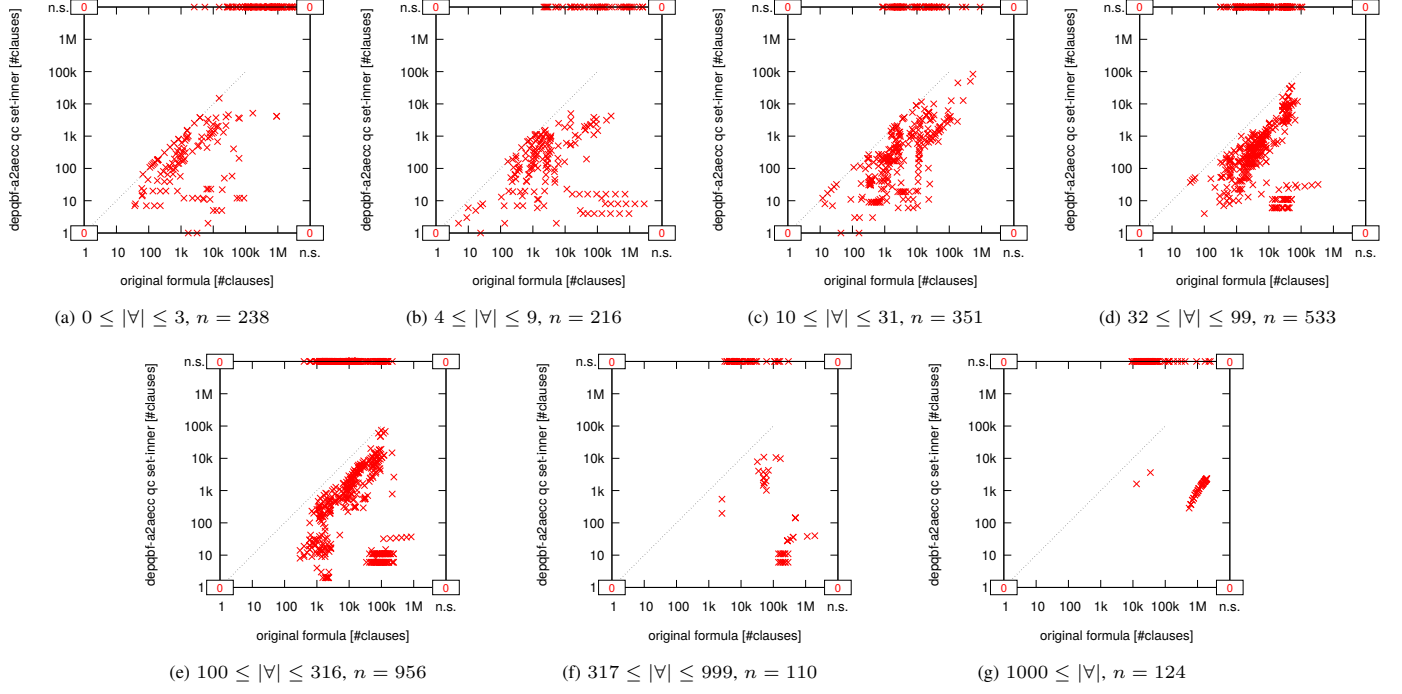


Fig. 350: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc set-inner partitioned by number of \forall quantified variables (number of clauses).

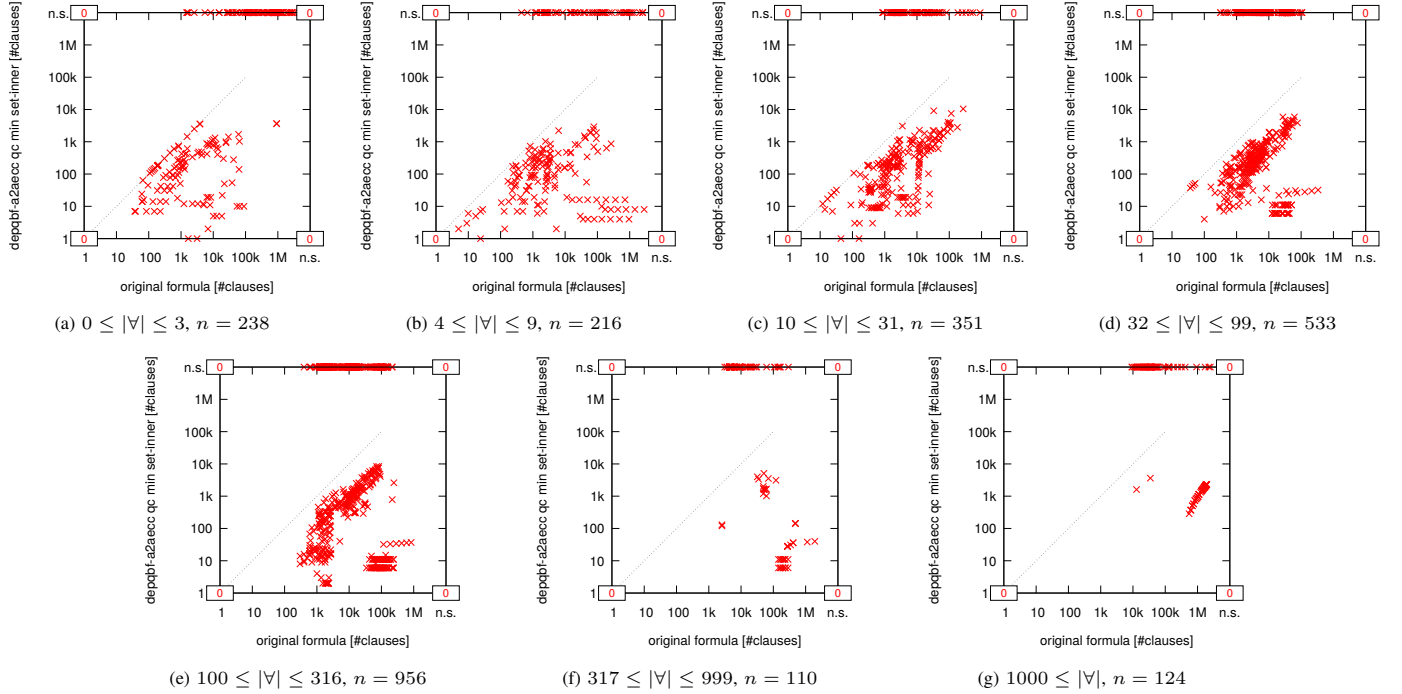


Fig. 351: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc min set-inner partitioned by number of \forall quantified variables (number of clauses).

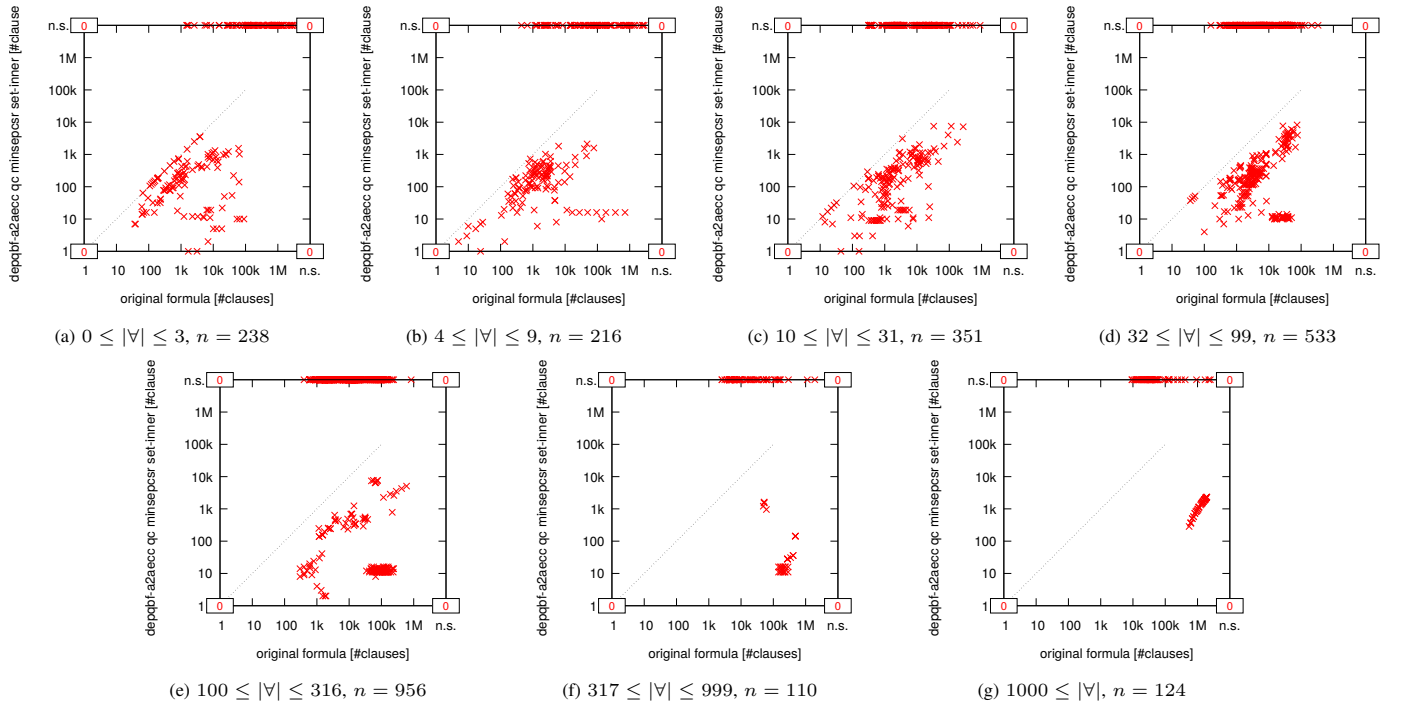


Fig. 352: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc minsepcsr set-inner partitioned by number of \forall quantified variables (number of clauses).

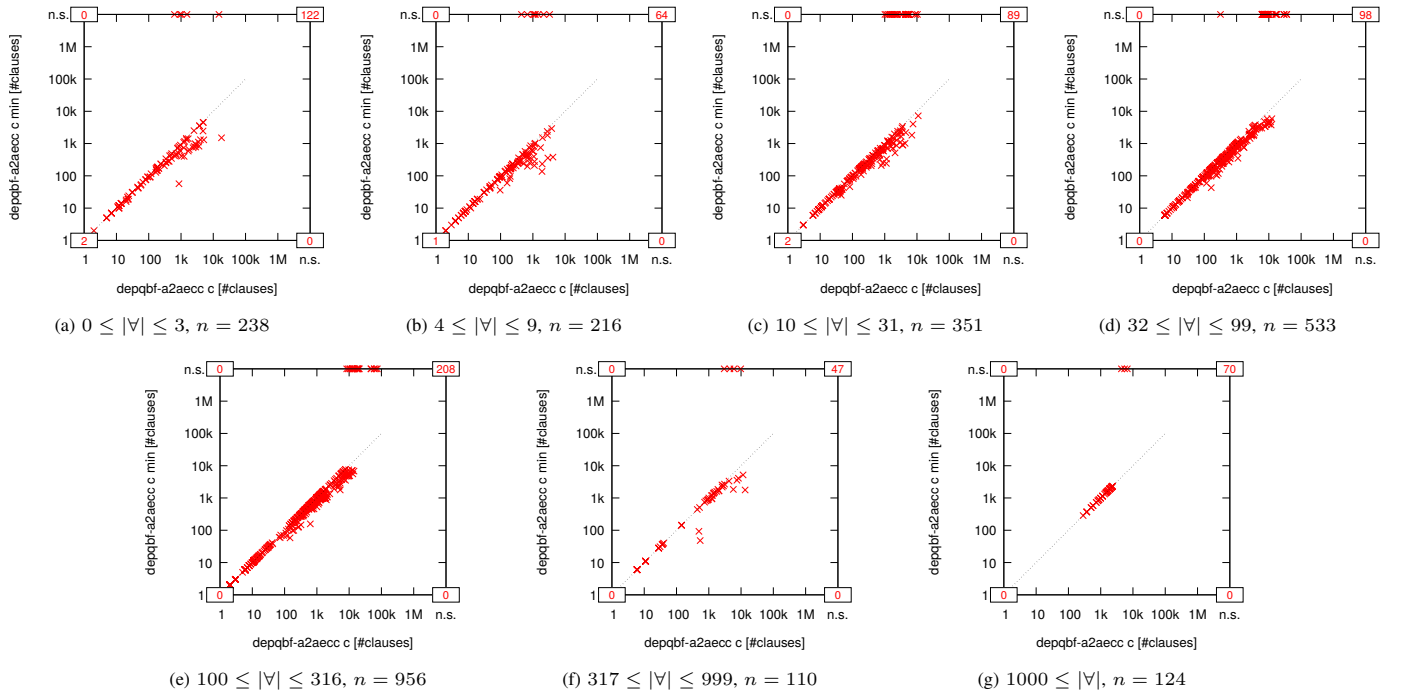


Fig. 353: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c versus mode c min partitioned by number of \forall quantified variables (number of clauses).

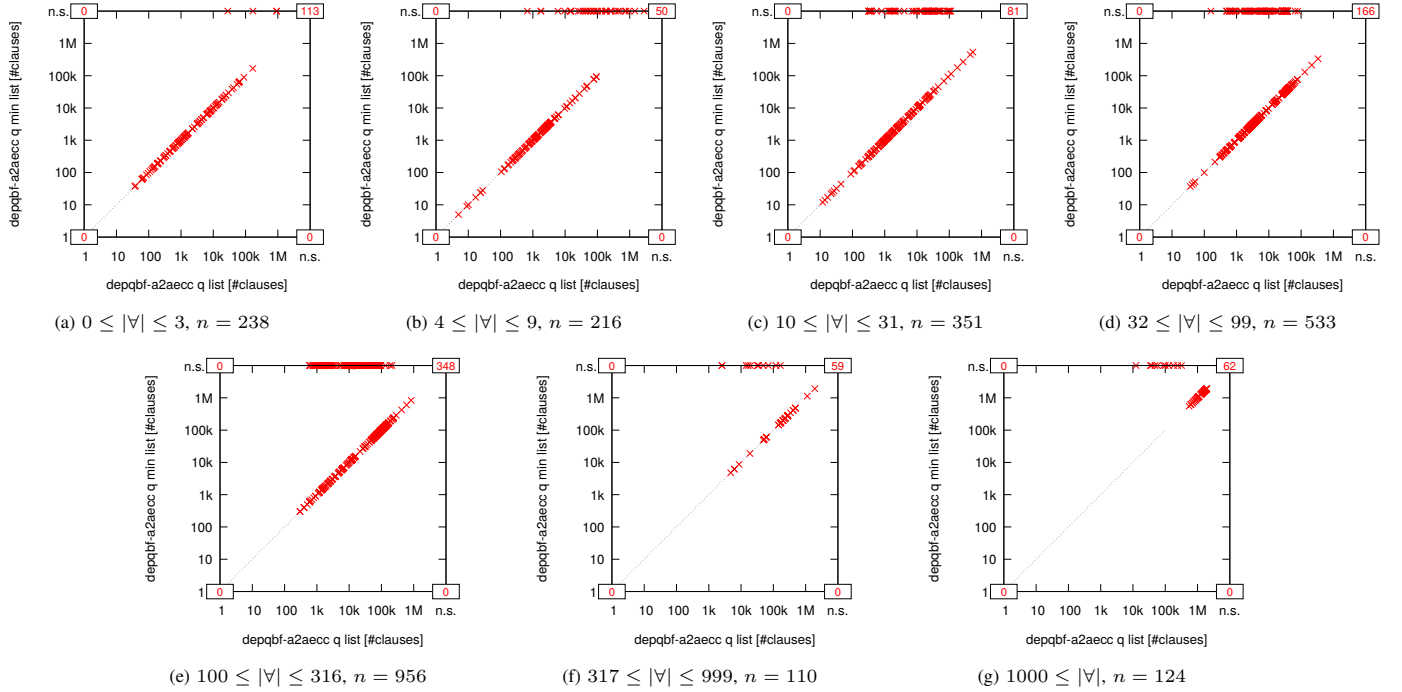


Fig. 354: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode q min list partitioned by number of \forall quantified variables (number of clauses).

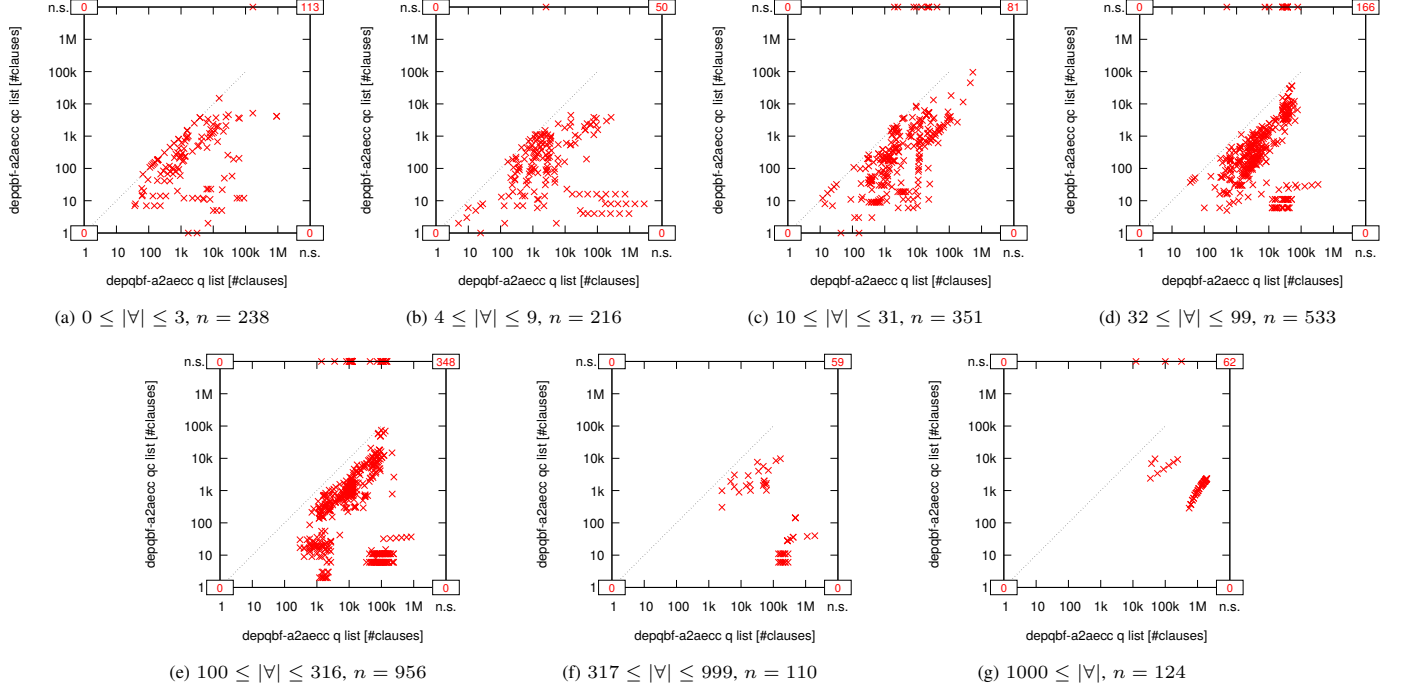


Fig. 355: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode qc list partitioned by number of \forall quantified variables (number of clauses).

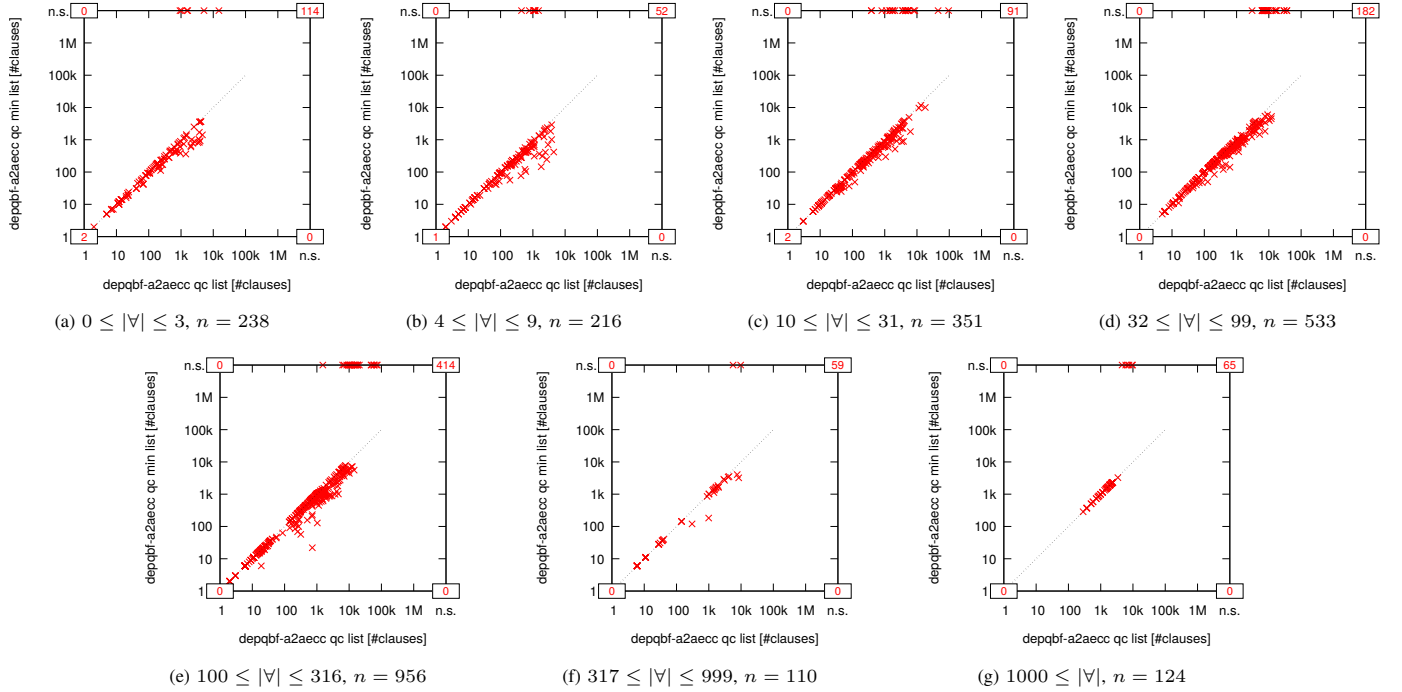


Fig. 356: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode qc min list partitioned by number of \forall quantified variables (number of clauses).

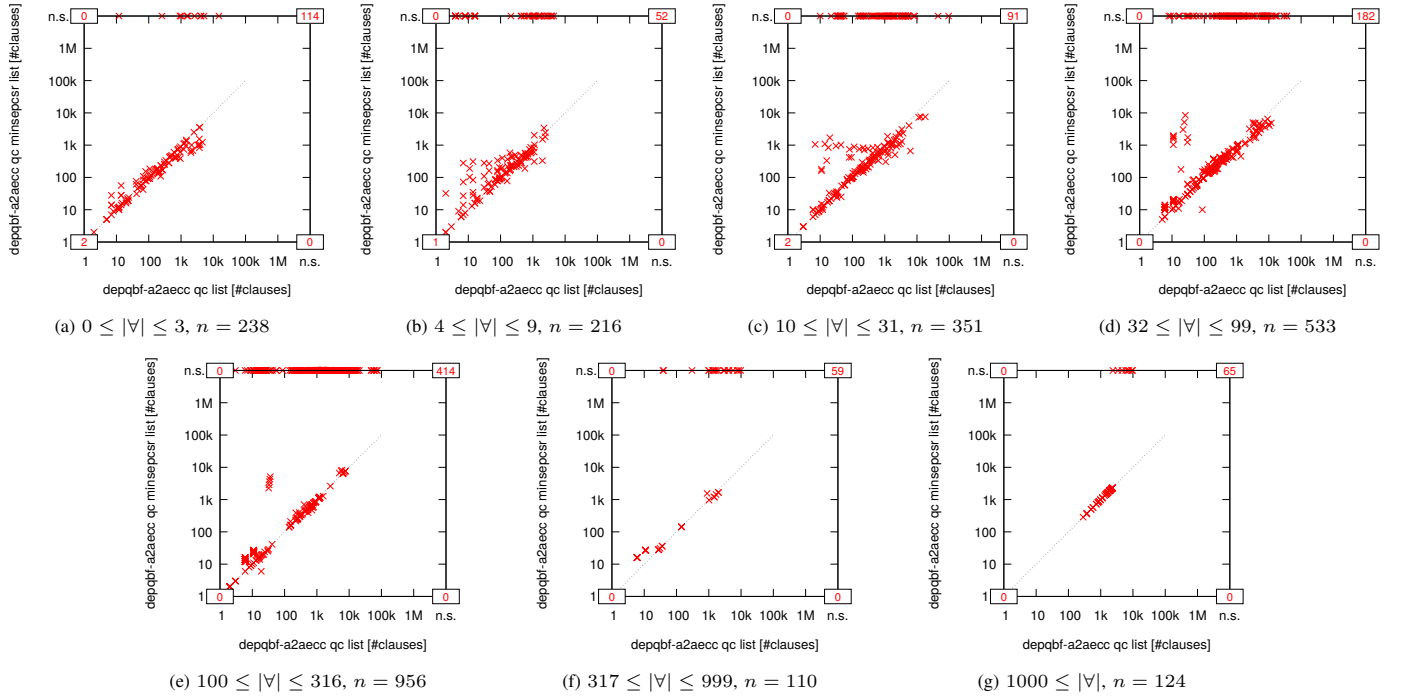


Fig. 357: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode qc minsepcsr list partitioned by number of \forall quantified variables (number of clauses).

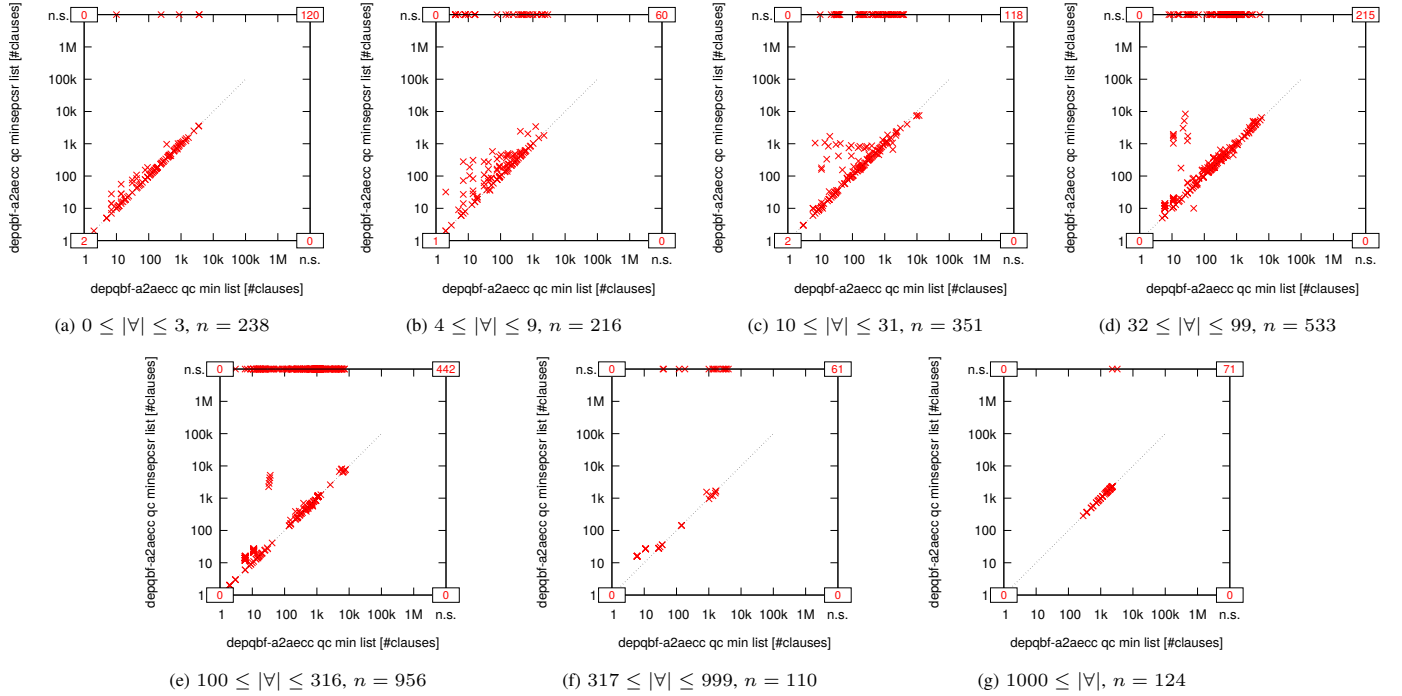


Fig. 358: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode qc minsepcsr list partitioned by number of \forall quantified variables (number of clauses).

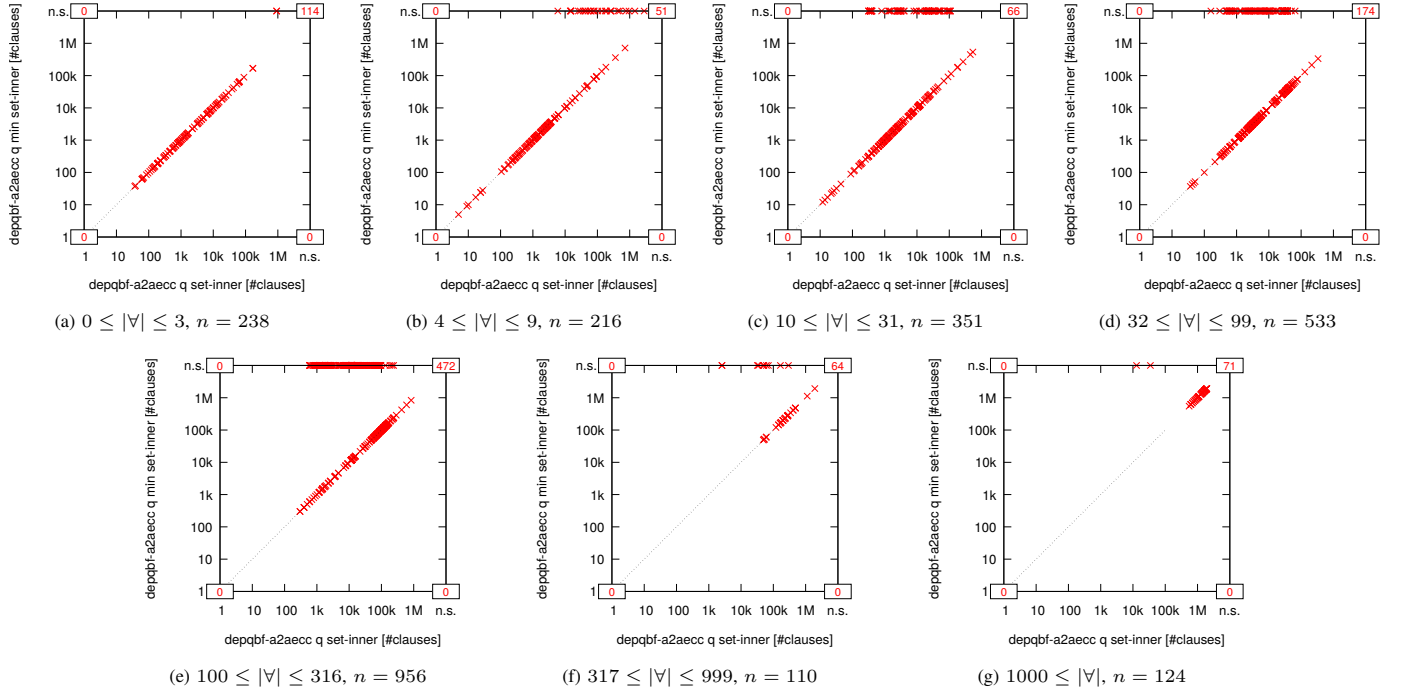


Fig. 359: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode q min set-inner partitioned by number of \forall quantified variables (number of clauses).

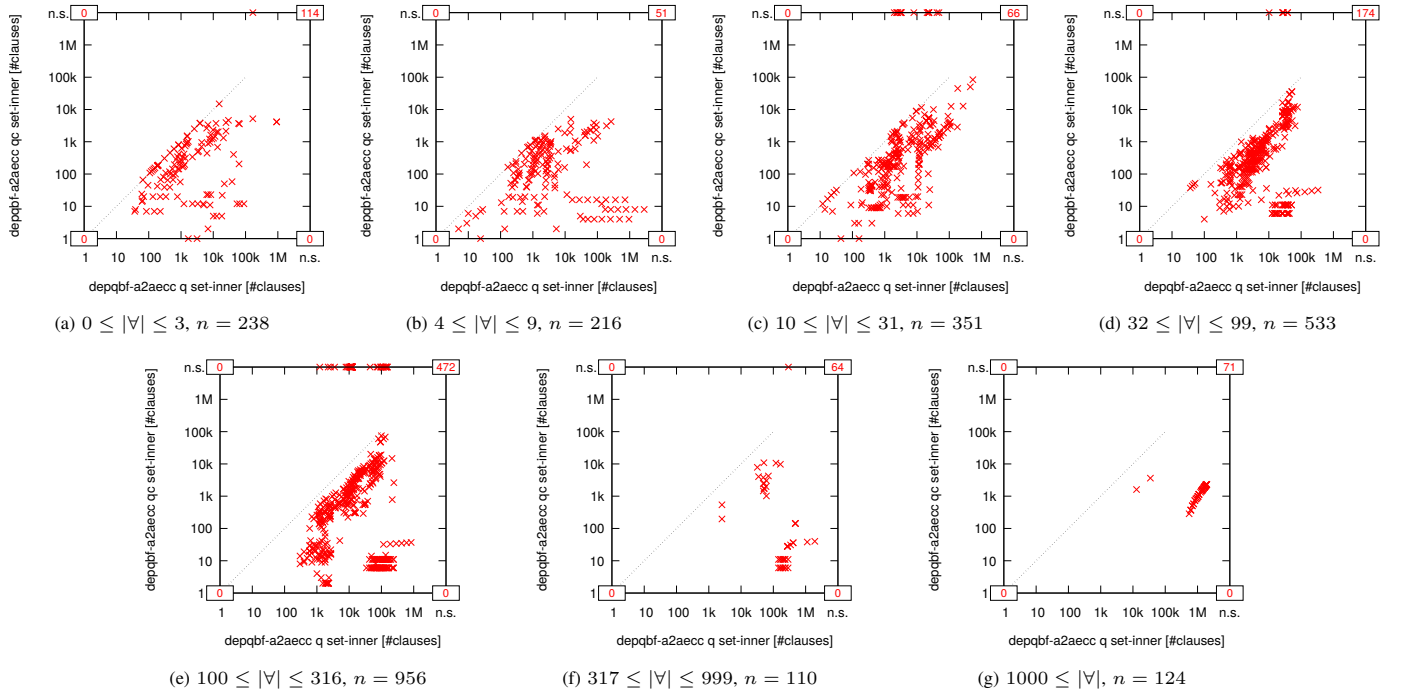


Fig. 360: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode qc set-inner partitioned by number of \forall quantified variables (number of clauses).

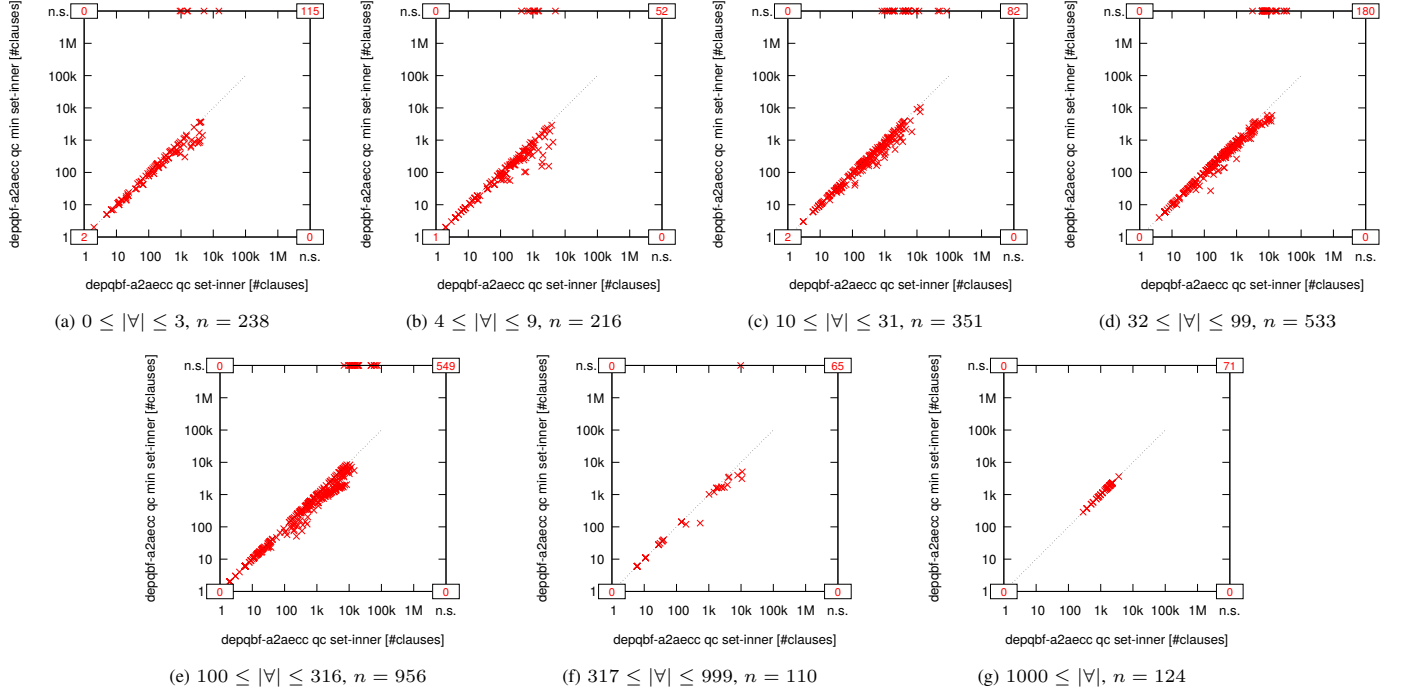


Fig. 361: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode qc min set-inner partitioned by number of \forall quantified variables (number of clauses).

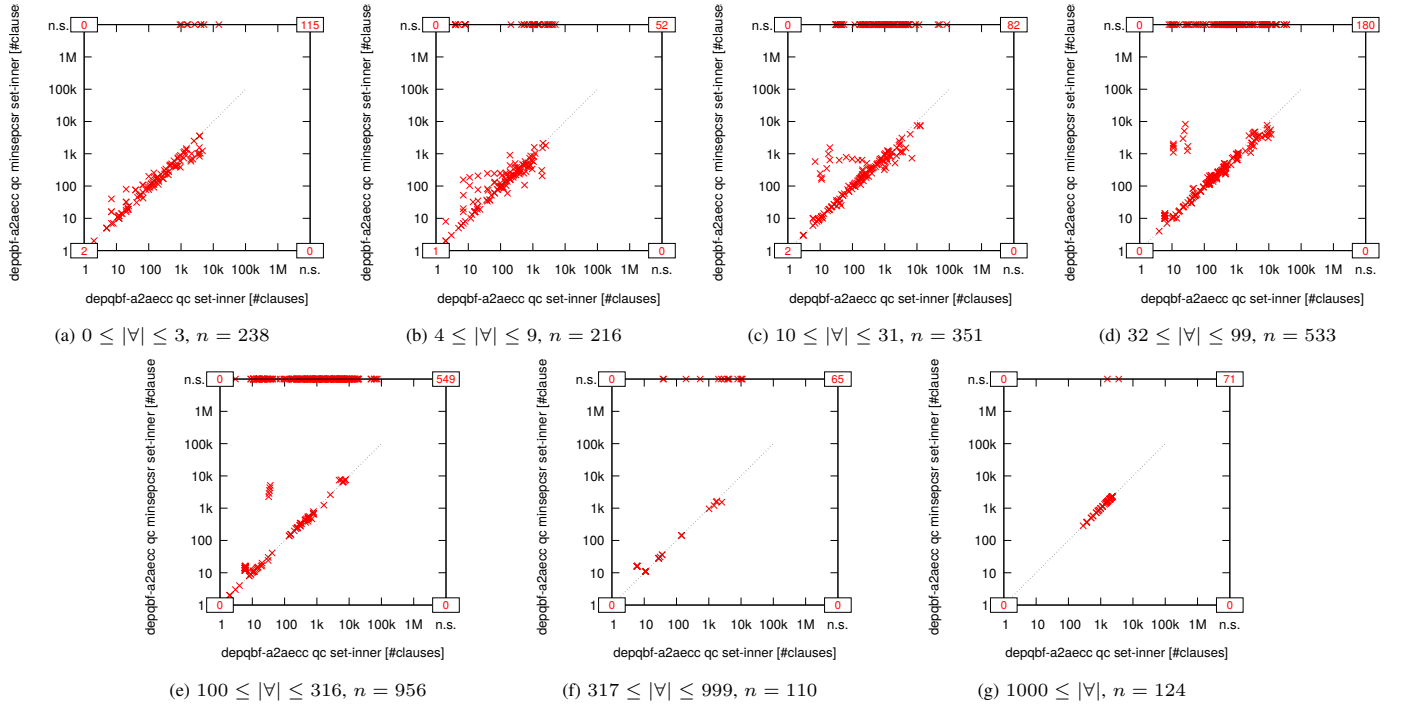


Fig. 362: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode qc minsepcsr set-inner partitioned by number of \forall quantified variables (number of clauses).

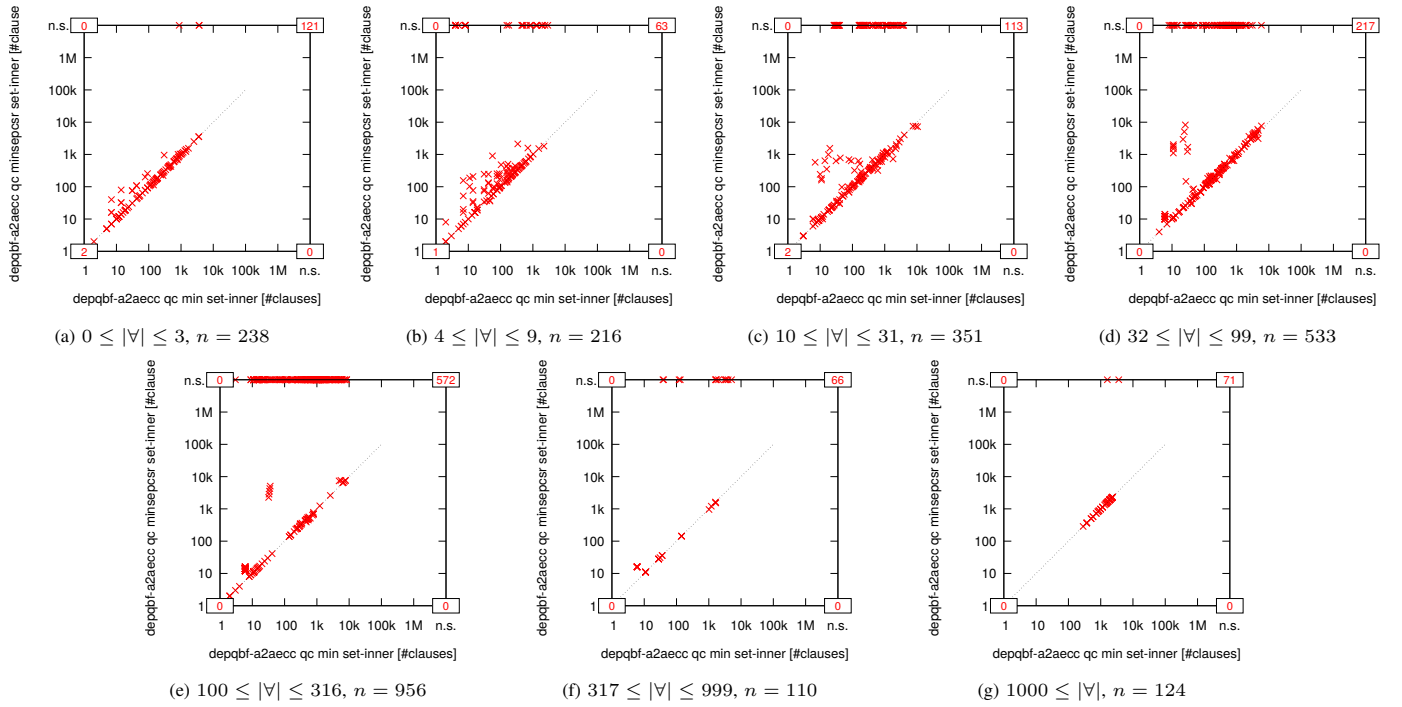


Fig. 363: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode qc minsepcsr set-inner partitioned by number of \forall quantified variables (number of clauses).

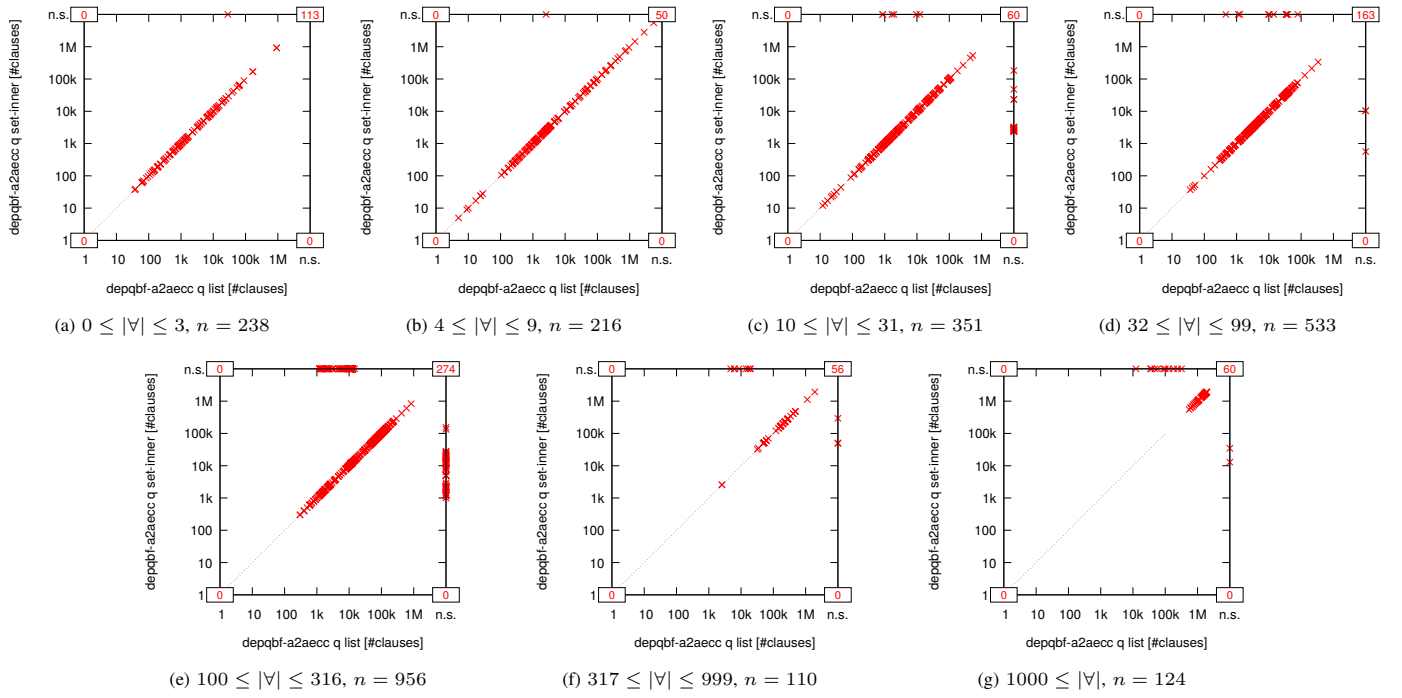


Fig. 364: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode q set-inner partitioned by number of \forall quantified variables (number of clauses).

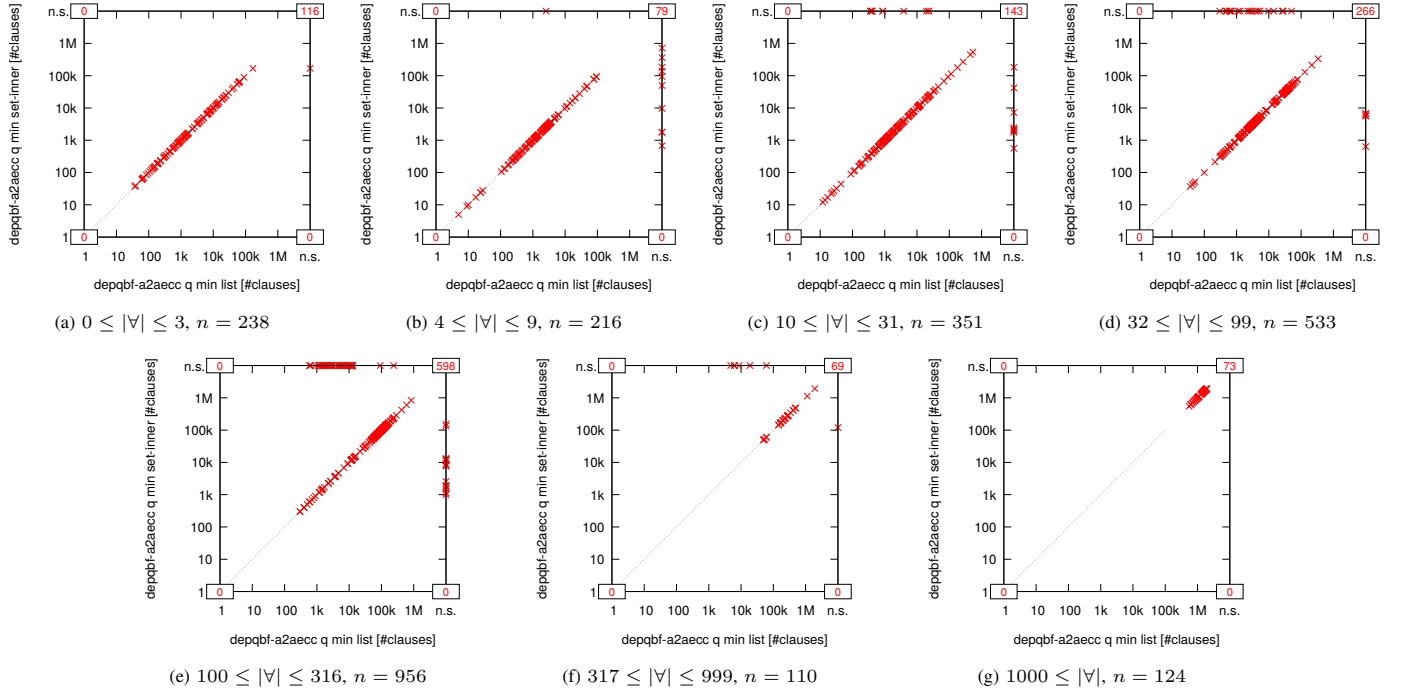


Fig. 365: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q min list versus mode q min set-inner partitioned by number of \forall quantified variables (number of clauses).

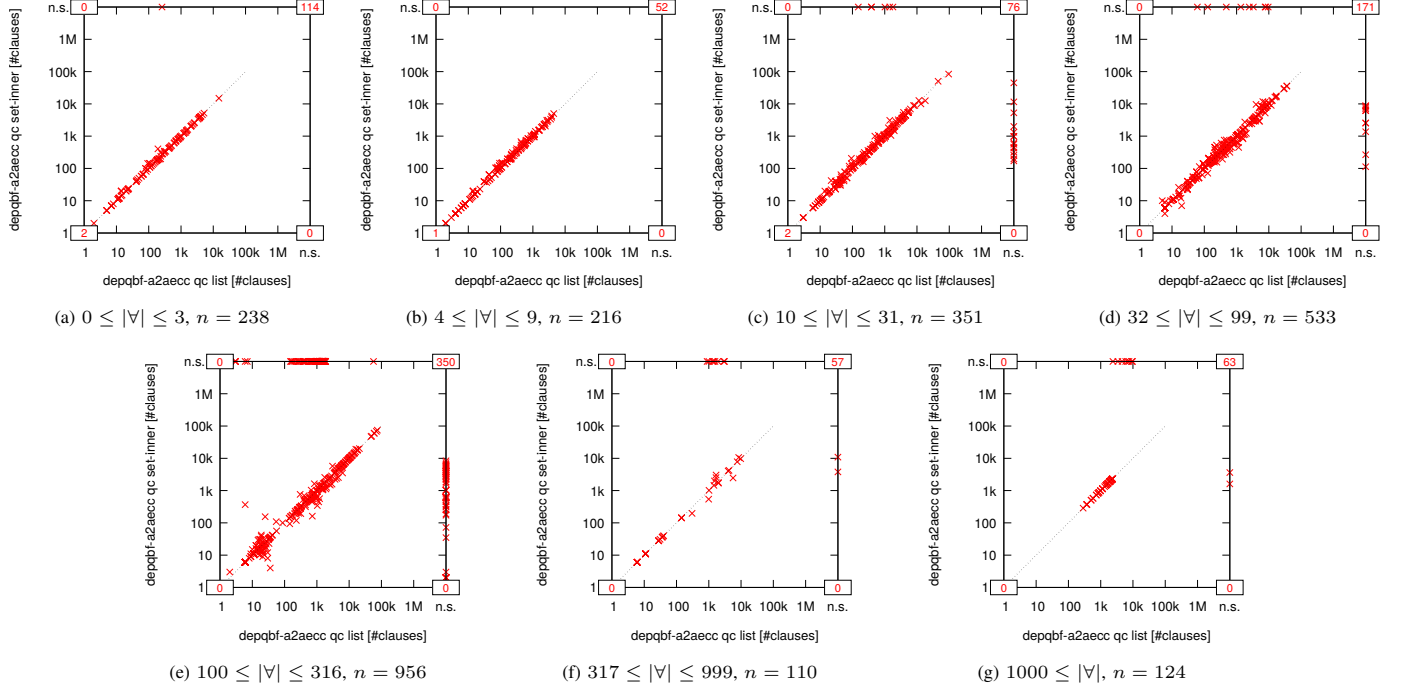


Fig. 366: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode qc set-inner partitioned by number of \forall quantified variables (number of clauses).

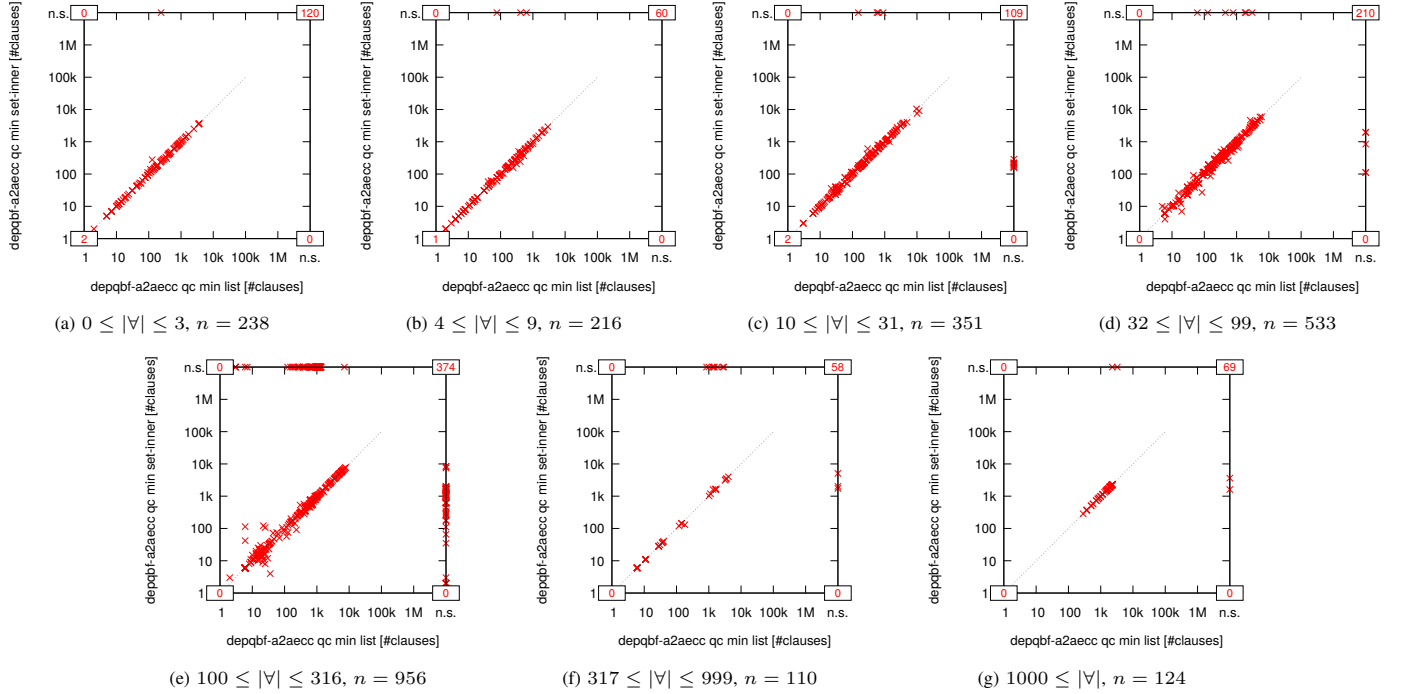


Fig. 367: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode qc min set-inner partitioned by number of \forall quantified variables (number of clauses).

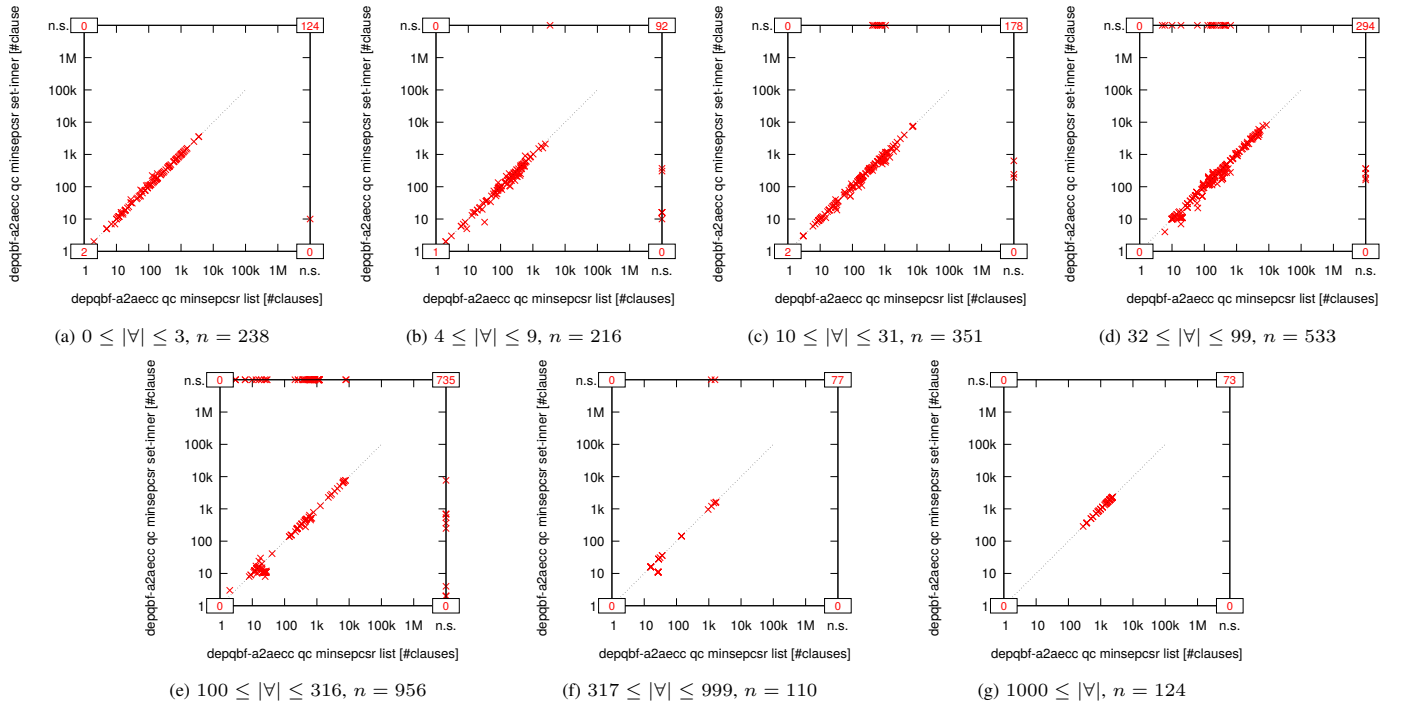


Fig. 368: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode qc minsepcsr set-inner partitioned by number of \forall quantified variables (number of clauses).

C. Partitioned by Alternation Depth

This subsection shows figures of the sizes of unsatisfiable cores with subfigures for partitions of the benchmarks according to their alternation depths.

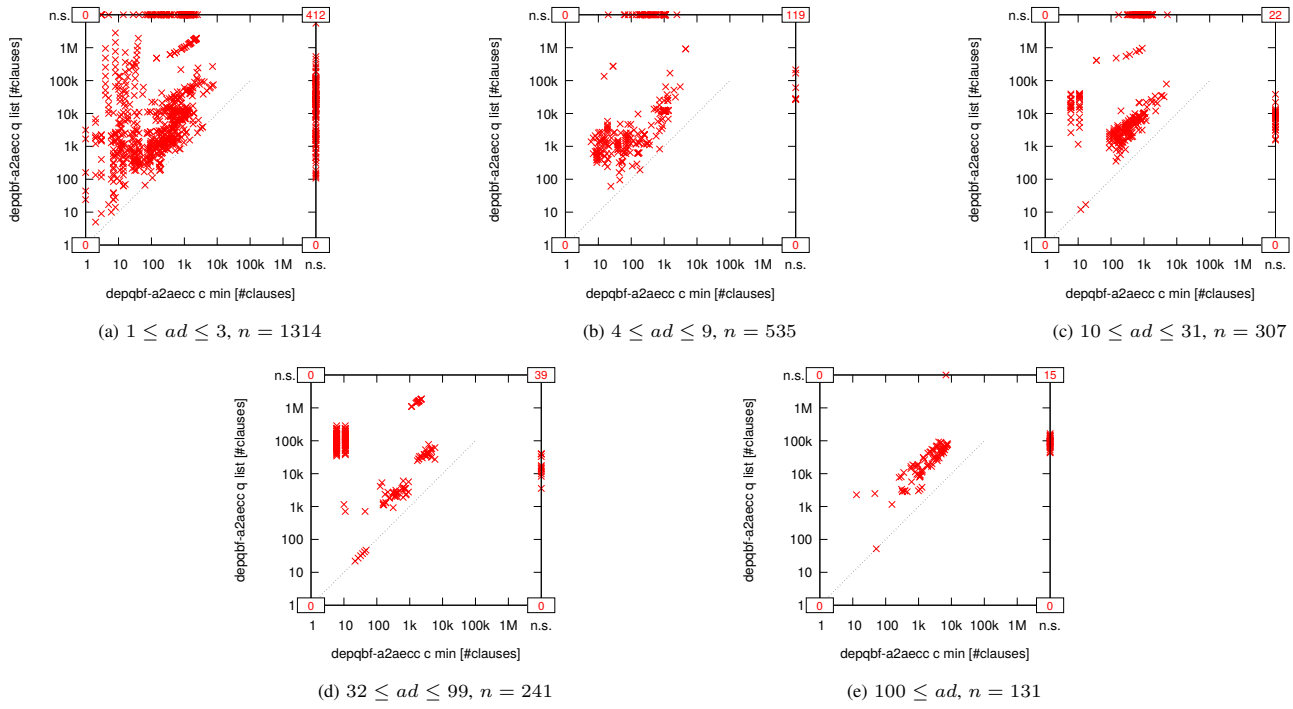


Fig. 369: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q list partitioned by alternation depth (number of clauses).

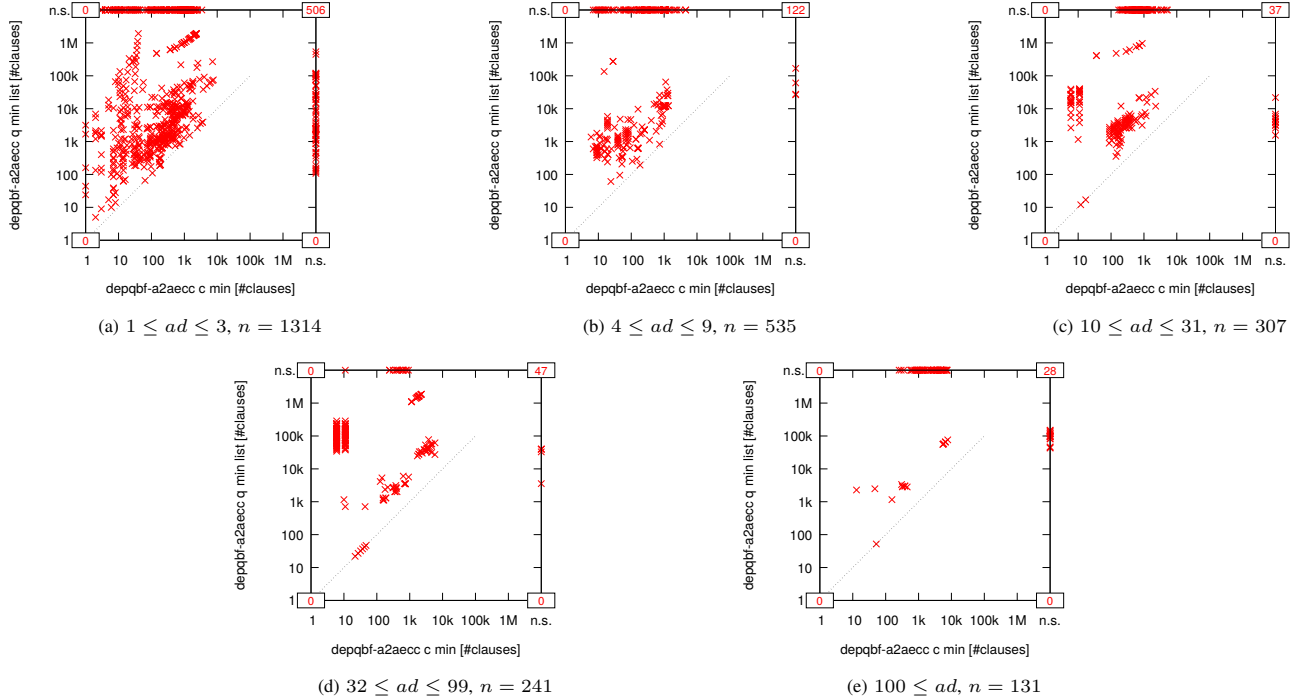


Fig. 370: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q min list partitioned by alternation depth (number of clauses).

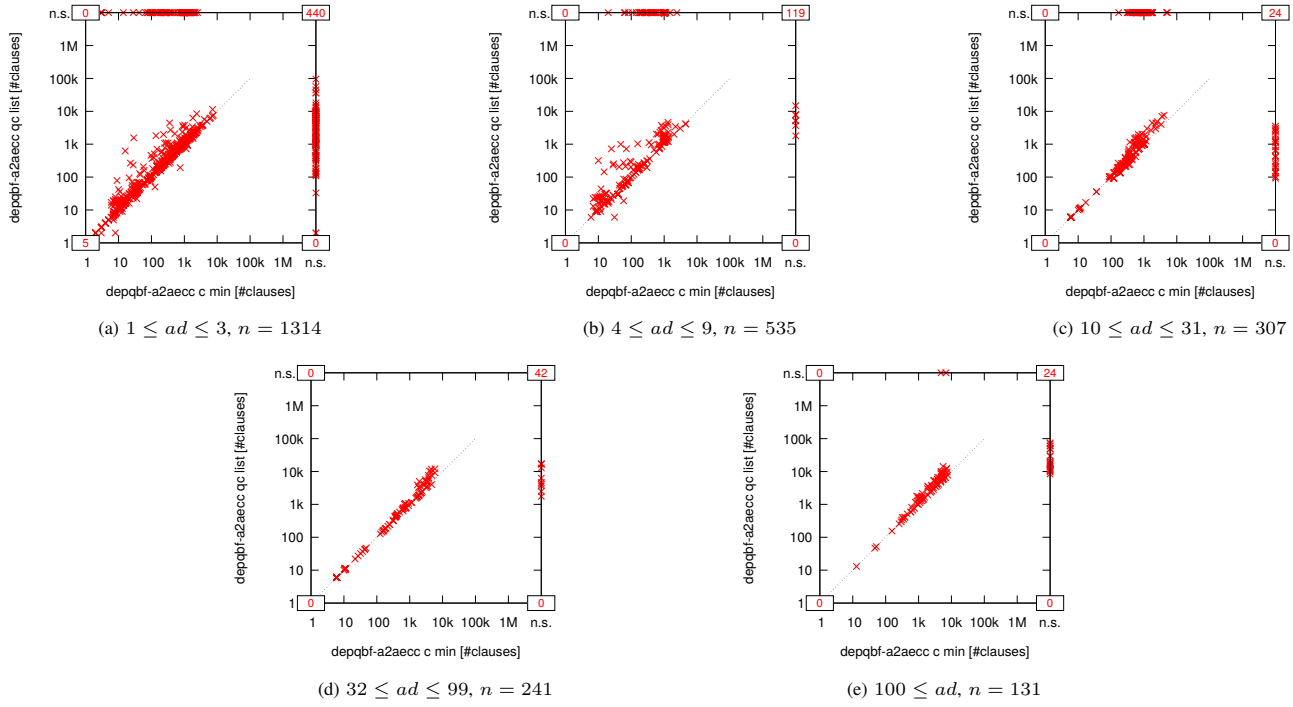


Fig. 371: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc list partitioned by alternation depth (number of clauses).

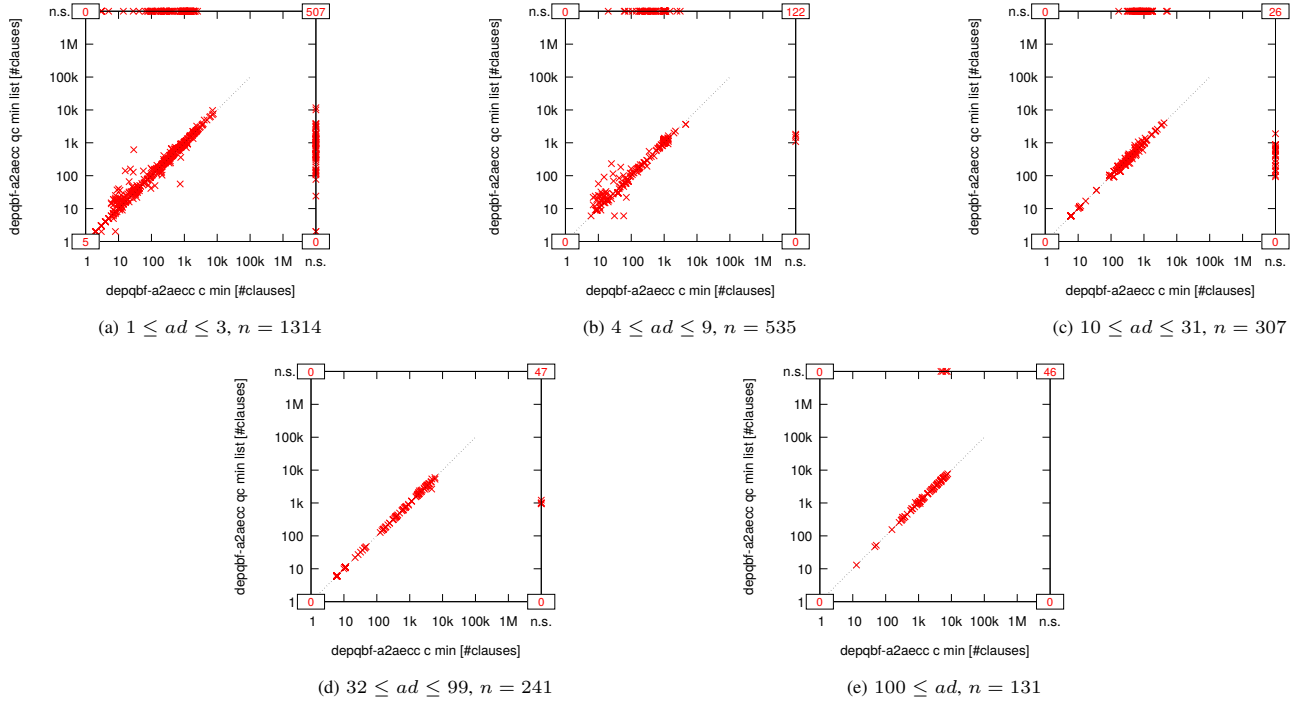


Fig. 372: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc min list partitioned by alternation depth (number of clauses).

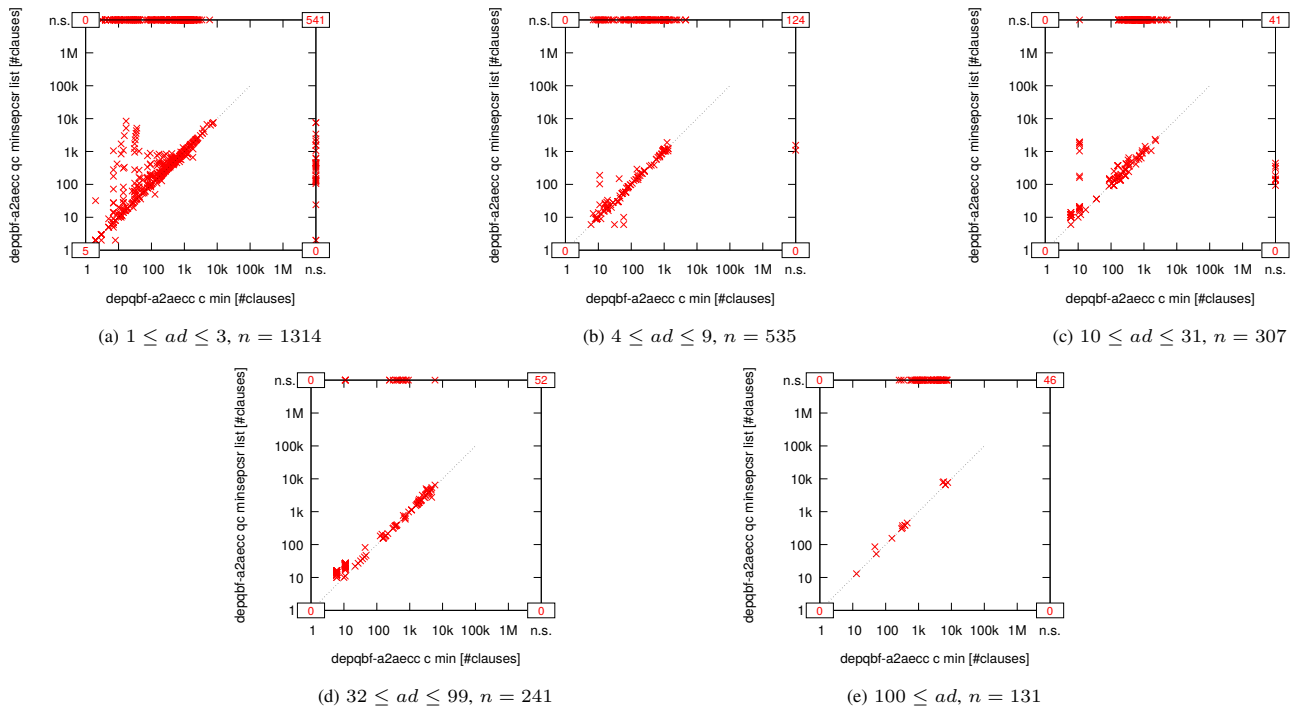


Fig. 373: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc minsepcsr list partitioned by alternation depth (number of clauses).

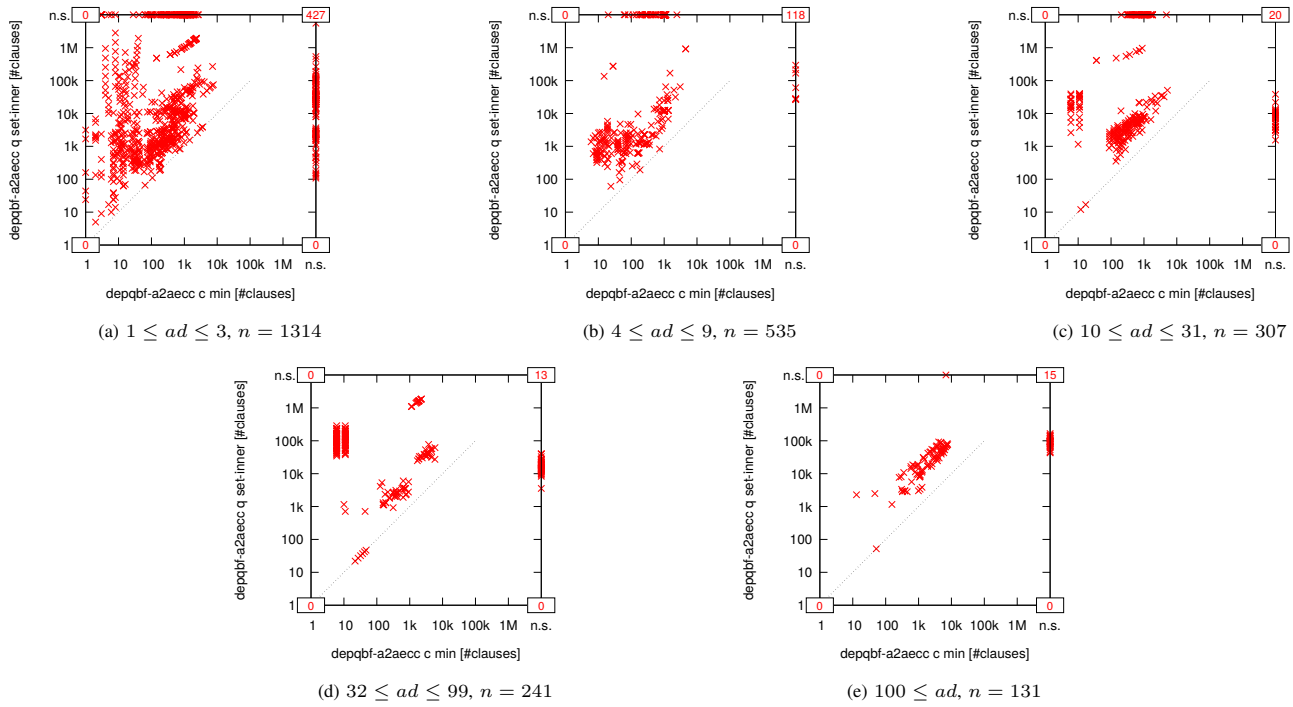


Fig. 374: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q set-inner partitioned by alternation depth (number of clauses).

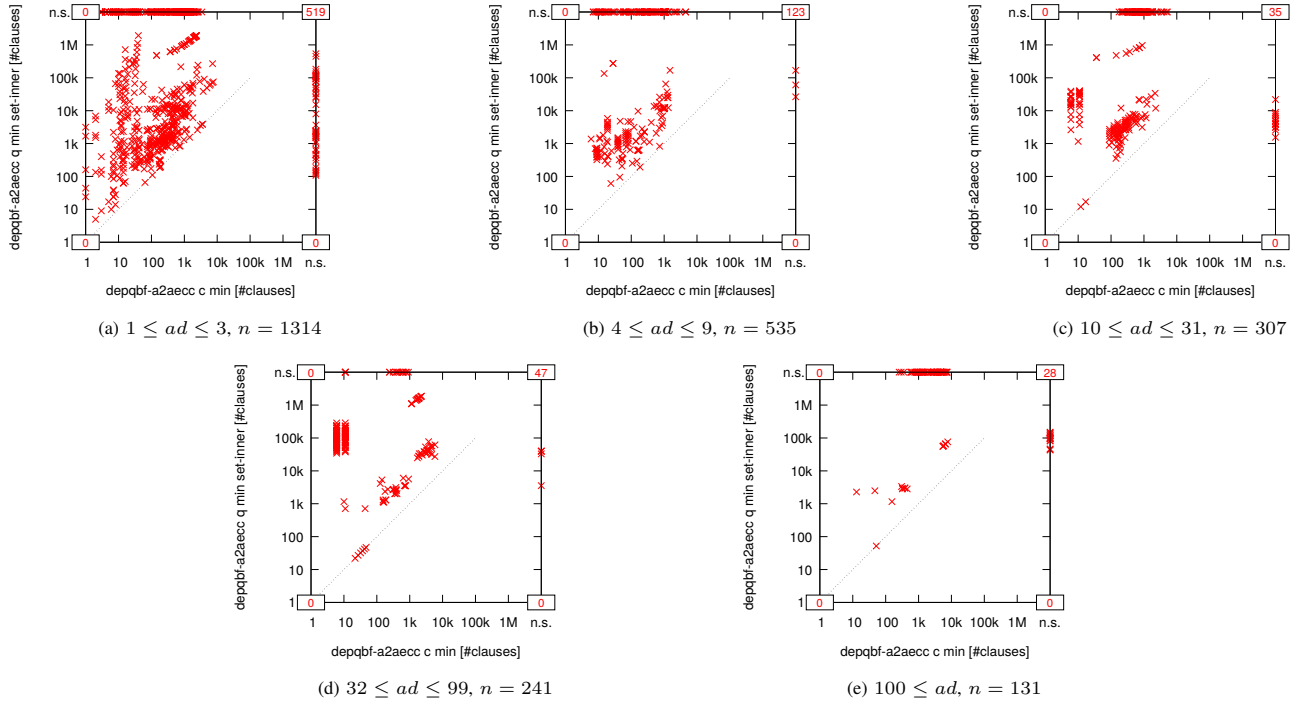


Fig. 375: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q min set-inner partitioned by alternation depth (number of clauses).

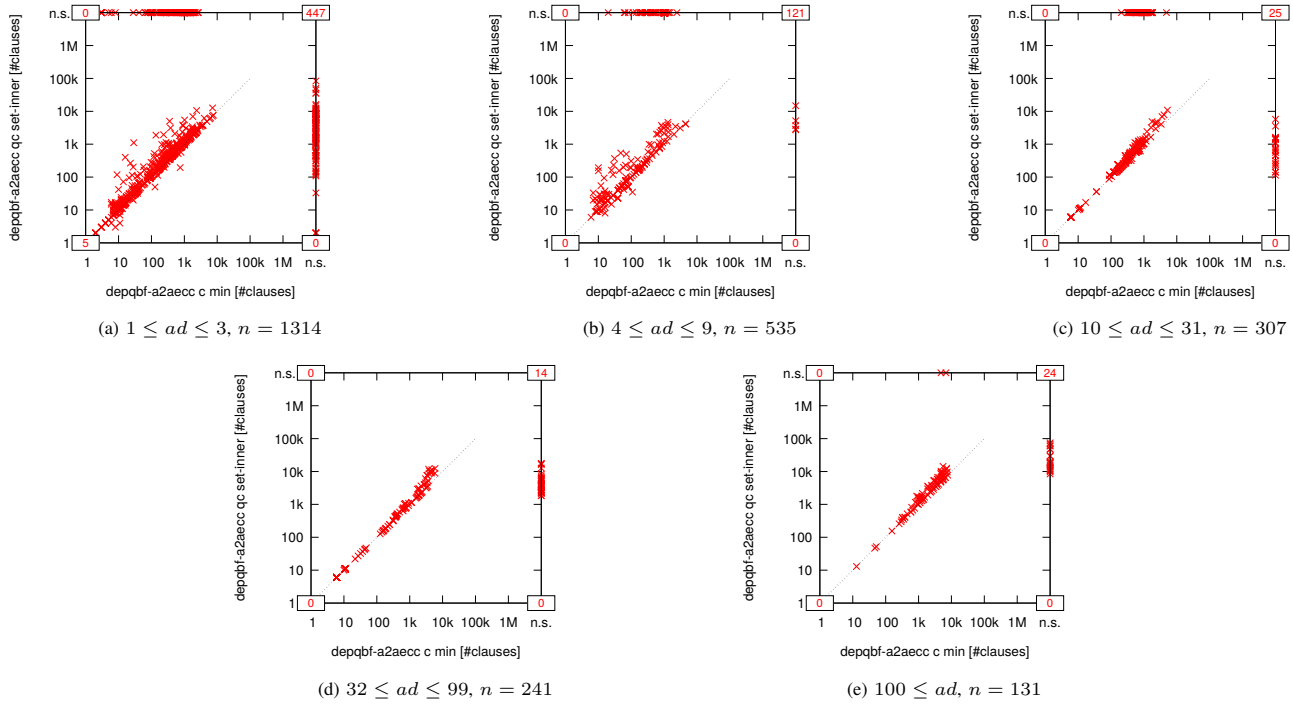


Fig. 376: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc set-inner partitioned by alternation depth (number of clauses).

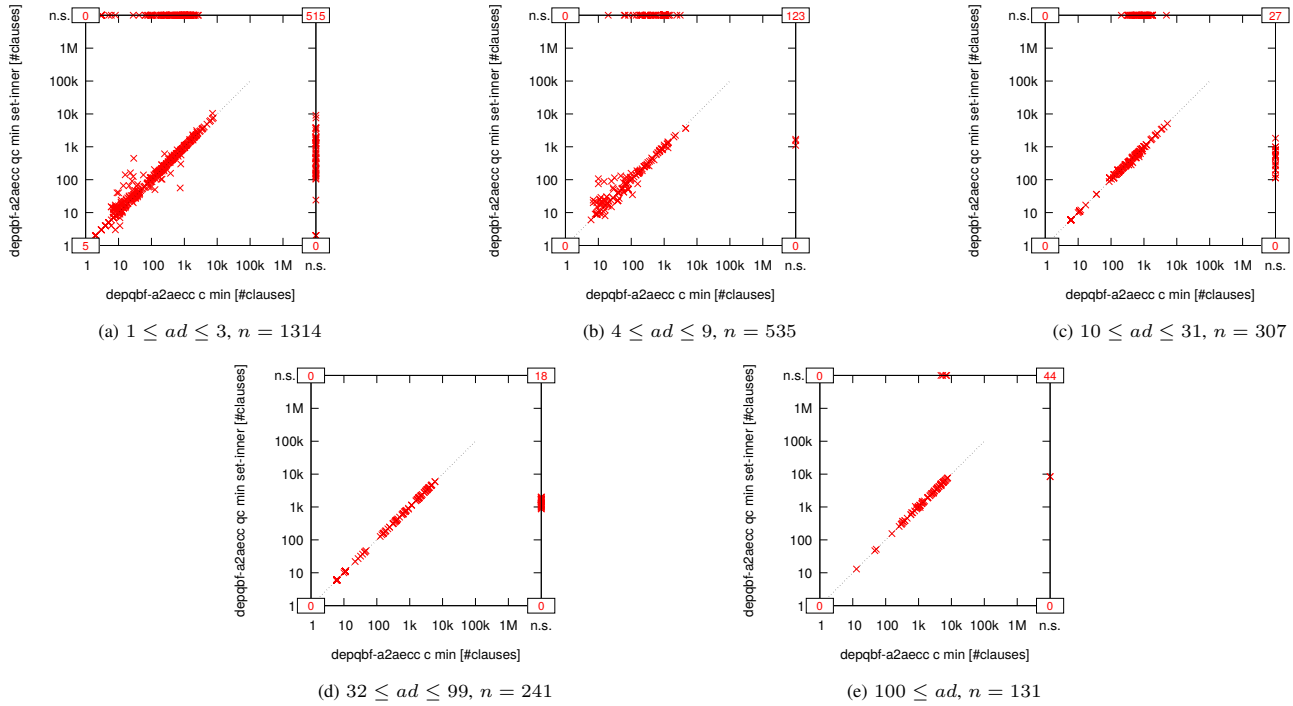


Fig. 377: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc min set-inner partitioned by alternation depth (number of clauses).

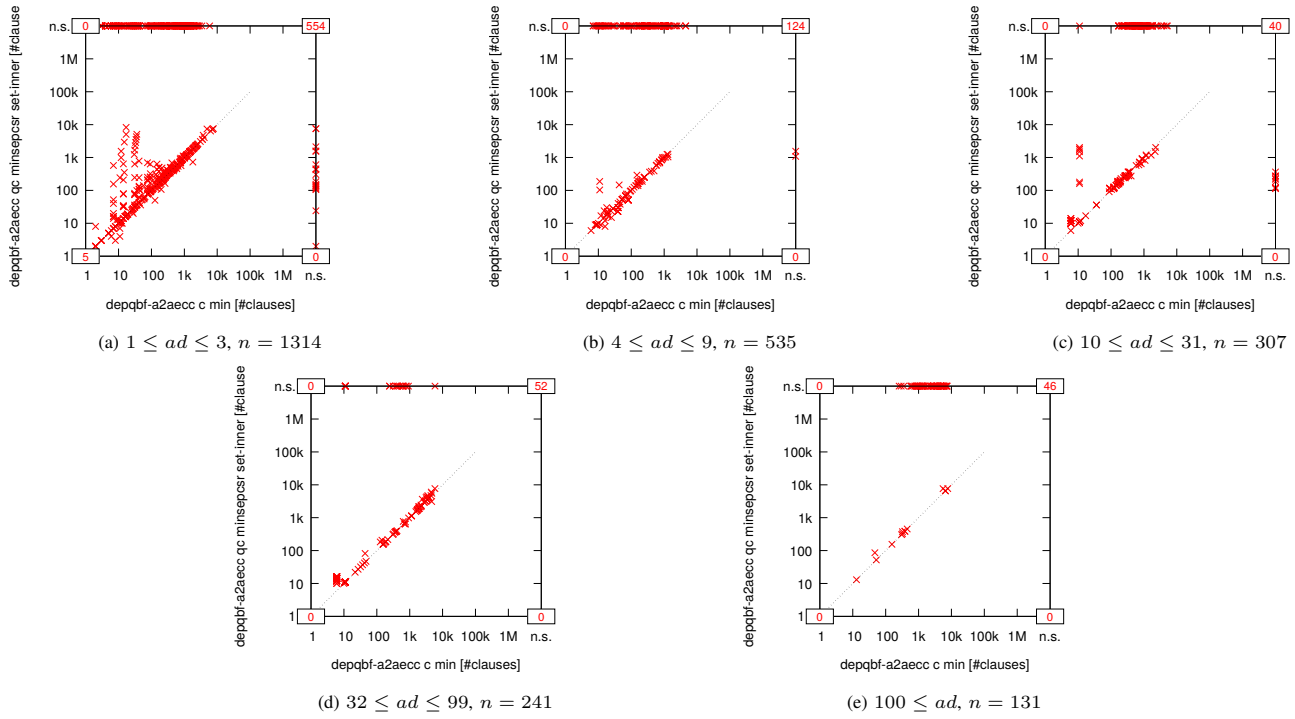


Fig. 378: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc minsepcsr set-inner partitioned by alternation depth (number of clauses).

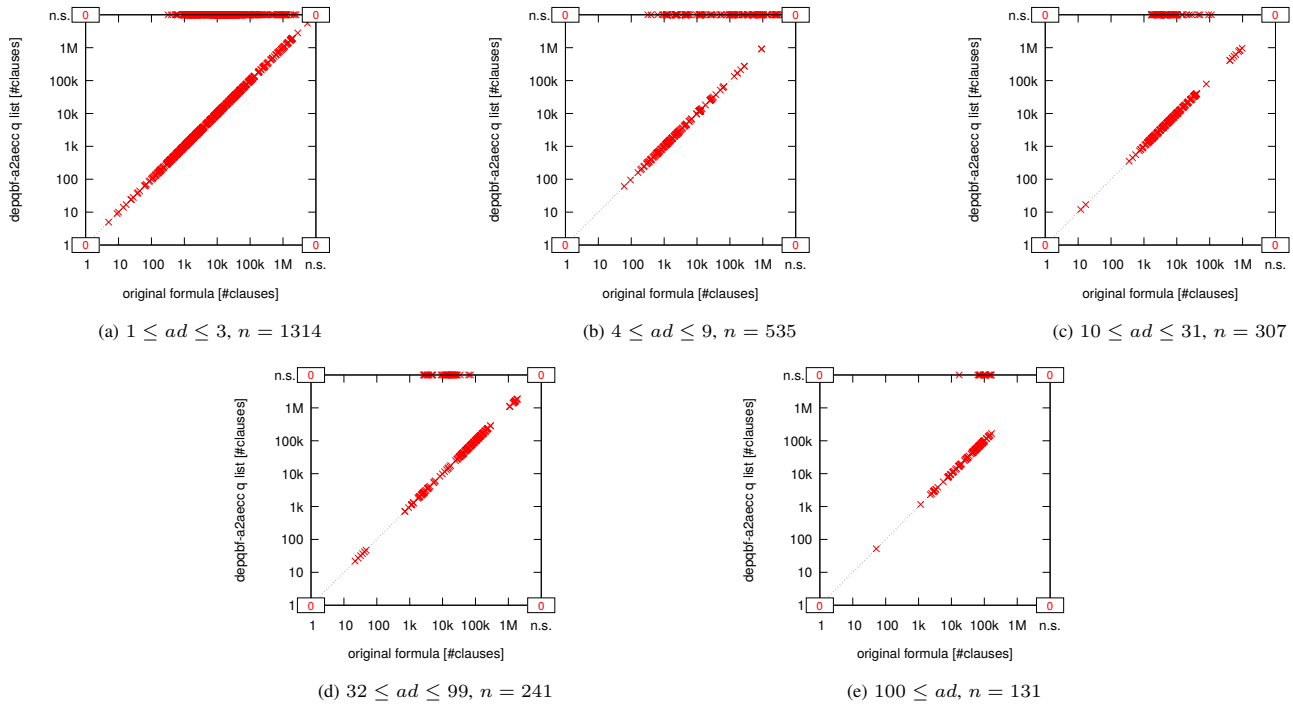


Fig. 379: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q list partitioned by alternation depth (number of clauses).

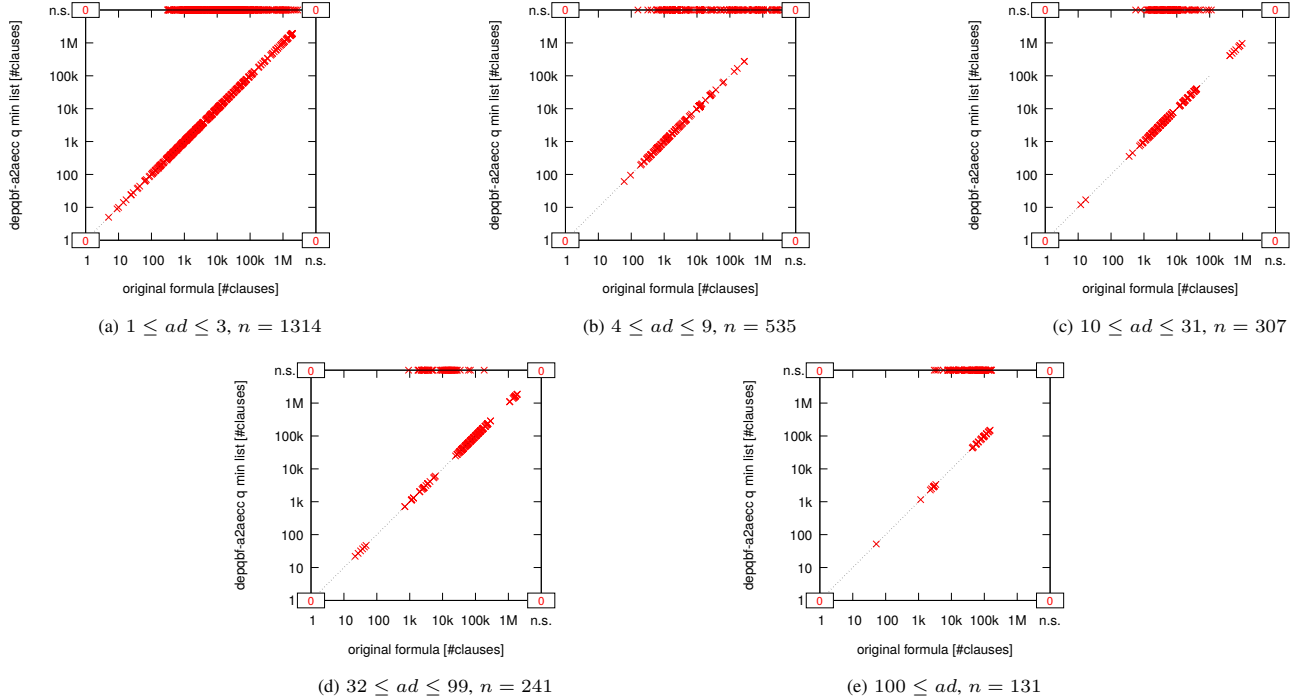


Fig. 380: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q min list partitioned by alternation depth (number of clauses).

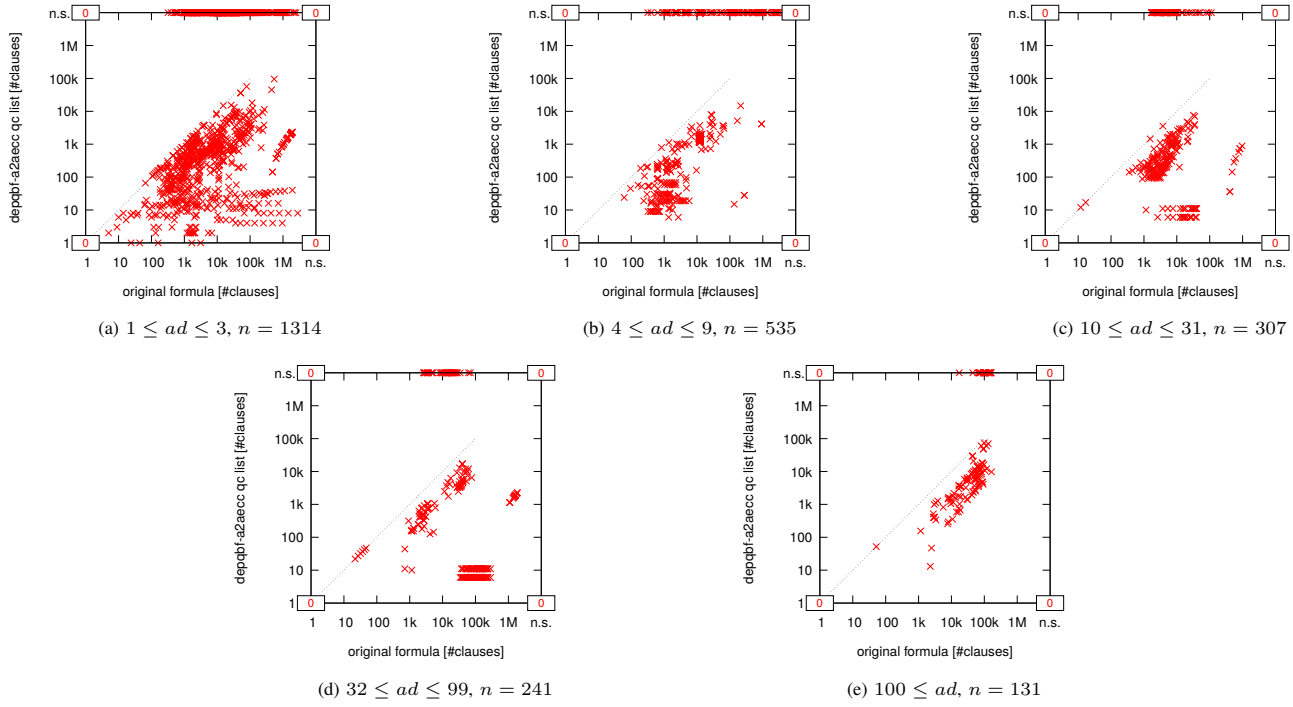


Fig. 381: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc list partitioned by alternation depth (number of clauses).

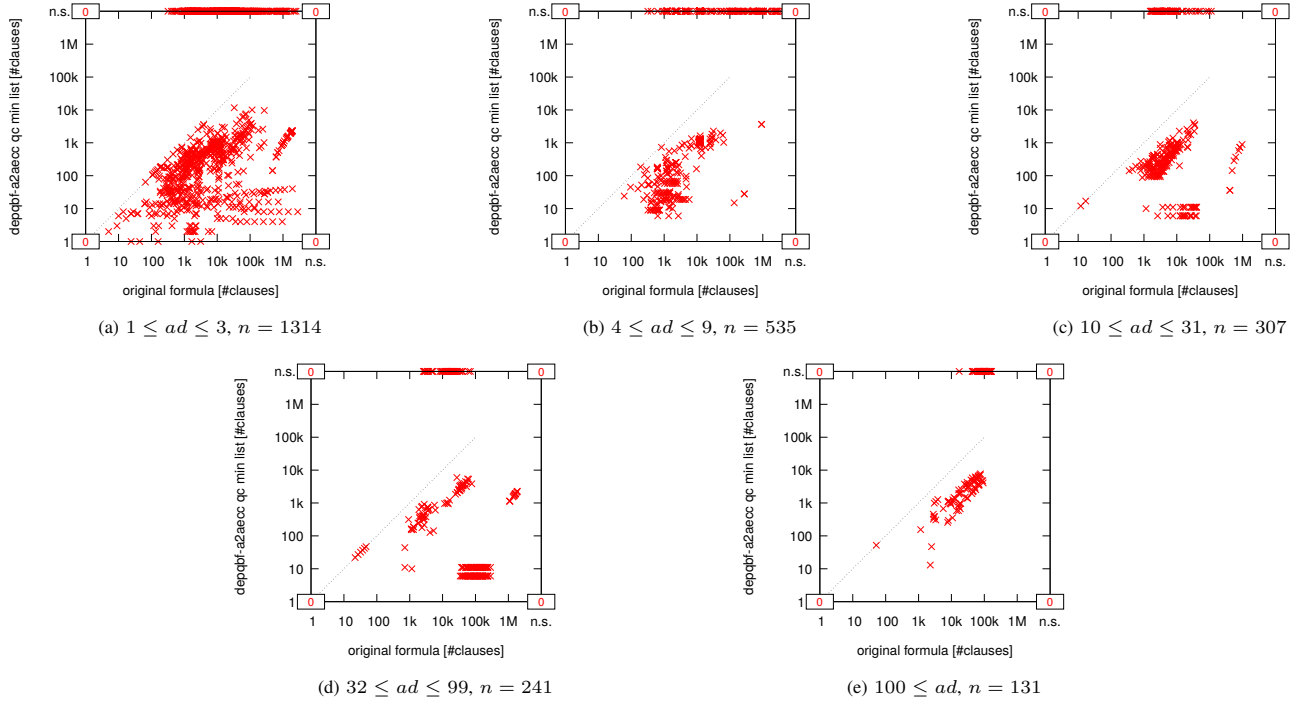


Fig. 382: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc min list partitioned by alternation depth (number of clauses).

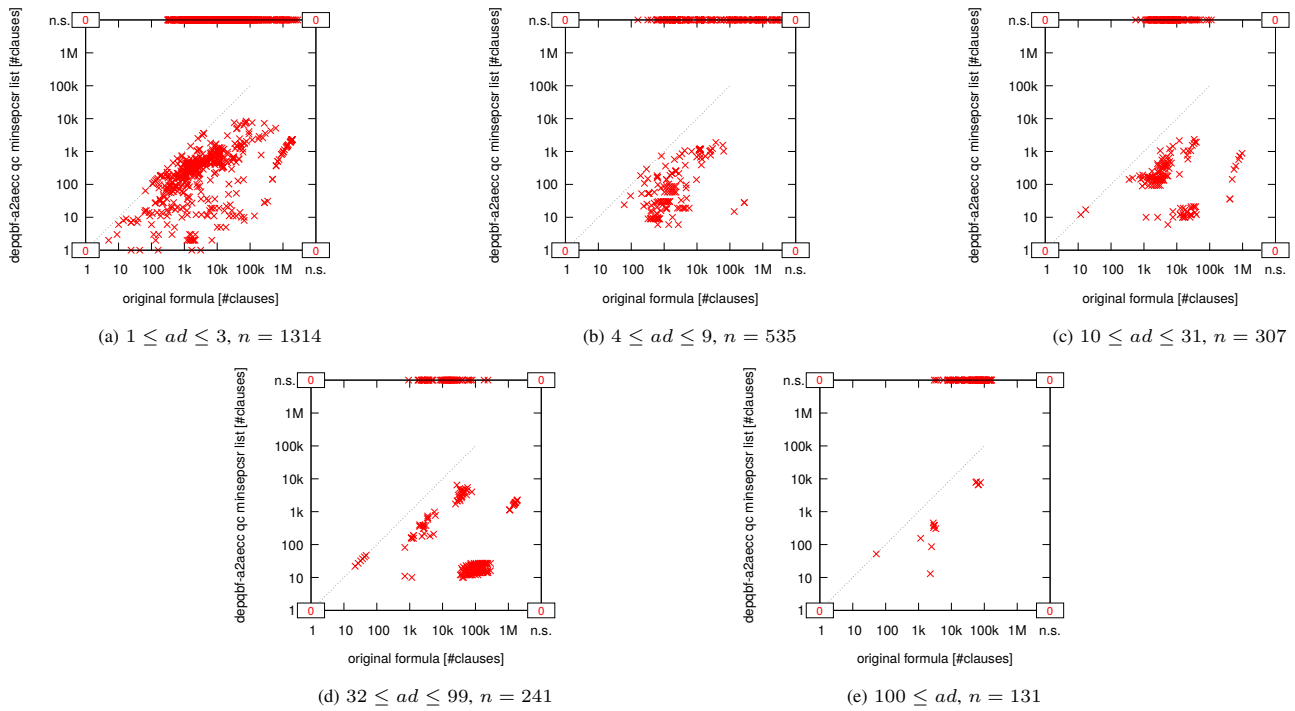


Fig. 383: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc minsepcsr list partitioned by alternation depth (number of clauses).

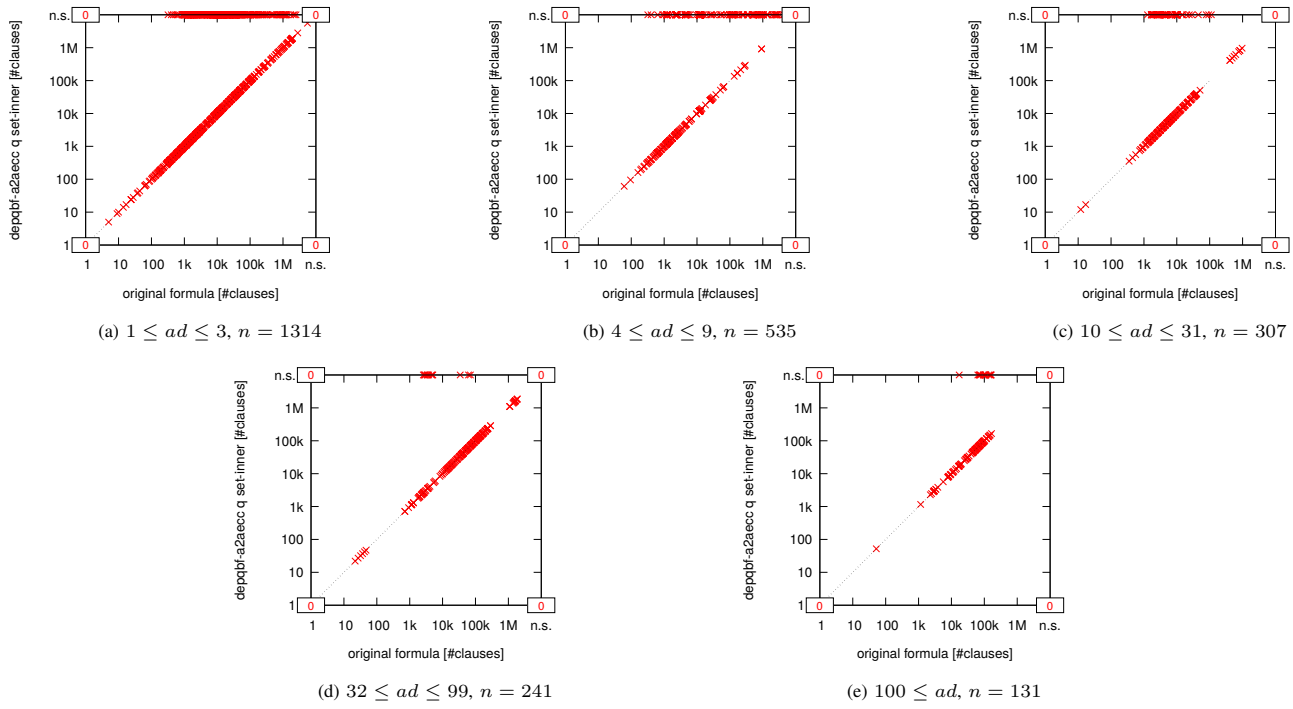


Fig. 384: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q set-inner partitioned by alternation depth (number of clauses).

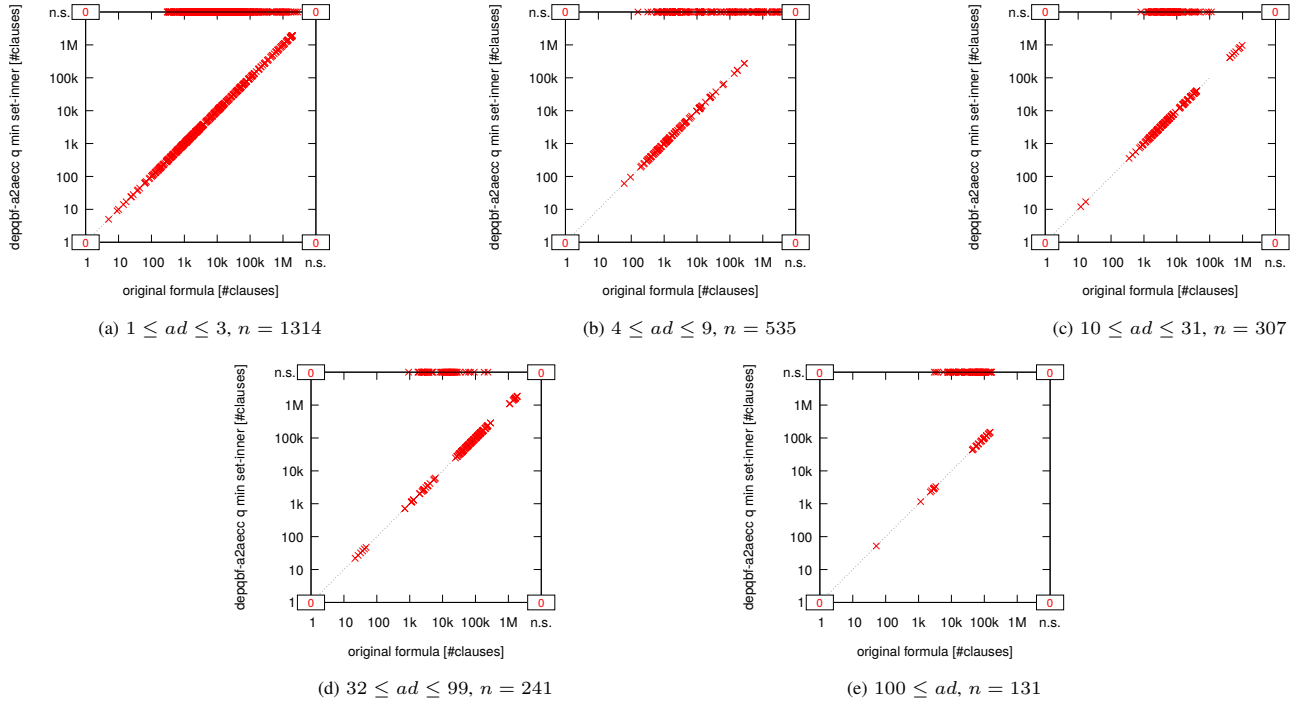


Fig. 385: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q min set-inner partitioned by alternation depth (number of clauses).

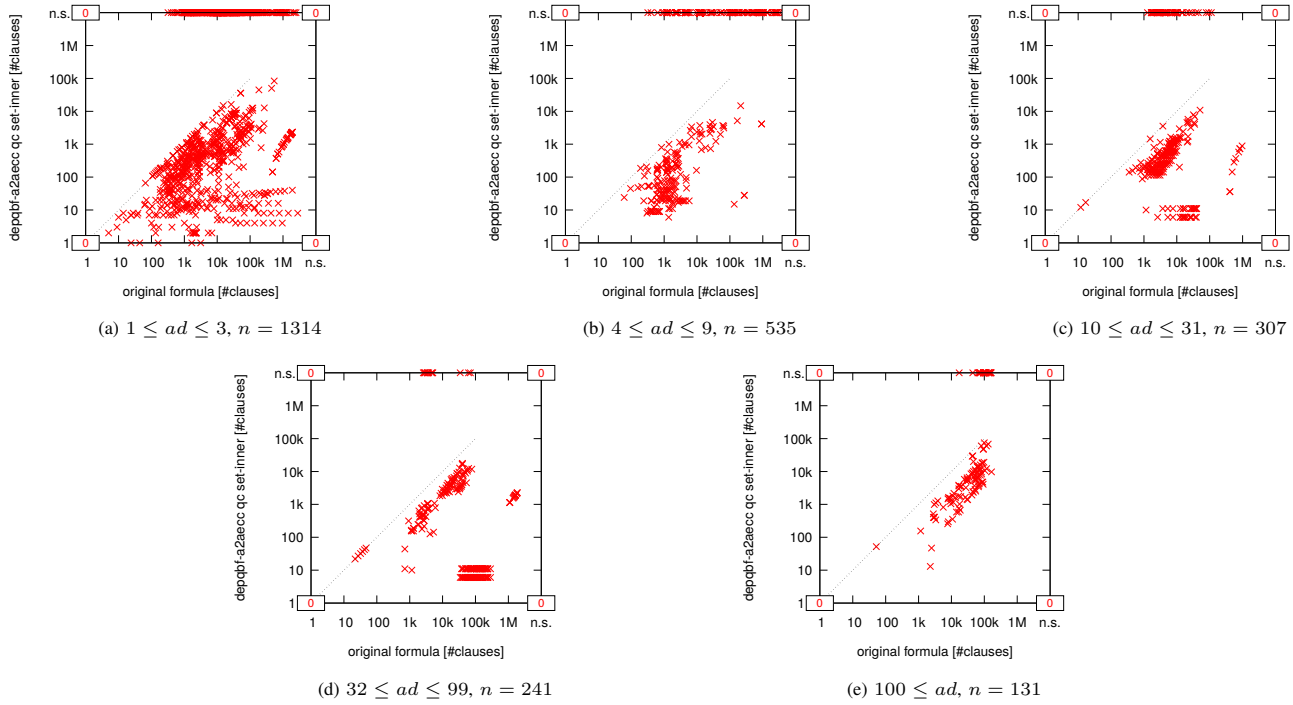


Fig. 386: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc set-inner partitioned by alternation depth (number of clauses).

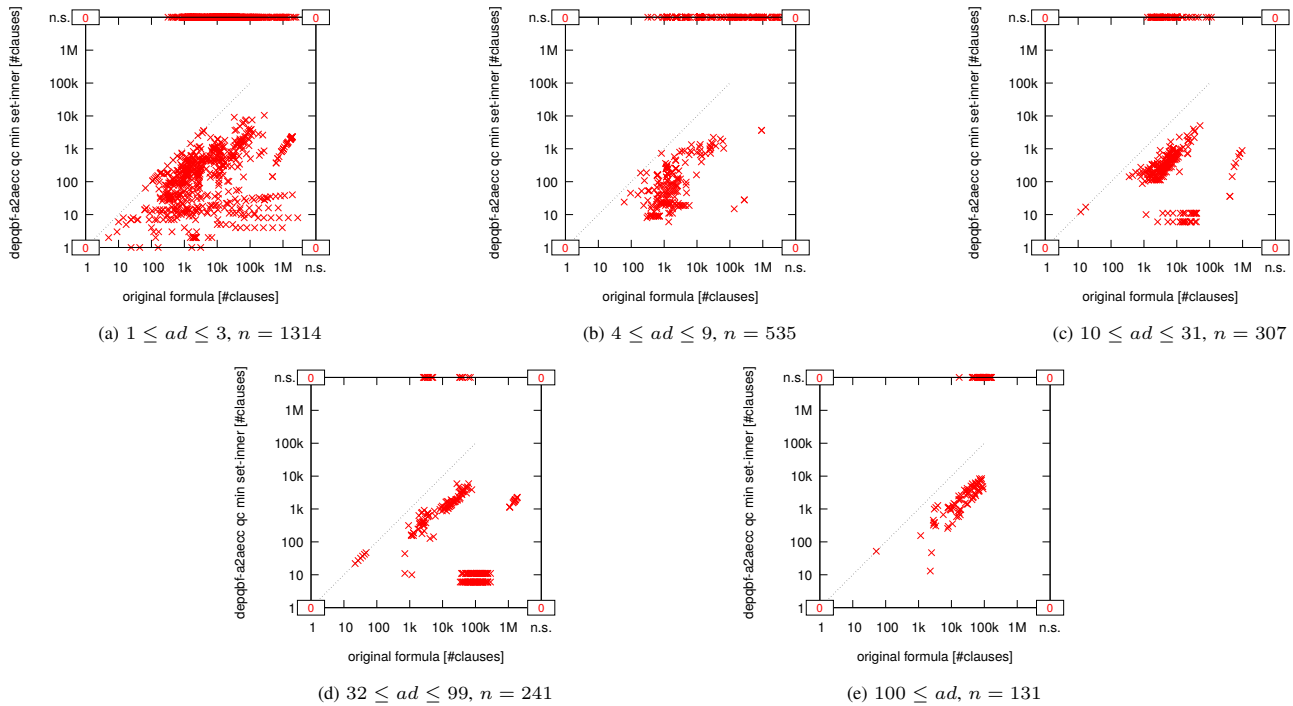


Fig. 387: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc min set-inner partitioned by alternation depth (number of clauses).

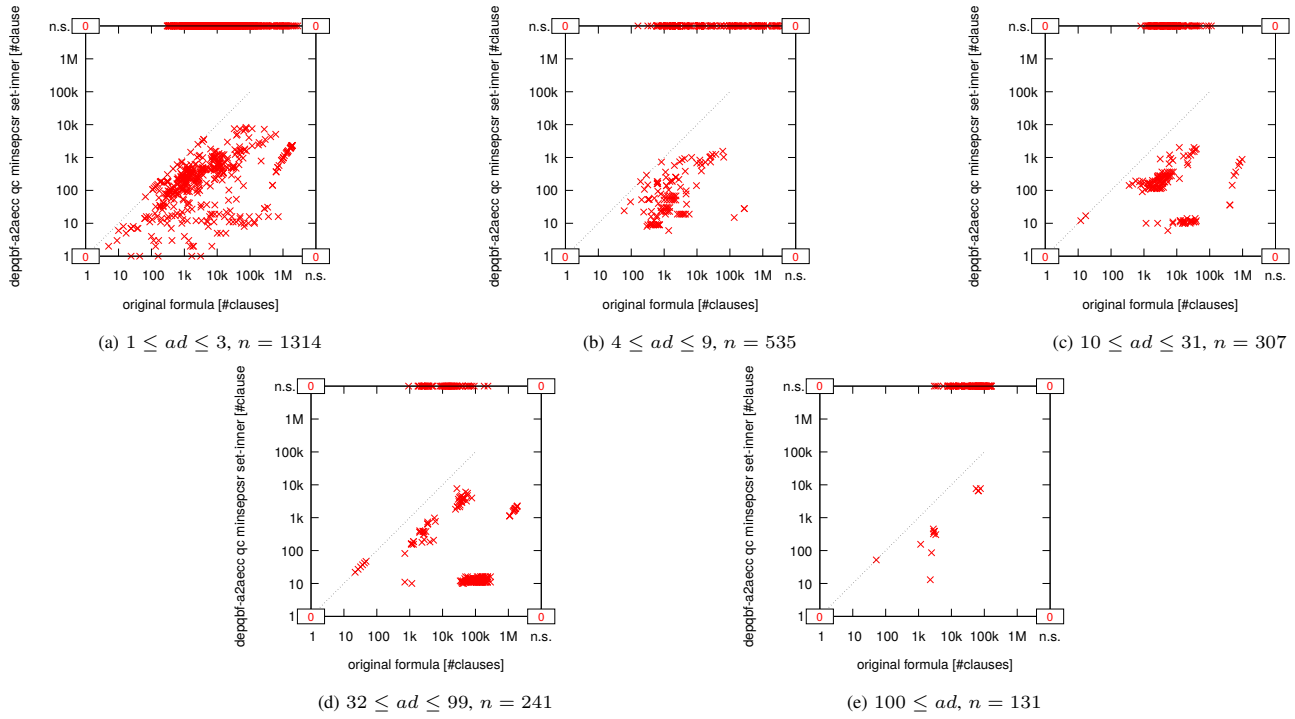


Fig. 388: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc minsepcsr set-inner partitioned by alternation depth (number of clauses).

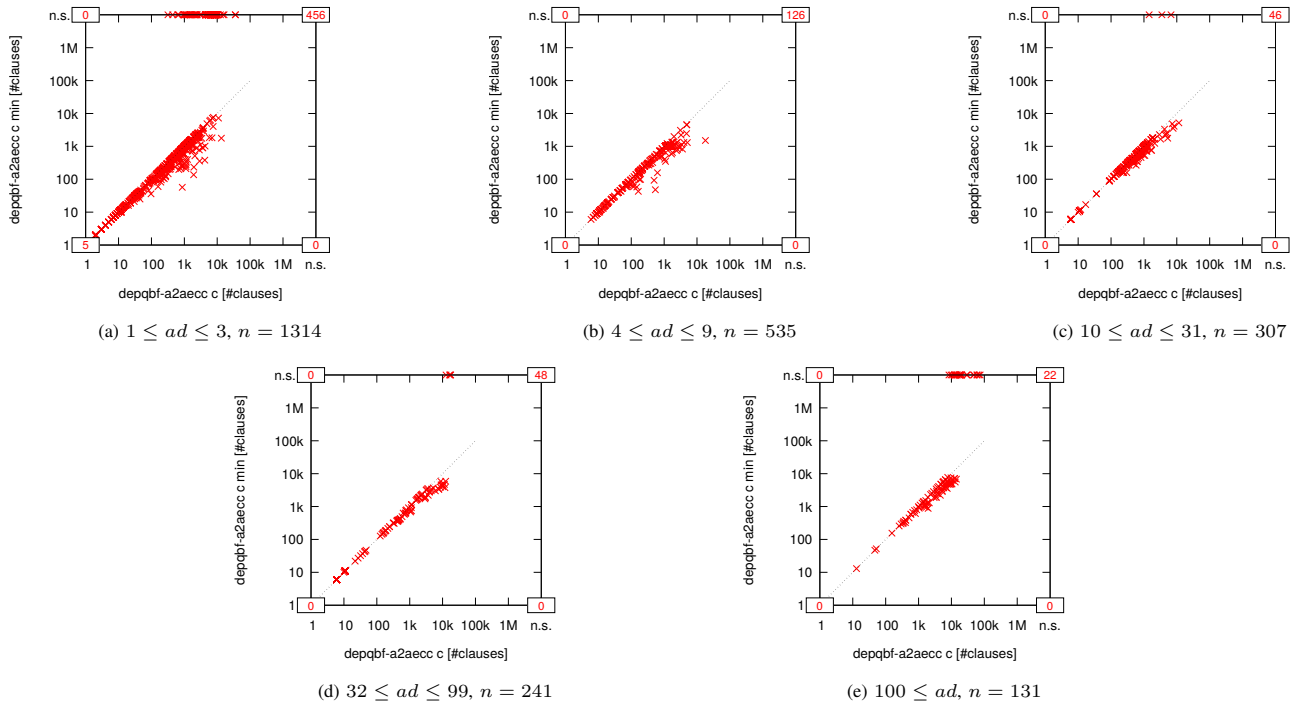


Fig. 389: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c versus mode c min partitioned by alternation depth (number of clauses).

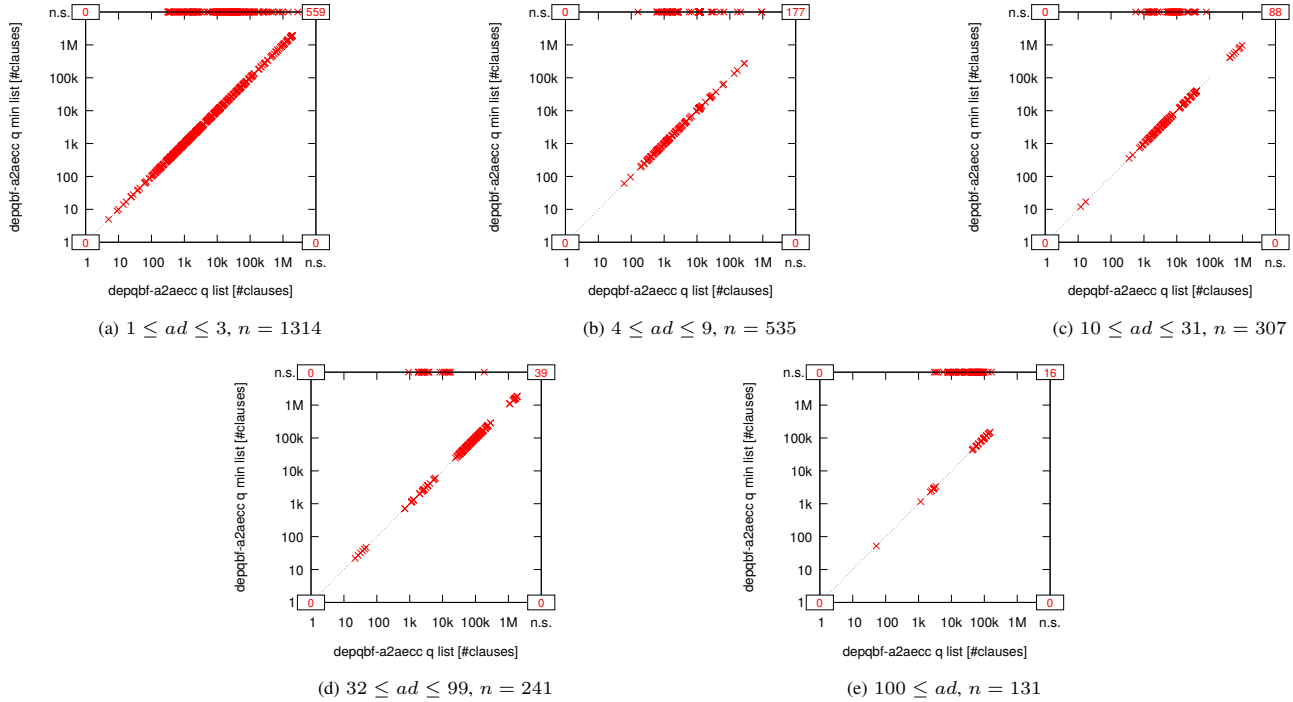


Fig. 390: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode q min list partitioned by alternation depth (number of clauses).

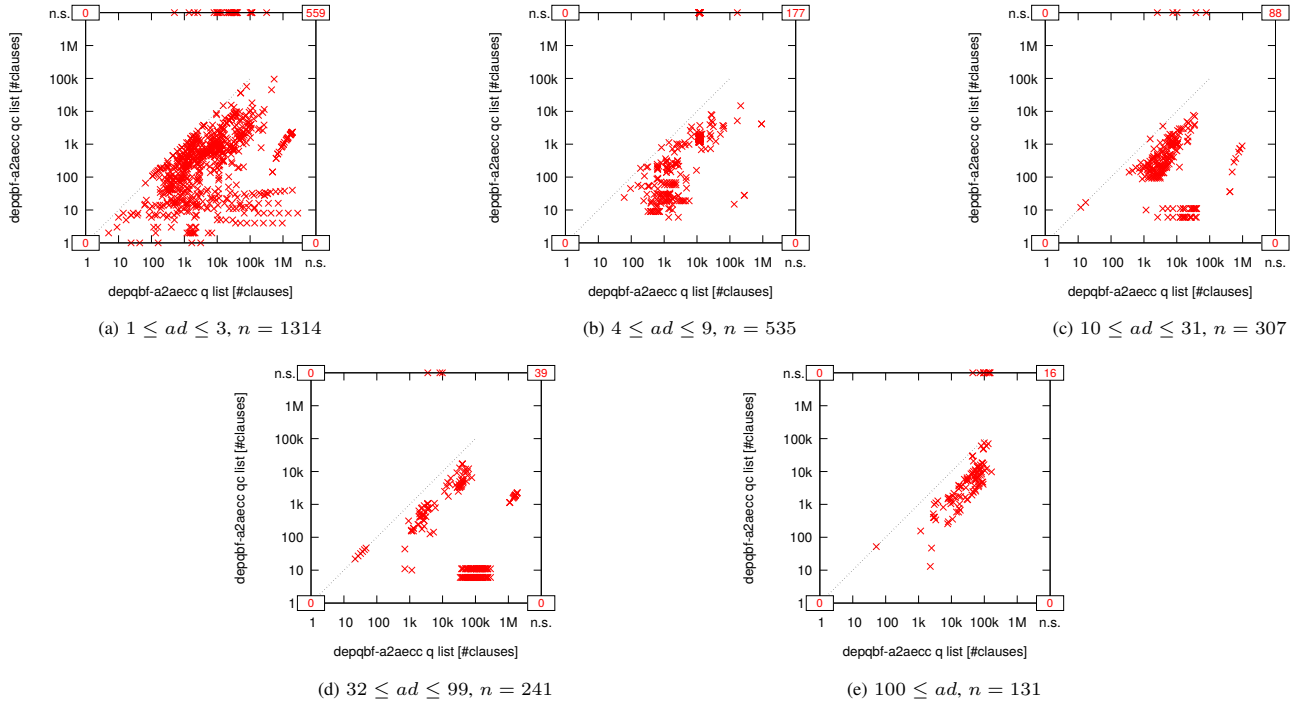


Fig. 391: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode qc list partitioned by alternation depth (number of clauses).

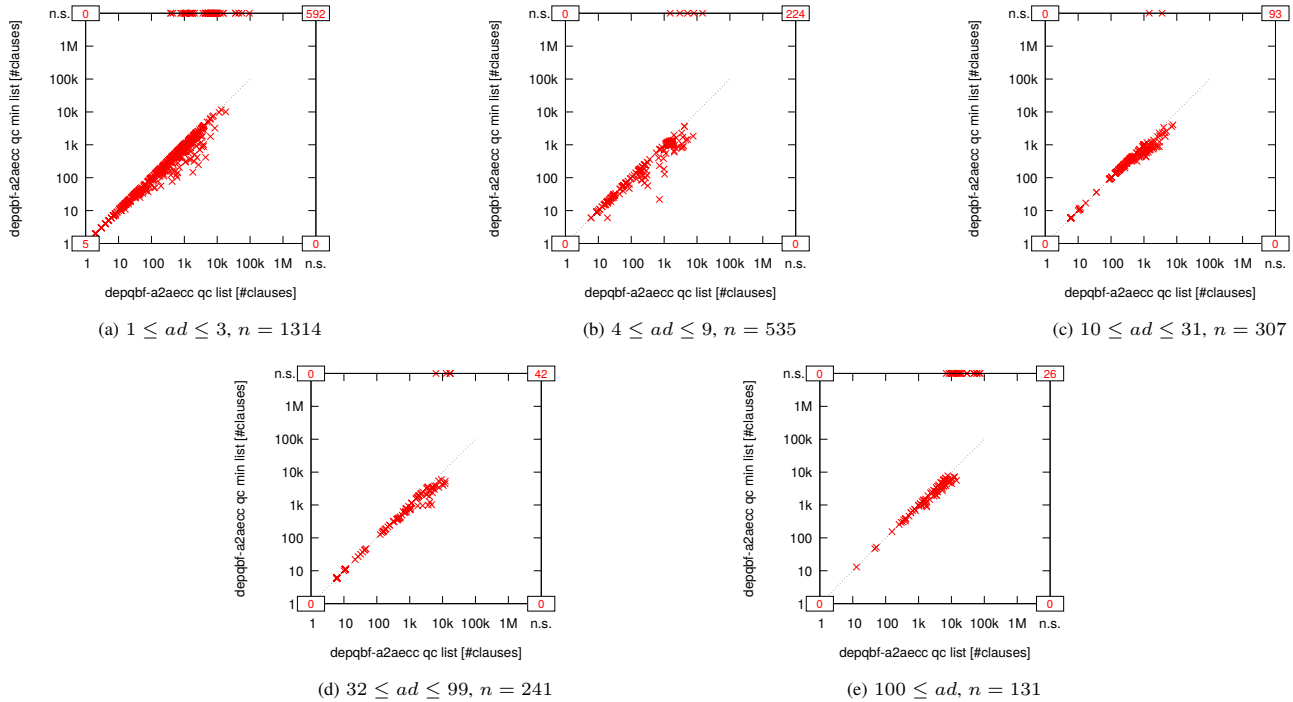


Fig. 392: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode qc min list partitioned by alternation depth (number of clauses).

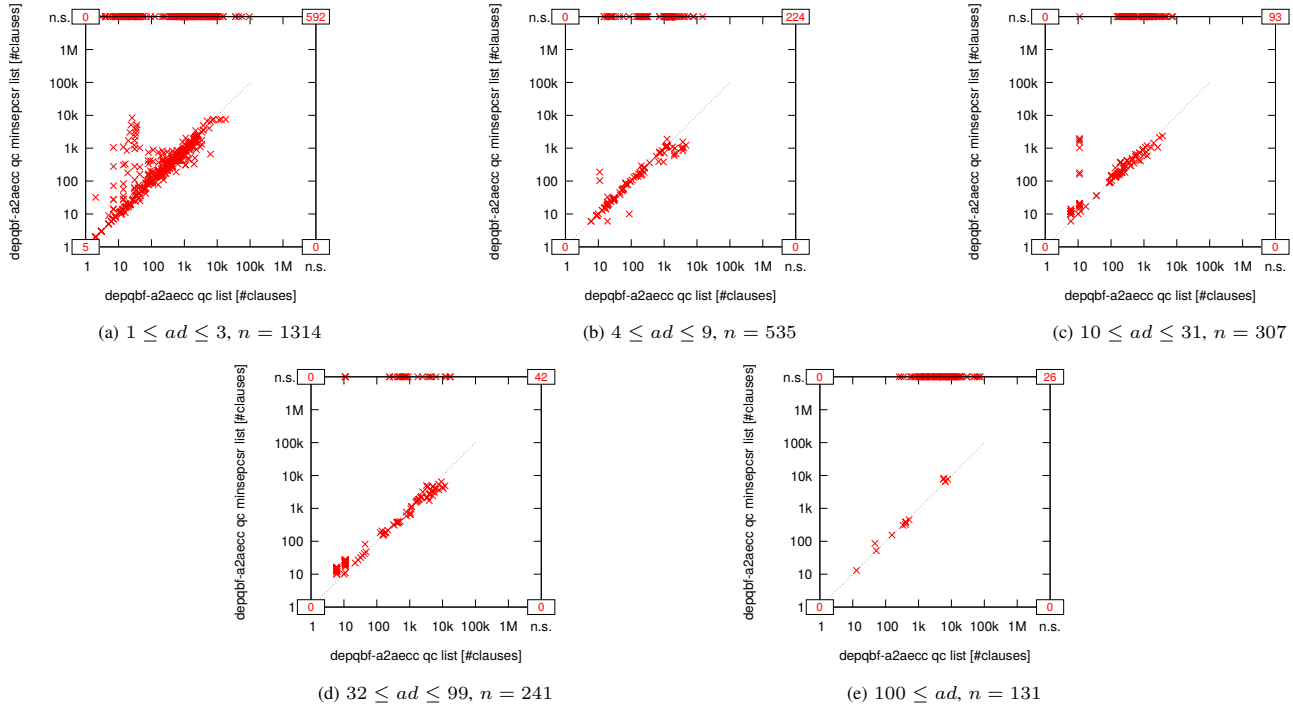


Fig. 393: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode qc minsepcsr list partitioned by alternation depth (number of clauses).

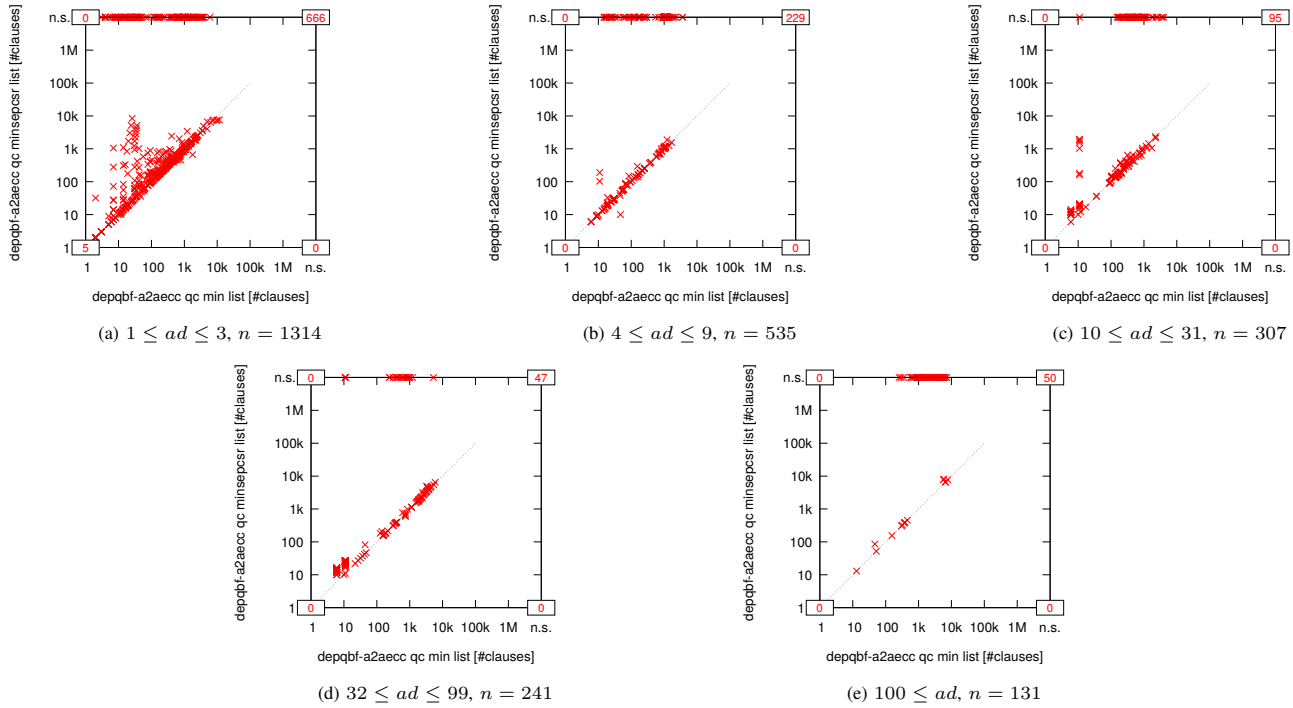


Fig. 394: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode qc minsepcsr list partitioned by alternation depth (number of clauses).

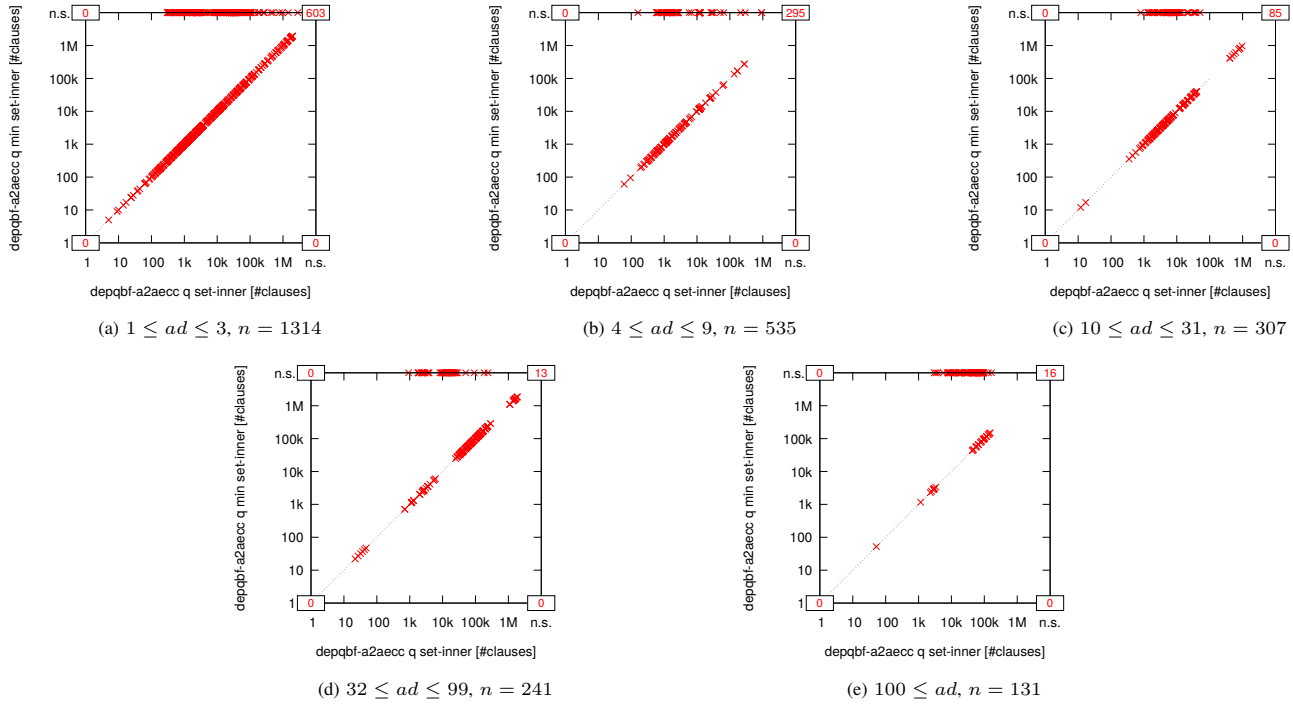


Fig. 395: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode q min set-inner partitioned by alternation depth (number of clauses).

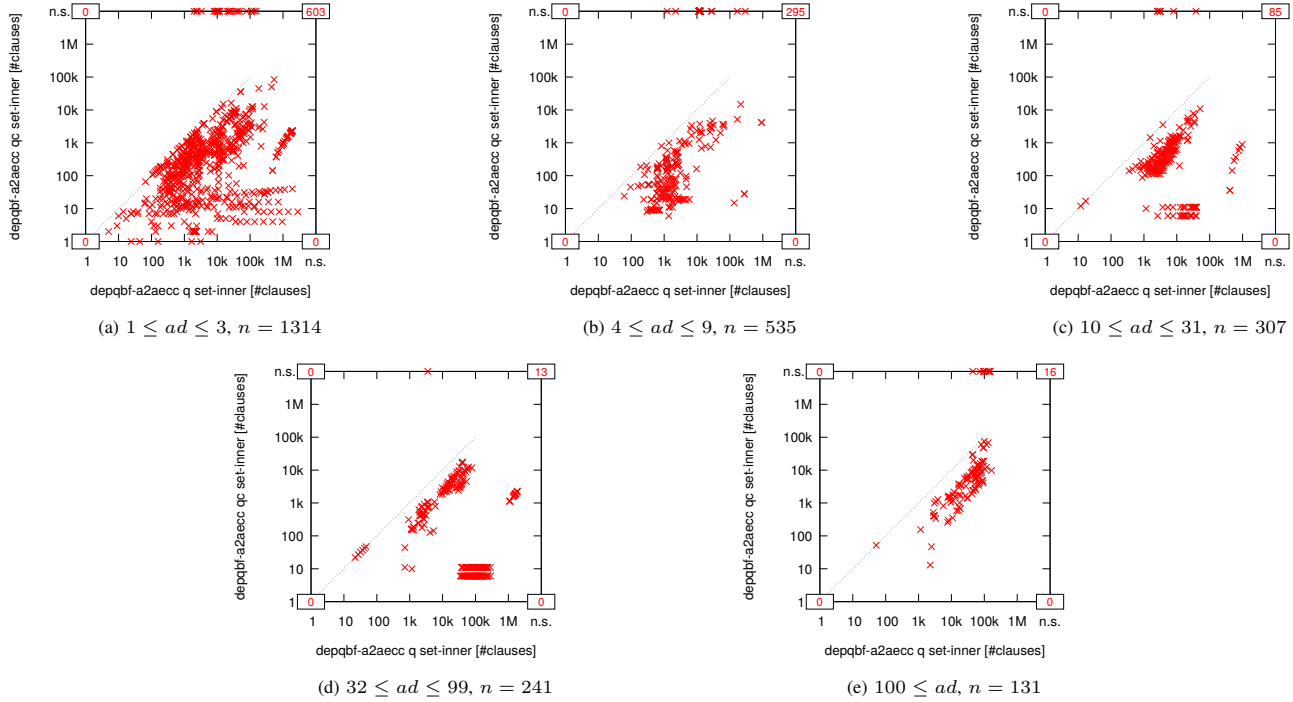


Fig. 396: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode qc set-inner partitioned by alternation depth (number of clauses).

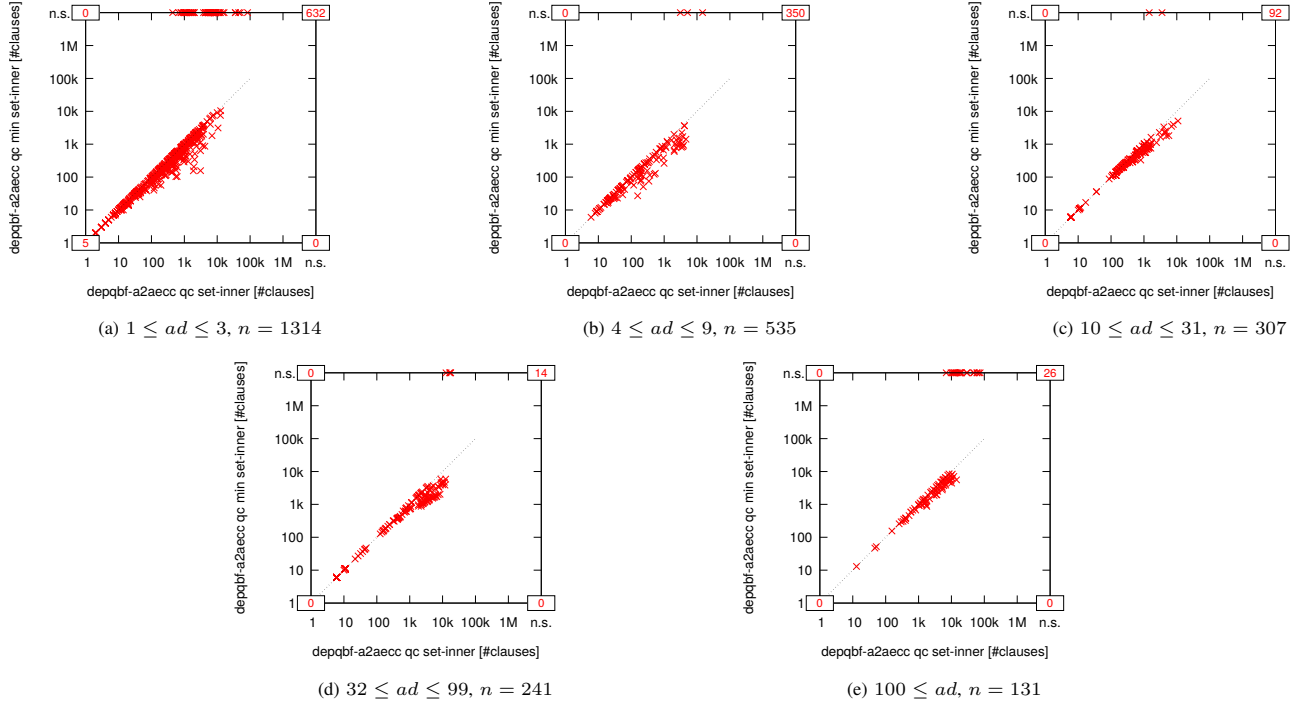


Fig. 397: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode qc min set-inner partitioned by alternation depth (number of clauses).

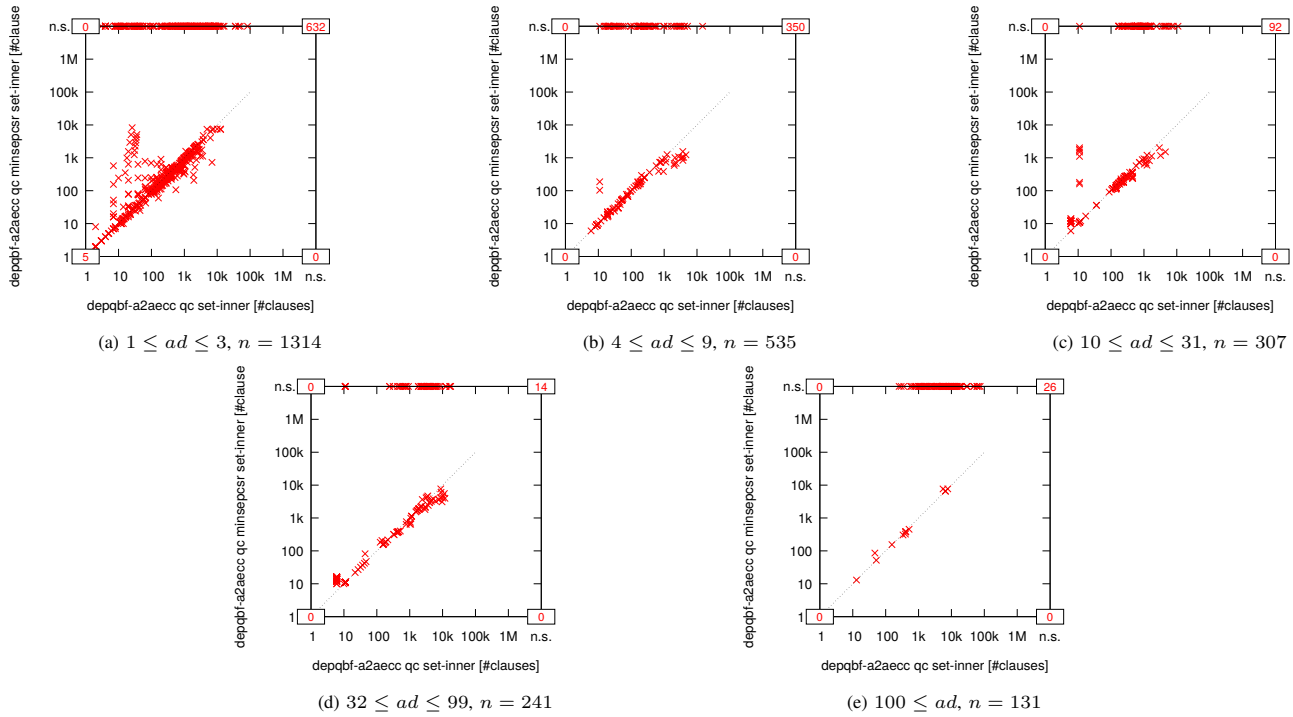


Fig. 398: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode qc minsepcsr set-inner partitioned by alternation depth (number of clauses).

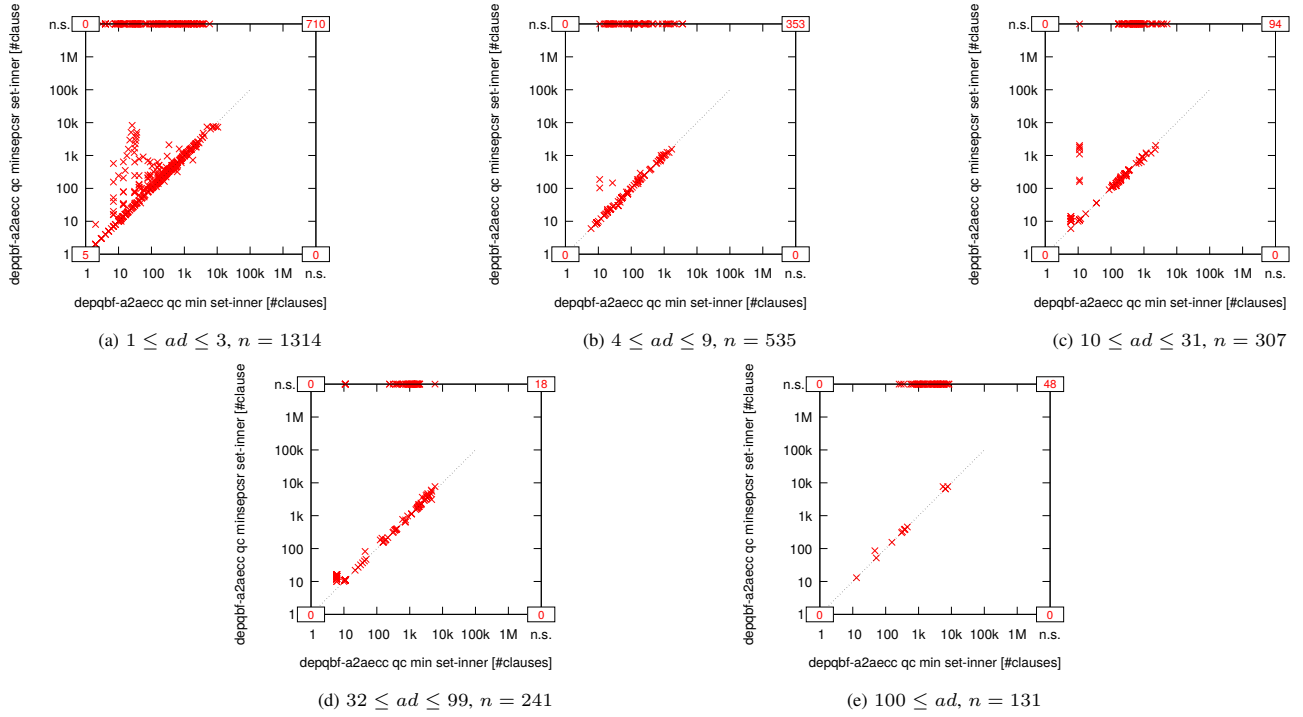


Fig. 399: Comparing sizes of unsatisfiable cores in `DepQBF-a2aecc` in mode `qc min set-inner` versus mode `qc minsepcsr set-inner` partitioned by alternation depth (number of clauses).

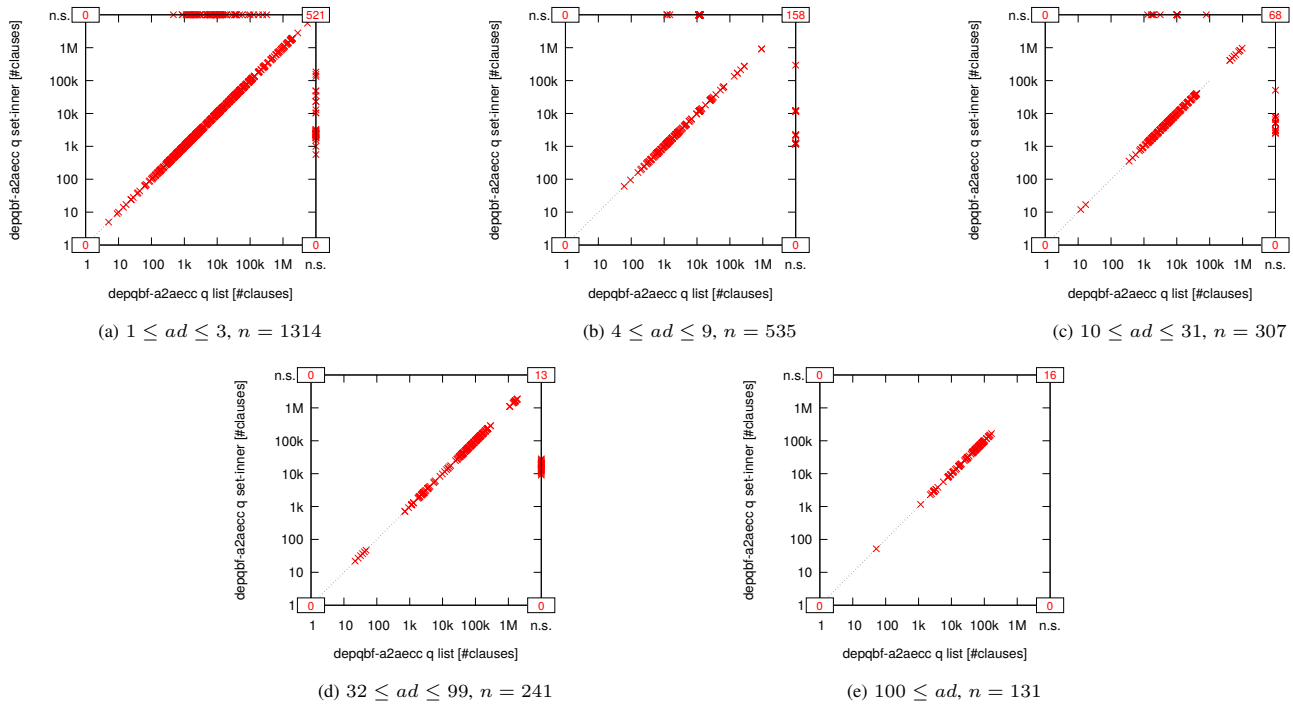


Fig. 400: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode q set-inner partitioned by alternation depth (number of clauses).

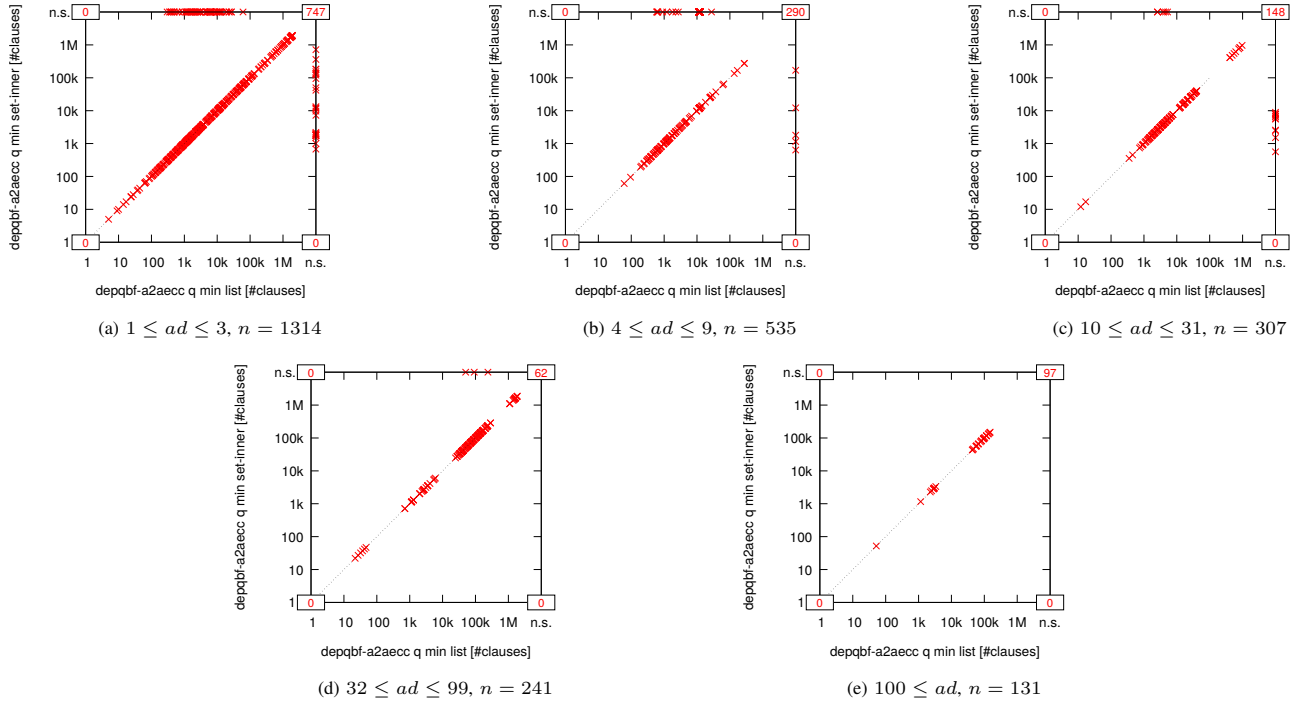


Fig. 401: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q min list versus mode q min set-inner partitioned by alternation depth (number of clauses).

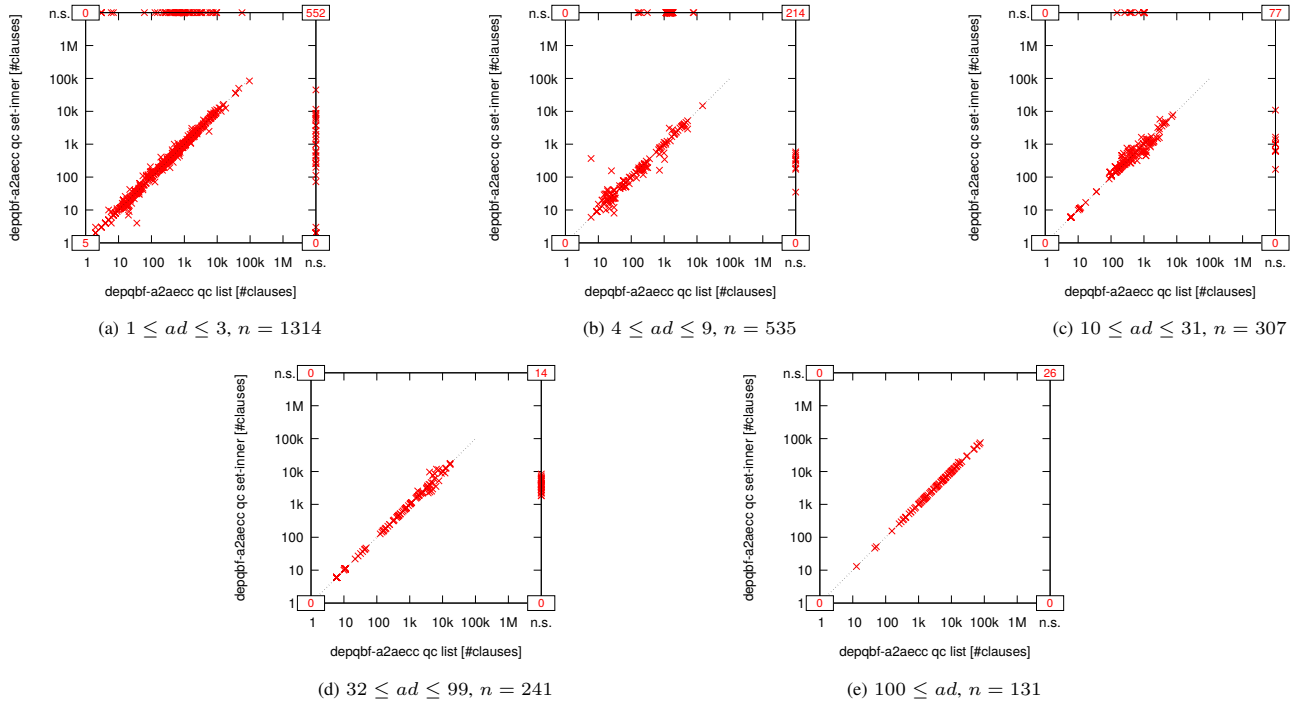


Fig. 402: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode qc set-inner partitioned by alternation depth (number of clauses).

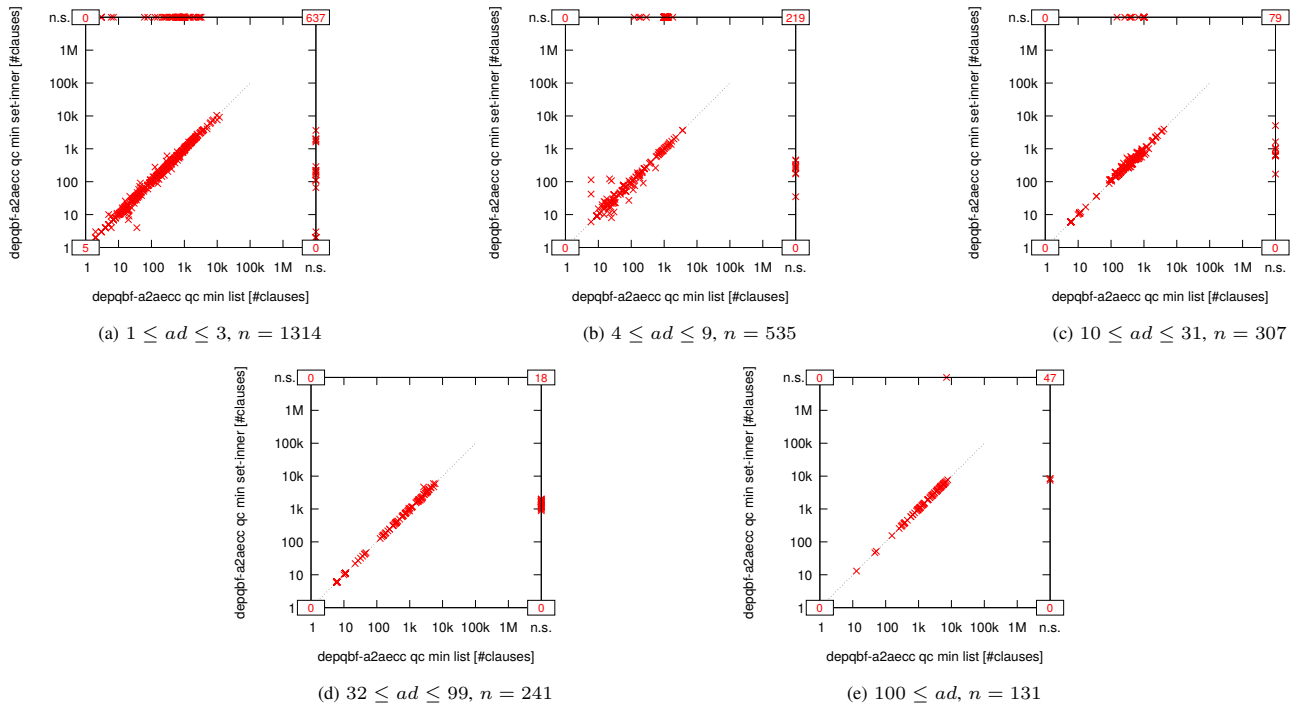


Fig. 403: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode qc min set-inner partitioned by alternation depth (number of clauses).

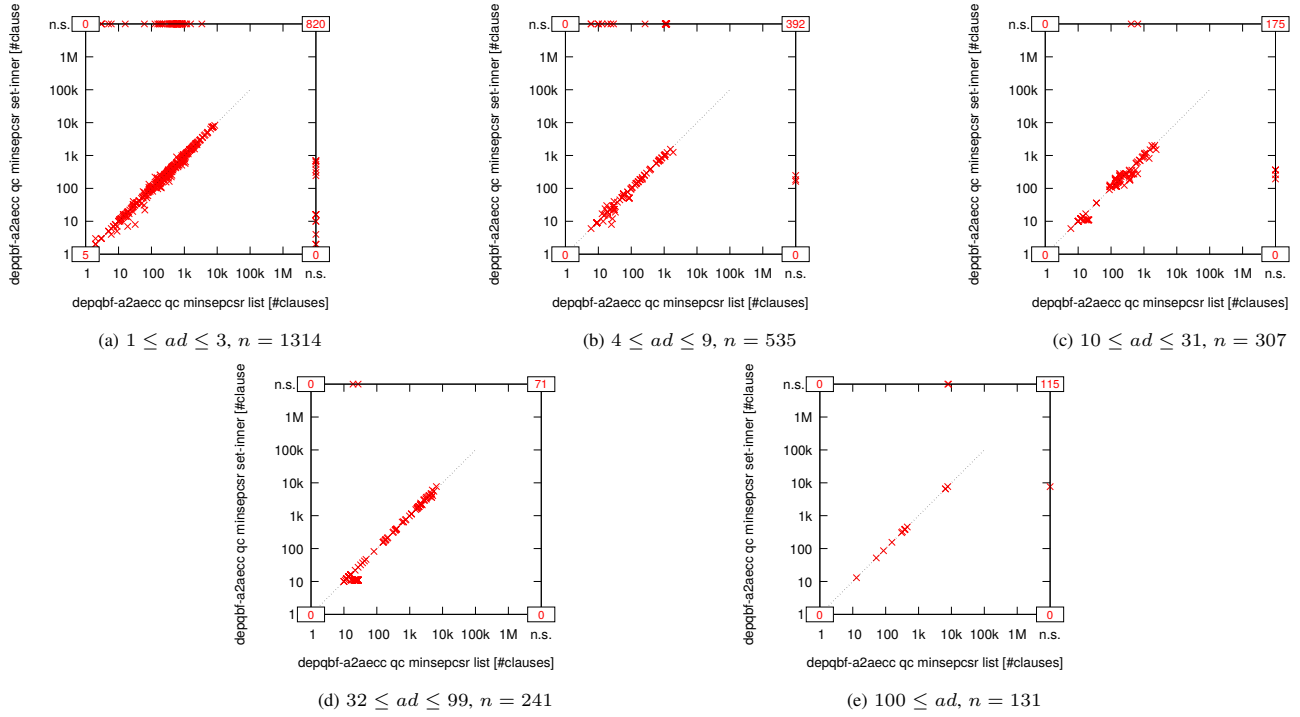


Fig. 404: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode qc minsepcsr set-inner partitioned by alternation depth (number of clauses).

D. Partitioned by Number of Clauses

This subsection shows figures of the sizes of unsatisfiable cores with subfigures for partitions of the benchmarks according to their number of clauses.

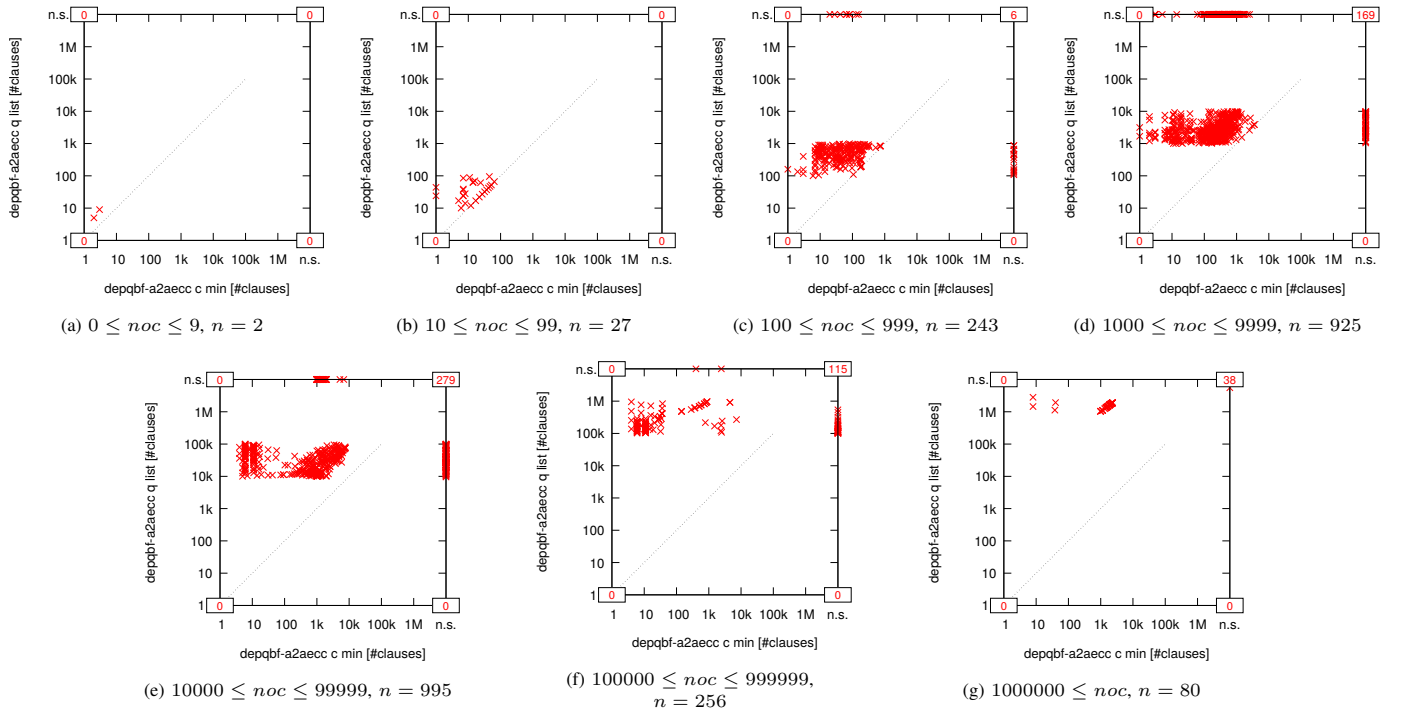


Fig. 405: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q list partitioned by number of clauses (number of clauses).

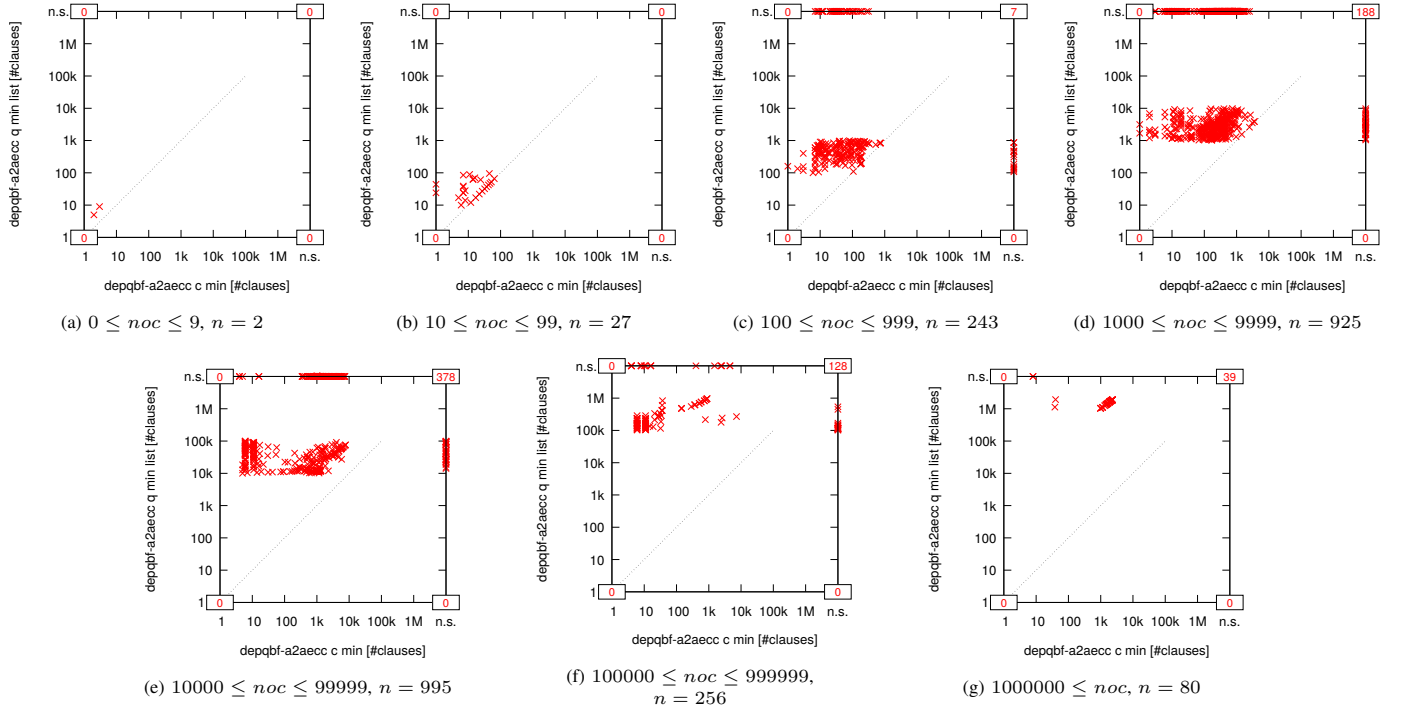


Fig. 406: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q min list partitioned by number of clauses (number of clauses).

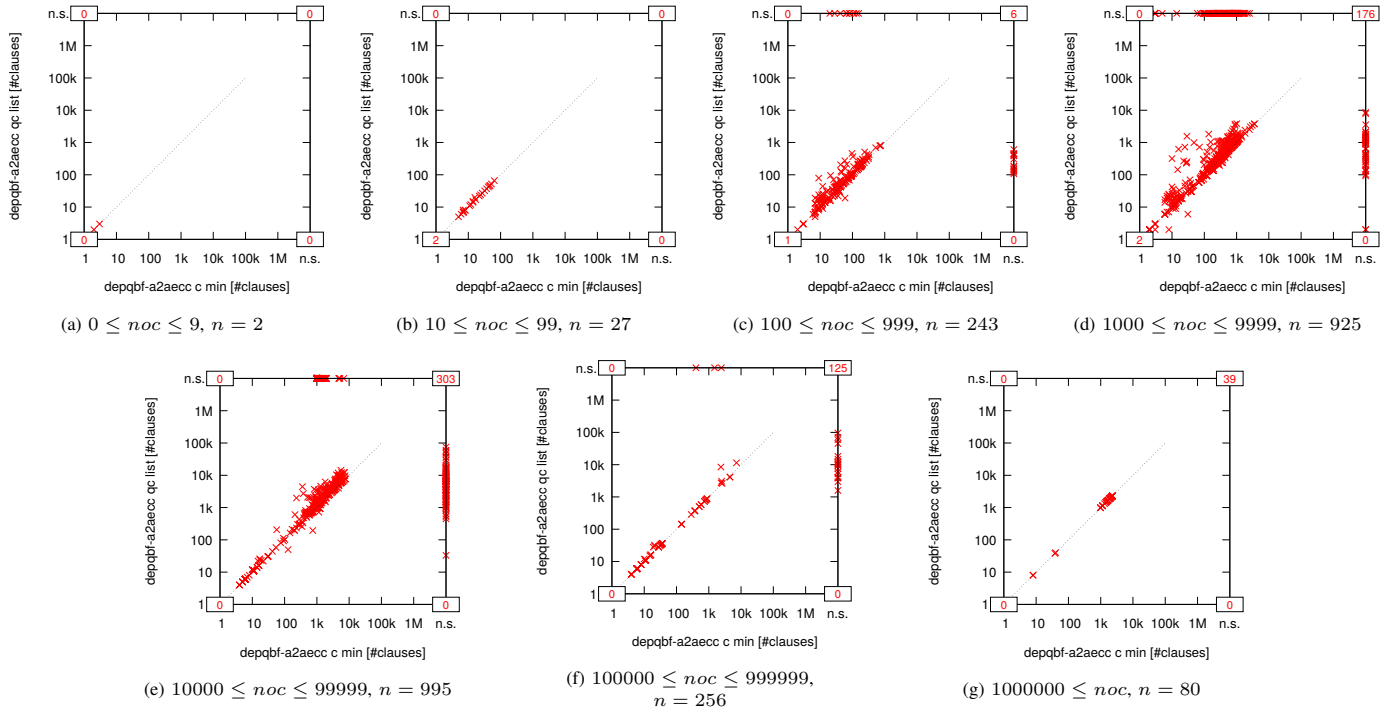


Fig. 407: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc list partitioned by number of clauses (number of clauses).

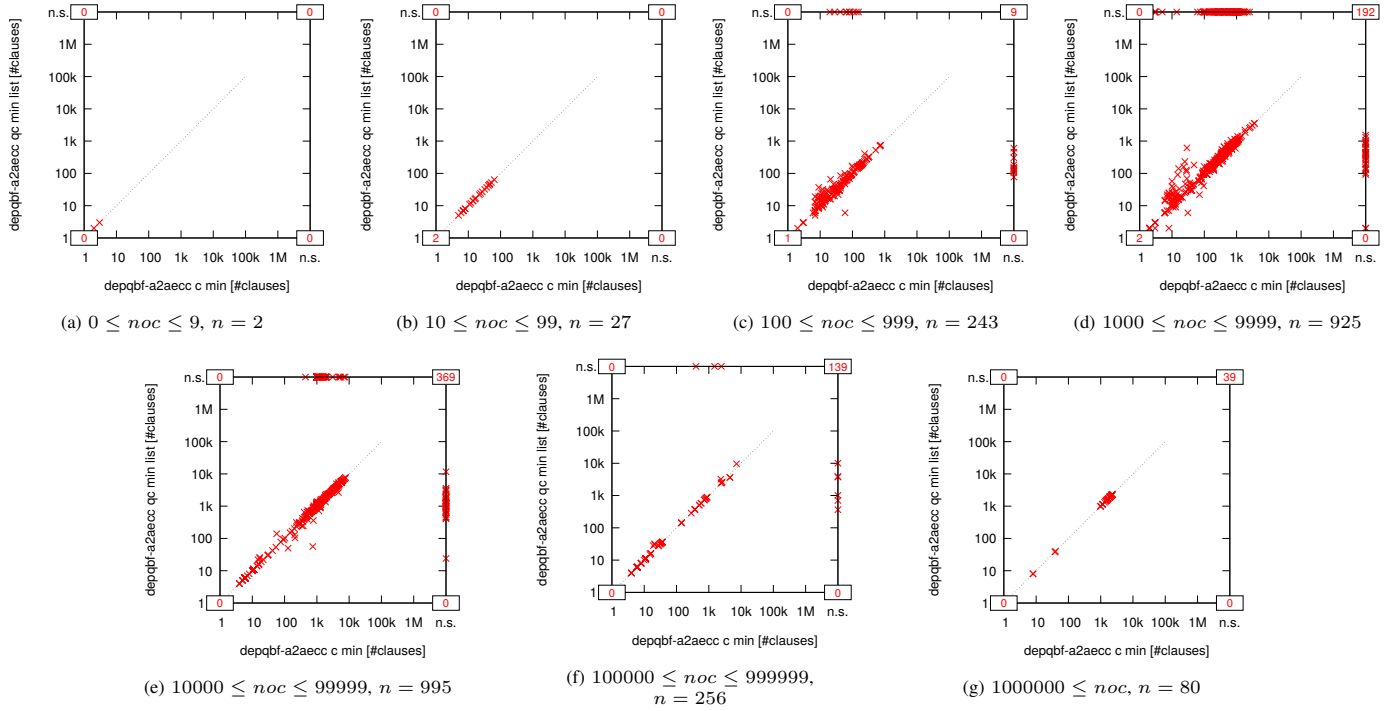


Fig. 408: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc min list partitioned by number of clauses (number of clauses).

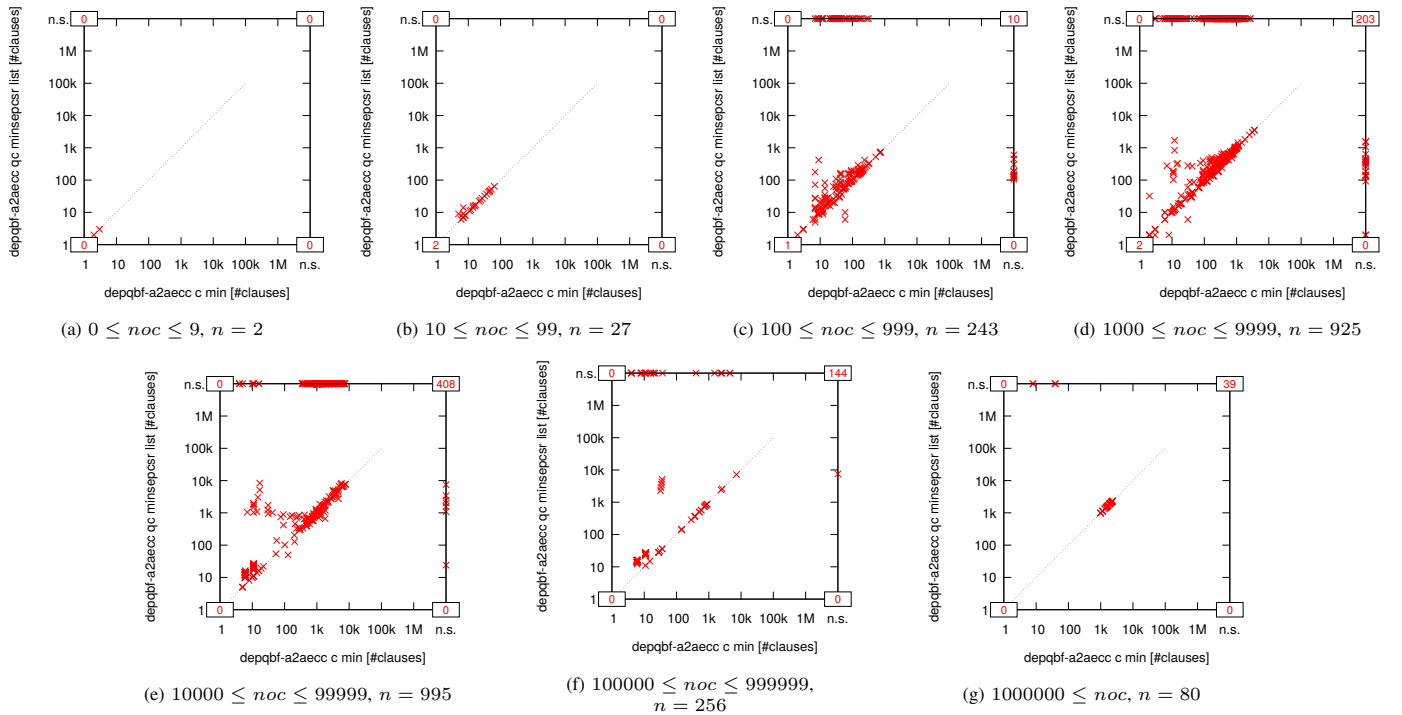


Fig. 409: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc minsepcsr list partitioned by number of clauses (number of clauses).

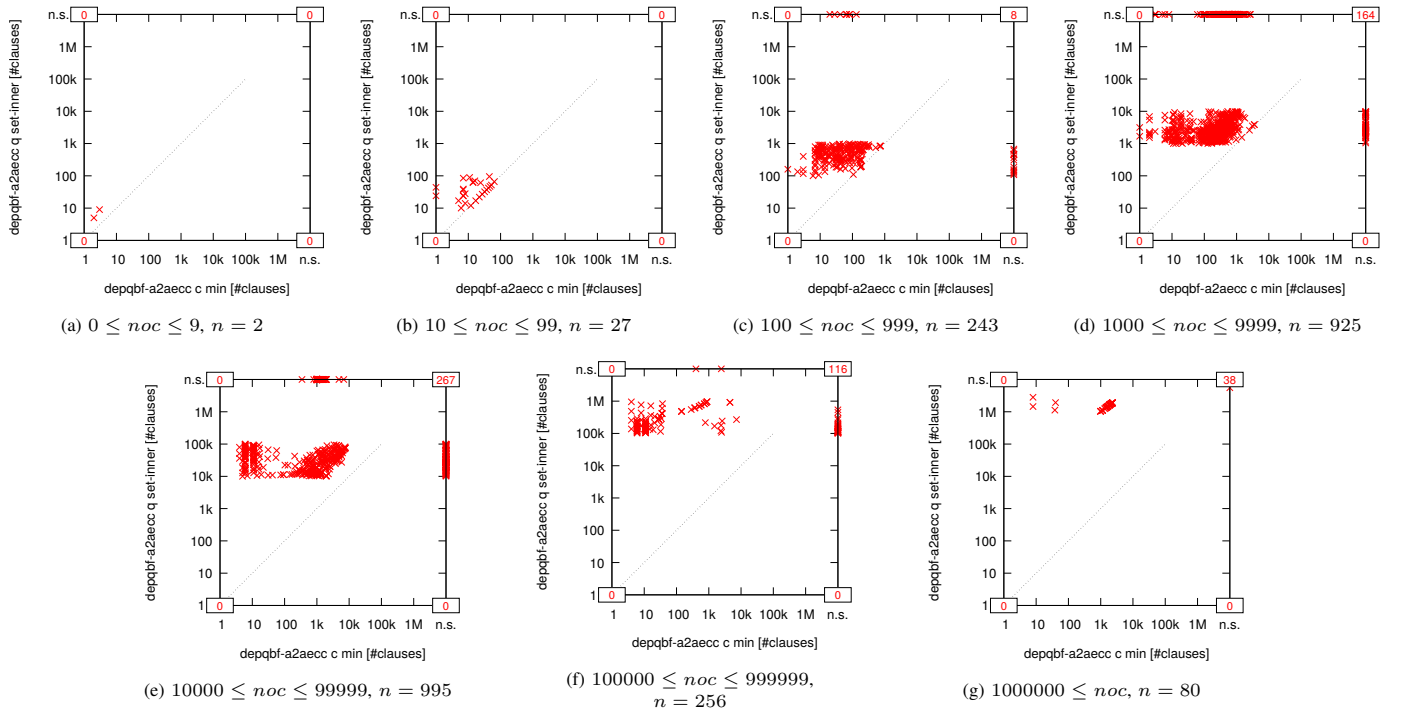


Fig. 410: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q set-inner partitioned by number of clauses (number of clauses).

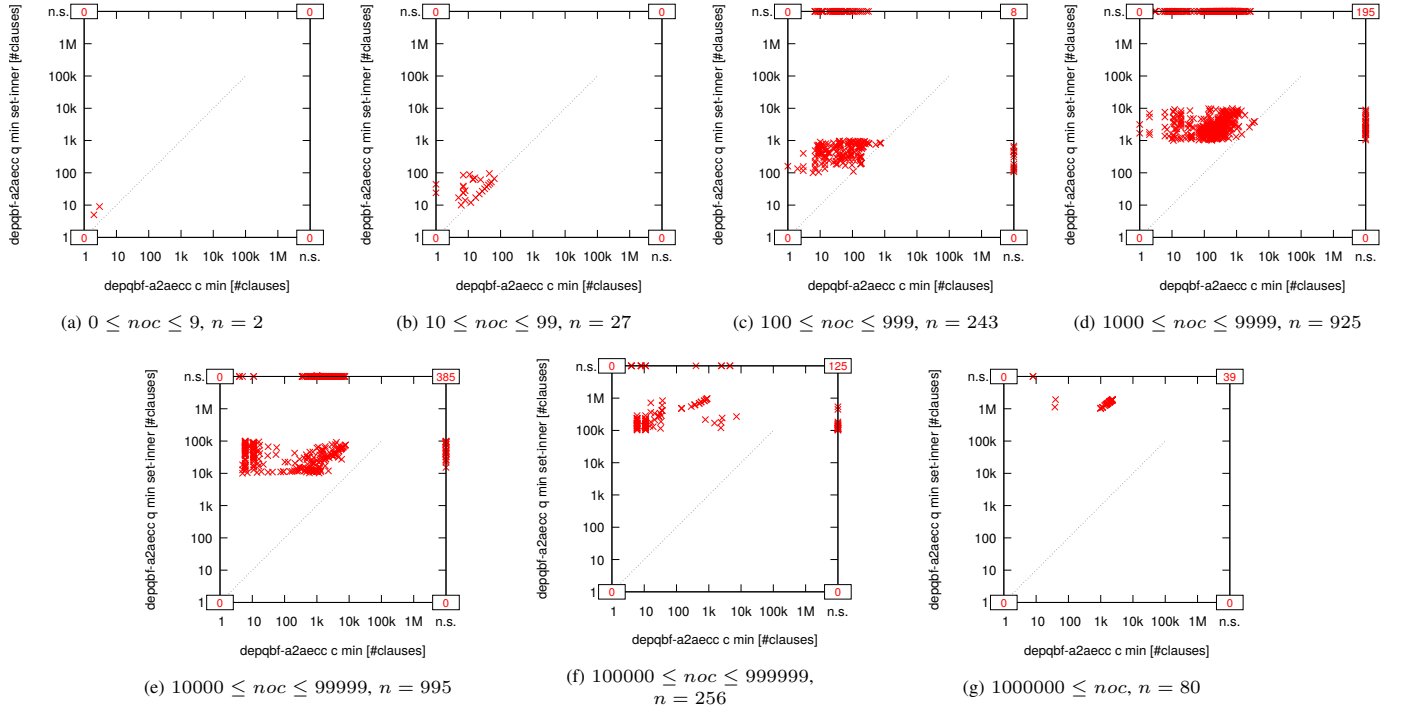


Fig. 411: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q min set-inner partitioned by number of clauses (number of clauses).

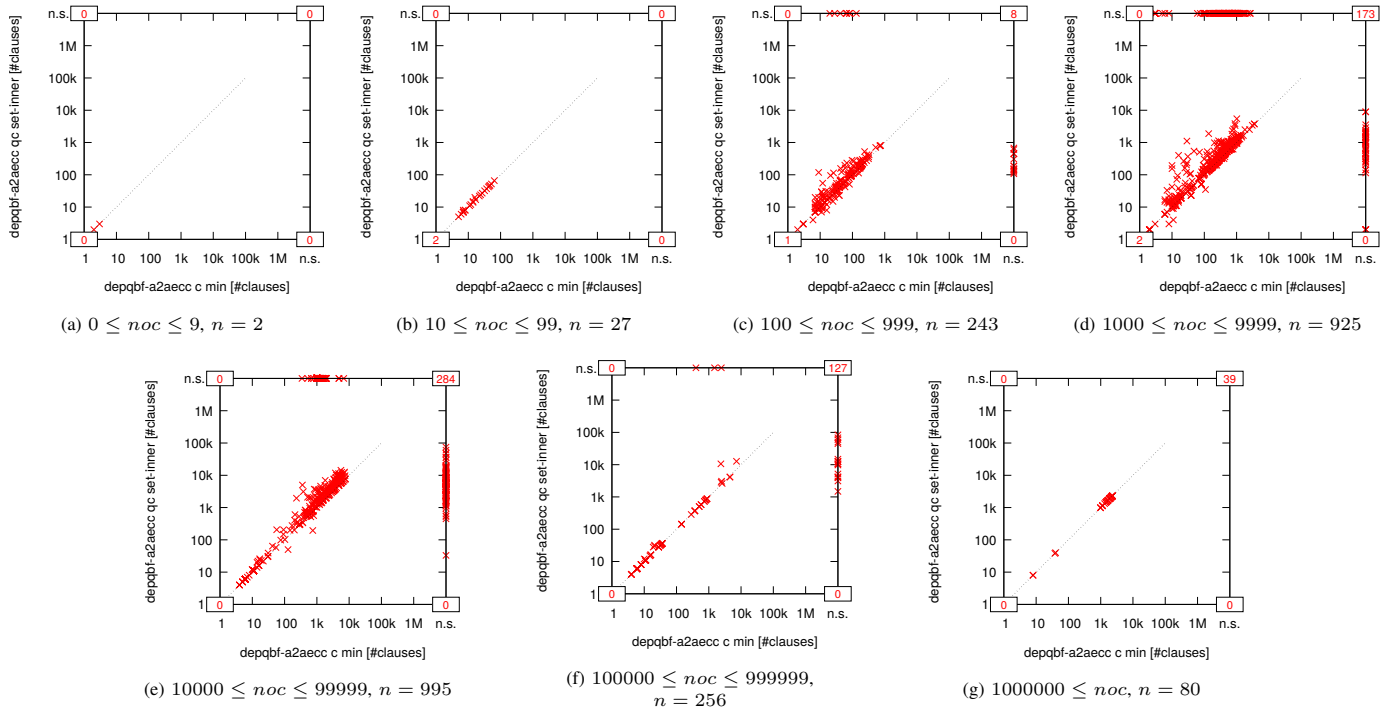


Fig. 412: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc set-inner partitioned by number of clauses (number of clauses).

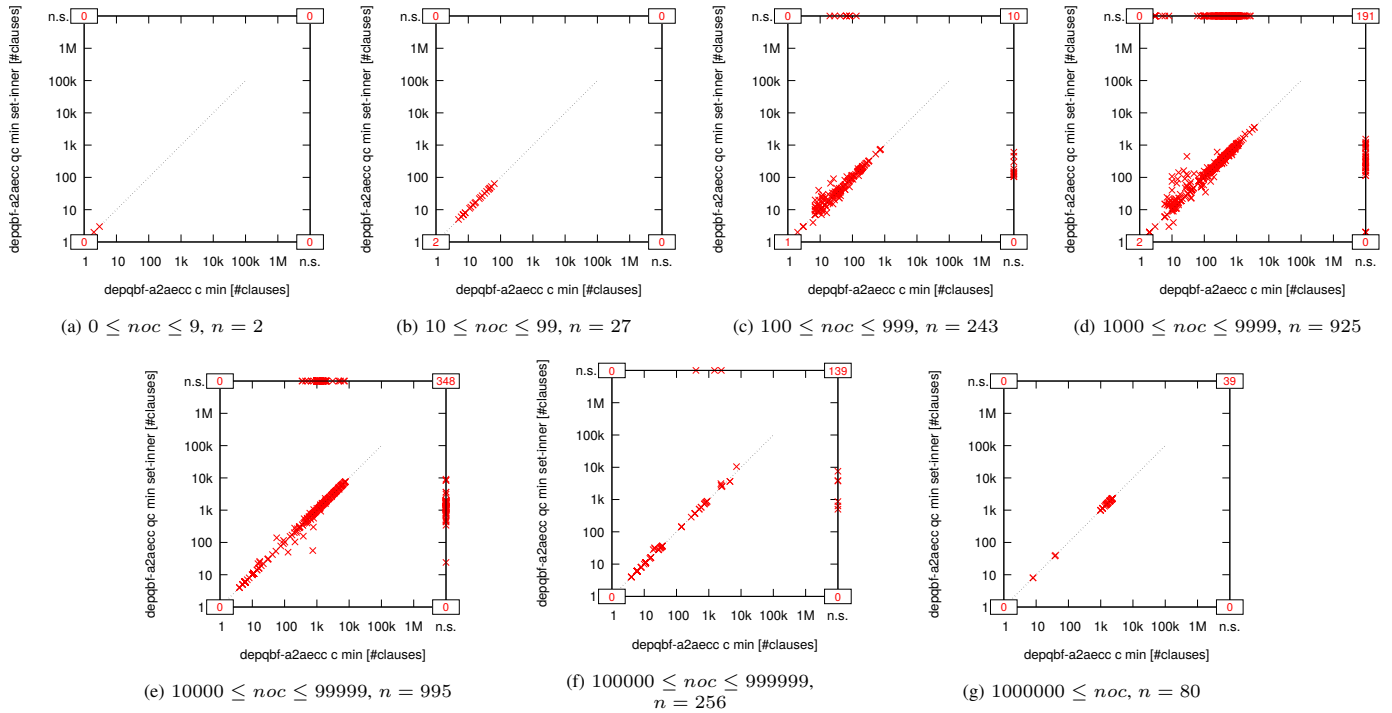


Fig. 413: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc min set-inner partitioned by number of clauses (number of clauses).

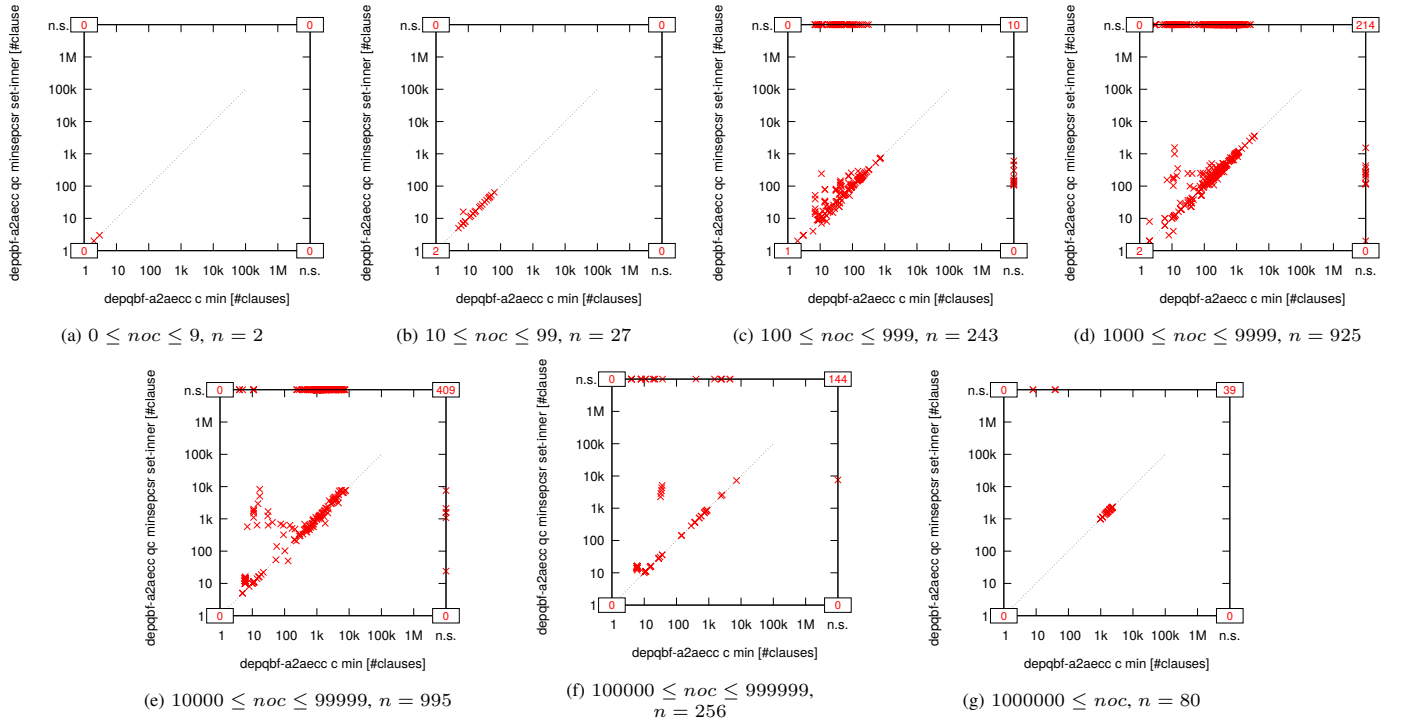


Fig. 414: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc minsepcsr set-inner partitioned by number of clauses (number of clauses).

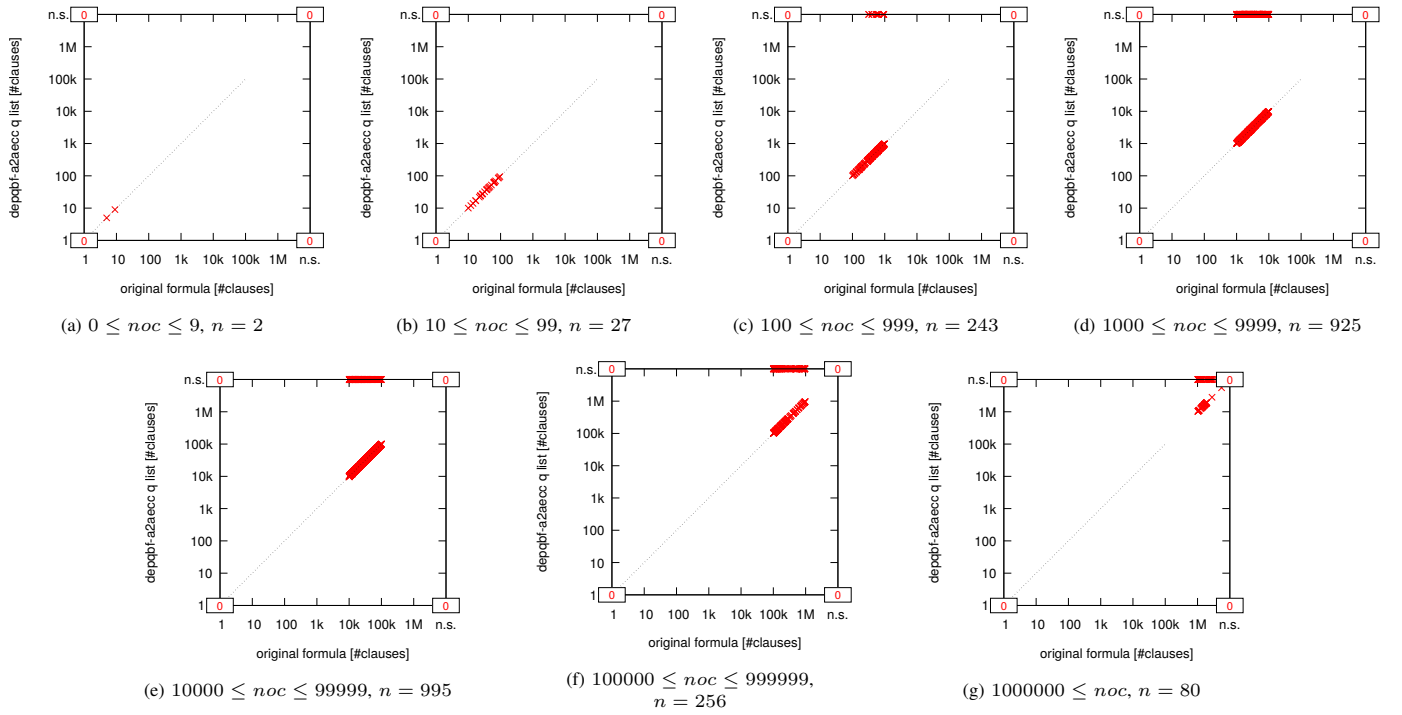


Fig. 415: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q list partitioned by number of clauses (number of clauses).

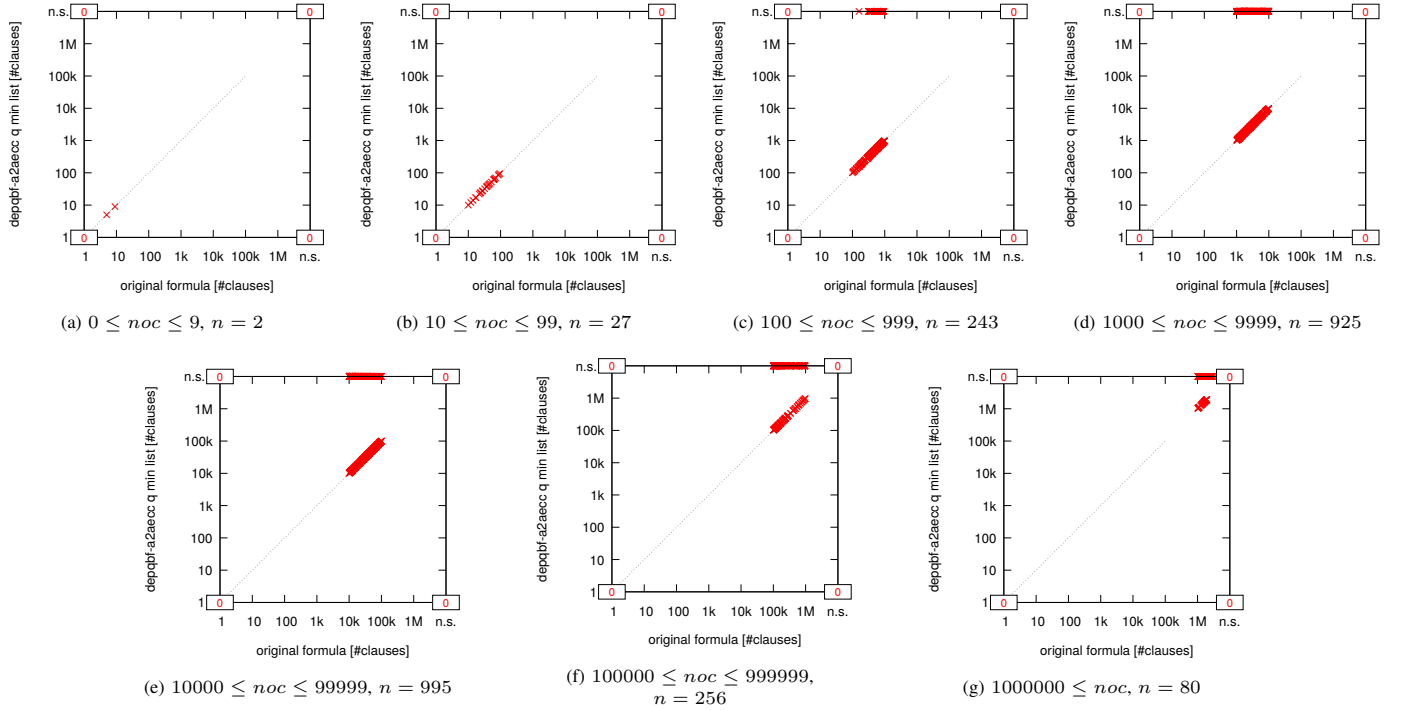


Fig. 416: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q min list partitioned by number of clauses (number of clauses).

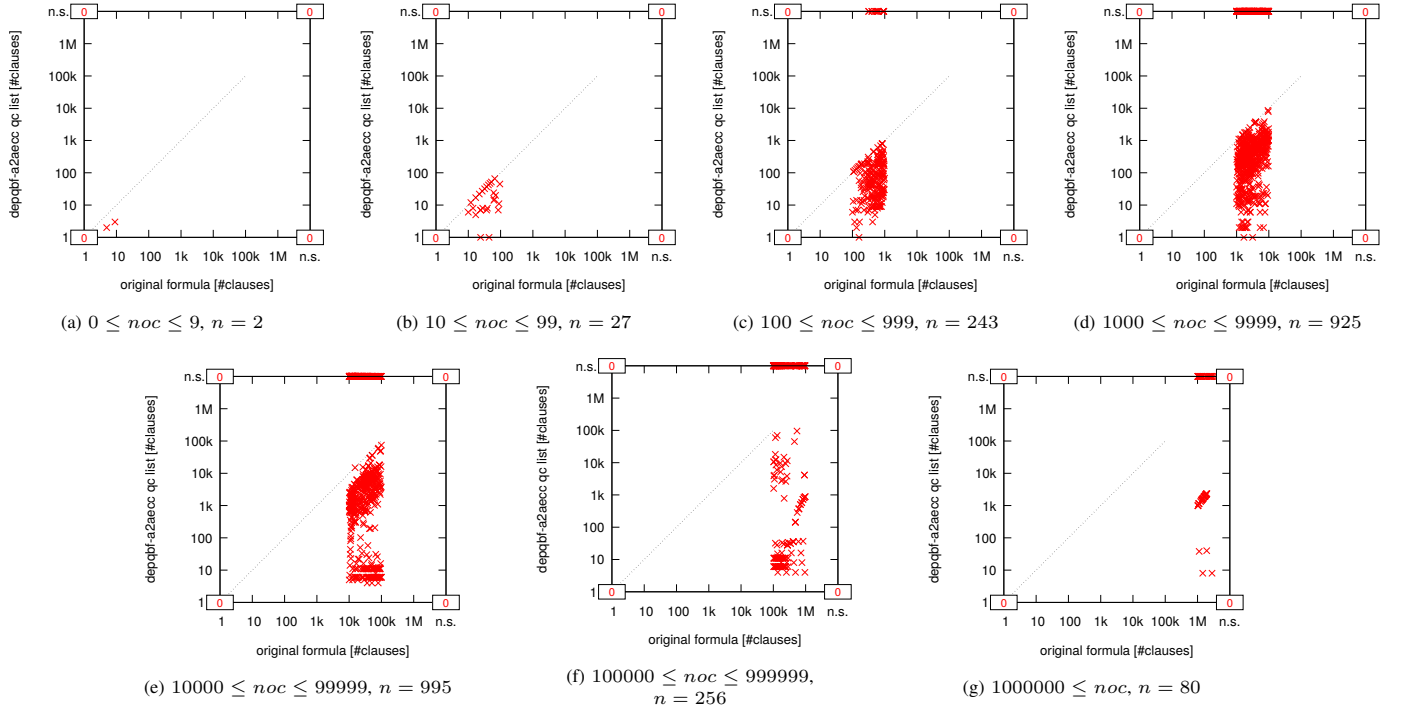


Fig. 417: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc list partitioned by number of clauses (number of clauses).

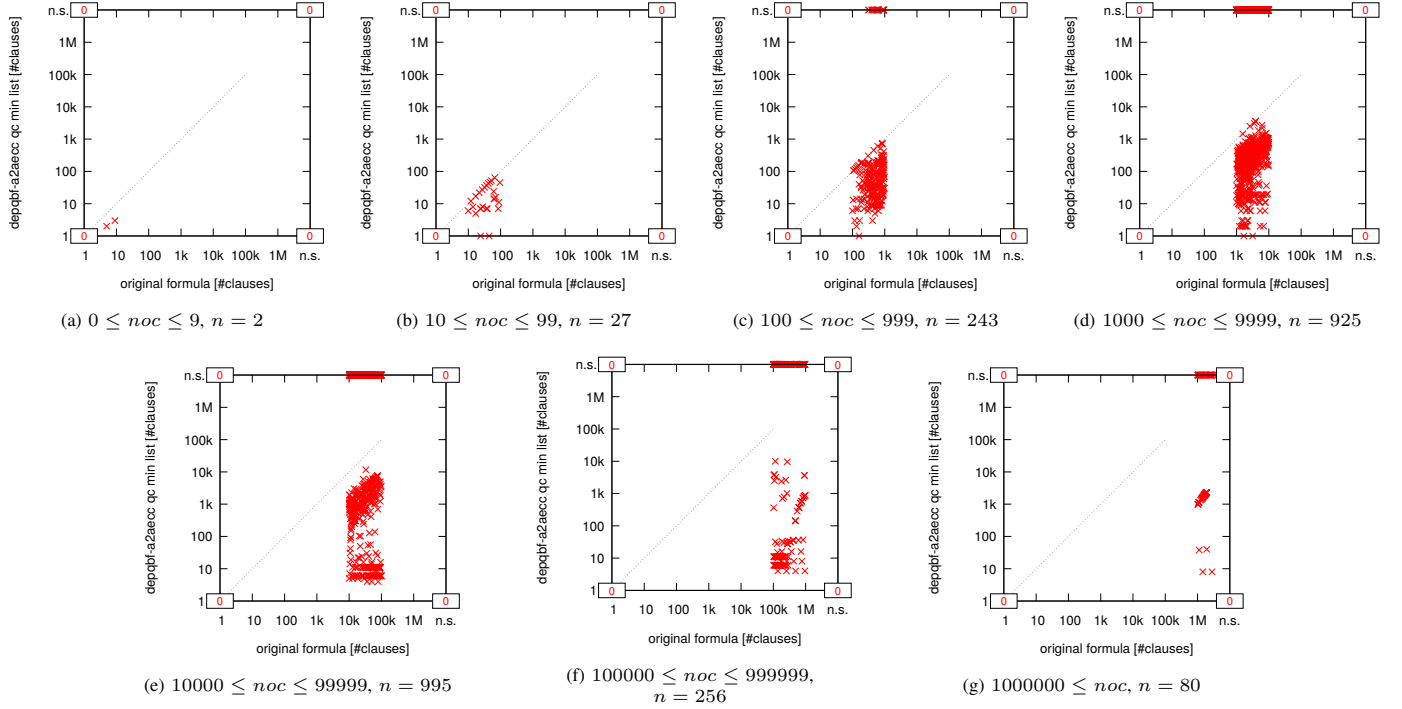


Fig. 418: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc min list partitioned by number of clauses (number of clauses).

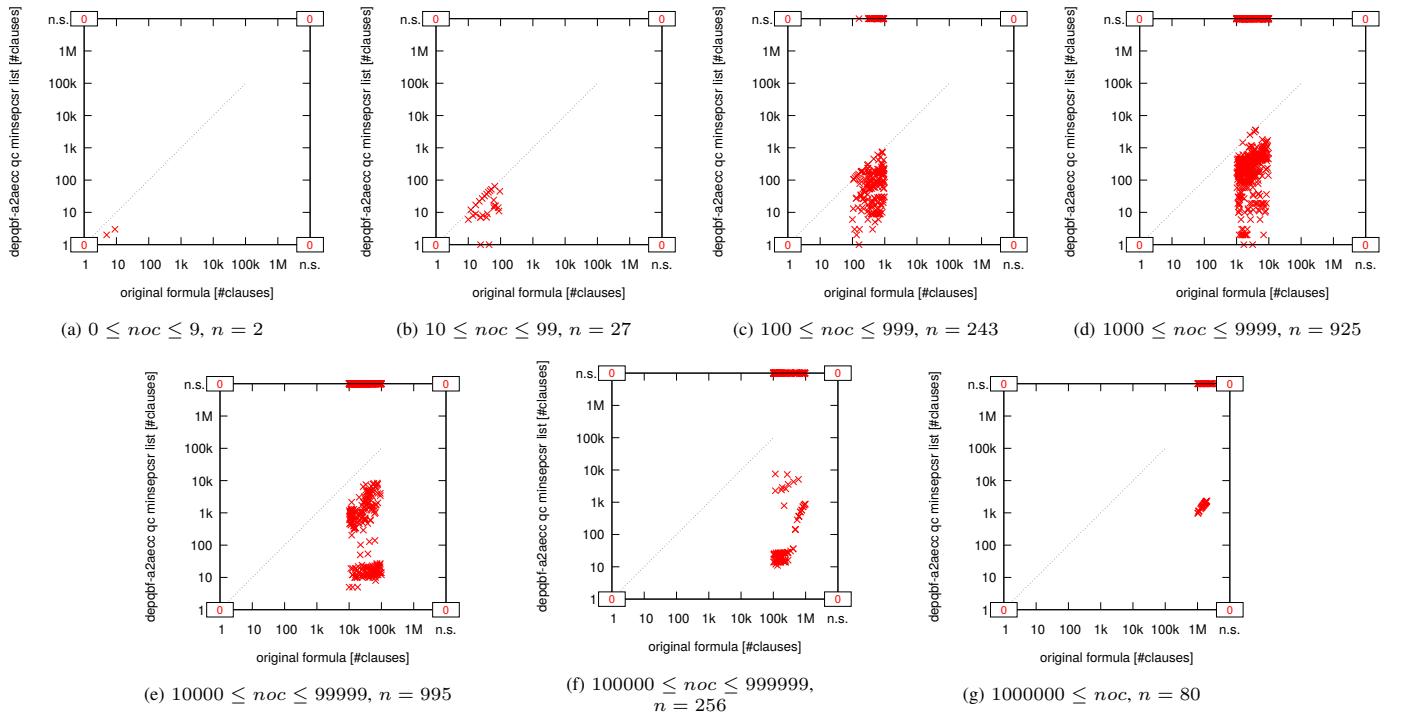


Fig. 419: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc minsepcsr list partitioned by number of clauses (number of clauses).

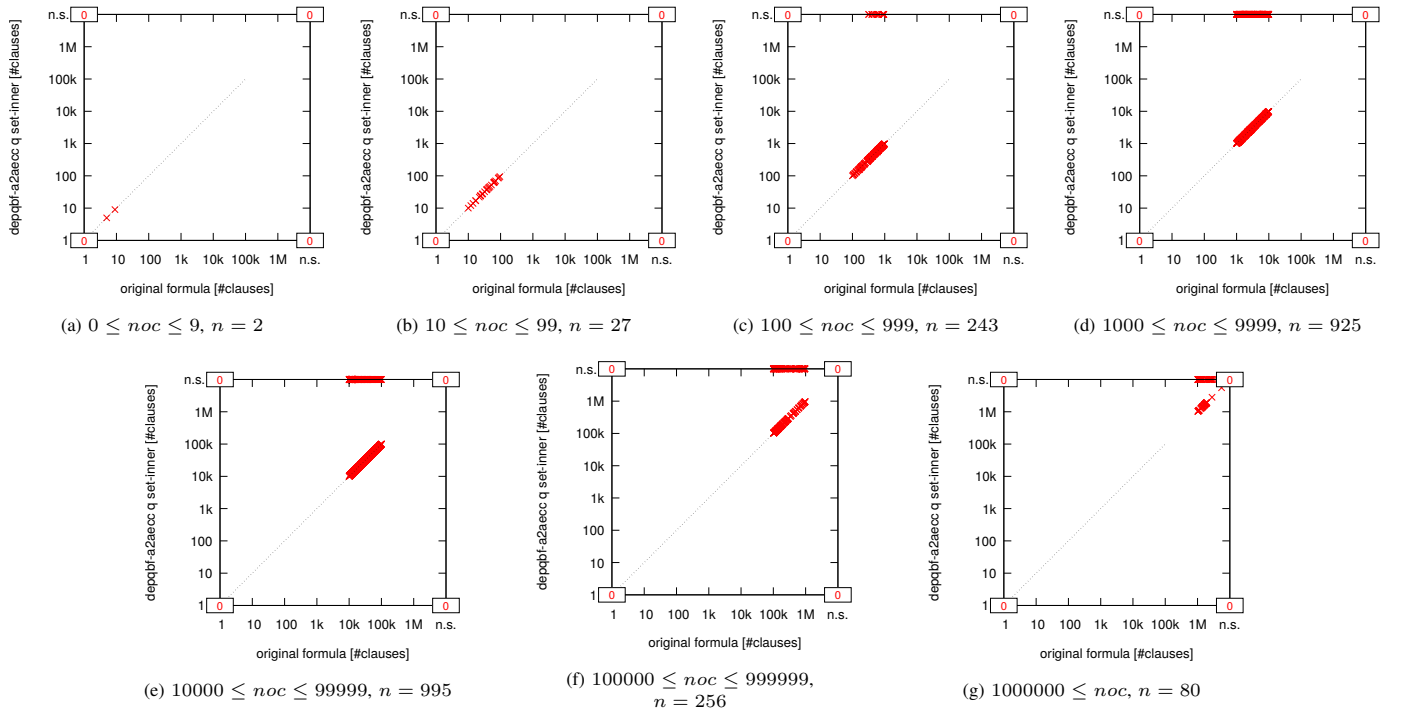


Fig. 420: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q set-inner partitioned by number of clauses (number of clauses).

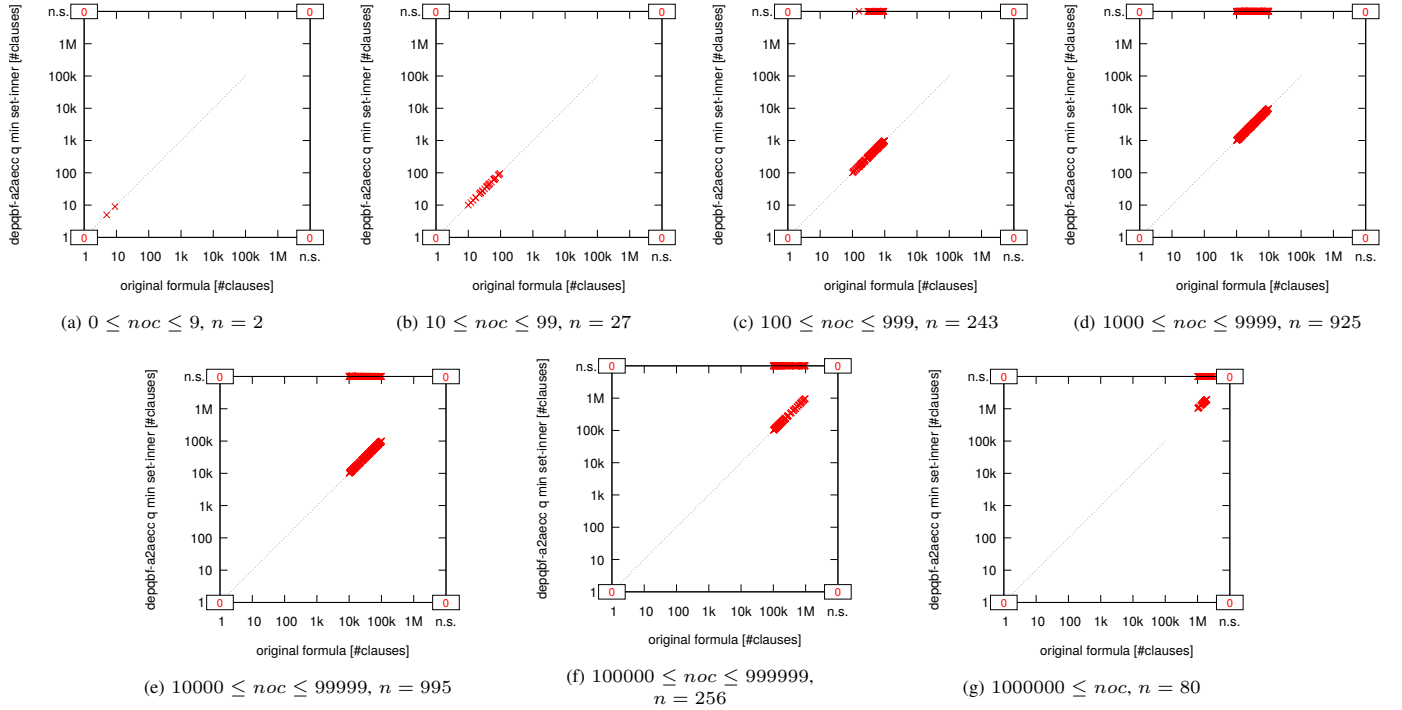


Fig. 421: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q min set-inner partitioned by number of clauses (number of clauses).

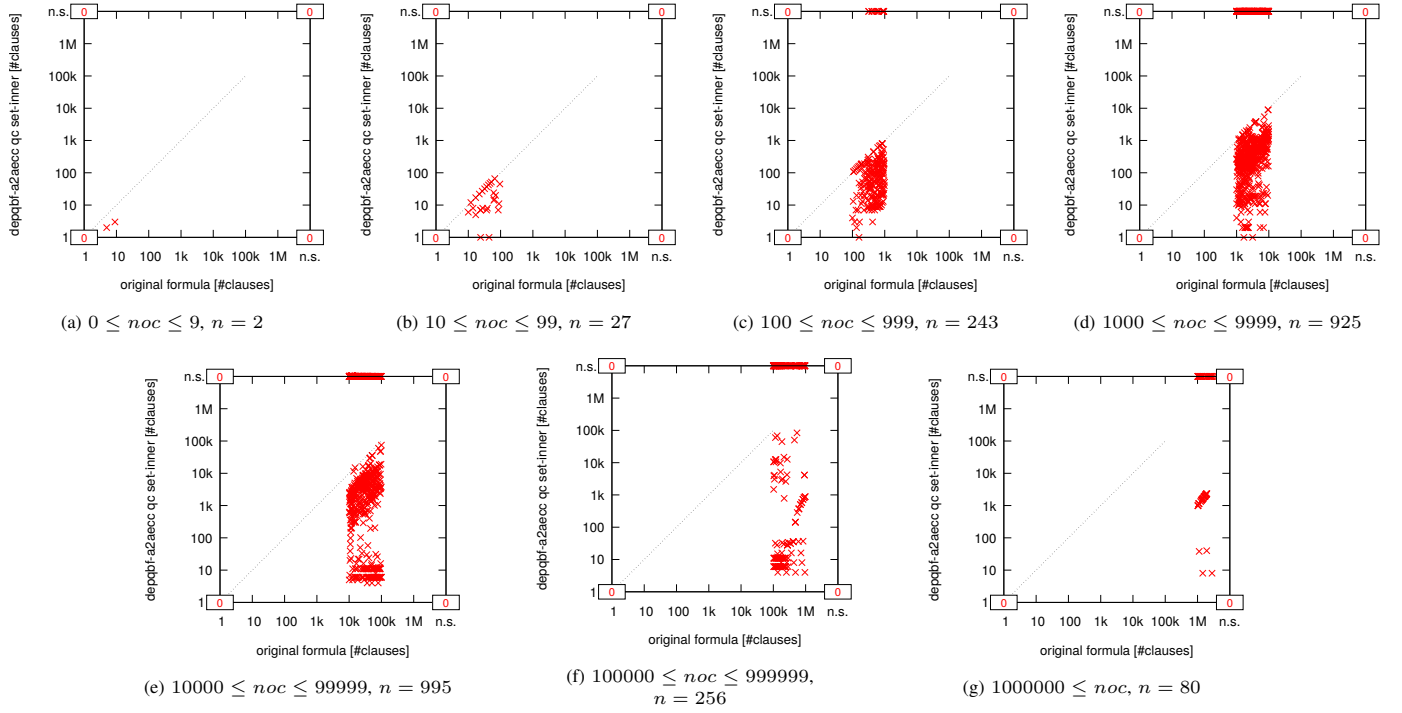


Fig. 422: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc set-inner partitioned by number of clauses (number of clauses).

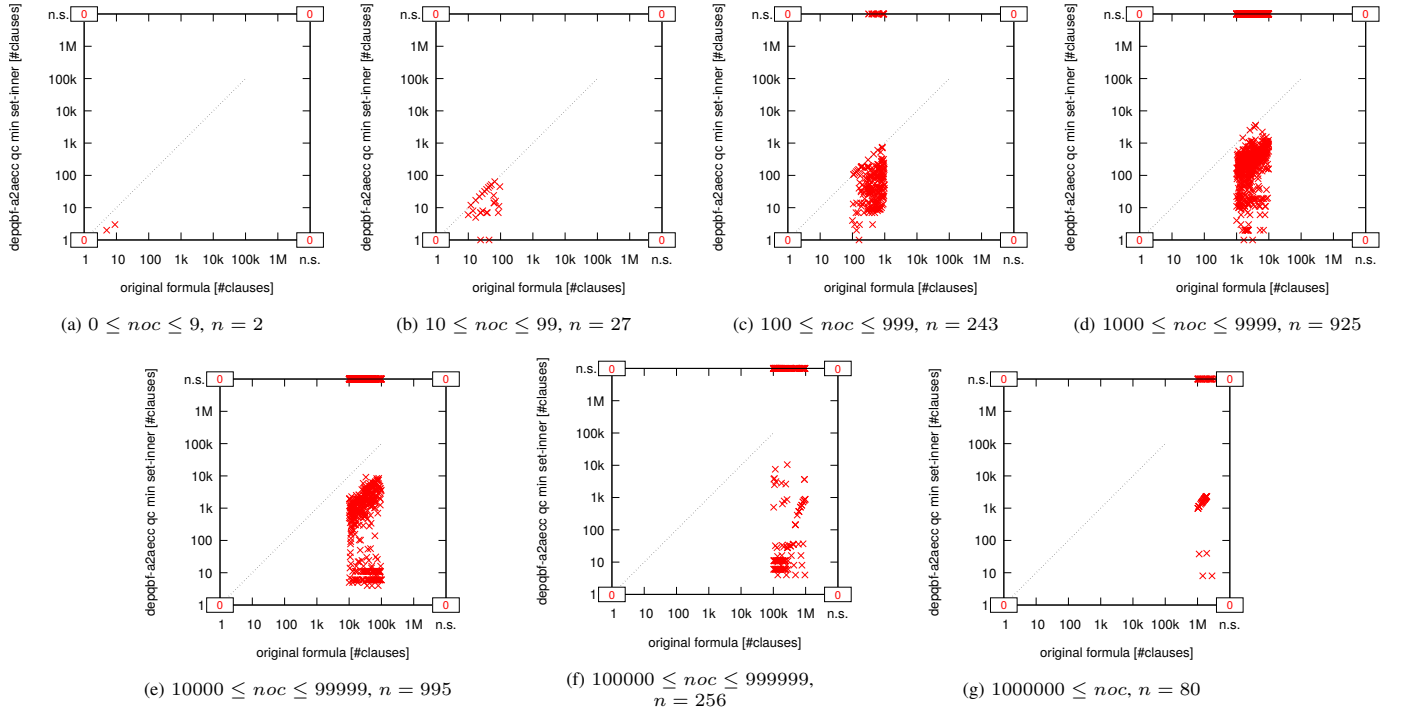


Fig. 423: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc min set-inner partitioned by number of clauses (number of clauses).

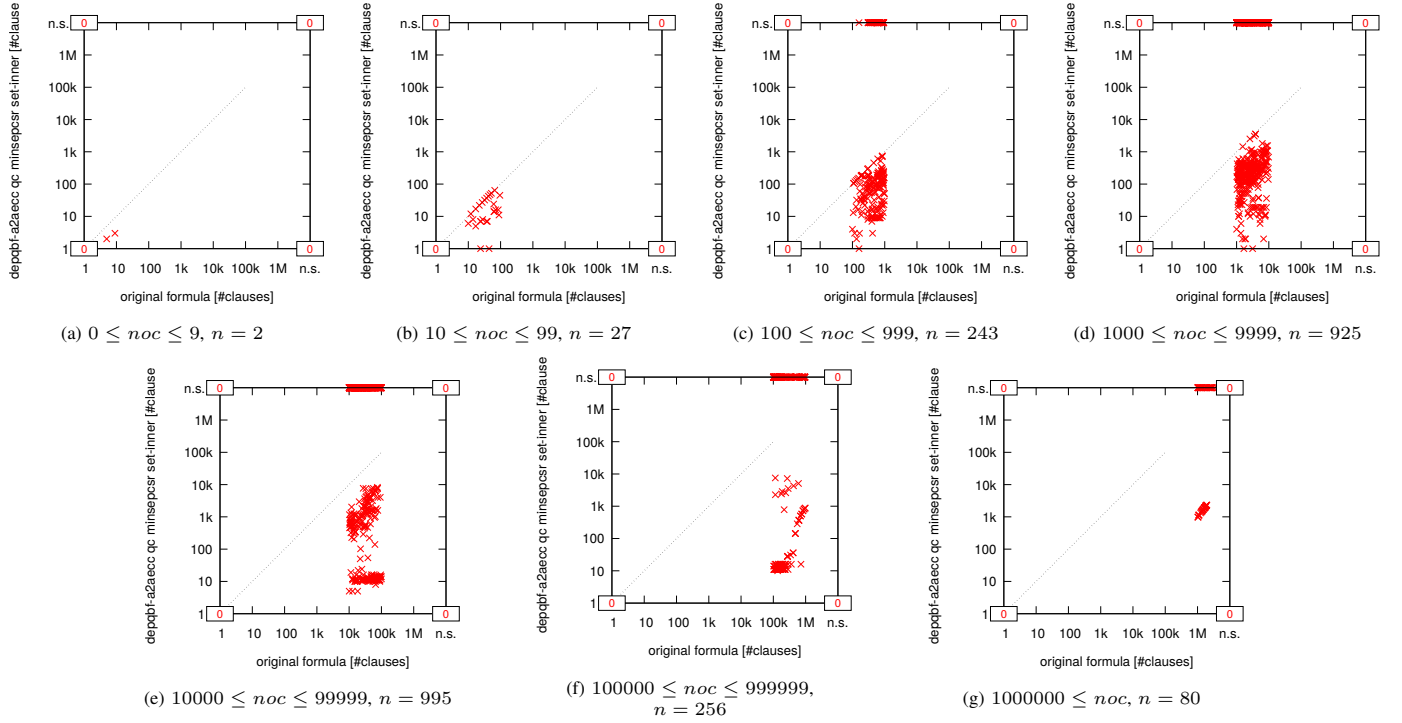


Fig. 424: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc minsepcsr set-inner partitioned by number of clauses (number of clauses).

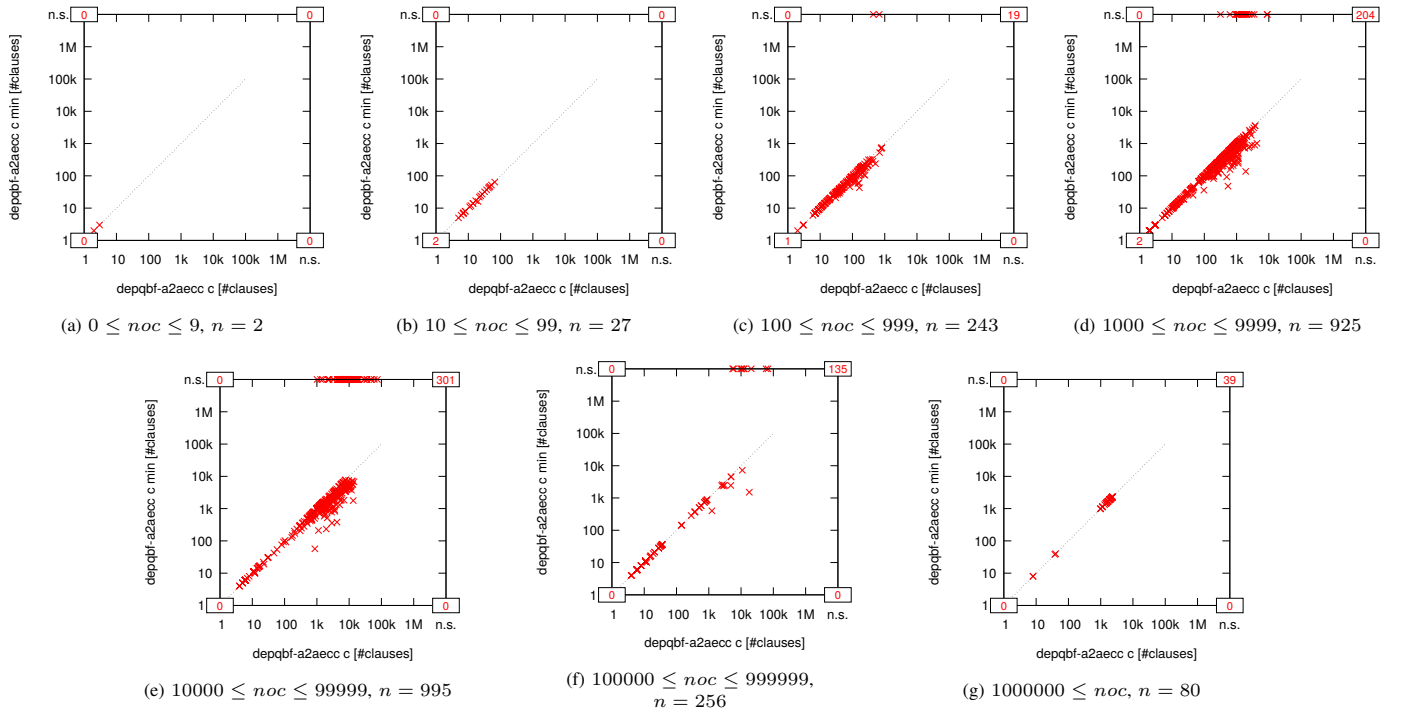


Fig. 425: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c versus mode c min partitioned by number of clauses (number of clauses).

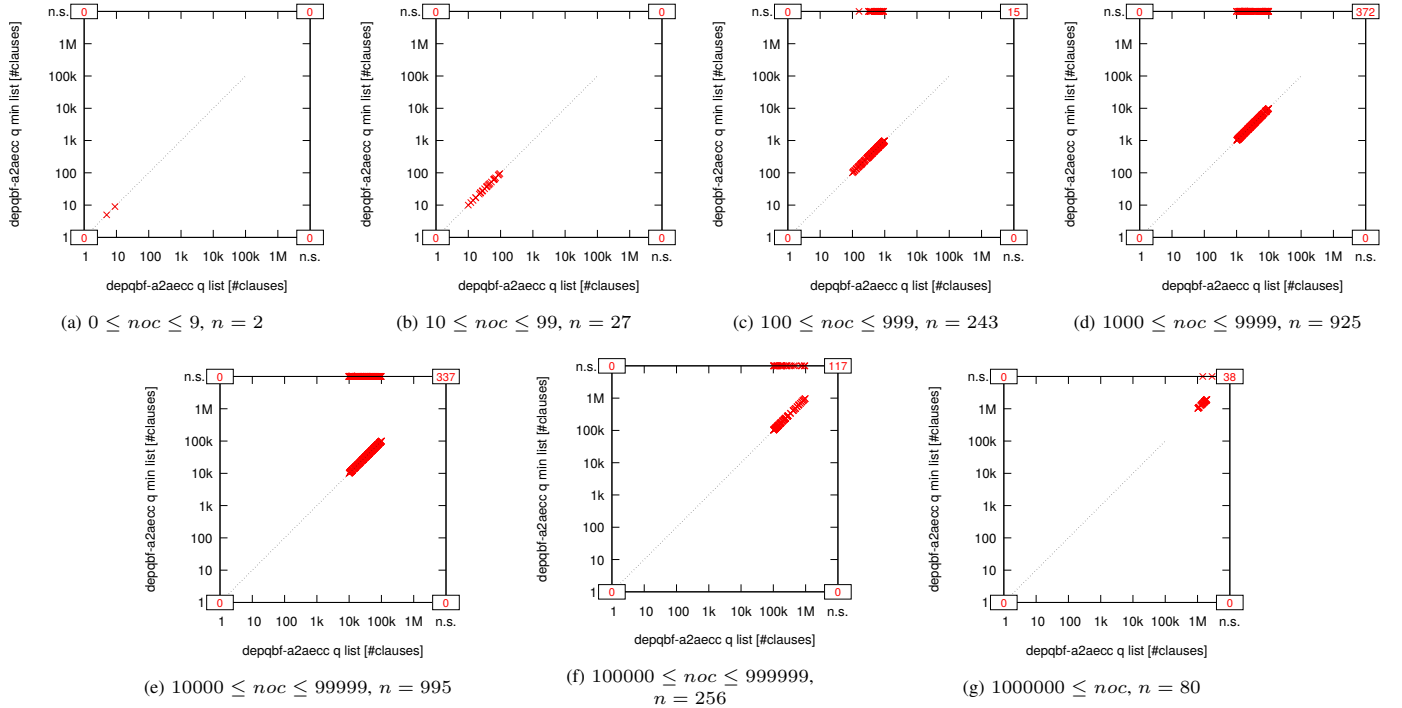


Fig. 426: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode q min list partitioned by number of clauses (number of clauses).

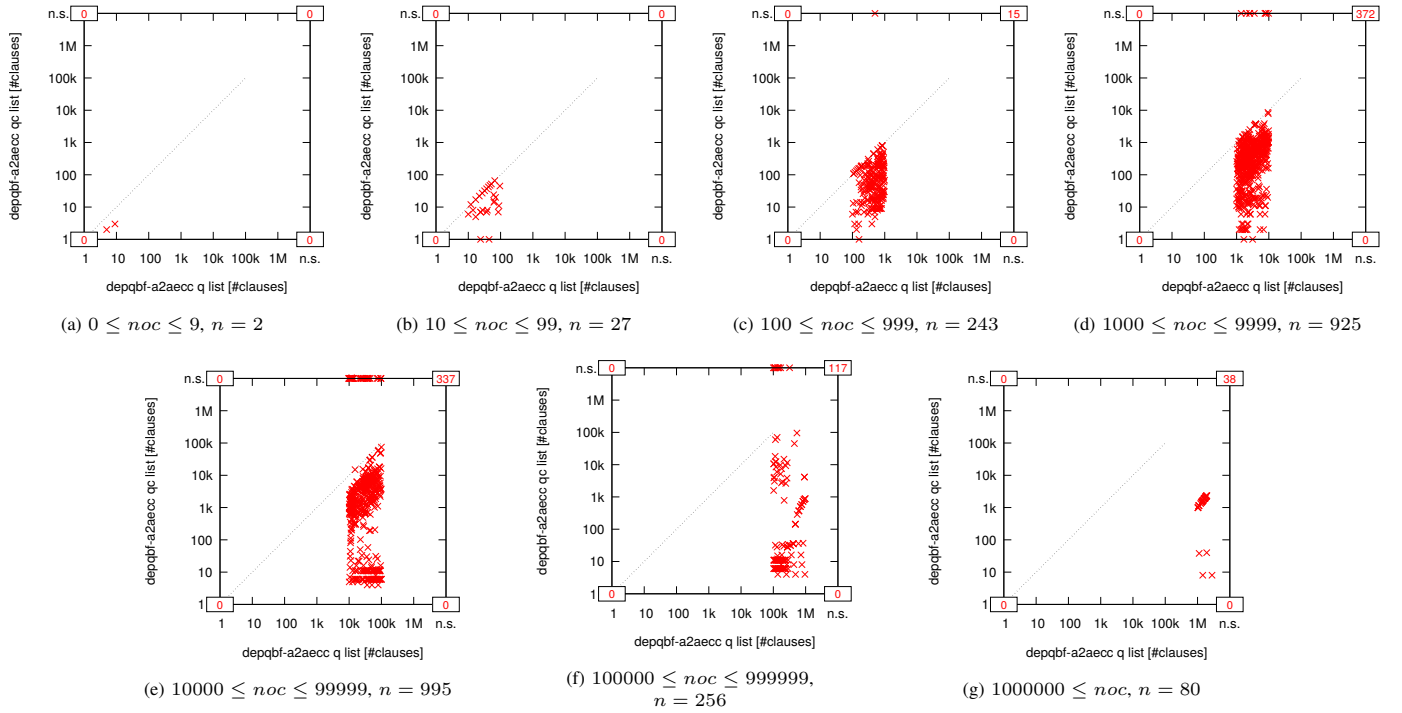


Fig. 427: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode qc list partitioned by number of clauses (number of clauses).

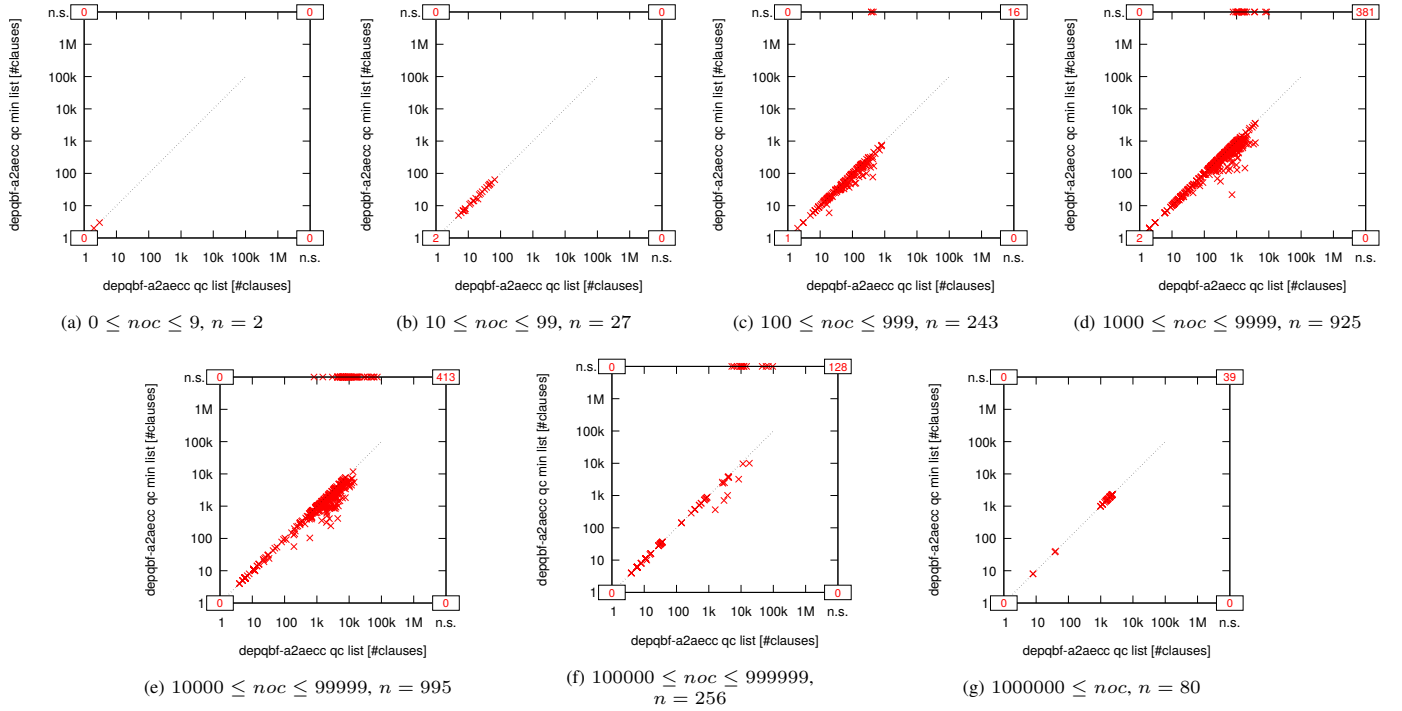


Fig. 428: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode qc min list partitioned by number of clauses (number of clauses).

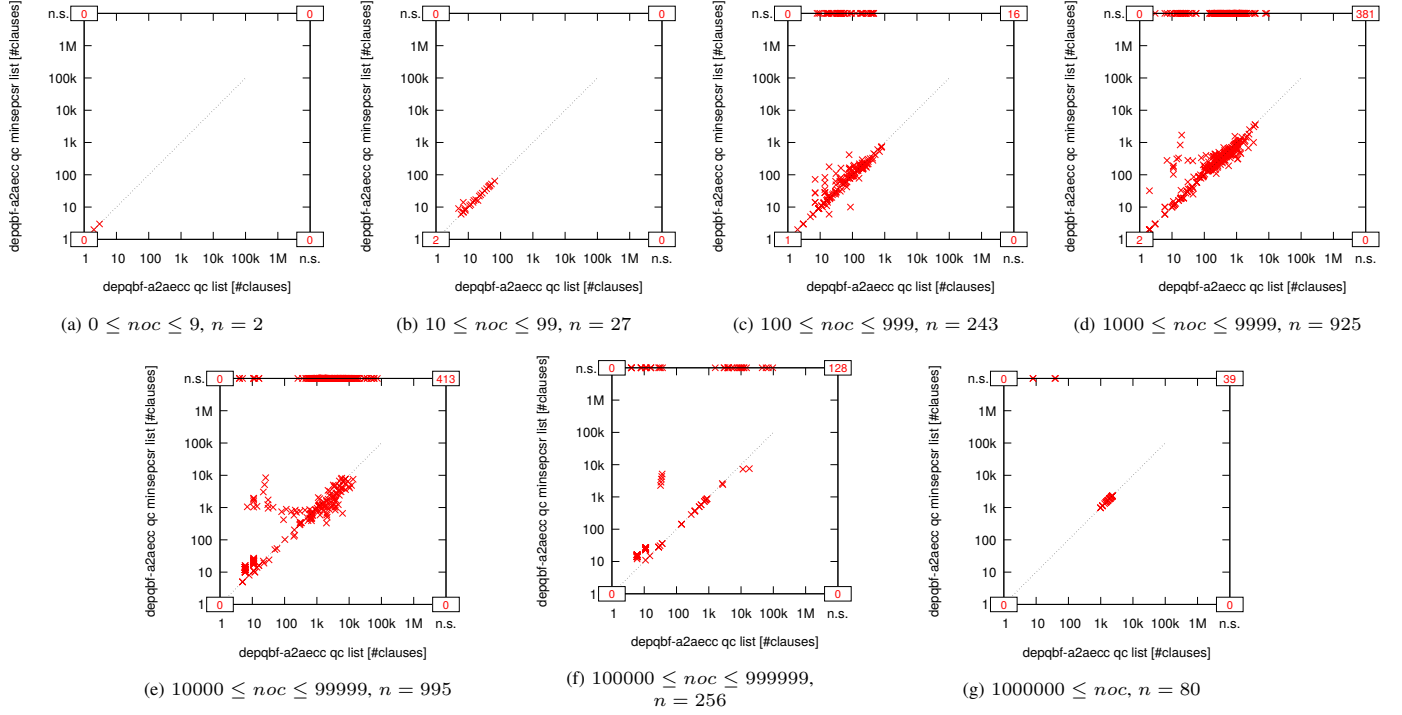


Fig. 429: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode qc minsepcsr list partitioned by number of clauses (number of clauses).

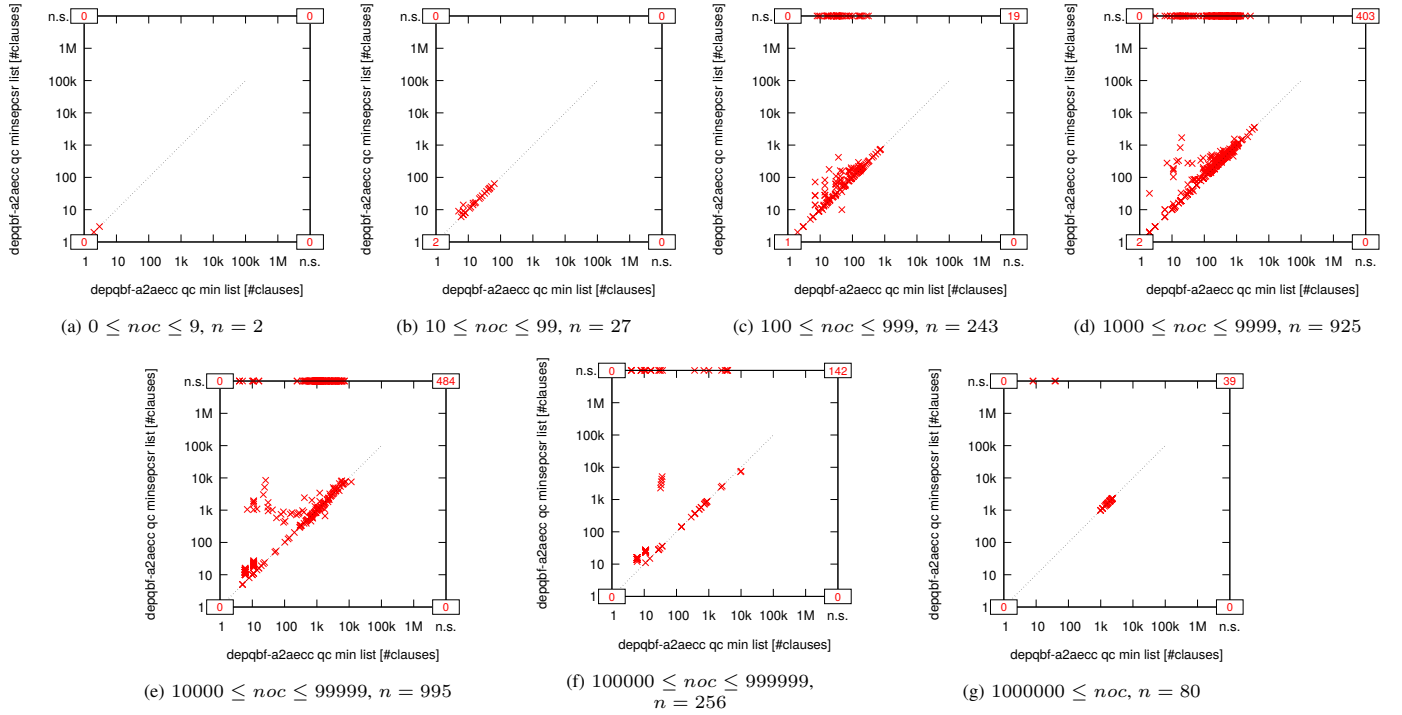


Fig. 430: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode qc minsepcsr list partitioned by number of clauses (number of clauses).

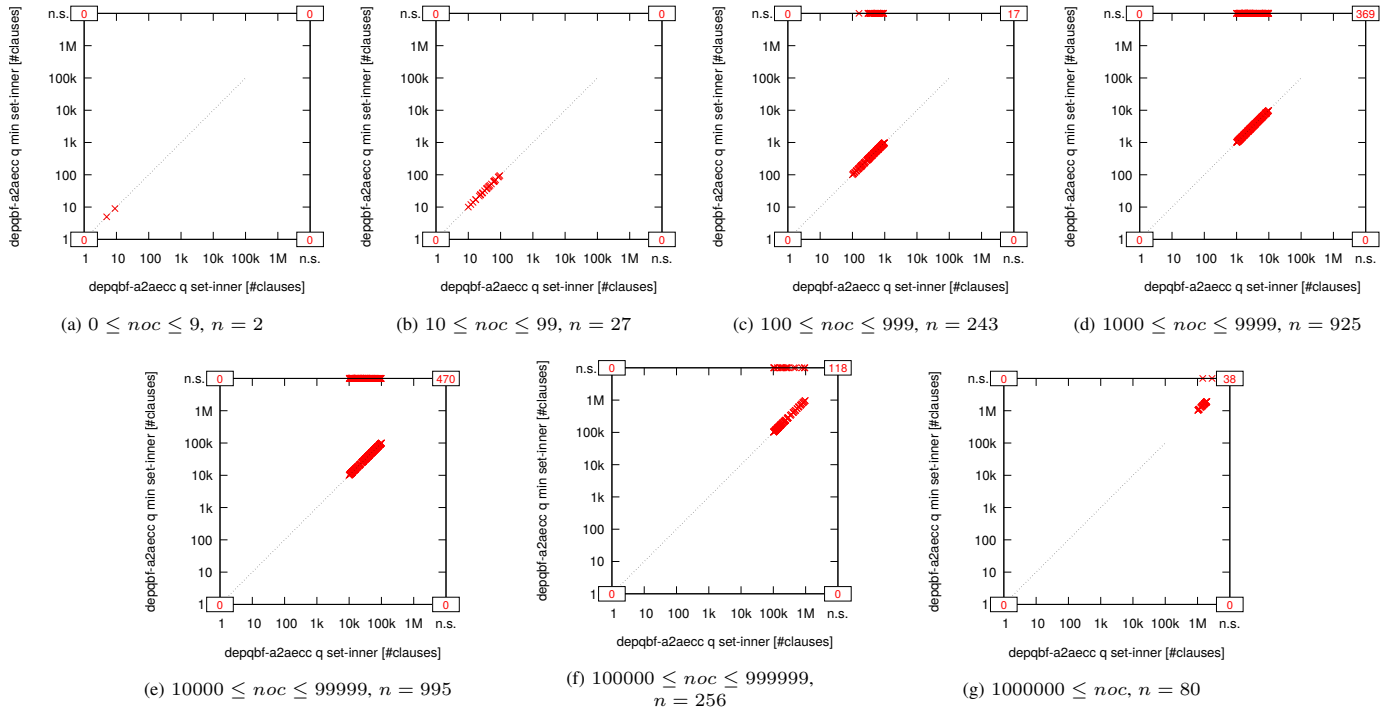


Fig. 431: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode q min set-inner partitioned by number of clauses (number of clauses).

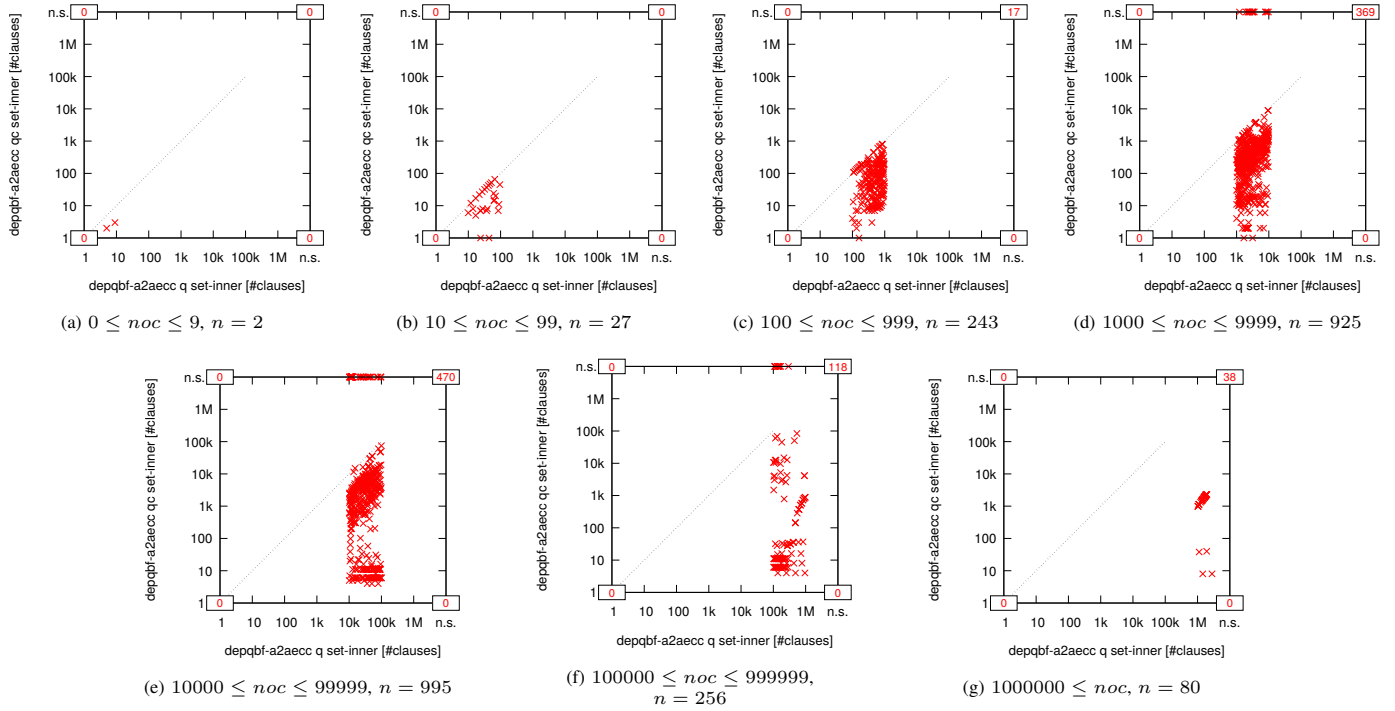


Fig. 432: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode qc set-inner partitioned by number of clauses (number of clauses).

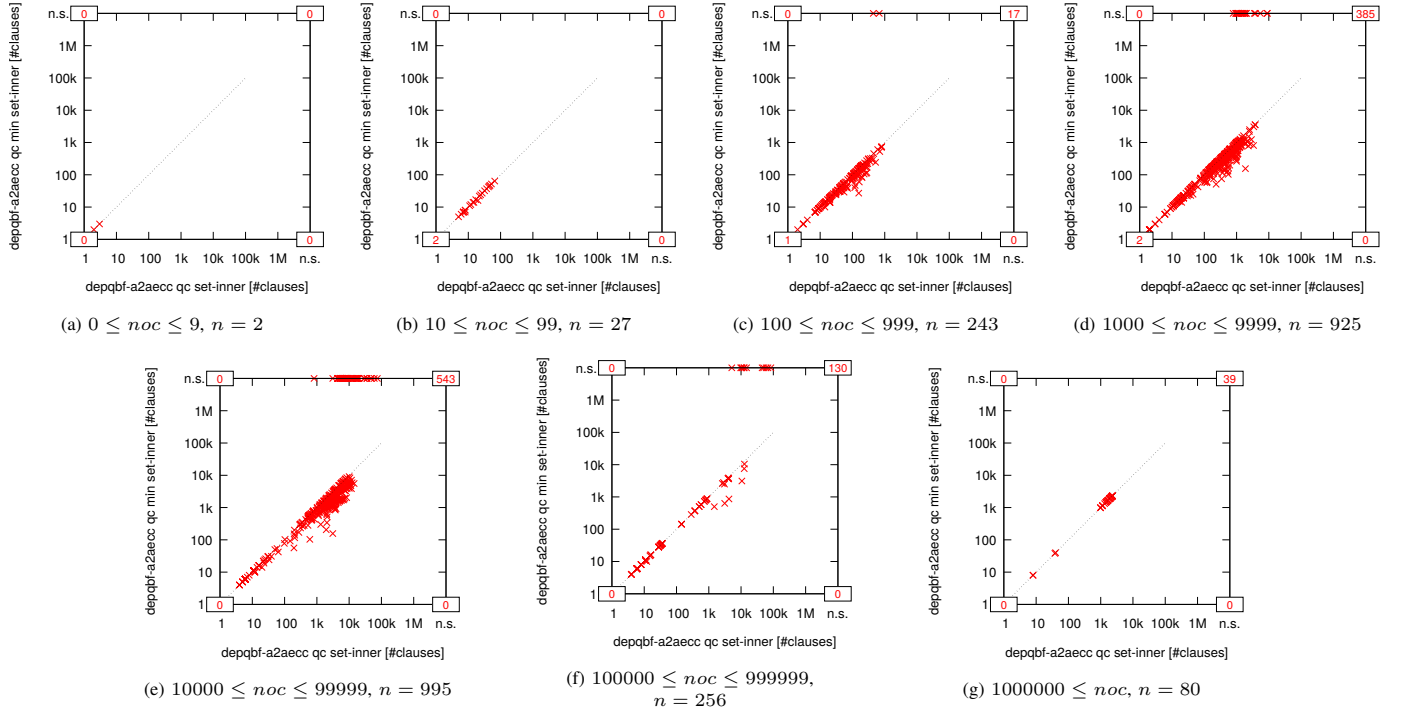


Fig. 433: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode qc min set-inner partitioned by number of clauses (number of clauses).

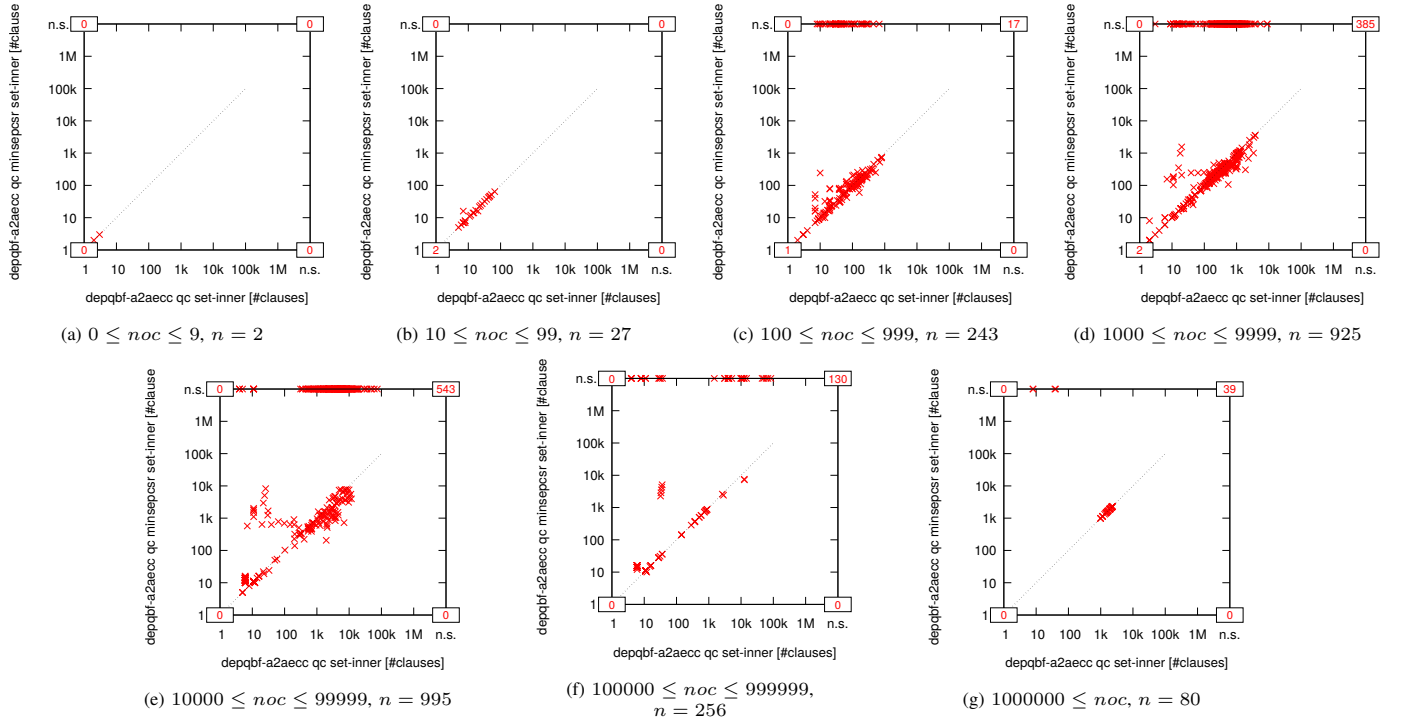


Fig. 434: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode qc minsepcsr set-inner partitioned by number of clauses (number of clauses).

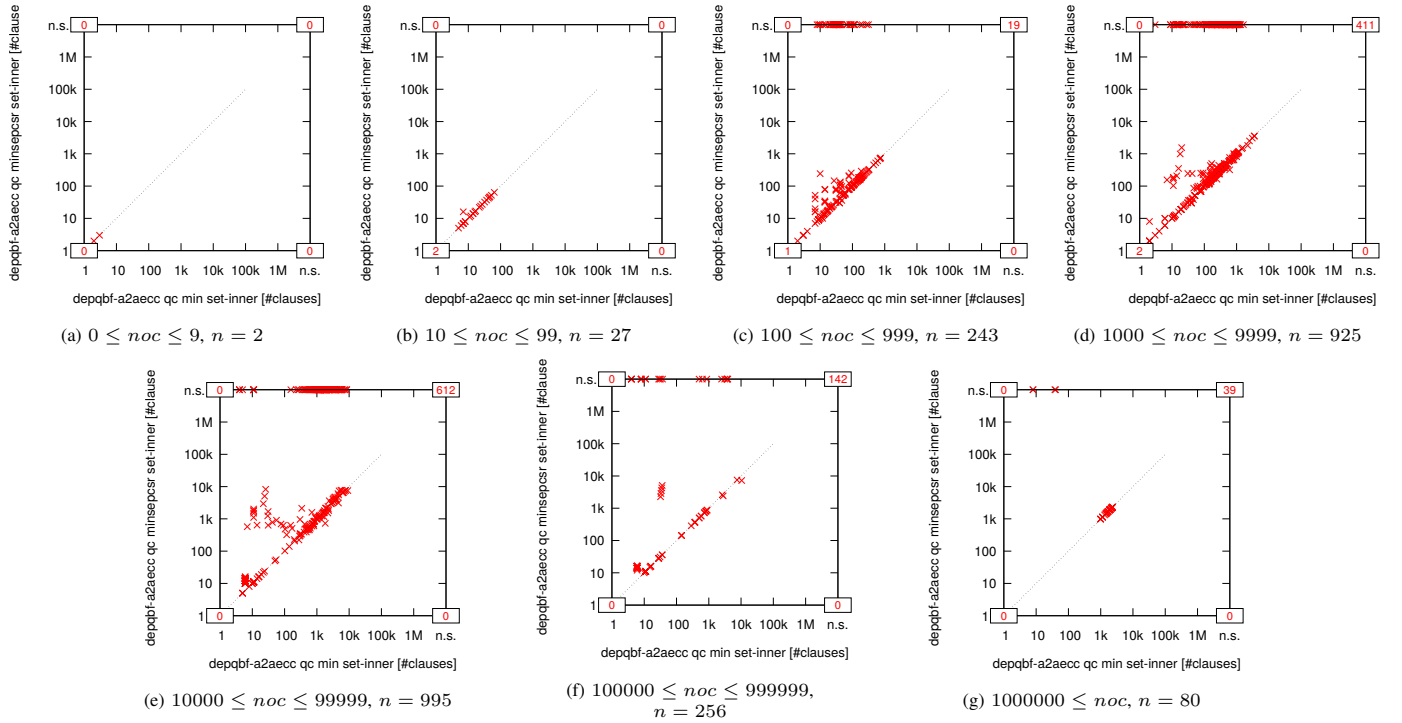


Fig. 435: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode qc minsepcsr set-inner partitioned by number of clauses (number of clauses).

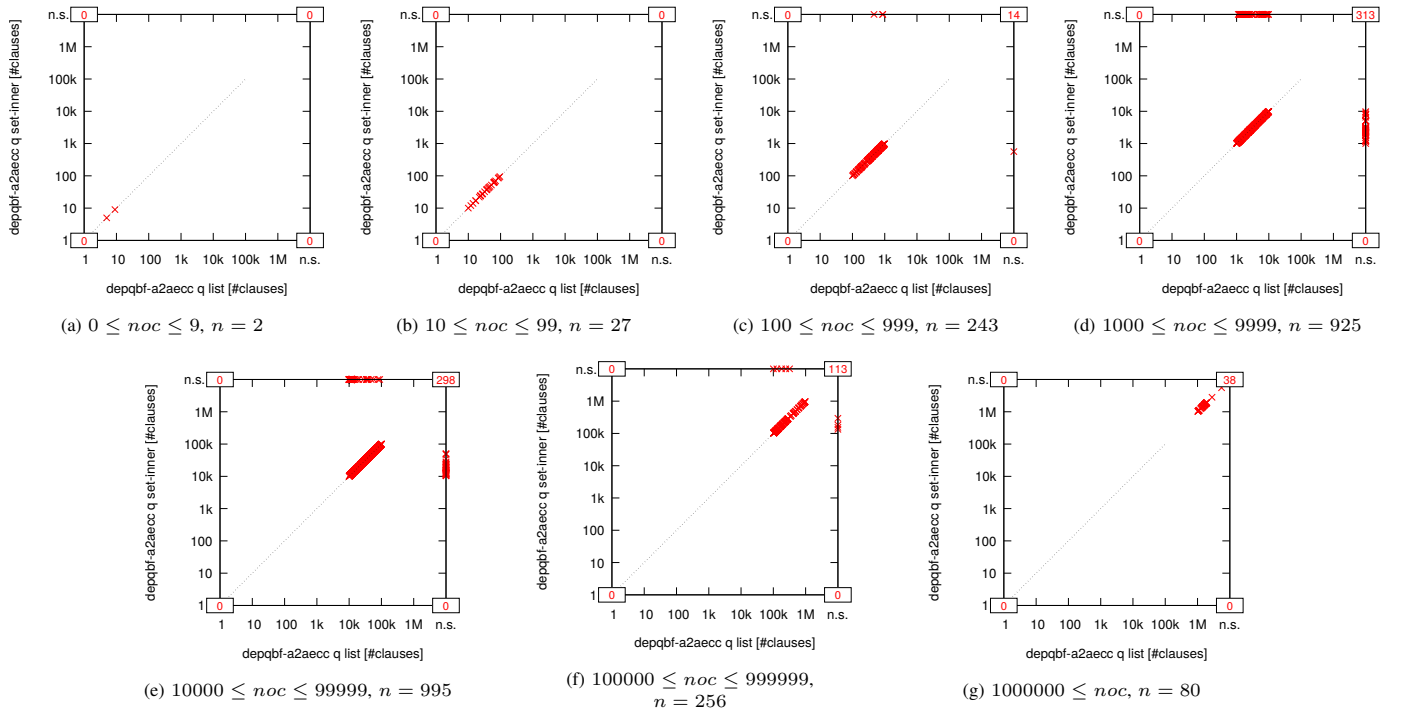


Fig. 436: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode q set-inner partitioned by number of clauses (number of clauses).

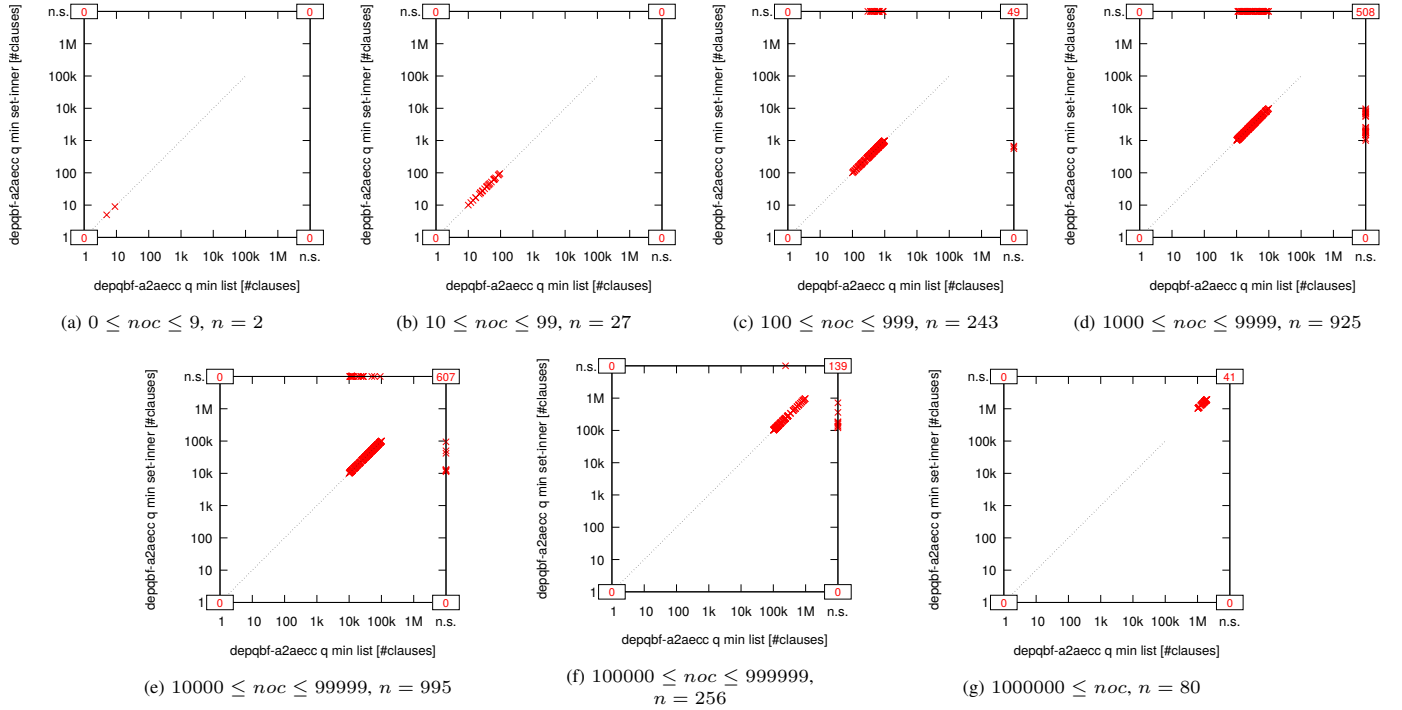


Fig. 437: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q min list versus mode q min set-inner partitioned by number of clauses (number of clauses).

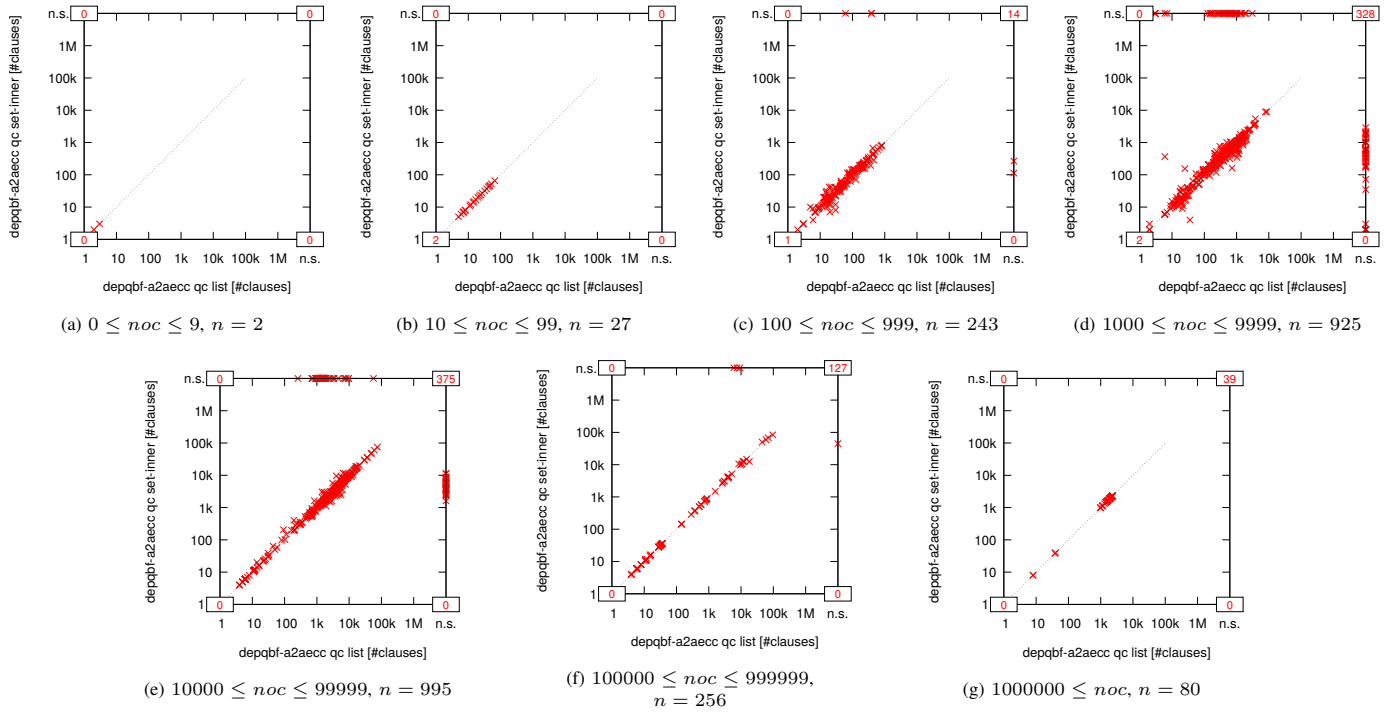


Fig. 438: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode qc set-inner partitioned by number of clauses (number of clauses).

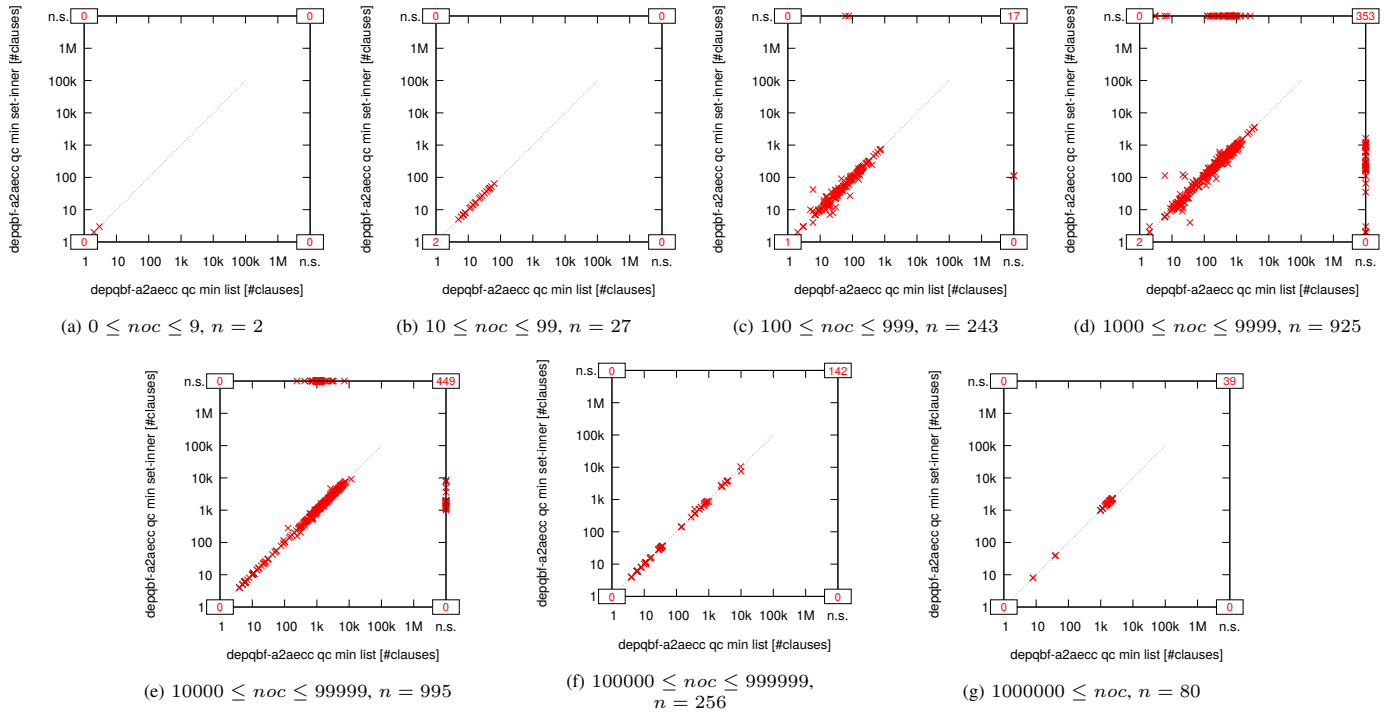


Fig. 439: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode qc min set-inner partitioned by number of clauses (number of clauses).

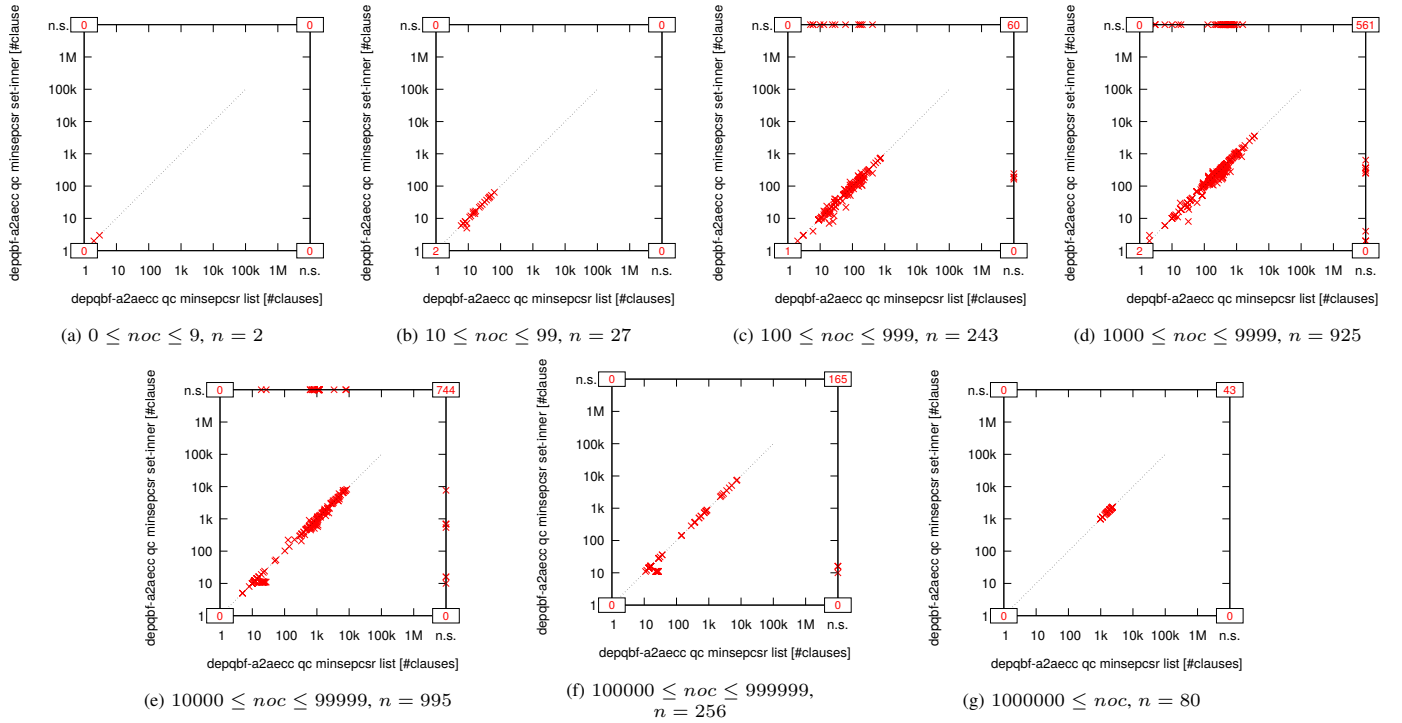


Fig. 440: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode qc minsepcsr set-inner partitioned by number of clauses (number of clauses).

E. Partitioned by Maximum Variable Index

This subsection shows figures of the sizes of unsatisfiable cores with subfigures for partitions of the benchmarks according to their maximum variable index.

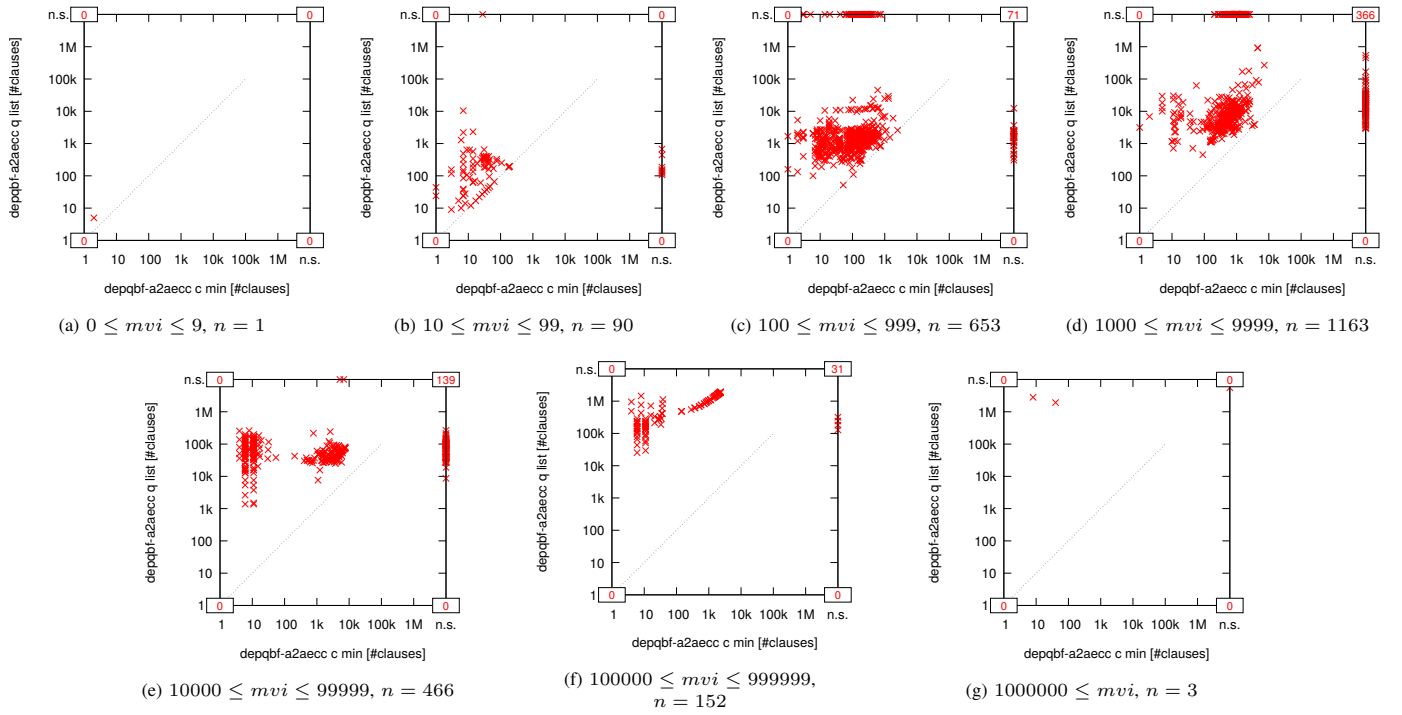


Fig. 441: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q list partitioned by maximum variable index (number of clauses).

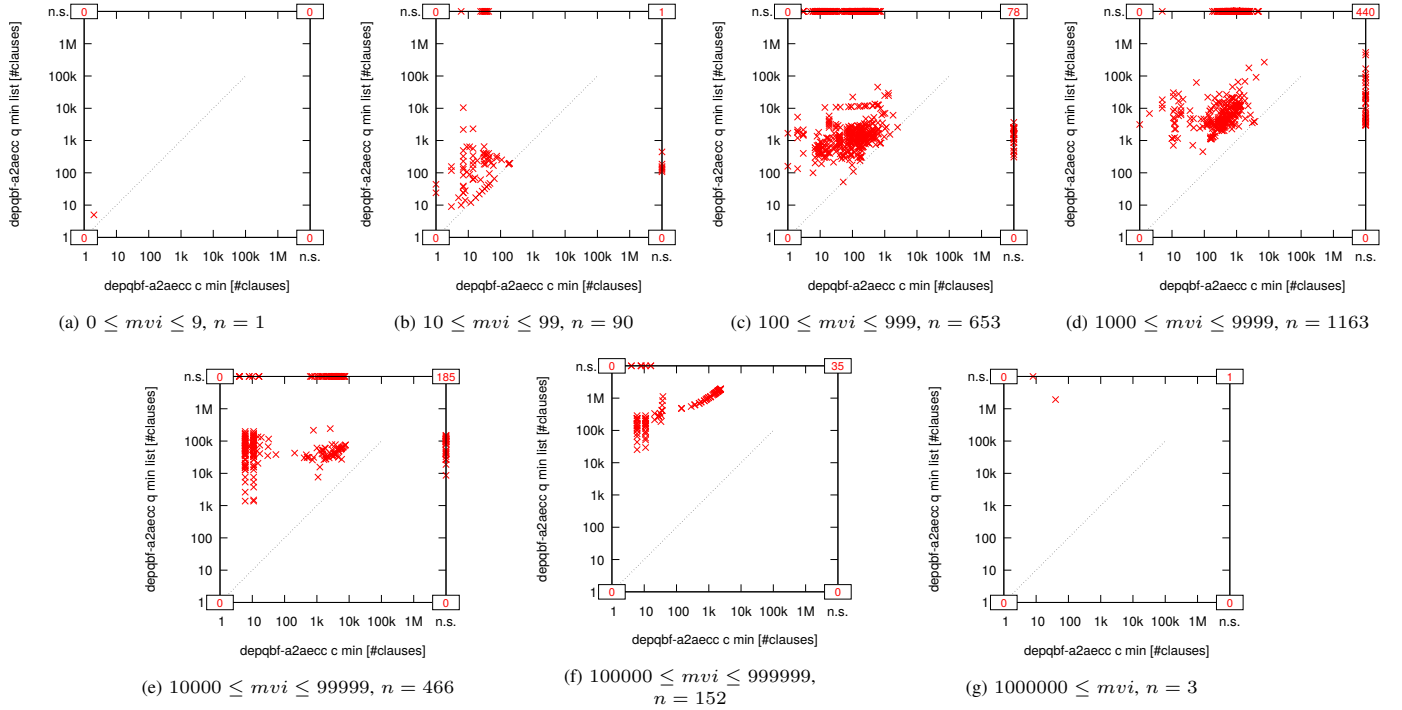


Fig. 442: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q min list partitioned by maximum variable index (number of clauses).

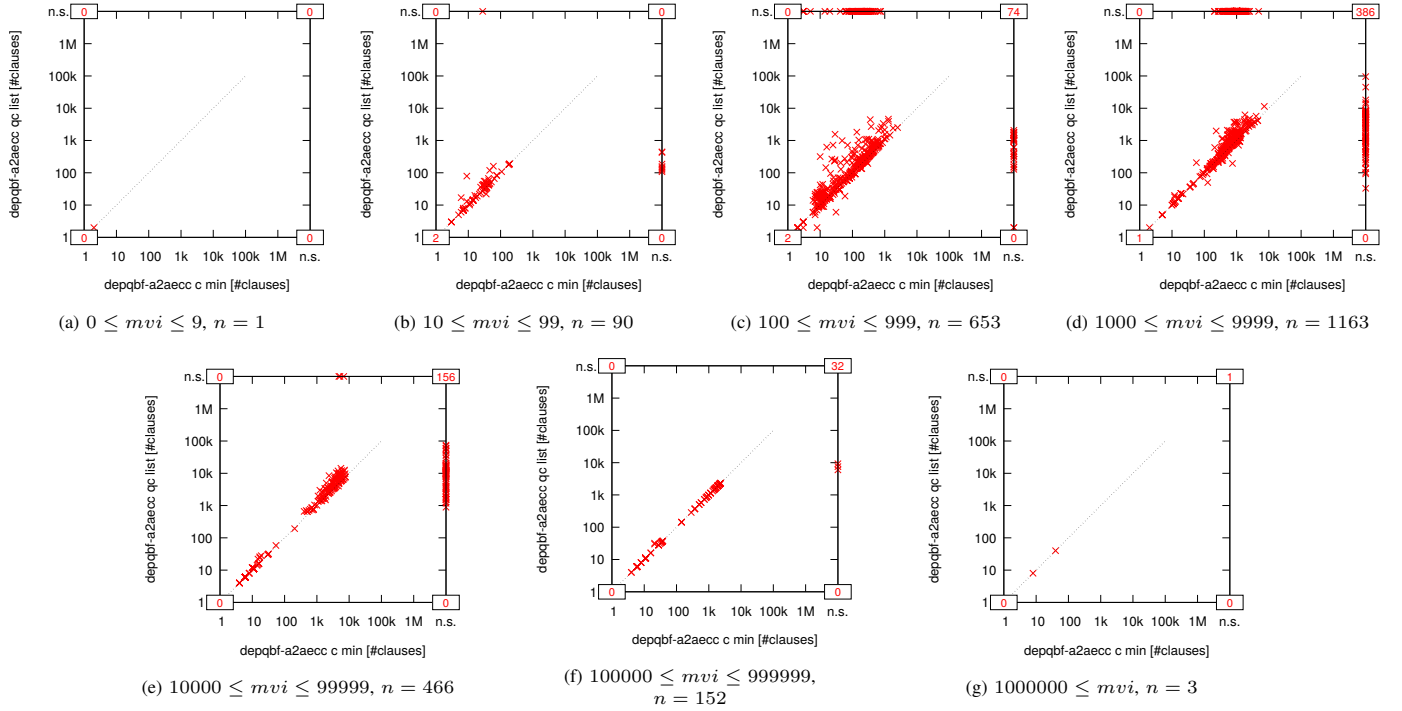


Fig. 443: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc list partitioned by maximum variable index (number of clauses).

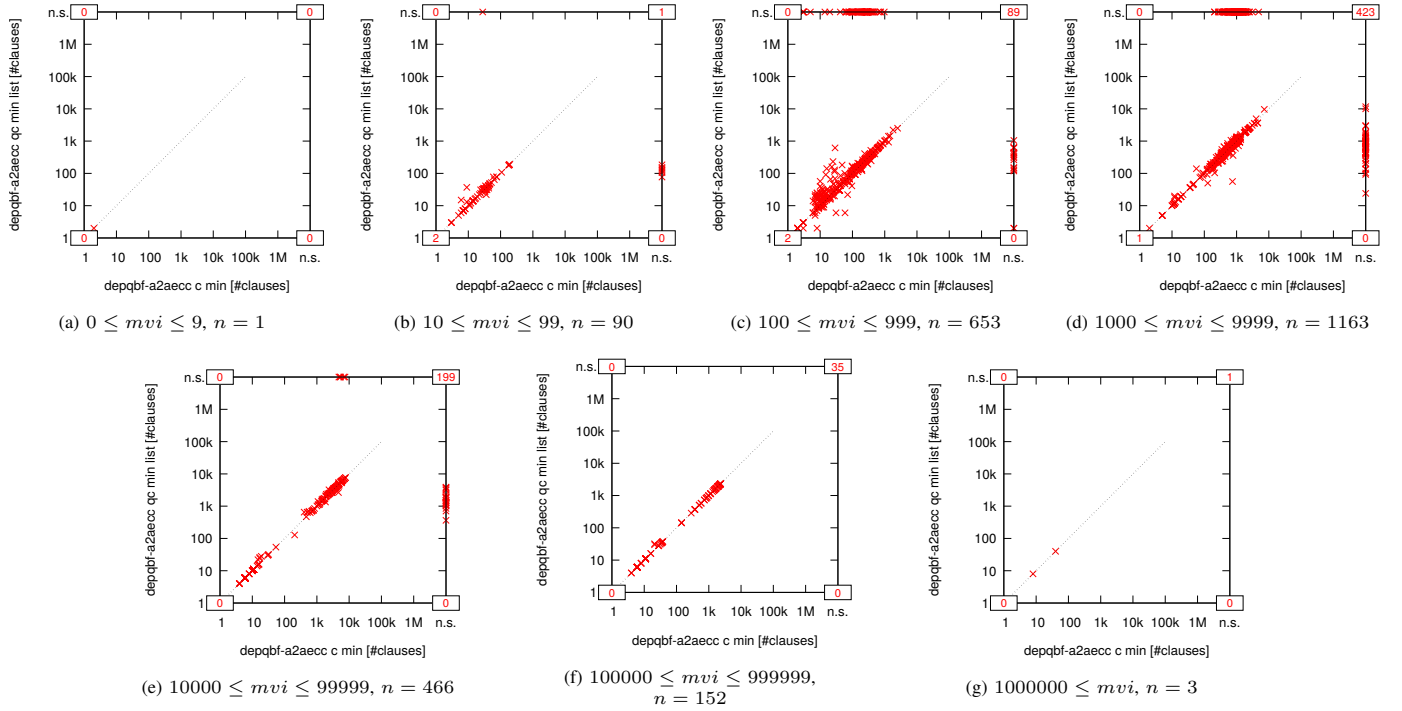


Fig. 444: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc min list partitioned by maximum variable index (number of clauses).

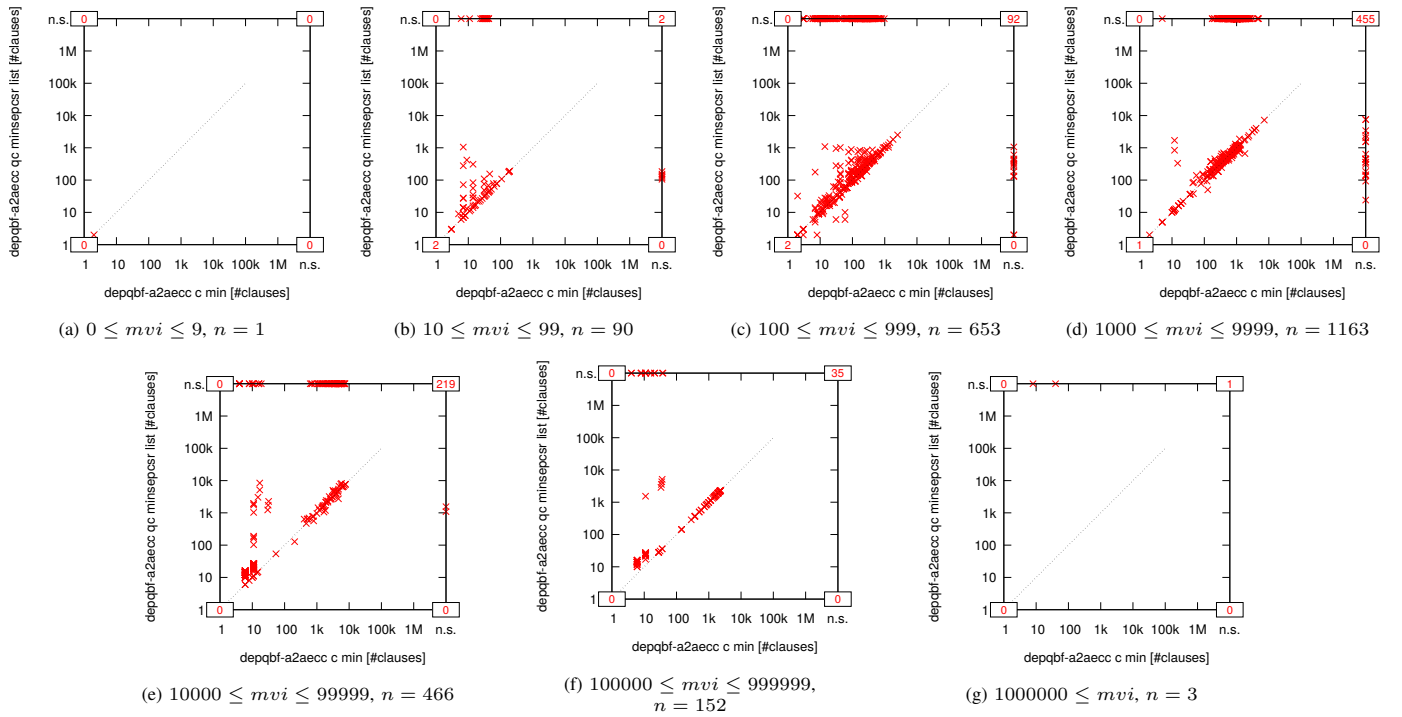


Fig. 445: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc minsepcsr list partitioned by maximum variable index (number of clauses).

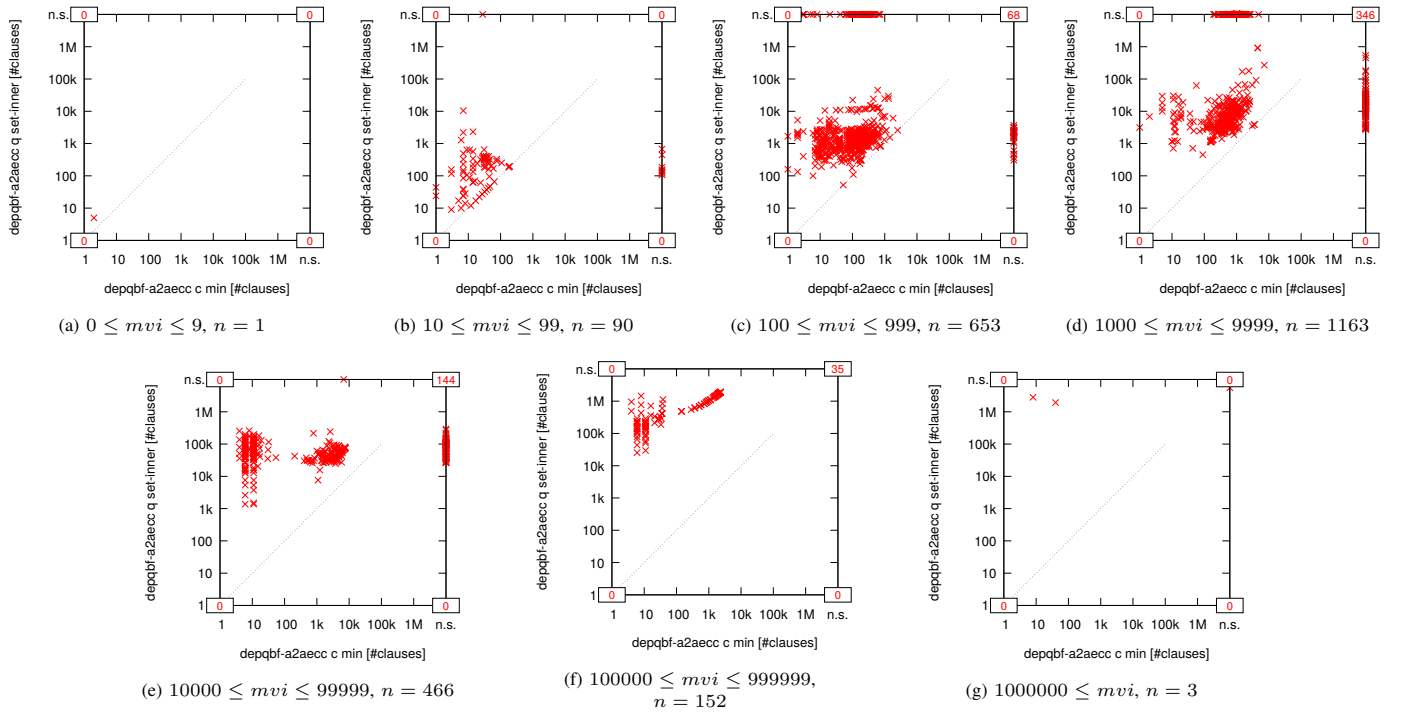


Fig. 446: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q set-inner partitioned by maximum variable index (number of clauses).

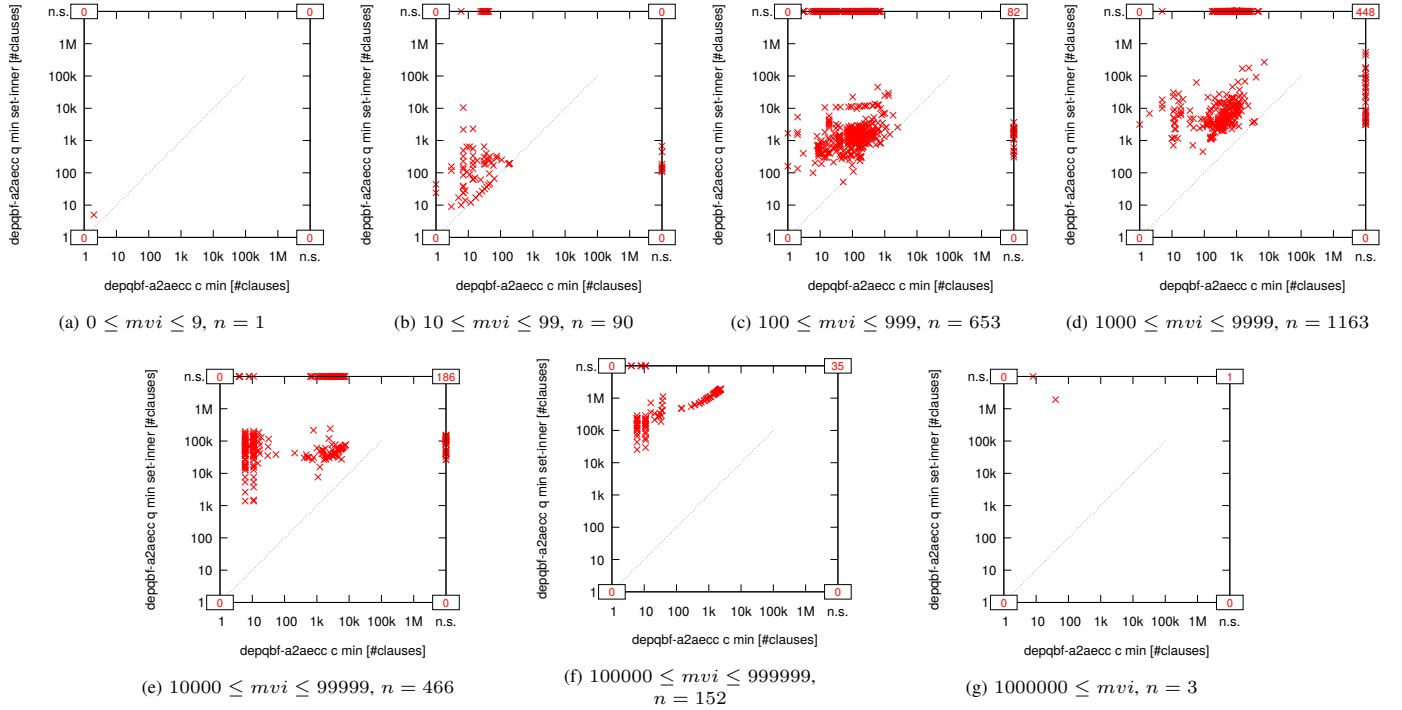


Fig. 447: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q min set-inner partitioned by maximum variable index (number of clauses).

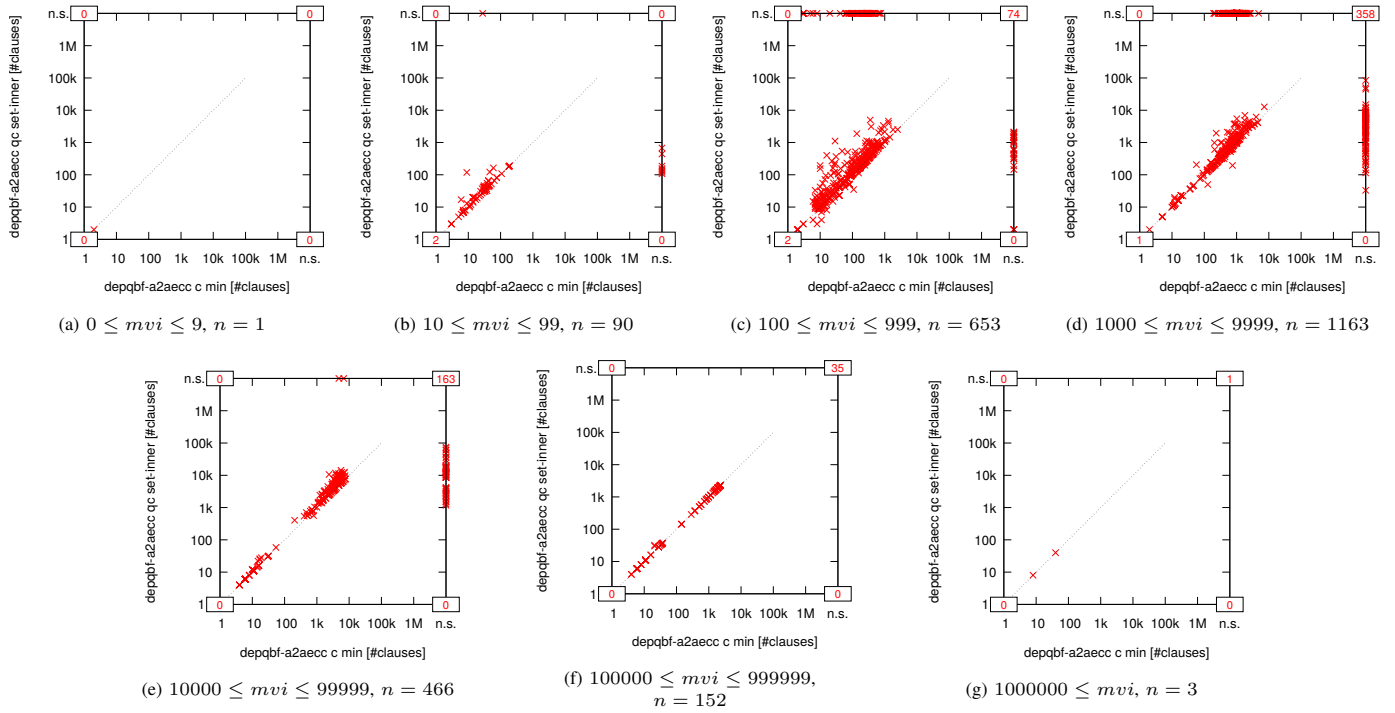


Fig. 448: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc set-inner partitioned by maximum variable index (number of clauses).

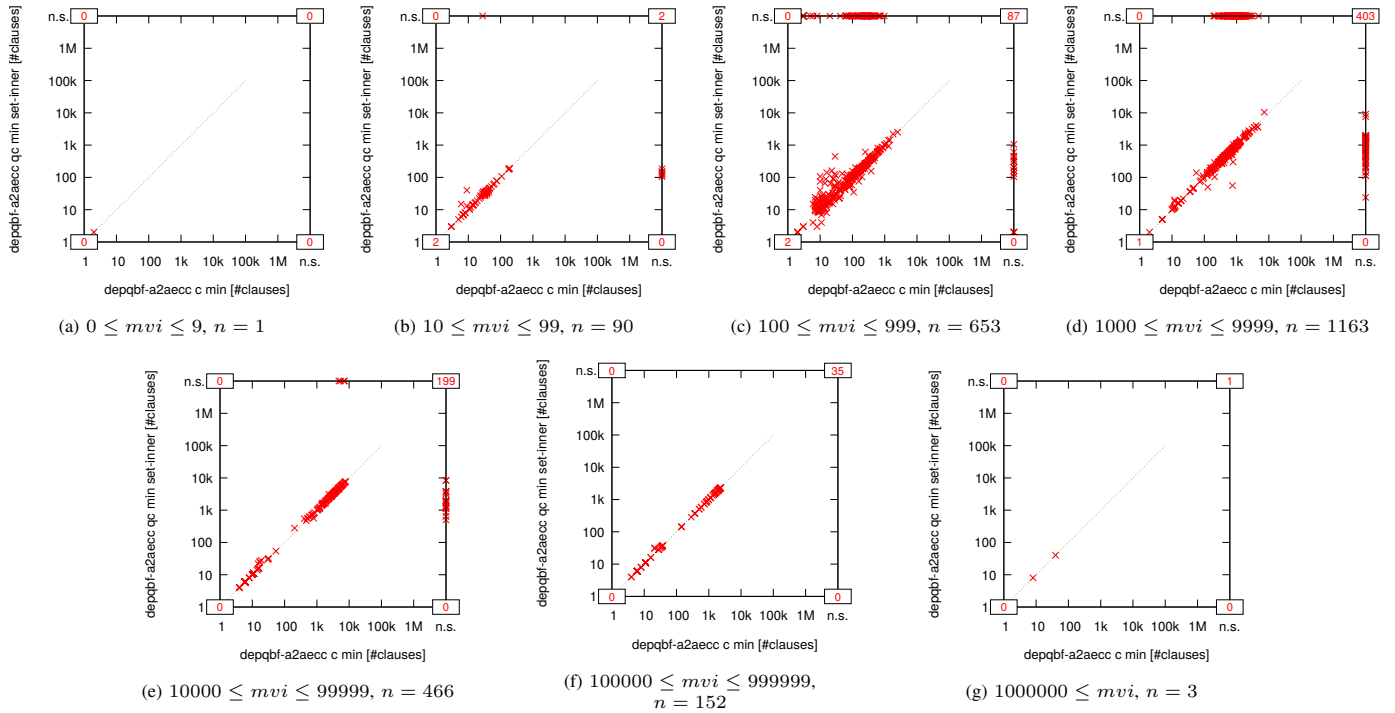


Fig. 449: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc min set-inner partitioned by maximum variable index (number of clauses).

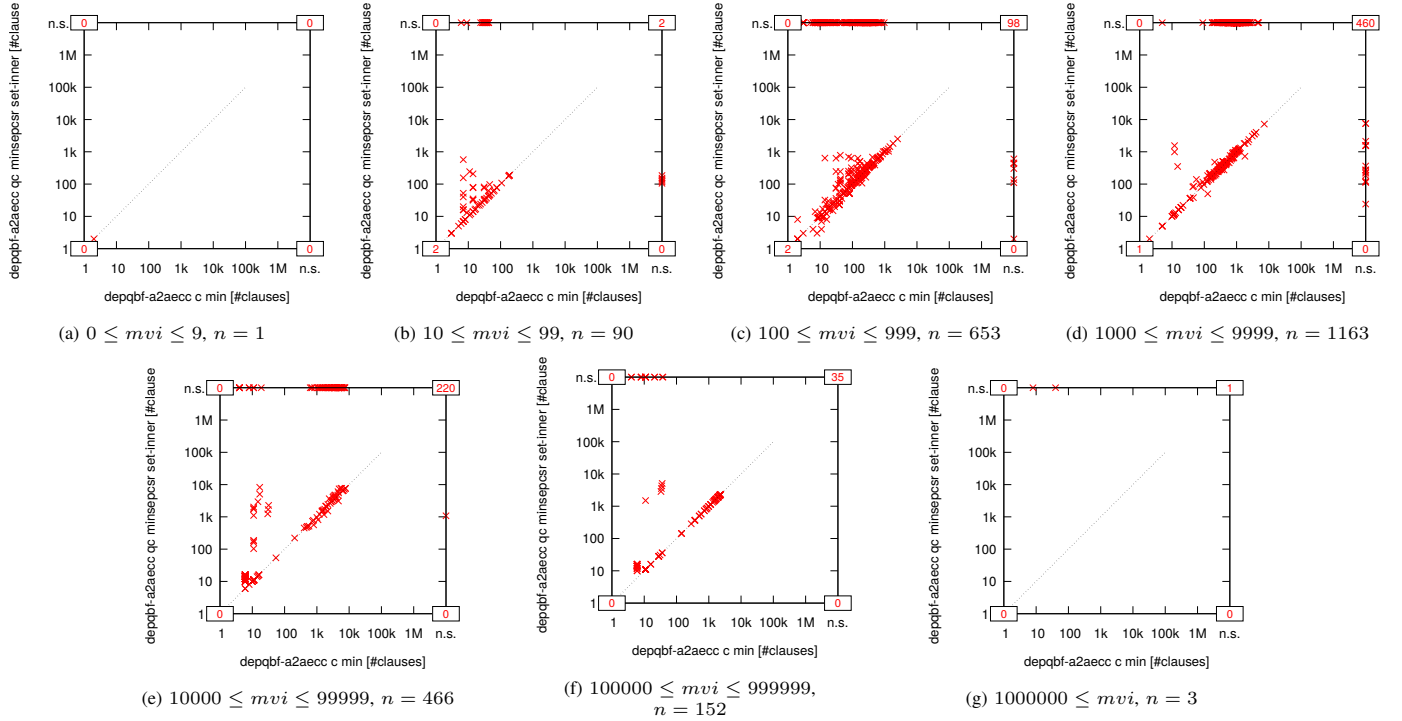


Fig. 450: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc minsepcsr set-inner partitioned by maximum variable index (number of clauses).

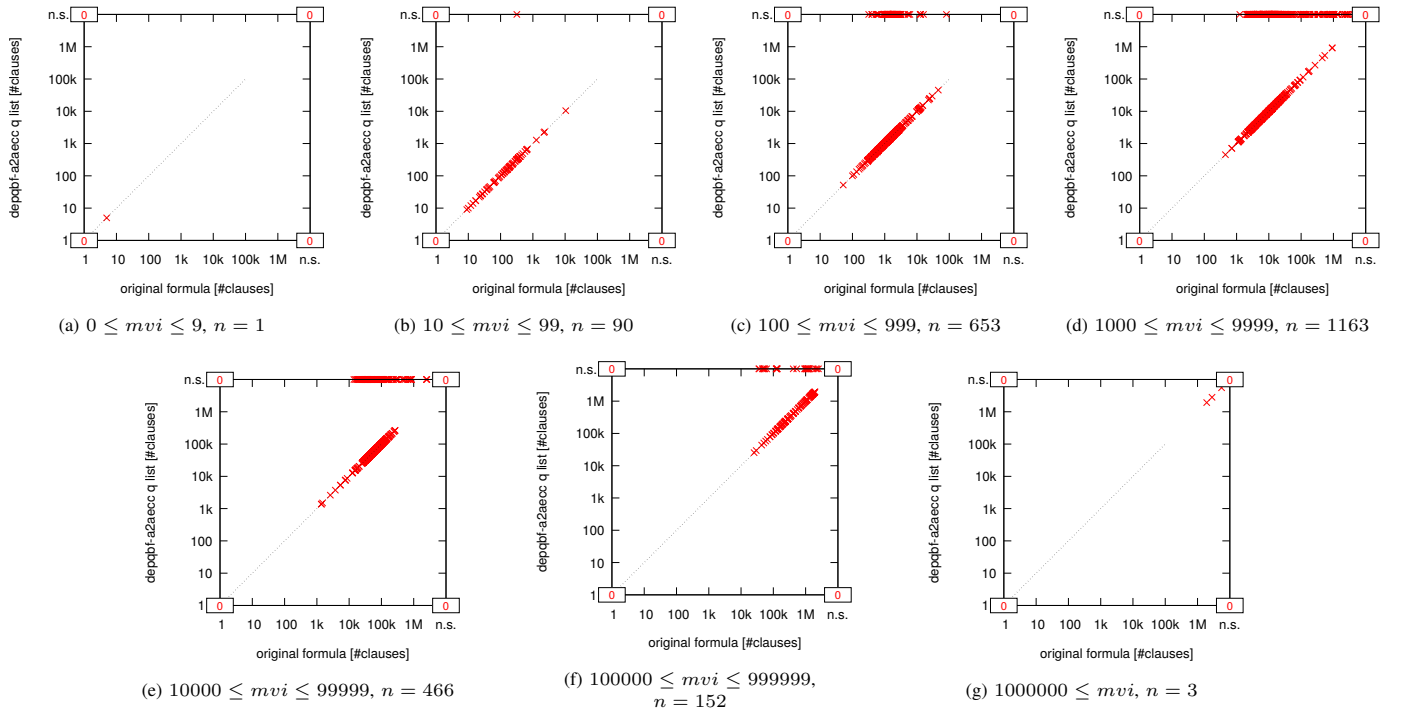


Fig. 451: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q list partitioned by maximum variable index (number of clauses).

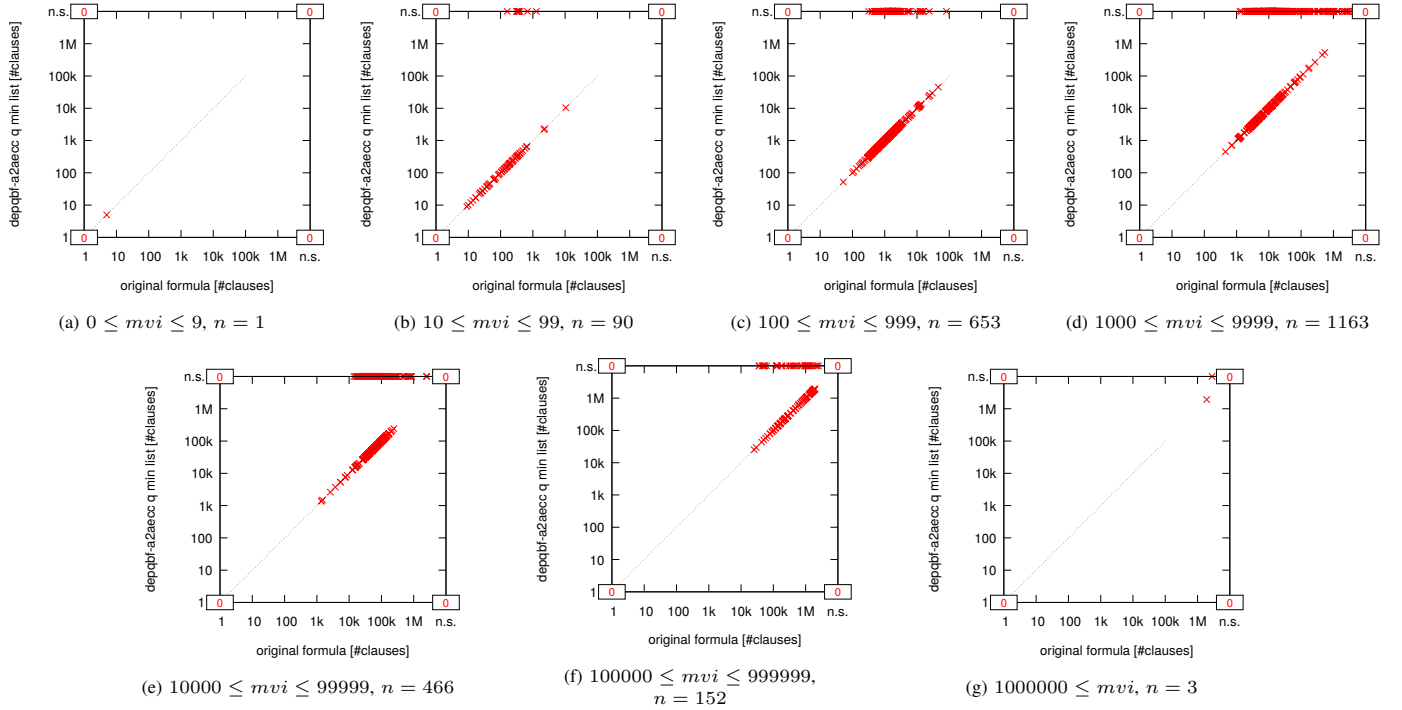


Fig. 452: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q min list partitioned by maximum variable index (number of clauses).

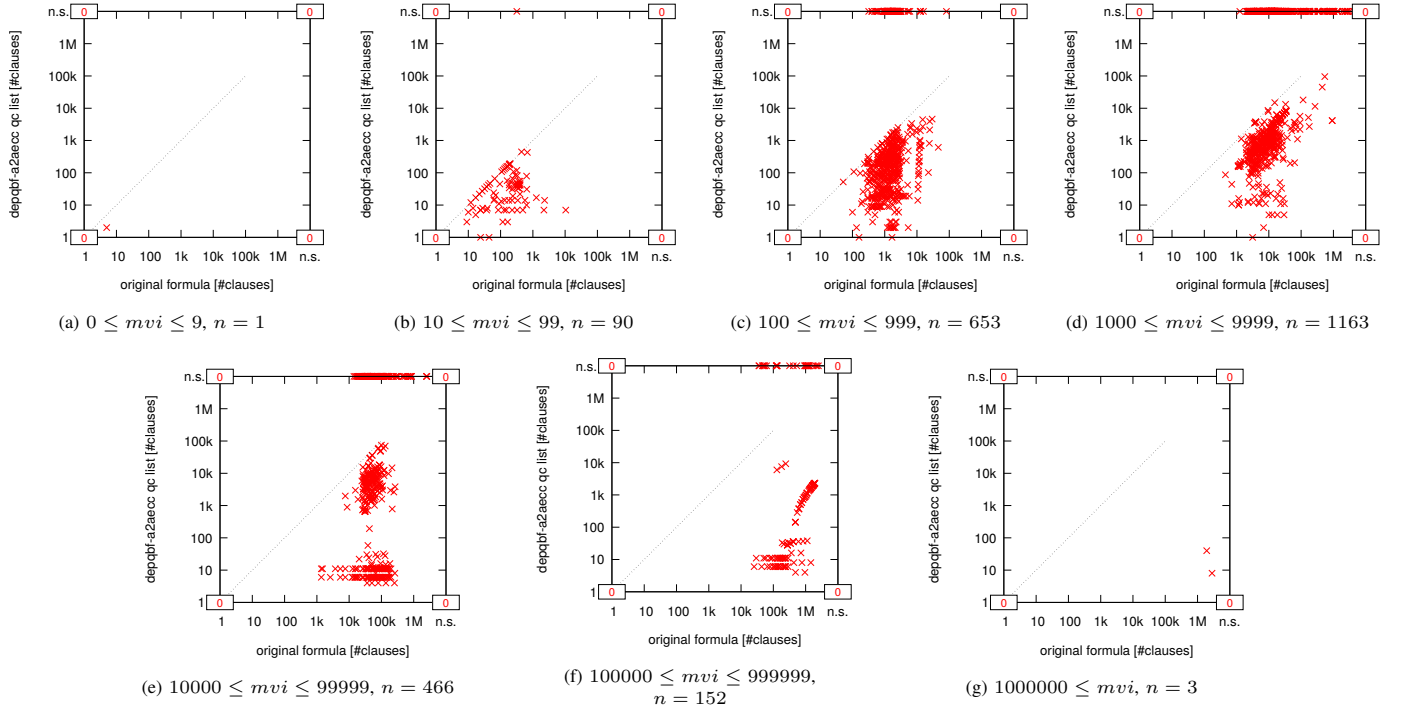


Fig. 453: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc list partitioned by maximum variable index (number of clauses).

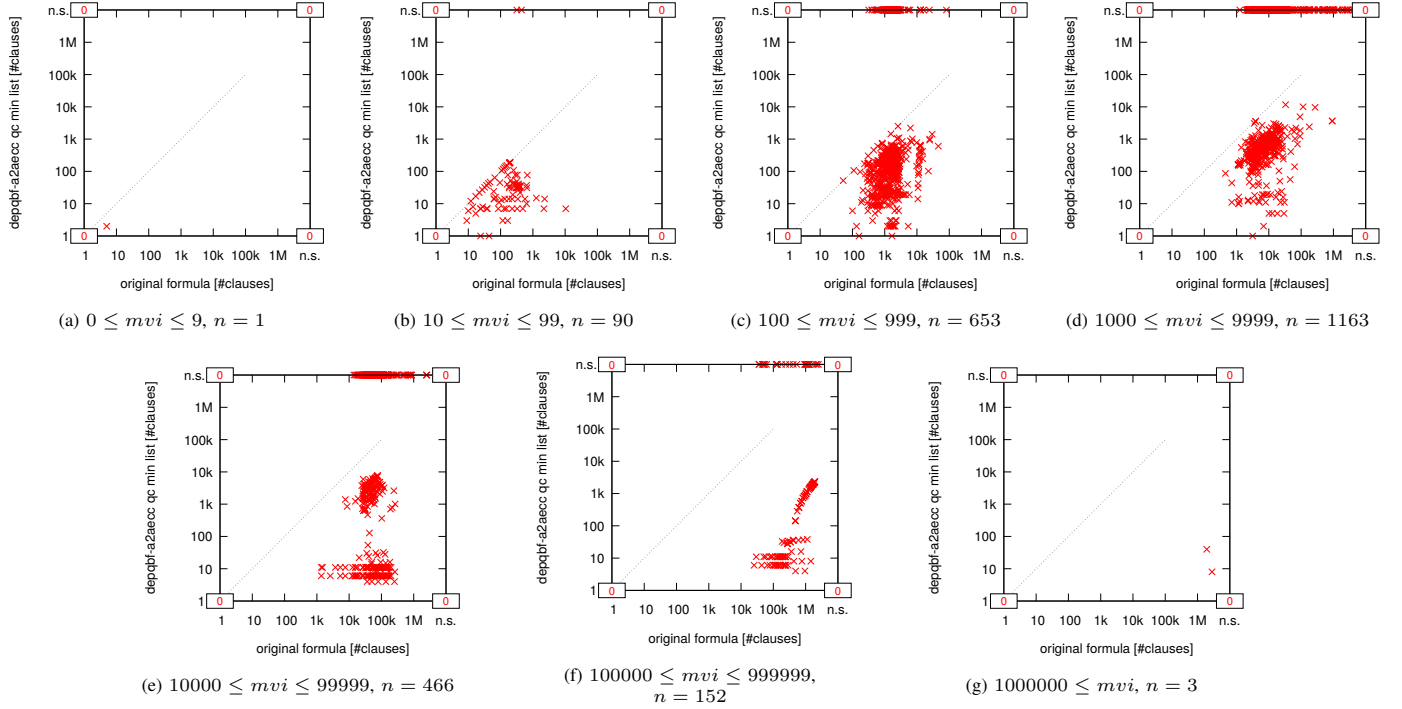


Fig. 454: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc min list partitioned by maximum variable index (number of clauses).

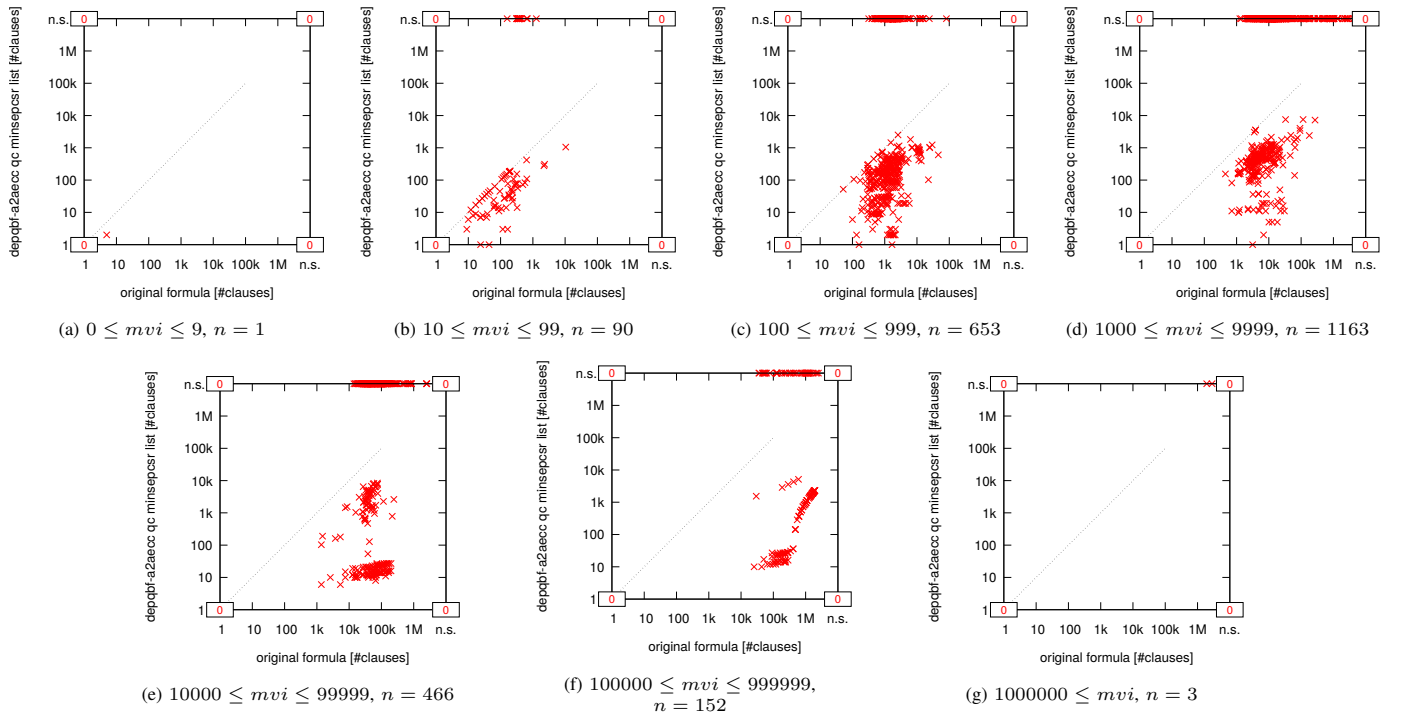


Fig. 455: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc minsepcsr list partitioned by maximum variable index (number of clauses).

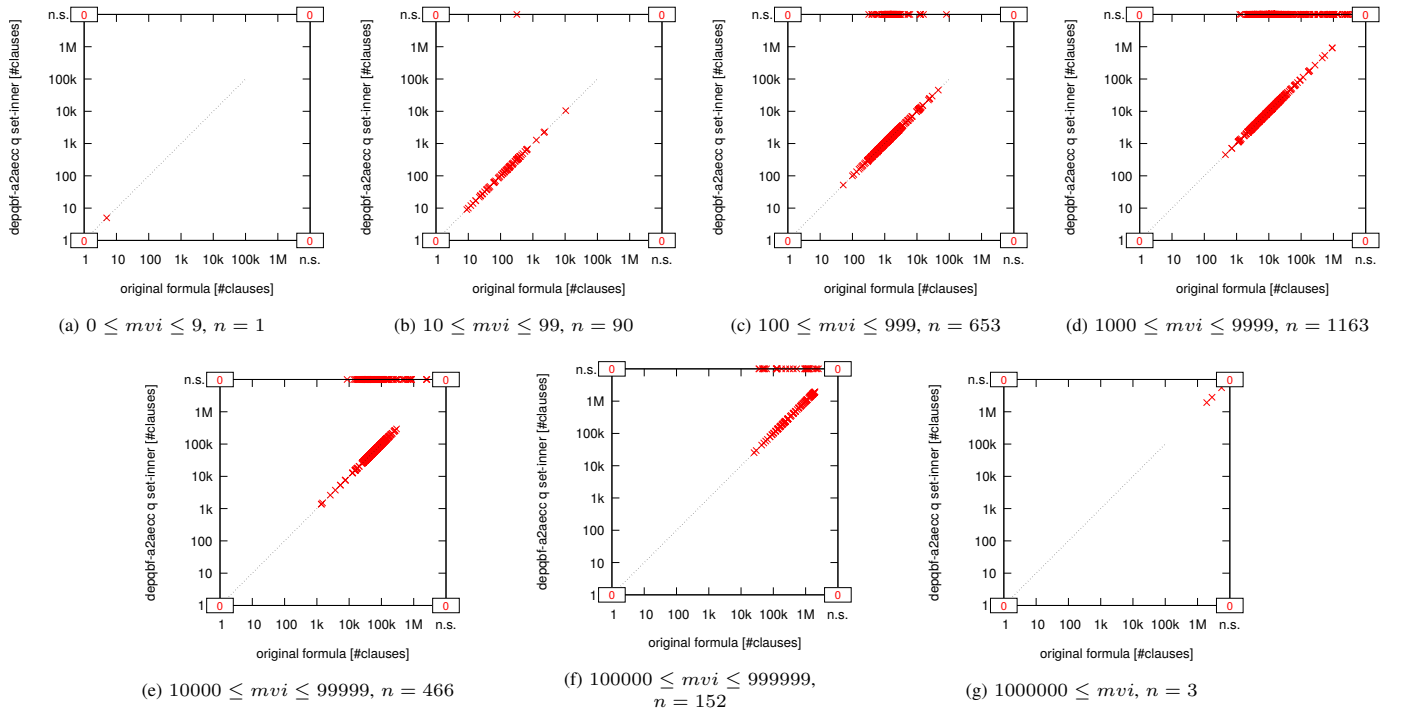


Fig. 456: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q set-inner partitioned by maximum variable index (number of clauses).

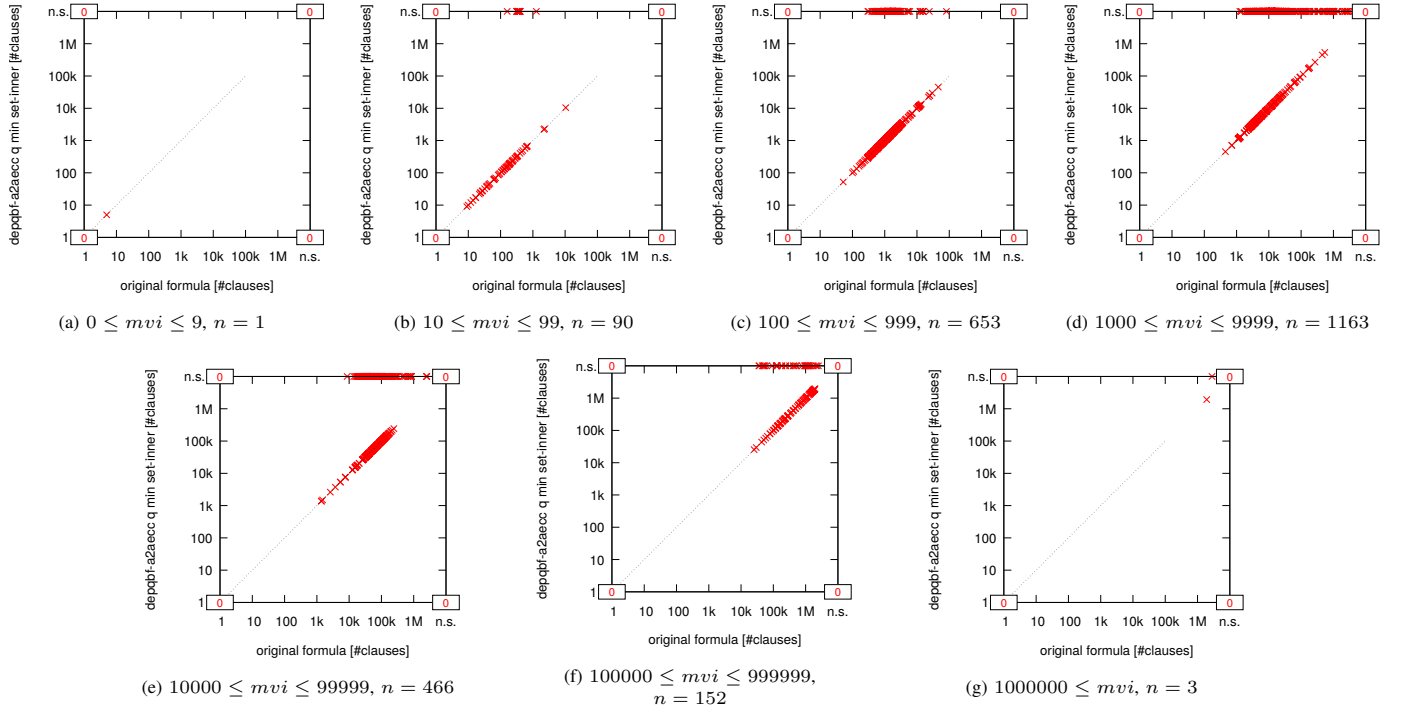


Fig. 457: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q min set-inner partitioned by maximum variable index (number of clauses).

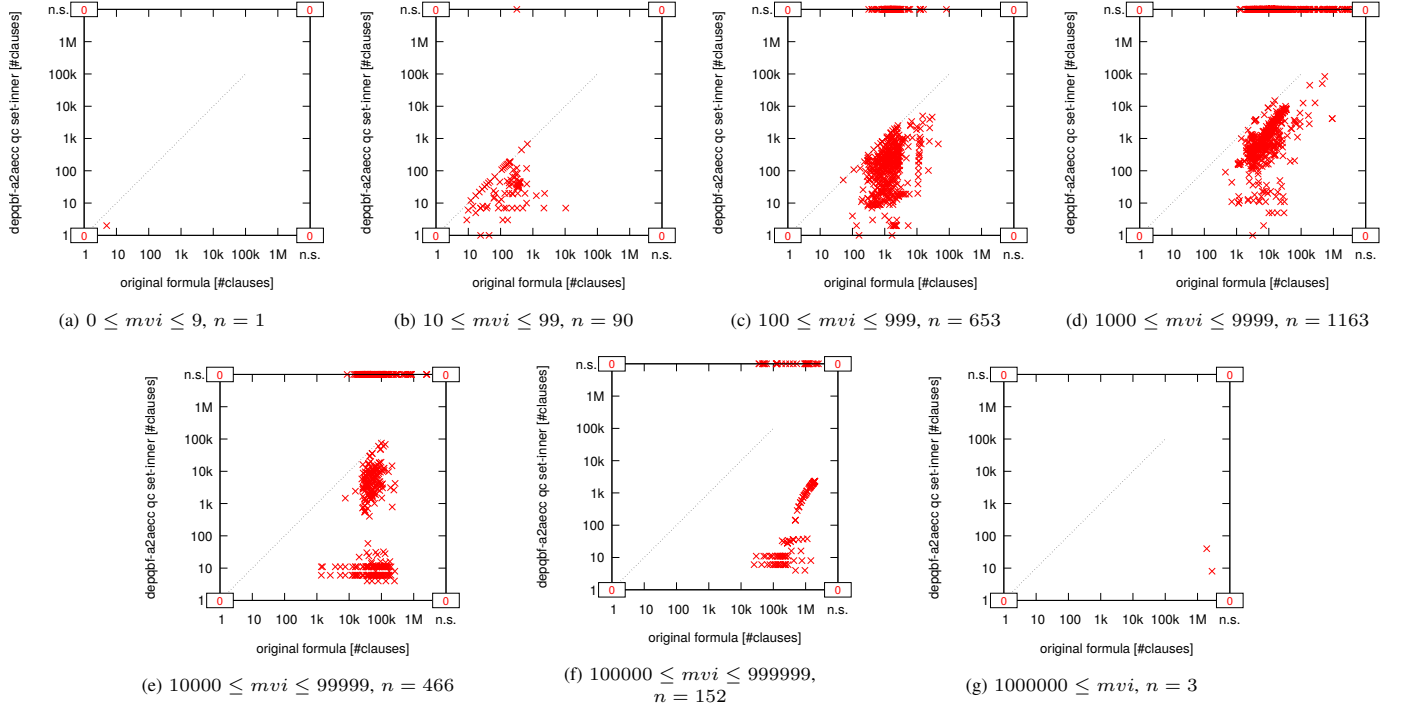


Fig. 458: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc set-inner partitioned by maximum variable index (number of clauses).

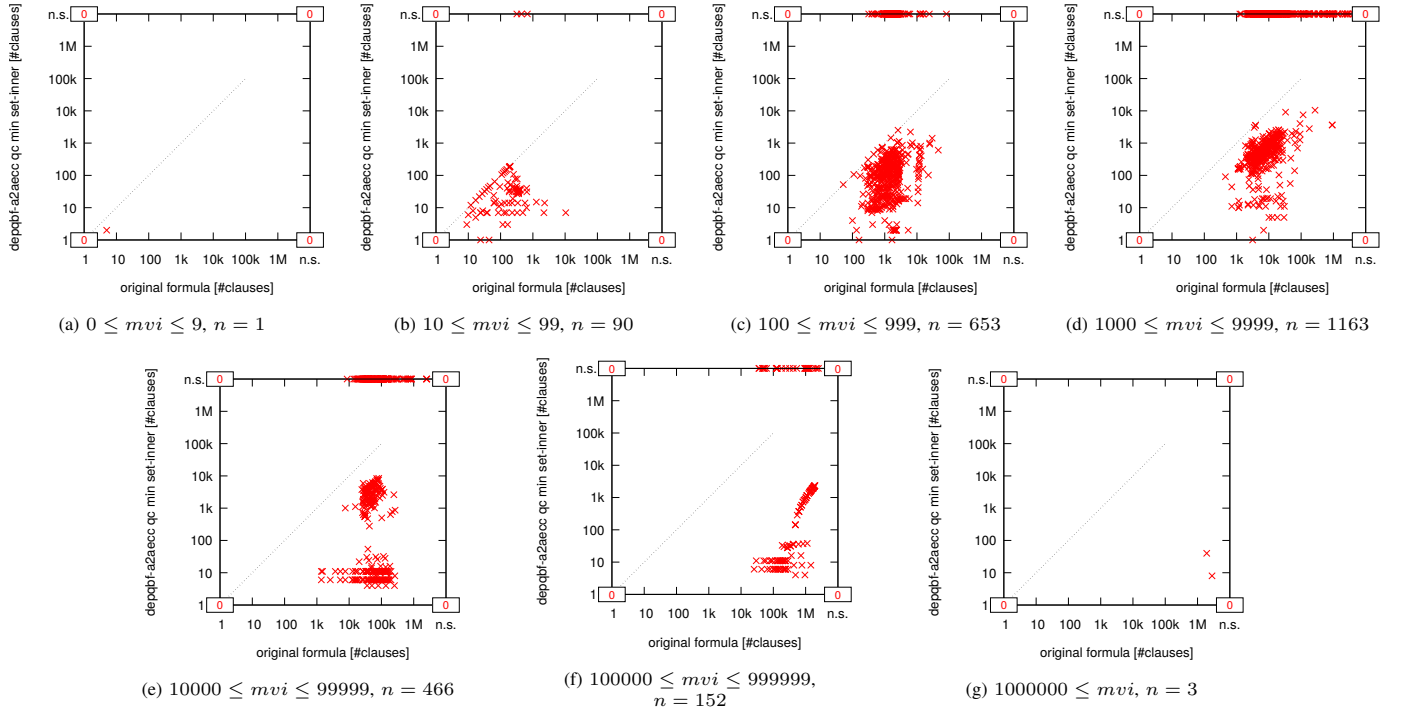


Fig. 459: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc min set-inner partitioned by maximum variable index (number of clauses).

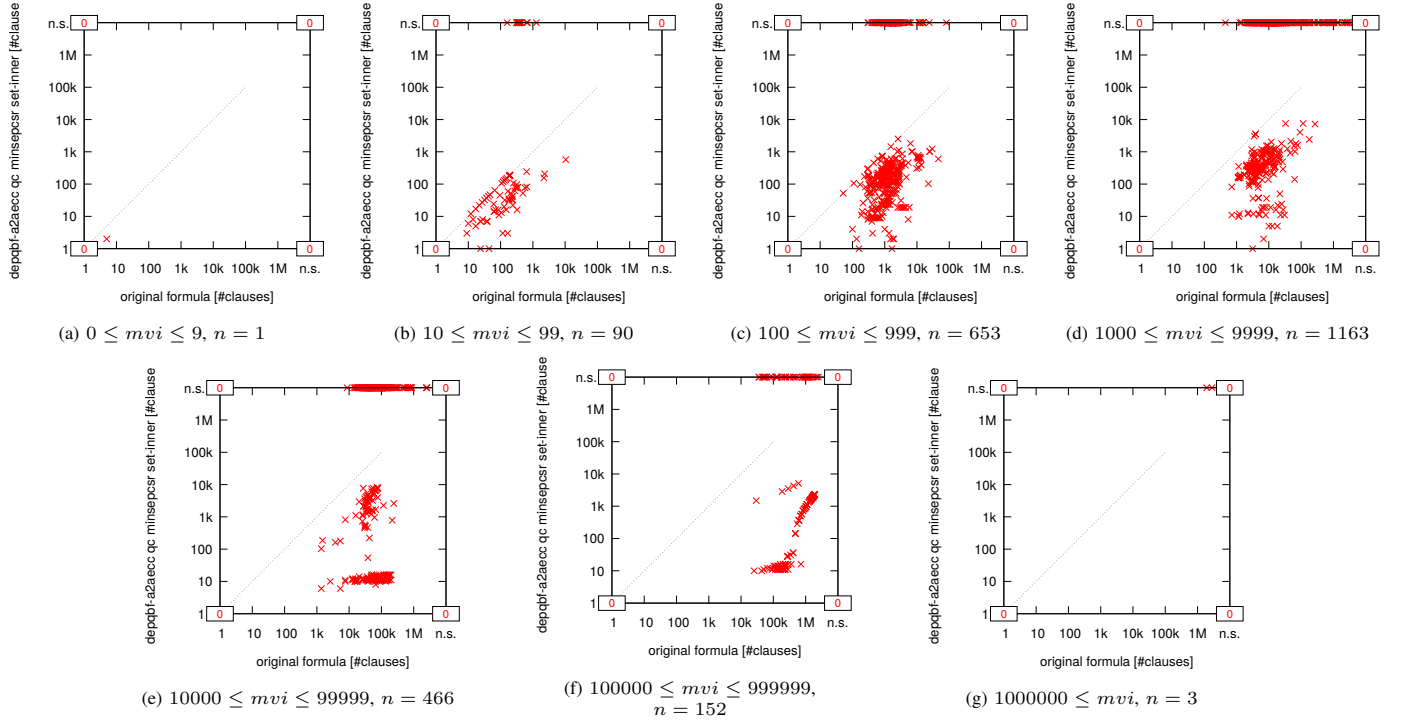


Fig. 460: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc minsepcsr set-inner partitioned by maximum variable index (number of clauses).

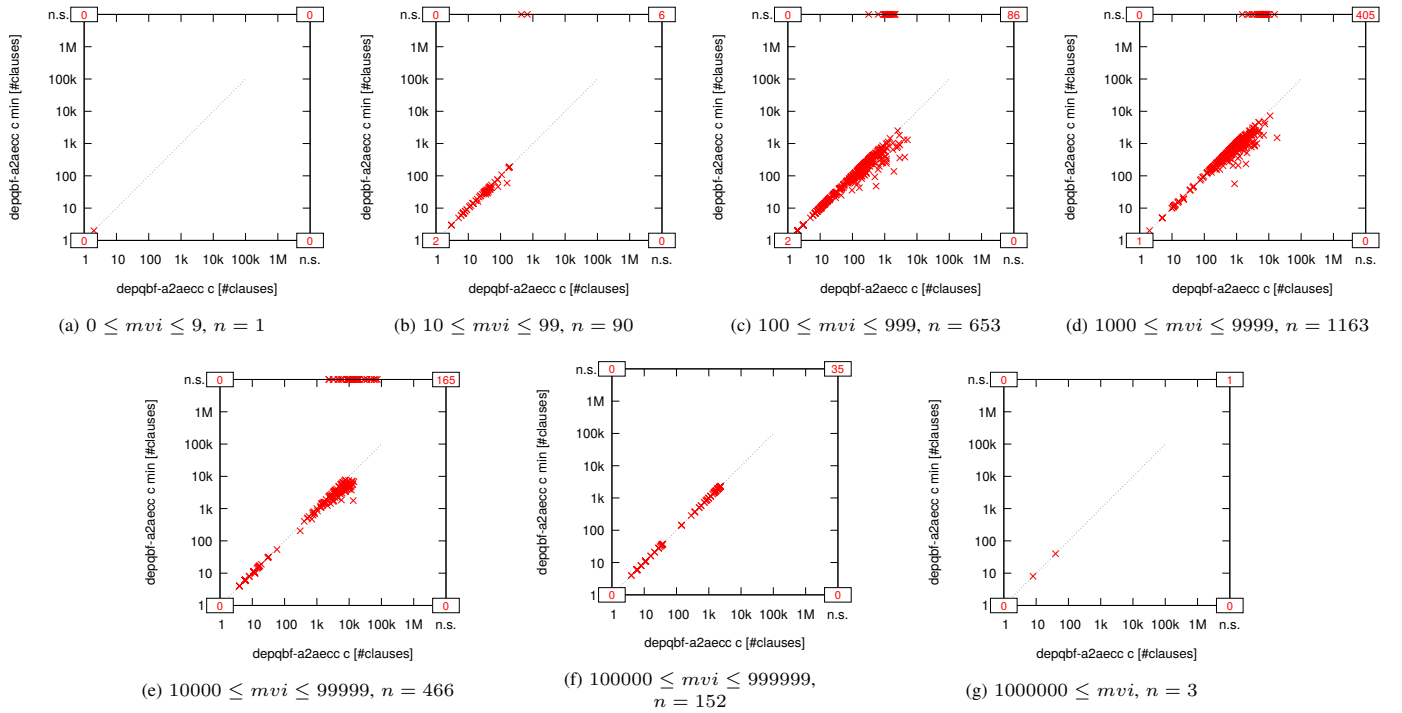


Fig. 461: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c versus mode c min partitioned by maximum variable index (number of clauses).

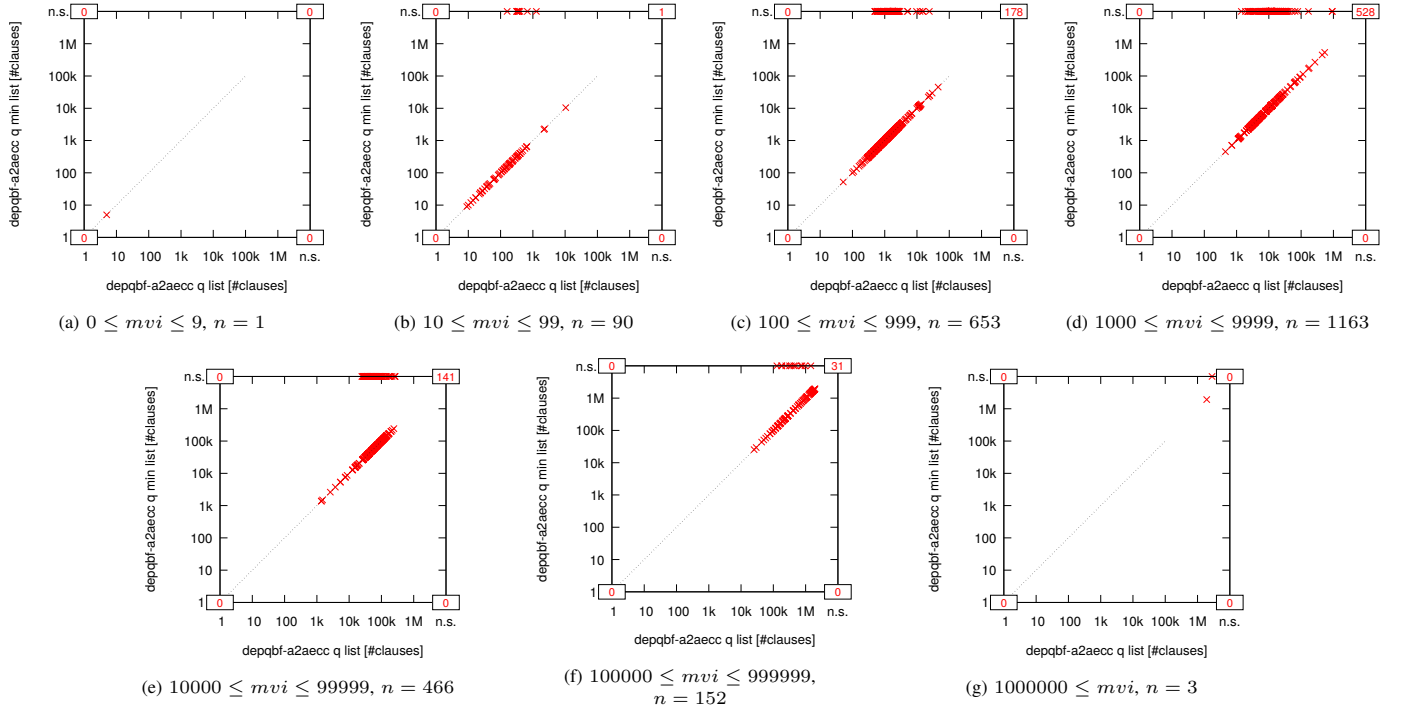


Fig. 462: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode q min list partitioned by maximum variable index (number of clauses).

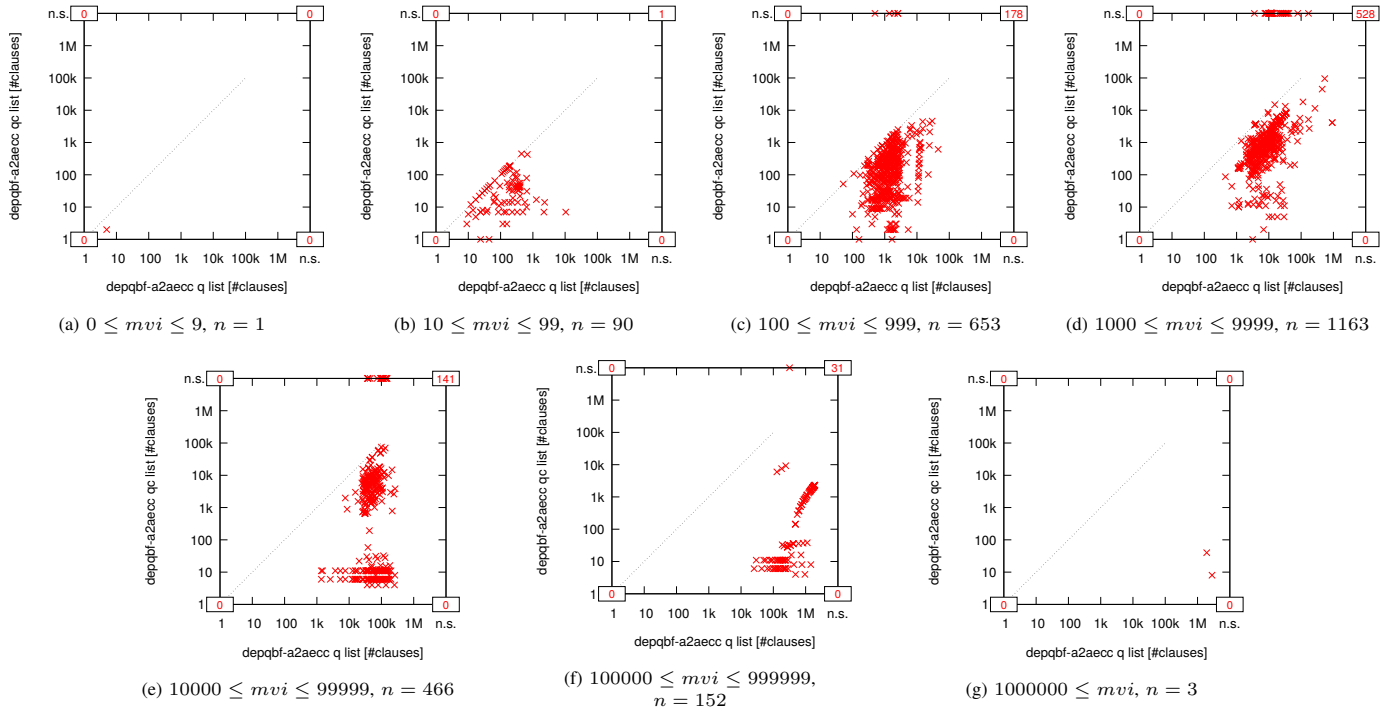


Fig. 463: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode qc list partitioned by maximum variable index (number of clauses).

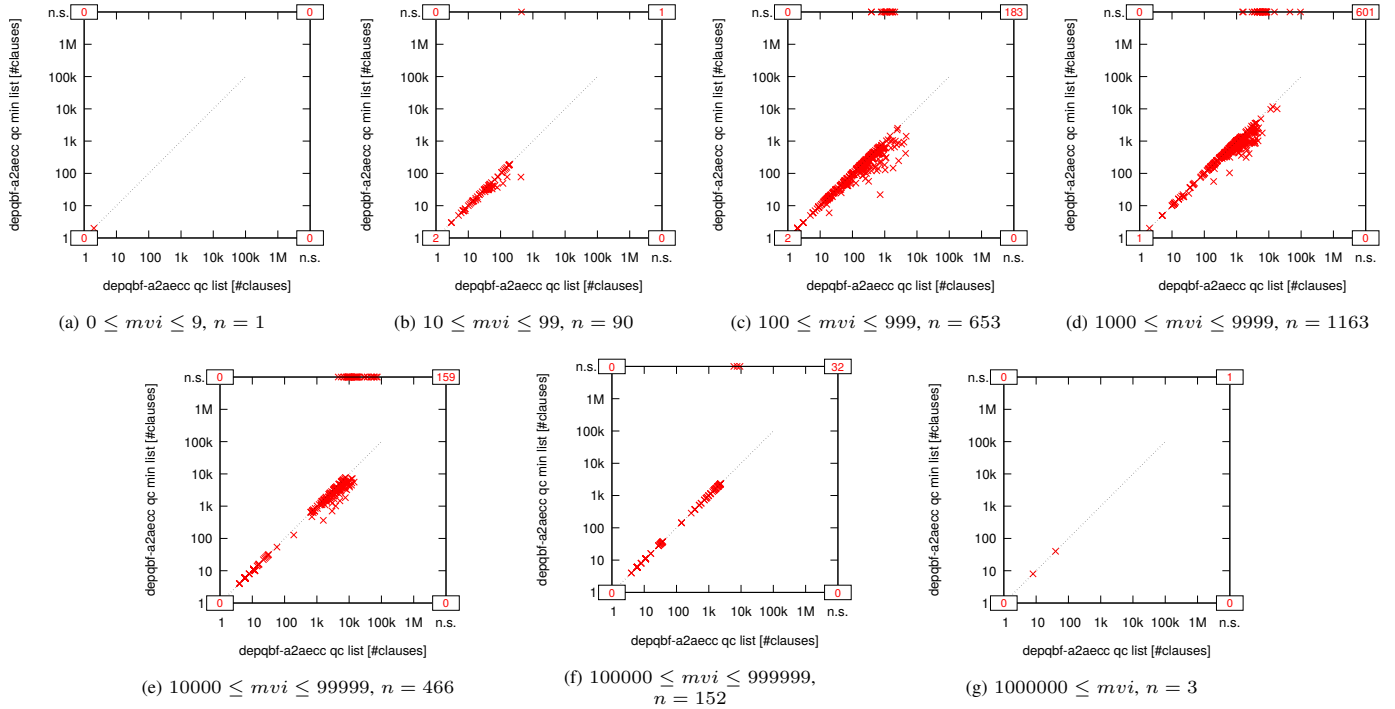


Fig. 464: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode qc min list partitioned by maximum variable index (number of clauses).

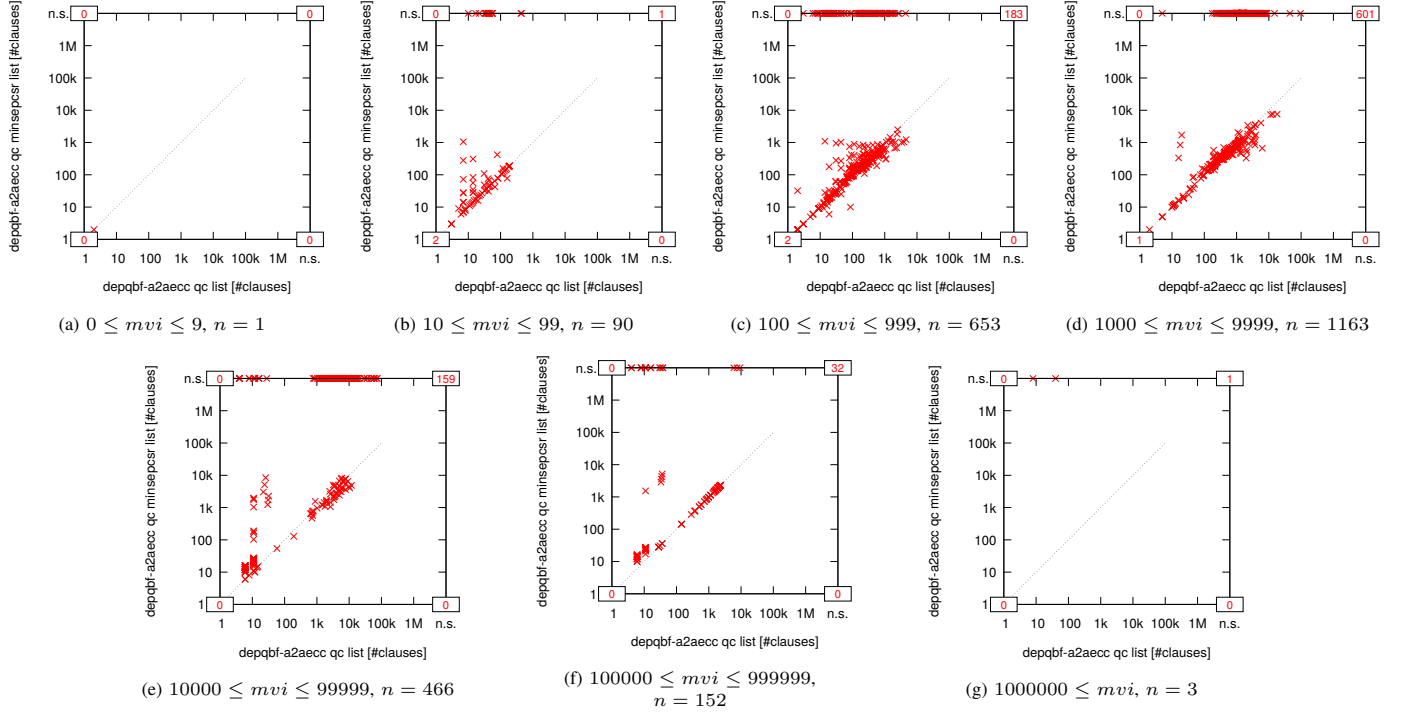


Fig. 465: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode qc minsepcsr list partitioned by maximum variable index (number of clauses).

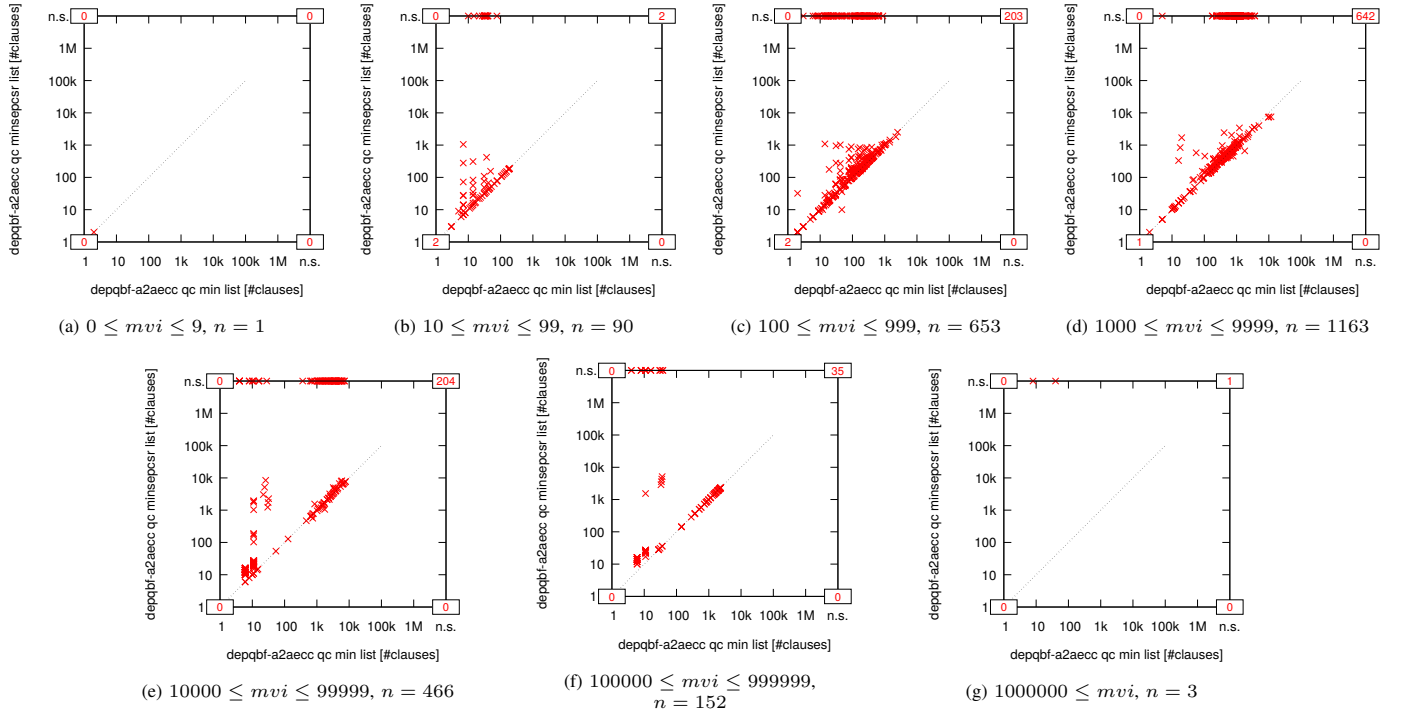


Fig. 466: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode qc minsepcsr list partitioned by maximum variable index (number of clauses).

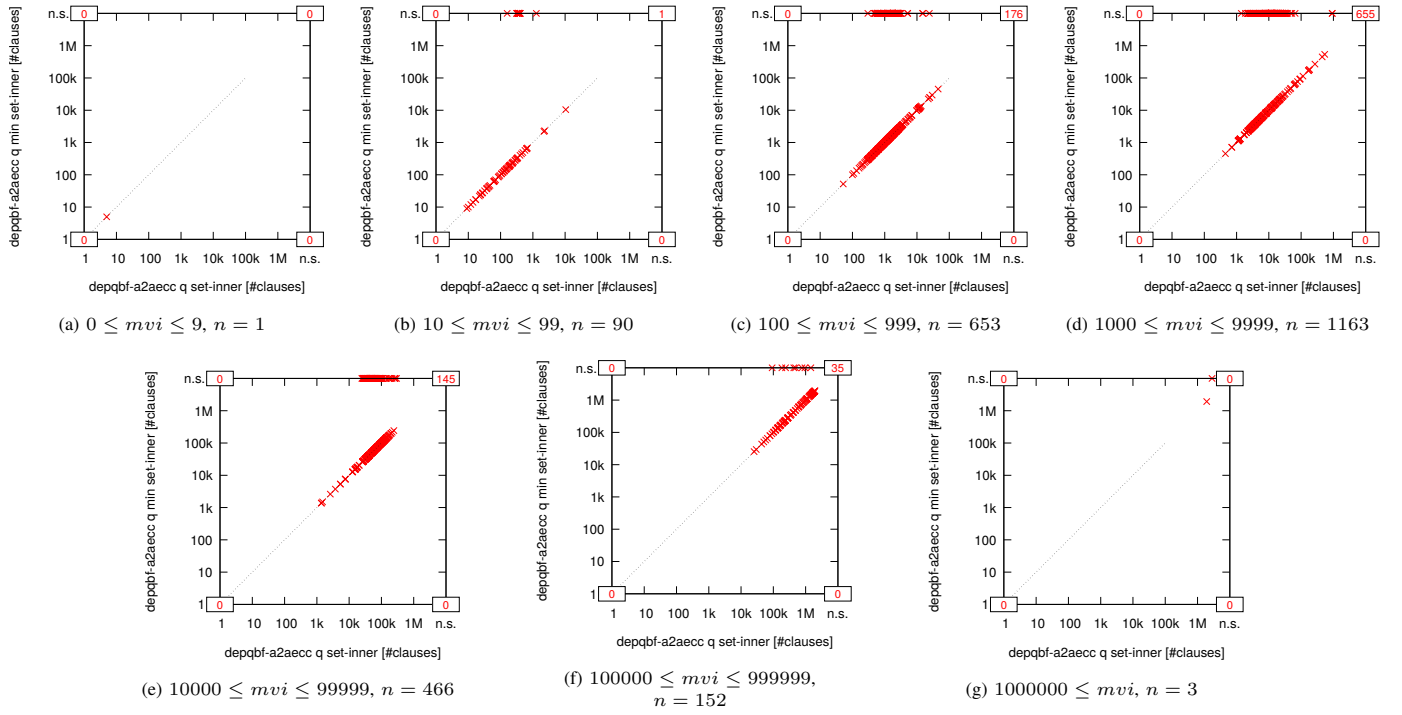


Fig. 467: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode q min set-inner partitioned by maximum variable index (number of clauses).

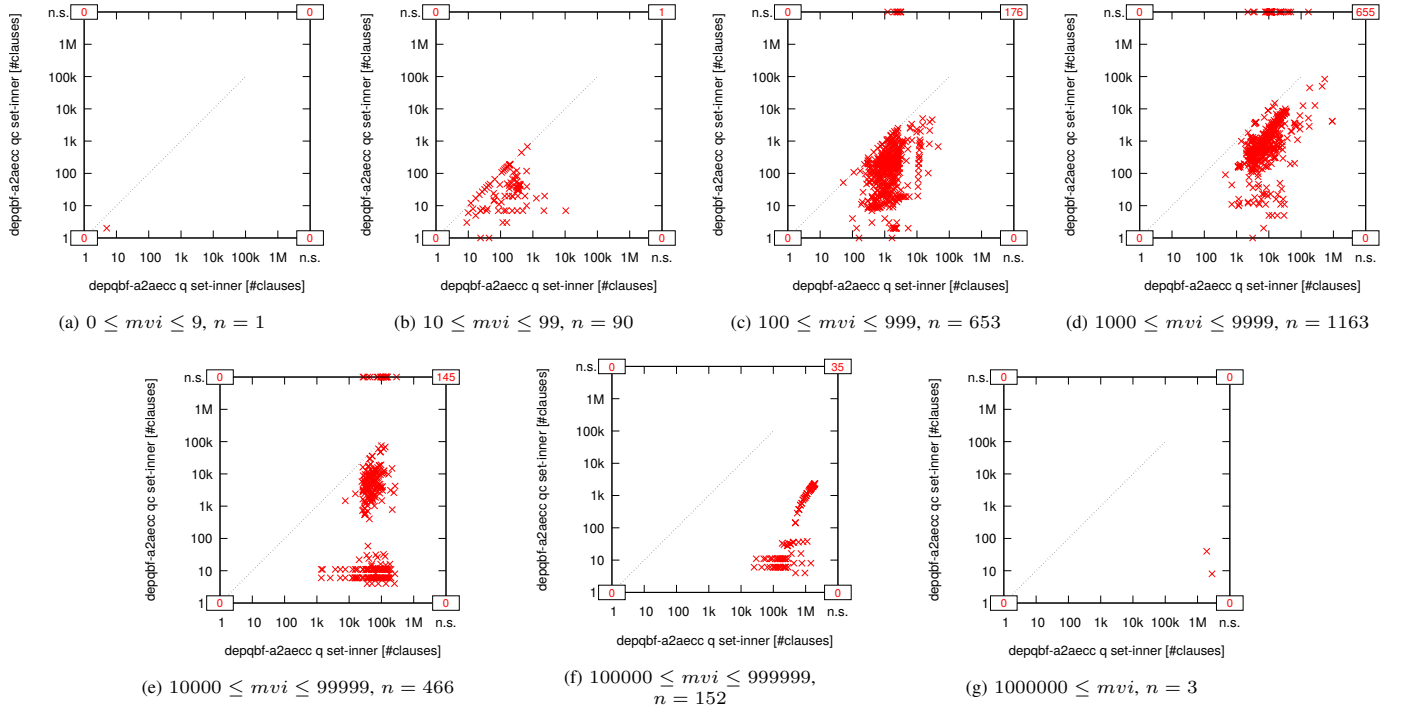


Fig. 468: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode qc set-inner partitioned by maximum variable index (number of clauses).

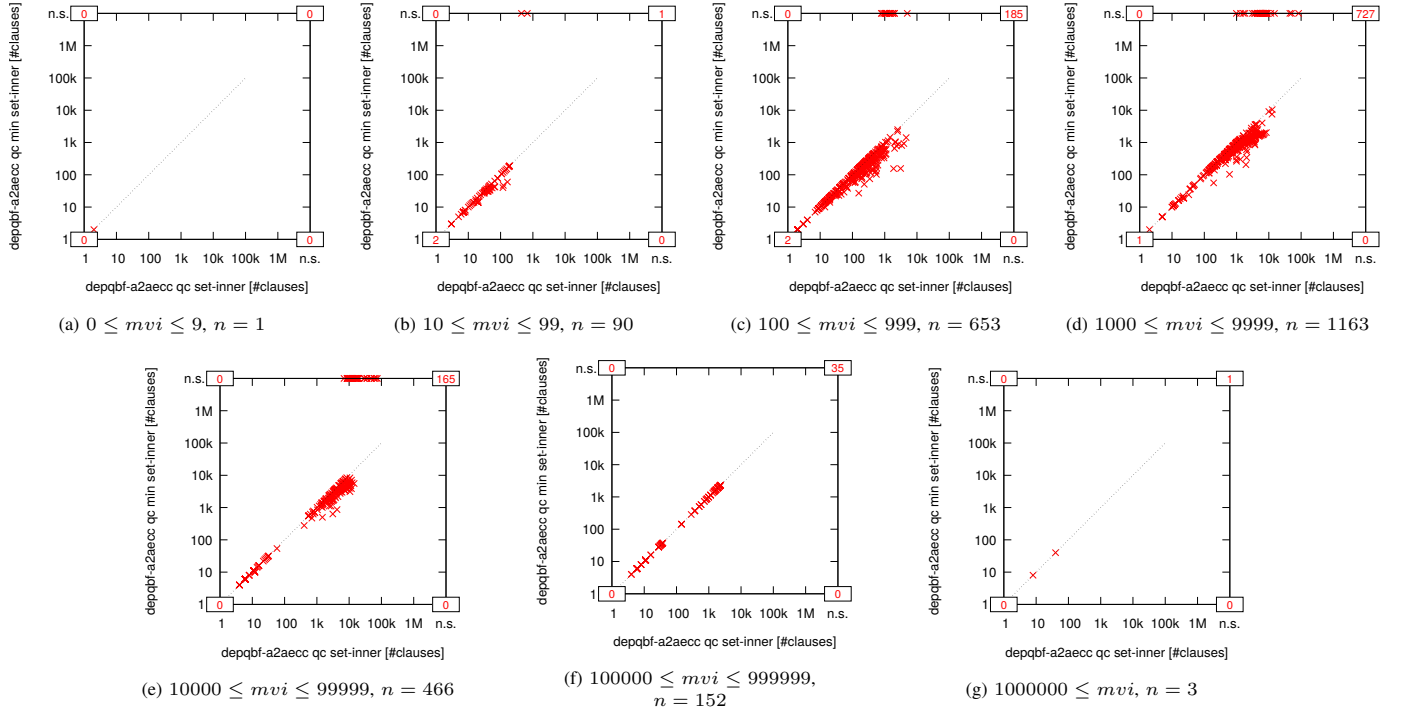


Fig. 469: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode qc min set-inner partitioned by maximum variable index (number of clauses).

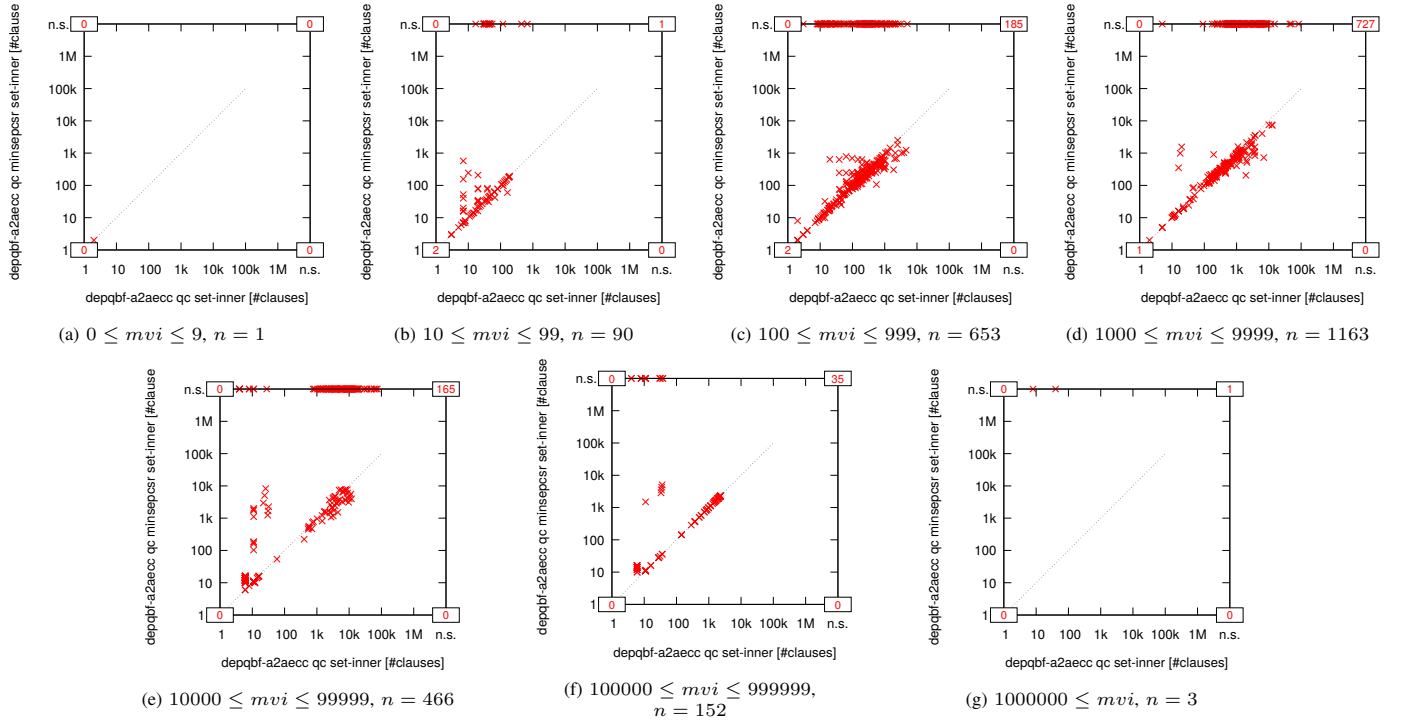


Fig. 470: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode qc minsepcsr set-inner partitioned by maximum variable index (number of clauses).

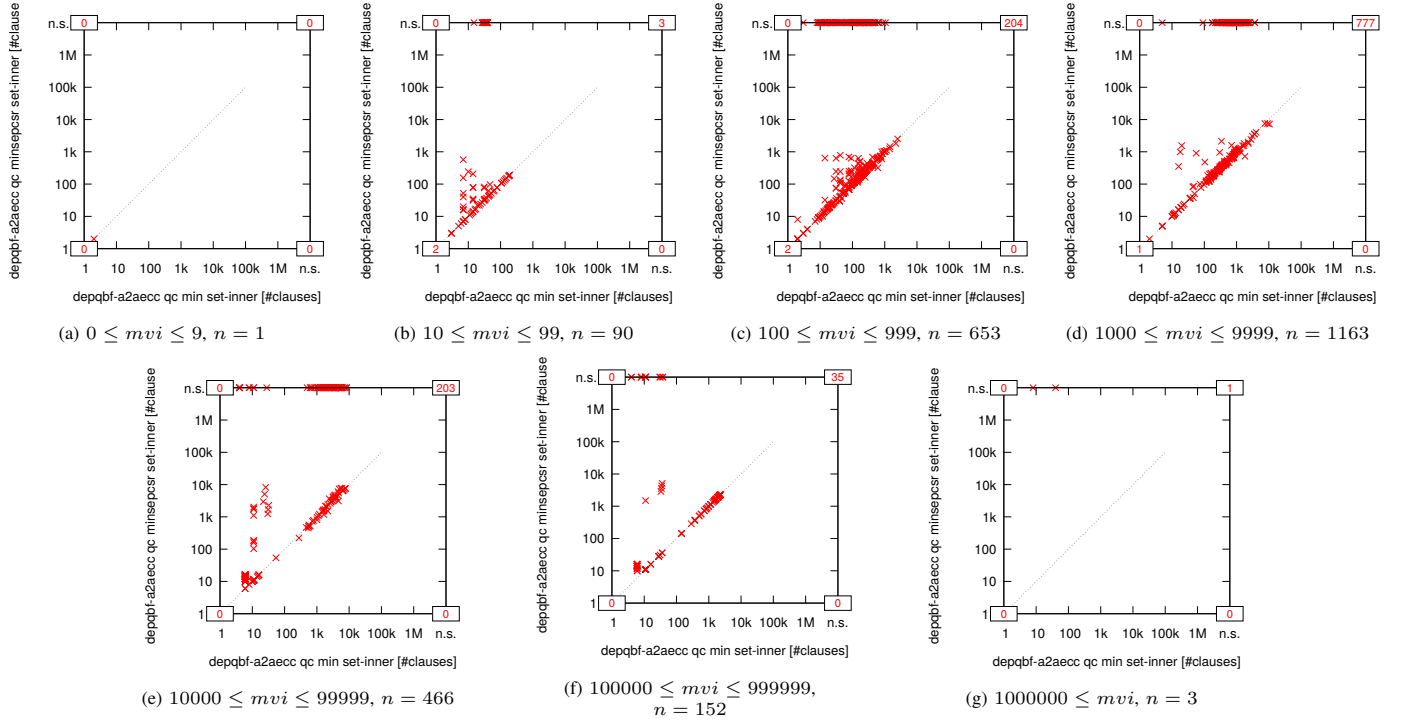


Fig. 471: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode qc minsepcsr set-inner partitioned by maximum variable index (number of clauses).

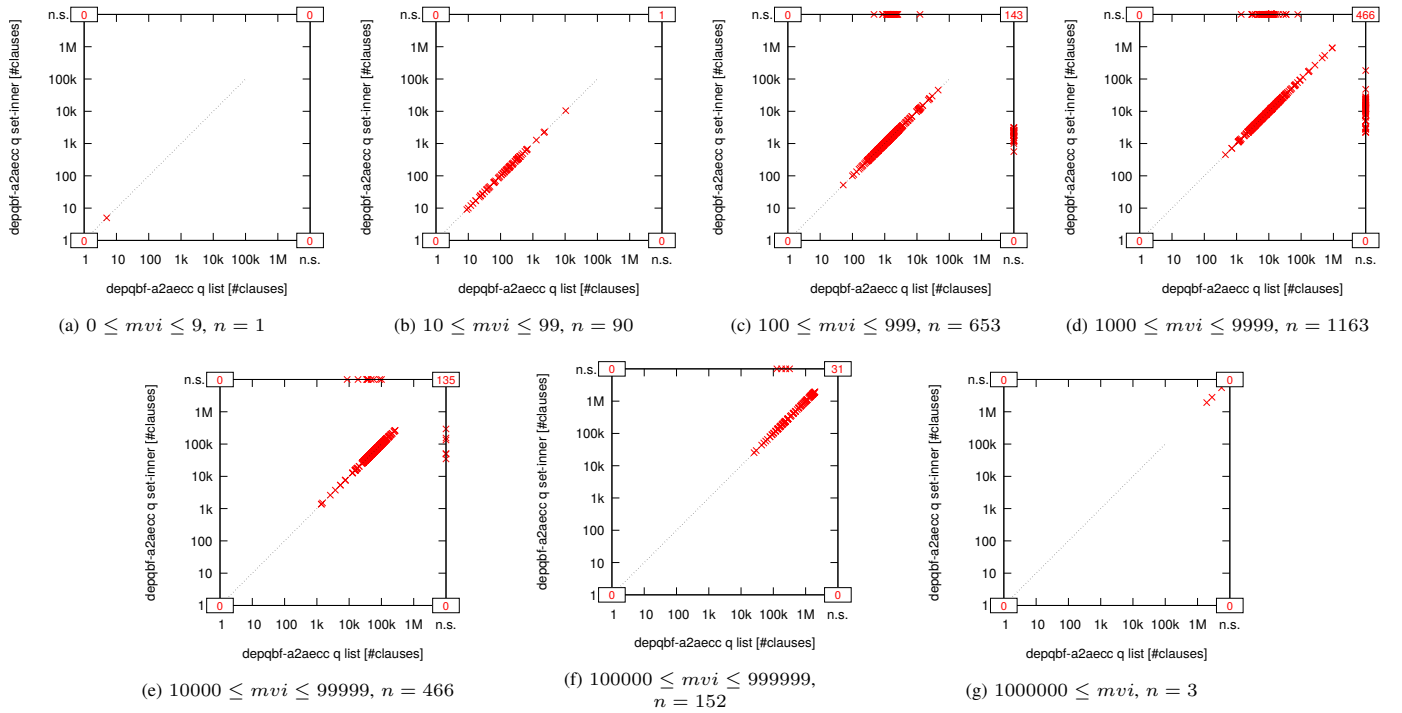


Fig. 472: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode q set-inner partitioned by maximum variable index (number of clauses).

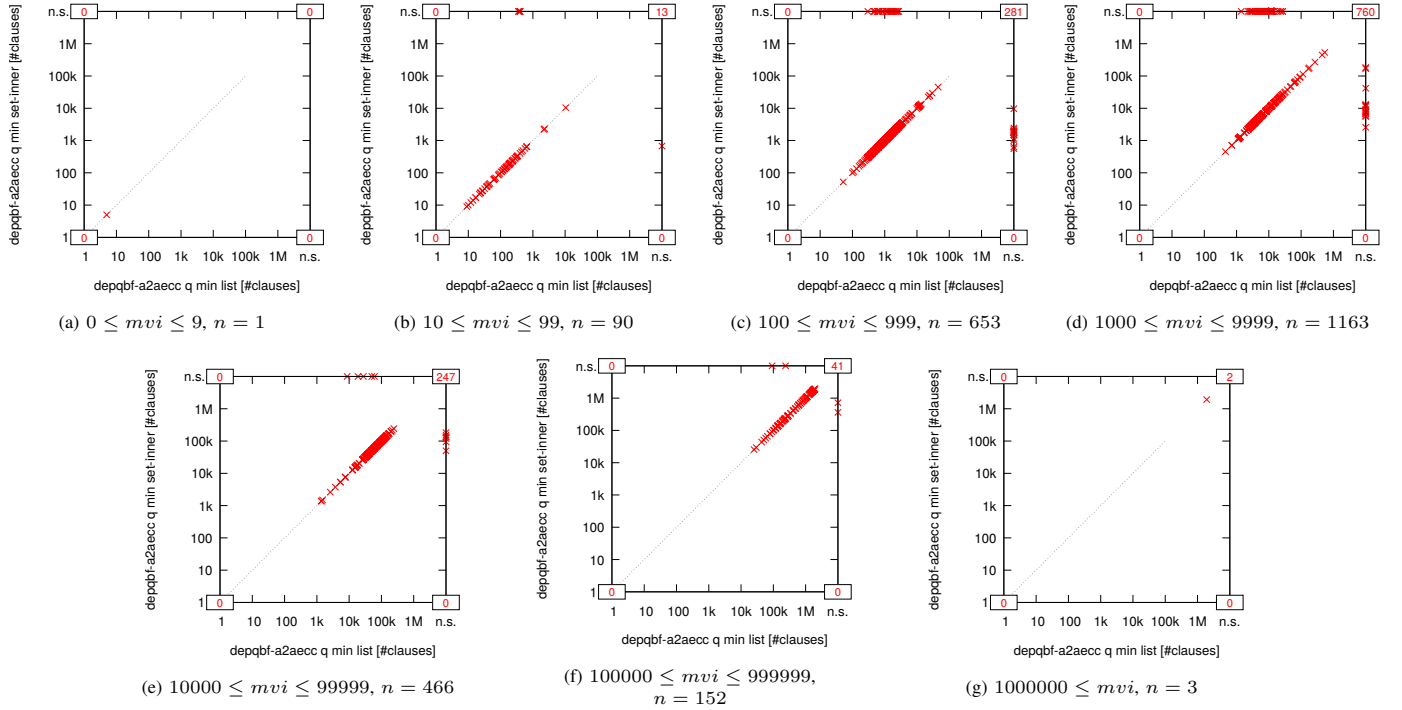


Fig. 473: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q min list versus mode q min set-inner partitioned by maximum variable index (number of clauses).

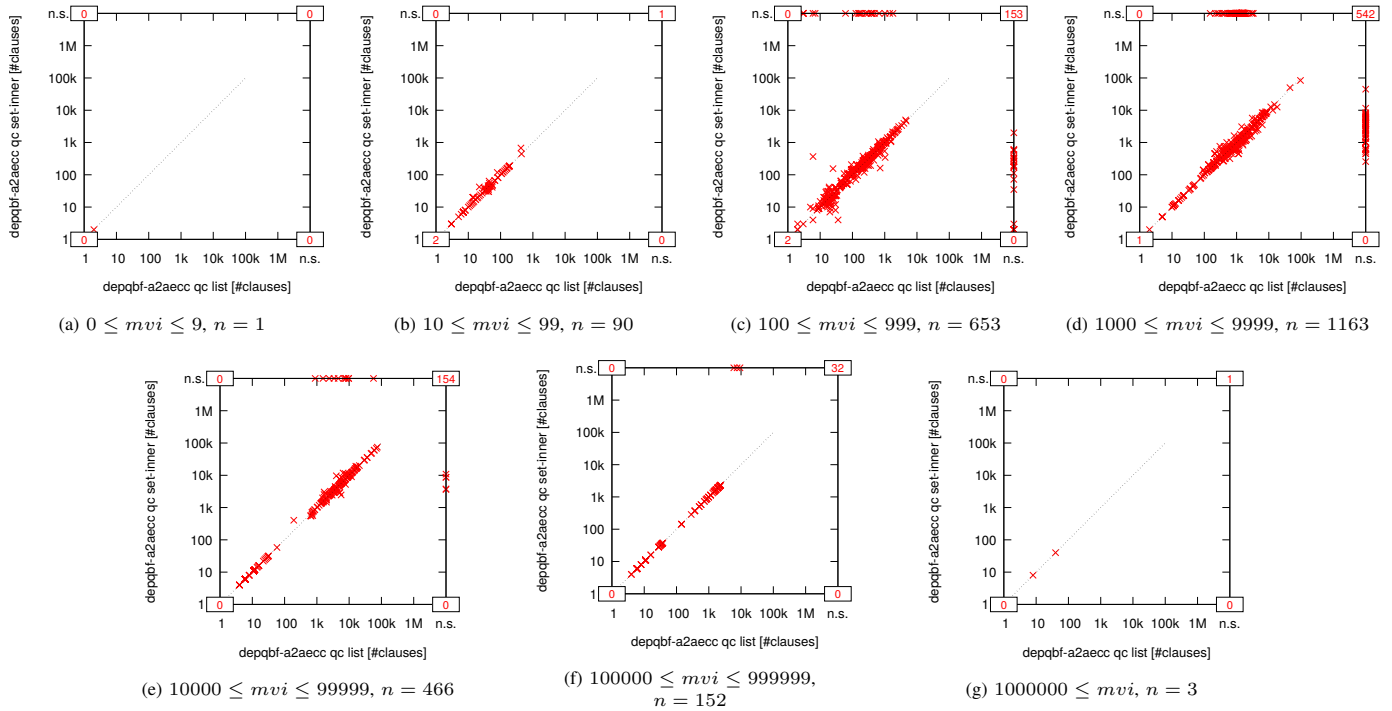


Fig. 474: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode qc set-inner partitioned by maximum variable index (number of clauses).

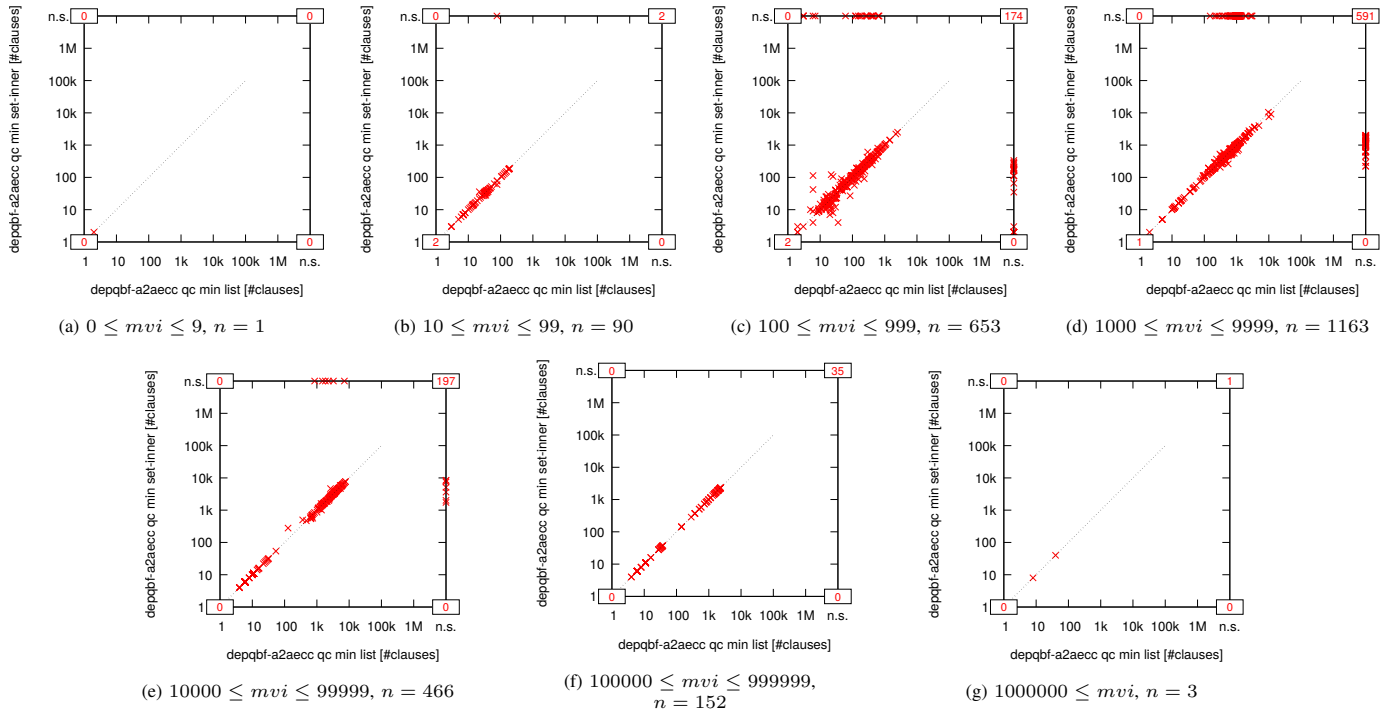


Fig. 475: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode qc min set-inner partitioned by maximum variable index (number of clauses).

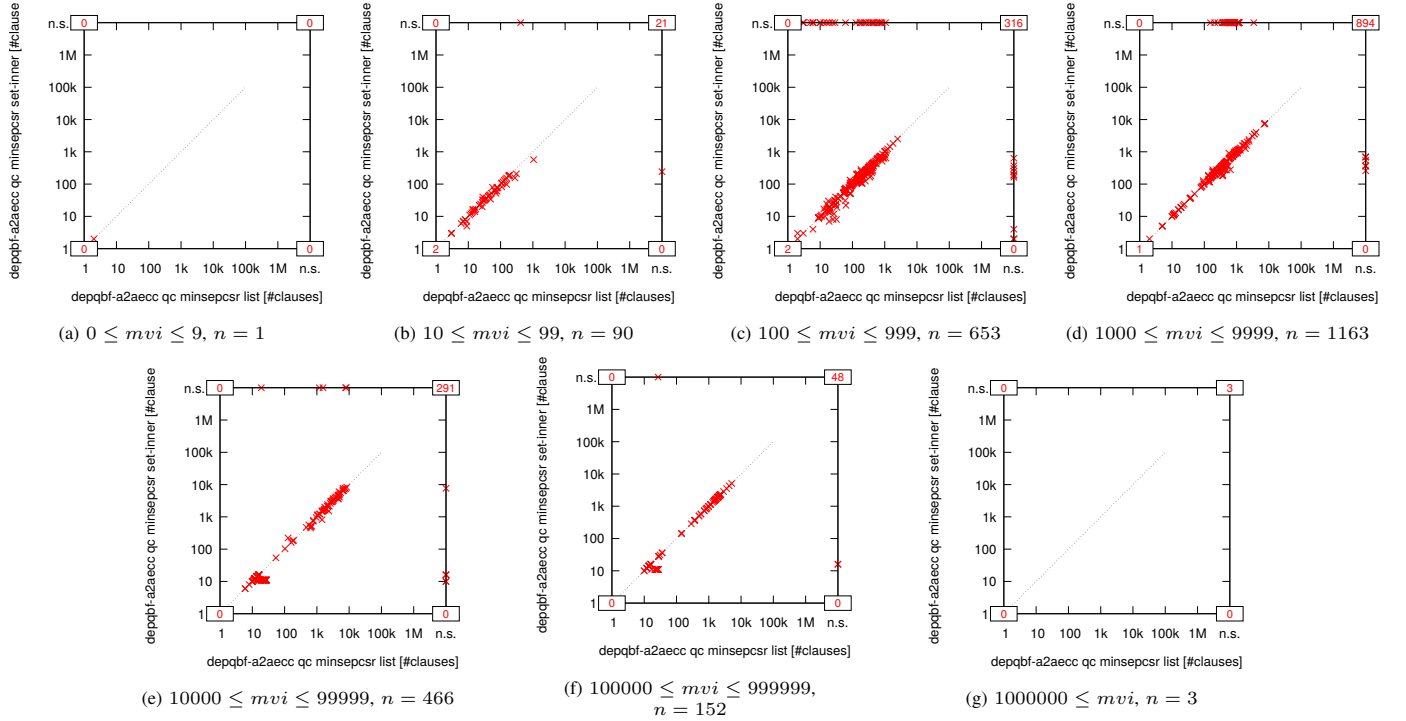


Fig. 476: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode qc minsepcsr set-inner partitioned by maximum variable index (number of clauses).

F. Partitioned by Number of Weakened \forall in Mode qc minsepcsr list

This subsection shows figures of the sizes of unsatisfiable cores with subfigures for partitions of the benchmarks according to their number of weakened \forall in mode qc minsepcsr list.

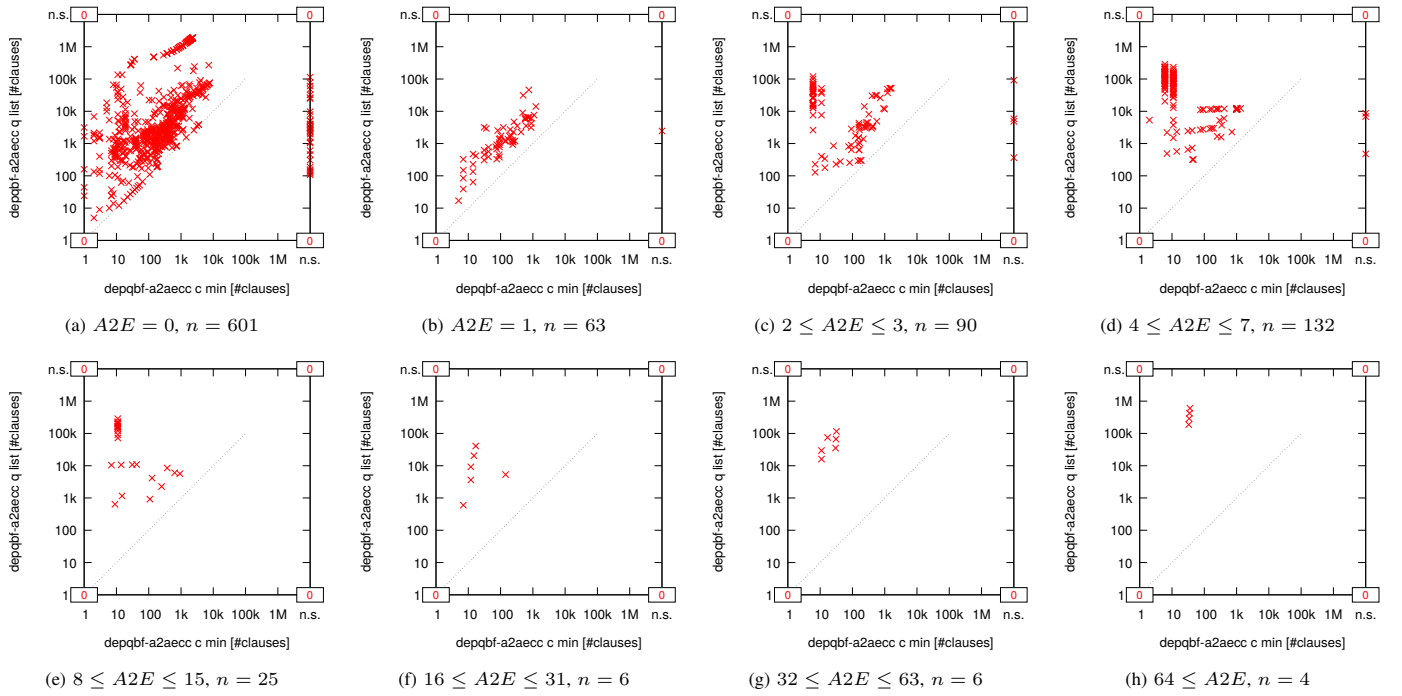


Fig. 477: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q list partitioned by number of weakened \forall in mode qc minsepcsr list (number of clauses).

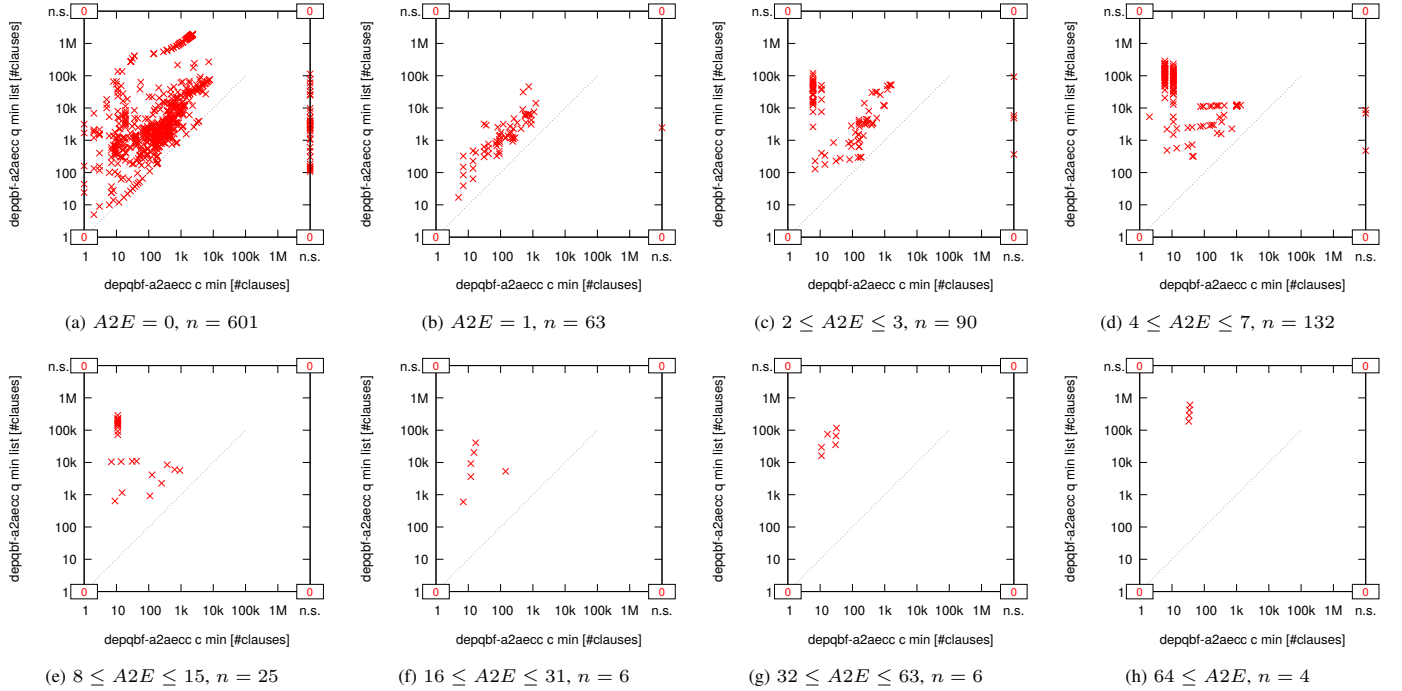


Fig. 478: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q min list partitioned by number of weakened \forall in mode qc minsepcsr list (number of clauses).

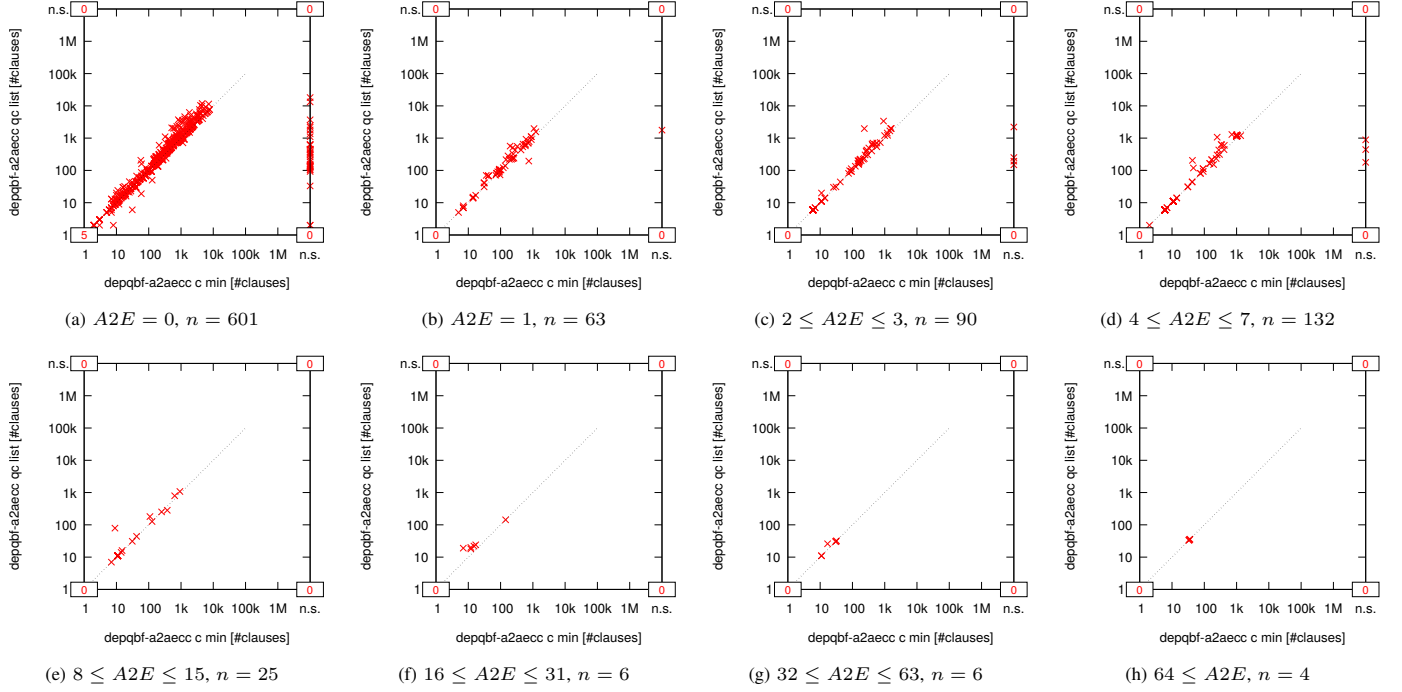


Fig. 479: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc list partitioned by number of weakened \forall in mode qc minsepcsr list (number of clauses).

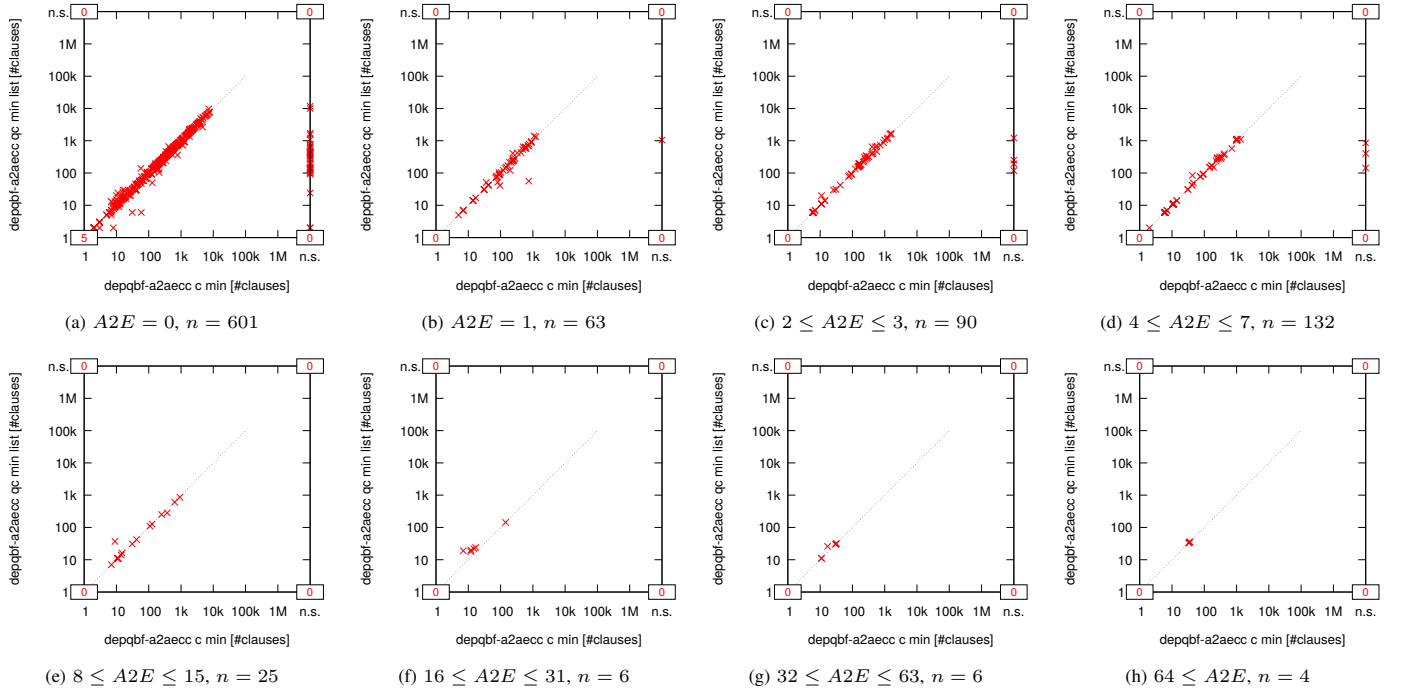


Fig. 480: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc min list partitioned by number of weakened \forall in mode qc minsepcsr list (number of clauses).

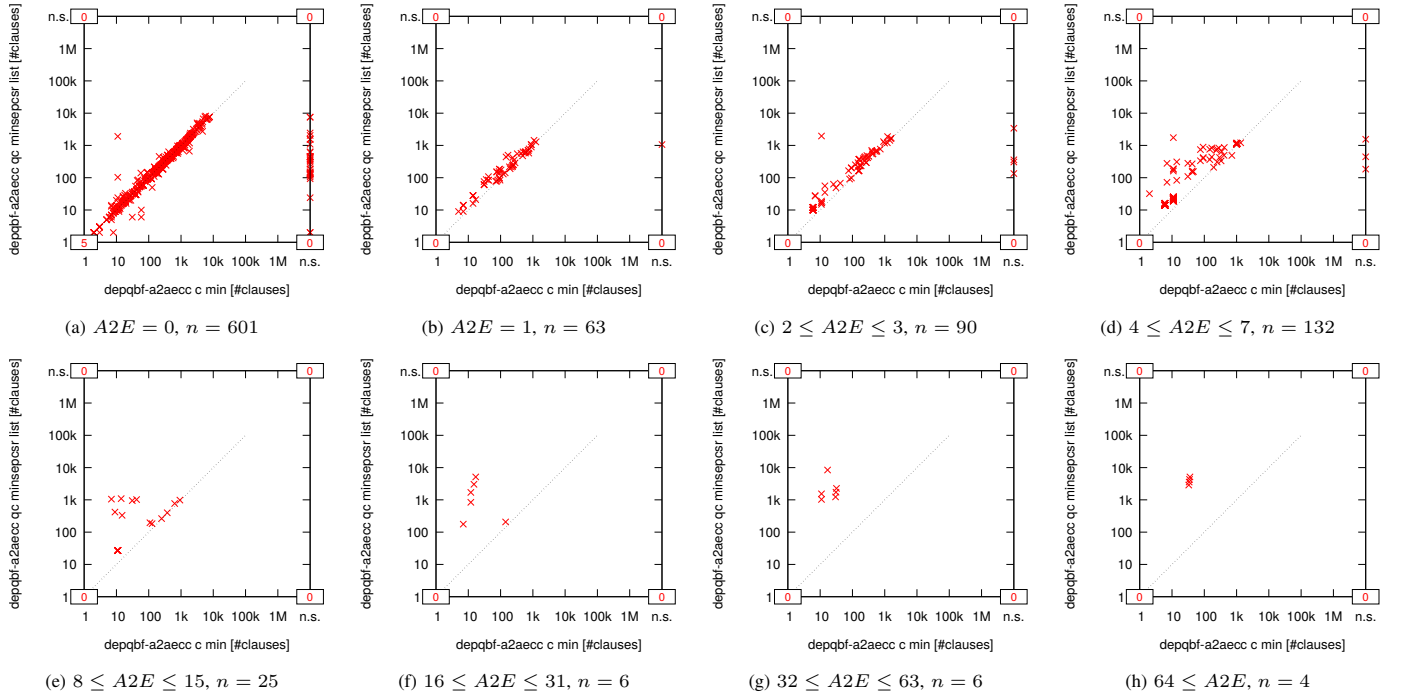


Fig. 481: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc minsepcsr list partitioned by number of weakened \forall in mode qc minsepcsr list (number of clauses).

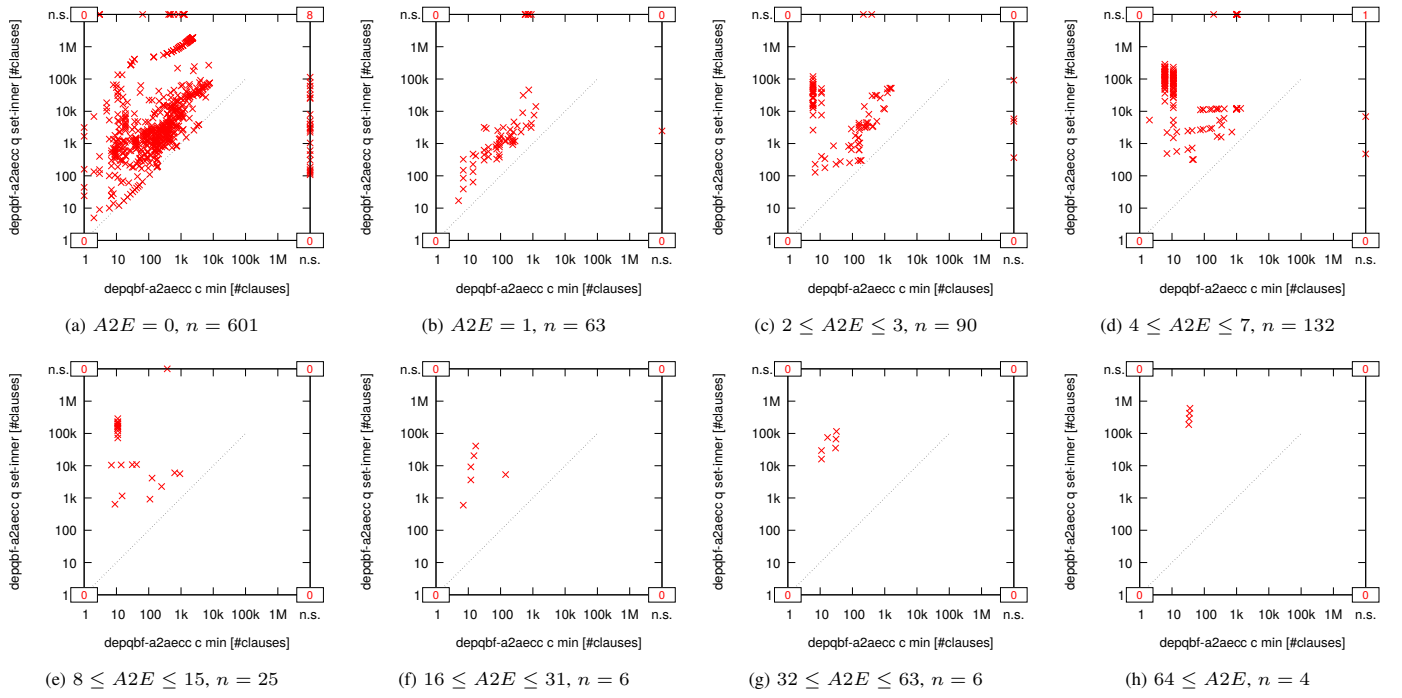


Fig. 482: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q set-inner partitioned by number of weakened \forall in mode qc minsepcsr list (number of clauses).

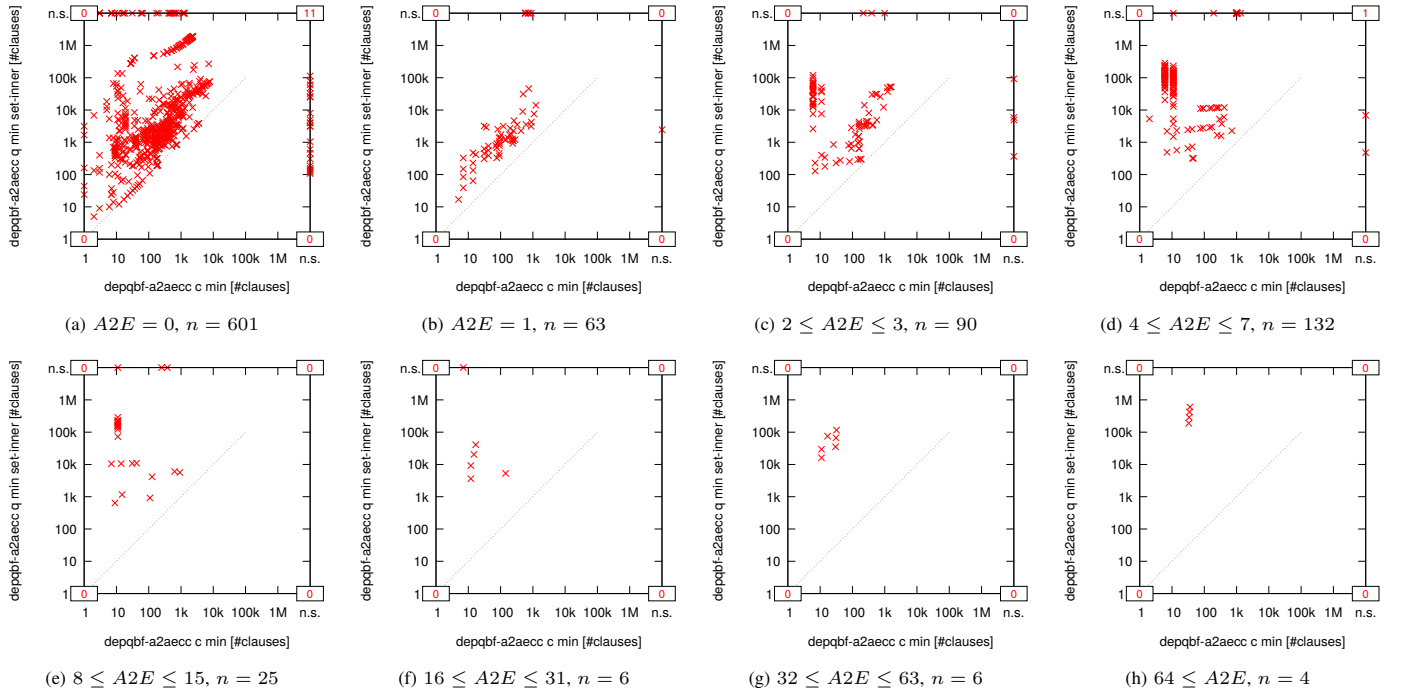


Fig. 483: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode q min set-inner partitioned by number of weakened \forall in mode qc minsepcsr list (number of clauses).

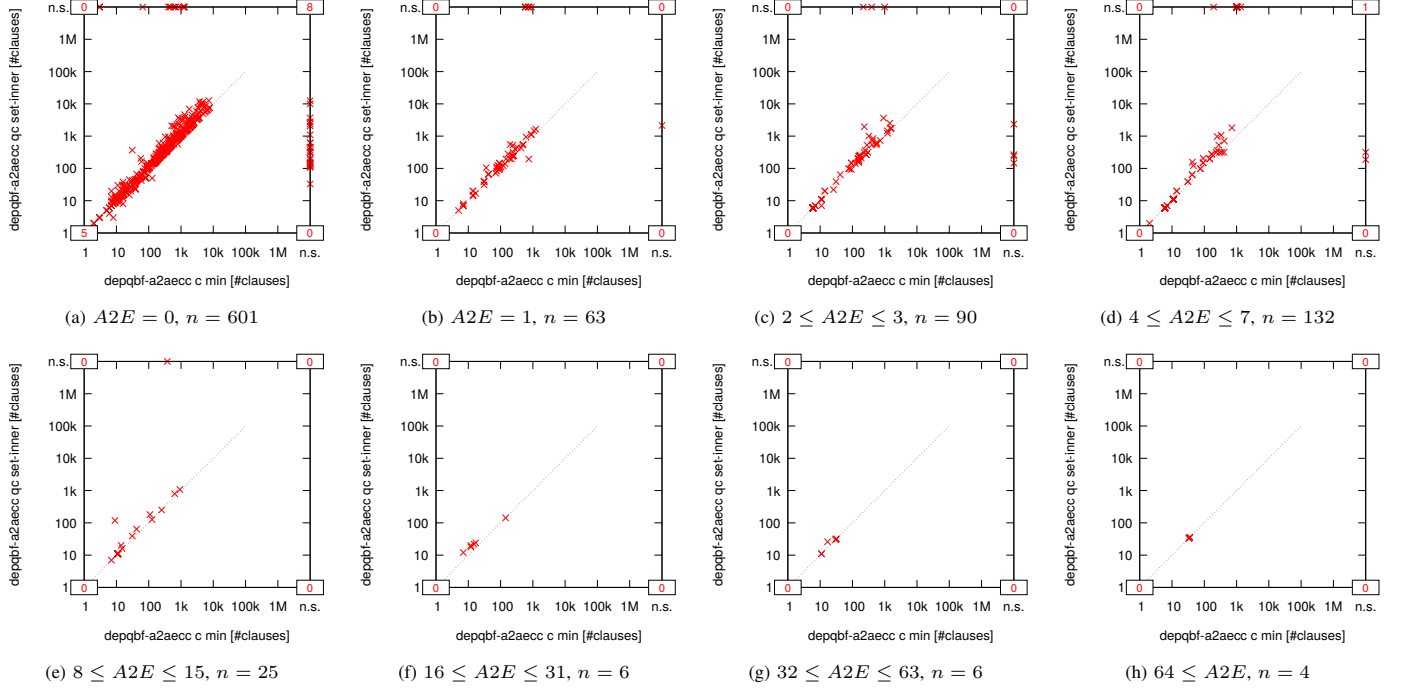


Fig. 484: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc set-inner partitioned by number of weakened ∇ in mode qc minsepcsr list (number of clauses).

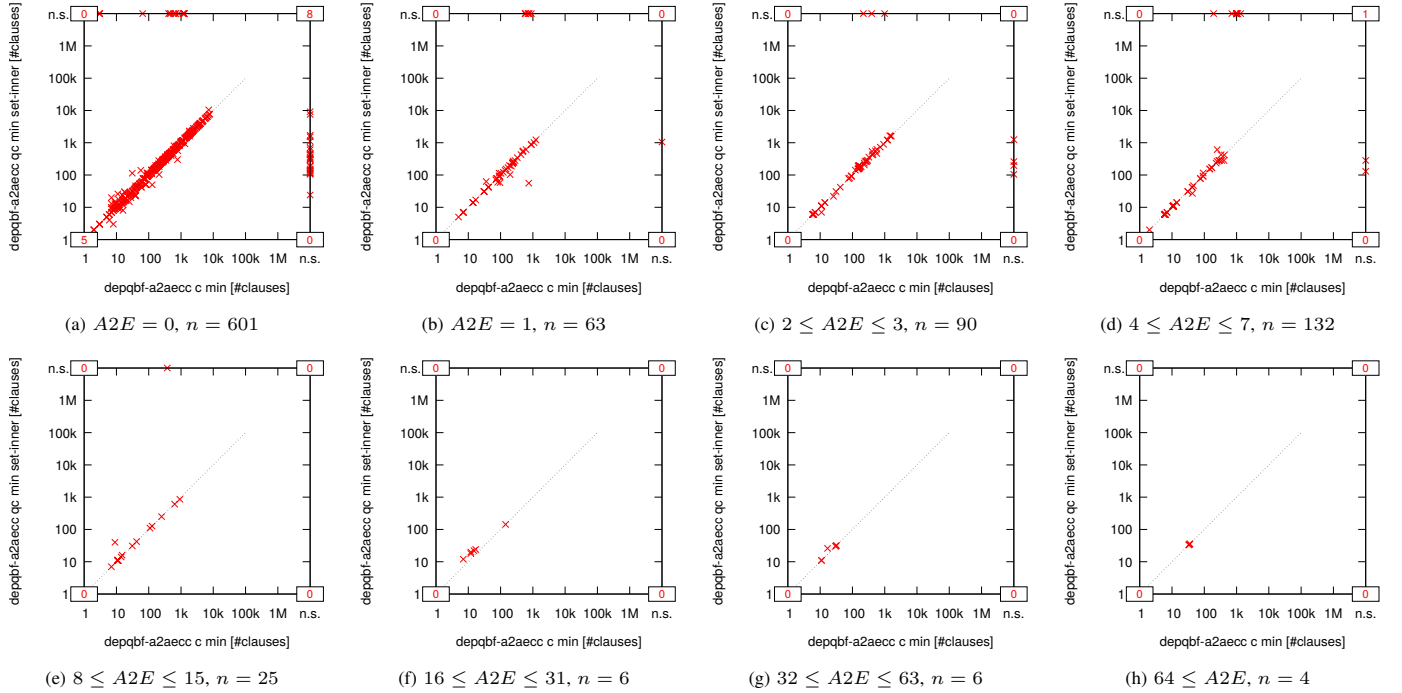


Fig. 485: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc min set-inner partitioned by number of weakened ∇ in mode qc minsepcsr list (number of clauses).

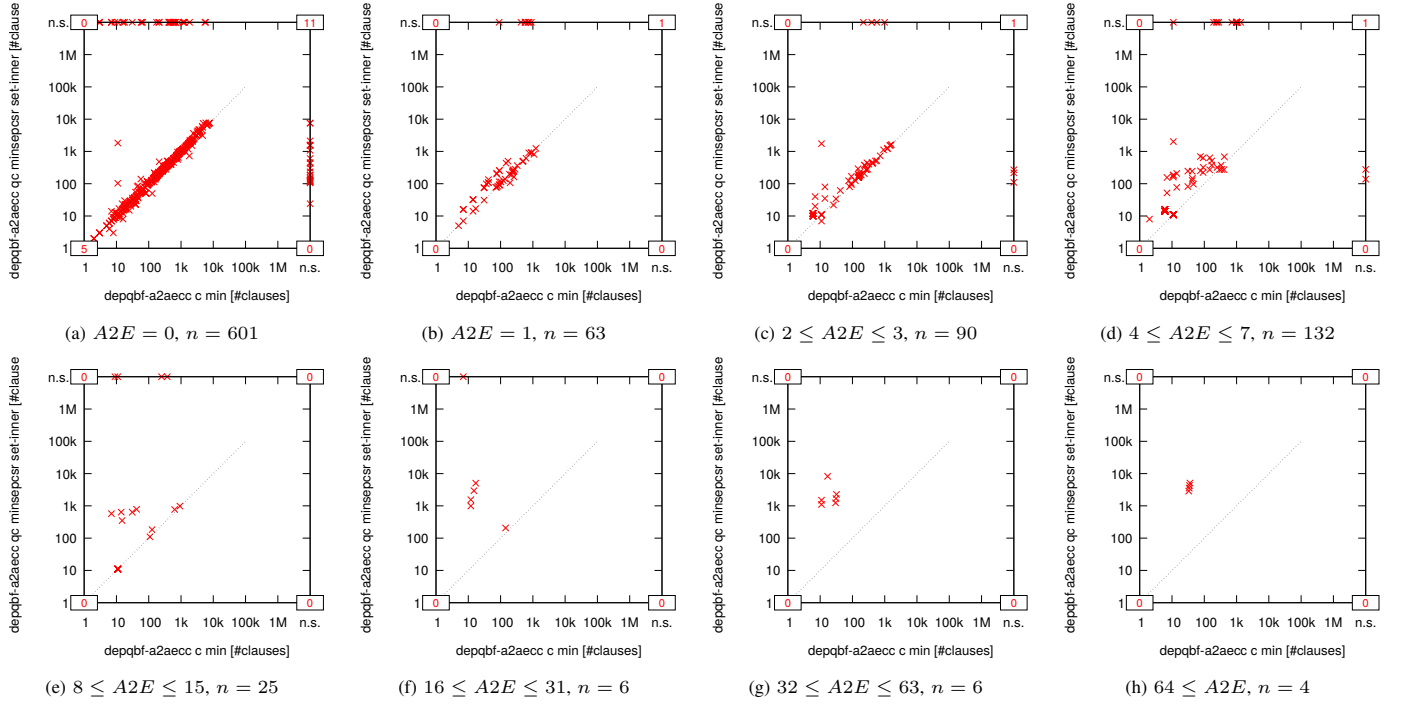


Fig. 486: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode qc minsepcsr set-inner partitioned by number of weakened \forall in mode qc minsepcsr list (number of clauses).

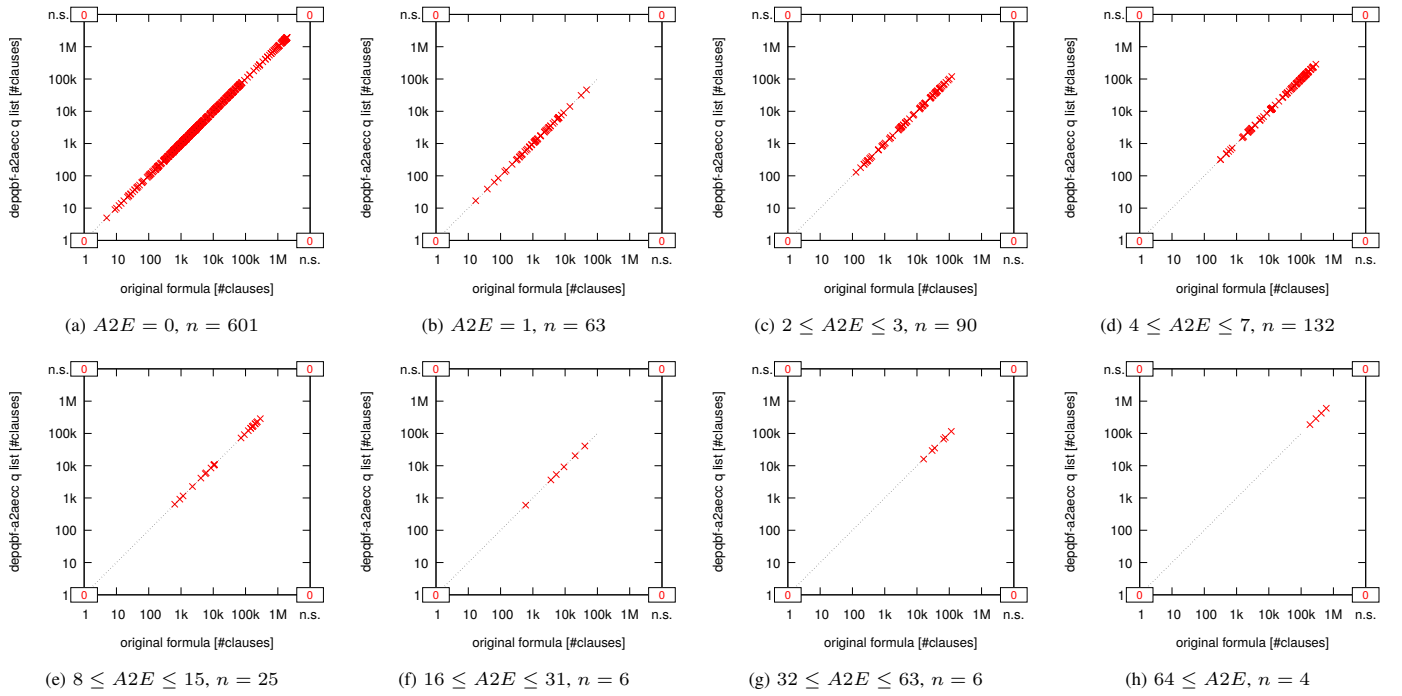


Fig. 487: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q list partitioned by number of weakened ∇ in mode qc minsepcsr list (number of clauses).

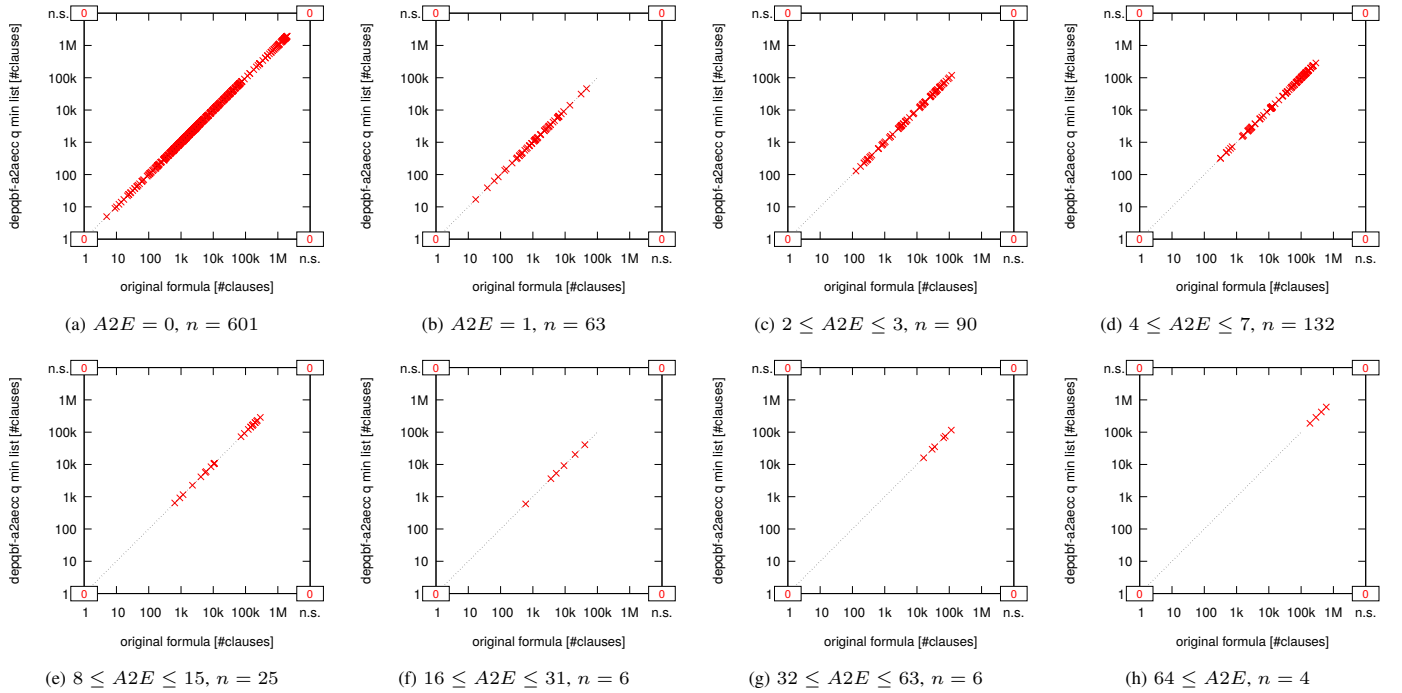


Fig. 488: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q min list partitioned by number of weakened ∇ in mode qc minsepcsr list (number of clauses).

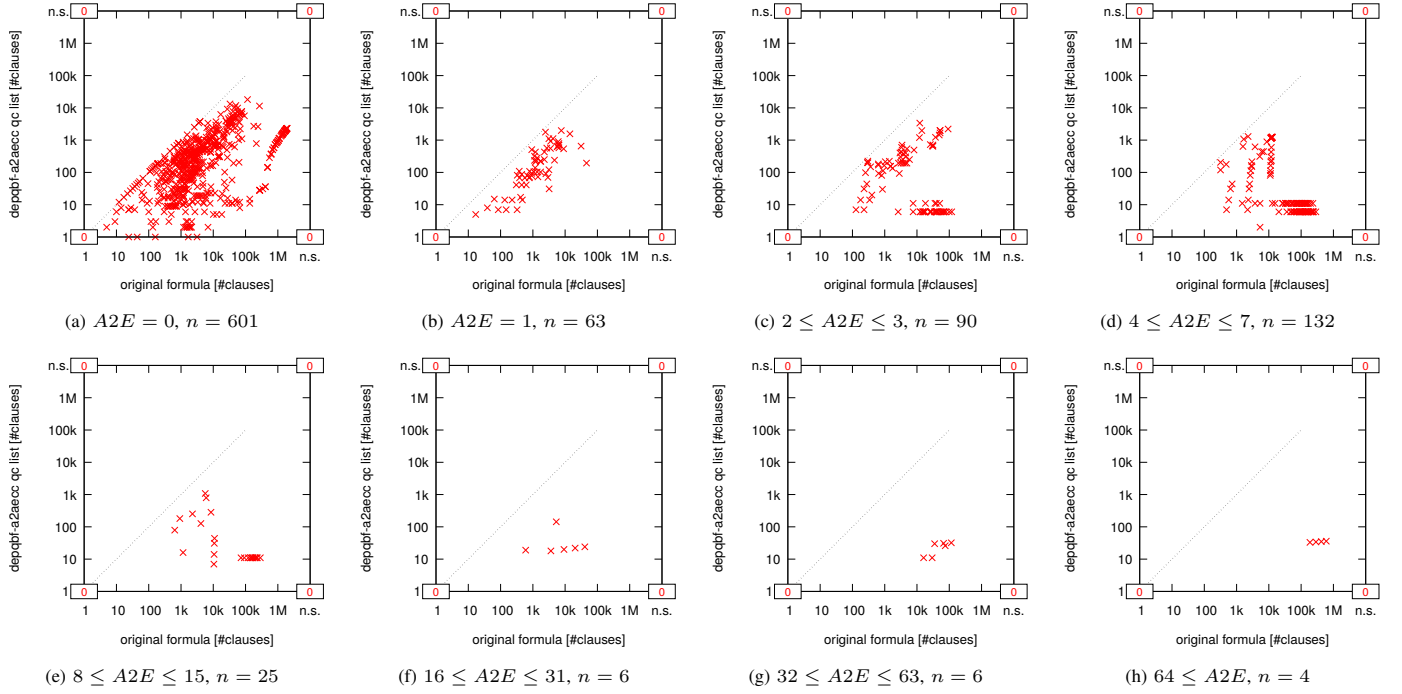


Fig. 489: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc list partitioned by number of weakened ∇ in mode qc minsepcsr list (number of clauses).

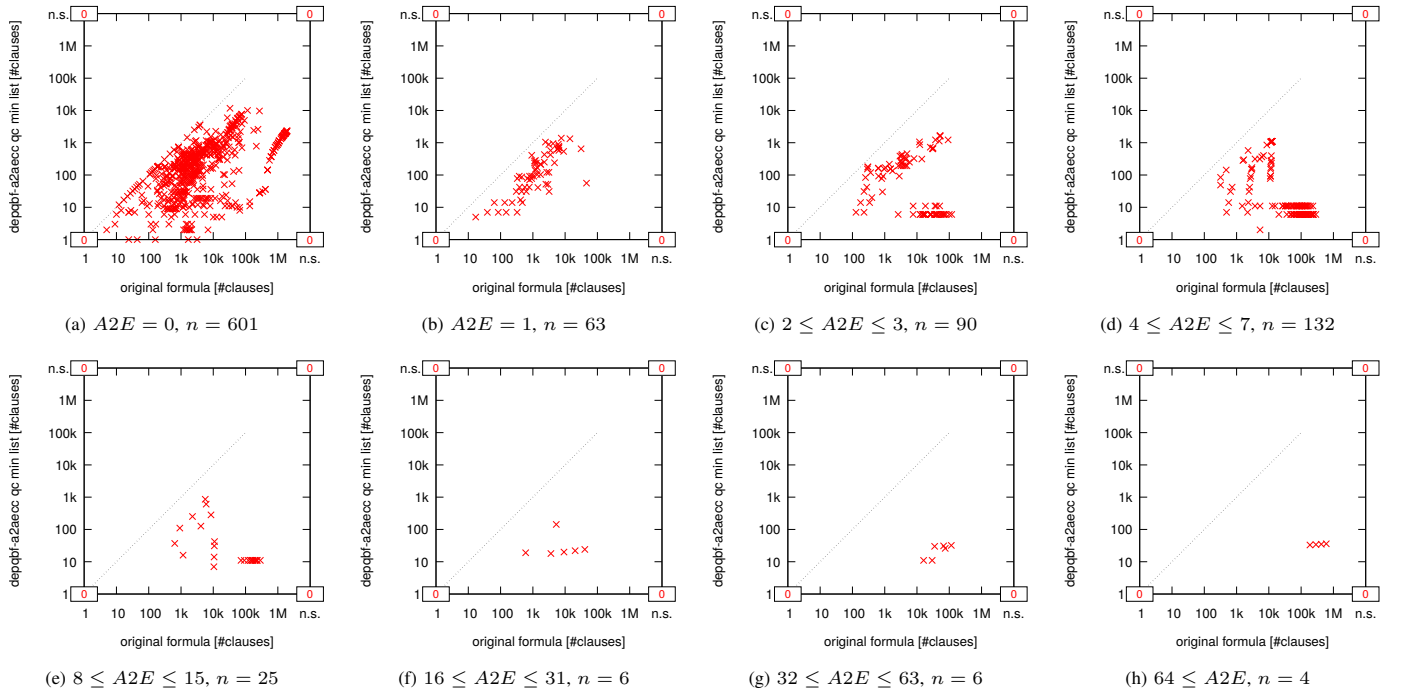


Fig. 490: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc min list partitioned by number of weakened ∇ in mode qc minsepcsr list (number of clauses).

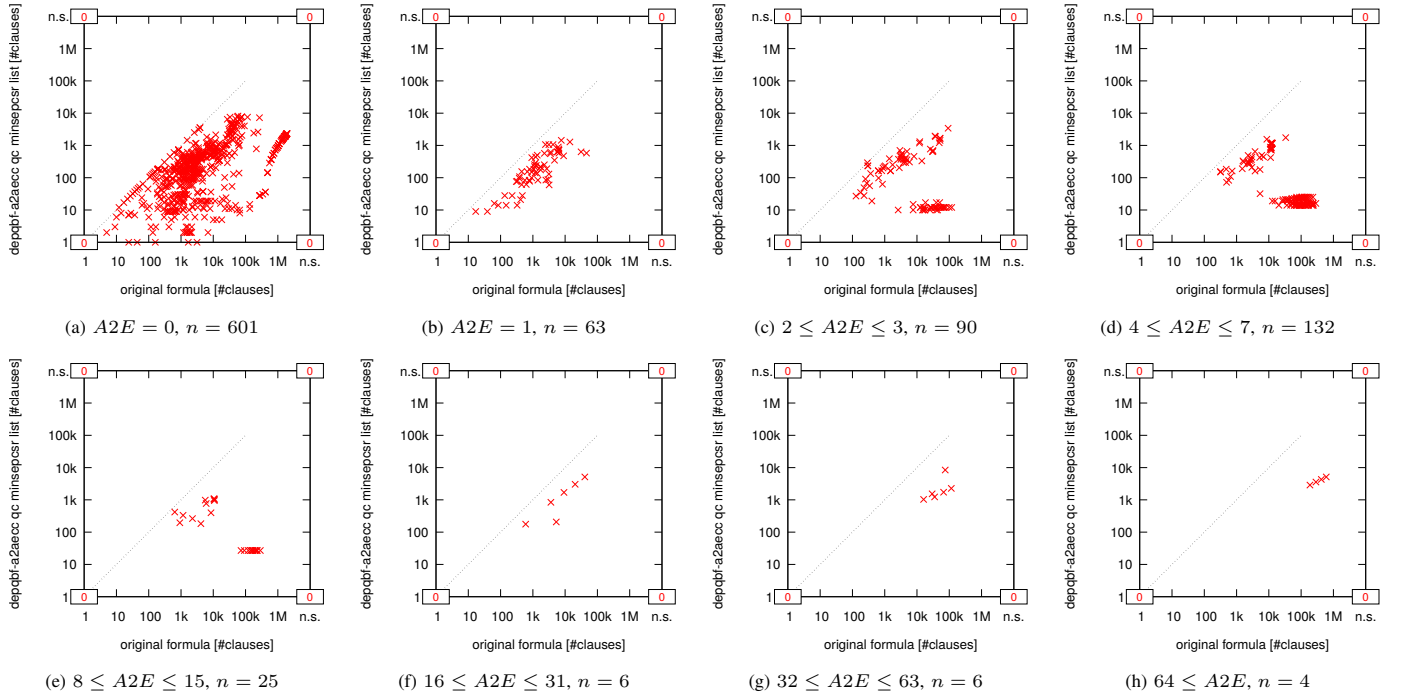


Fig. 491: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc minsepcsr list partitioned by number of weakened \forall in mode qc minsepcsr list (number of clauses).

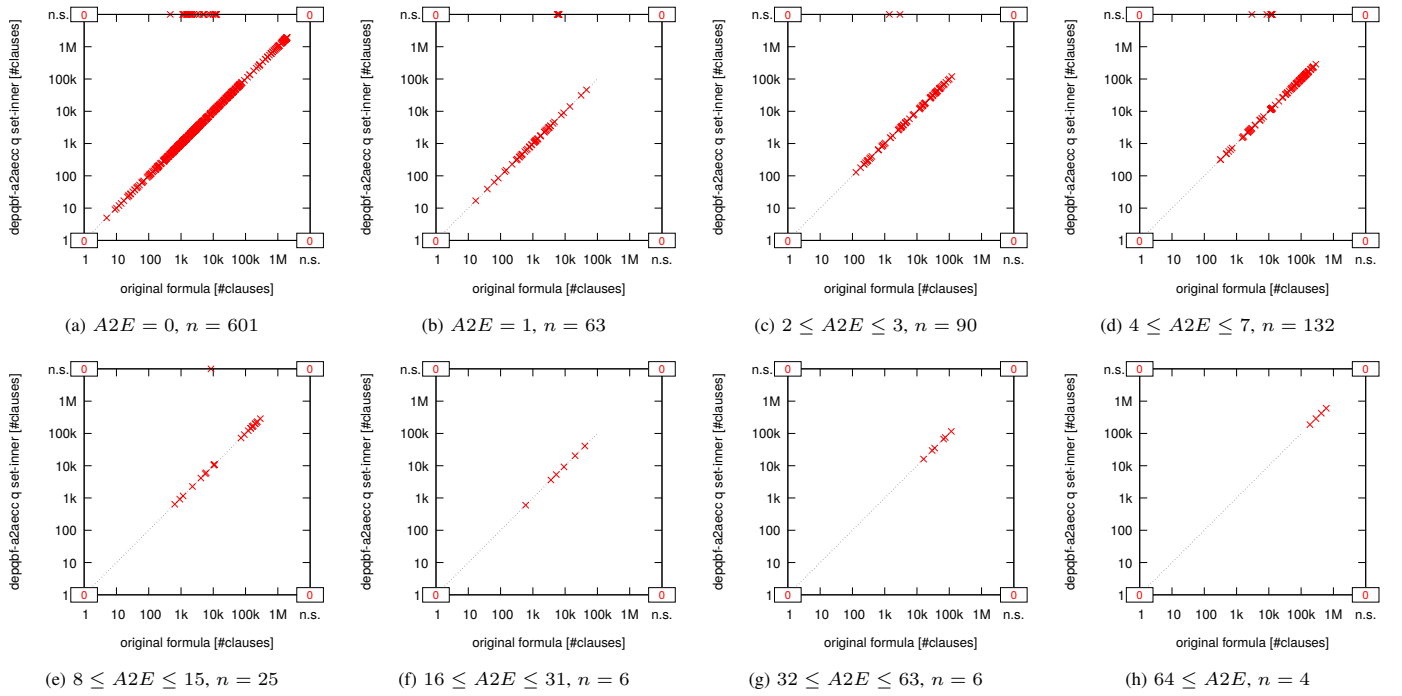


Fig. 492: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q set-inner partitioned by number of weakened \forall in mode qc minsepcsr list (number of clauses).

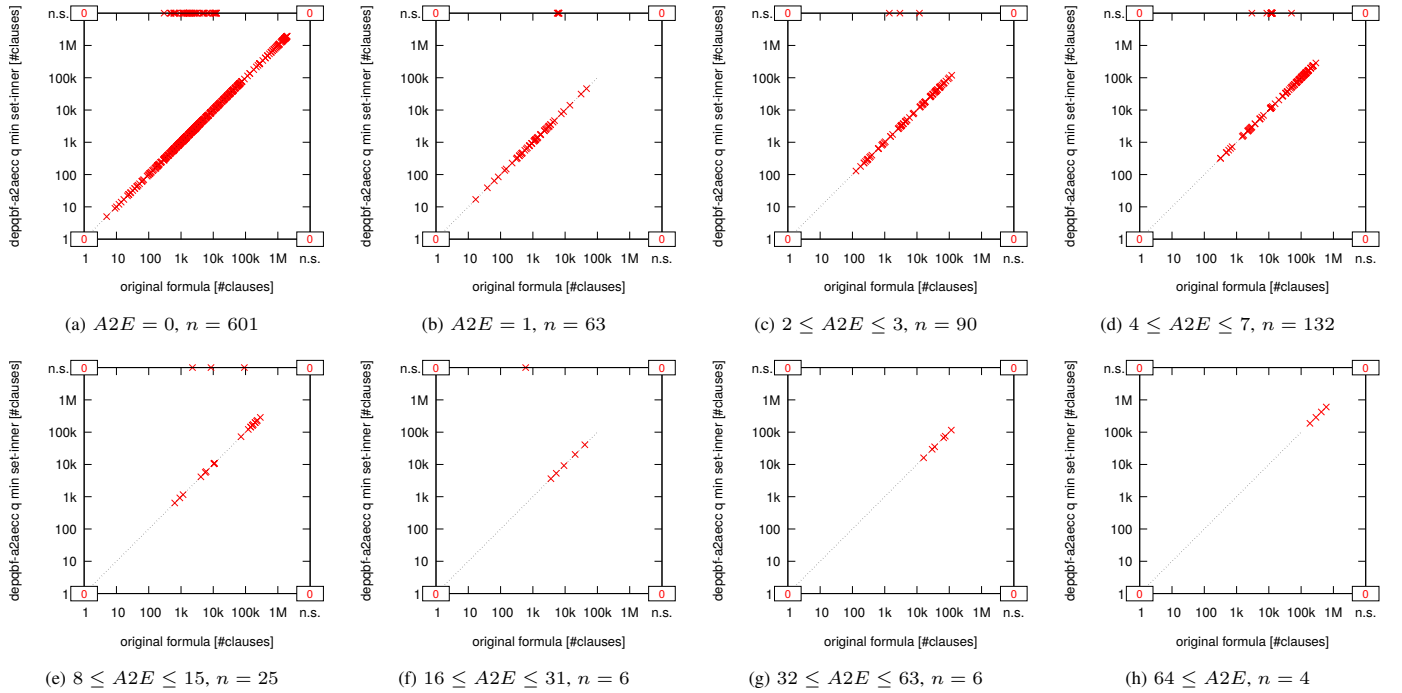


Fig. 493: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode q min set-inner partitioned by number of weakened \forall in mode qc minsepcsr list (number of clauses).

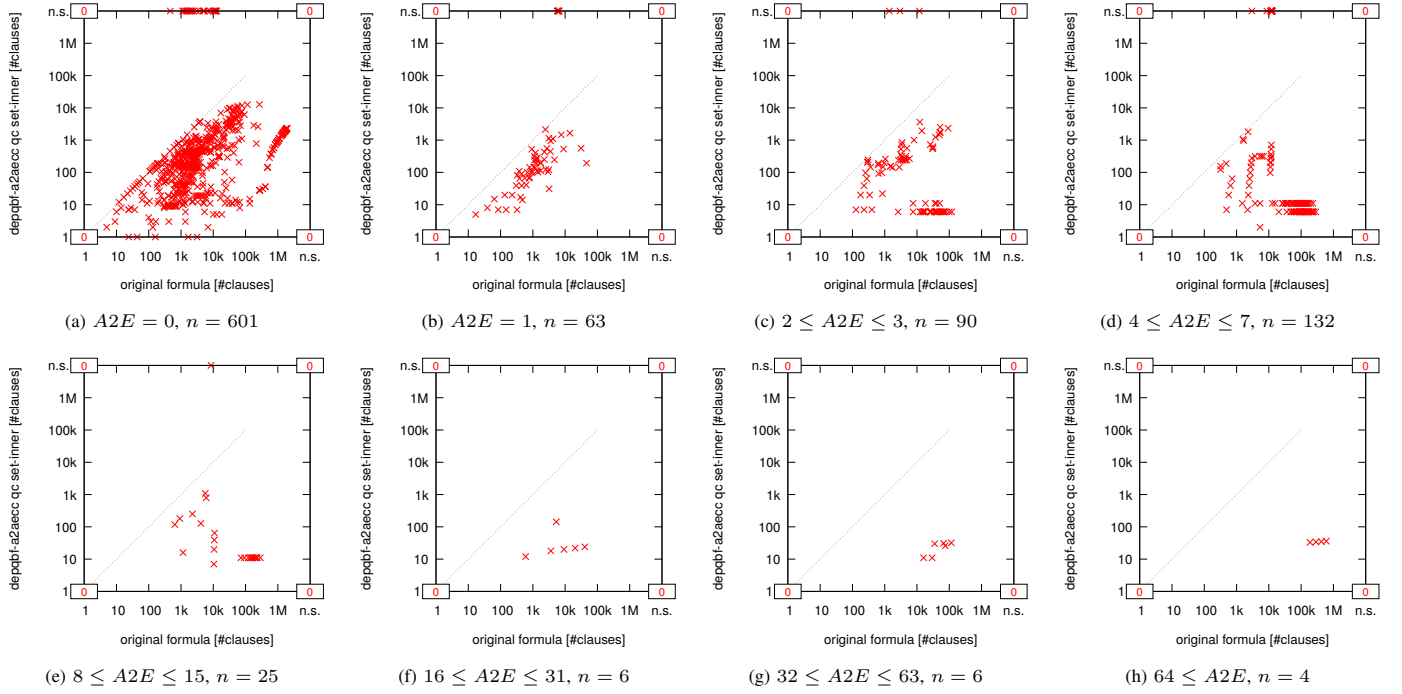


Fig. 494: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc set-inner partitioned by number of weakened \forall in mode qc minsepcsr list (number of clauses).

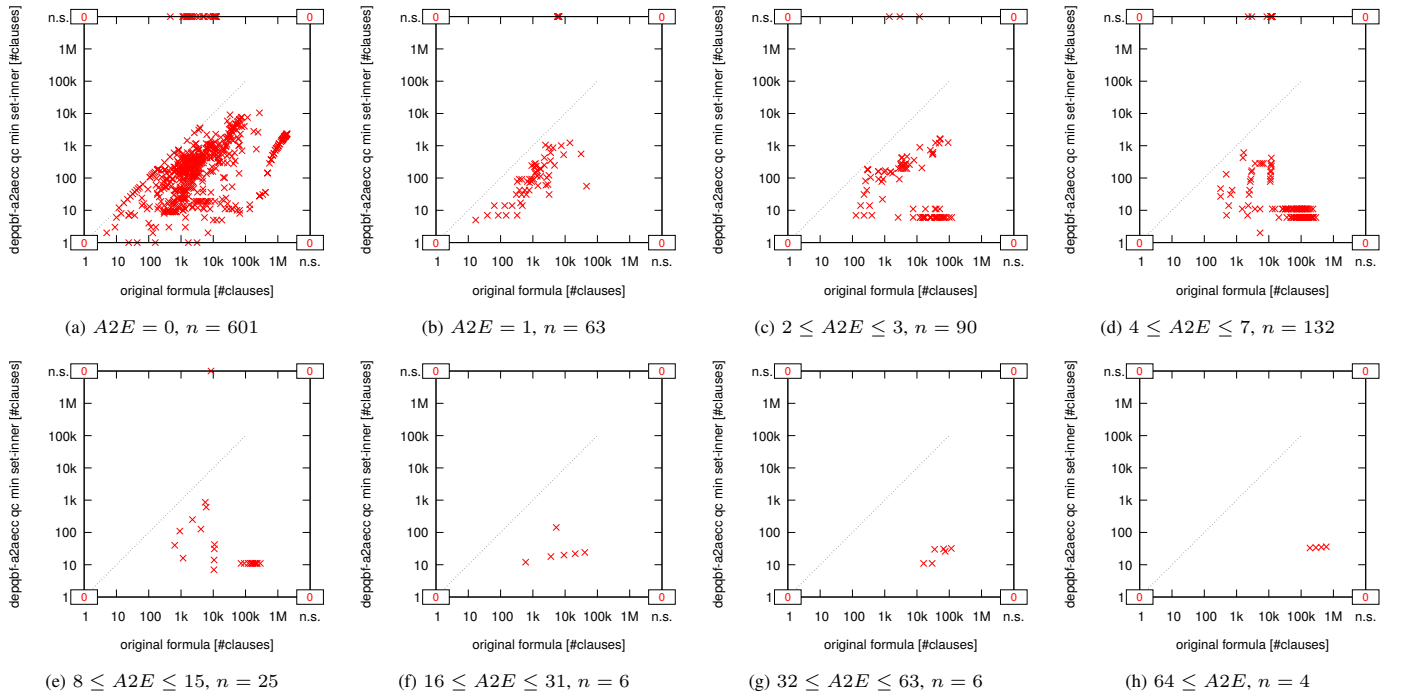


Fig. 495: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc min set-inner partitioned by number of weakened \forall in mode qc minsepcsr list (number of clauses).

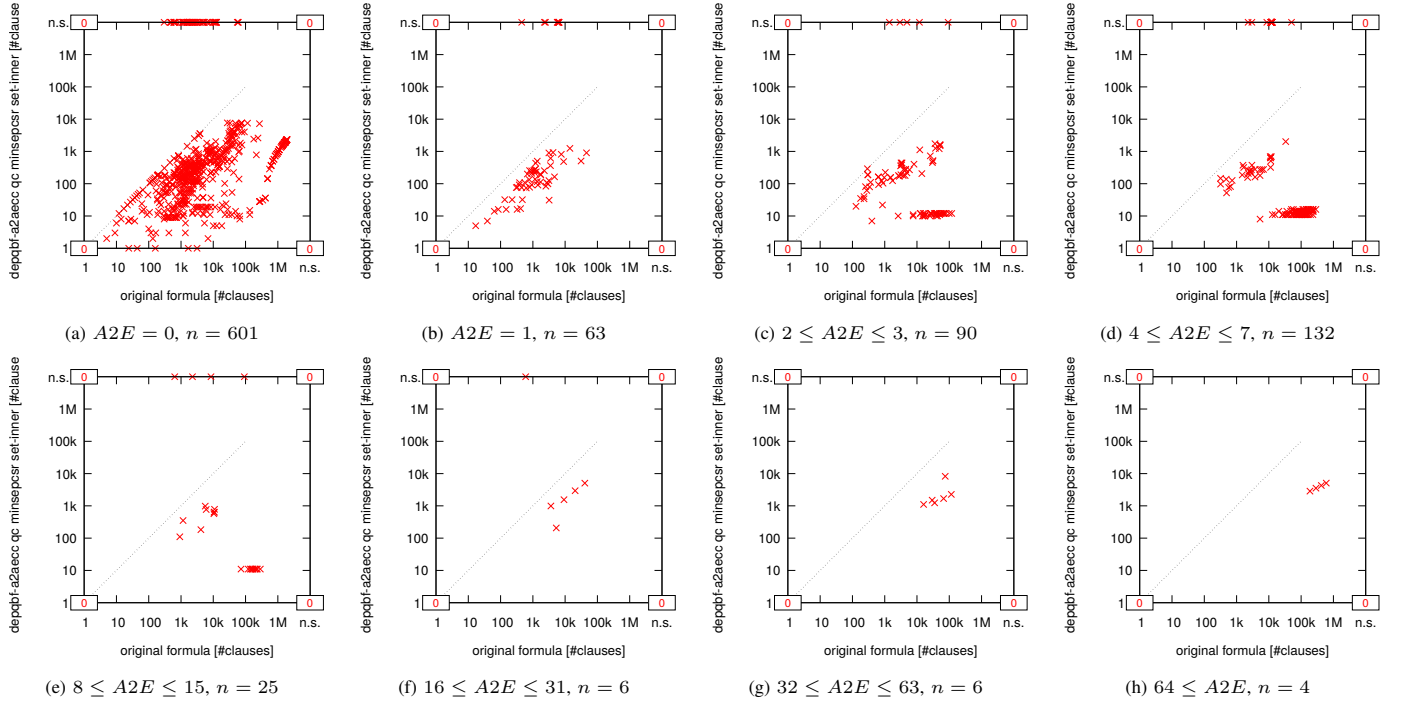


Fig. 496: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode original versus mode qc minsepcsr set-inner partitioned by number of weakened \forall in mode qc minsepcsr list (number of clauses).

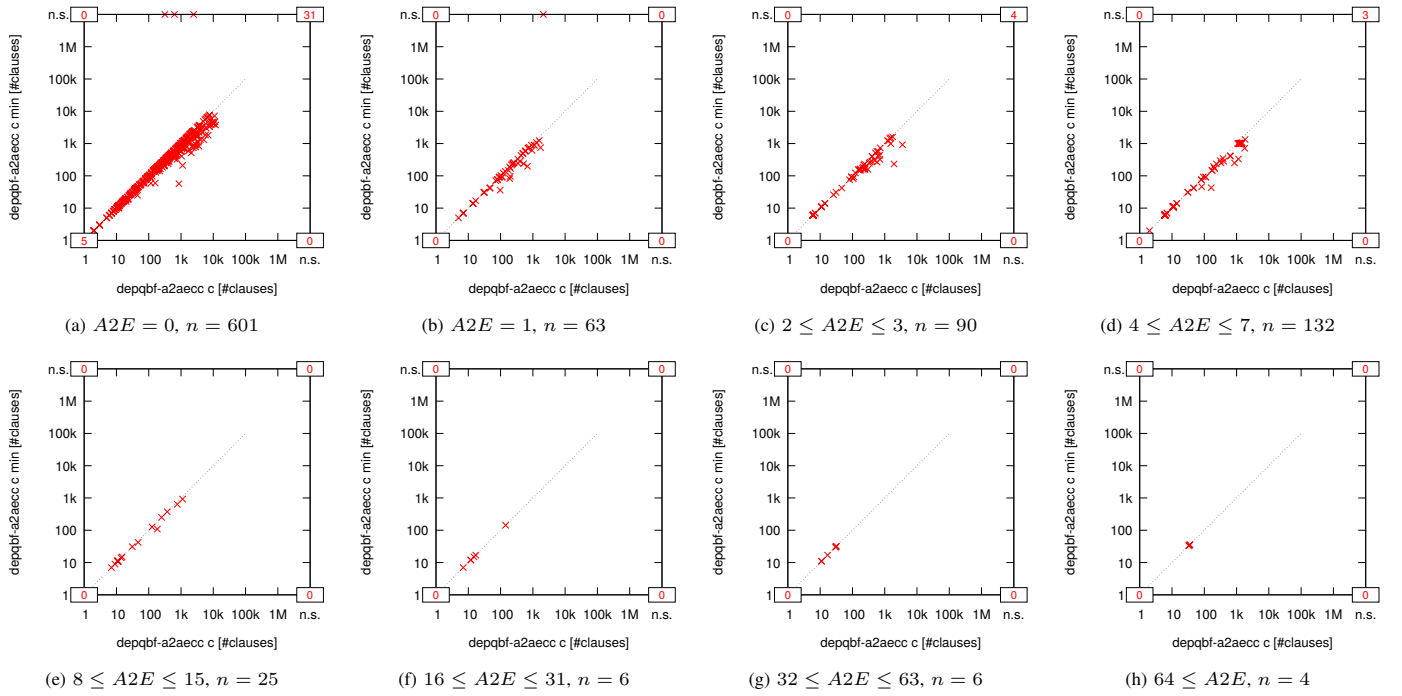


Fig. 497: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode c versus mode c min partitioned by number of weakened \forall in mode qc minsepcsr list (number of clauses).

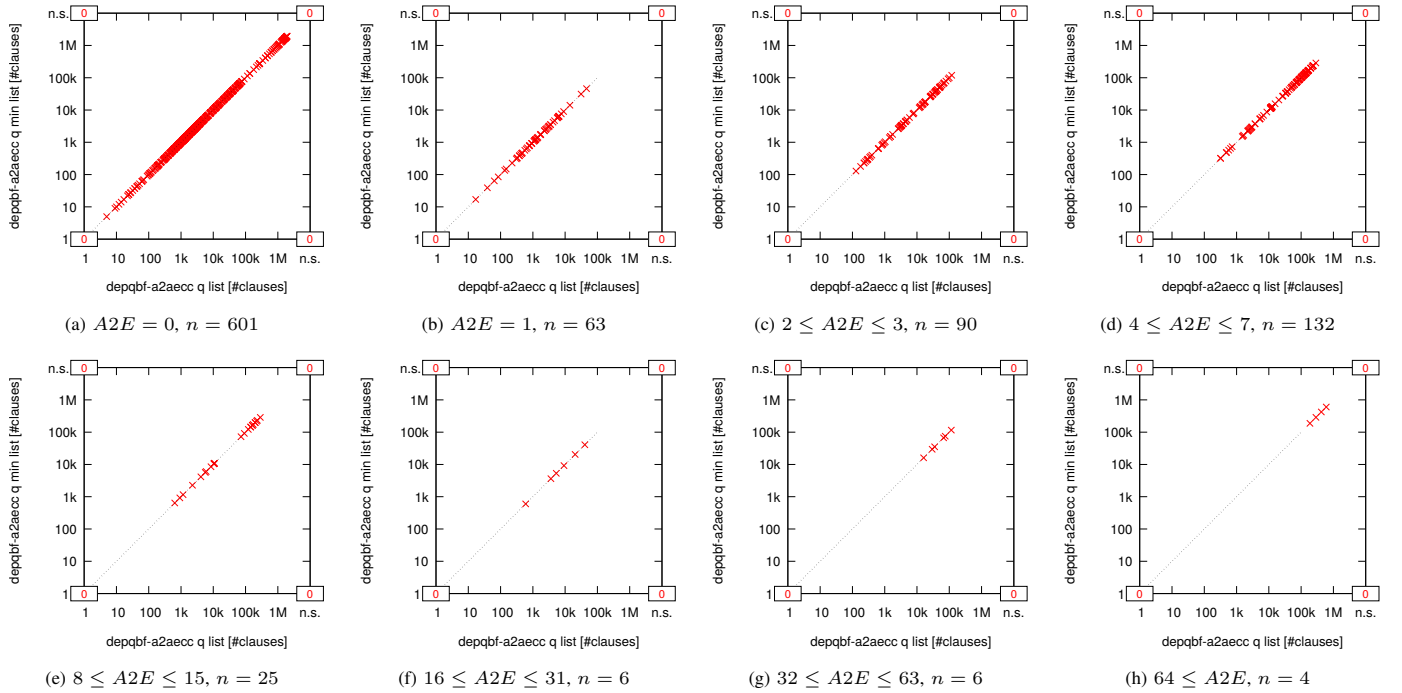


Fig. 498: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode q min list partitioned by number of weakened \forall in mode qc minsepcsr list (number of clauses).

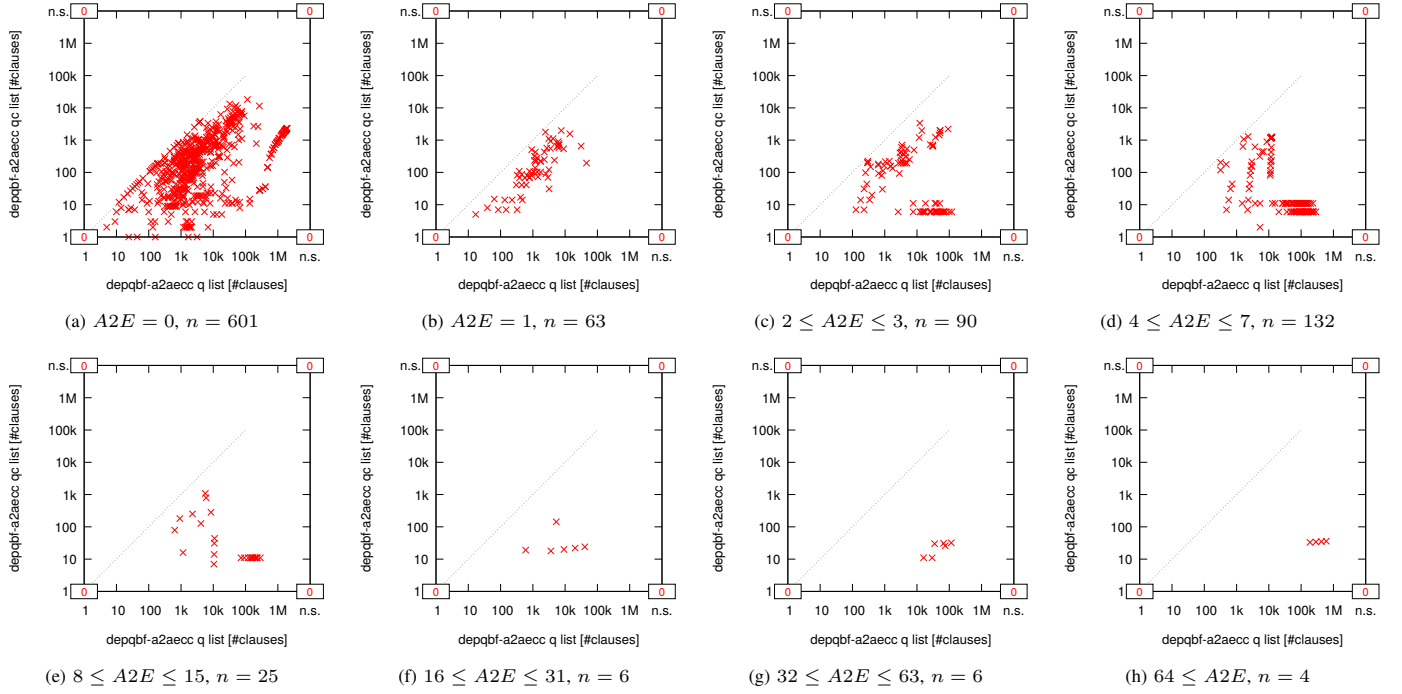


Fig. 499: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode qc list partitioned by number of weakened ∇ in mode qc minsepcsr list (number of clauses).

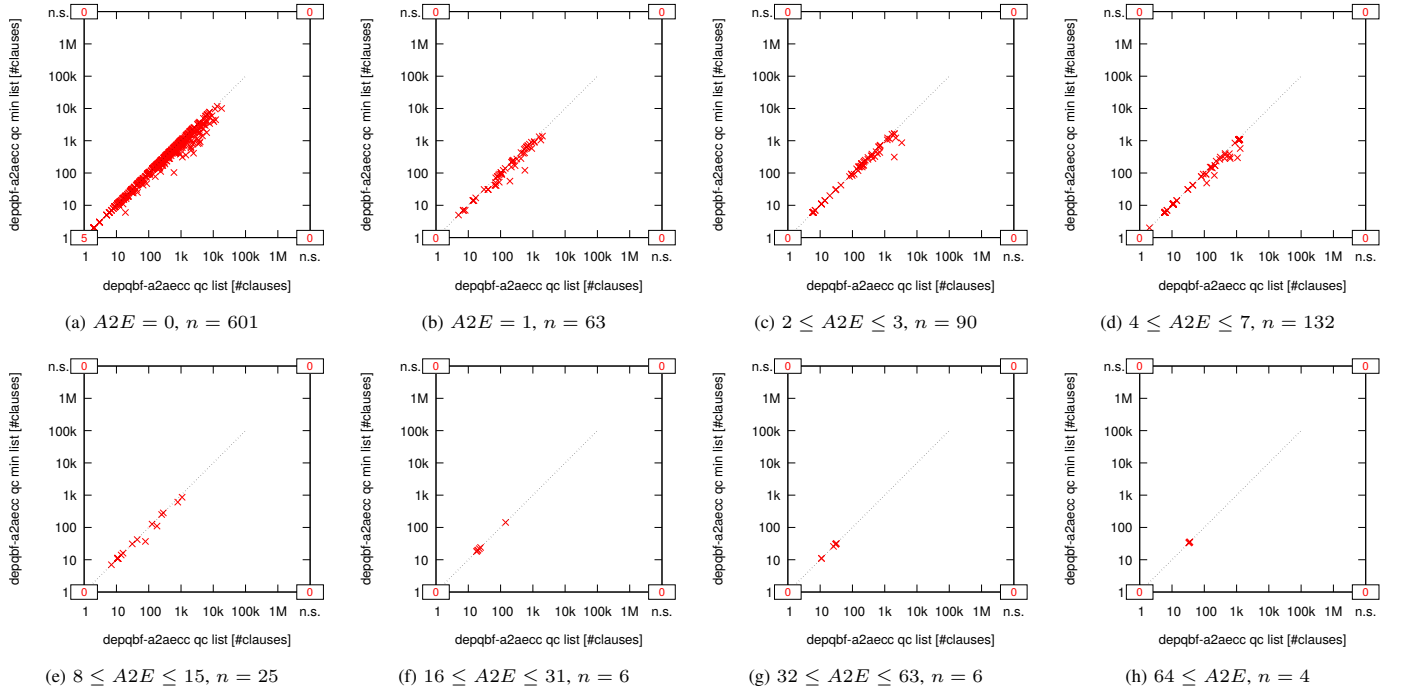


Fig. 500: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode qc min list partitioned by number of weakened ∇ in mode qc minsepcsr list (number of clauses).

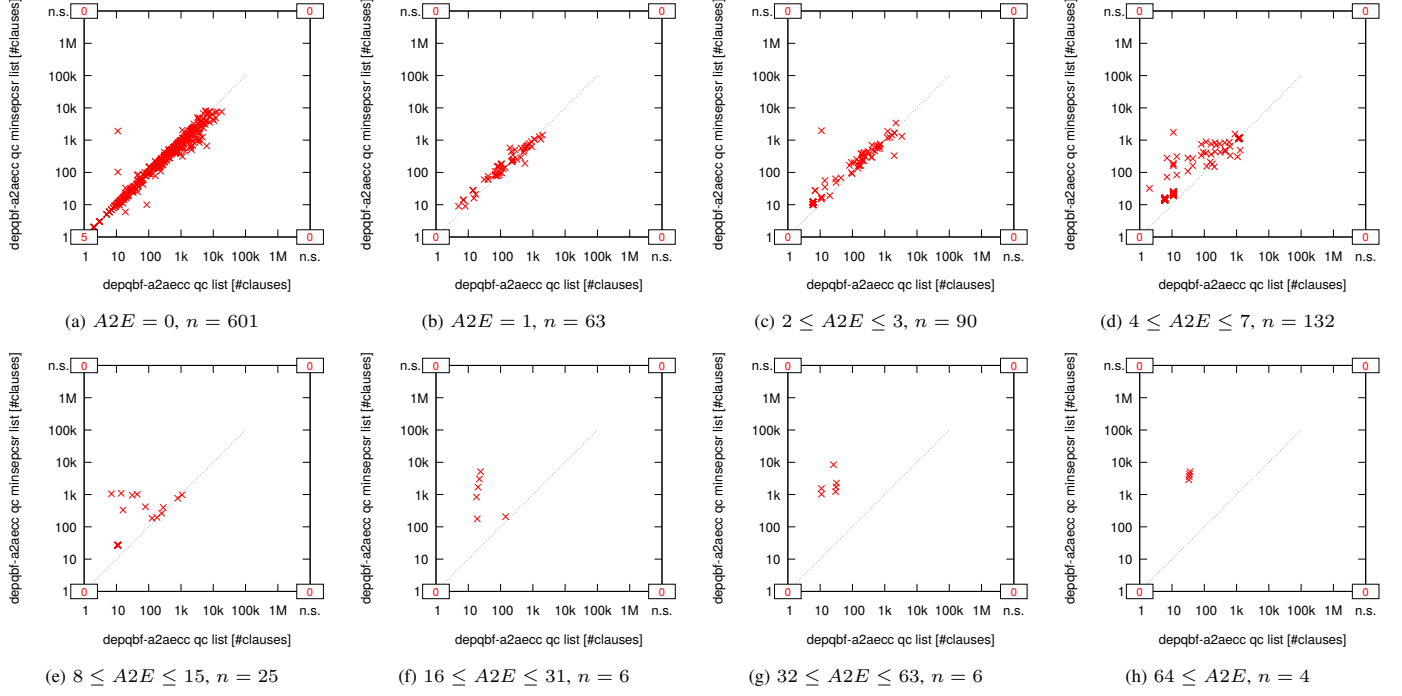


Fig. 501: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode qc minsepcsr list partitioned by number of weakened ∇ in mode qc minsepcsr list (number of clauses).

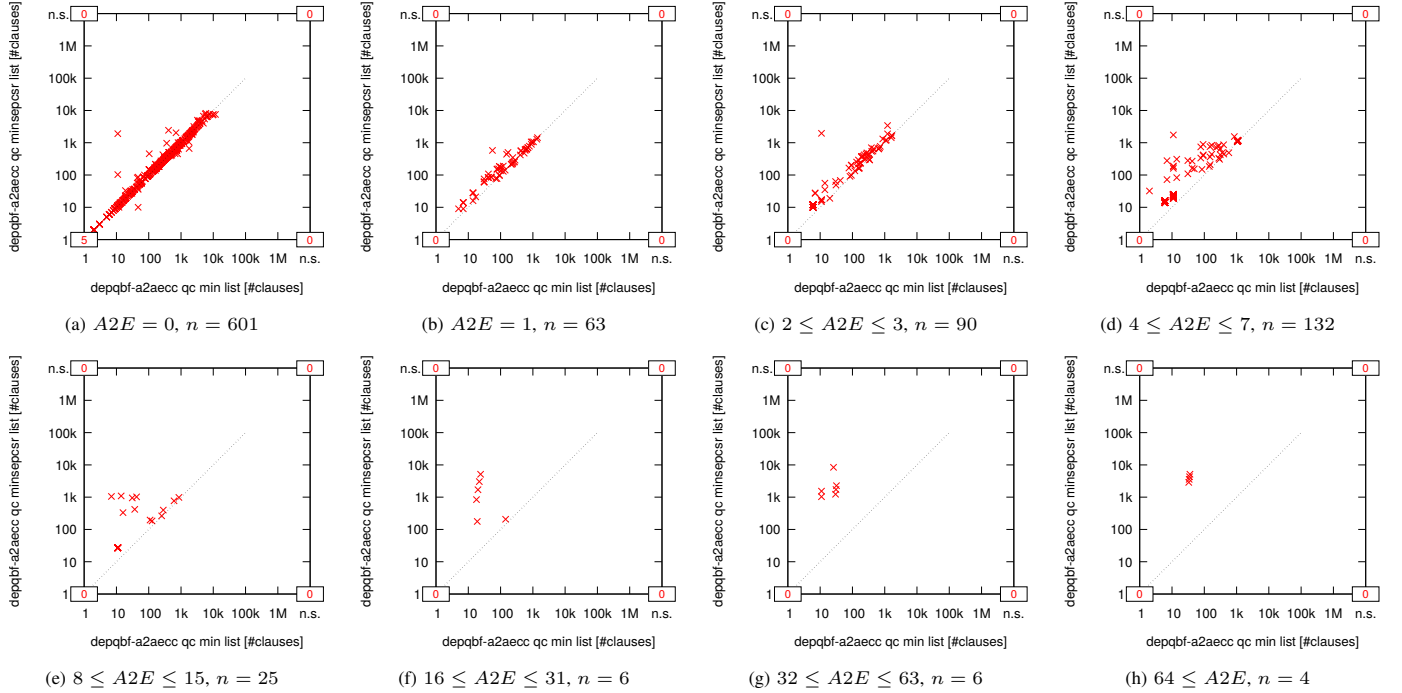


Fig. 502: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode qc minsepcsr list partitioned by number of weakened ∇ in mode qc minsepcsr list (number of clauses).

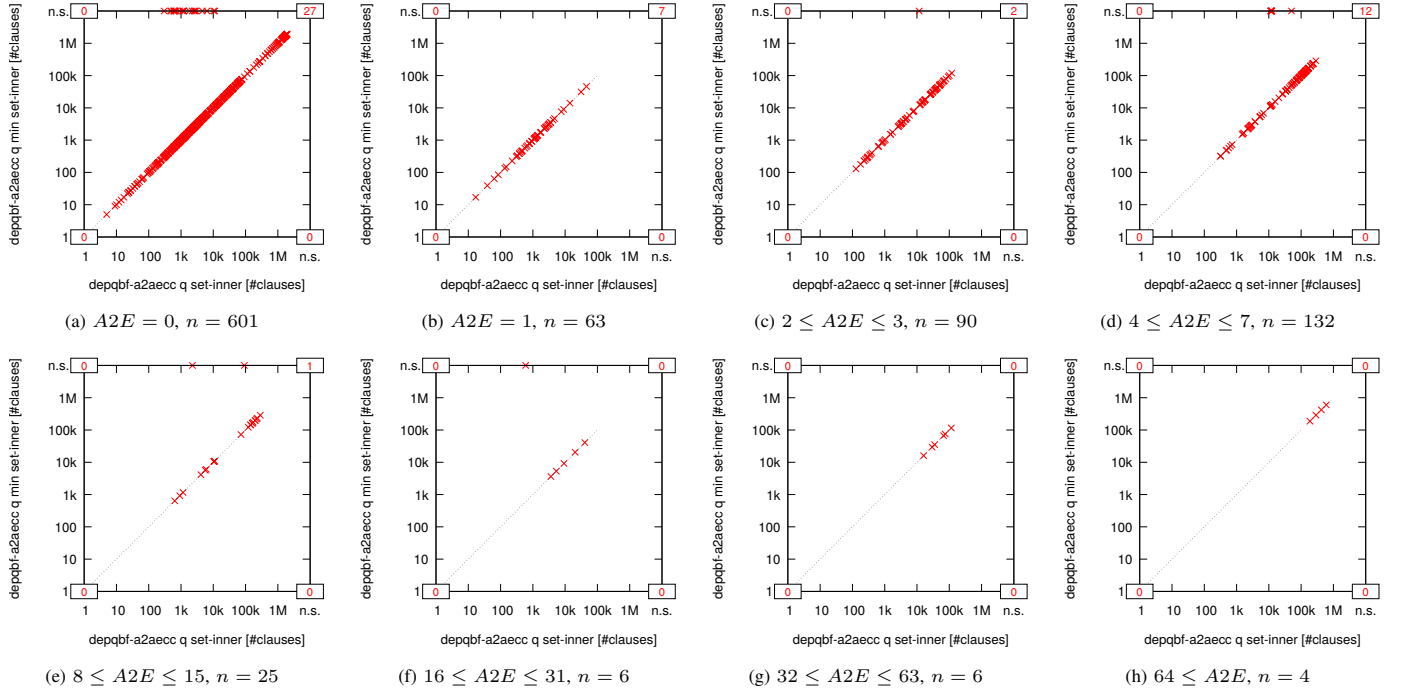


Fig. 503: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode q min set-inner partitioned by number of weakened ∇ in mode qc minsepcsr list (number of clauses).

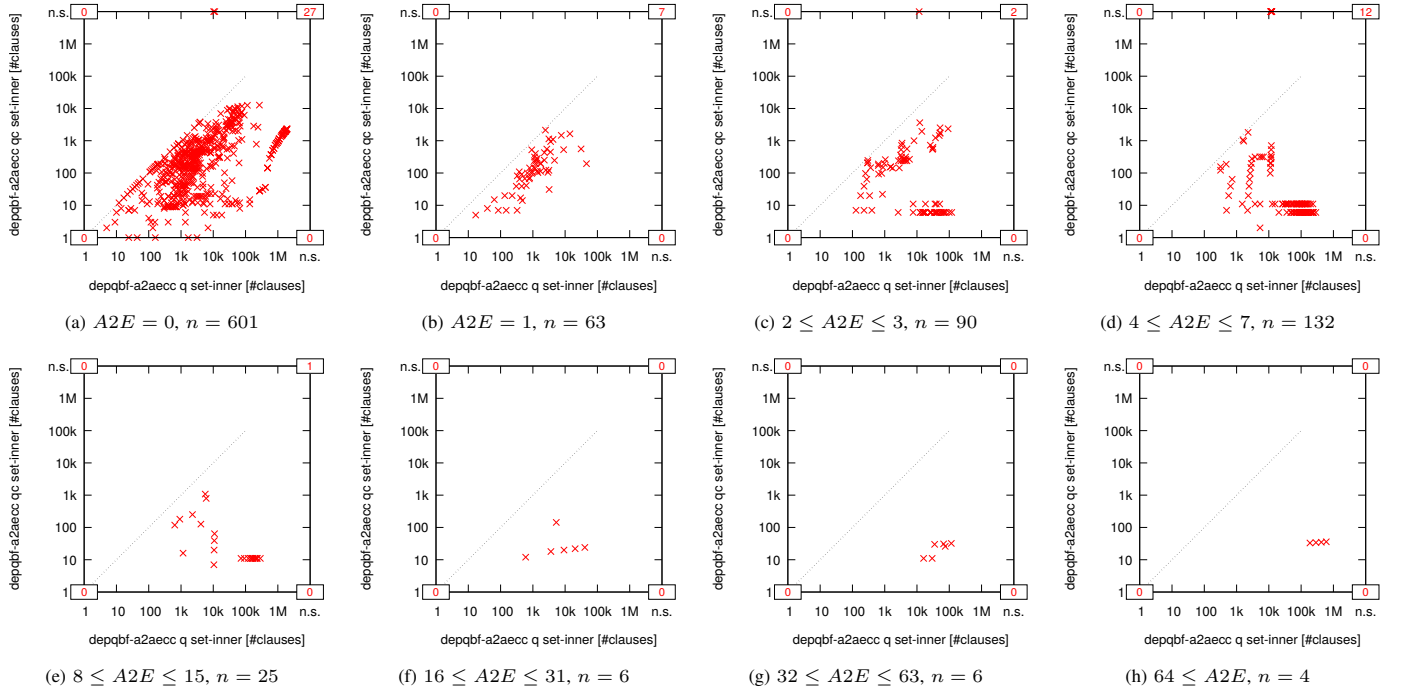


Fig. 504: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode qc set-inner partitioned by number of weakened ∇ in mode qc minsepcsr list (number of clauses).

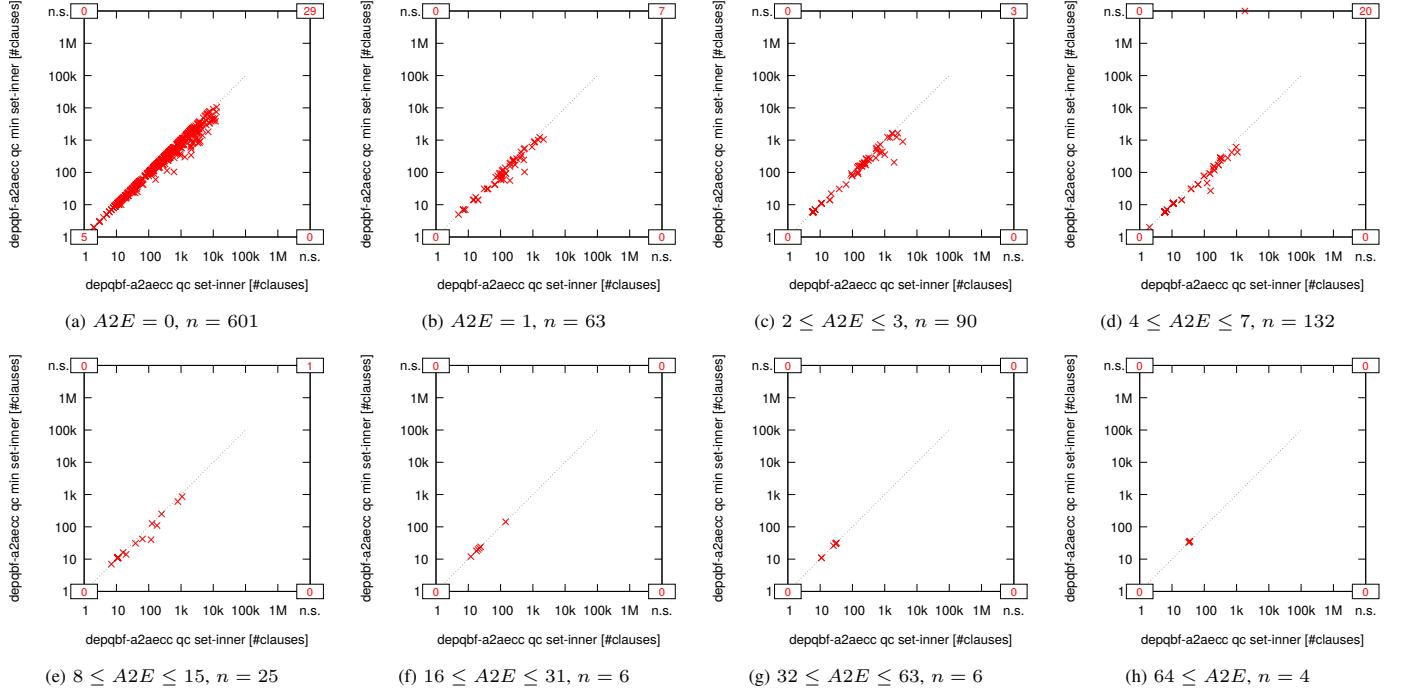


Fig. 505: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode qc min set-inner partitioned by number of weakened ∇ in mode qc minsepcsr list (number of clauses).

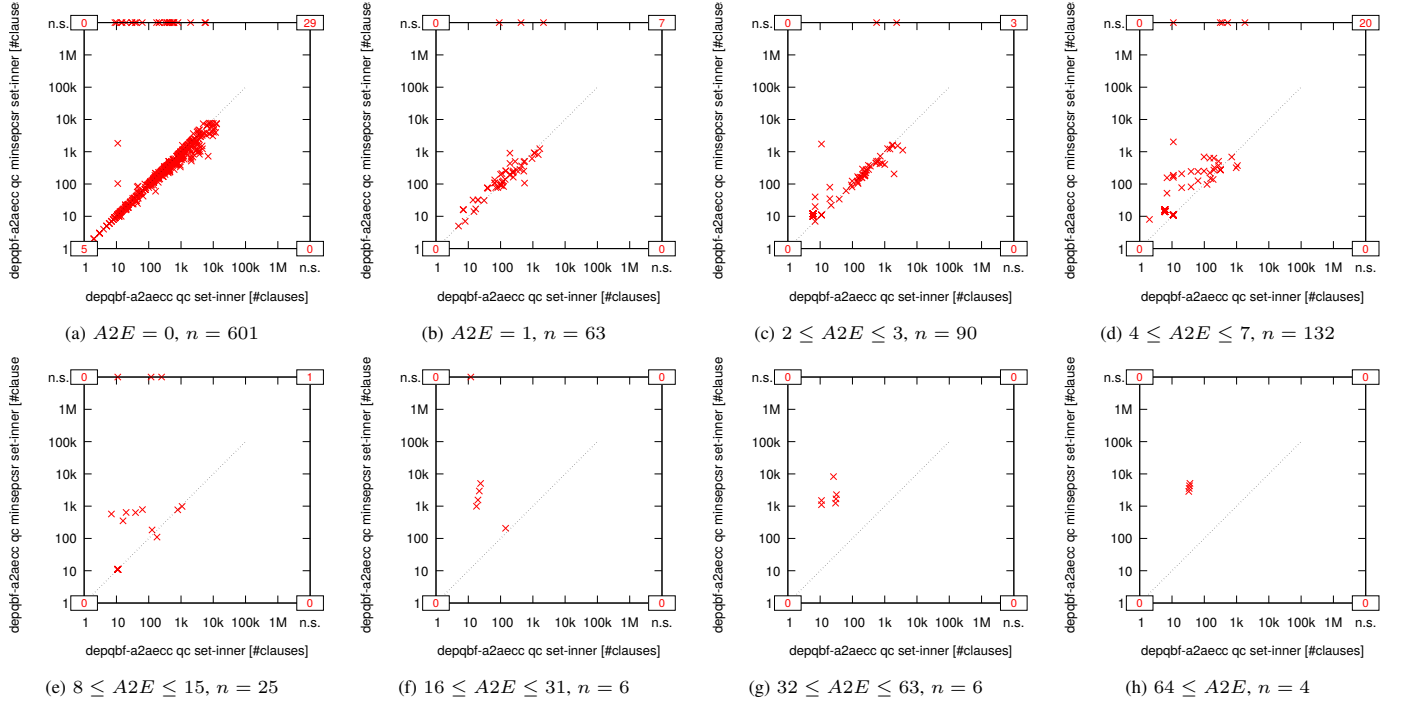


Fig. 506: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode qc minsepcsr set-inner partitioned by number of weakened ∇ in mode qc minsepcsr list (number of clauses).

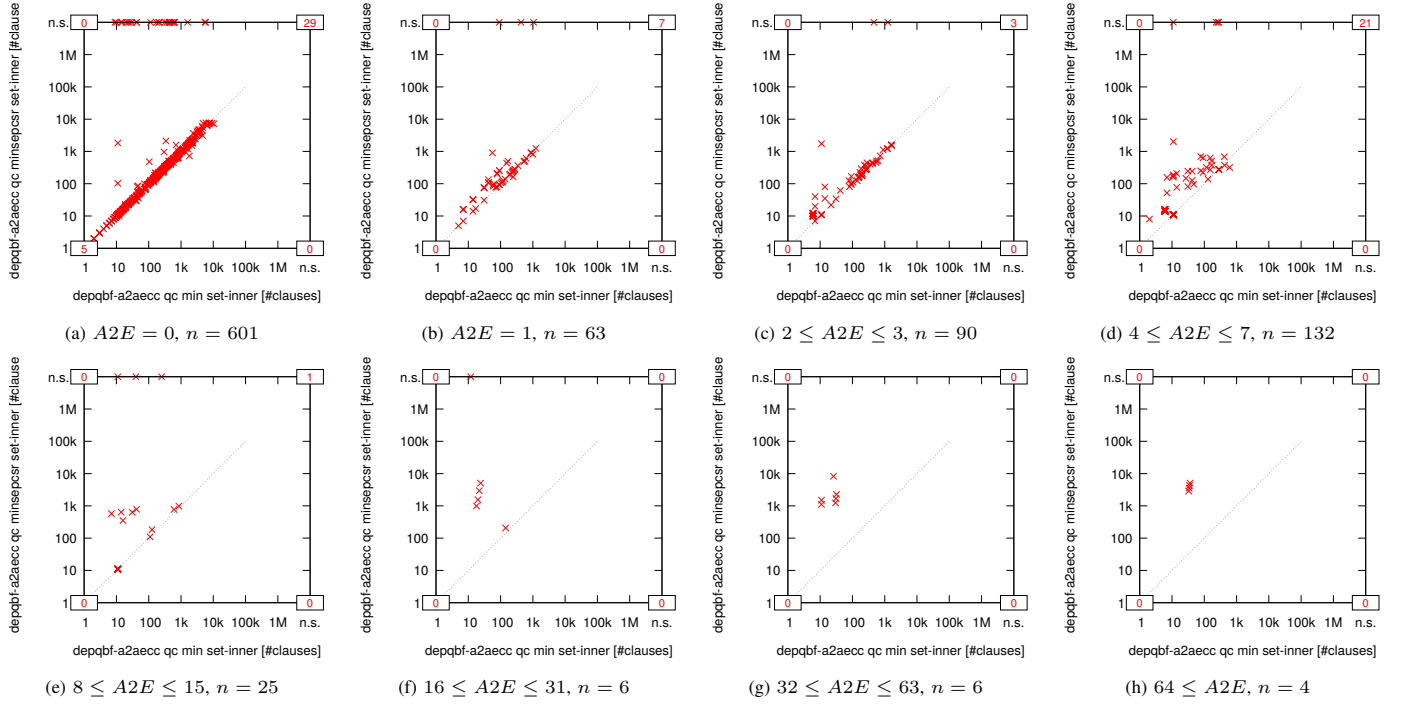


Fig. 507: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode qc minsepcsr set-inner partitioned by number of weakened \forall in mode qc minsepcsr list (number of clauses).

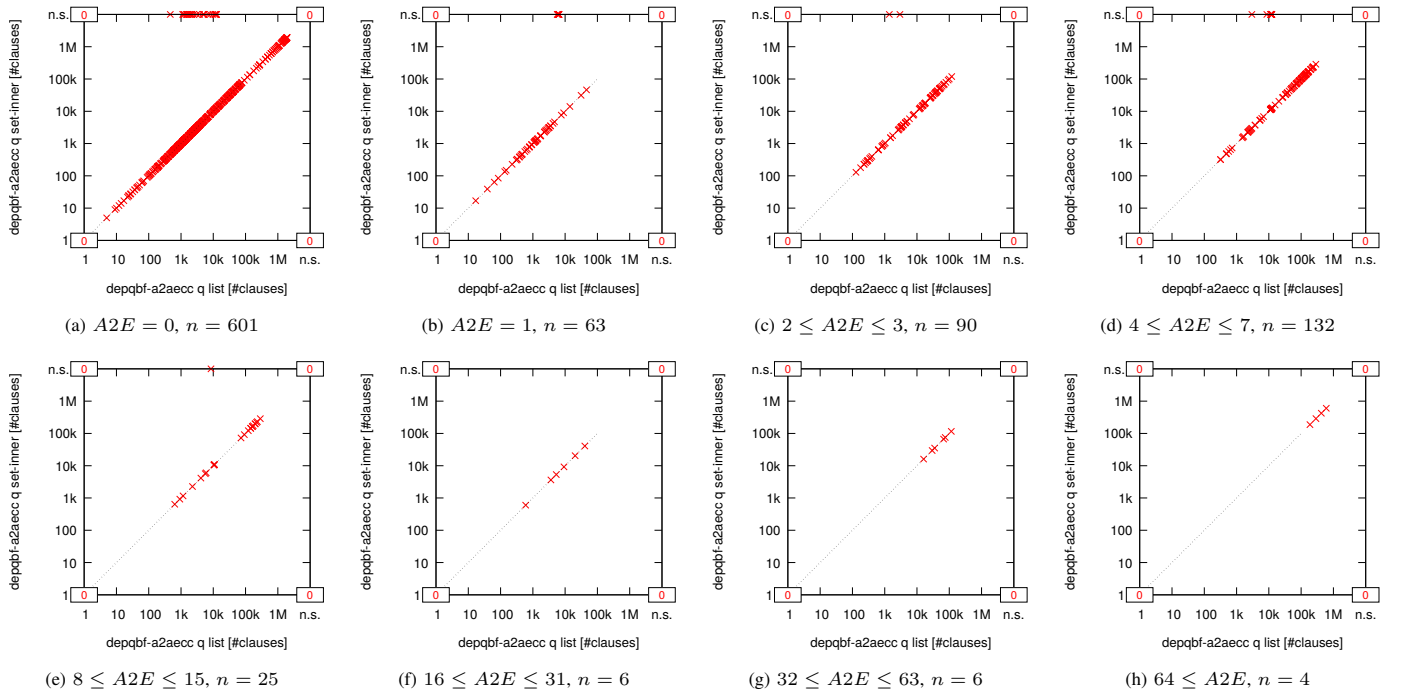


Fig. 508: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode q set-inner partitioned by number of weakened ∇ in mode qc minsepcsr list (number of clauses).

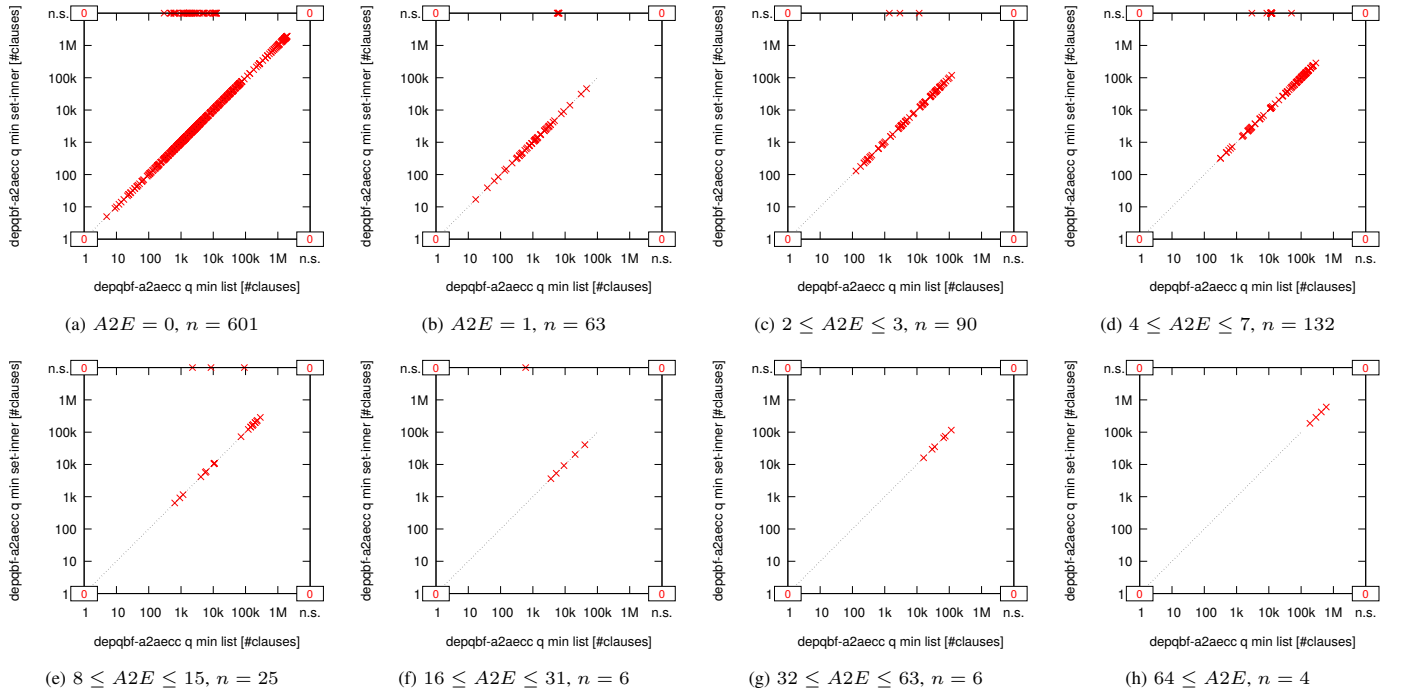


Fig. 509: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode q min list versus mode q min set-inner partitioned by number of weakened ∇ in mode qc minsepcsr list (number of clauses).

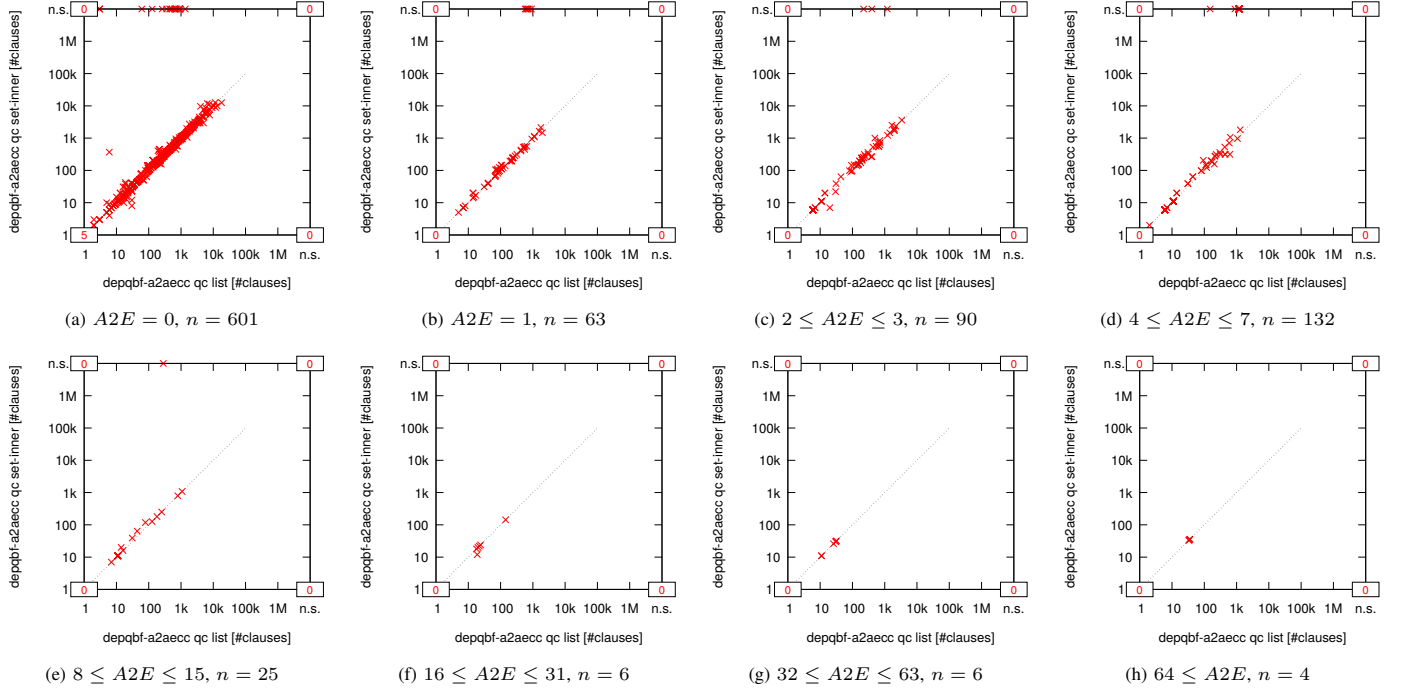


Fig. 510: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode qc set-inner partitioned by number of weakened ∇ in mode qc minsepcsr list (number of clauses).

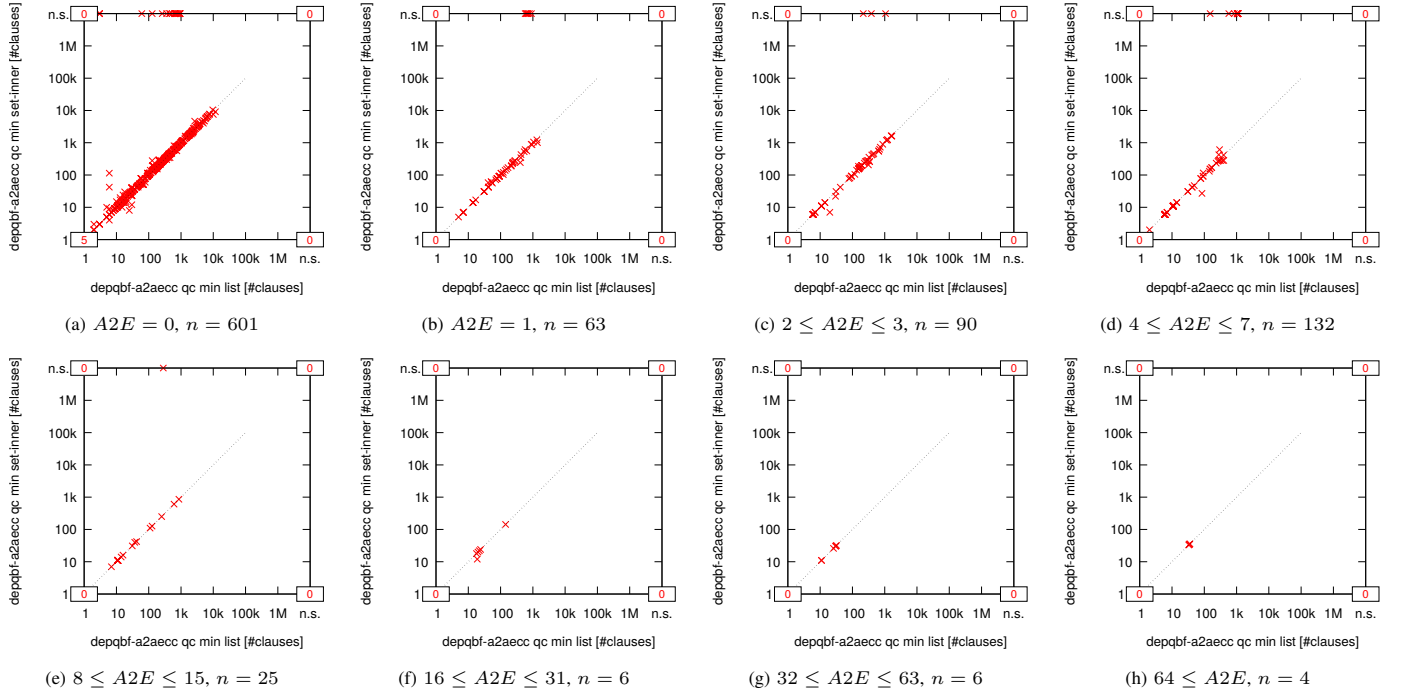


Fig. 511: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode qc min set-inner partitioned by number of weakened ∇ in mode qc minsepcsr list (number of clauses).

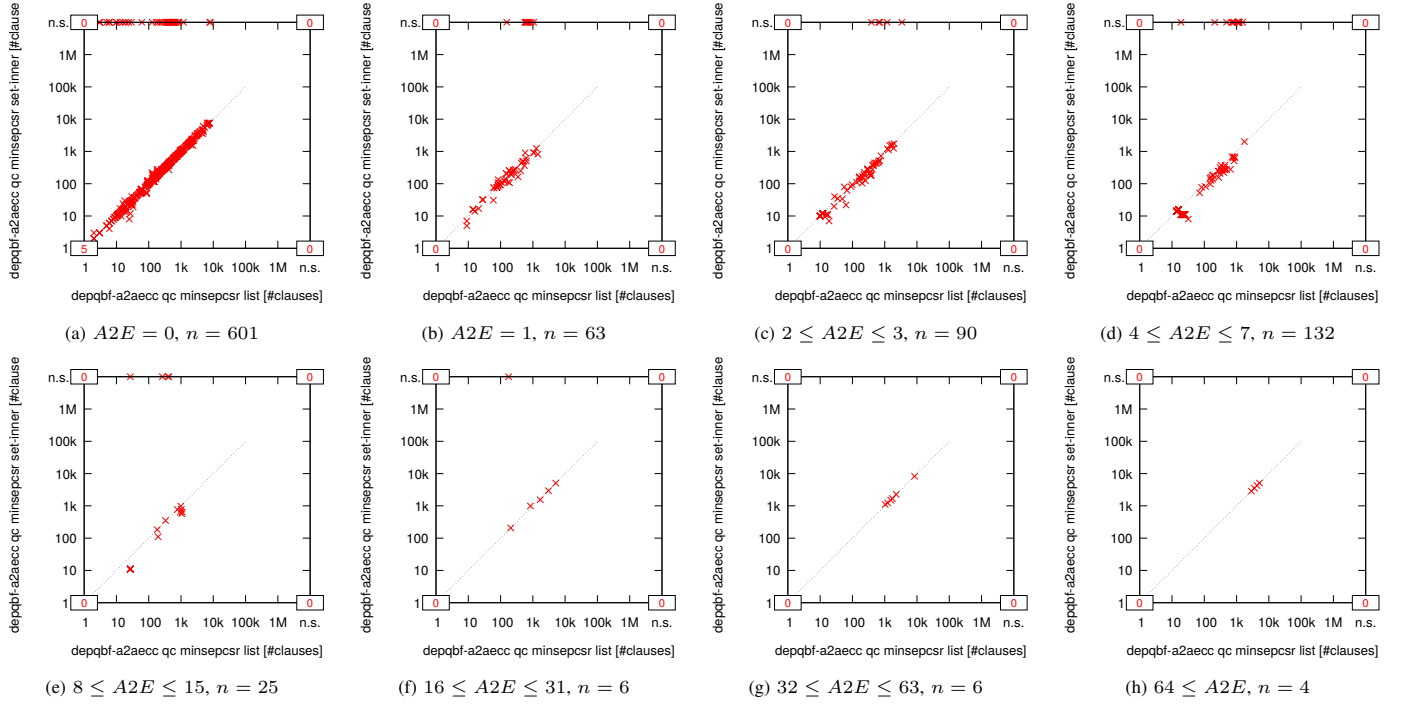


Fig. 512: Comparing sizes of unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode qc minsepcsr set-inner partitioned by number of weakened \forall in mode qc minsepcsr list (number of clauses).

APPENDIX E
OVERHEAD OF EXTRACTING UNSATISFIABLE CORES IN DepQBF-a2aecc

In this appendix we show additional plots along the lines of Fig. 3 that partition the set of benchmark instances by

- benchmark suite,
- number of \forall quantified variables,
- alternation depth,
- number of clauses, and
- maximum variable index.

When we provide a number of benchmarks for a plot or a number of plots ($n = \dots$), then in this appendix this is the number of all benchmarks relevant to the plot that have been found to be unsatisfiable by any solver in our experiments.

A. Partitioned by Benchmark Suite

In this subsection there are 14 figures for each benchmark suite with subfigures for different modes of unsatisfiable core extraction:

- 1) Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics.
- 2) Memory usage for 1.
- 3) As 1. but with set-inner semantics.
- 4) Memory usage for 3.
- 5) Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline.
- 6) Memory usage for 5.
- 7) As 5. but with set-inner semantics.
- 8) Memory usage for 7.
- 9) Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline.
- 10) Memory usage for 9.
- 11) As 9. but with set-inner semantics.
- 12) Memory usage for 11.
- 13) Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics.
- 14) Memory usage for 13.

1) All ($n = 2528$):

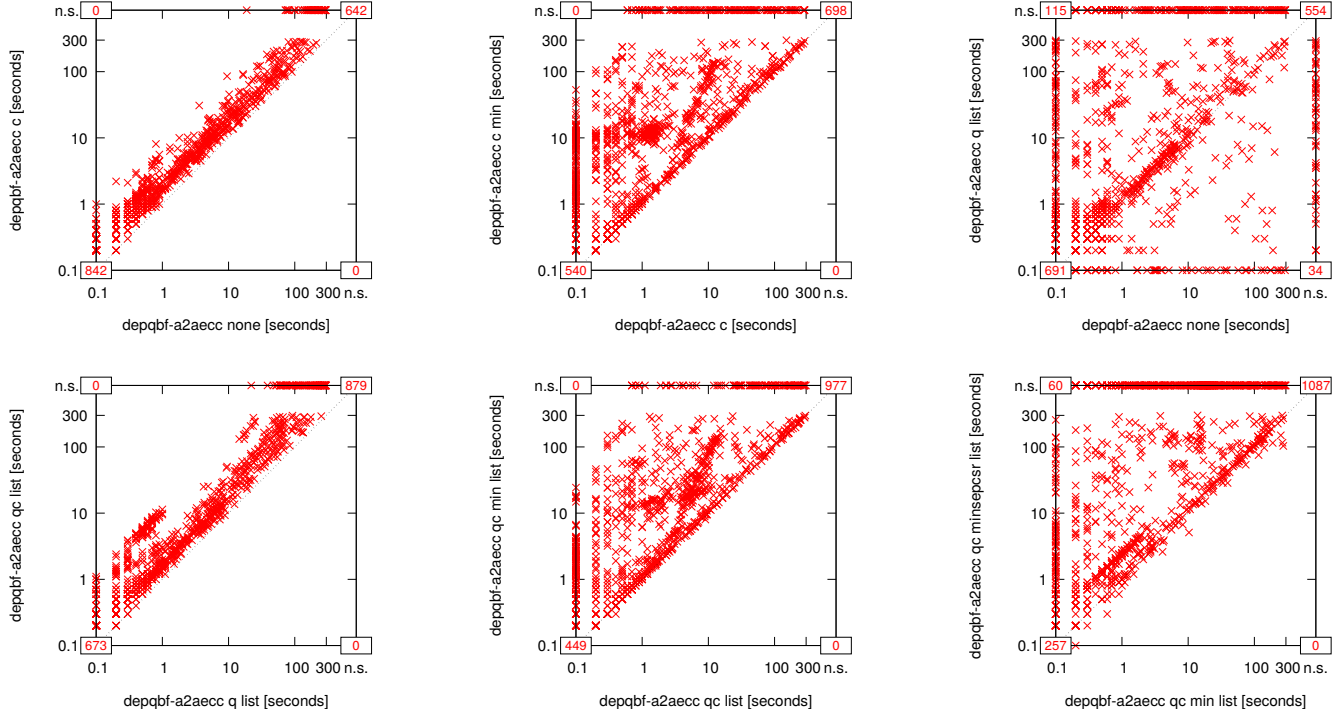


Fig. 513: Suite All ($n = 2528$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

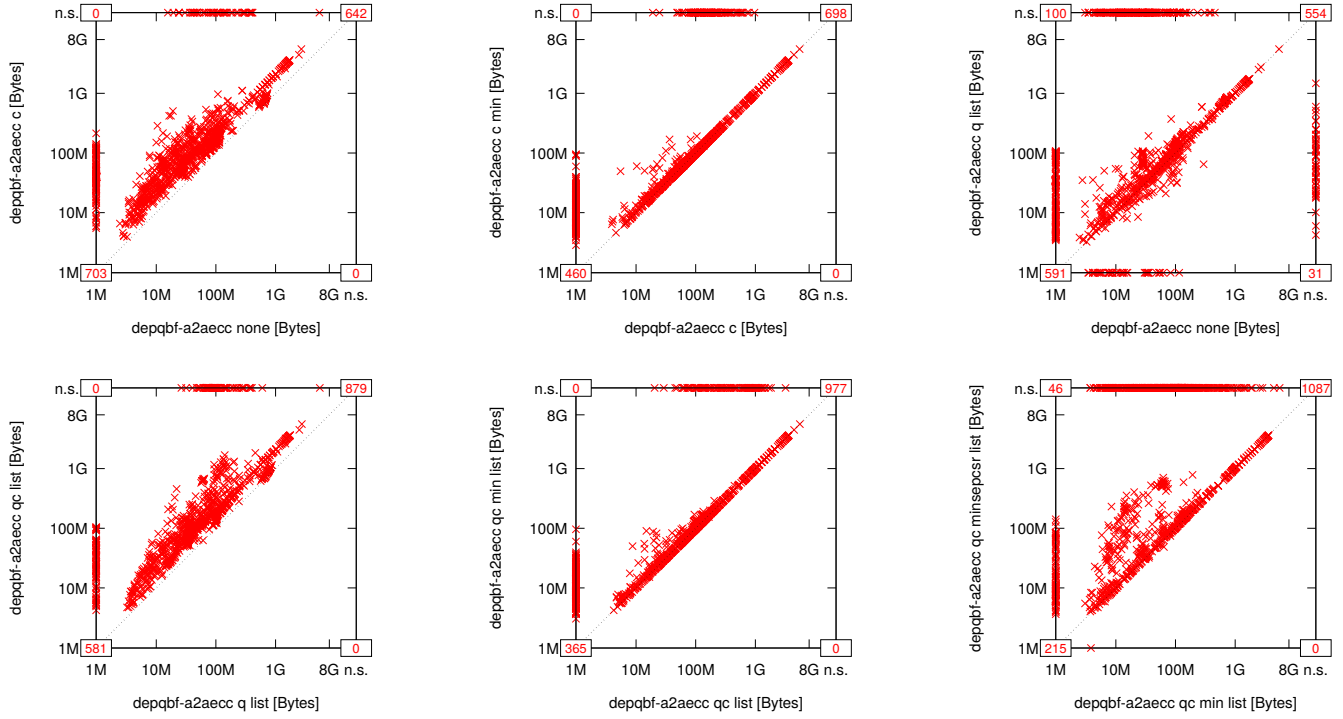


Fig. 514: Suite All ($n = 2528$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

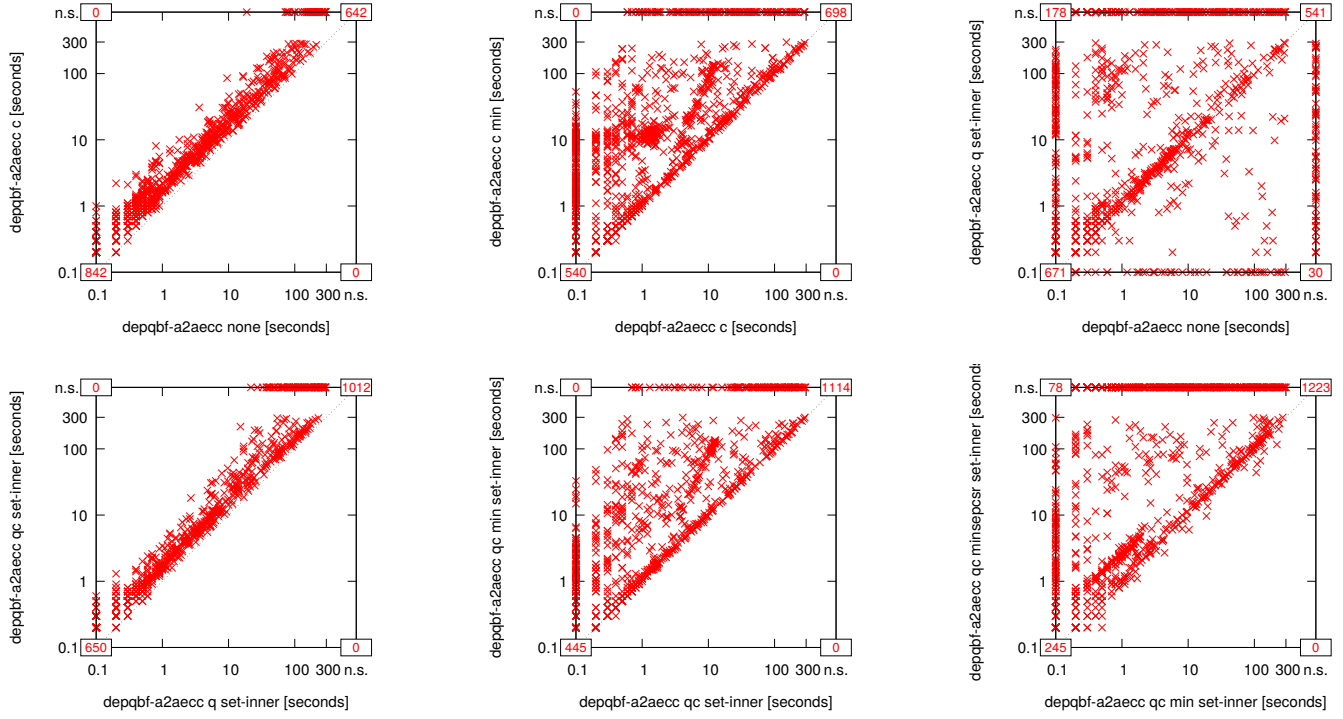


Fig. 515: Suite All ($n = 2528$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

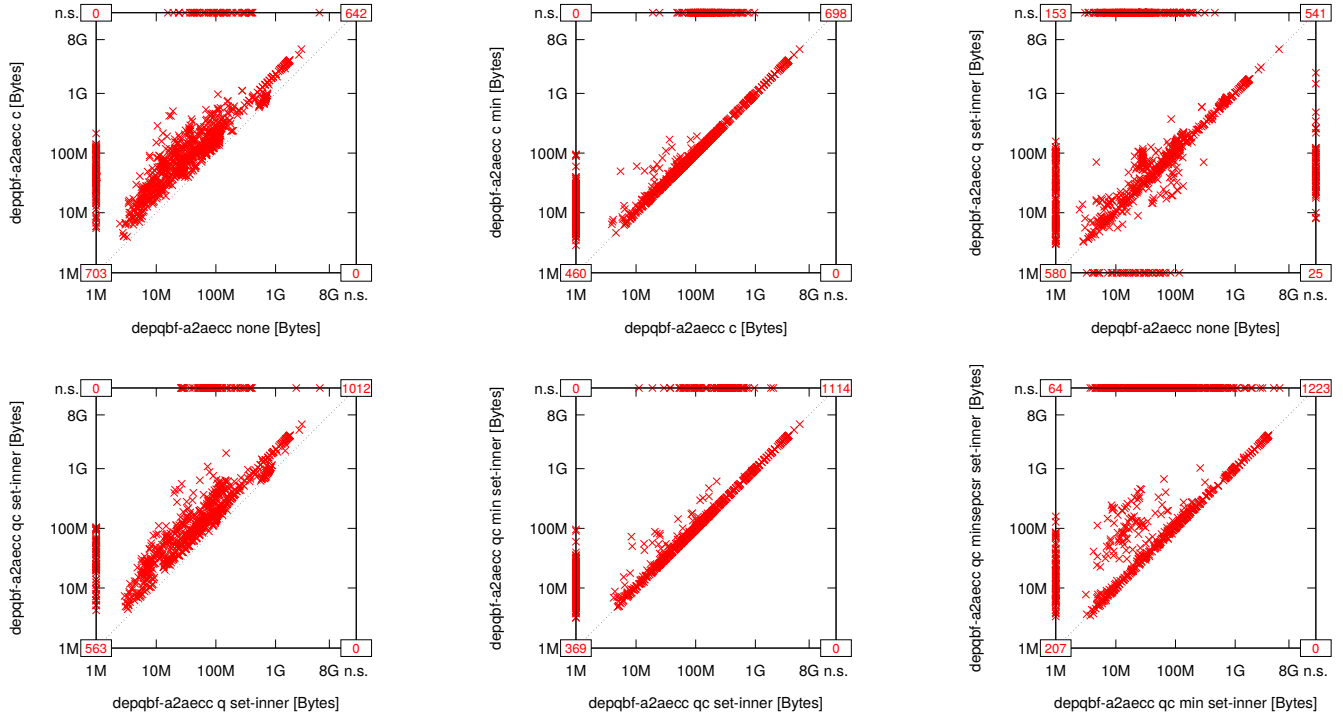


Fig. 516: Suite All ($n = 2528$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

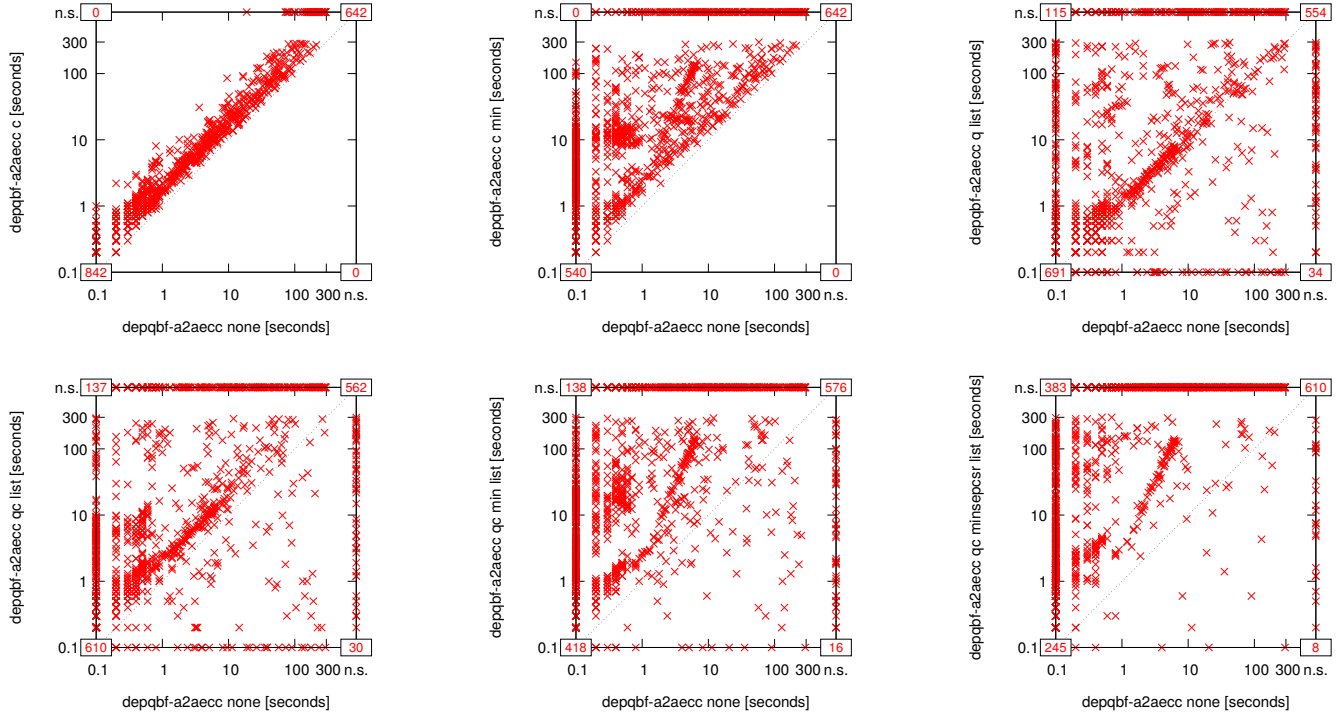


Fig. 517: Suite All ($n = 2528$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

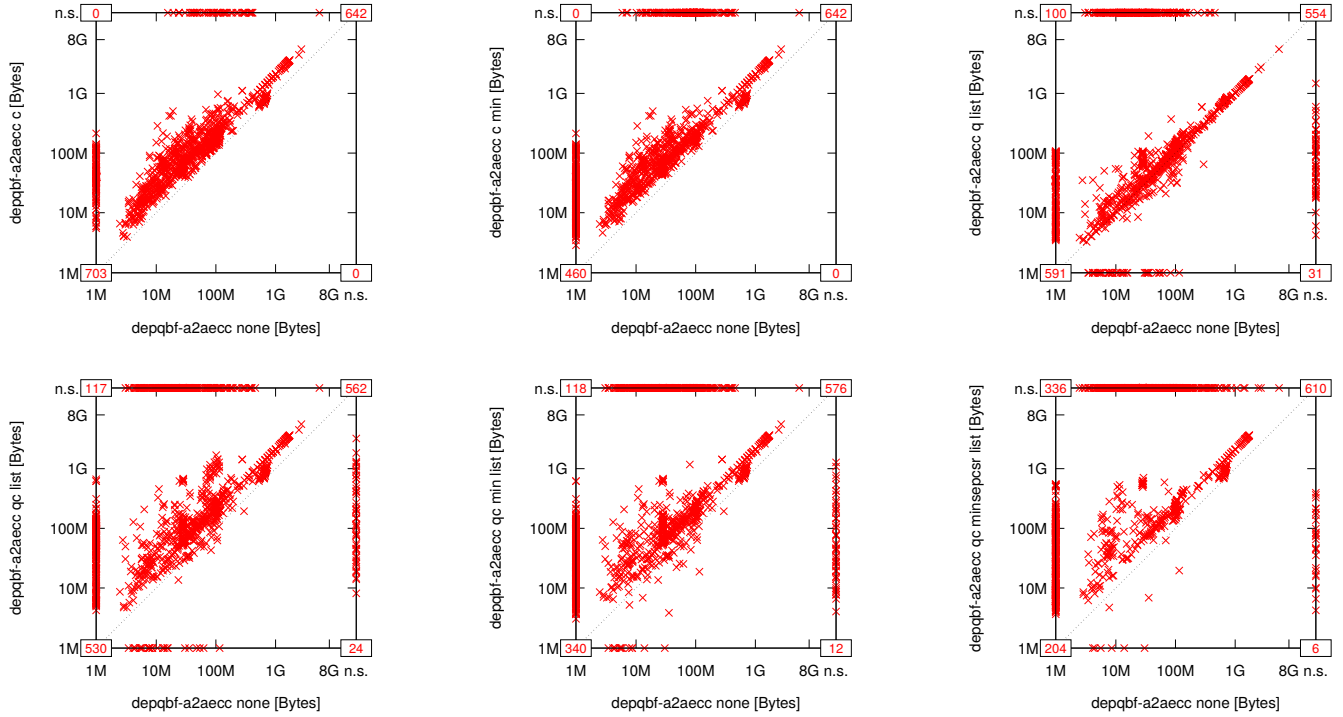


Fig. 518: Suite All ($n = 2528$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

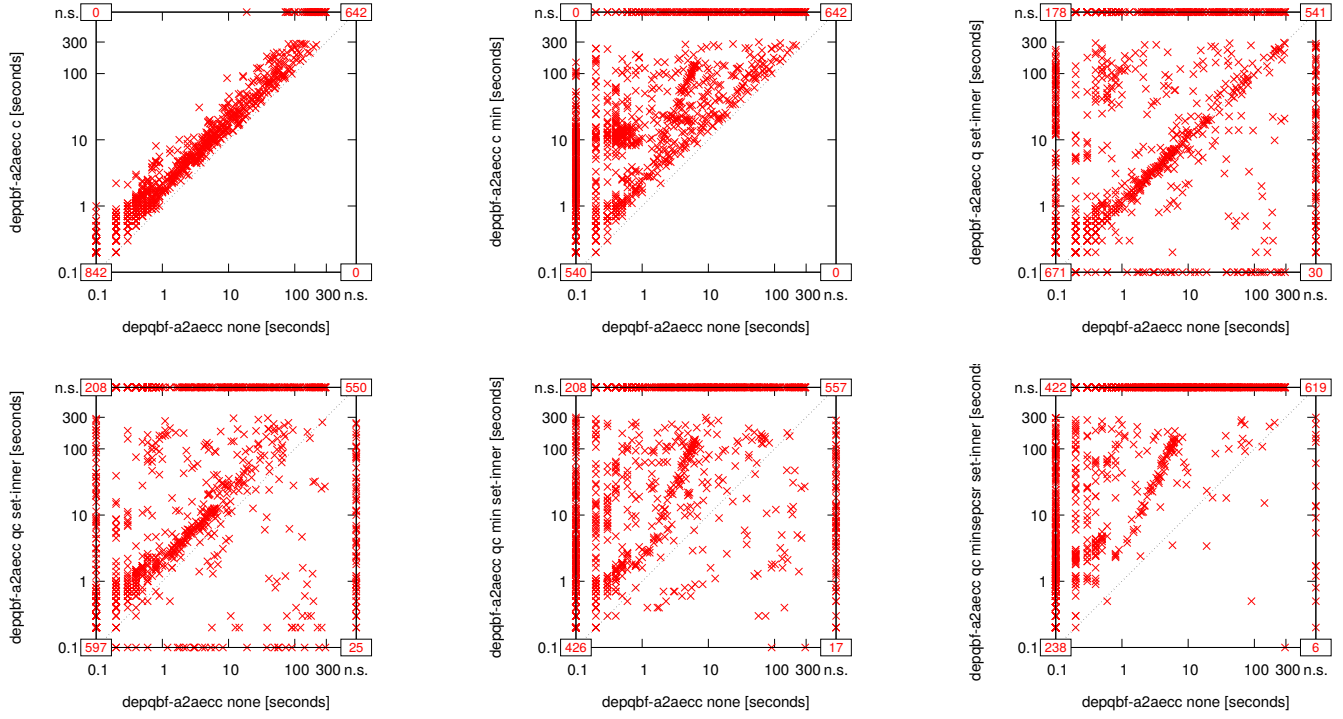


Fig. 519: Suite All ($n = 2528$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

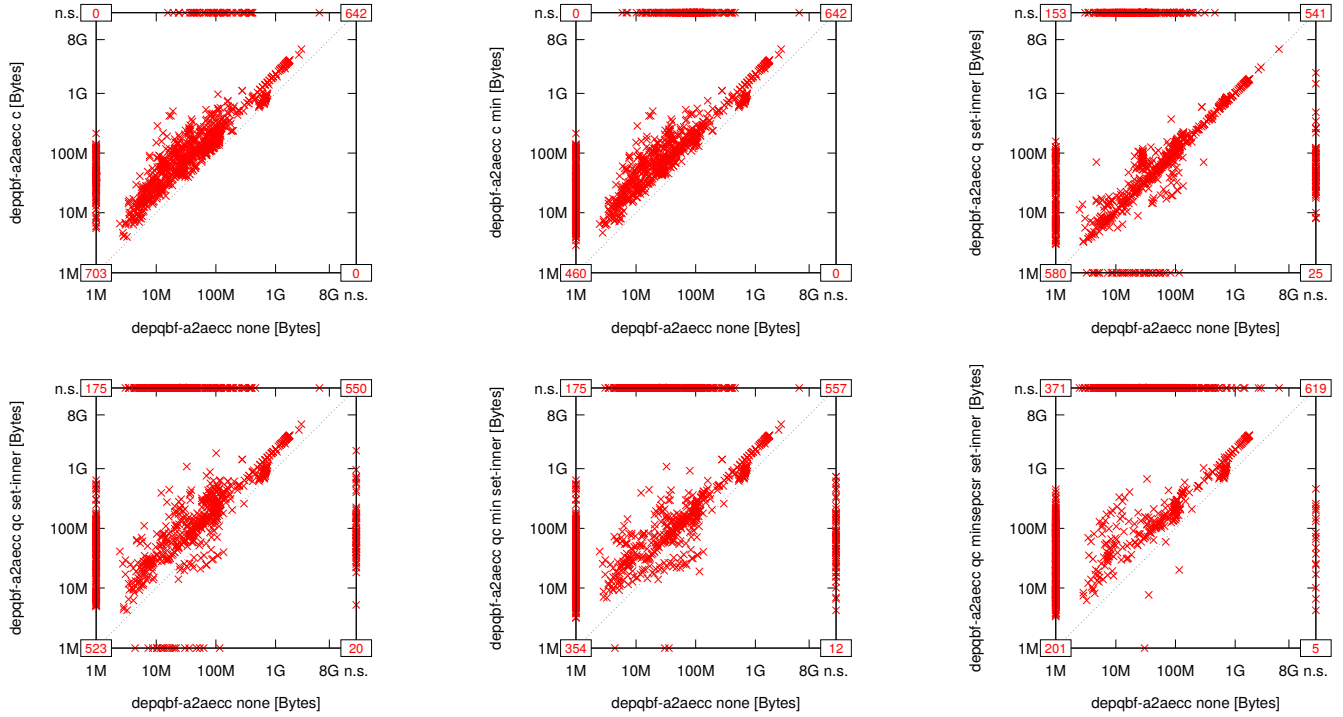


Fig. 520: Suite All ($n = 2528$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

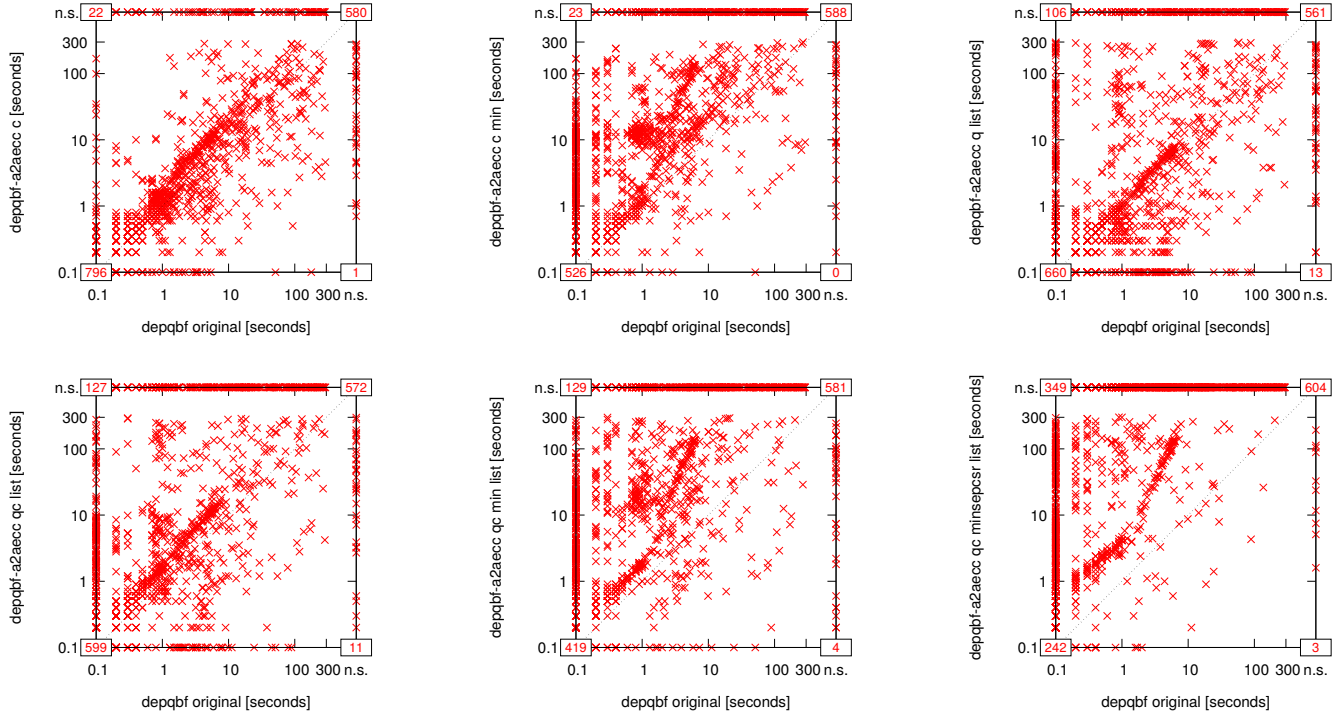


Fig. 521: Suite All ($n = 2528$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

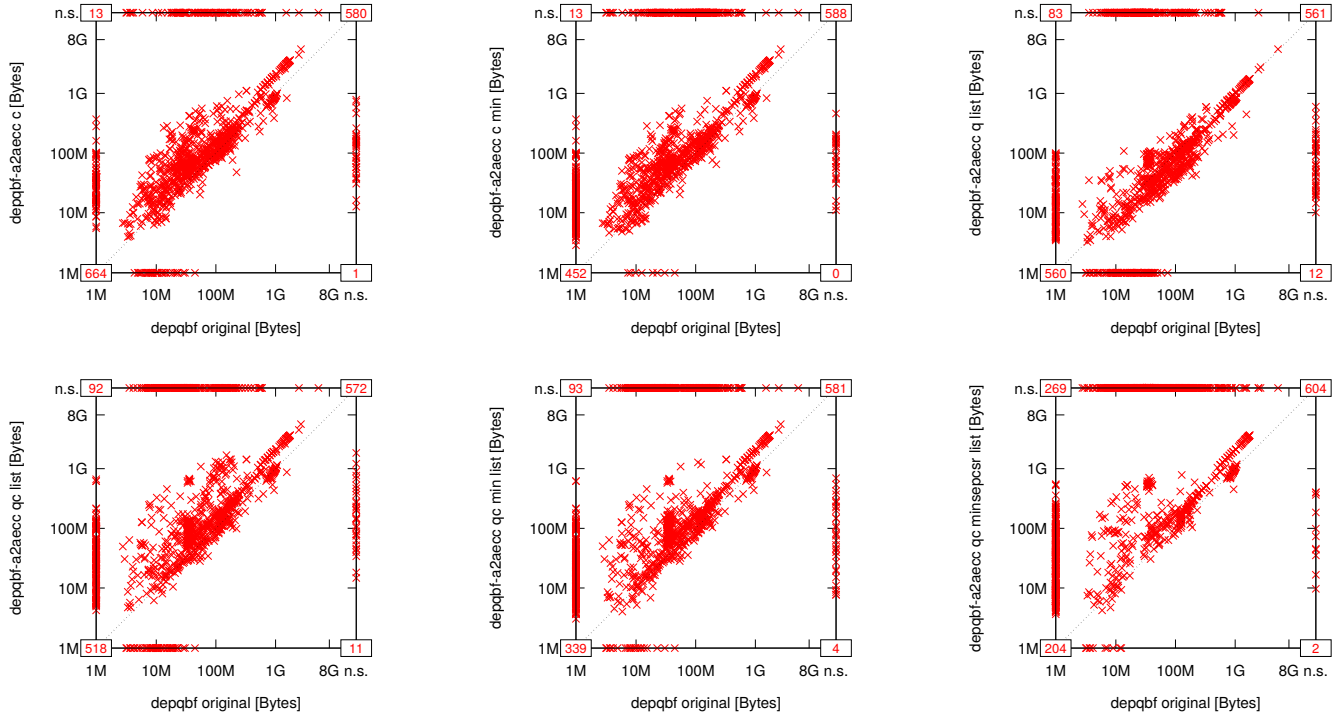


Fig. 522: Suite All ($n = 2528$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

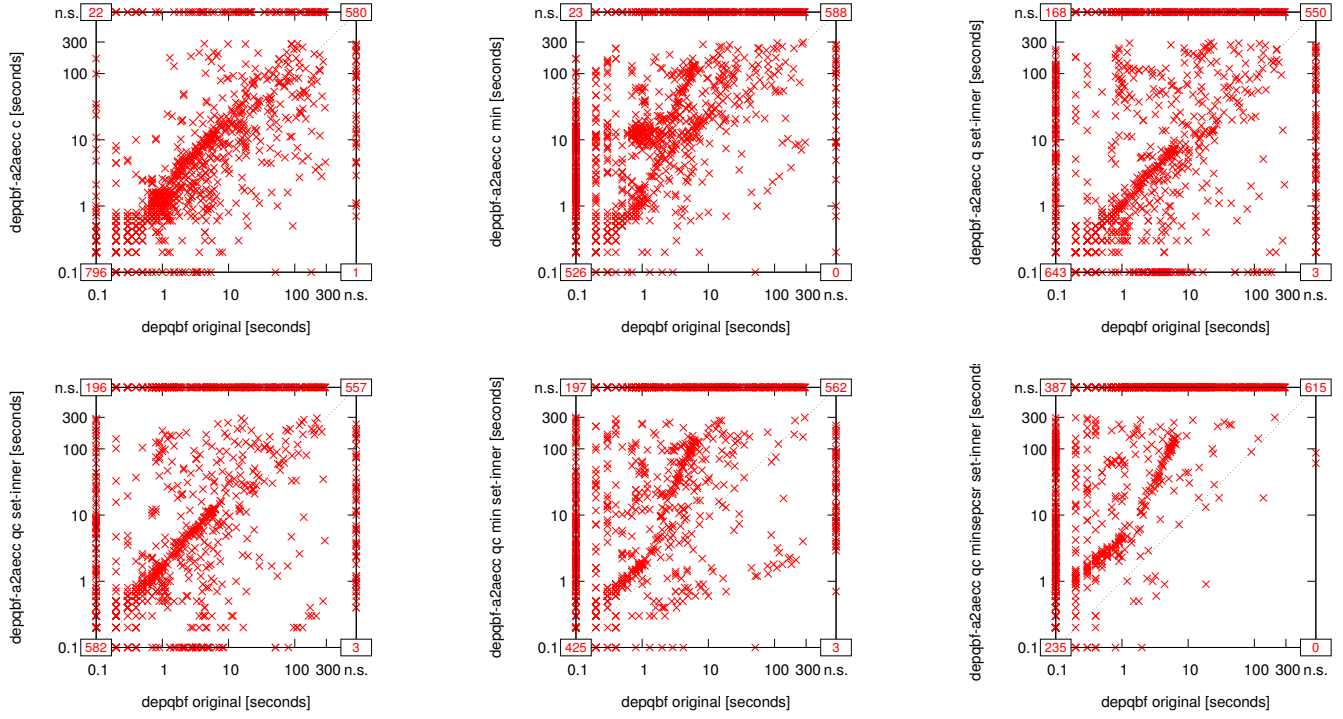


Fig. 523: Suite All ($n = 2528$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

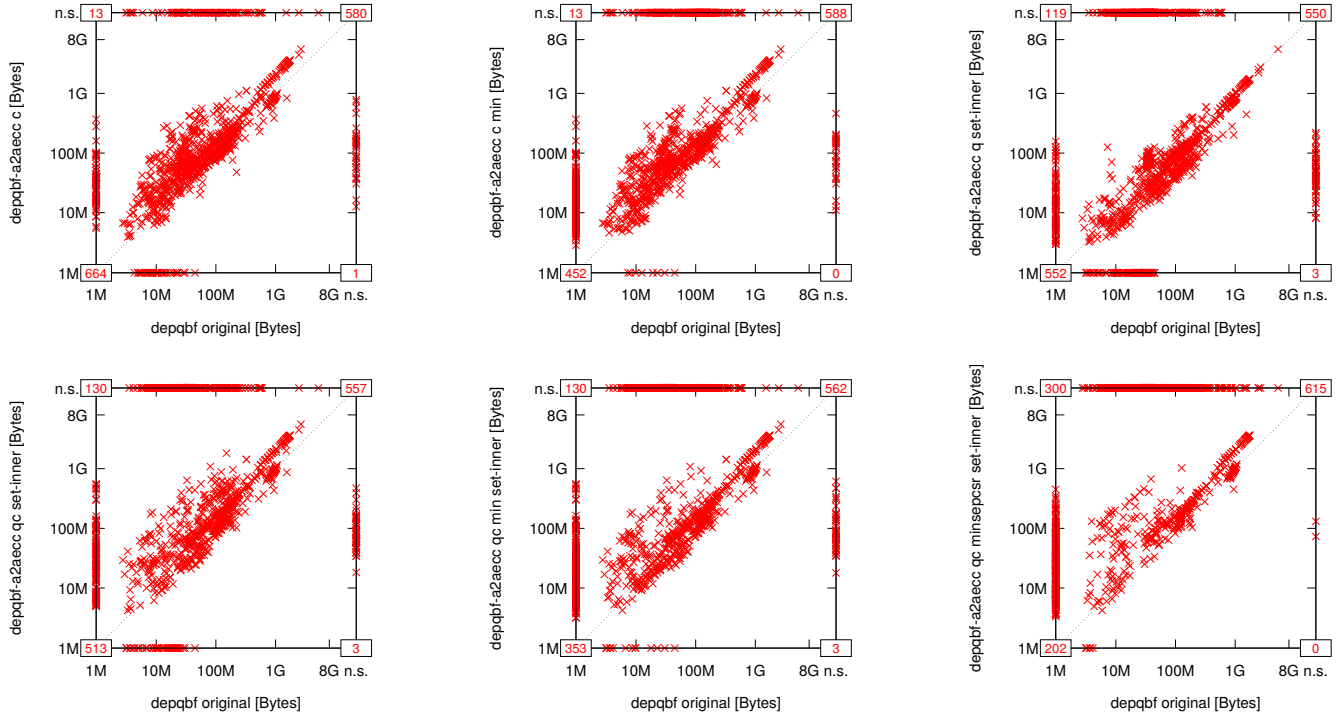


Fig. 524: Suite All ($n = 2528$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

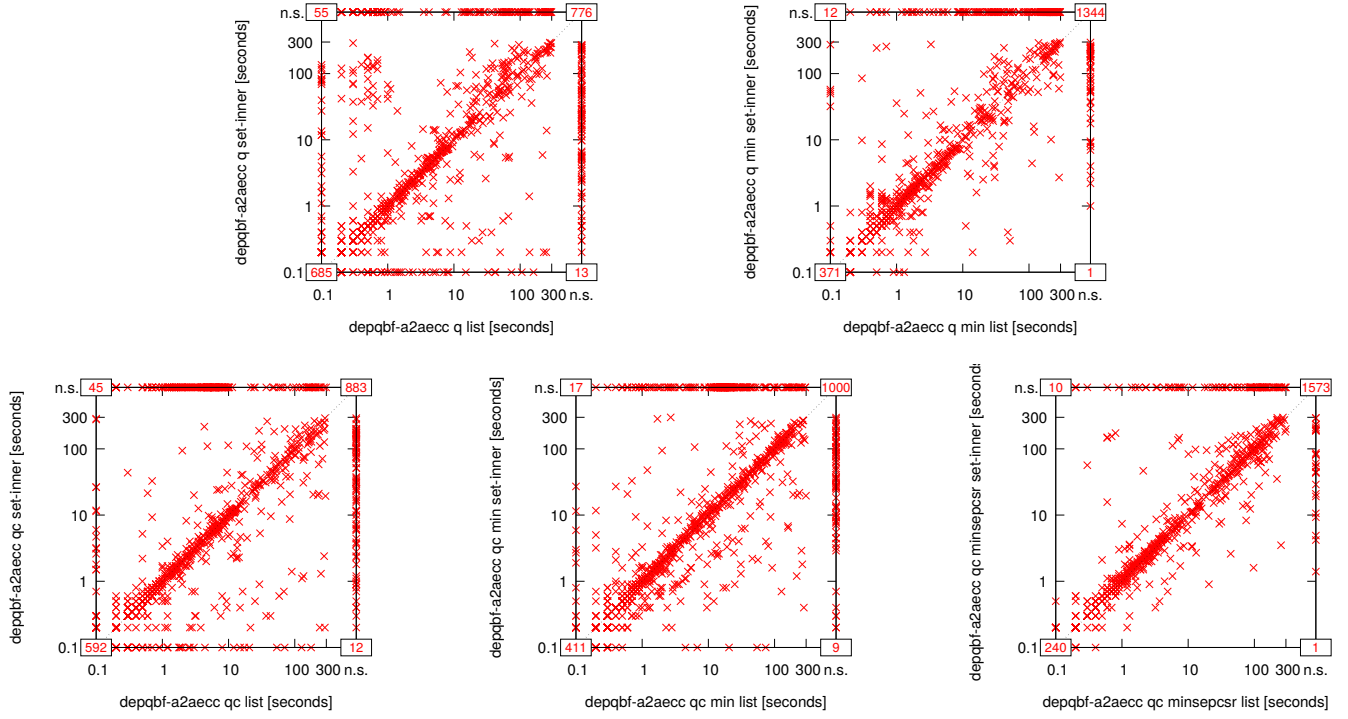


Fig. 525: Suite All ($n = 2528$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

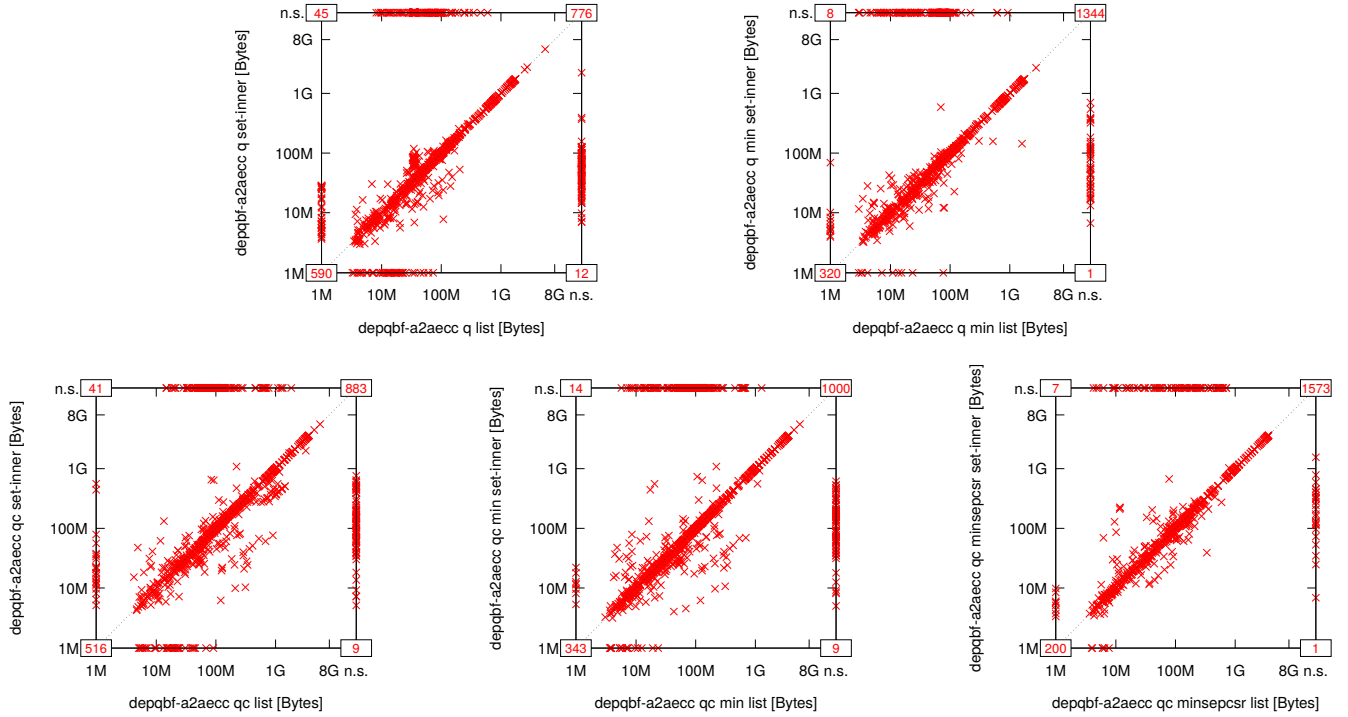


Fig. 526: Suite All ($n = 2528$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

2) Akshay-Chakraborty-John-Shah-Rabe ($n = 2$):

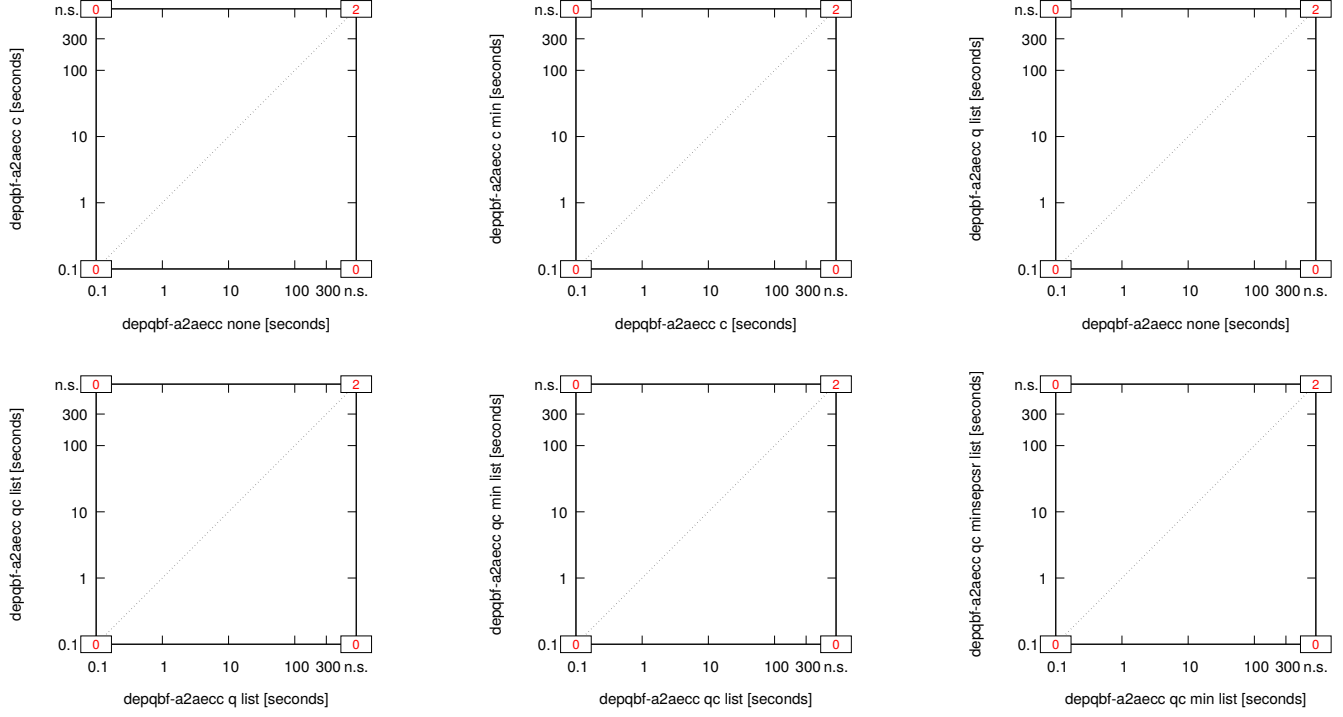


Fig. 527: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

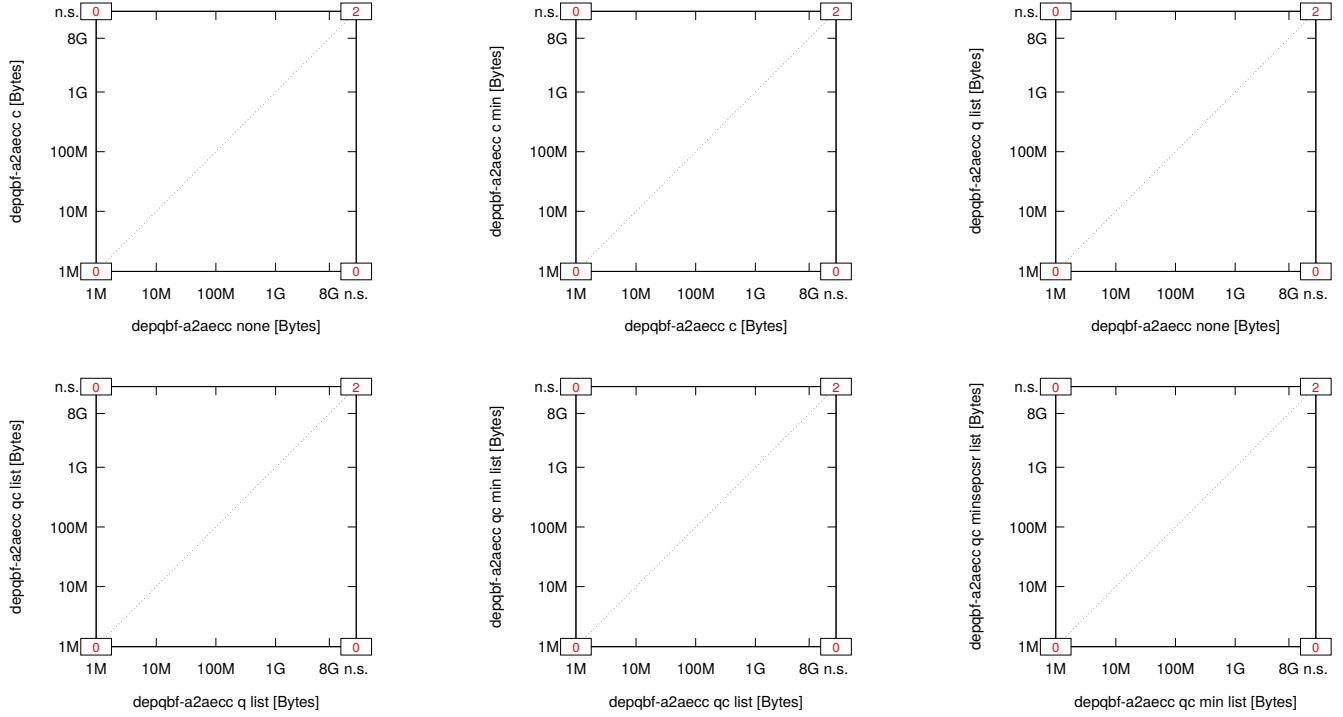


Fig. 528: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

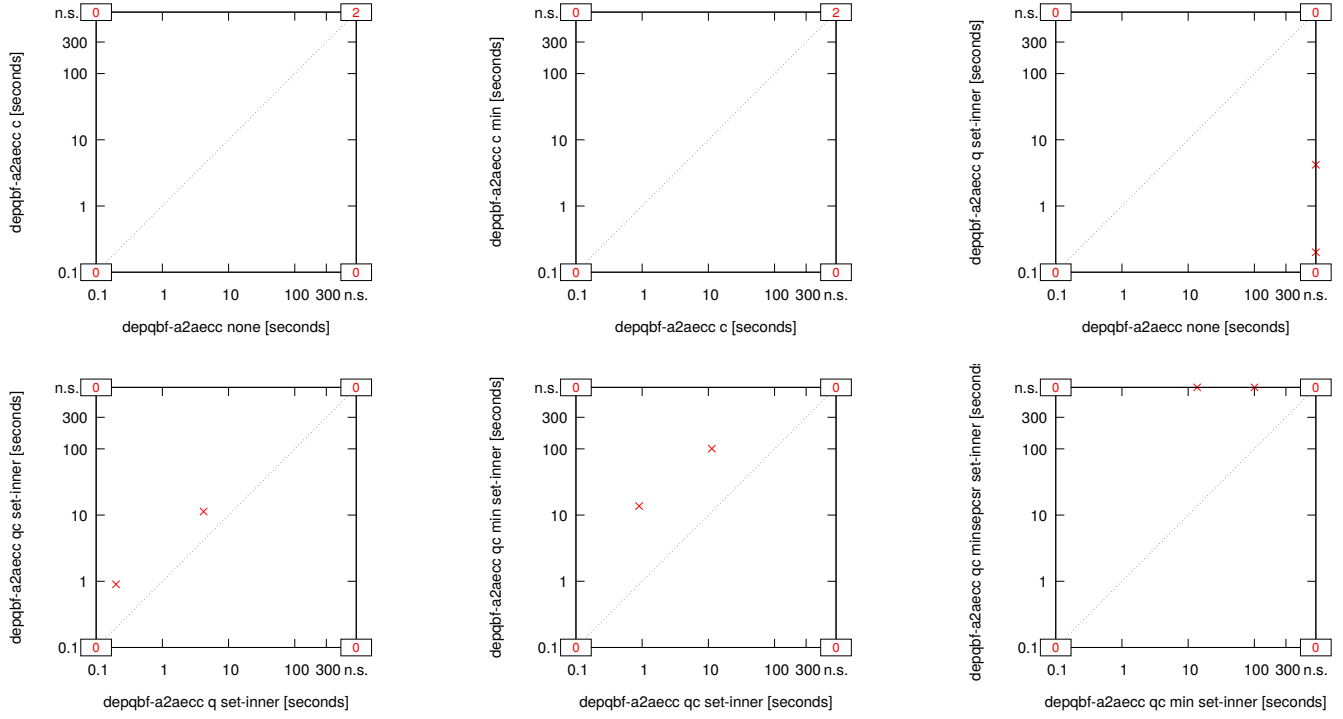


Fig. 529: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

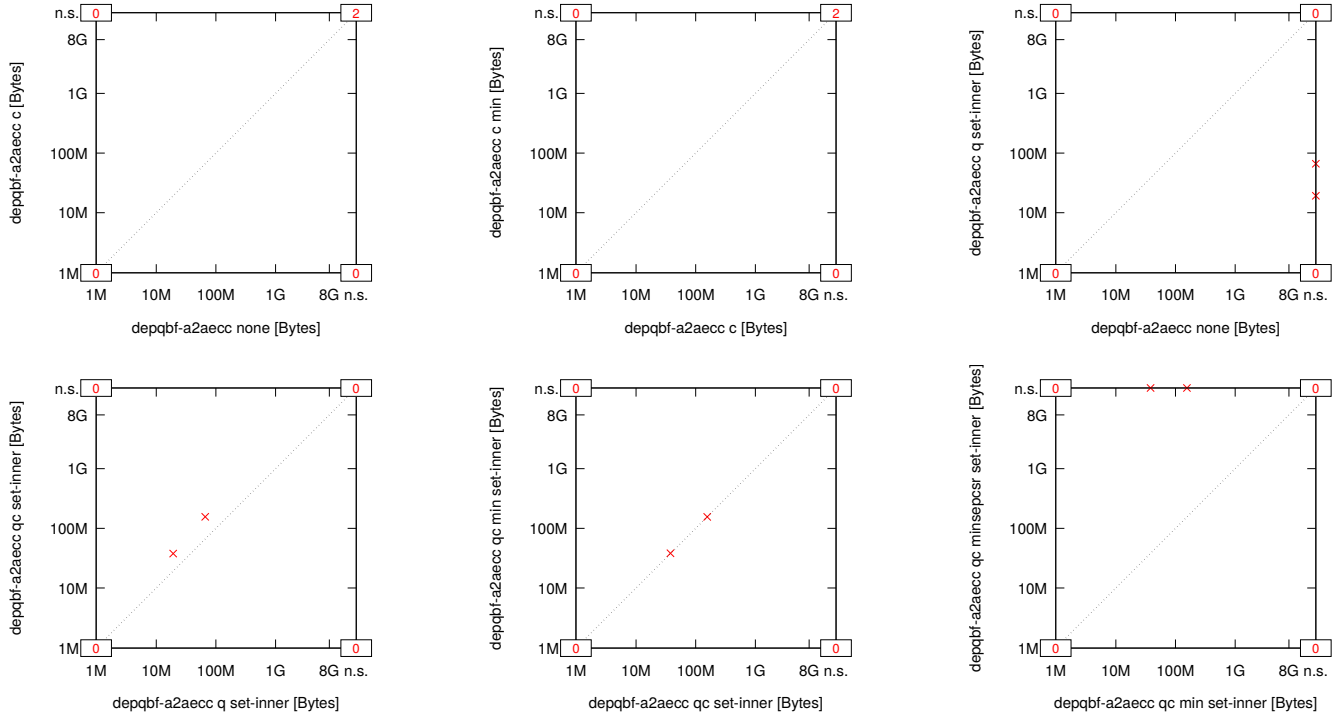


Fig. 530: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

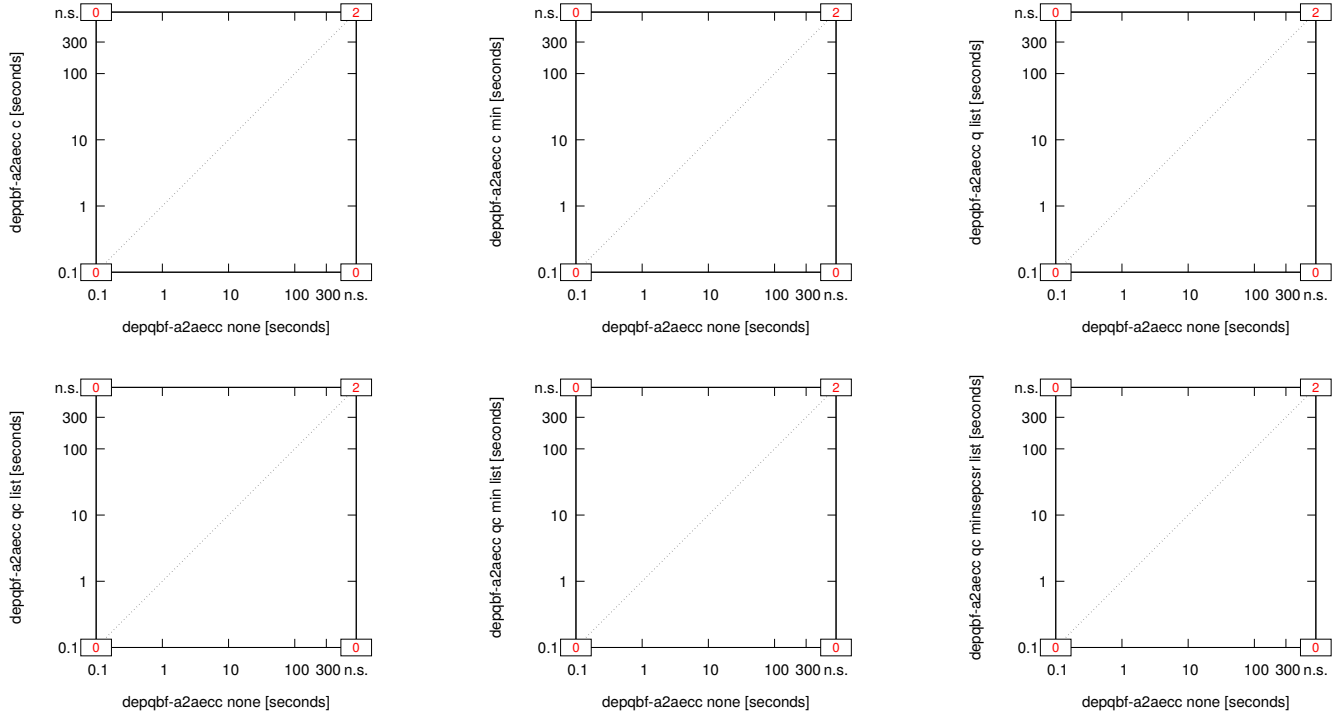


Fig. 531: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

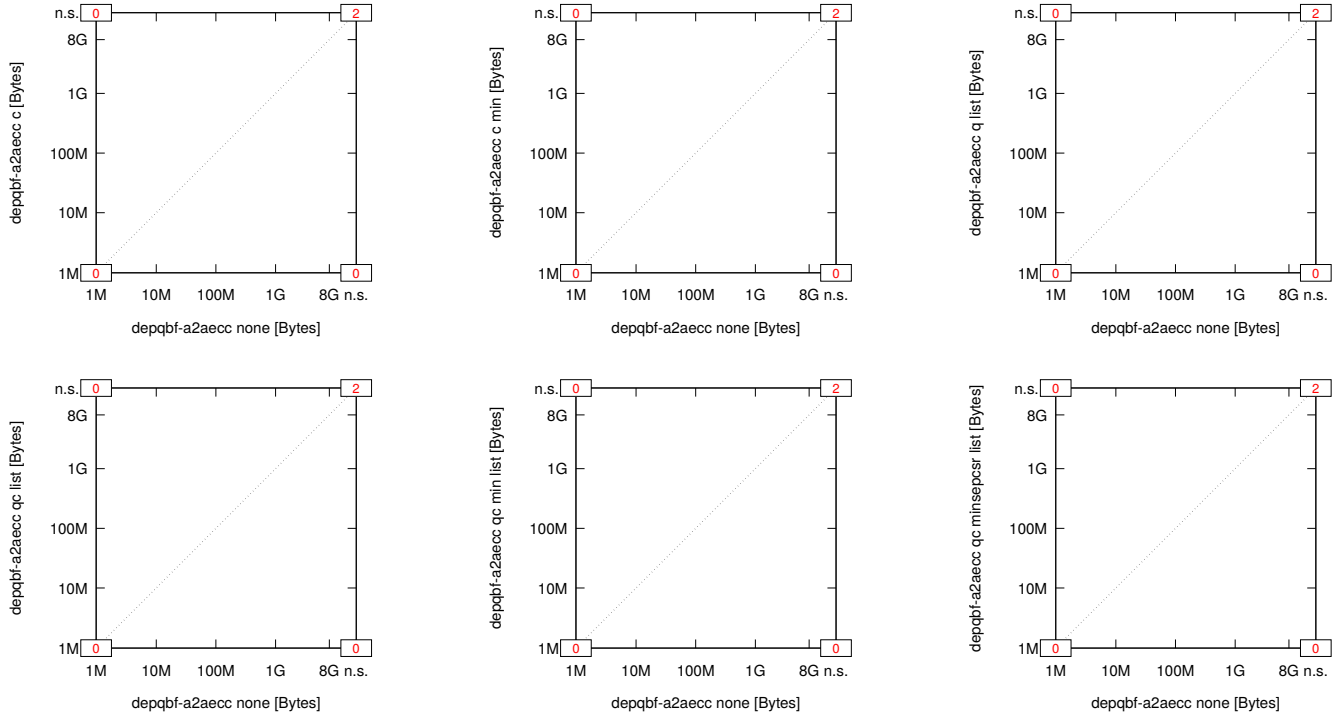


Fig. 532: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

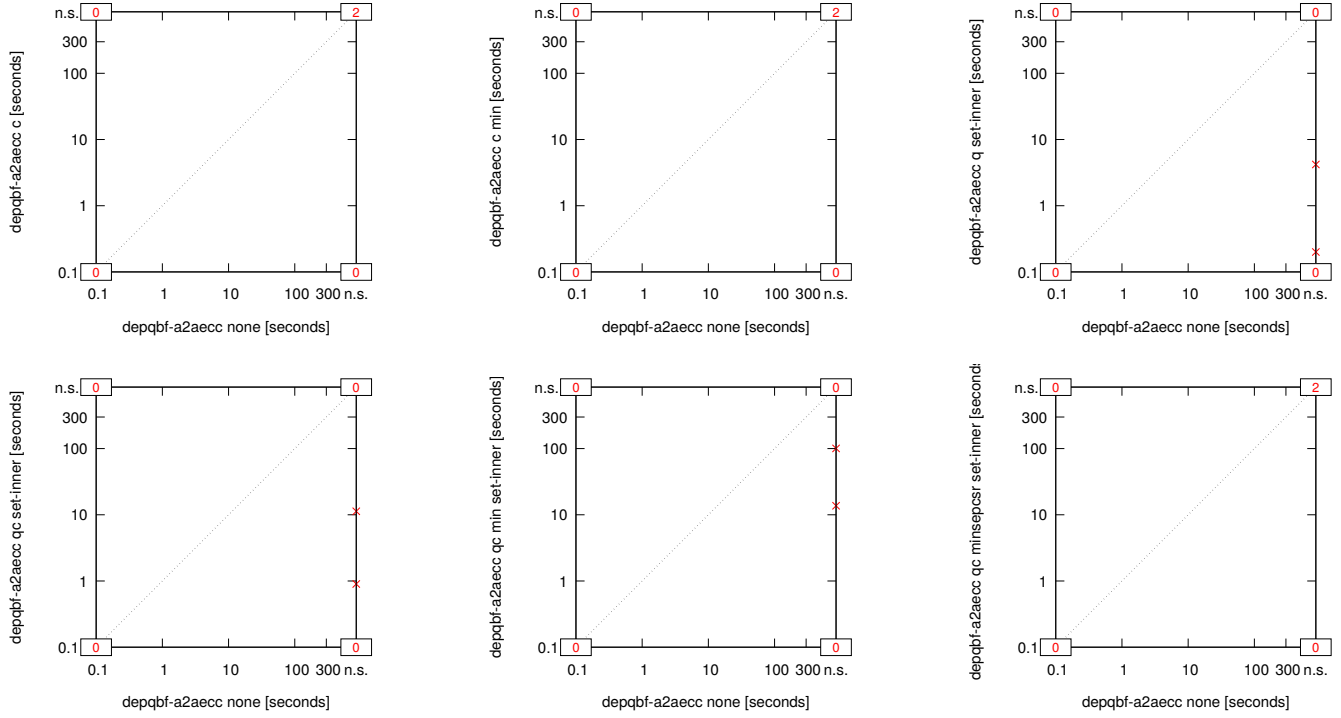


Fig. 533: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

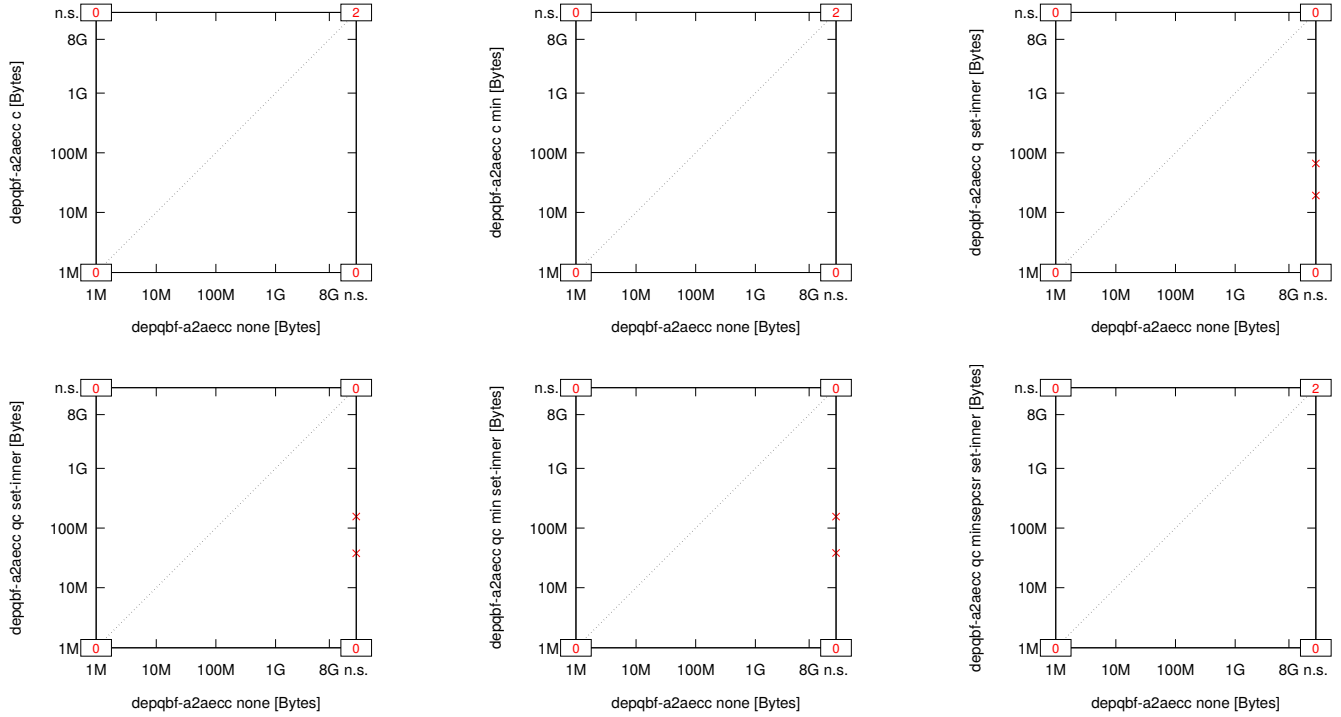


Fig. 534: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

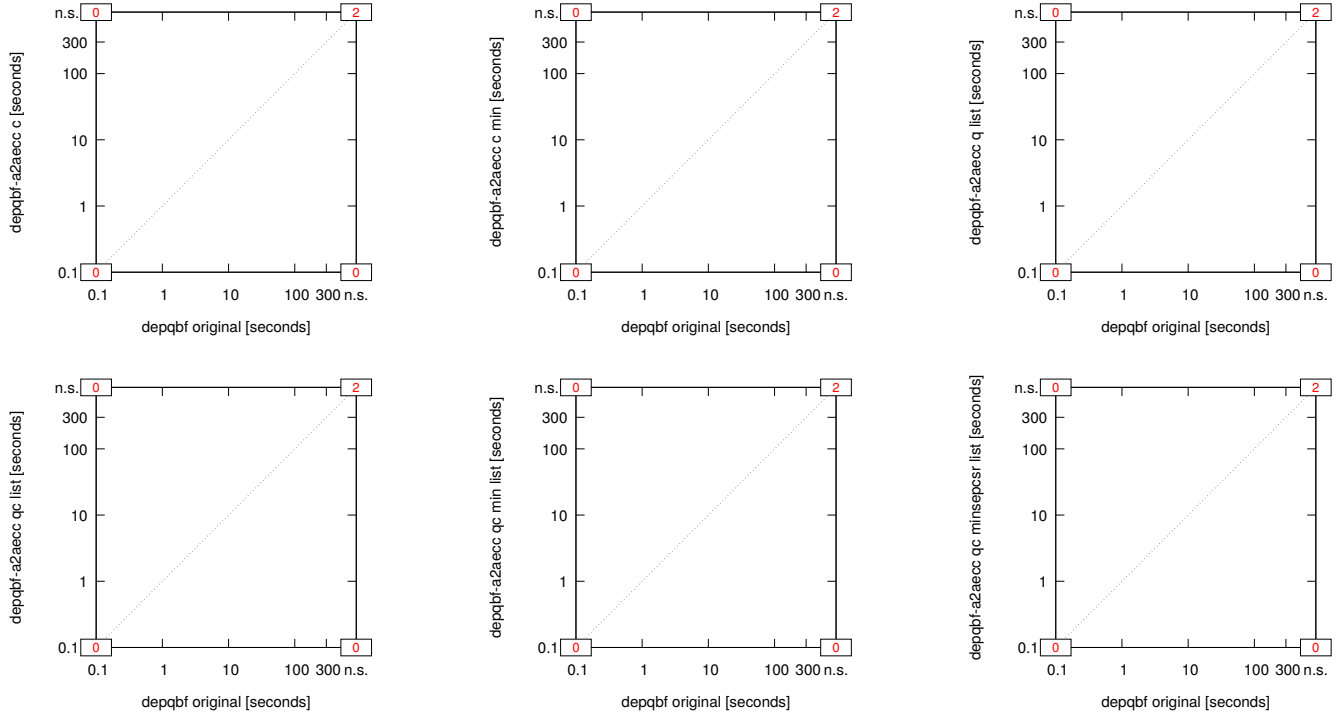


Fig. 535: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

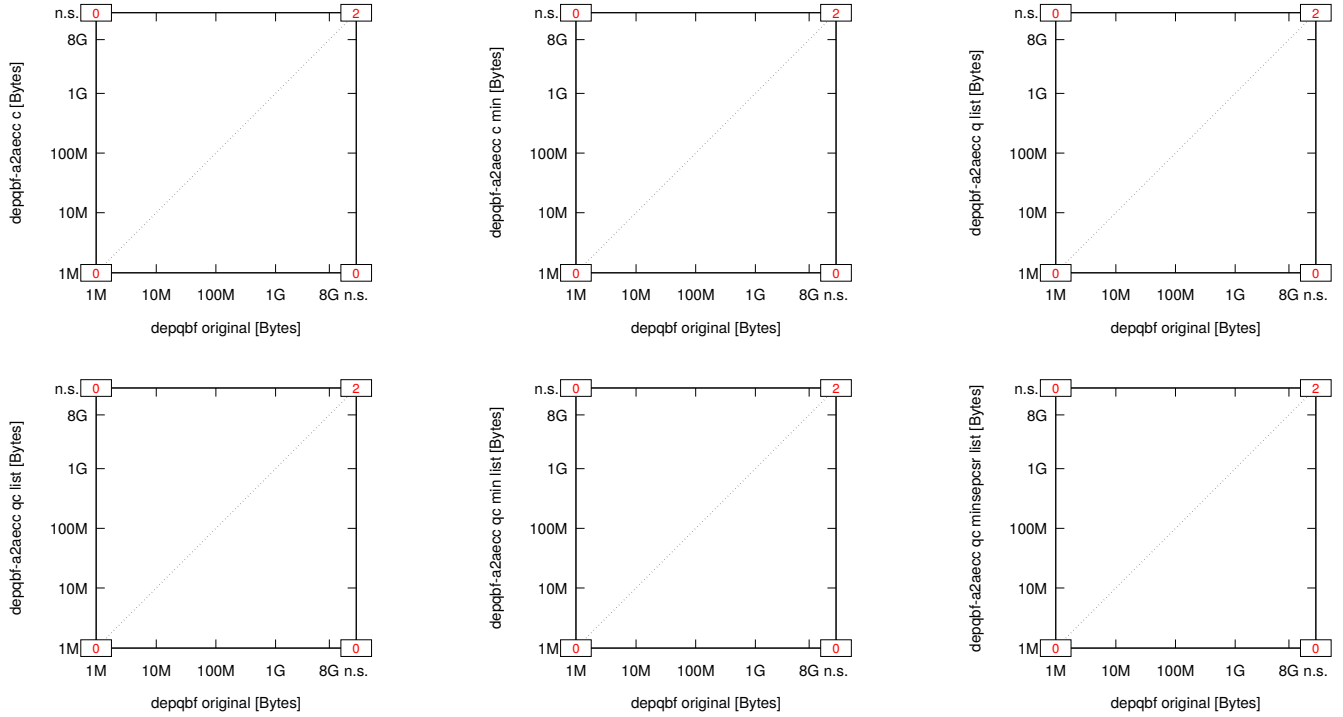


Fig. 536: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

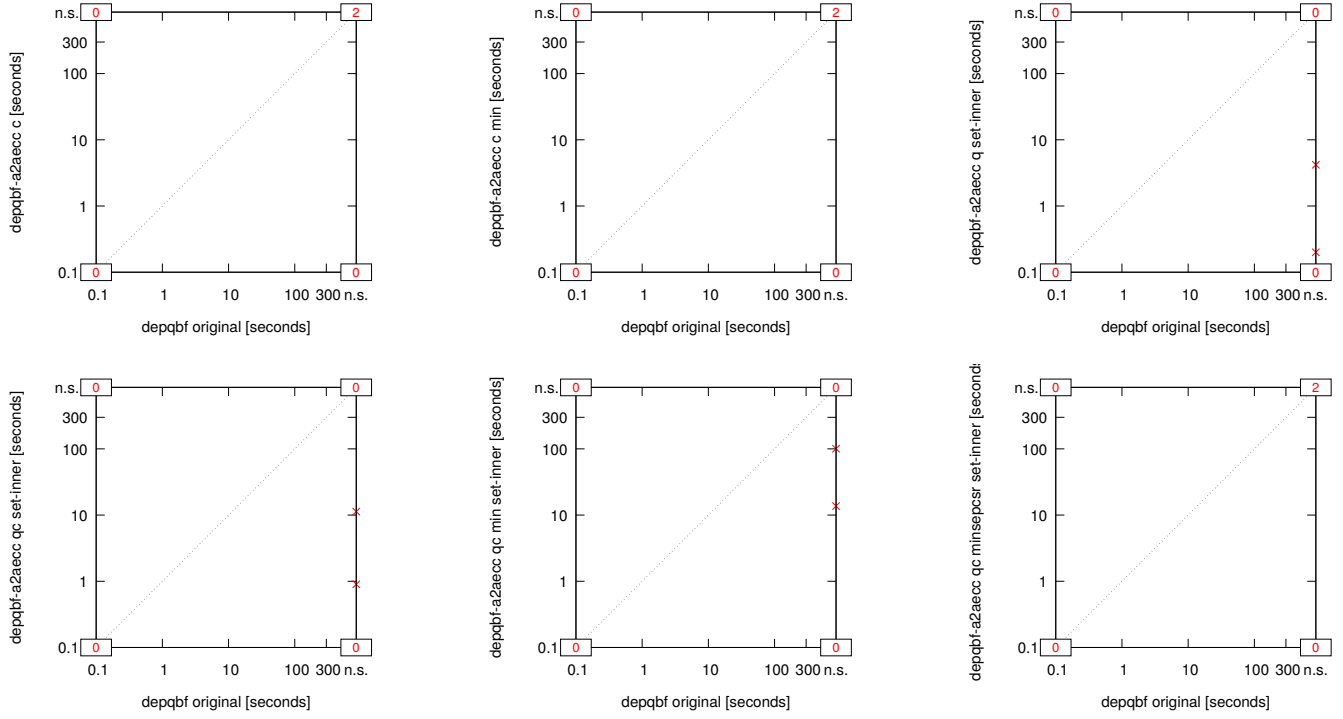


Fig. 537: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

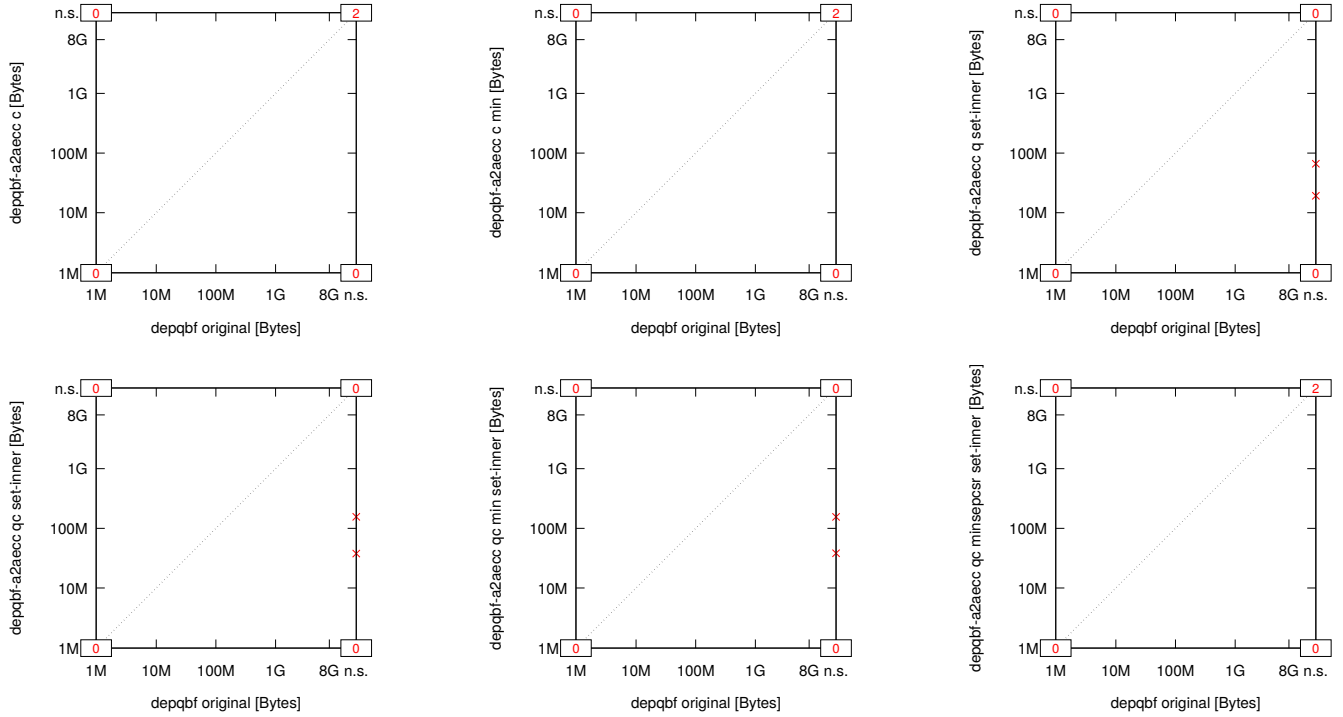


Fig. 538: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

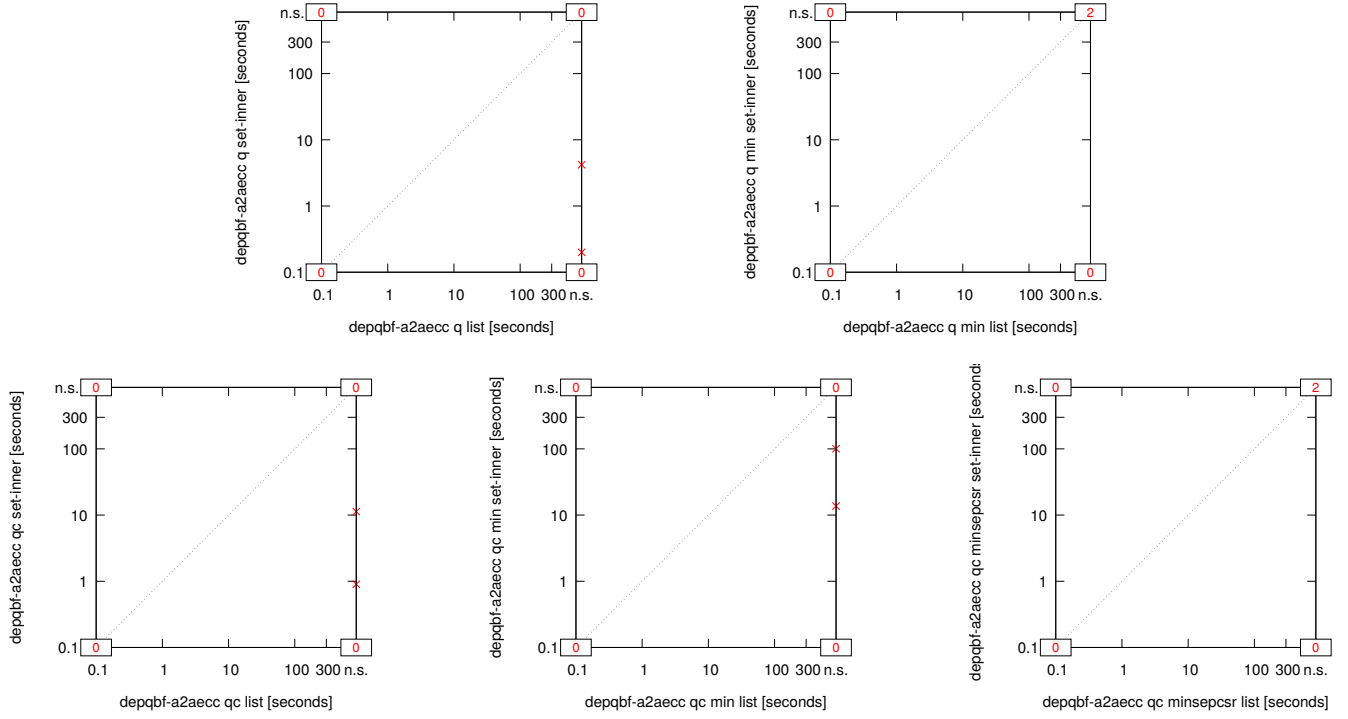


Fig. 539: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

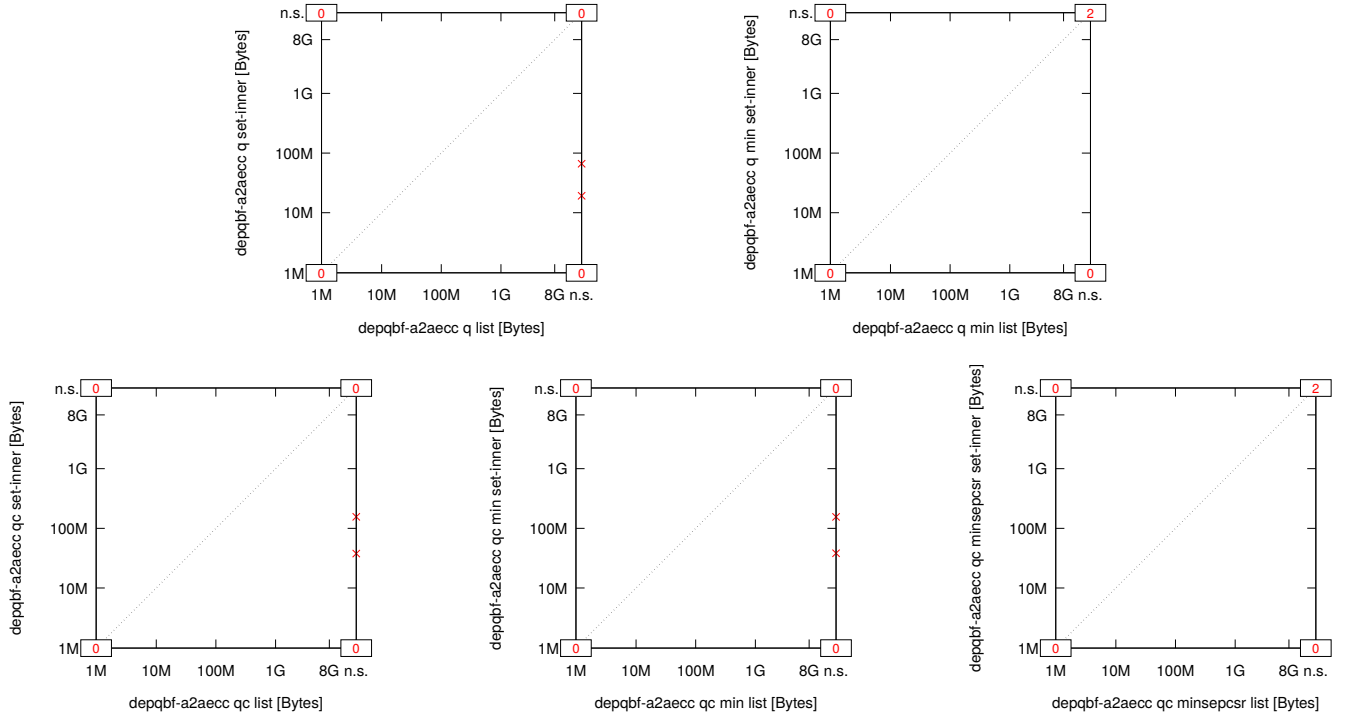


Fig. 540: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 2$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

3) Amendola-Ricca-Truszczyński ($n = 9$):

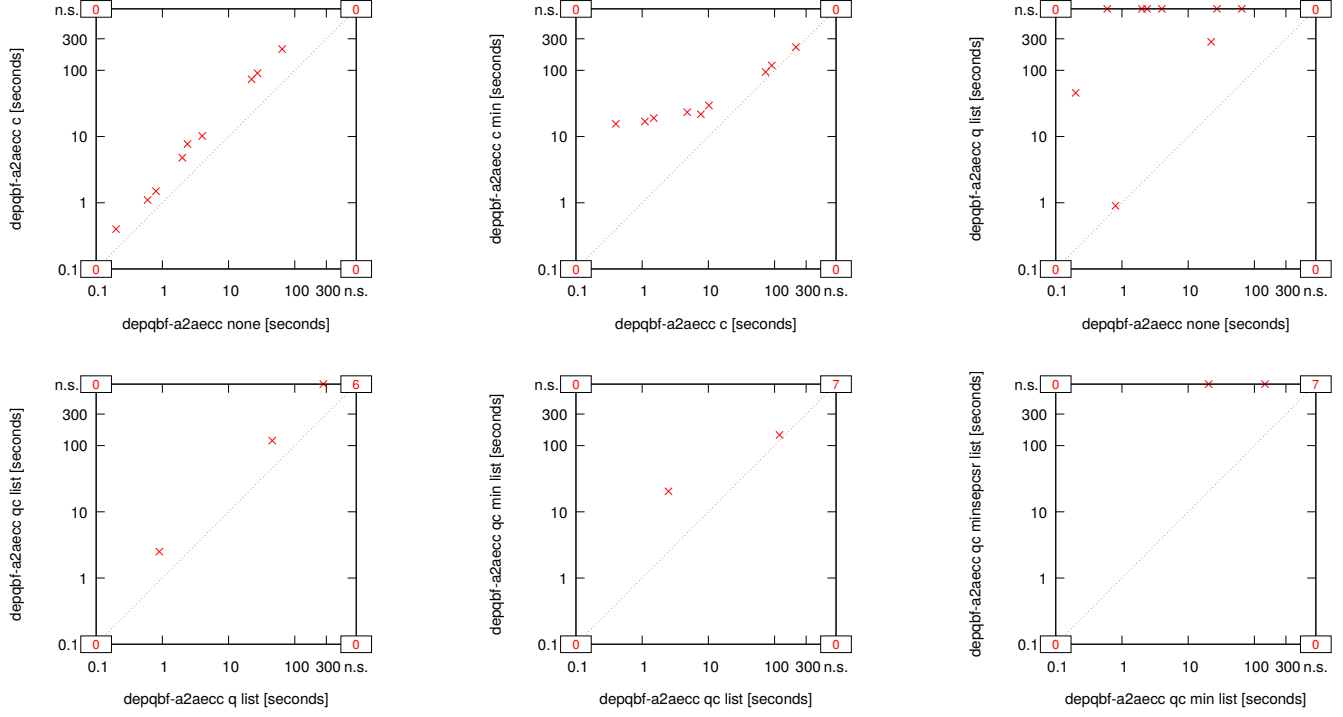


Fig. 541: Suite Amendola-Ricca-Truszczyński ($n = 9$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

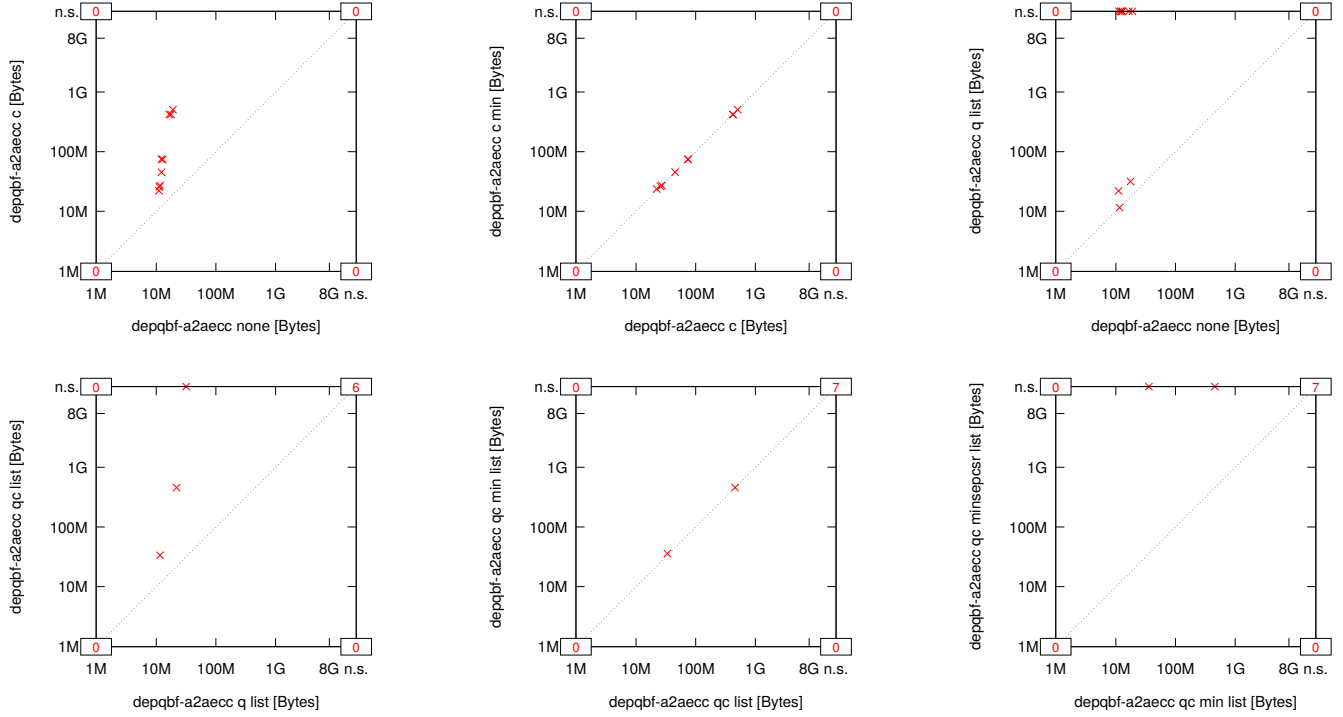


Fig. 542: Suite Amendola-Ricca-Truszczyński ($n = 9$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

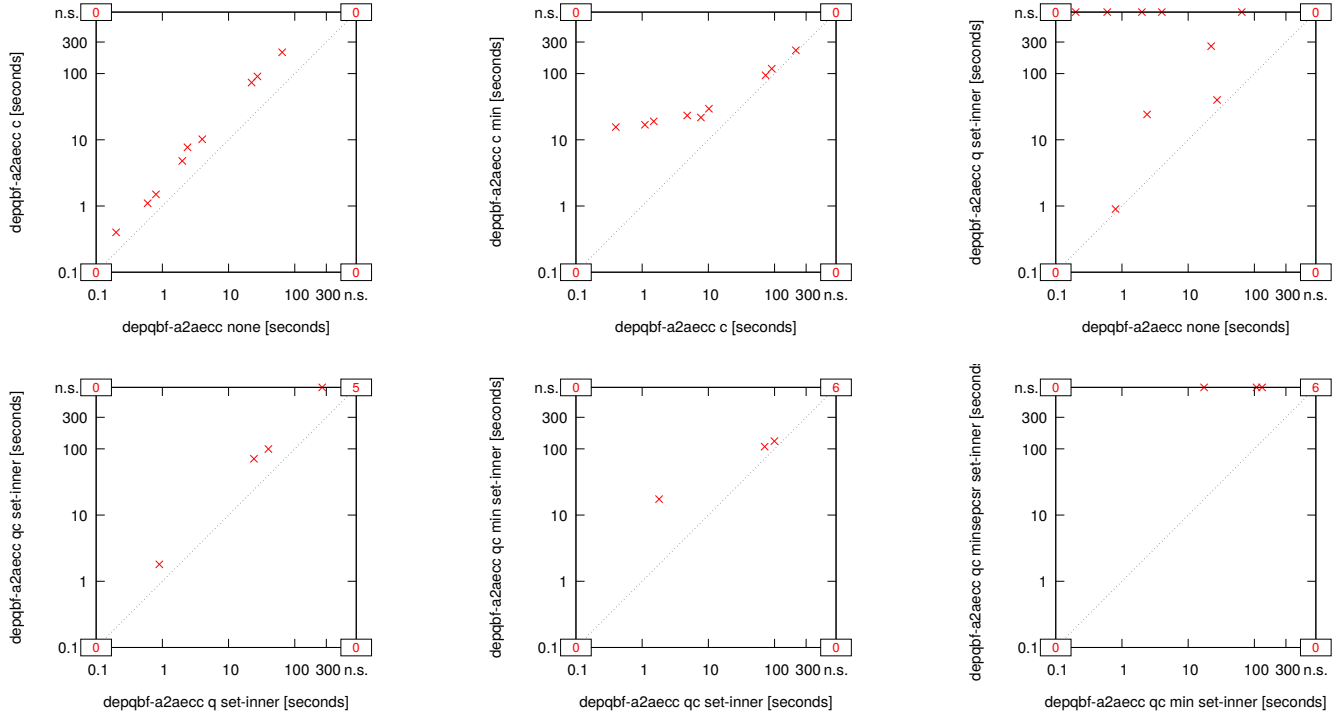


Fig. 543: Suite Amendola-Ricca-Truszczyński ($n = 9$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

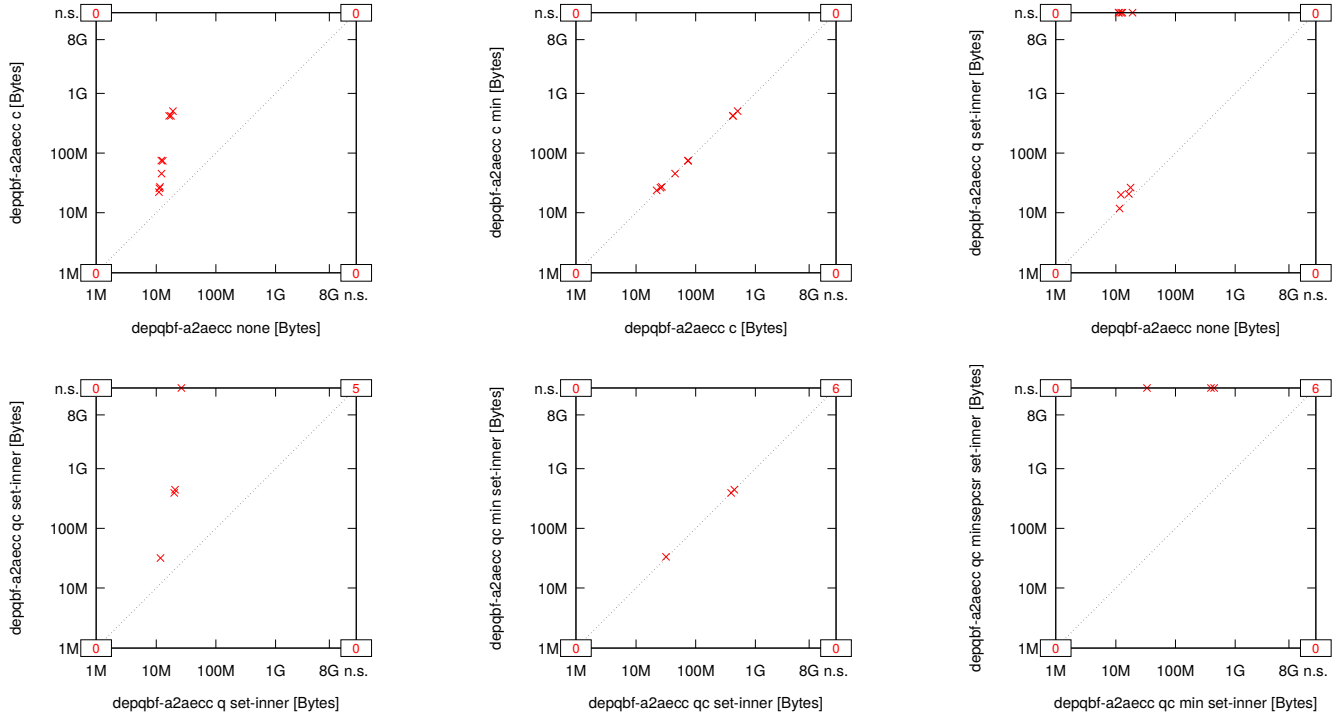


Fig. 544: Suite Amendola-Ricca-Truszczyński ($n = 9$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

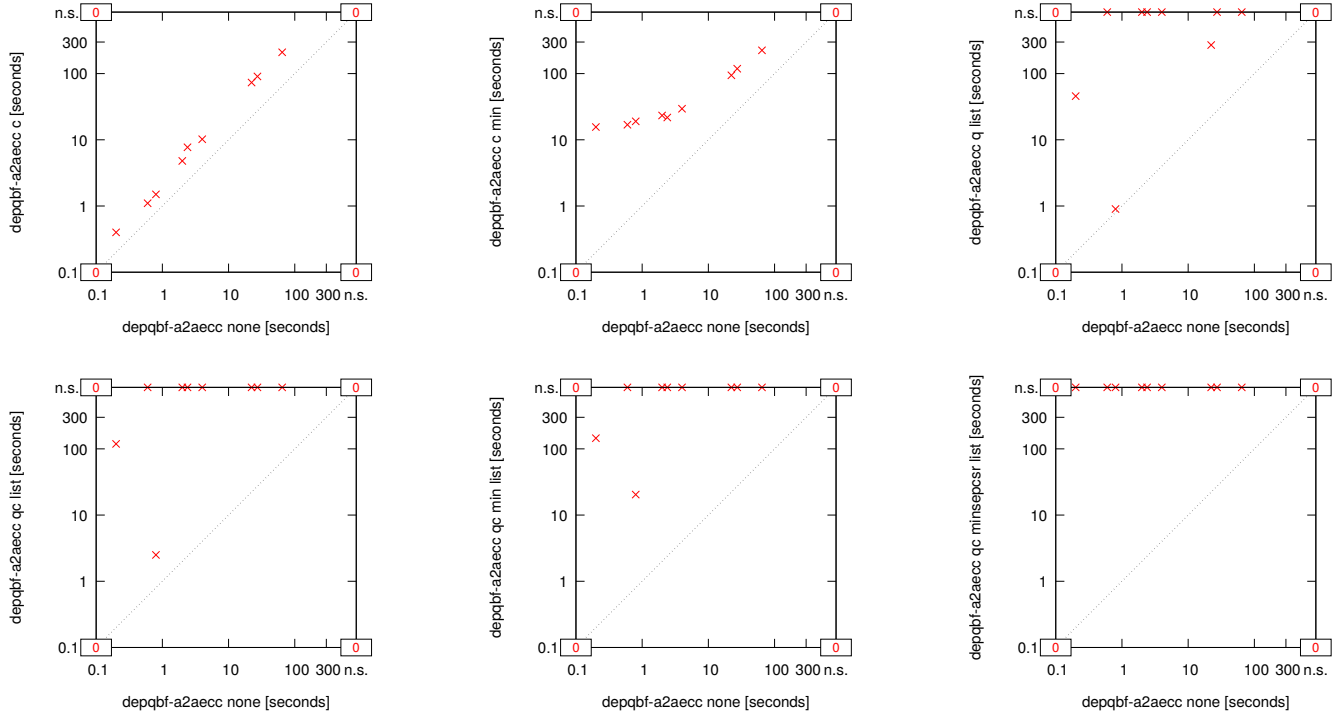


Fig. 545: Suite Amendola-Ricca-Truszczyński ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

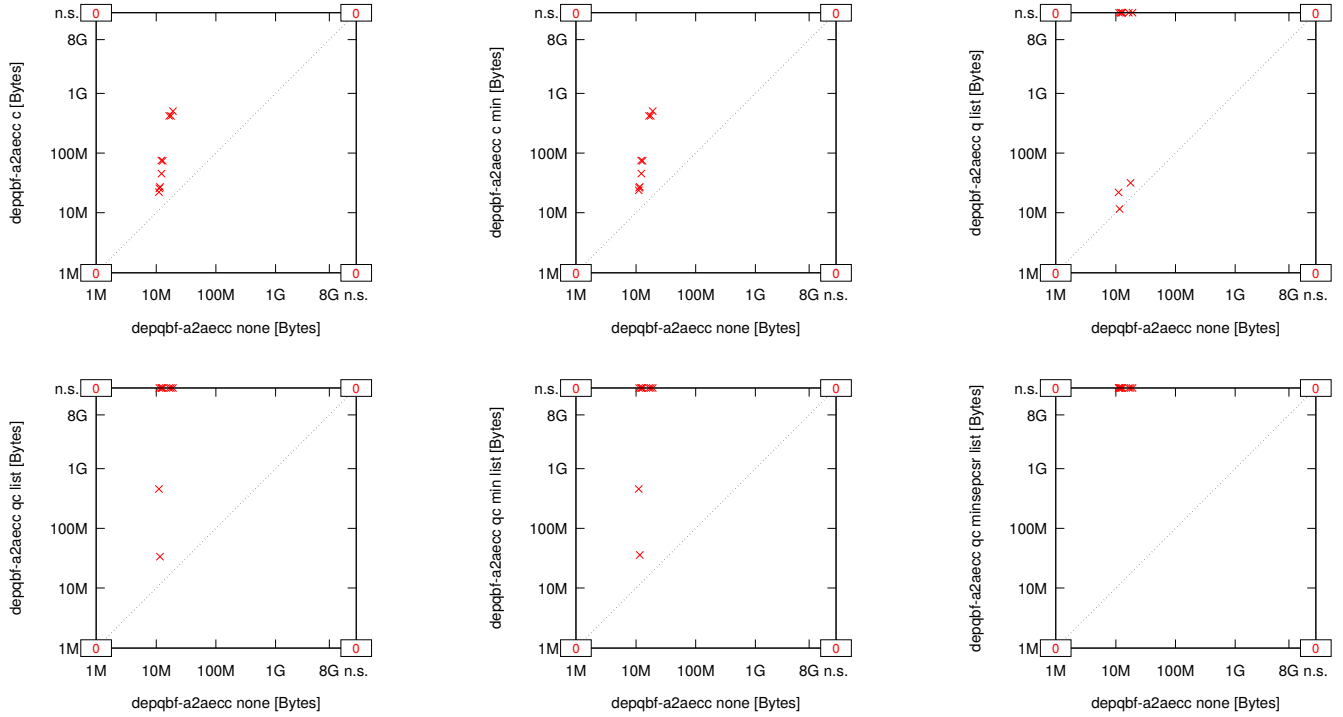


Fig. 546: Suite Amendola-Ricca-Truszczyński ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

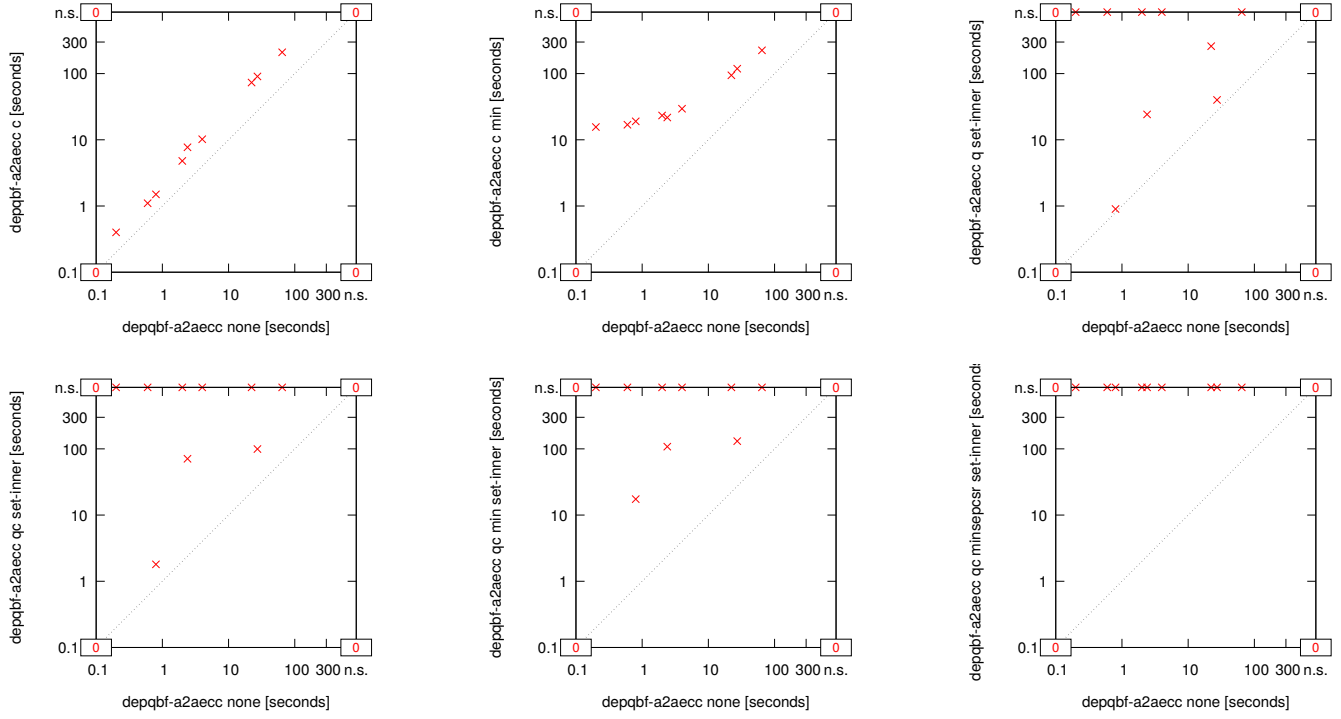


Fig. 547: Suite Amendola-Ricca-Truszczyński ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

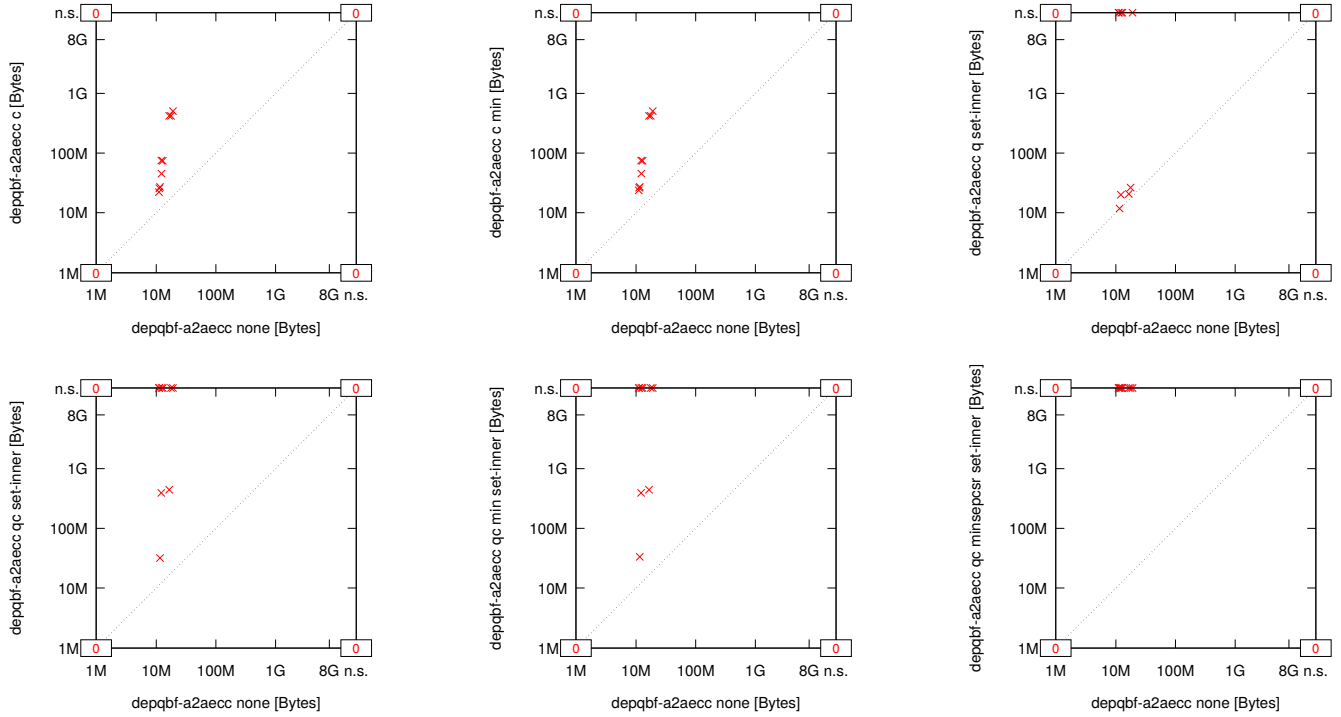


Fig. 548: Suite Amendola-Ricca-Truszczyński ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

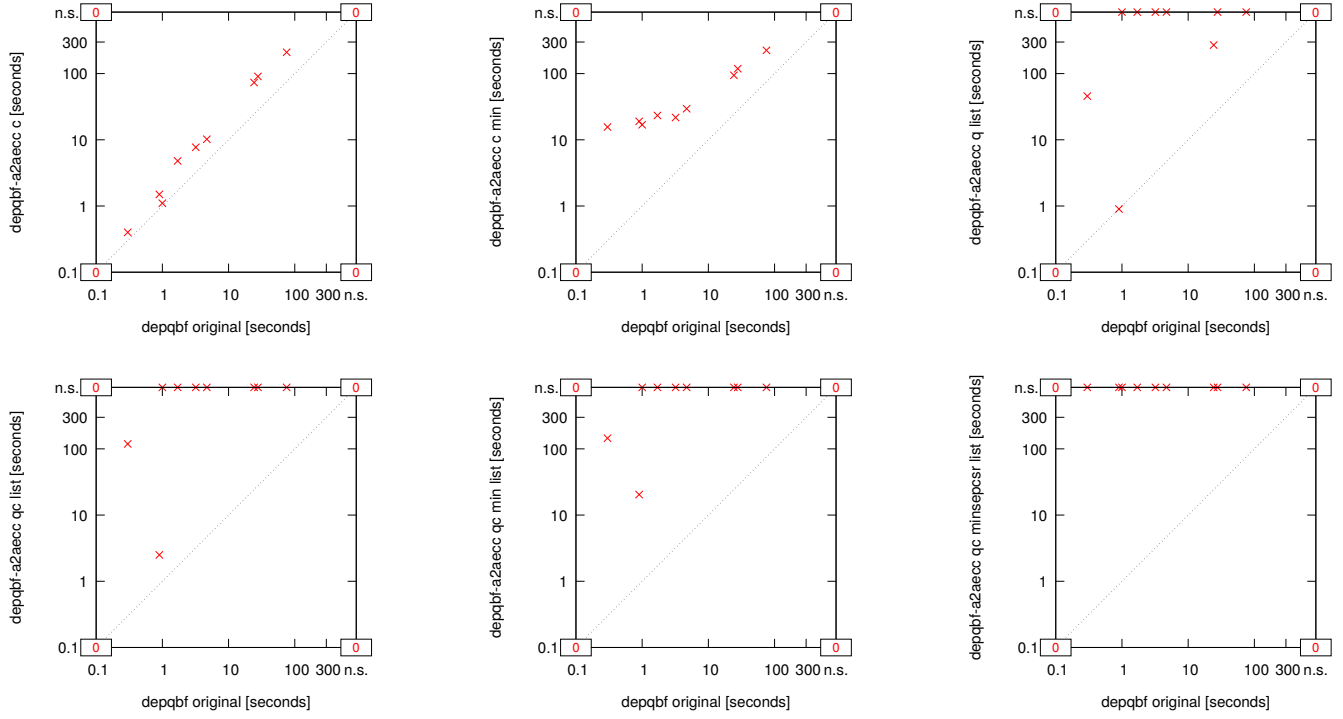


Fig. 549: Suite Amendola-Ricca-Truszczyński ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

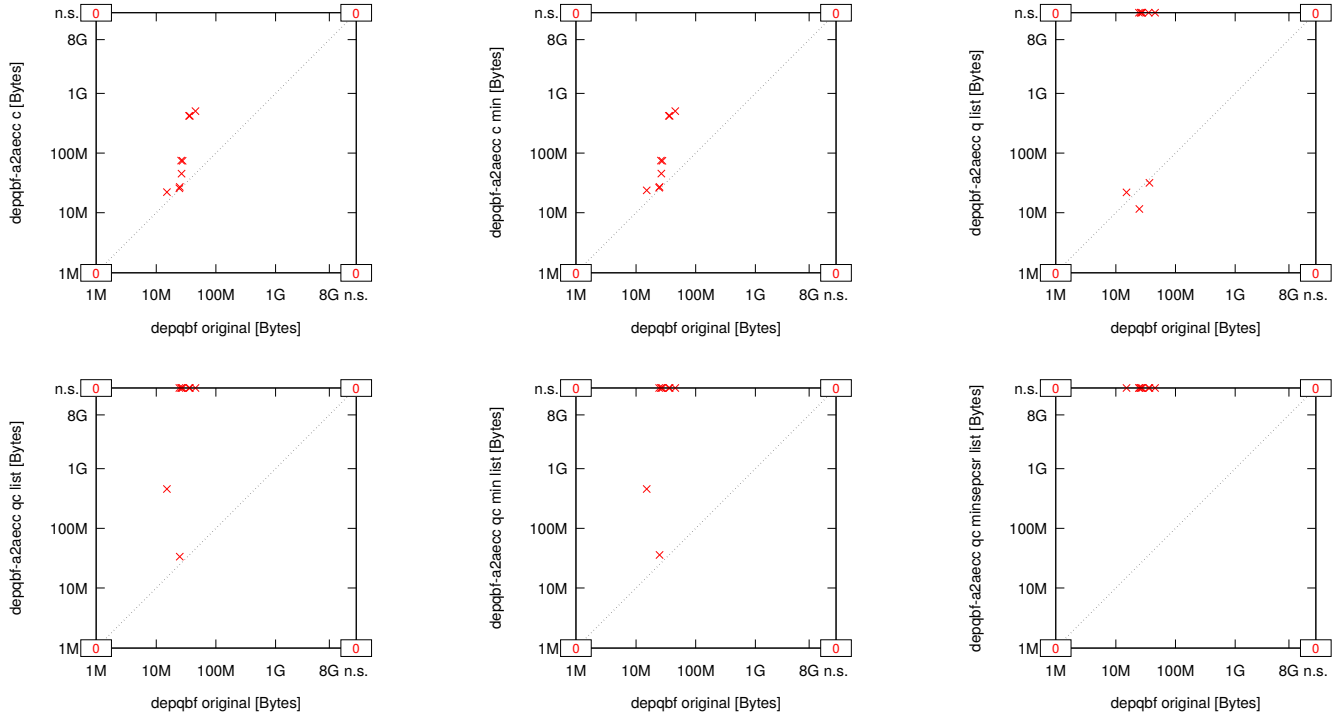


Fig. 550: Suite Amendola-Ricca-Truszczyński ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

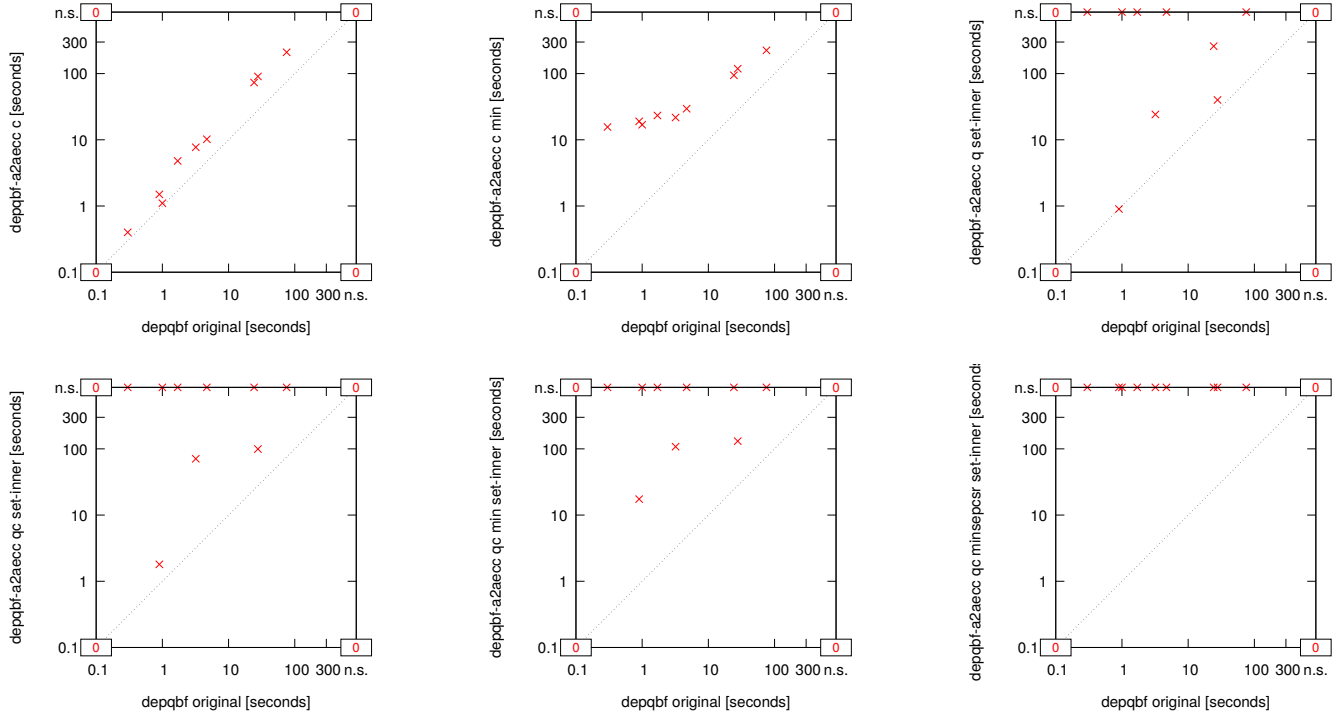


Fig. 551: Suite Amendola-Ricca-Truszczyński ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

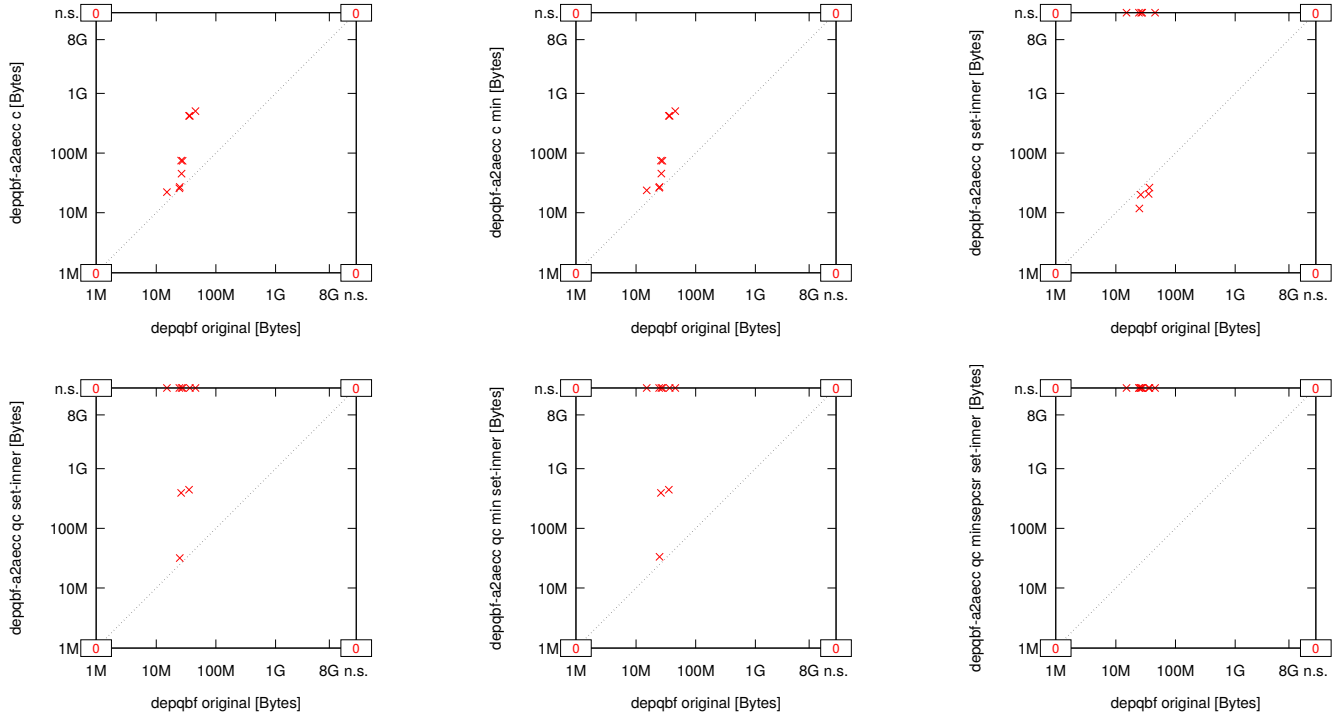


Fig. 552: Suite Amendola-Ricca-Truszczyński ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

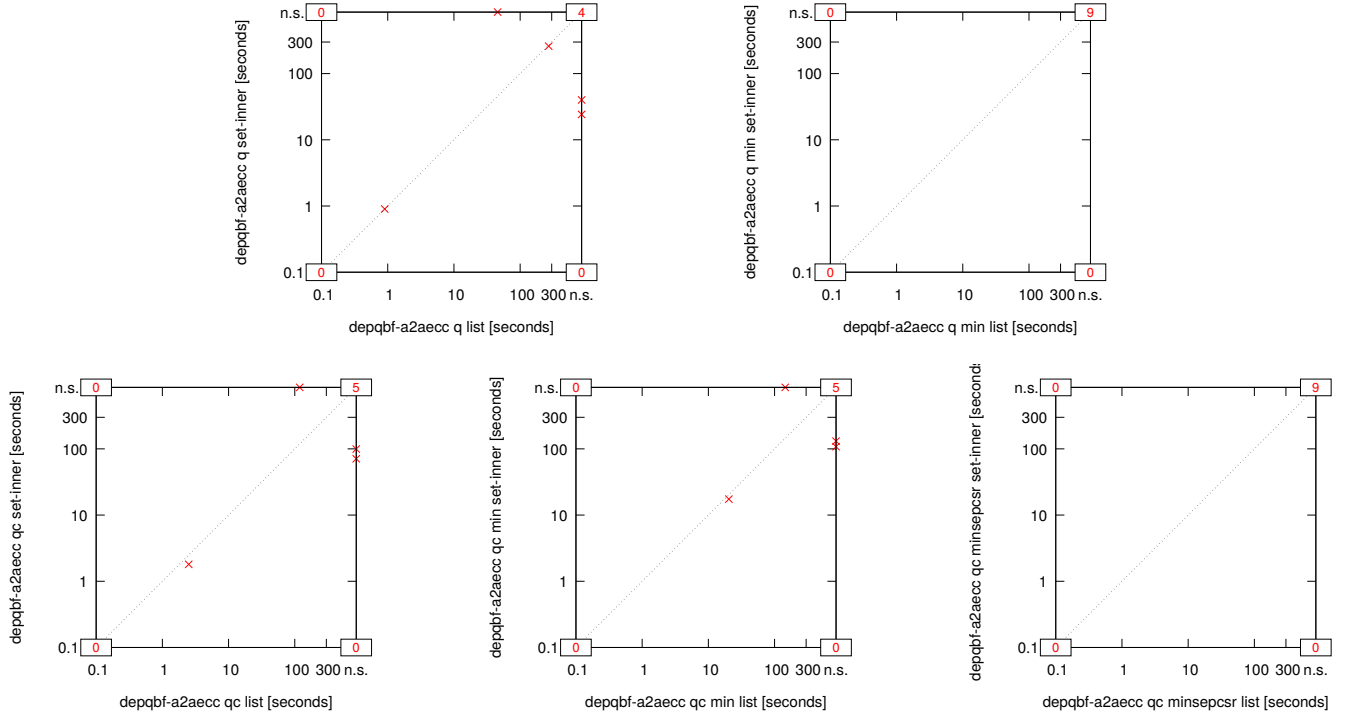


Fig. 553: Suite Amendola-Ricca-Truszczyński ($n = 9$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

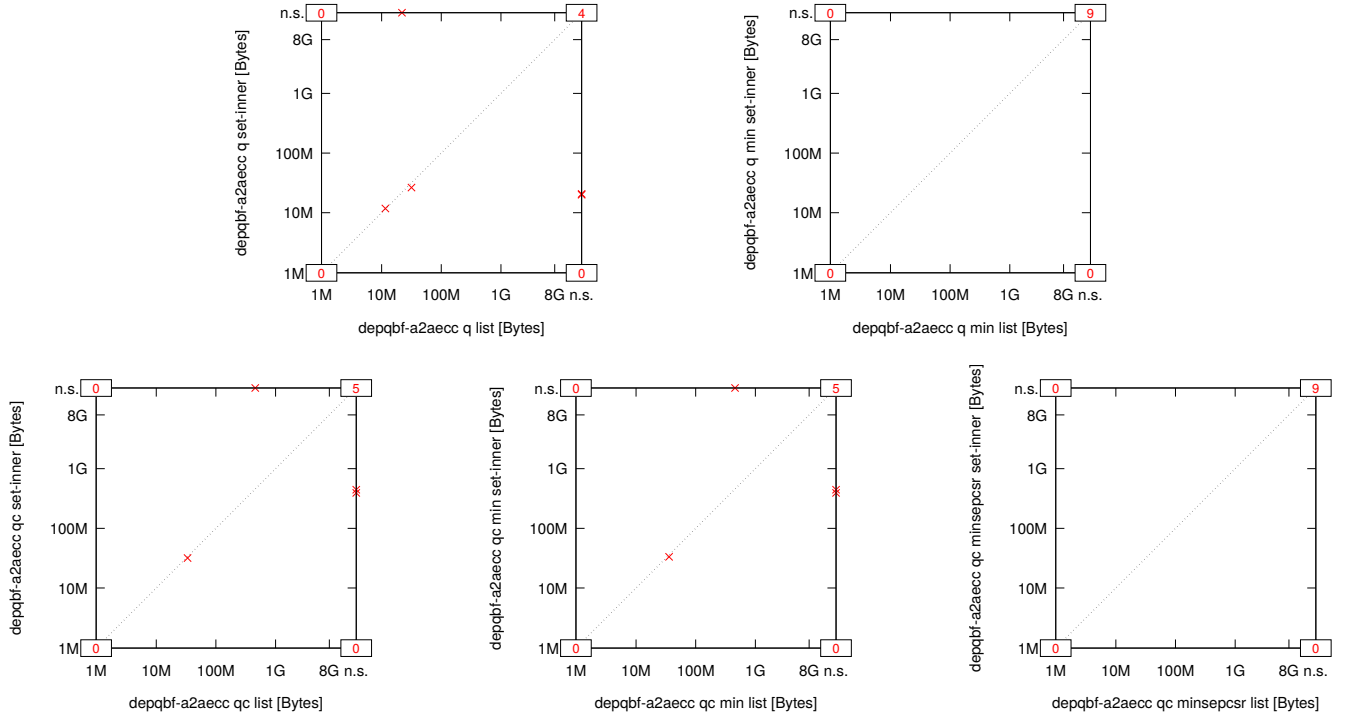


Fig. 554: Suite Amendola-Ricca-Truszczyński ($n = 9$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

4) Ansotegui ($n = 12$):

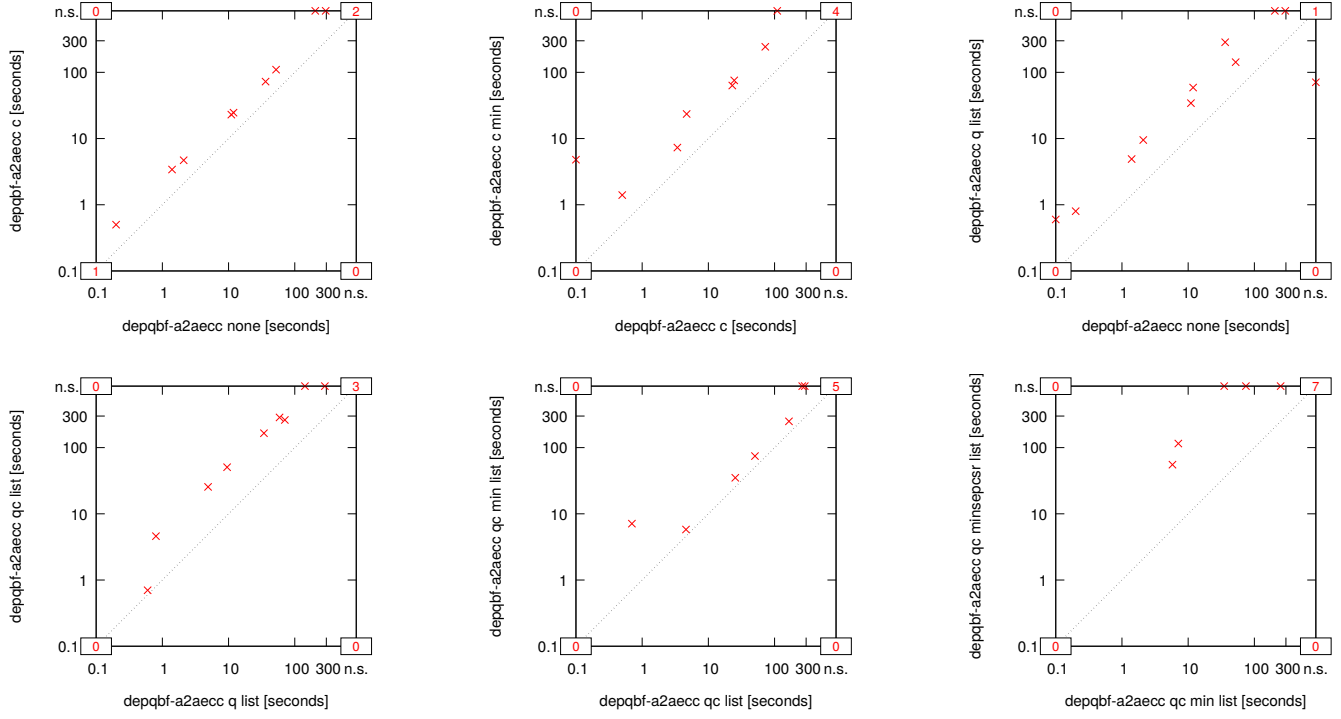


Fig. 555: Suite Ansotegui ($n = 12$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

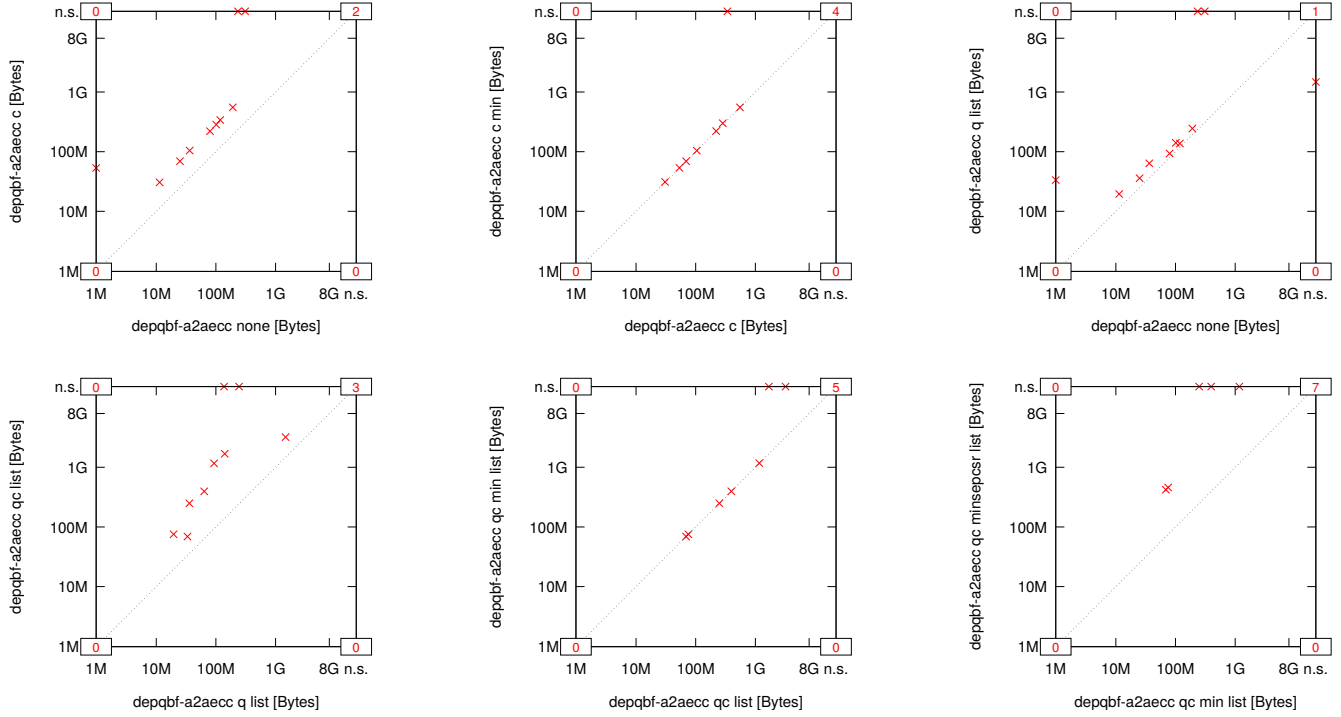


Fig. 556: Suite Ansotegui ($n = 12$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

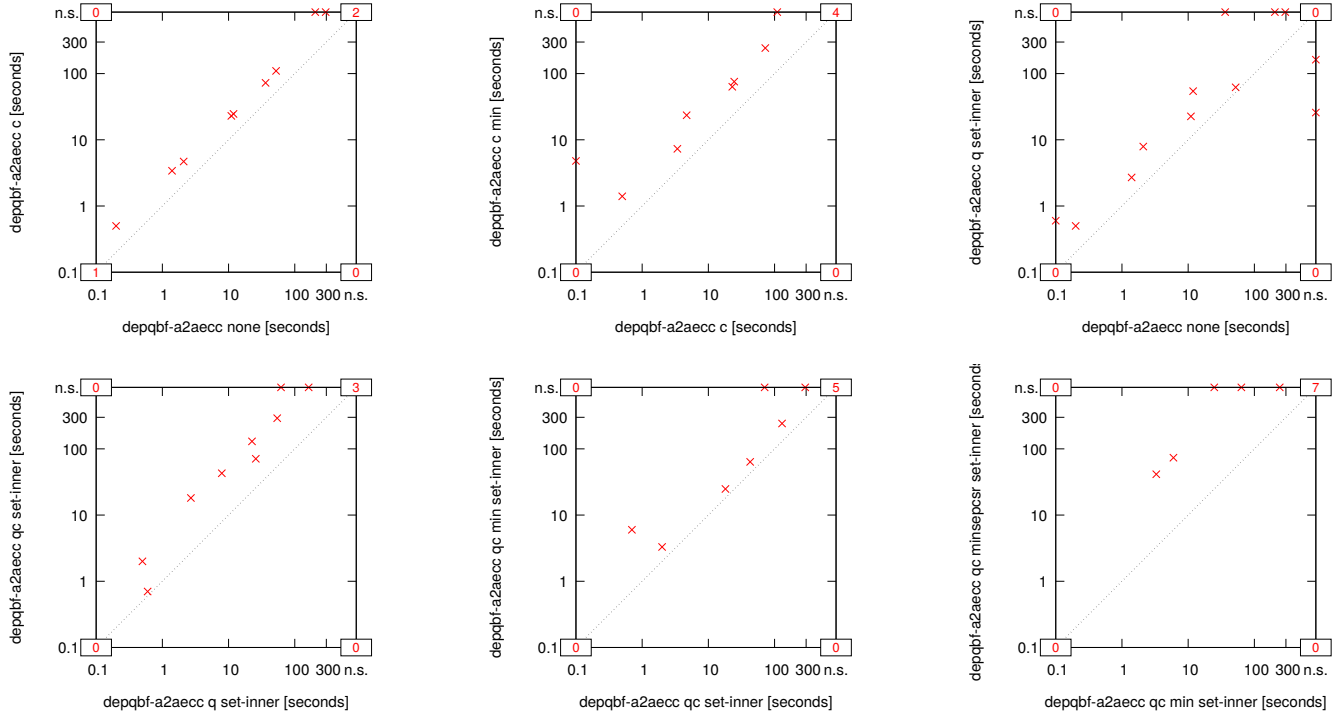


Fig. 557: Suite Ansotegui ($n = 12$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

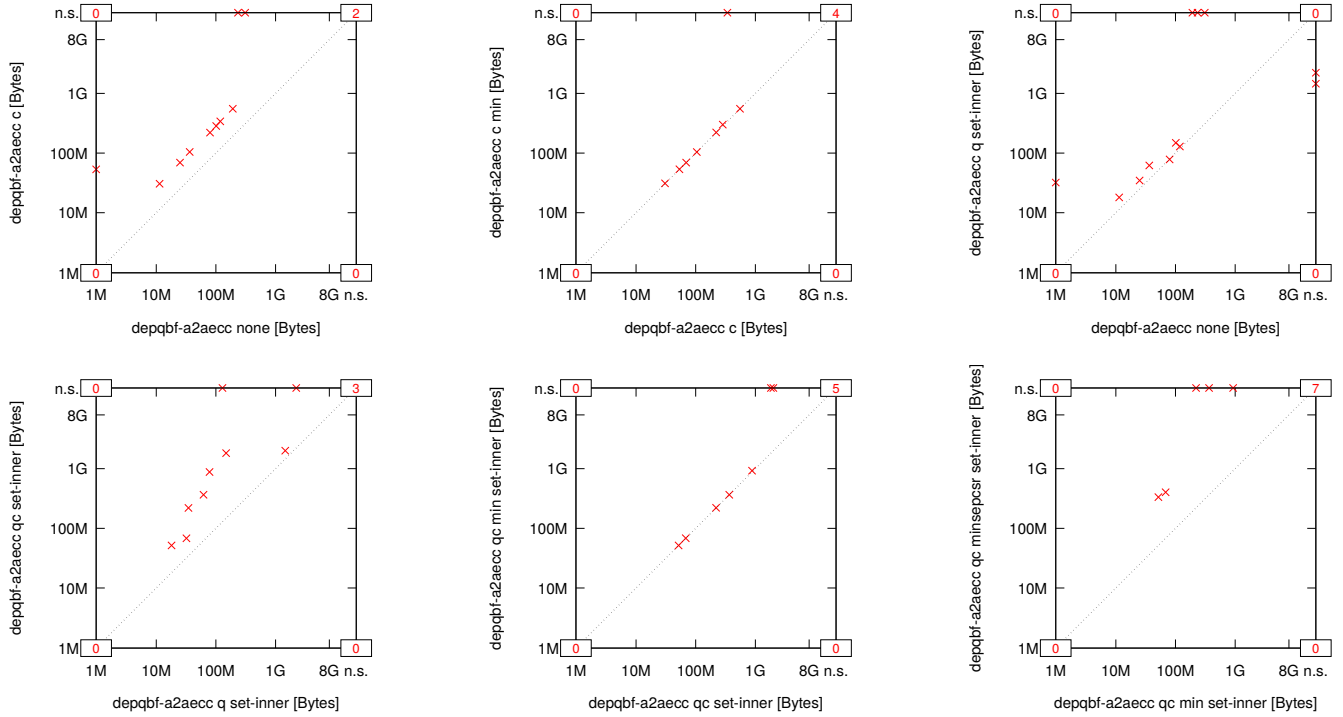


Fig. 558: Suite Ansotegui ($n = 12$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

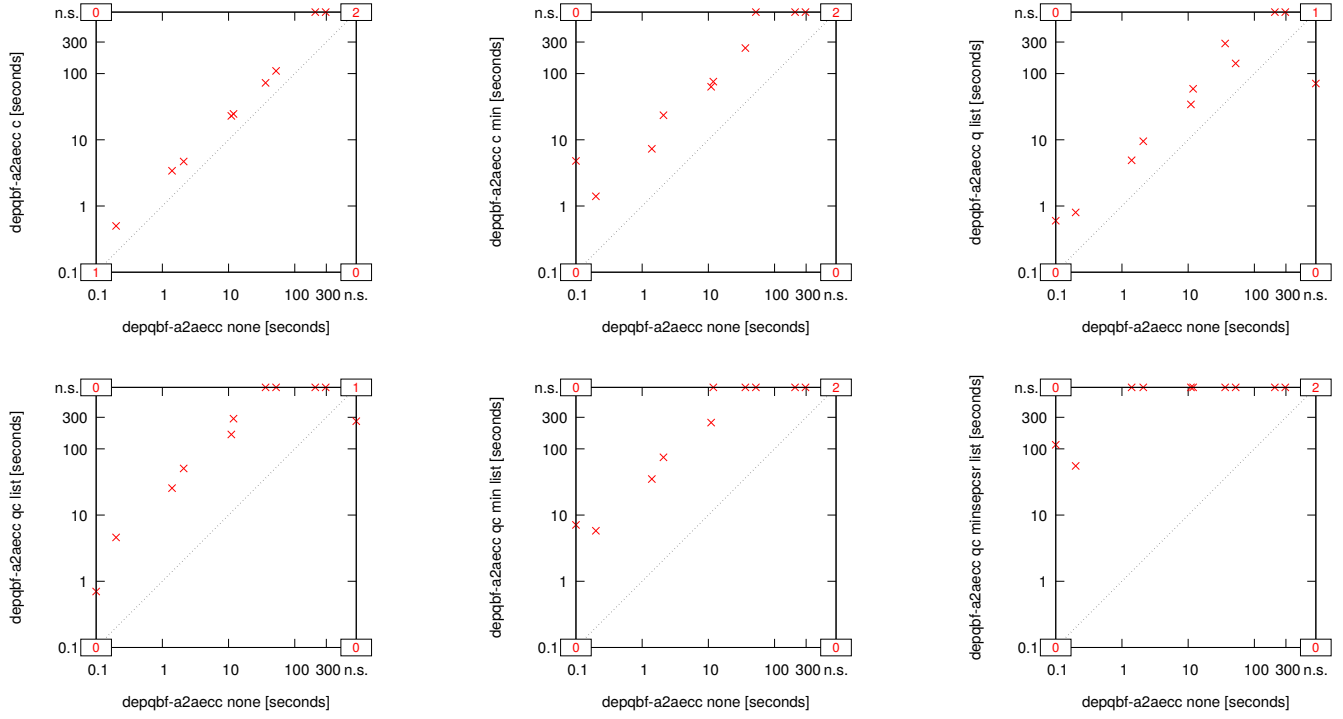


Fig. 559: Suite Ansotegui ($n = 12$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

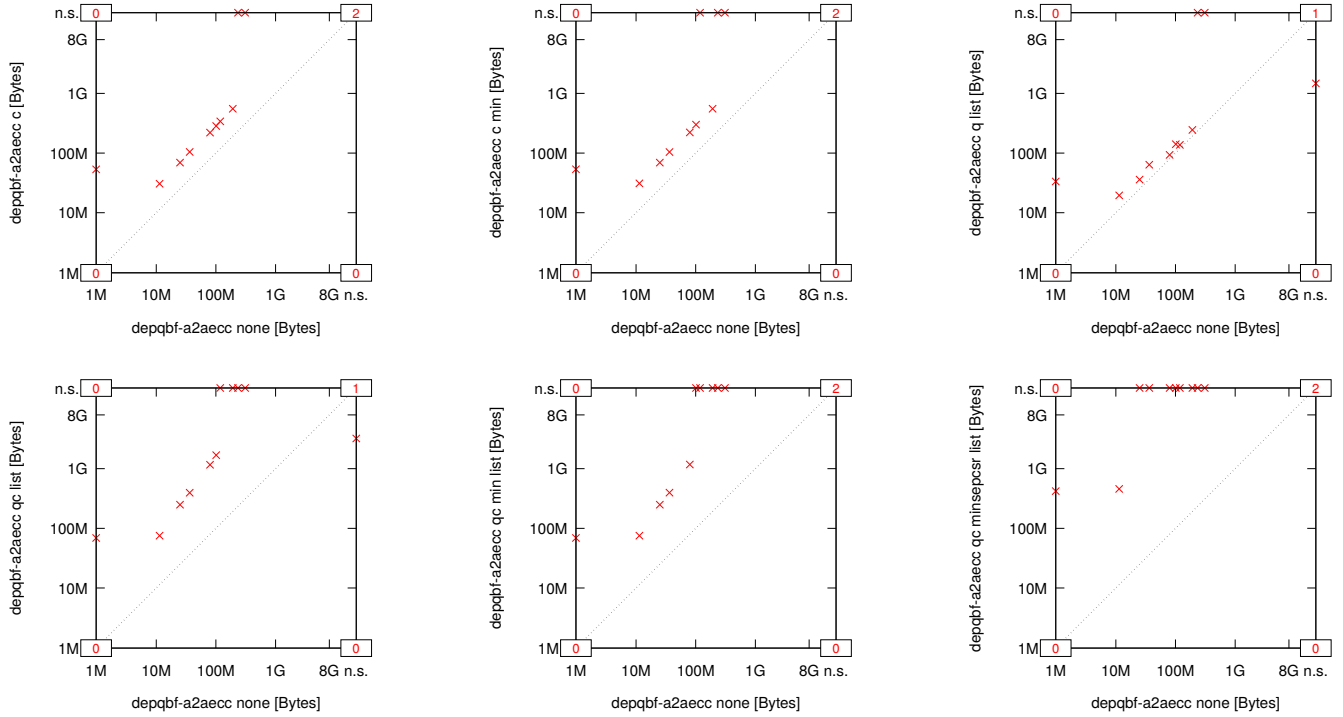


Fig. 560: Suite Ansotegui ($n = 12$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

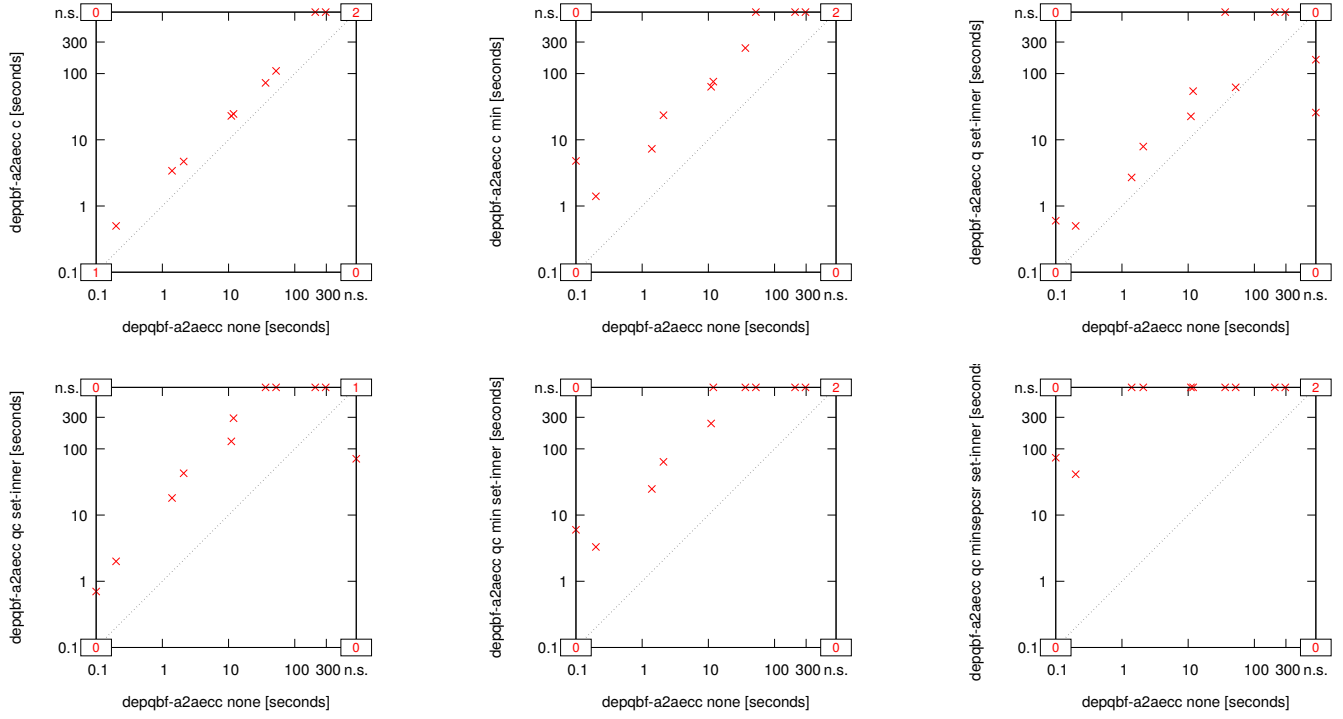


Fig. 561: Suite Ansotegui ($n = 12$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

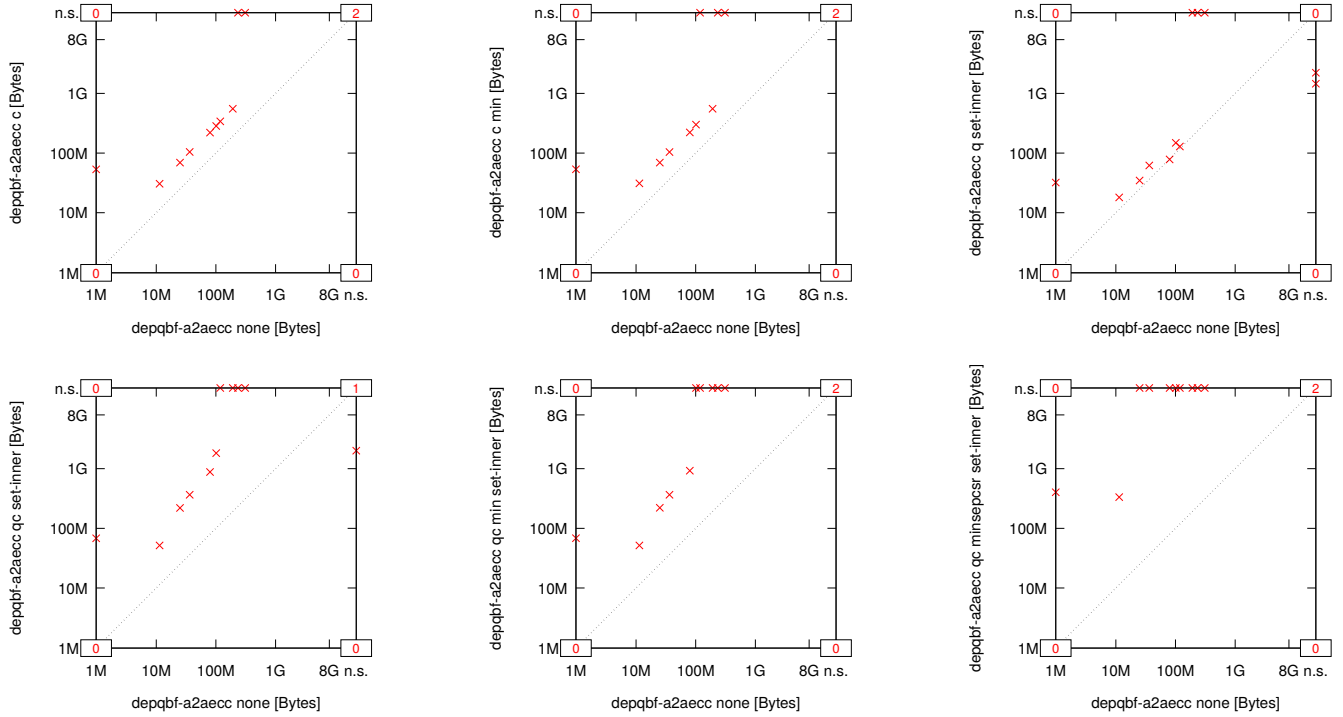


Fig. 562: Suite Ansotegui ($n = 12$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

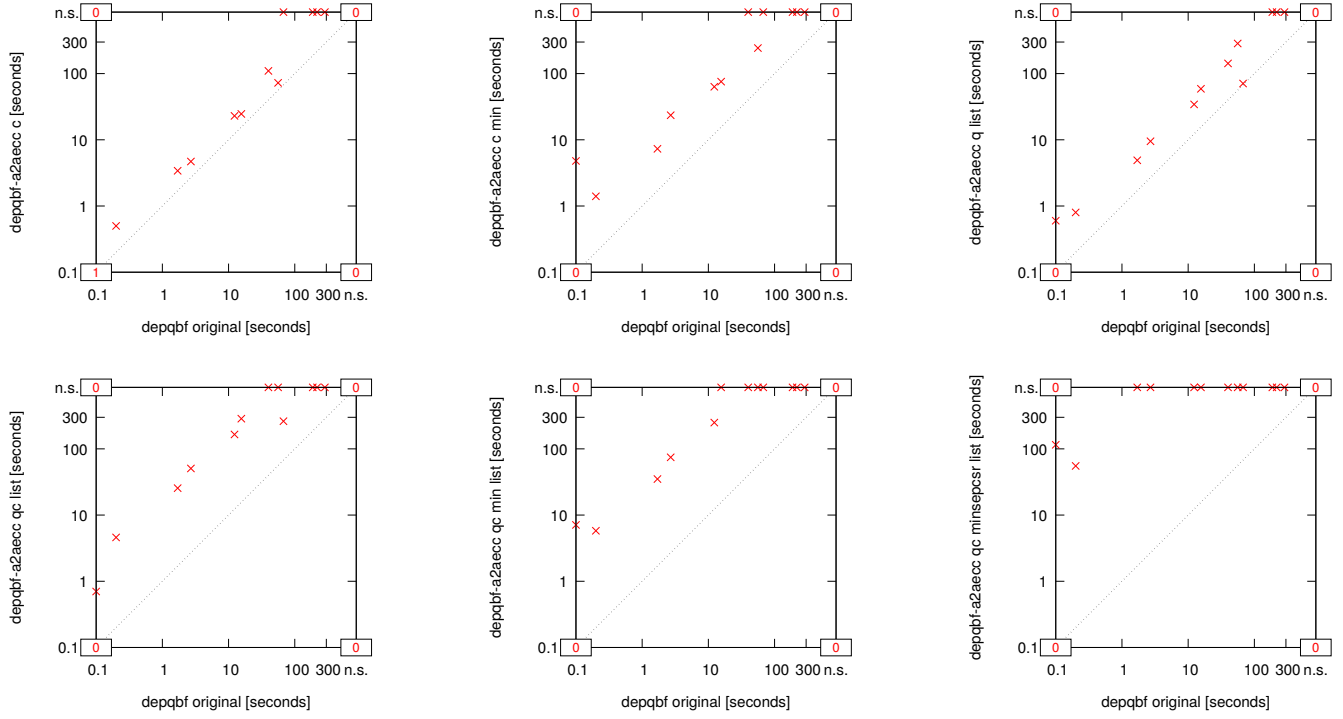


Fig. 563: Suite Ansotegui ($n = 12$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

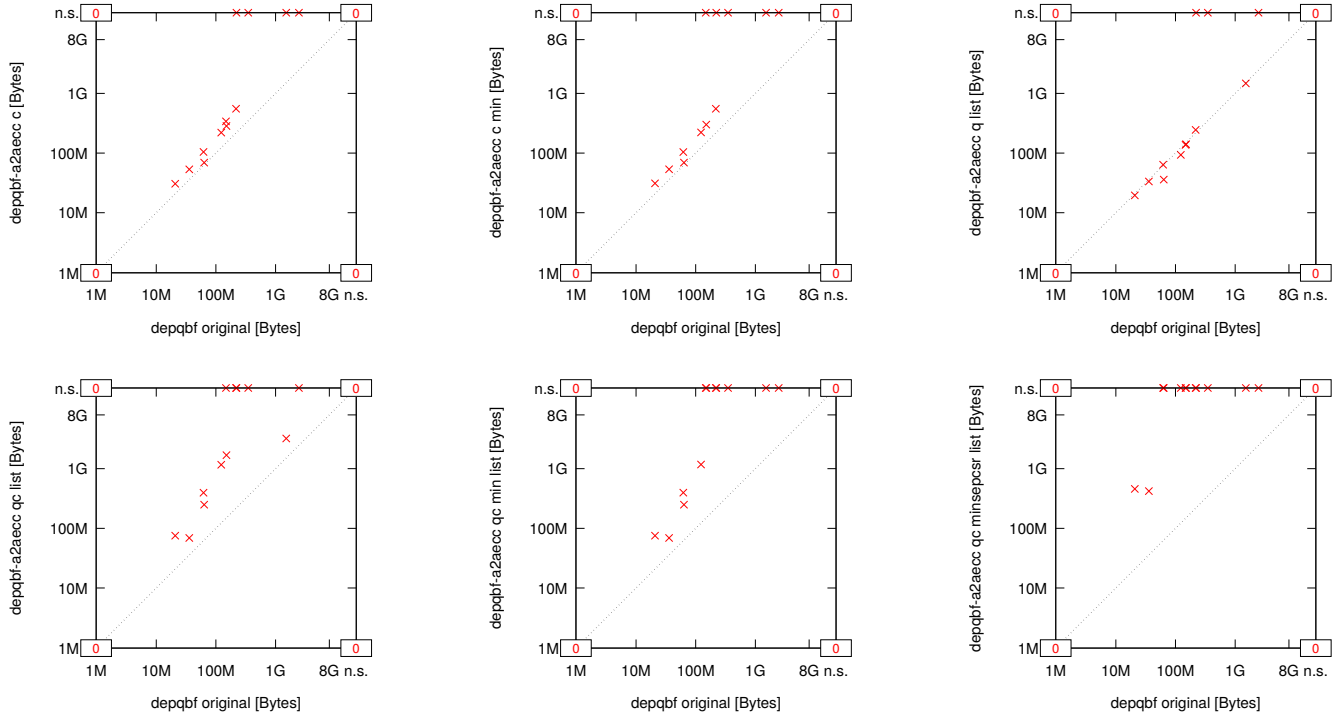


Fig. 564: Suite Ansotegui ($n = 12$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

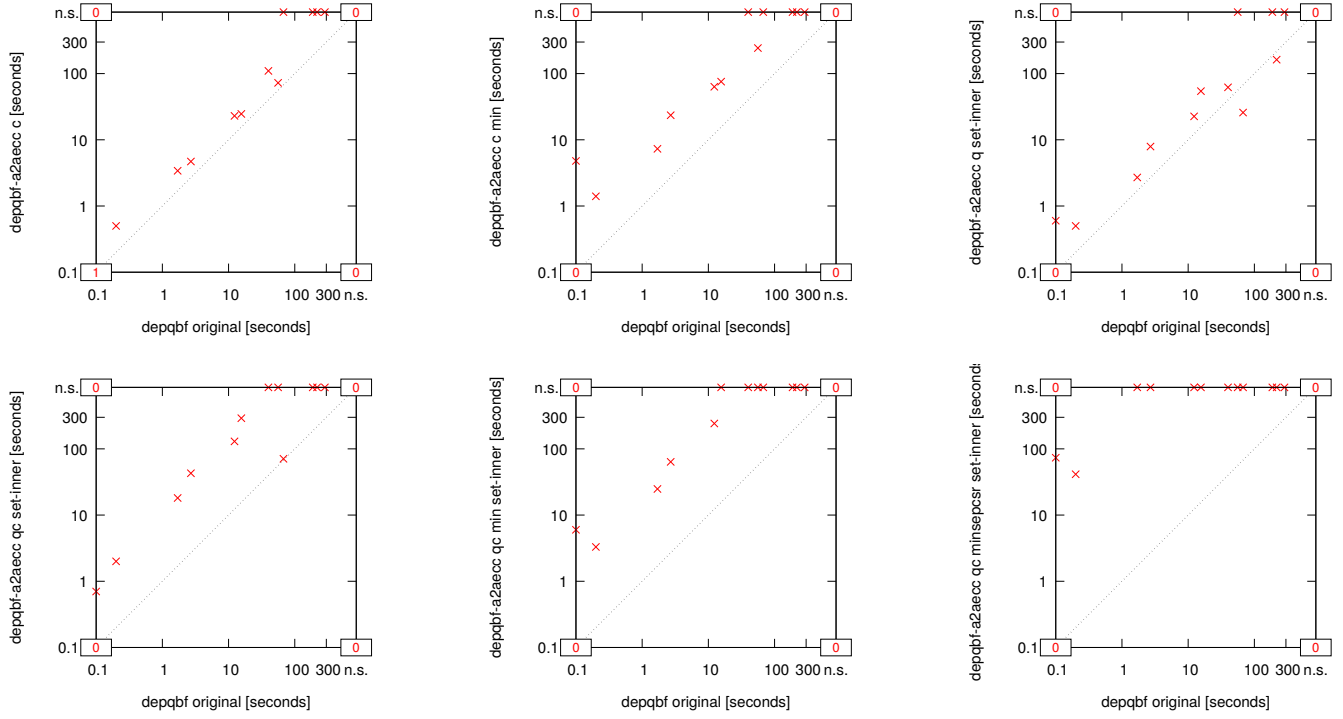


Fig. 565: Suite Ansotegui ($n = 12$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

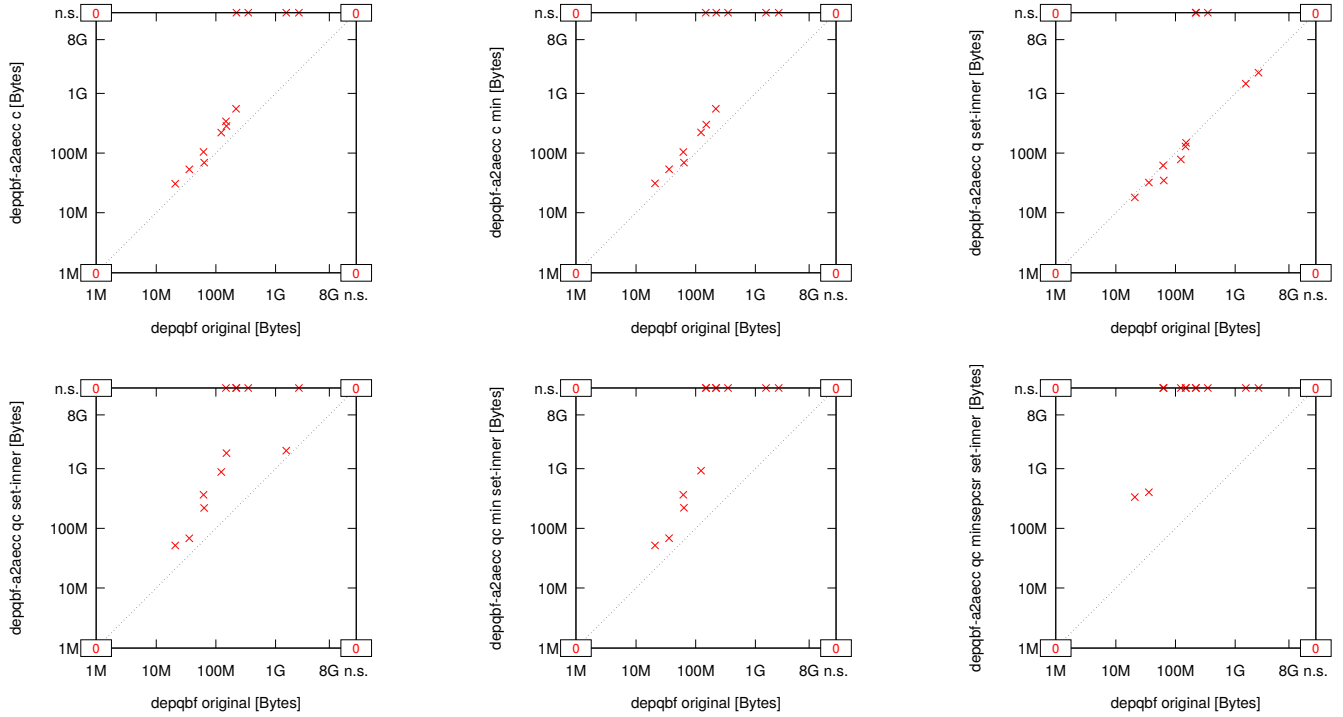


Fig. 566: Suite Ansotegui ($n = 12$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

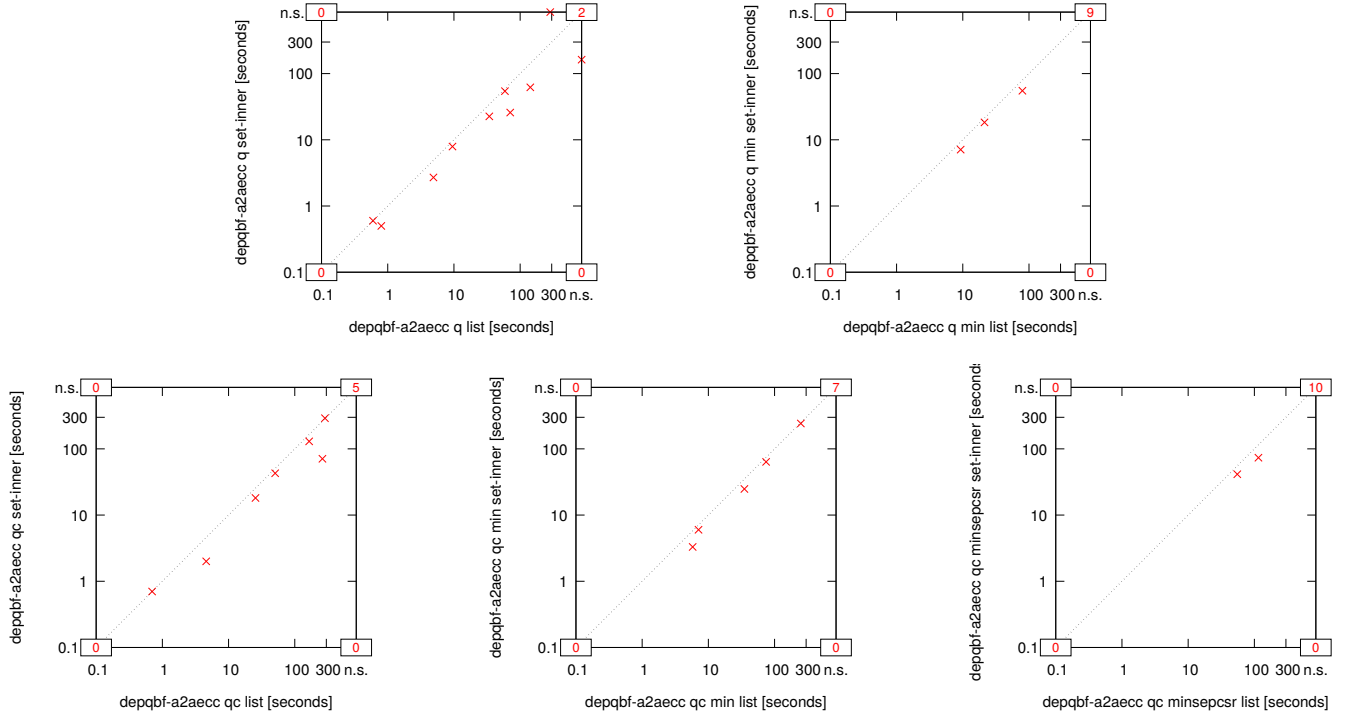


Fig. 567: Suite Ansotegui ($n = 12$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

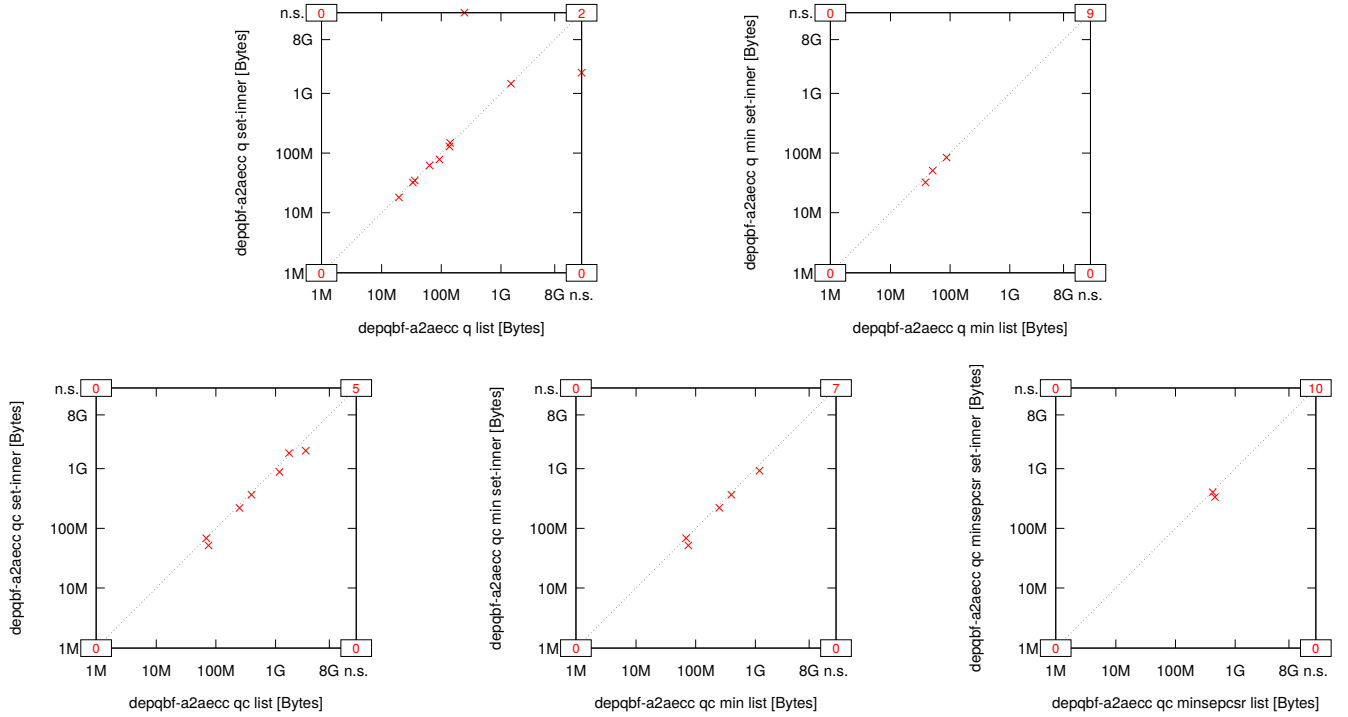


Fig. 568: Suite Ansotegui ($n = 12$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

5) *Ayari* ($n = 48$):

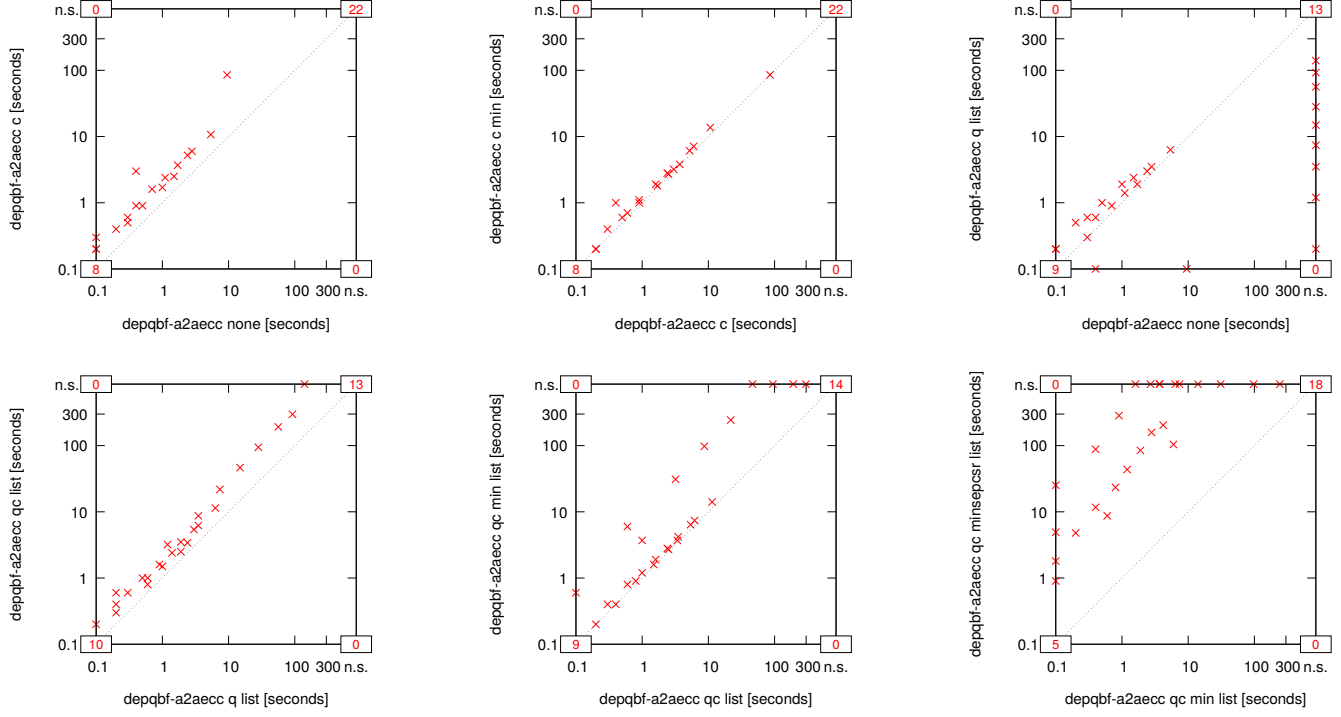


Fig. 569: Suite Ayari ($n = 48$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

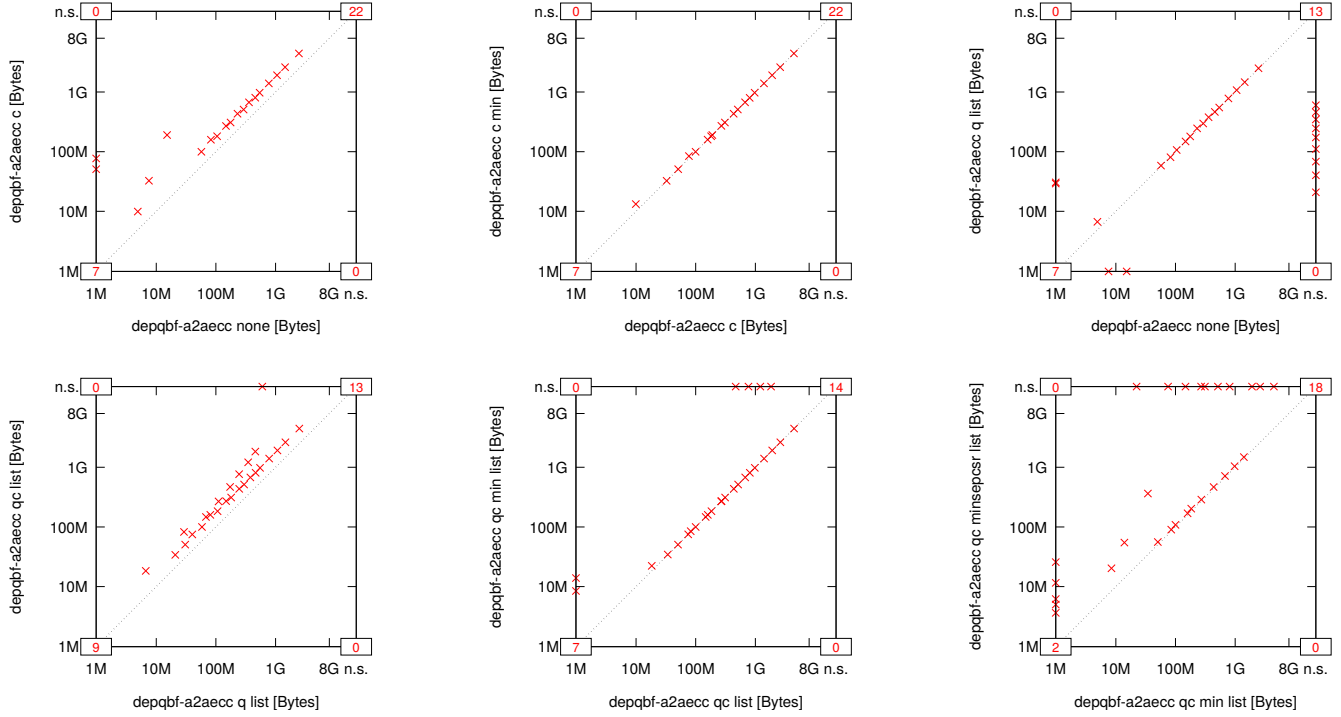


Fig. 570: Suite Ayari ($n = 48$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

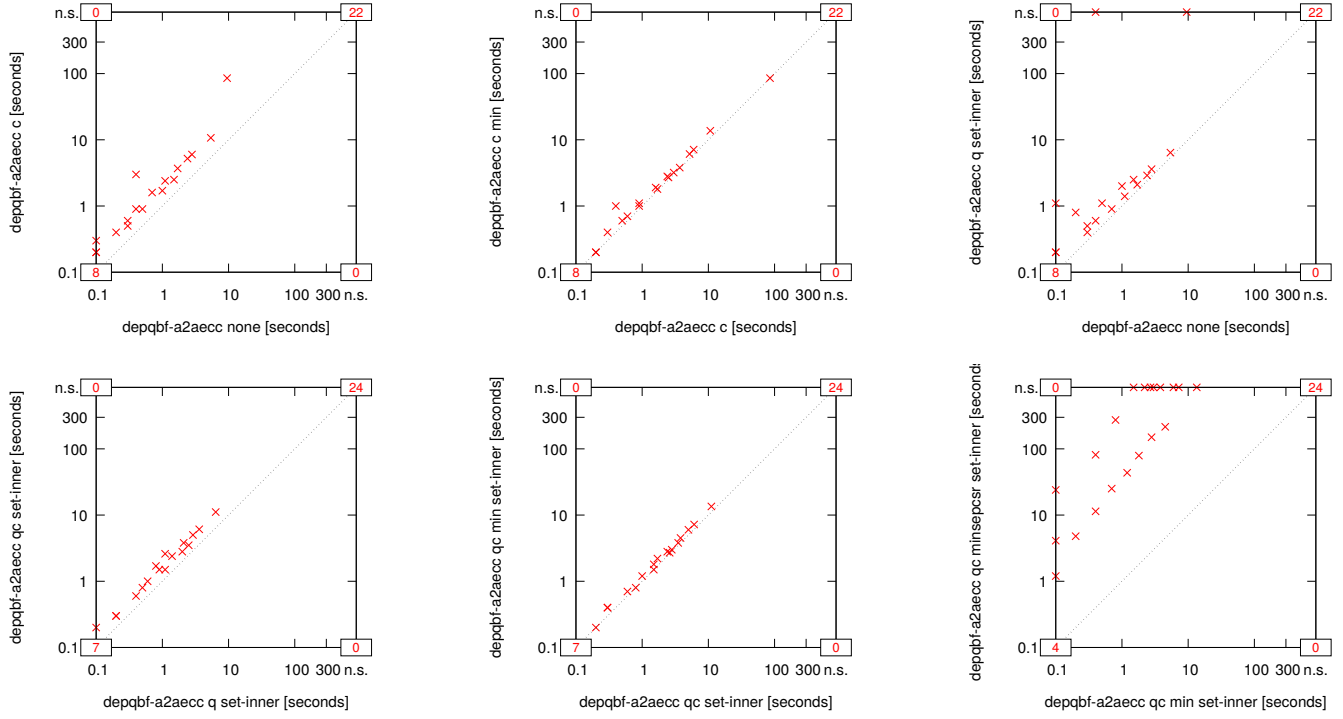


Fig. 571: Suite Ayari ($n = 48$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

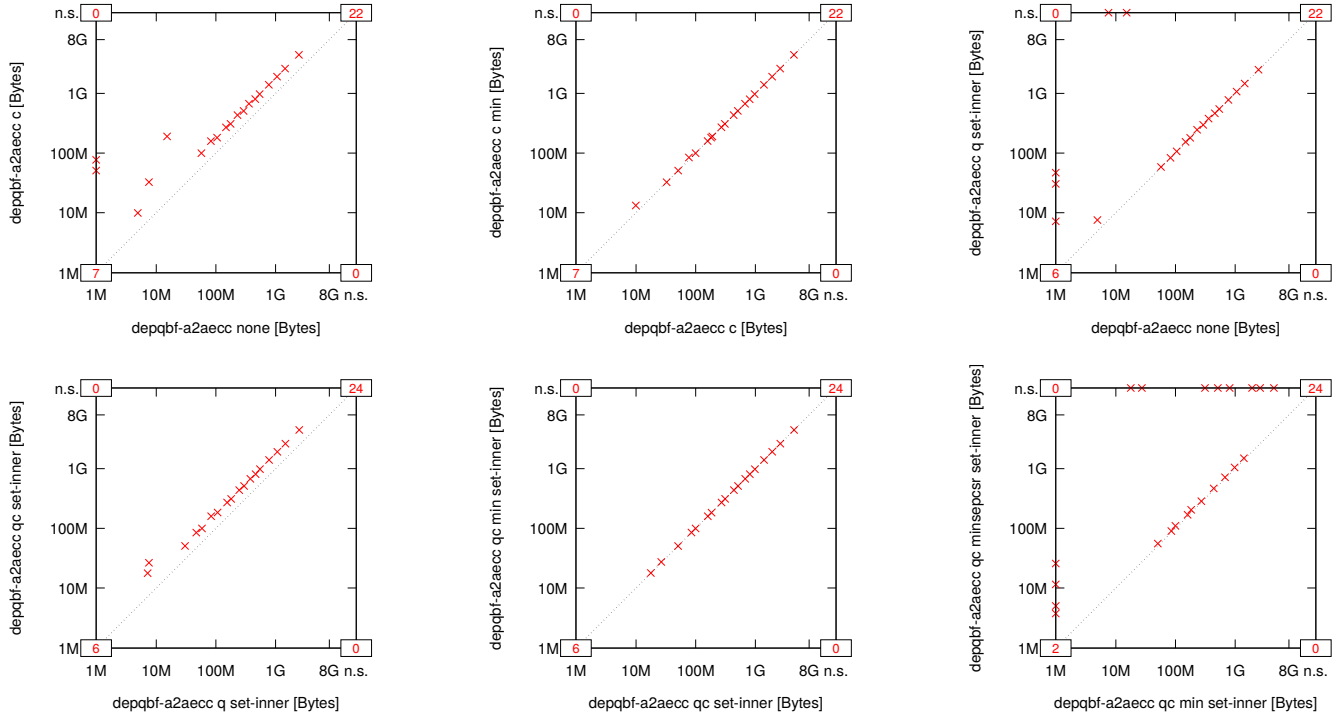


Fig. 572: Suite Ayari ($n = 48$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

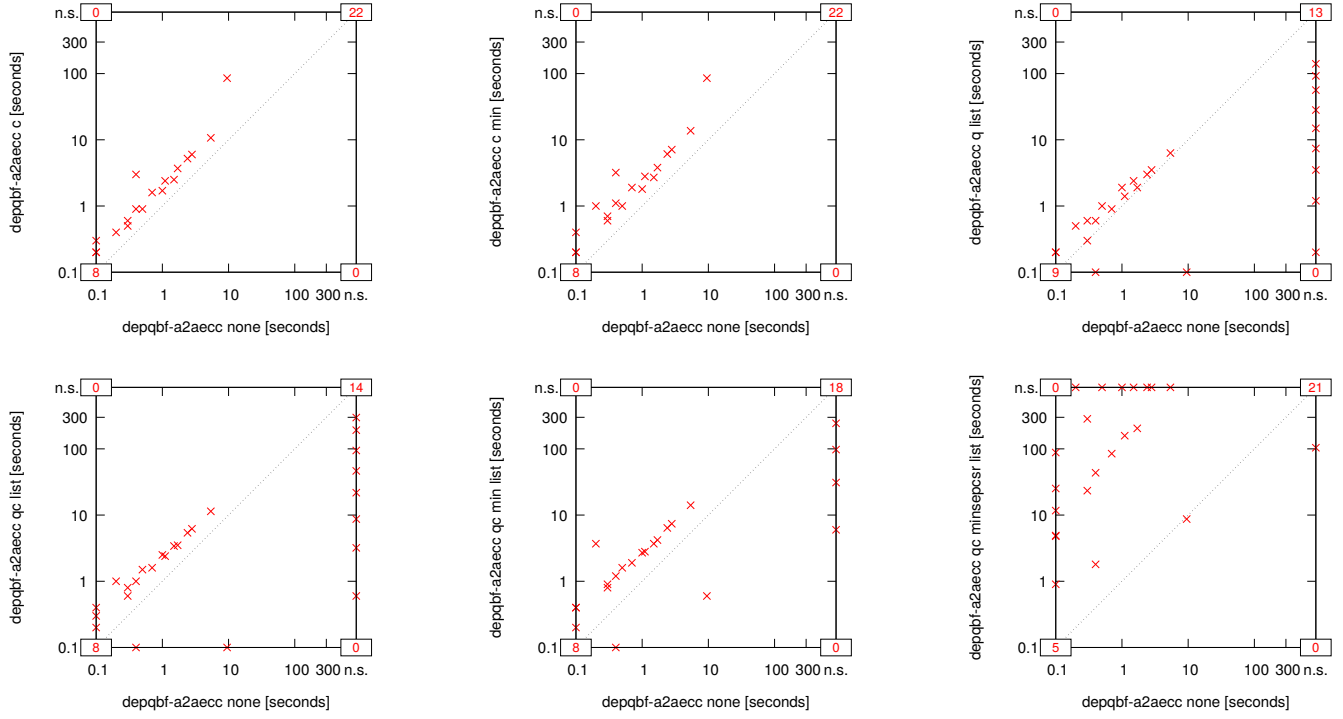


Fig. 573: Suite Ayari ($n = 48$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

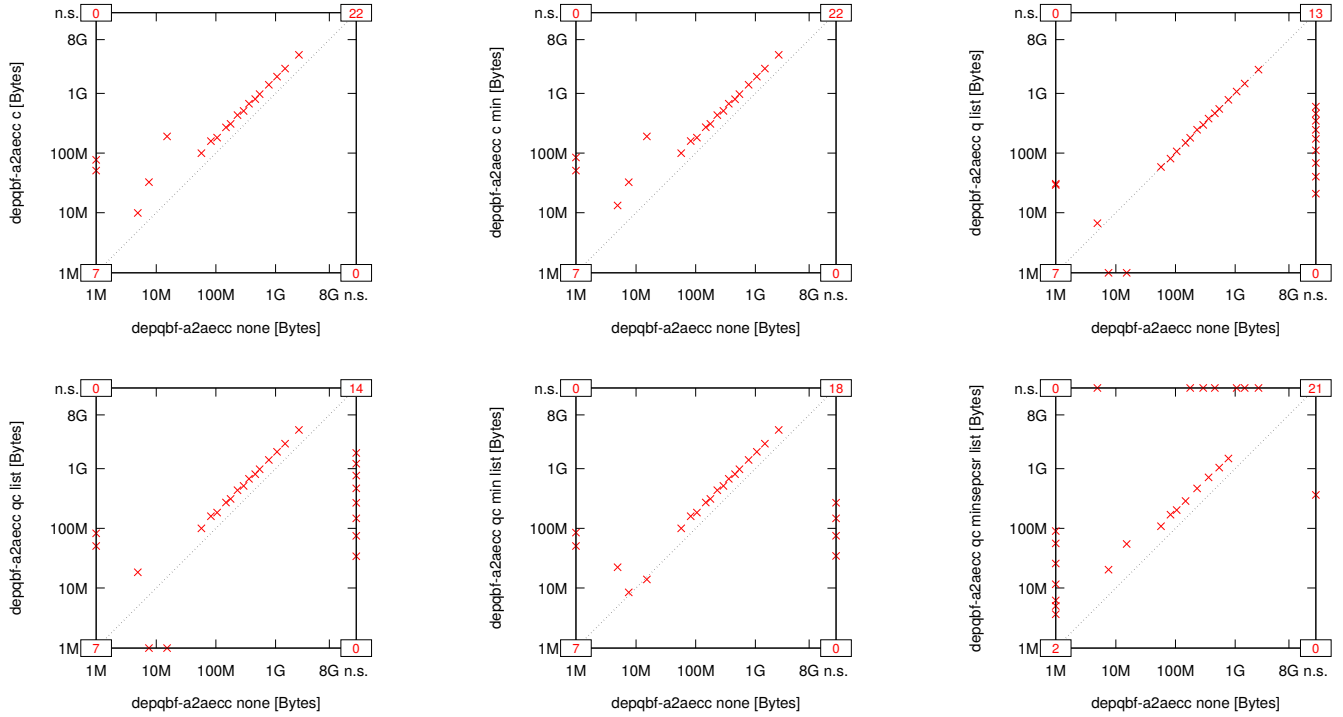


Fig. 574: Suite Ayari ($n = 48$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

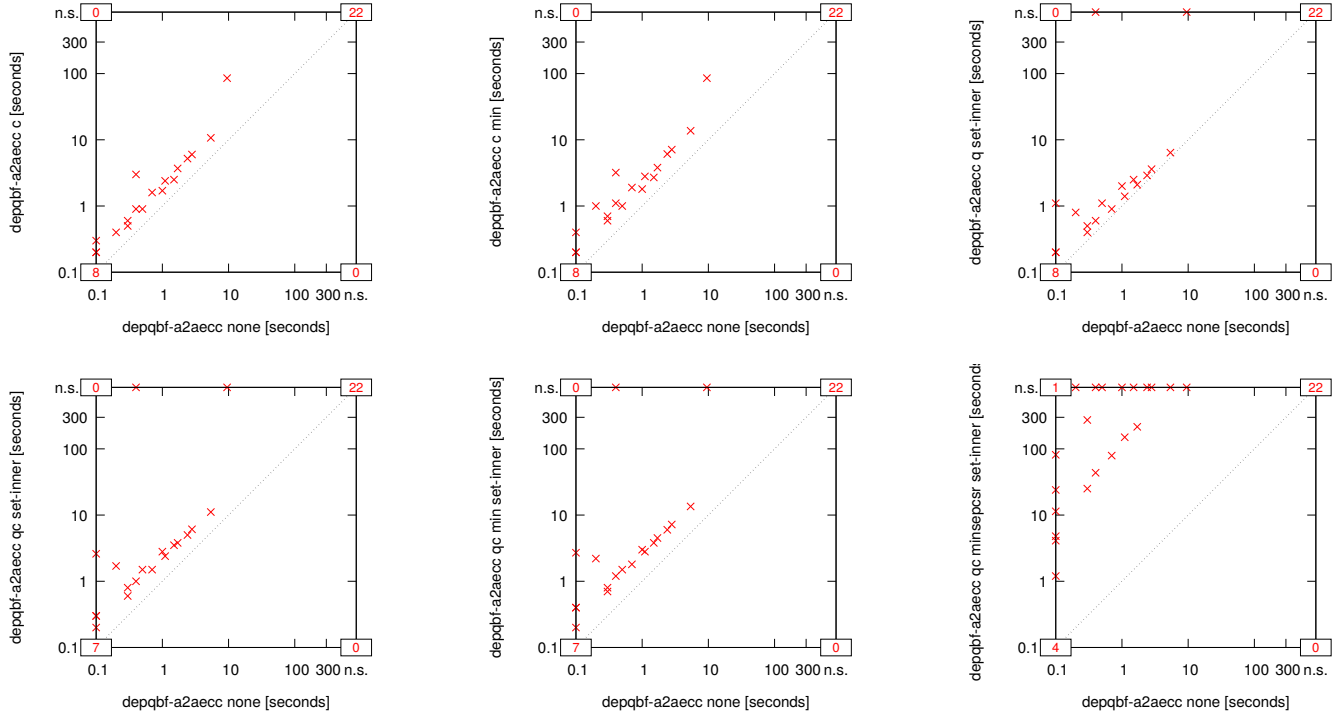


Fig. 575: Suite Ayari ($n = 48$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

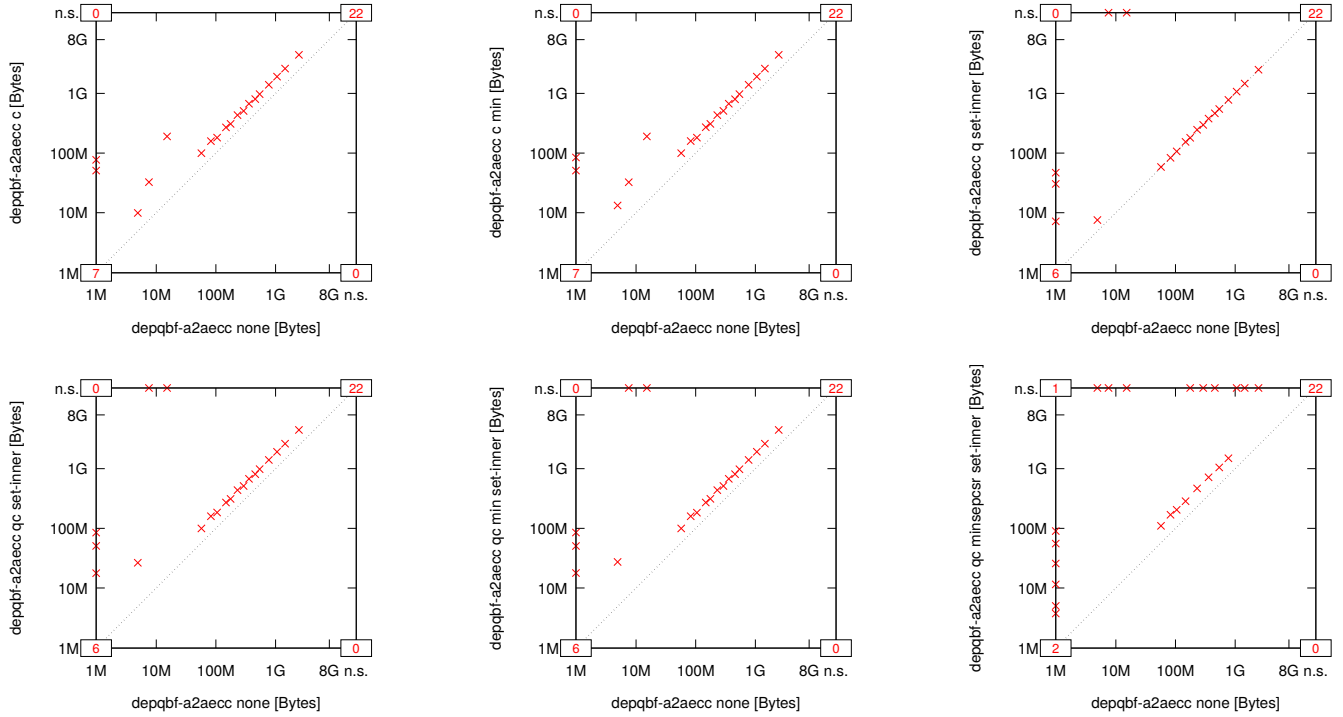


Fig. 576: Suite Ayari ($n = 48$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

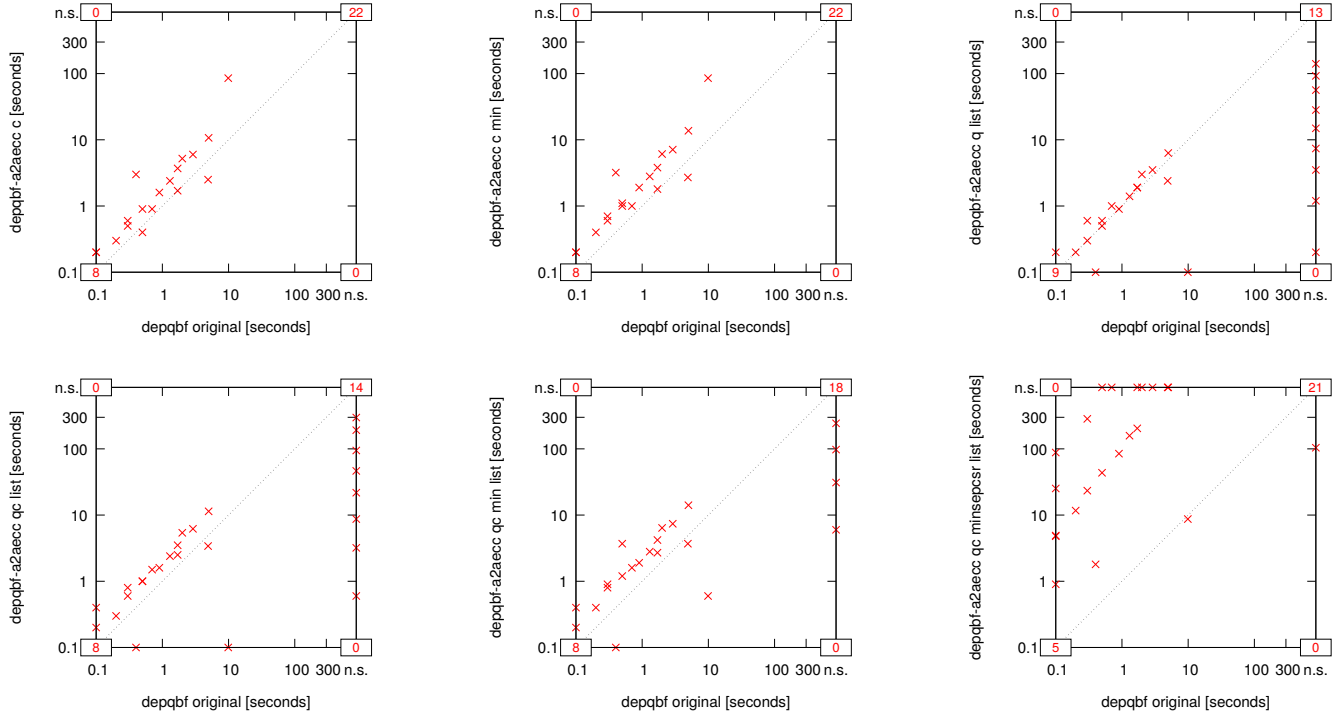


Fig. 577: Suite Ayari ($n = 48$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

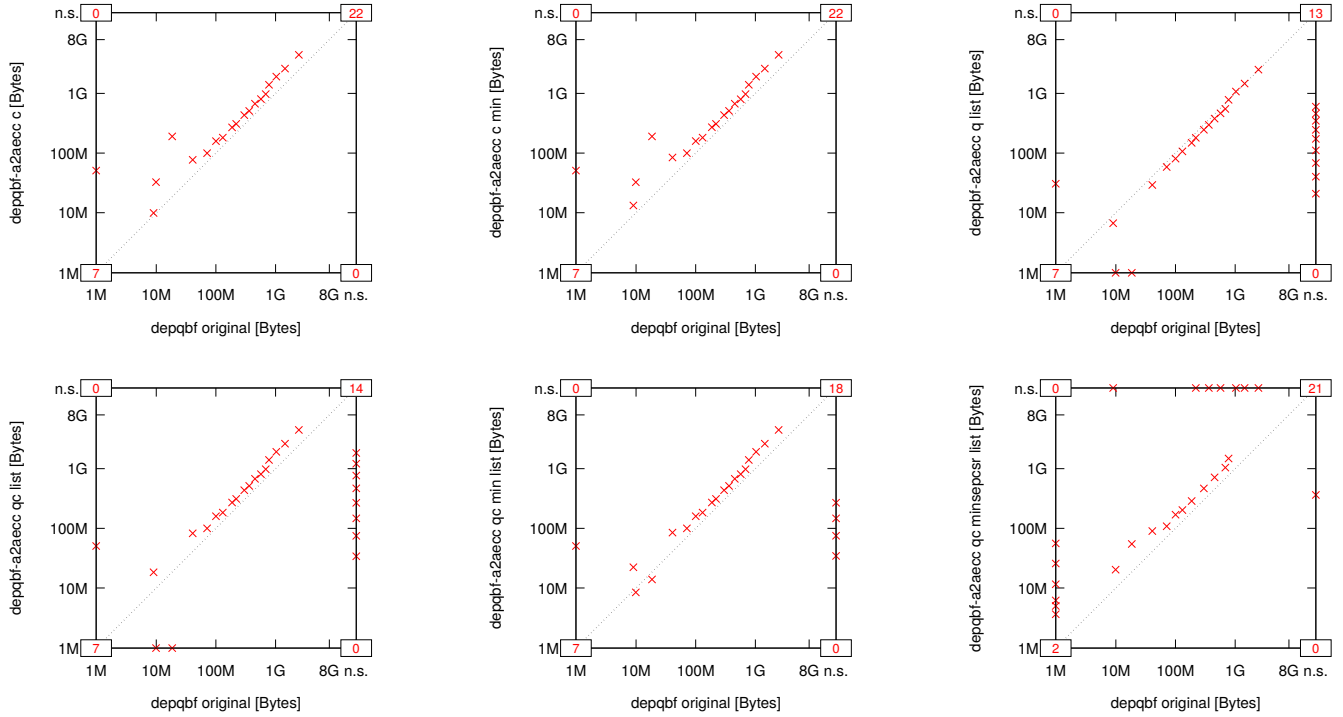


Fig. 578: Suite Ayari ($n = 48$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

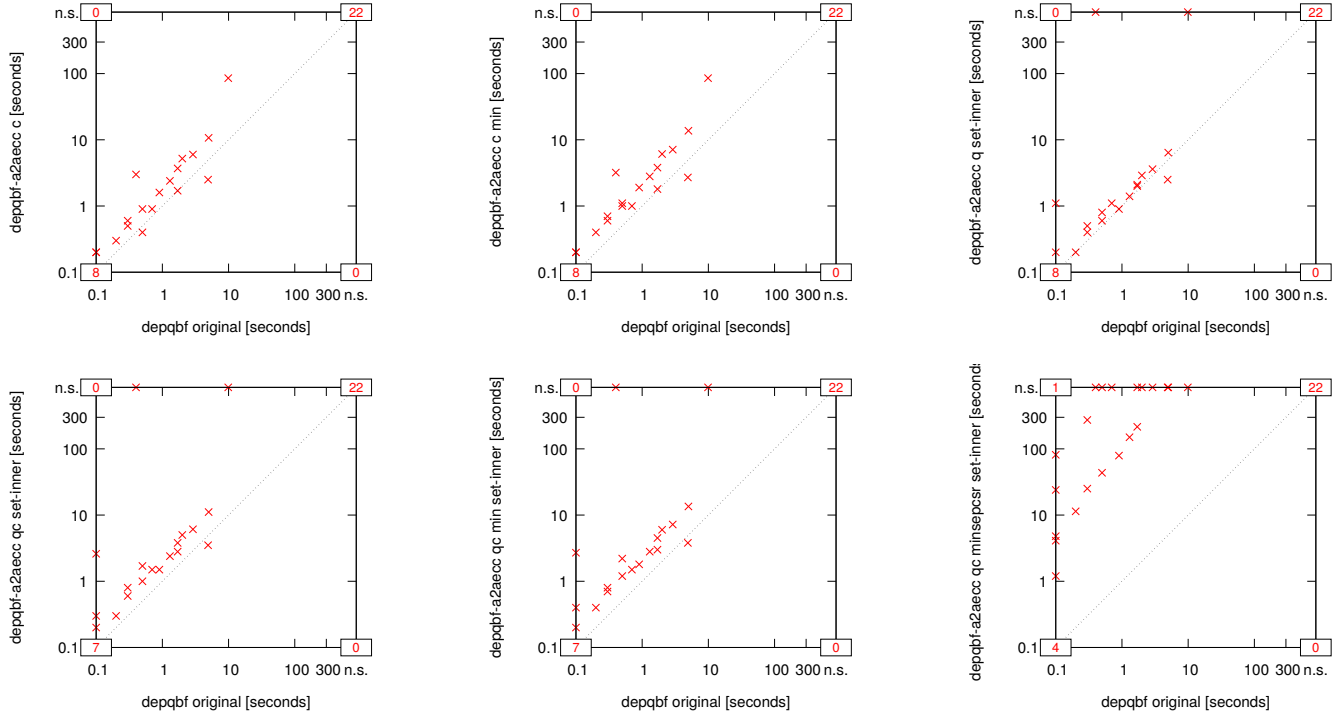


Fig. 579: Suite Ayari ($n = 48$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

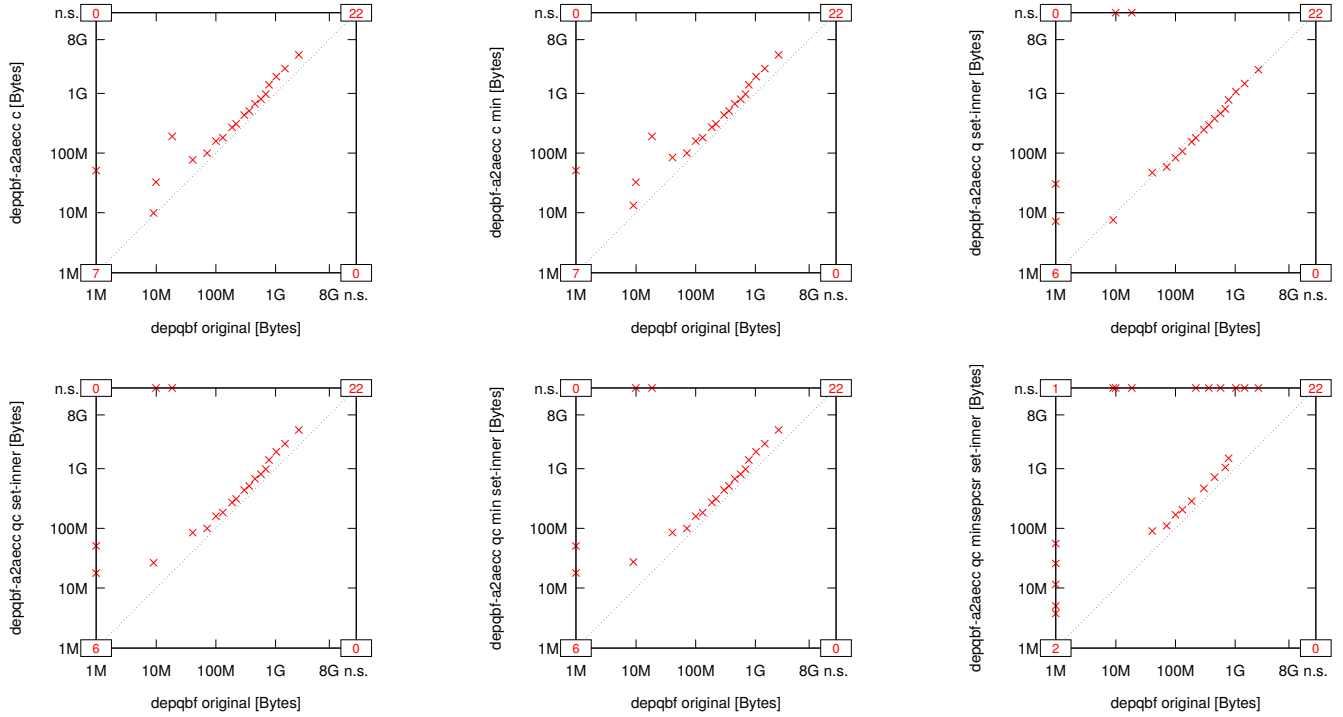


Fig. 580: Suite Ayari ($n = 48$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

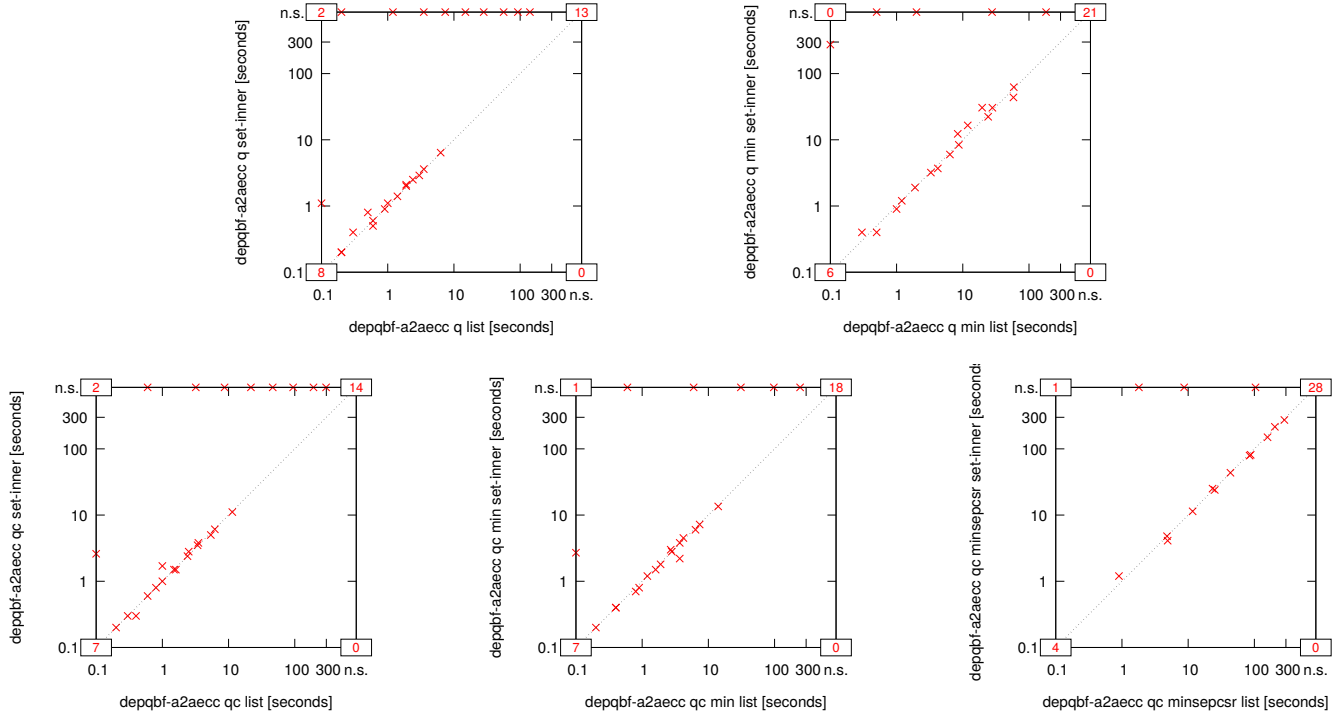


Fig. 581: Suite Ayari ($n = 48$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

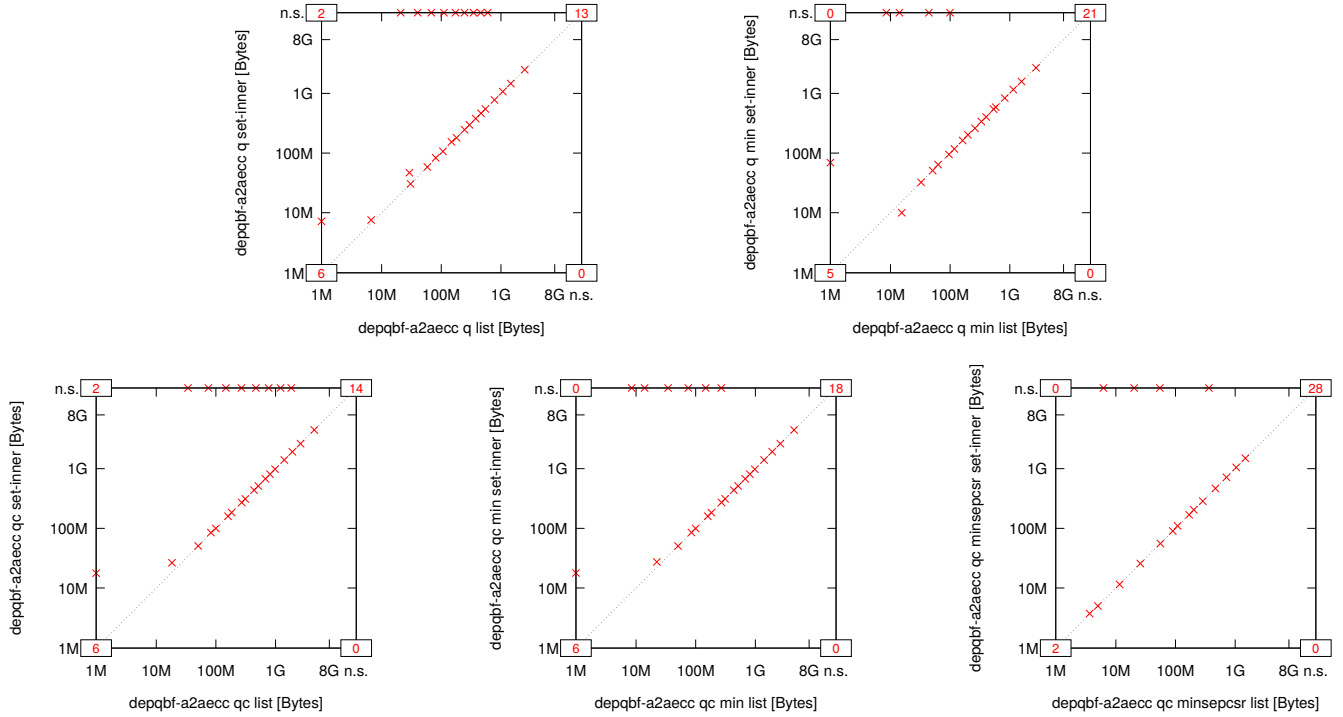


Fig. 582: Suite Ayari ($n = 48$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

6) Basler ($n = 75$):

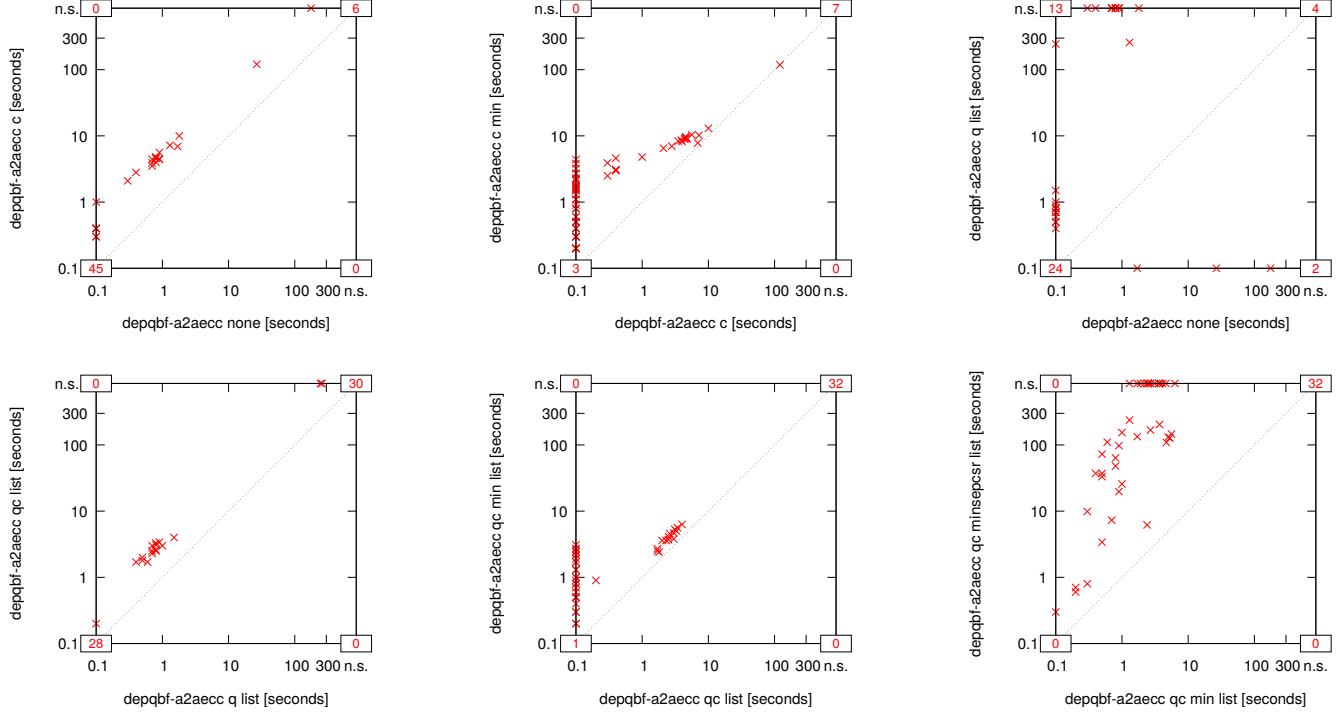


Fig. 583: Suite Basler ($n = 75$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

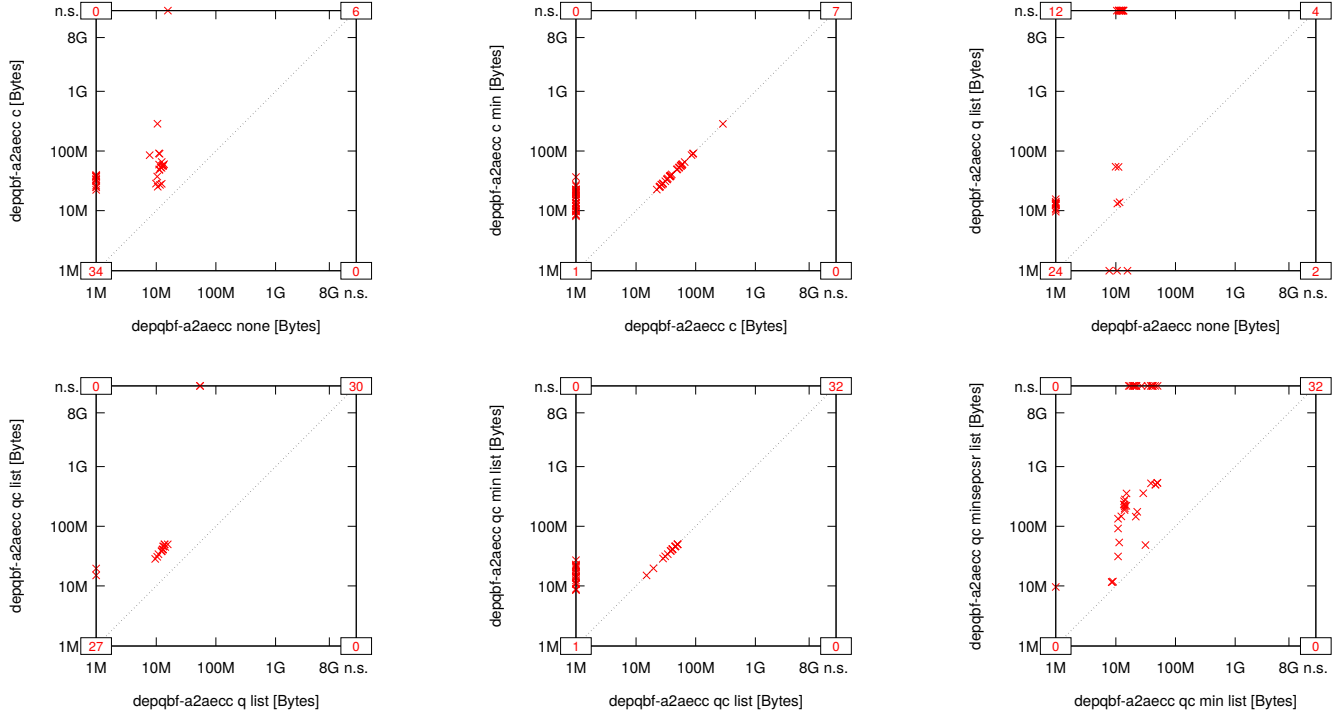


Fig. 584: Suite Basler ($n = 75$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

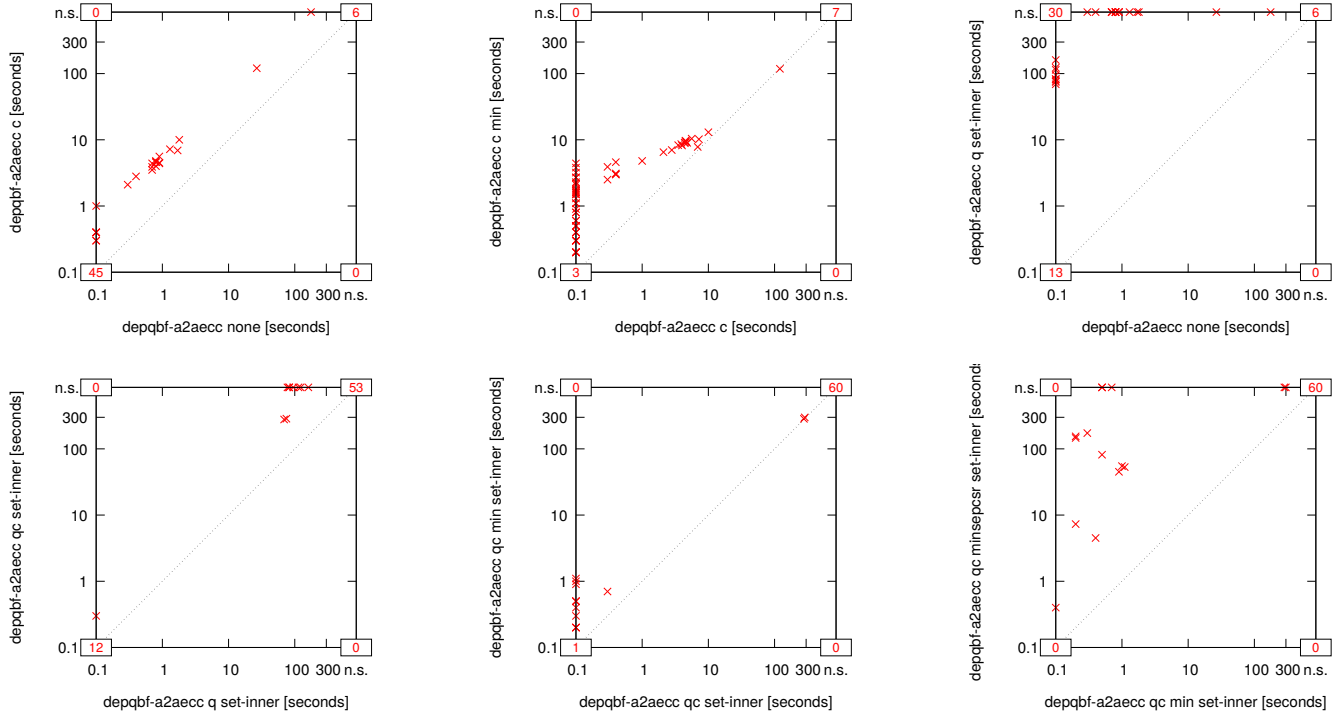


Fig. 585: Suite Basler ($n = 75$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

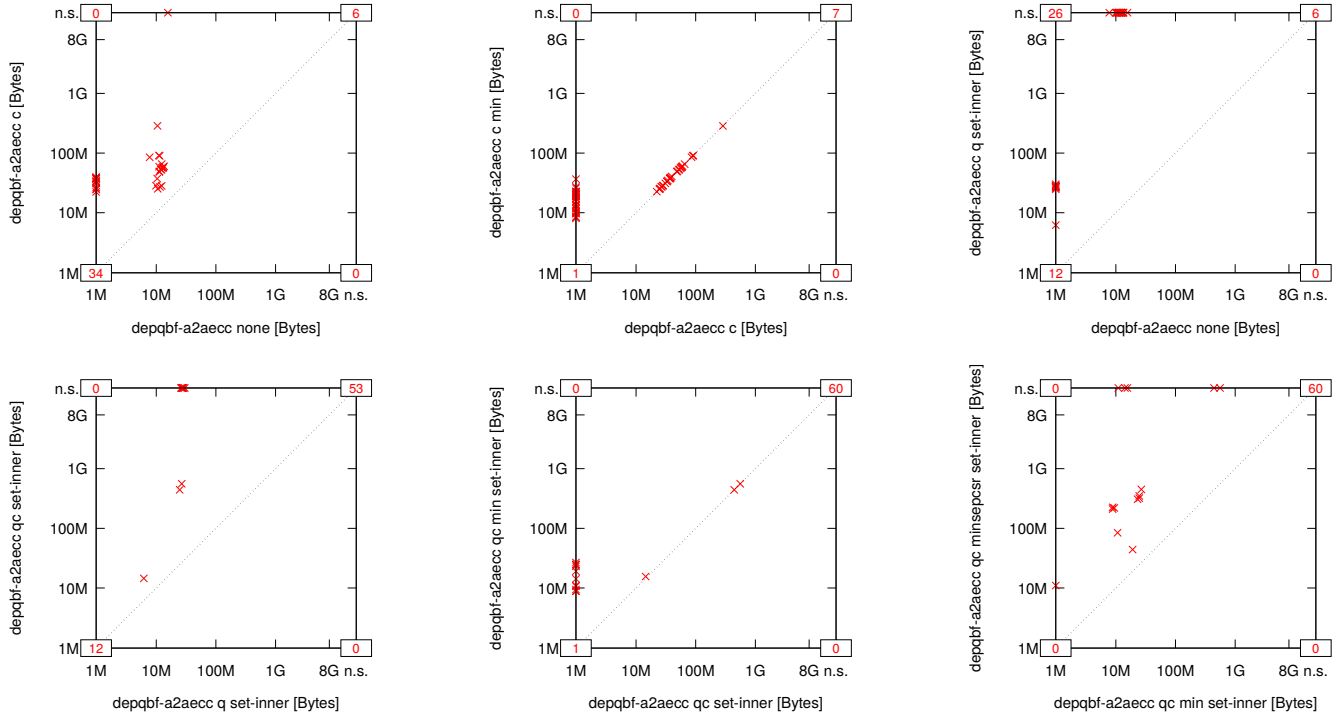


Fig. 586: Suite Basler ($n = 75$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

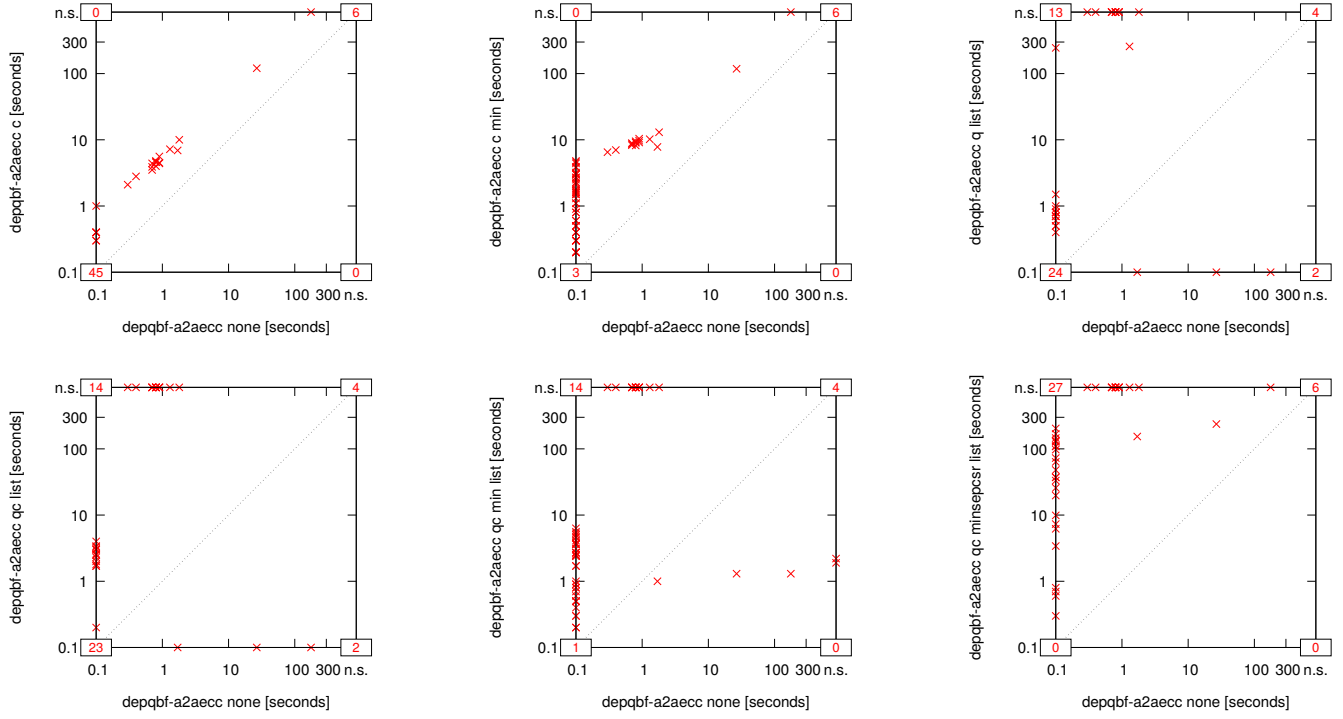


Fig. 587: Suite Basler ($n = 75$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

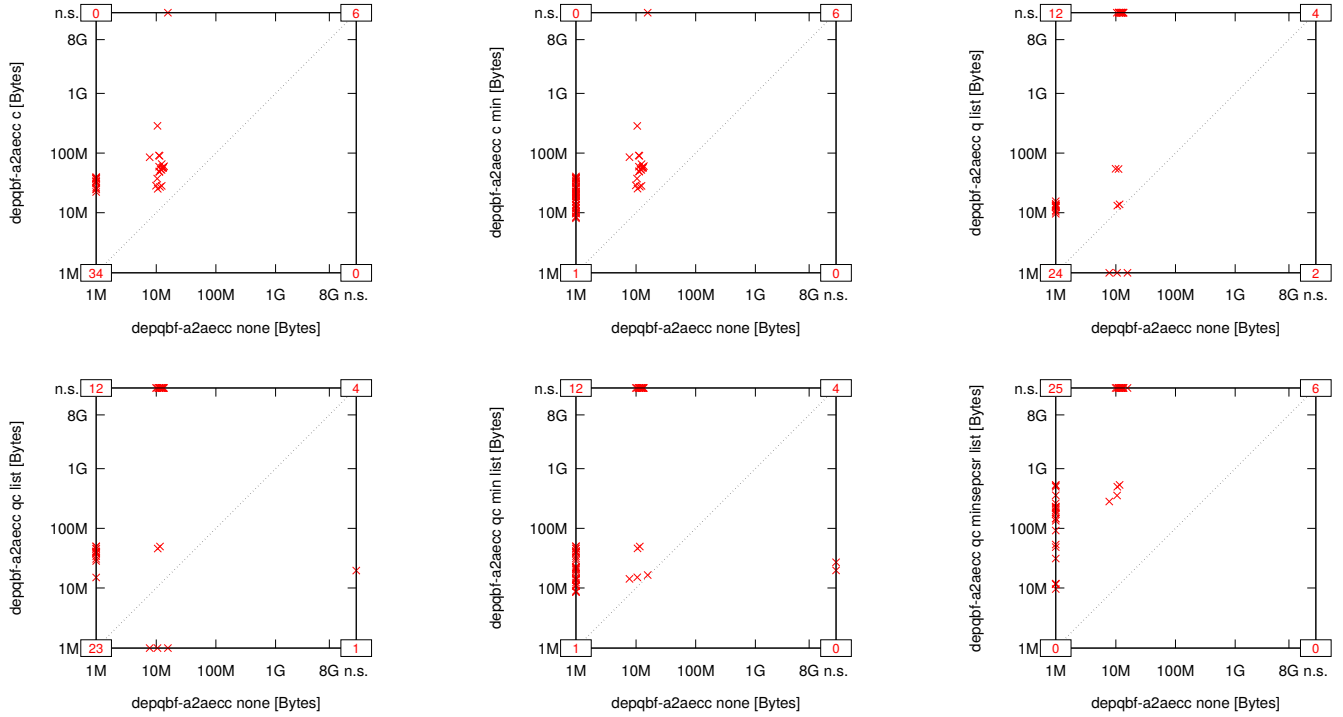


Fig. 588: Suite Basler ($n = 75$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

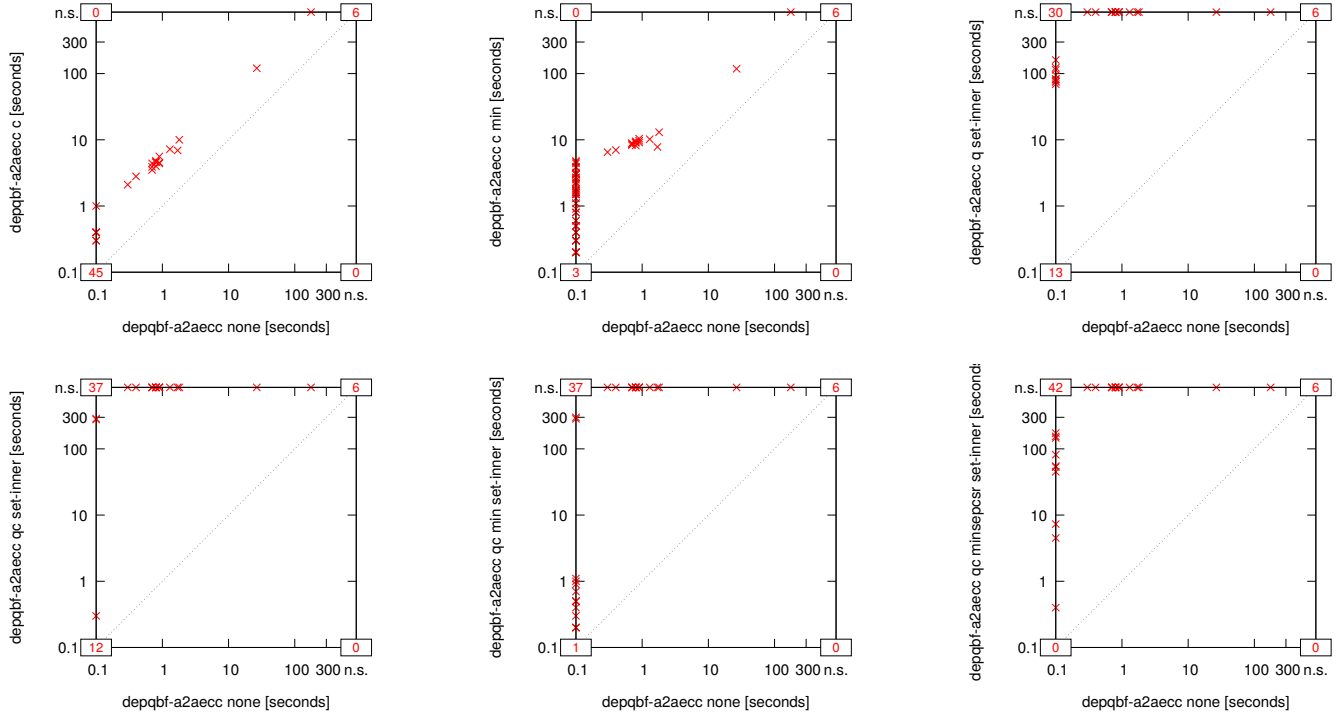


Fig. 589: Suite Basler ($n = 75$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

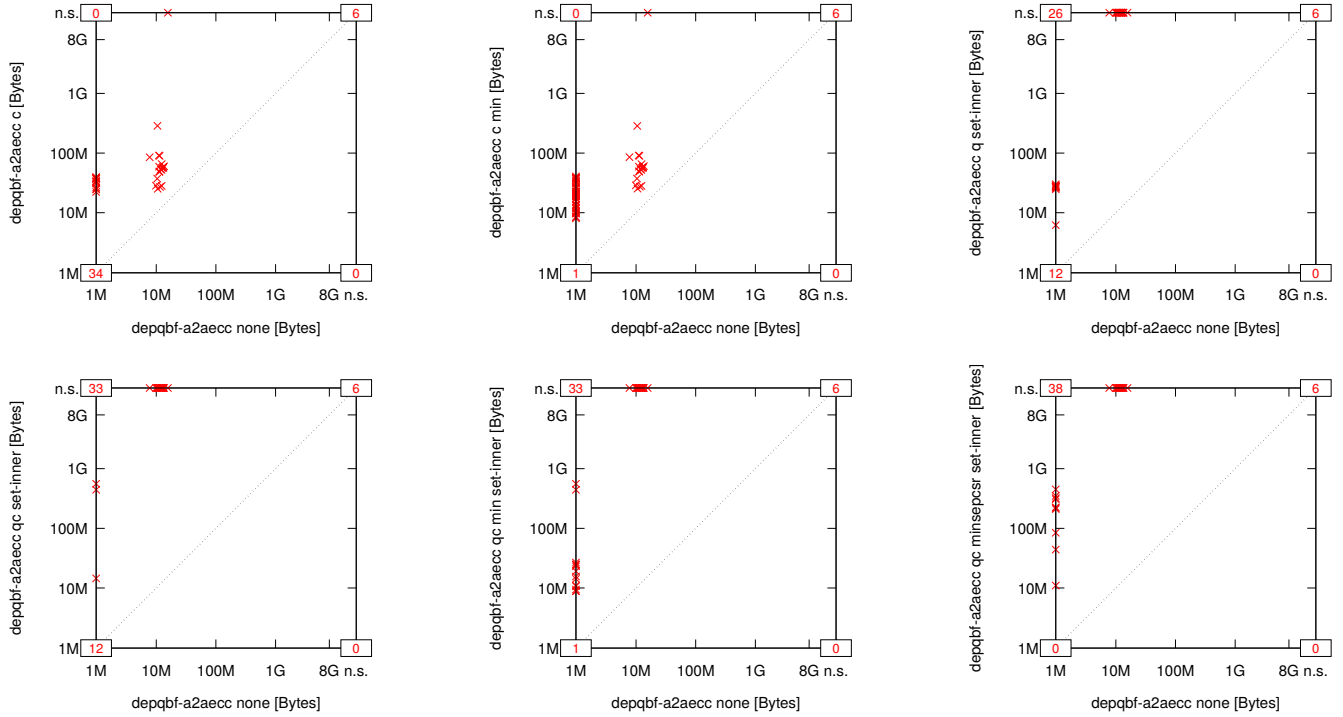


Fig. 590: Suite Basler ($n = 75$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

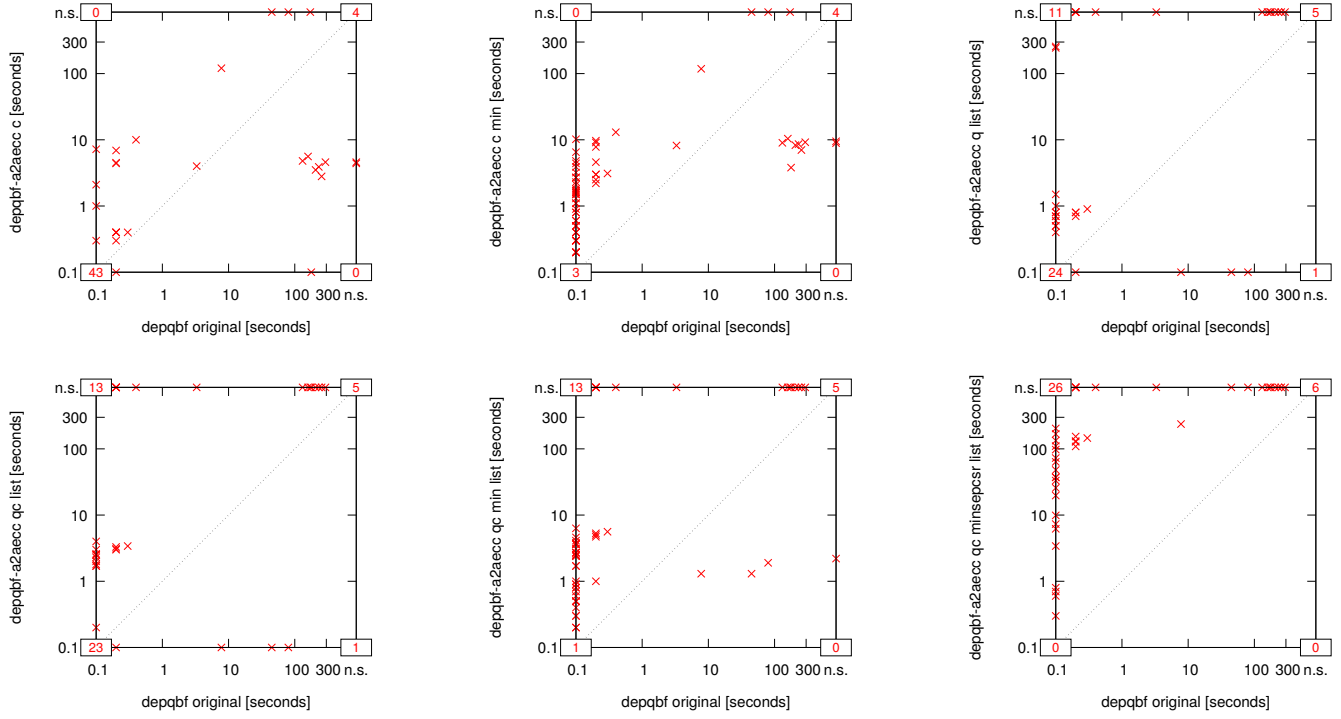


Fig. 591: Suite Basler ($n = 75$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

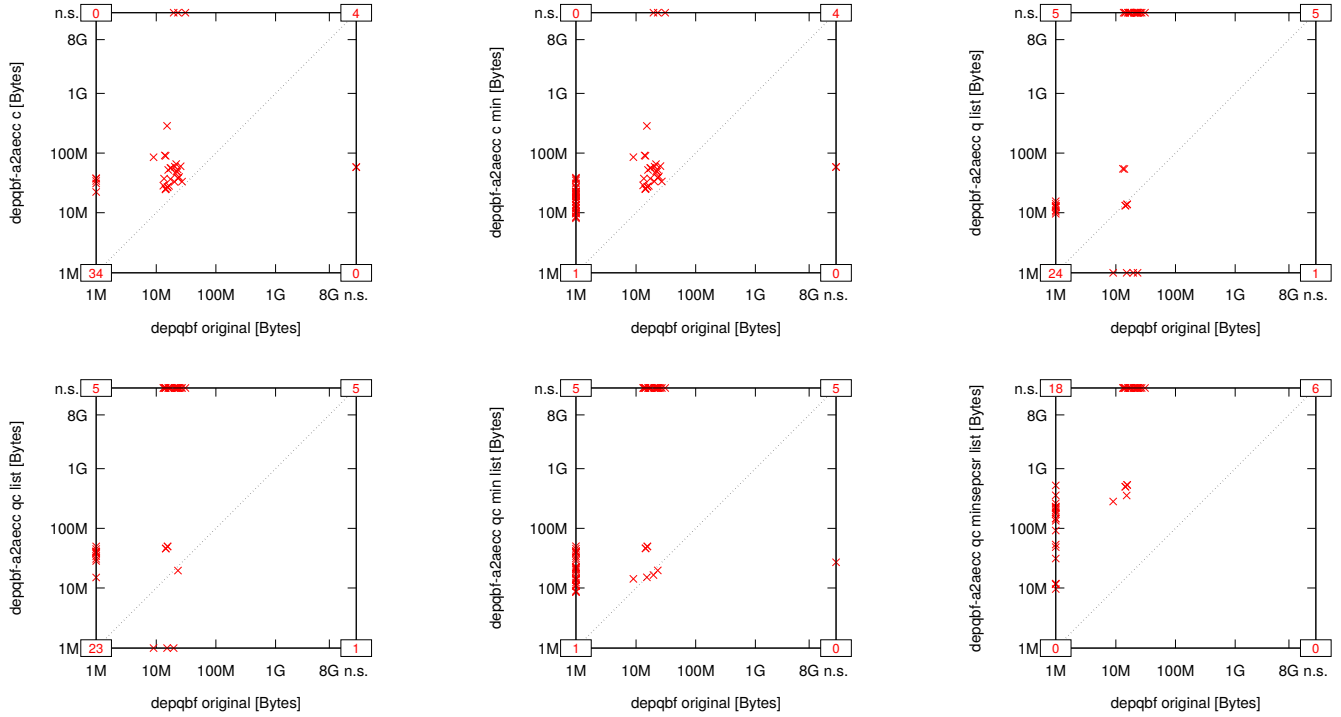


Fig. 592: Suite Basler ($n = 75$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

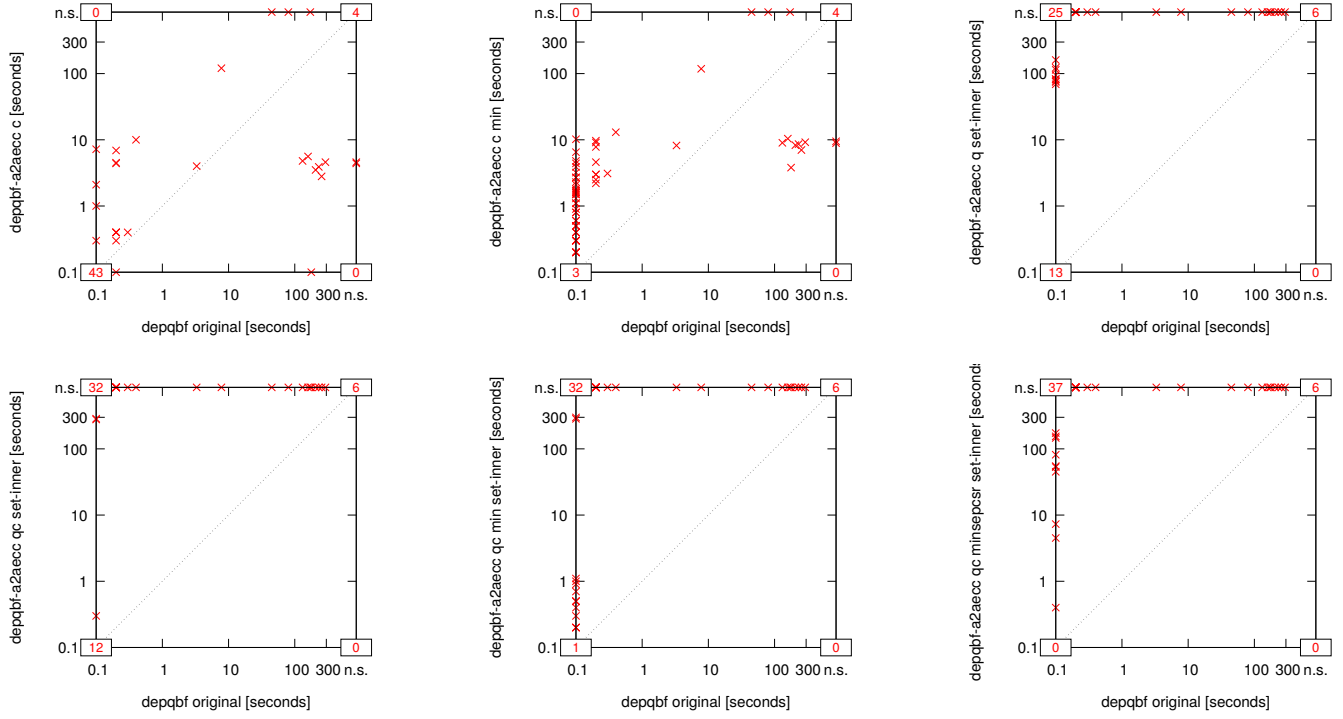


Fig. 593: Suite Basler ($n = 75$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

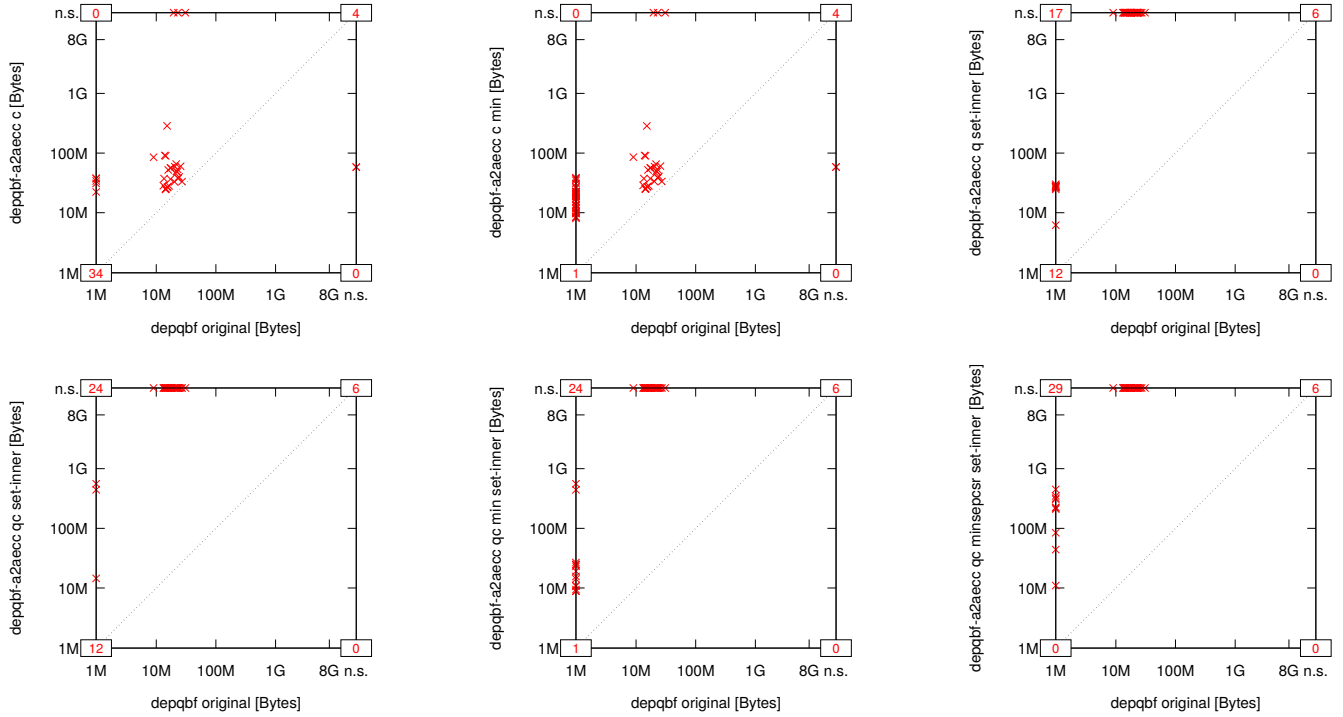


Fig. 594: Suite Basler ($n = 75$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

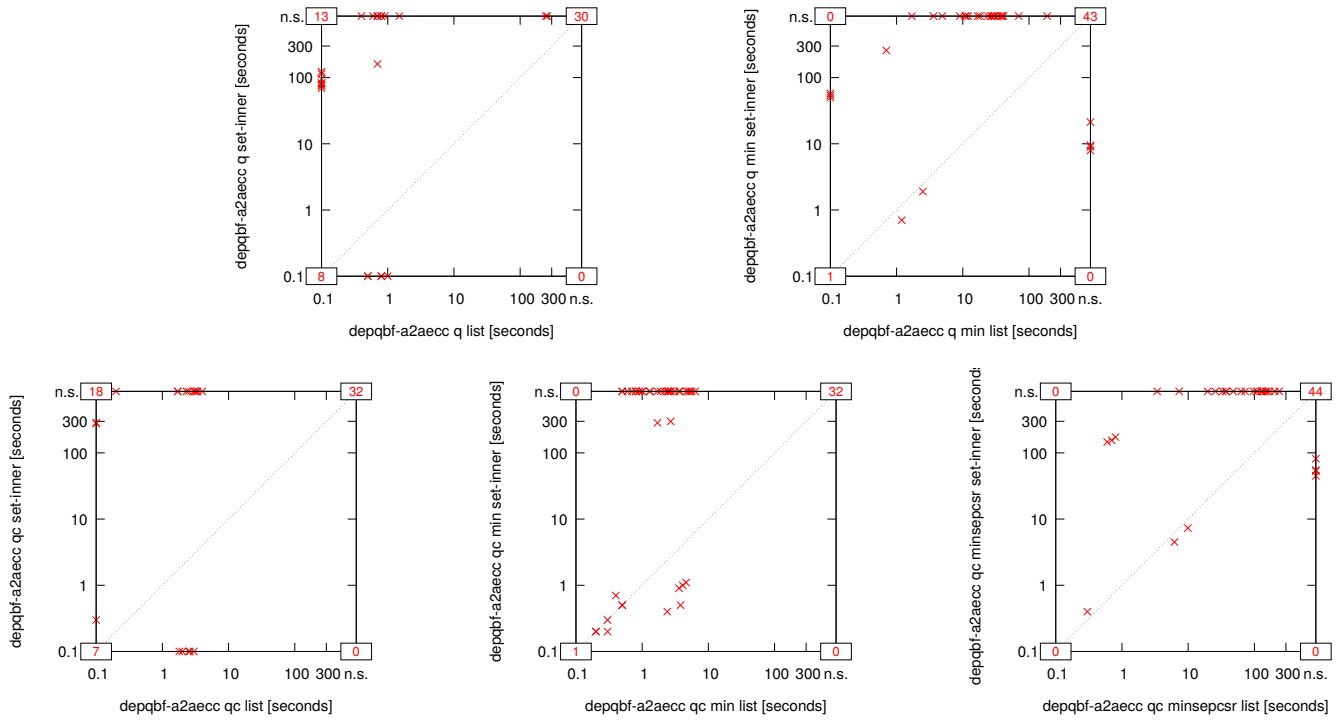


Fig. 595: Suite Basler ($n = 75$): Extracting and minimizing unsatisfiable cores in `DepQBF-a2aecc` with set-inner versus list semantics (run time in seconds).

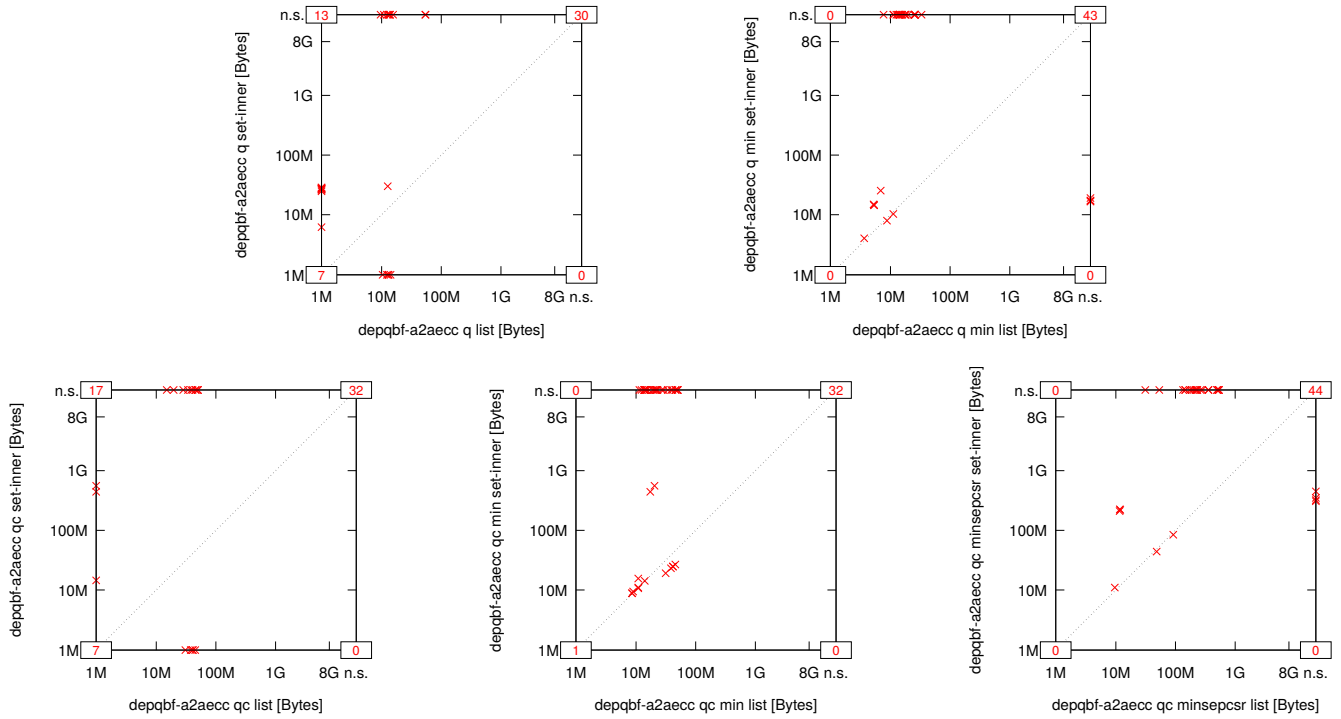


Fig. 596: Suite Basler ($n = 75$): Extracting and minimizing unsatisfiable cores in `DepQBF-a2aecc` with set-inner versus list semantics (memory usage in Bytes).

7) *Biere* ($n = 25$):

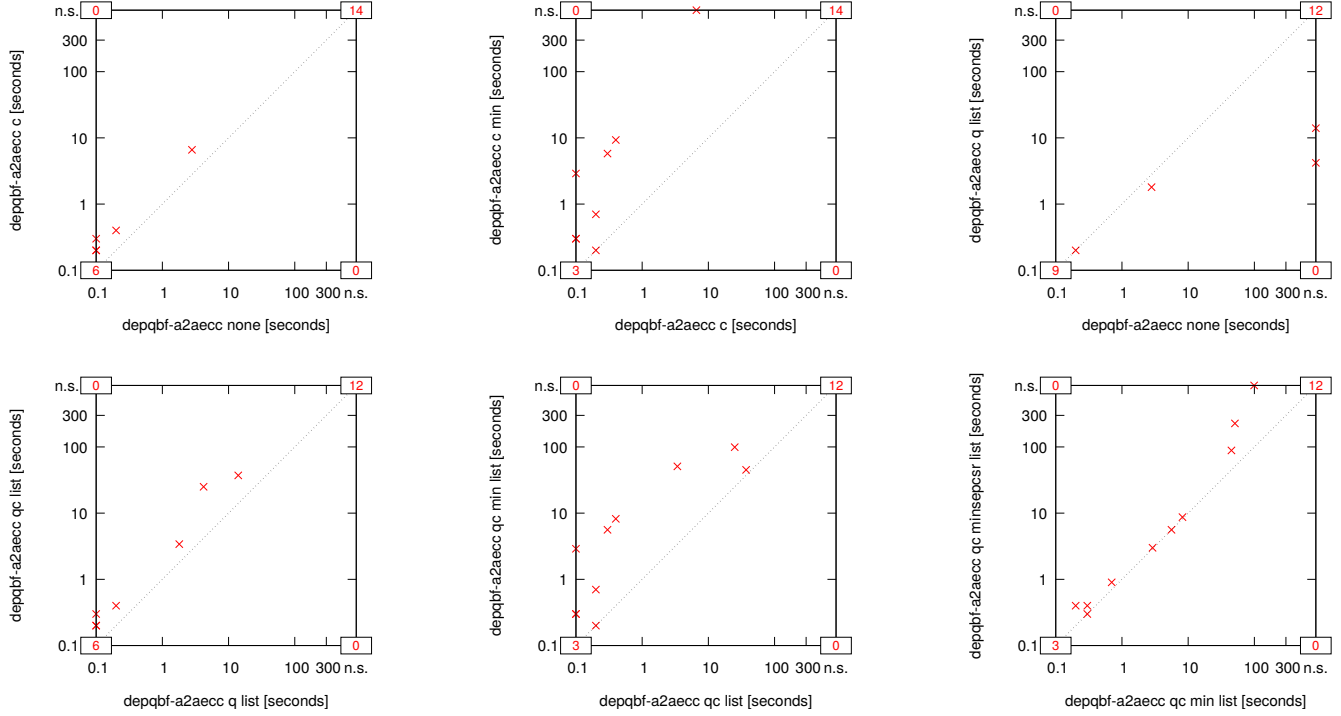


Fig. 597: Suite Biere ($n = 25$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

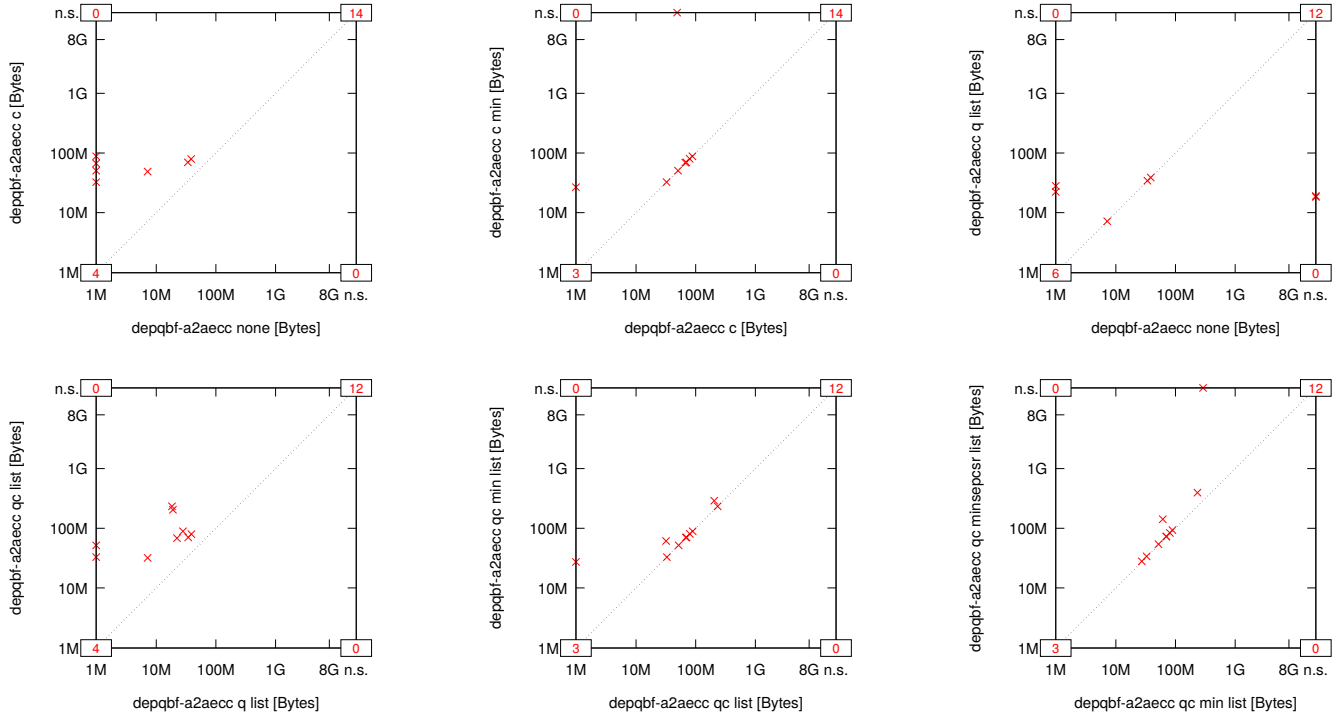


Fig. 598: Suite Biere ($n = 25$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

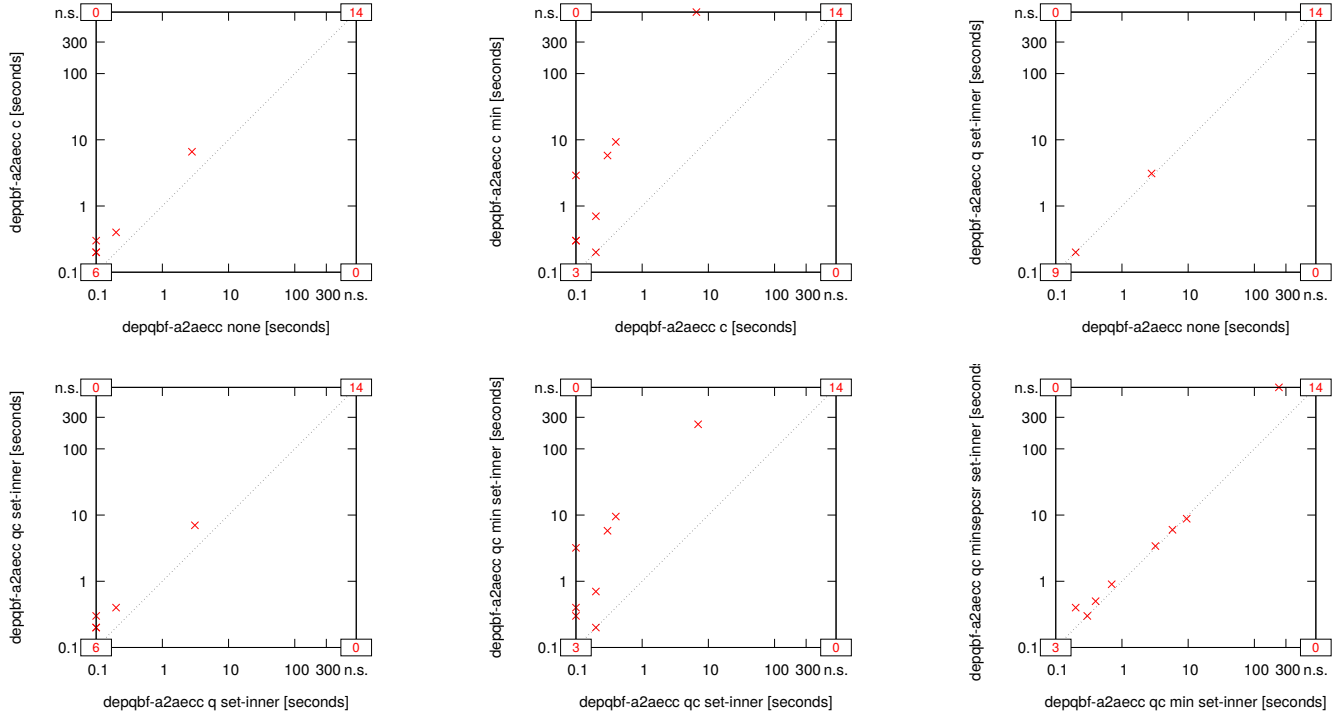


Fig. 599: Suite Biere ($n = 25$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

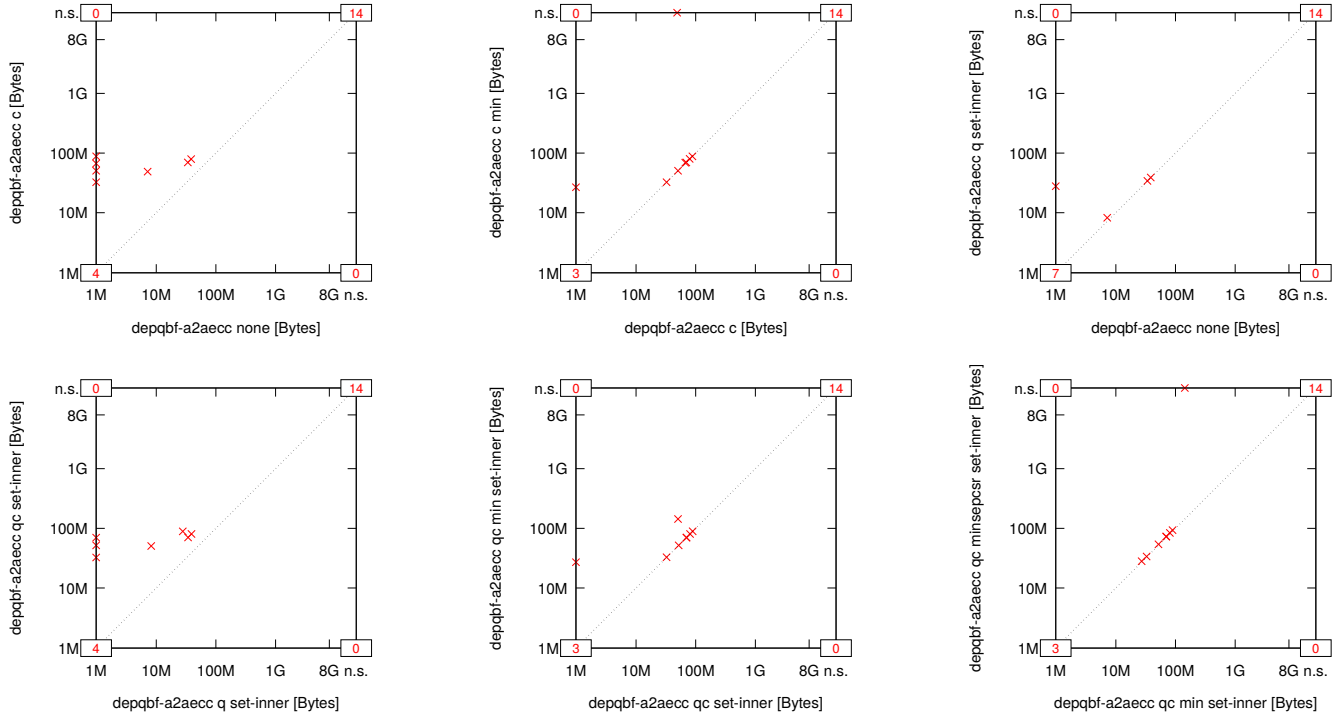


Fig. 600: Suite Biere ($n = 25$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

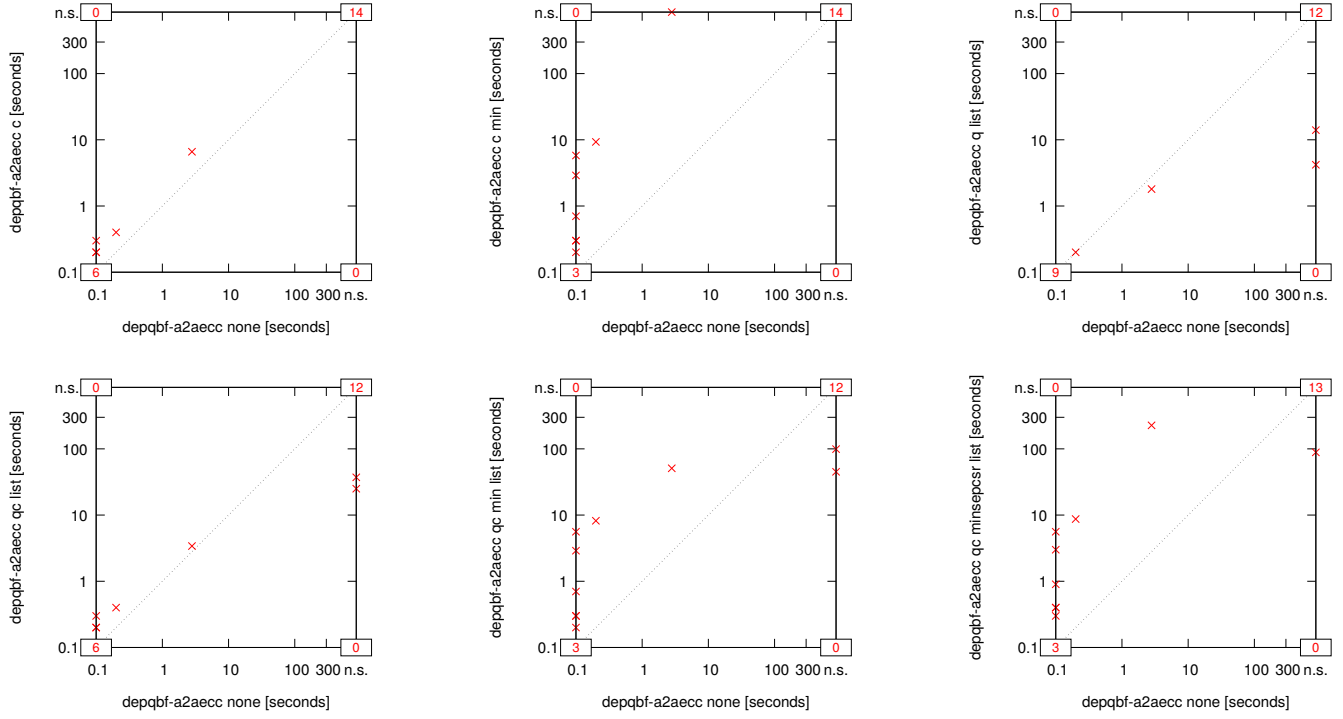


Fig. 601: Suite Biere ($n = 25$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

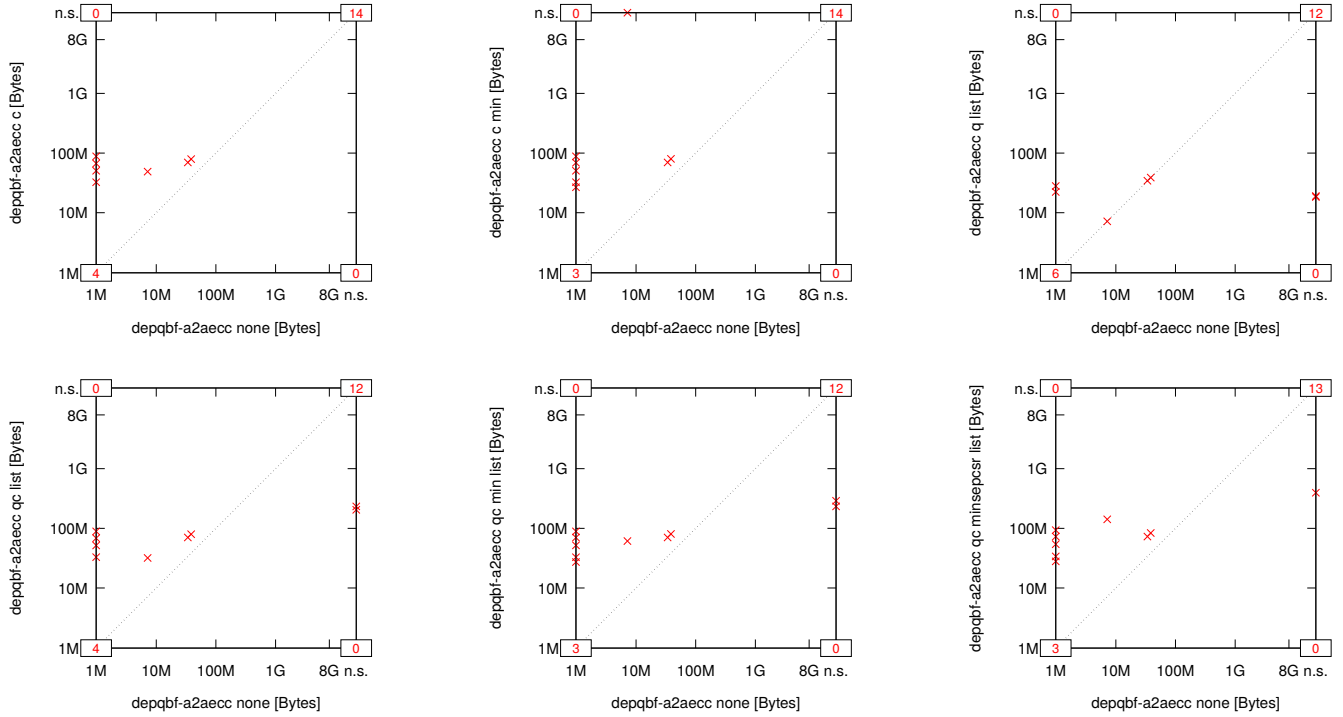


Fig. 602: Suite Biere ($n = 25$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

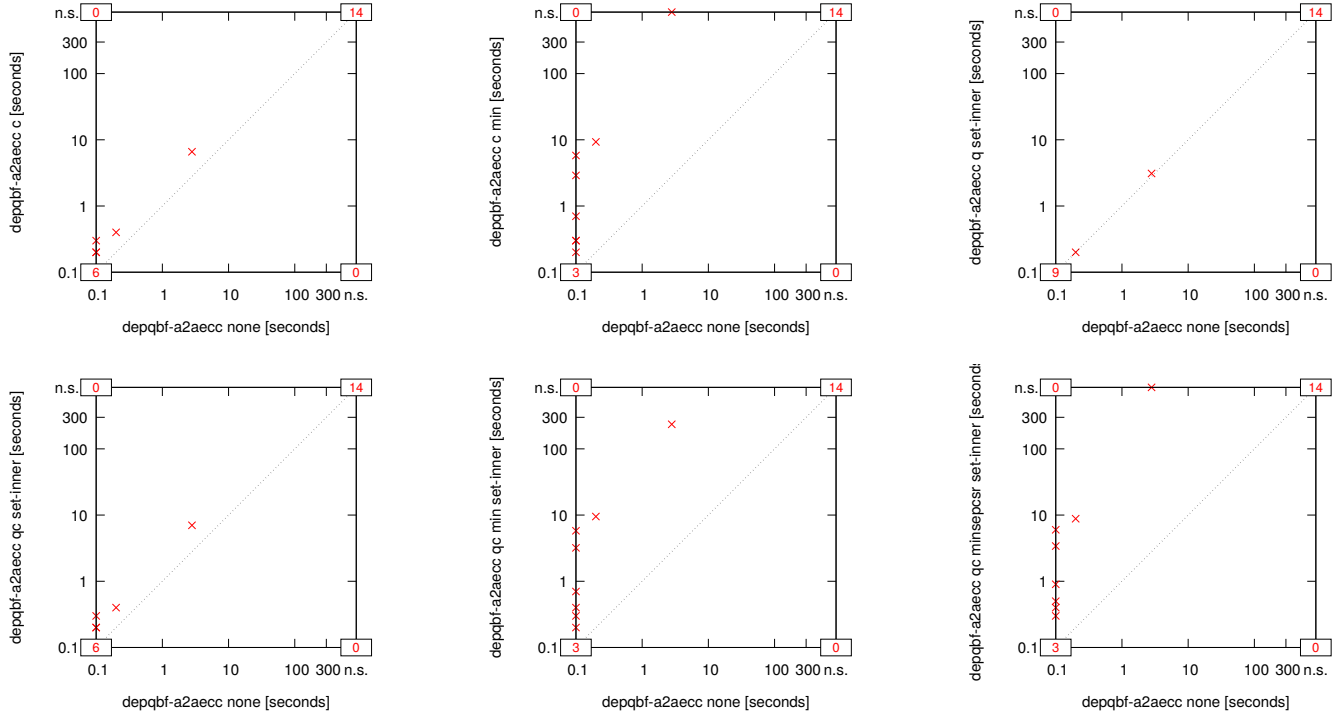


Fig. 603: Suite Biere ($n = 25$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

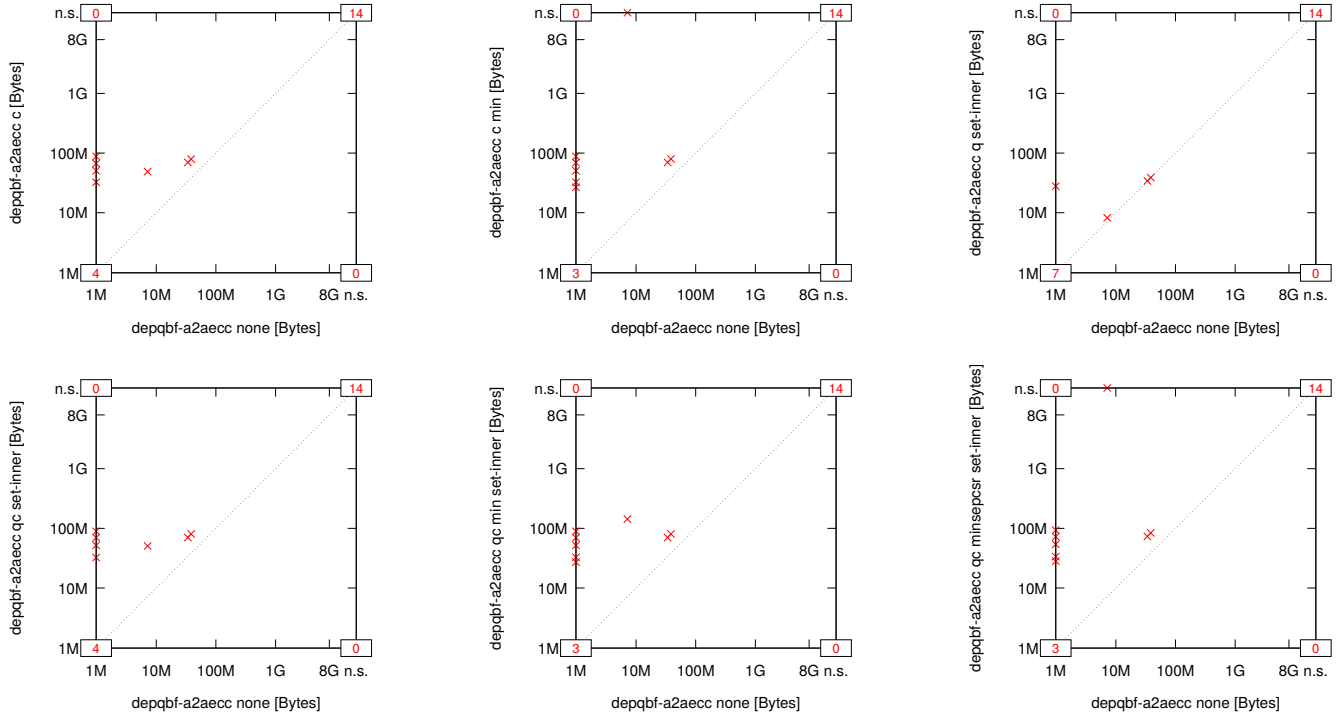


Fig. 604: Suite Biere ($n = 25$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

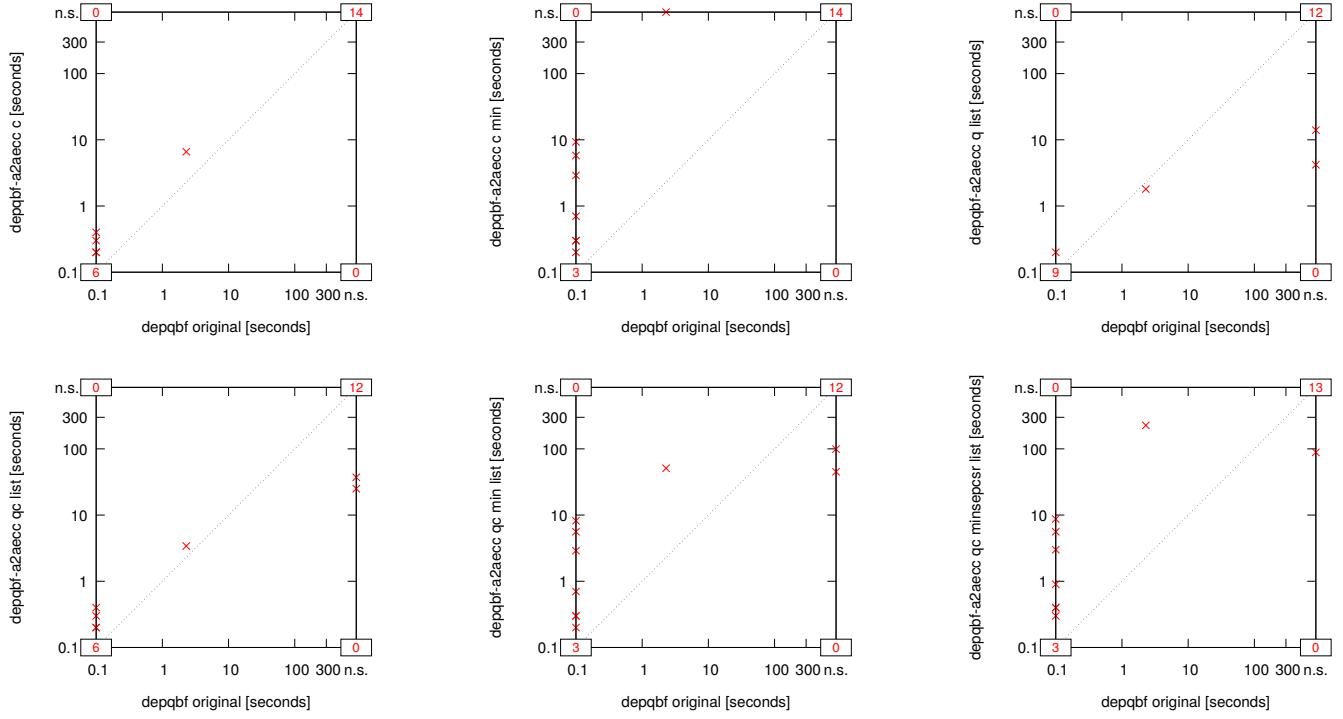


Fig. 605: Suite Biere ($n = 25$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

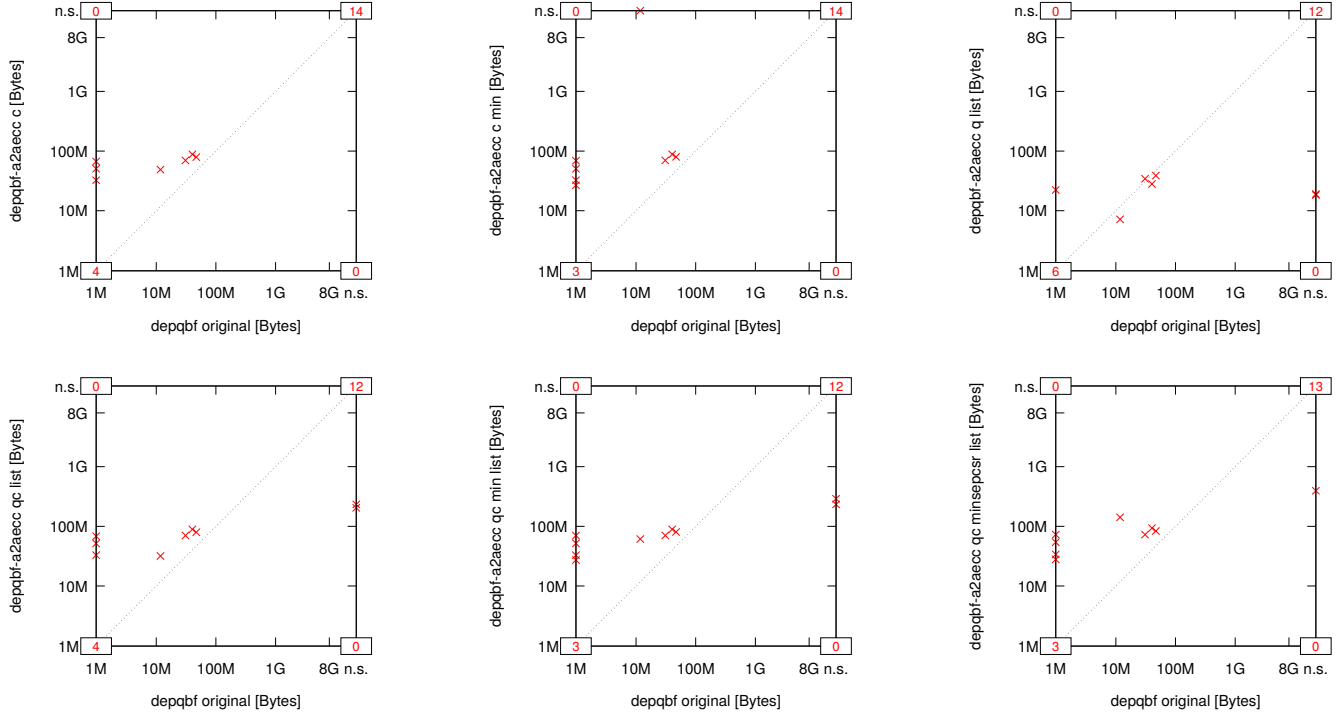


Fig. 606: Suite Biere ($n = 25$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

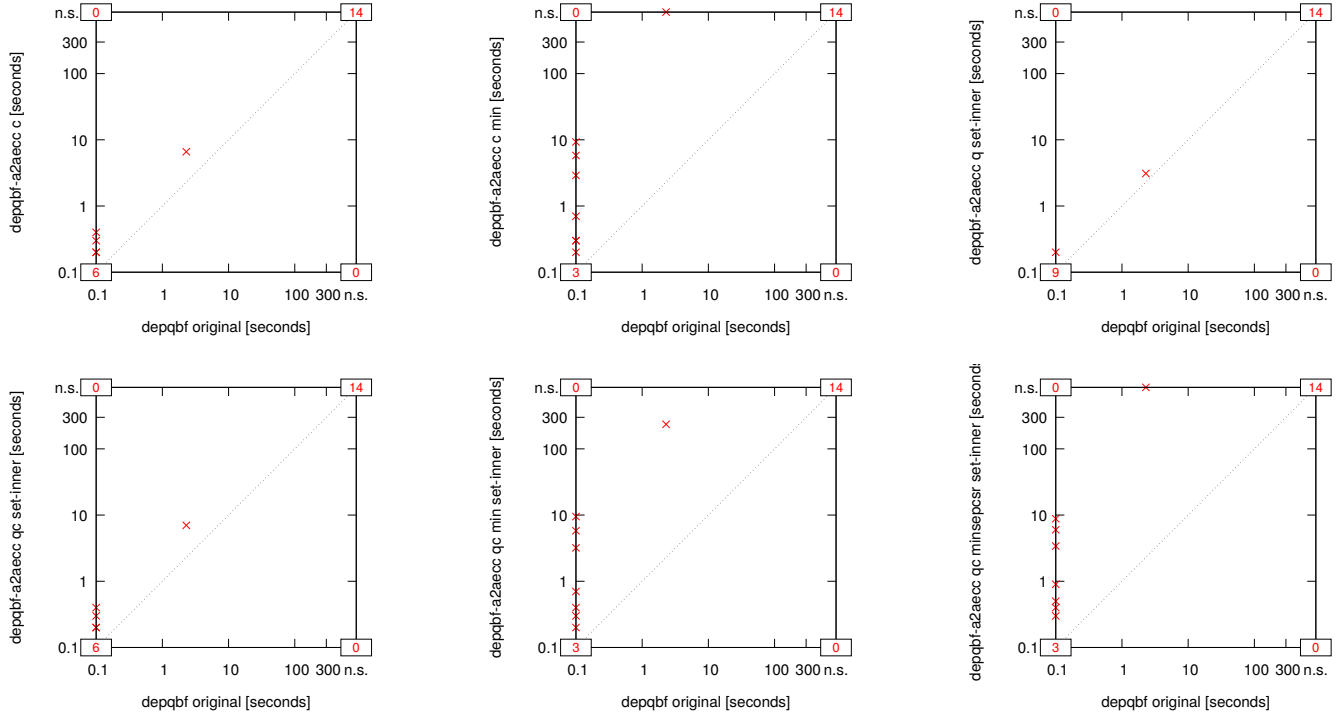


Fig. 607: Suite Biere ($n = 25$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

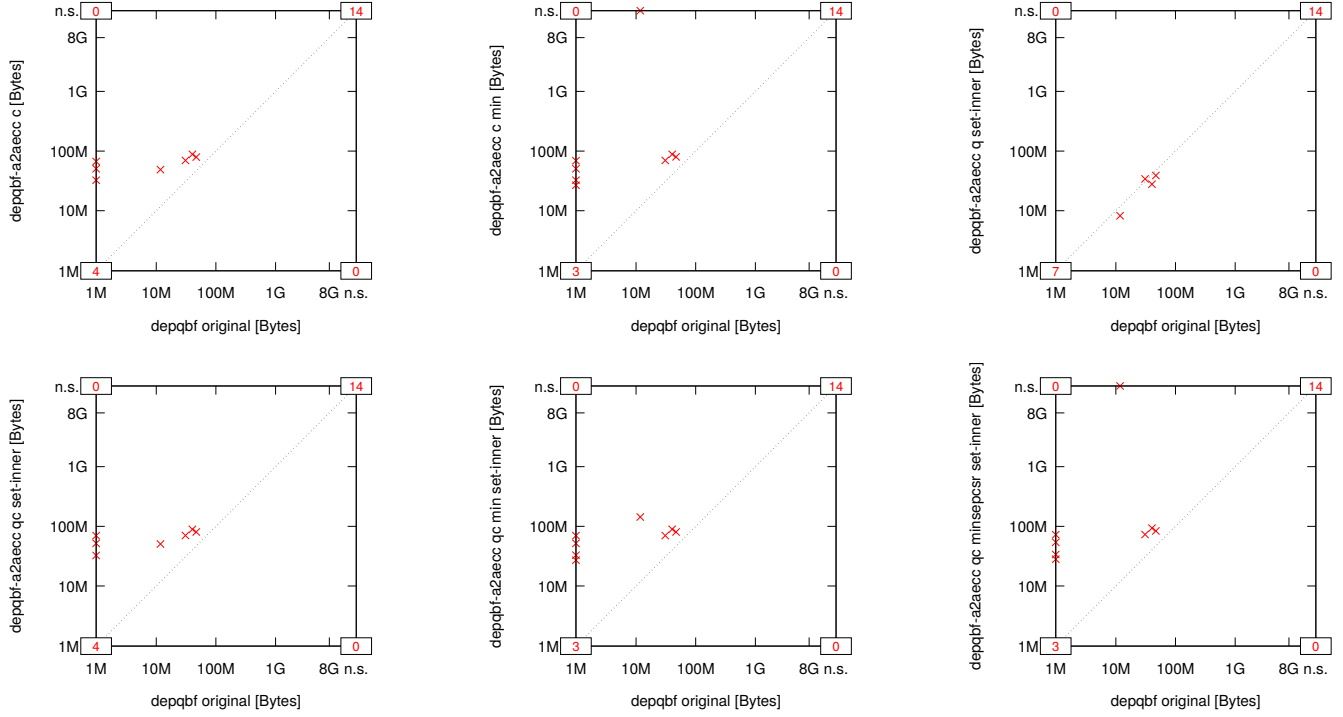


Fig. 608: Suite Biere ($n = 25$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

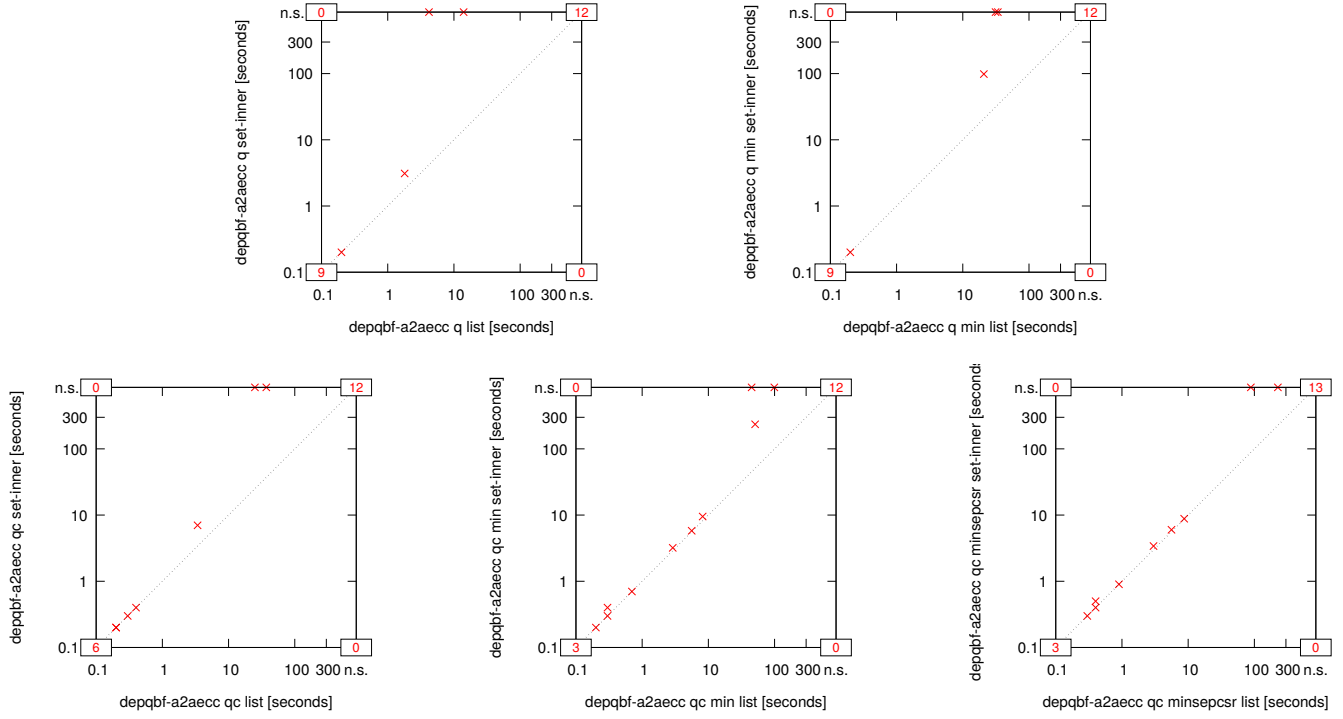


Fig. 609: Suite Biere ($n = 25$): Extracting and minimizing unsatisfiable cores in `DepQBF-a2aecc` with set-inner versus list semantics (run time in seconds).

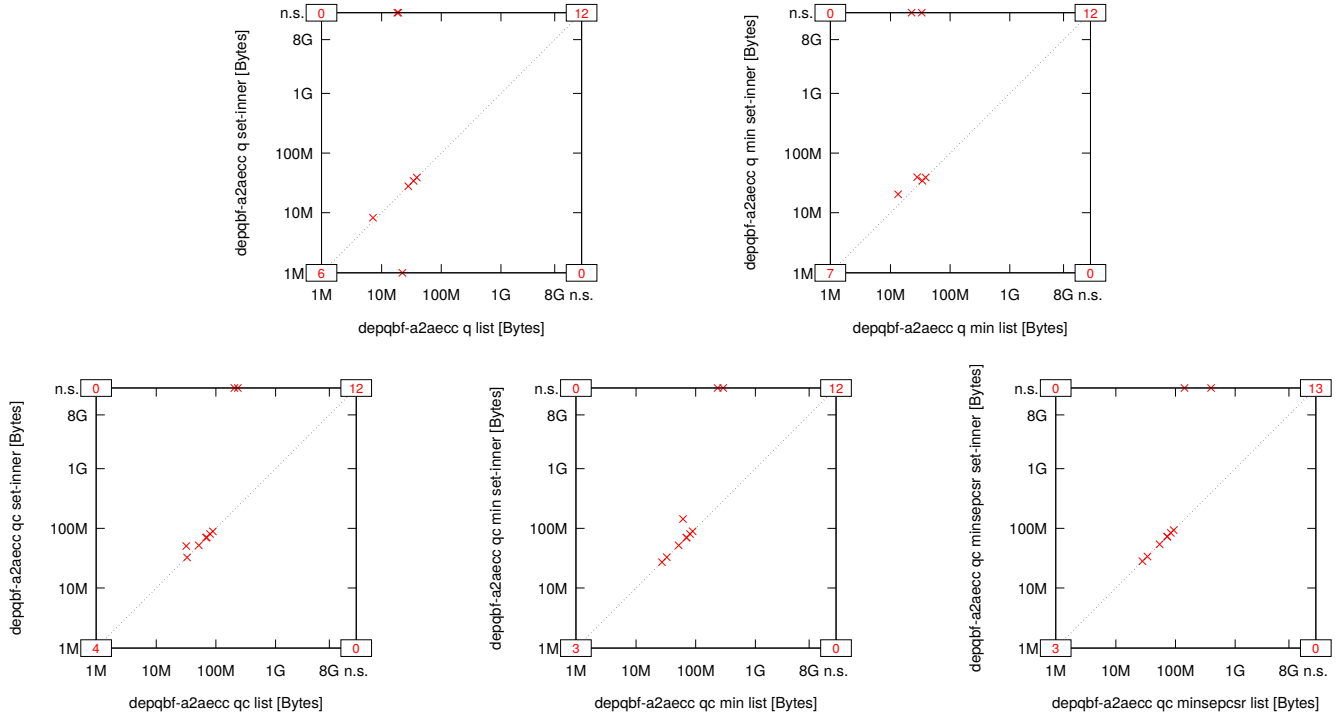


Fig. 610: Suite Biere ($n = 25$): Extracting and minimizing unsatisfiable cores in `DepQBF-a2aecc` with set-inner versus list semantics (memory usage in Bytes).

8) *Cashmore-Fox-Giunchiglia* ($n = 110$):

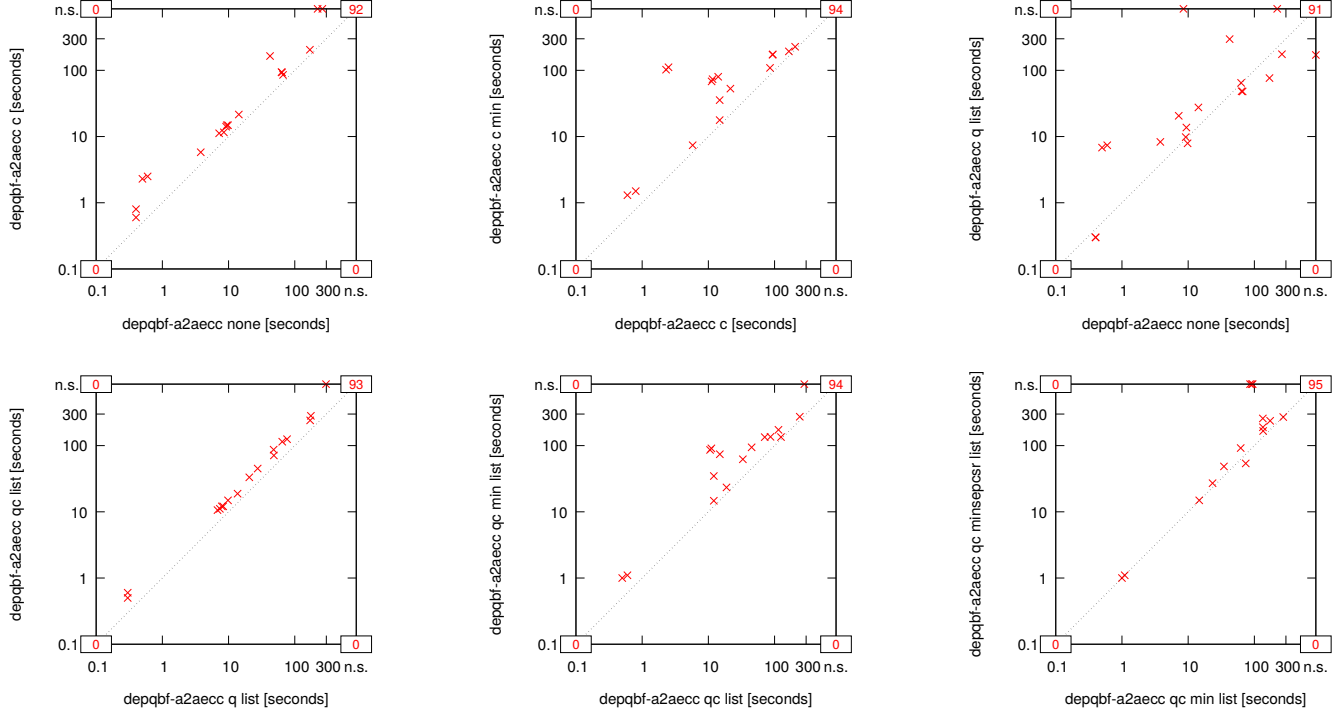


Fig. 611: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

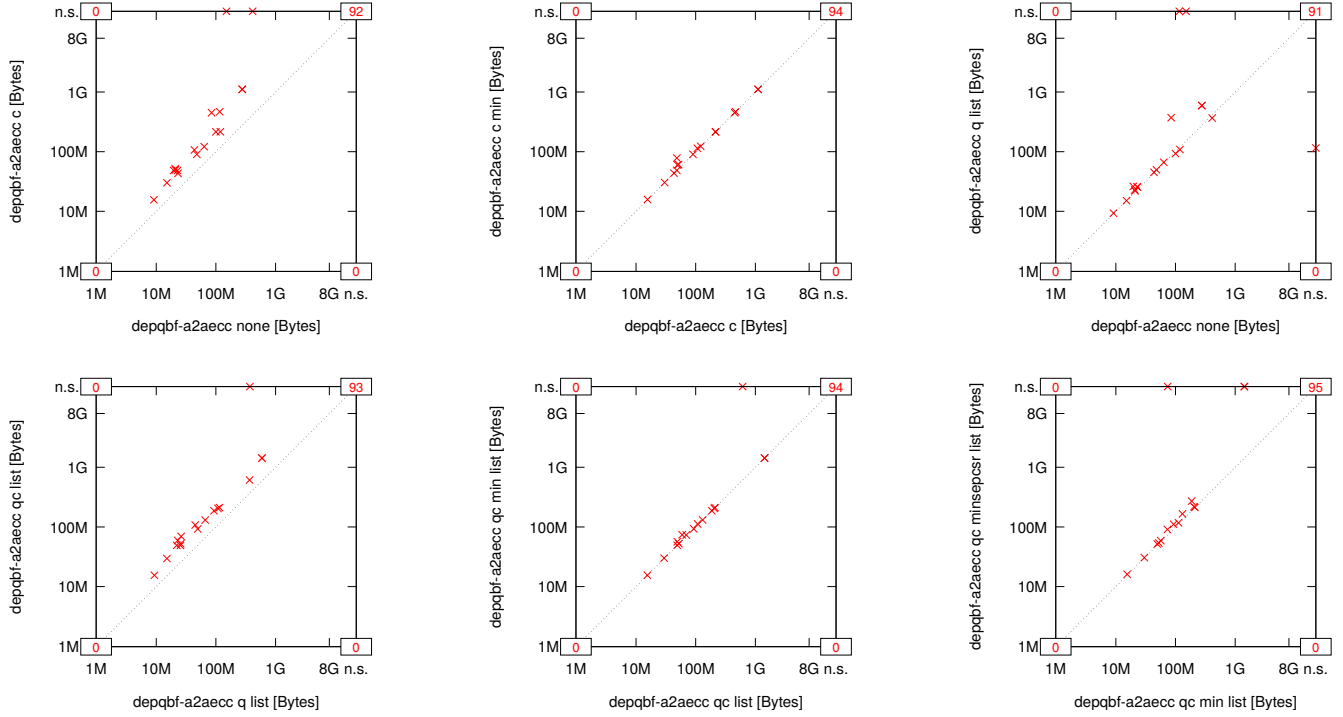


Fig. 612: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

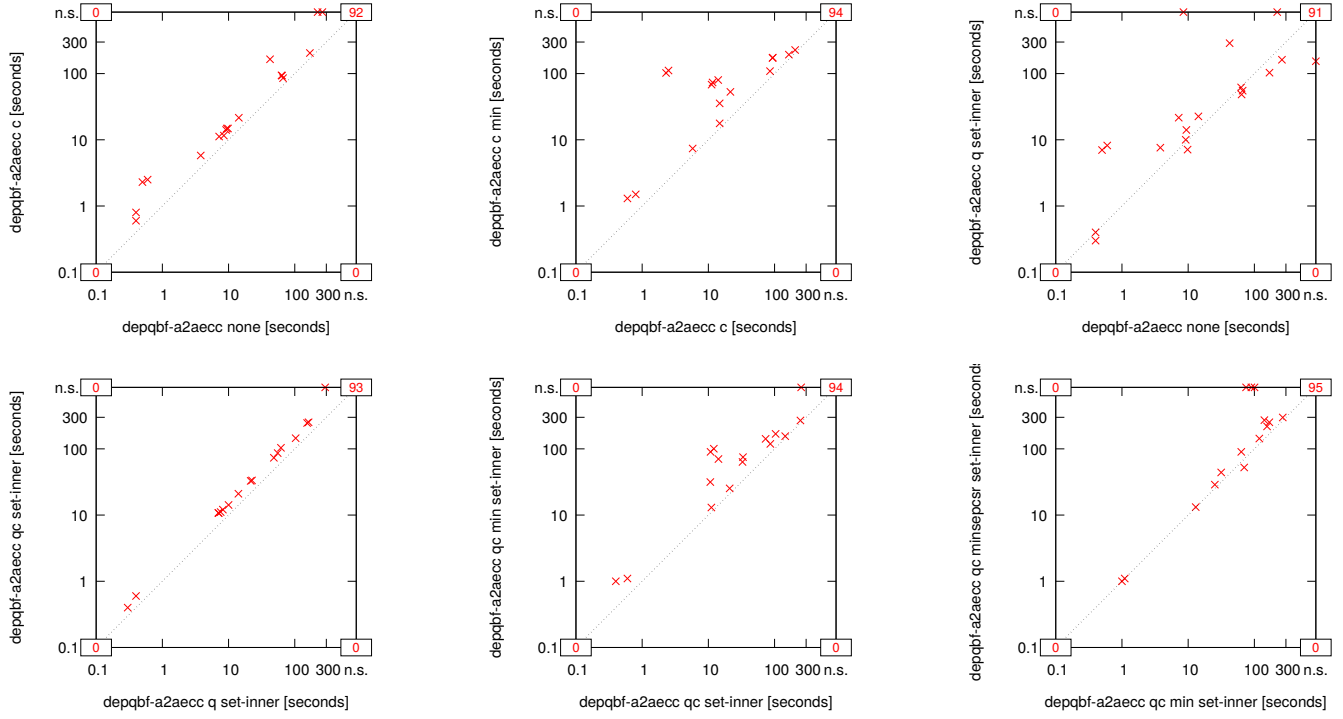


Fig. 613: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

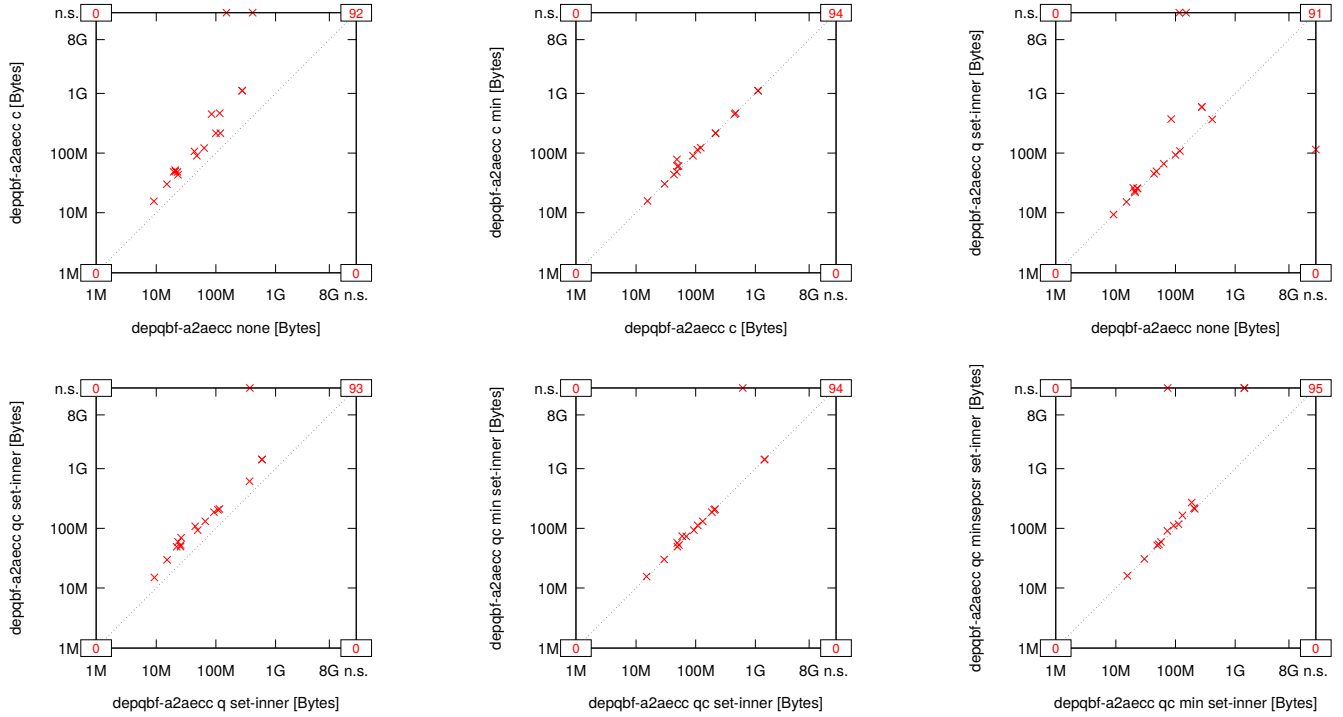


Fig. 614: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

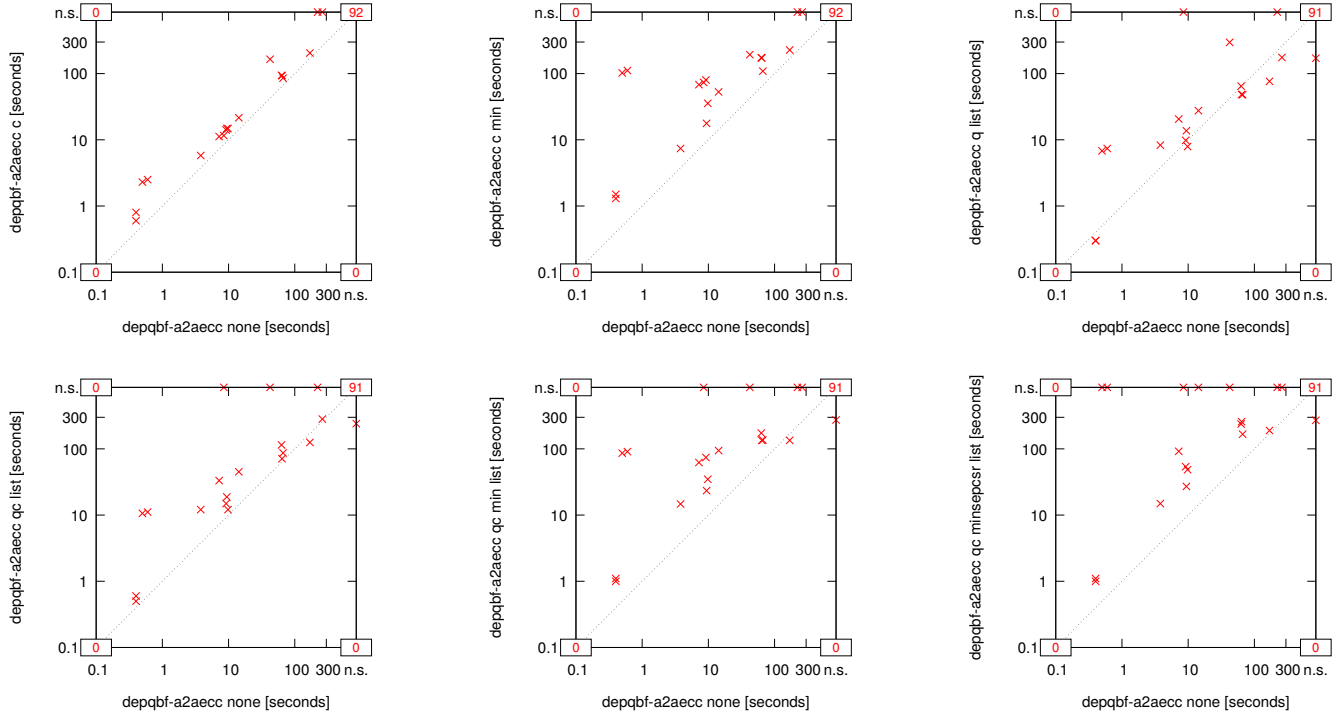


Fig. 615: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

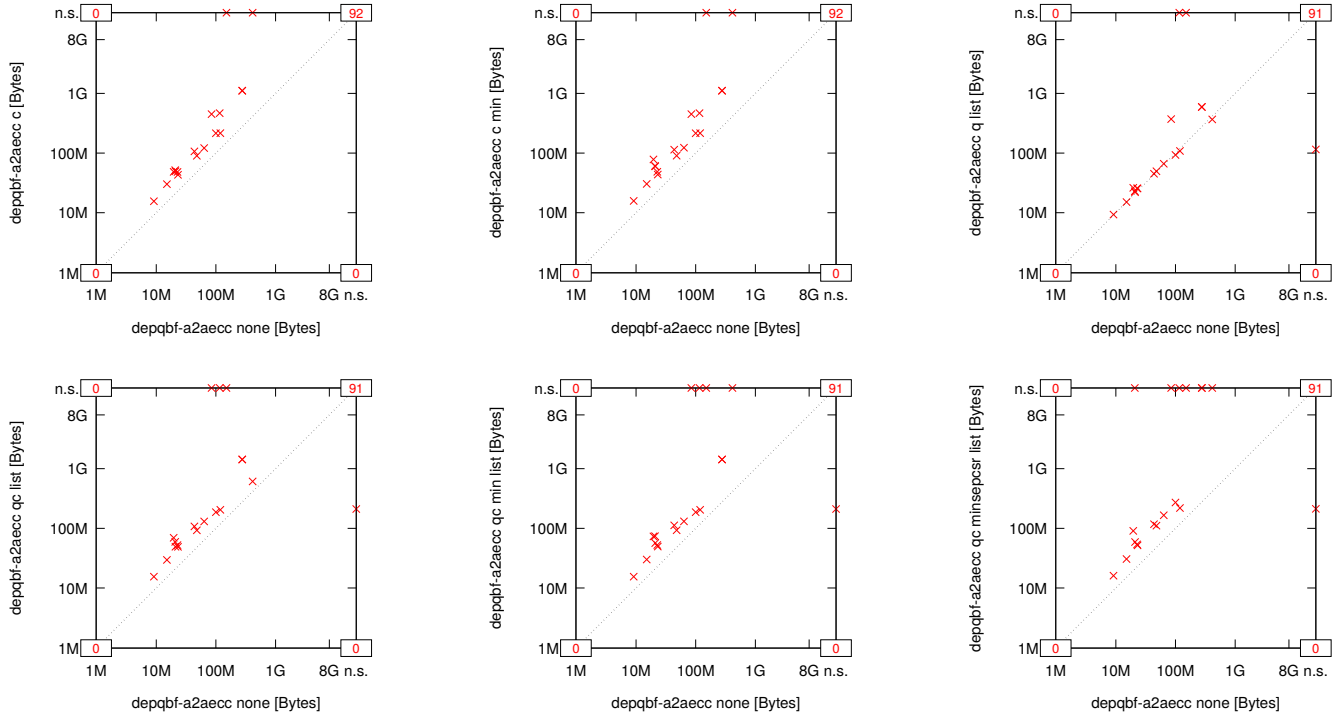


Fig. 616: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

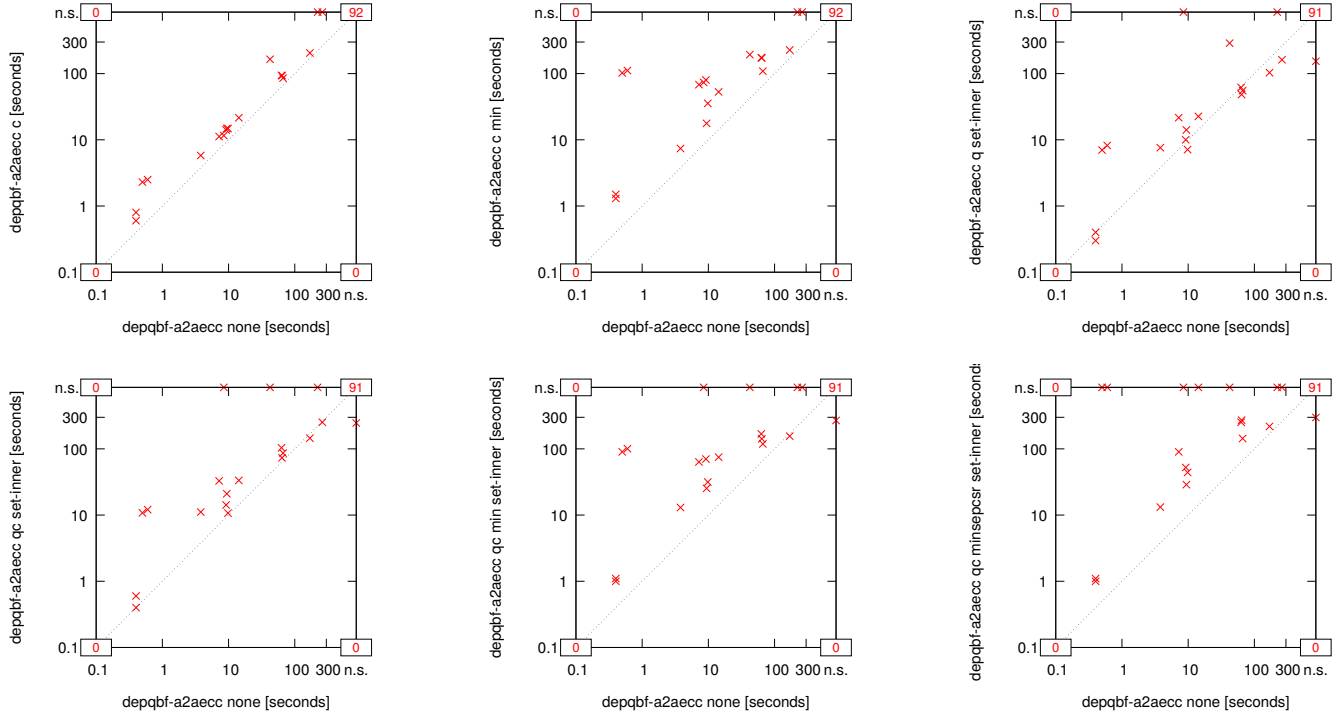


Fig. 617: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

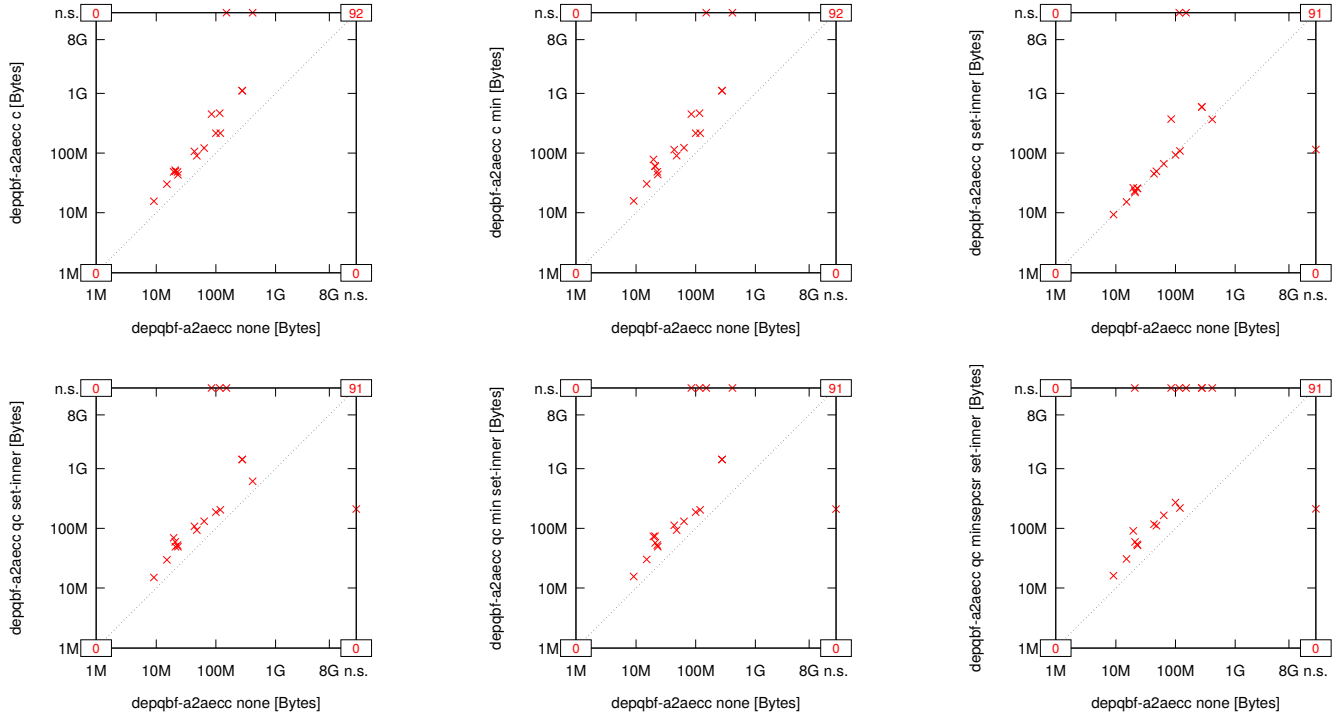


Fig. 618: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

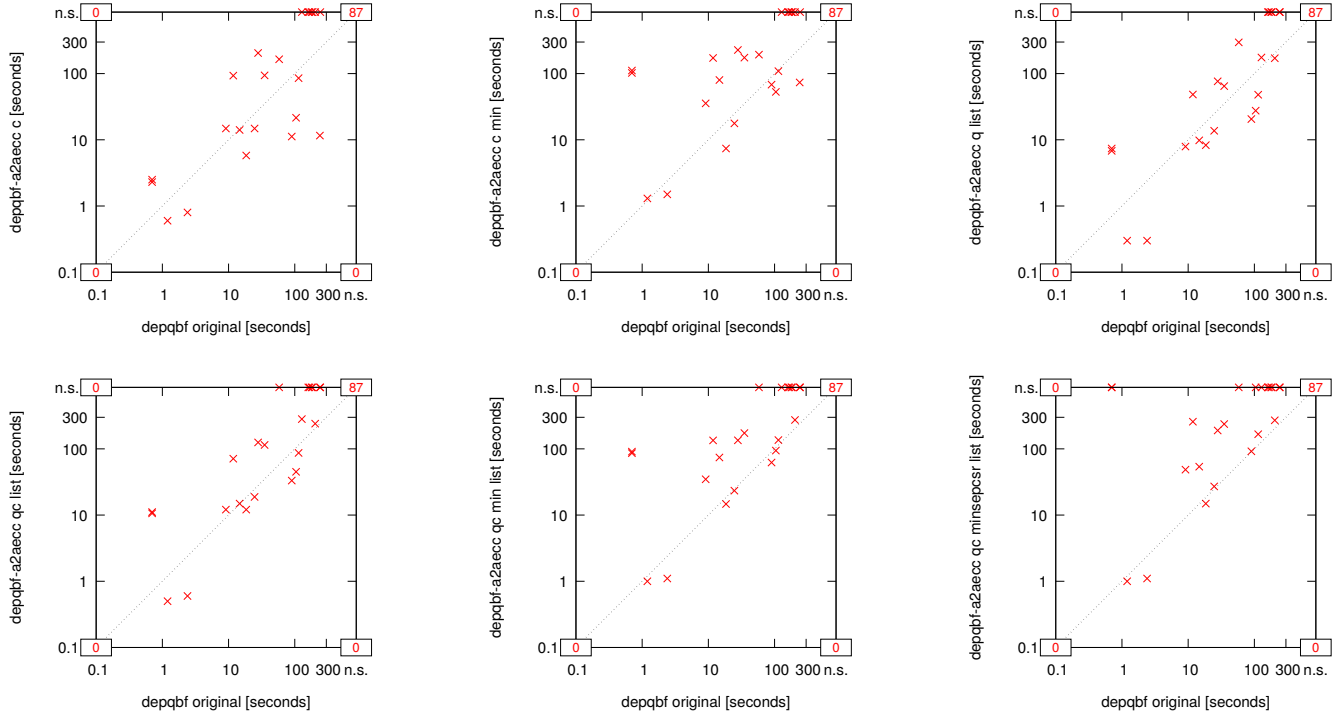


Fig. 619: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

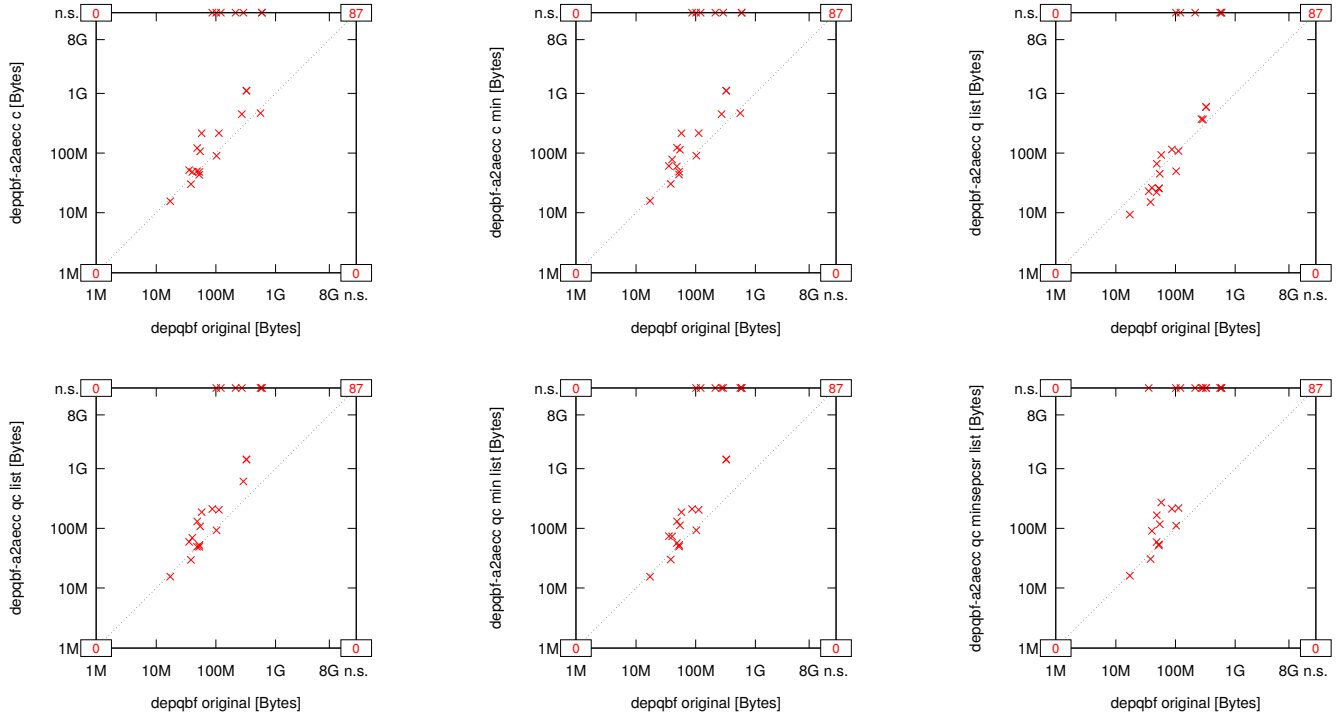


Fig. 620: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

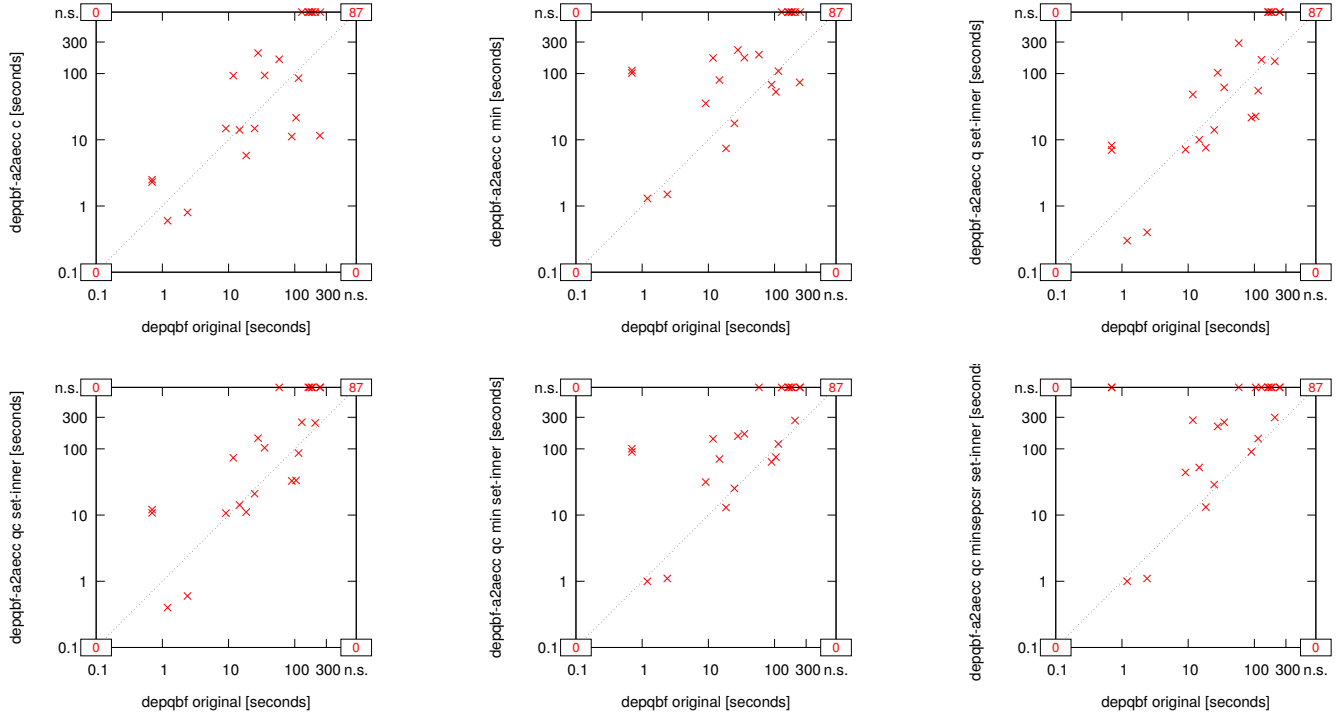


Fig. 621: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

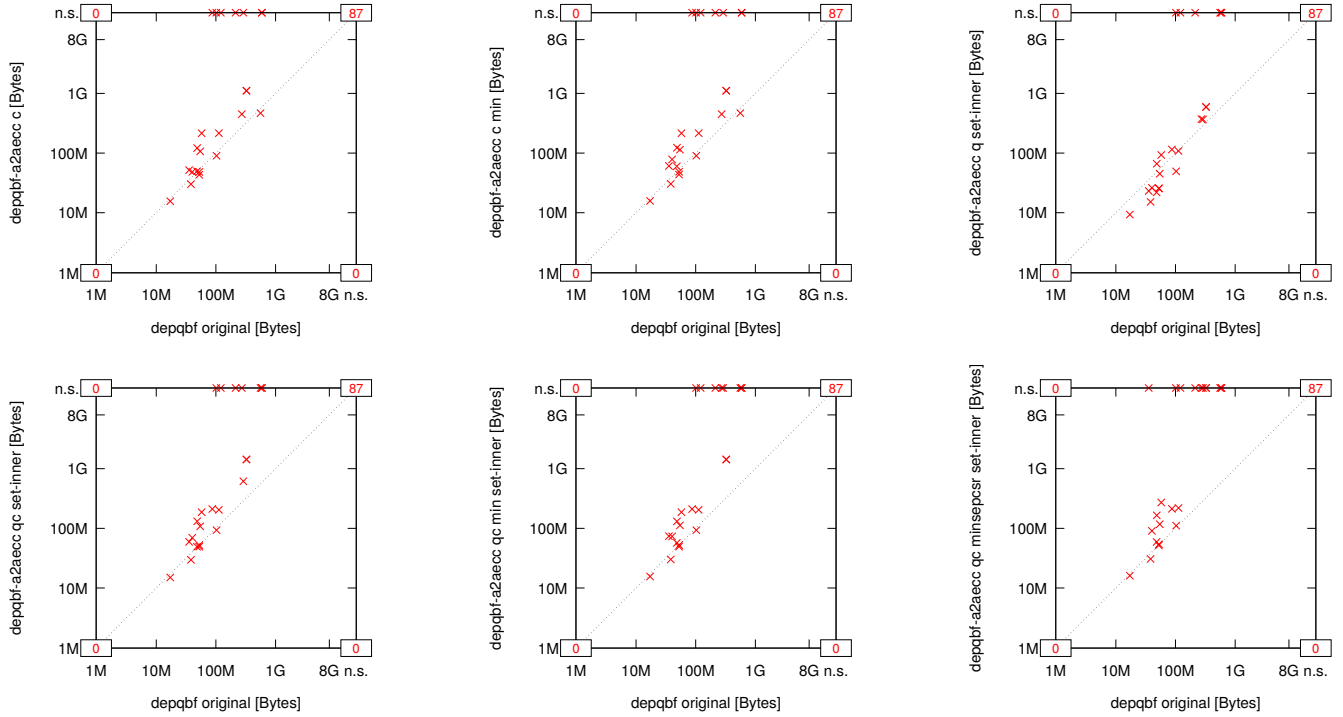


Fig. 622: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

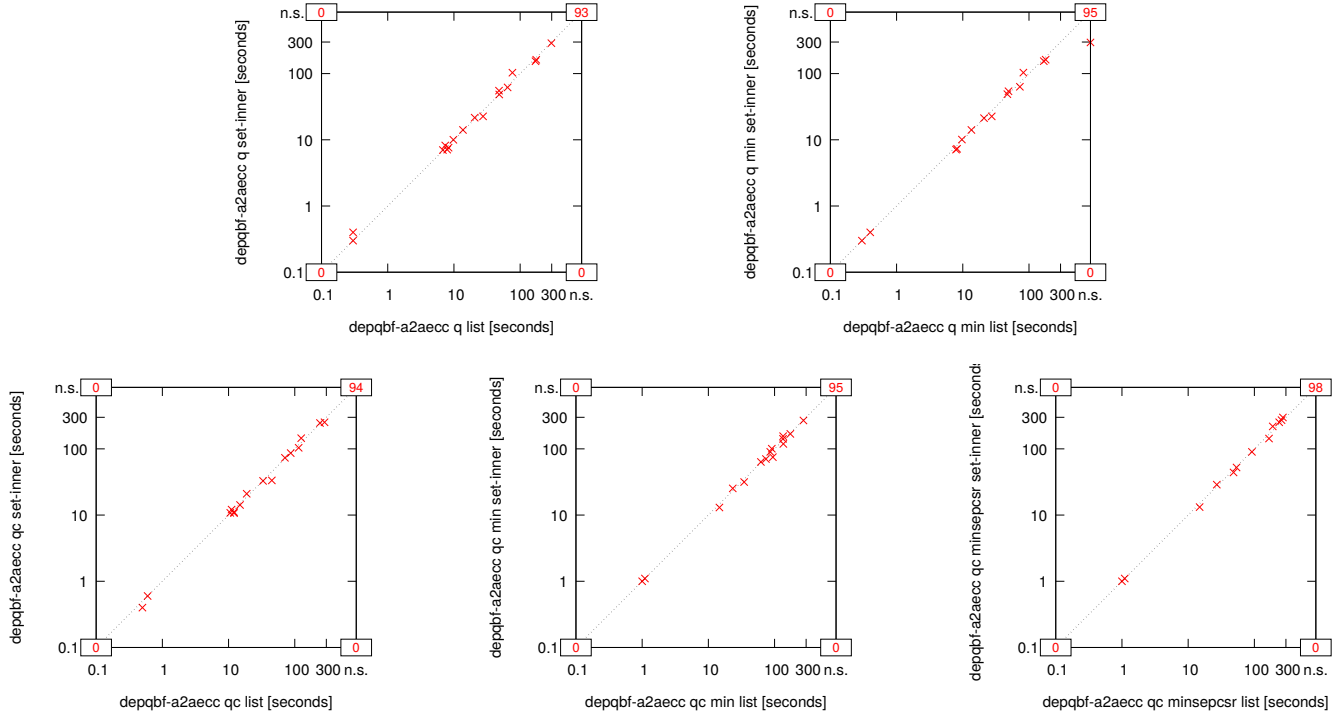


Fig. 623: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

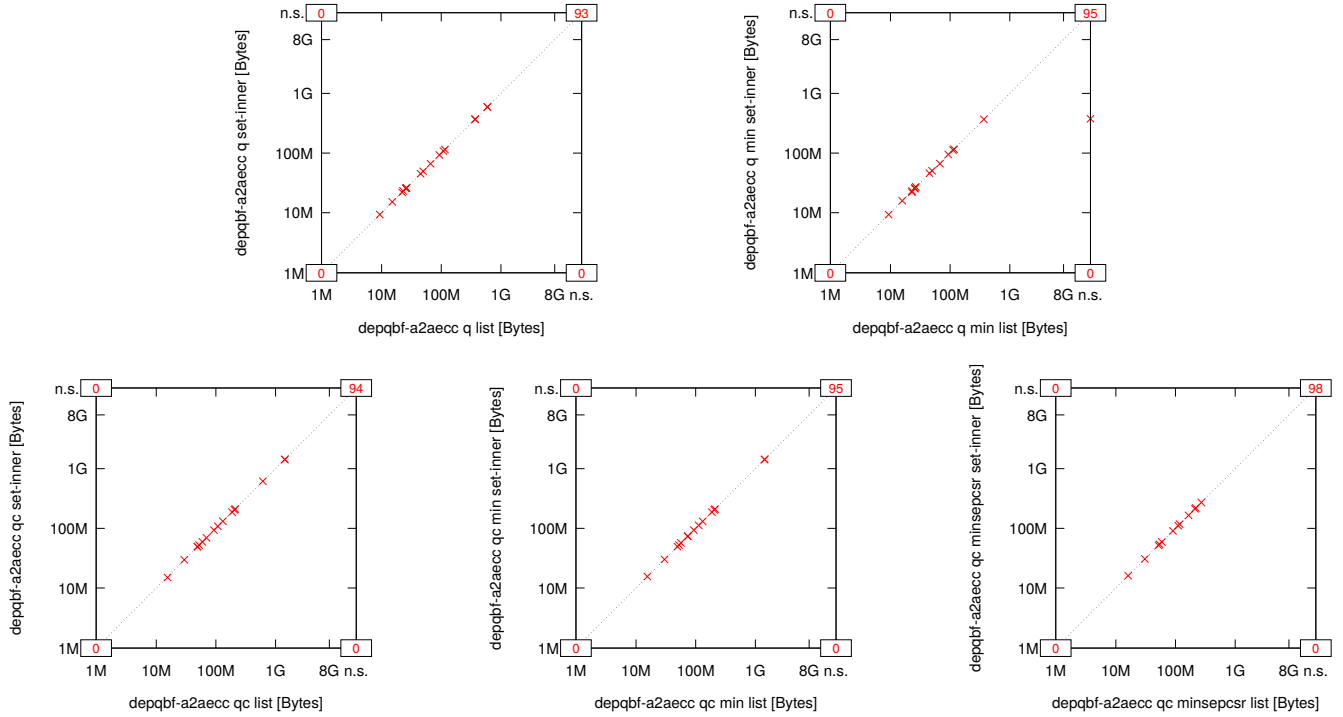


Fig. 624: Suite Cashmore-Fox-Giunchiglia ($n = 110$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

9) Castellini ($n = 112$):

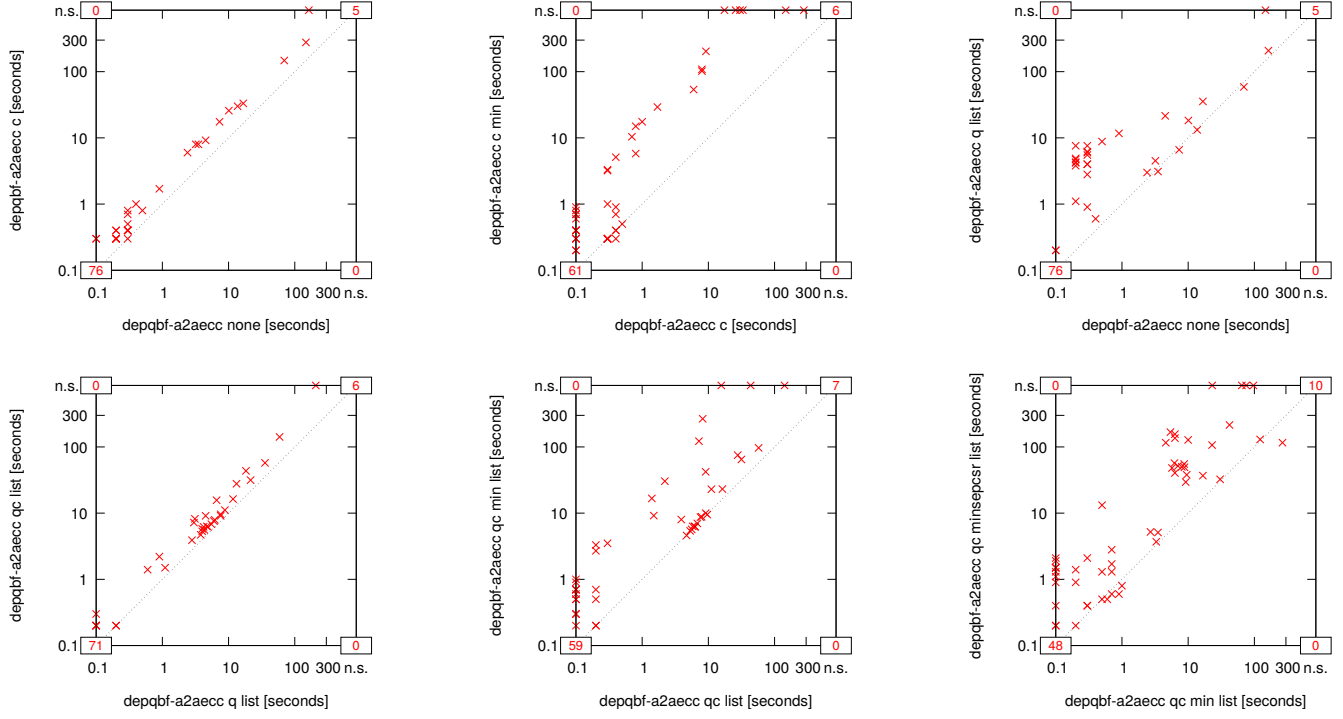


Fig. 625: Suite Castellini ($n = 112$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

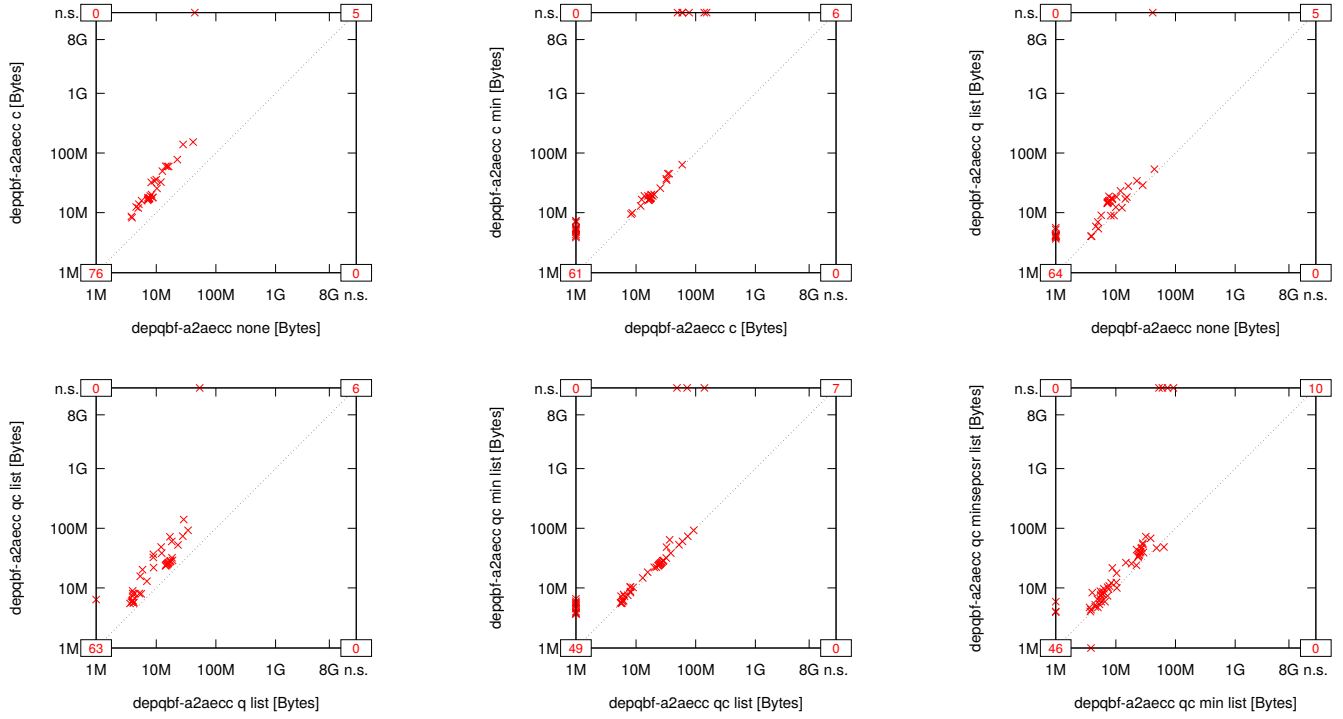


Fig. 626: Suite Castellini ($n = 112$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

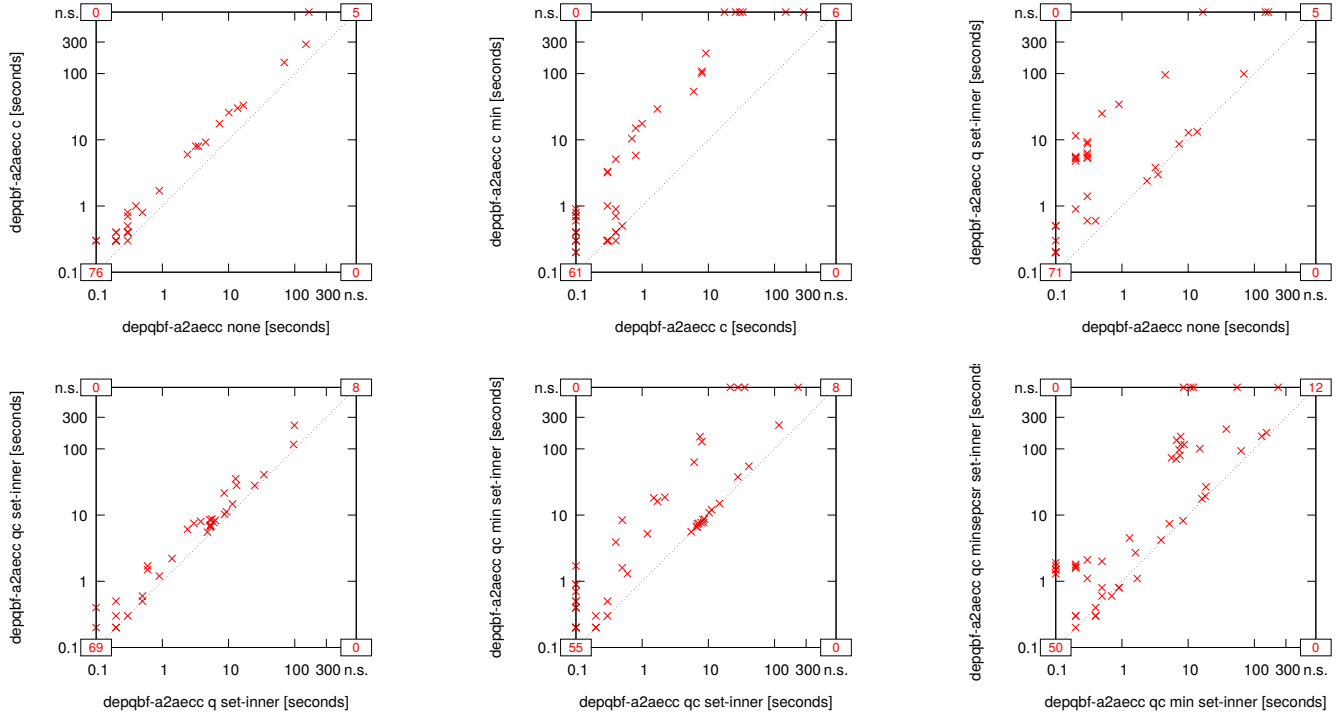


Fig. 627: Suite Castellini ($n = 112$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

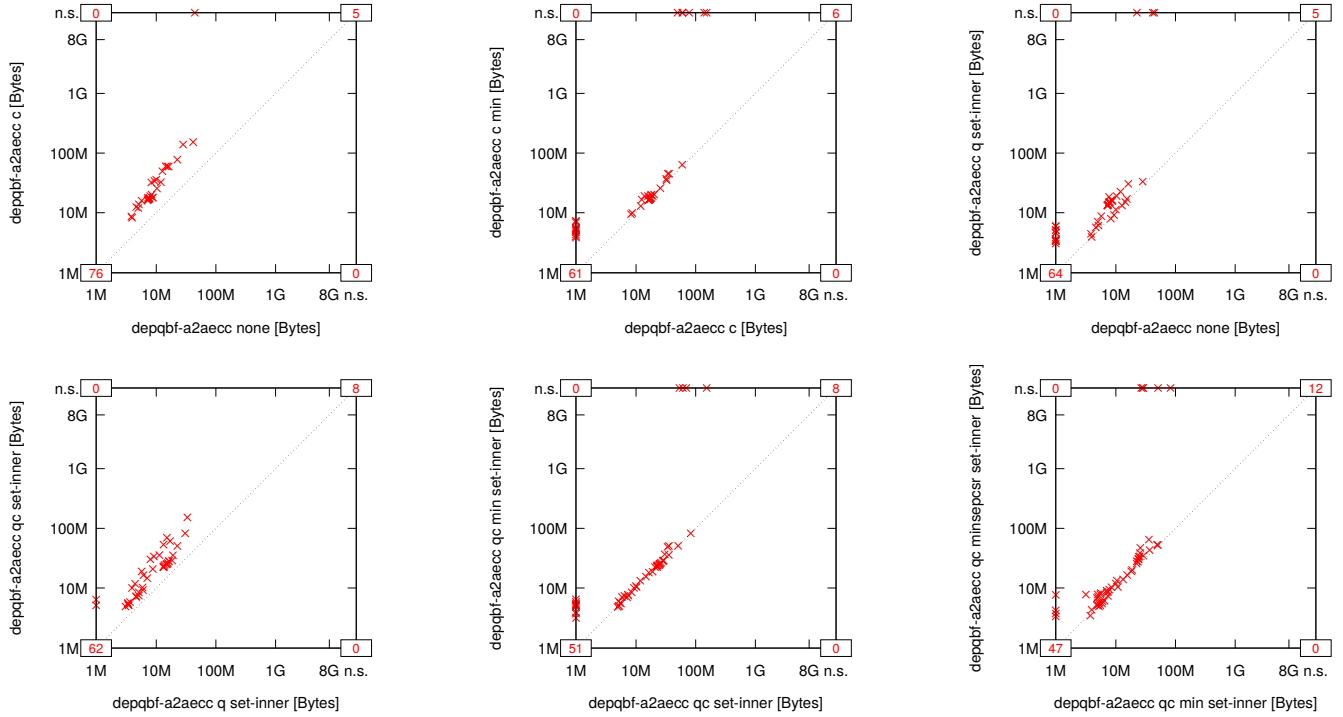


Fig. 628: Suite Castellini ($n = 112$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

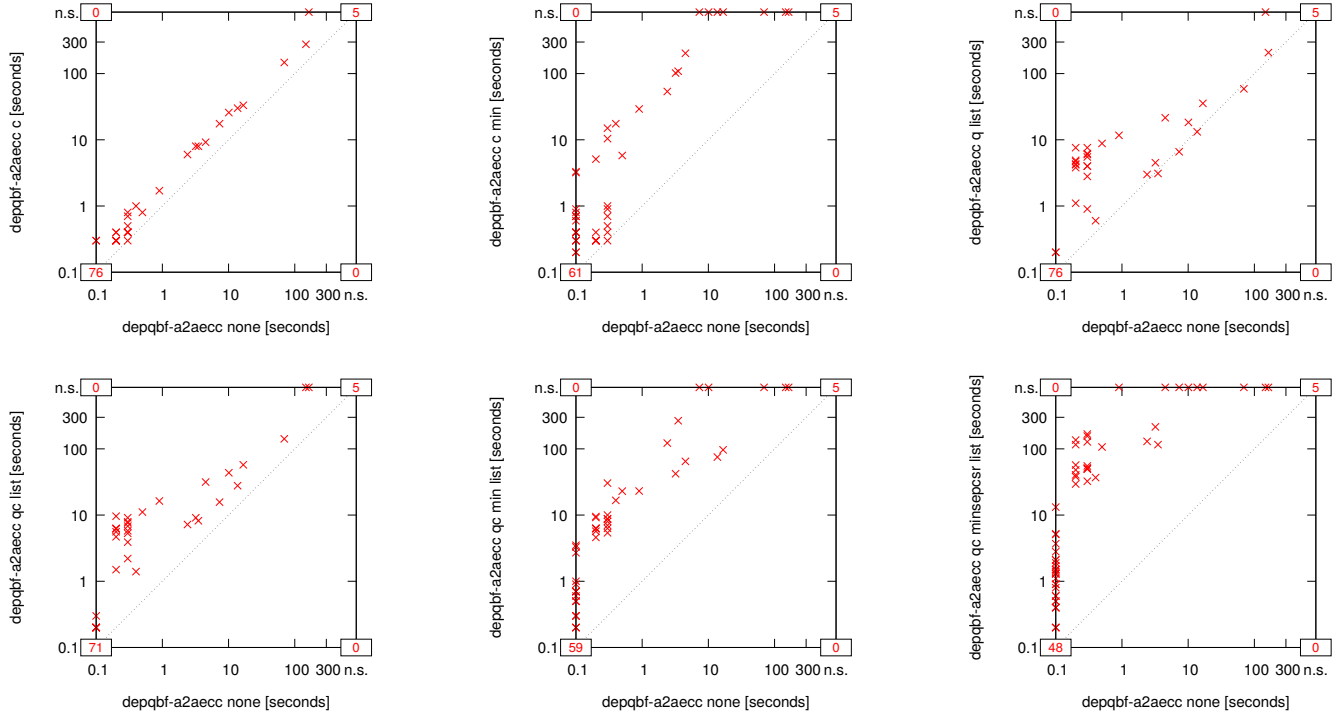


Fig. 629: Suite Castellini ($n = 112$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

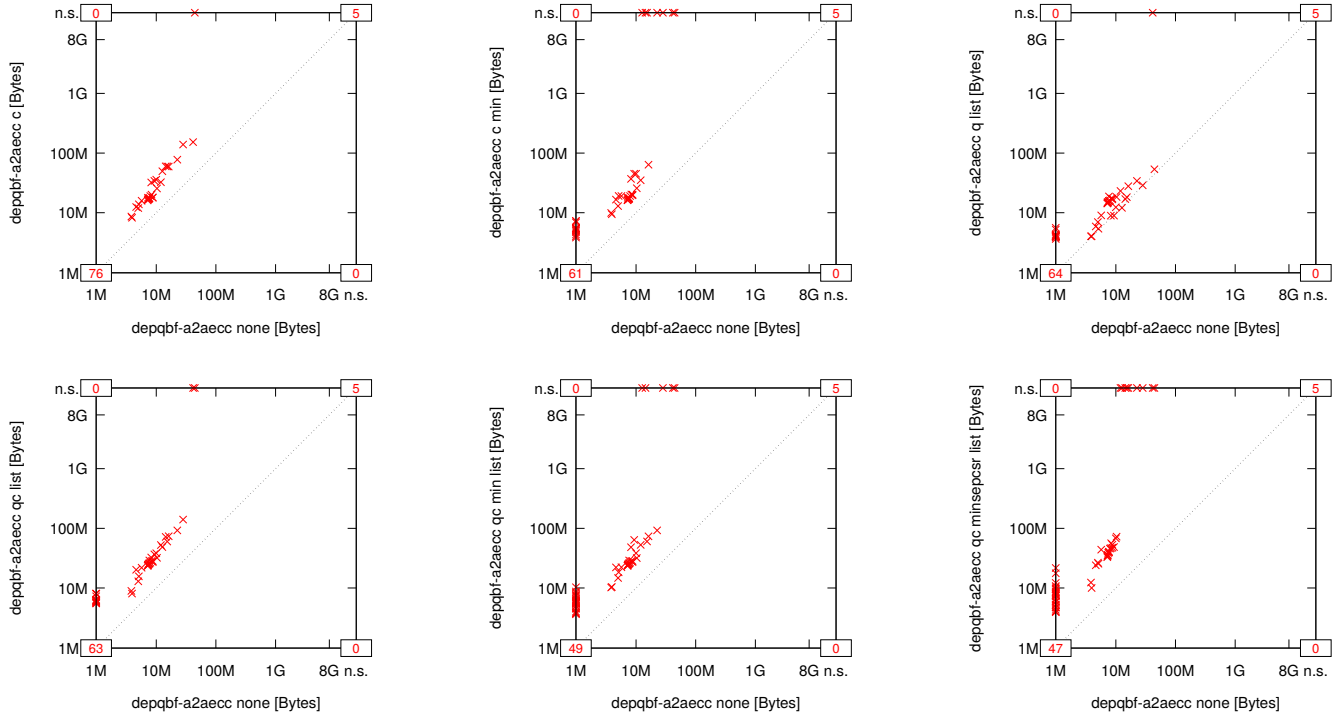


Fig. 630: Suite Castellini ($n = 112$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

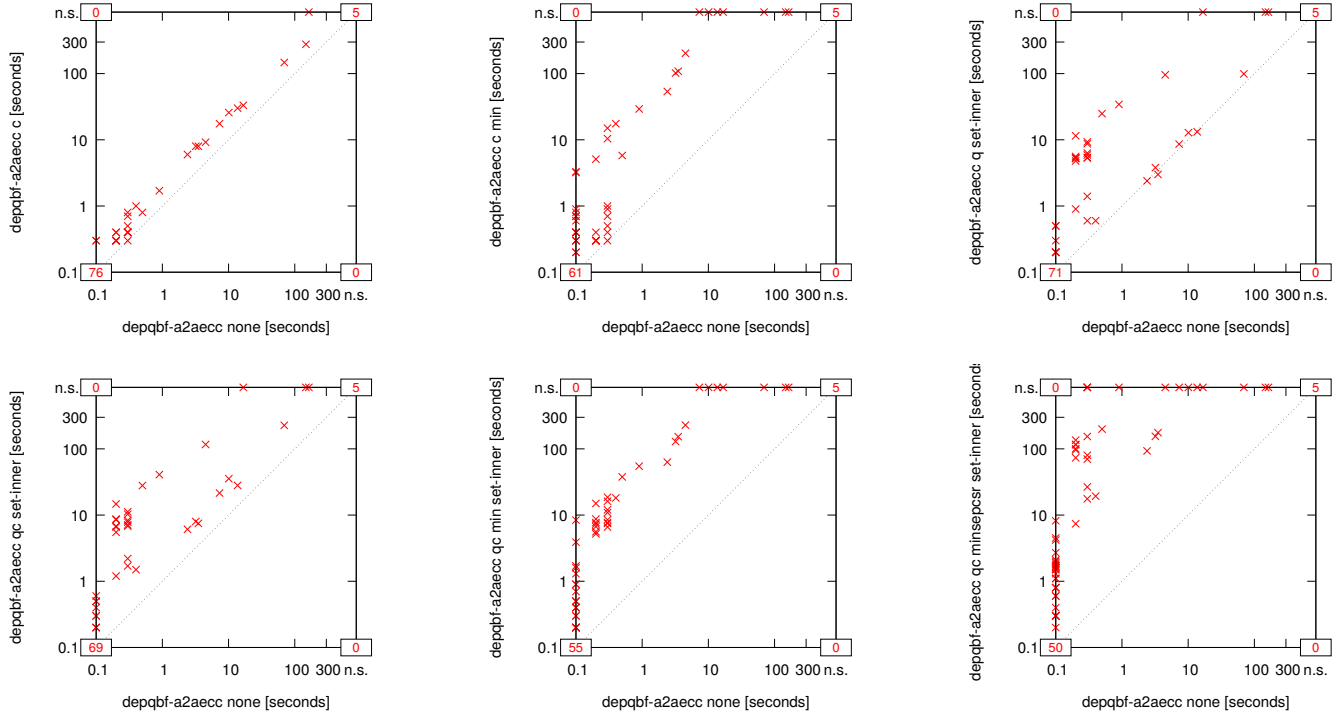


Fig. 631: Suite Castellini ($n = 112$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

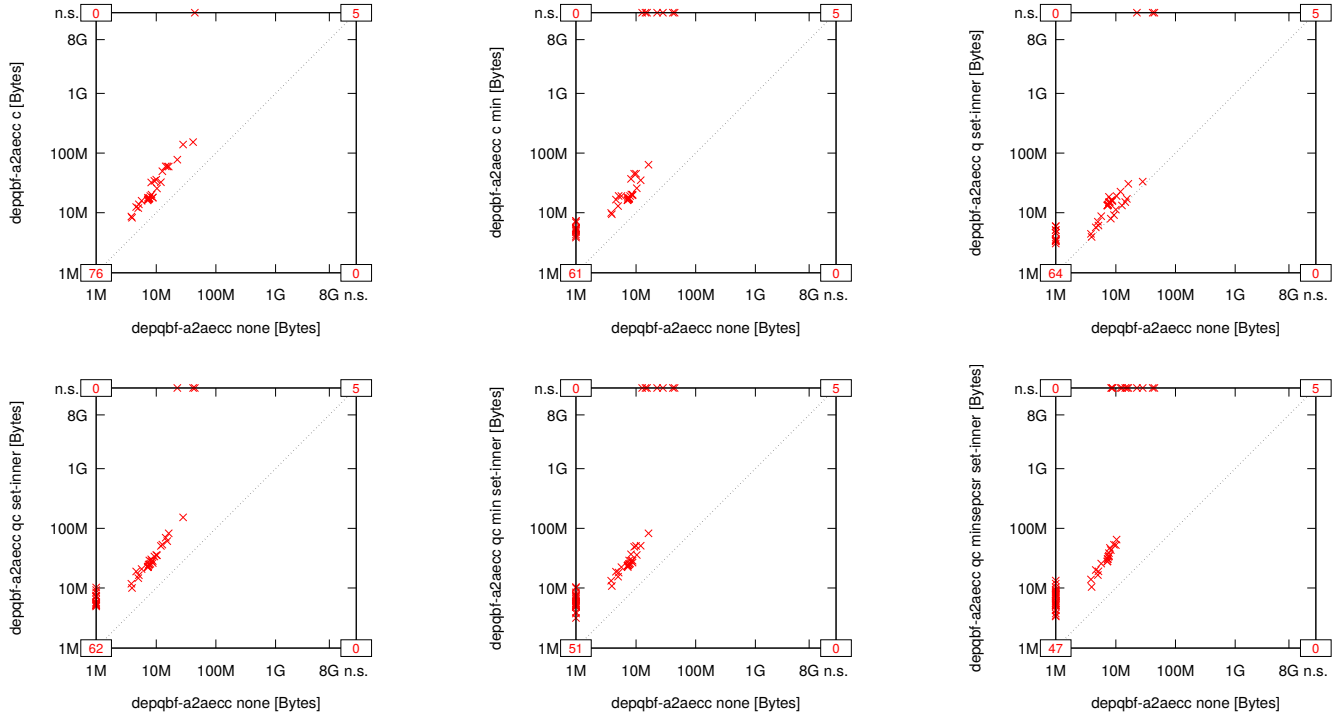


Fig. 632: Suite Castellini ($n = 112$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

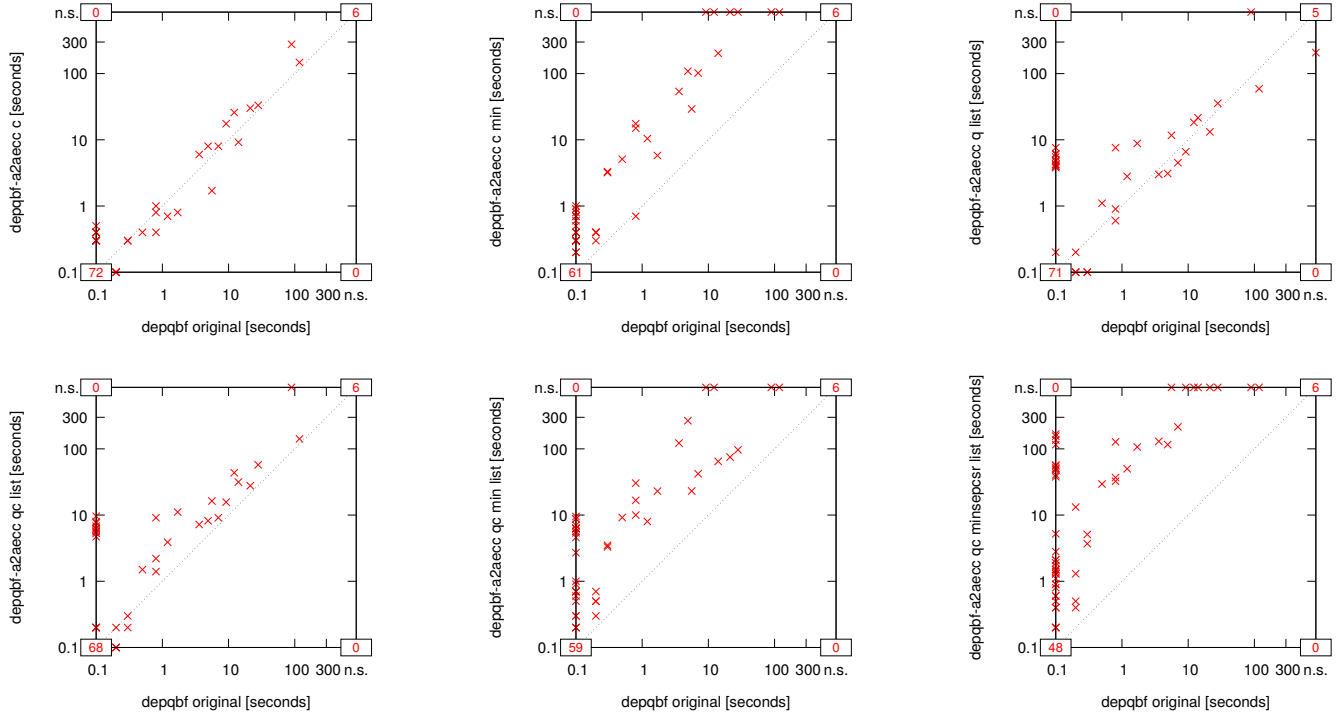


Fig. 633: Suite Castellini ($n = 112$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

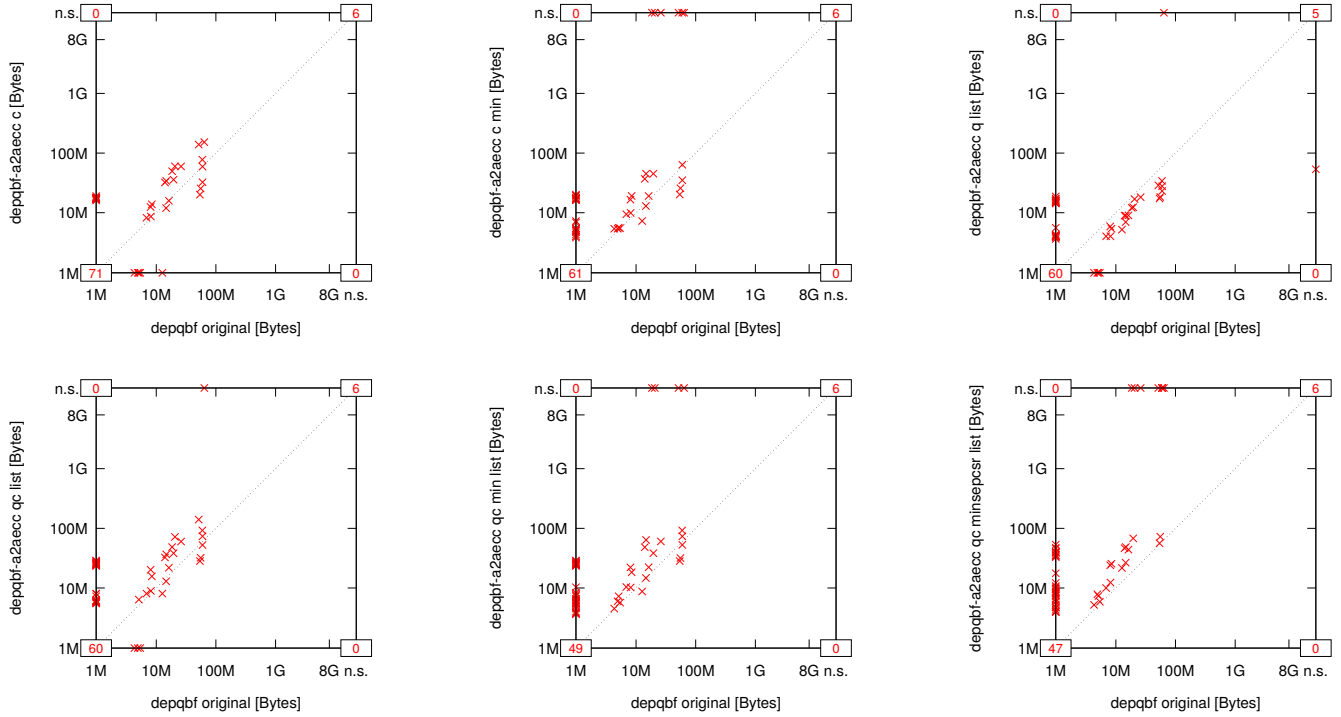


Fig. 634: Suite Castellini ($n = 112$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

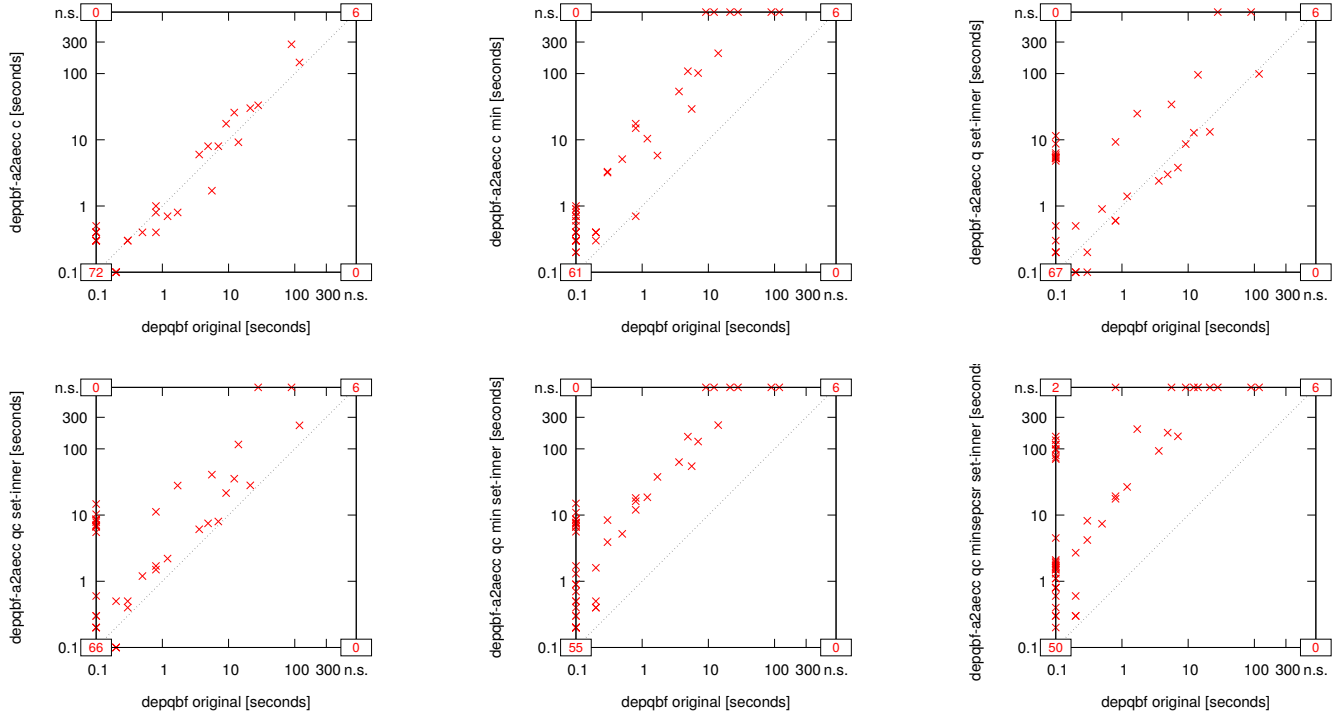


Fig. 635: Suite Castellini ($n = 112$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

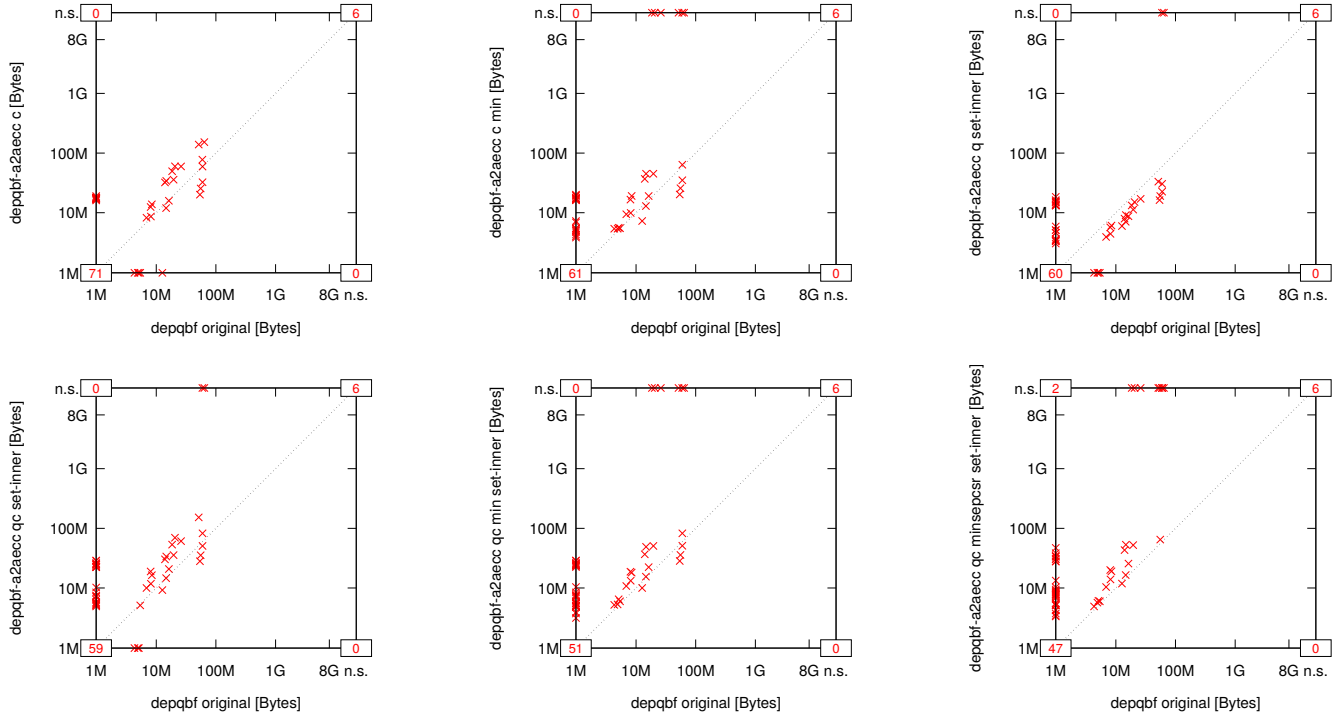


Fig. 636: Suite Castellini ($n = 112$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

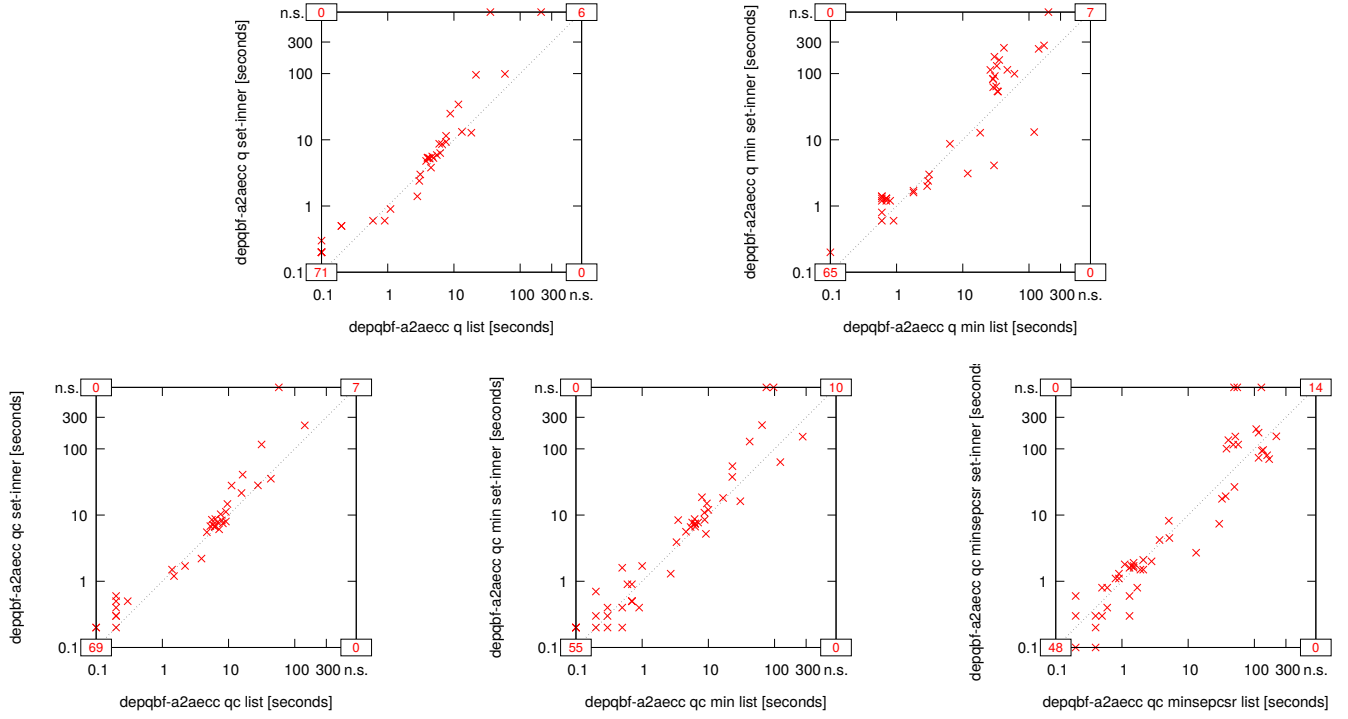


Fig. 637: Suite Castellini ($n = 112$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

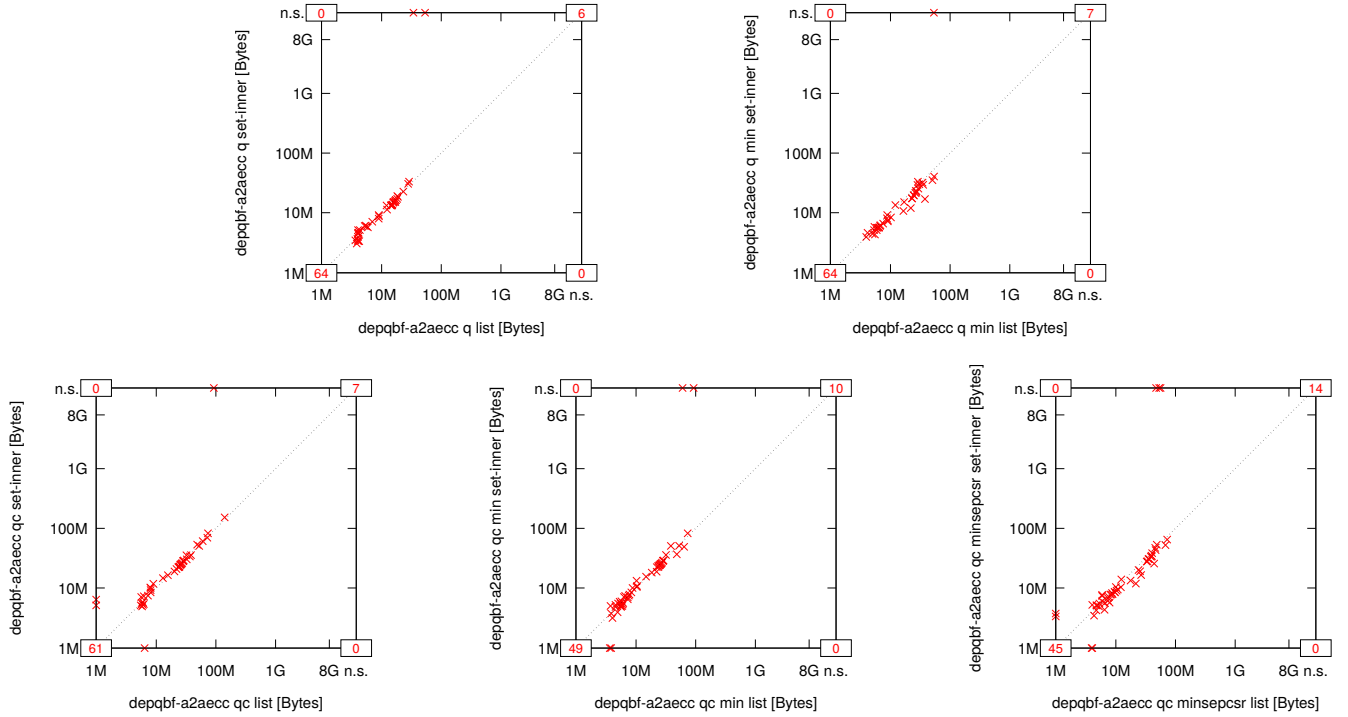


Fig. 638: Suite Castellini ($n = 112$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

10) Chen-Interian ($n = 16$):

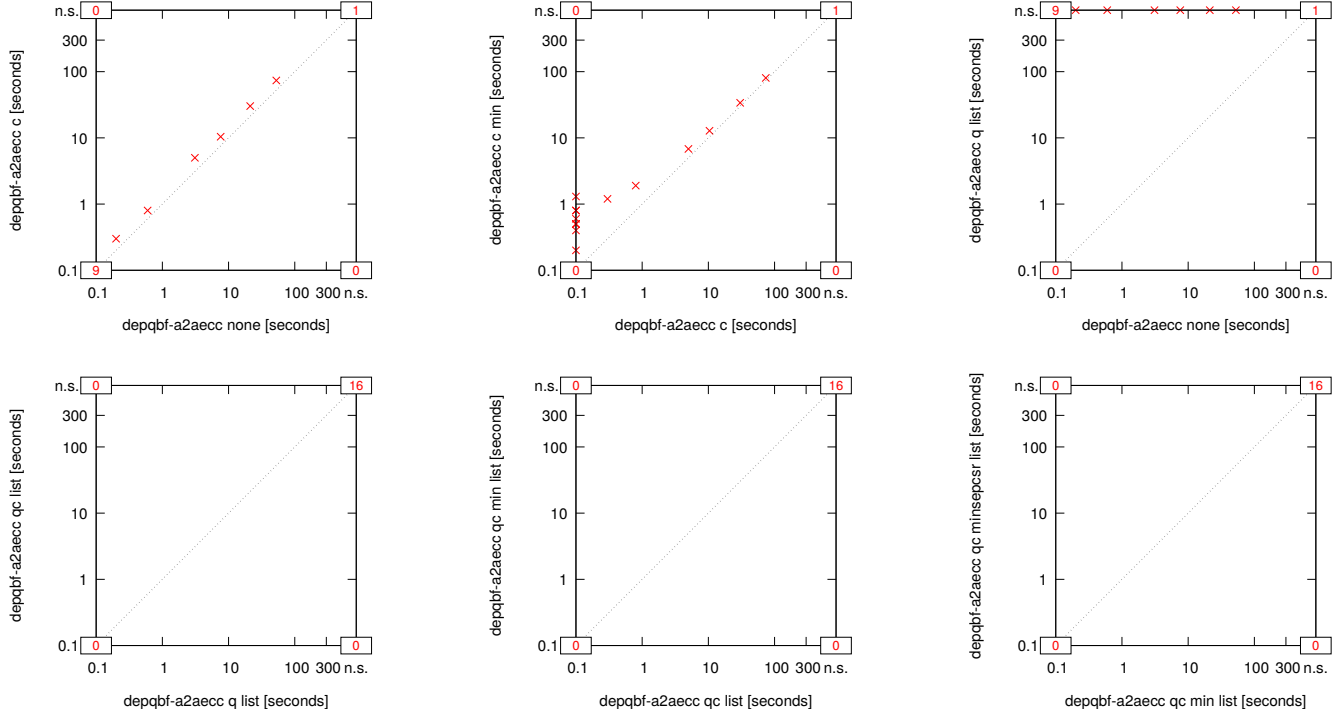


Fig. 639: Suite Chen-Interian ($n = 16$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

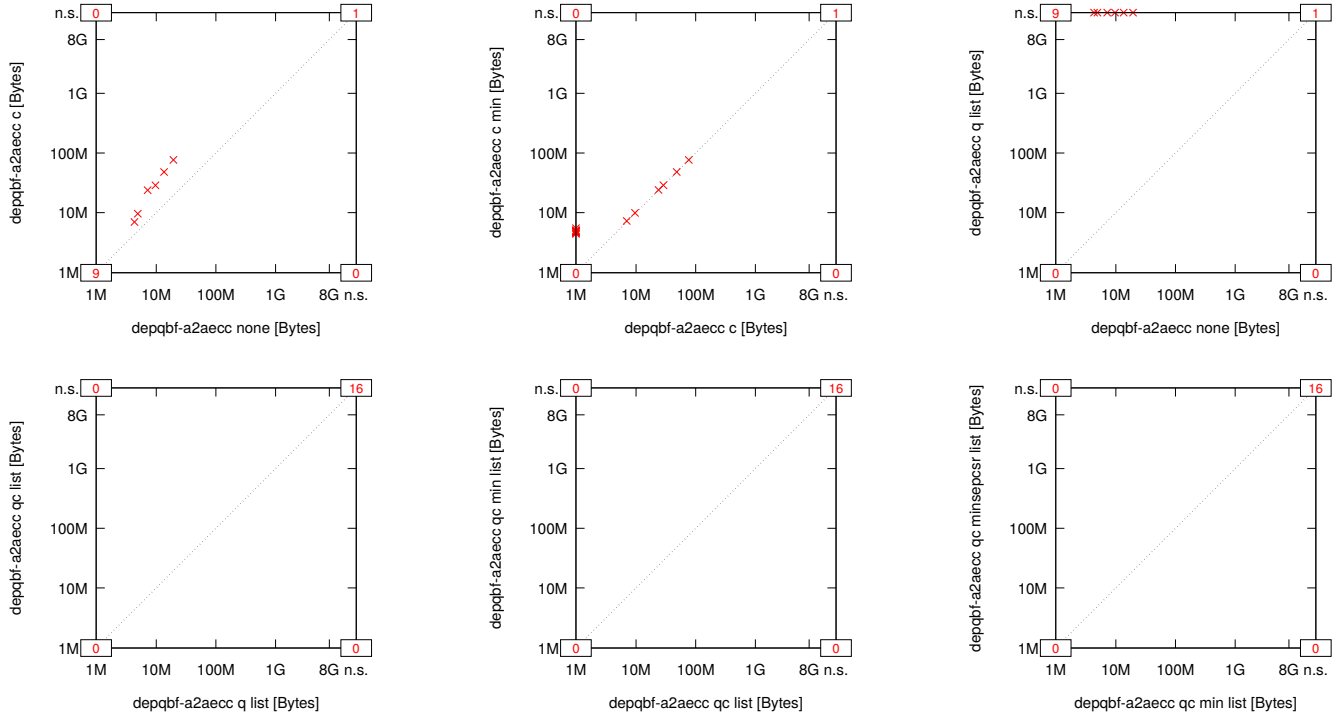


Fig. 640: Suite Chen-Interian ($n = 16$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

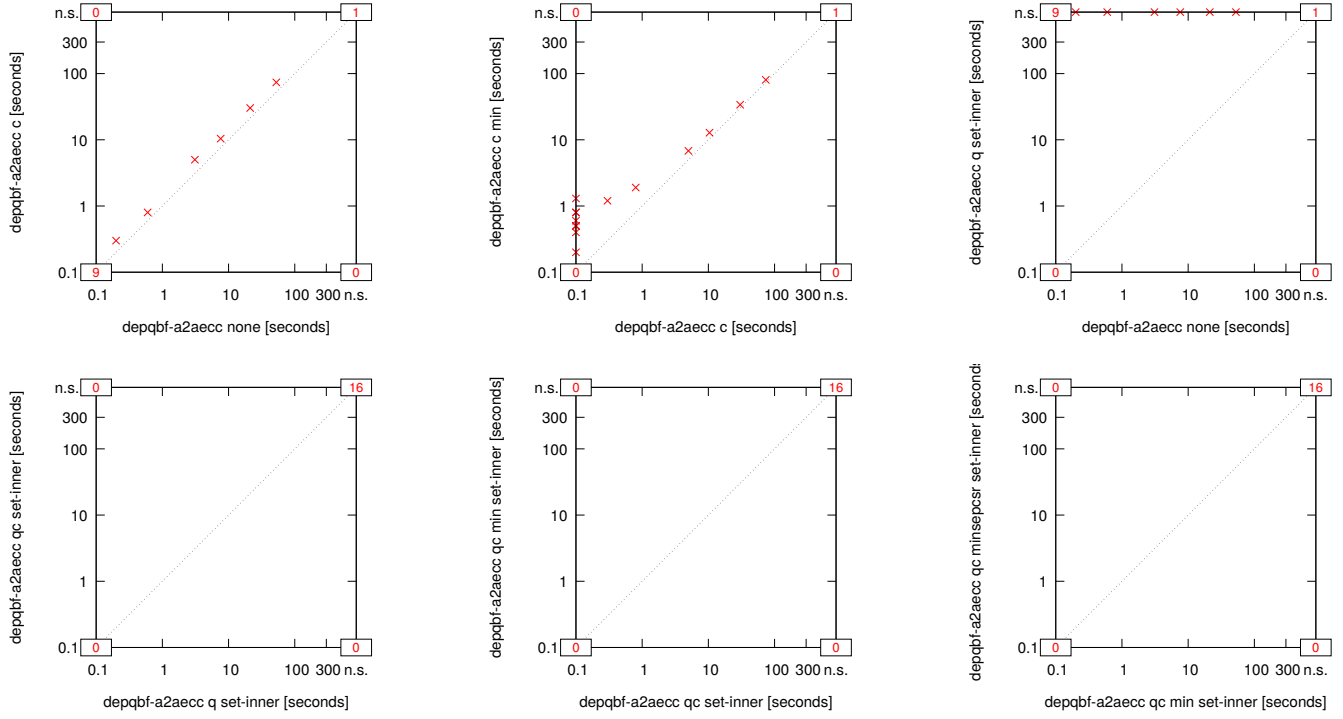


Fig. 641: Suite Chen-Interian ($n = 16$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

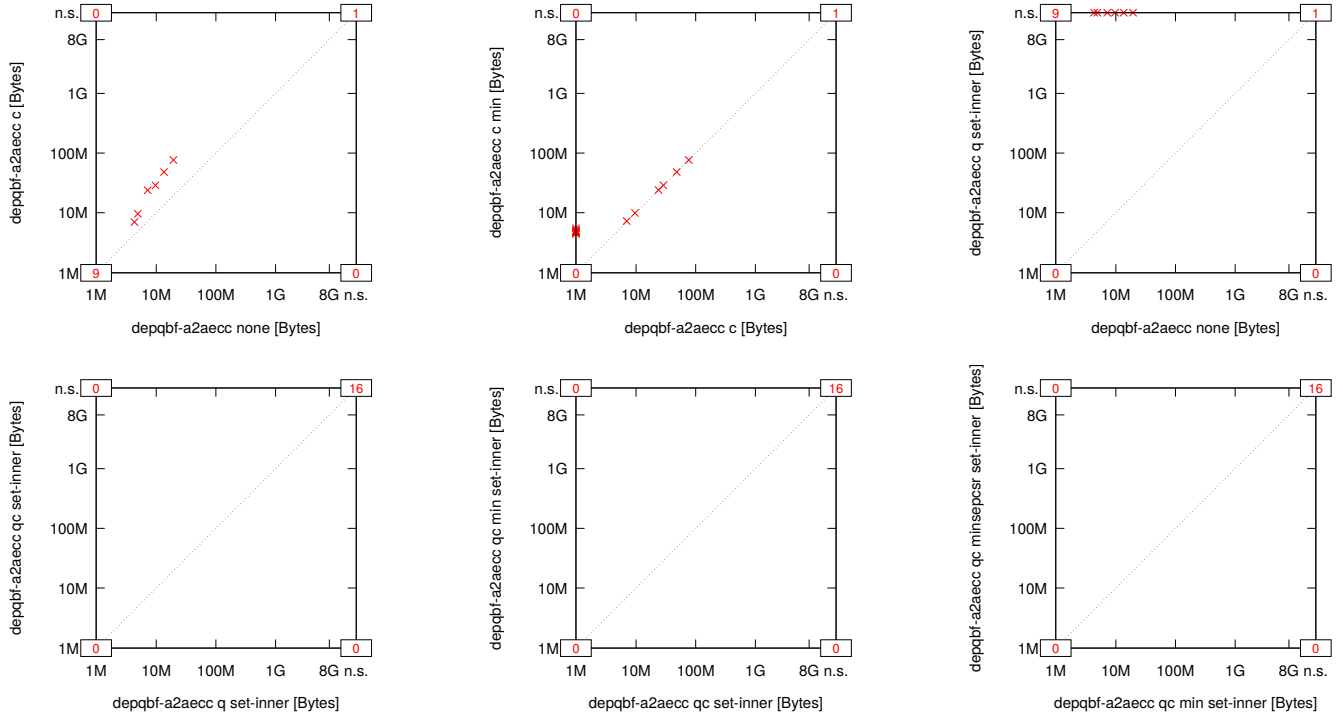


Fig. 642: Suite Chen-Interian ($n = 16$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

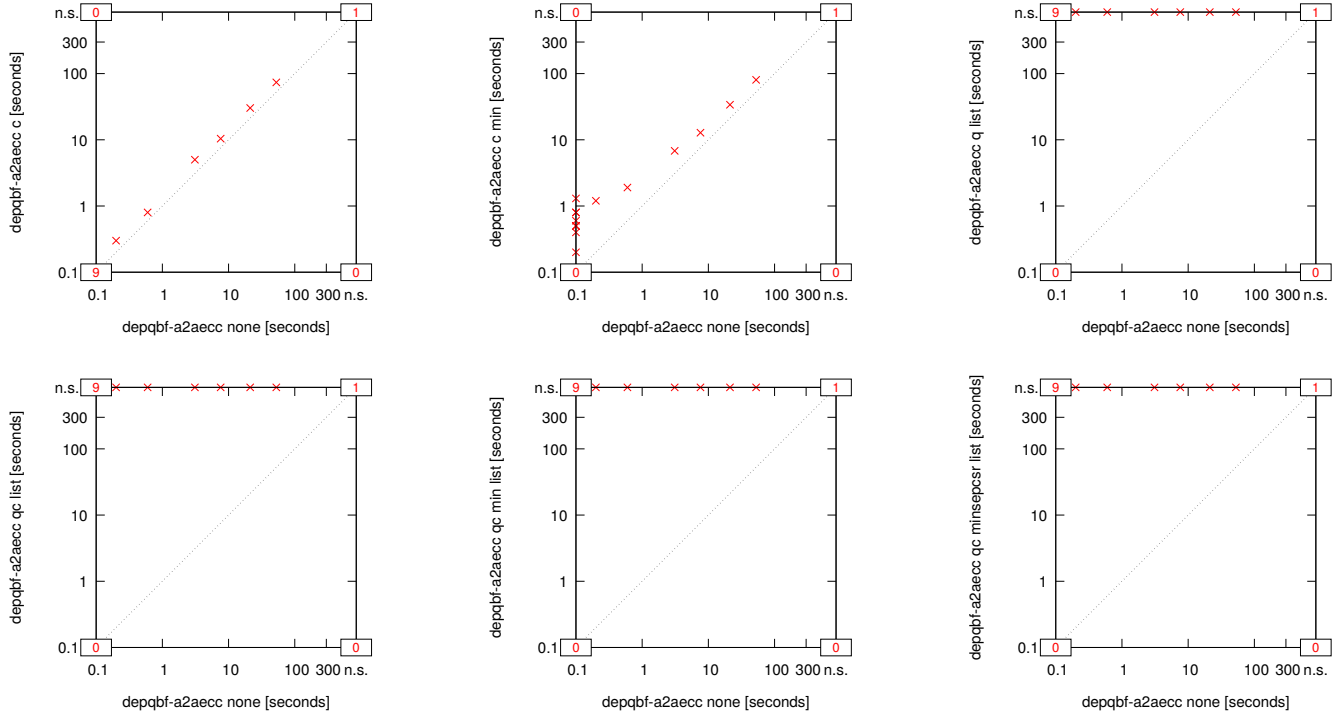


Fig. 643: Suite Chen-Interian ($n = 16$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

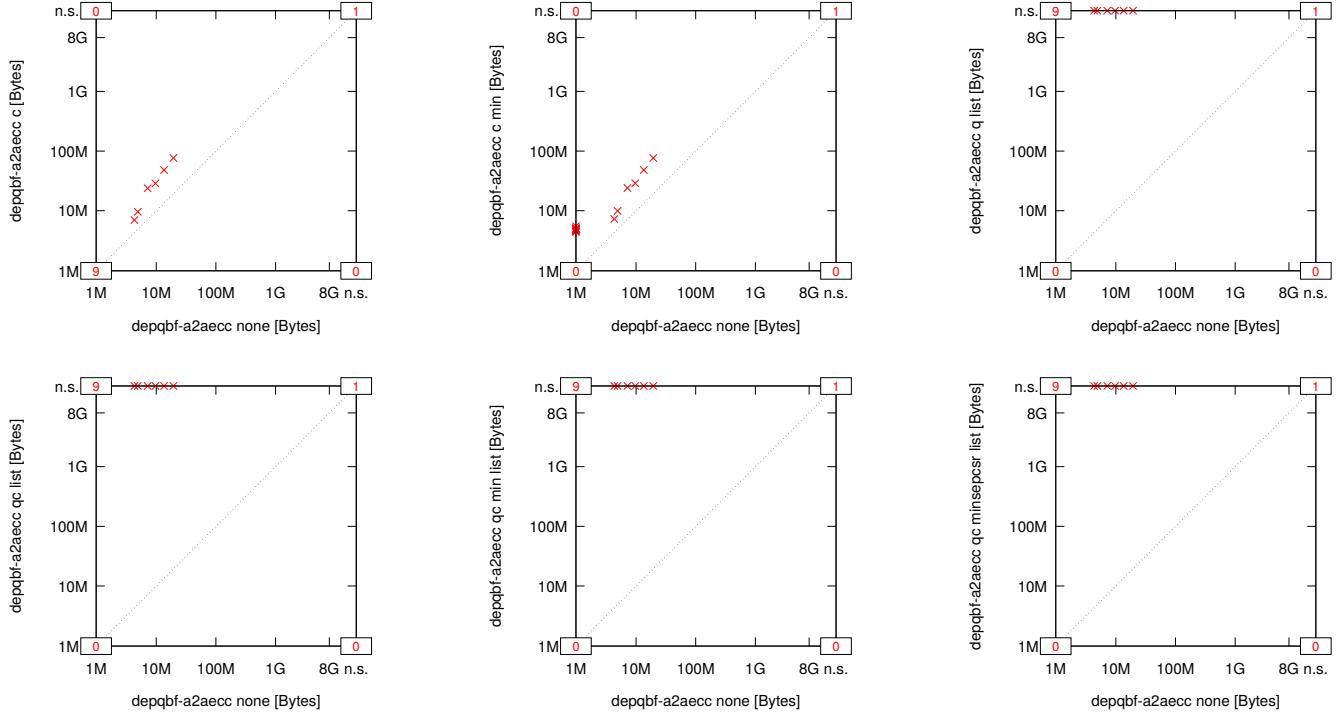


Fig. 644: Suite Chen-Interian ($n = 16$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

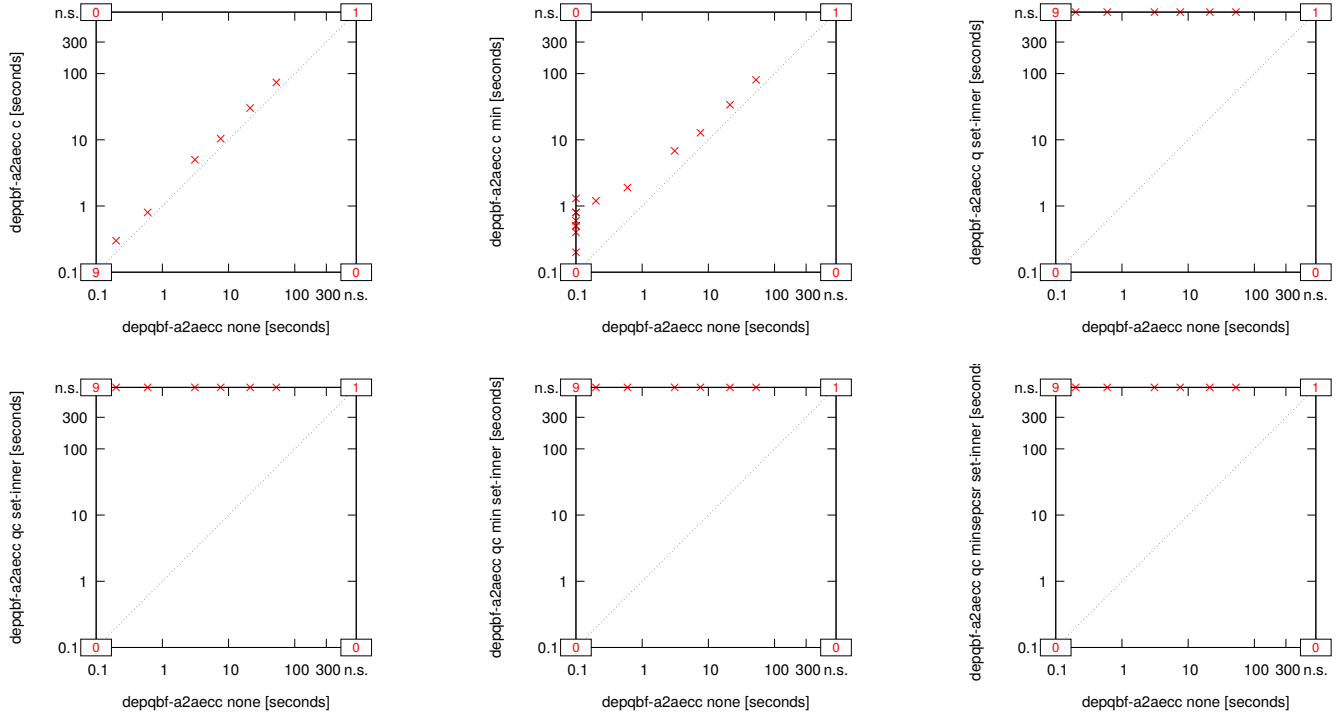


Fig. 645: Suite Chen-Interian ($n = 16$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

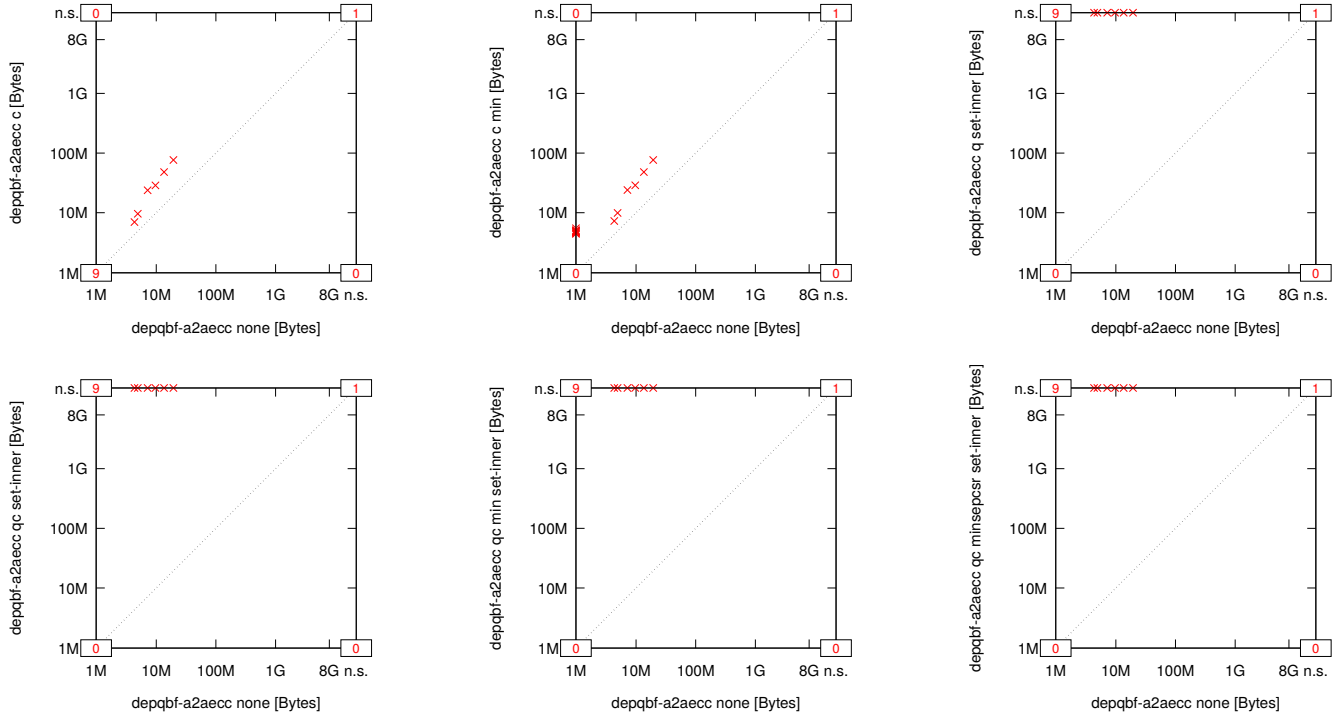


Fig. 646: Suite Chen-Interian ($n = 16$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

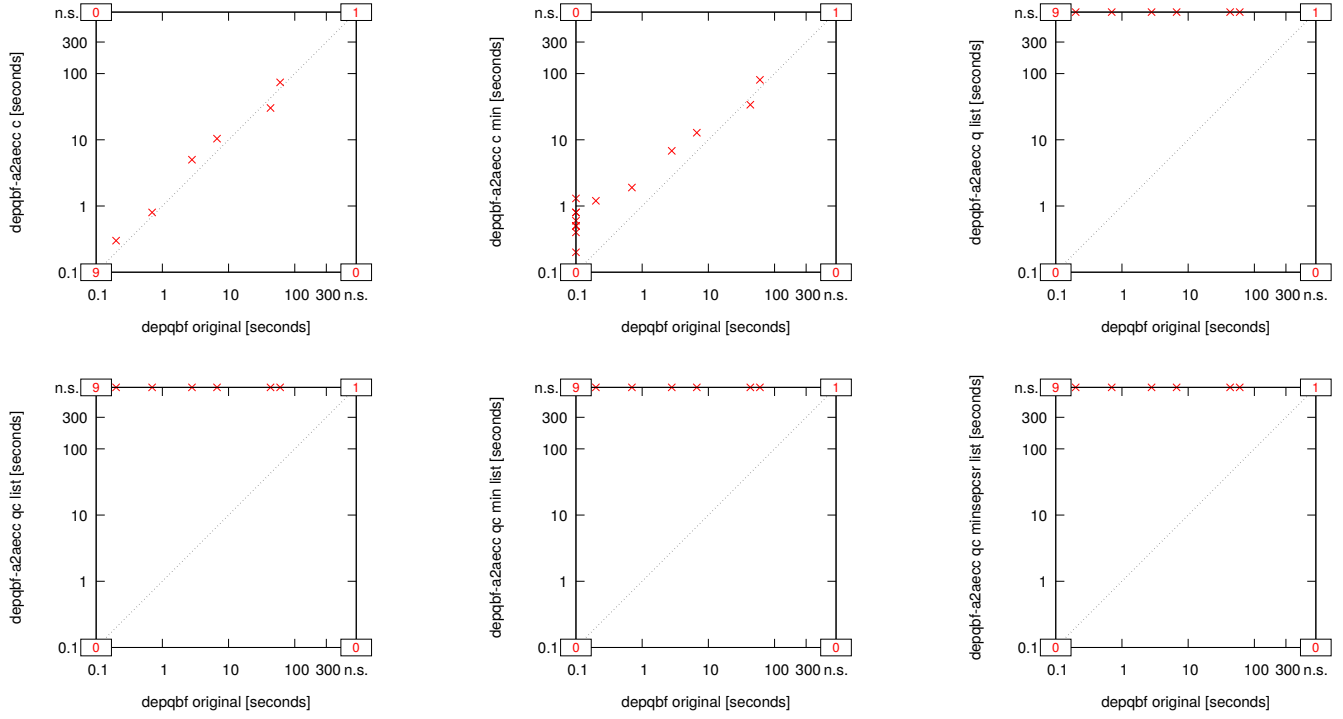


Fig. 647: Suite Chen-Interian ($n = 16$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

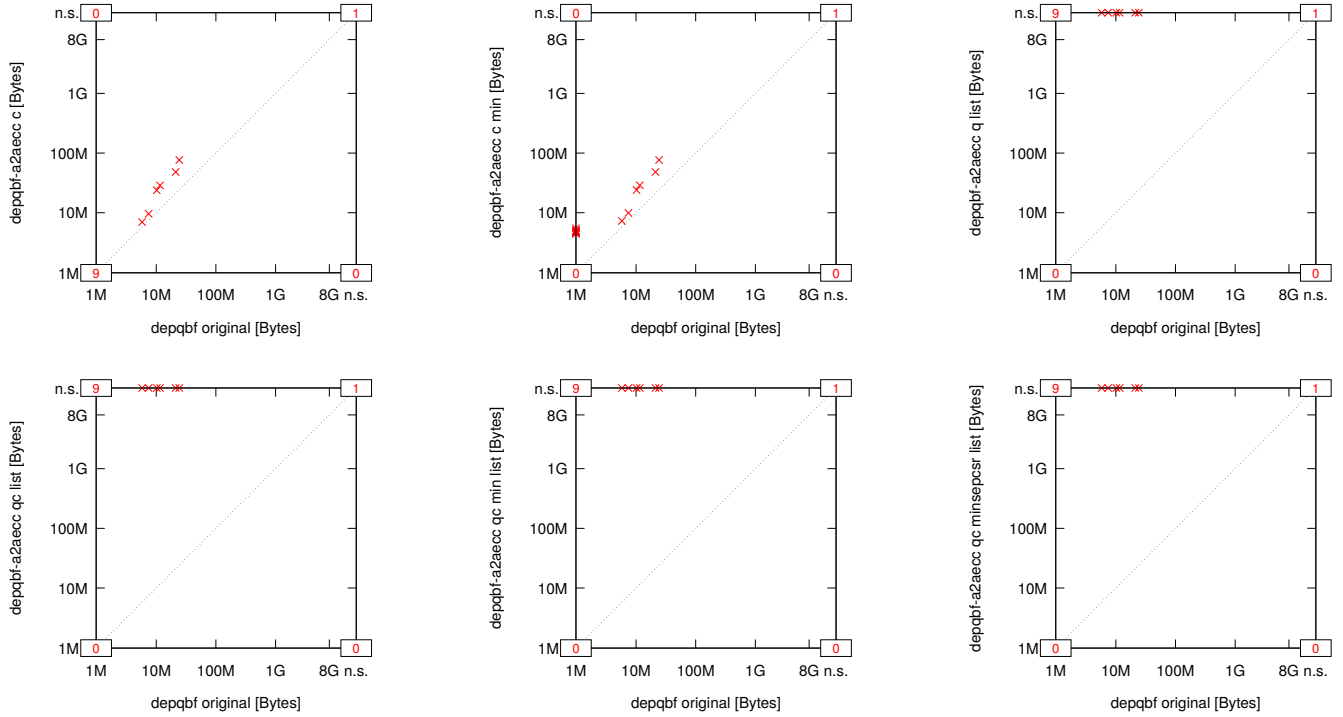


Fig. 648: Suite Chen-Interian ($n = 16$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

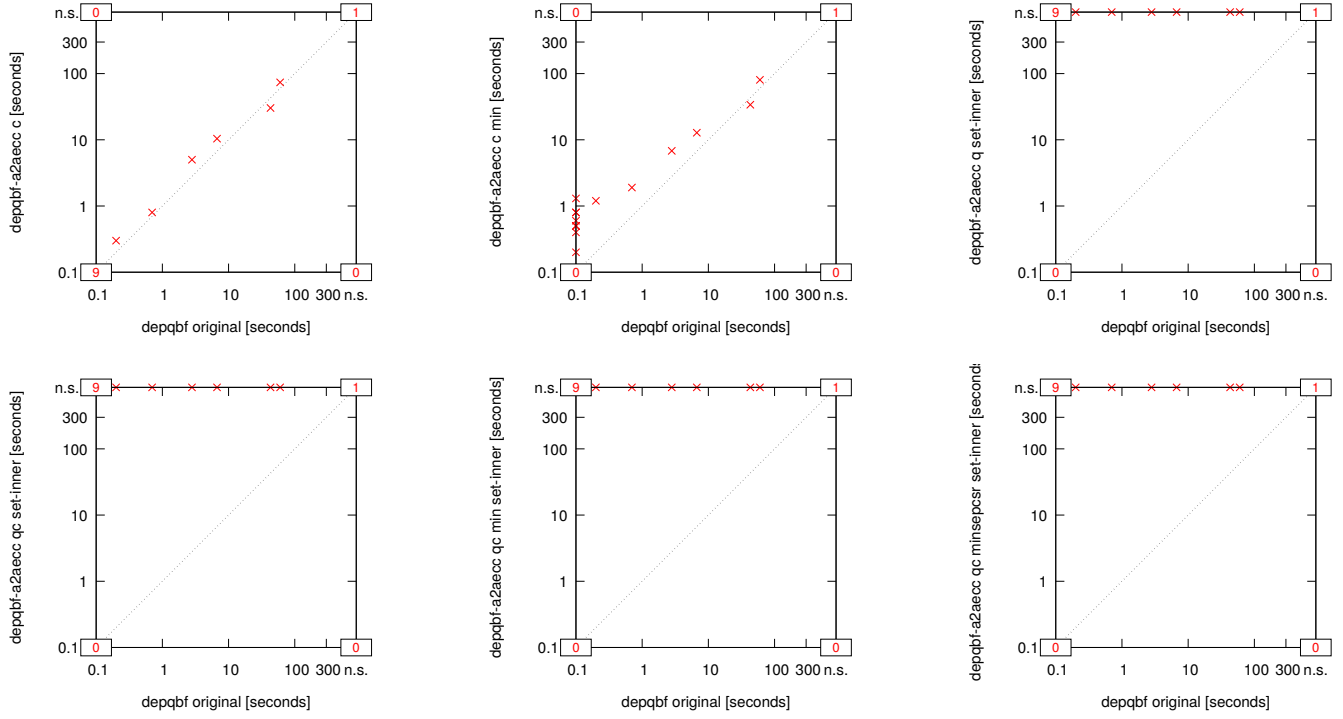


Fig. 649: Suite Chen-Interian ($n = 16$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

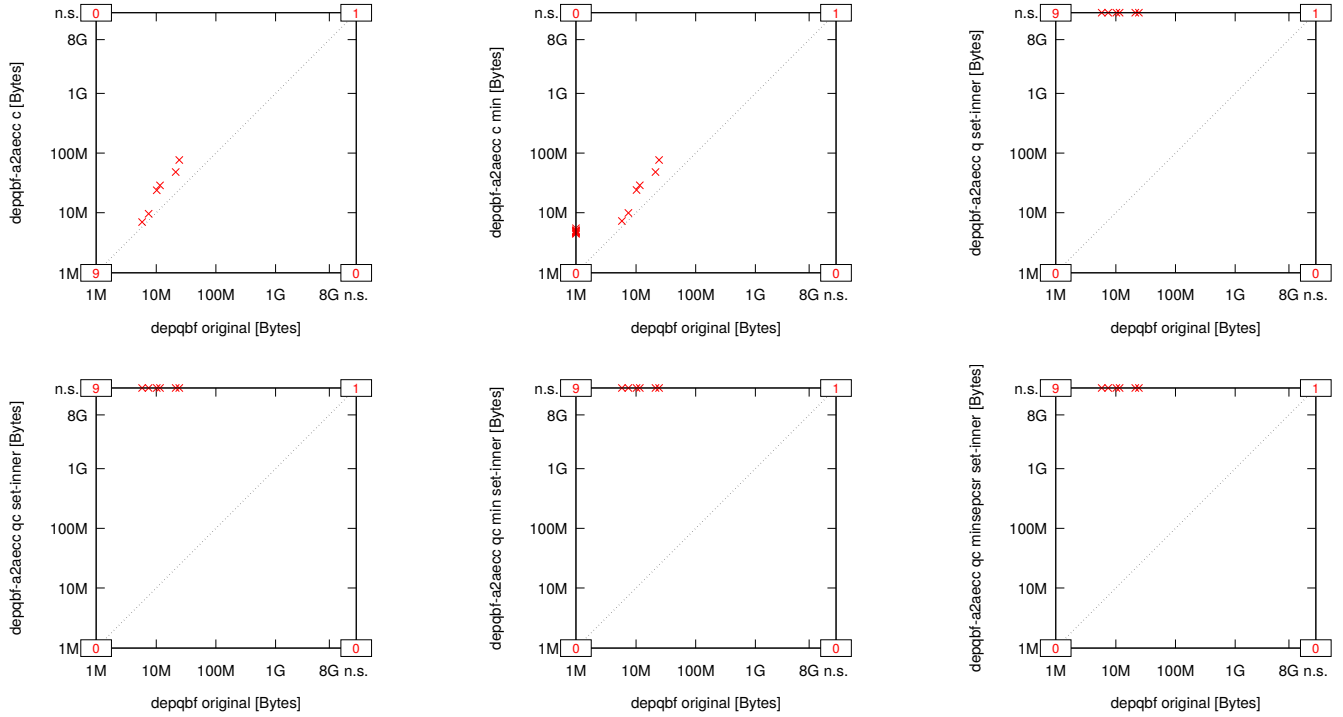


Fig. 650: Suite Chen-Interian ($n = 16$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

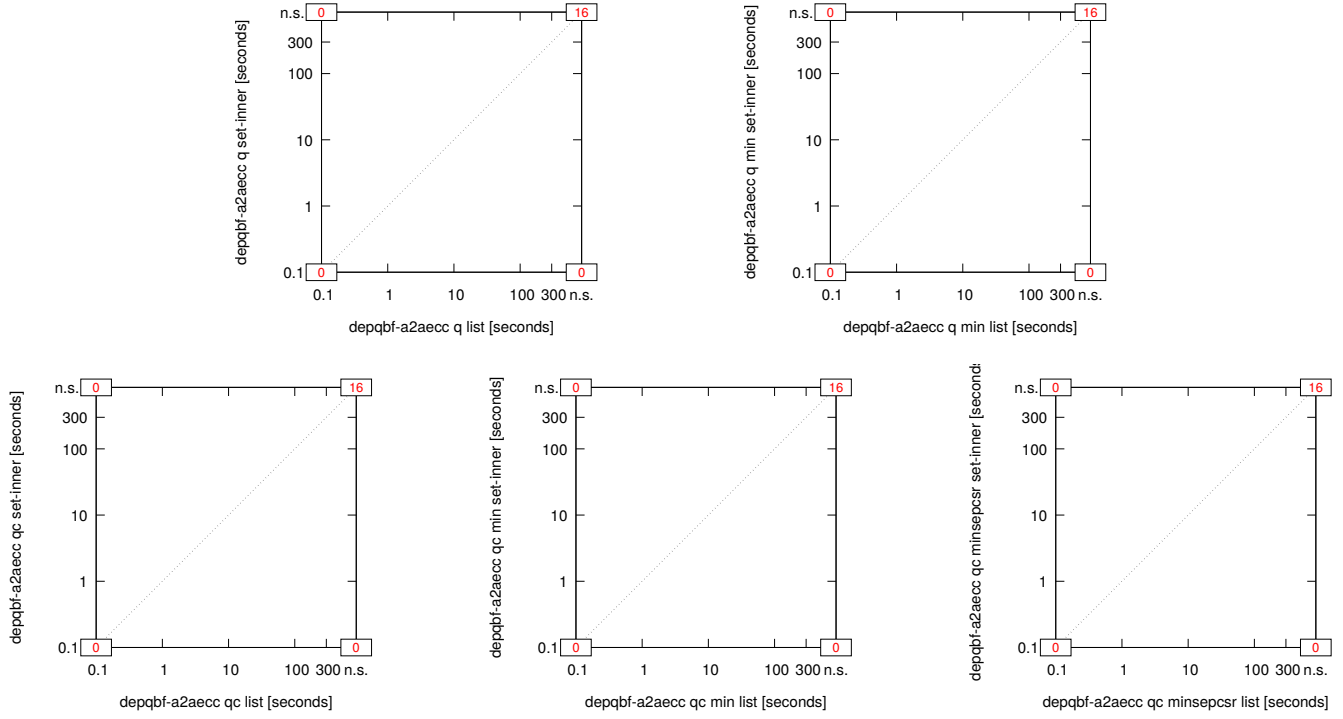


Fig. 651: Suite Chen-Interian ($n = 16$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

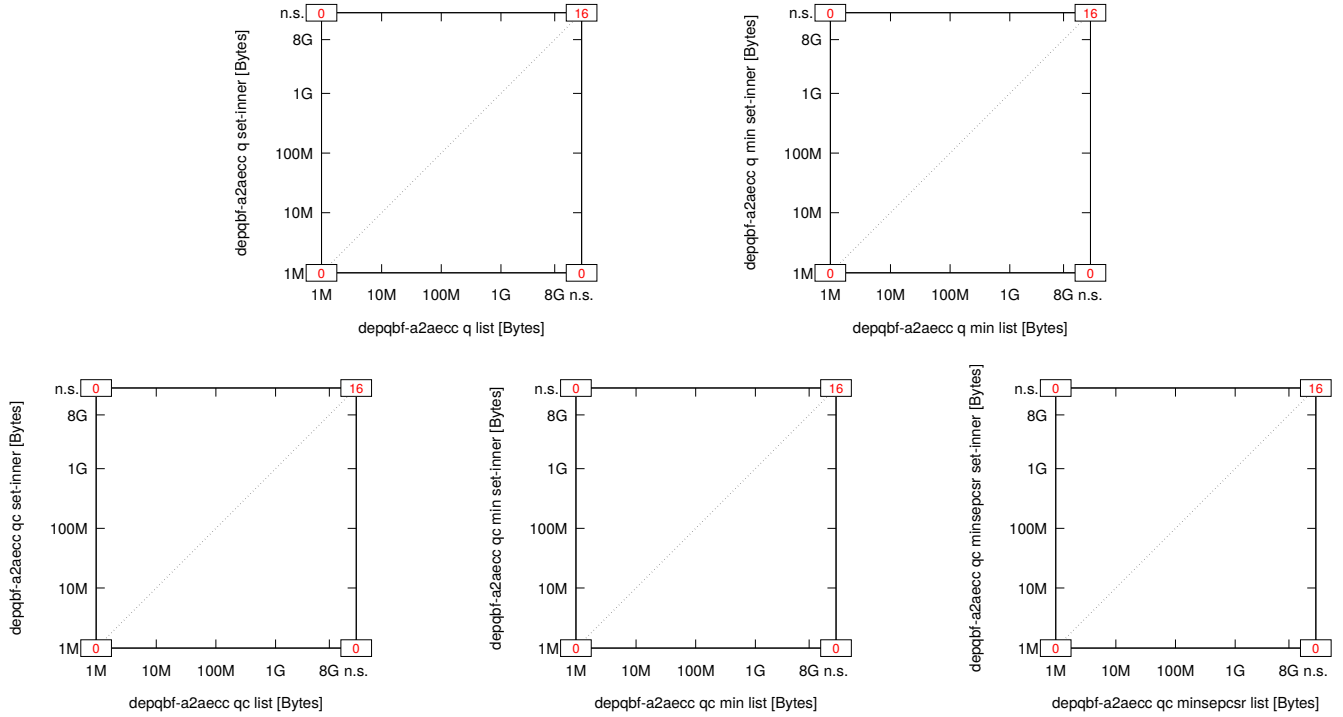


Fig. 652: Suite Chen-Interian ($n = 16$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

11) *Diptarama-Jordan-Shinohara* ($n = 11$):

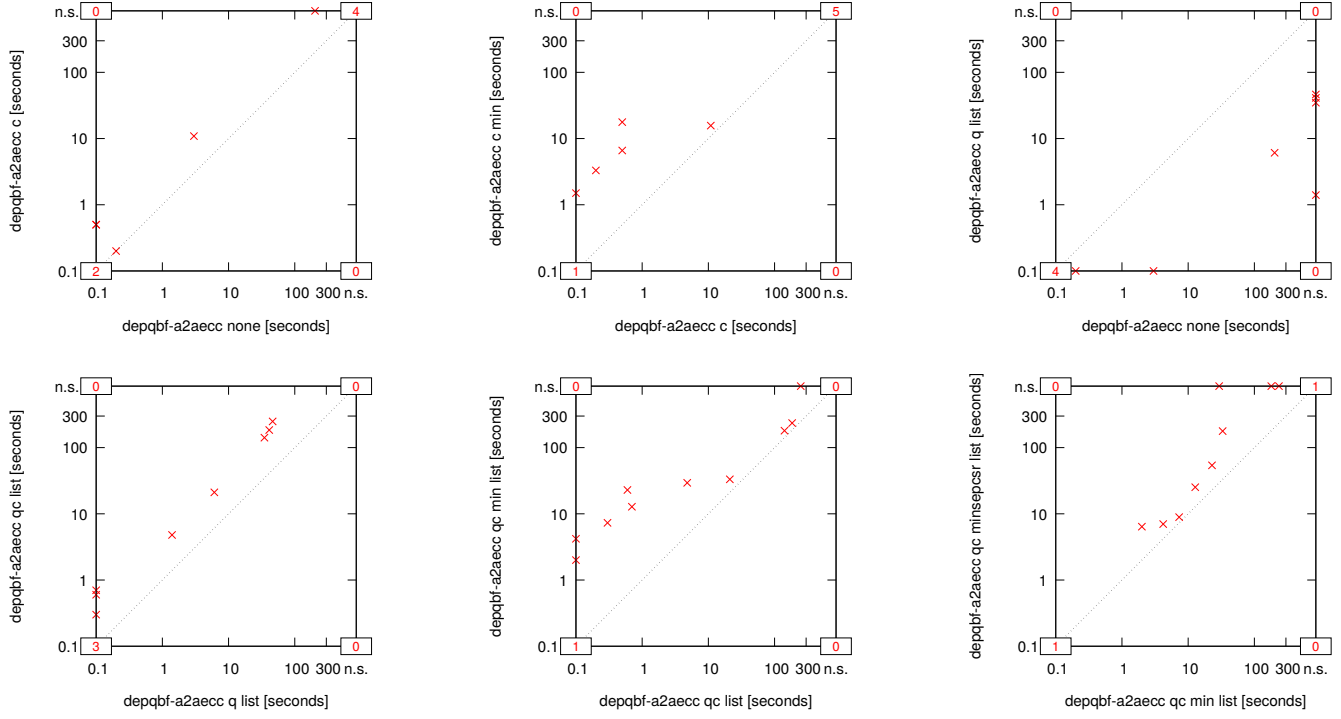


Fig. 653: Suite Diptarama-Jordan-Shinohara ($n = 11$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

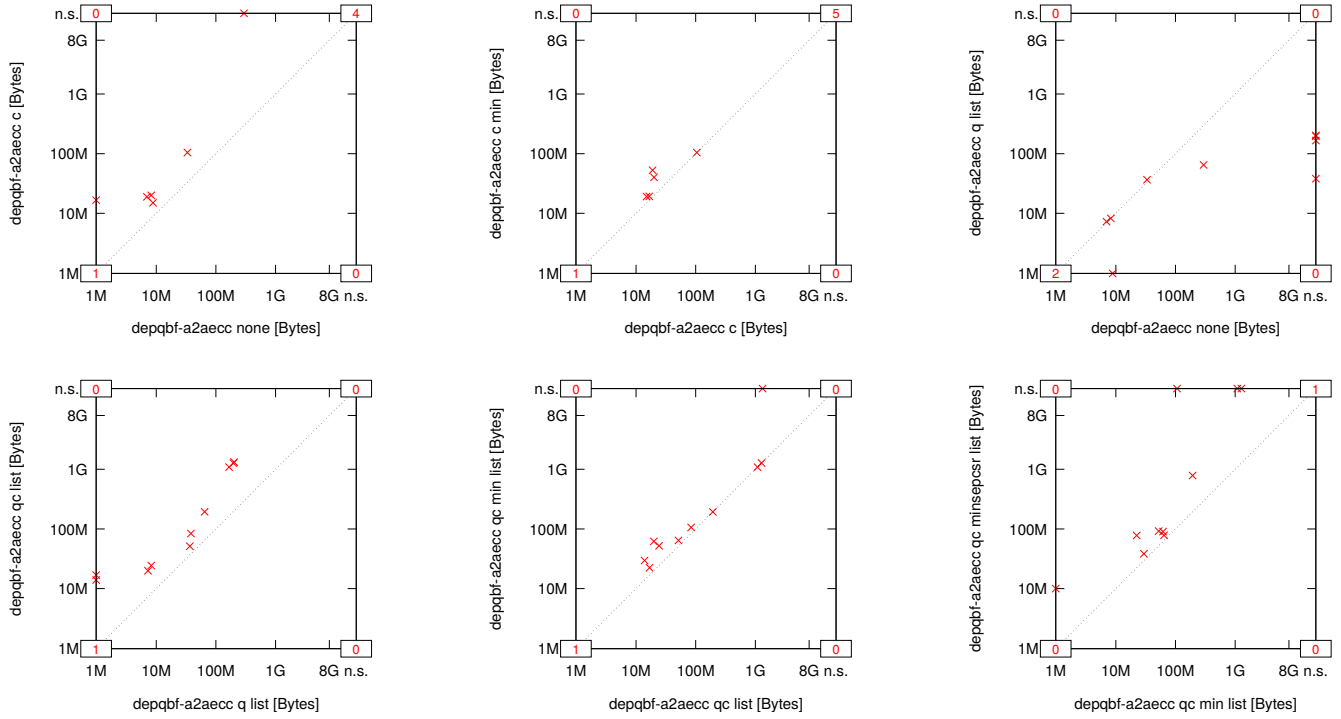


Fig. 654: Suite Diptarama-Jordan-Shinohara ($n = 11$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

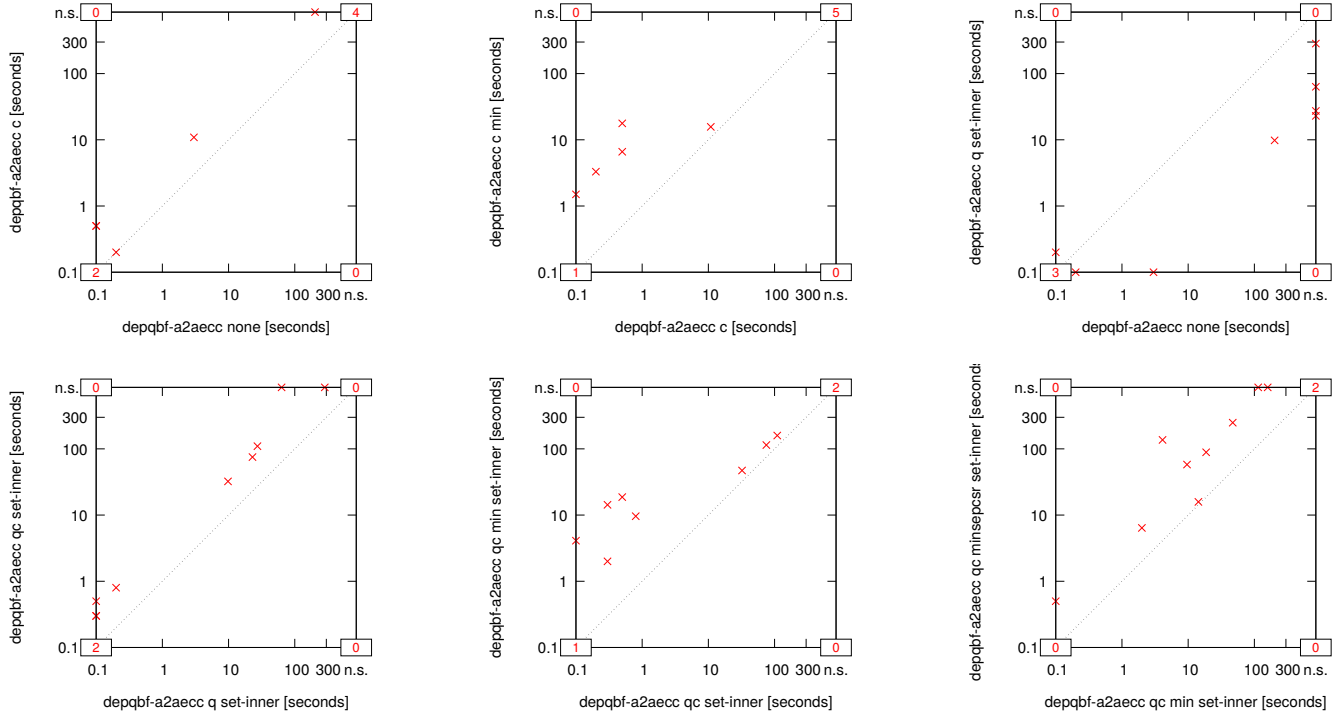


Fig. 655: Suite Diptarama-Jordan-Shinohara ($n = 11$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

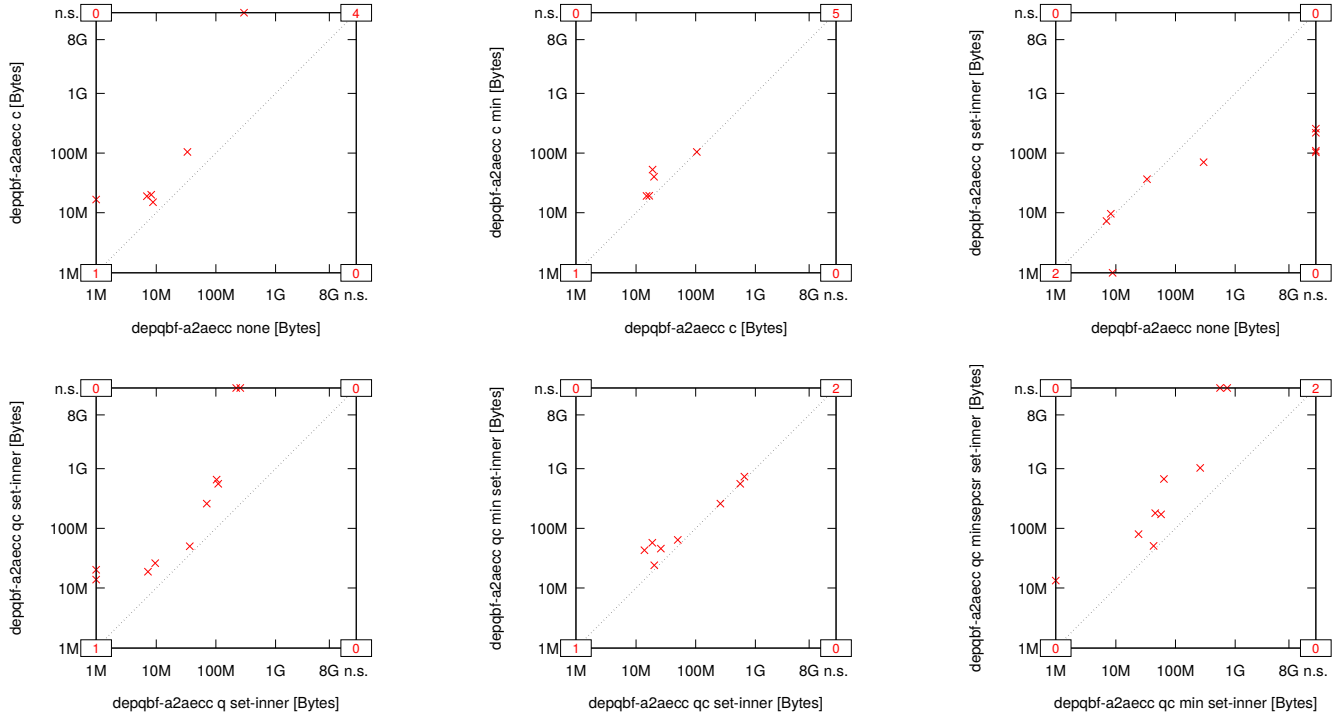


Fig. 656: Suite Diptarama-Jordan-Shinohara ($n = 11$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

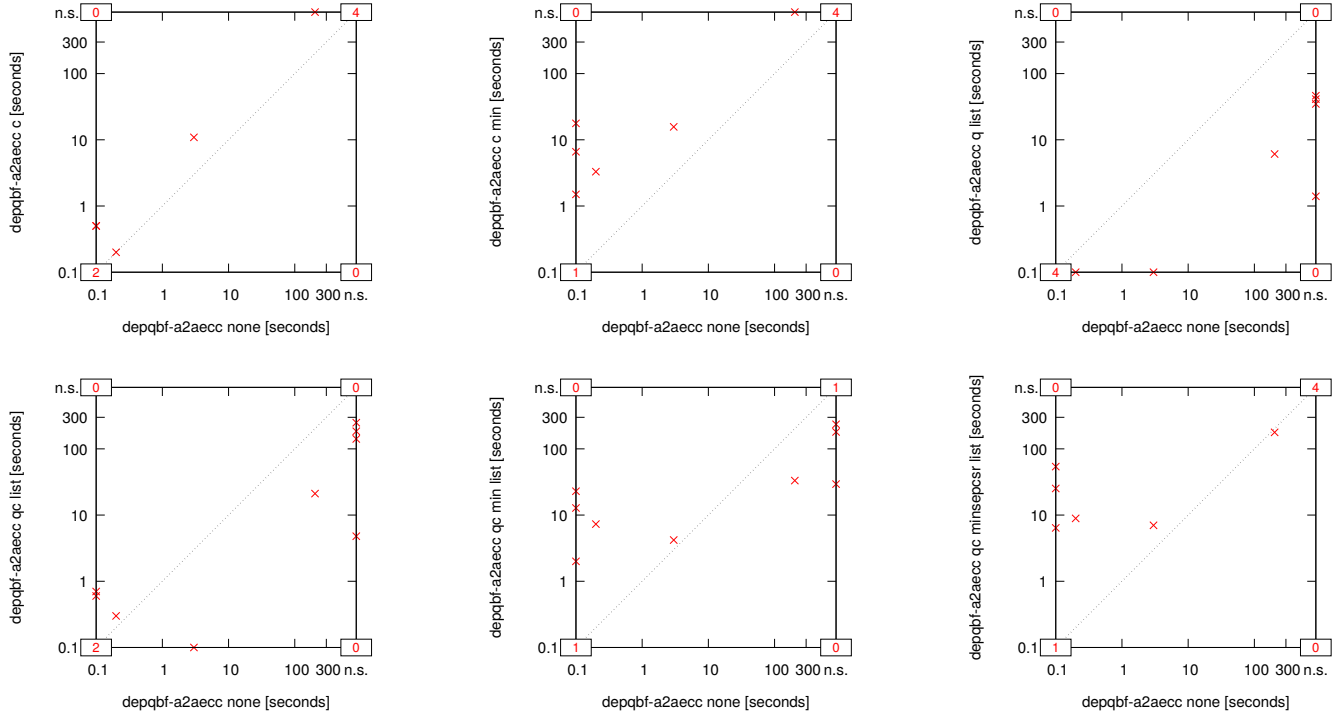


Fig. 657: Suite Diptarama-Jordan-Shinohara ($n = 11$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

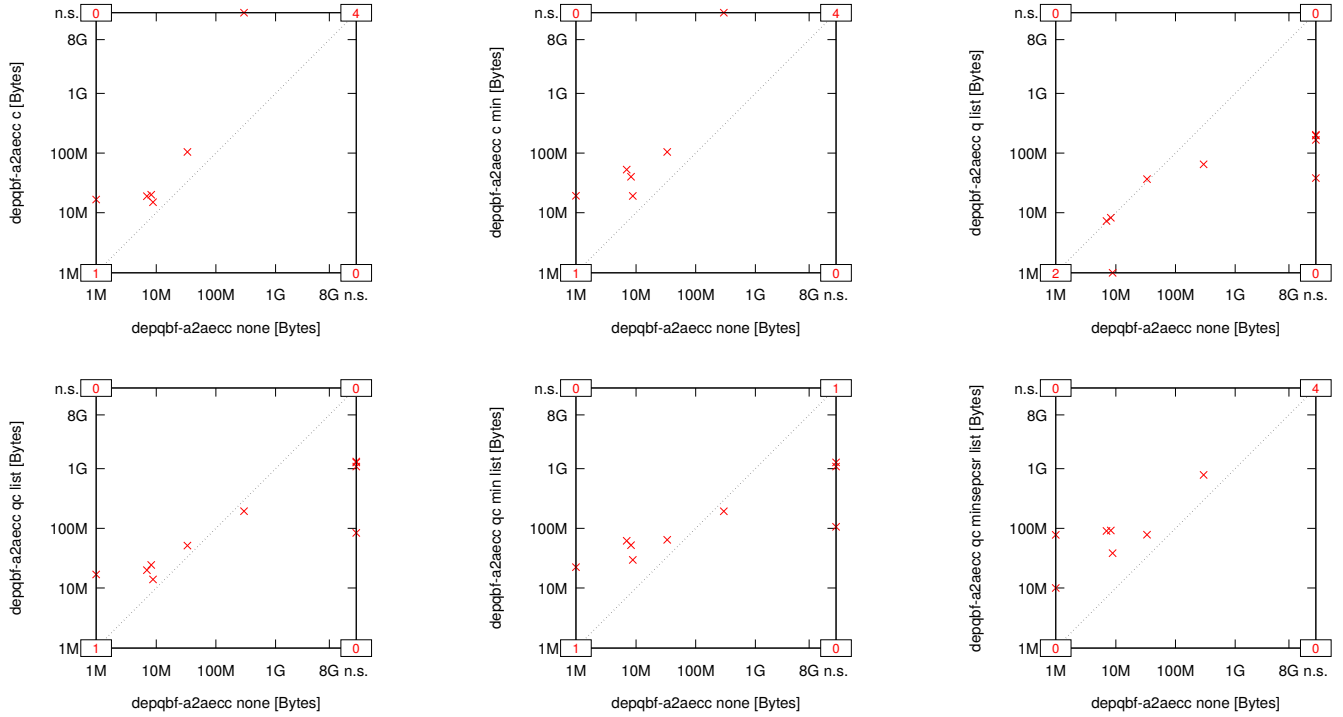


Fig. 658: Suite Diptarama-Jordan-Shinohara ($n = 11$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

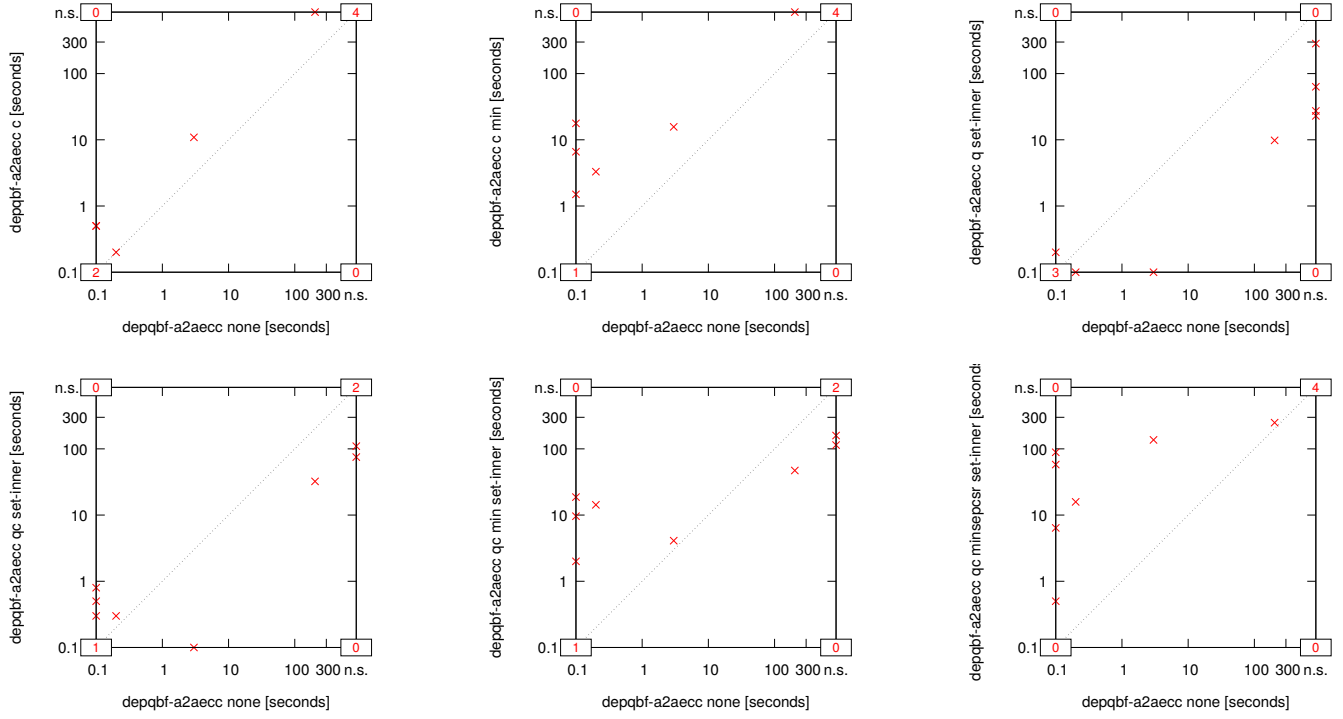


Fig. 659: Suite Diptarama-Jordan-Shinohara ($n = 11$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

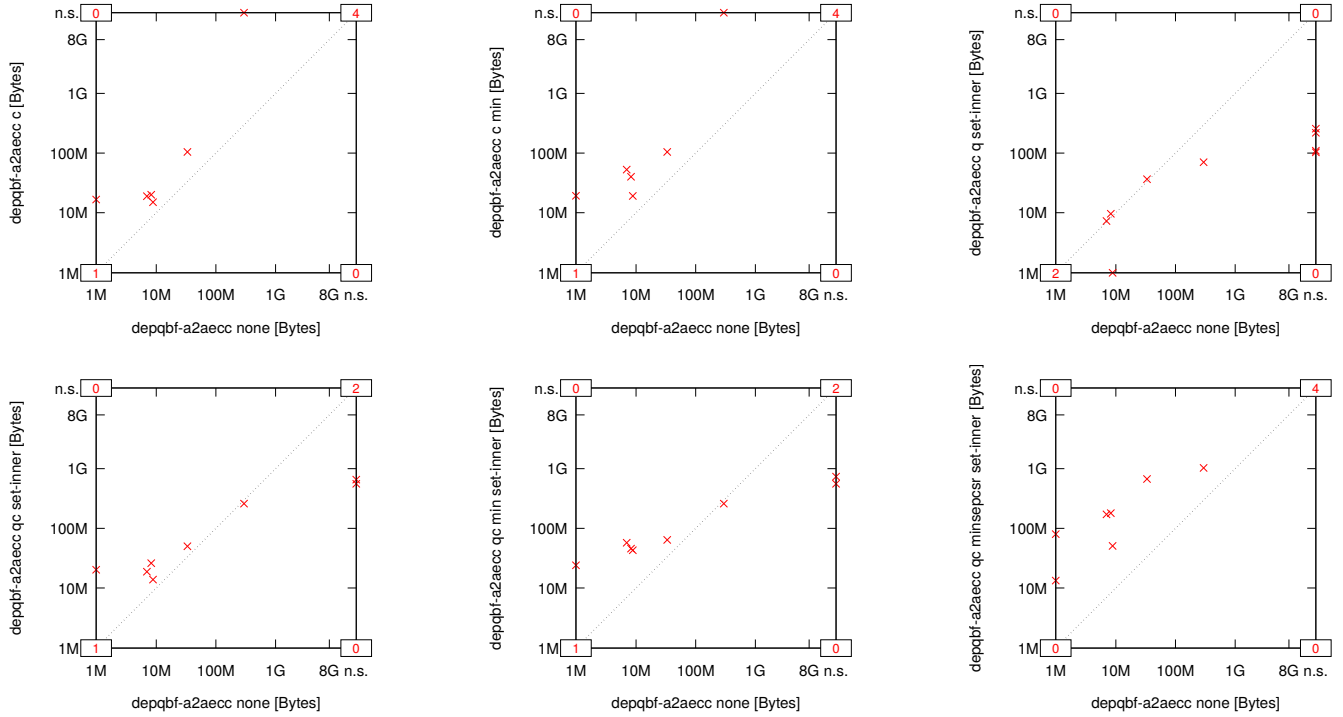


Fig. 660: Suite Diptarama-Jordan-Shinohara ($n = 11$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

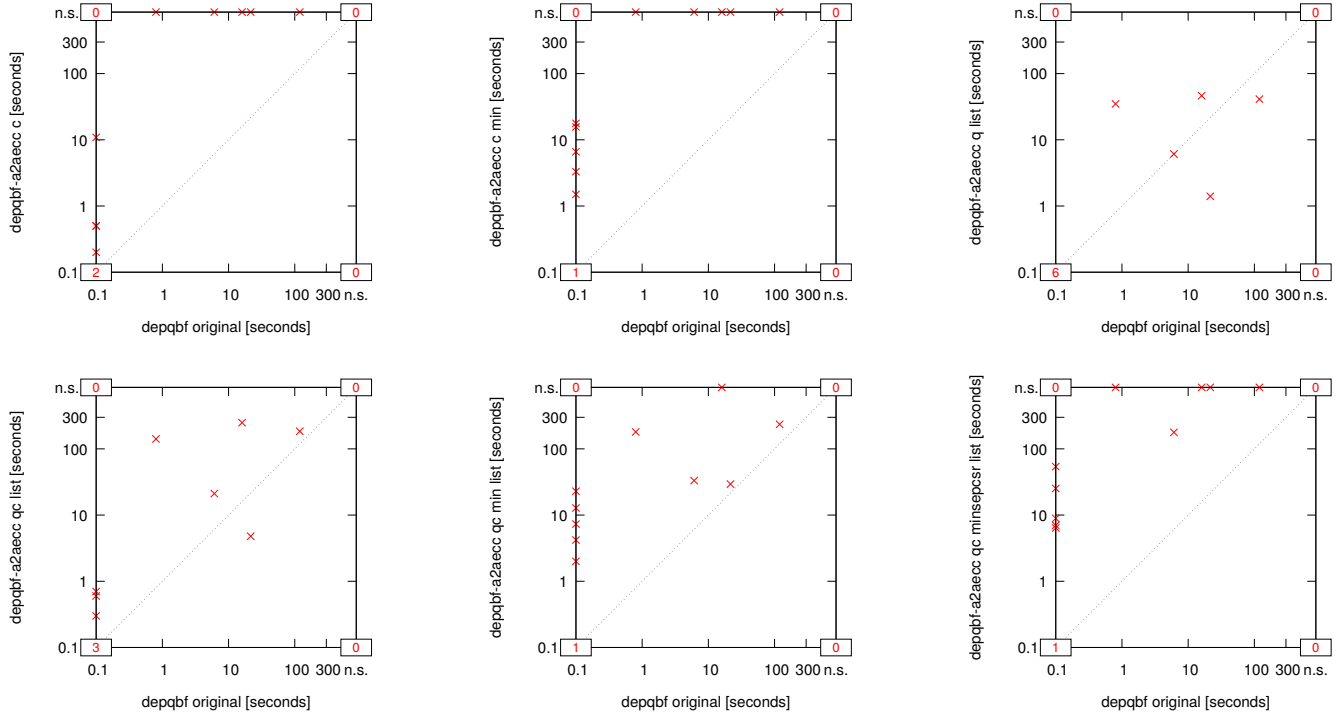


Fig. 661: Suite Diptarama-Jordan-Shinohara ($n = 11$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

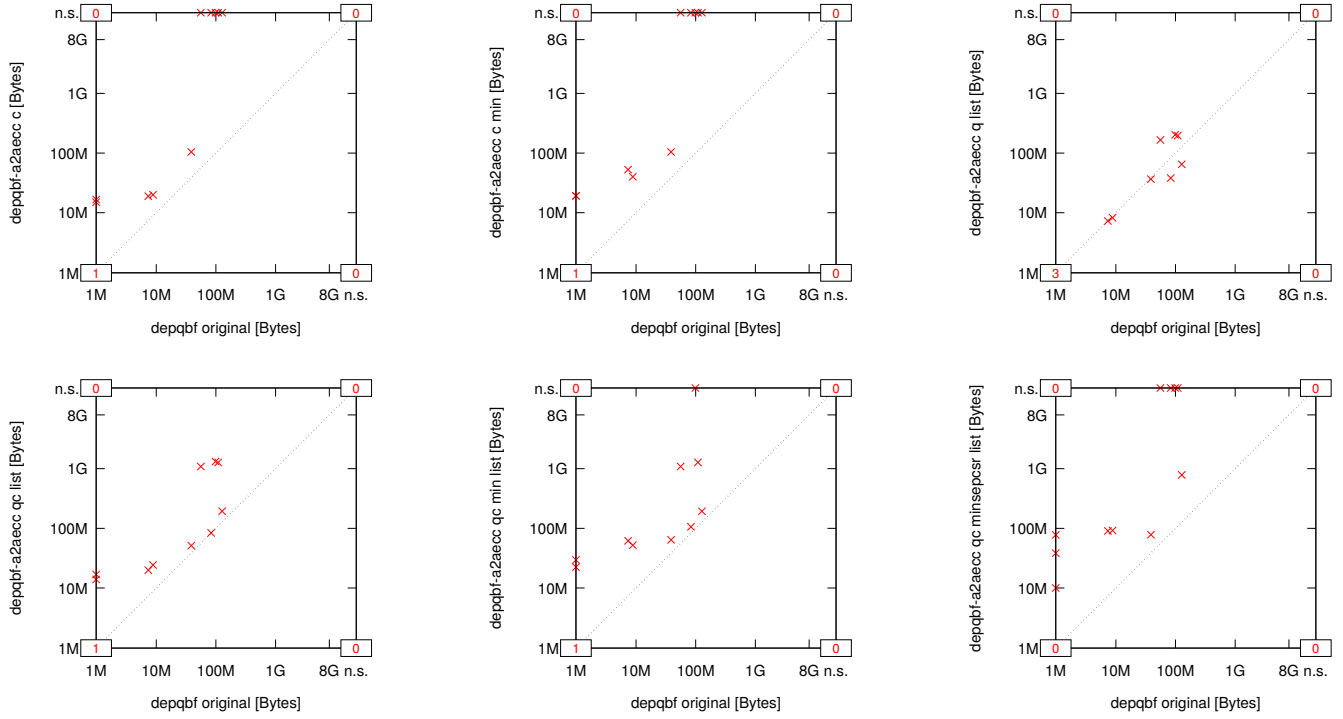


Fig. 662: Suite Diptarama-Jordan-Shinohara ($n = 11$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

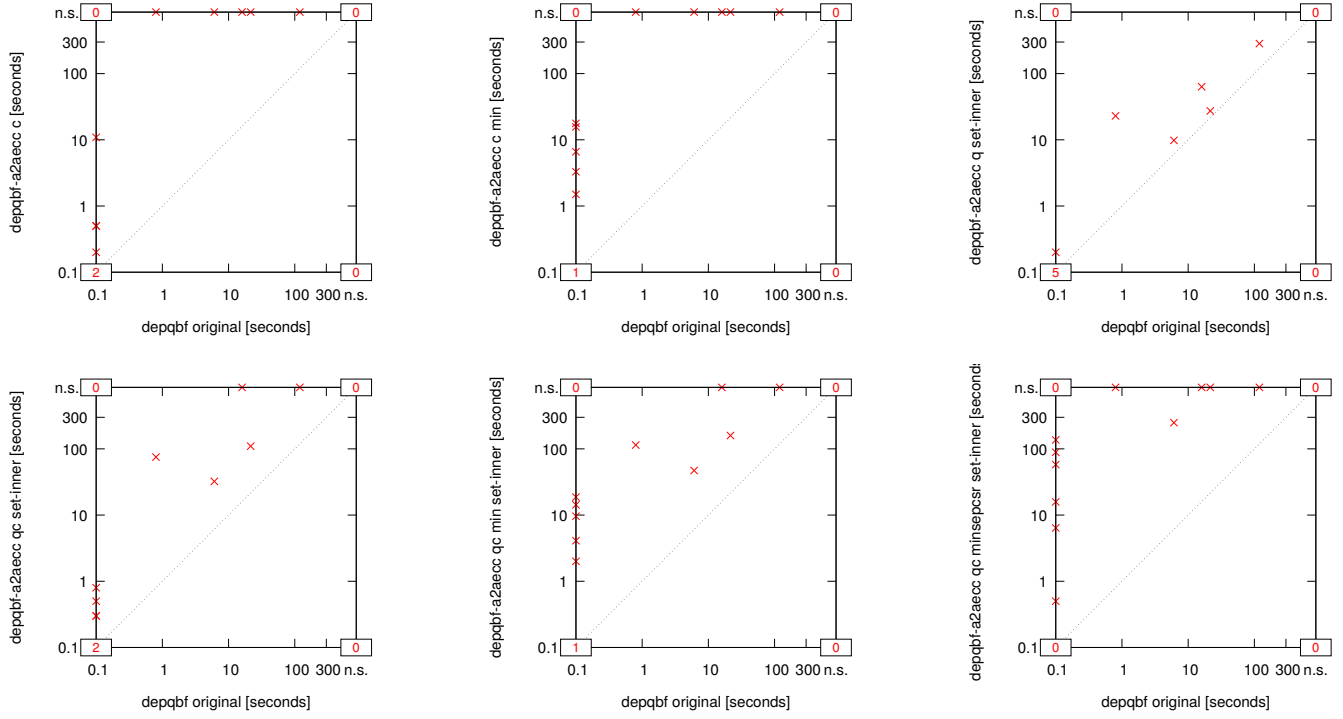


Fig. 663: Suite Diptarama-Jordan-Shinohara ($n = 11$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

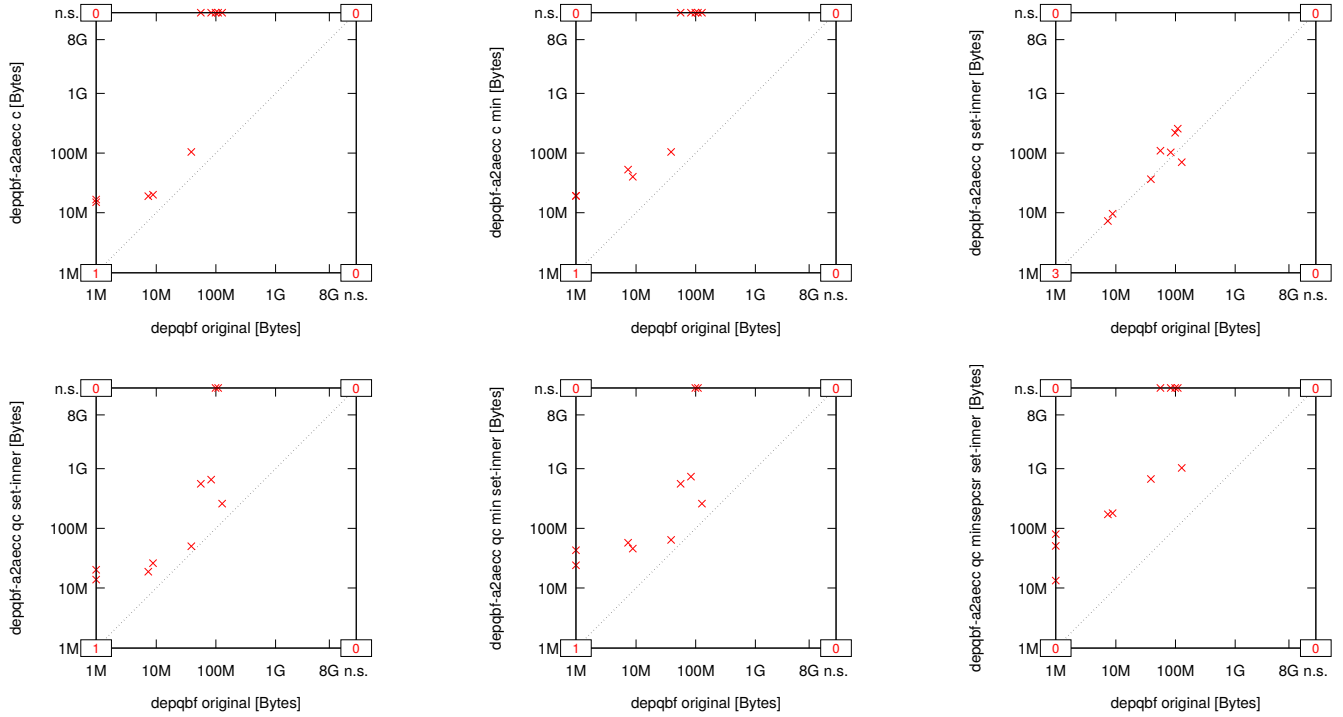


Fig. 664: Suite Diptarama-Jordan-Shinohara ($n = 11$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

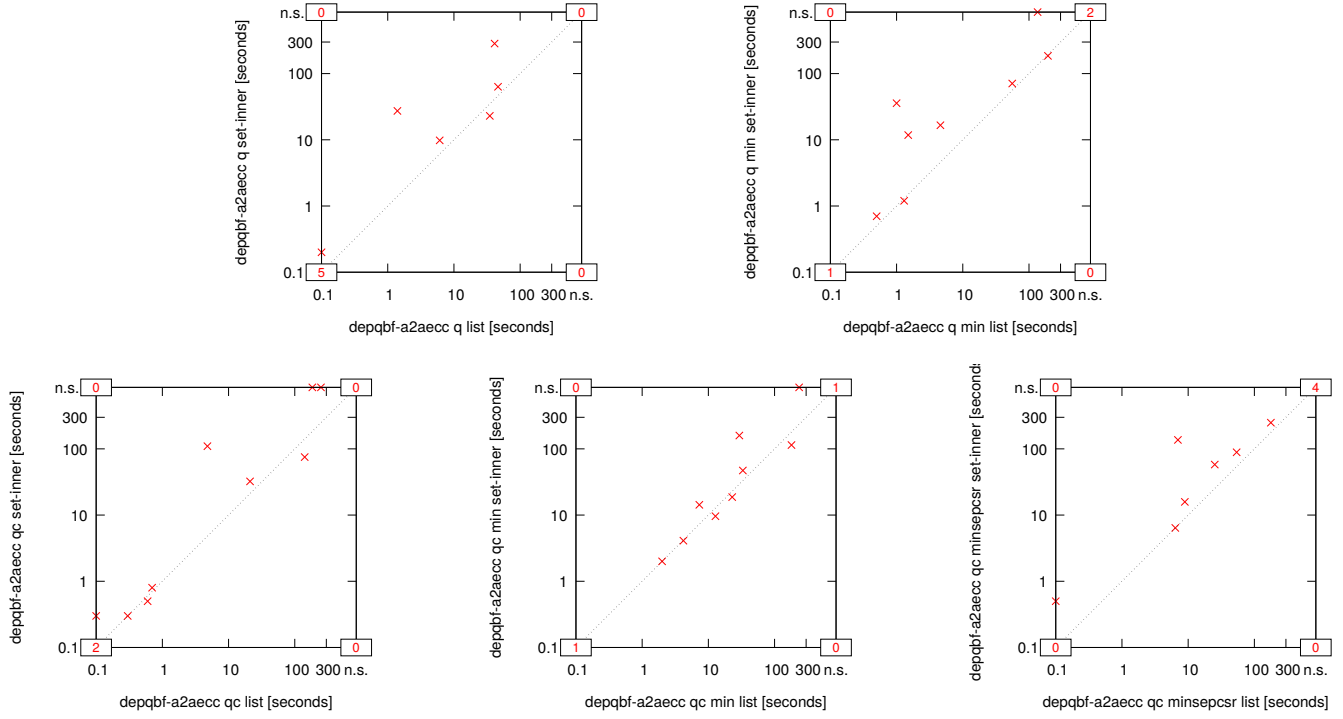


Fig. 665: Suite Diptarama-Jordan-Shinohara ($n = 11$): Extracting and minimizing unsatisfiable cores in `DepQBF-a2aecc` with `set-inner` versus list semantics (run time in seconds).

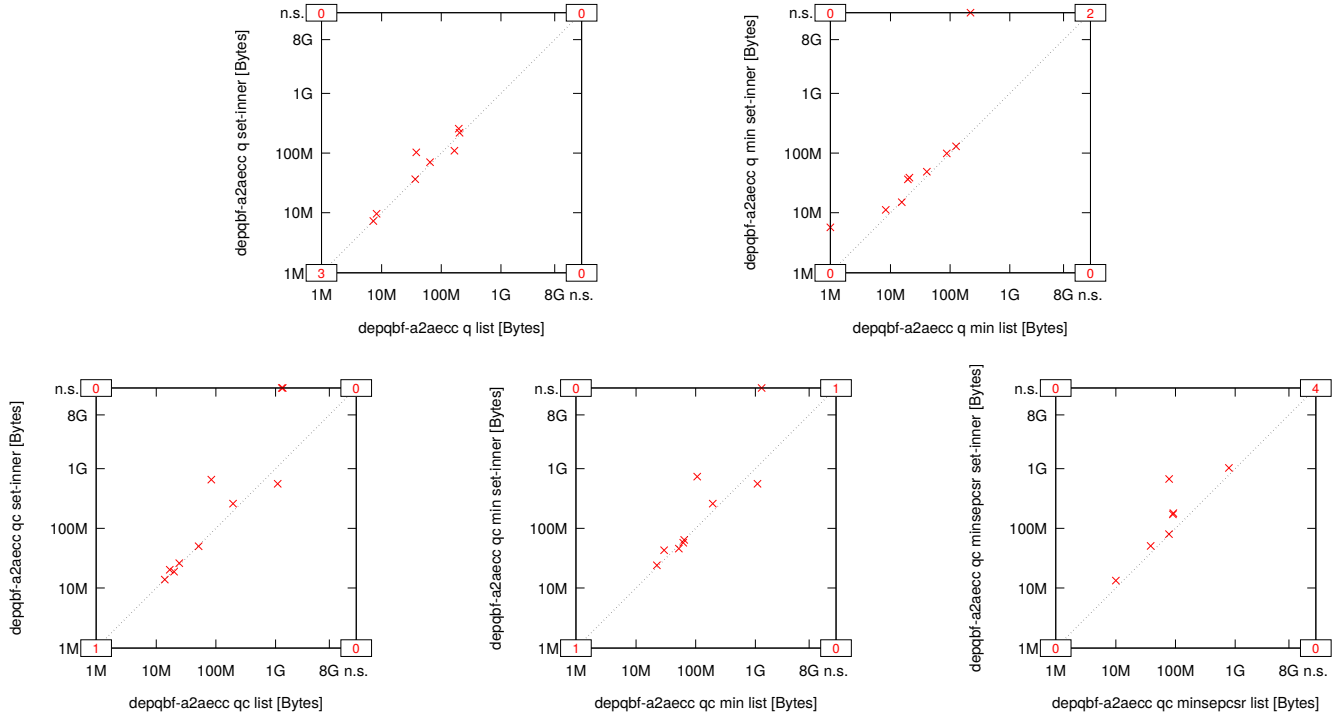


Fig. 666: Suite Diptarama-Jordan-Shinohara ($n = 11$): Extracting and minimizing unsatisfiable cores in `DepQBF-a2aecc` with `set-inner` versus list semantics (memory usage in Bytes).

12) Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$):

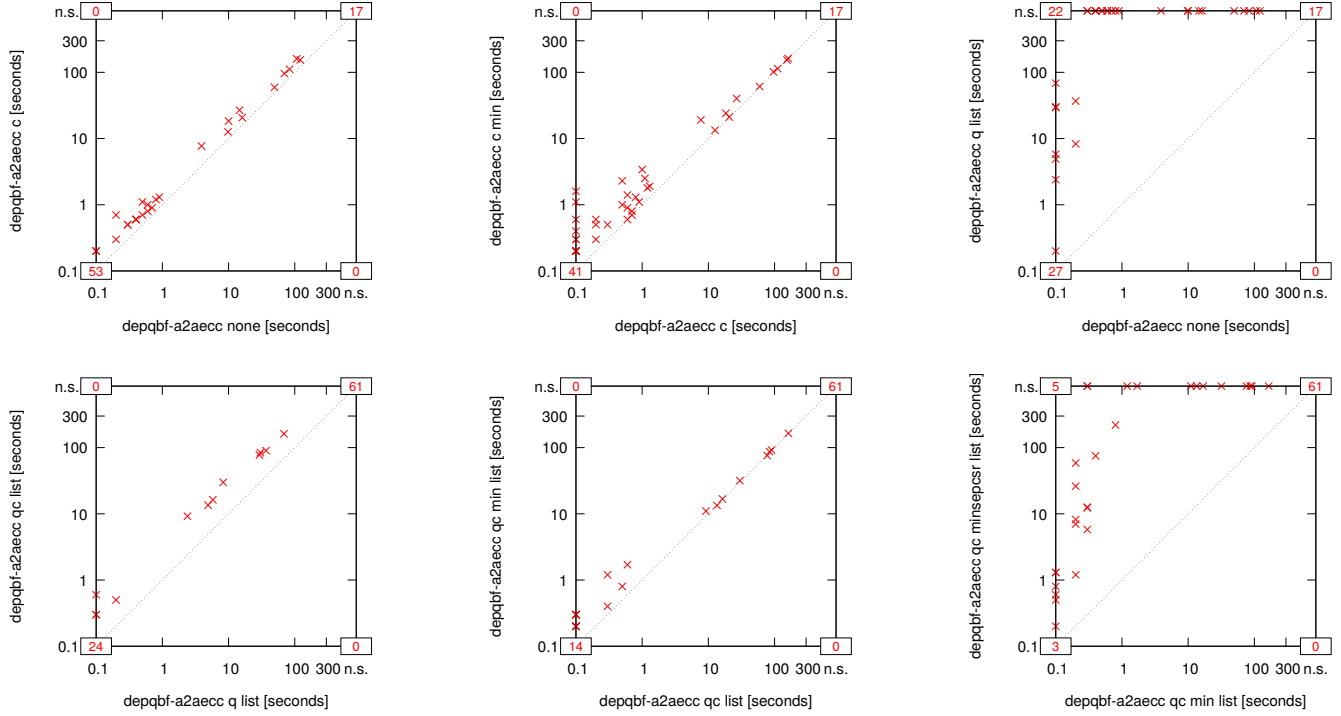


Fig. 667: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Comparing run times for extracting different unsatisfiable cores in DepQBF- $-a2aecc$ with list semantics (run time in seconds).

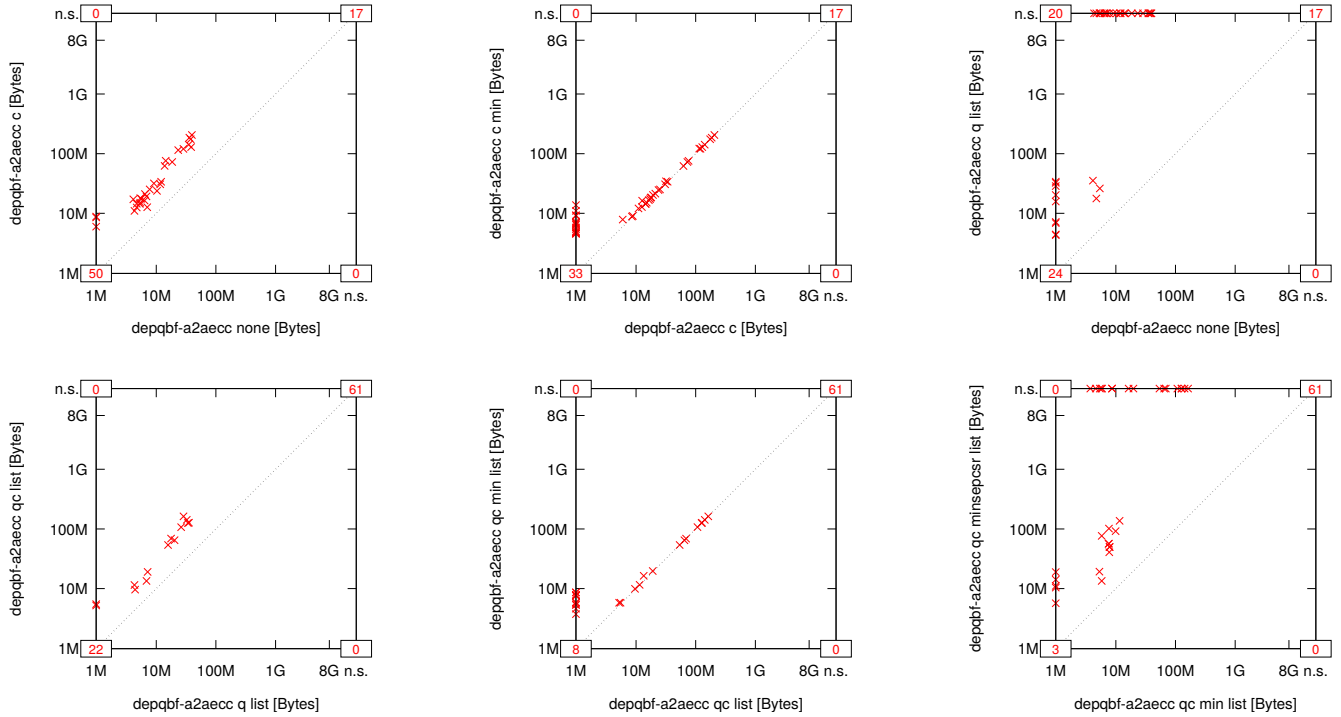


Fig. 668: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF- $-a2aecc$ with list semantics (memory usage in Bytes).

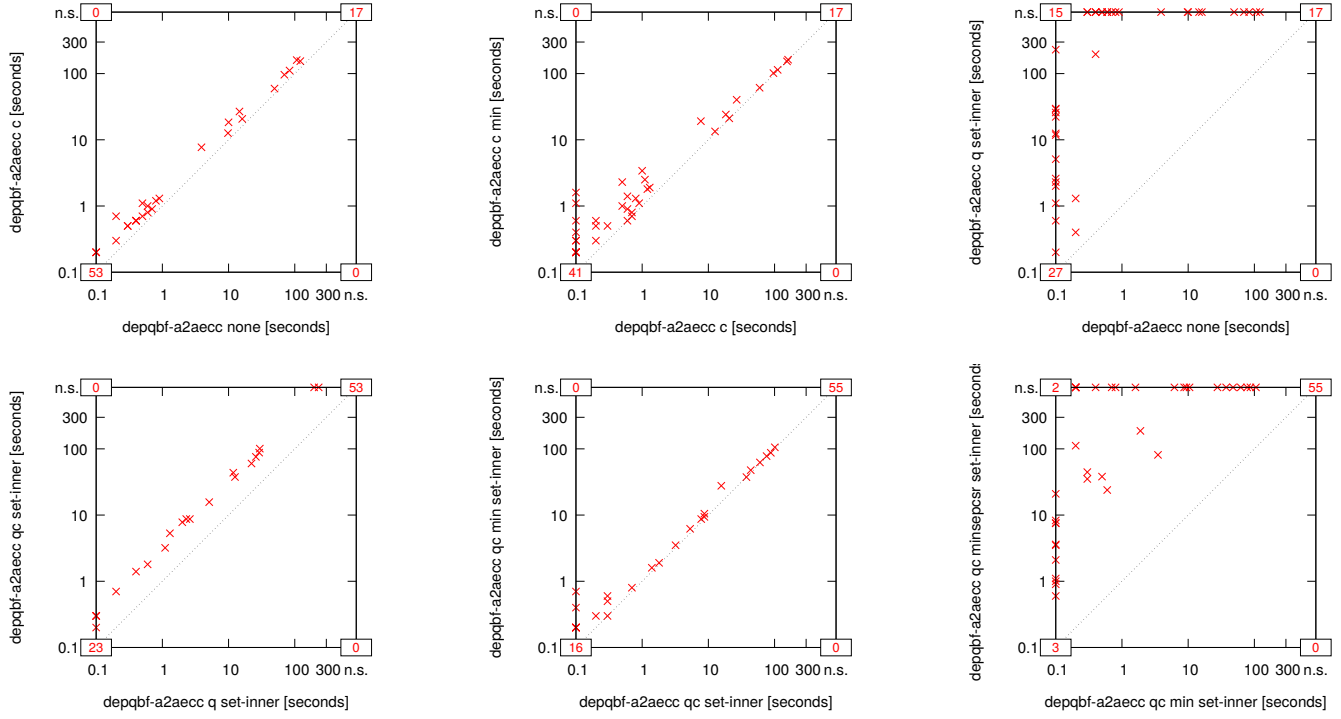


Fig. 669: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

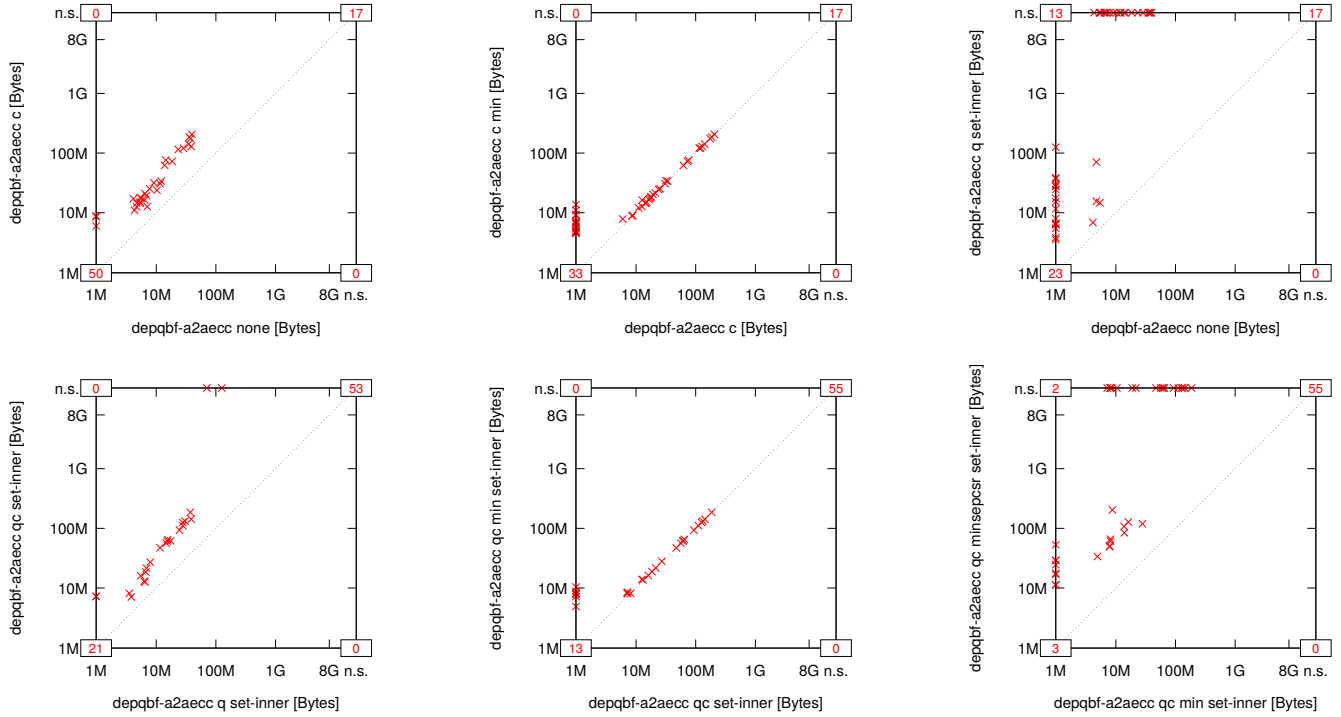


Fig. 670: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

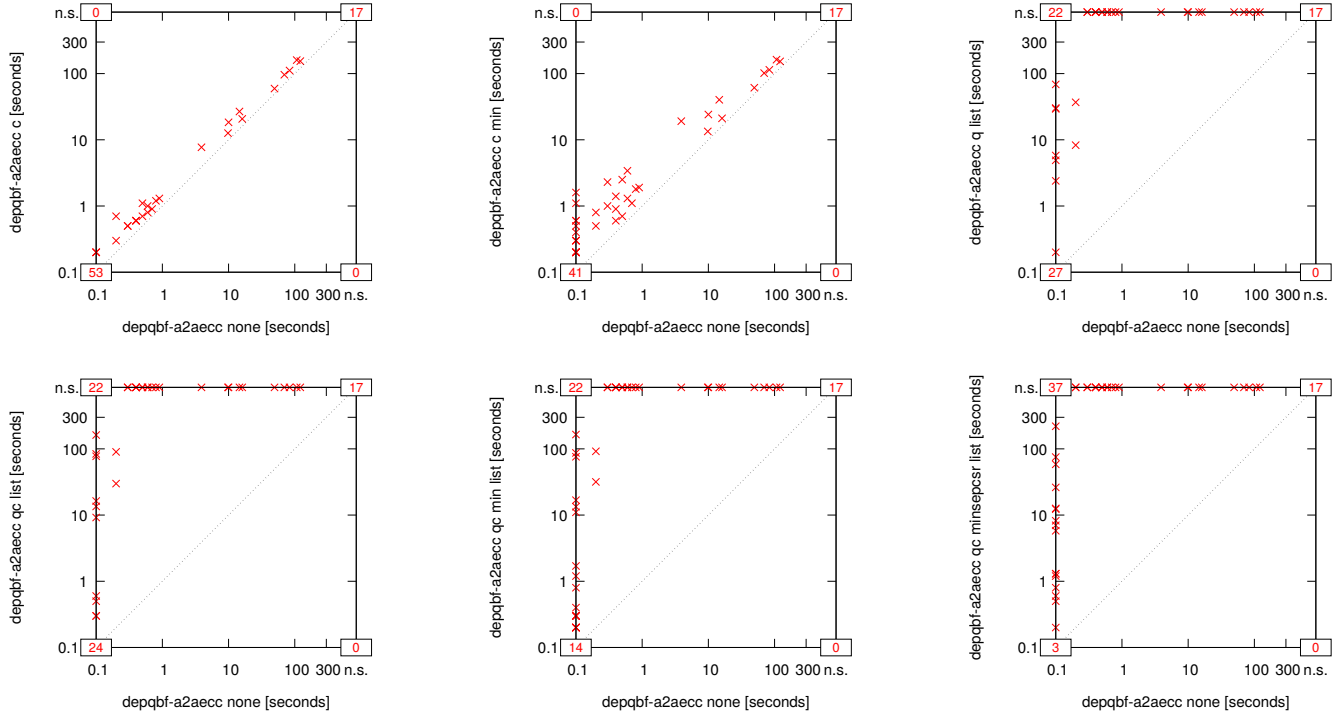


Fig. 671: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

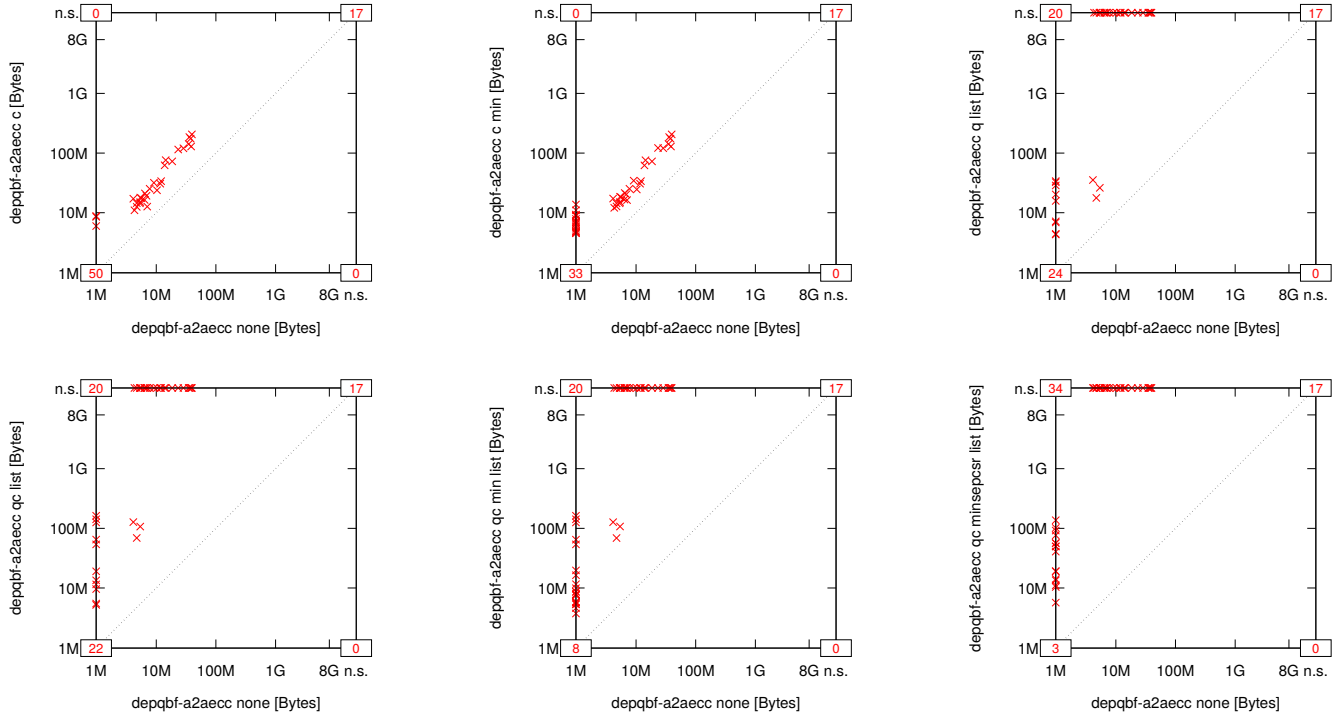


Fig. 672: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

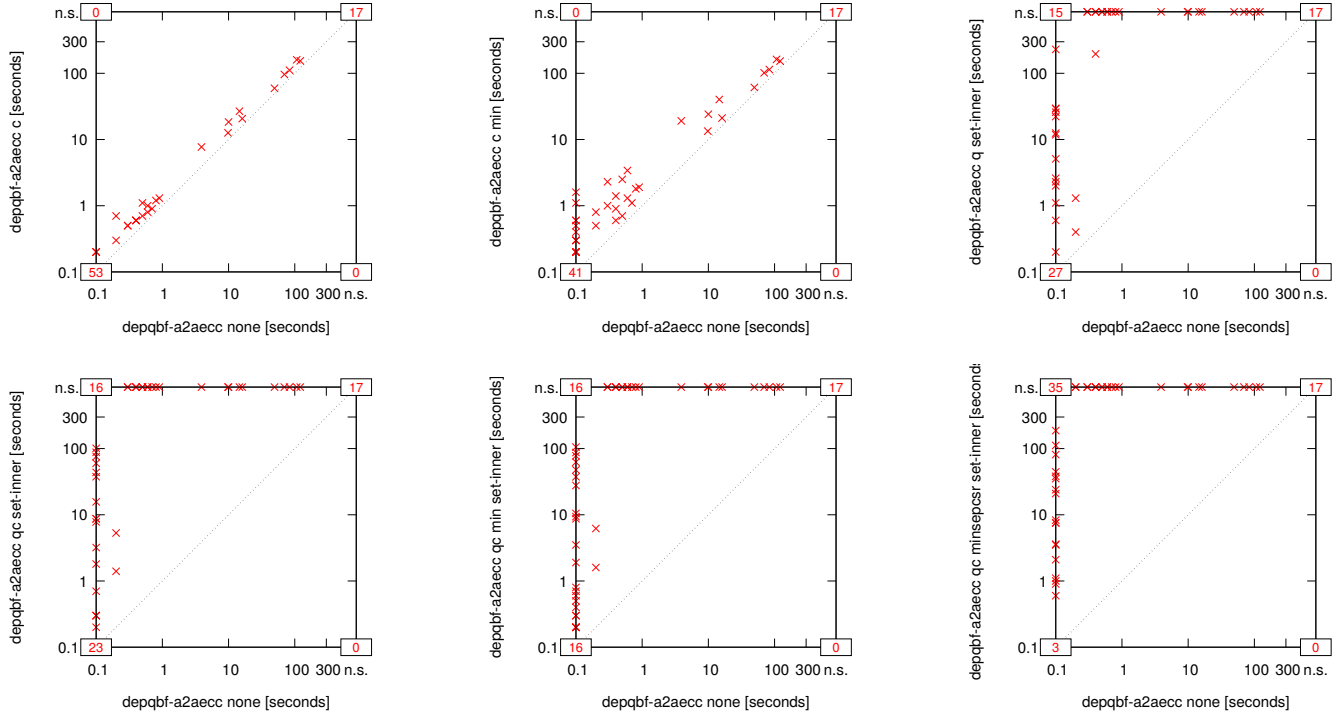


Fig. 673: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

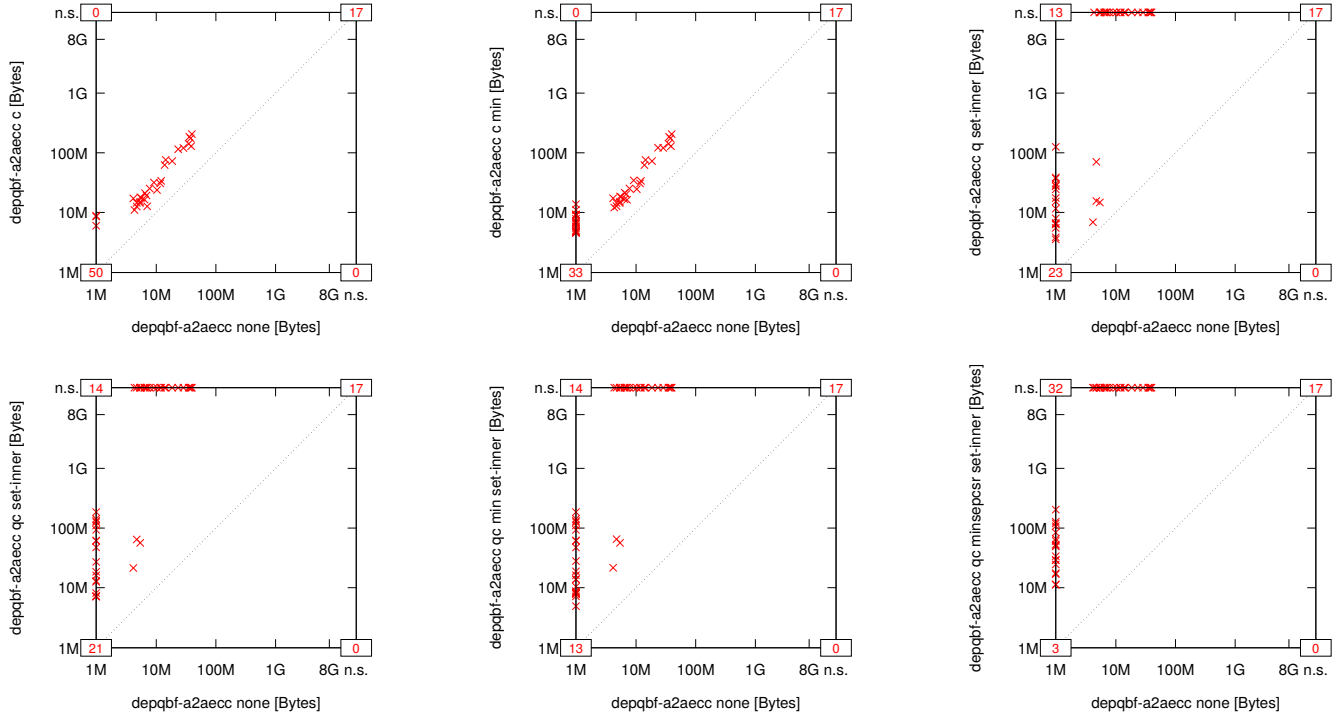


Fig. 674: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

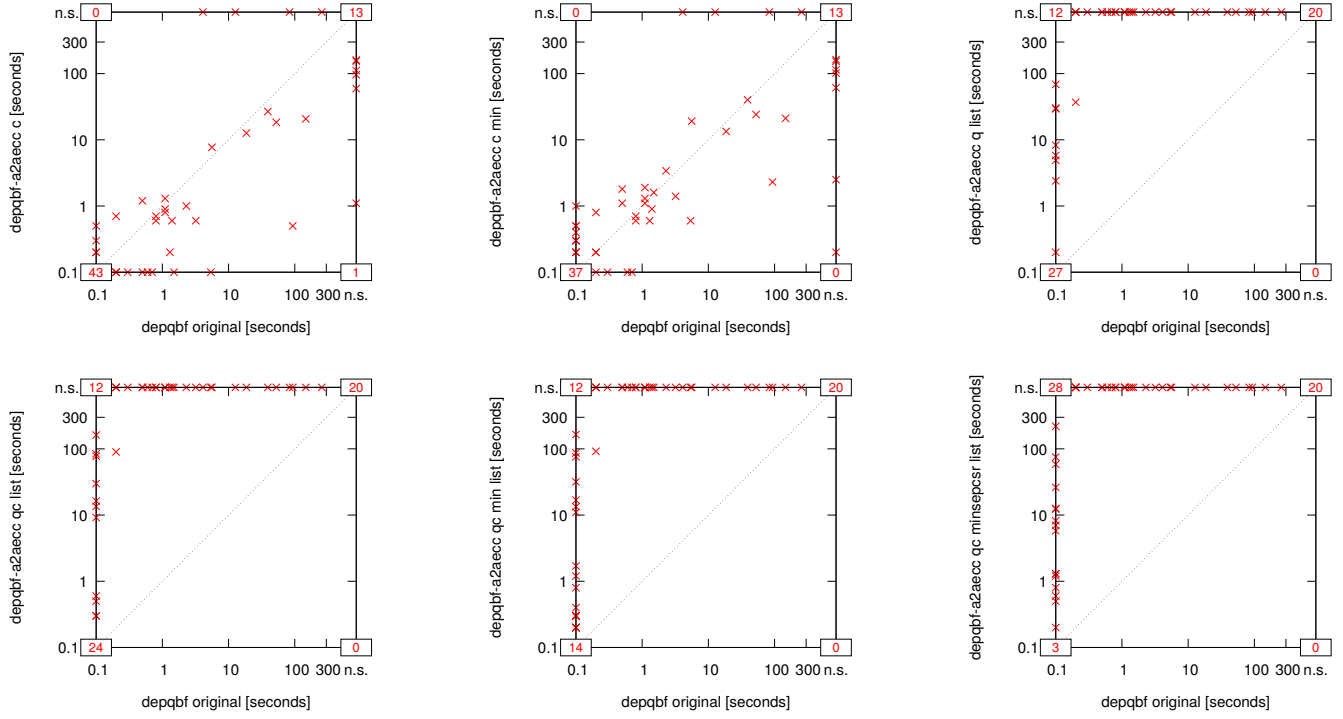


Fig. 675: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

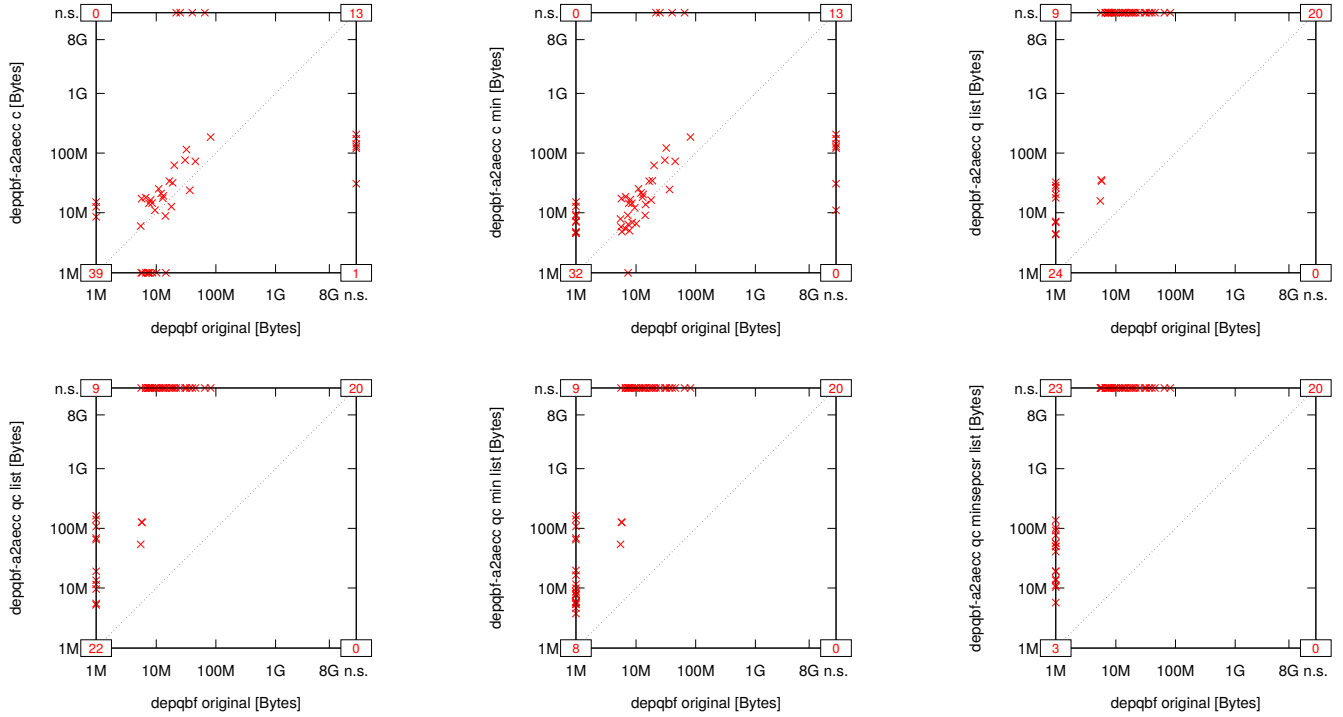


Fig. 676: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

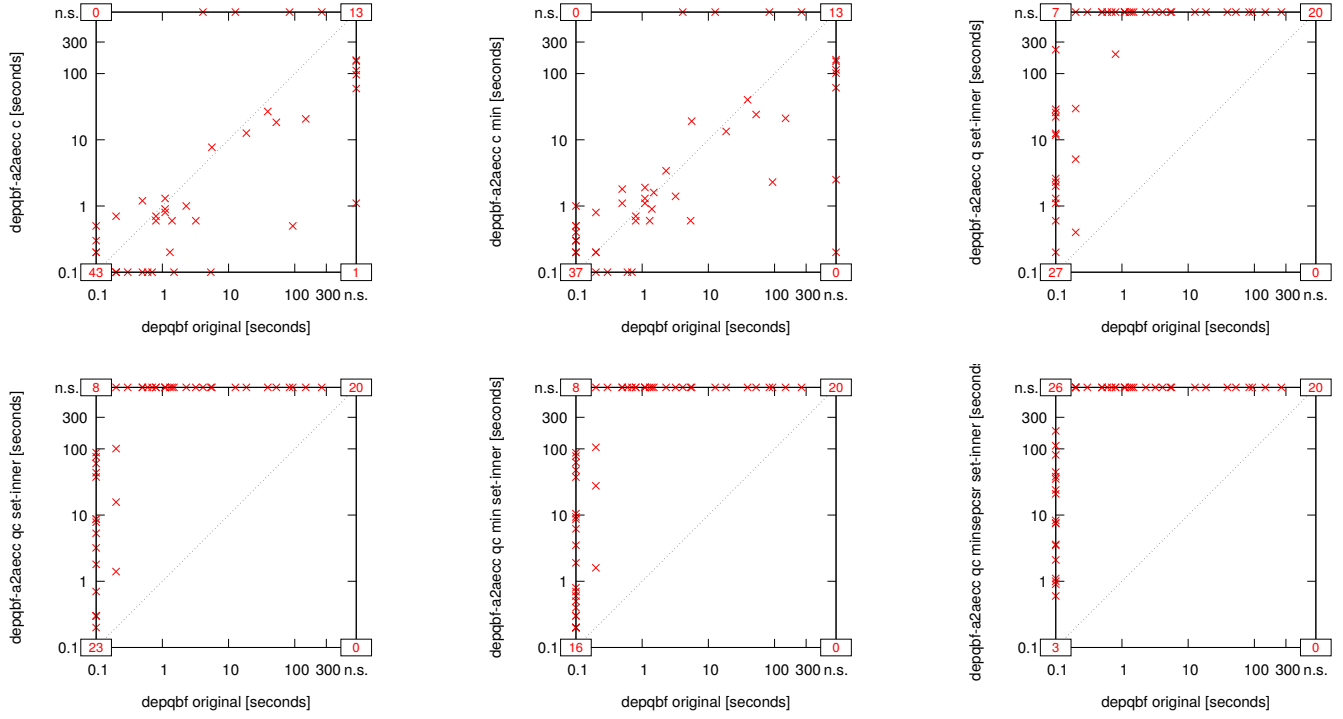


Fig. 677: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

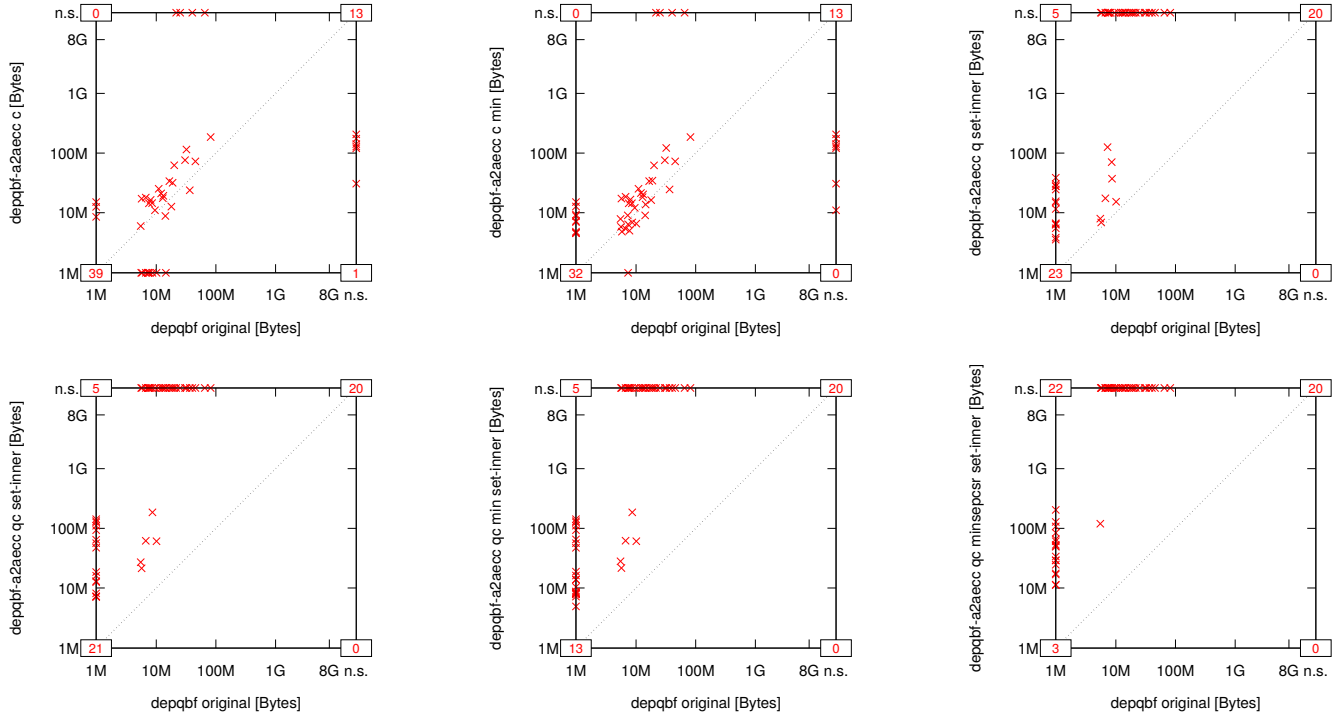


Fig. 678: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

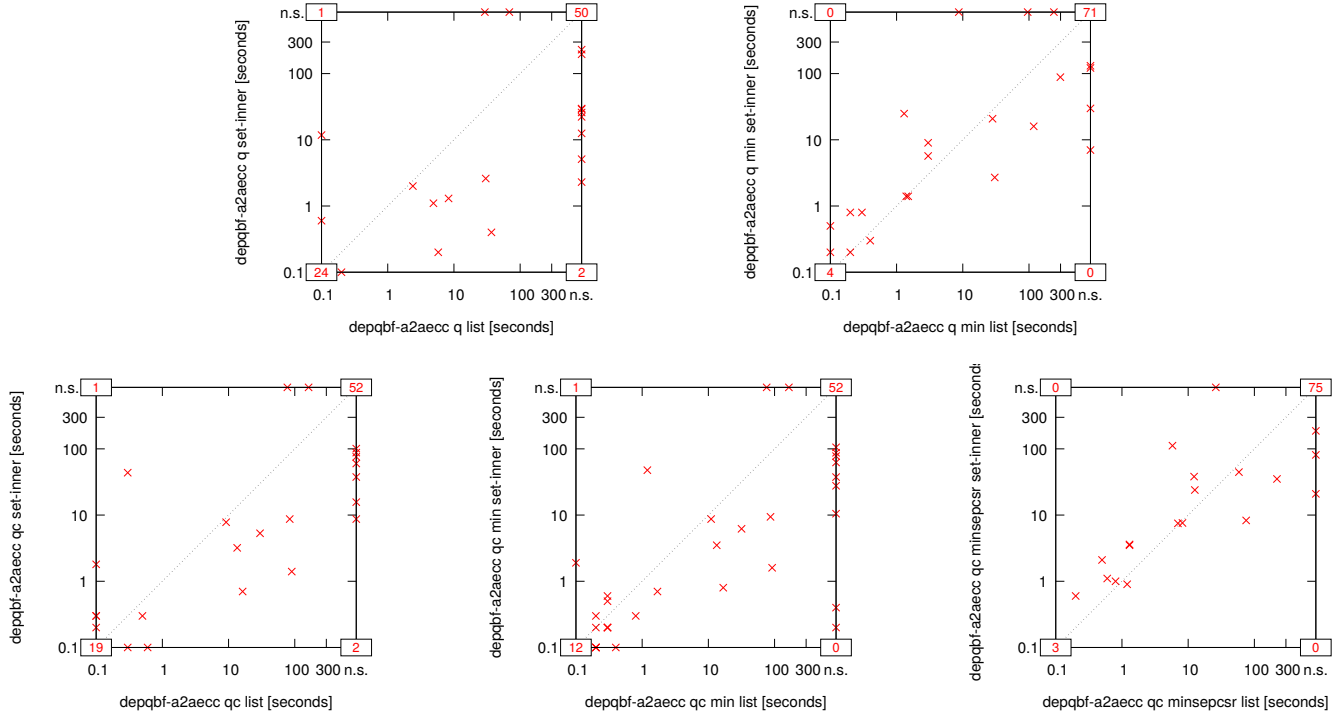


Fig. 679: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

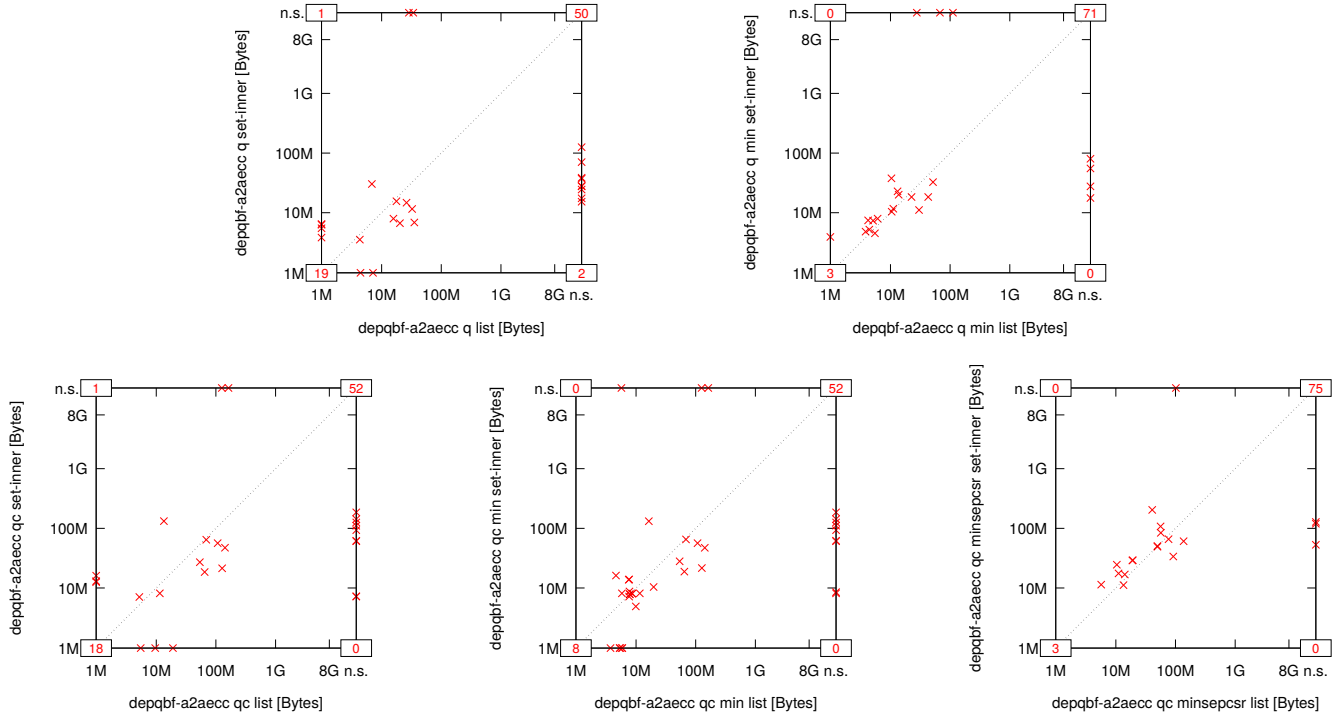


Fig. 680: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 97$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

13) *Faber-Leone-Maratea-Ricca* ($n = 128$):

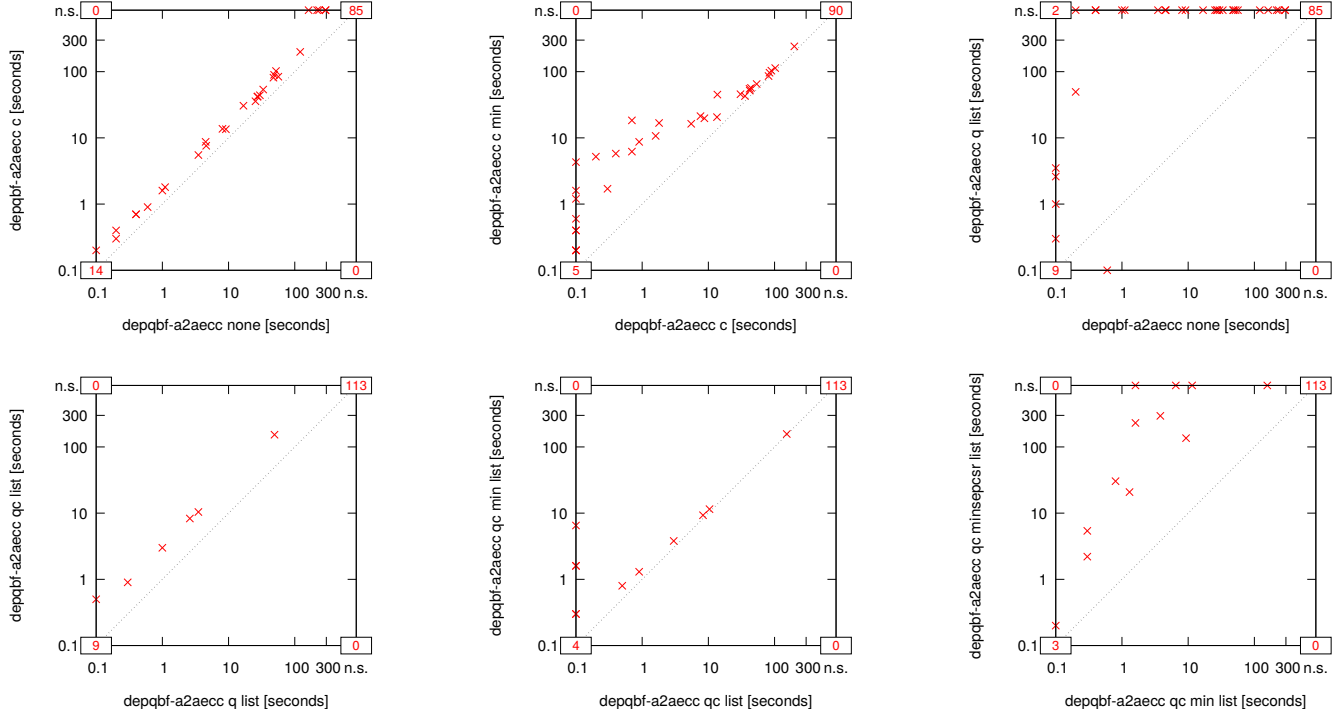


Fig. 681: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

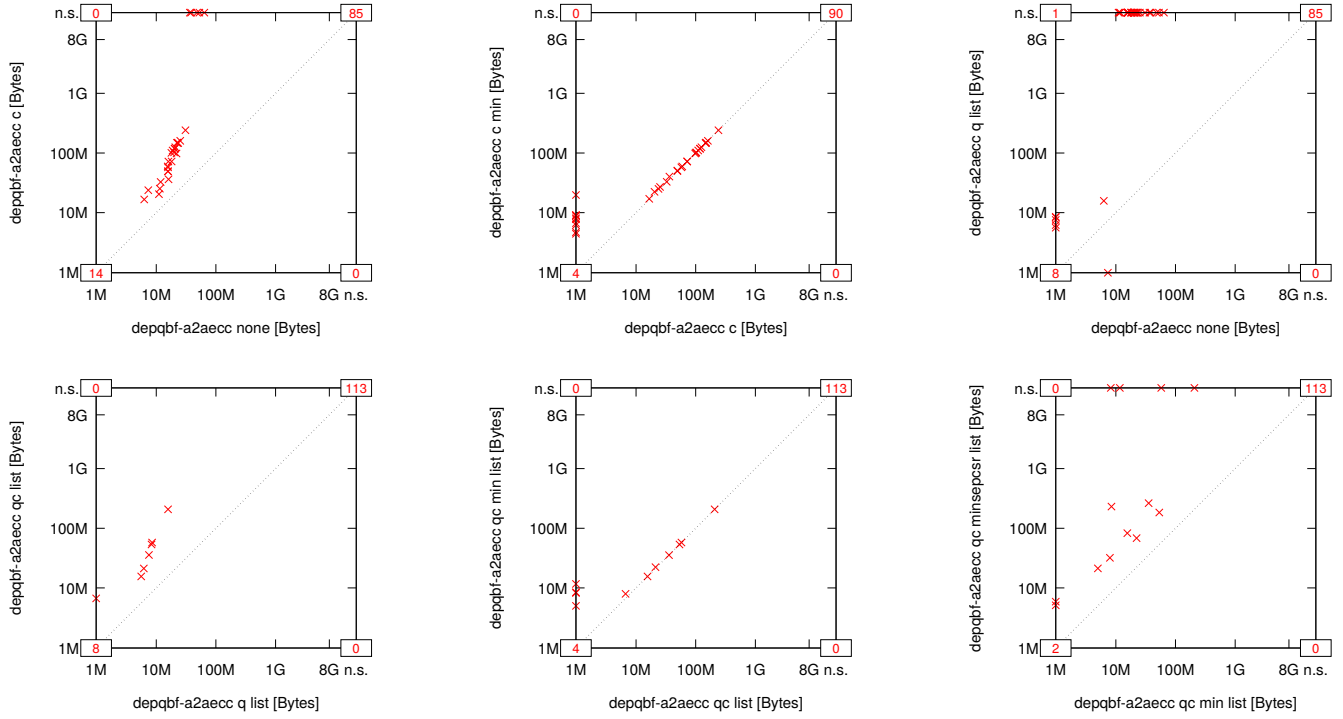


Fig. 682: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

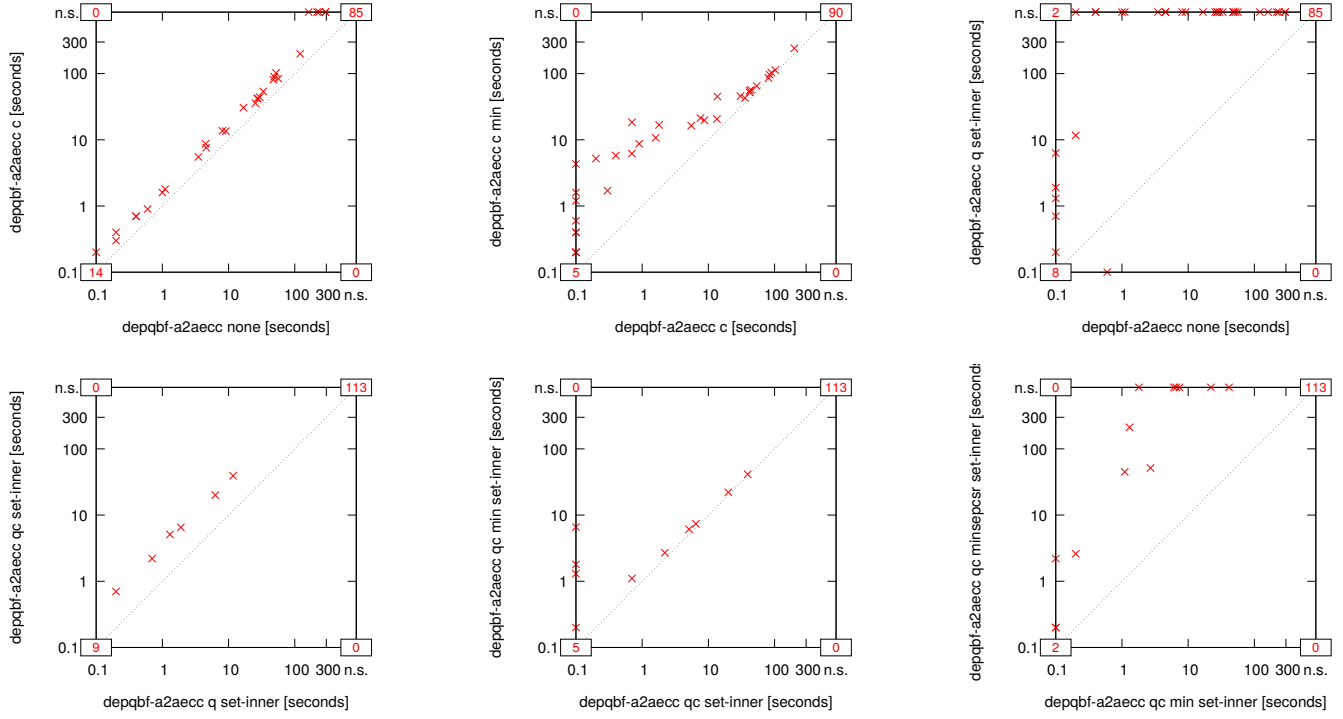


Fig. 683: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

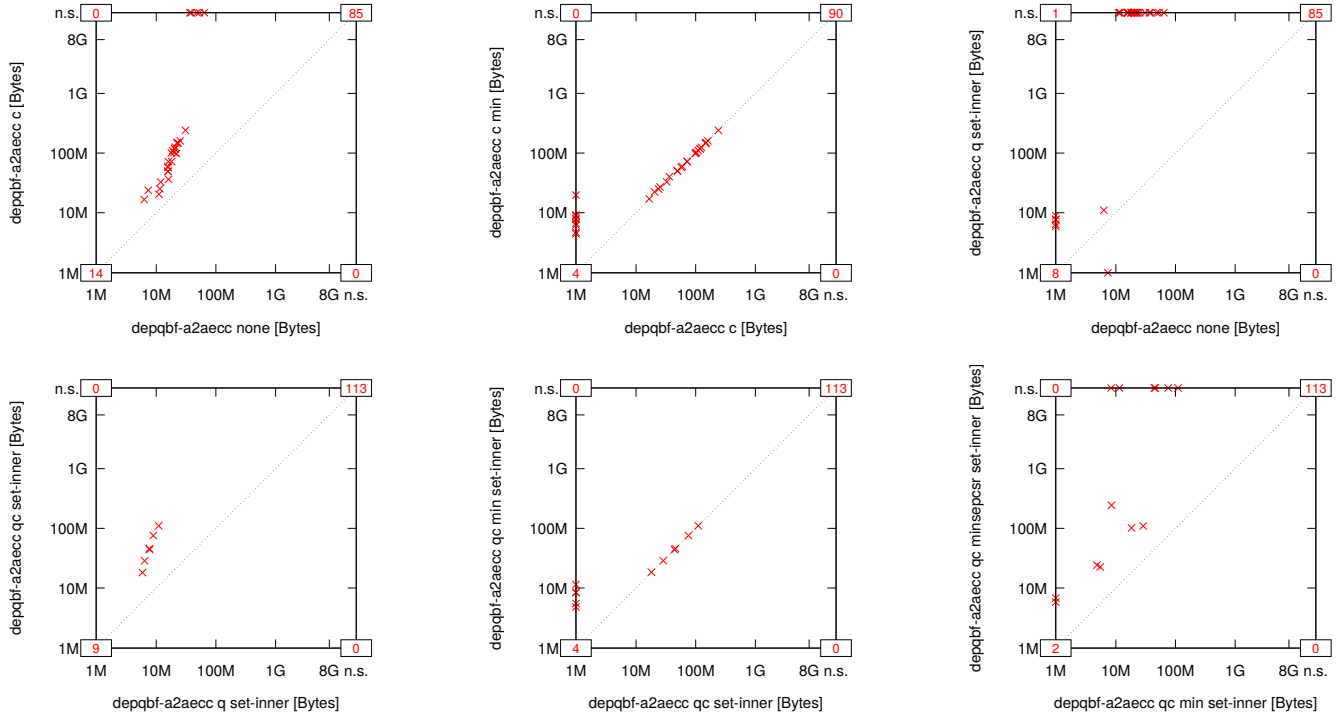


Fig. 684: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

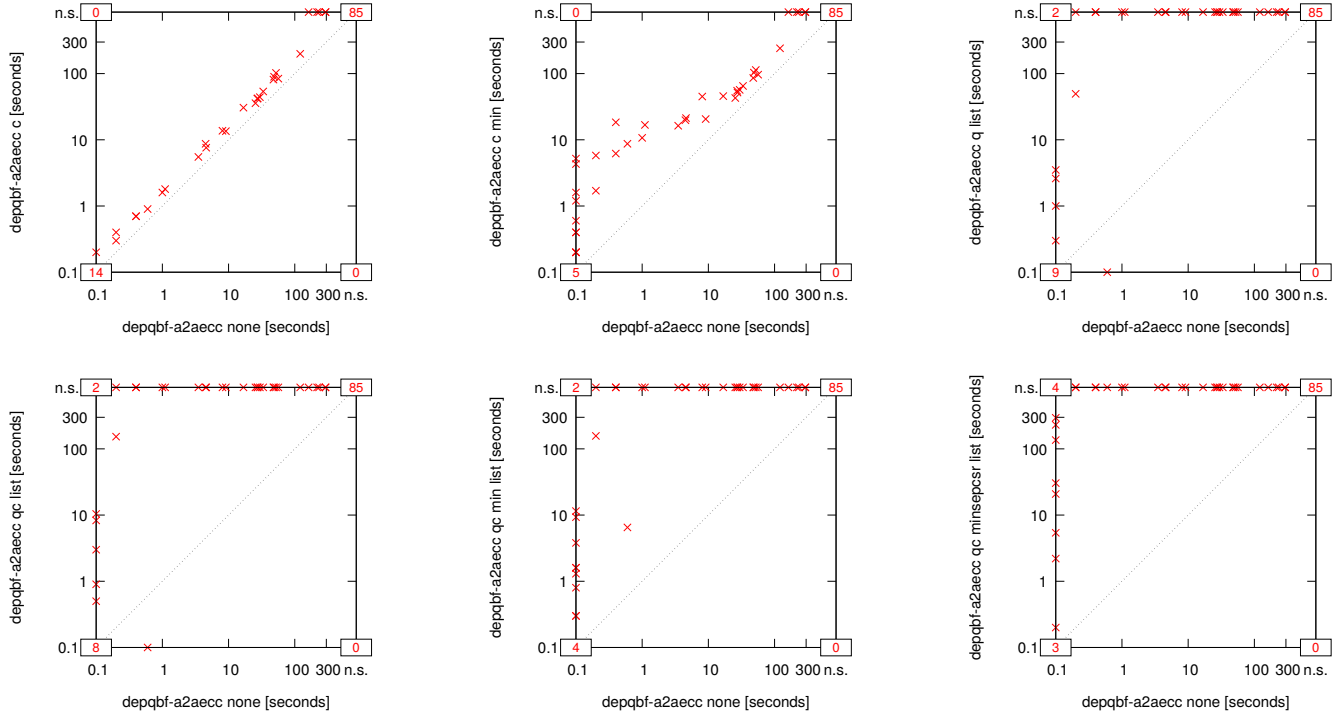


Fig. 685: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

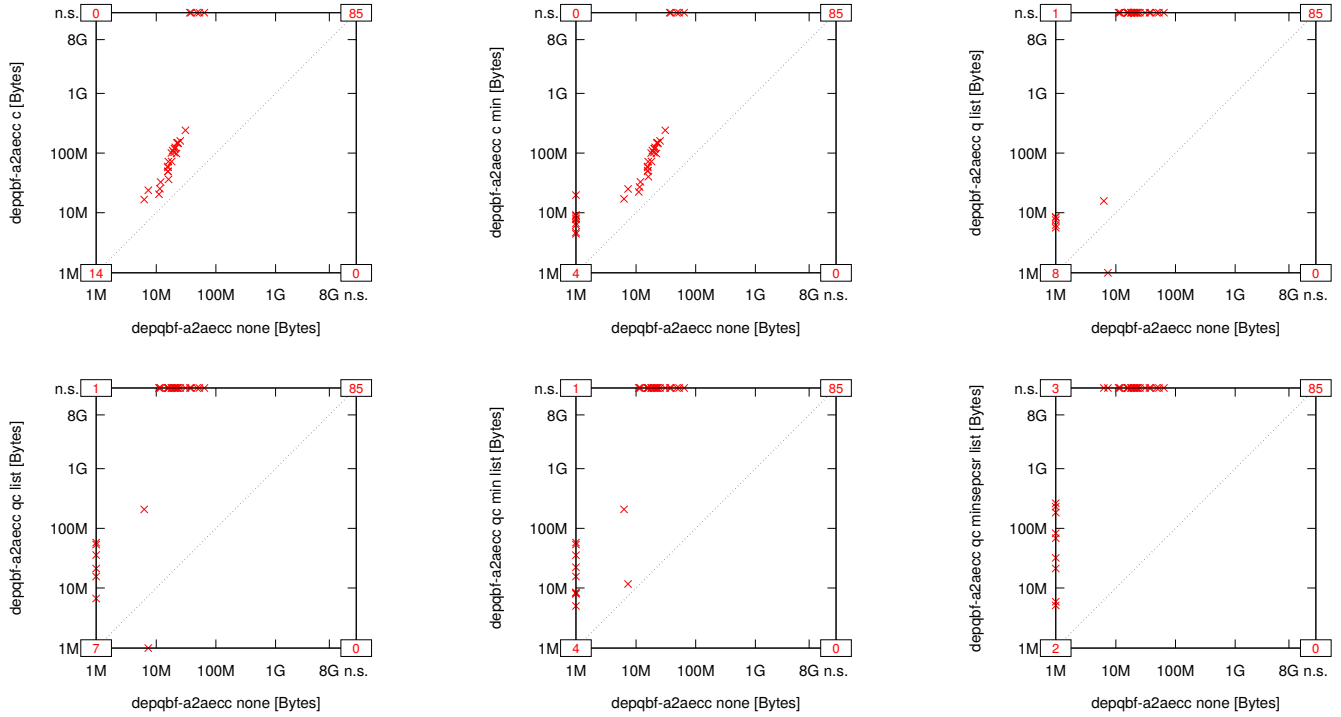


Fig. 686: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

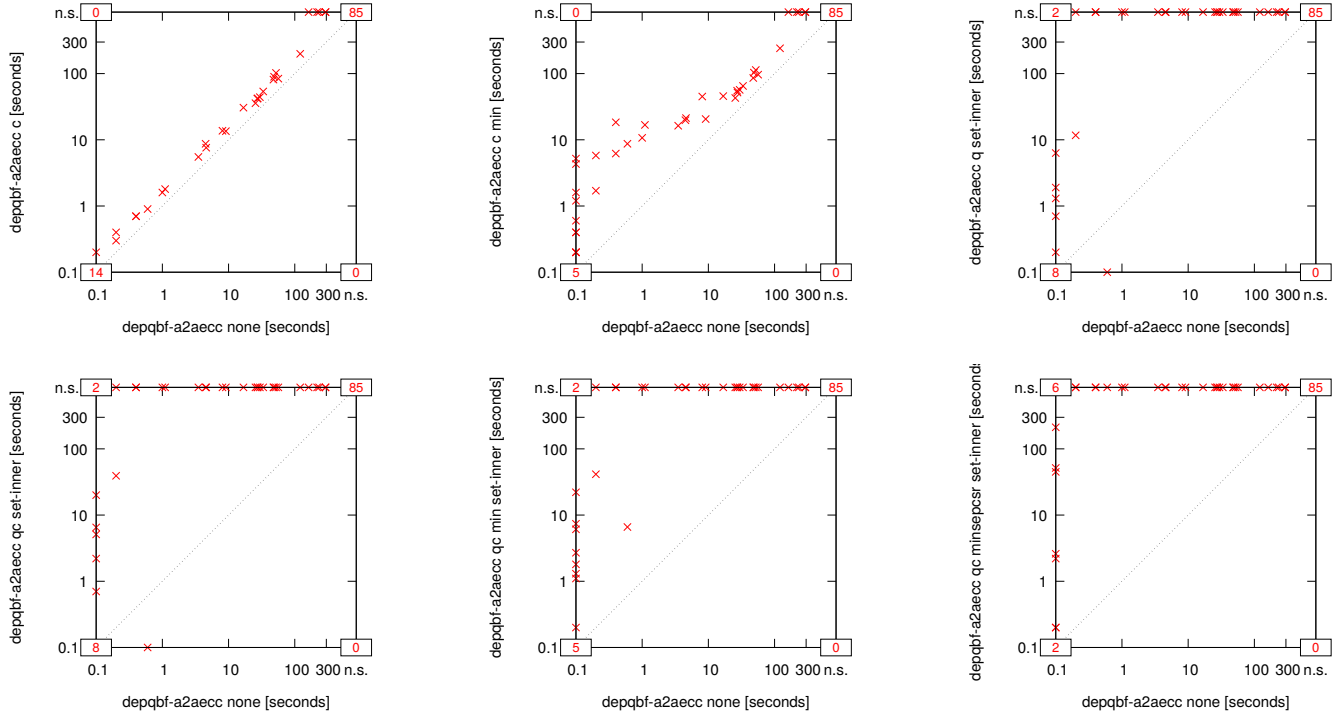


Fig. 687: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

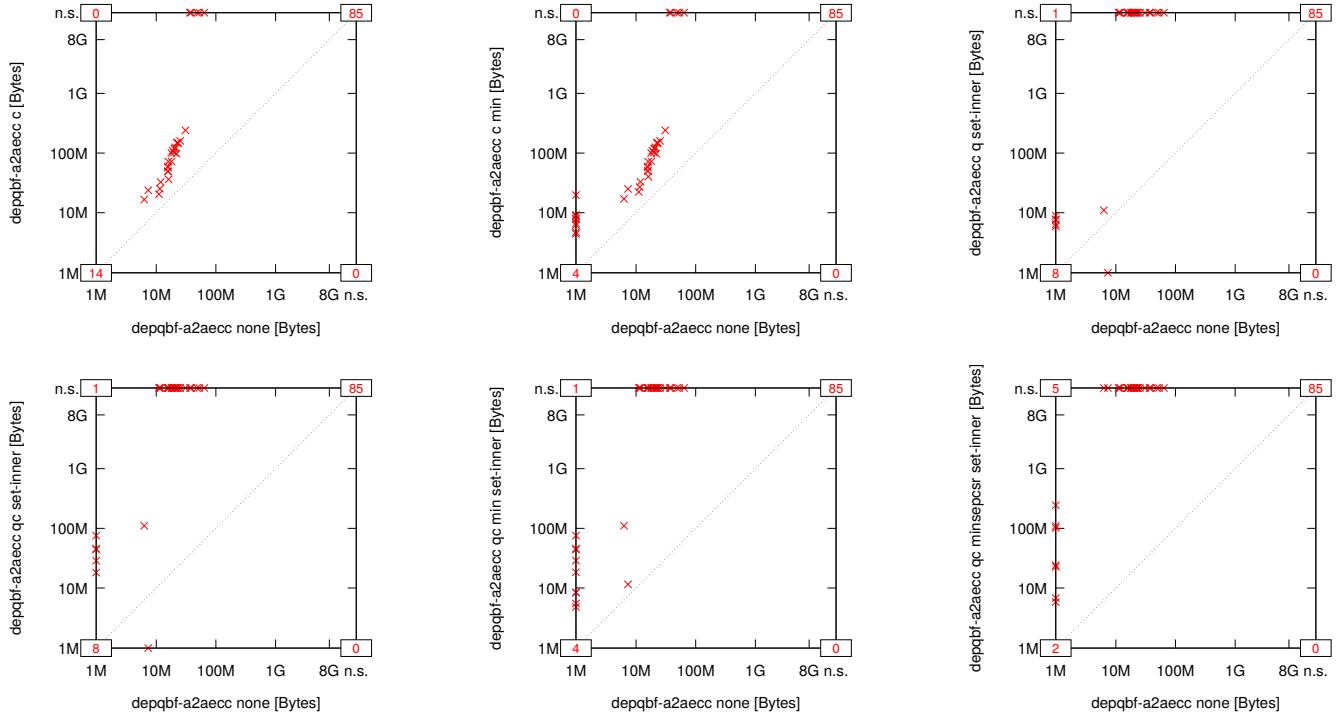


Fig. 688: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

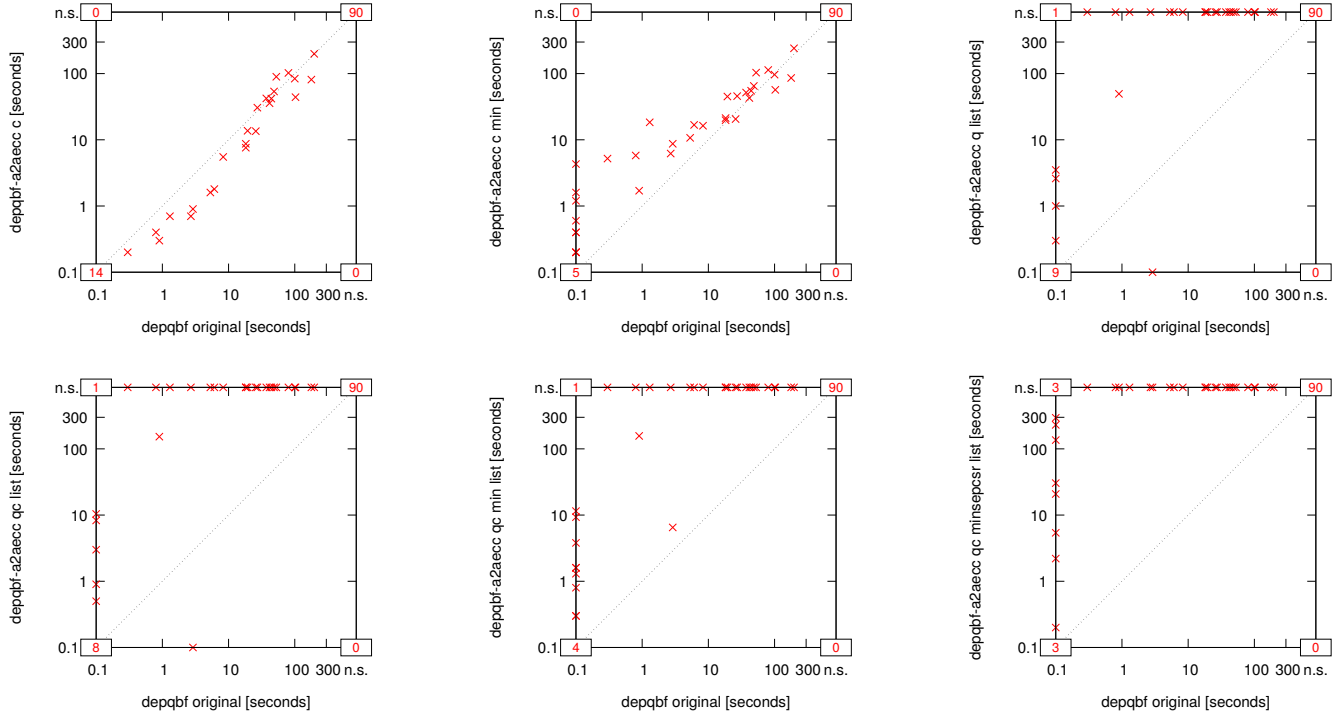


Fig. 689: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

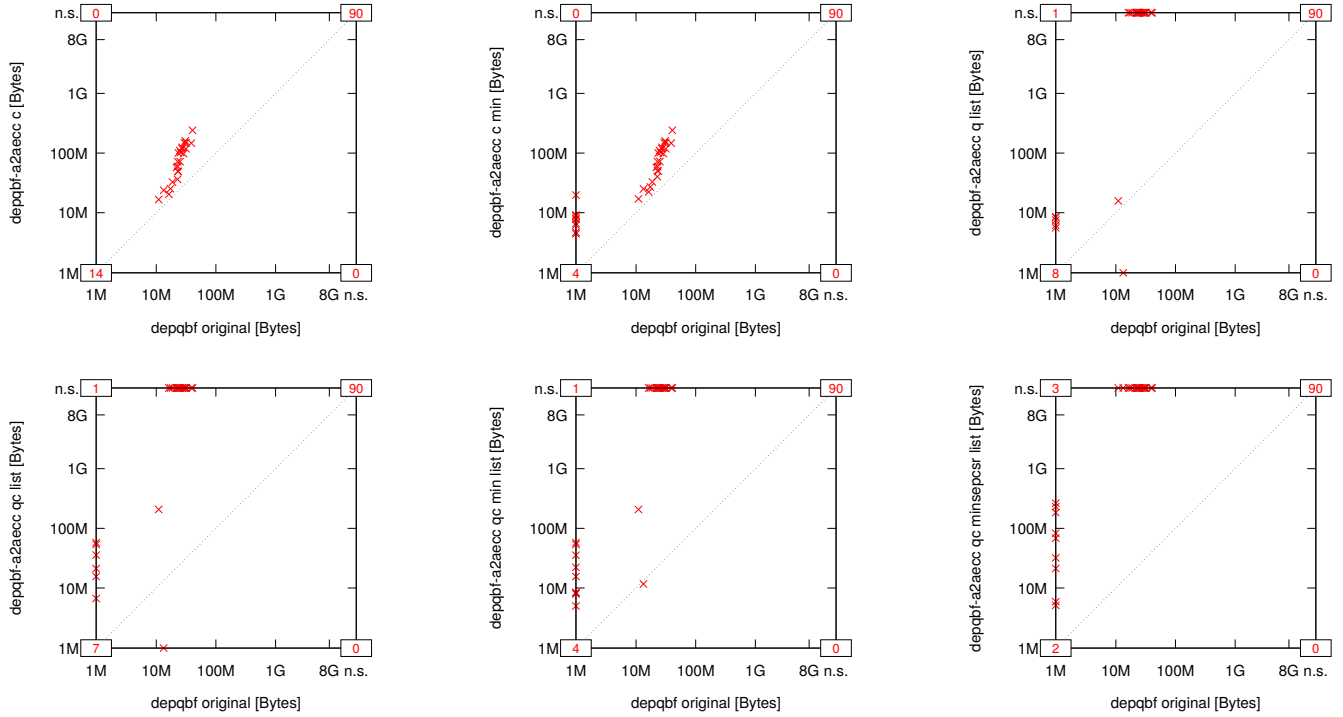


Fig. 690: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

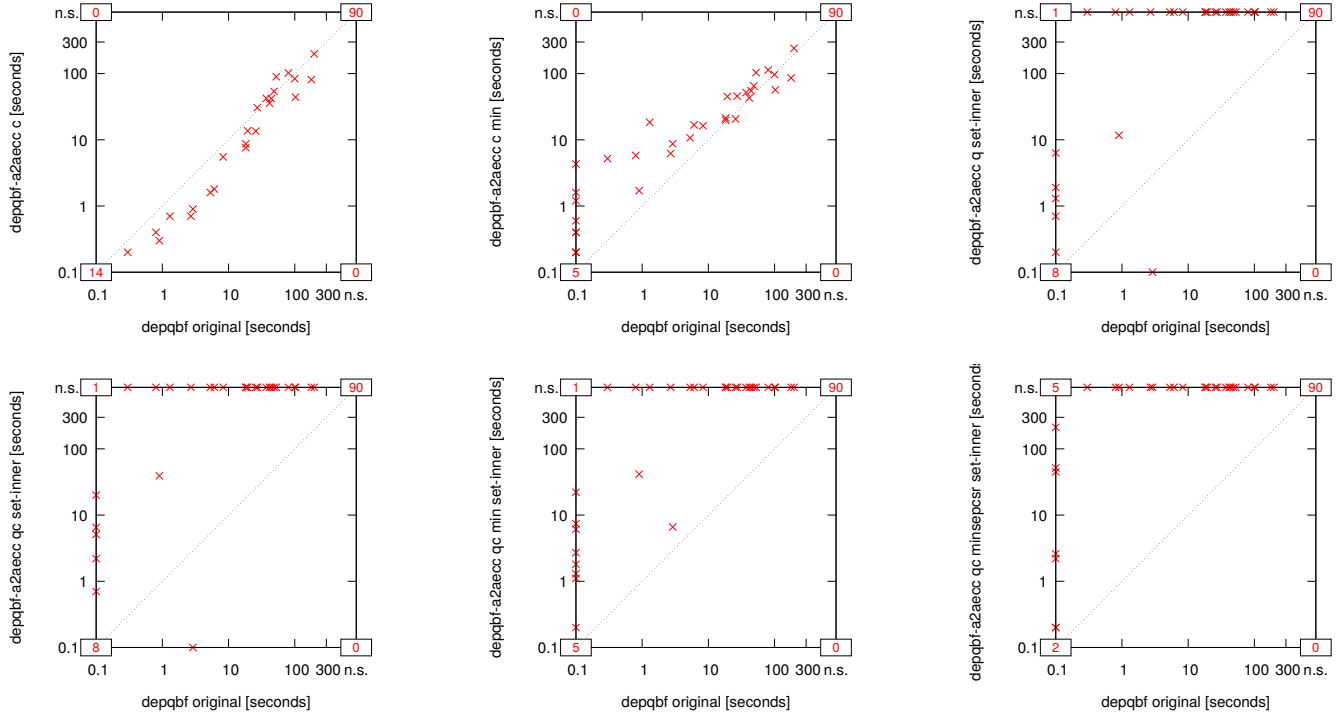


Fig. 691: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

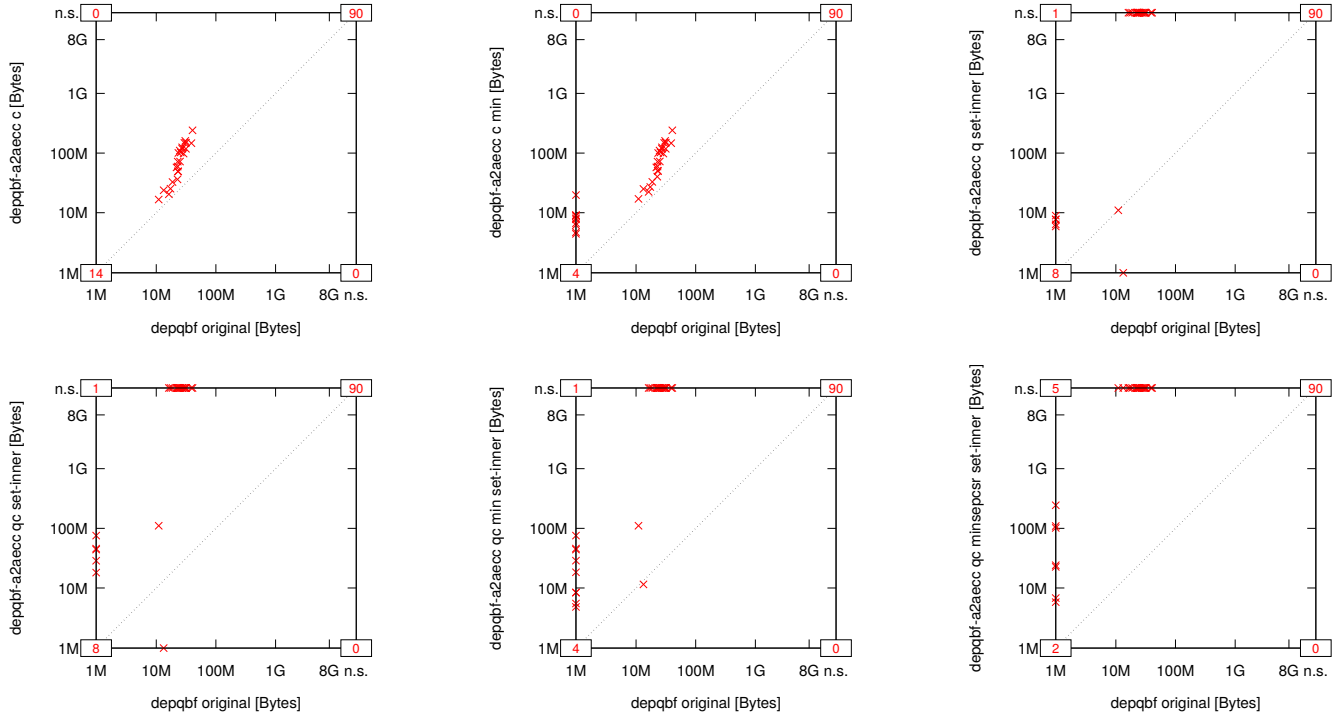


Fig. 692: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

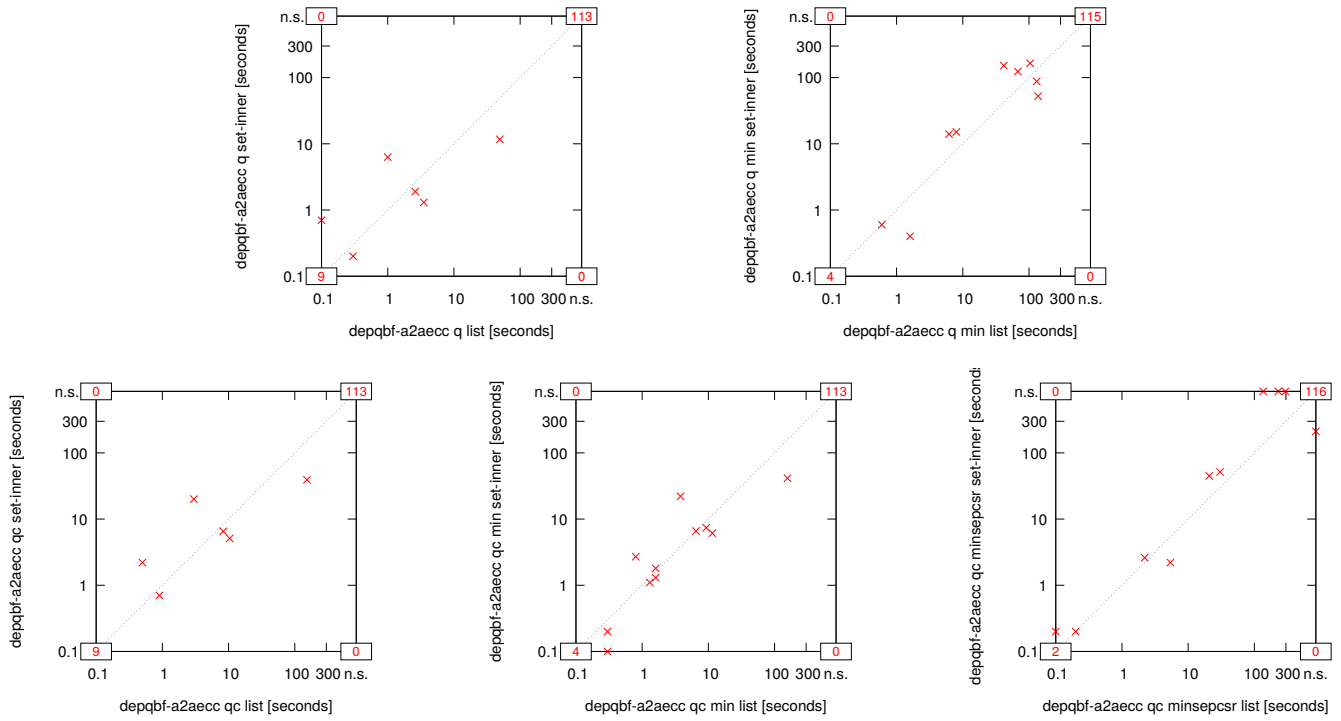


Fig. 693: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

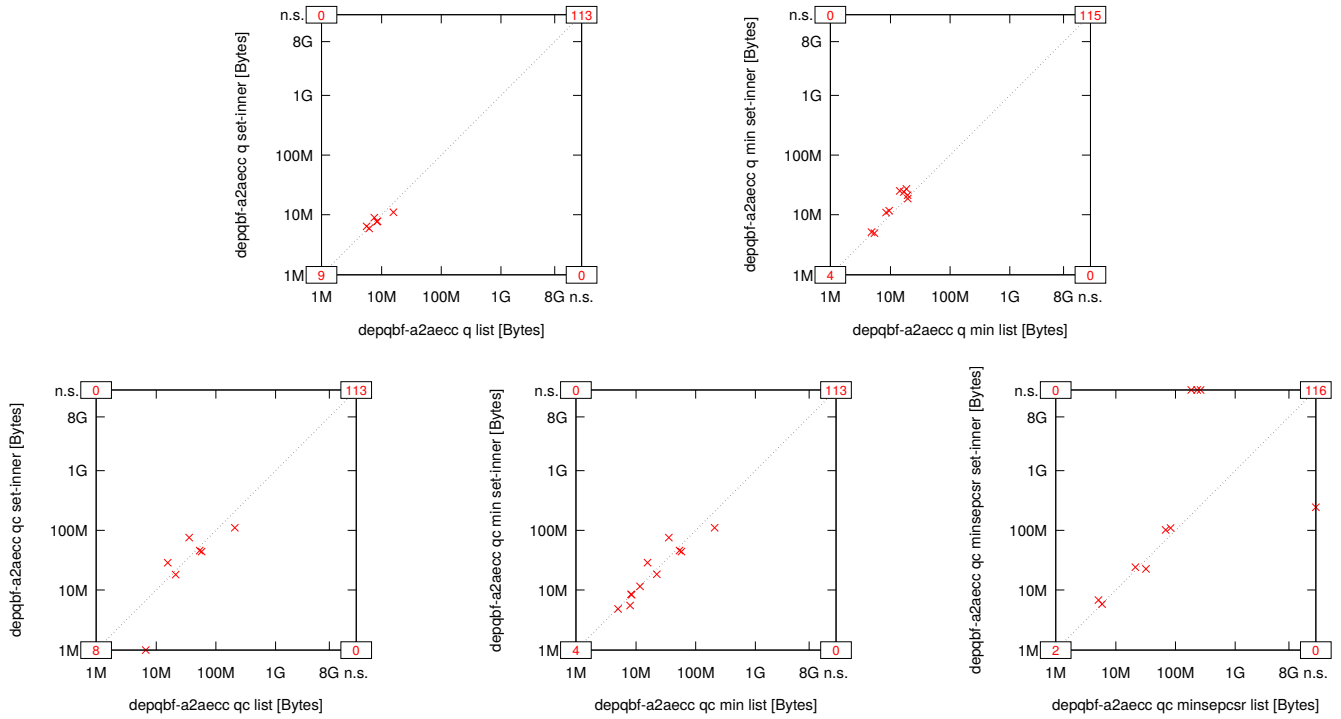


Fig. 694: Suite Faber-Leone-Maratea-Ricca ($n = 128$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

14) Gent-Rowley ($n = 142$):

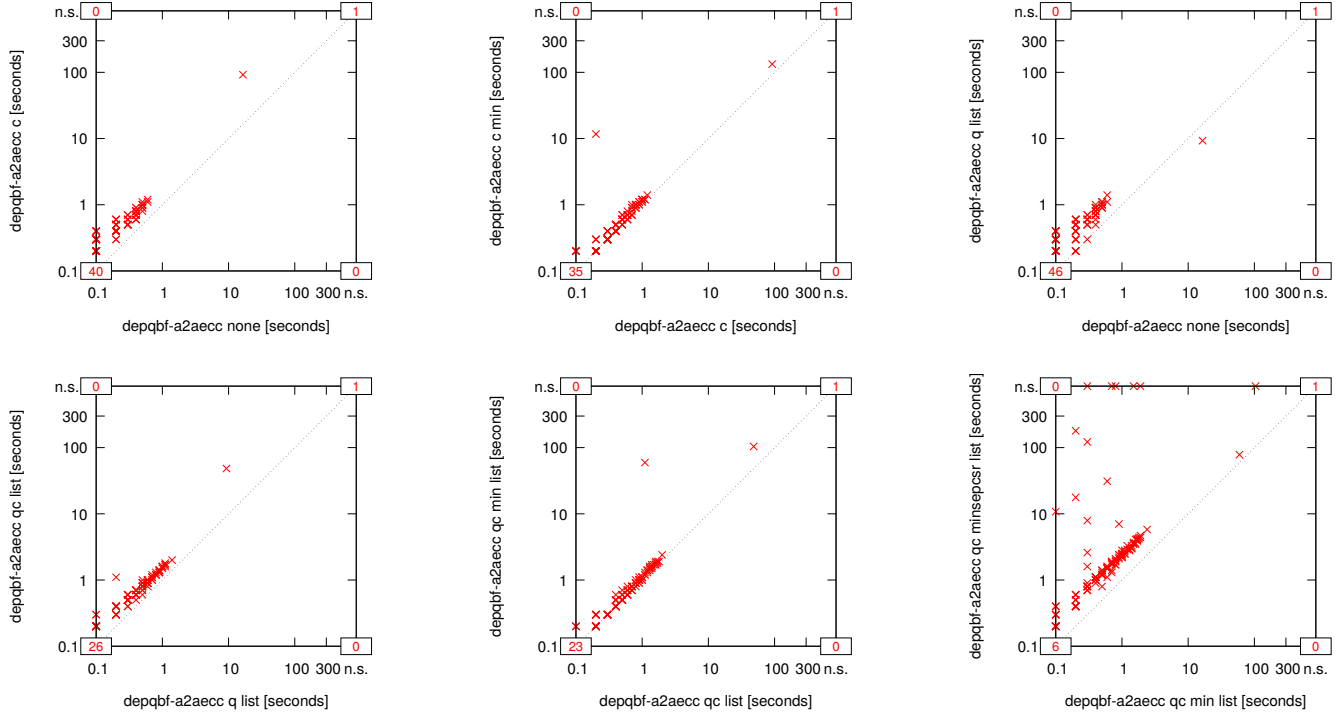


Fig. 695: Suite Gent-Rowley ($n = 142$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

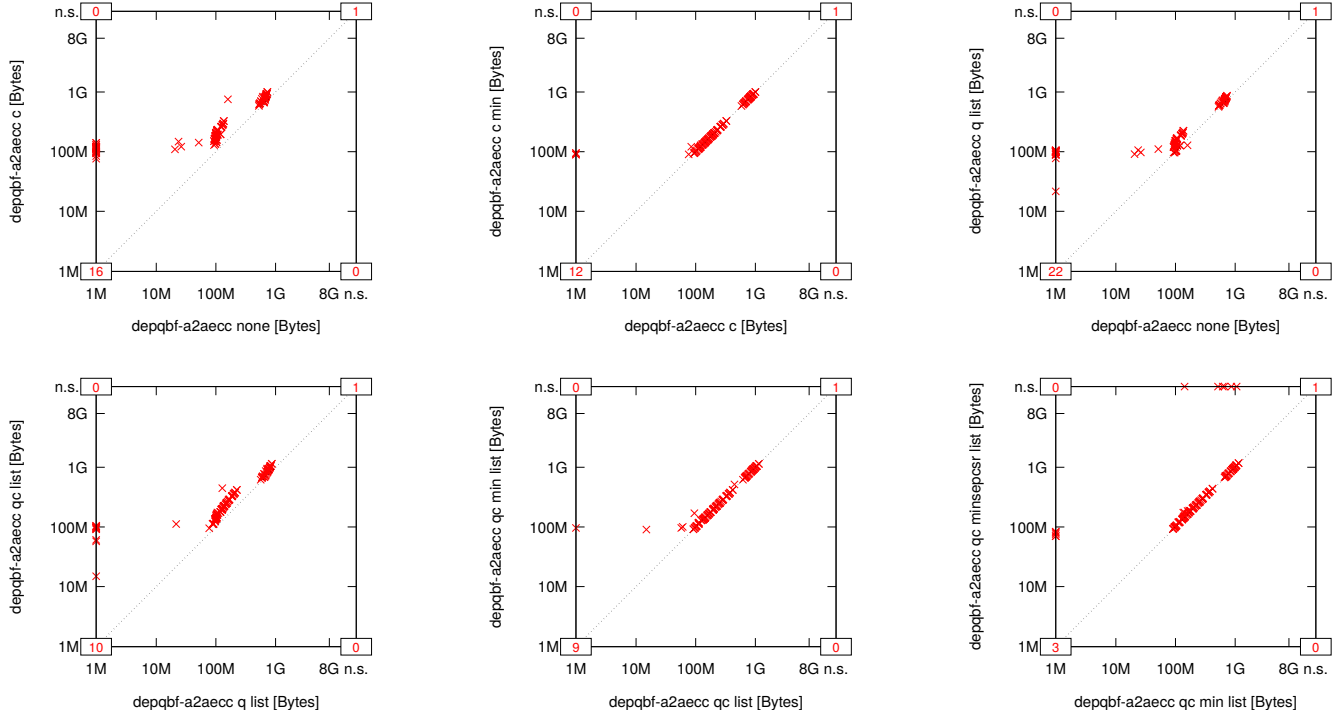


Fig. 696: Suite Gent-Rowley ($n = 142$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

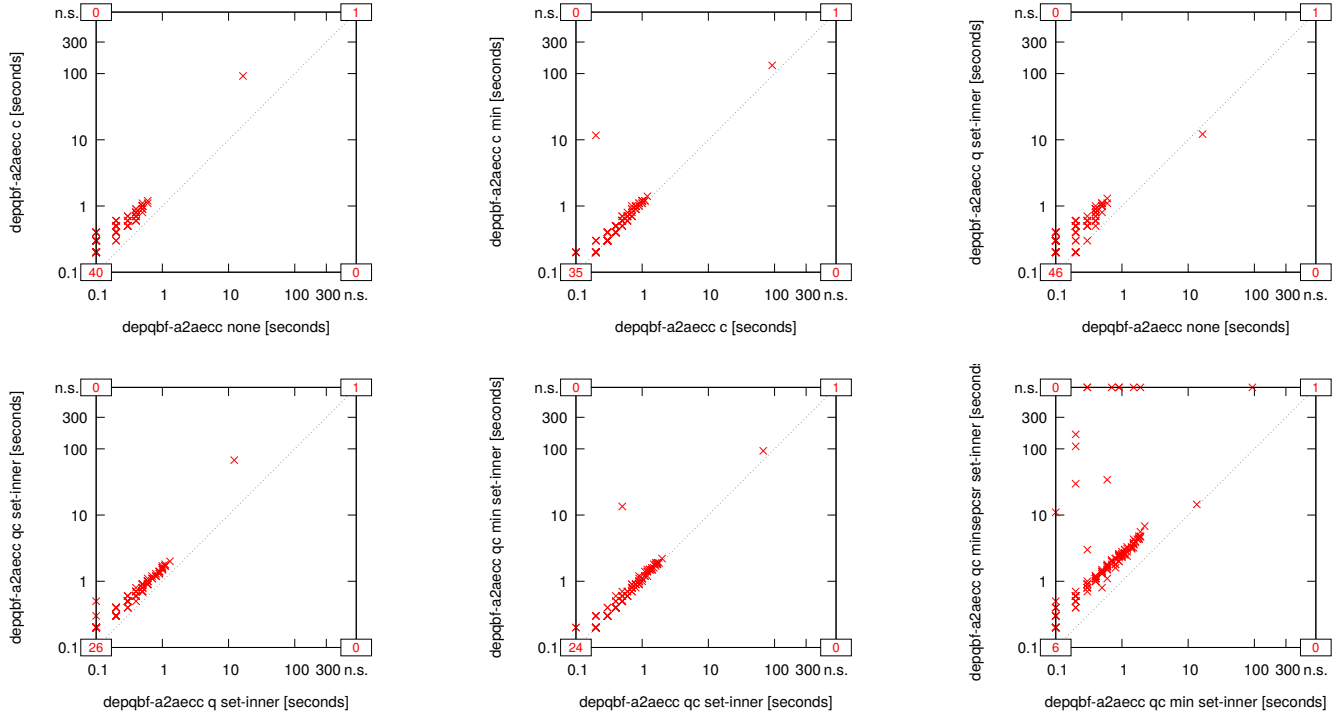


Fig. 697: Suite Gent-Rowley ($n = 142$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

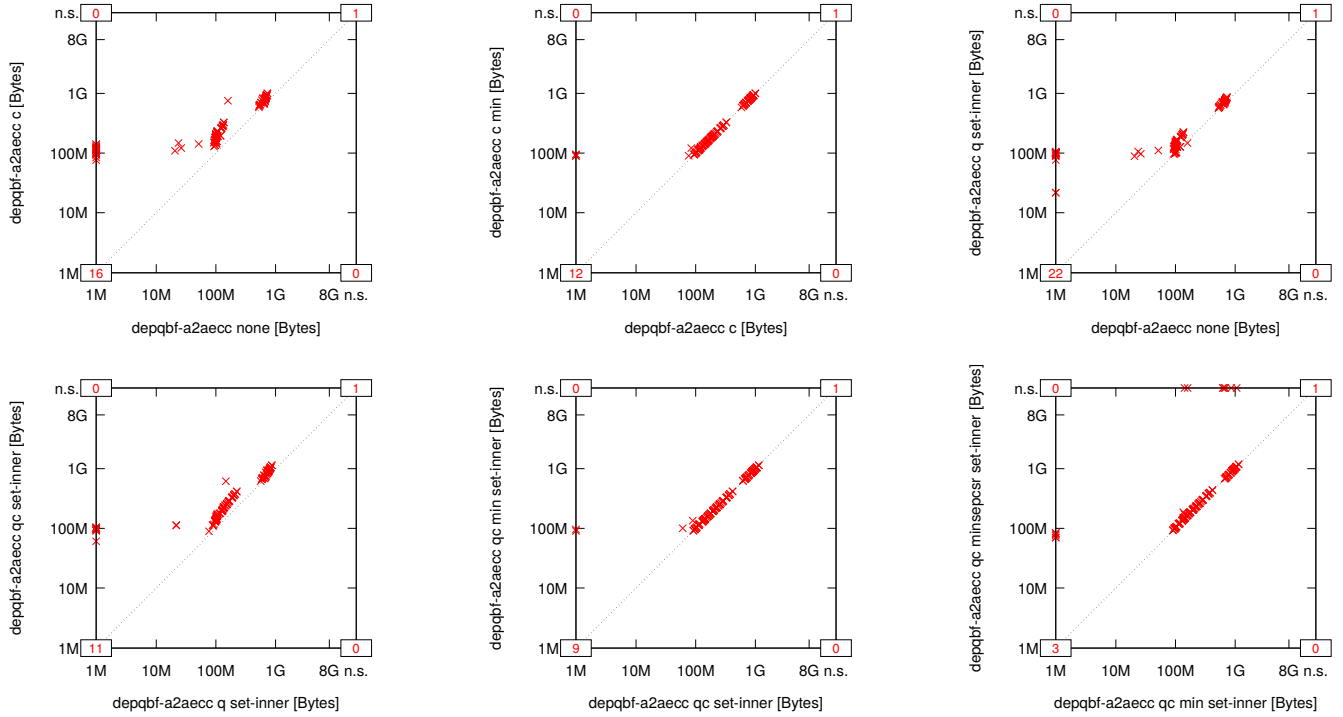


Fig. 698: Suite Gent-Rowley ($n = 142$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

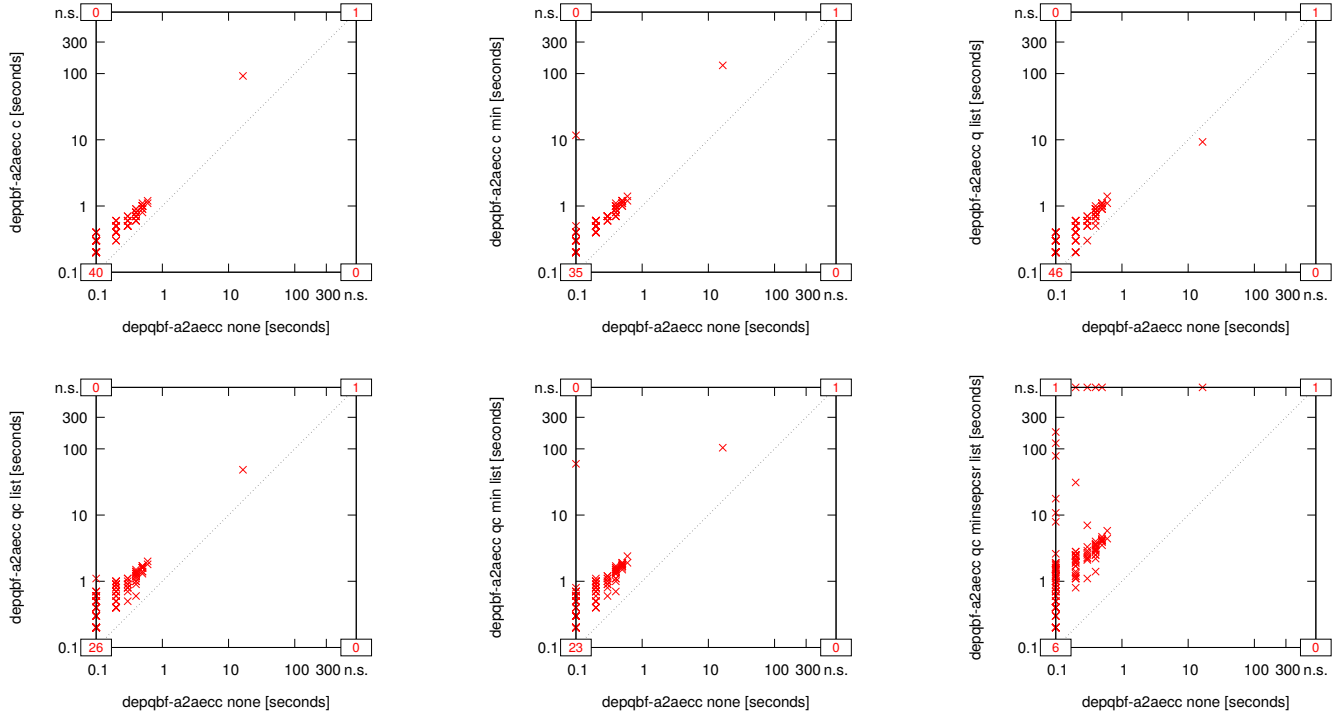


Fig. 699: Suite Gent-Rowley ($n = 142$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

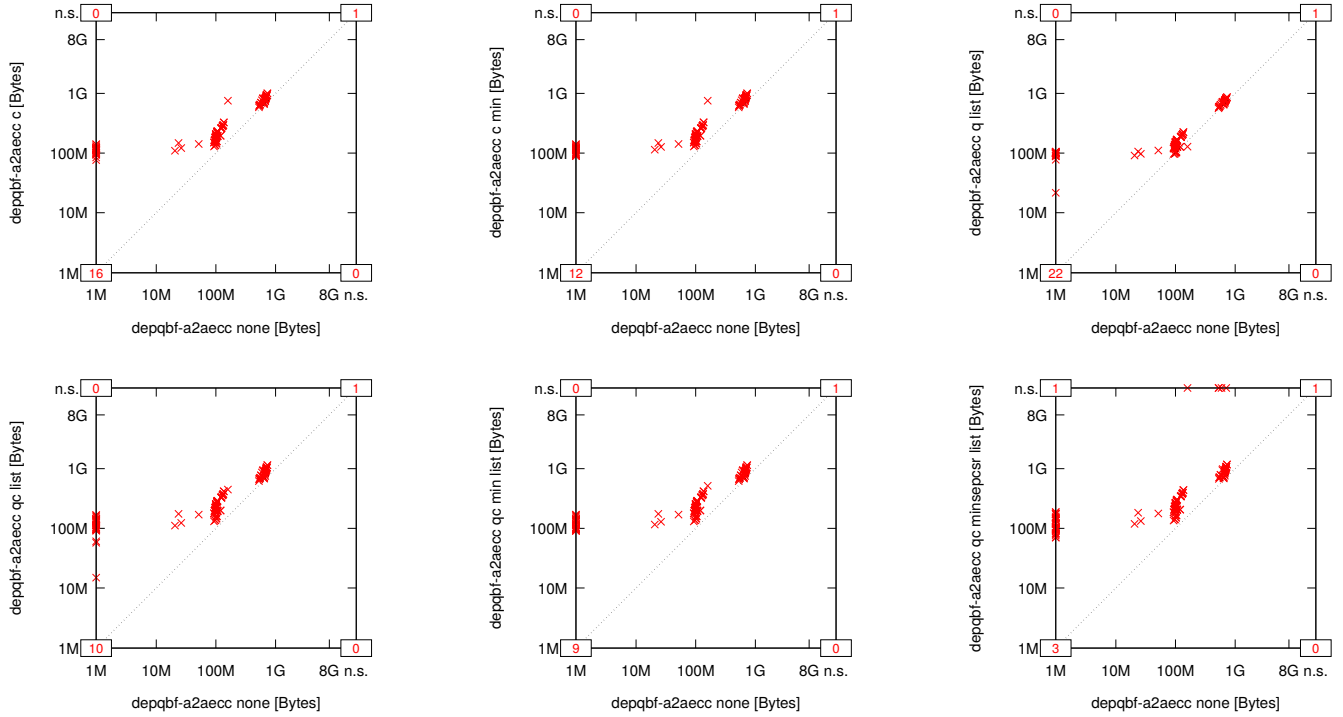


Fig. 700: Suite Gent-Rowley ($n = 142$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

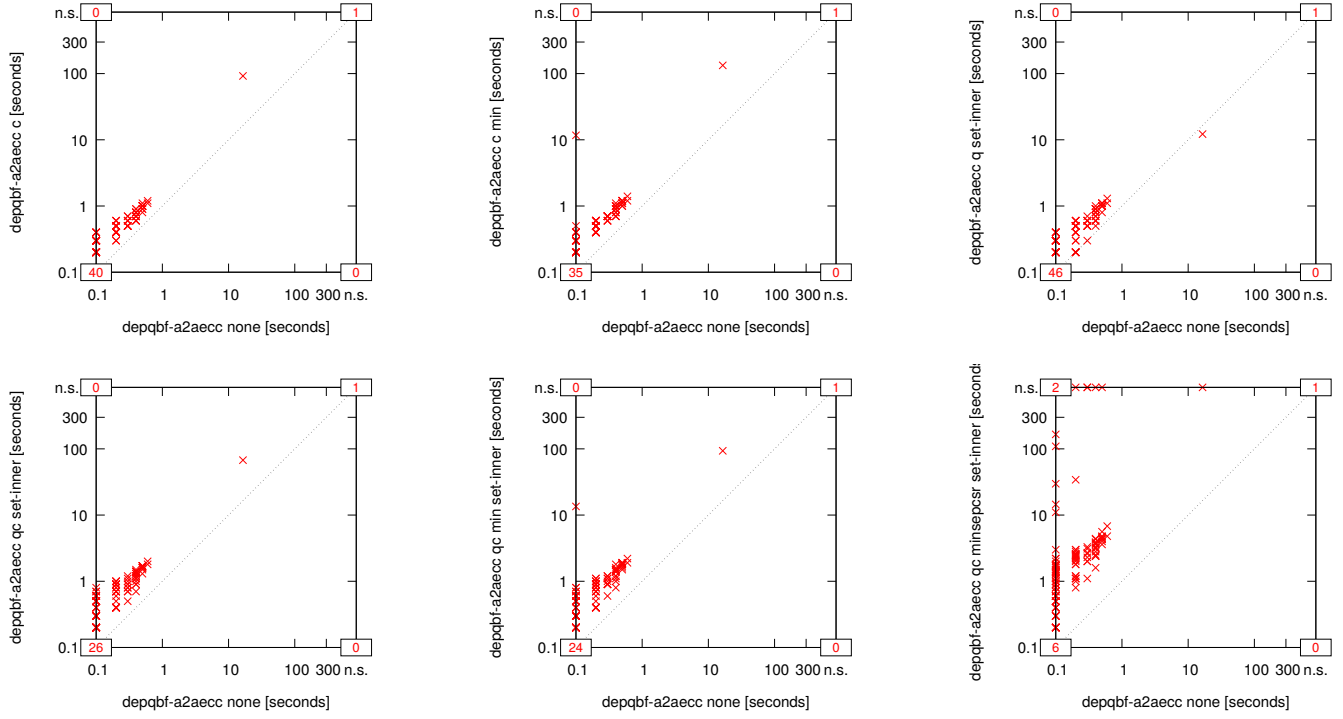


Fig. 701: Suite Gent-Rowley ($n = 142$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. Dep-QBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

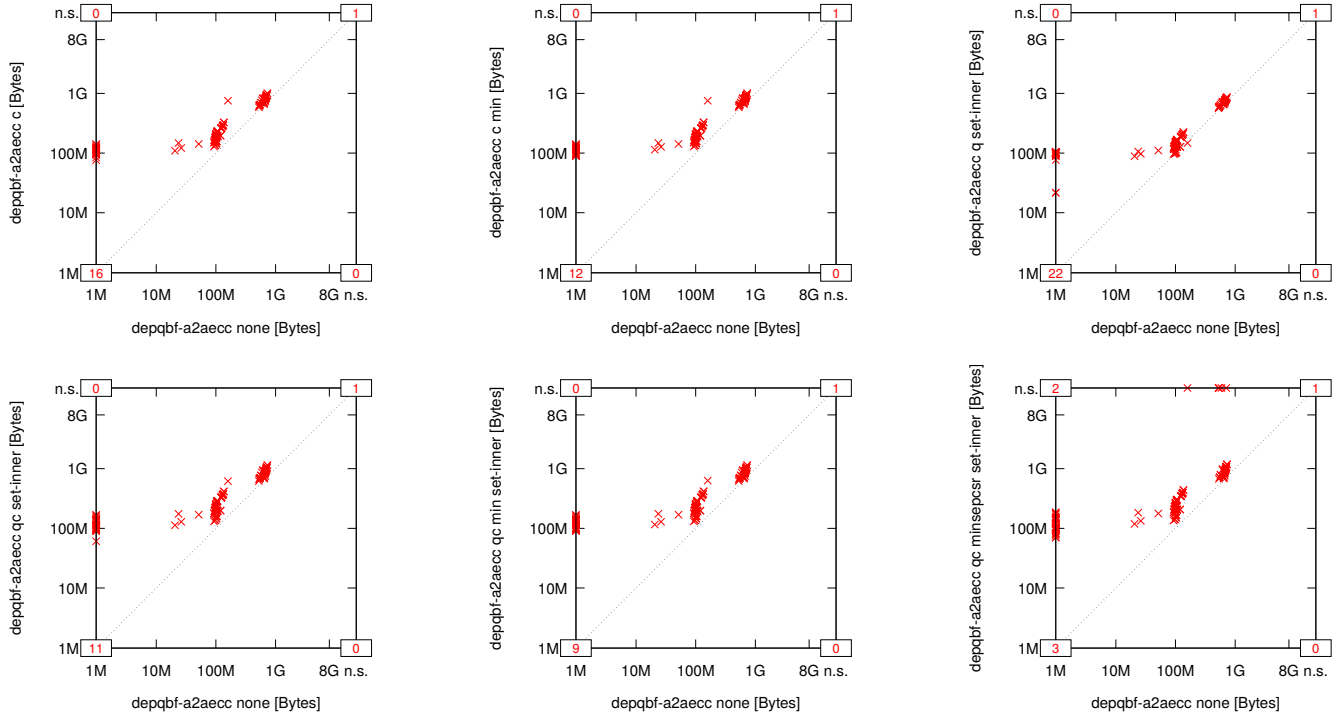


Fig. 702: Suite Gent-Rowley ($n = 142$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. Dep-QBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

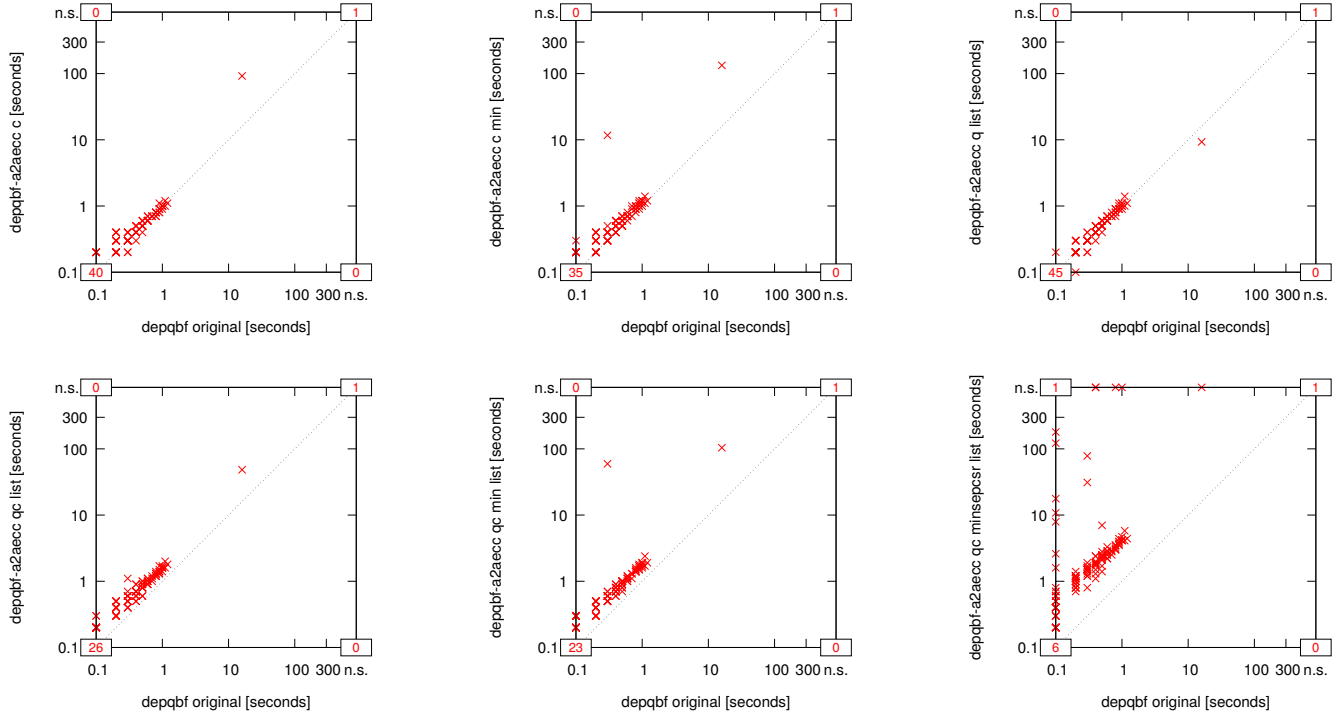


Fig. 703: Suite Gent-Rowley ($n = 142$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

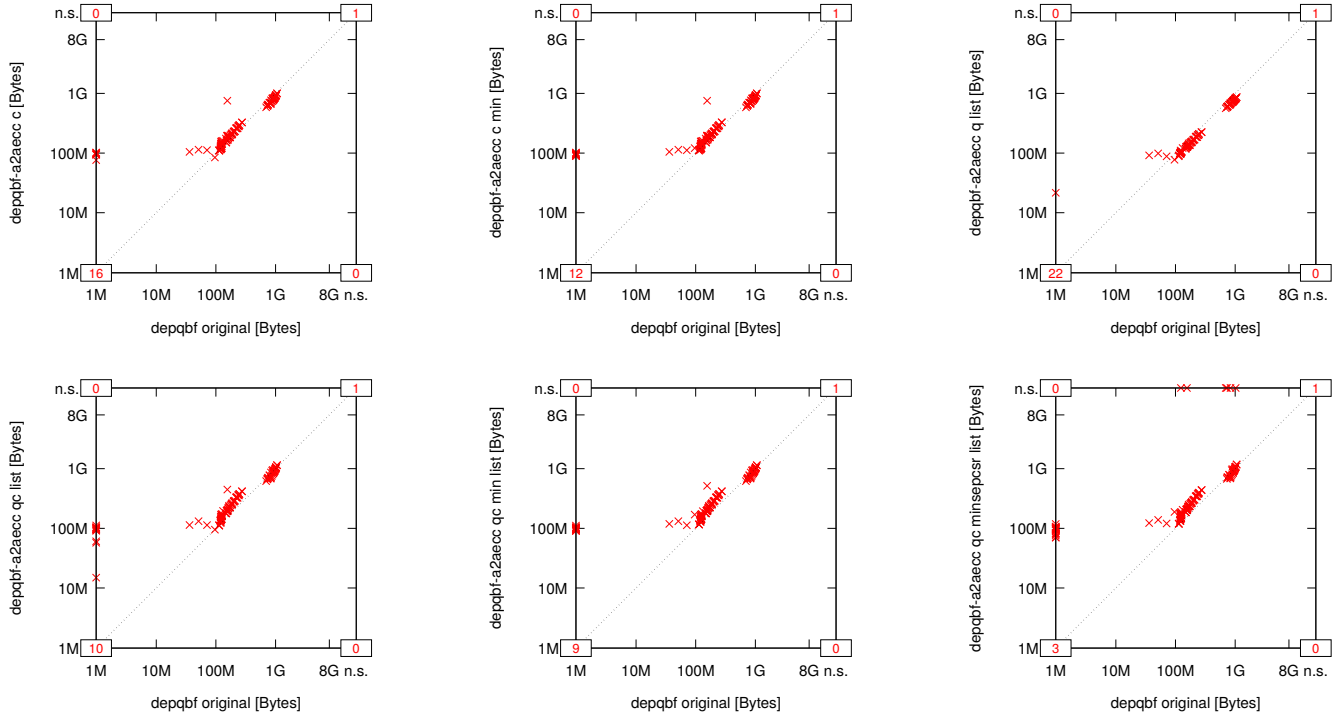


Fig. 704: Suite Gent-Rowley ($n = 142$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

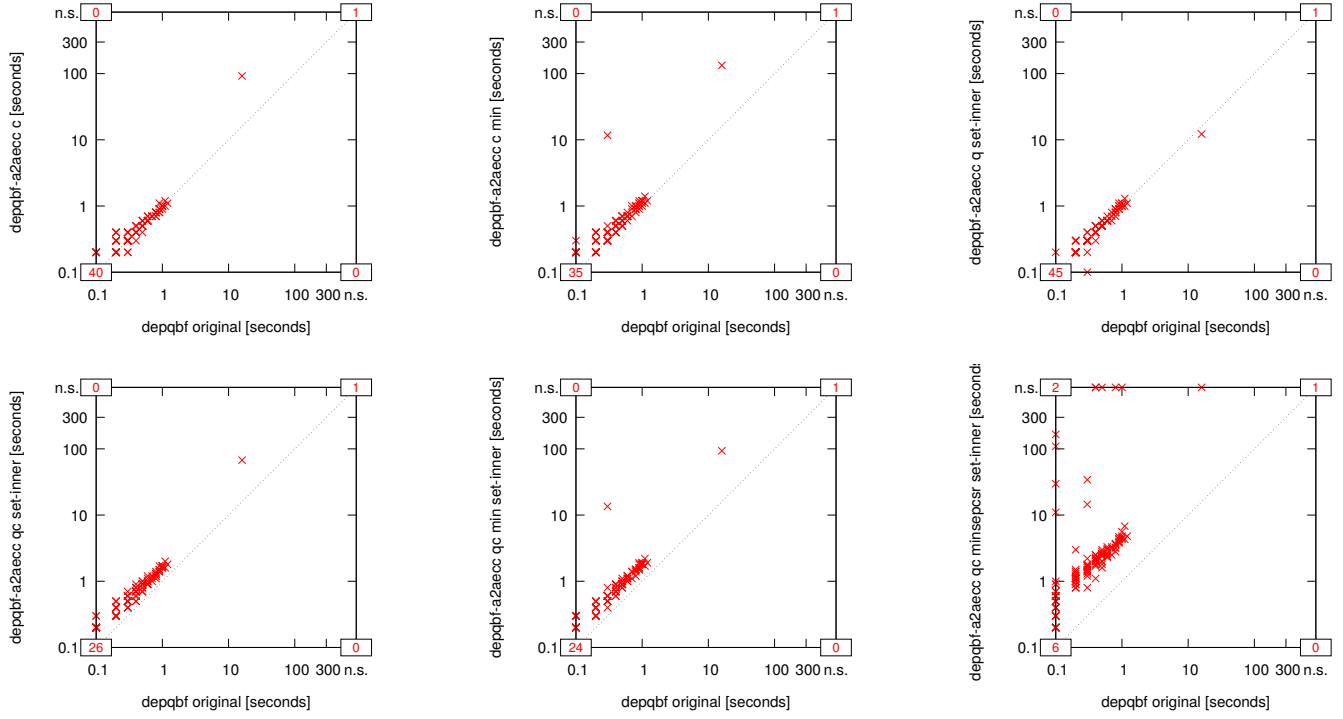


Fig. 705: Suite Gent-Rowley ($n = 142$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

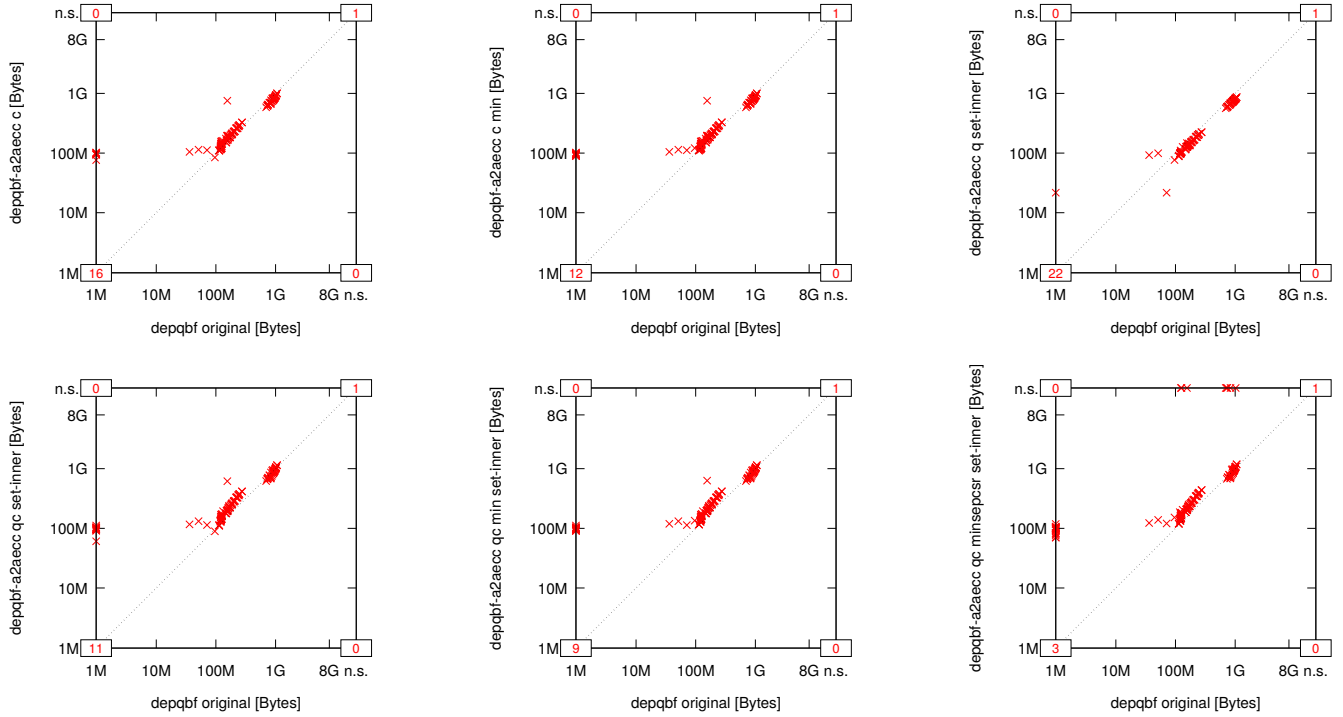


Fig. 706: Suite Gent-Rowley ($n = 142$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

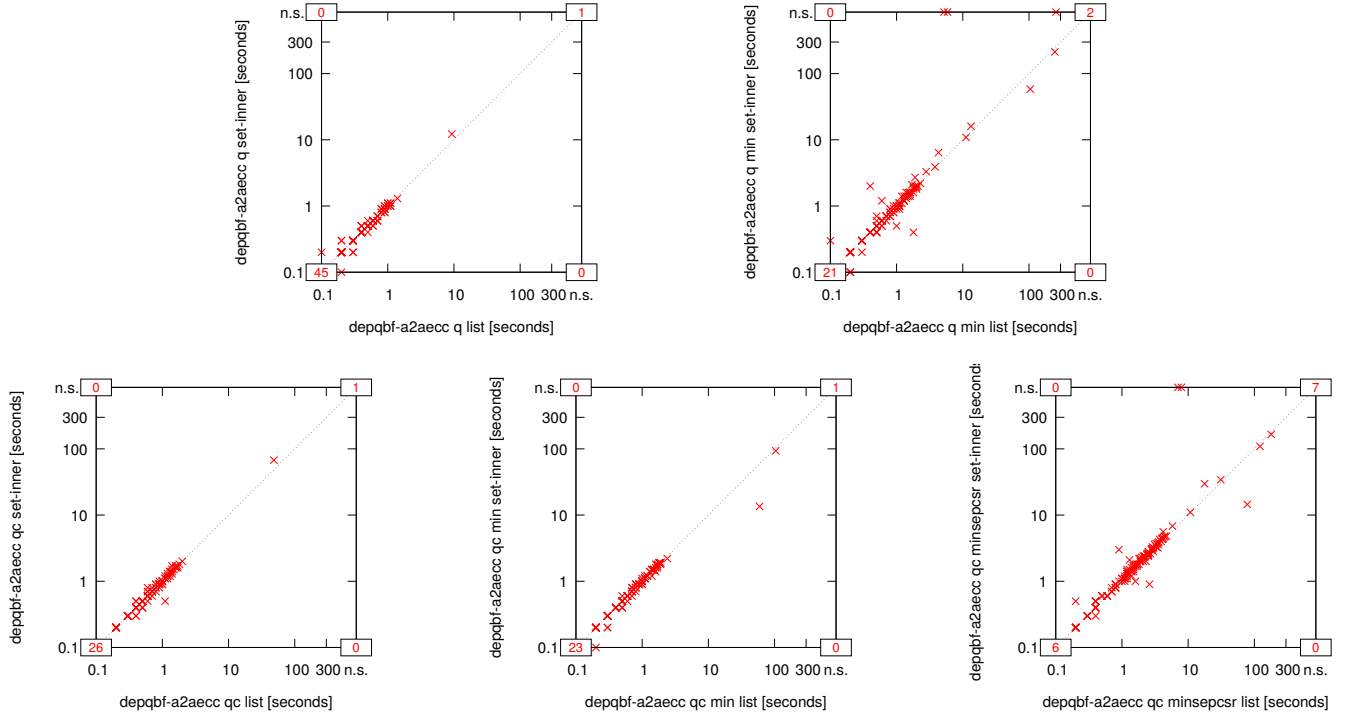


Fig. 707: Suite Gent-Rowley ($n = 142$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

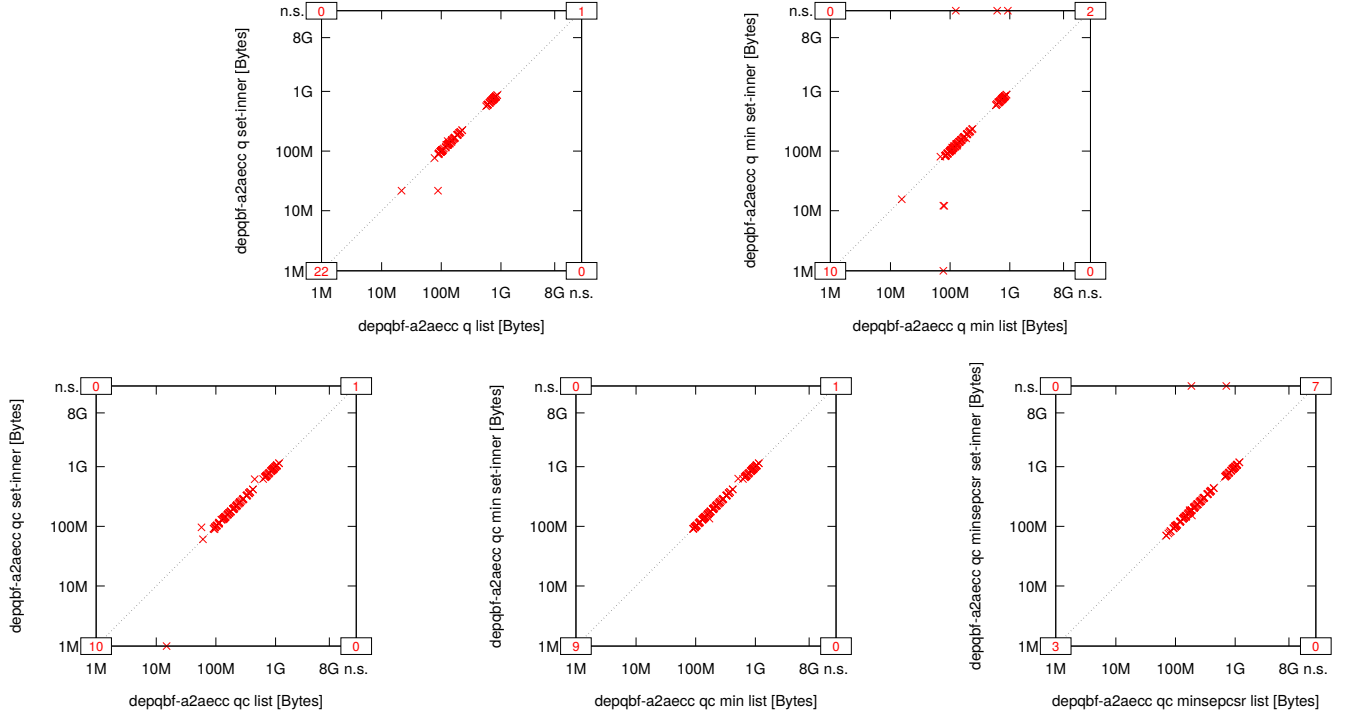


Fig. 708: Suite Gent-Rowley ($n = 142$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

15) *Herbstritt* ($n = 157$):

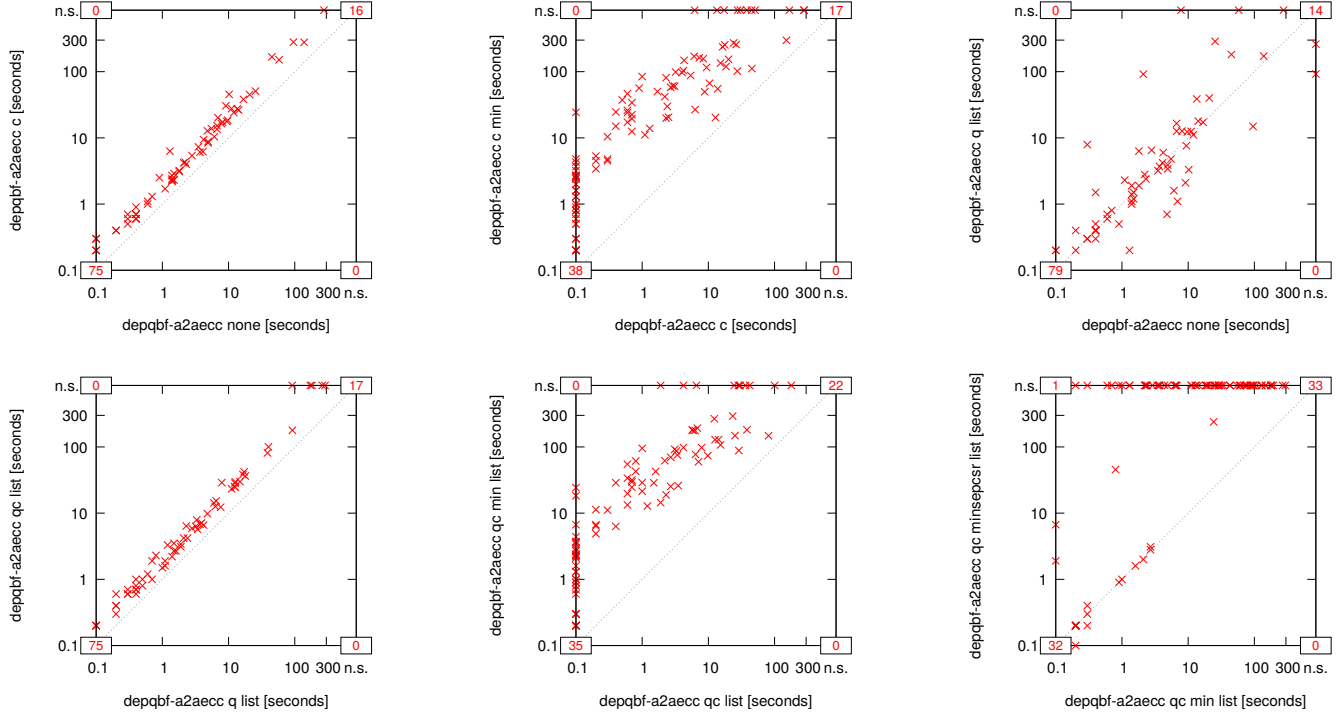


Fig. 709: Suite *Herbstritt* ($n = 157$): Comparing run times for extracting different unsatisfiable cores in *DepQBF-a2aecc* with list semantics (run time in seconds).

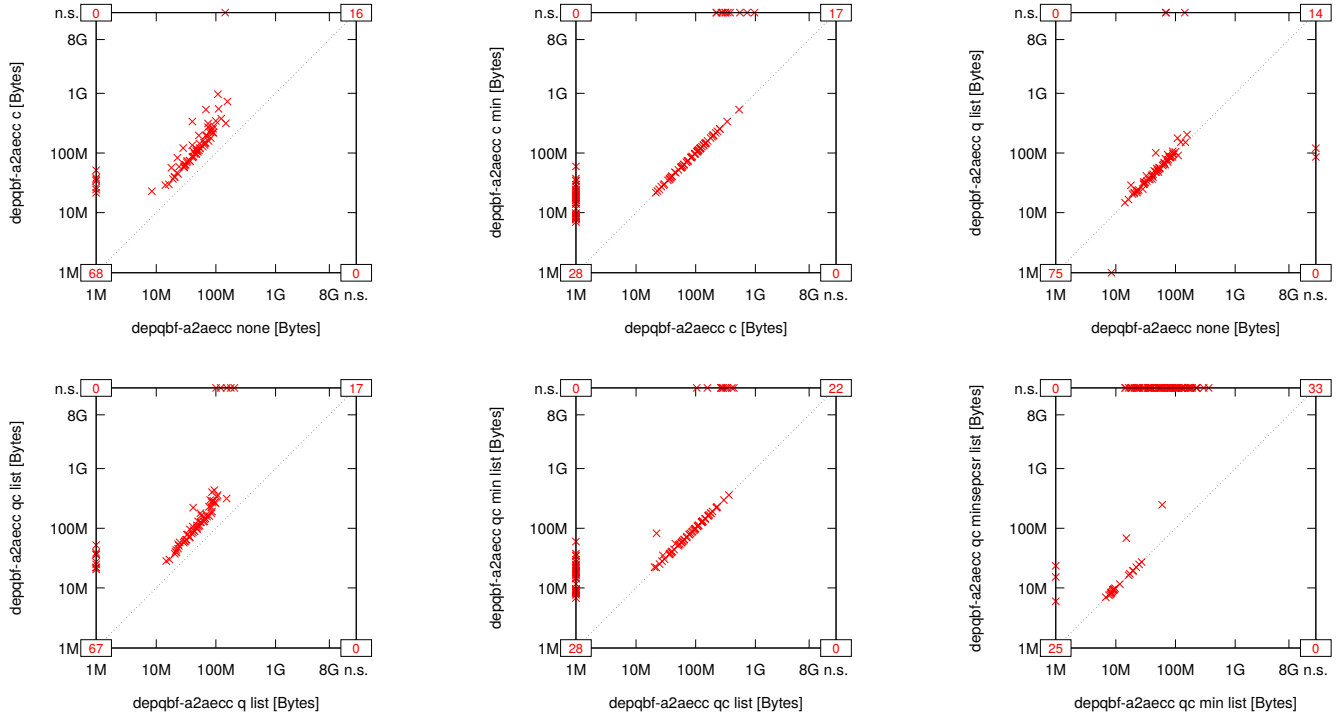


Fig. 710: Suite *Herbstritt* ($n = 157$): Comparing memory usage for extracting different unsatisfiable cores in *DepQBF-a2aecc* with list semantics (memory usage in Bytes).

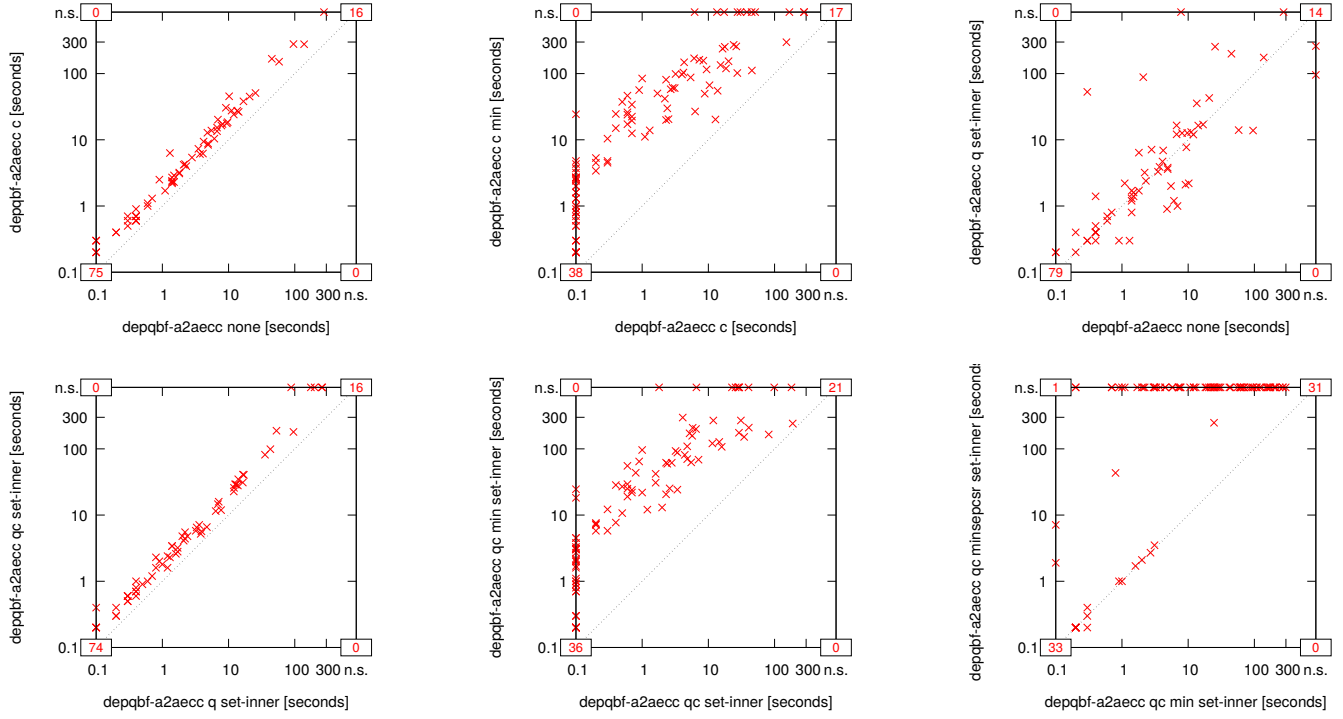


Fig. 711: Suite Herbstritt ($n = 157$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

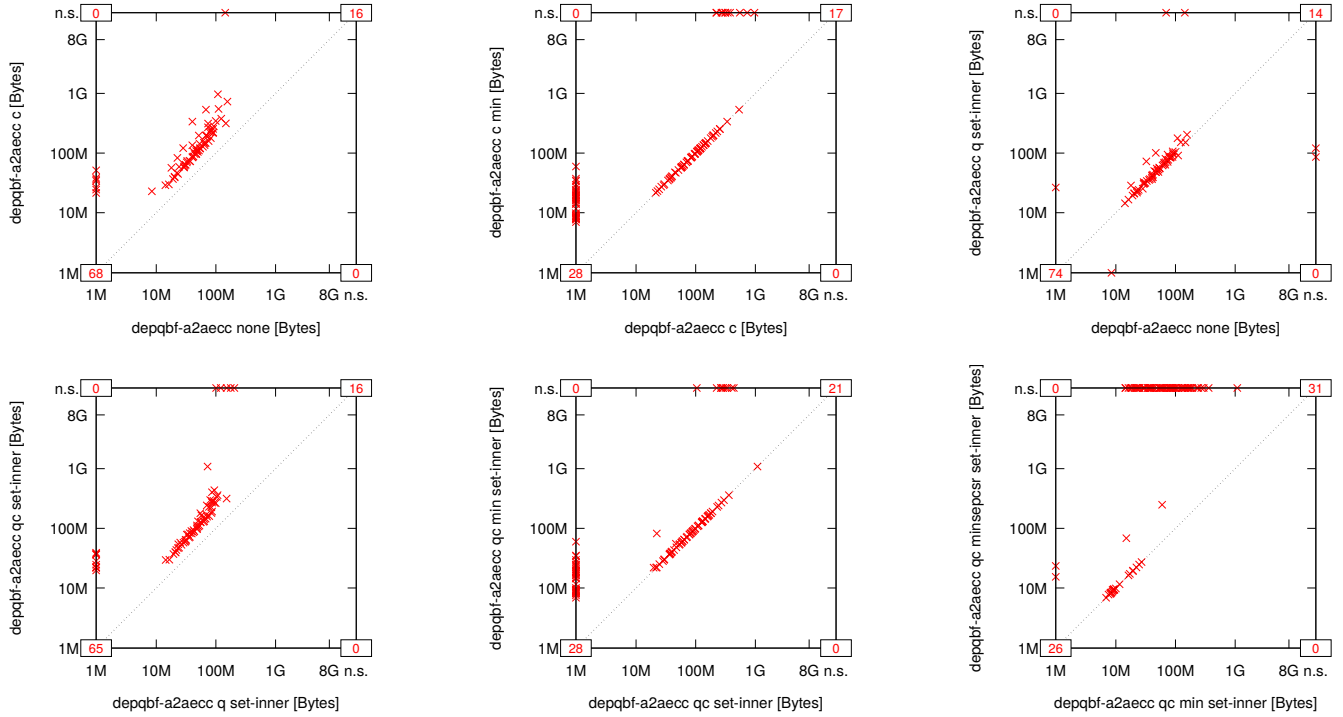


Fig. 712: Suite Herbstritt ($n = 157$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

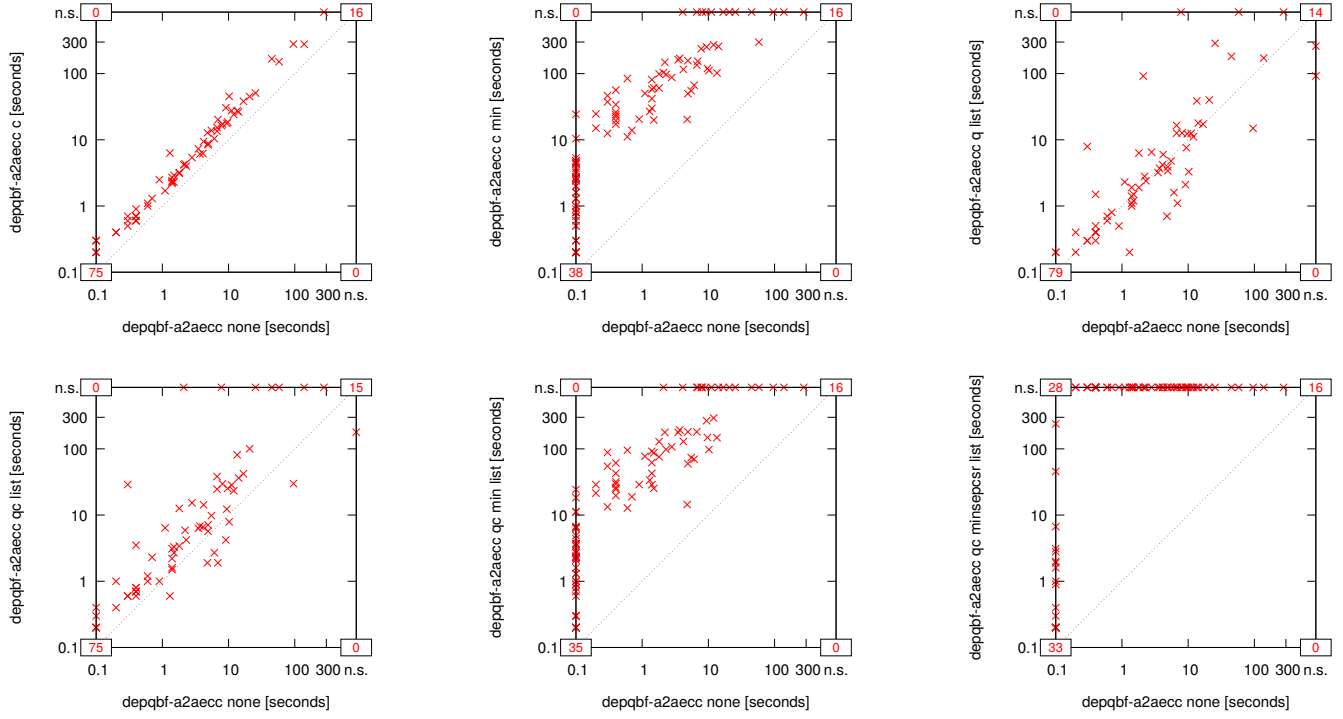


Fig. 713: Suite Herbstritt ($n = 157$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

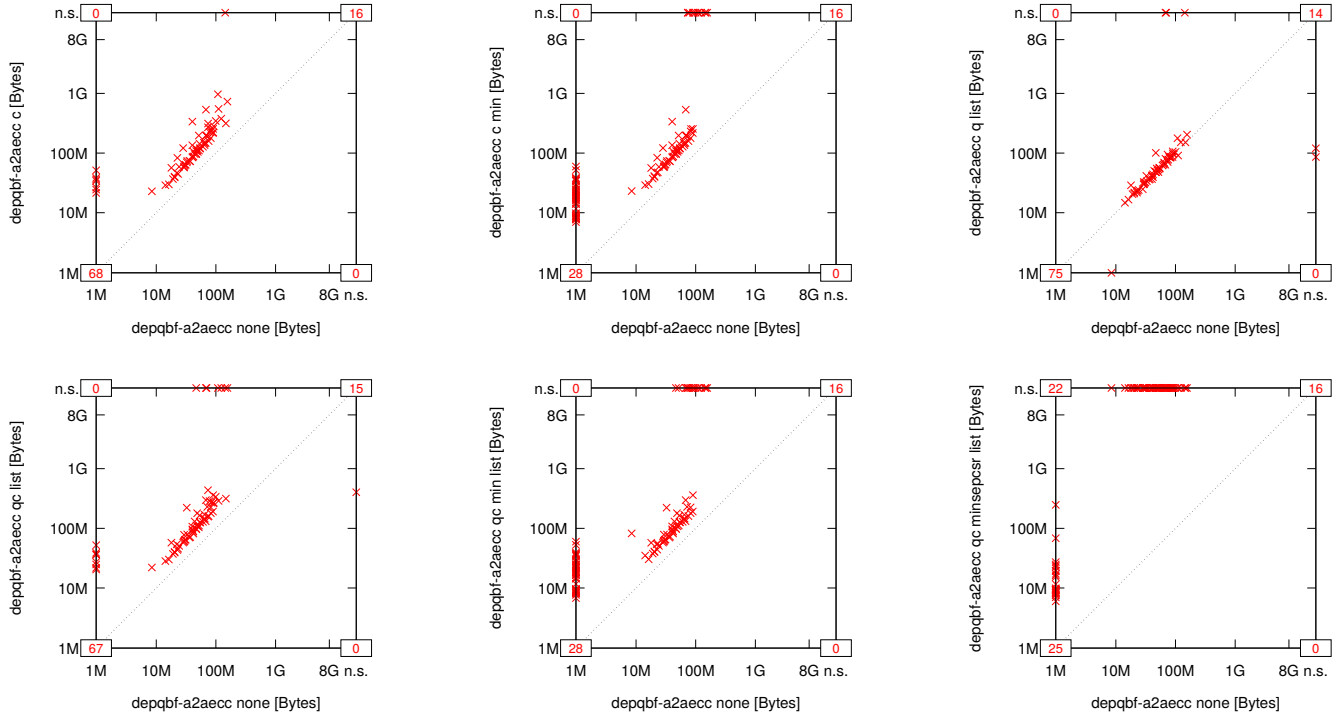


Fig. 714: Suite Herbstritt ($n = 157$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

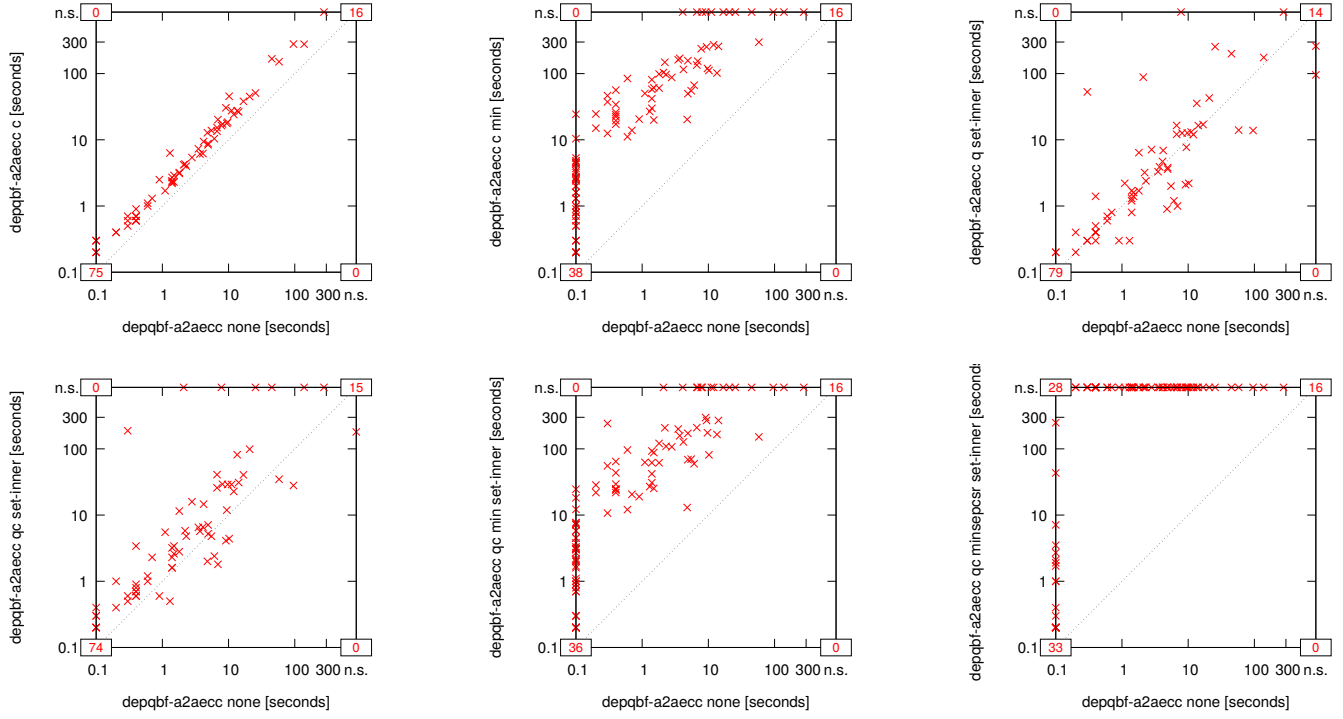


Fig. 715: Suite Herbstritt ($n = 157$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

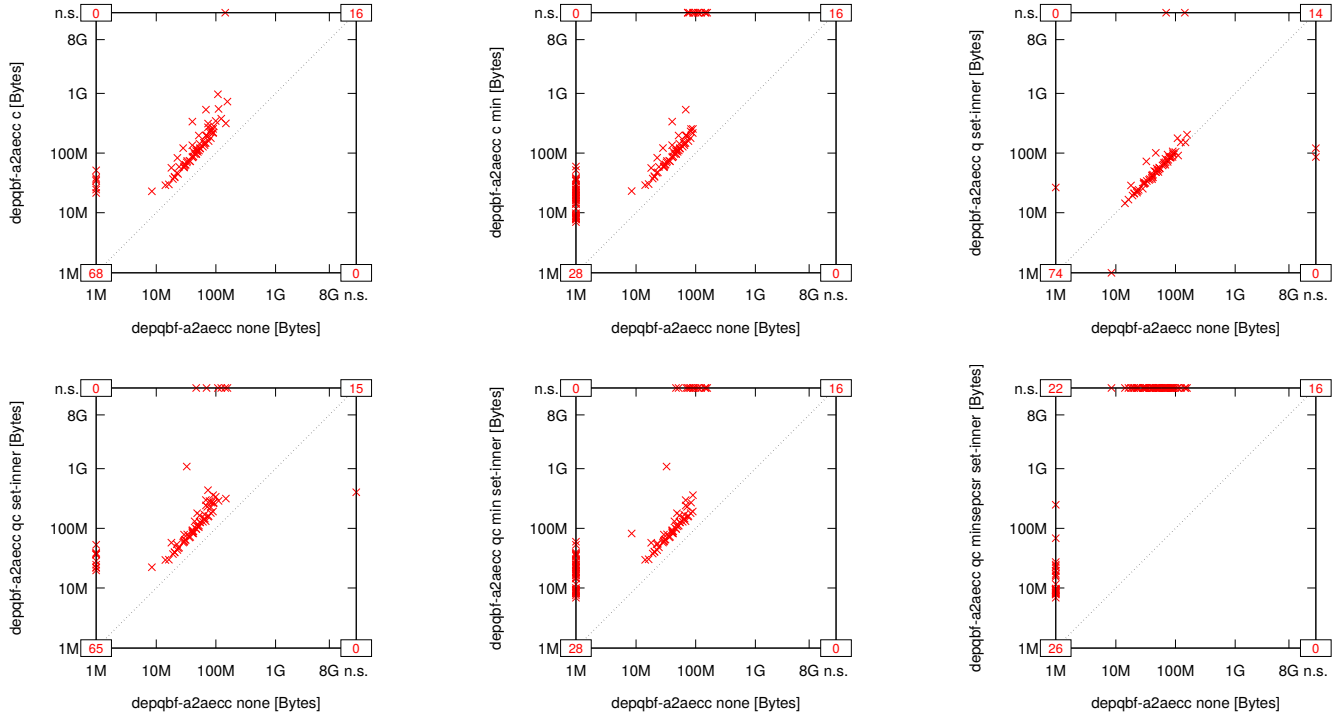


Fig. 716: Suite Herbstritt ($n = 157$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

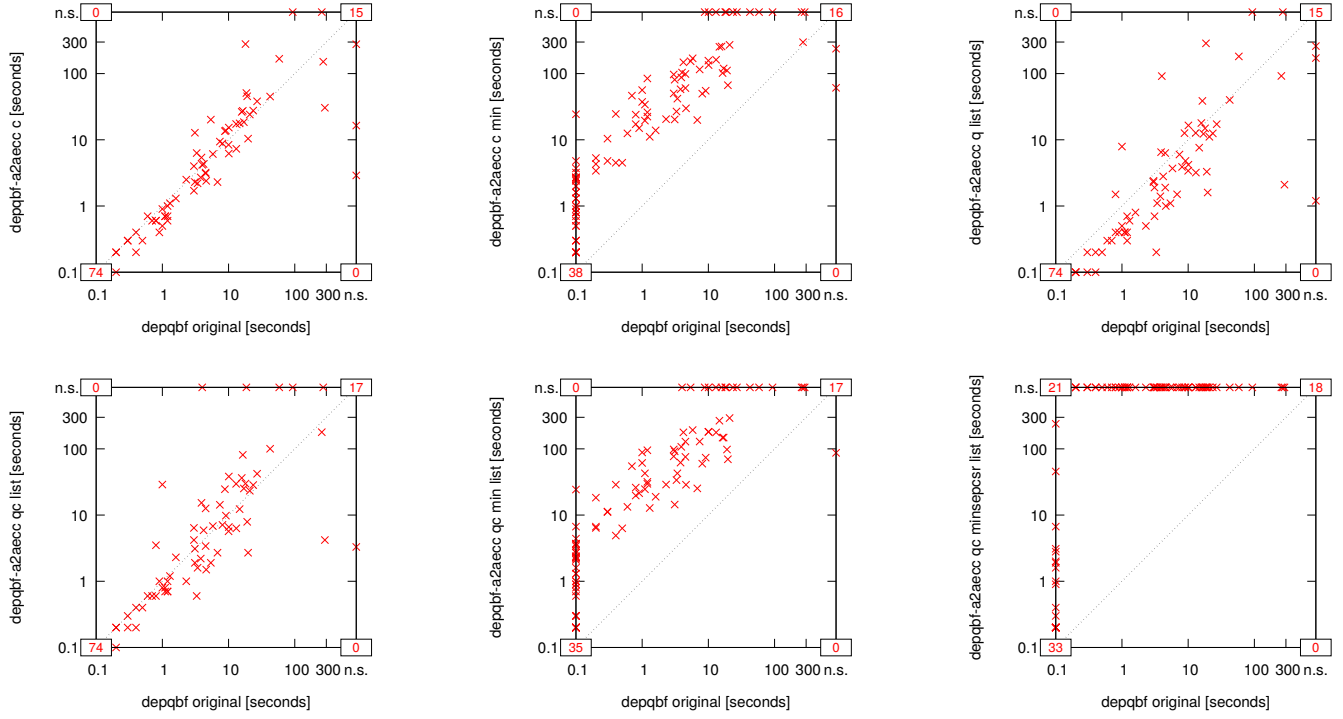


Fig. 717: Suite Herbstritt ($n = 157$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

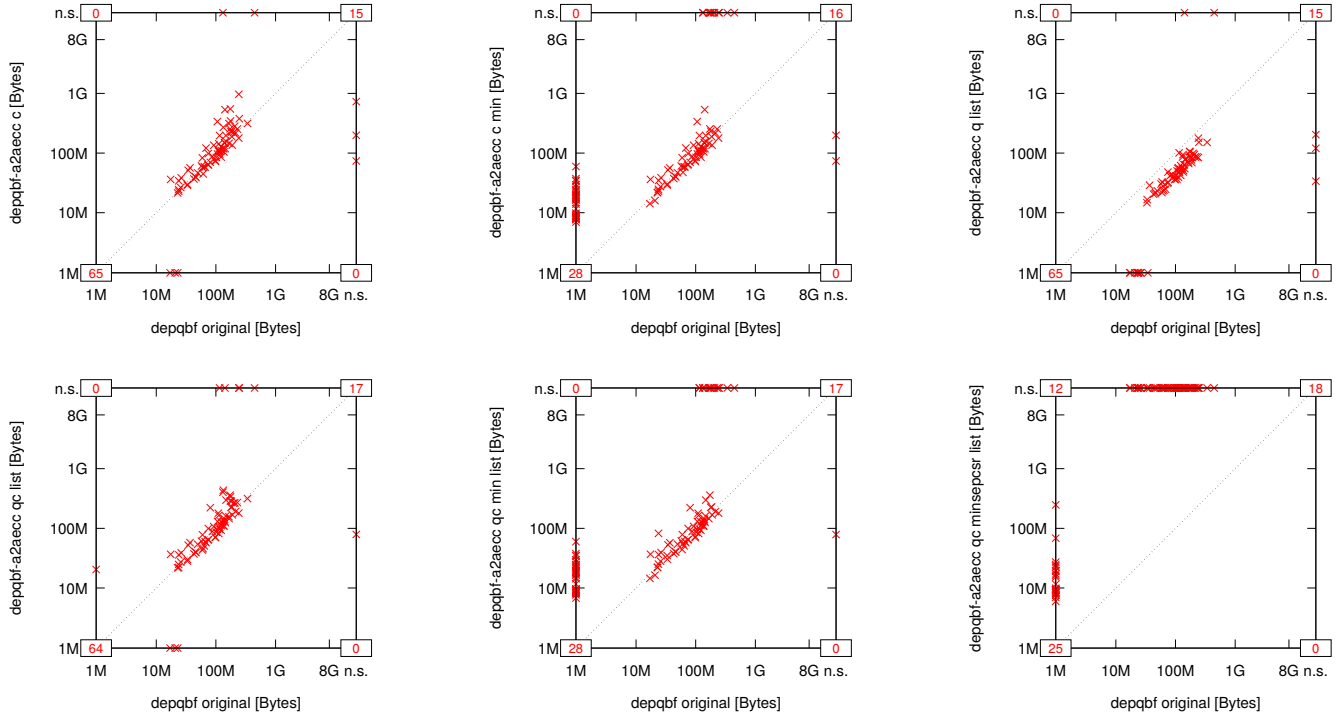


Fig. 718: Suite Herbstritt ($n = 157$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

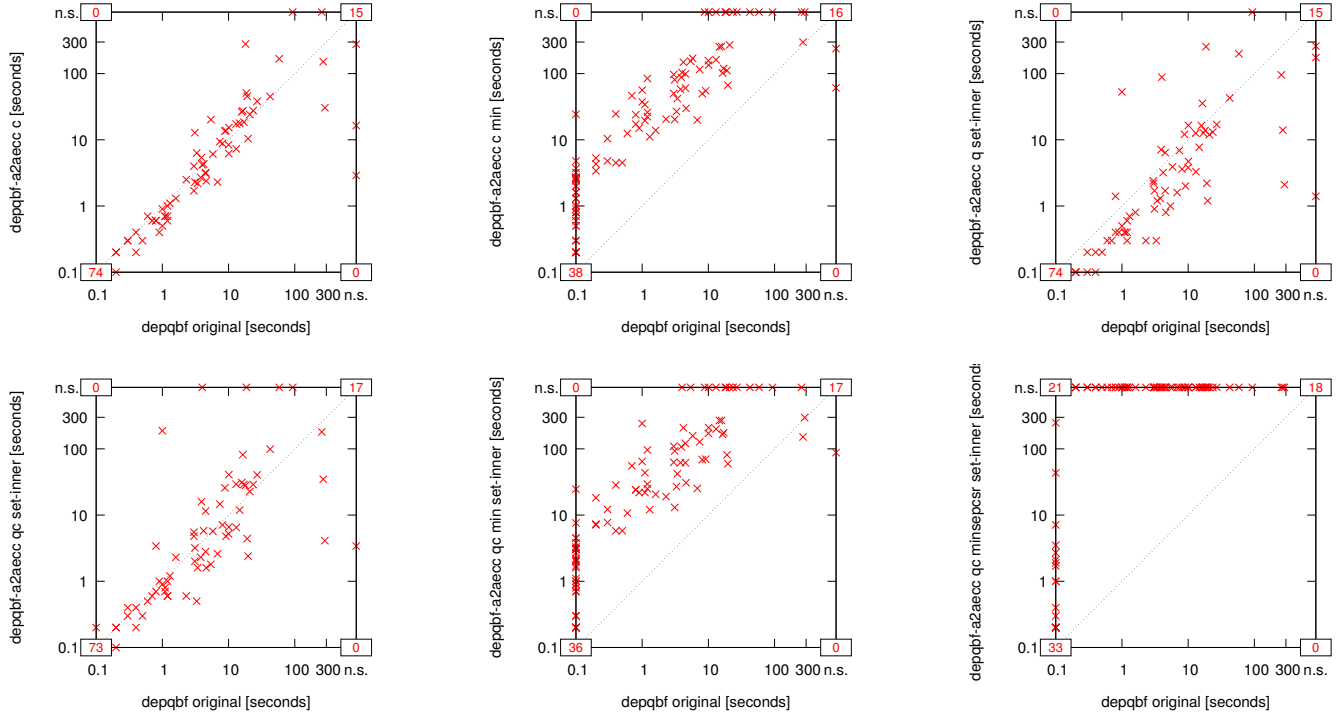


Fig. 719: Suite Herbstritt ($n = 157$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

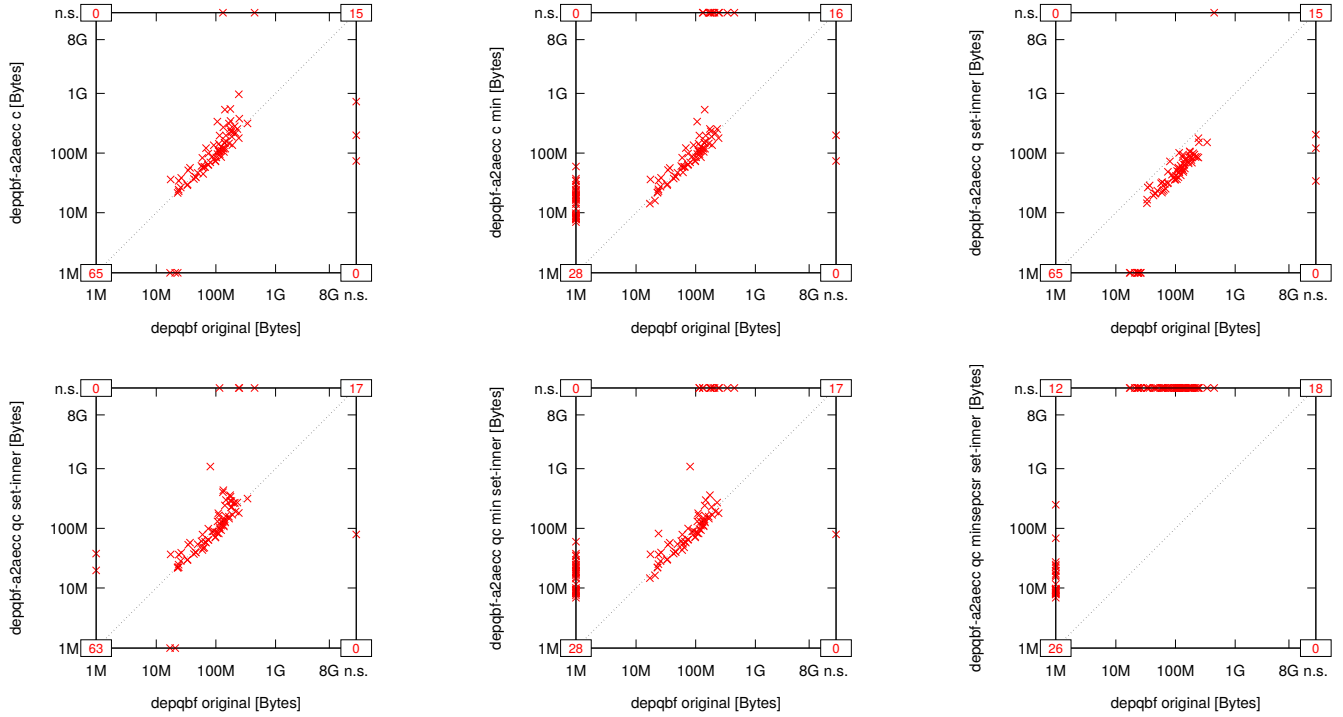


Fig. 720: Suite Herbstritt ($n = 157$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

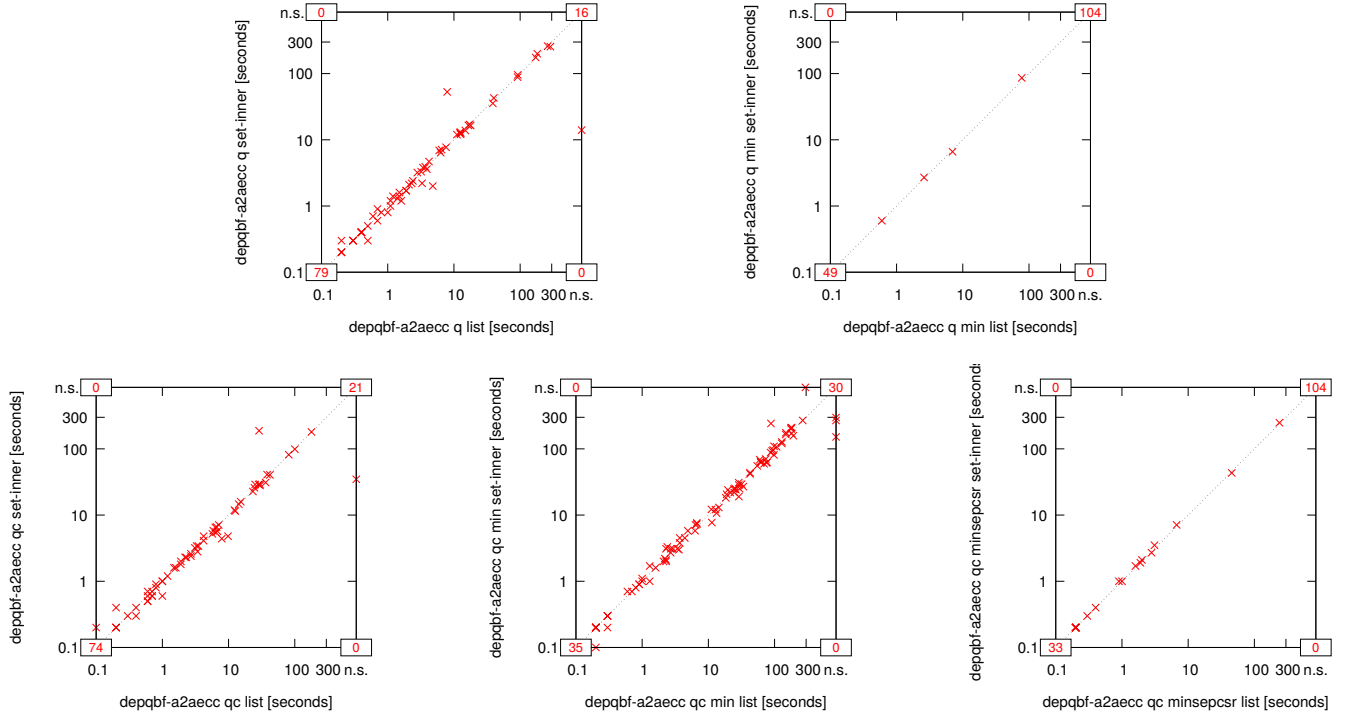


Fig. 721: Suite Herbstritt ($n = 157$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

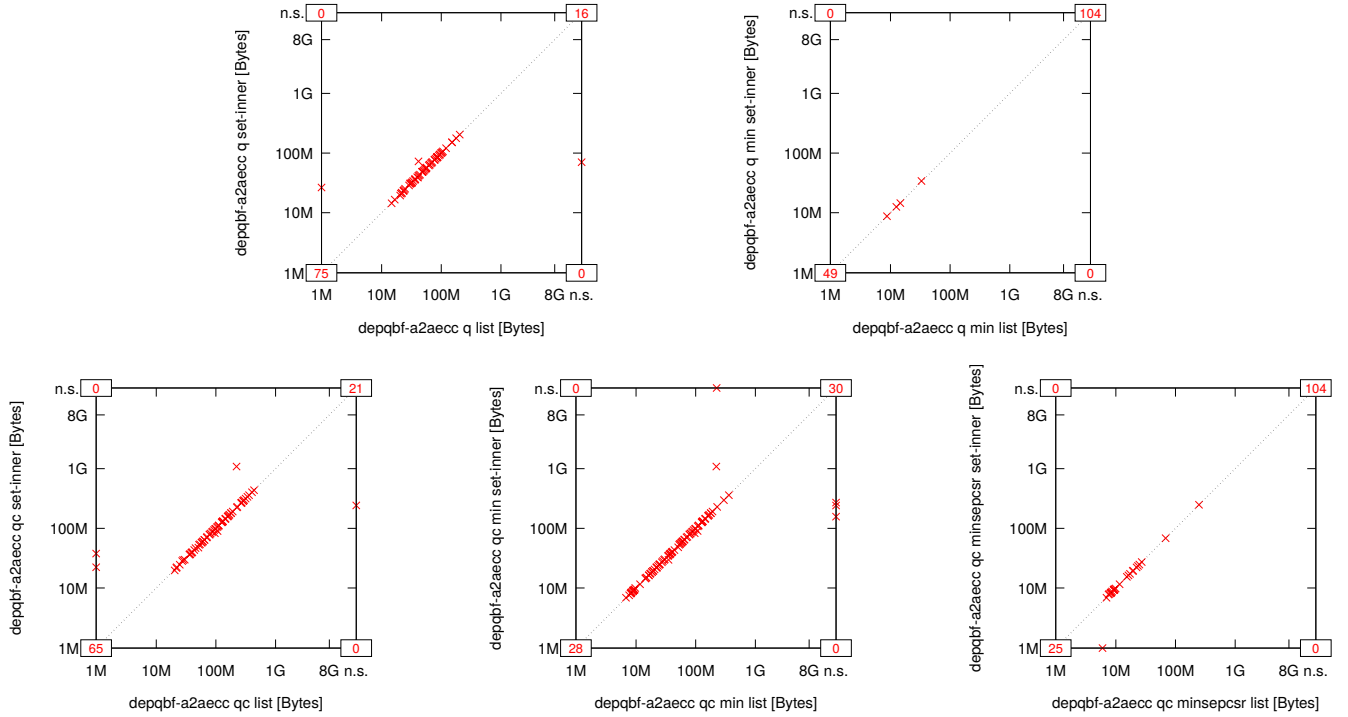


Fig. 722: Suite Herbstritt ($n = 157$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

16) *Interian* ($n = 24$):

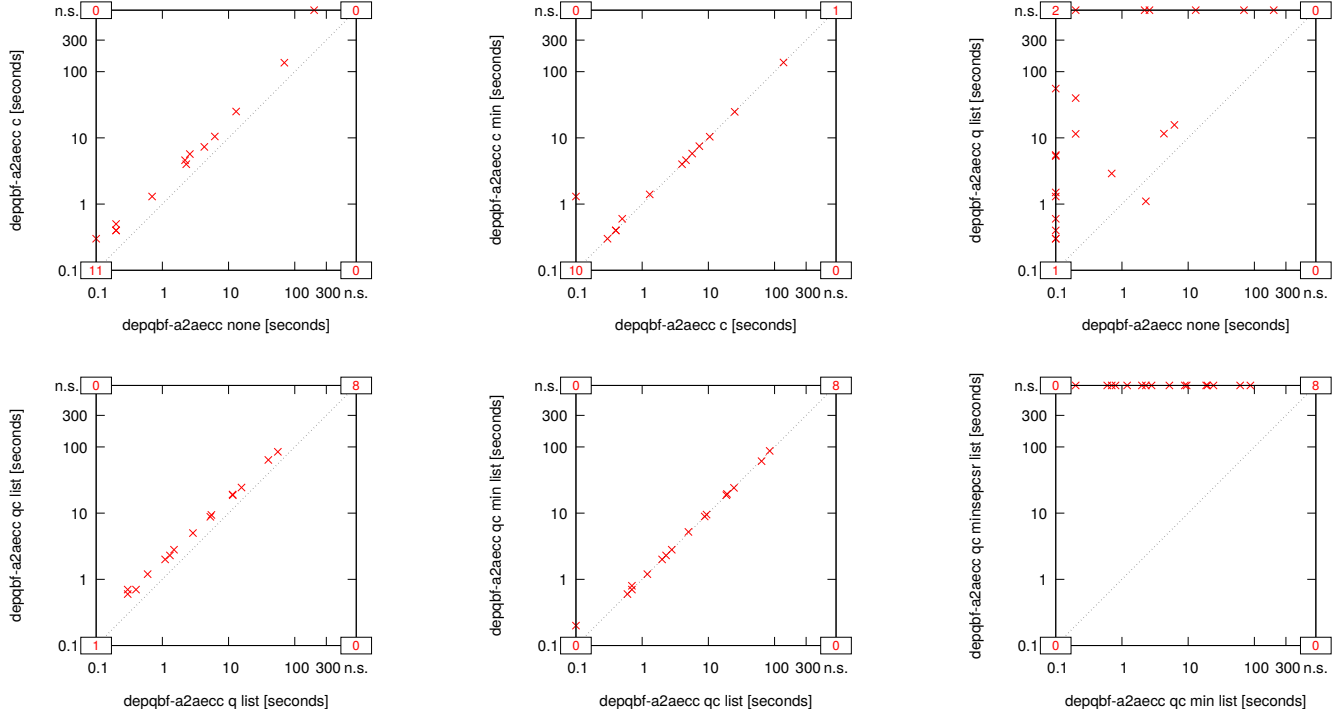


Fig. 723: Suite *Interian* ($n = 24$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

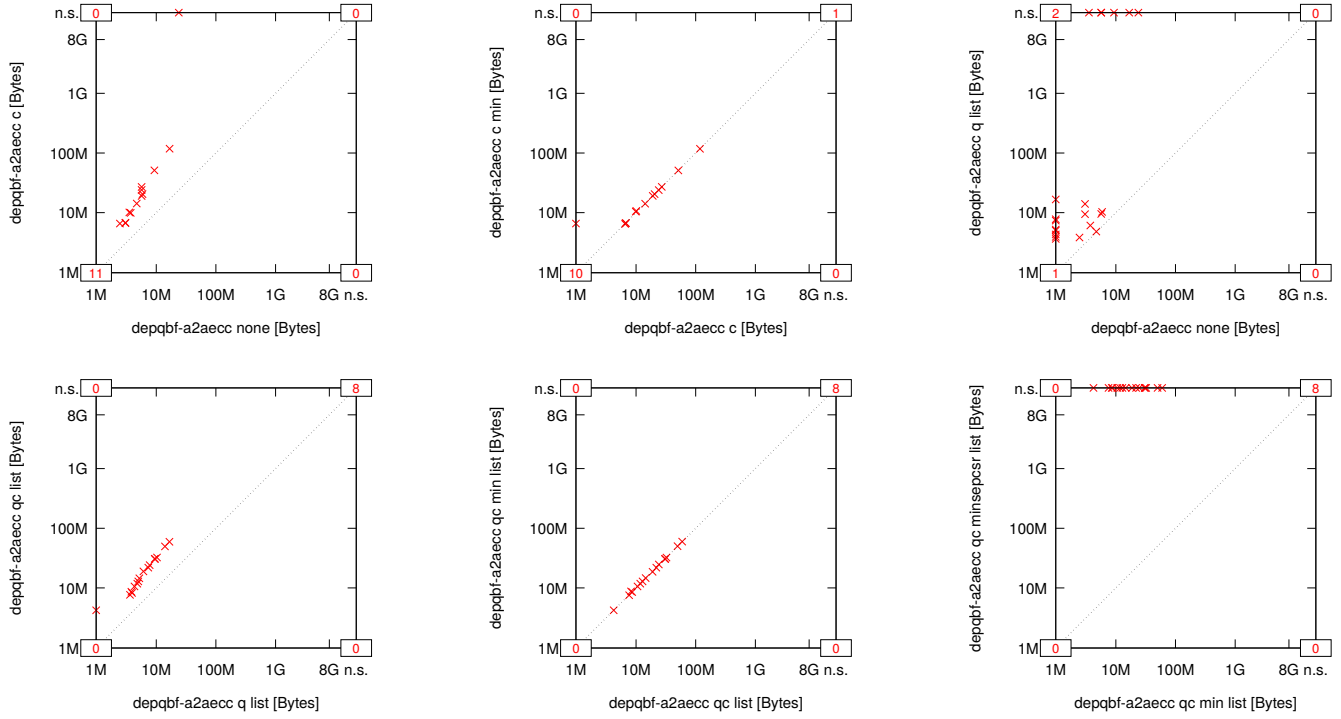


Fig. 724: Suite *Interian* ($n = 24$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

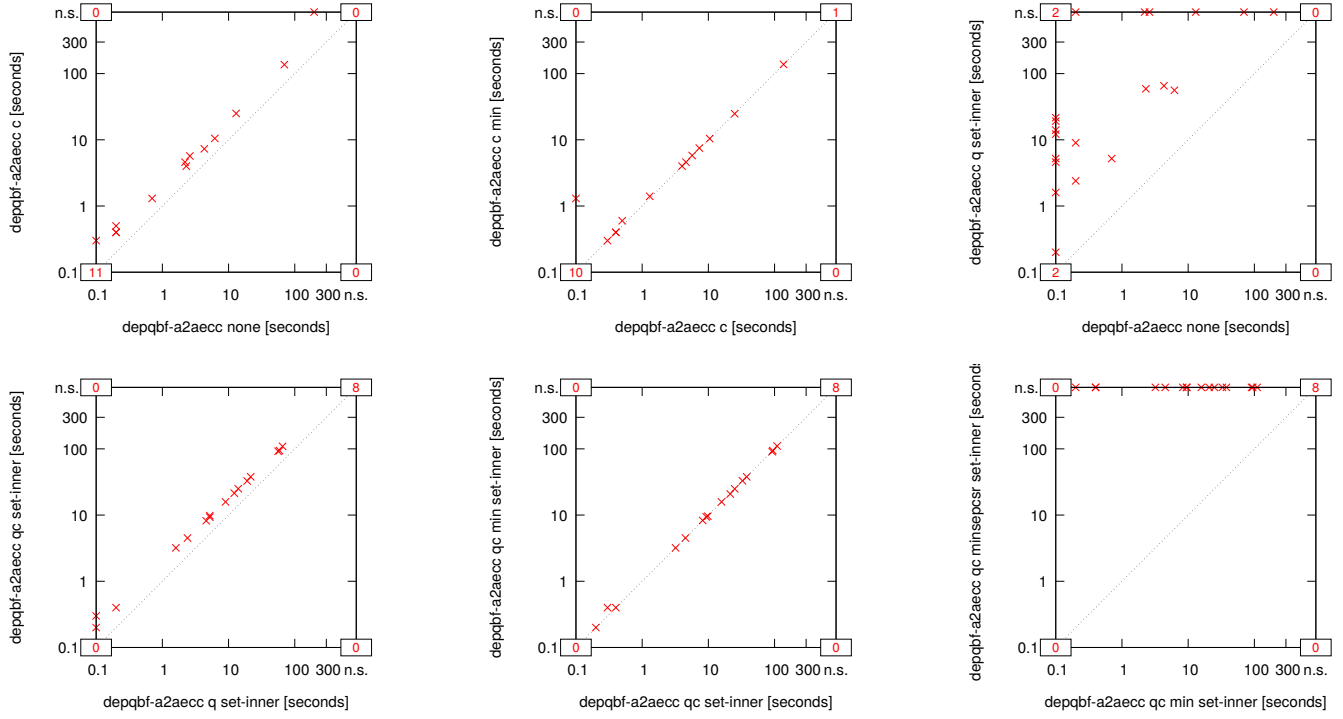


Fig. 725: Suite Interian ($n = 24$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

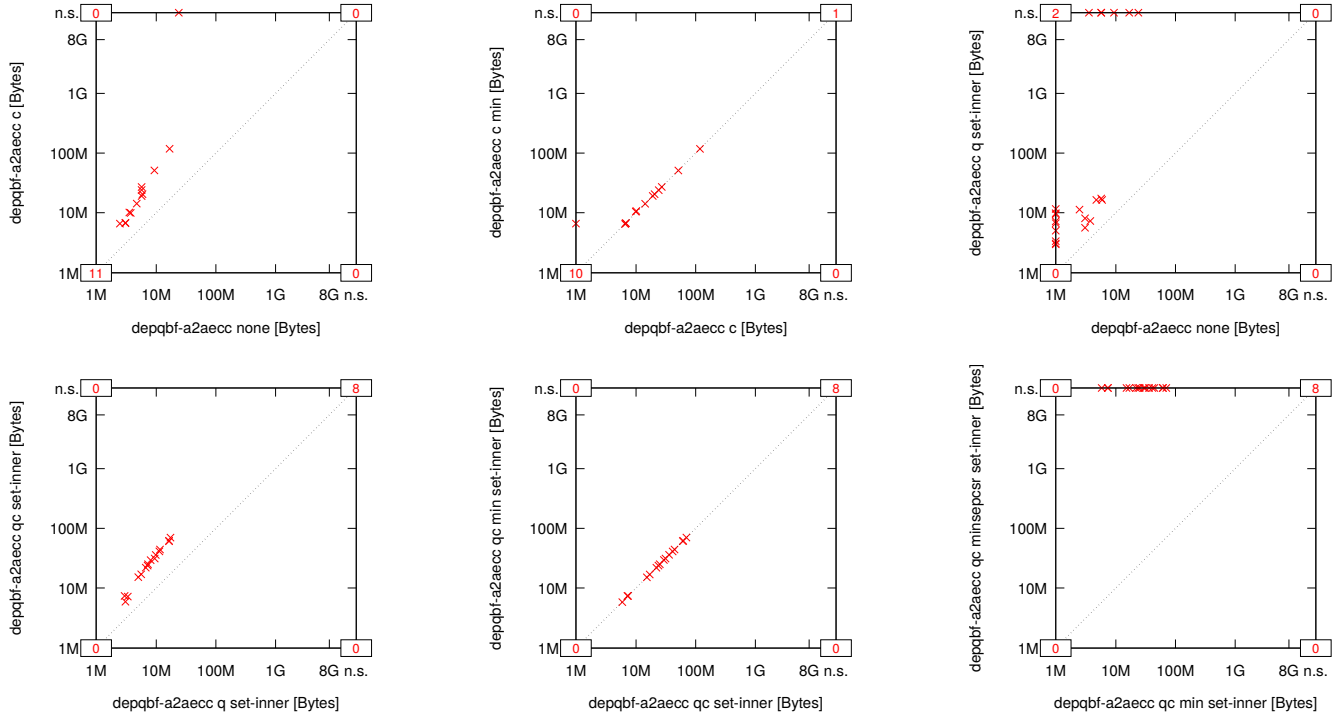


Fig. 726: Suite Interian ($n = 24$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

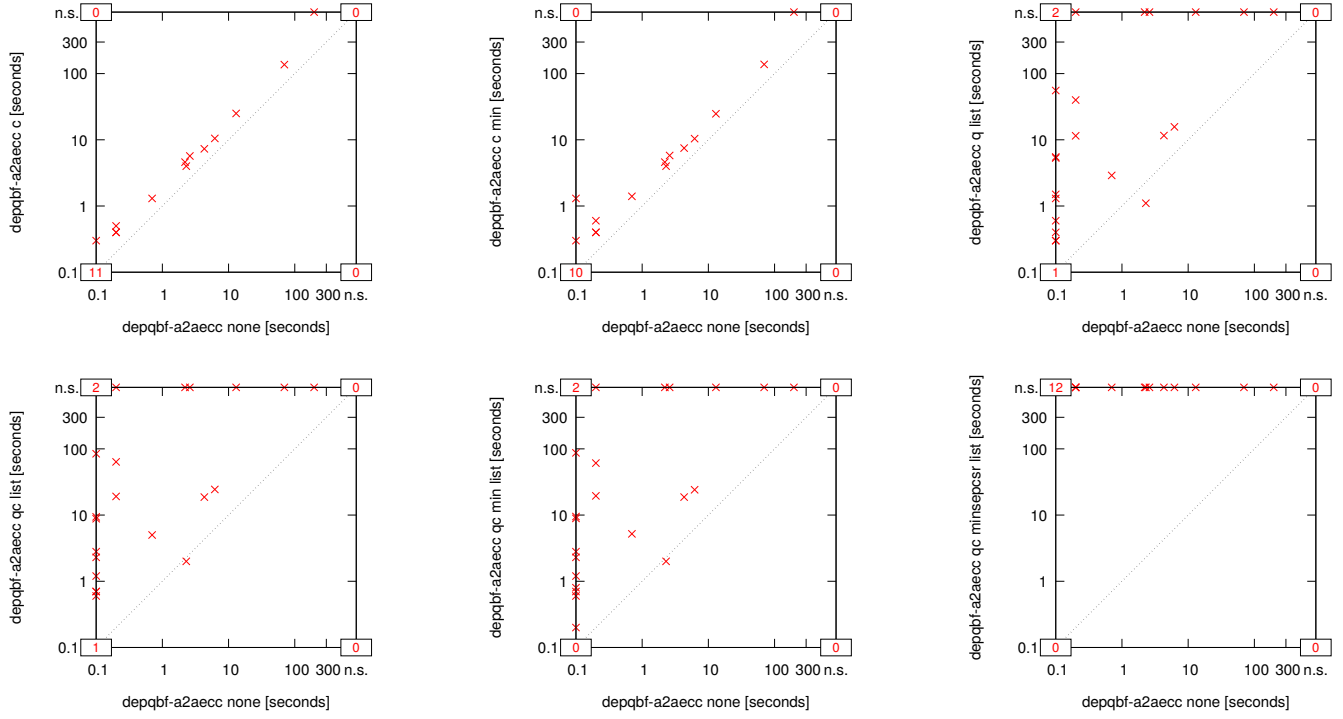


Fig. 727: Suite Interian ($n = 24$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

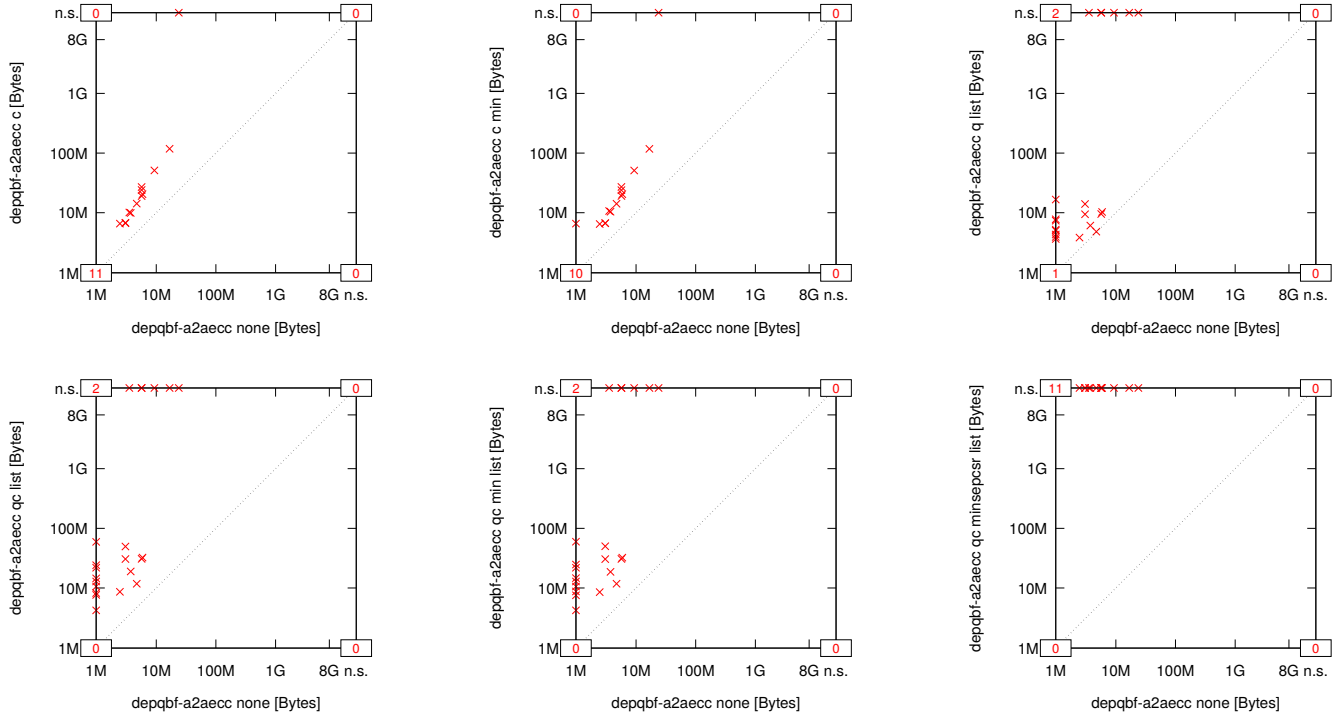


Fig. 728: Suite Interian ($n = 24$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

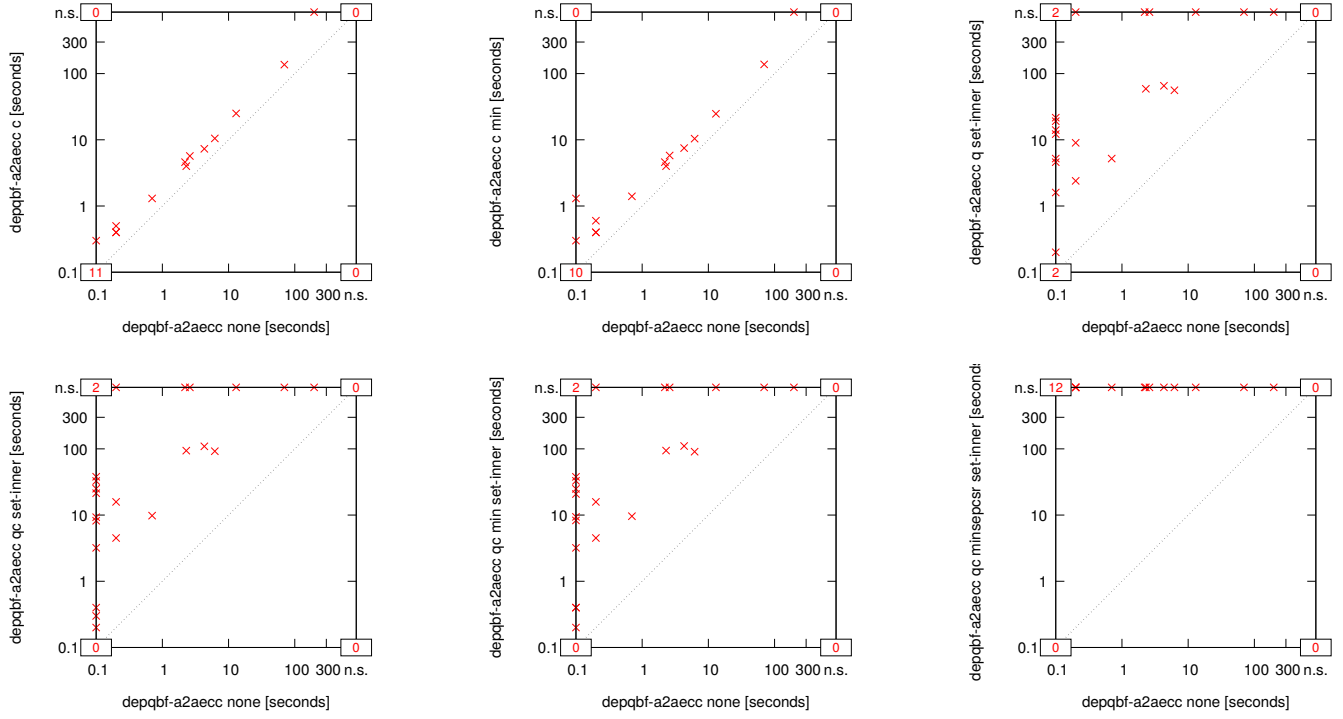


Fig. 729: Suite Interian ($n = 24$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

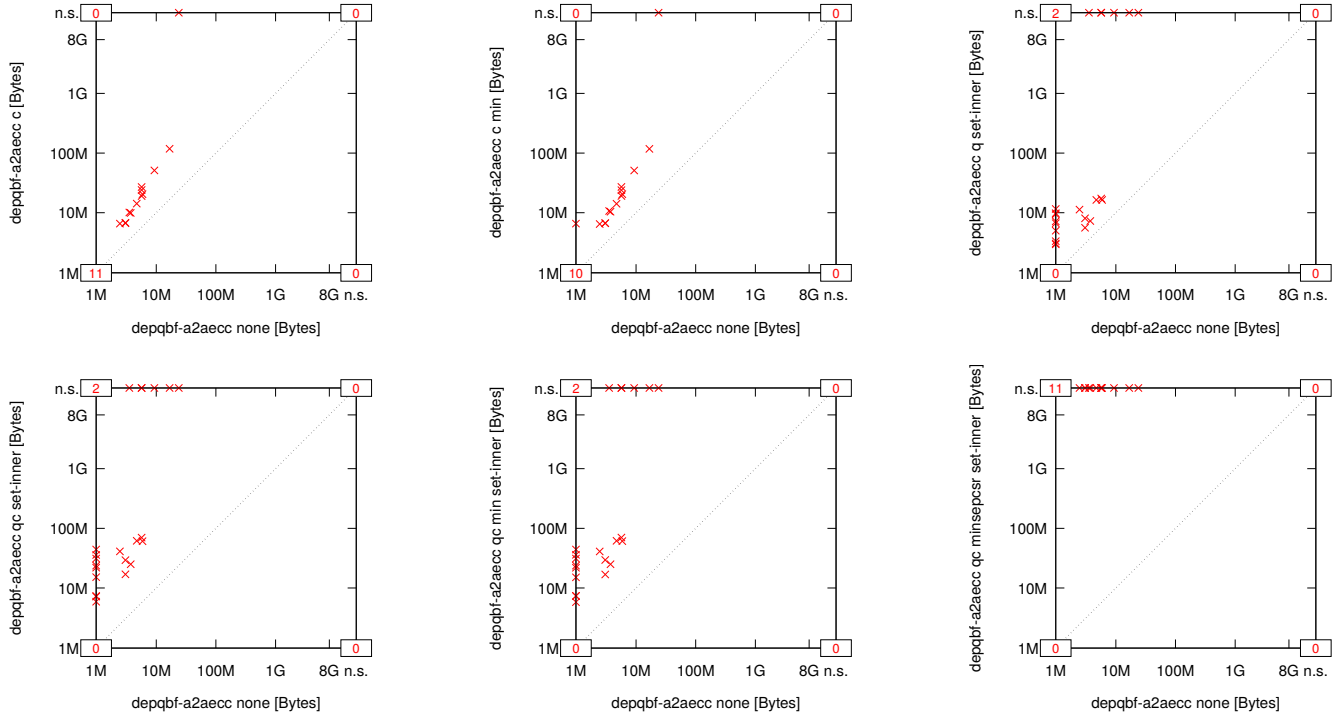


Fig. 730: Suite Interian ($n = 24$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

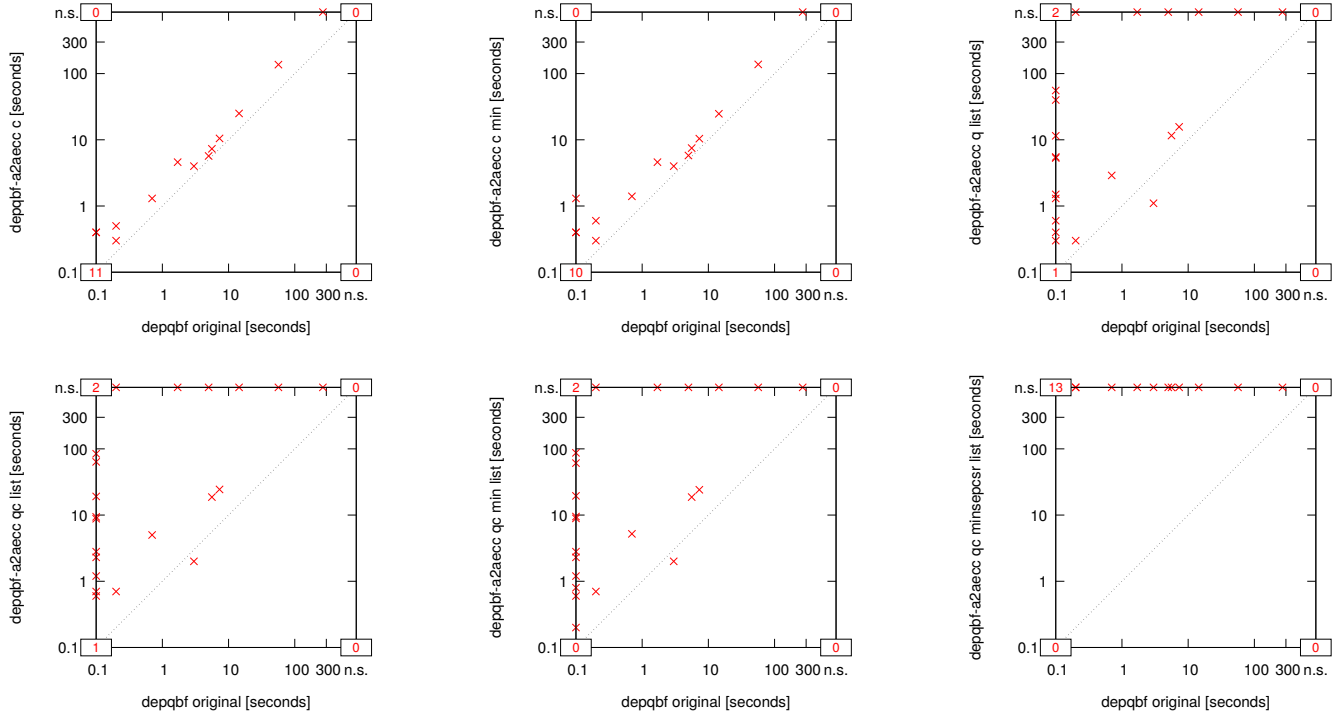


Fig. 731: Suite Interian ($n = 24$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

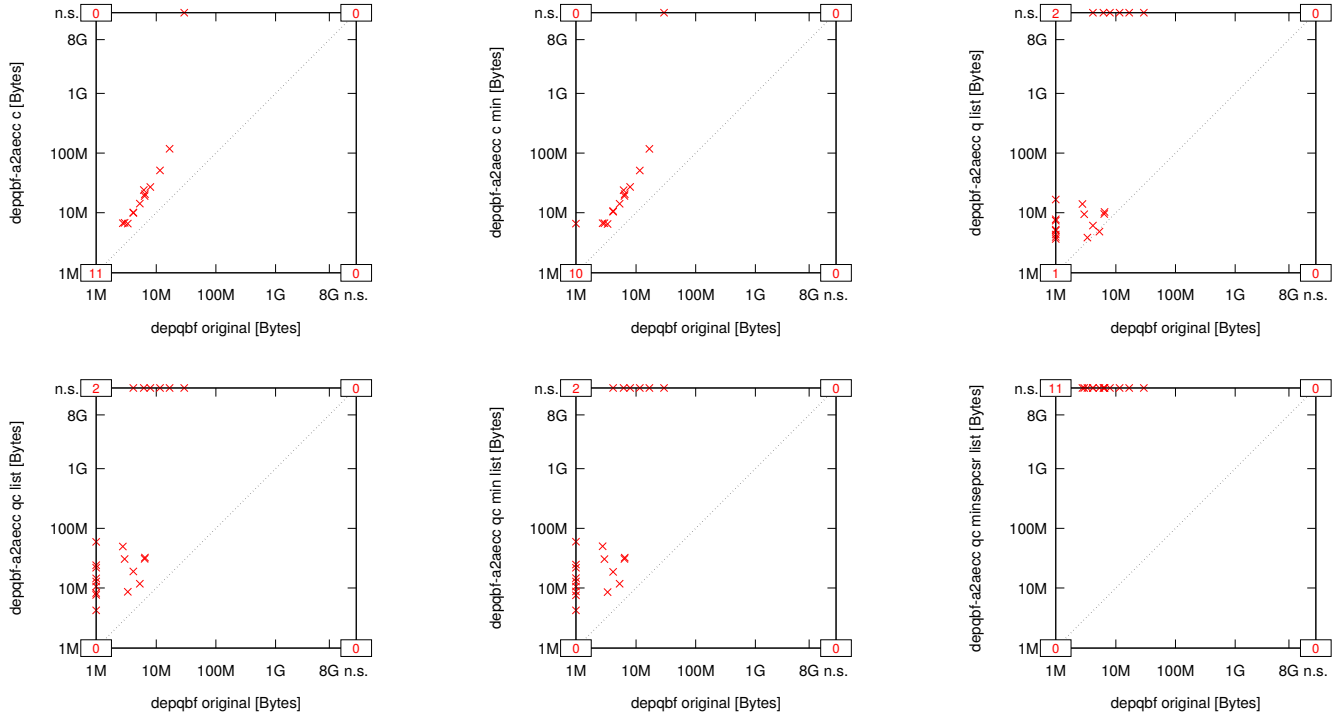


Fig. 732: Suite Interian ($n = 24$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

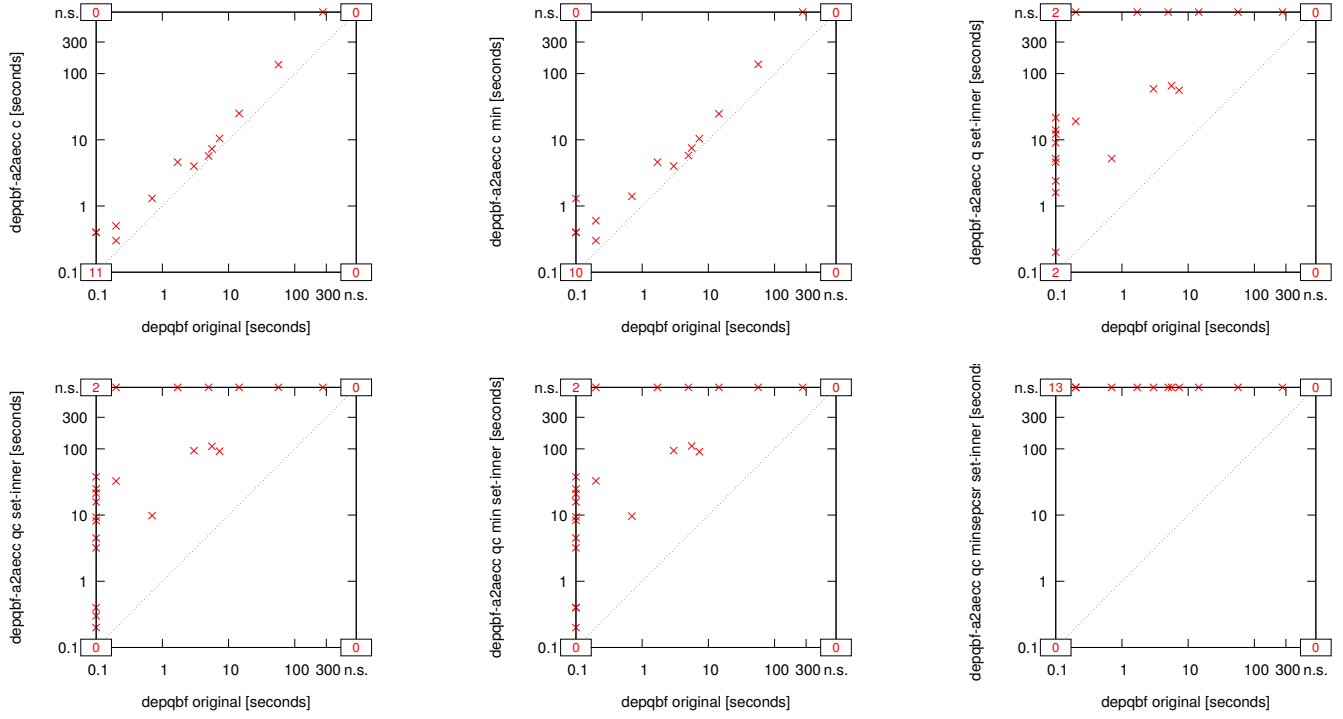


Fig. 733: Suite Interian ($n = 24$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

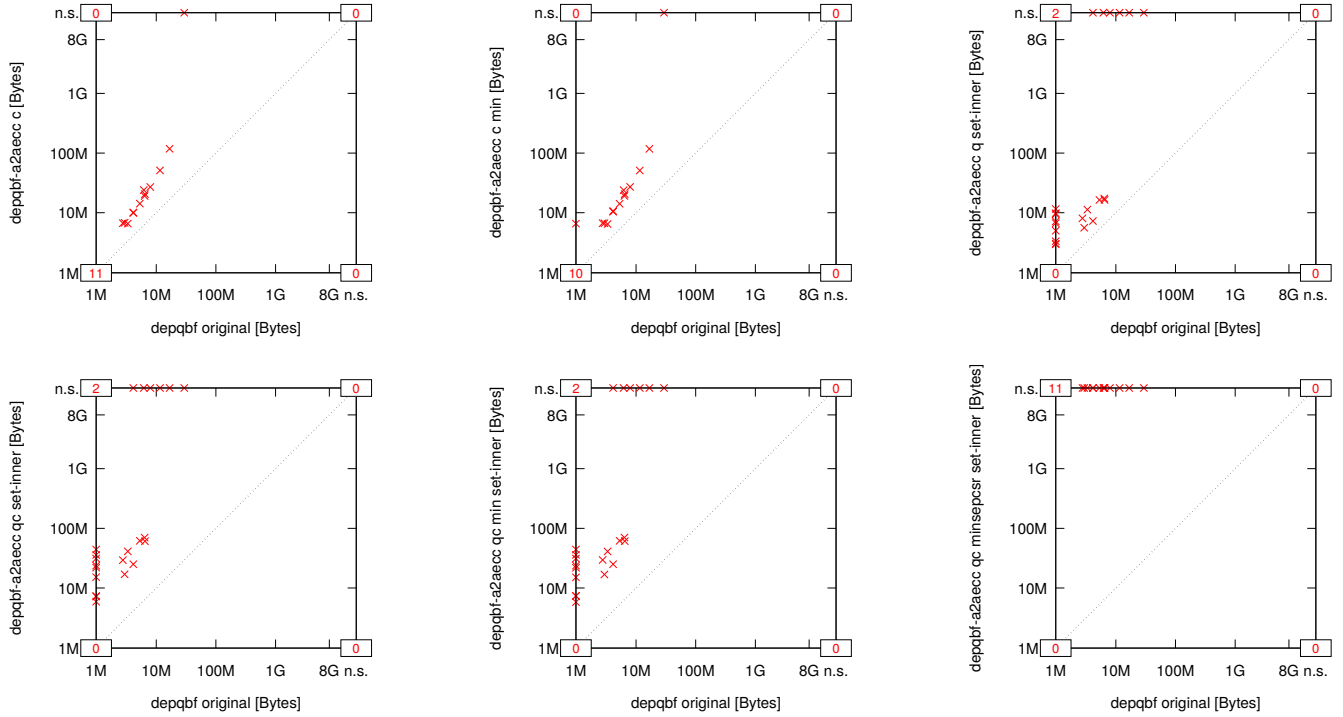


Fig. 734: Suite Interian ($n = 24$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

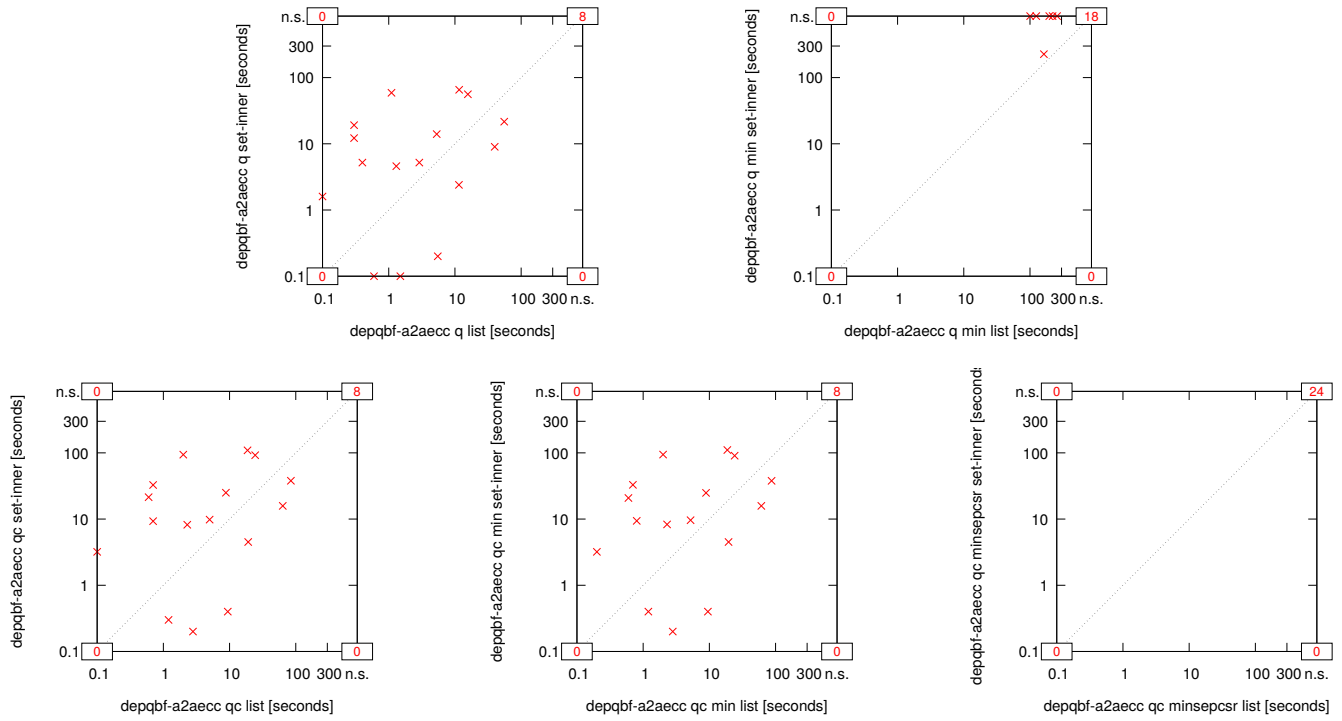


Fig. 735: Suite Interian ($n = 24$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

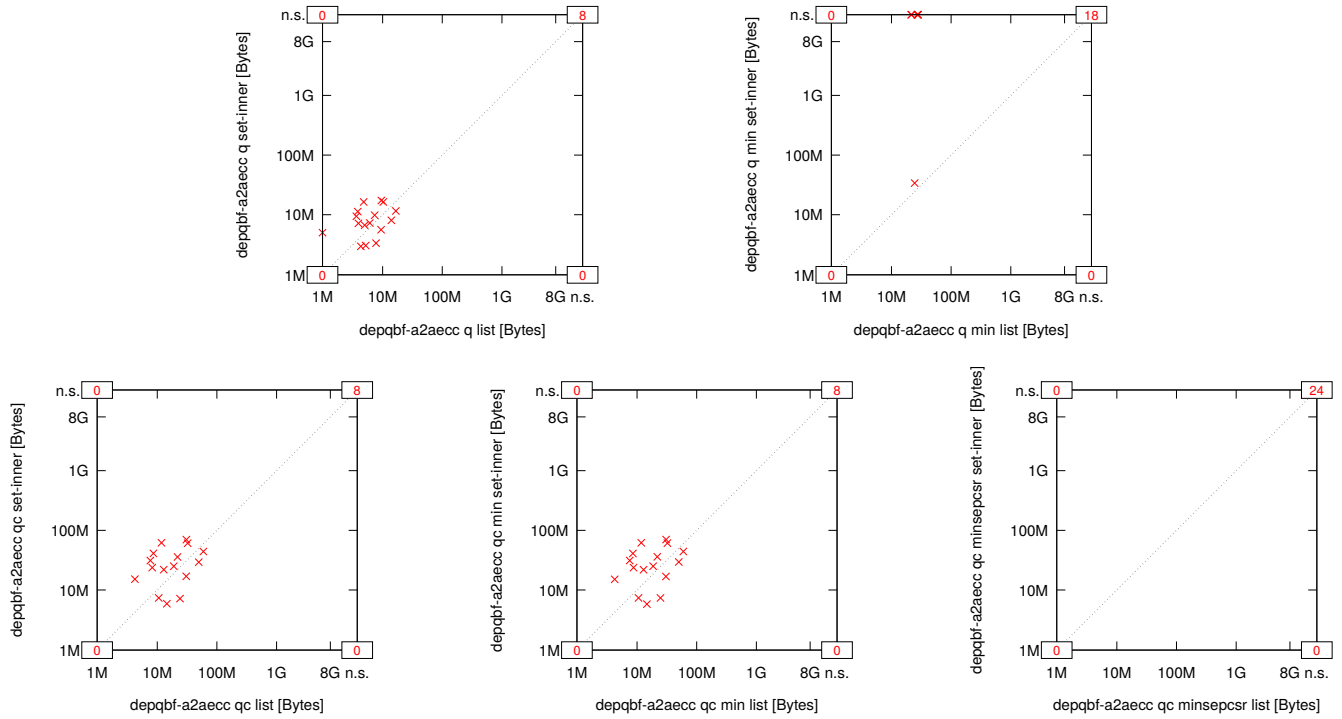


Fig. 736: Suite Interian ($n = 24$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

17) Jordan-Kaiser ($n = 83$):

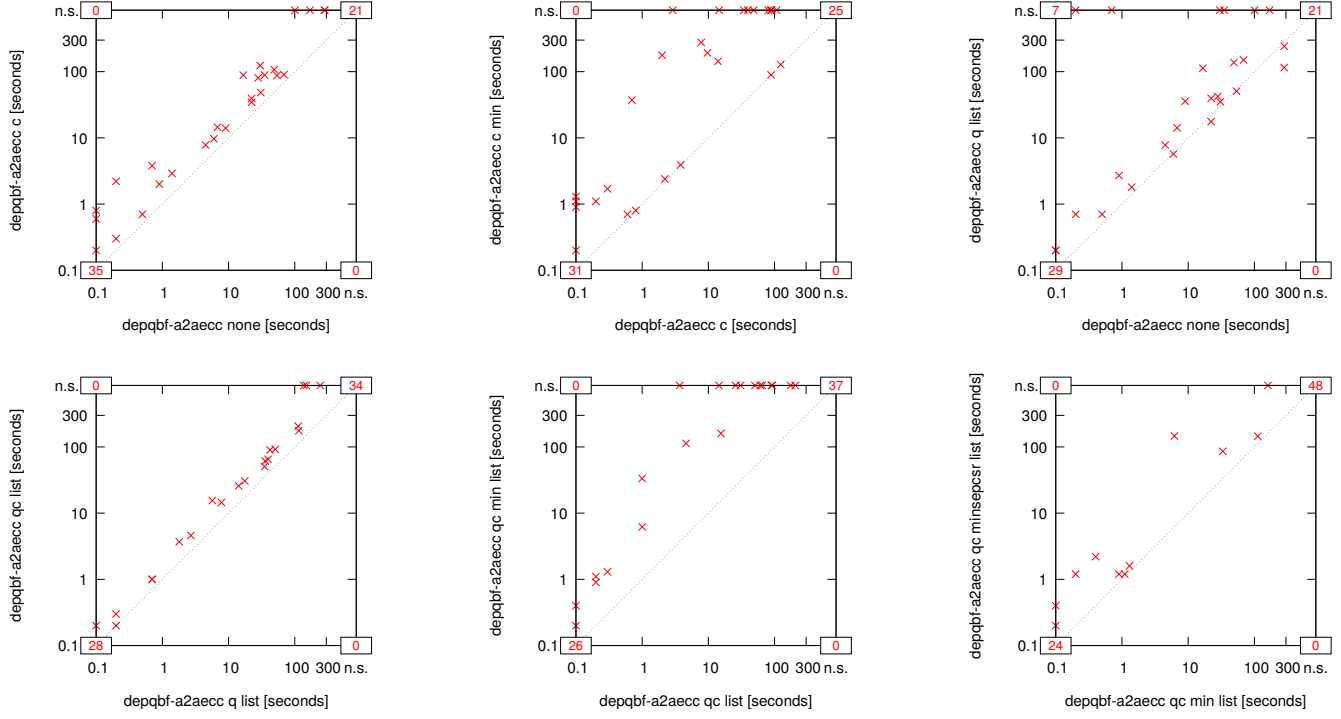


Fig. 737: Suite Jordan-Kaiser ($n = 83$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

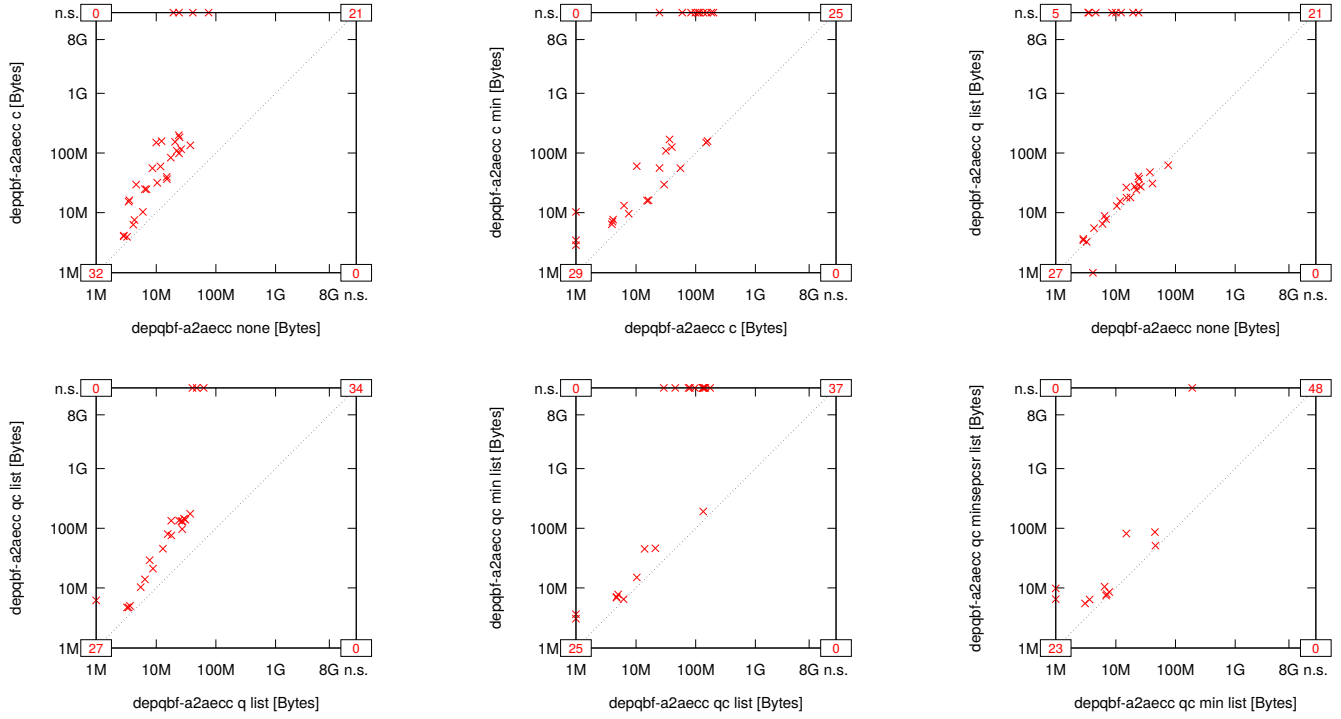


Fig. 738: Suite Jordan-Kaiser ($n = 83$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

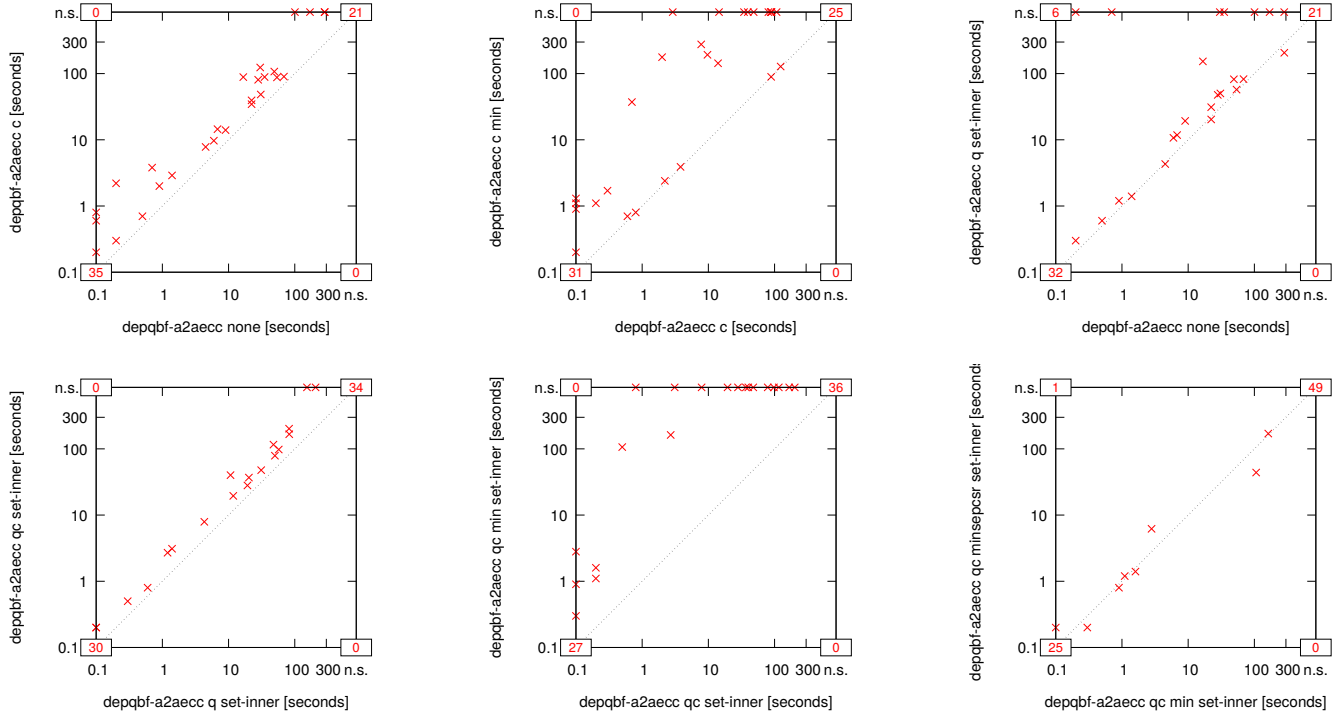


Fig. 739: Suite Jordan-Kaiser ($n = 83$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

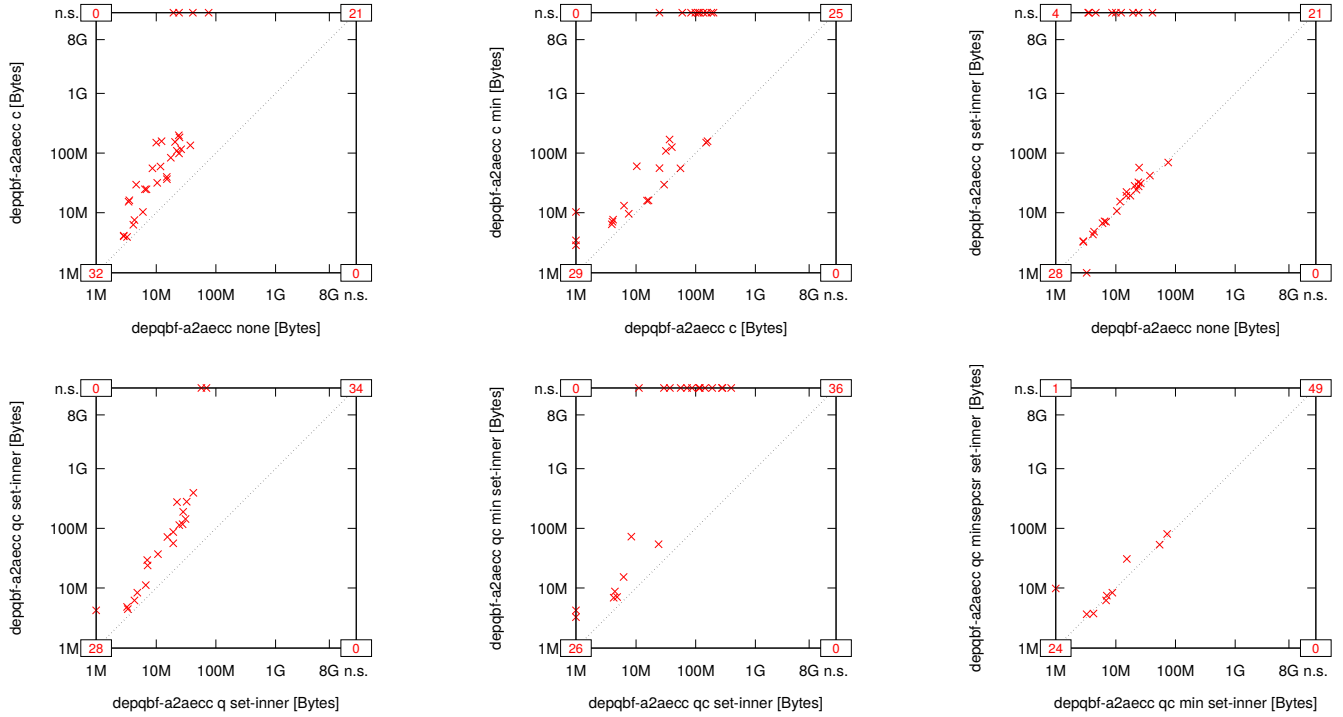


Fig. 740: Suite Jordan-Kaiser ($n = 83$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

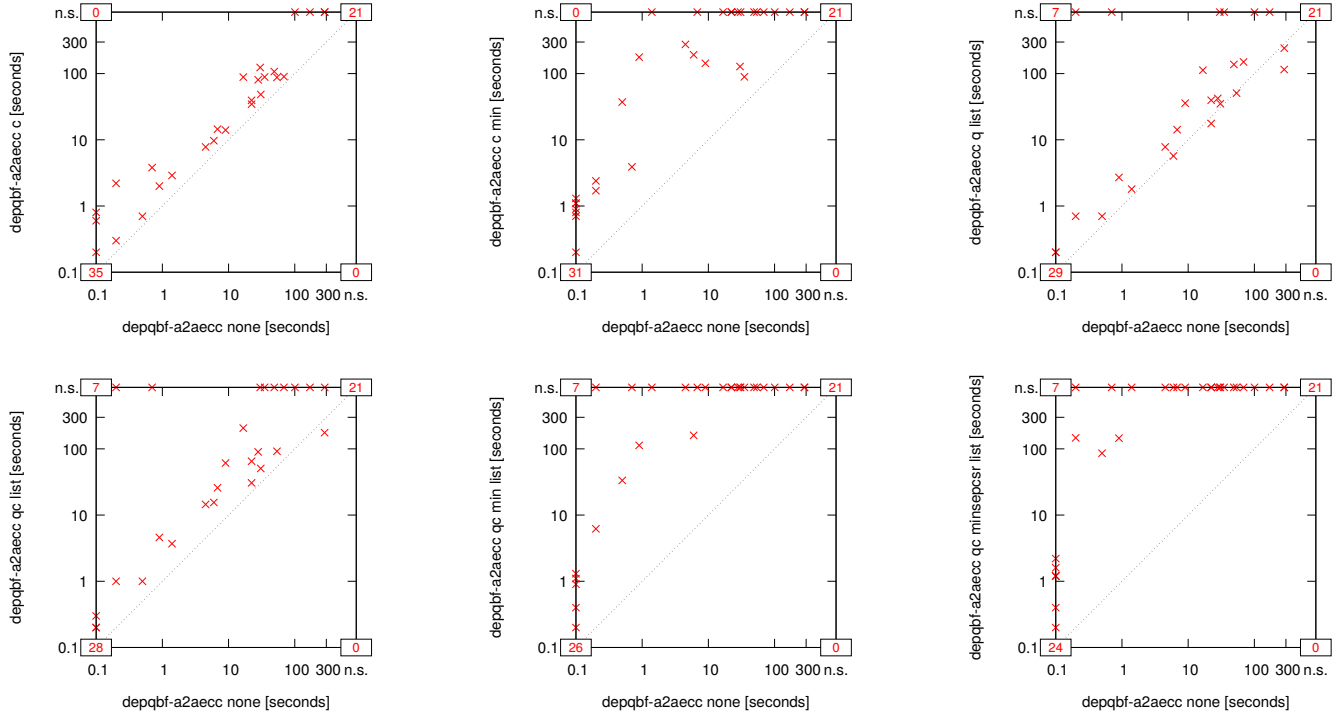


Fig. 741: Suite Jordan-Kaiser ($n = 83$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

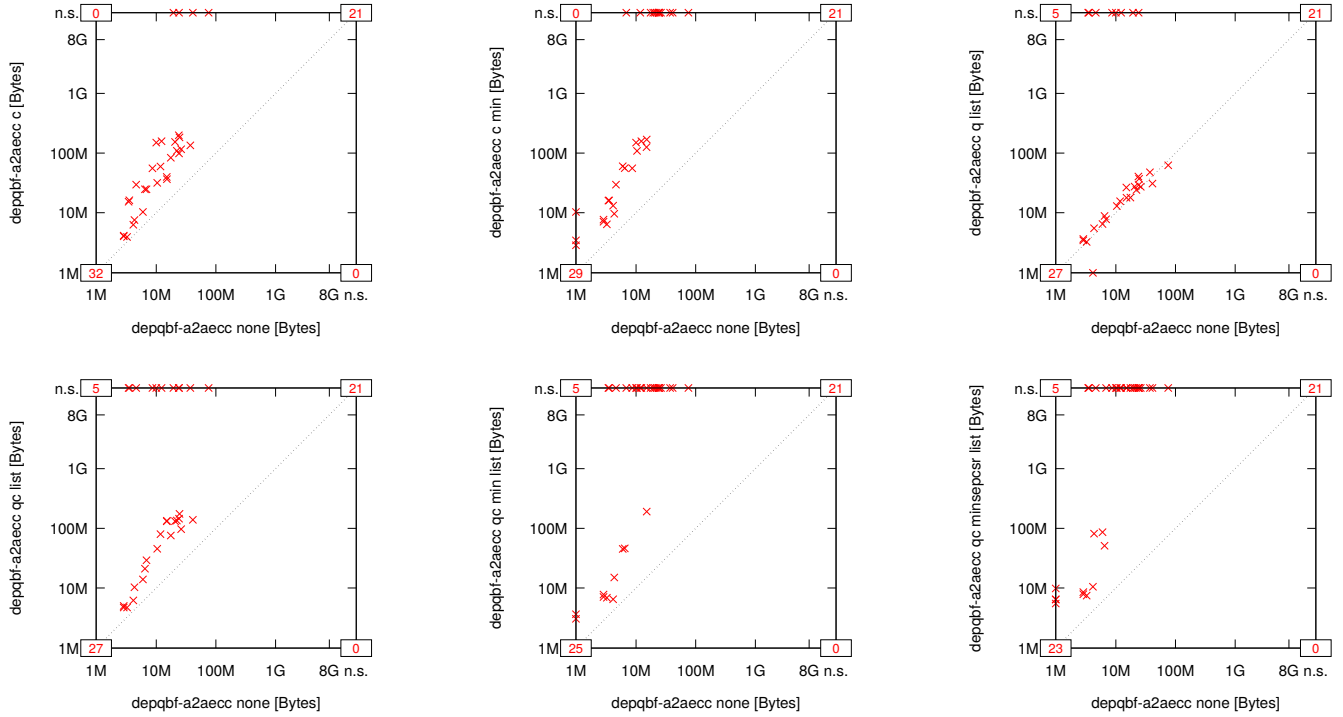


Fig. 742: Suite Jordan-Kaiser ($n = 83$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

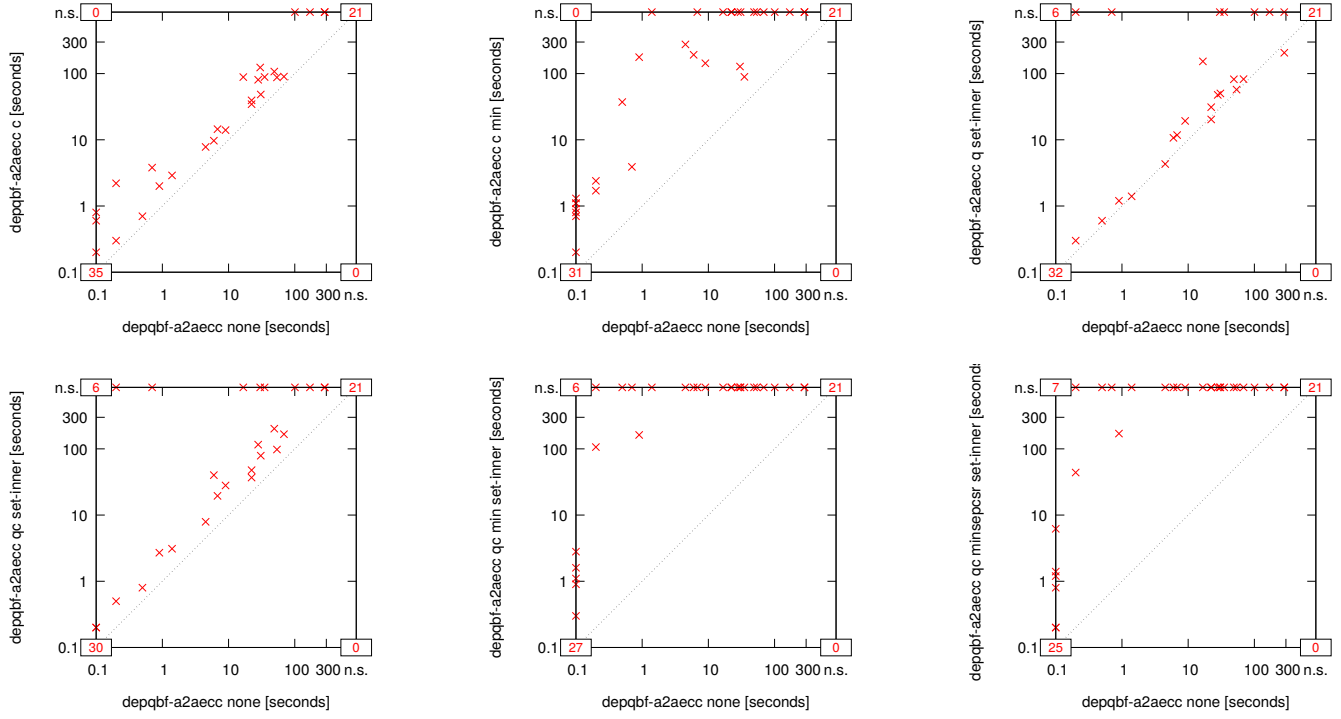


Fig. 743: Suite Jordan-Kaiser ($n = 83$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

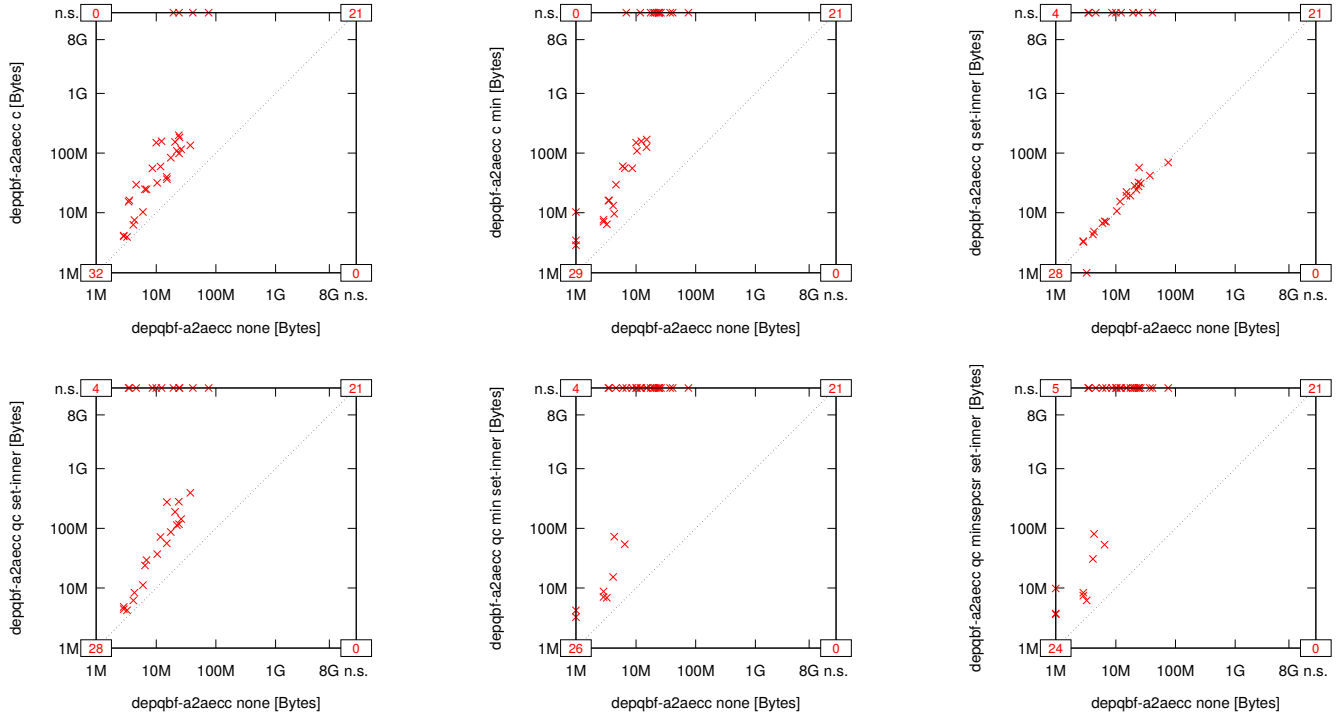


Fig. 744: Suite Jordan-Kaiser ($n = 83$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

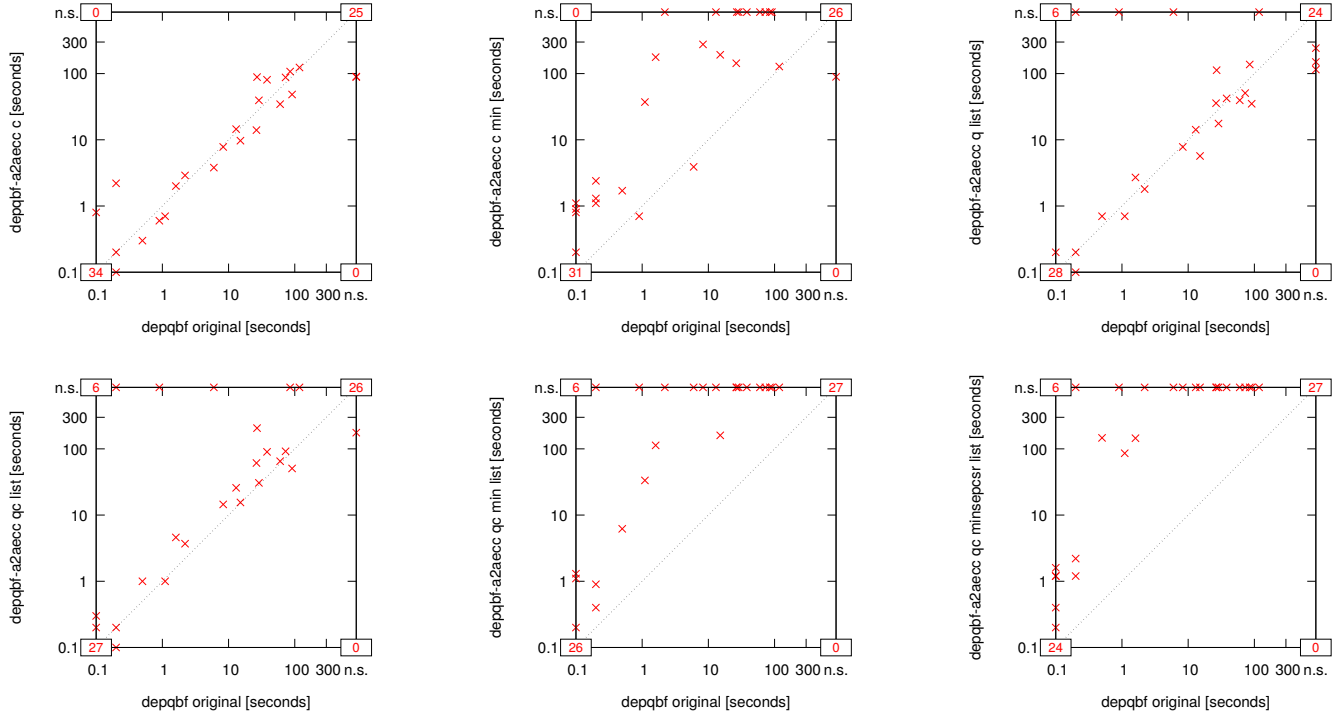


Fig. 745: Suite Jordan-Kaiser ($n = 83$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

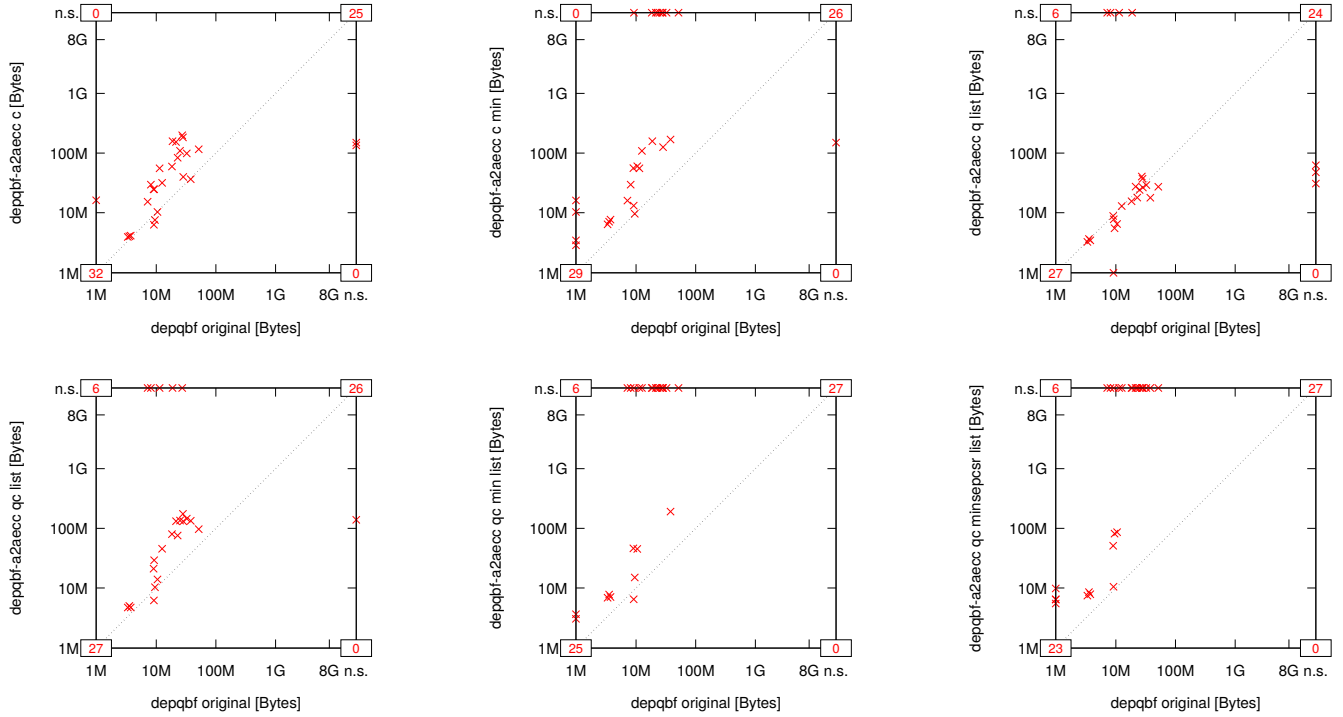


Fig. 746: Suite Jordan-Kaiser ($n = 83$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

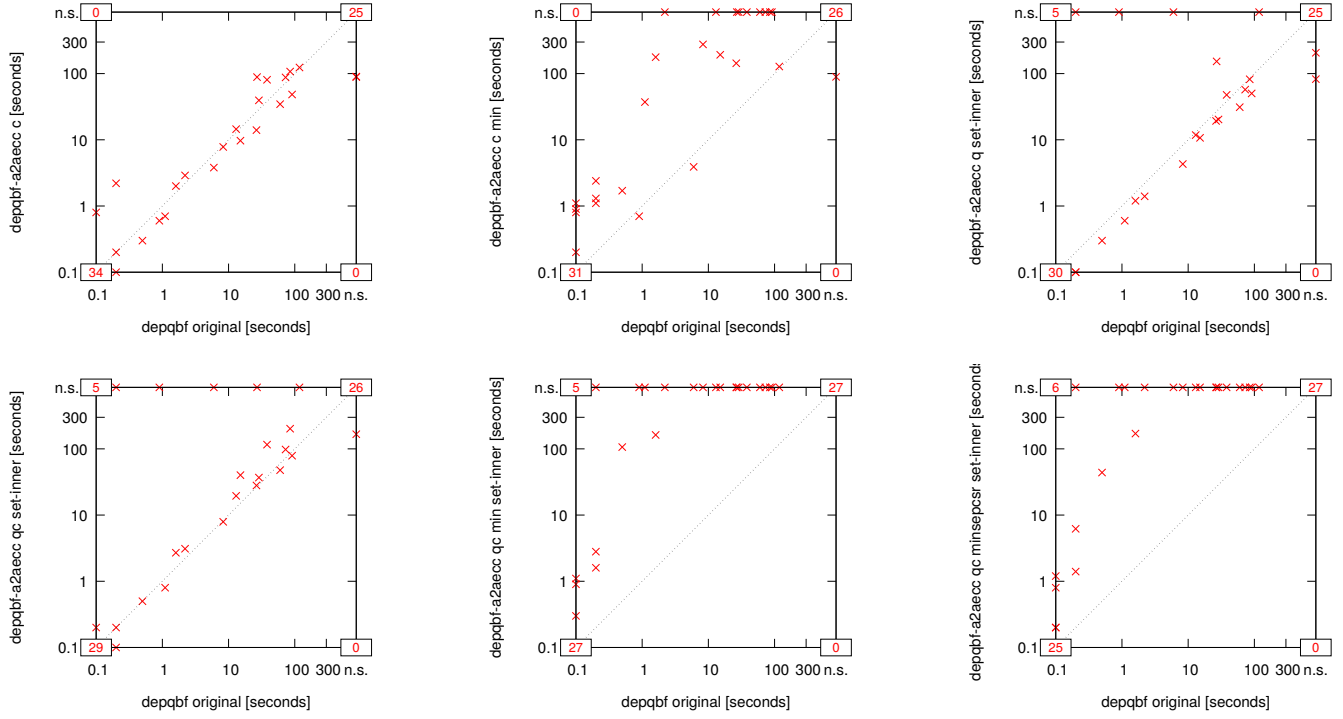


Fig. 747: Suite Jordan-Kaiser ($n = 83$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

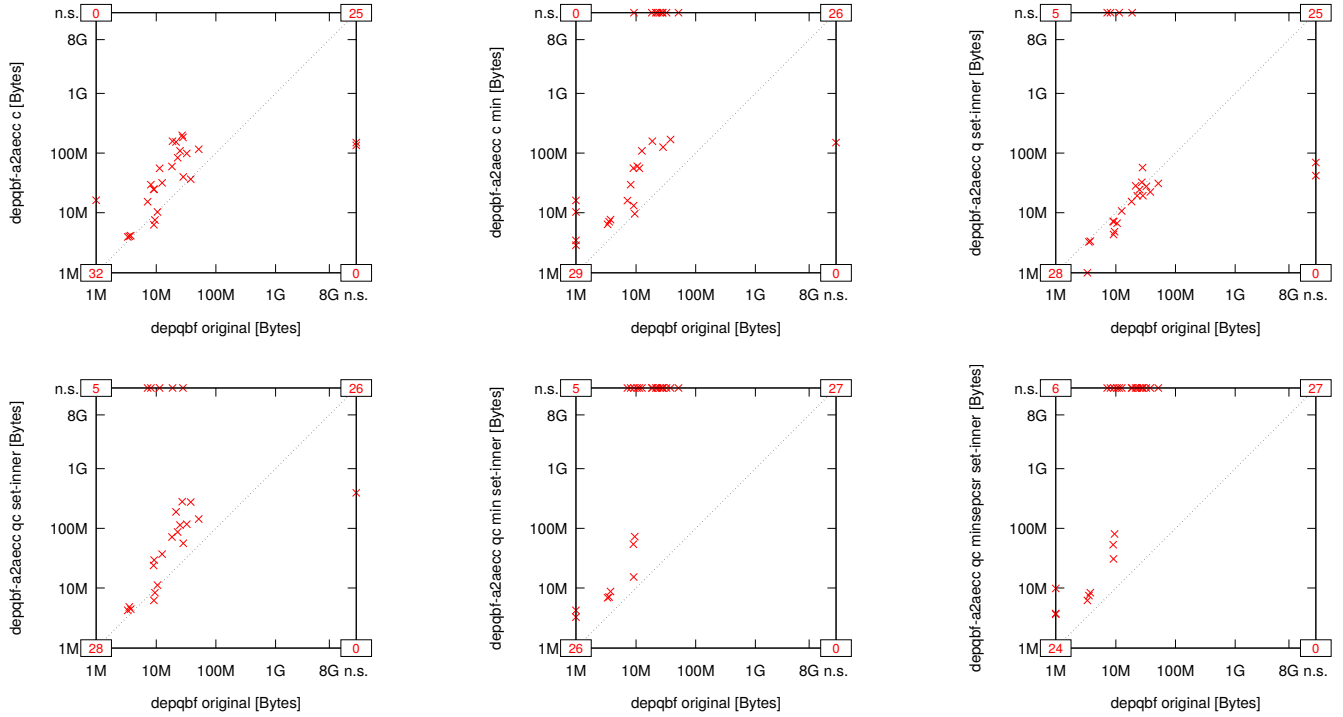


Fig. 748: Suite Jordan-Kaiser ($n = 83$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

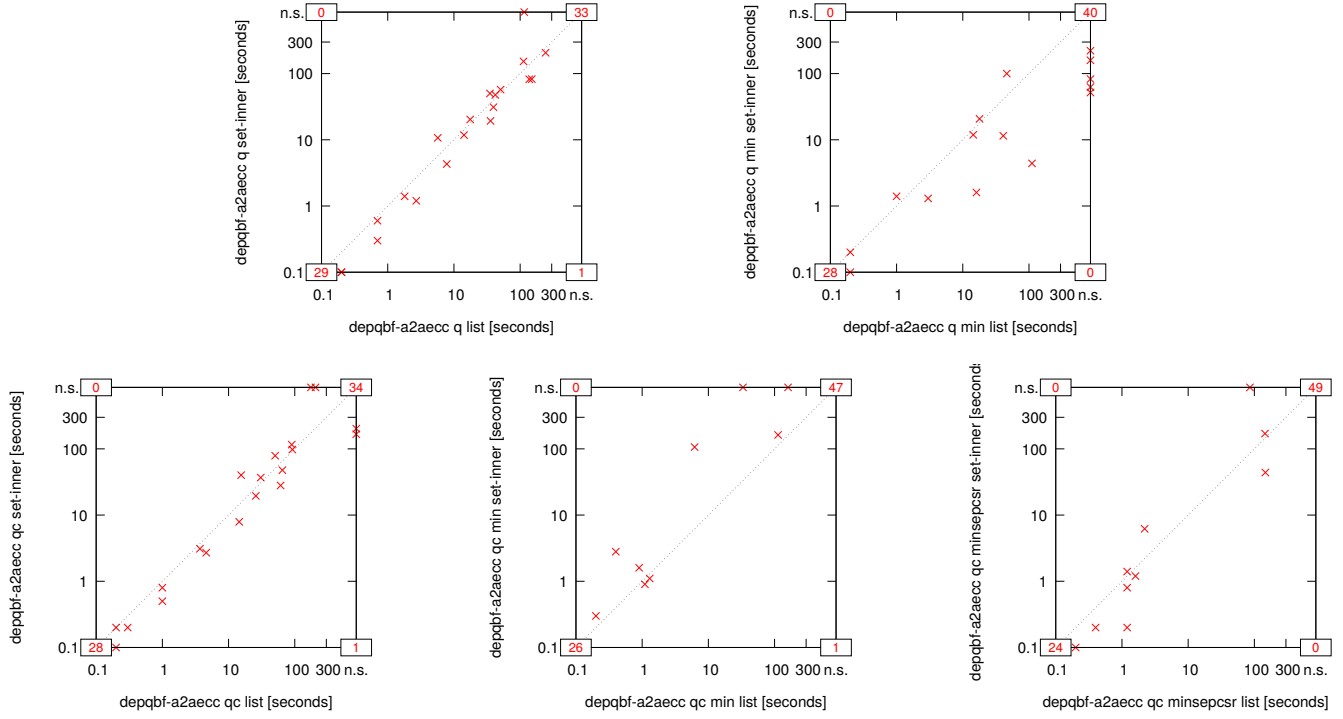


Fig. 749: Suite Jordan-Kaiser ($n = 83$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

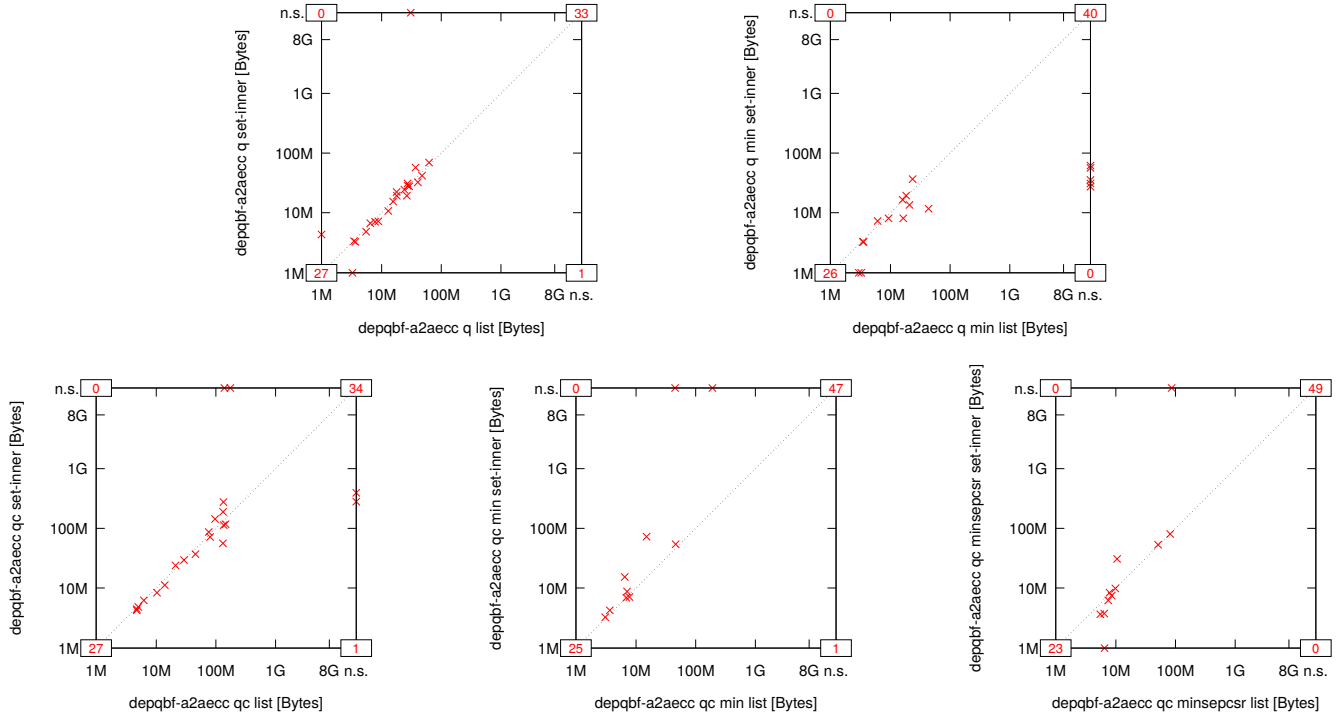


Fig. 750: Suite Jordan-Kaiser ($n = 83$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

18) Katz ($n = 8$):

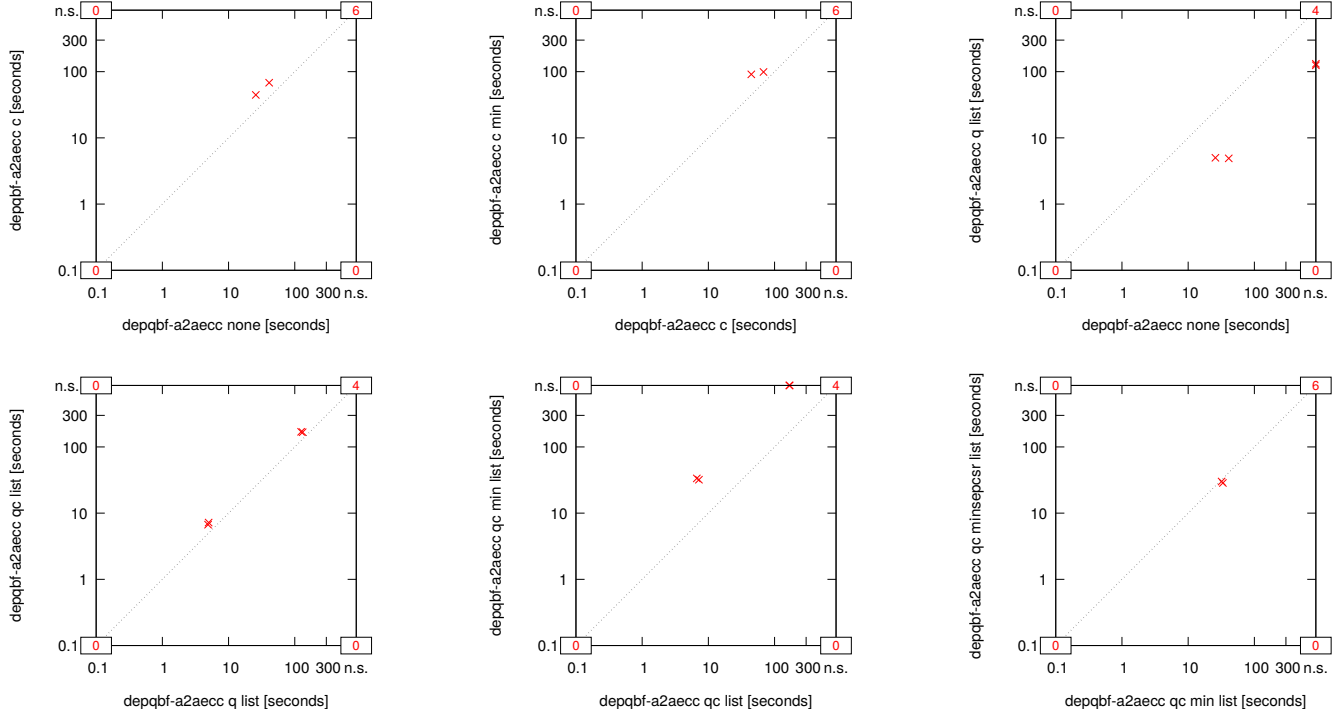


Fig. 751: Suite Katz ($n = 8$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

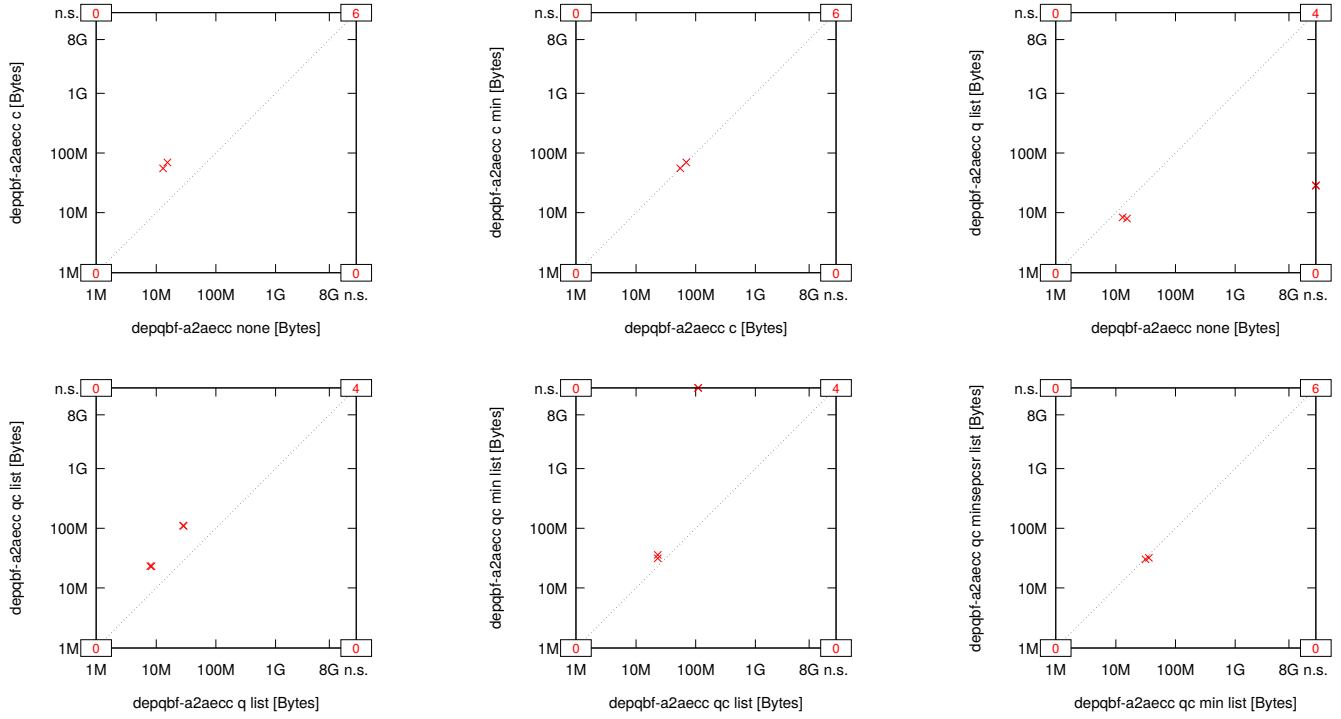


Fig. 752: Suite Katz ($n = 8$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

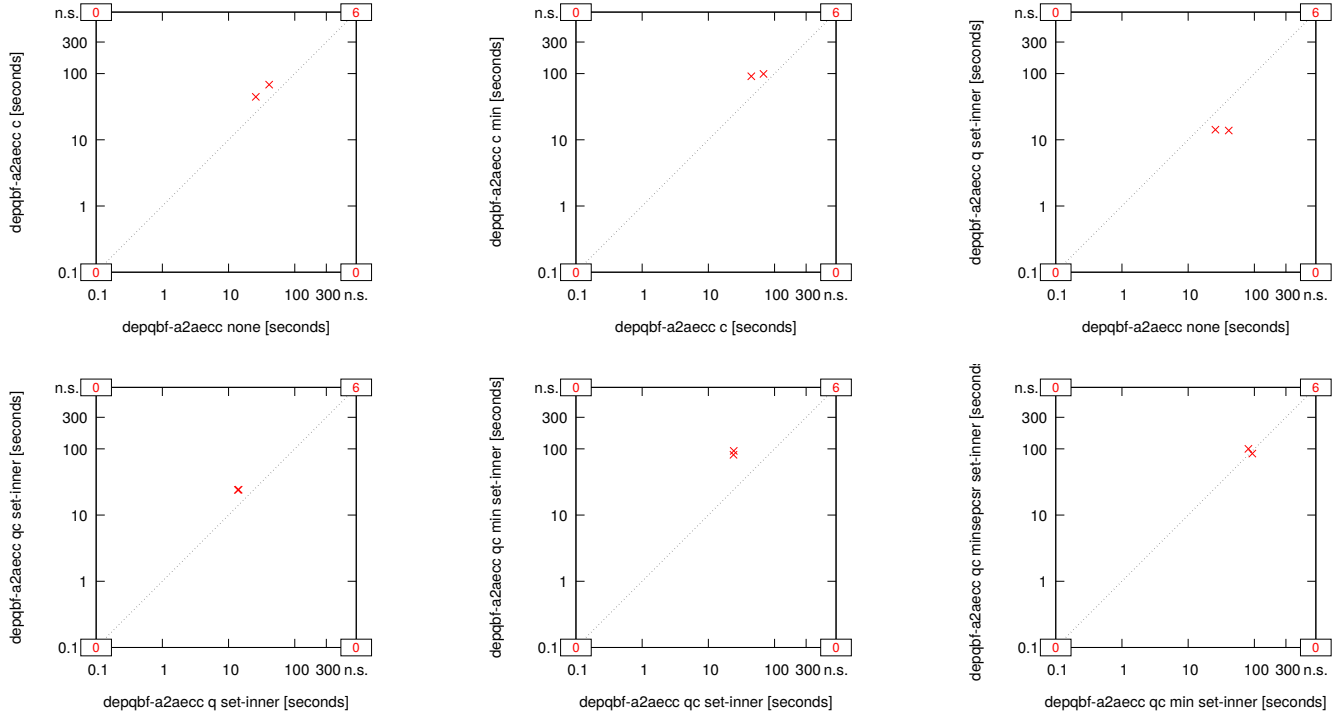


Fig. 753: Suite Katz ($n = 8$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

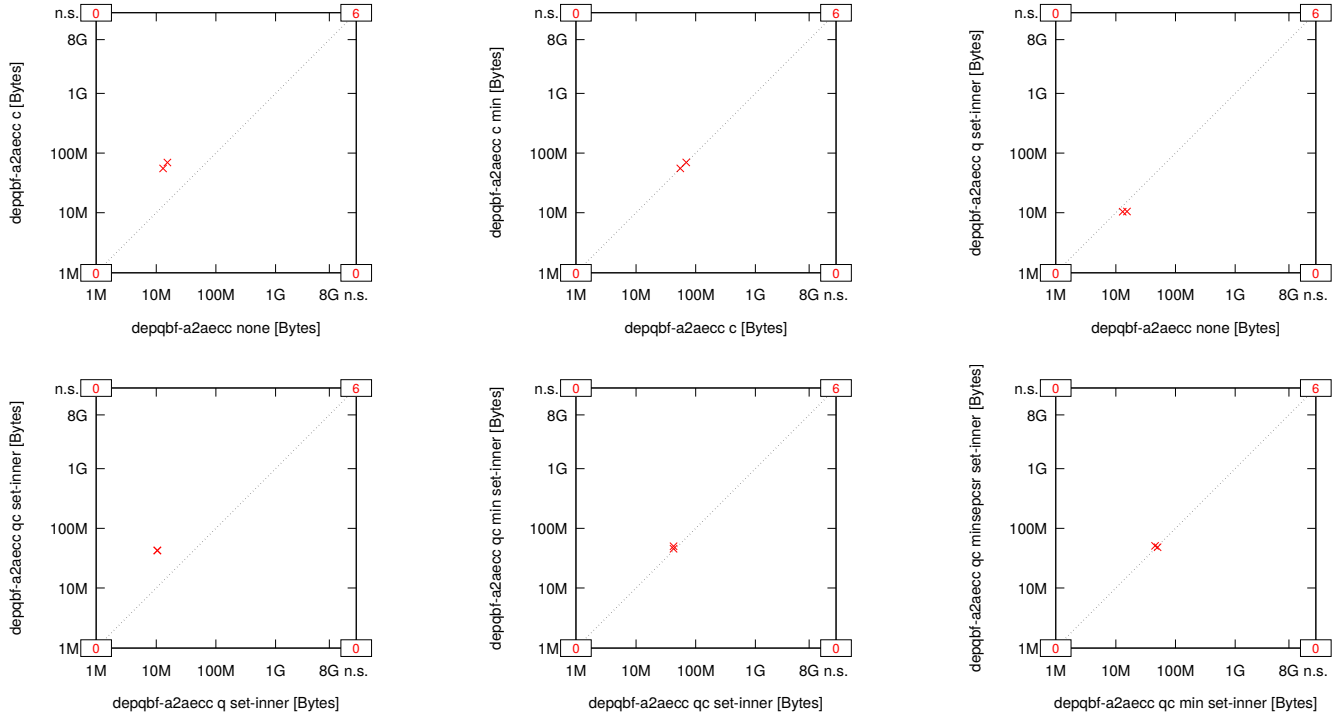


Fig. 754: Suite Katz ($n = 8$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

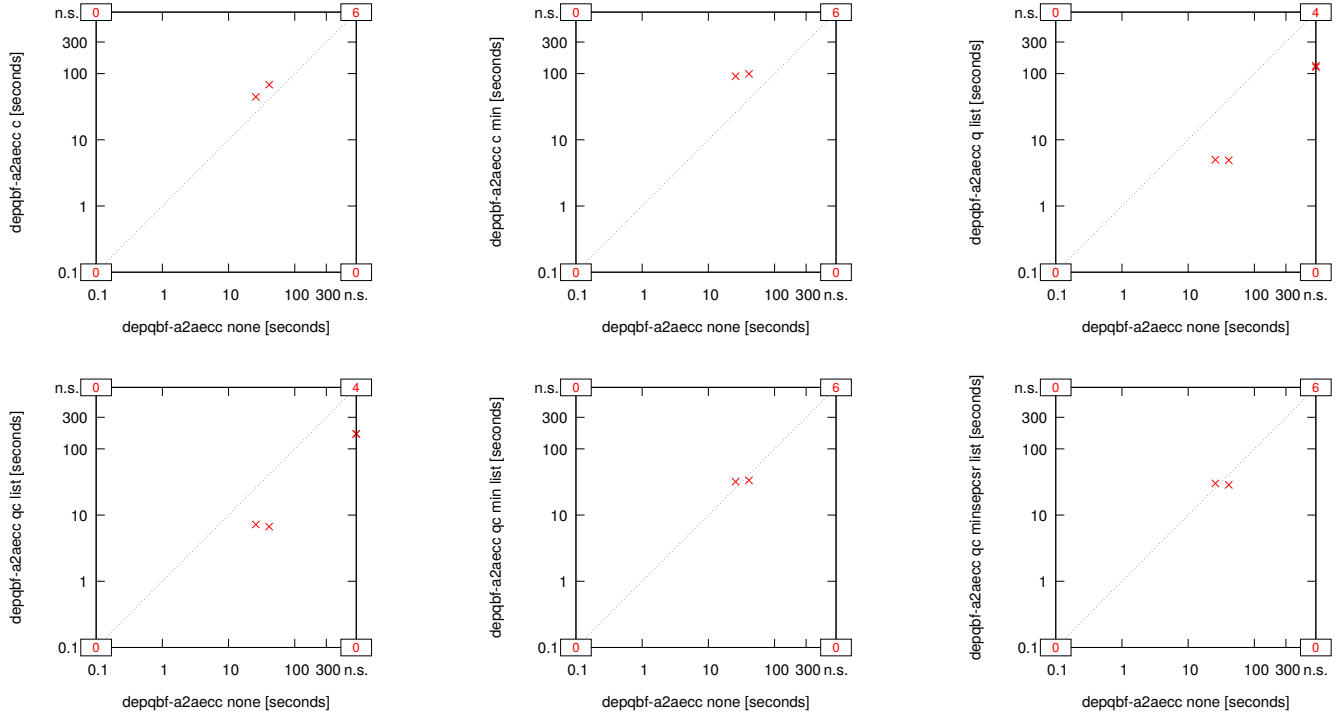


Fig. 755: Suite Katz ($n = 8$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

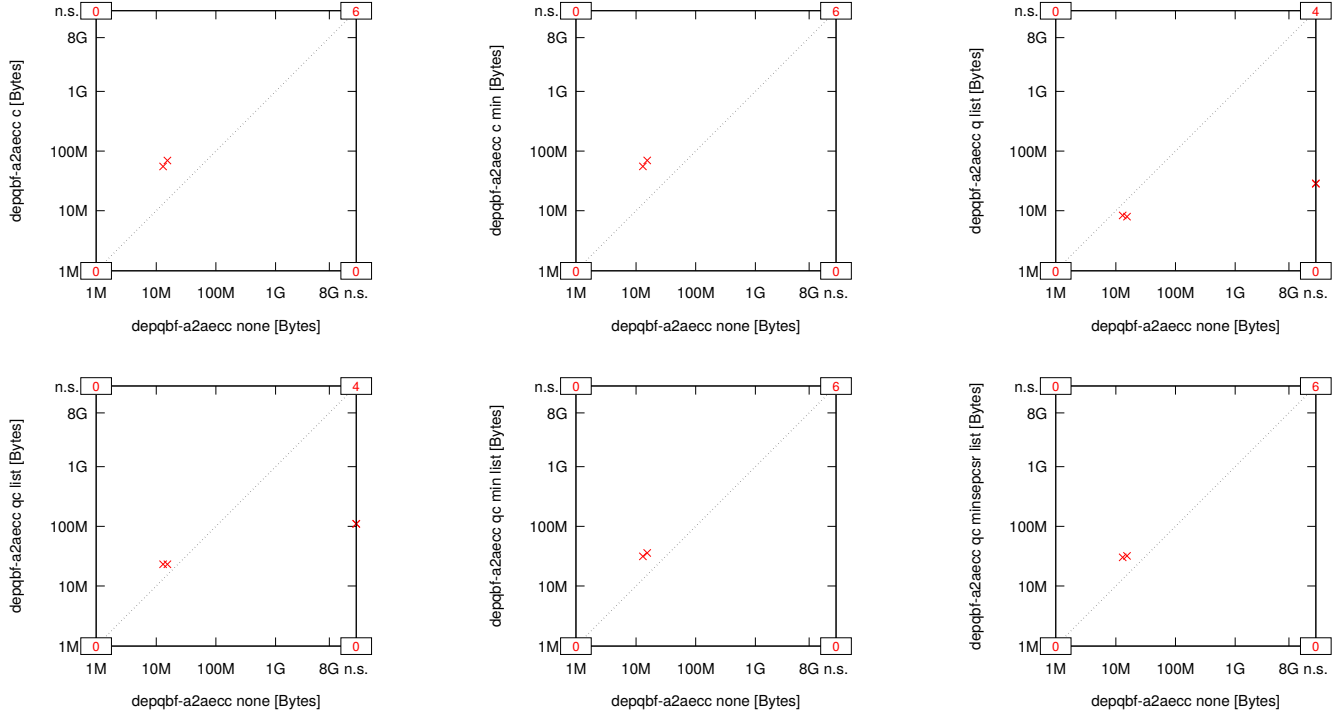


Fig. 756: Suite Katz ($n = 8$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

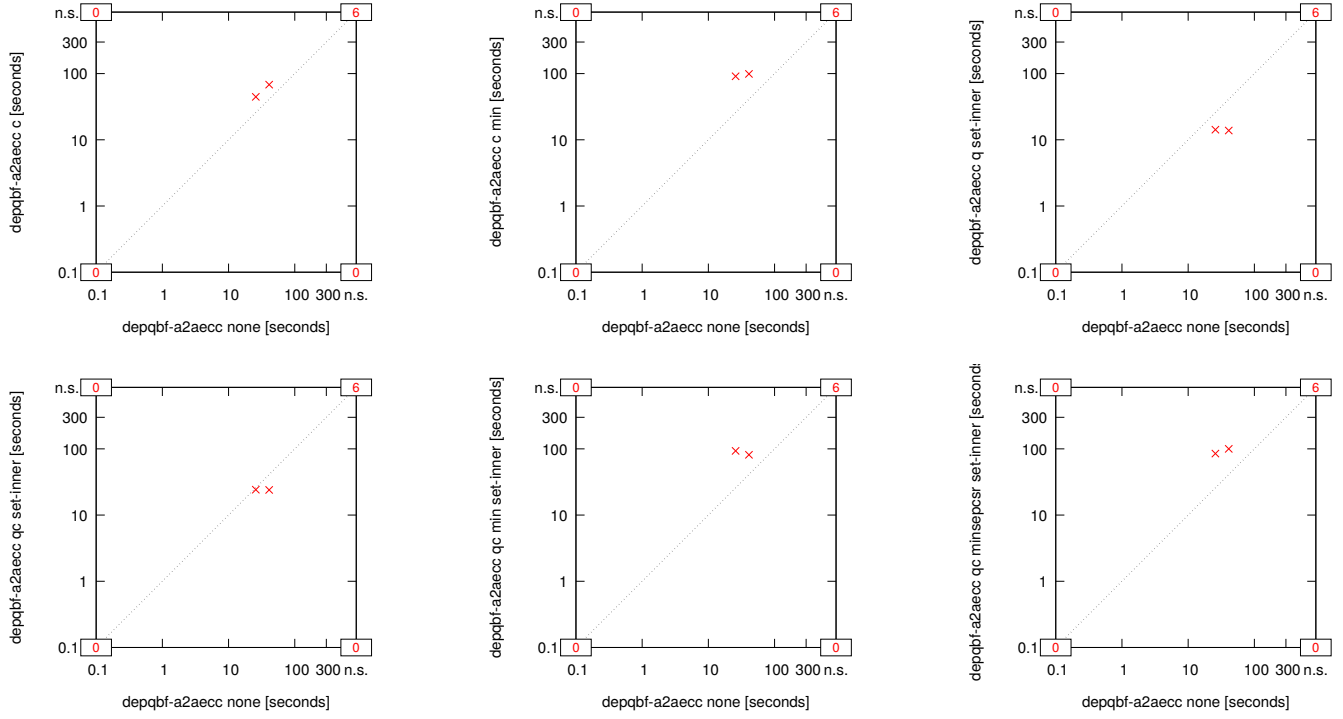


Fig. 757: Suite Katz ($n = 8$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

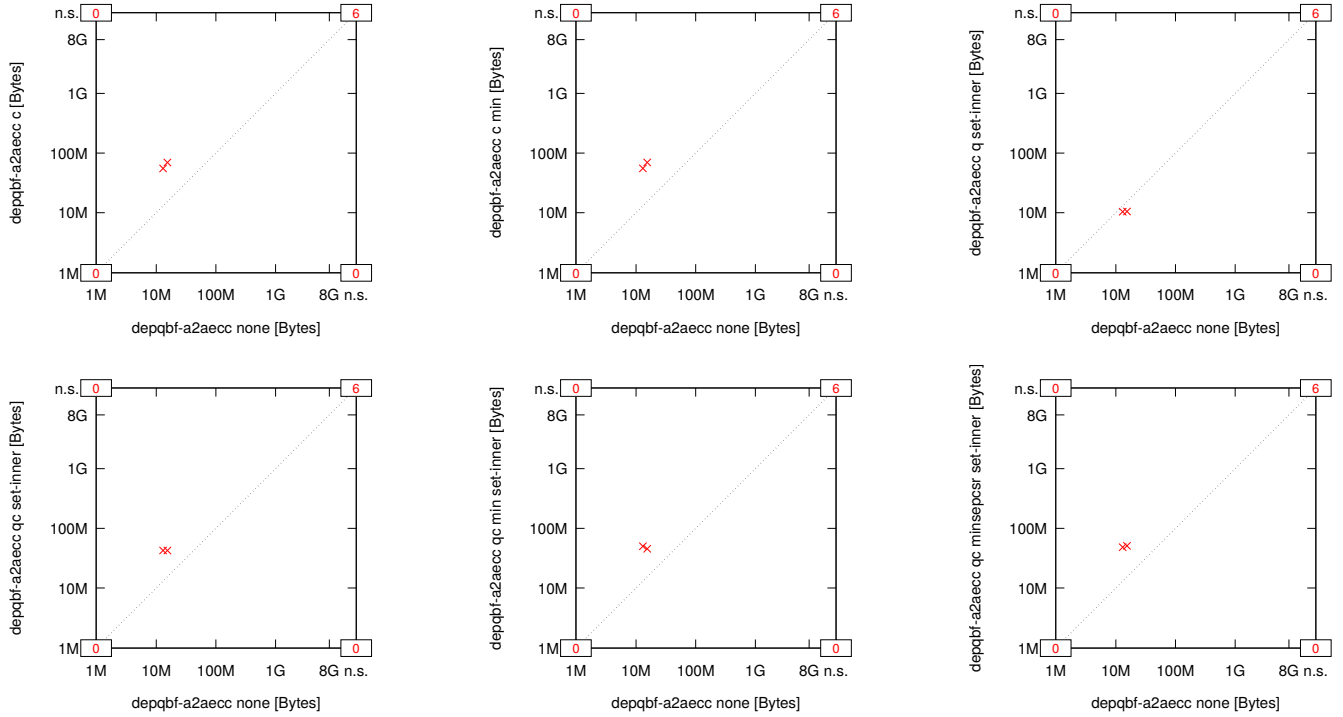


Fig. 758: Suite Katz ($n = 8$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

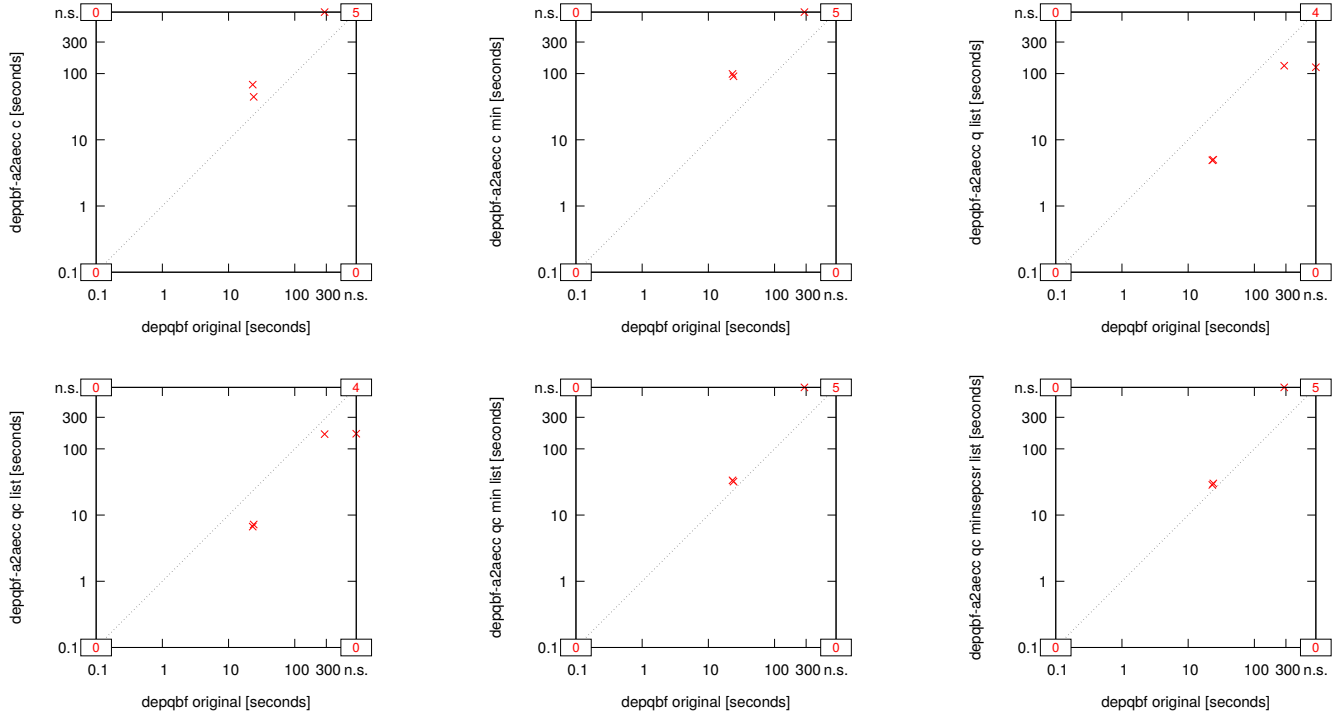


Fig. 759: Suite Katz ($n = 8$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

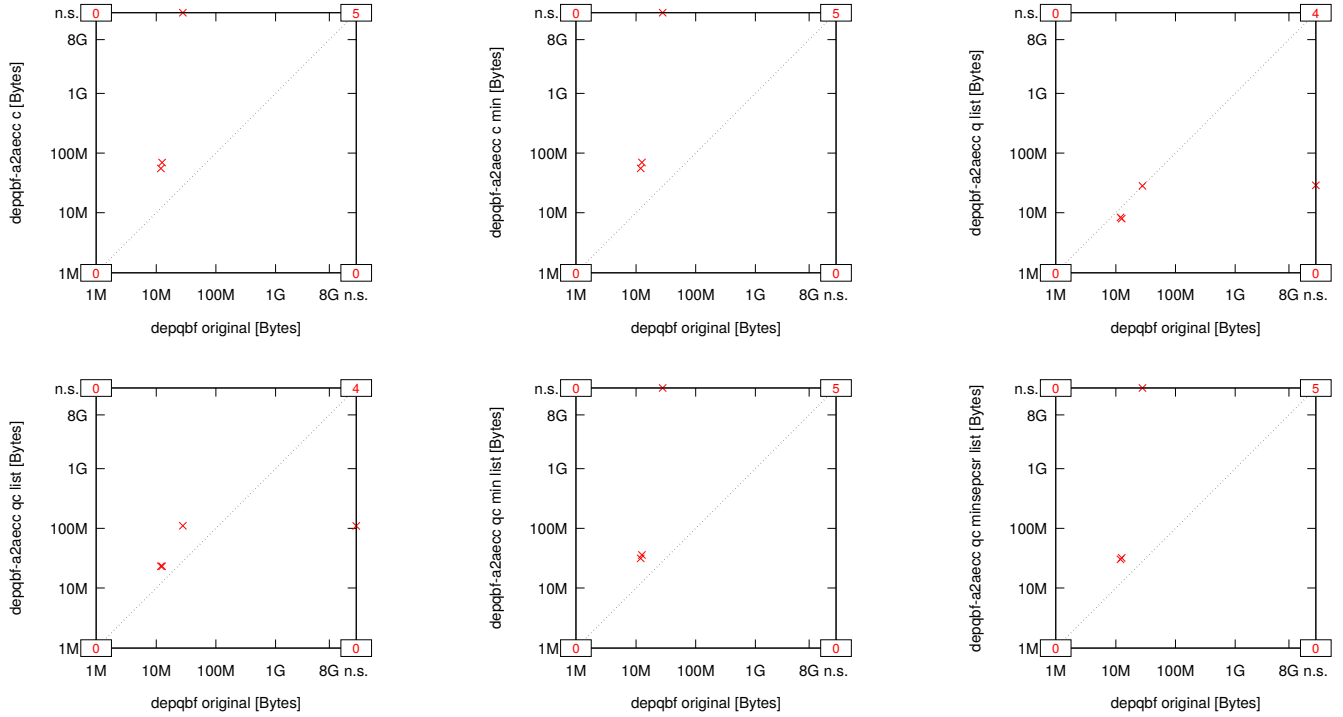


Fig. 760: Suite Katz ($n = 8$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

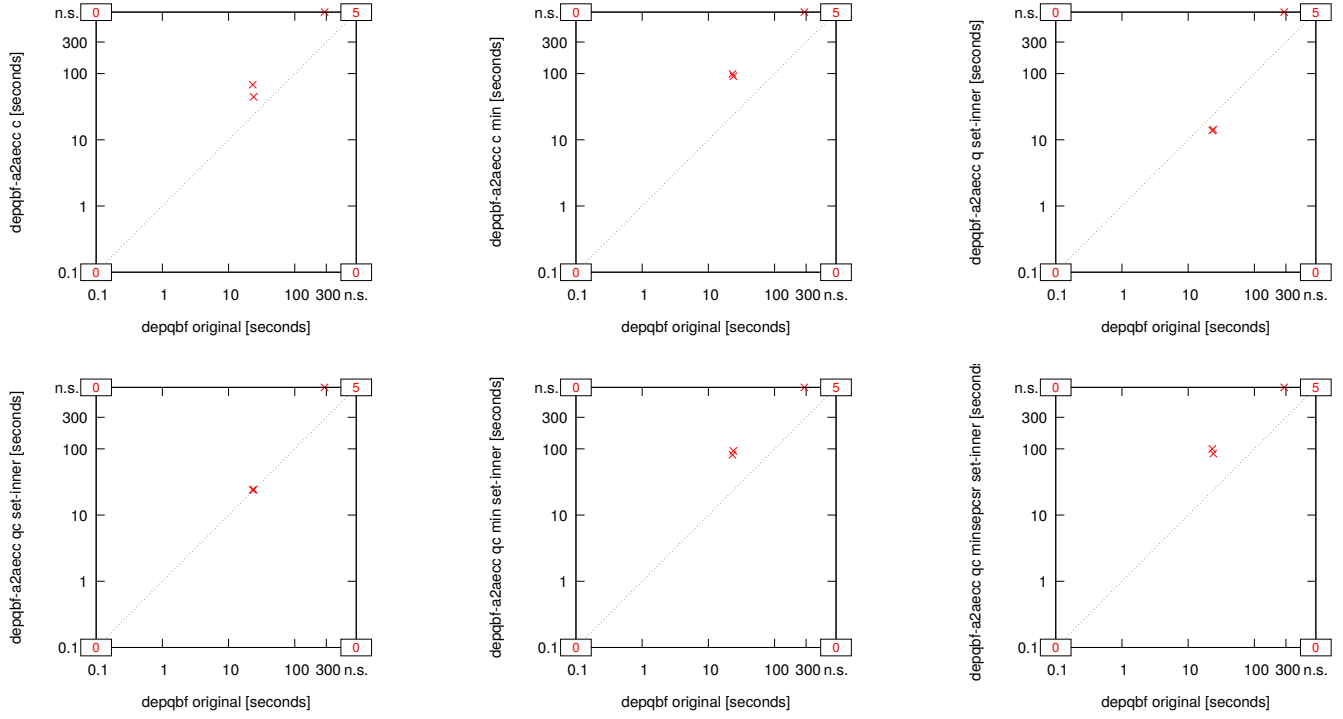


Fig. 761: Suite Katz ($n = 8$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

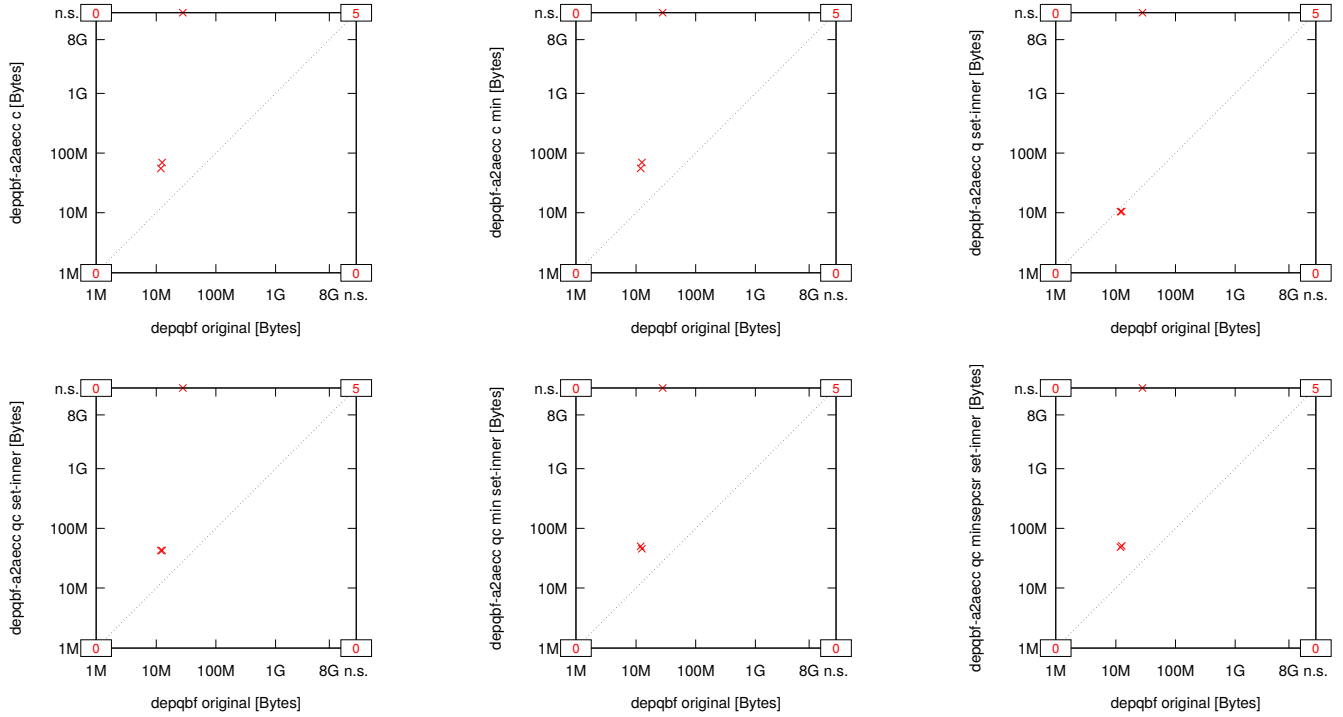


Fig. 762: Suite Katz ($n = 8$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

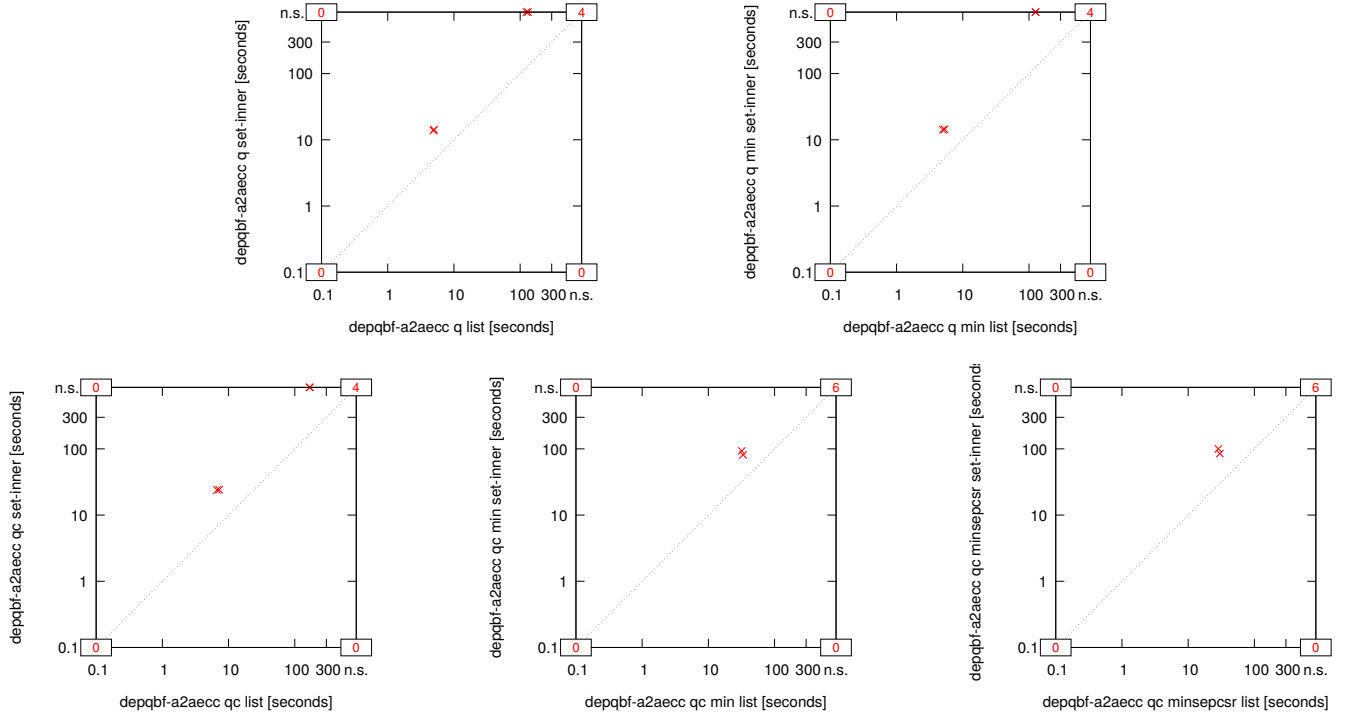


Fig. 763: Suite Katz ($n = 8$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

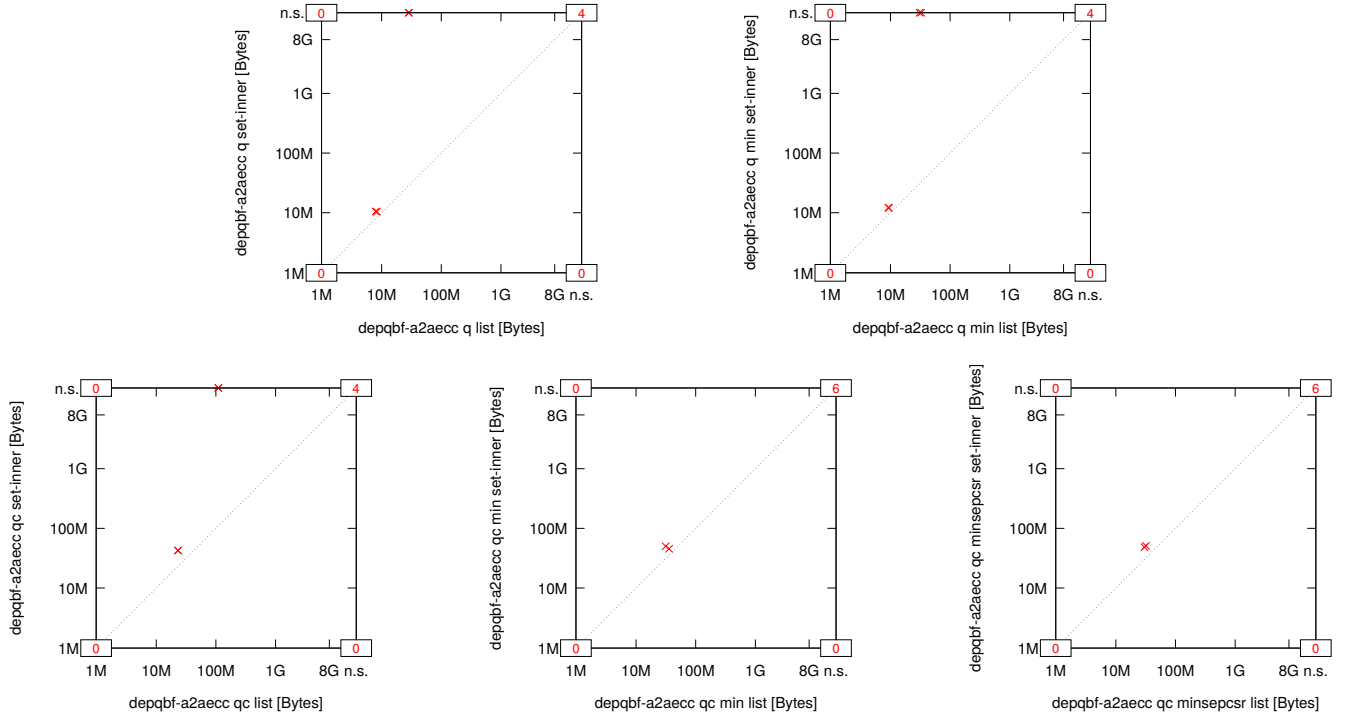


Fig. 764: Suite Katz ($n = 8$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

19) Klieber ($n = 15$):

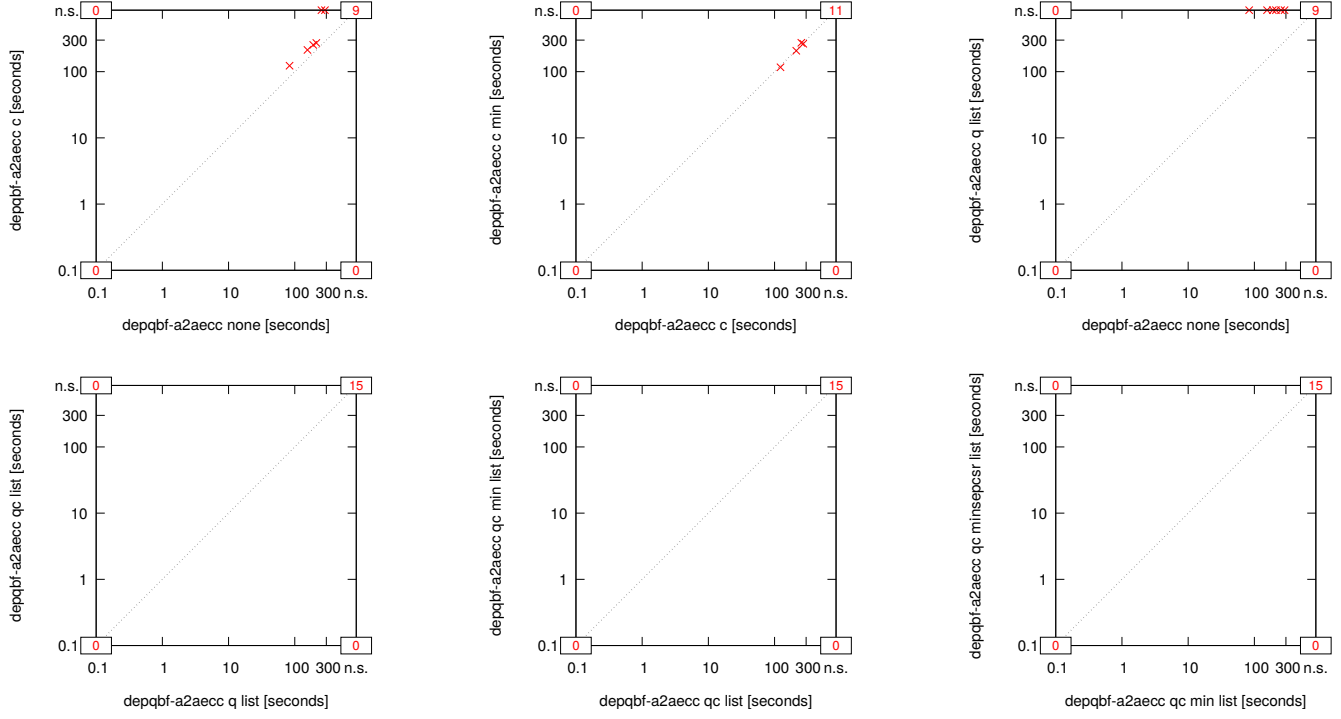


Fig. 765: Suite Klieber ($n = 15$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

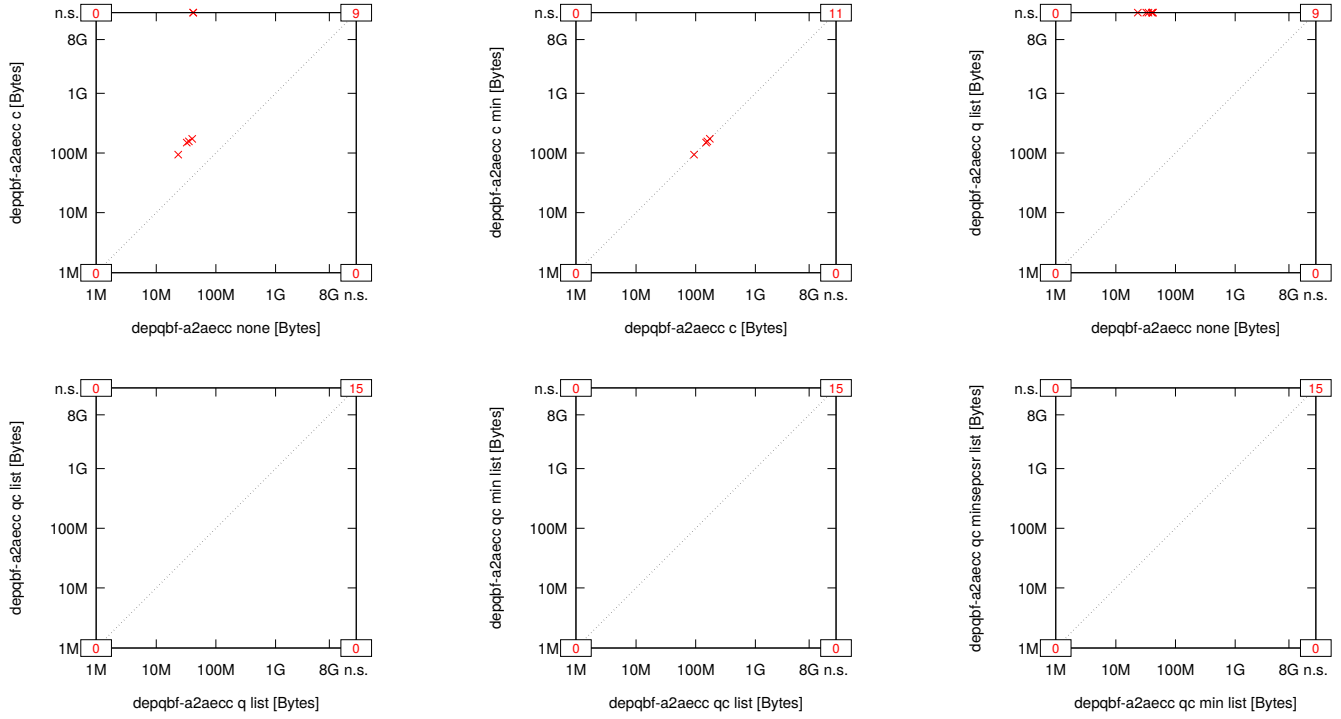


Fig. 766: Suite Klieber ($n = 15$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

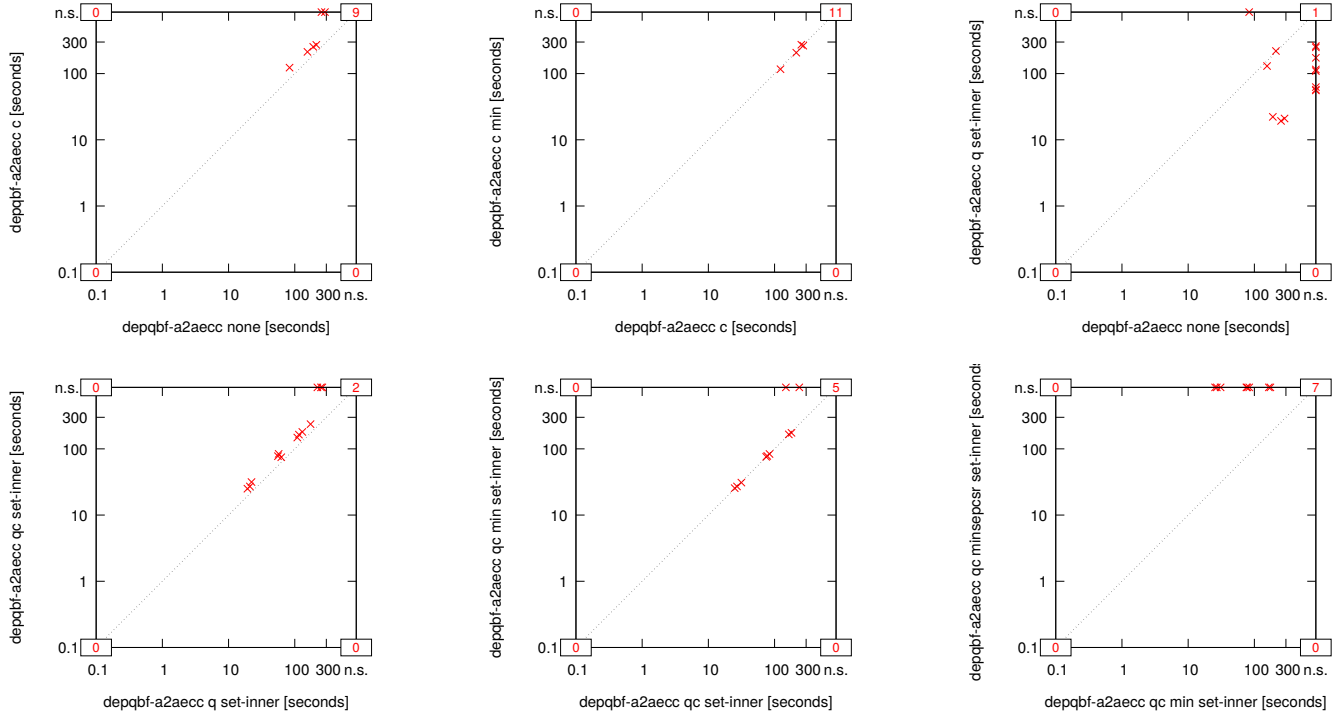


Fig. 767: Suite Klieber ($n = 15$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

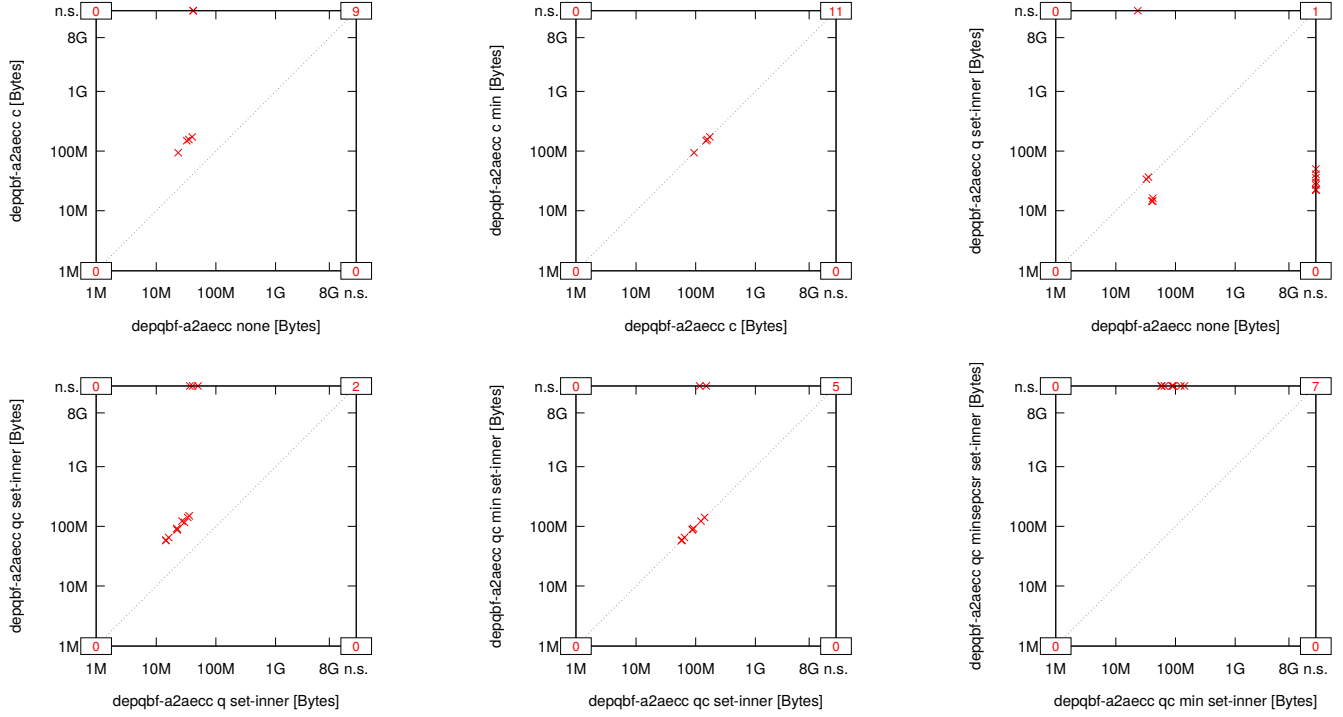


Fig. 768: Suite Klieber ($n = 15$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

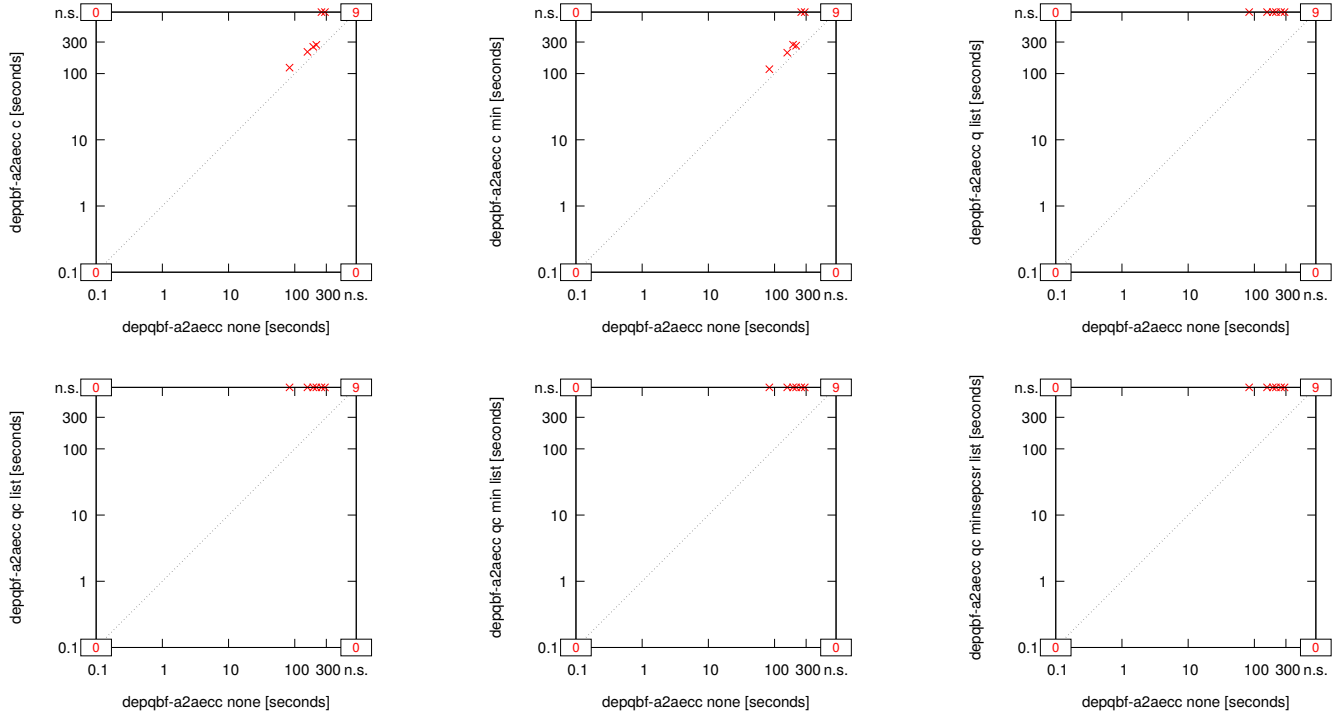


Fig. 769: Suite Klieber ($n = 15$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

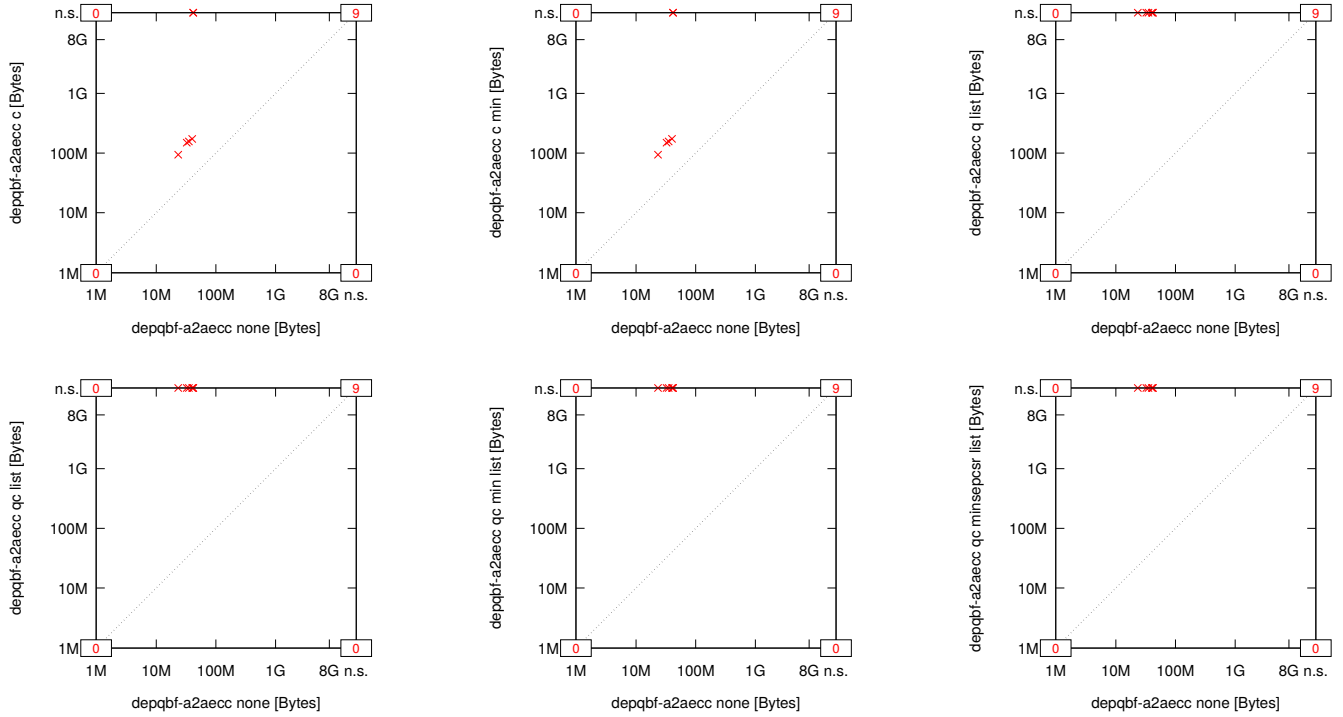


Fig. 770: Suite Klieber ($n = 15$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

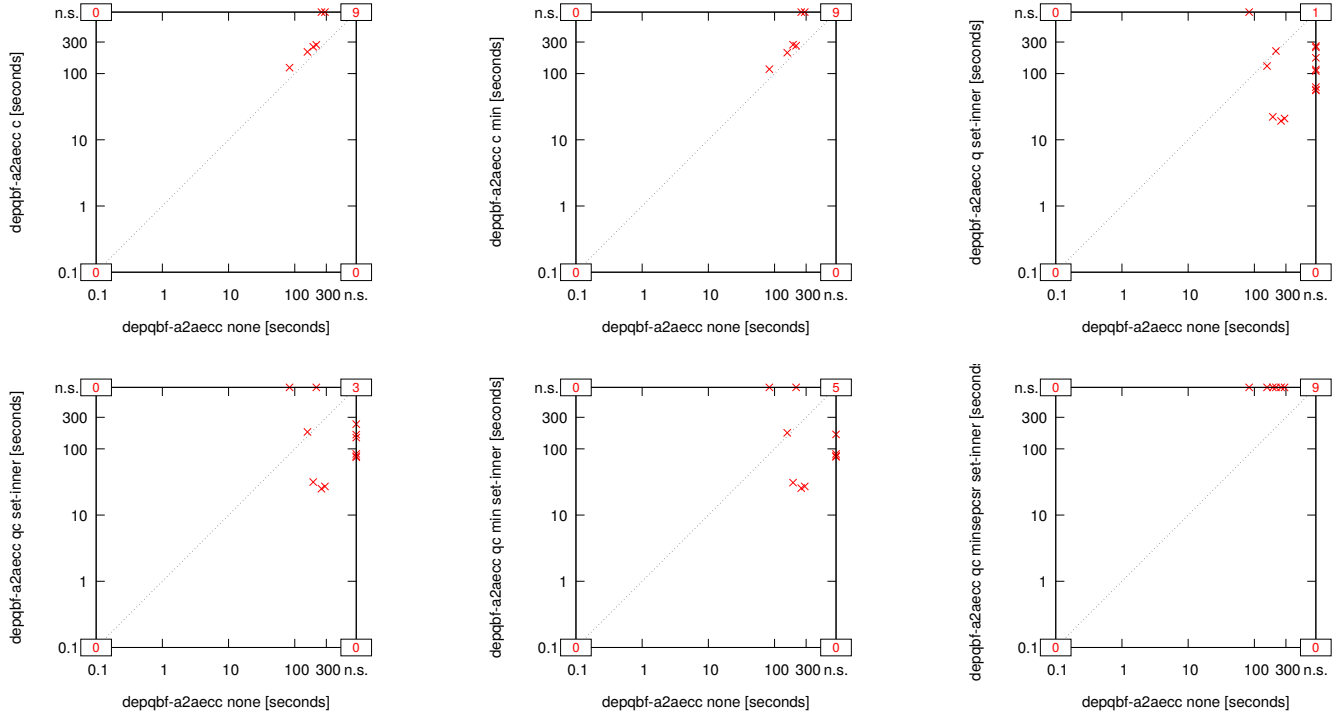


Fig. 771: Suite Klieber ($n = 15$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

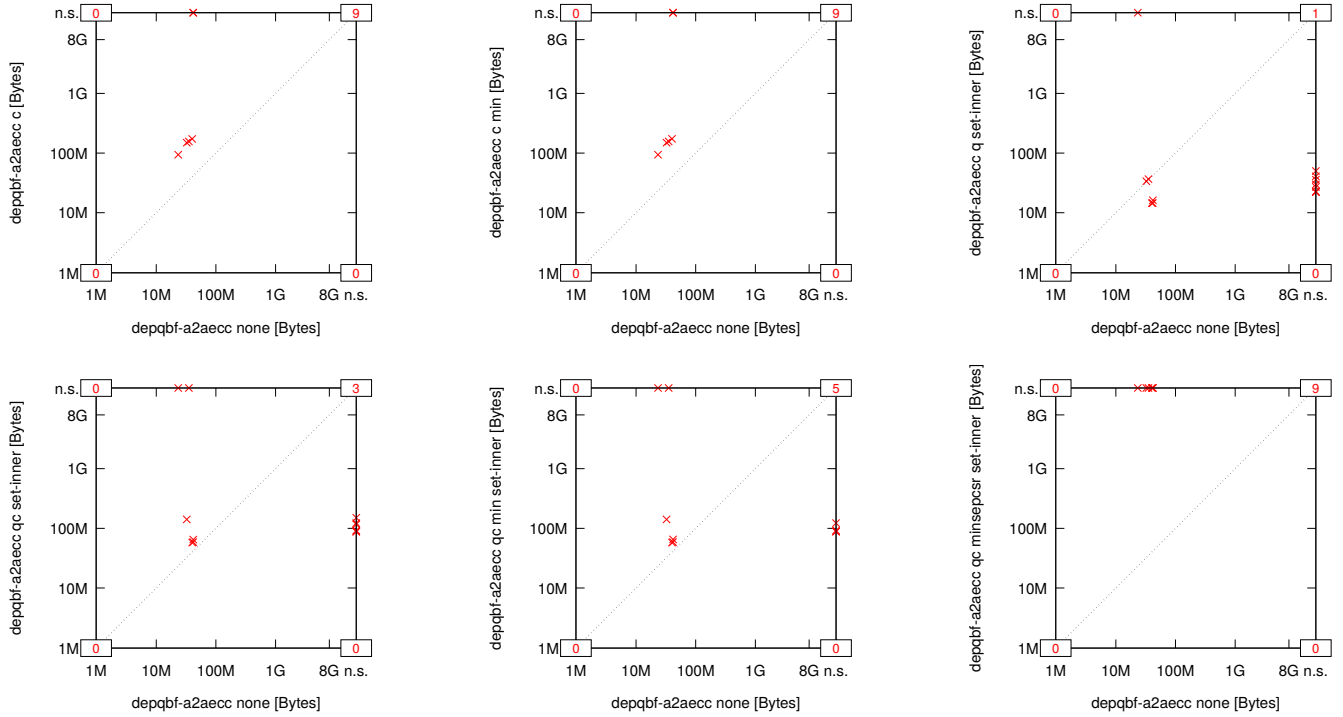


Fig. 772: Suite Klieber ($n = 15$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

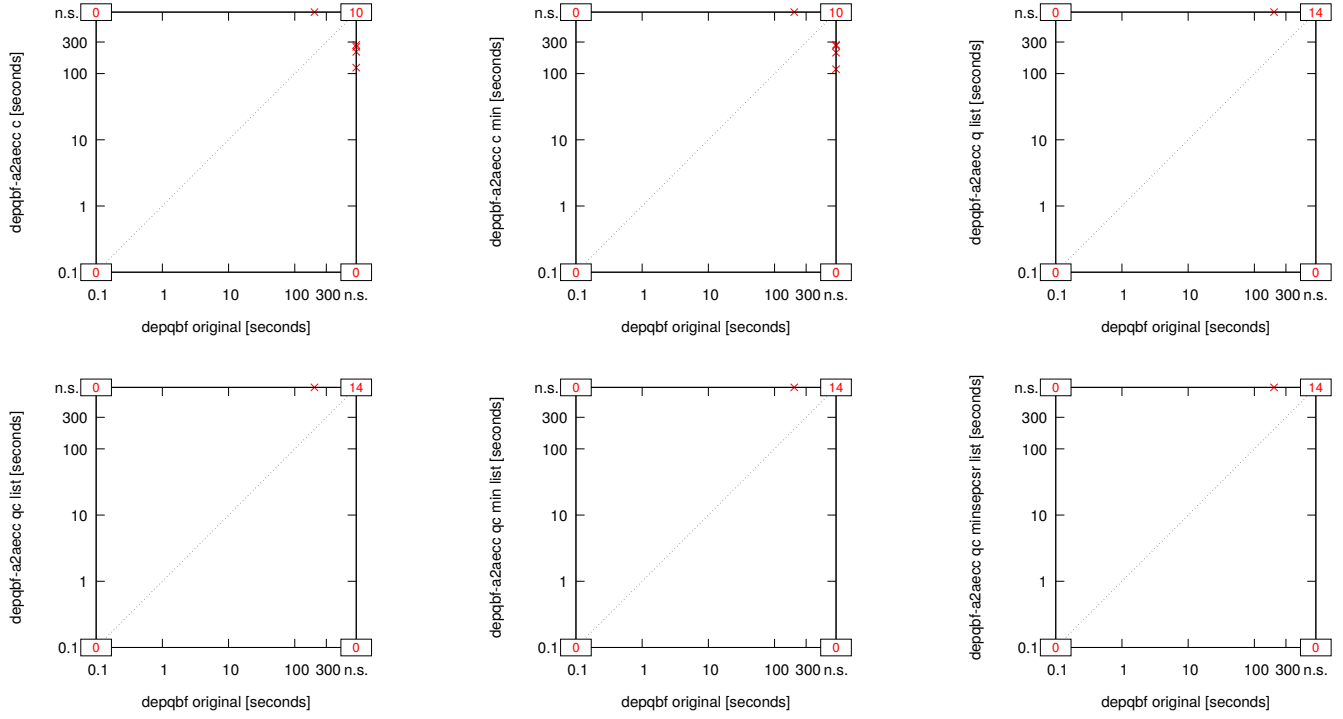


Fig. 773: Suite Klieber ($n = 15$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

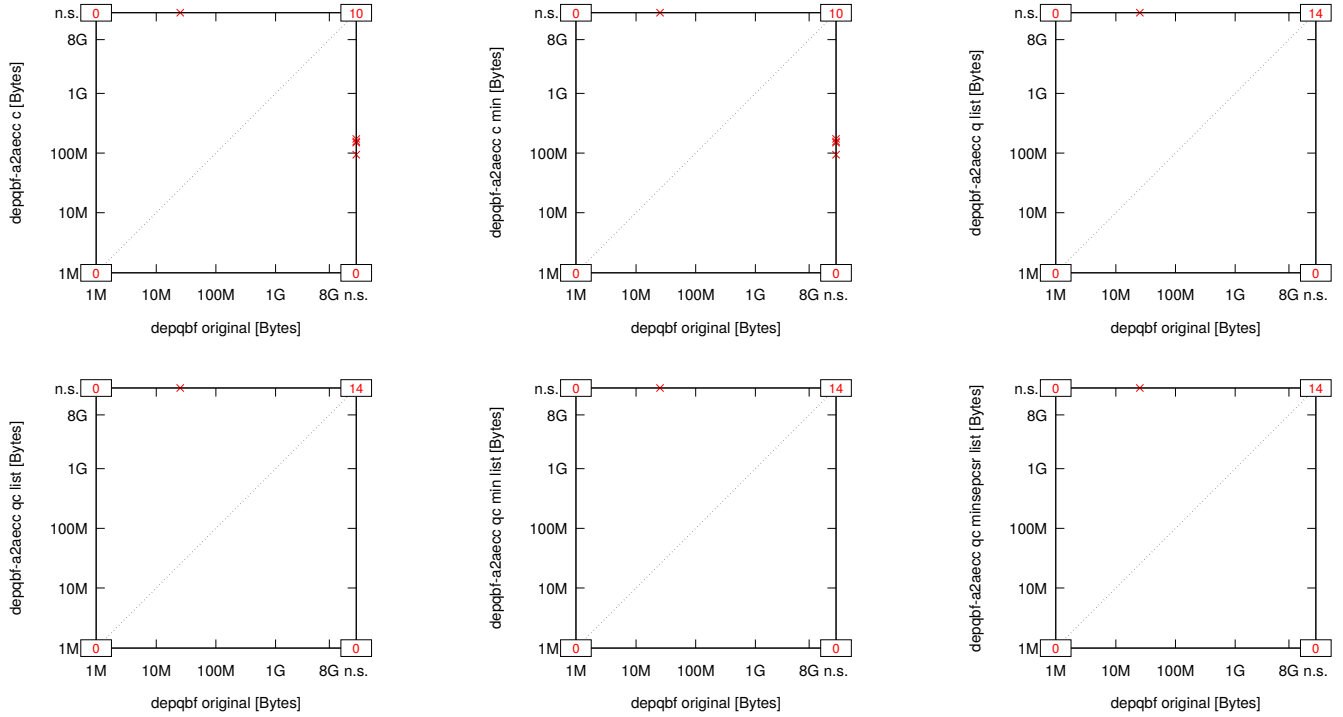


Fig. 774: Suite Klieber ($n = 15$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

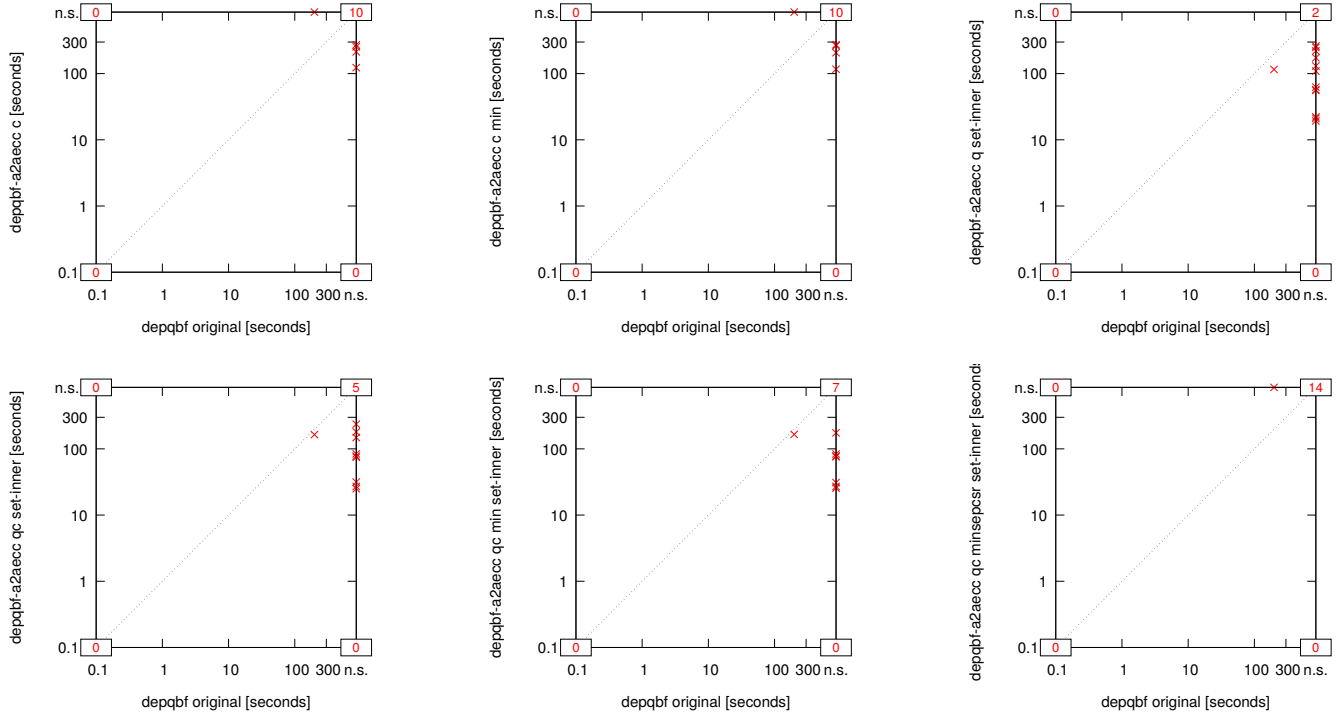


Fig. 775: Suite Klieber ($n = 15$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

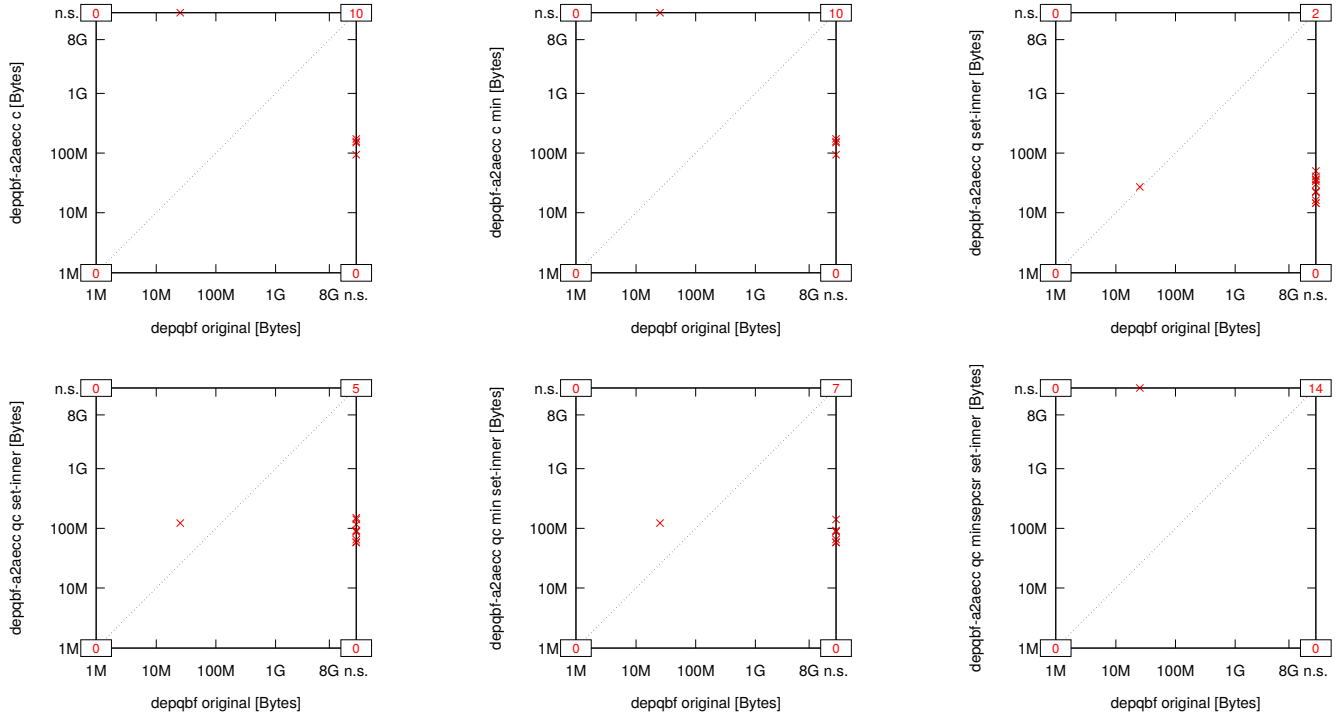


Fig. 776: Suite Klieber ($n = 15$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

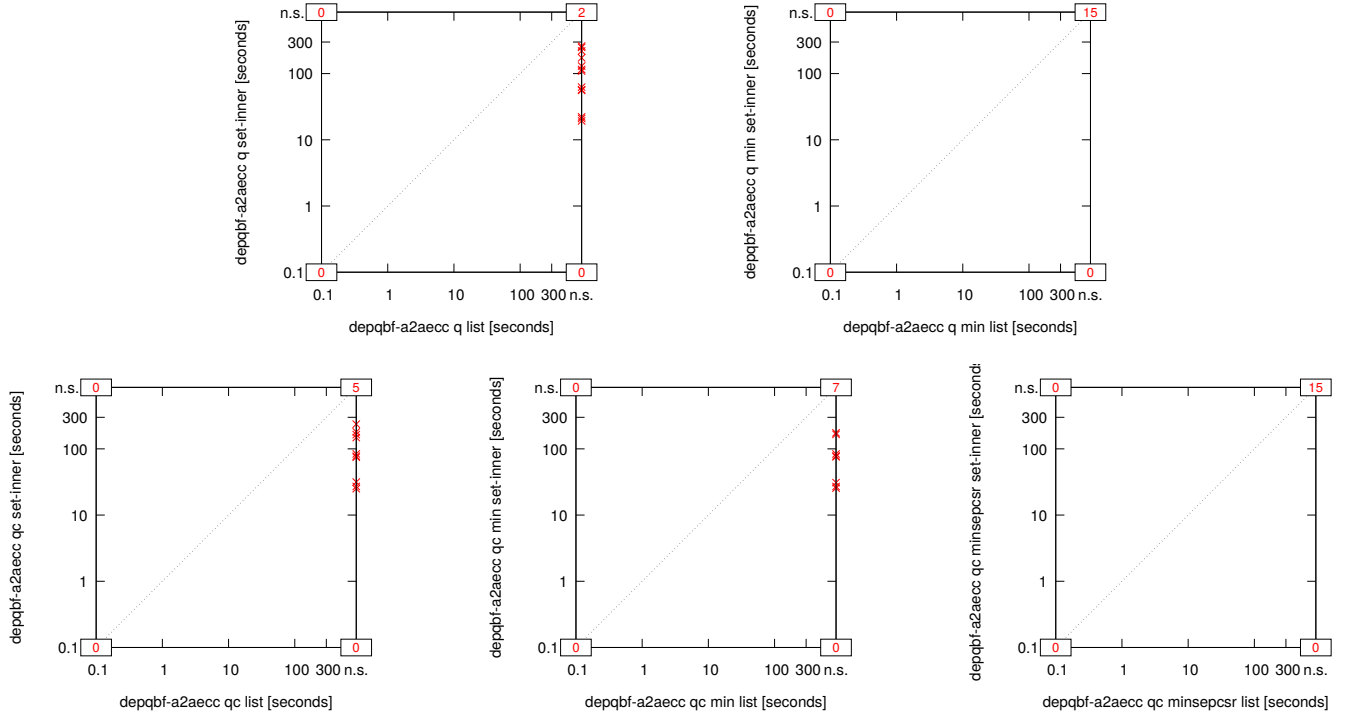


Fig. 777: Suite Klieber ($n = 15$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

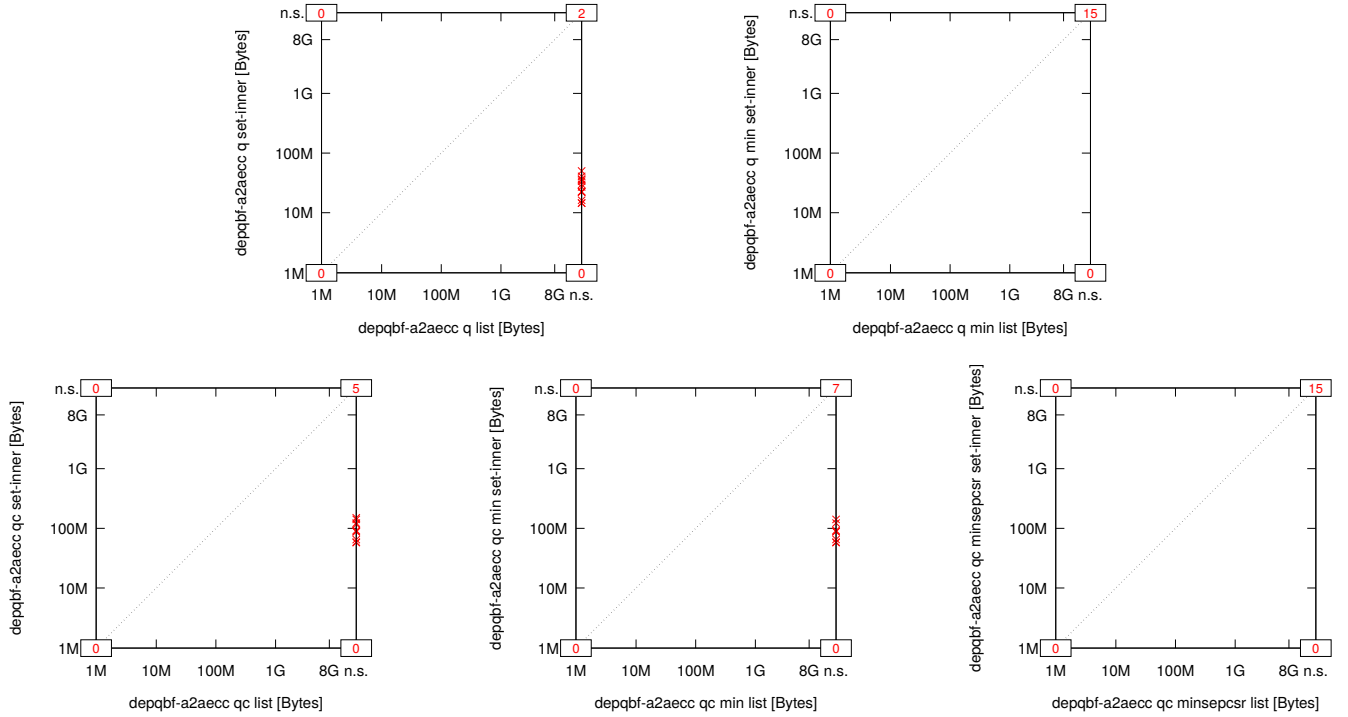


Fig. 778: Suite Klieber ($n = 15$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

20) Kontchakov ($n = 70$):

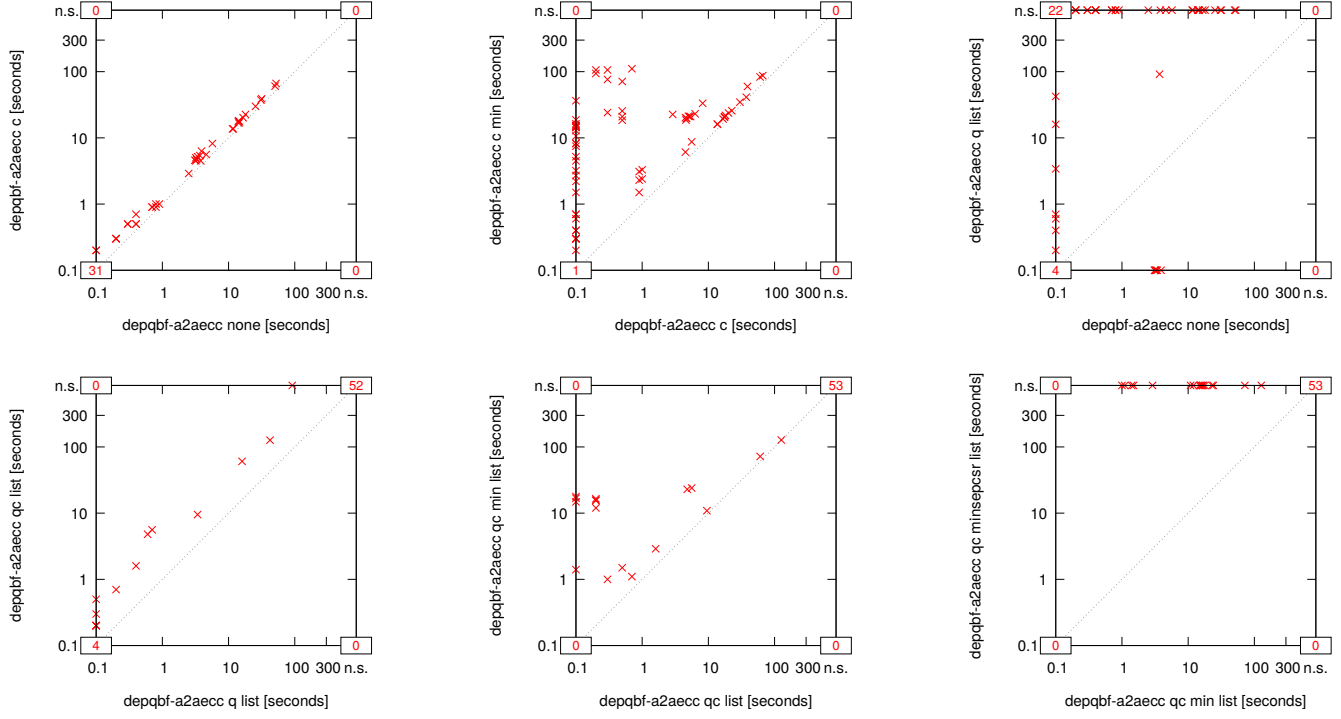


Fig. 779: Suite Kontchakov ($n = 70$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

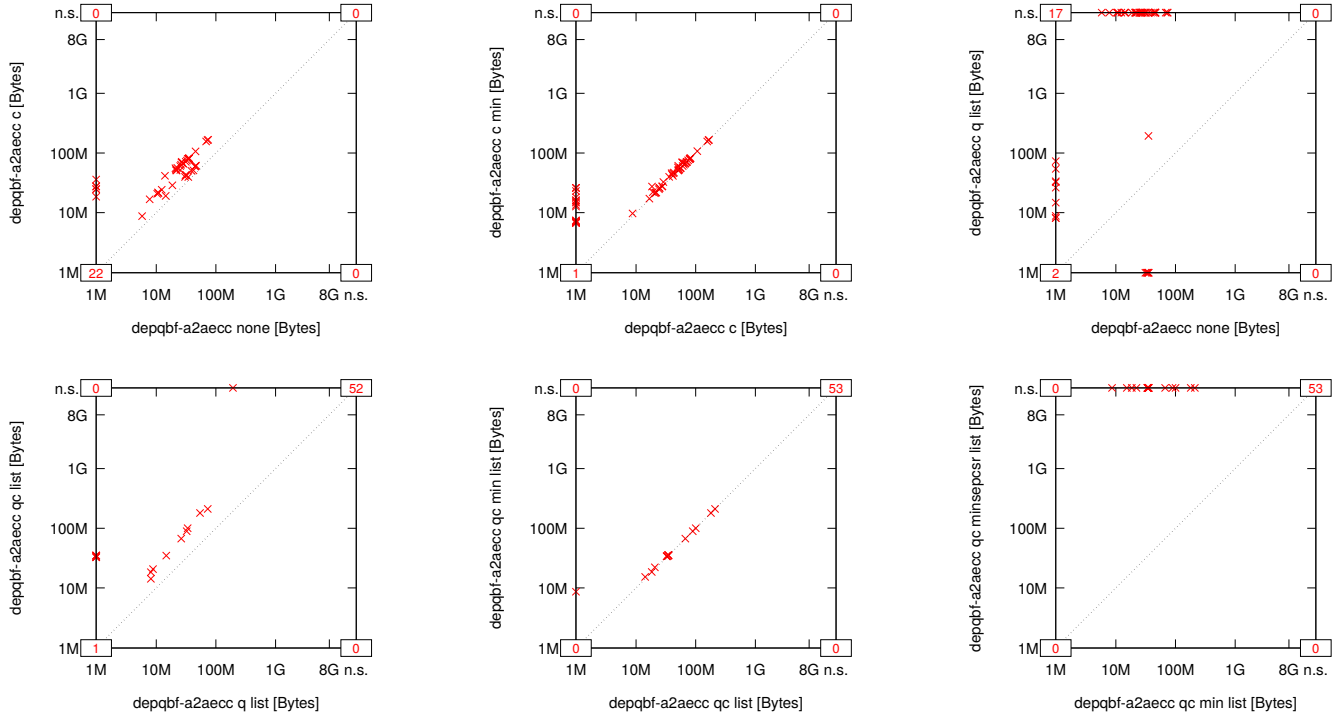


Fig. 780: Suite Kontchakov ($n = 70$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

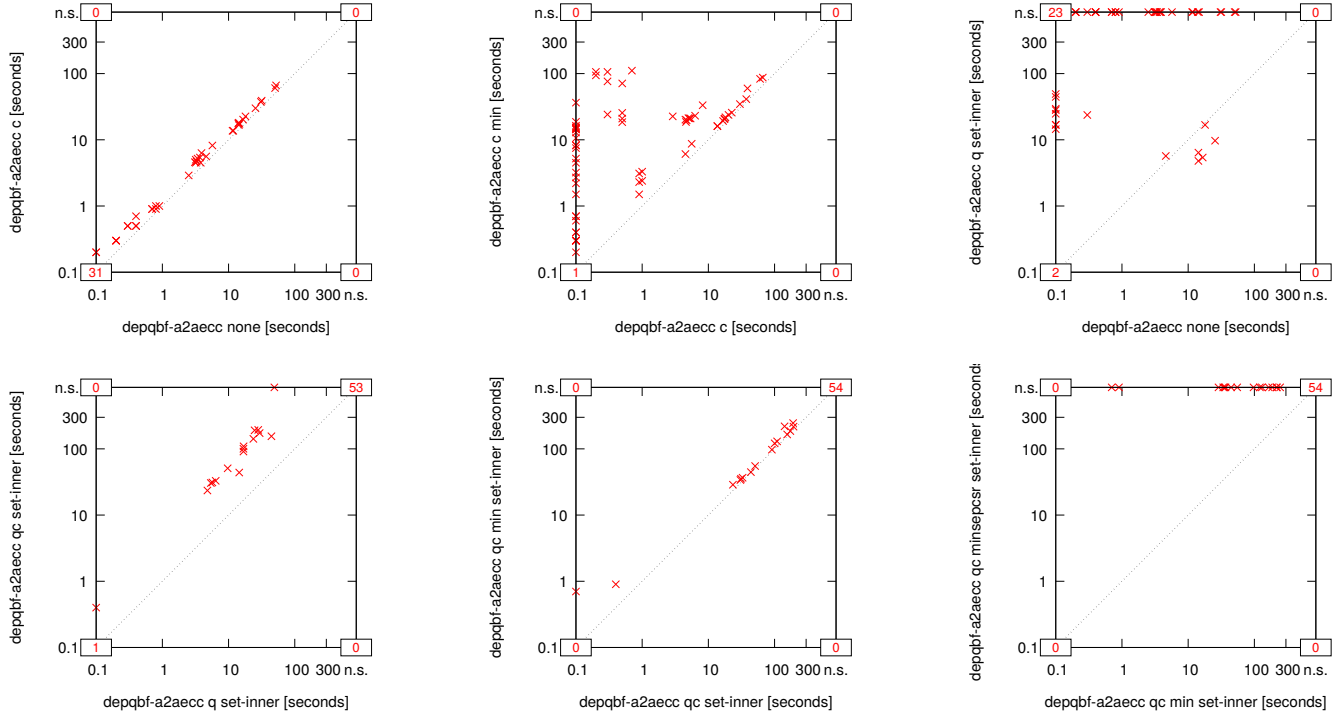


Fig. 781: Suite Kontchakov ($n = 70$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

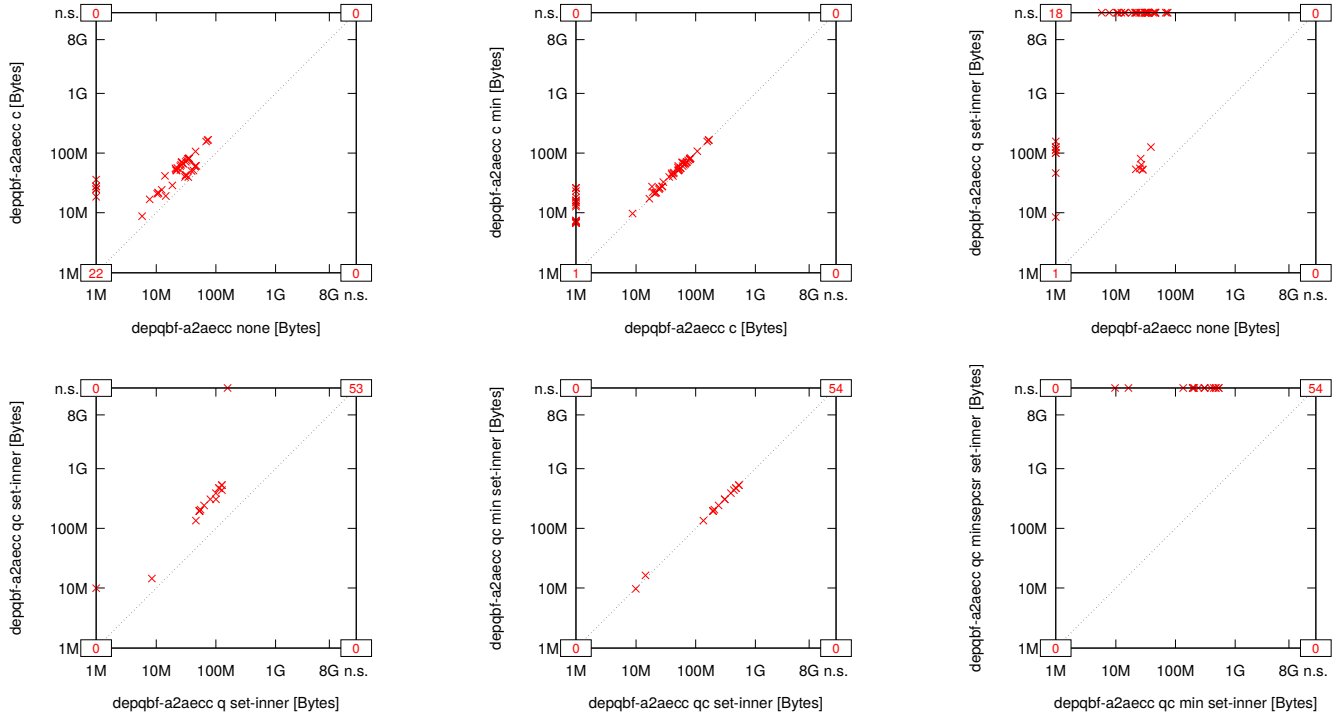


Fig. 782: Suite Kontchakov ($n = 70$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

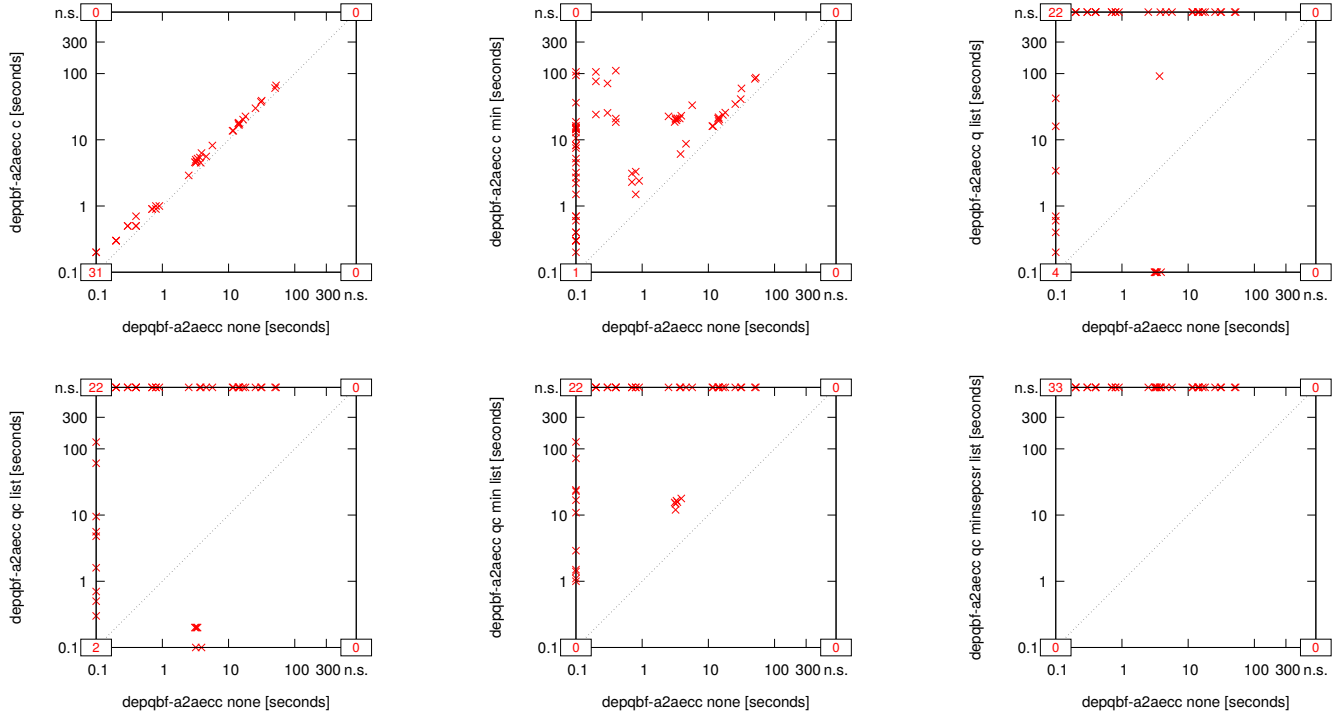


Fig. 783: Suite Kontchakov ($n = 70$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

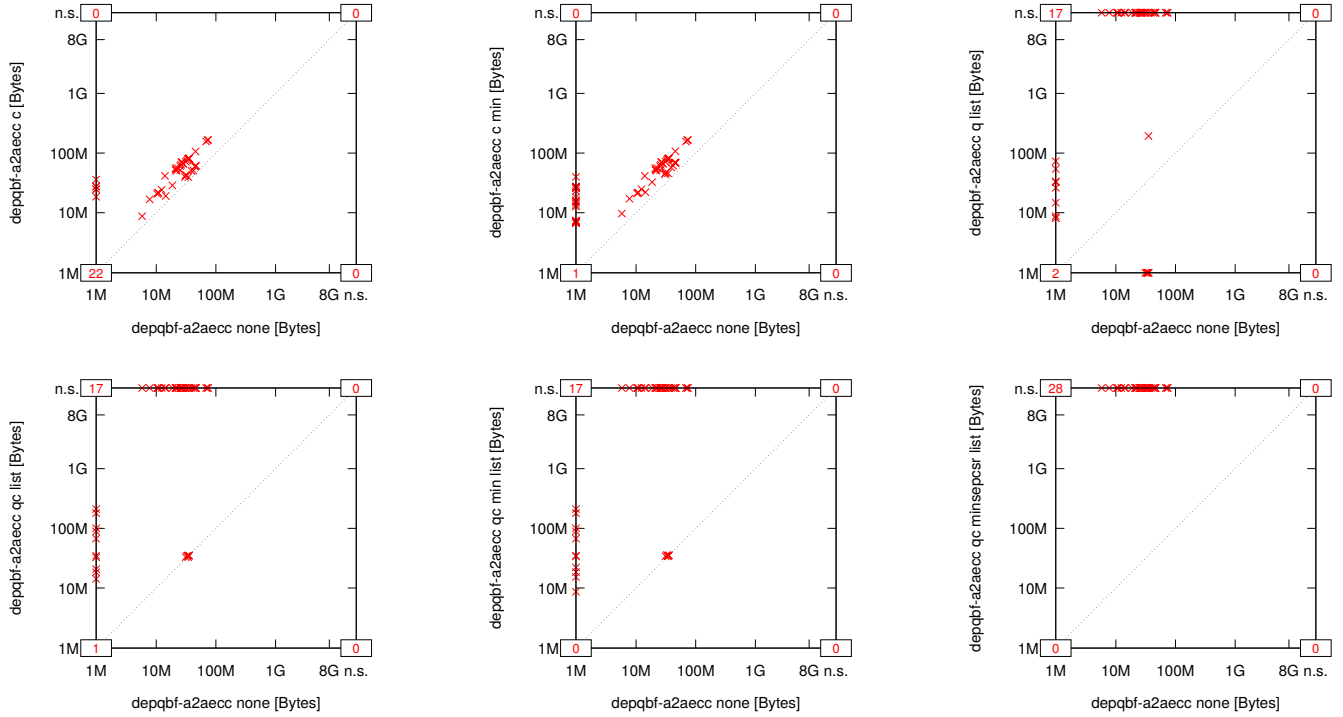


Fig. 784: Suite Kontchakov ($n = 70$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

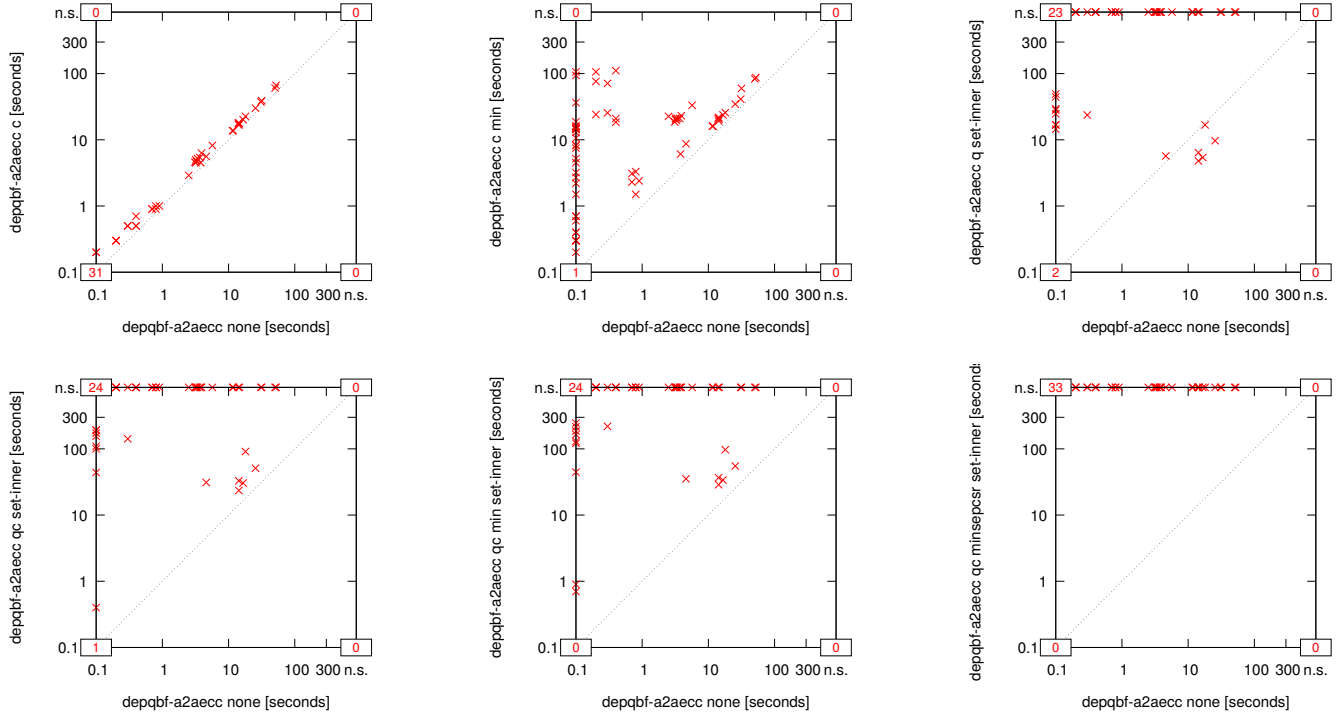


Fig. 785: Suite Kontchakov ($n = 70$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

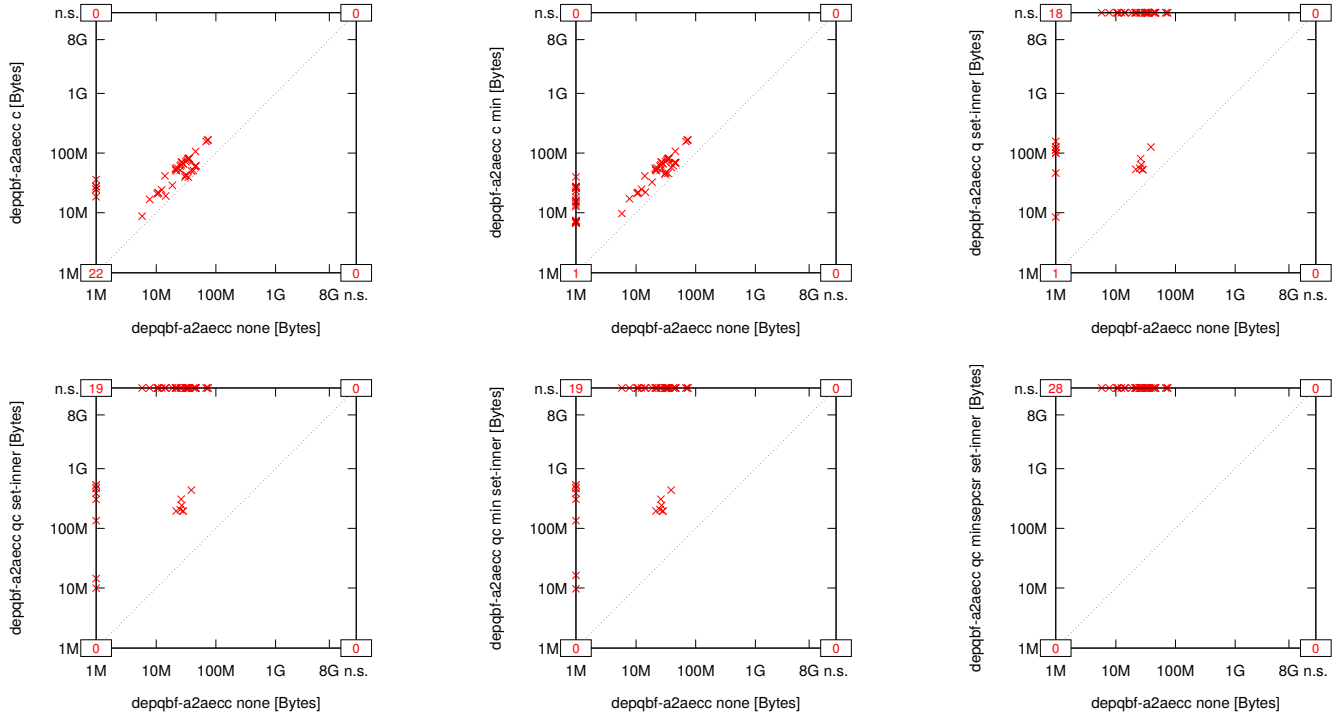


Fig. 786: Suite Kontchakov ($n = 70$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

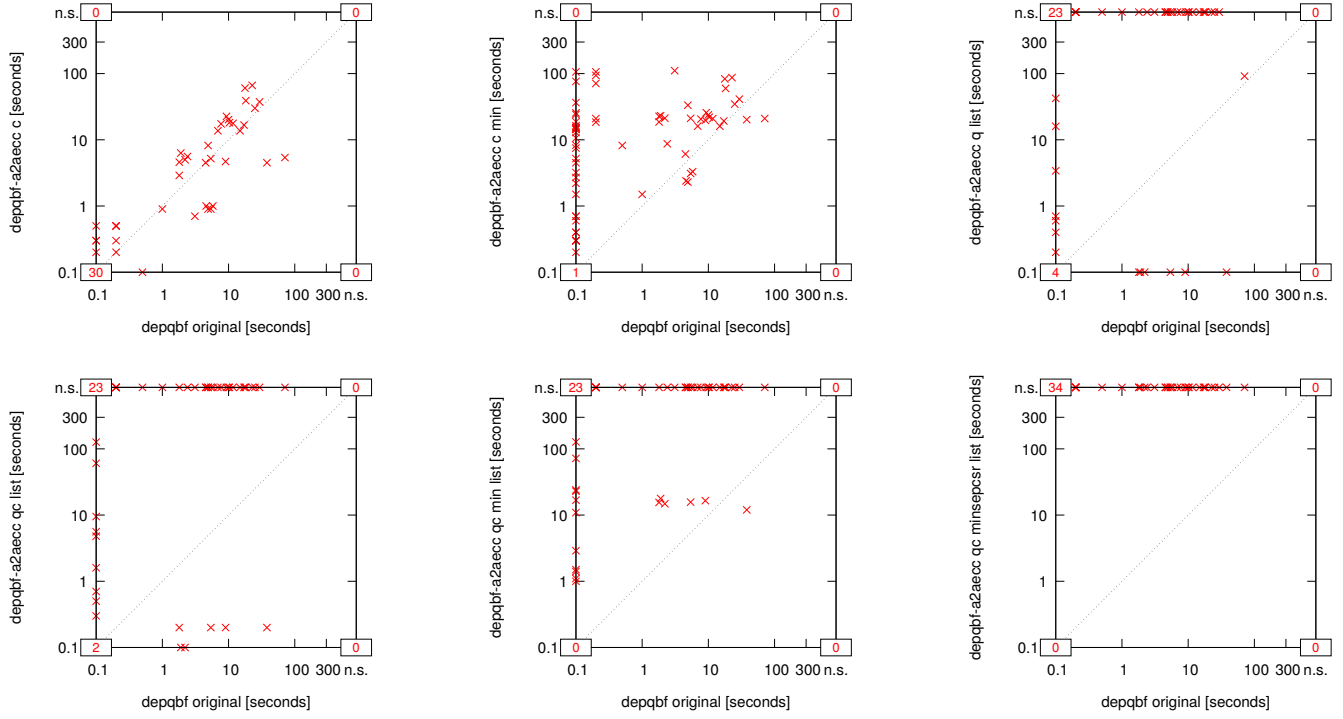


Fig. 787: Suite Kontchakov ($n = 70$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

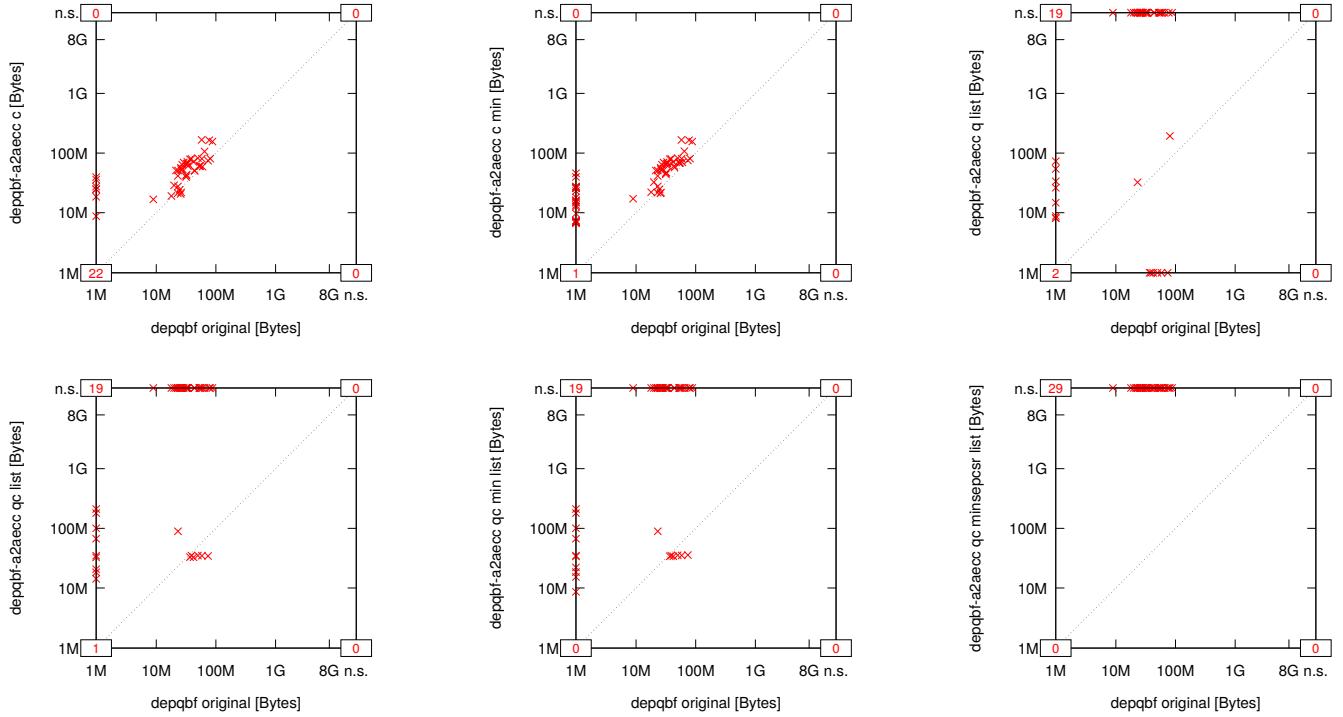


Fig. 788: Suite Kontchakov ($n = 70$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

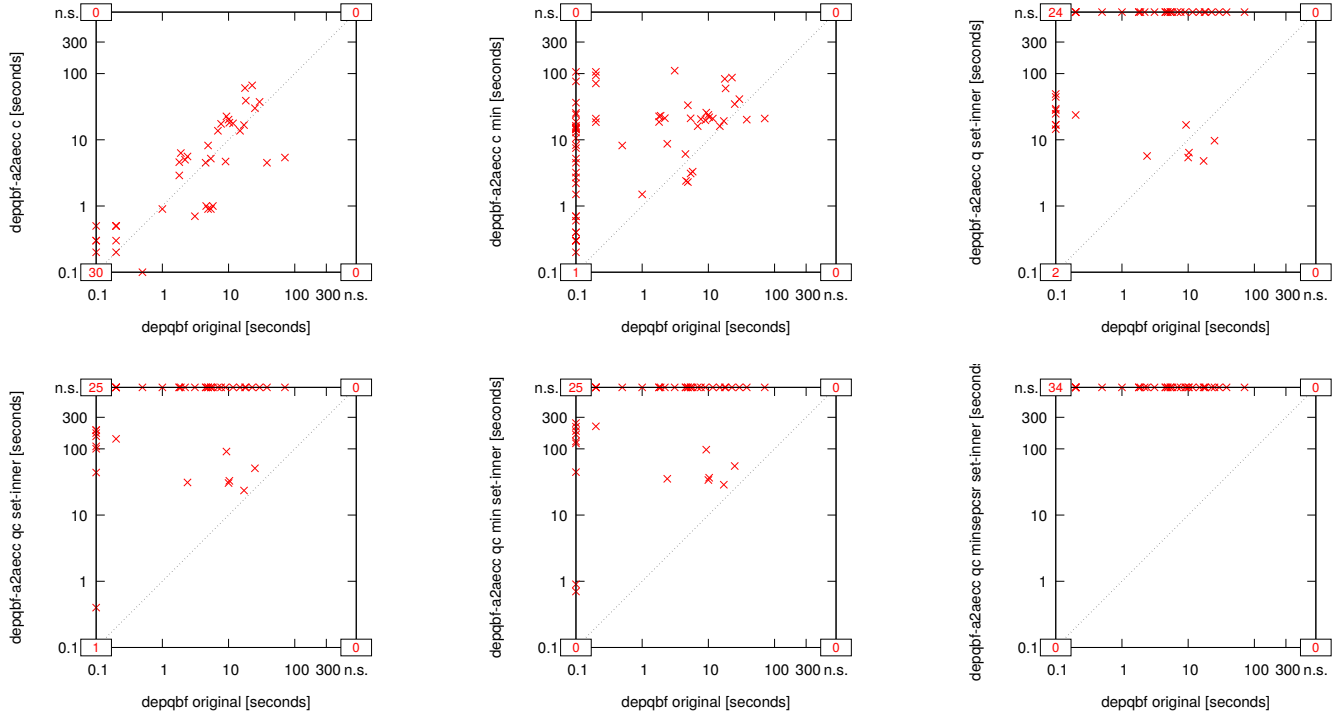


Fig. 789: Suite Kontchakov ($n = 70$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

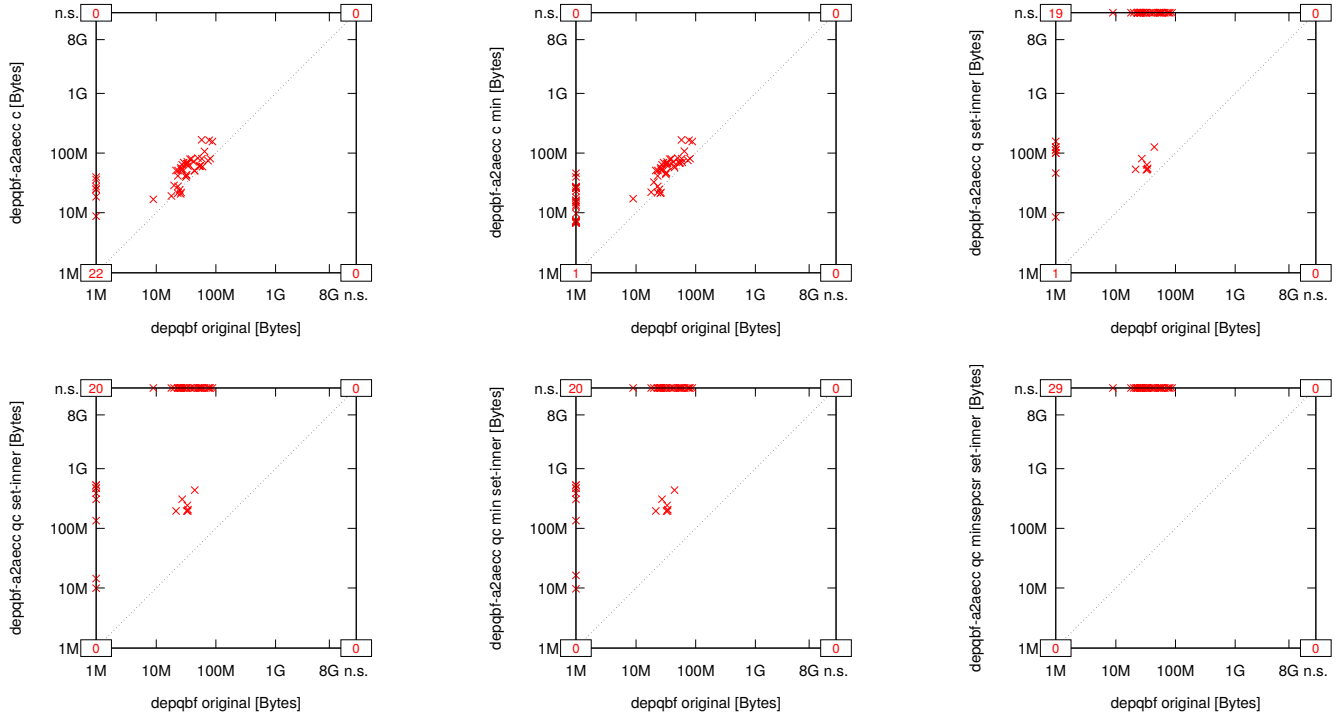


Fig. 790: Suite Kontchakov ($n = 70$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

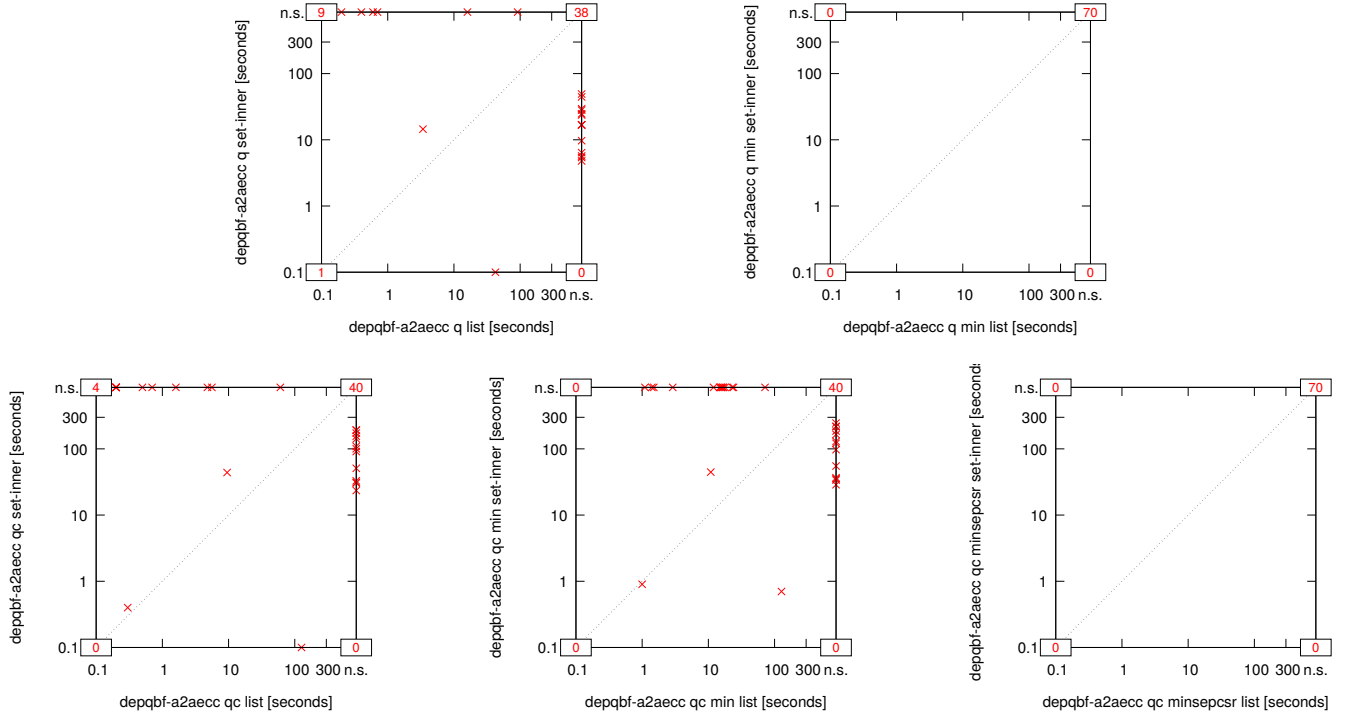


Fig. 791: Suite Kontchakov ($n = 70$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

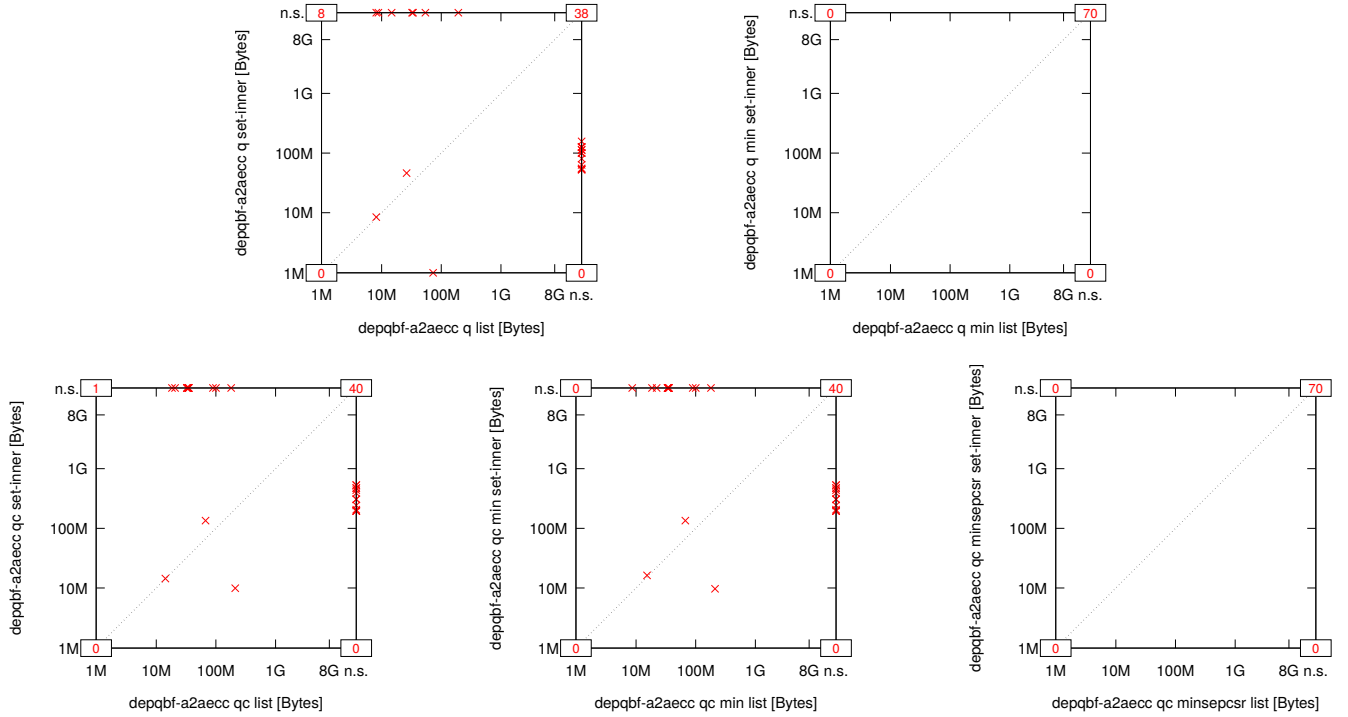


Fig. 792: Suite Kontchakov ($n = 70$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

21) Kronegger-Pfandler-Pichler ($n = 133$):

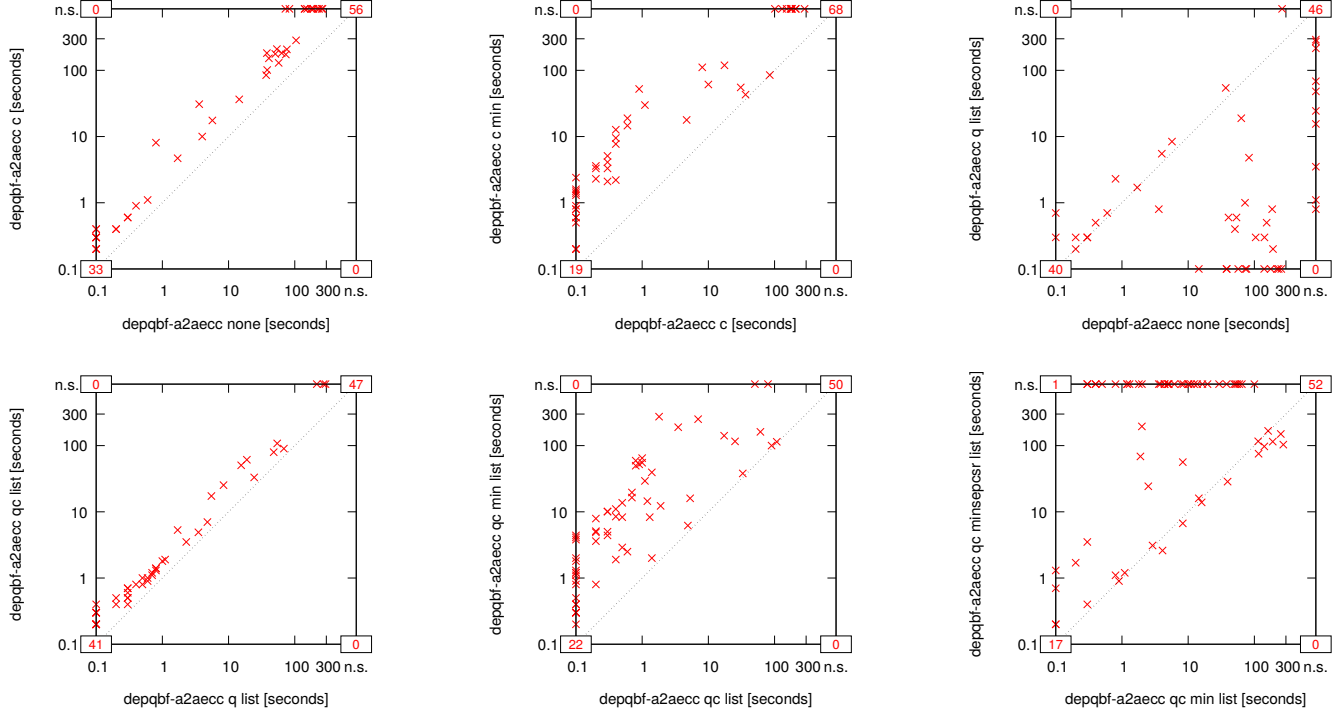


Fig. 793: Suite Kronegger-Pfandler-Pichler ($n = 133$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

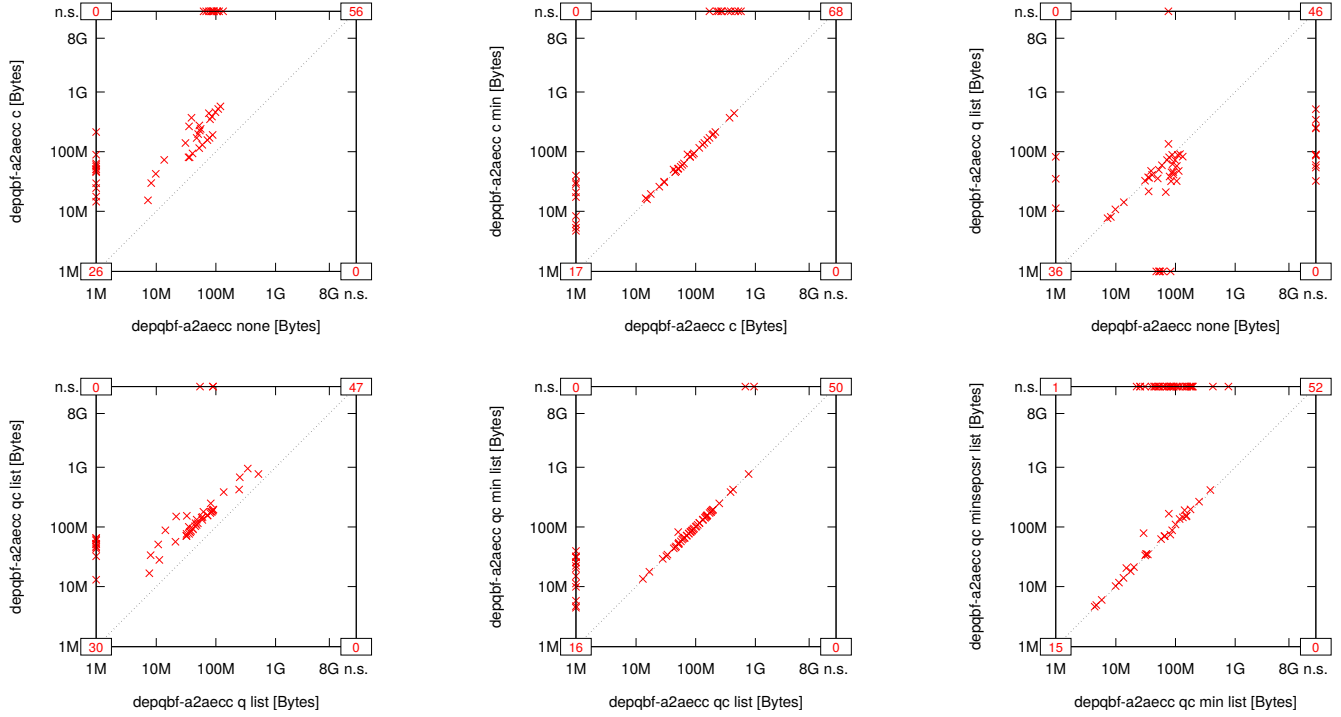


Fig. 794: Suite Kronegger-Pfandler-Pichler ($n = 133$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

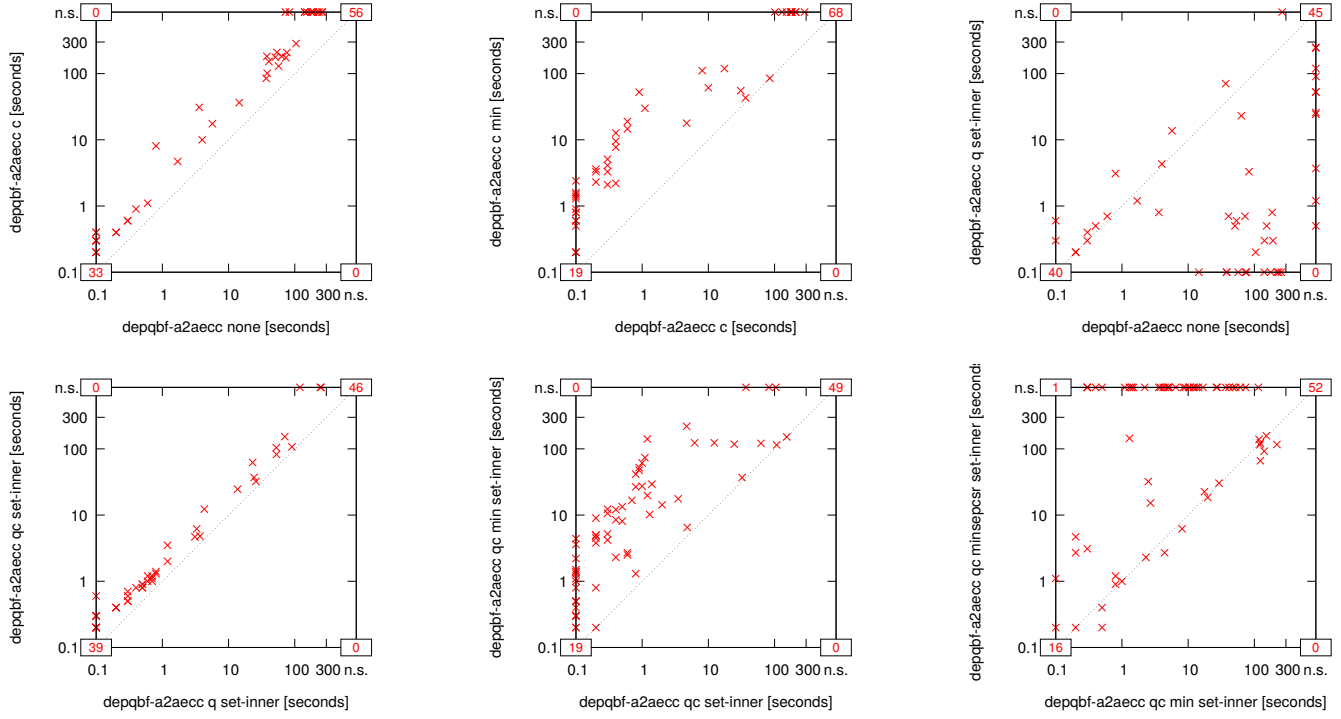


Fig. 795: Suite Kronegger-Pfandler-Pichler ($n = 133$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

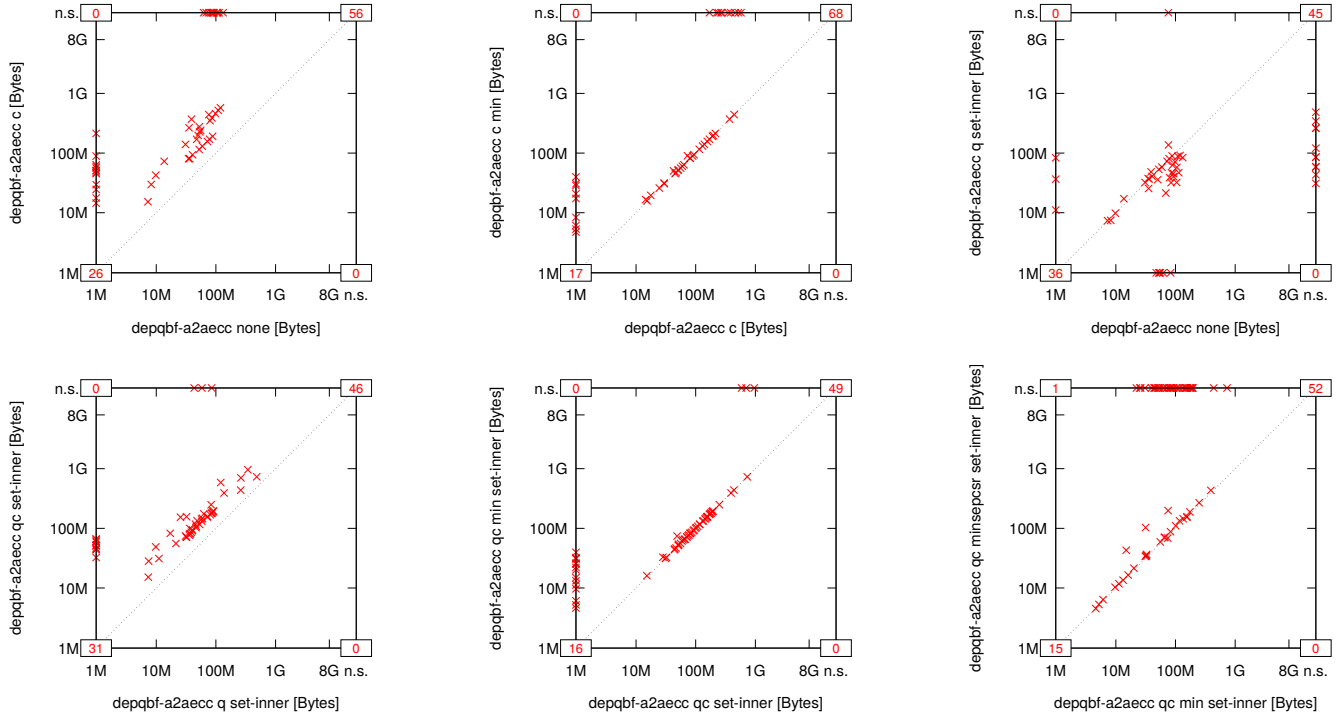


Fig. 796: Suite Kronegger-Pfandler-Pichler ($n = 133$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

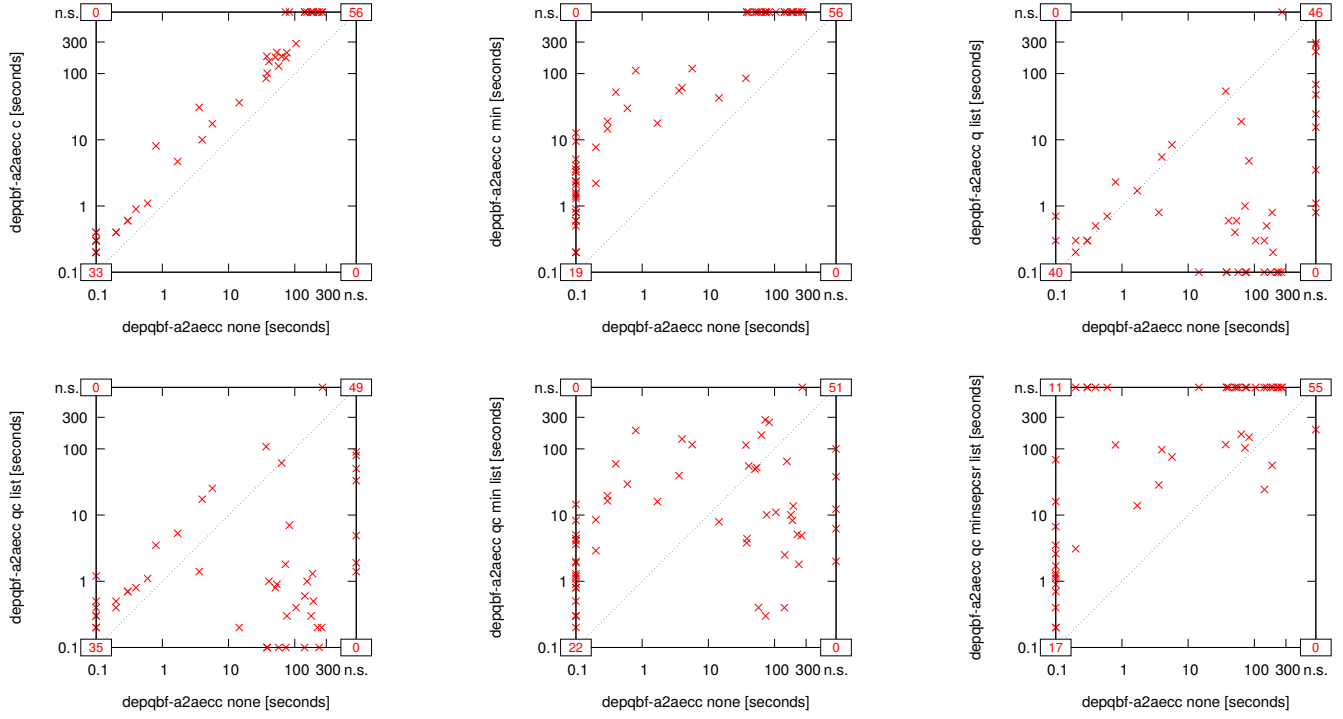


Fig. 797: Suite Kronegger-Pfandler-Pichler ($n = 133$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

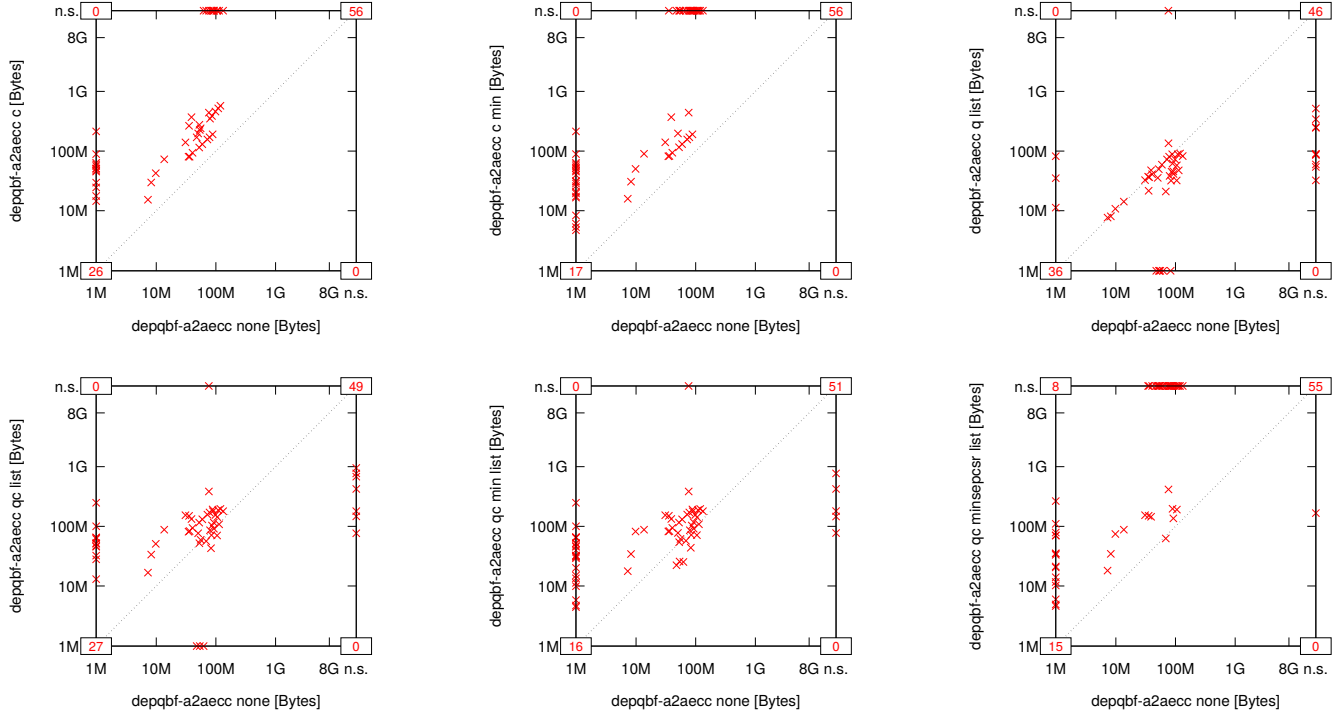


Fig. 798: Suite Kronegger-Pfandler-Pichler ($n = 133$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

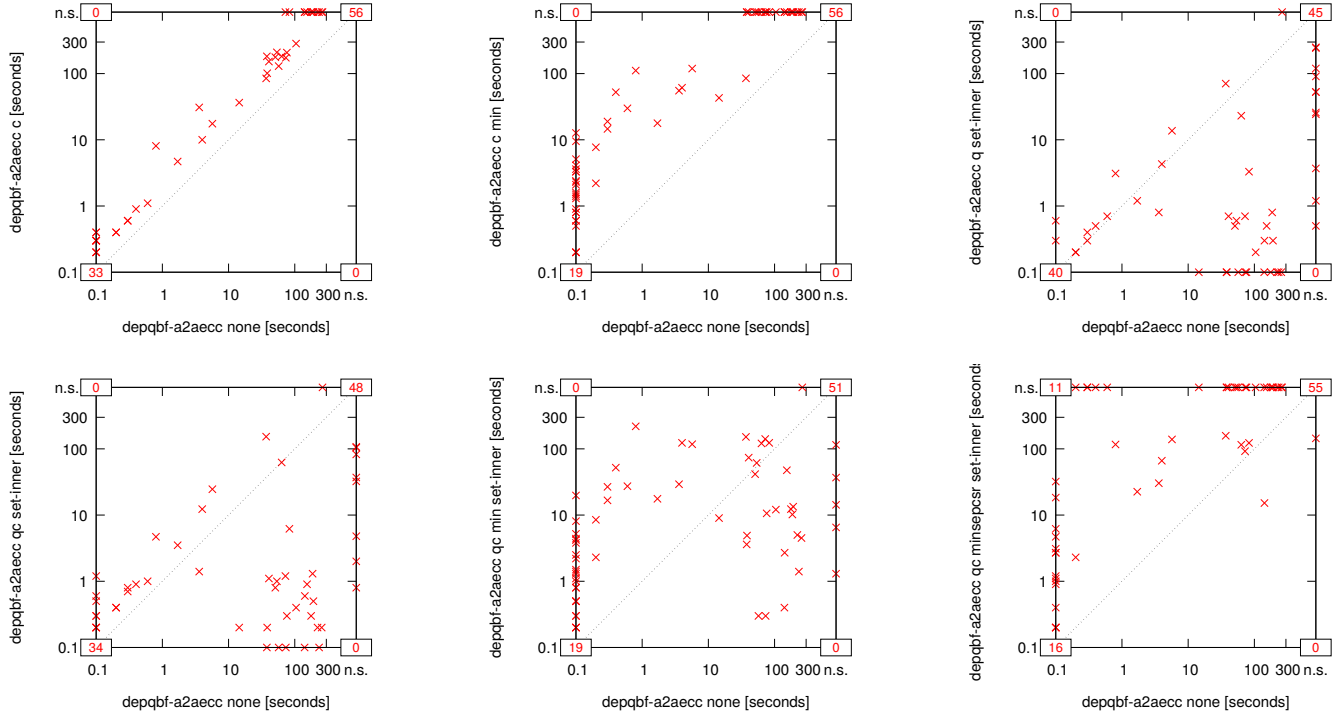


Fig. 799: Suite Kronegger-Pfandler-Pichler ($n = 133$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

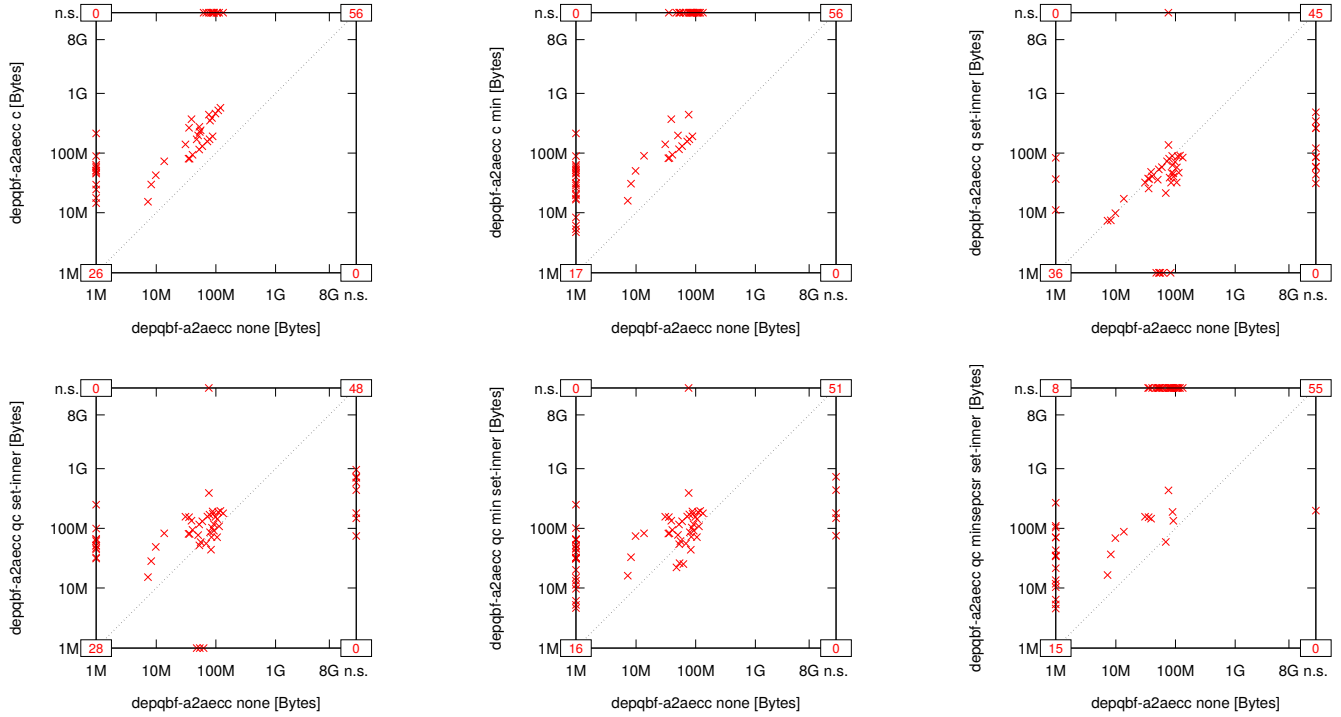


Fig. 800: Suite Kronegger-Pfandler-Pichler ($n = 133$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

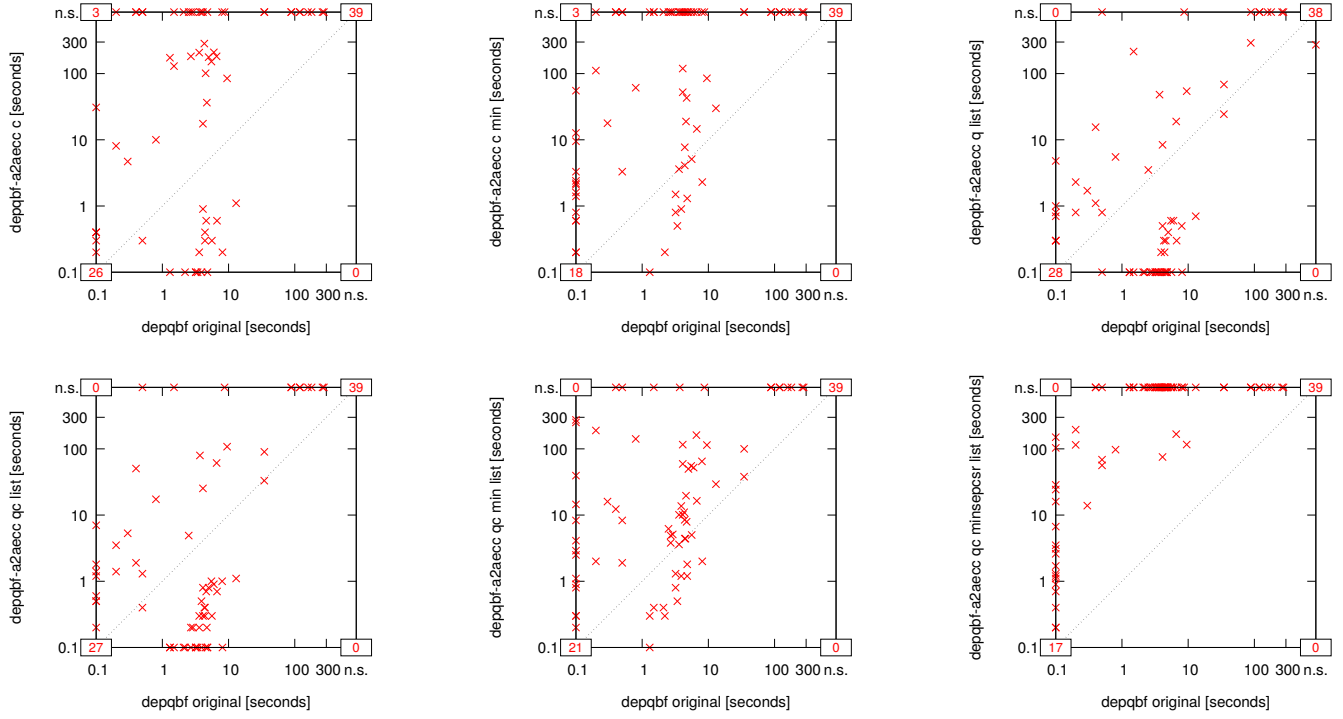


Fig. 801: Suite Kronegger-Pfandler-Pichler ($n = 133$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

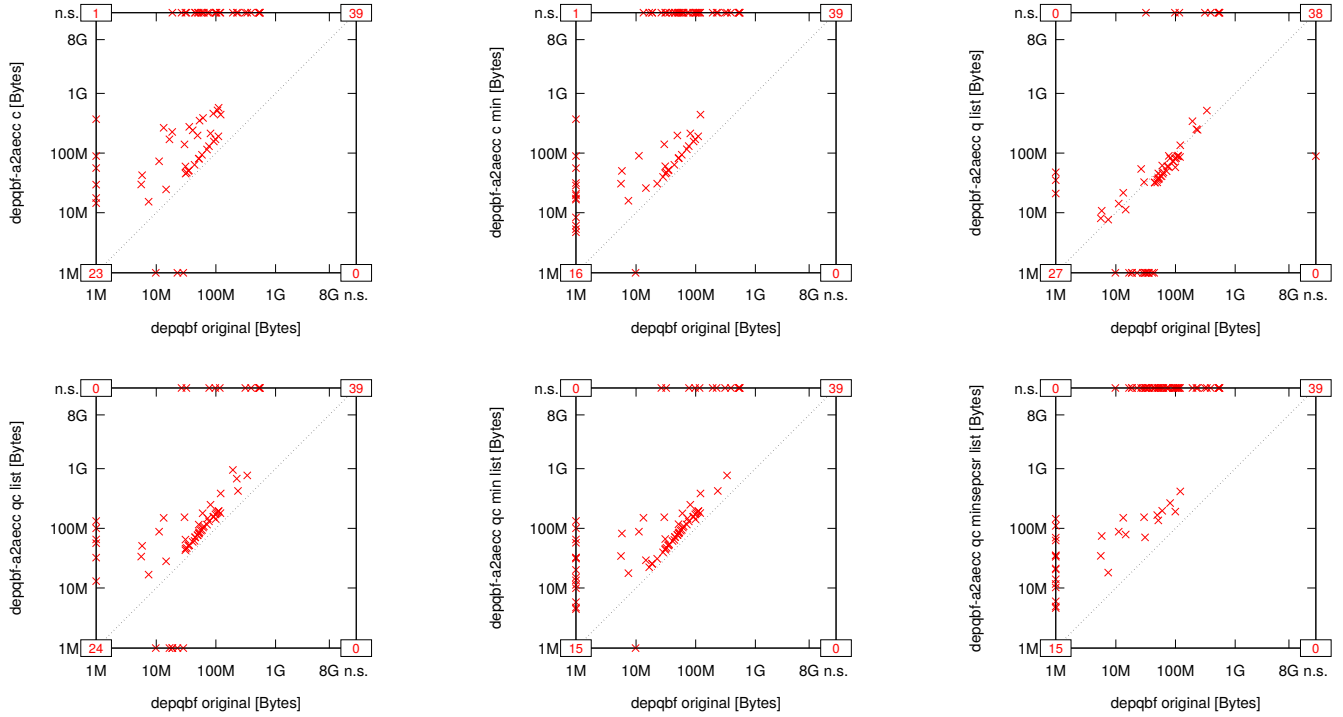


Fig. 802: Suite Kronegger-Pfandler-Pichler ($n = 133$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

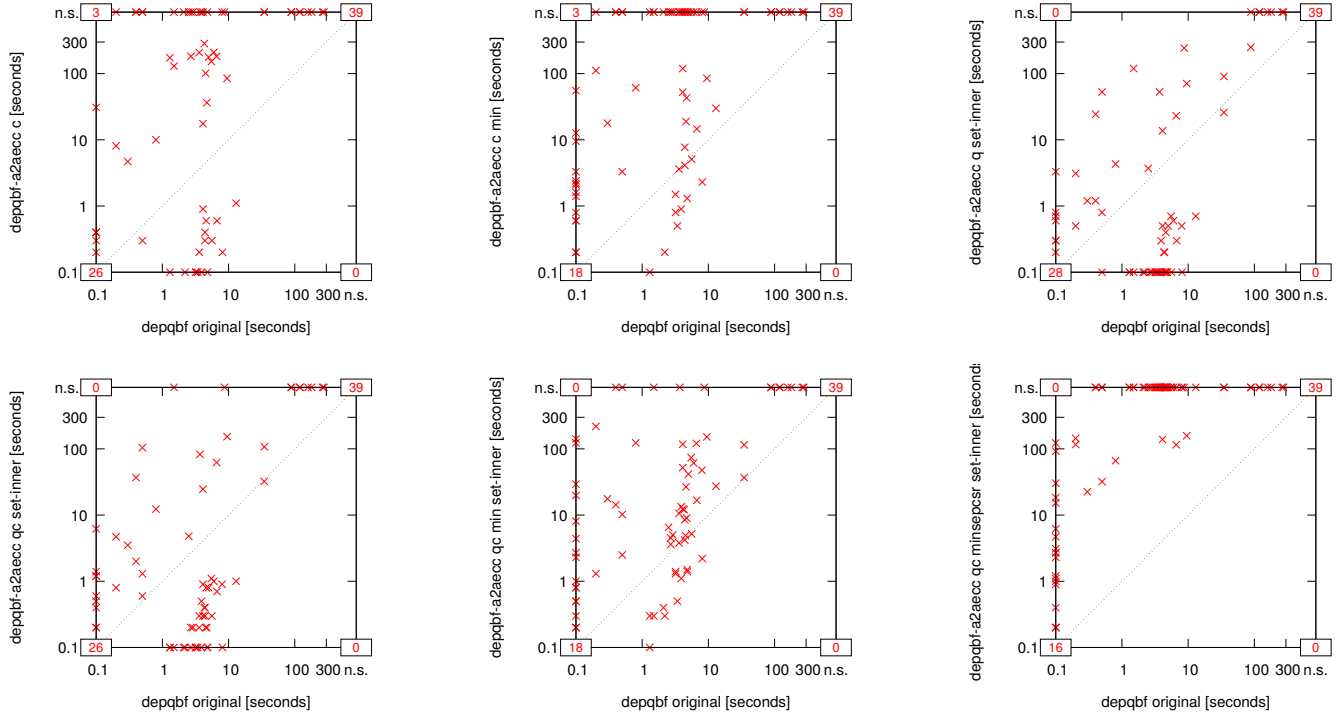


Fig. 803: Suite Kronegger-Pfandler-Pichler ($n = 133$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

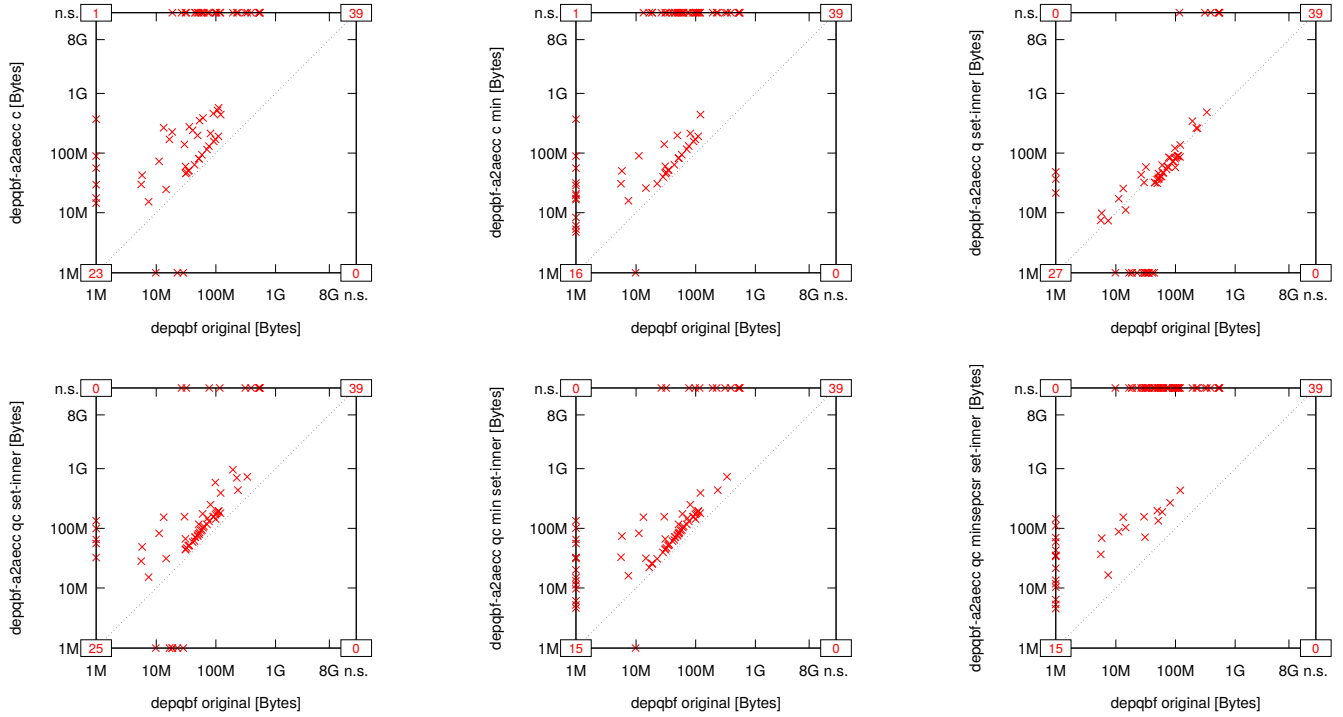


Fig. 804: Suite Kronegger-Pfandler-Pichler ($n = 133$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

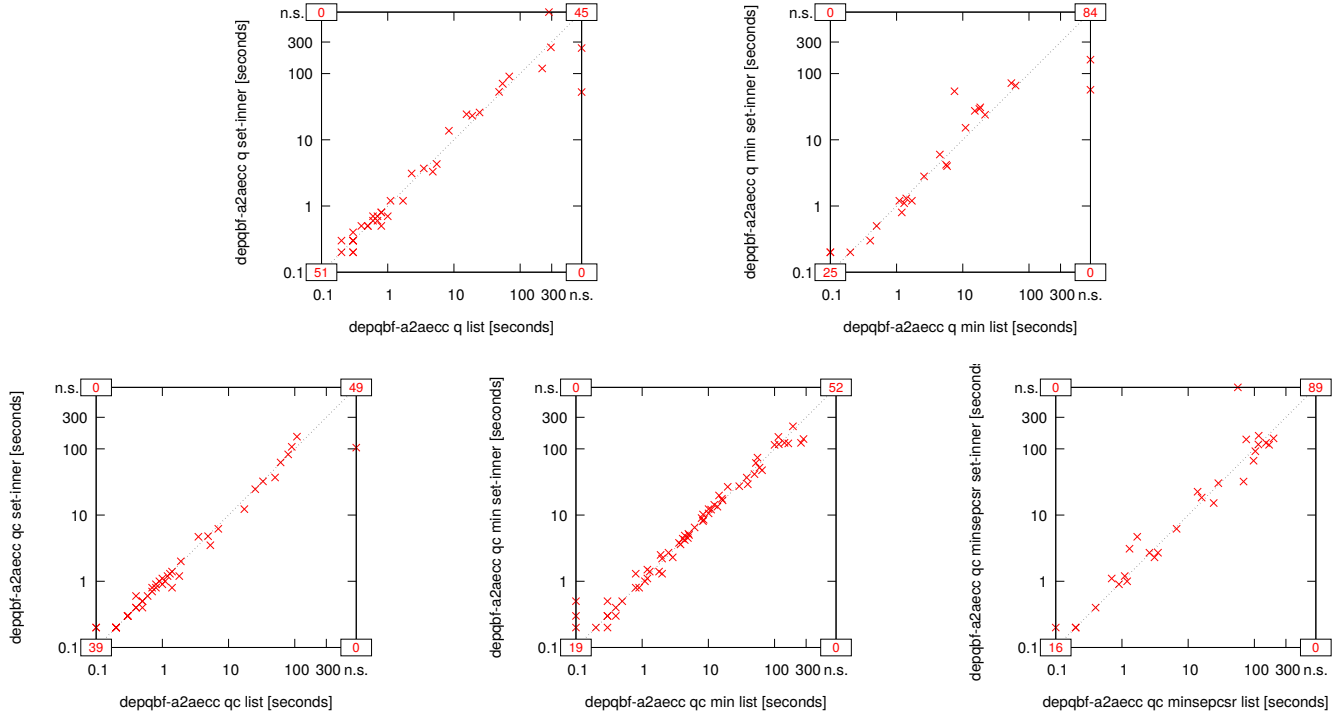


Fig. 805: Suite Kronegger-Pfandler-Pichler ($n = 133$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

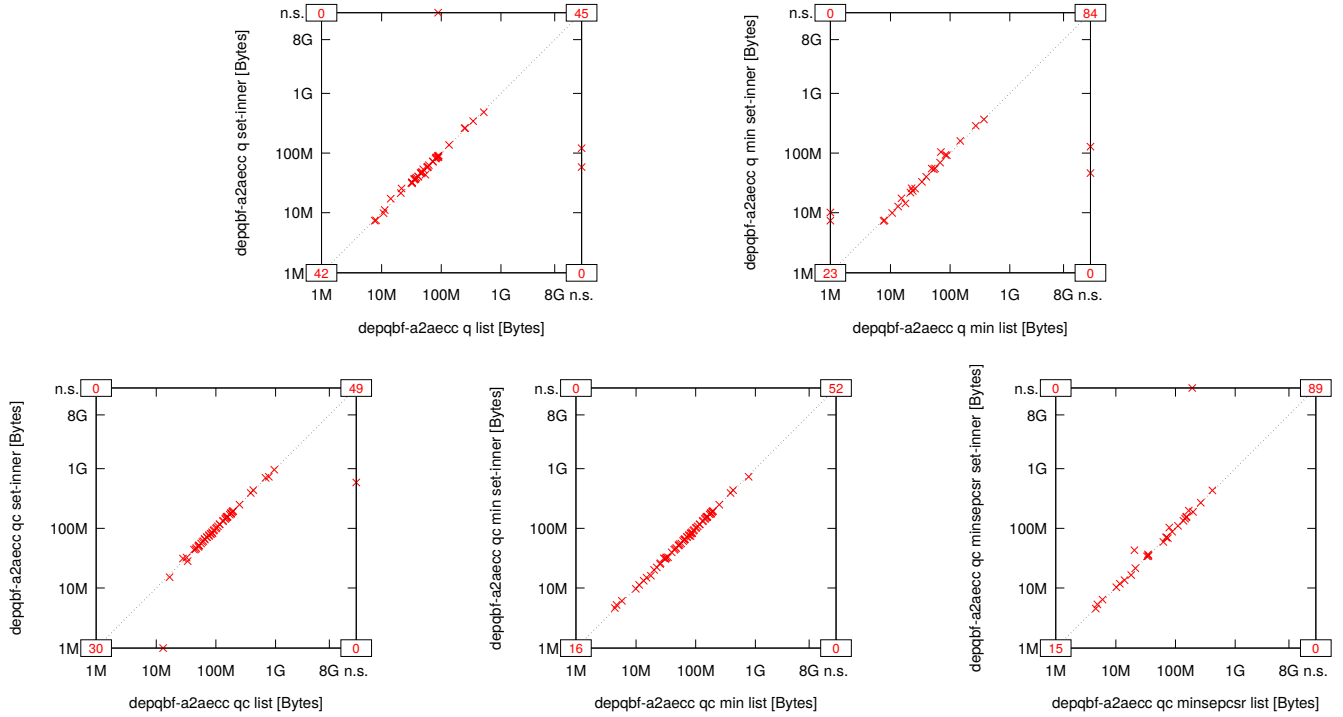


Fig. 806: Suite Kronegger-Pfandler-Pichler ($n = 133$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

22) Lahiri-Seshia ($n = 1$):

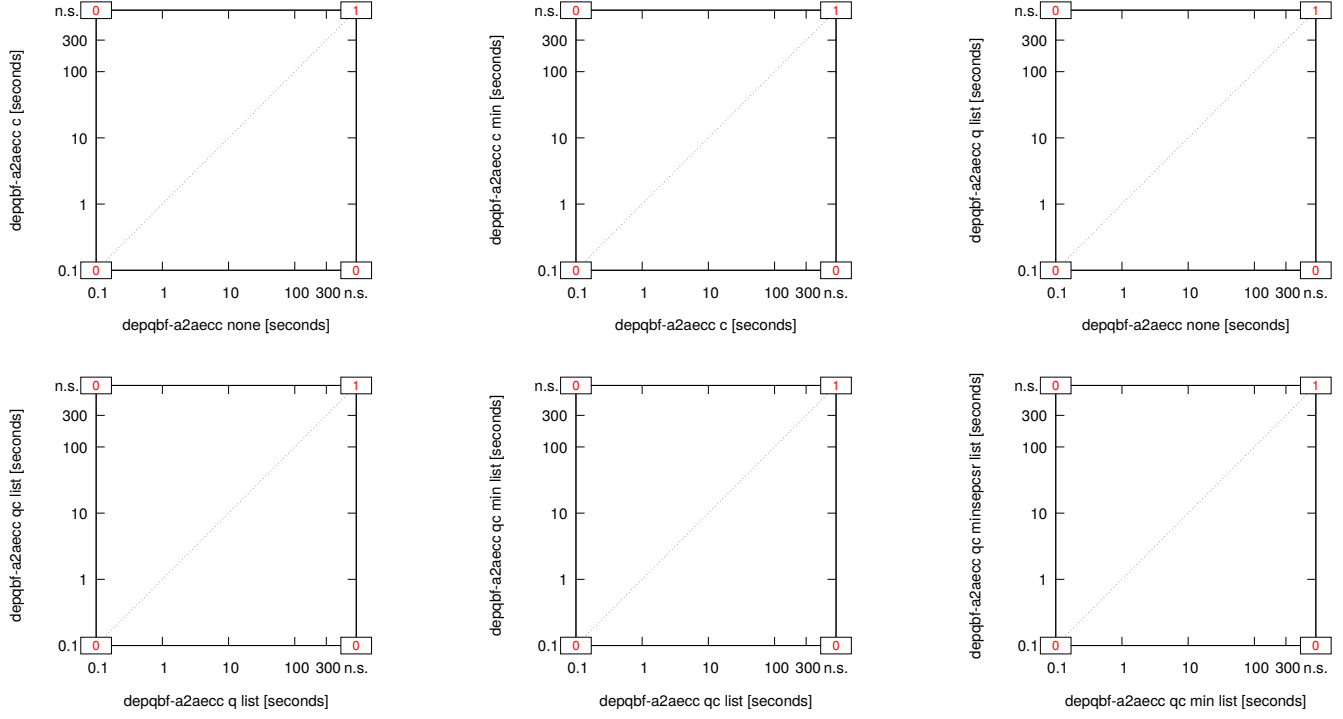


Fig. 807: Suite Lahiri-Seshia ($n = 1$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

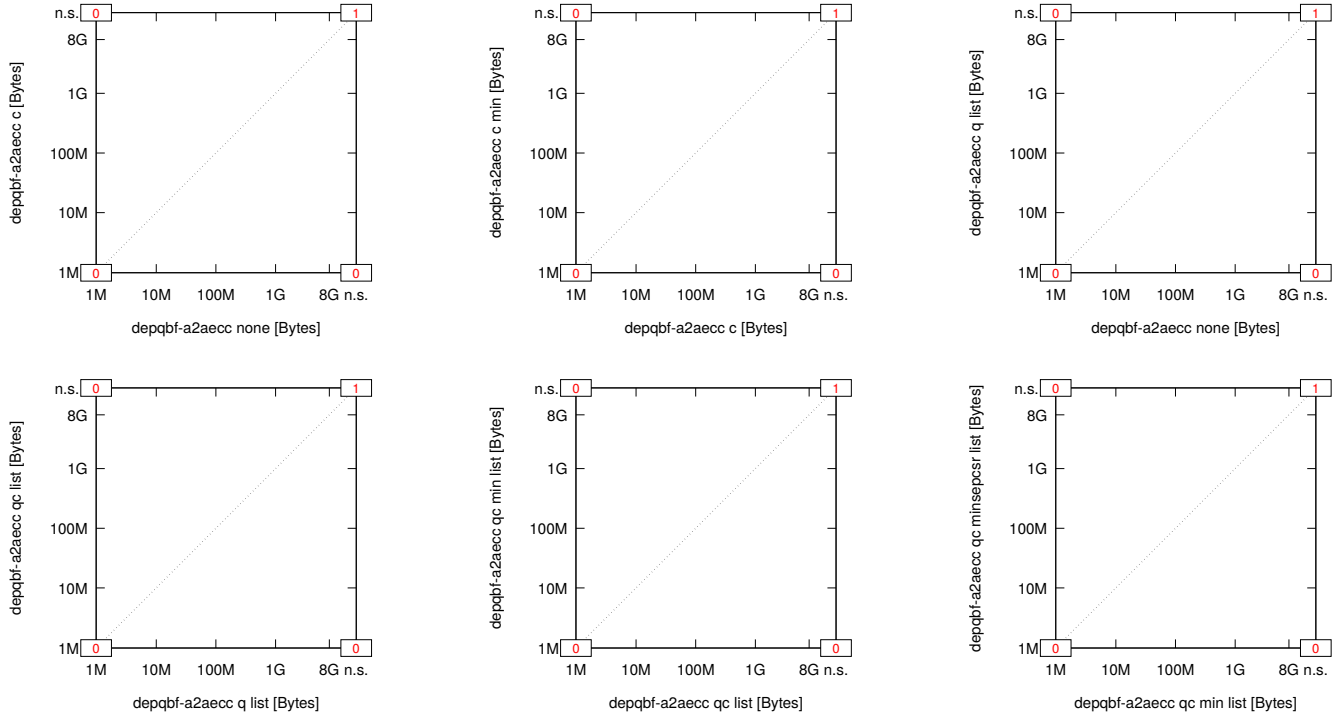


Fig. 808: Suite Lahiri-Seshia ($n = 1$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

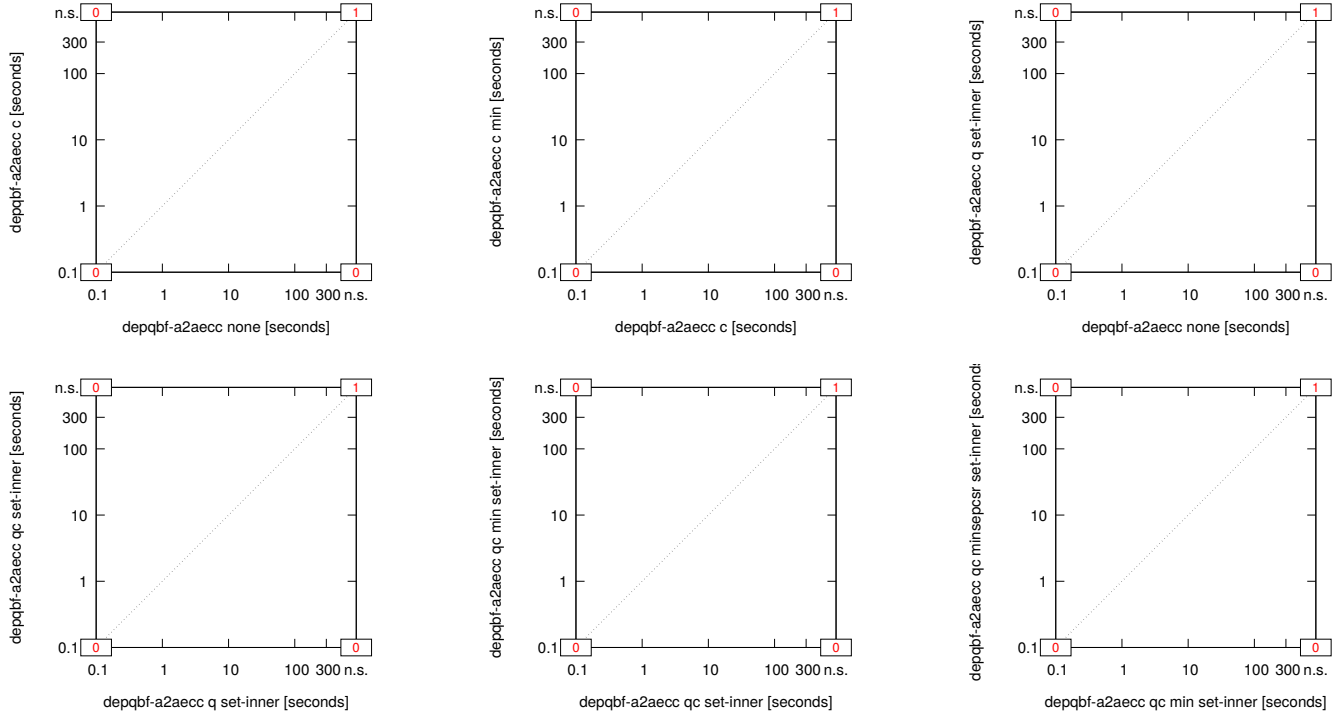


Fig. 809: Suite Lahiri-Seshia ($n = 1$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

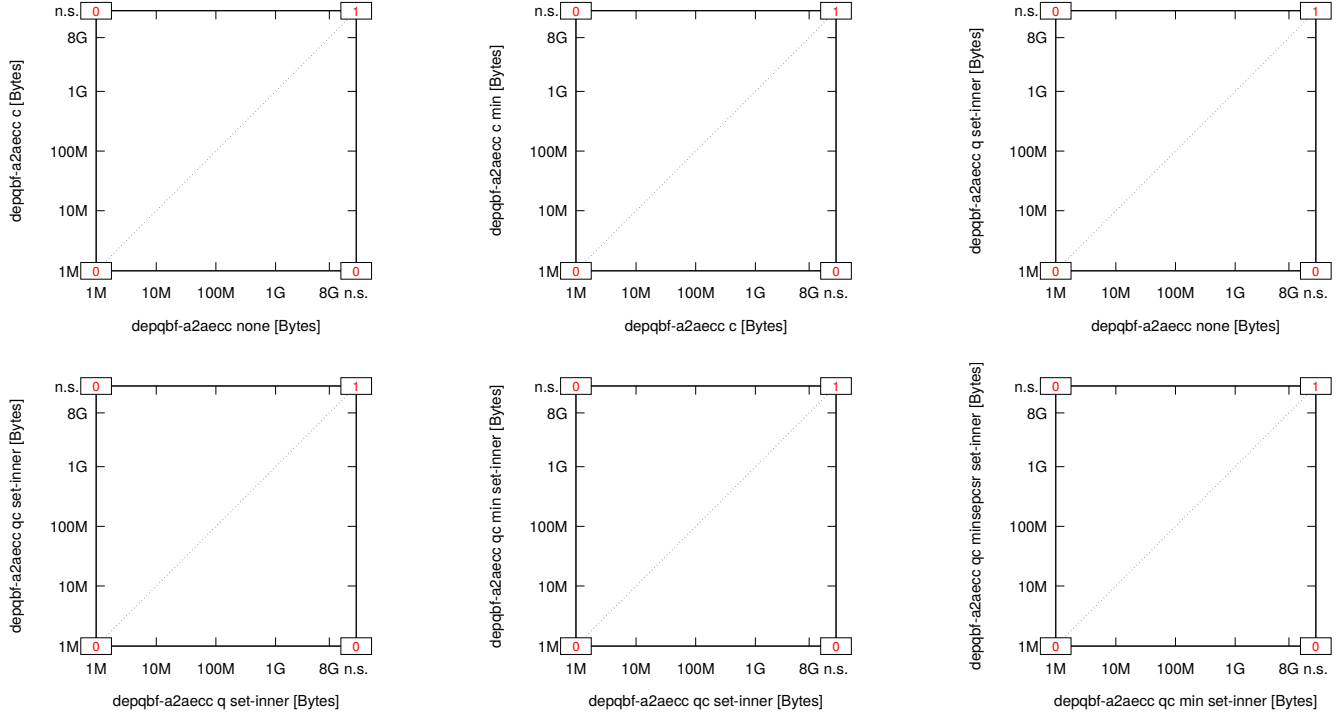


Fig. 810: Suite Lahiri-Seshia ($n = 1$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

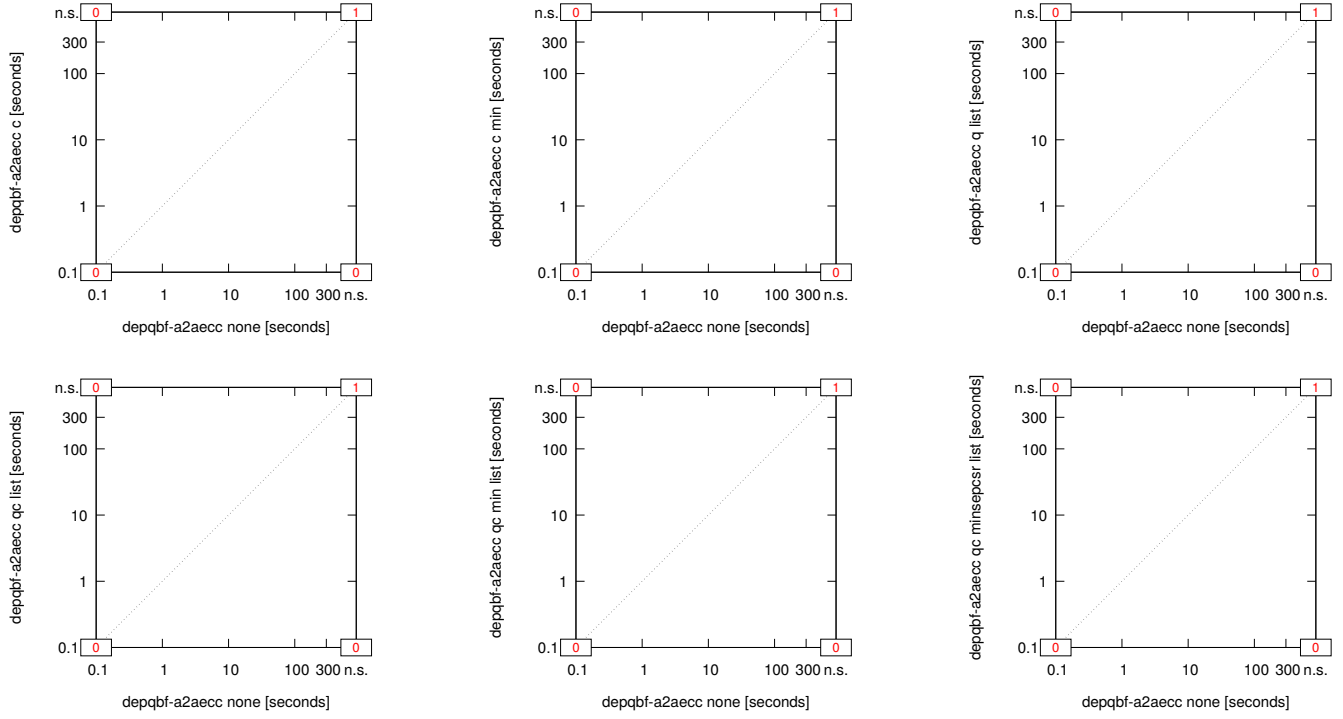


Fig. 811: Suite Lahiri-Seshia ($n = 1$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

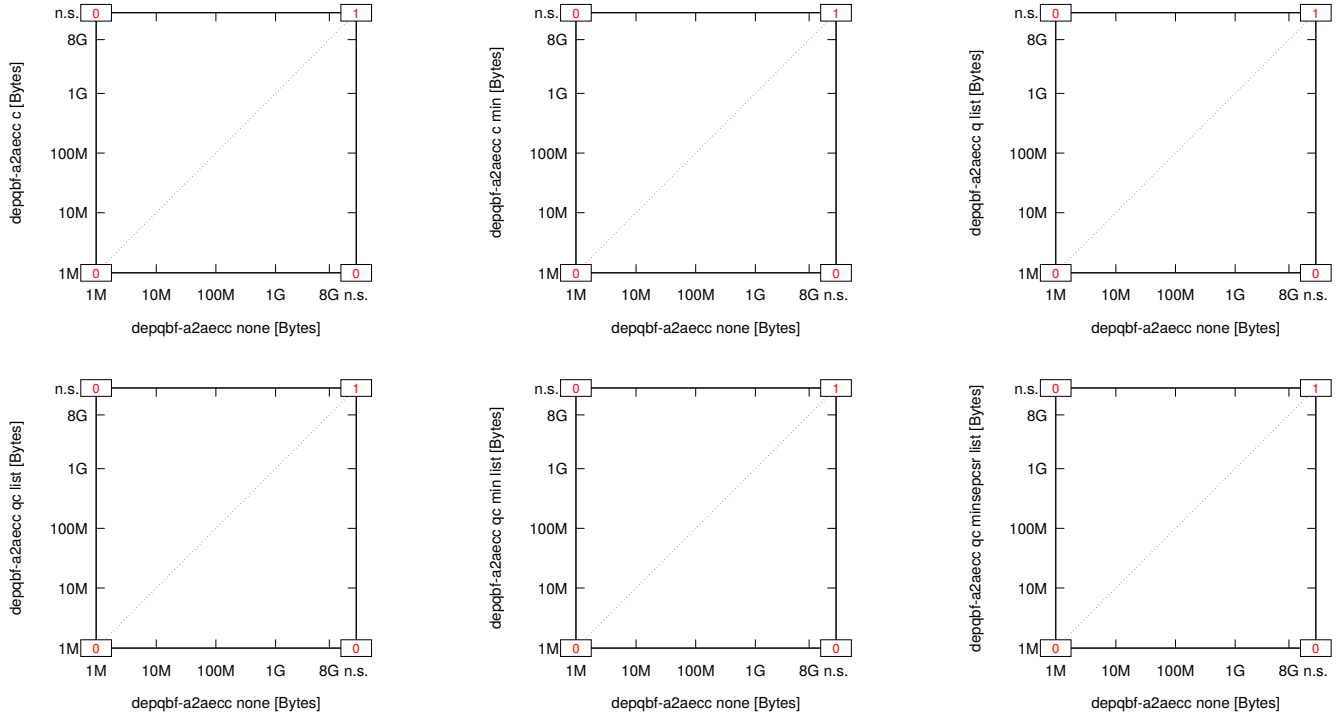


Fig. 812: Suite Lahiri-Seshia ($n = 1$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

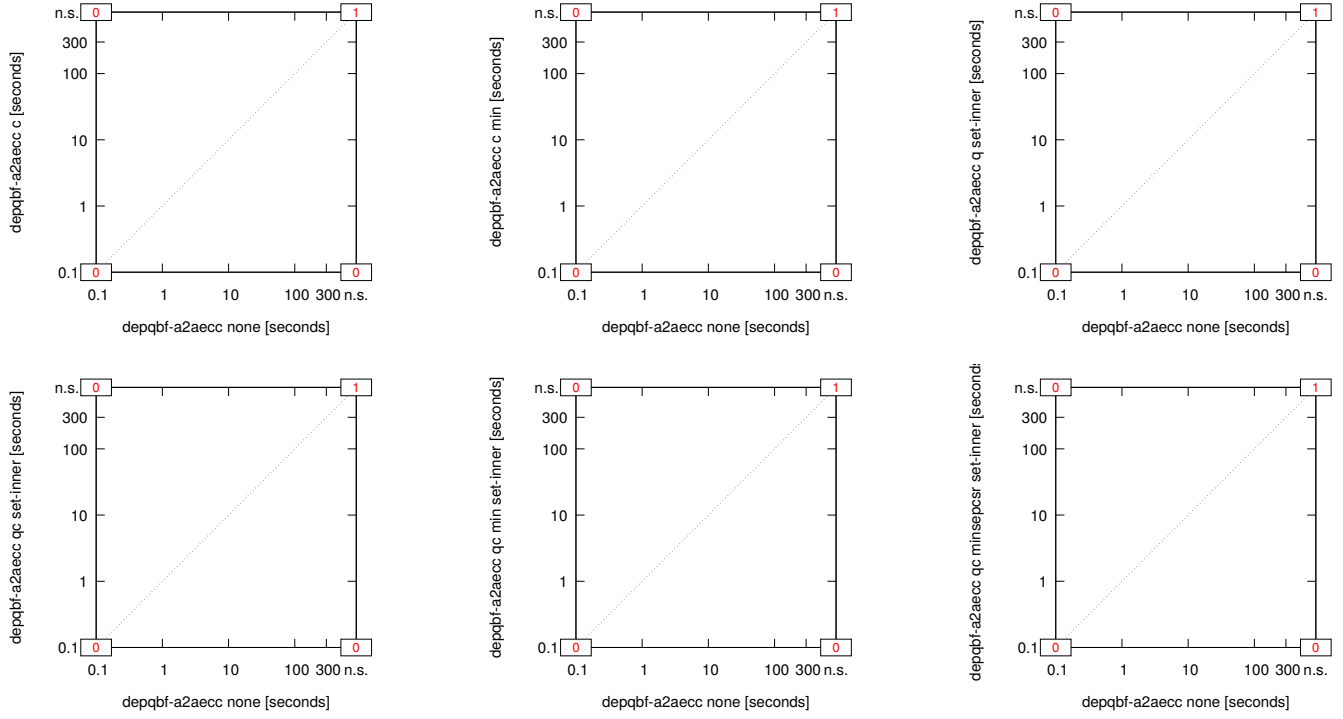


Fig. 813: Suite Lahiri-Seshia ($n = 1$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

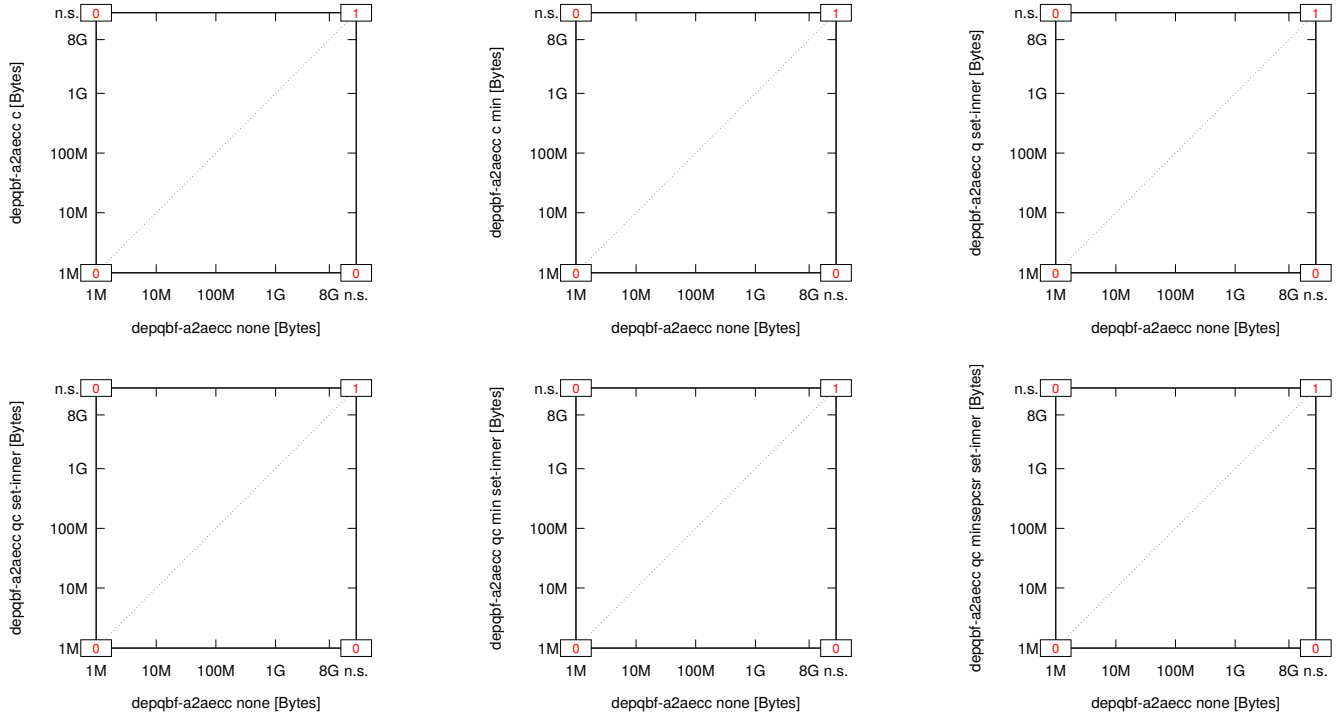


Fig. 814: Suite Lahiri-Seshia ($n = 1$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

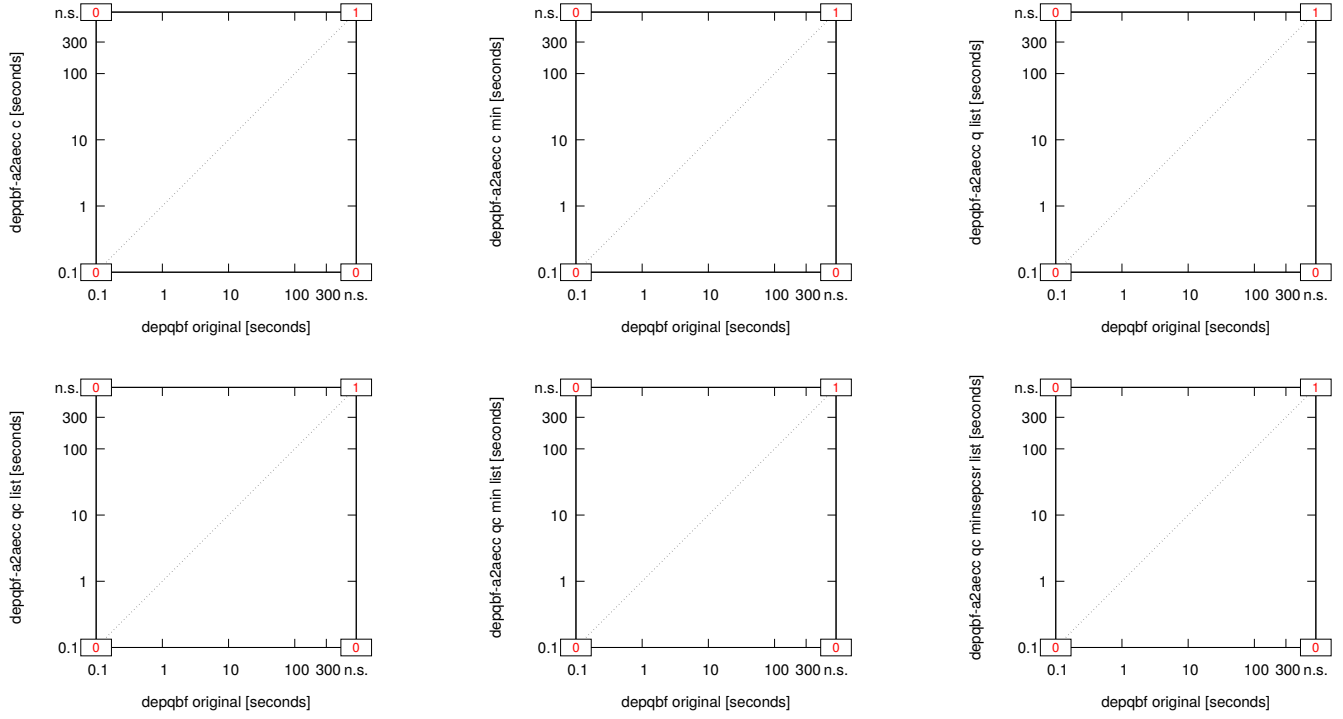


Fig. 815: Suite Lahiri-Seshia ($n = 1$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

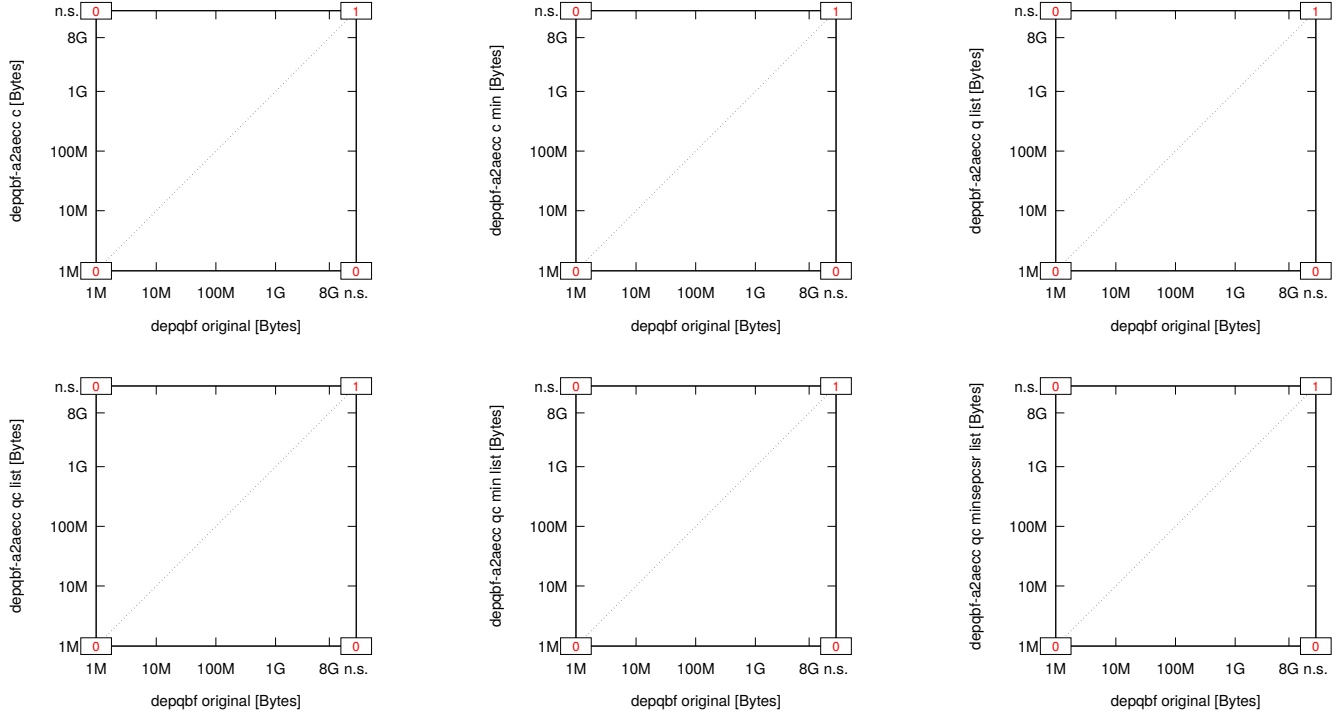


Fig. 816: Suite Lahiri-Seshia ($n = 1$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

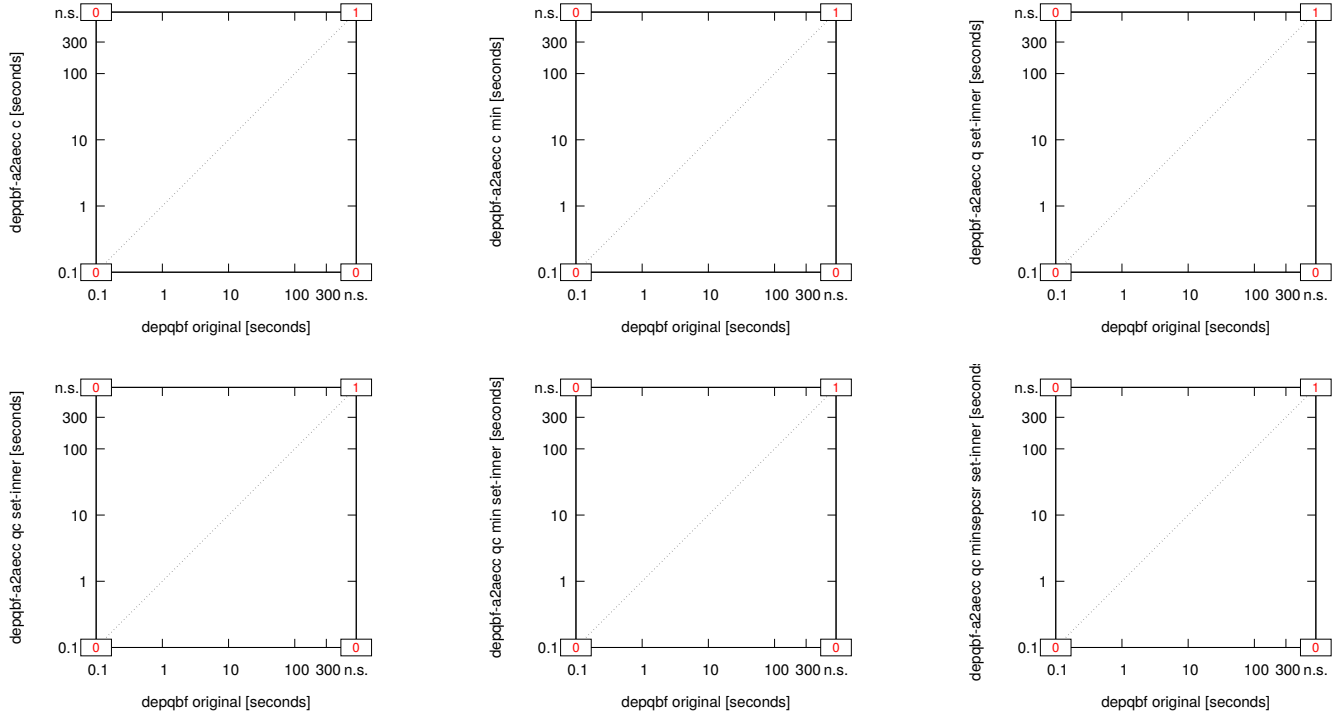


Fig. 817: Suite Lahiri-Seshia ($n = 1$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

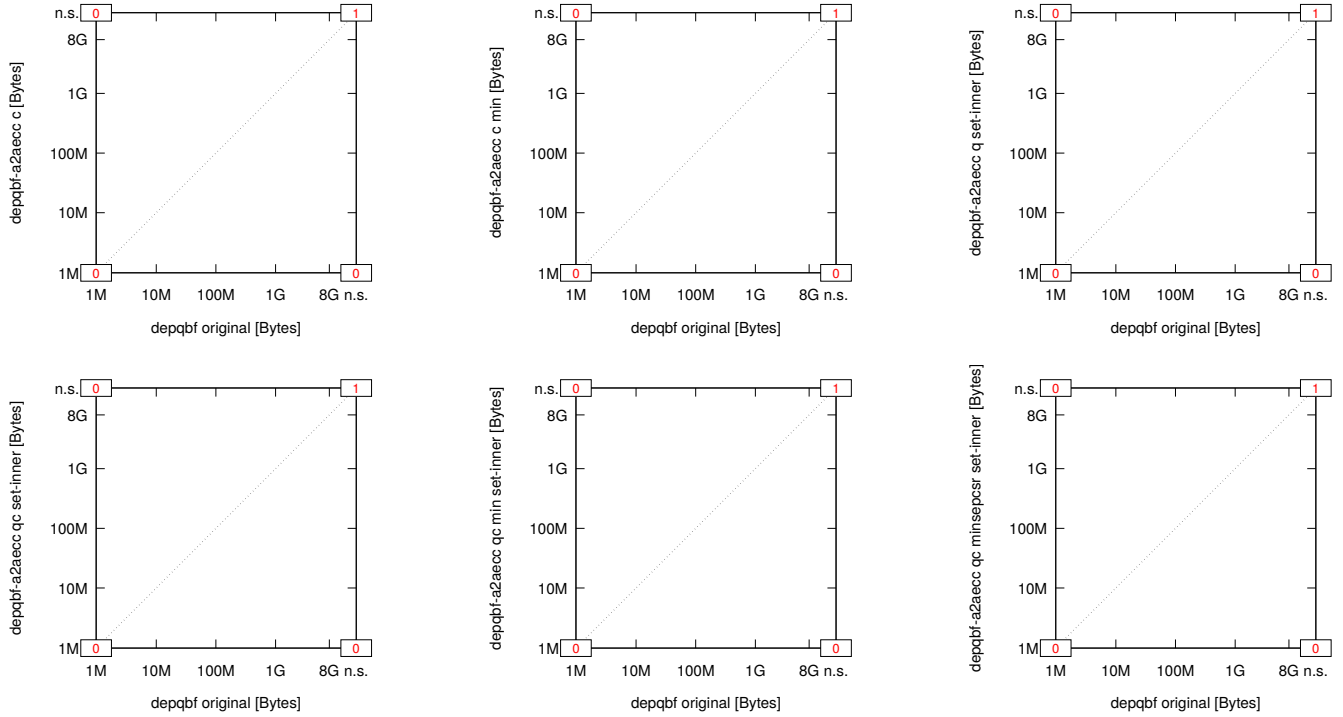


Fig. 818: Suite Lahiri-Seshia ($n = 1$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

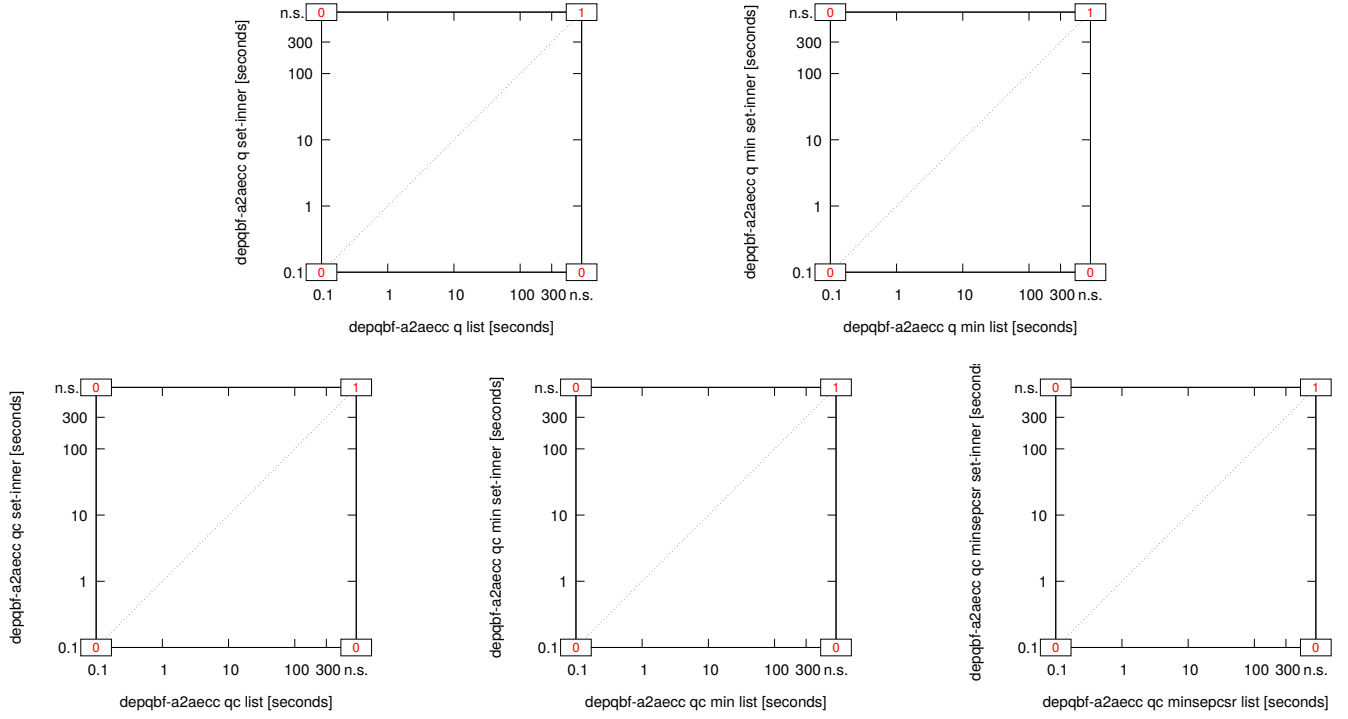


Fig. 819: Suite Lahiri-Seshia ($n = 1$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

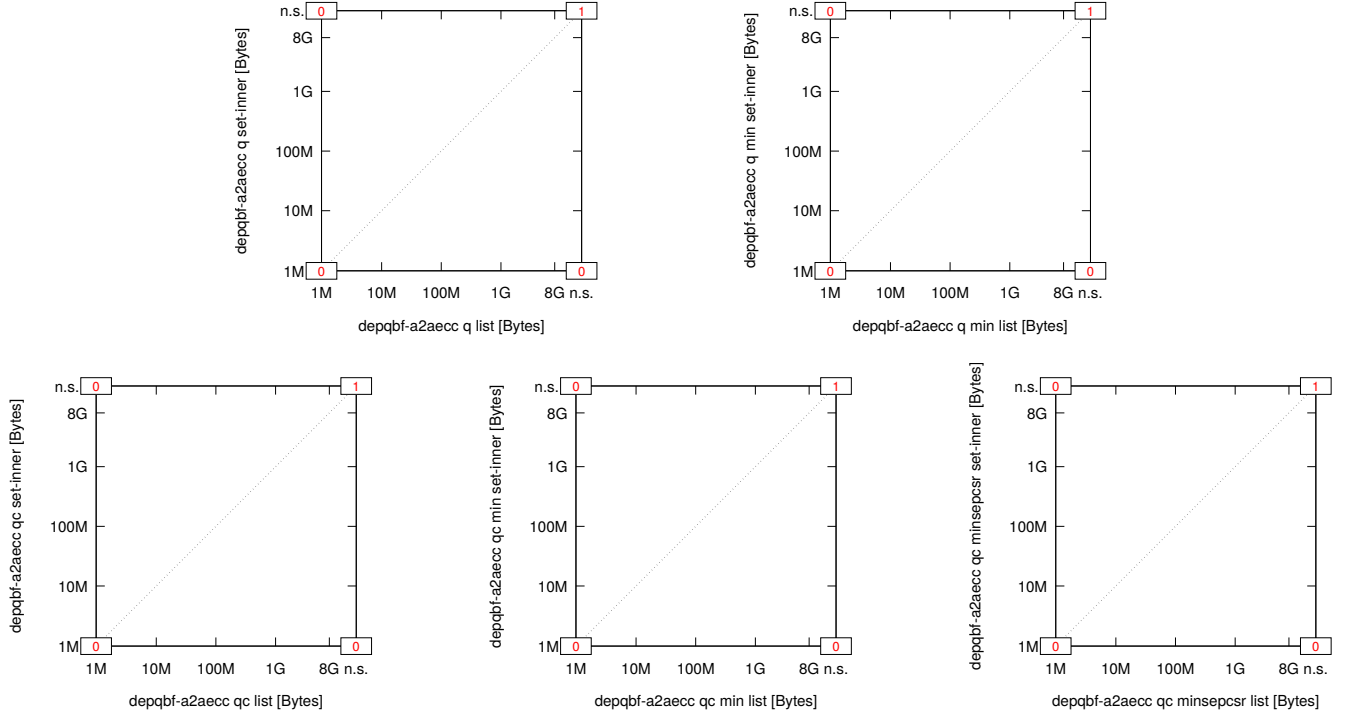


Fig. 820: Suite Lahiri-Seshia ($n = 1$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

23) Lee-Jiang ($n = 2$):

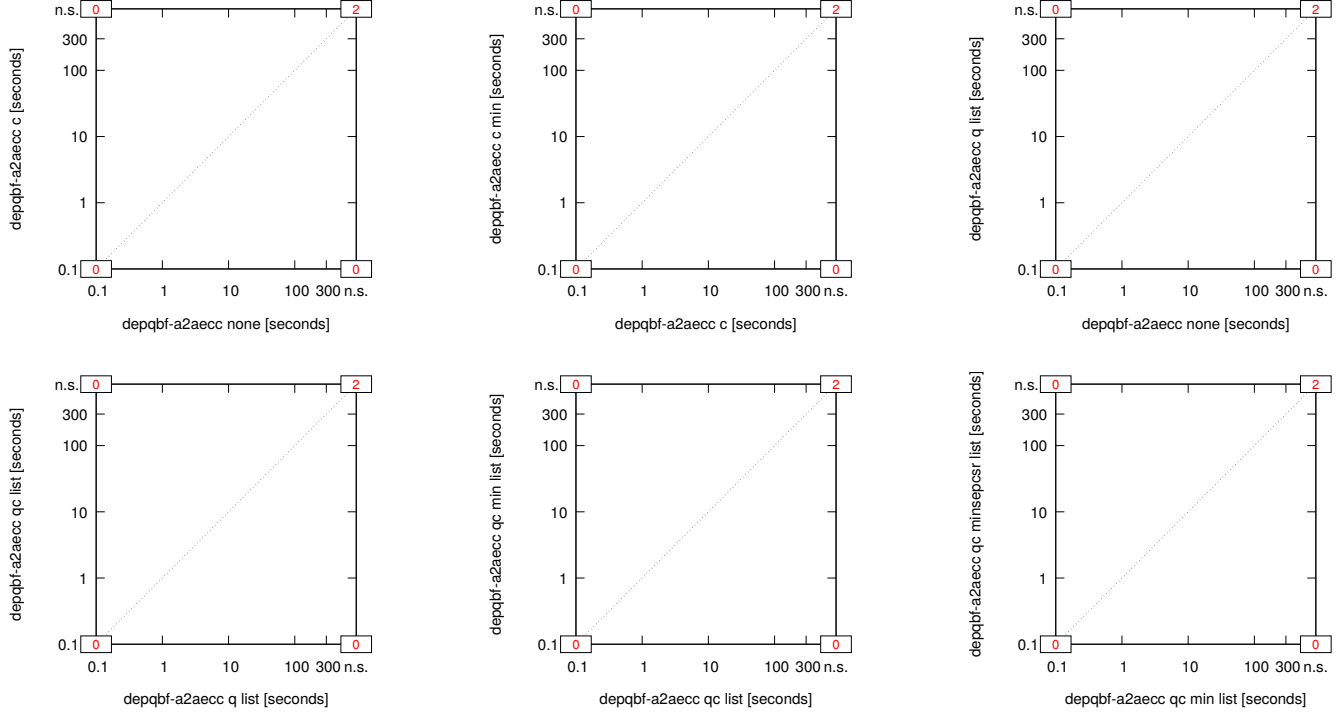


Fig. 821: Suite Lee-Jiang ($n = 2$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

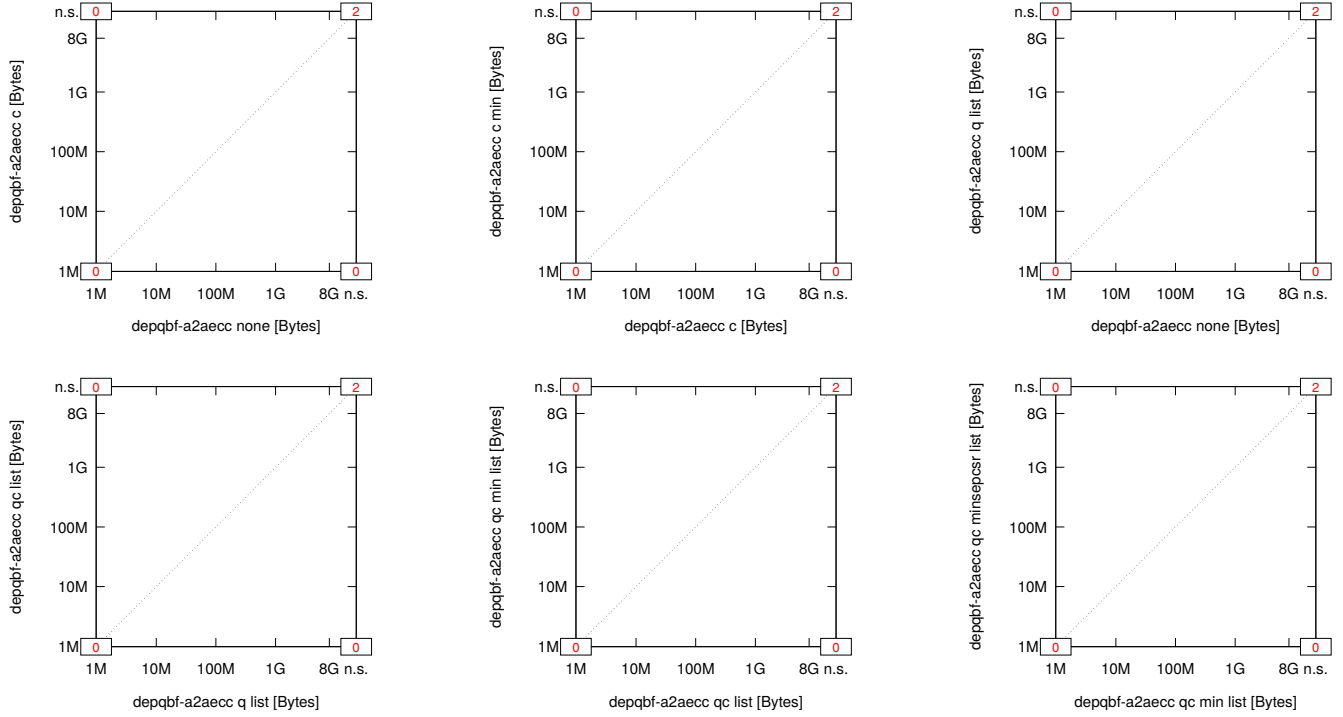


Fig. 822: Suite Lee-Jiang ($n = 2$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

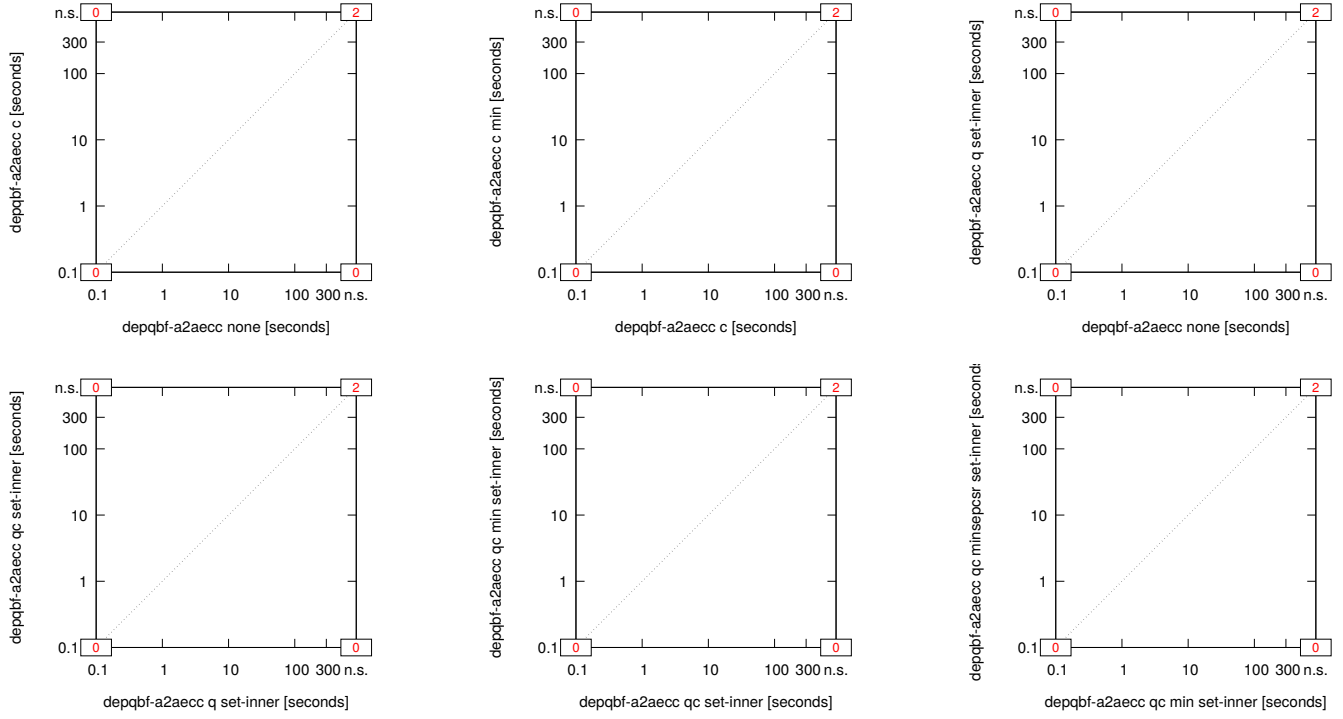


Fig. 823: Suite Lee-Jiang ($n = 2$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

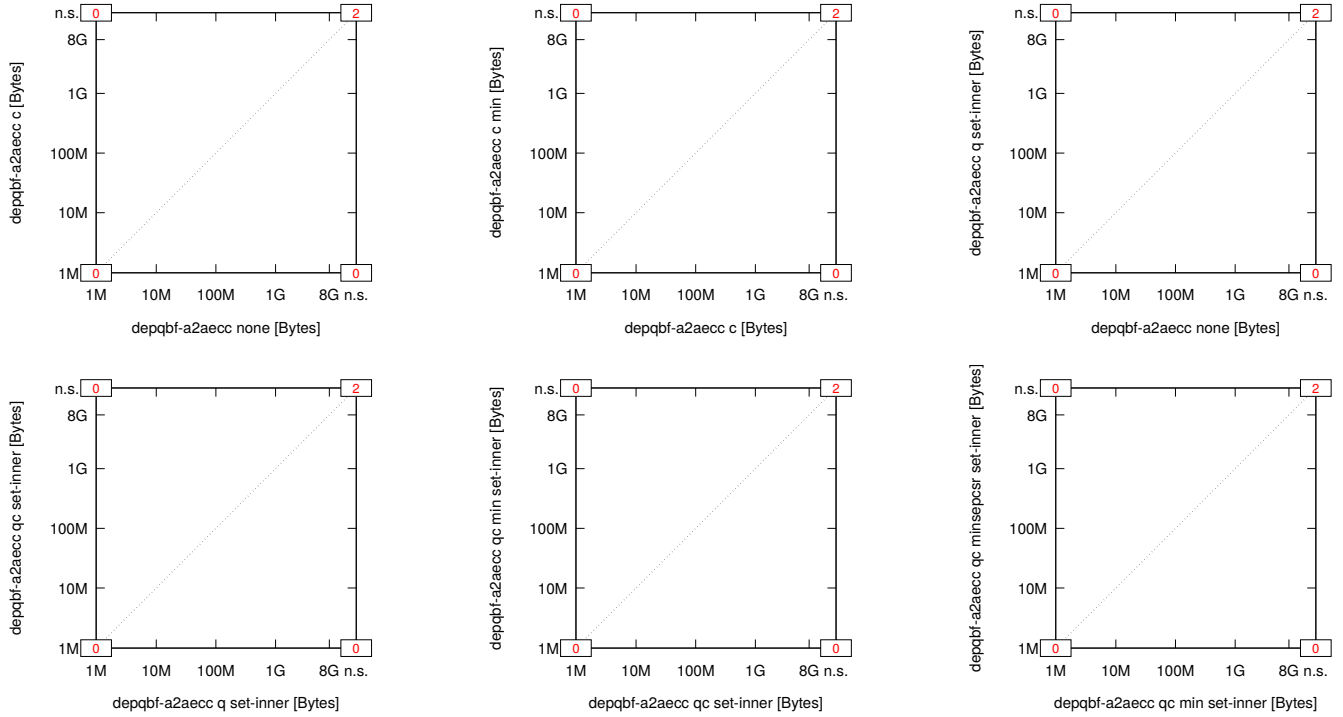


Fig. 824: Suite Lee-Jiang ($n = 2$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

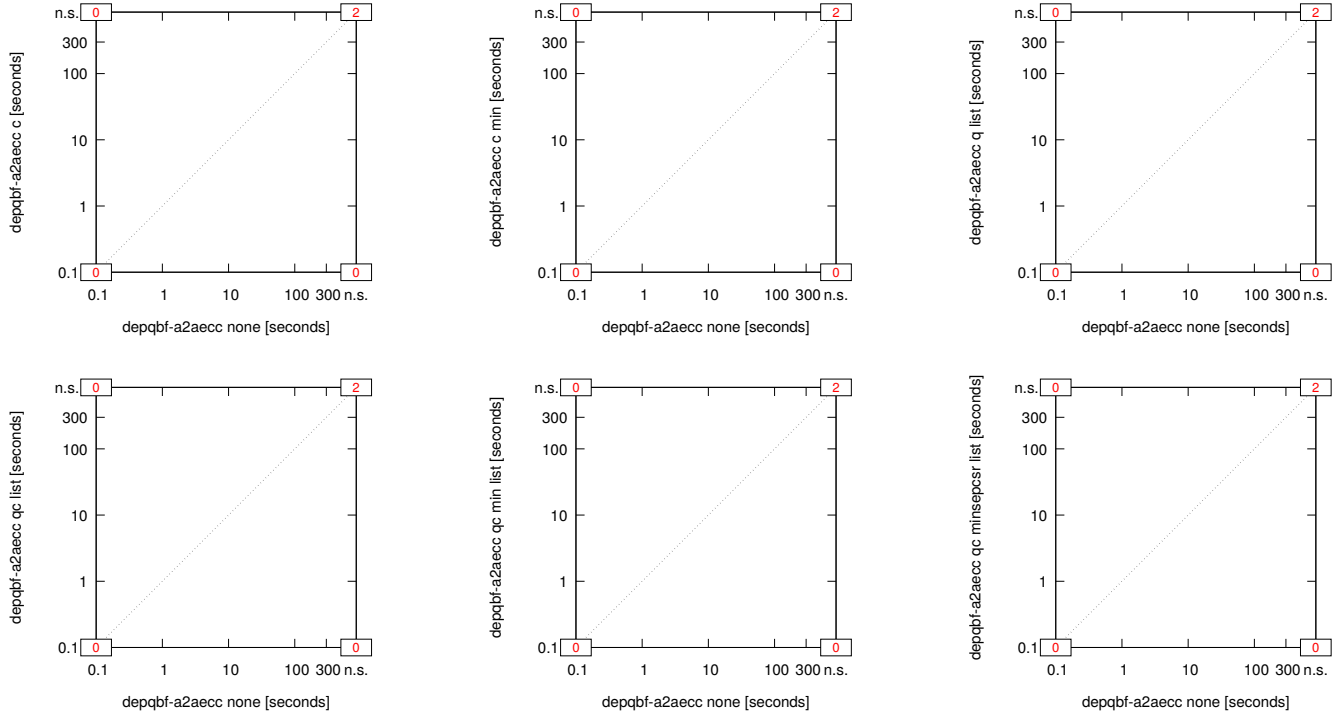


Fig. 825: Suite Lee-Jiang ($n = 2$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

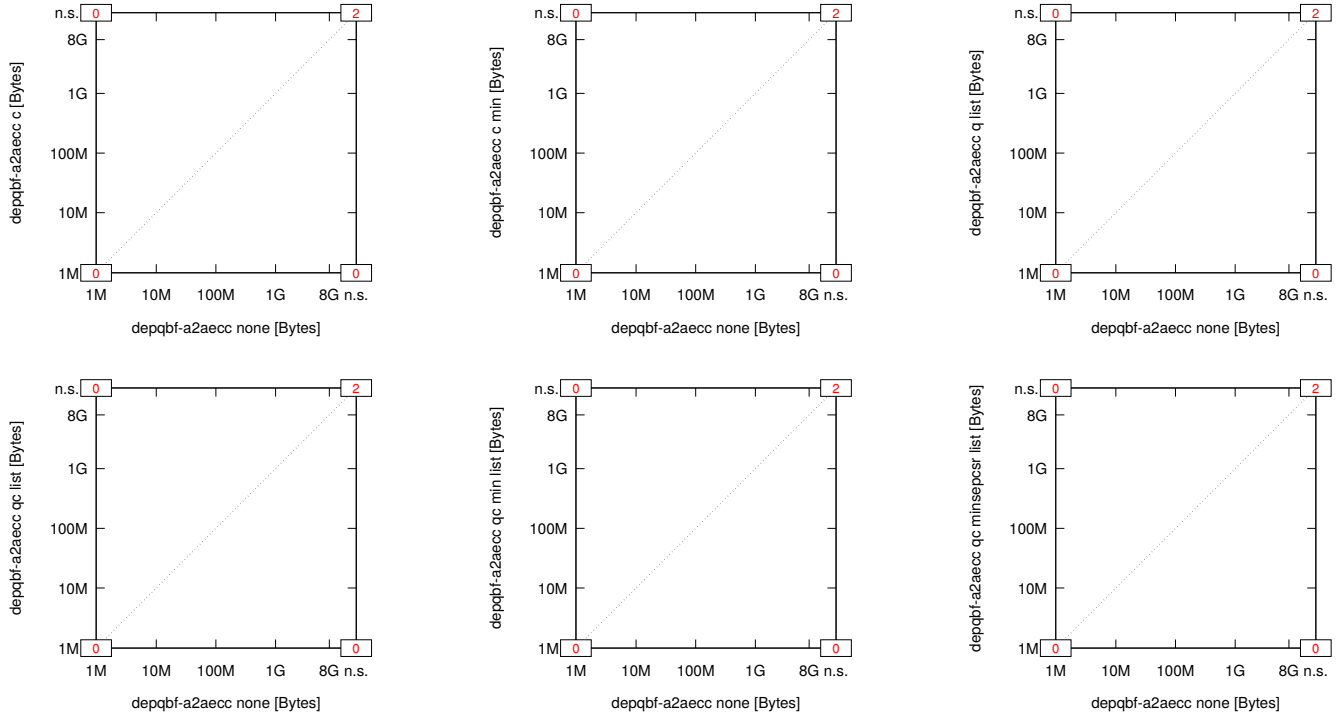


Fig. 826: Suite Lee-Jiang ($n = 2$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

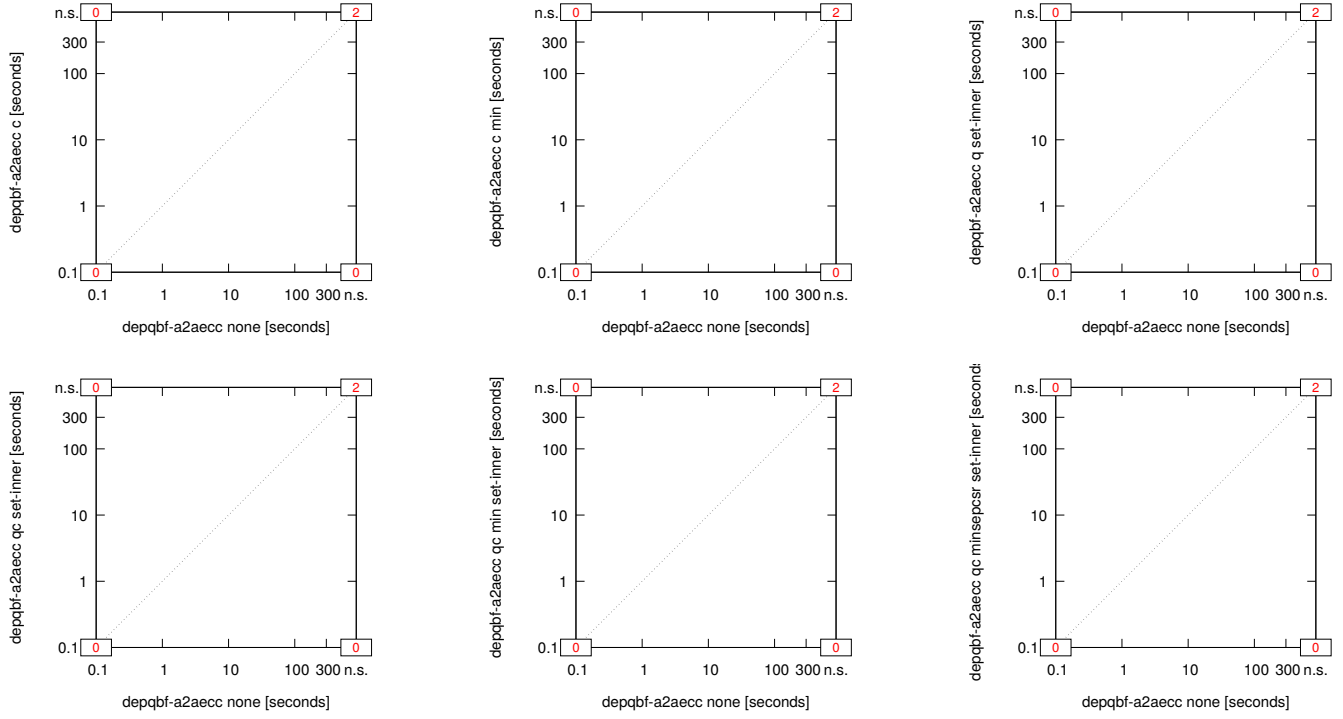


Fig. 827: Suite Lee-Jiang ($n = 2$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

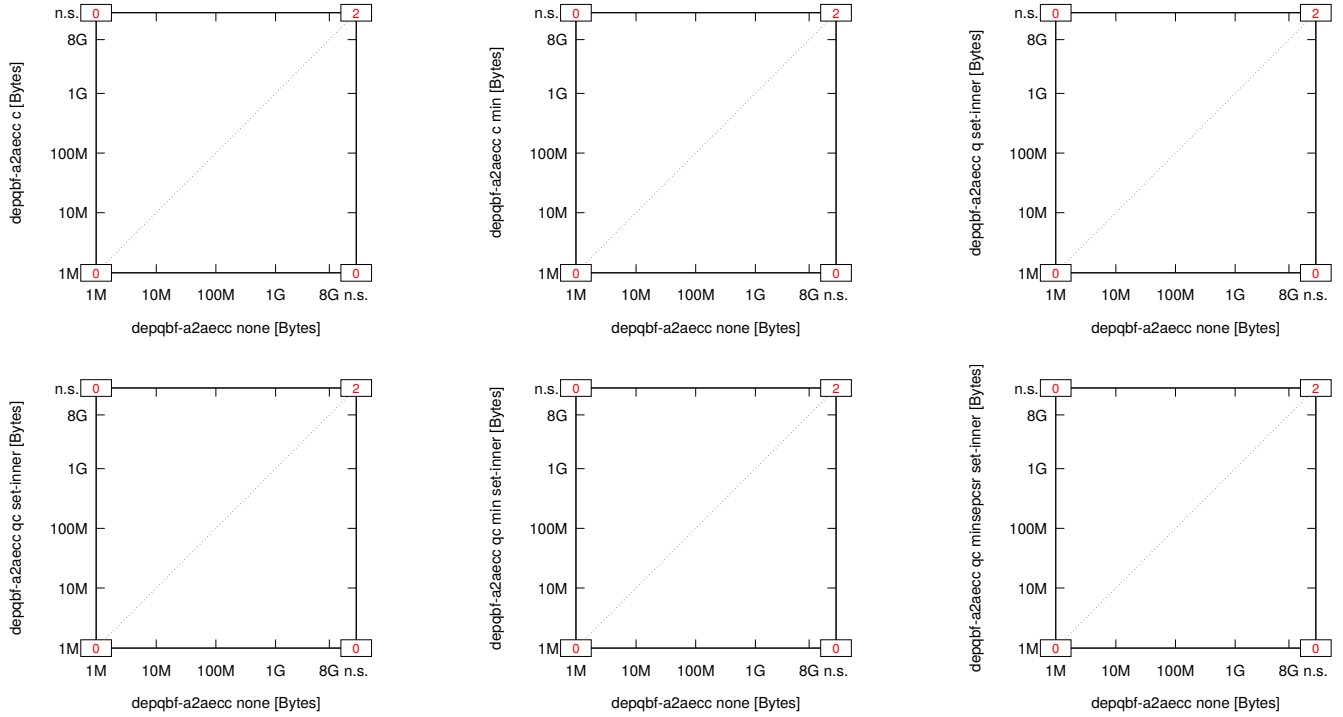


Fig. 828: Suite Lee-Jiang ($n = 2$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

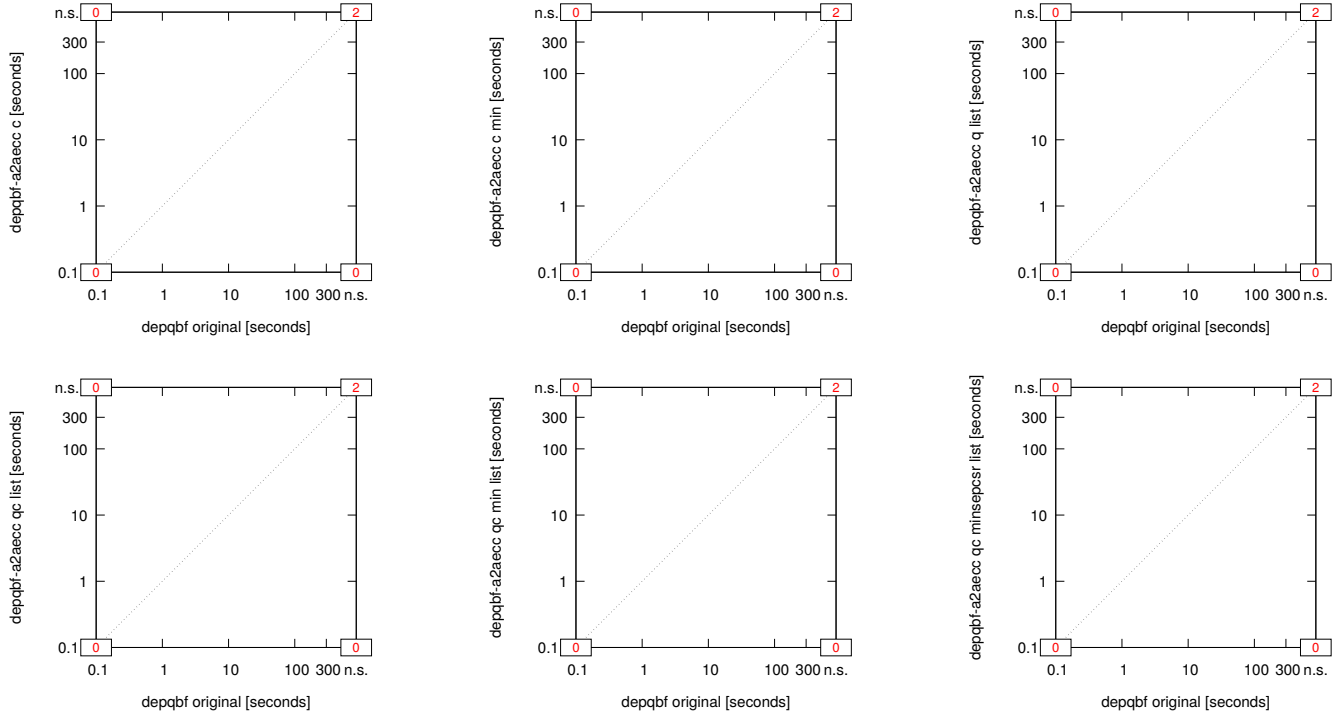


Fig. 829: Suite Lee-Jiang ($n = 2$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

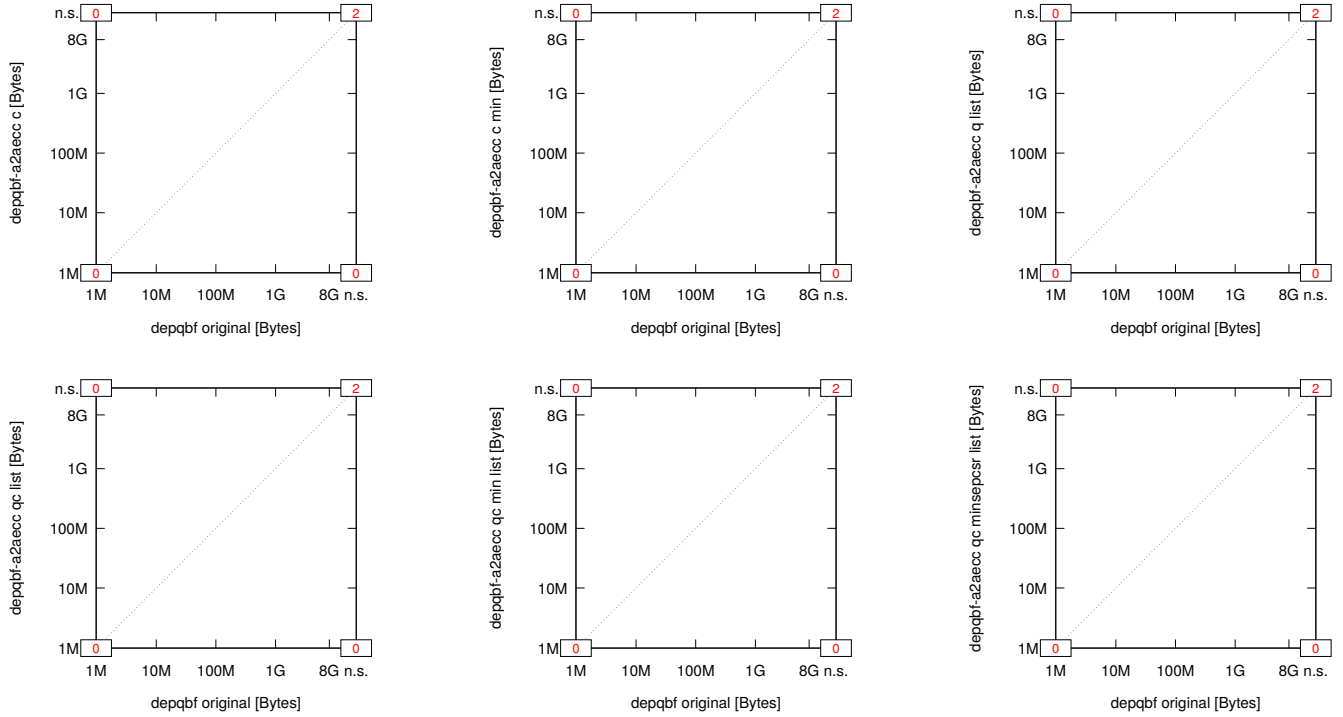


Fig. 830: Suite Lee-Jiang ($n = 2$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

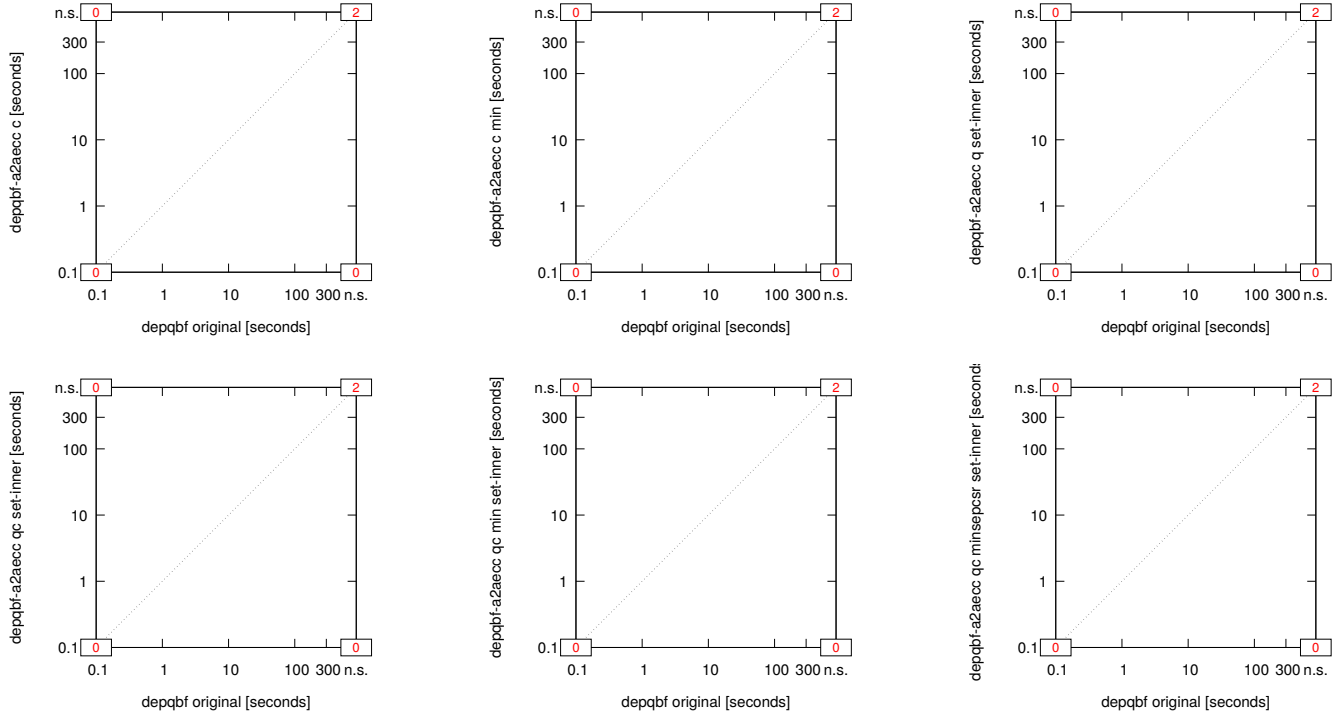


Fig. 831: Suite Lee-Jiang ($n = 2$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

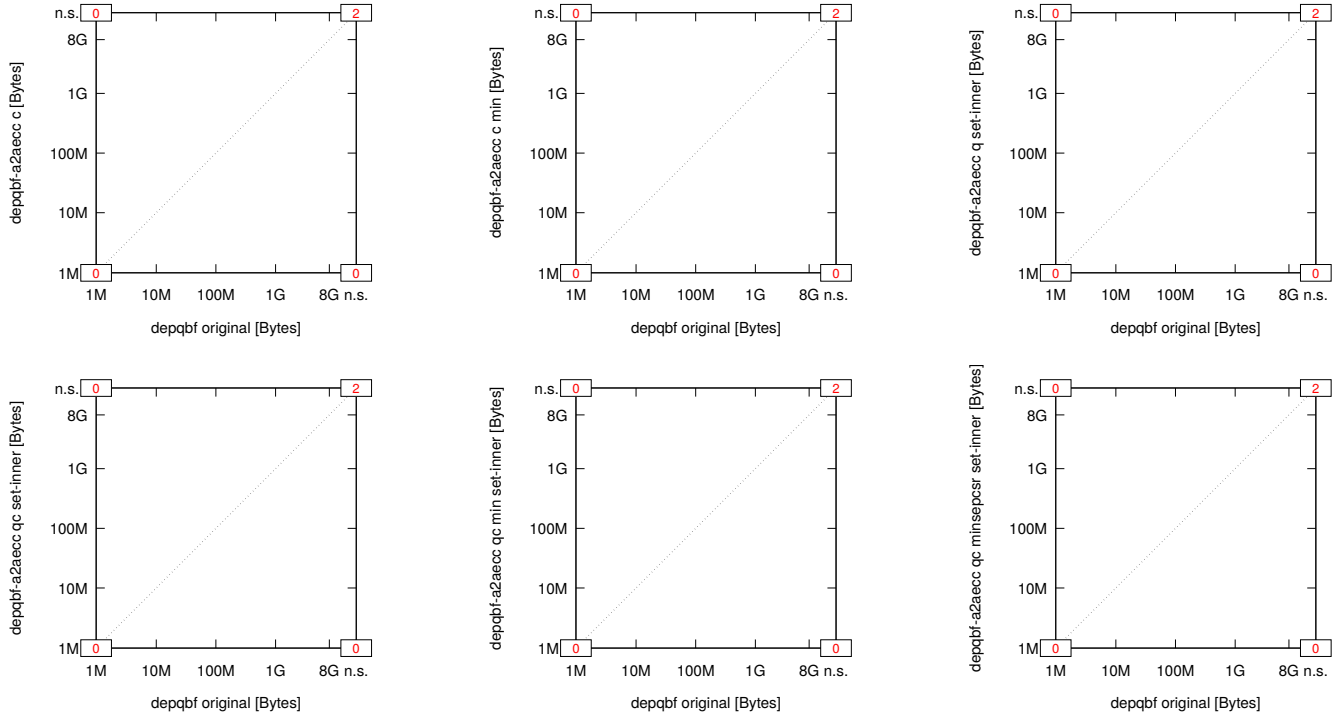


Fig. 832: Suite Lee-Jiang ($n = 2$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

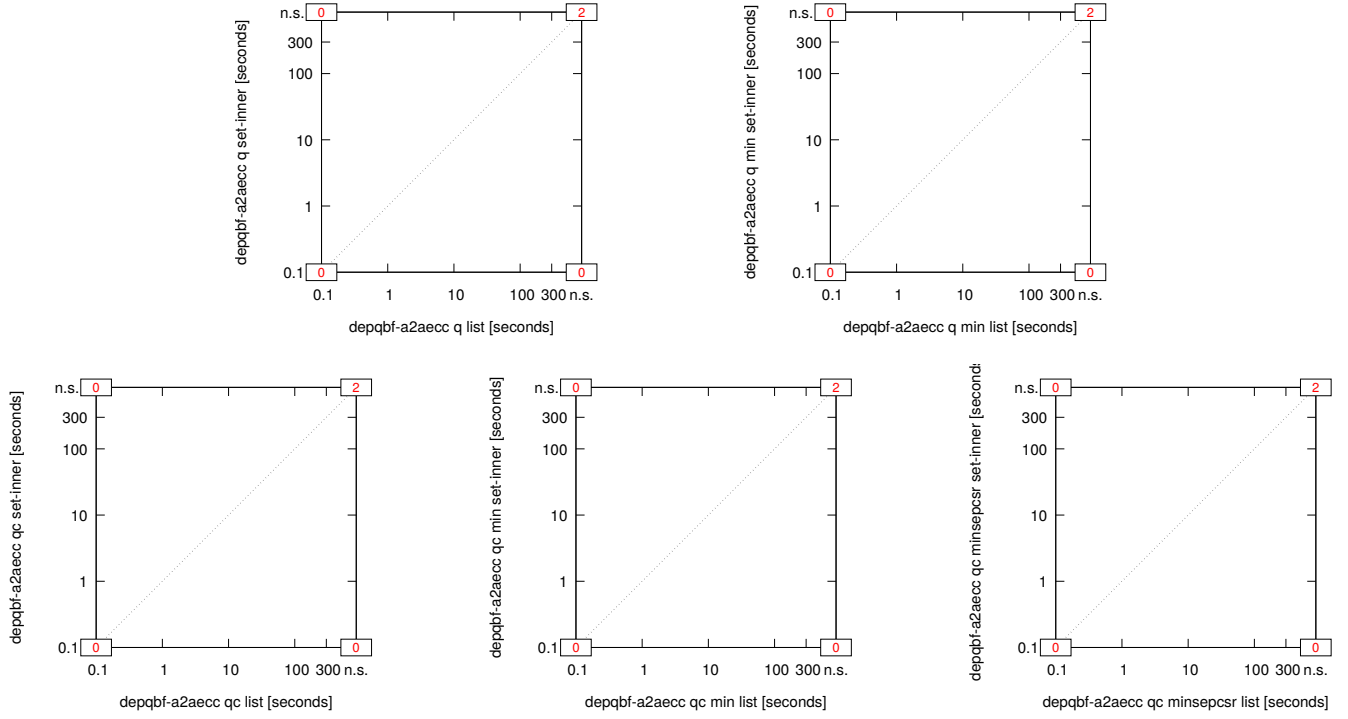


Fig. 833: Suite Lee-Jiang ($n = 2$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

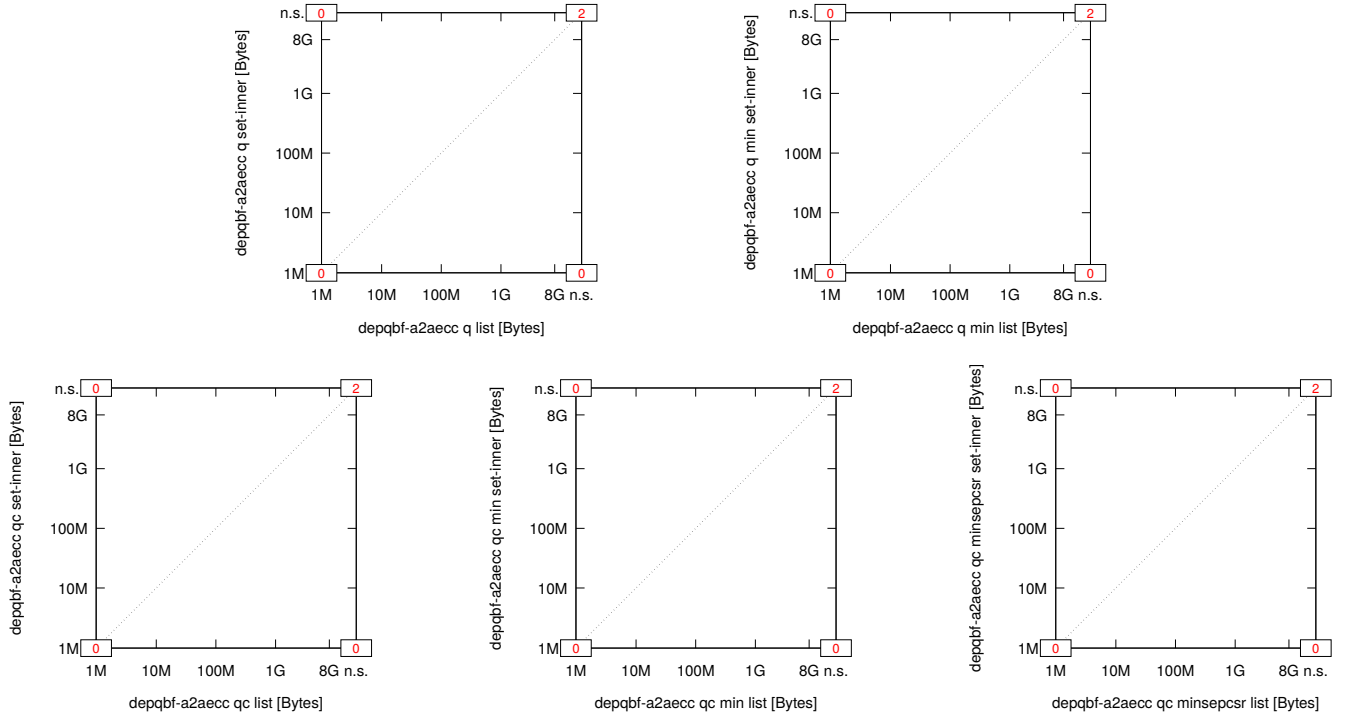


Fig. 834: Suite Lee-Jiang ($n = 2$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

24) *Letombe* ($n = 85$):

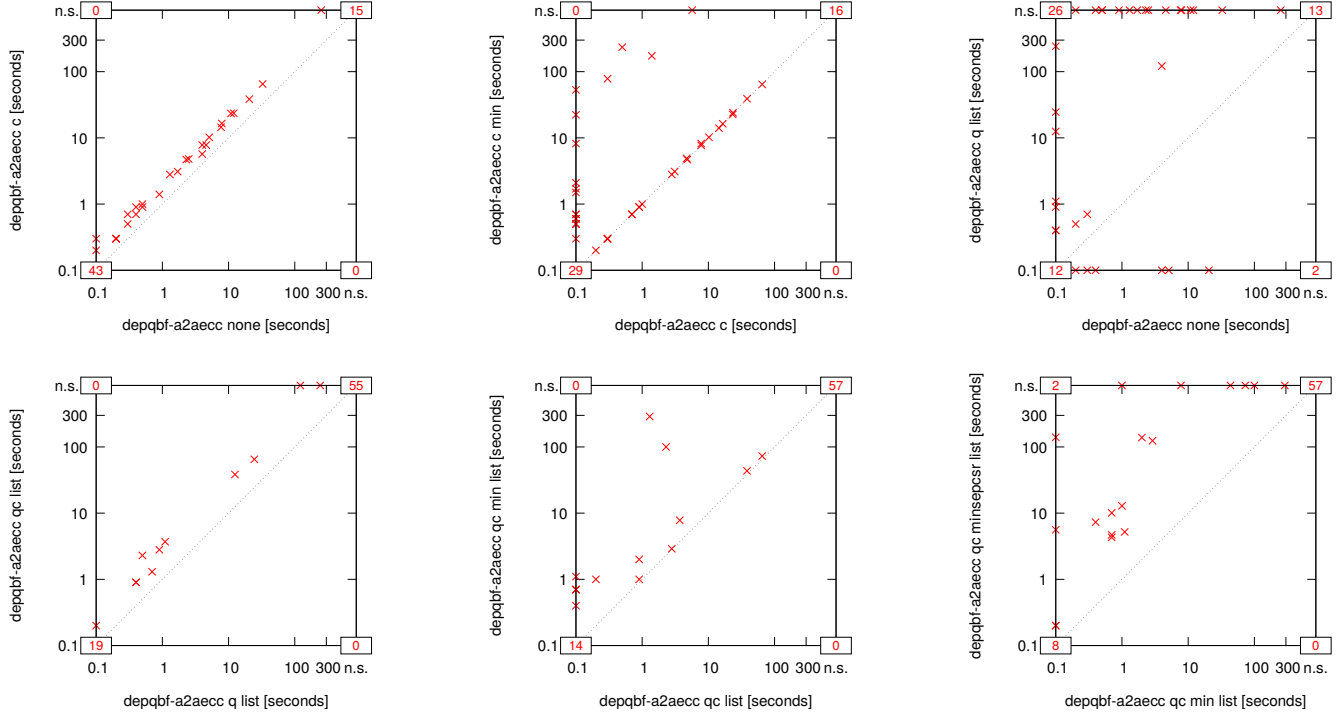


Fig. 835: Suite Letombe ($n = 85$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

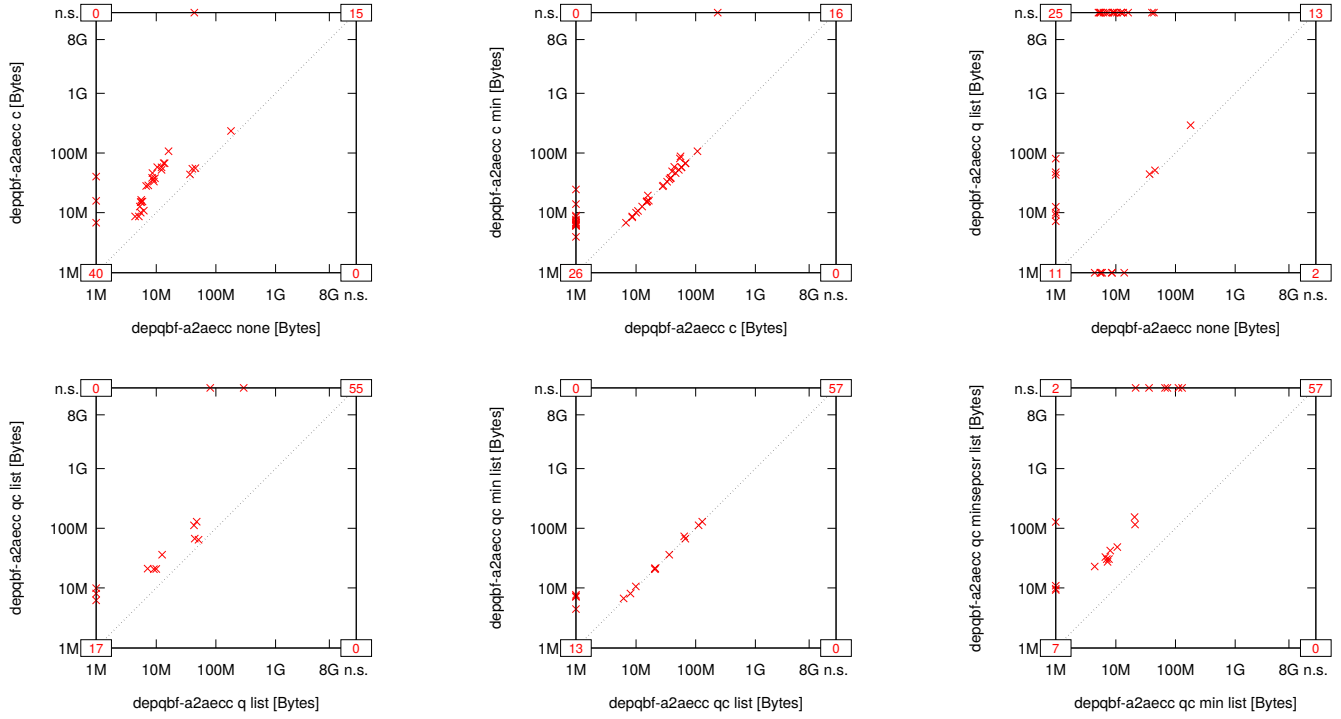


Fig. 836: Suite Letombe ($n = 85$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

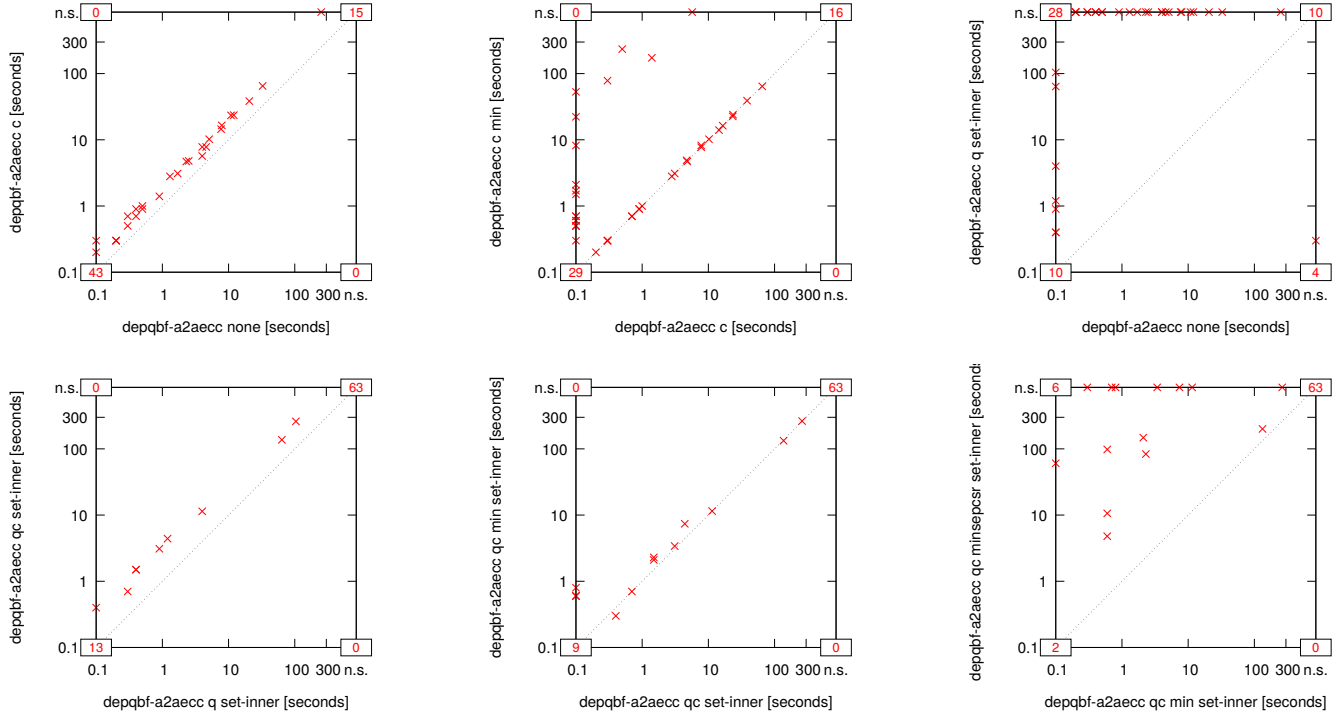


Fig. 837: Suite Letombe ($n = 85$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

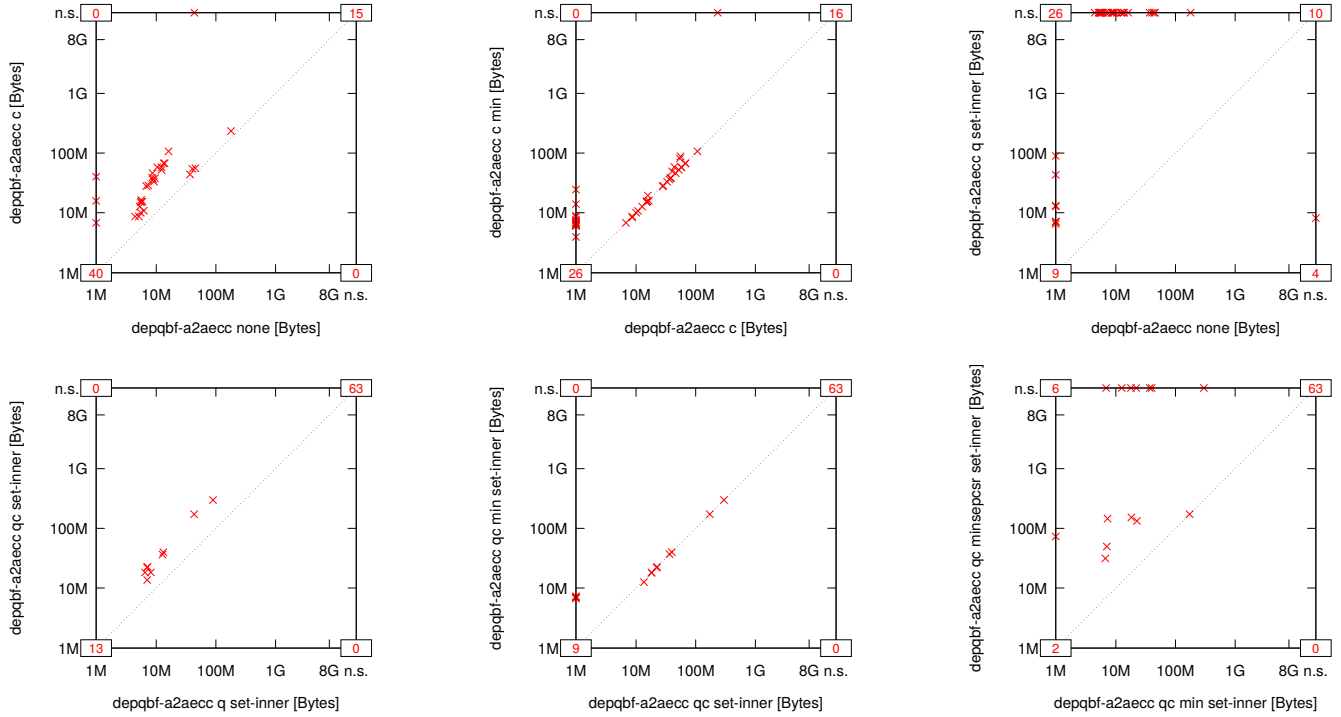


Fig. 838: Suite Letombe ($n = 85$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

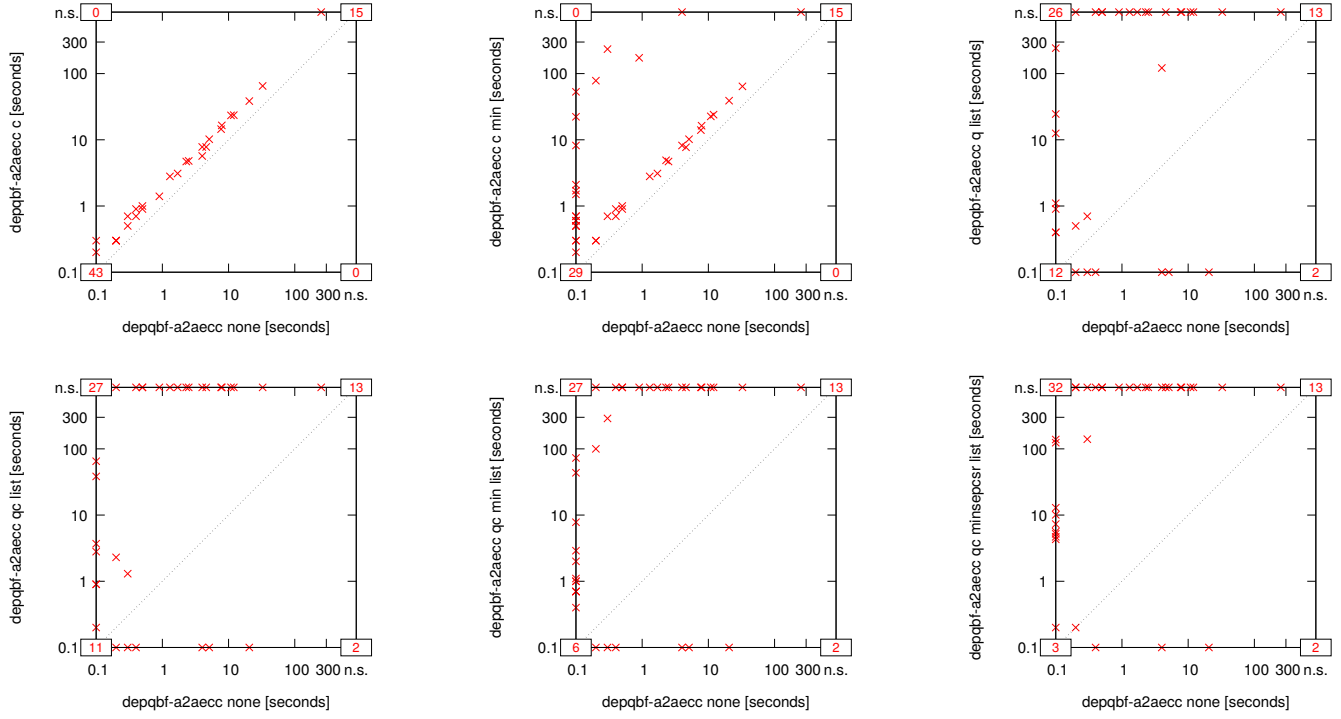


Fig. 839: Suite Letombe ($n = 85$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

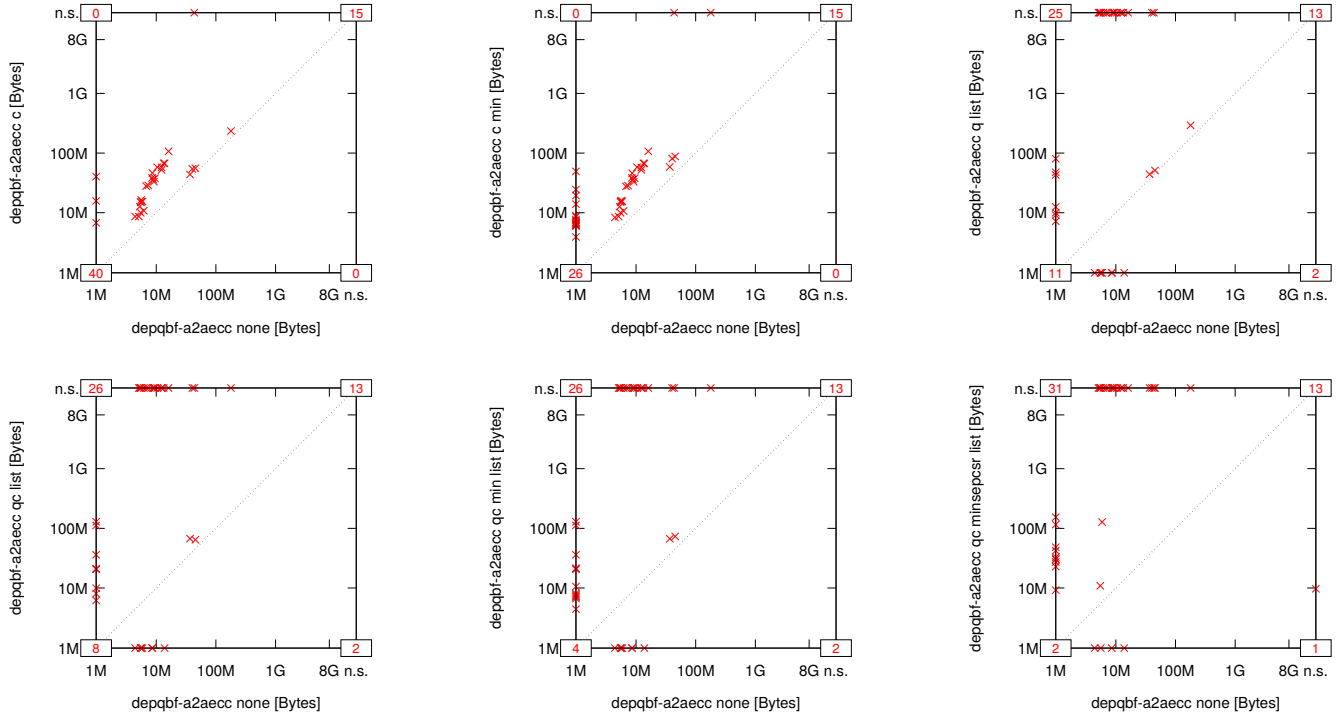


Fig. 840: Suite Letombe ($n = 85$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

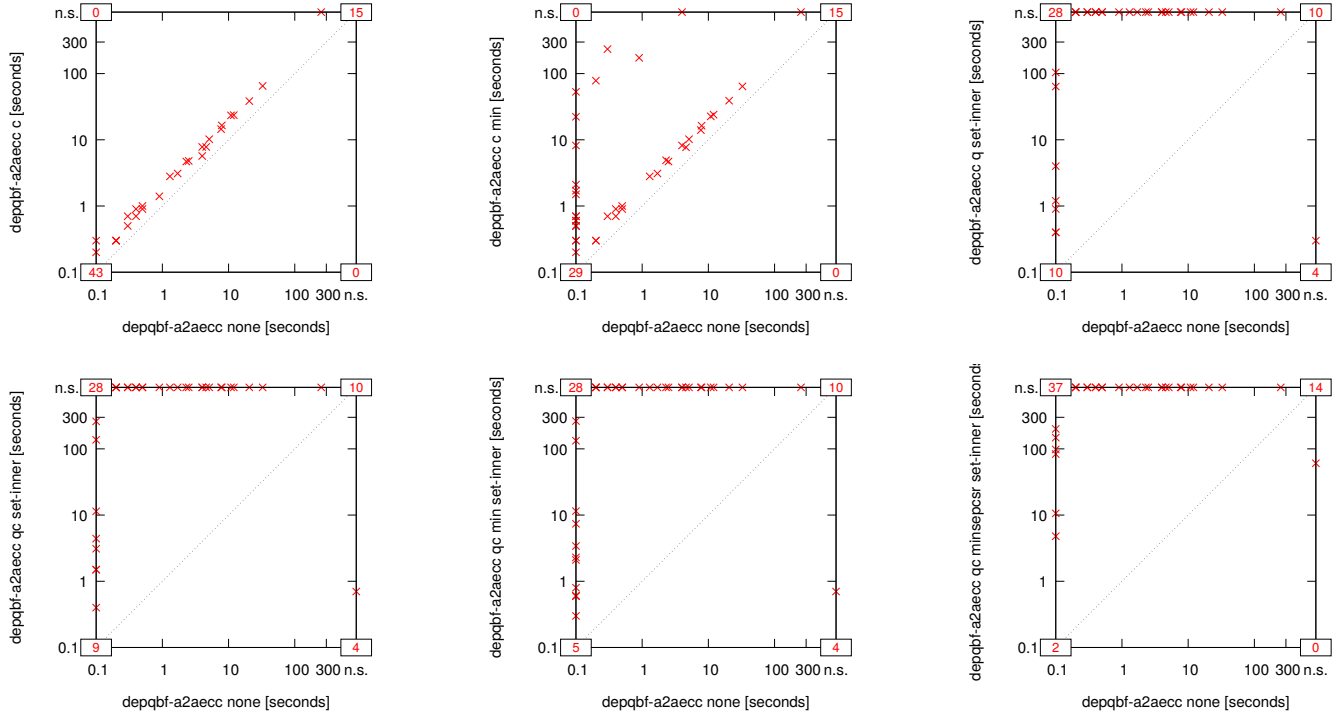


Fig. 841: Suite Letombe ($n = 85$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

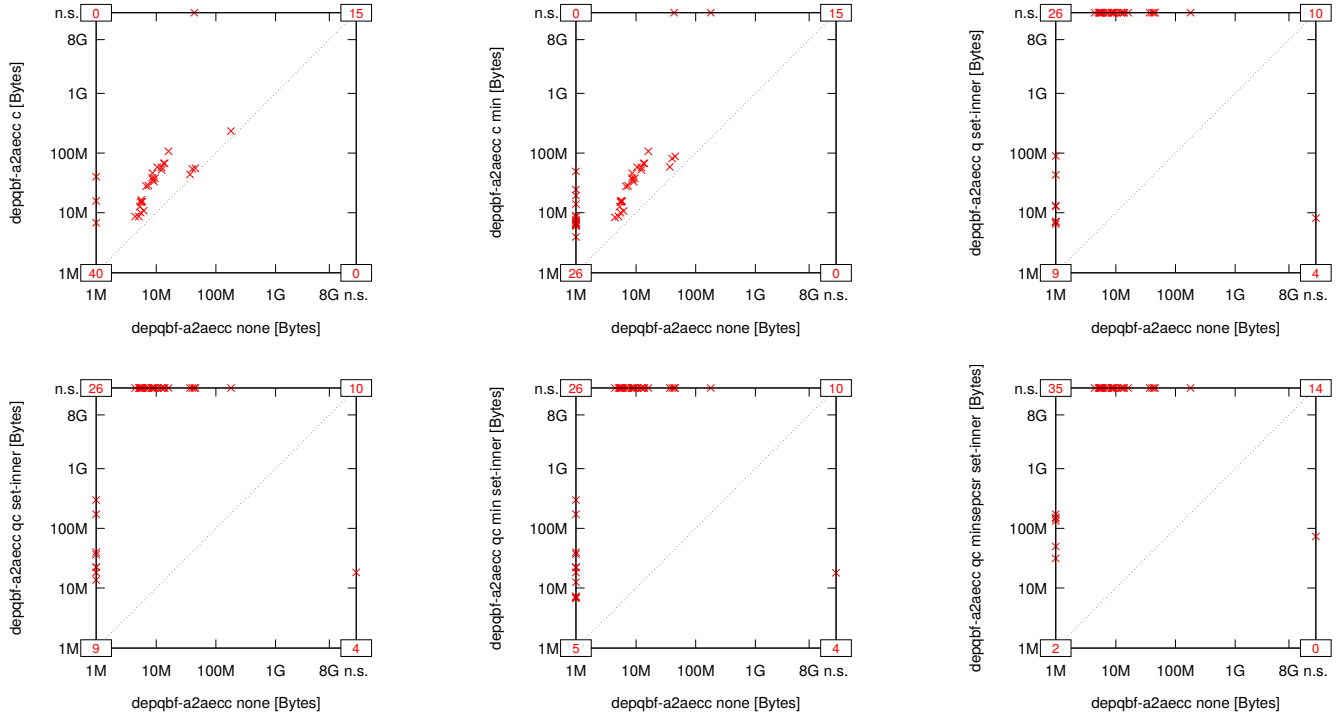


Fig. 842: Suite Letombe ($n = 85$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

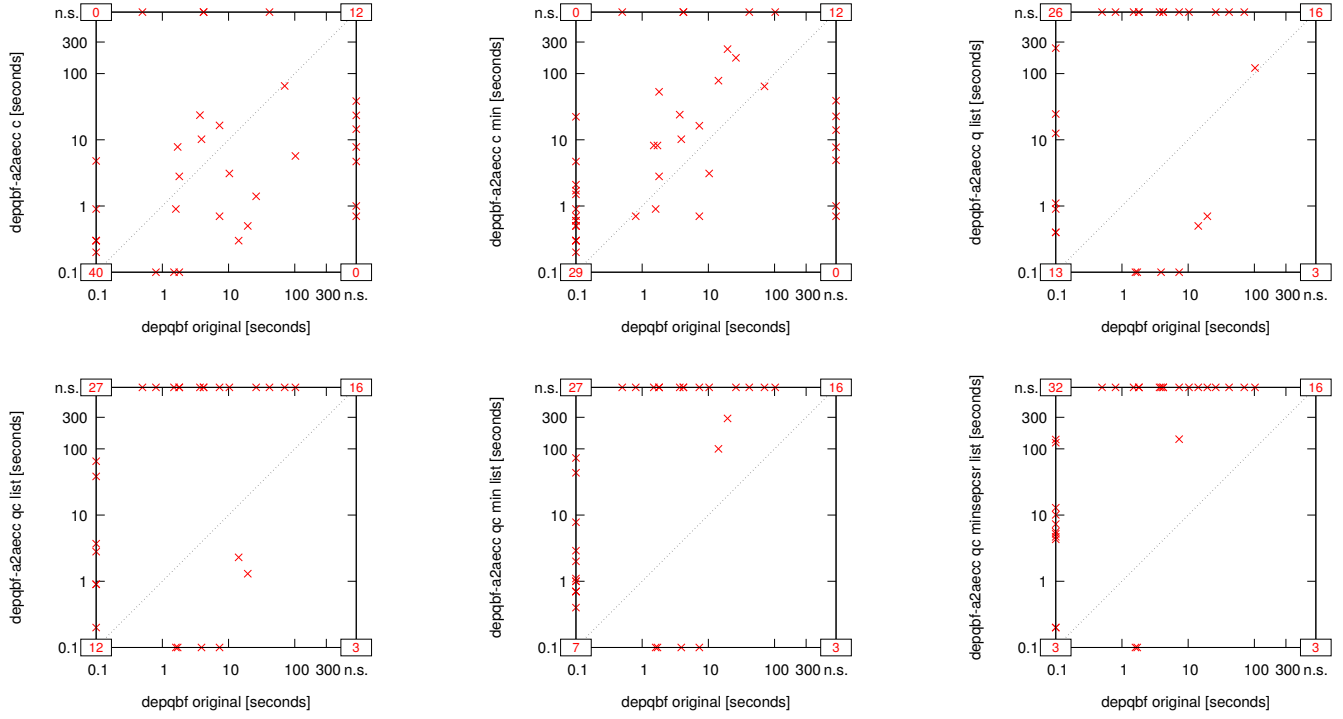


Fig. 843: Suite Letombe ($n = 85$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

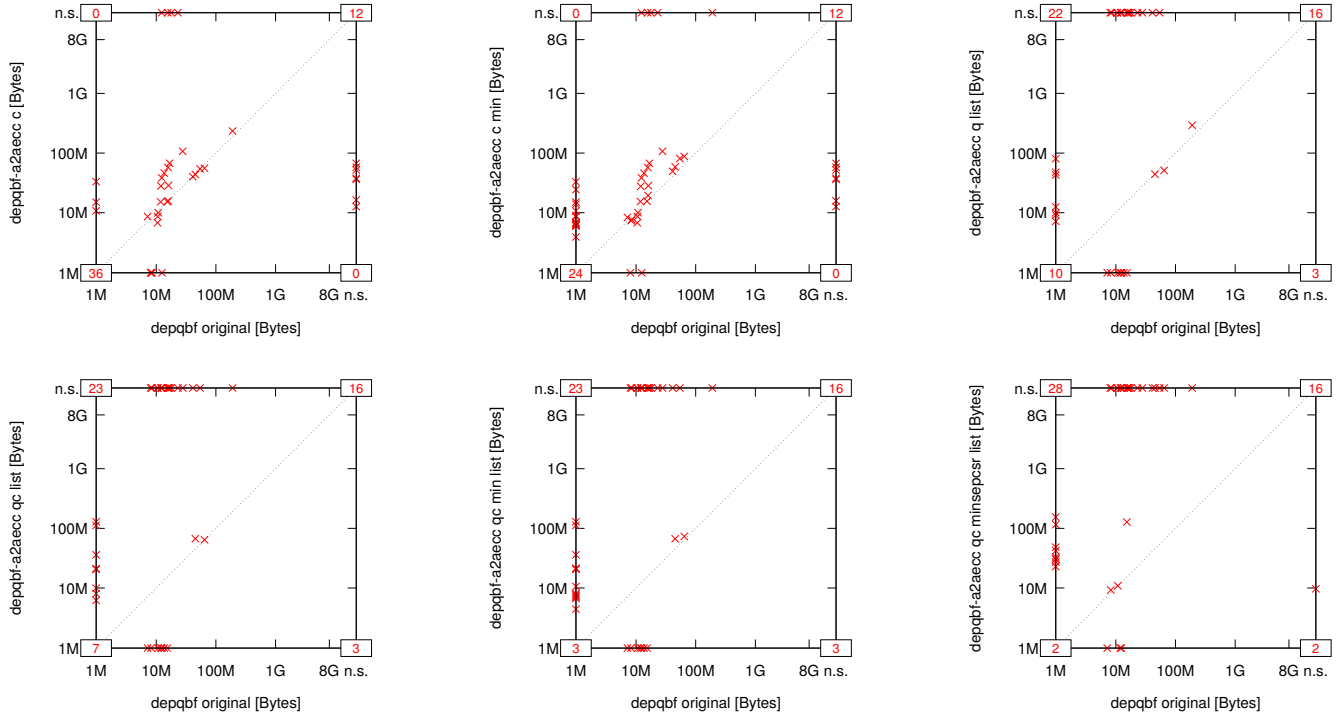


Fig. 844: Suite Letombe ($n = 85$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

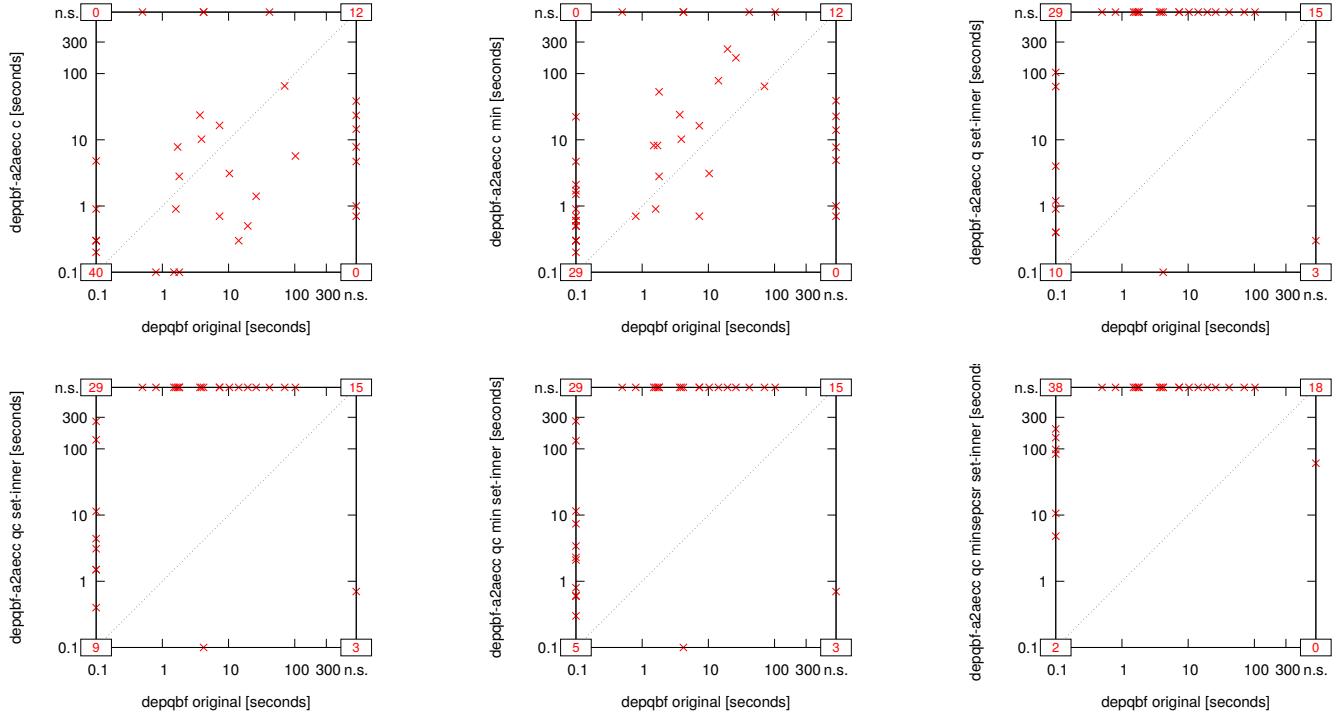


Fig. 845: Suite Letombe ($n = 85$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

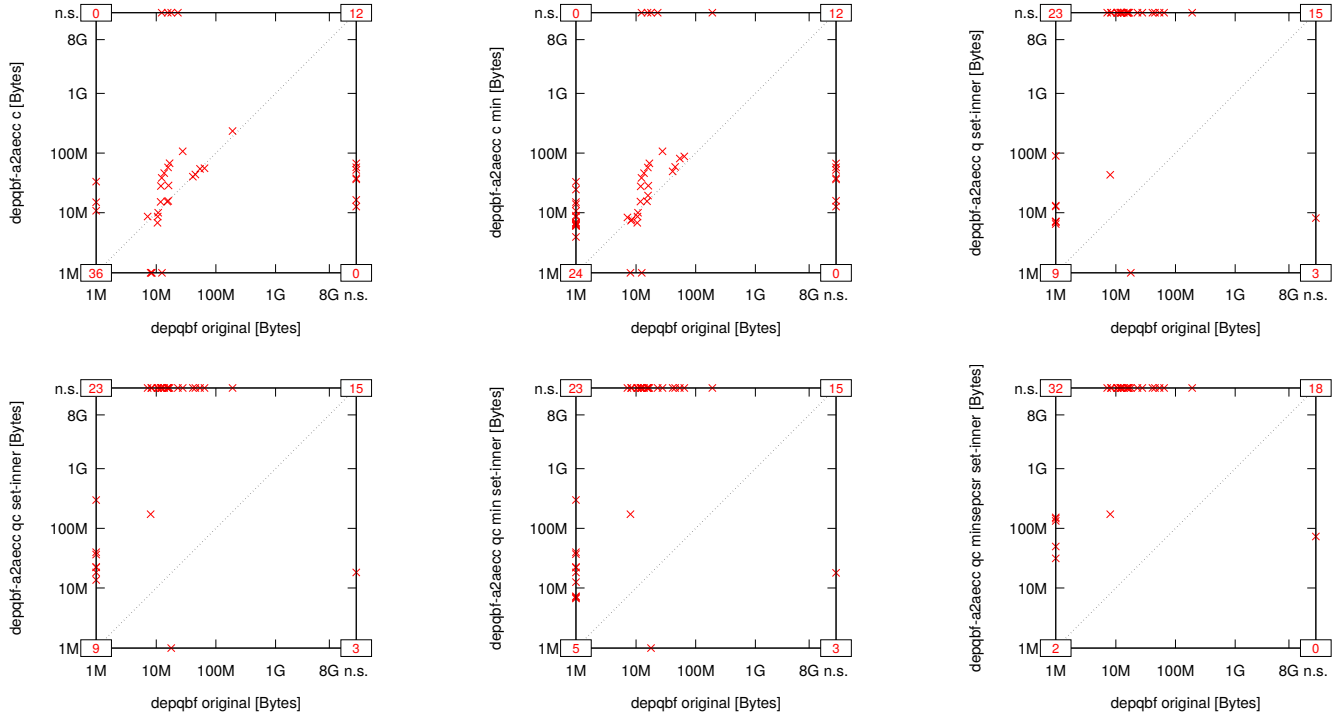


Fig. 846: Suite Letombe ($n = 85$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

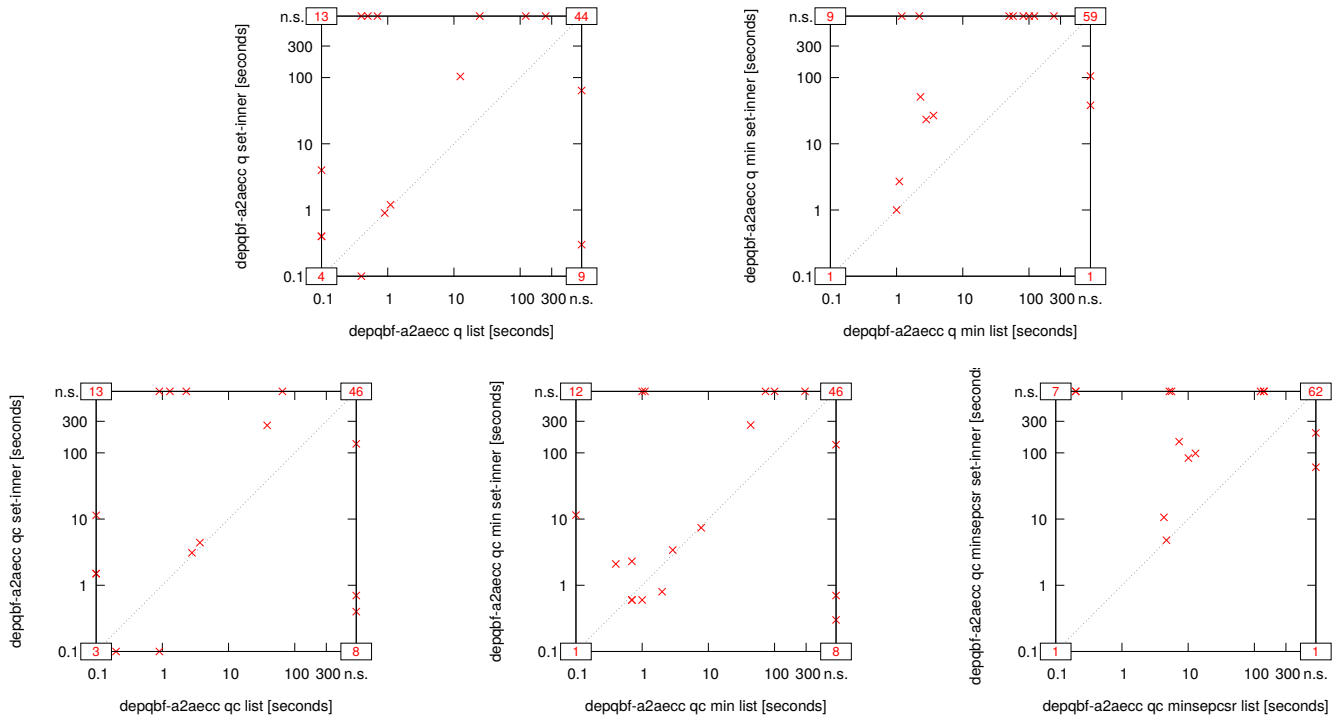


Fig. 847: Suite Letombe ($n = 85$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

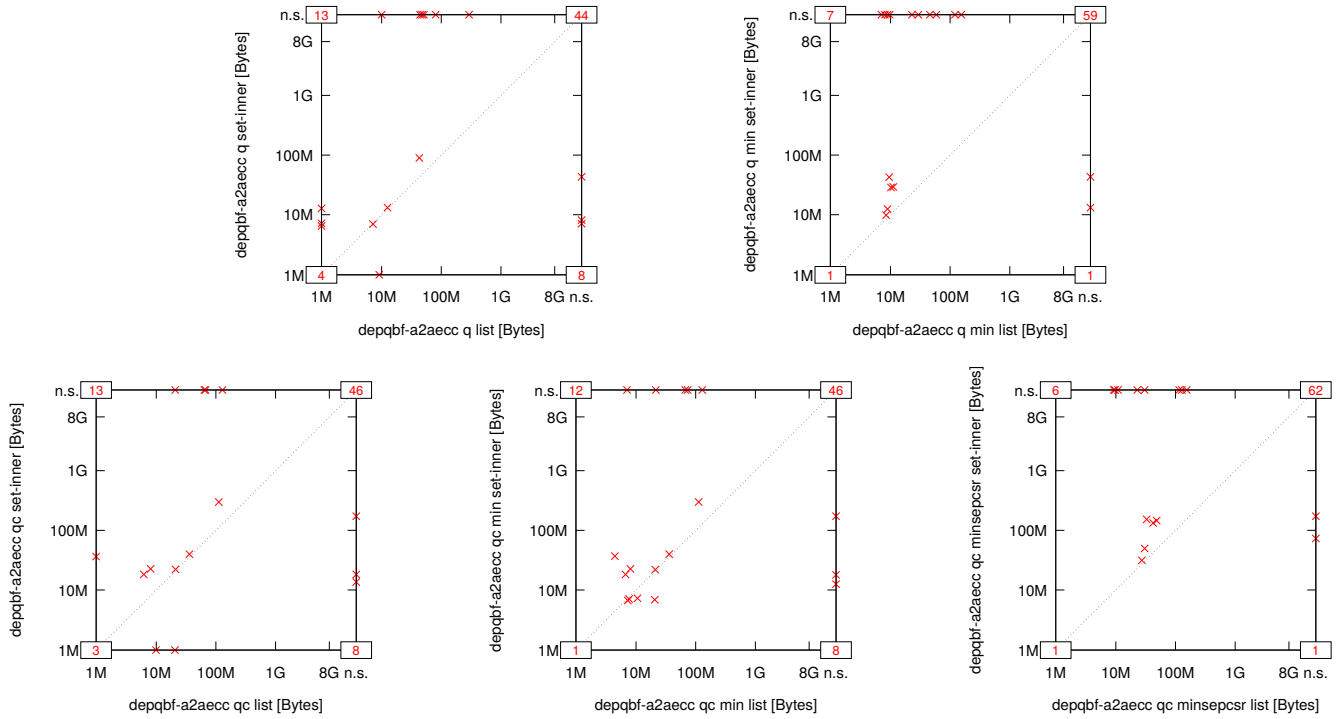


Fig. 848: Suite Letombe ($n = 85$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

25) Letz ($n = 9$):

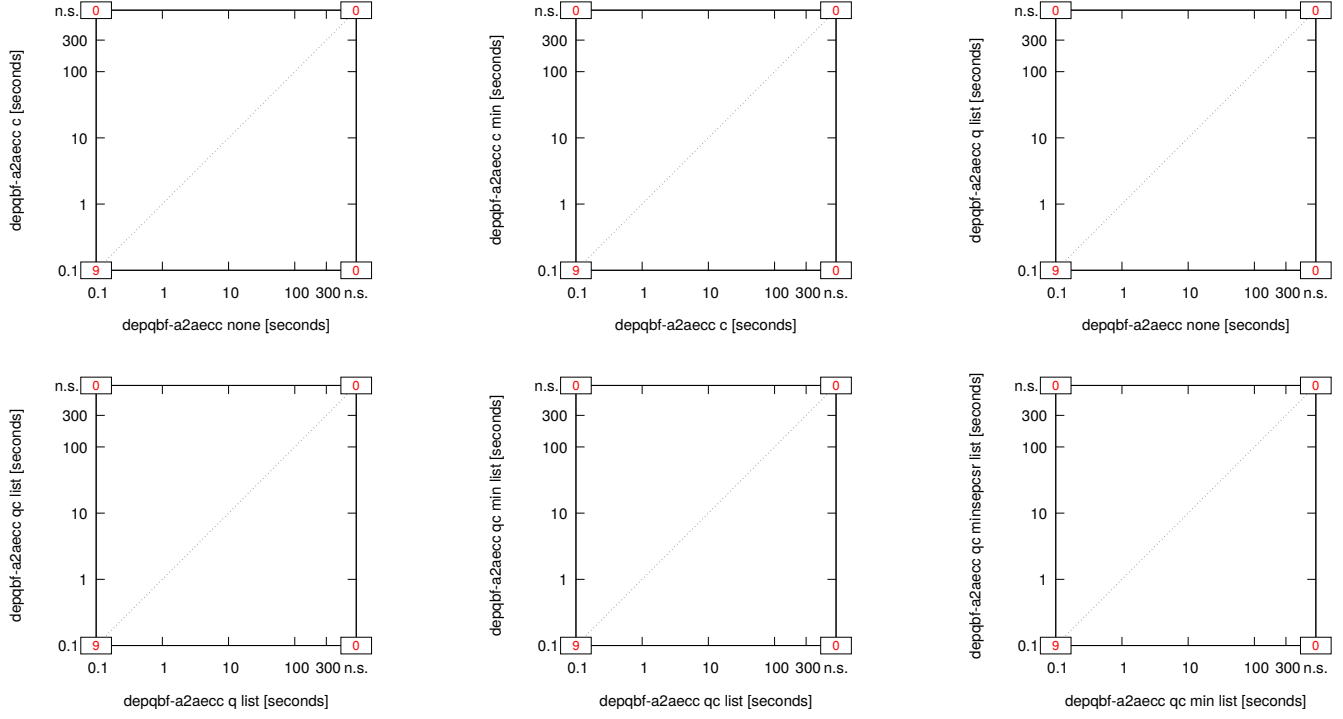


Fig. 849: Suite Letz ($n = 9$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

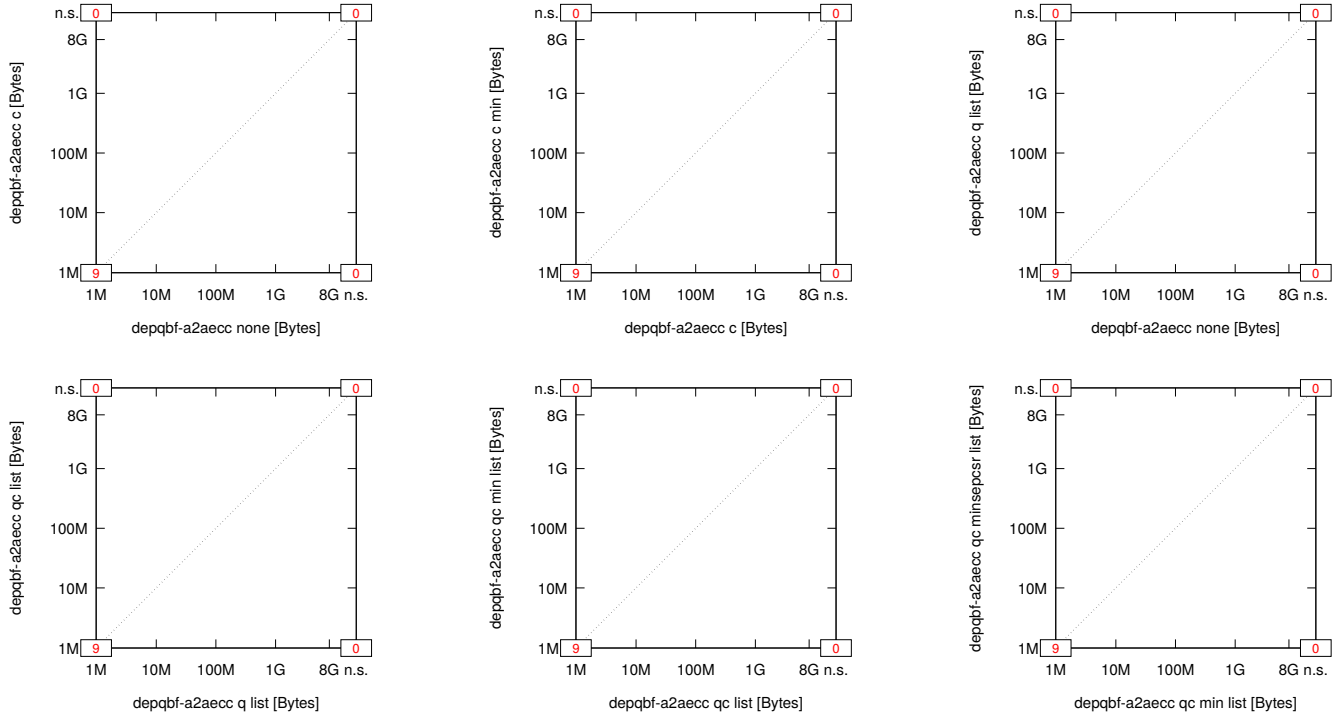


Fig. 850: Suite Letz ($n = 9$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

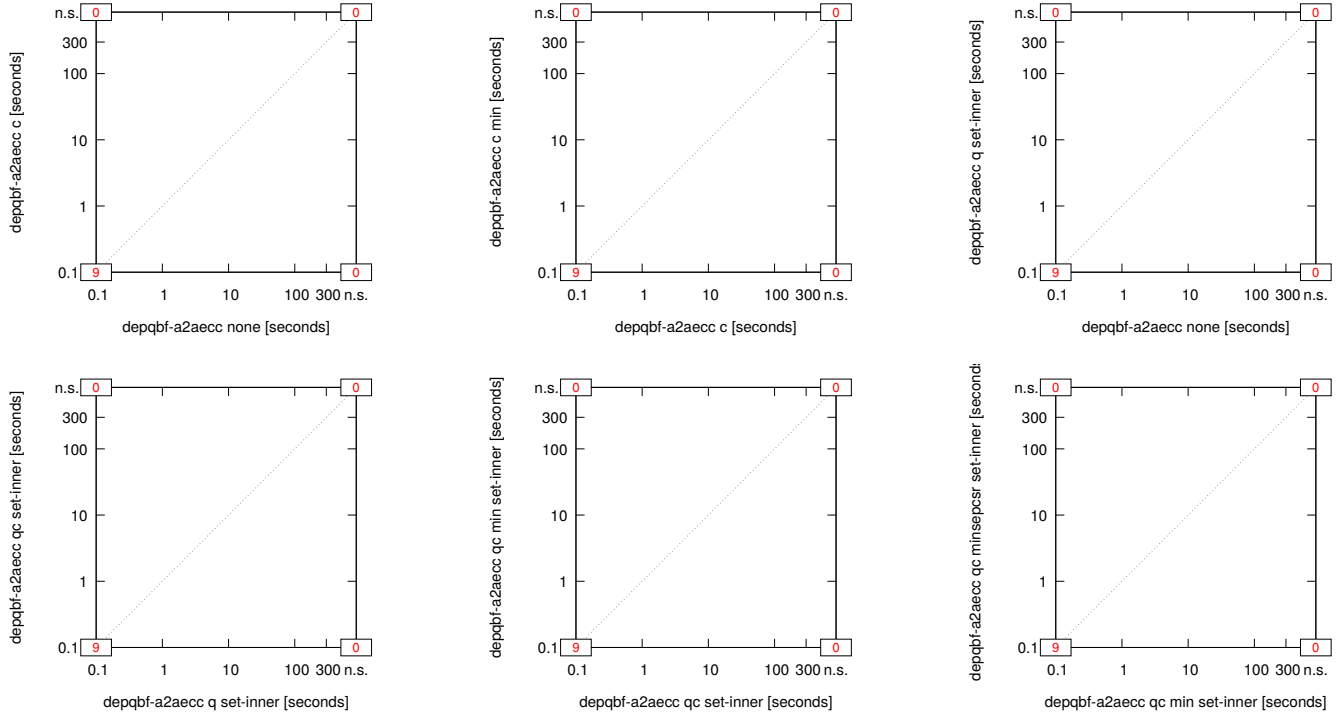


Fig. 851: Suite Letz ($n = 9$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

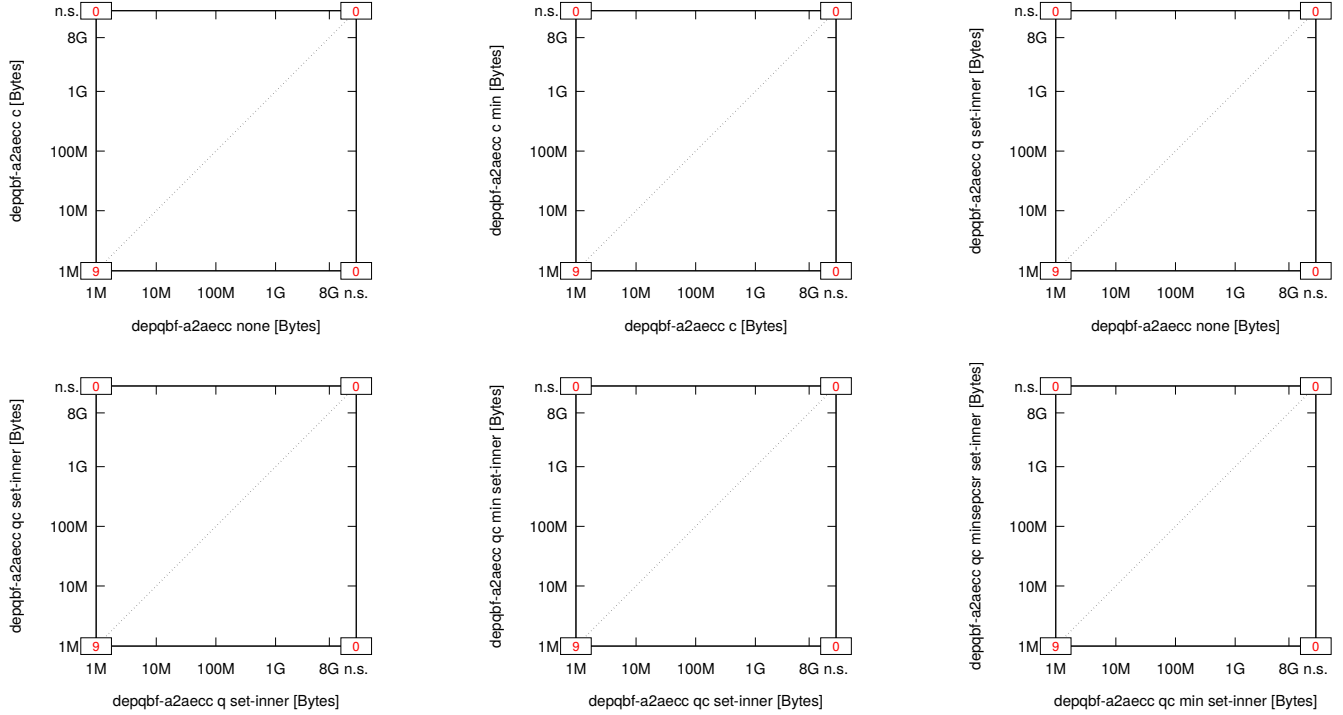


Fig. 852: Suite Letz ($n = 9$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

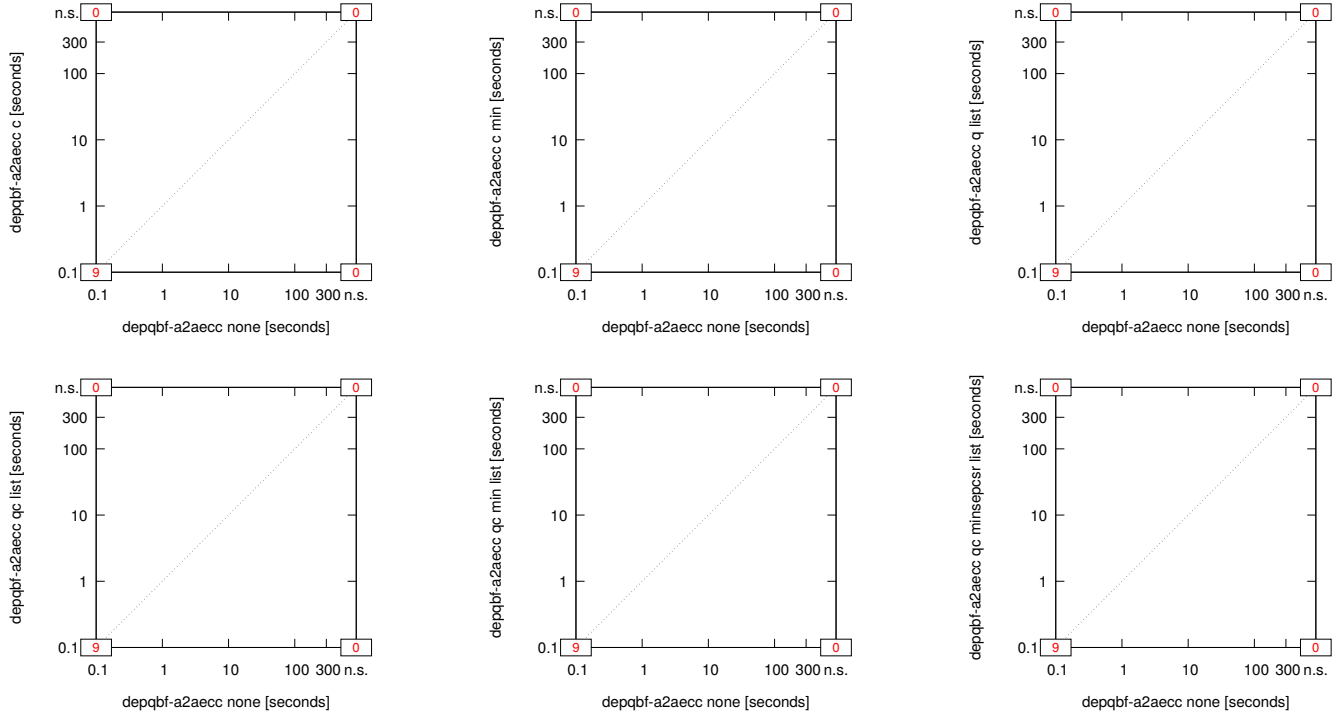


Fig. 853: Suite Letz ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

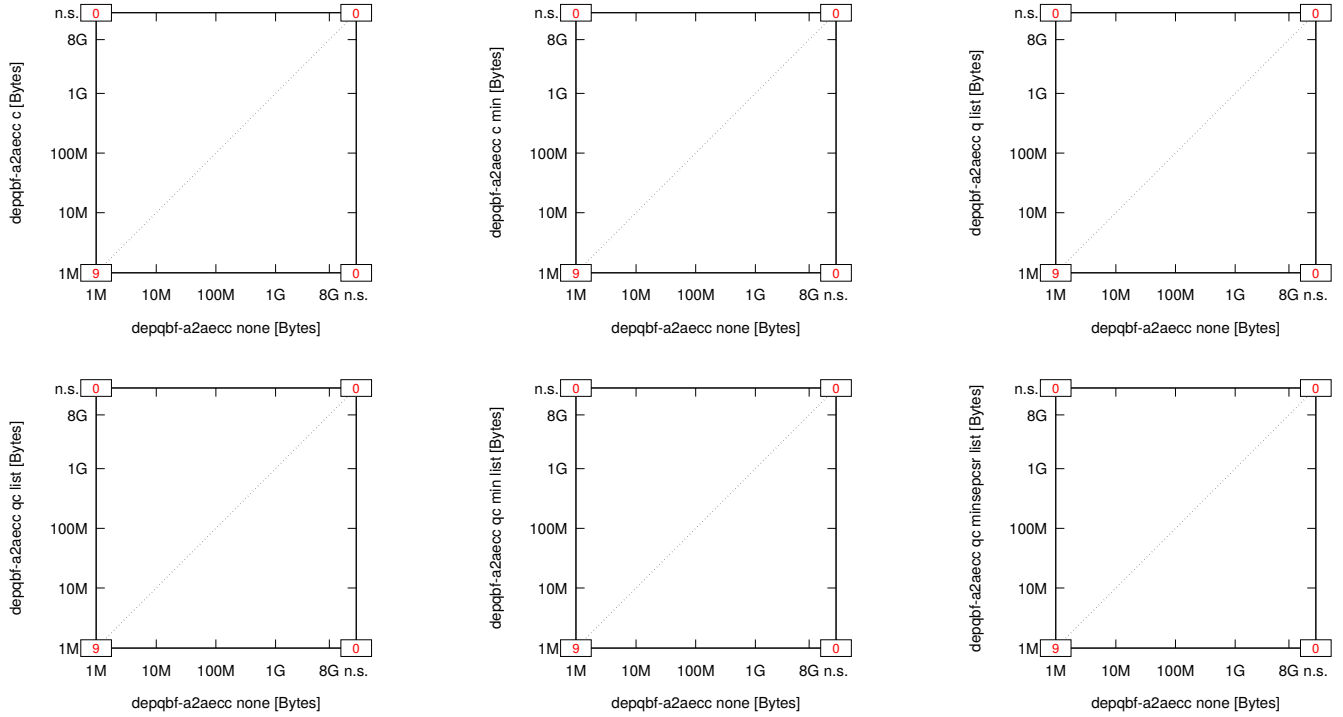


Fig. 854: Suite Letz ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

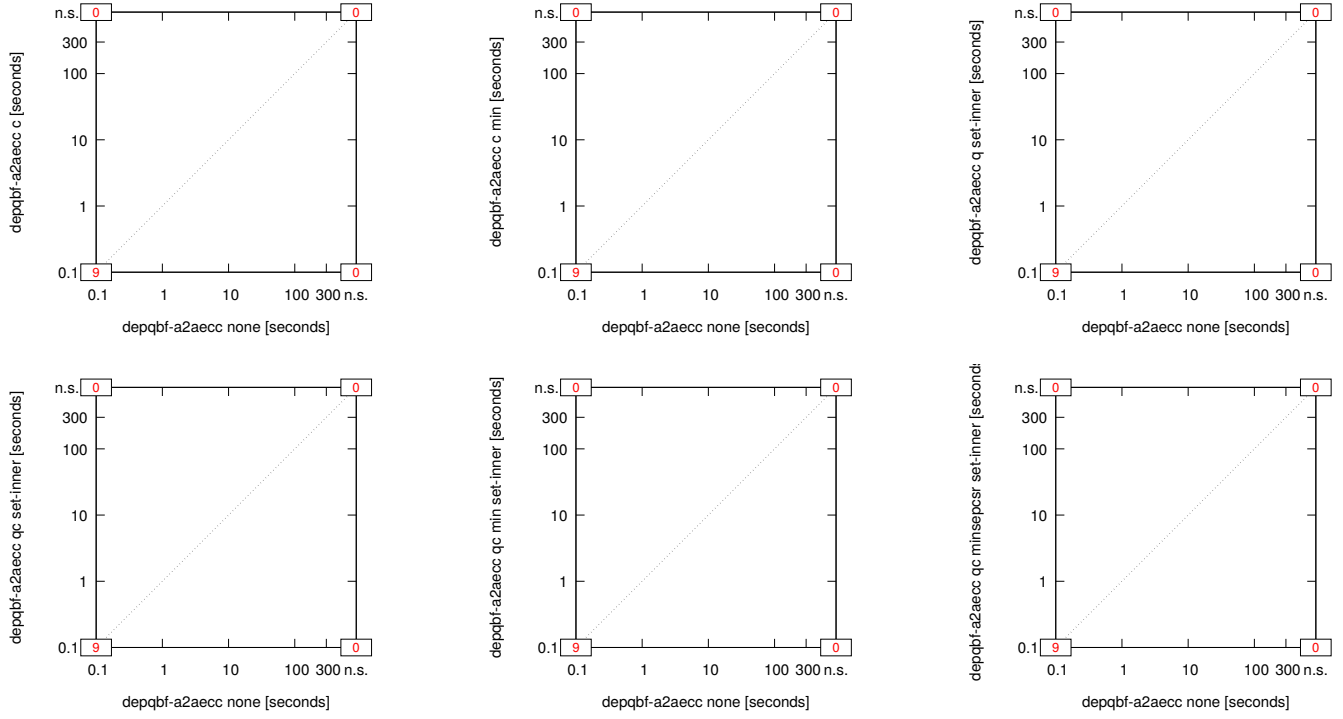


Fig. 855: Suite Letz ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

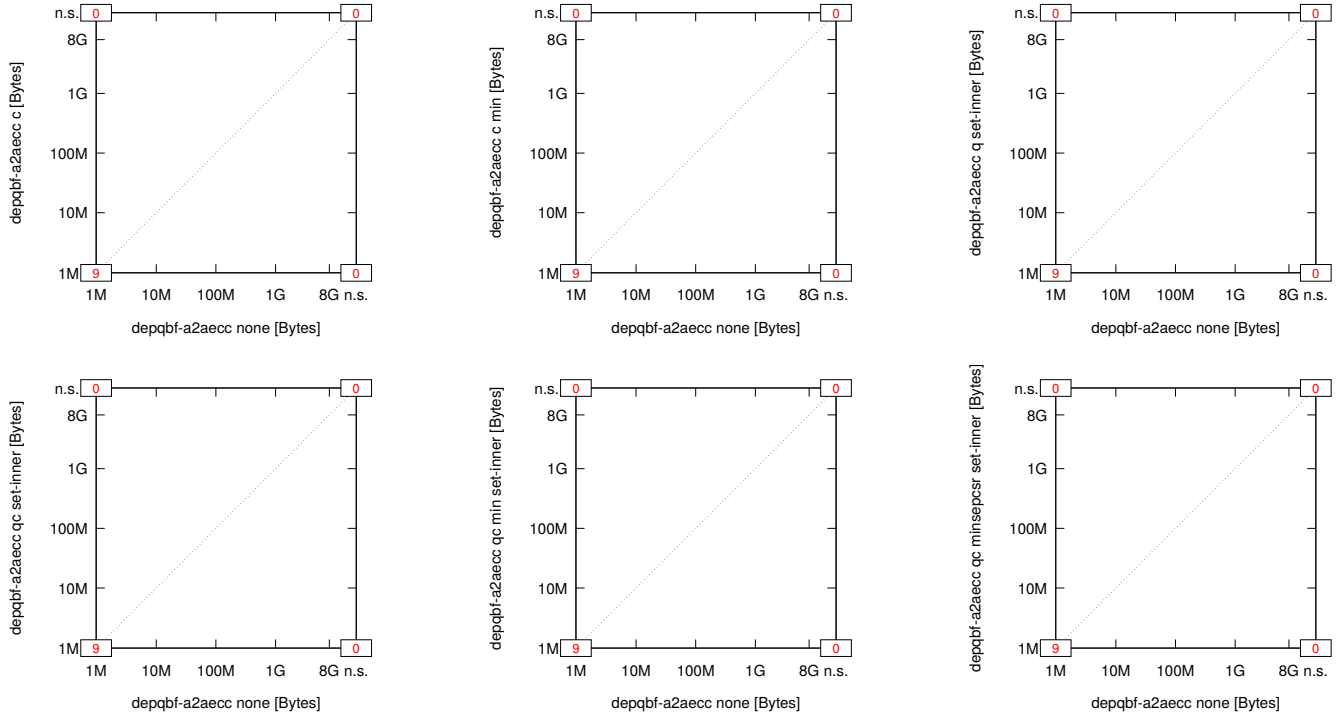


Fig. 856: Suite Letz ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

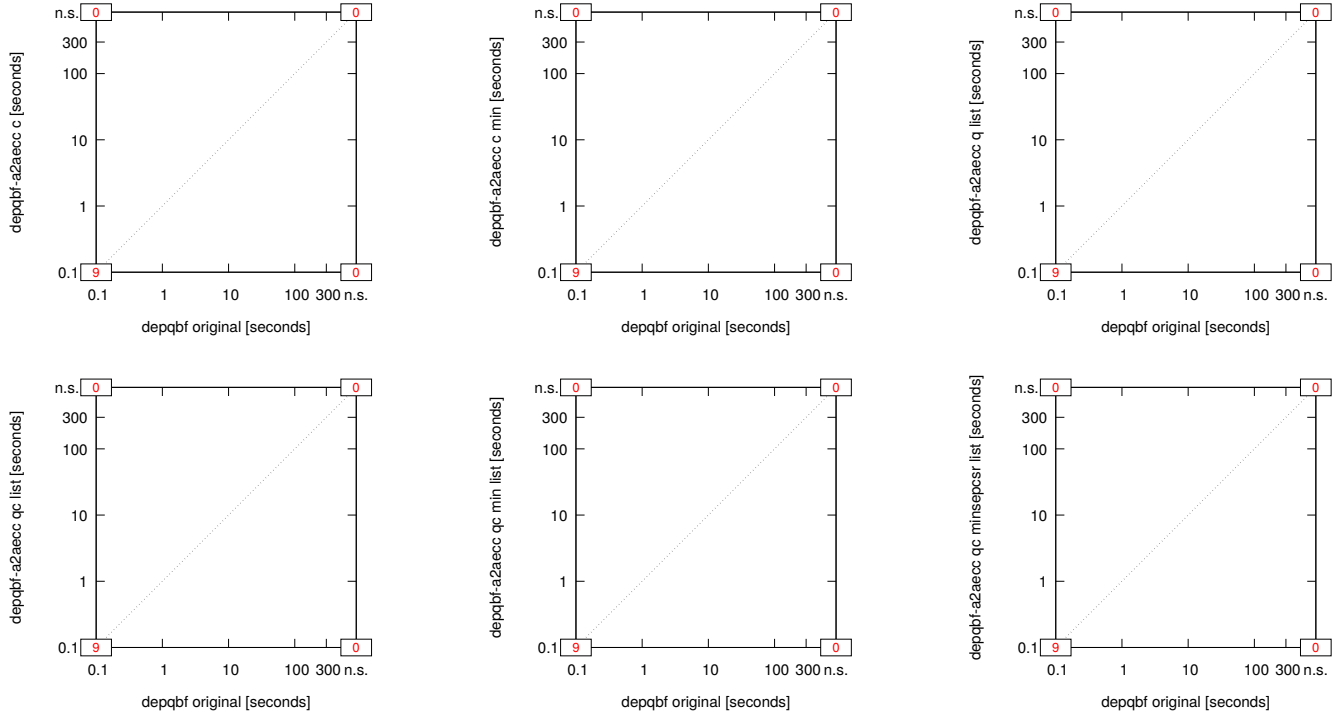


Fig. 857: Suite Letz ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

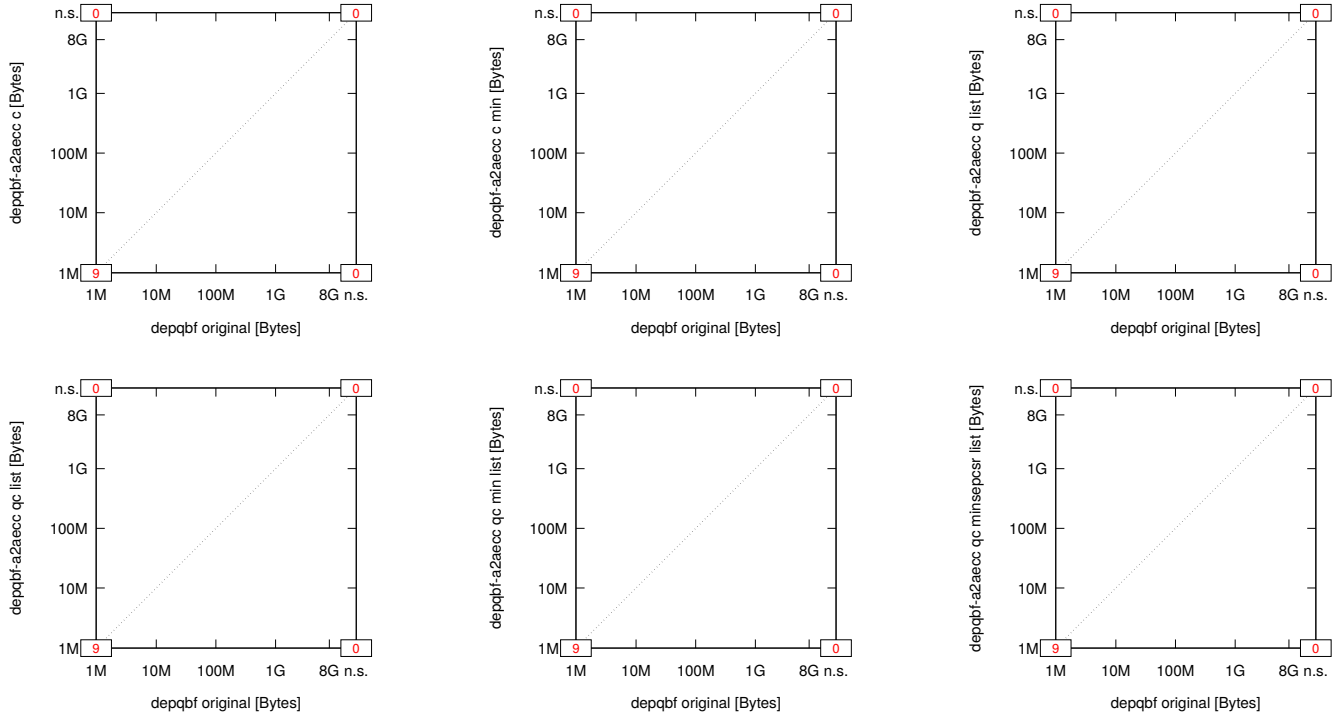


Fig. 858: Suite Letz ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

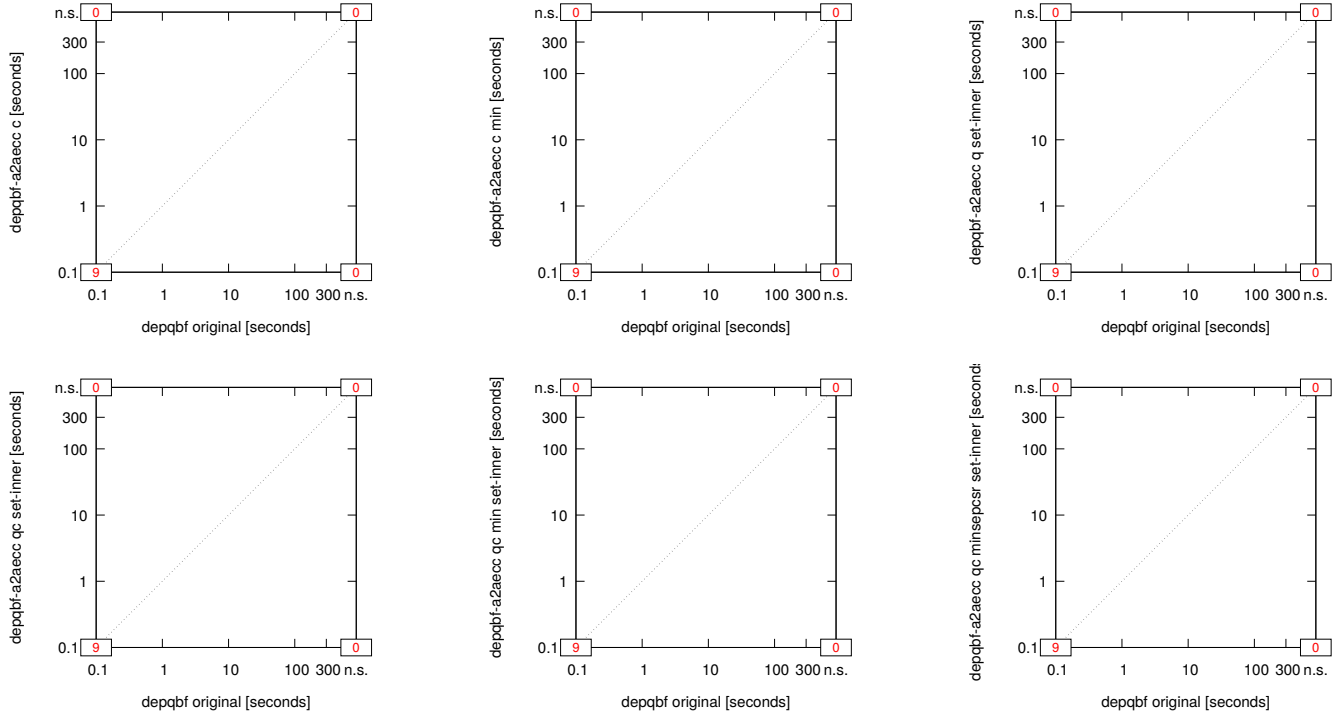


Fig. 859: Suite Letz ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

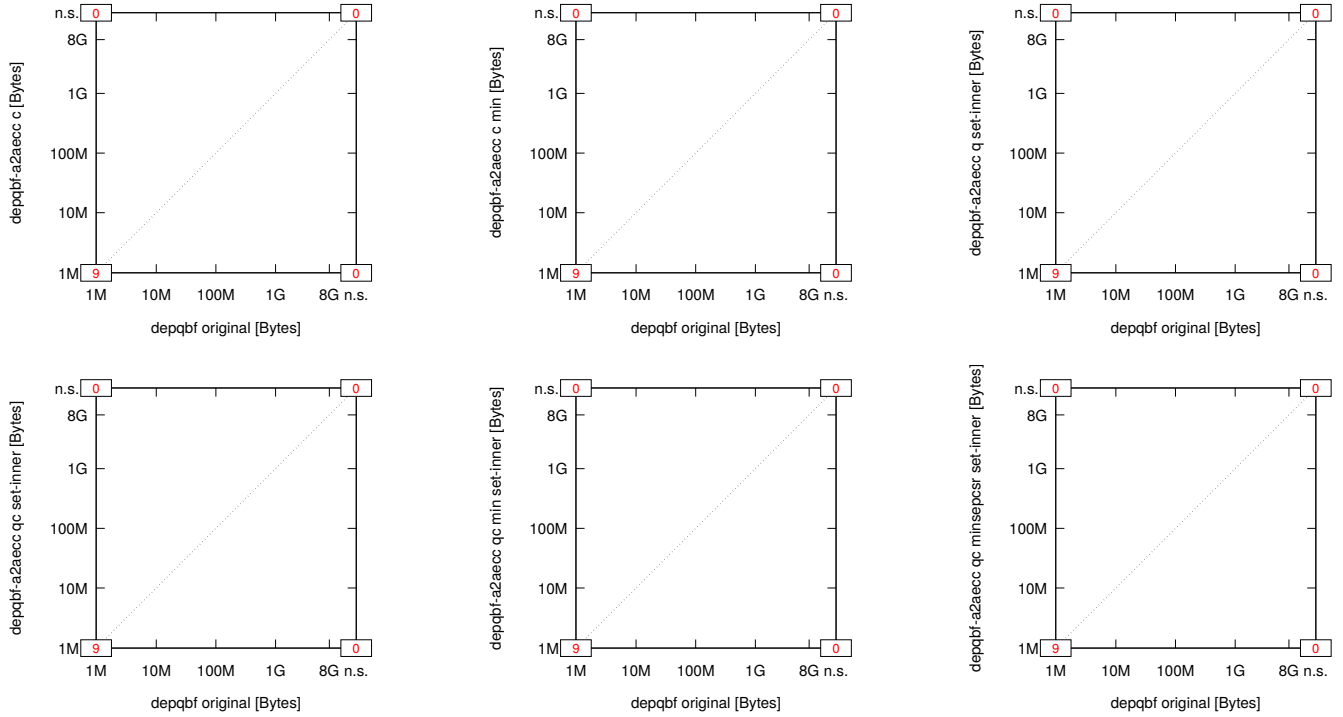


Fig. 860: Suite Letz ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

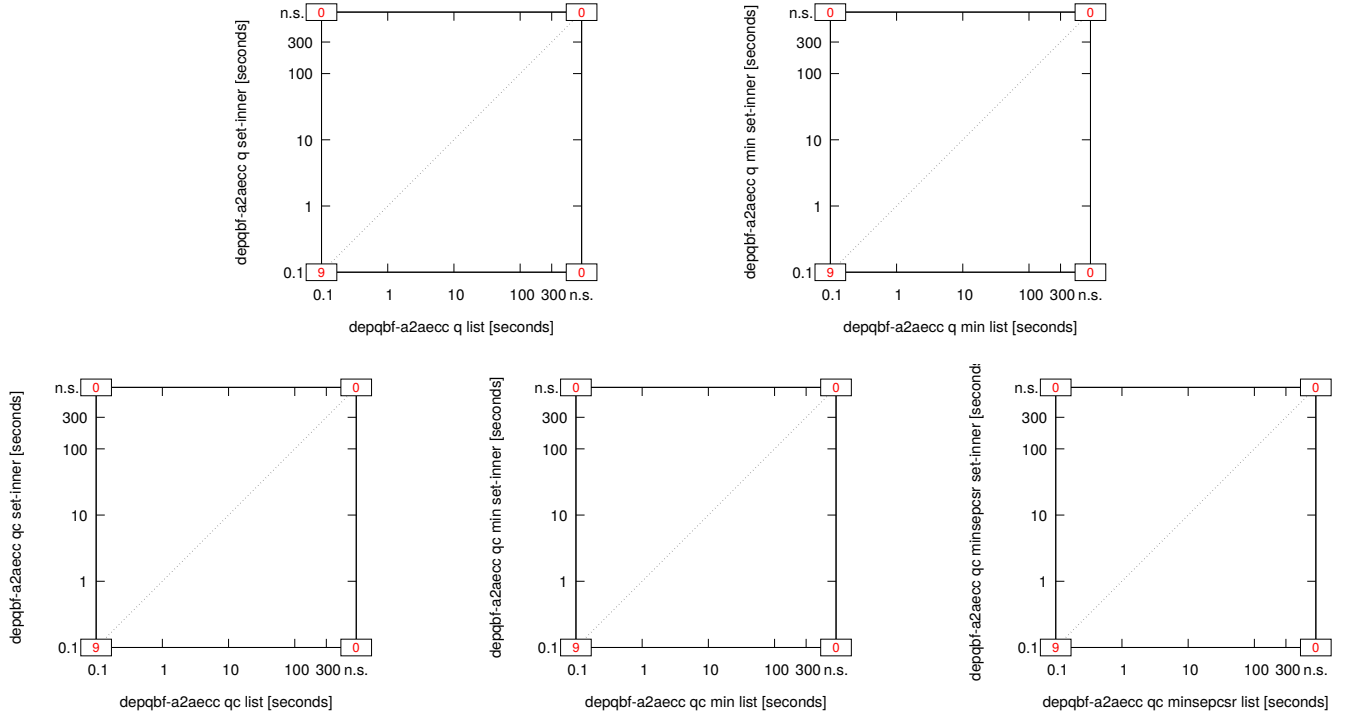


Fig. 861: Suite Letz ($n = 9$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

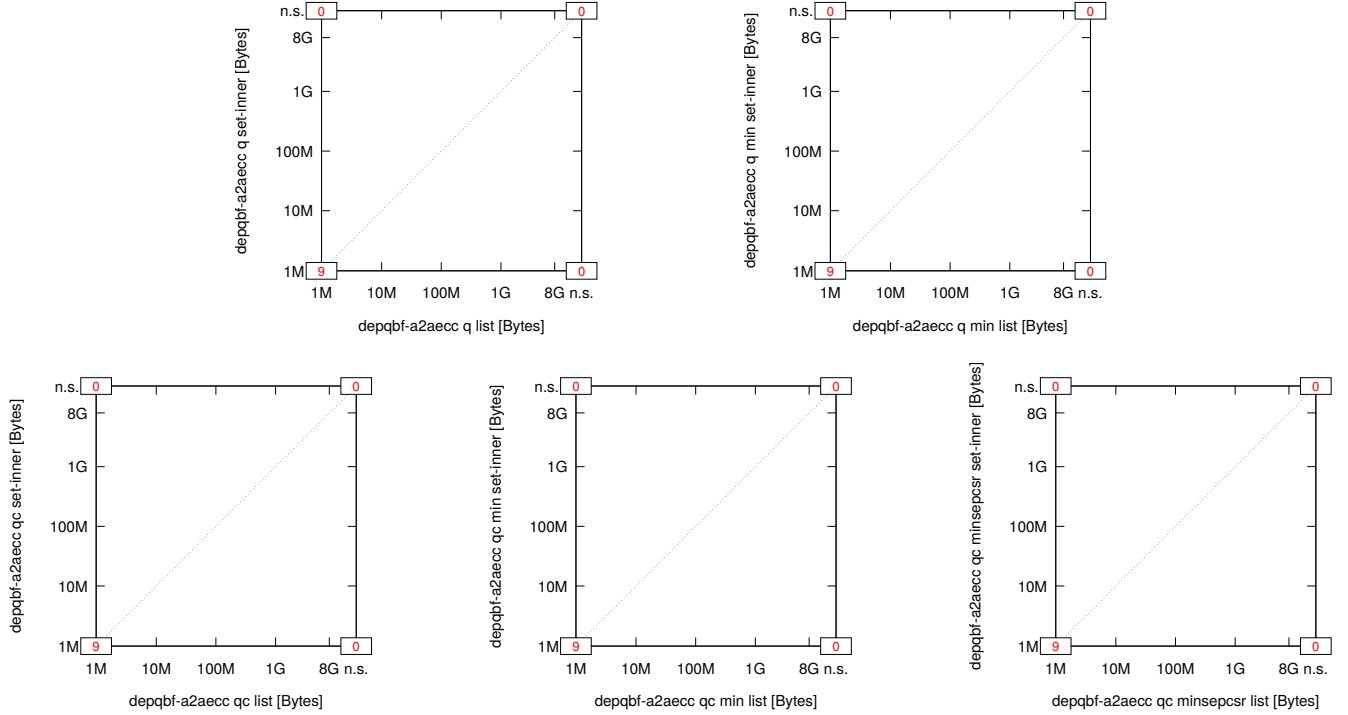


Fig. 862: Suite Letz ($n = 9$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

26) Ling ($n = 3$):

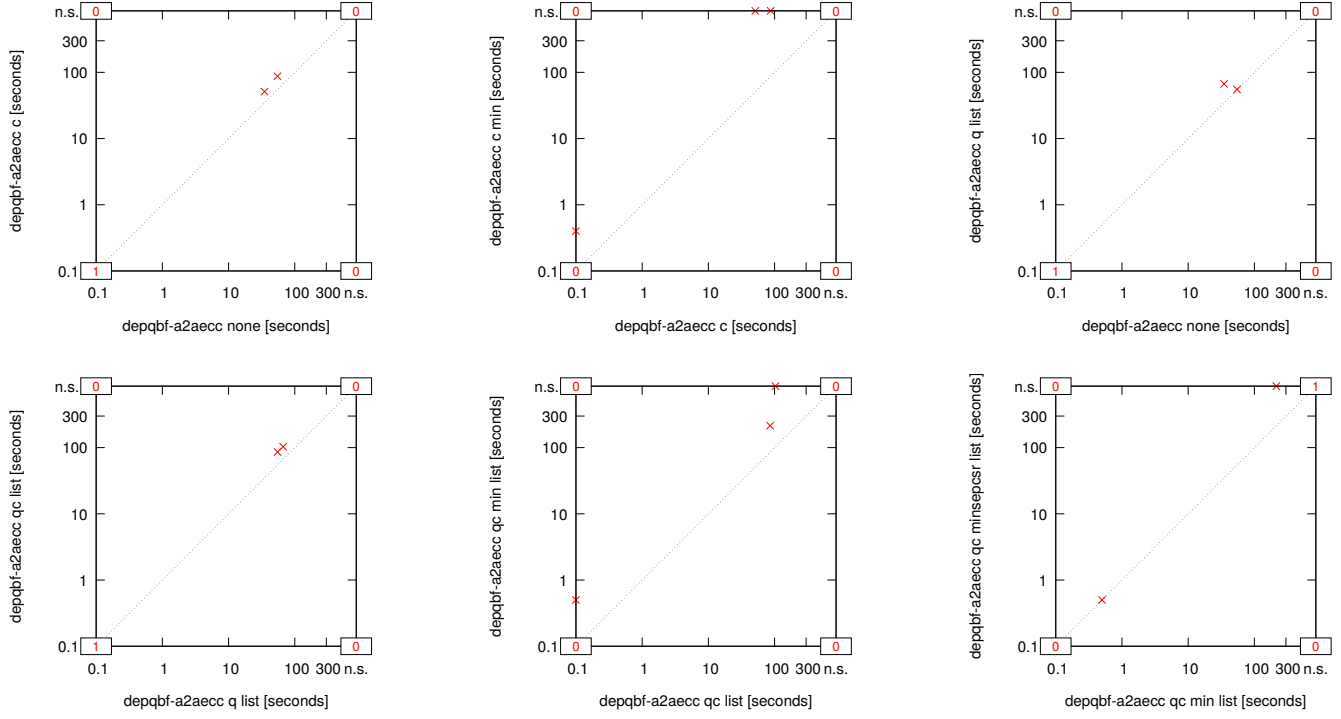


Fig. 863: Suite Ling ($n = 3$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

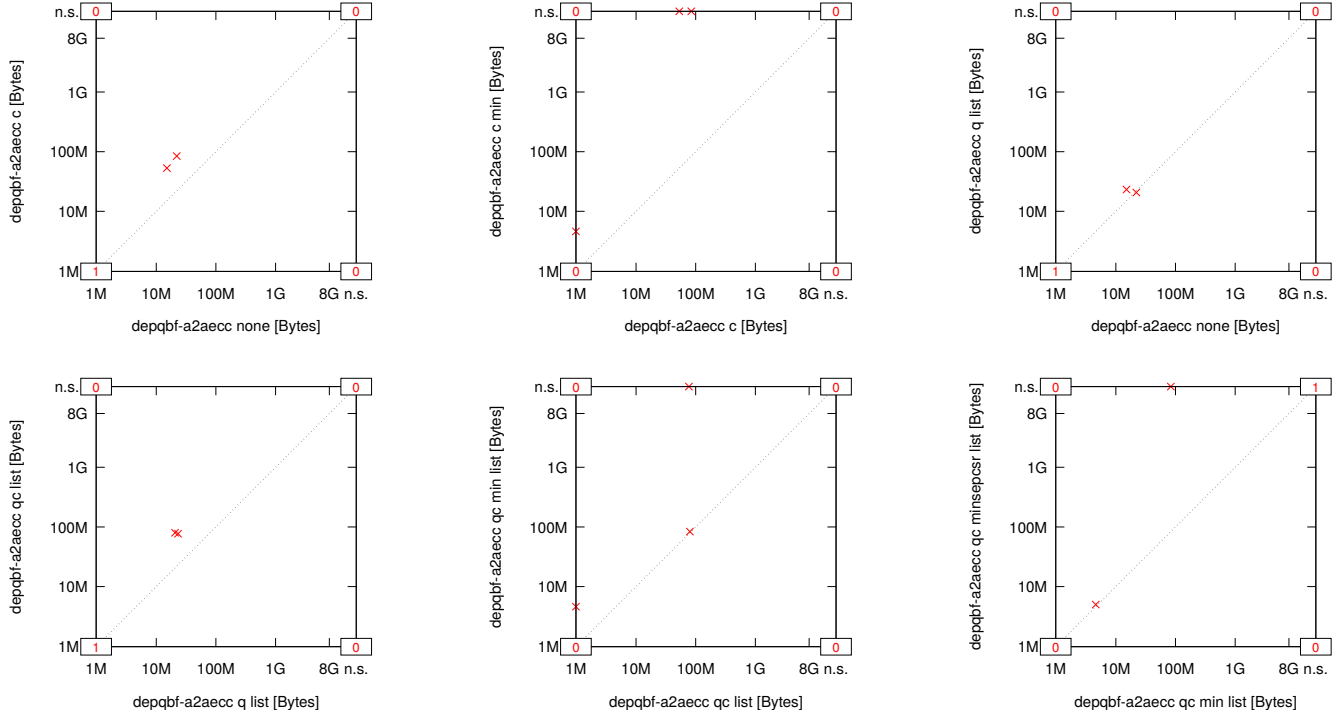


Fig. 864: Suite Ling ($n = 3$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

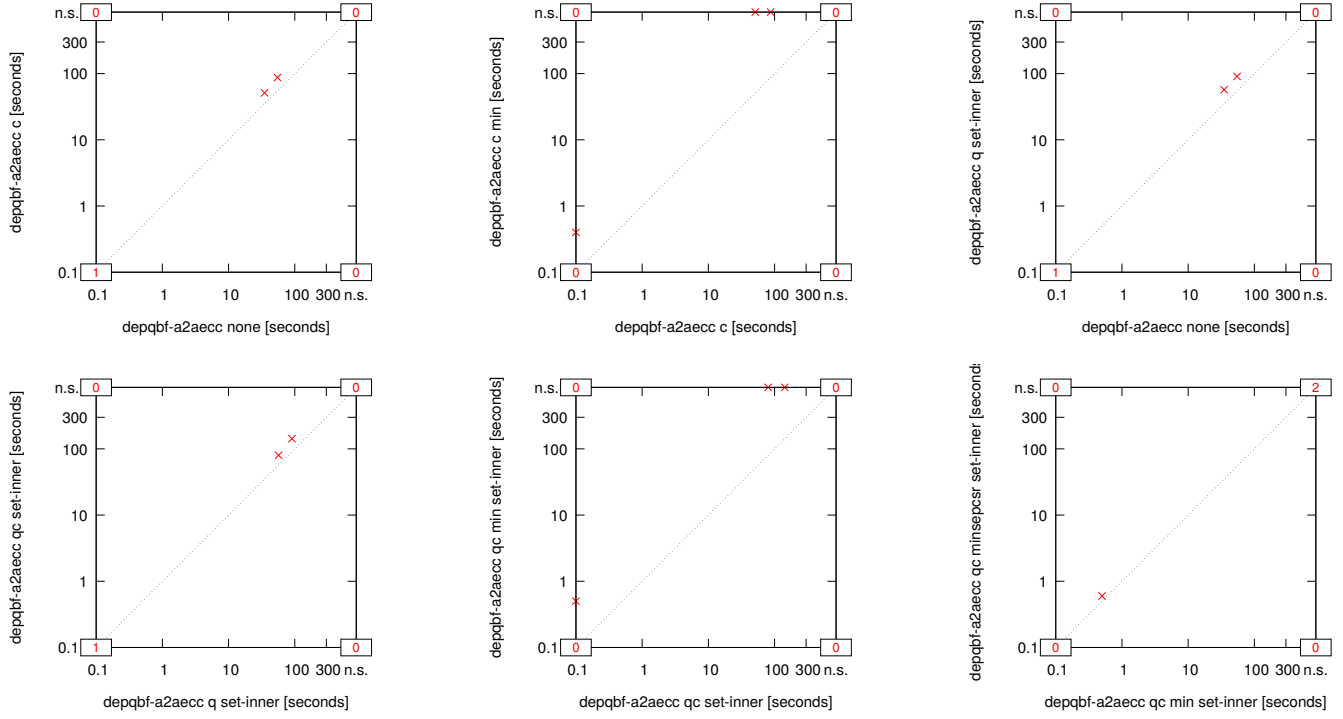


Fig. 865: Suite Ling ($n = 3$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

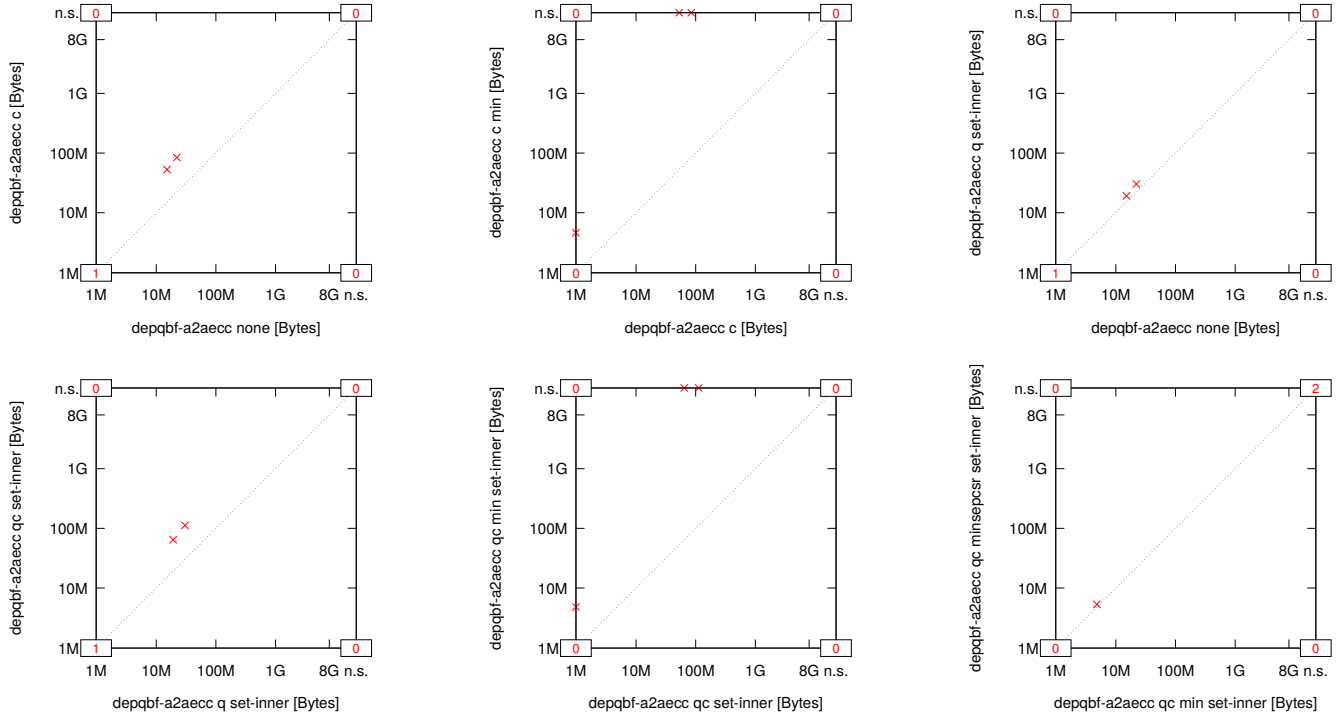


Fig. 866: Suite Ling ($n = 3$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

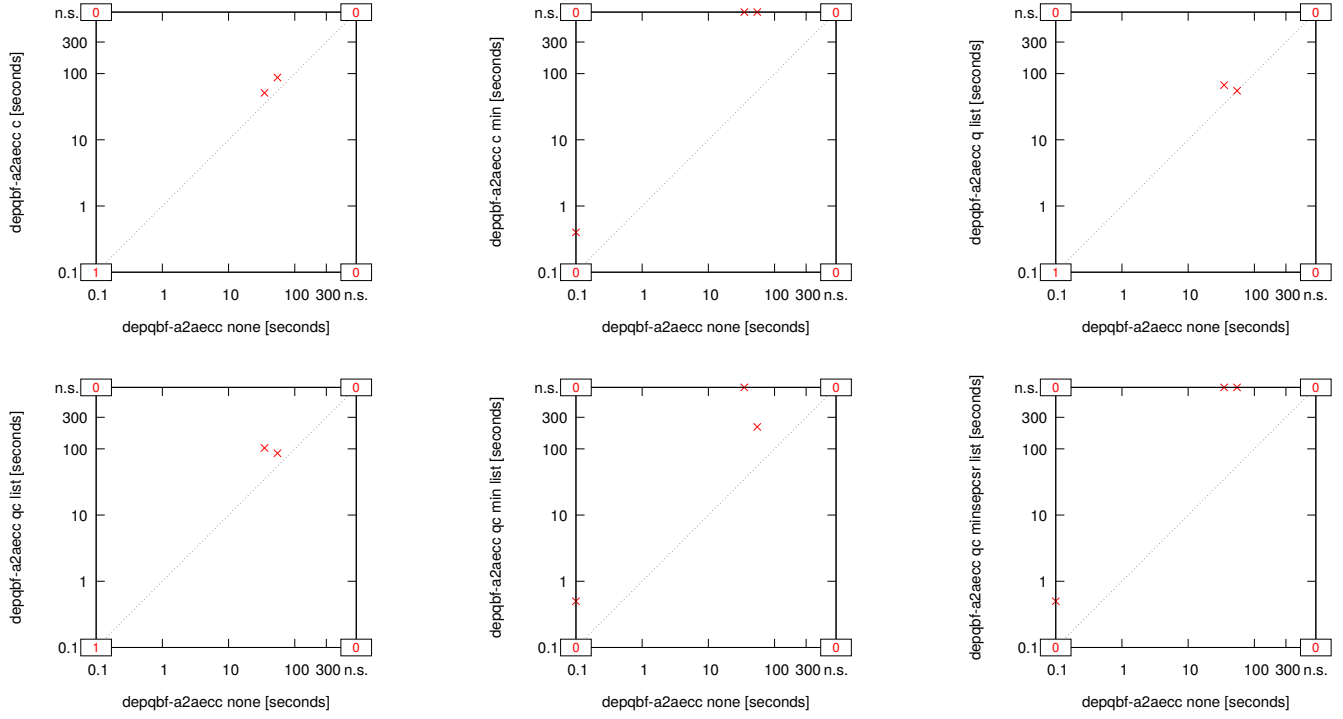


Fig. 867: Suite Ling ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

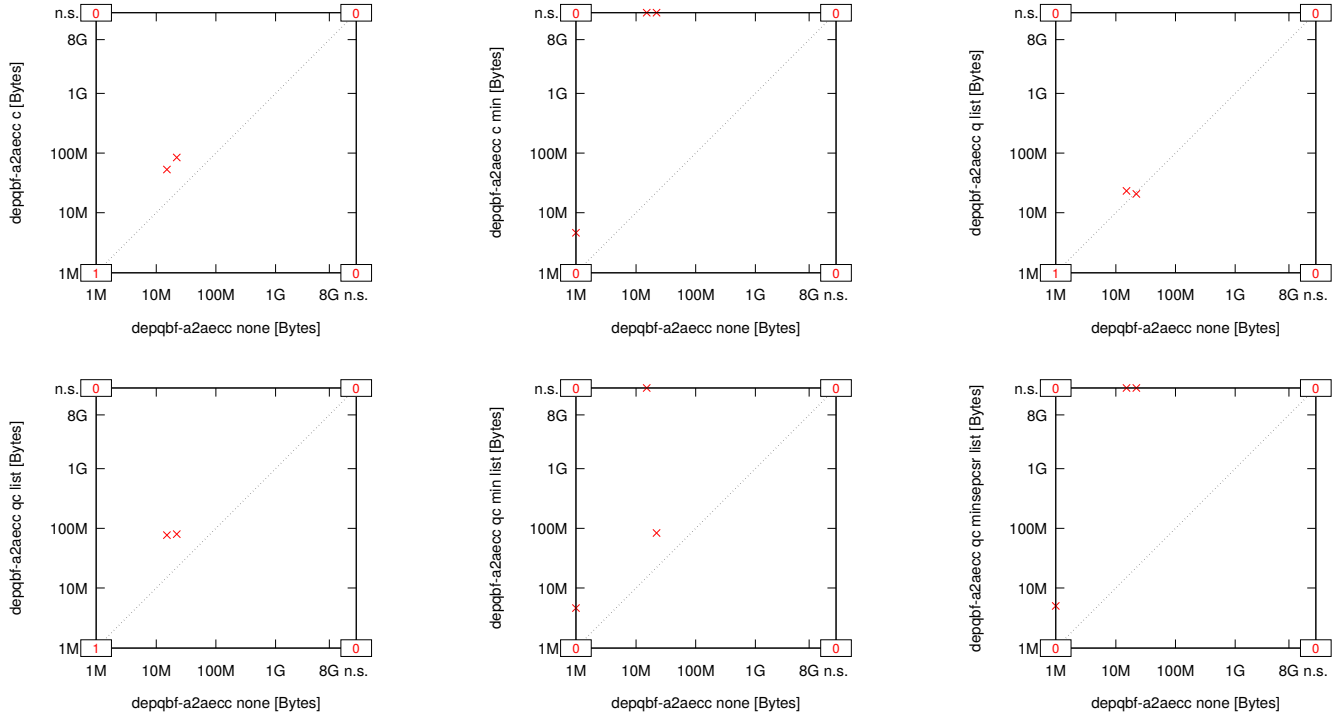


Fig. 868: Suite Ling ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

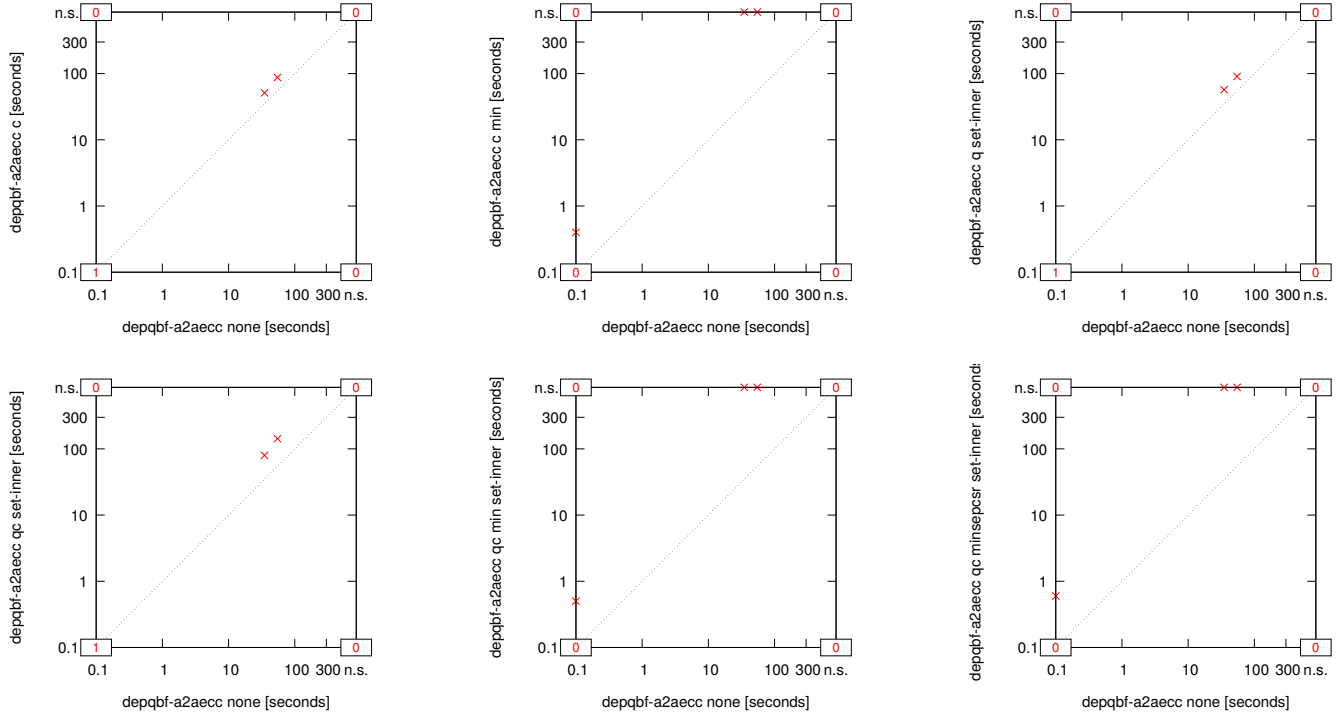


Fig. 869: Suite Ling ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

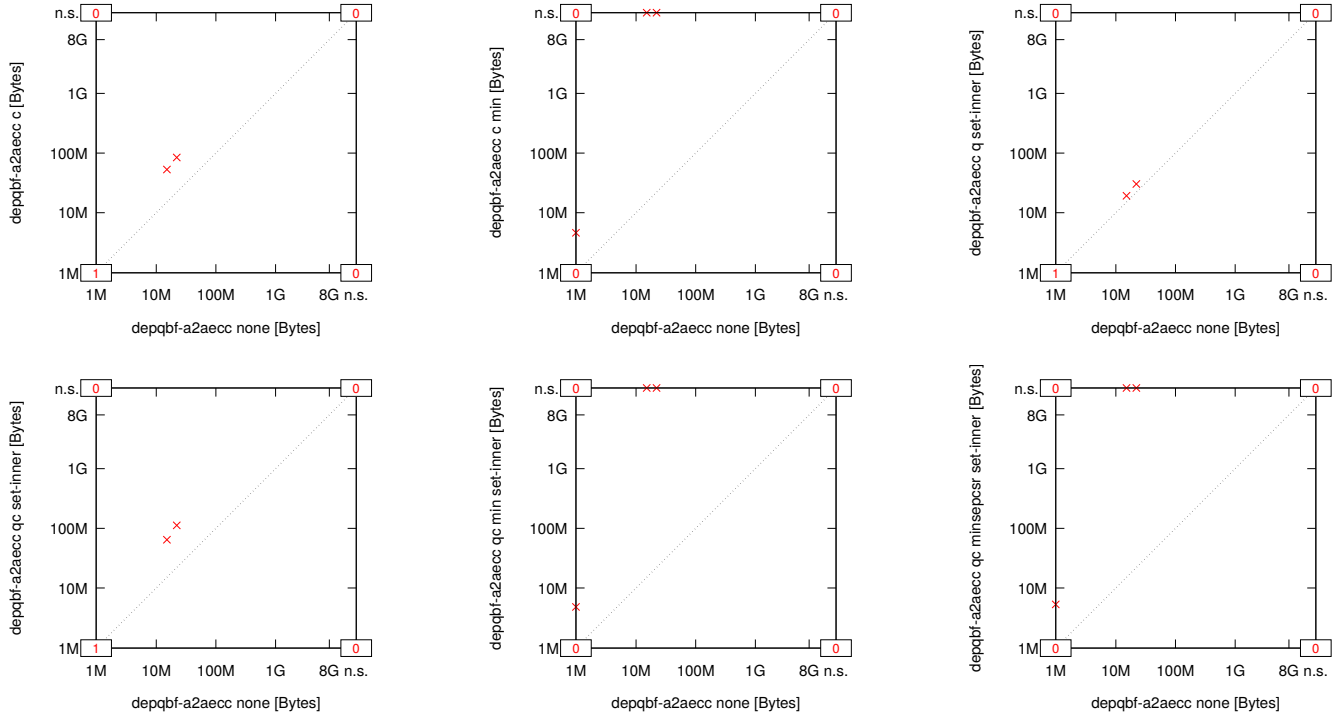


Fig. 870: Suite Ling ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

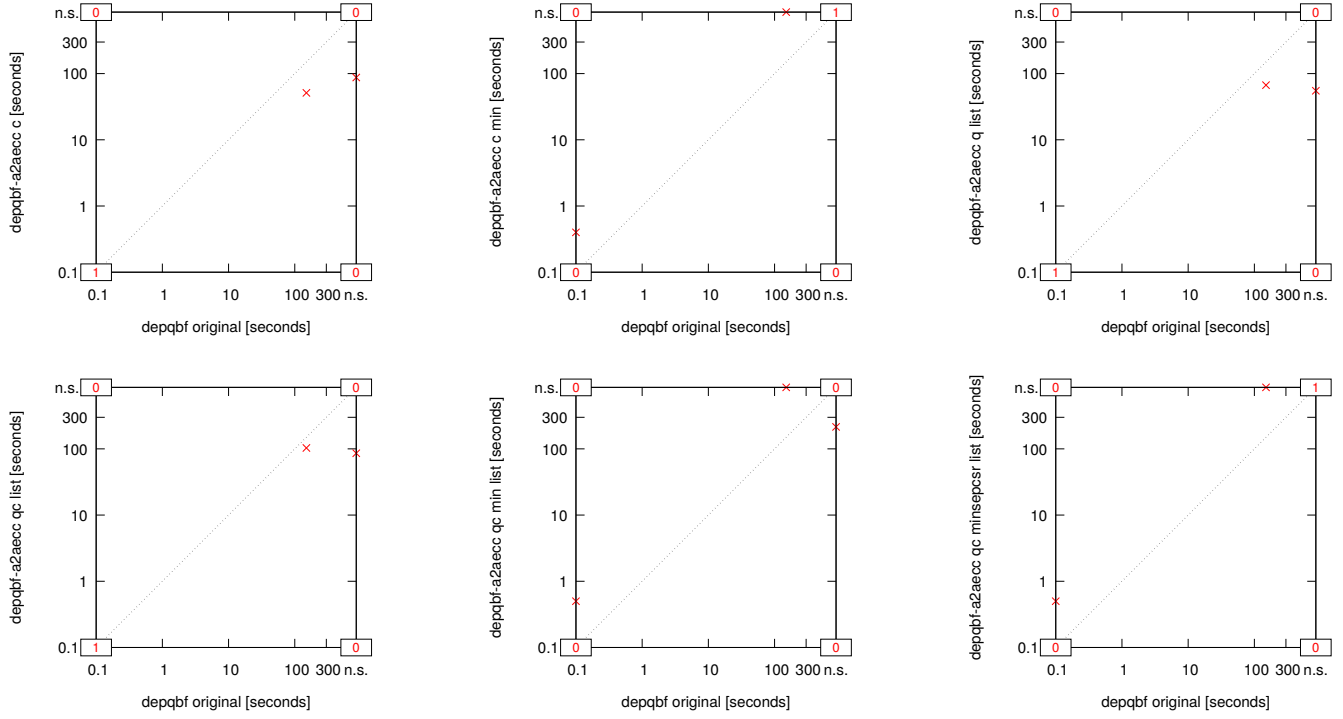


Fig. 871: Suite Ling ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

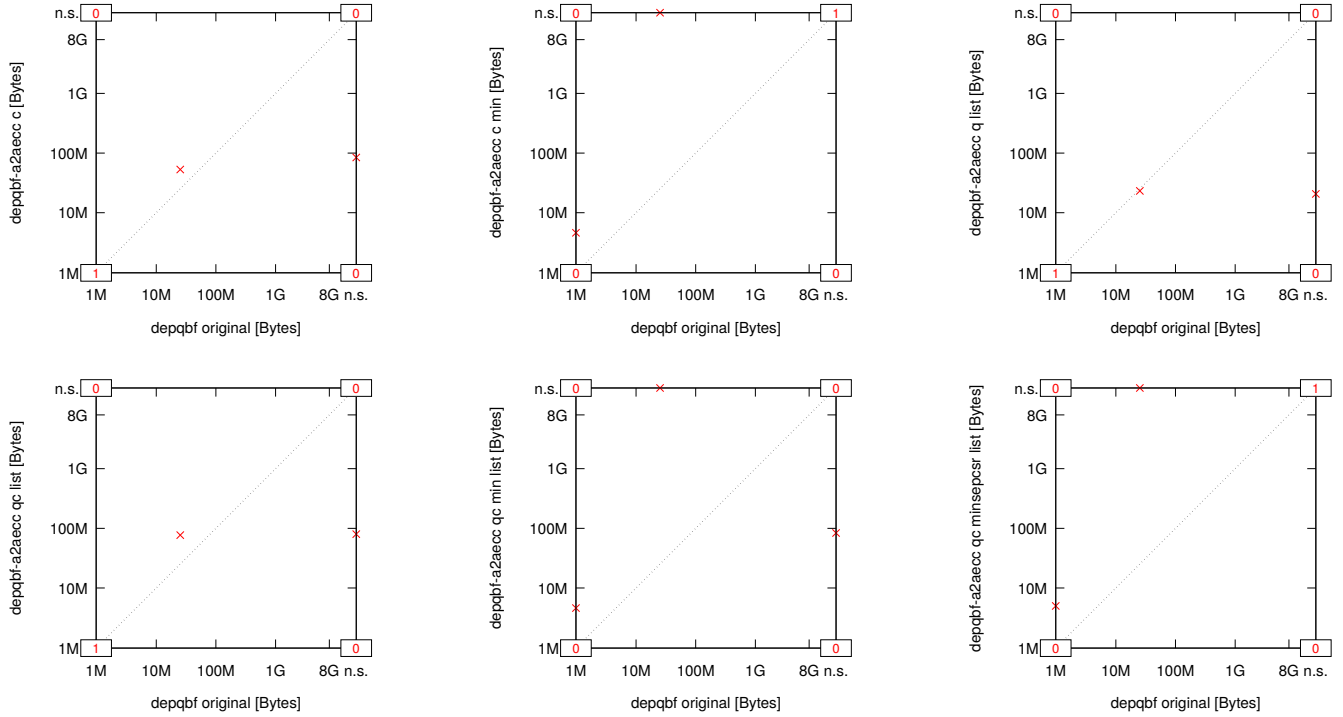


Fig. 872: Suite Ling ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

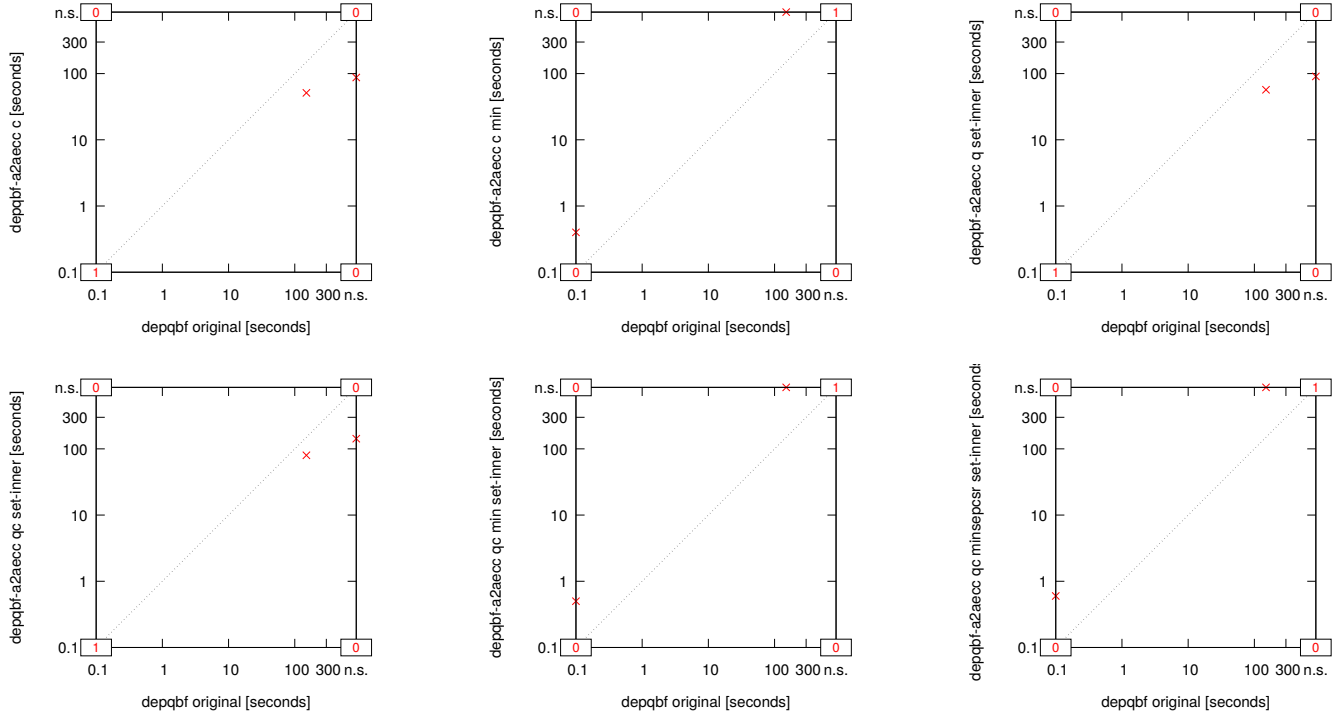


Fig. 873: Suite Ling ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

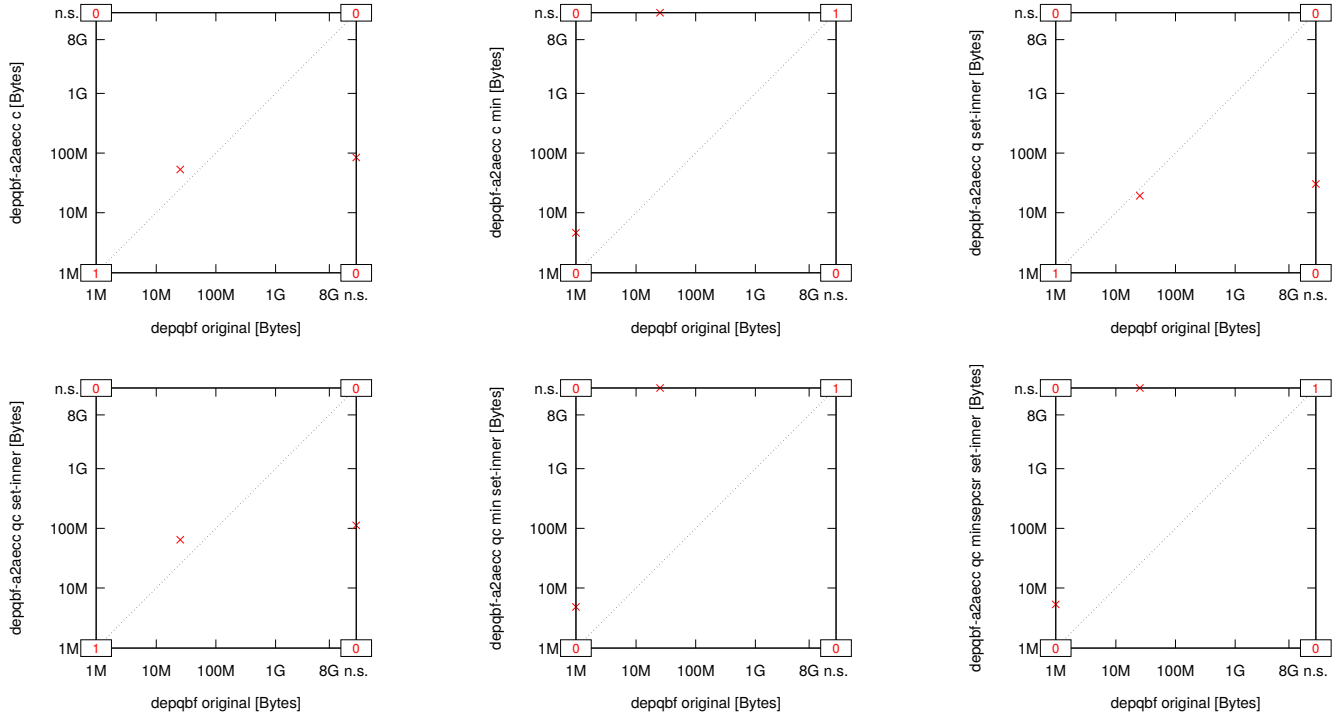


Fig. 874: Suite Ling ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

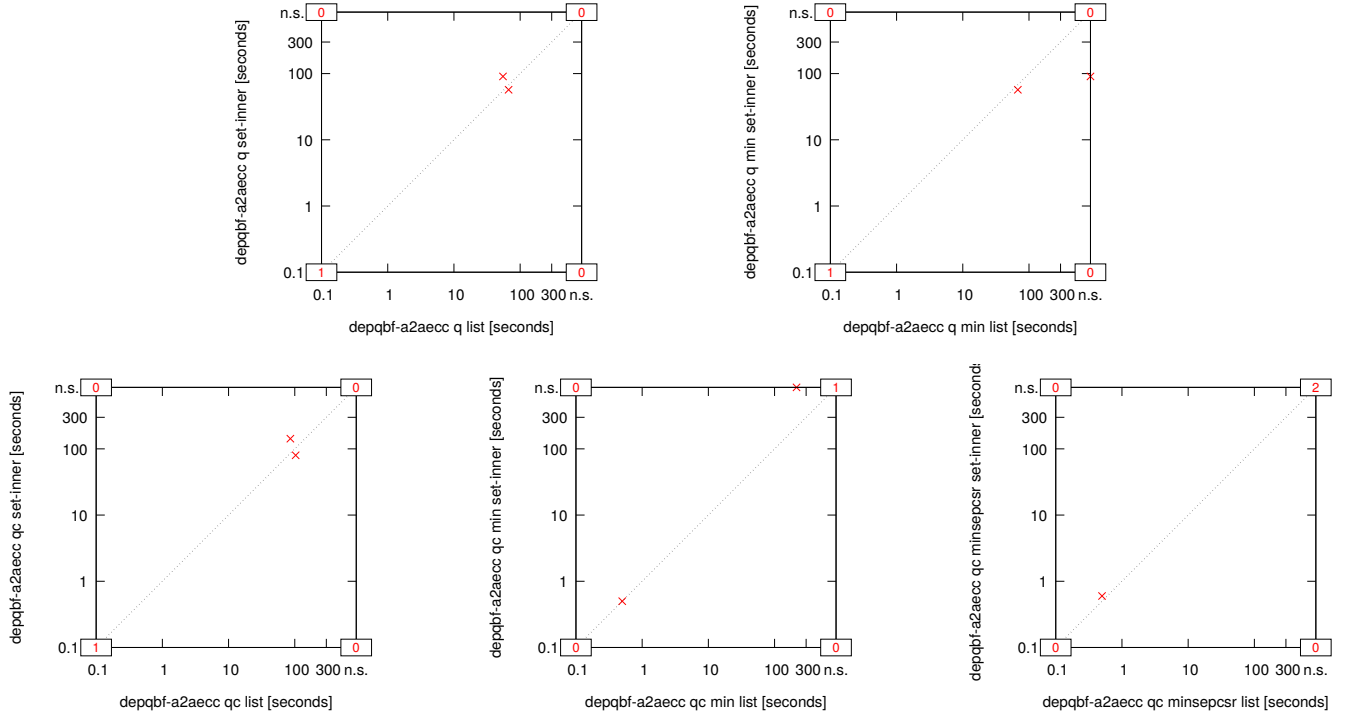


Fig. 875: Suite Ling ($n = 3$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

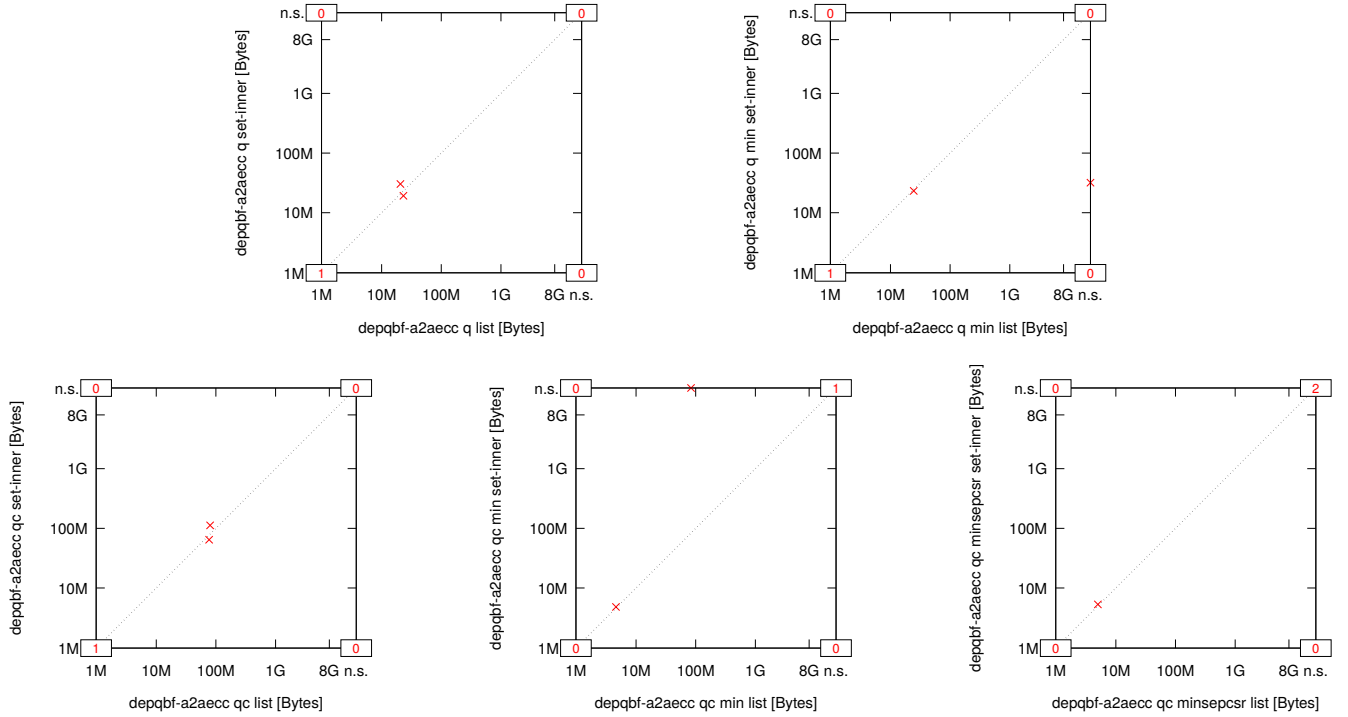


Fig. 876: Suite Ling ($n = 3$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

27) Mangassarian-Veneris ($n = 60$):

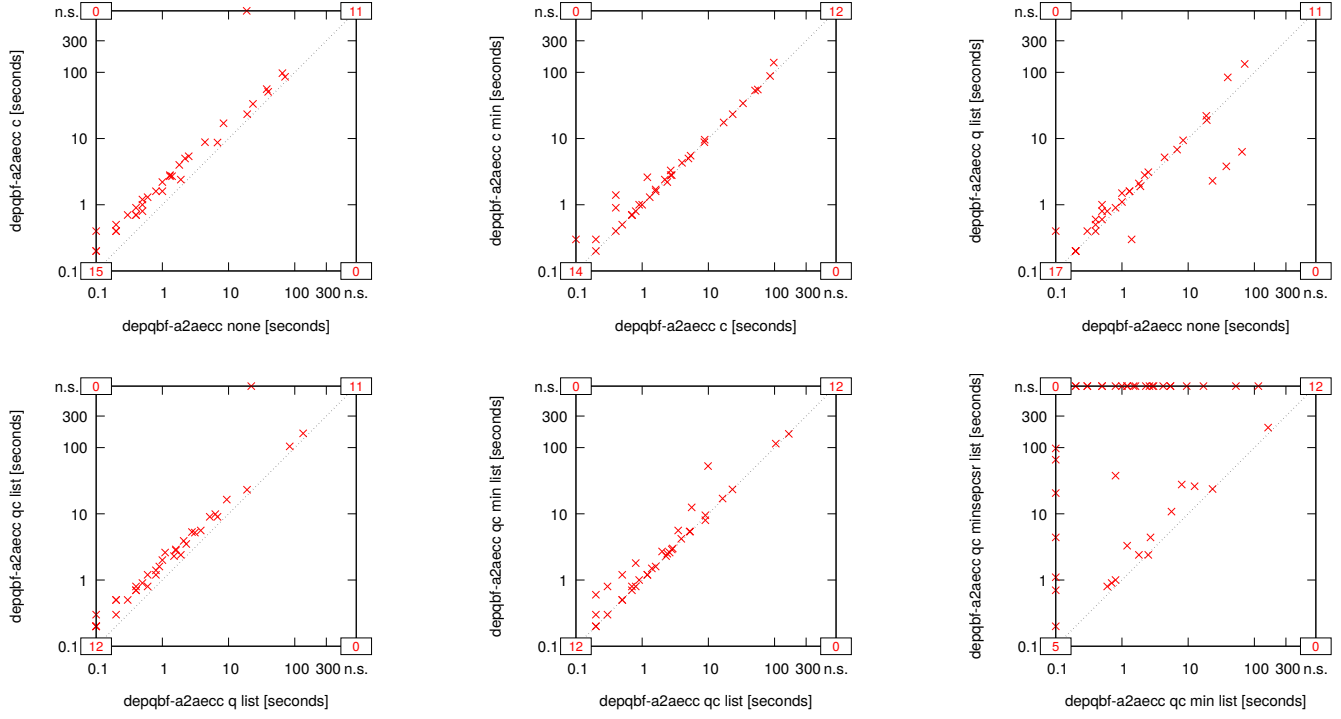


Fig. 877: Suite Mangassarian-Veneris ($n = 60$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

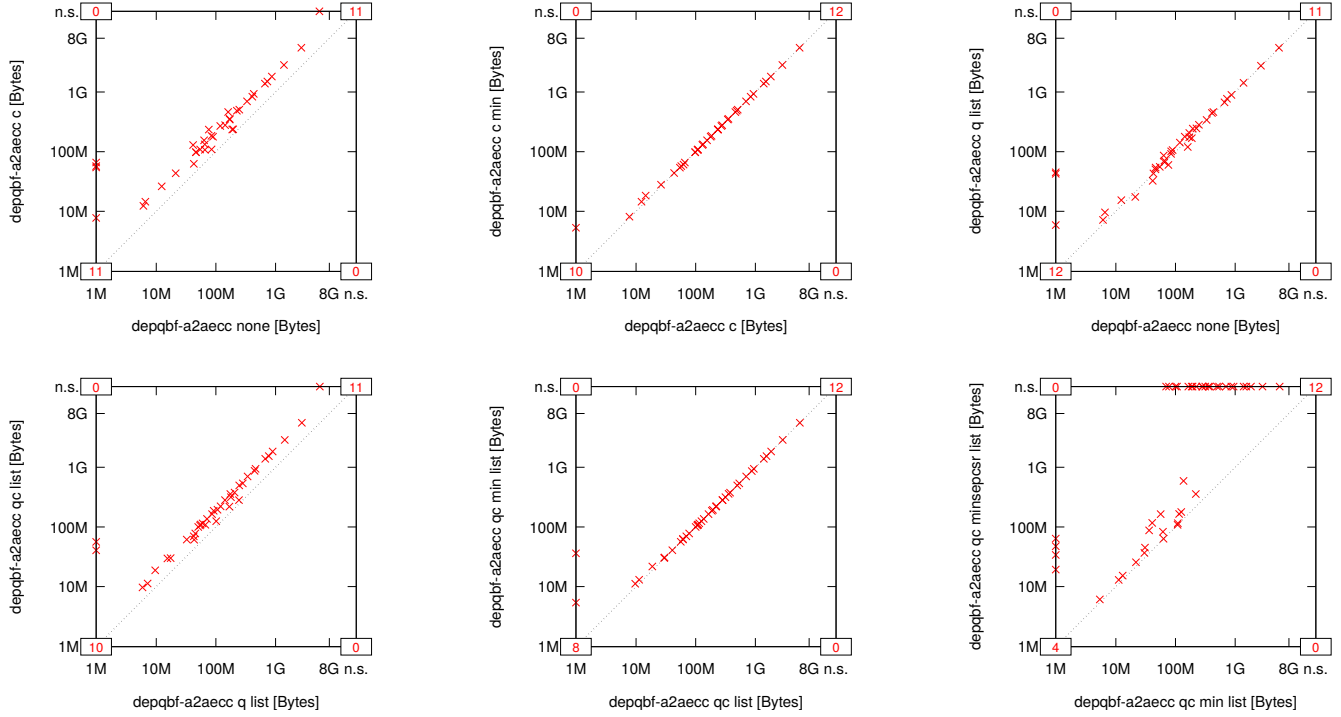


Fig. 878: Suite Mangassarian-Veneris ($n = 60$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

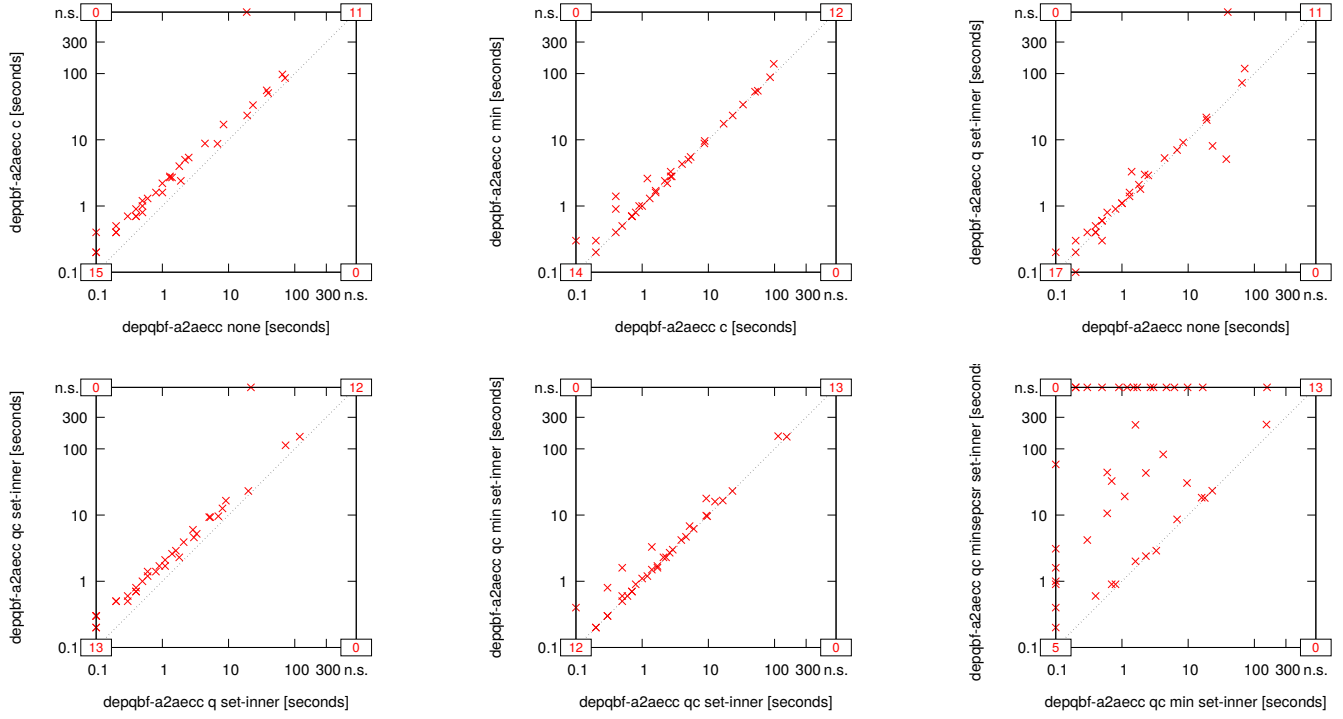


Fig. 879: Suite Mangassarian-Veneris ($n = 60$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

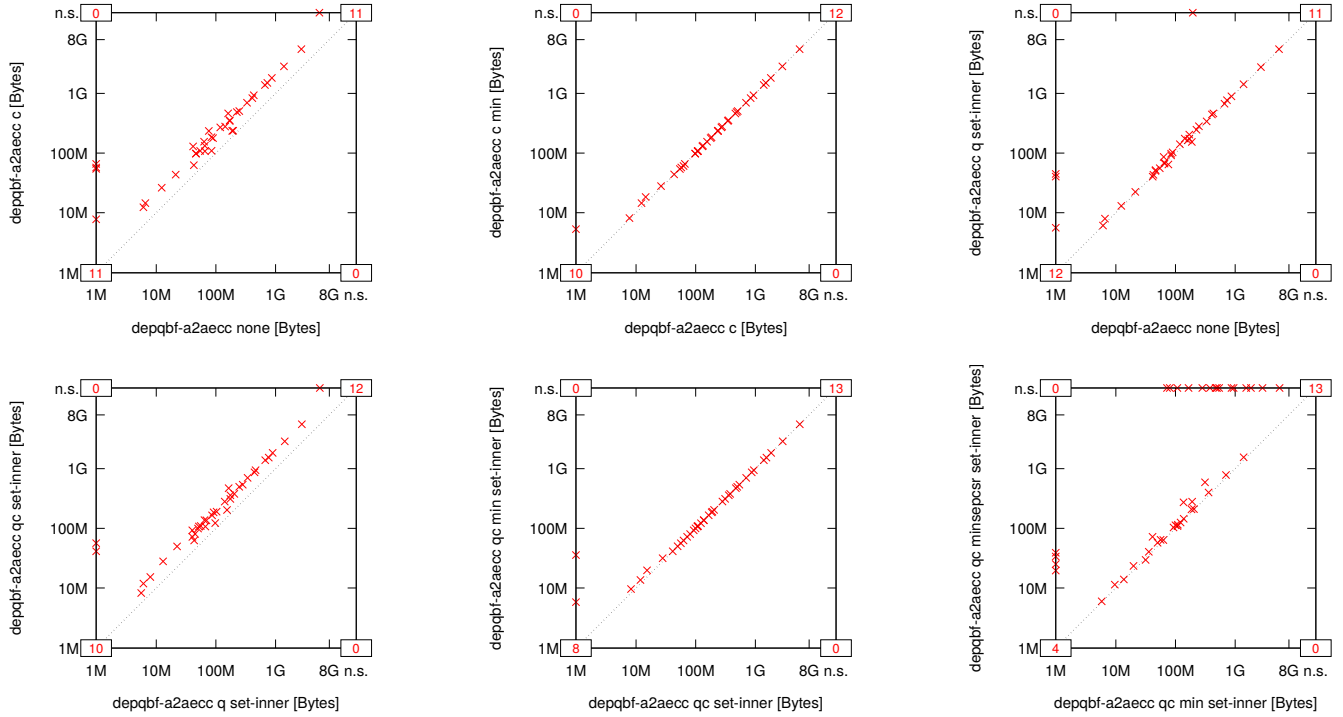


Fig. 880: Suite Mangassarian-Veneris ($n = 60$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

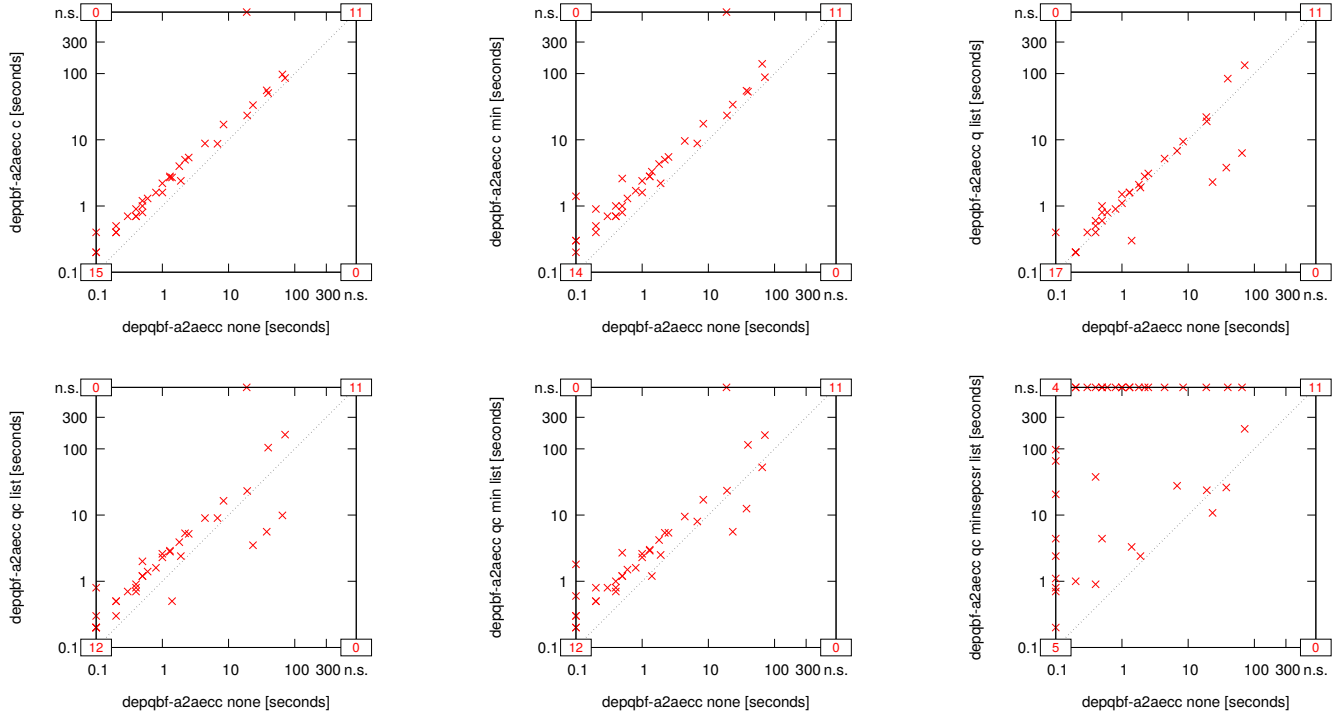


Fig. 881: Suite Mangassarian-Veneris ($n = 60$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. Dep-QBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

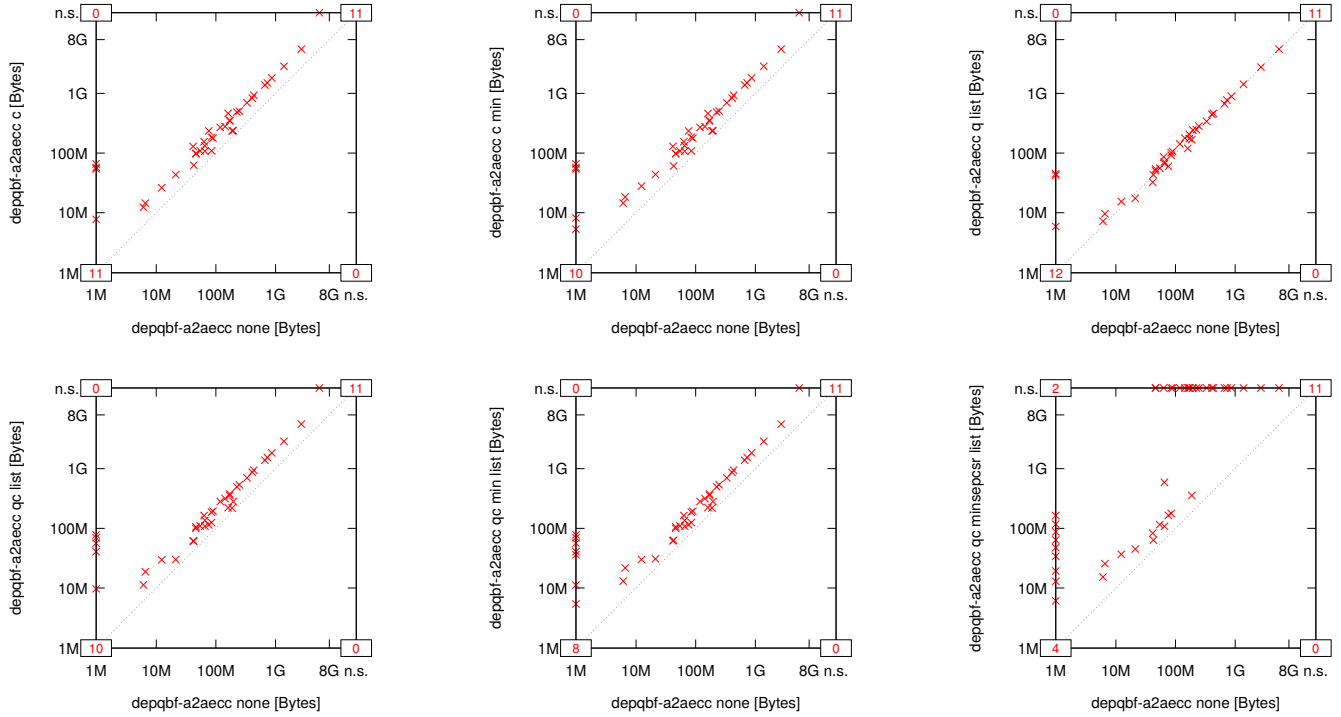


Fig. 882: Suite Mangassarian-Veneris ($n = 60$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. Dep-QBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

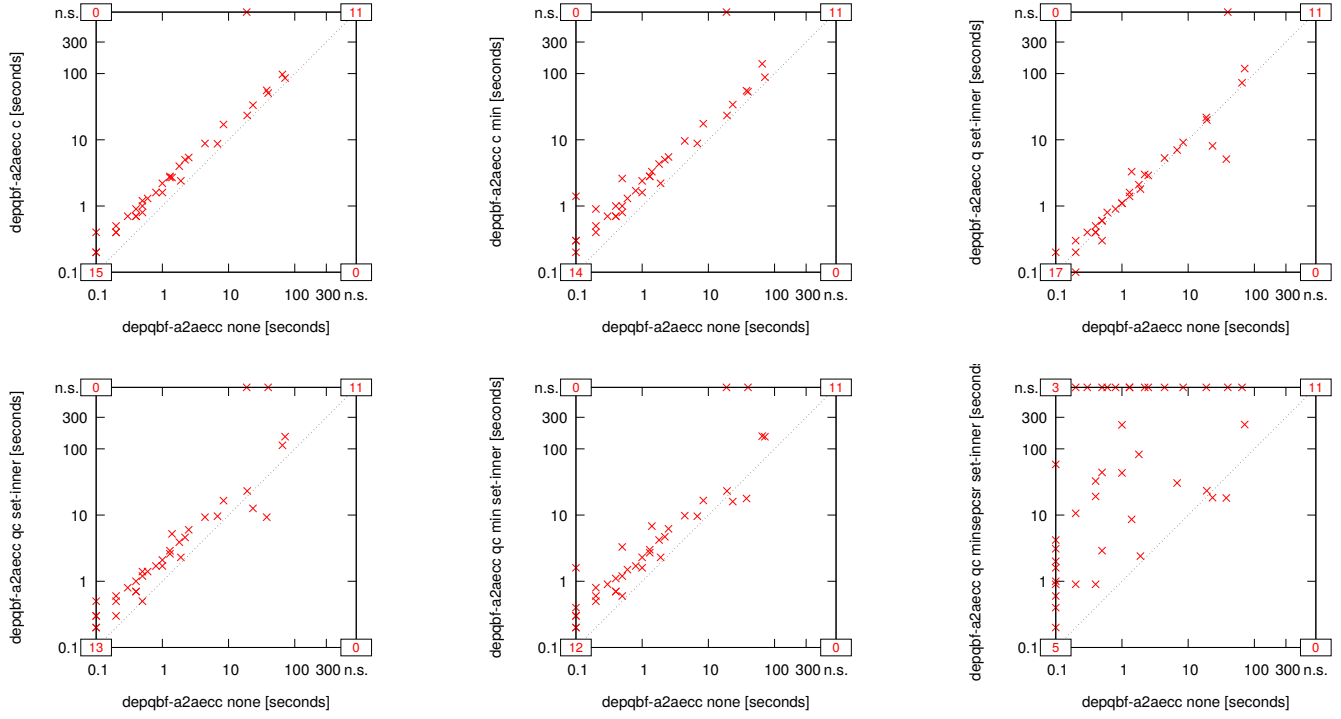


Fig. 883: Suite Mangassarian-Veneris ($n = 60$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

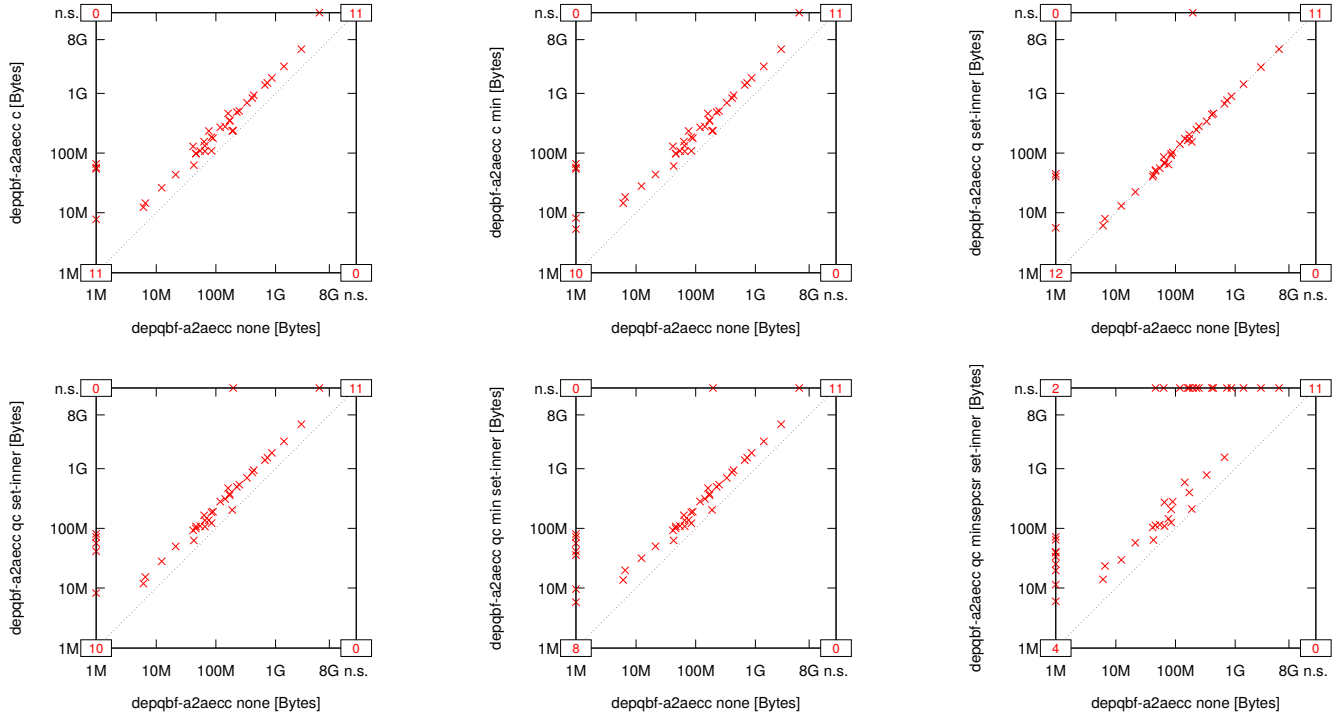


Fig. 884: Suite Mangassarian-Veneris ($n = 60$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

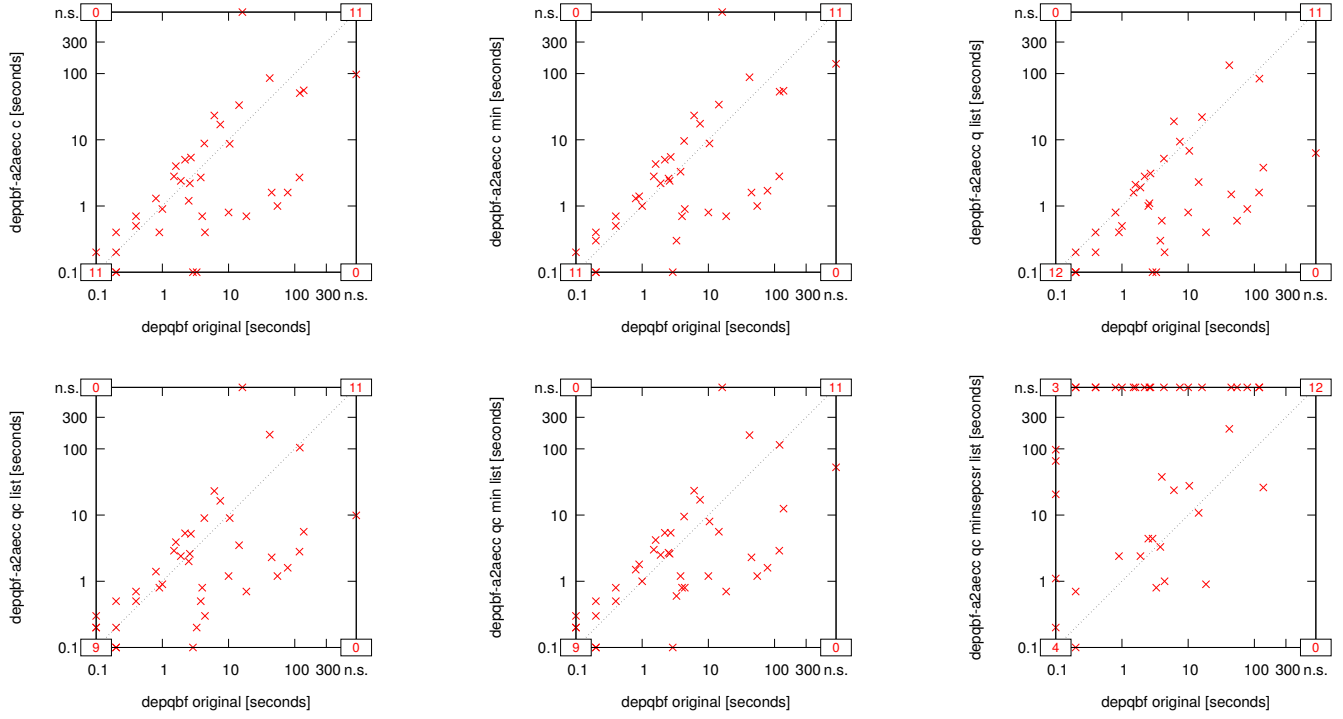


Fig. 885: Suite Mangassarian-Veneris ($n = 60$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

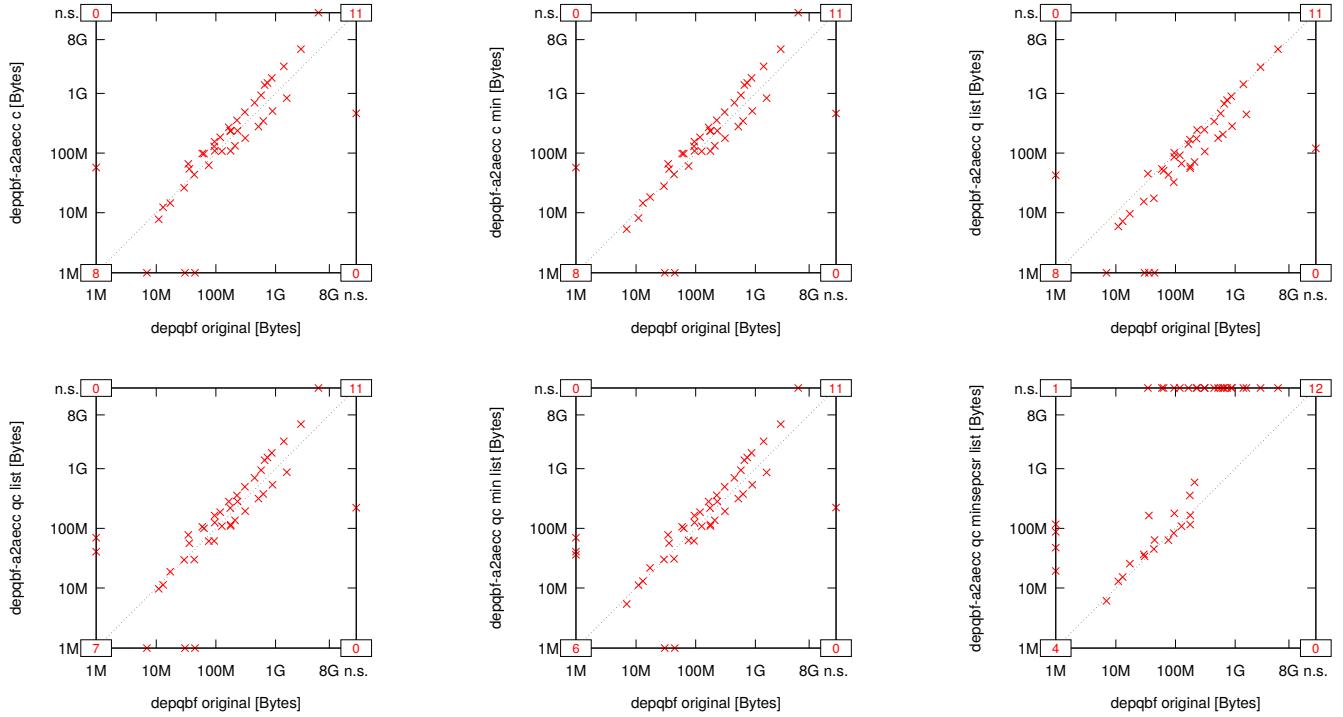


Fig. 886: Suite Mangassarian-Veneris ($n = 60$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

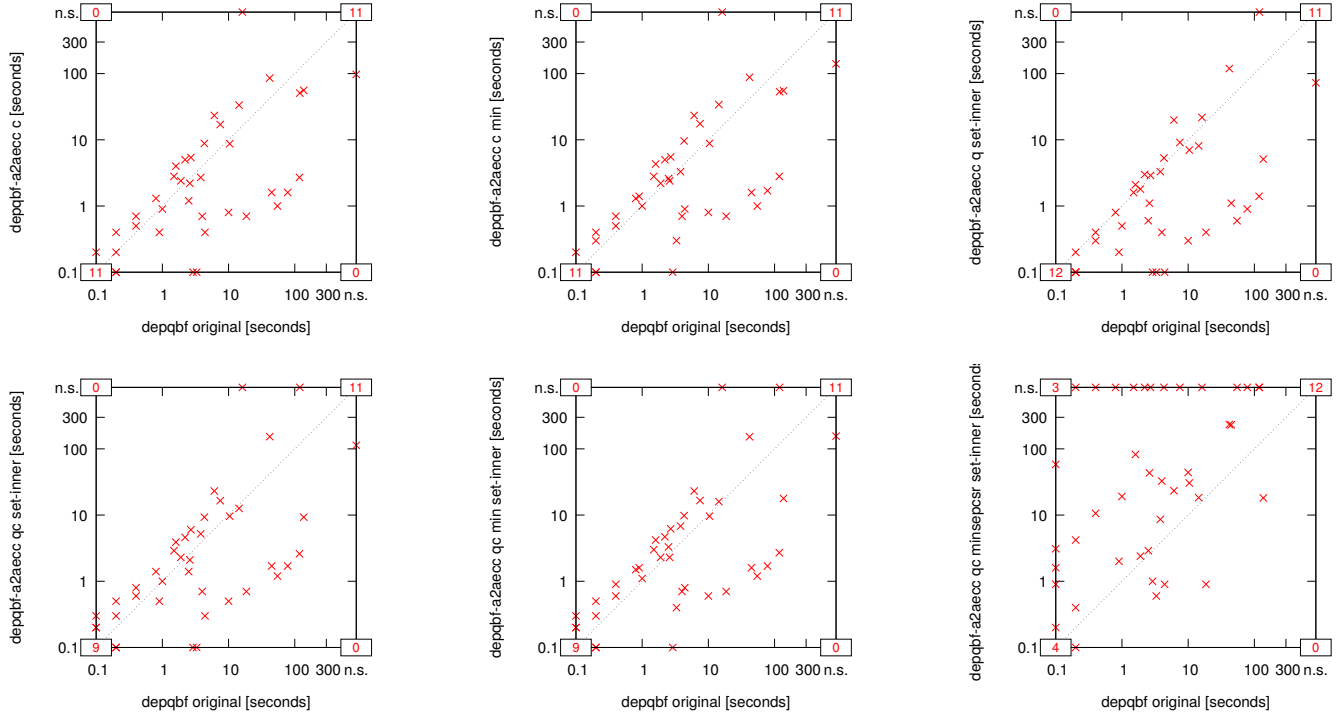


Fig. 887: Suite Mangassarian-Veneris ($n = 60$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

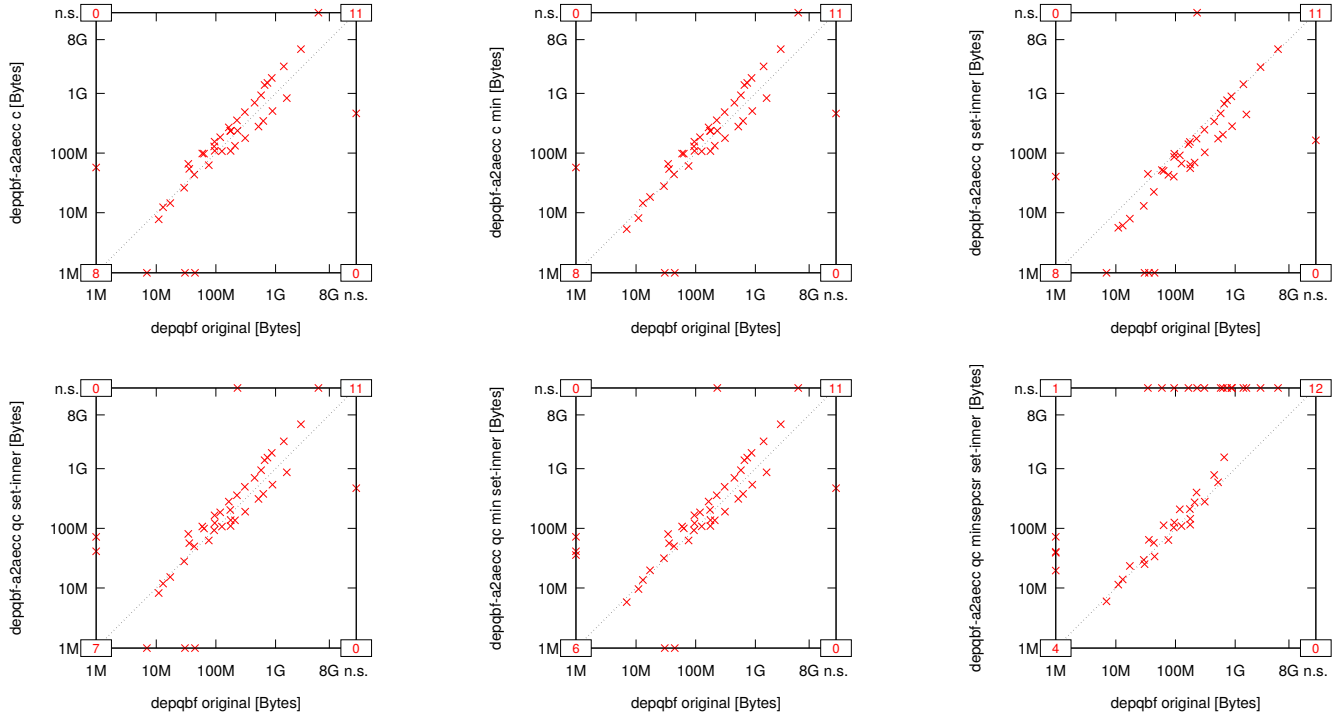


Fig. 888: Suite Mangassarian-Veneris ($n = 60$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

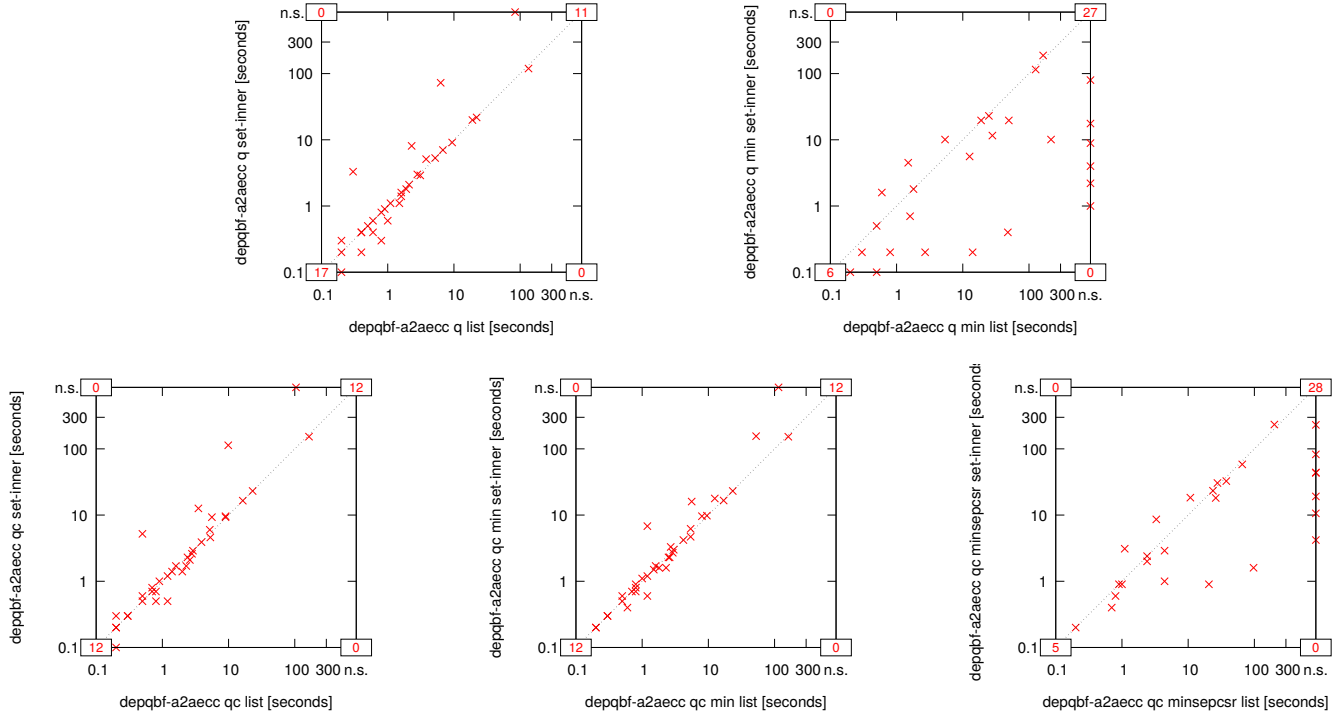


Fig. 889: Suite Mangassarian-Veneris ($n = 60$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

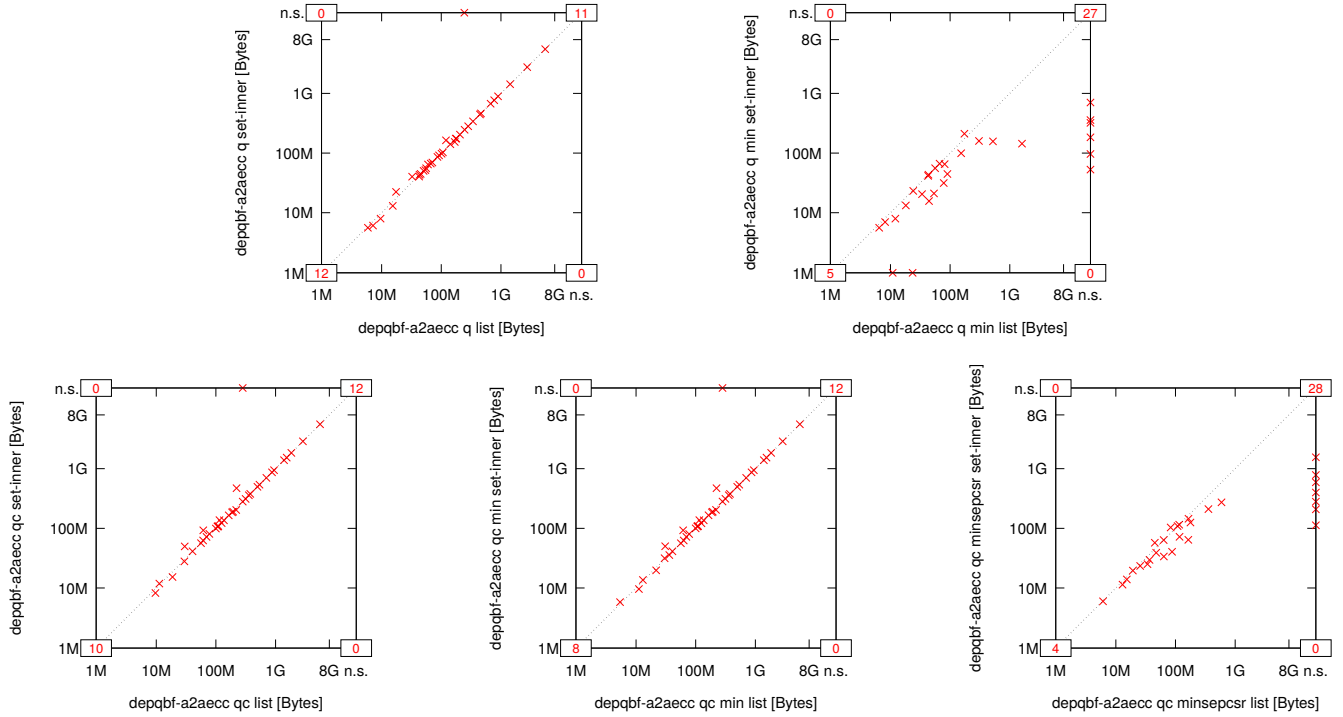


Fig. 890: Suite Mangassarian-Veneris ($n = 60$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

28) MayerEichberger-Saffidine ($n = 3$):

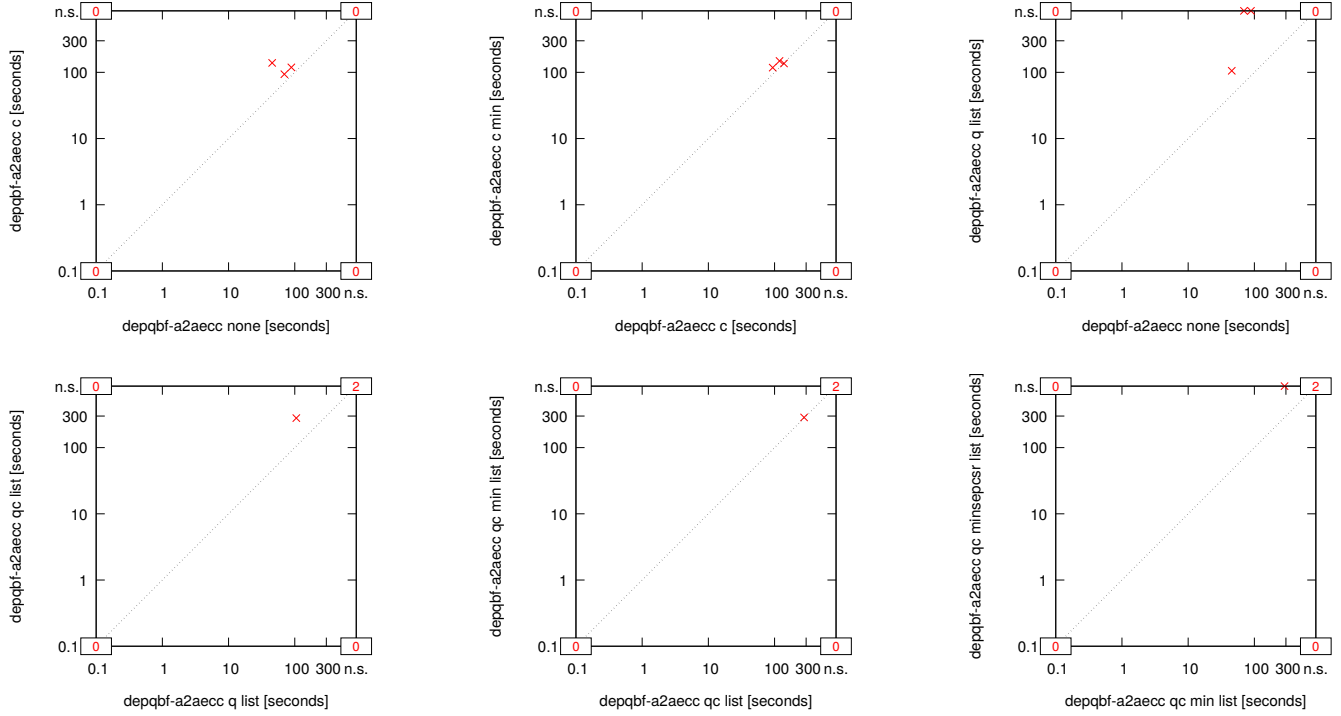


Fig. 891: Suite MayerEichberger-Saffidine ($n = 3$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

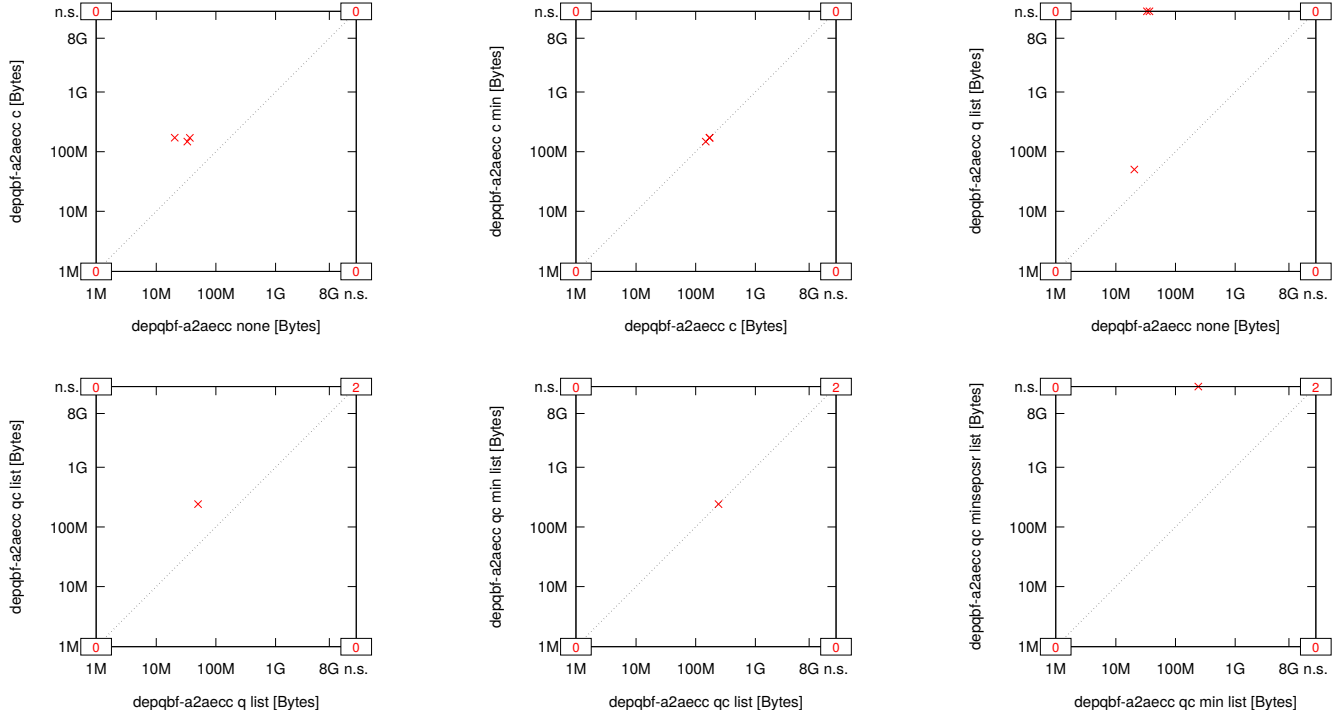


Fig. 892: Suite MayerEichberger-Saffidine ($n = 3$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

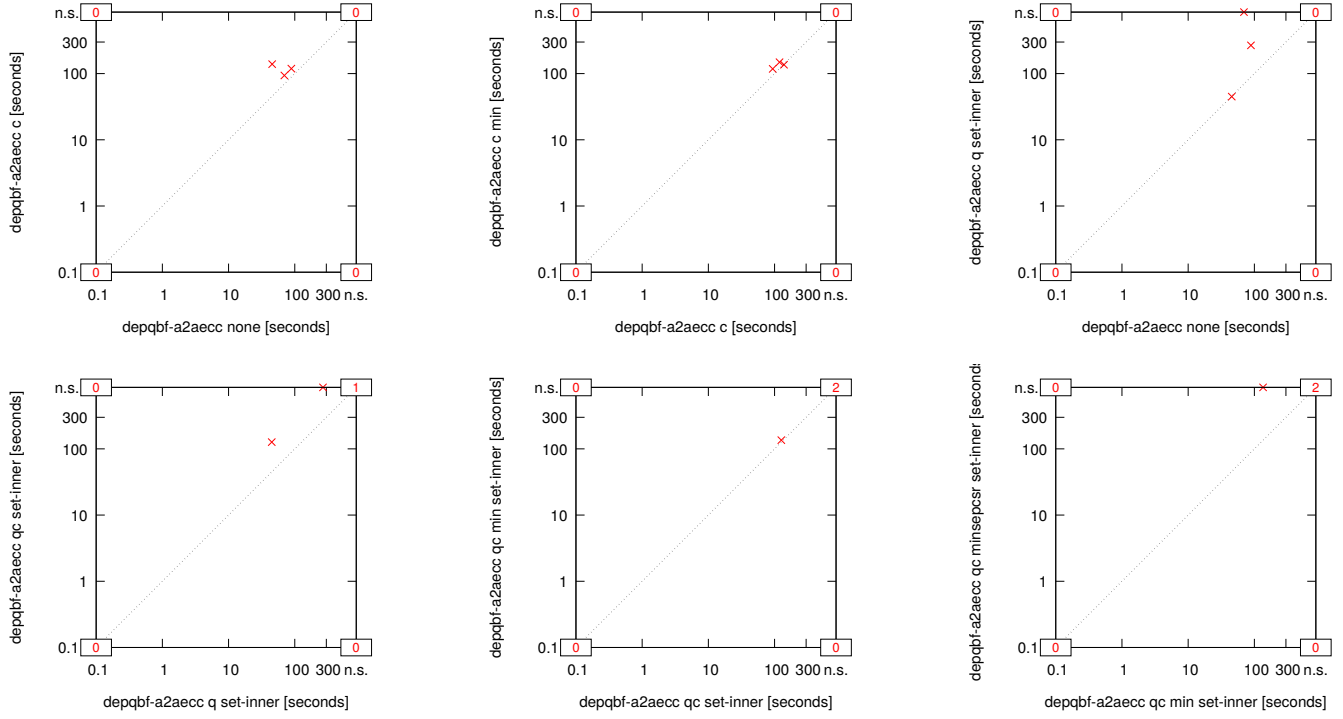


Fig. 893: Suite MayerEichberger-Saffidine ($n = 3$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

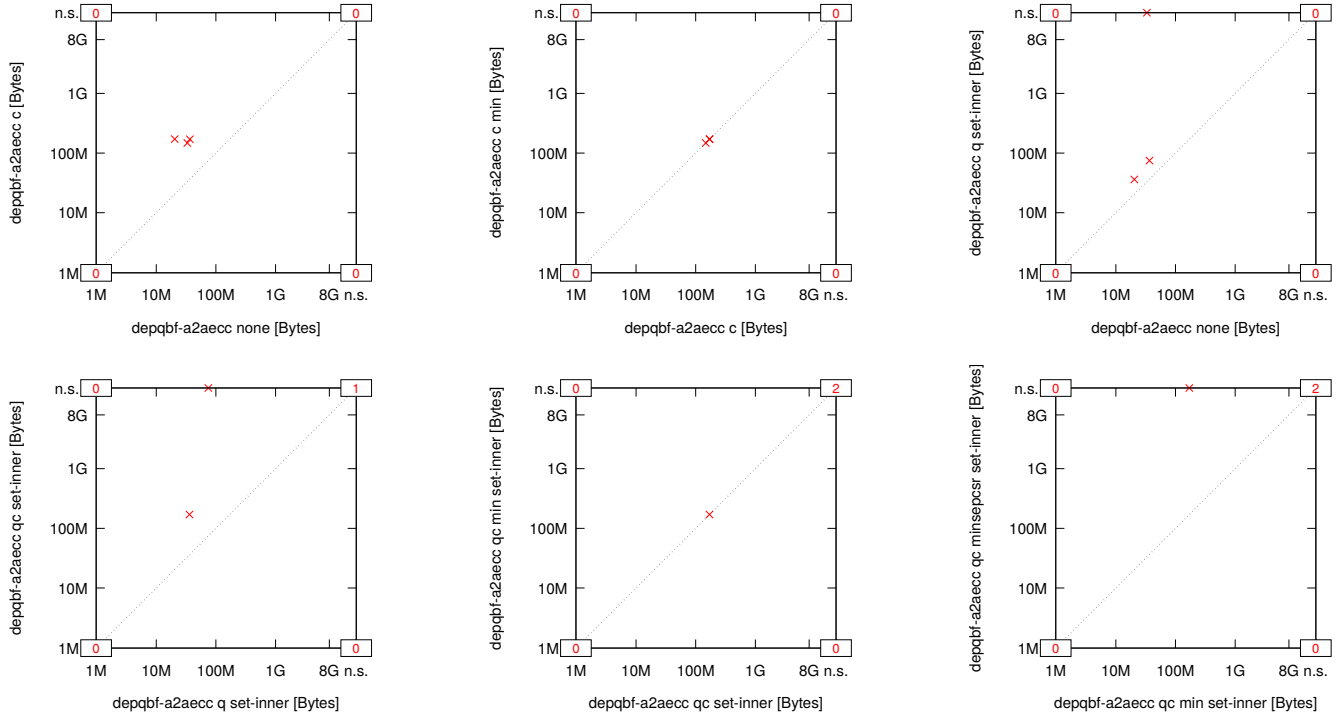


Fig. 894: Suite MayerEichberger-Saffidine ($n = 3$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

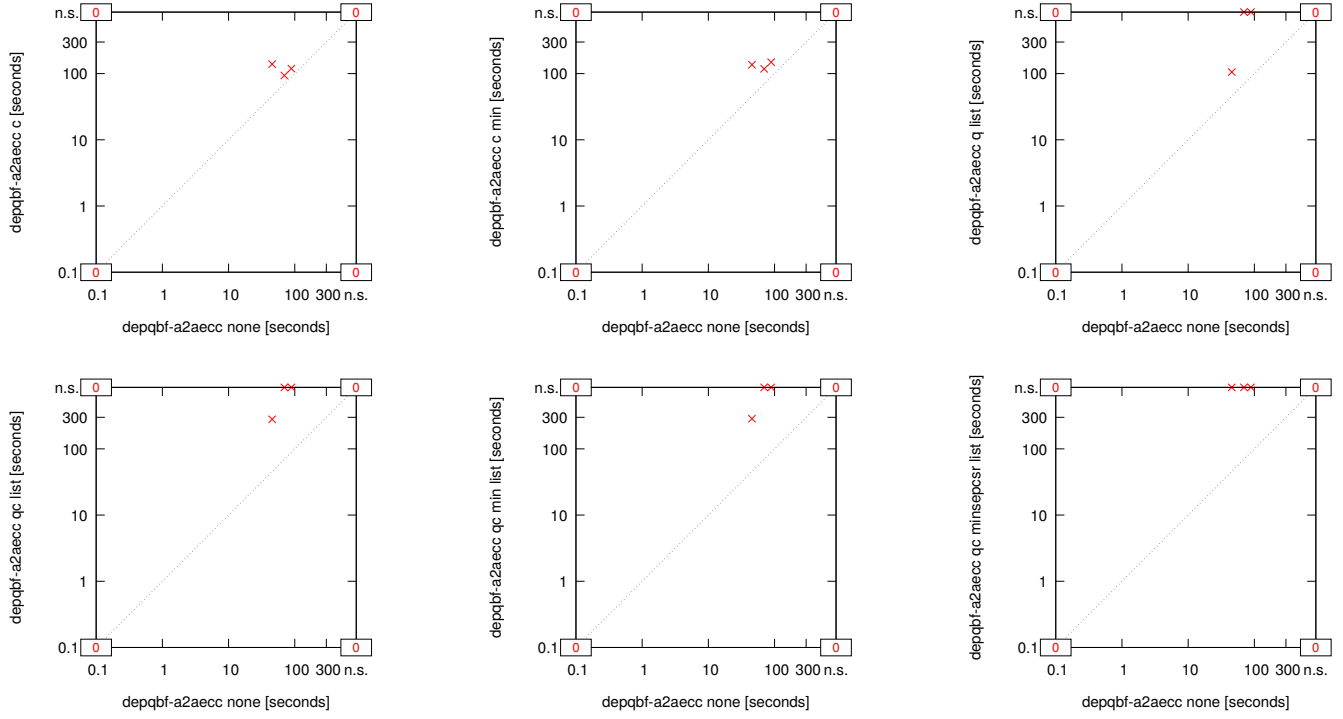


Fig. 895: Suite MayerEichberger-Saffidine ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. Dep-QBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

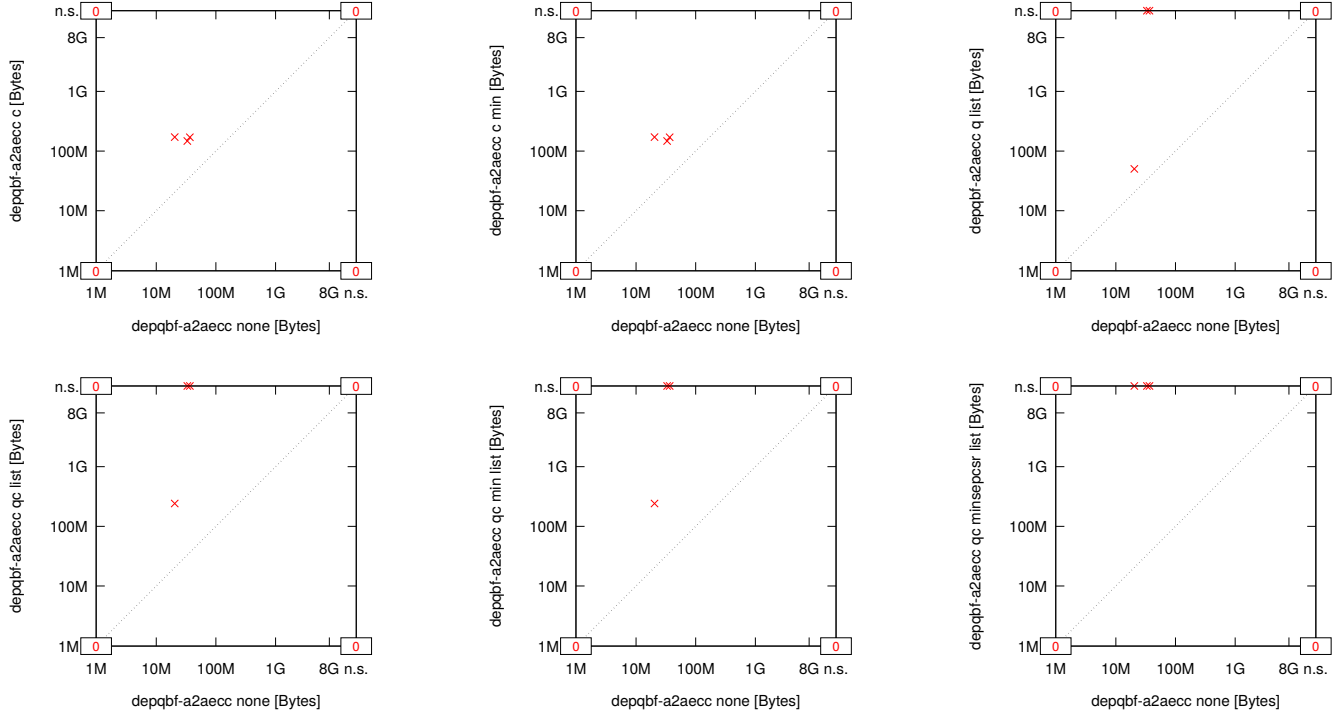


Fig. 896: Suite MayerEichberger-Saffidine ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. Dep-QBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

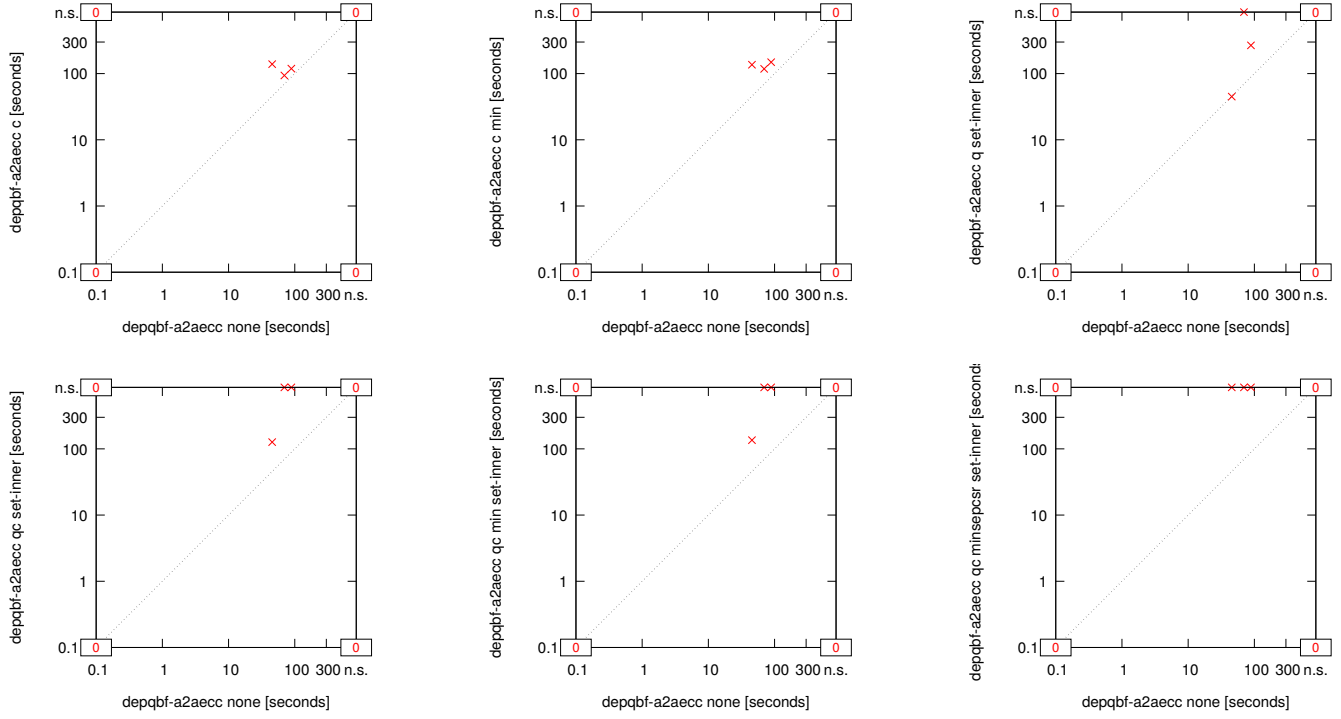


Fig. 897: Suite MayerEichberger-Saffidine ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

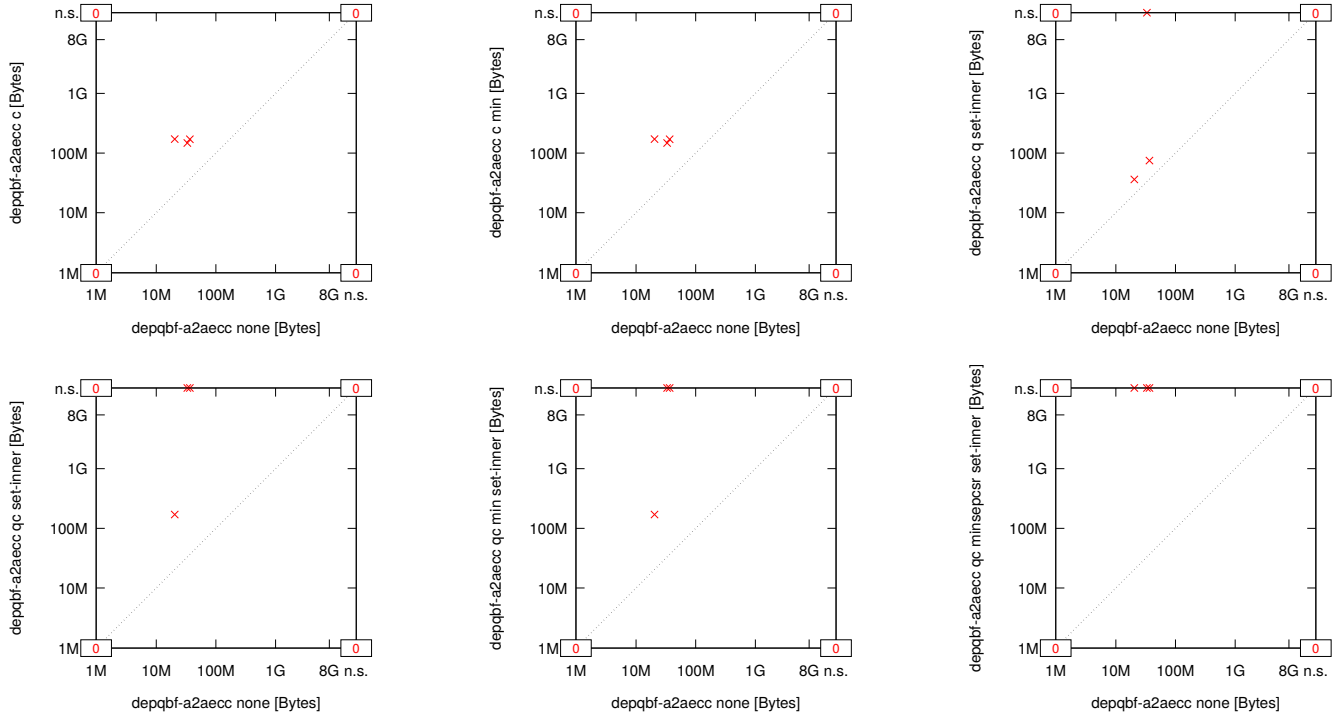


Fig. 898: Suite MayerEichberger-Saffidine ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

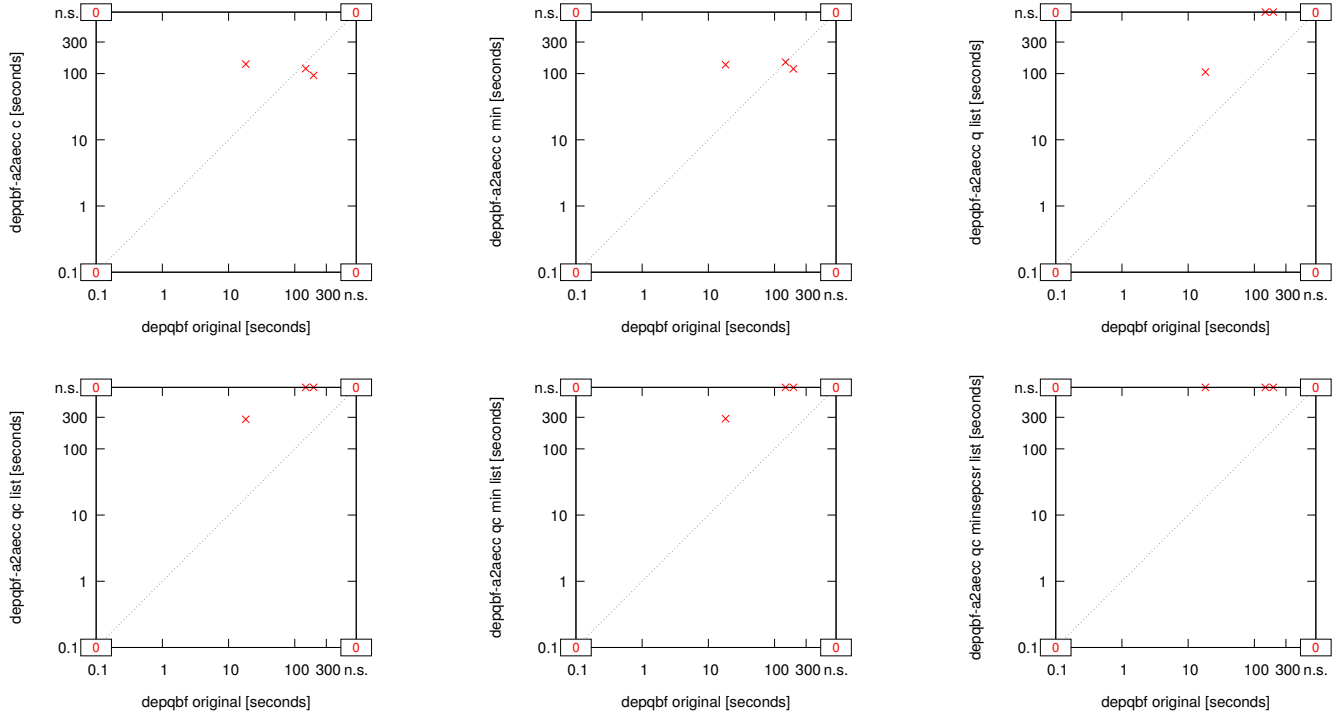


Fig. 899: Suite MayerEichberger-Saffidine ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

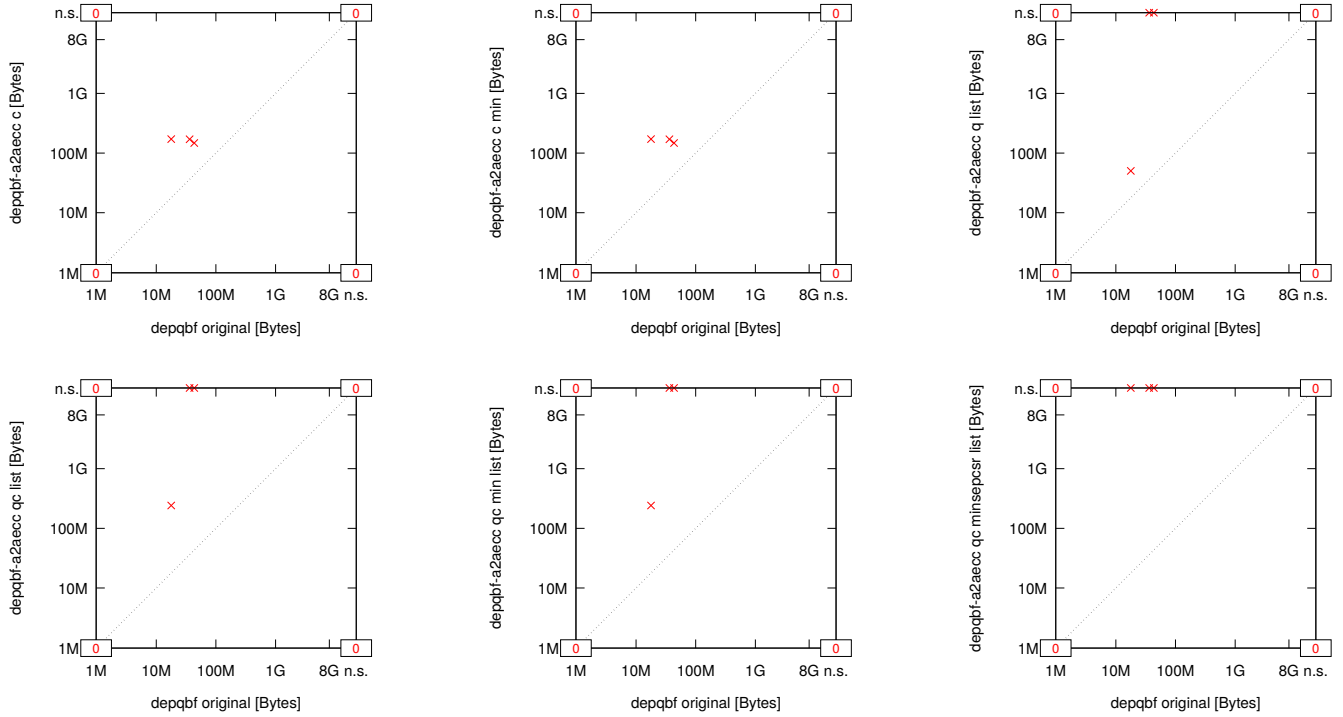


Fig. 900: Suite MayerEichberger-Saffidine ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

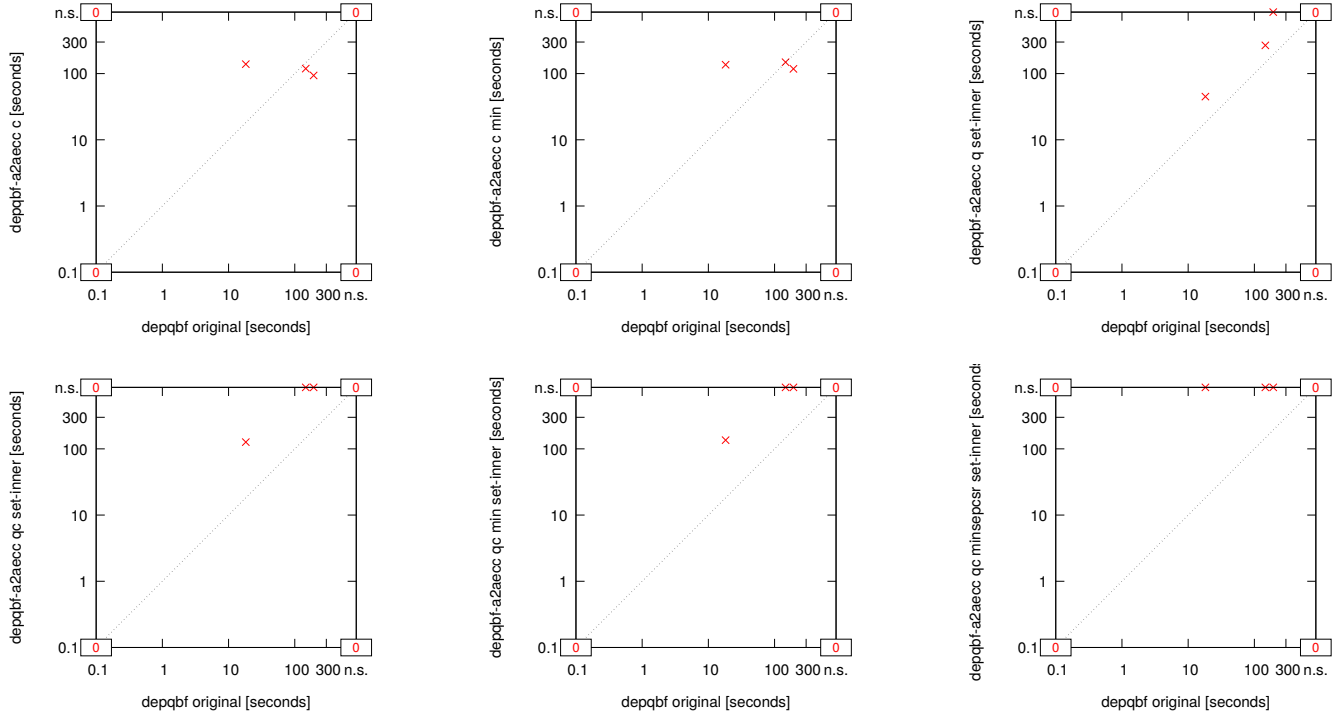


Fig. 901: Suite MayerEichberger-Saffidine ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

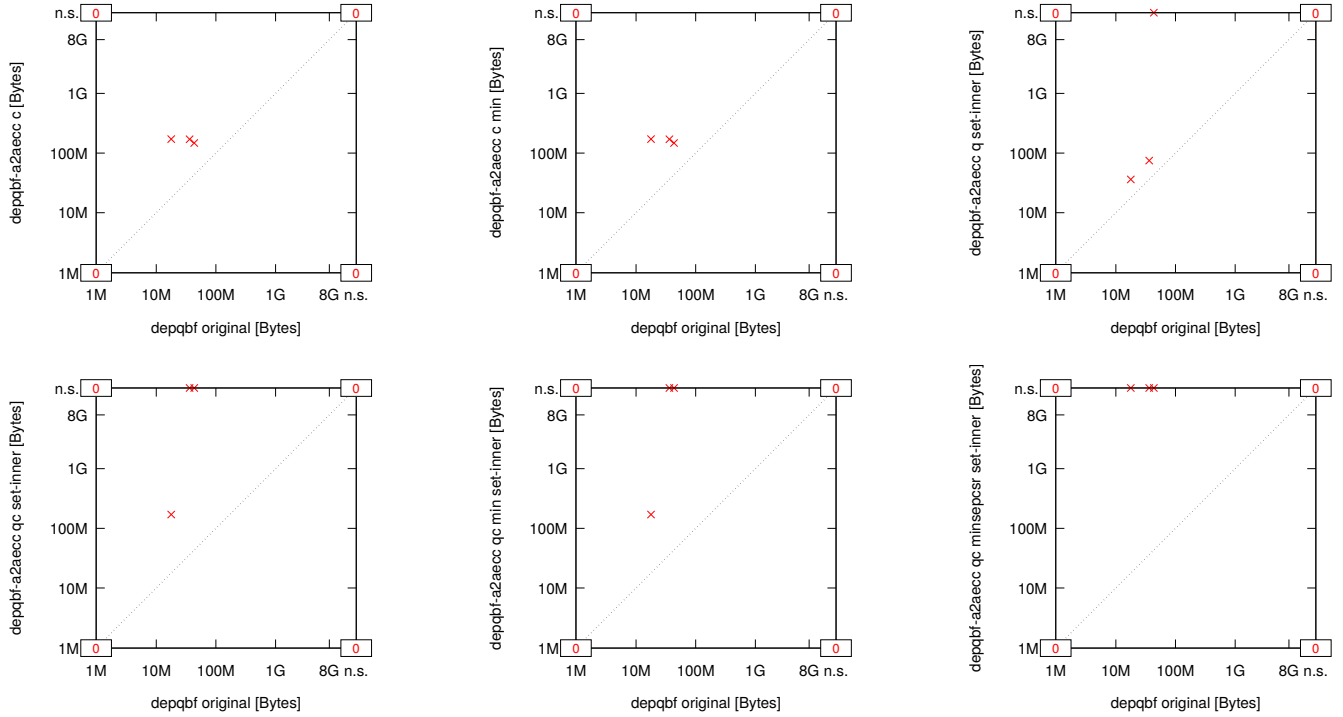


Fig. 902: Suite MayerEichberger-Saffidine ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

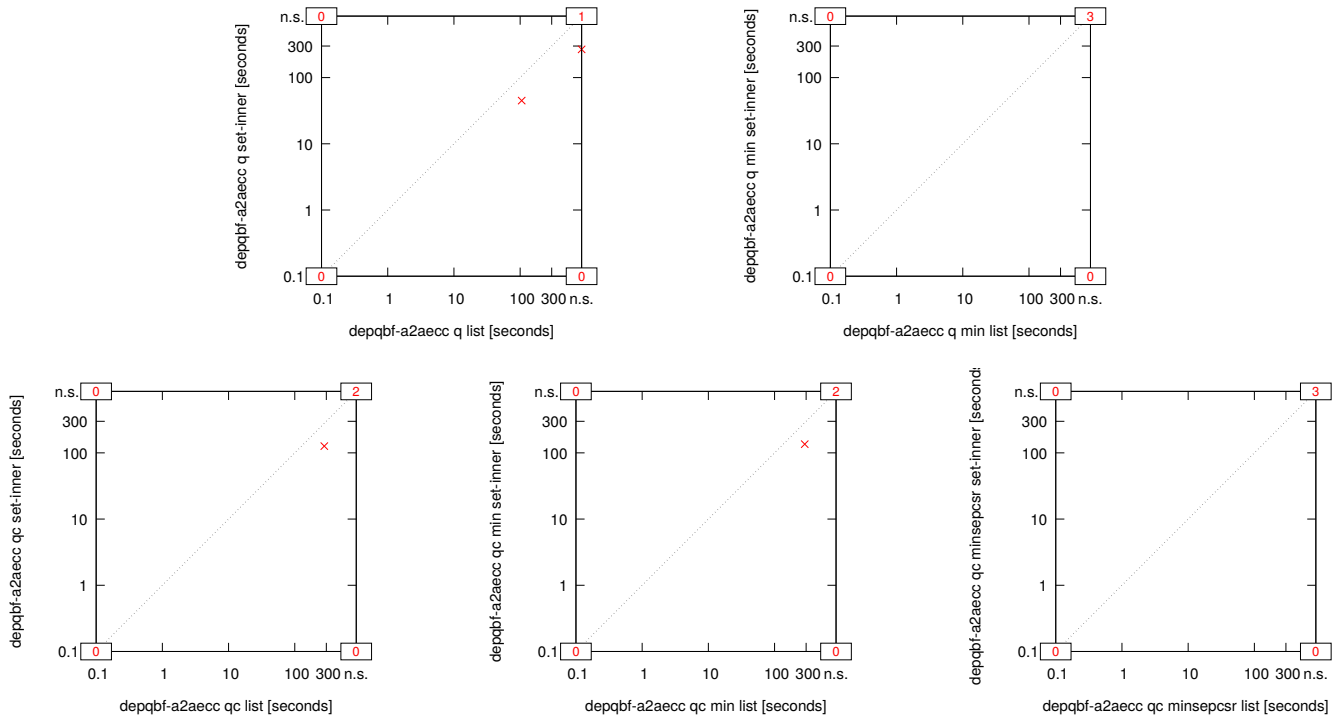


Fig. 903: Suite MayerEichberger-Saffidine ($n = 3$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

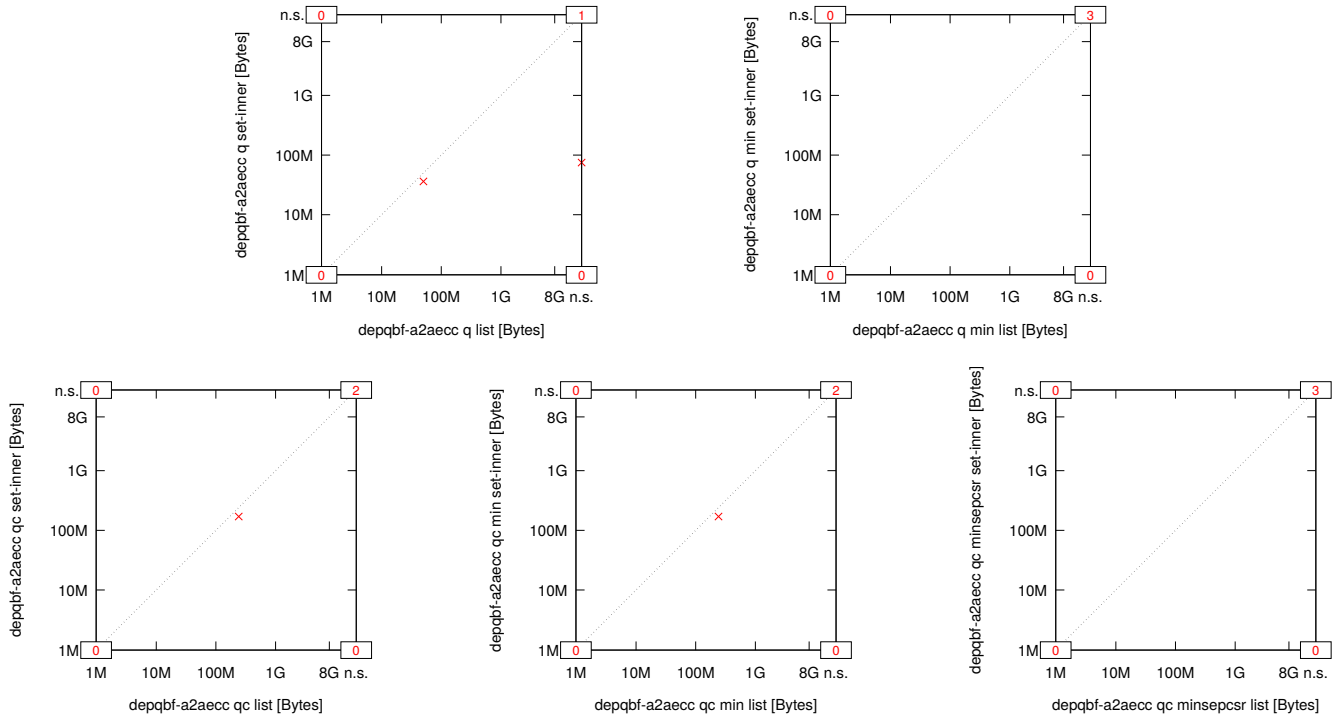


Fig. 904: Suite MayerEichberger-Saffidine ($n = 3$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

29) *Messenger* ($n = 0$):

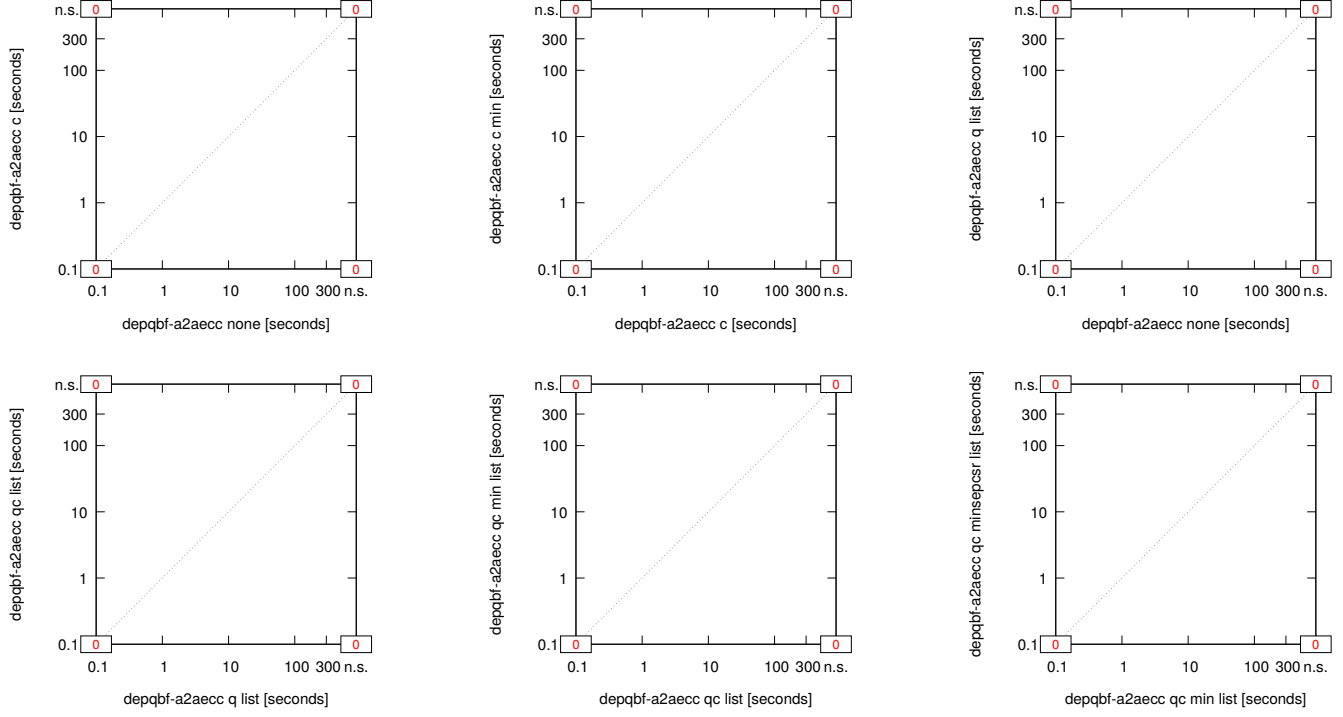


Fig. 905: Suite *Messenger* ($n = 0$): Comparing run times for extracting different unsatisfiable cores in *DepQBF-a2aecc* with list semantics (run time in seconds).

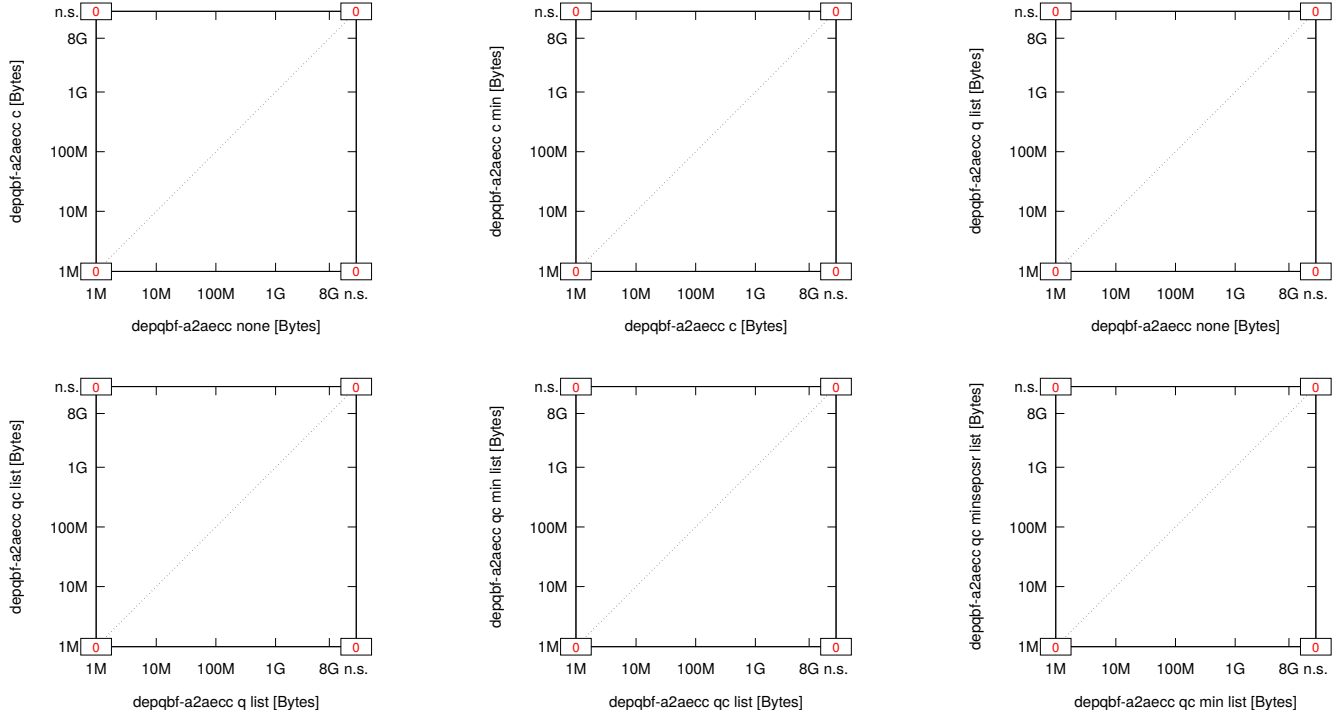


Fig. 906: Suite *Messenger* ($n = 0$): Comparing memory usage for extracting different unsatisfiable cores in *DepQBF-a2aecc* with list semantics (memory usage in Bytes).

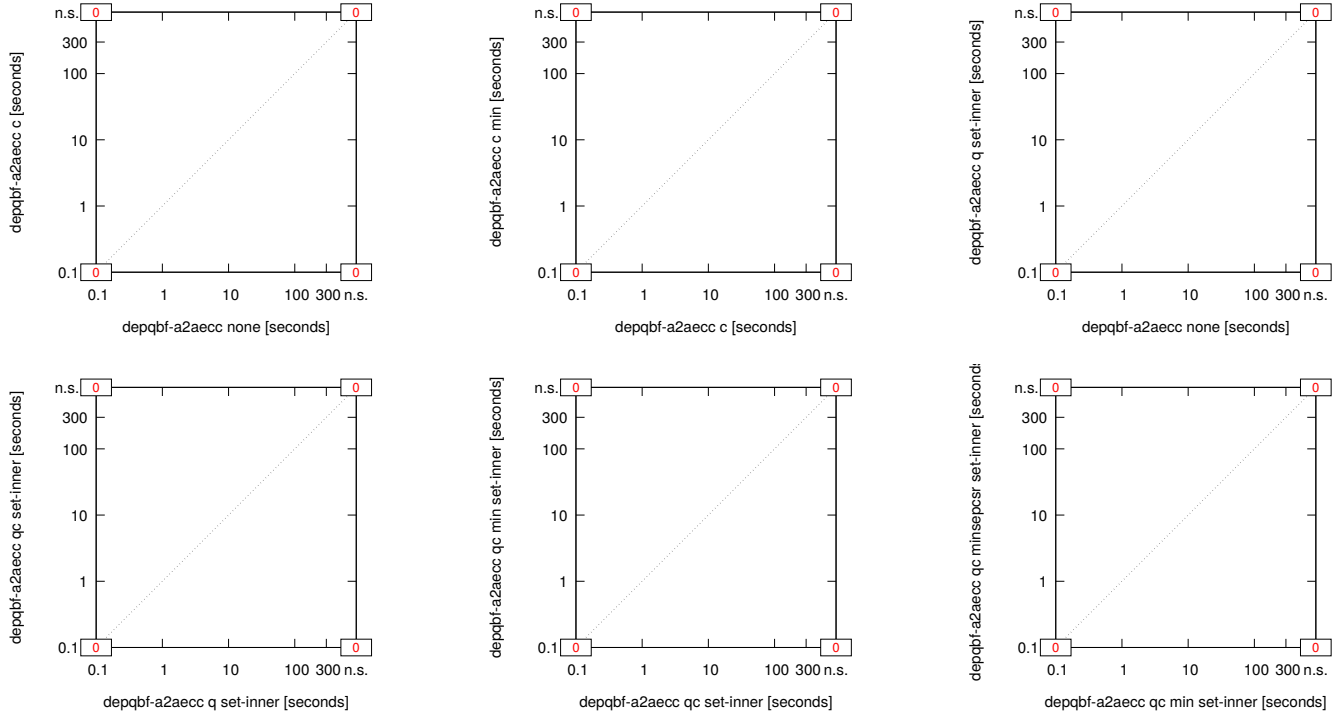


Fig. 907: Suite Messinger ($n = 0$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

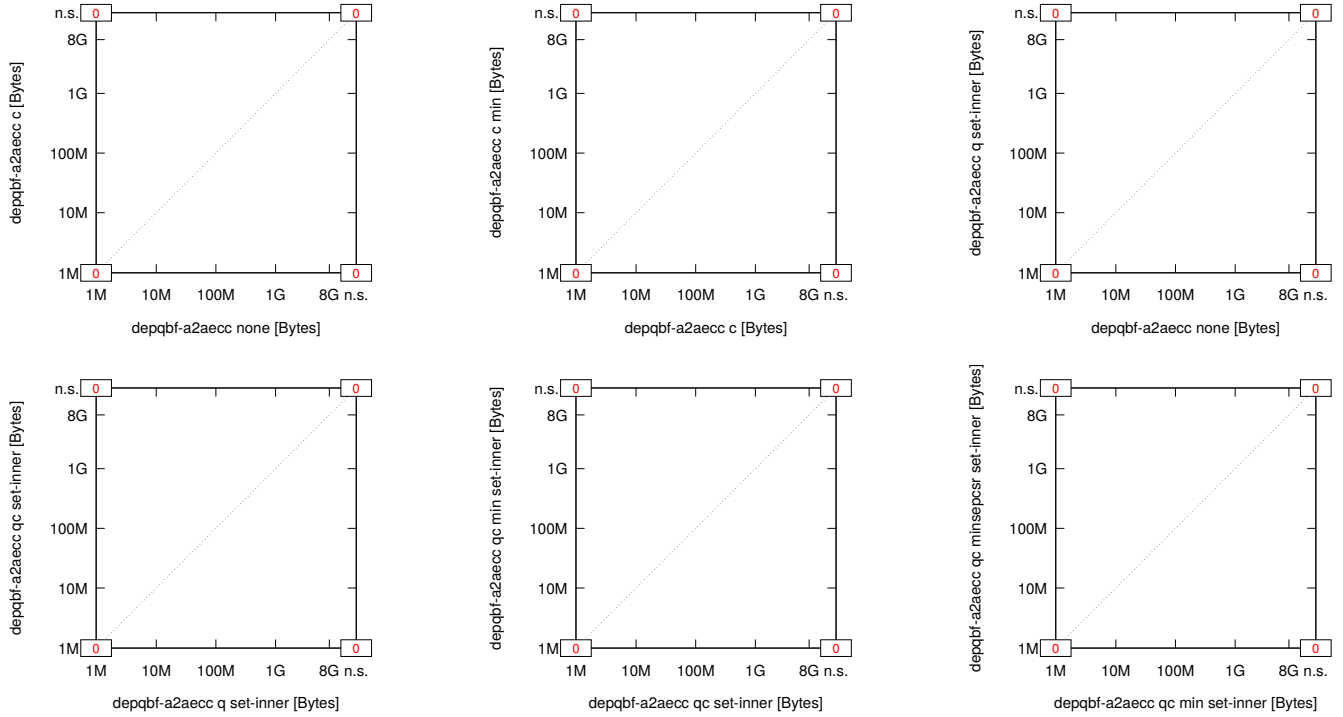


Fig. 908: Suite Messinger ($n = 0$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

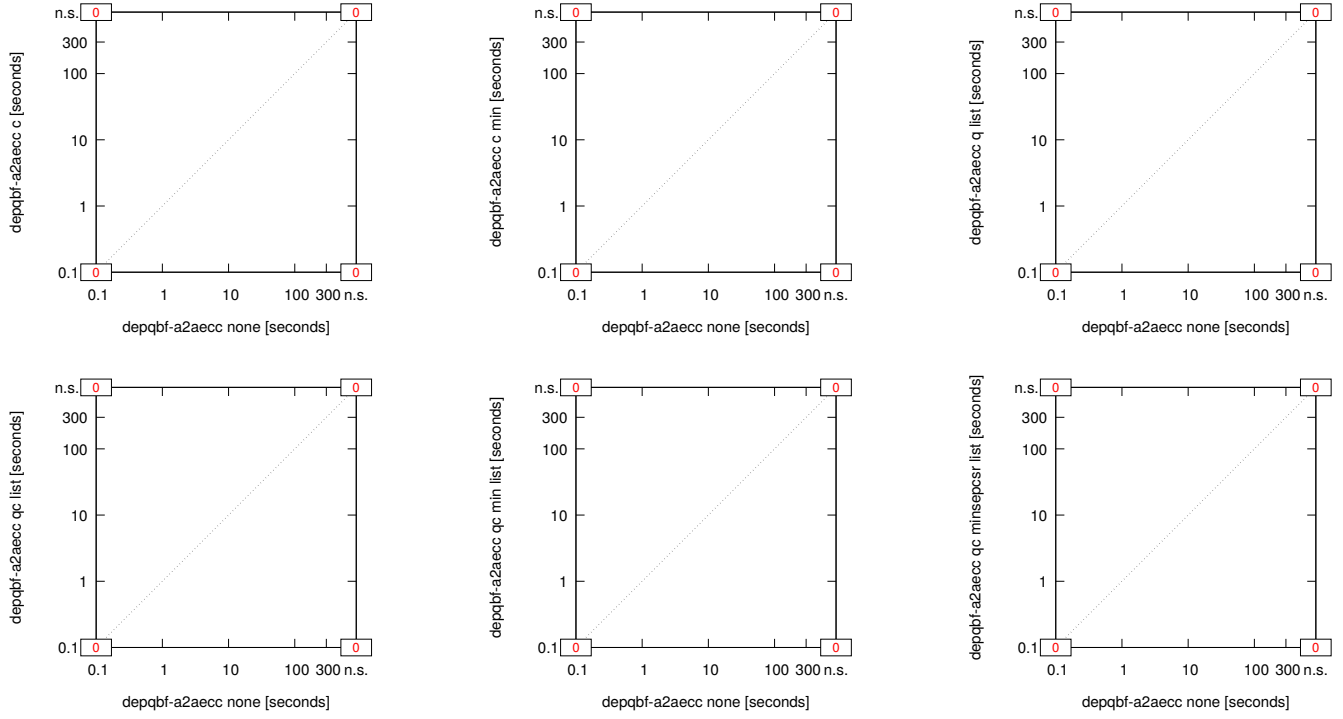


Fig. 909: Suite Messenger ($n = 0$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

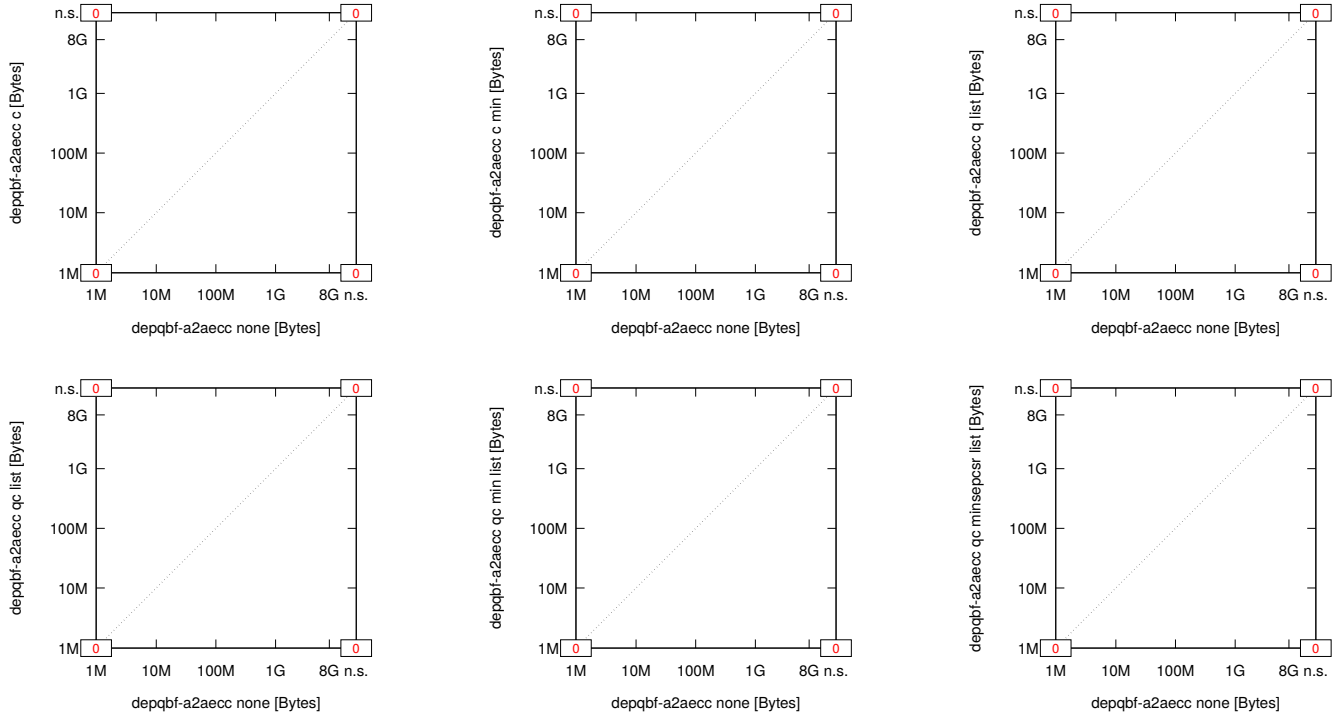


Fig. 910: Suite Messenger ($n = 0$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

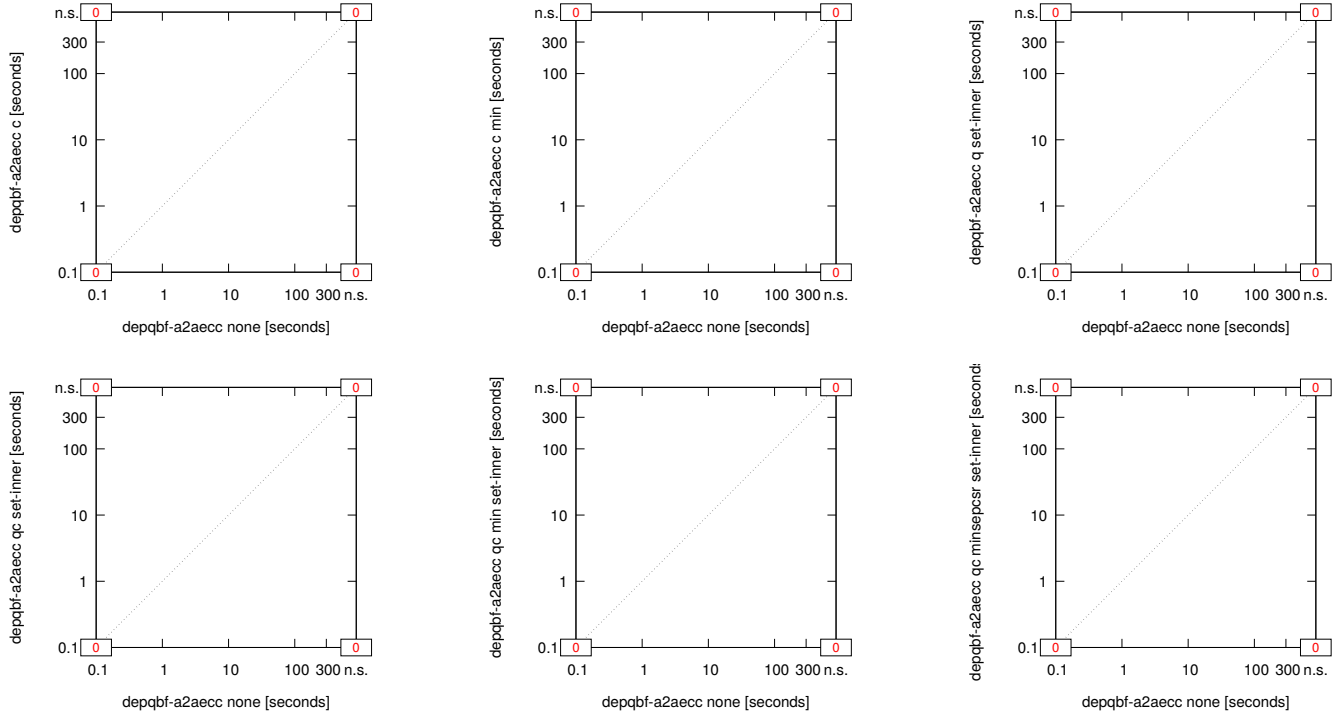


Fig. 911: Suite Messinger ($n = 0$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

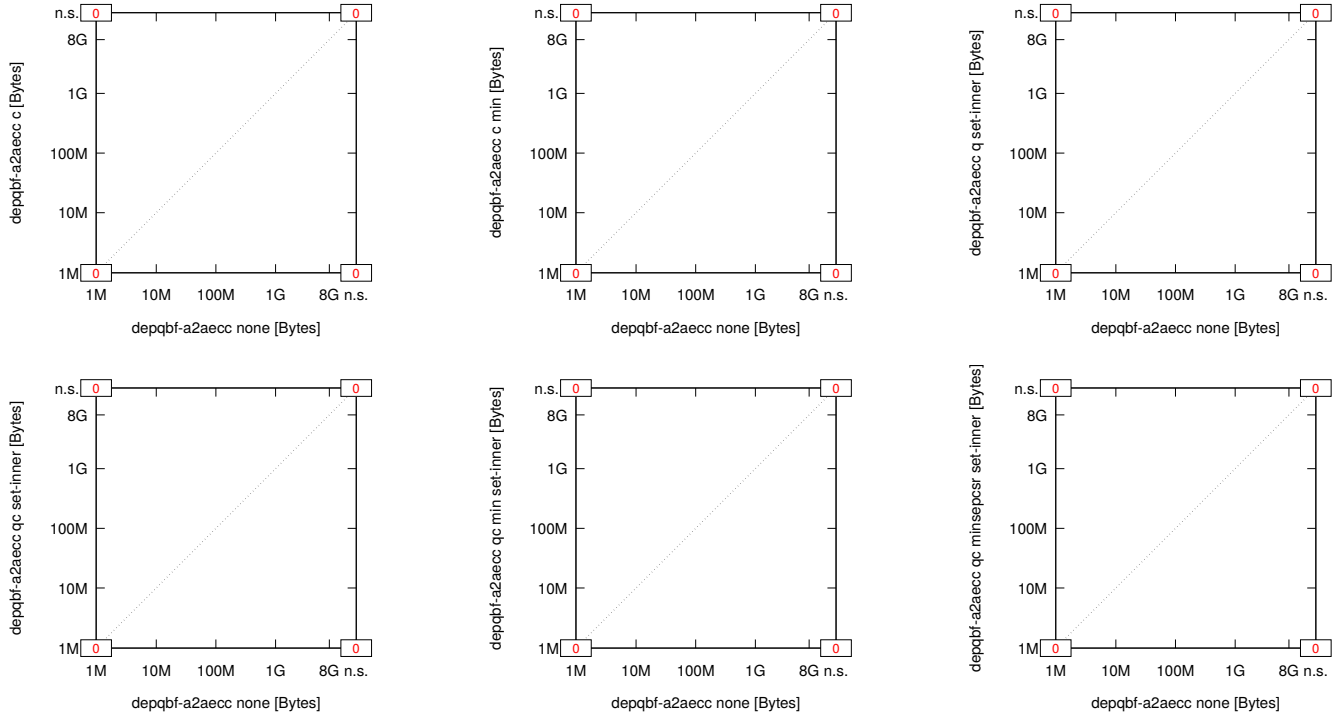


Fig. 912: Suite Messinger ($n = 0$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

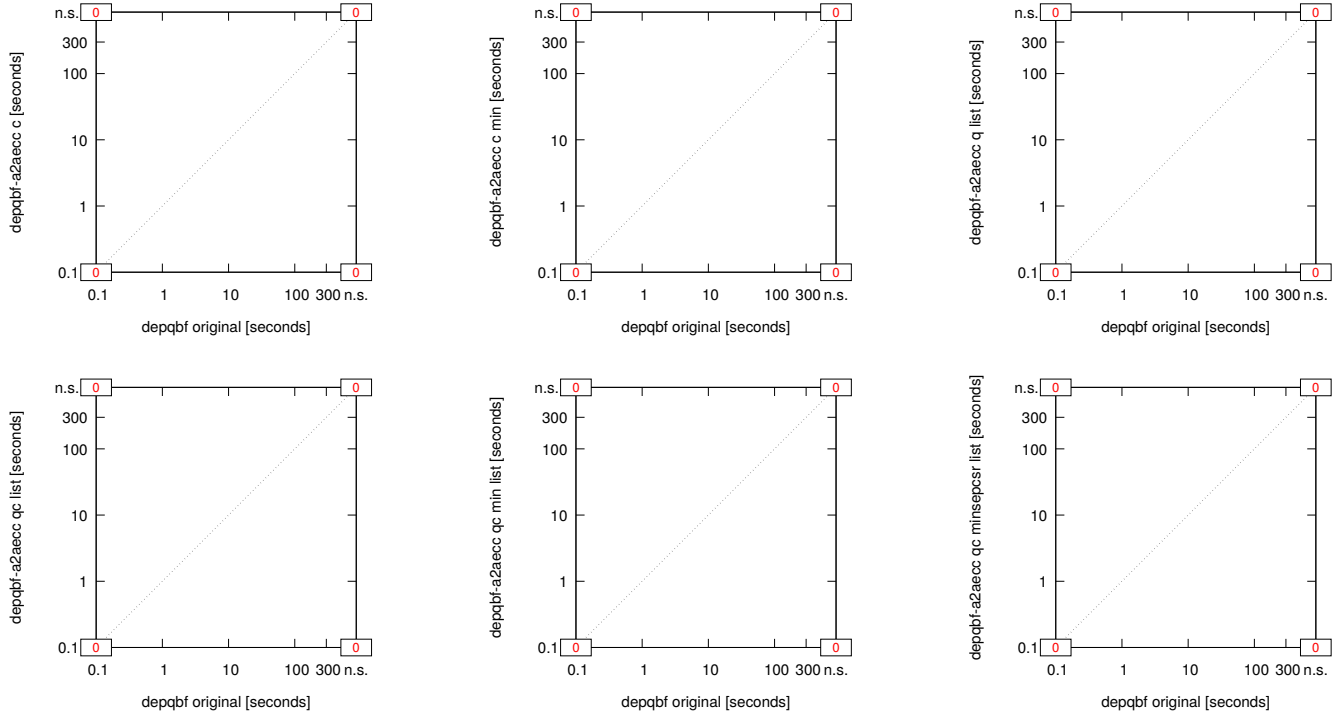


Fig. 913: Suite Messenger ($n = 0$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

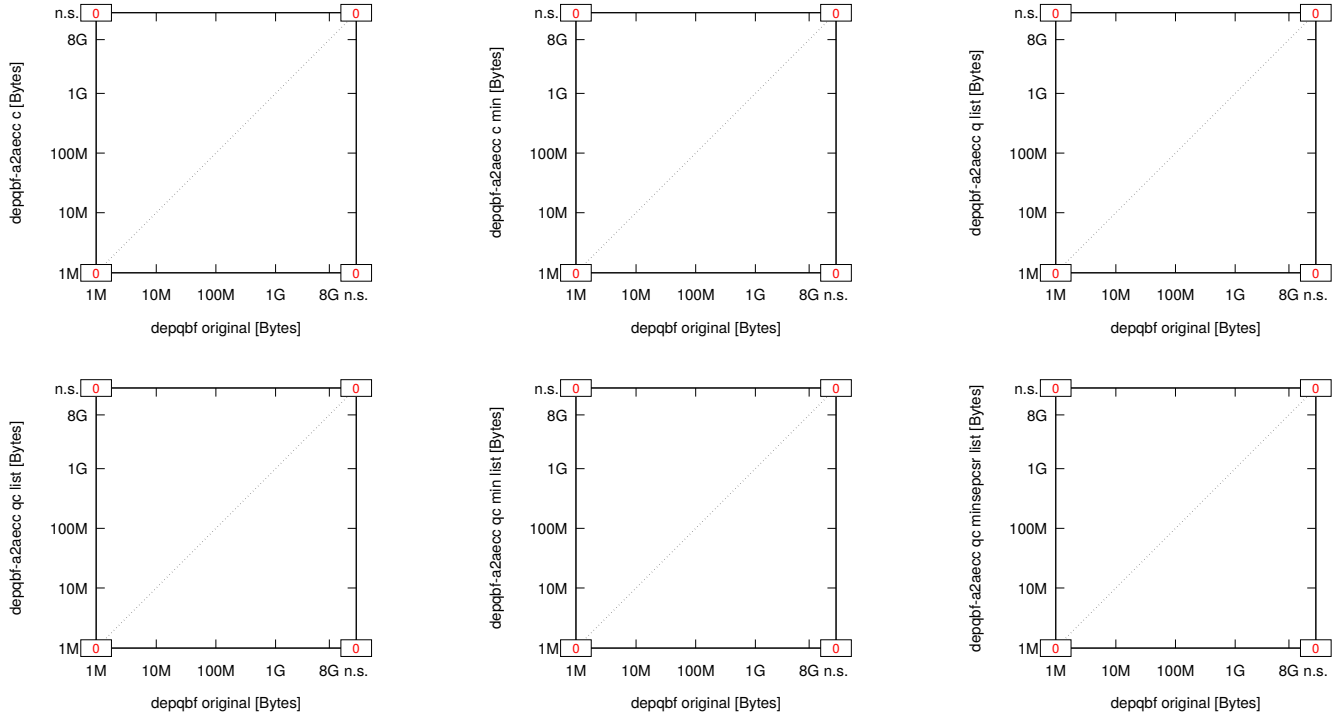


Fig. 914: Suite Messenger ($n = 0$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

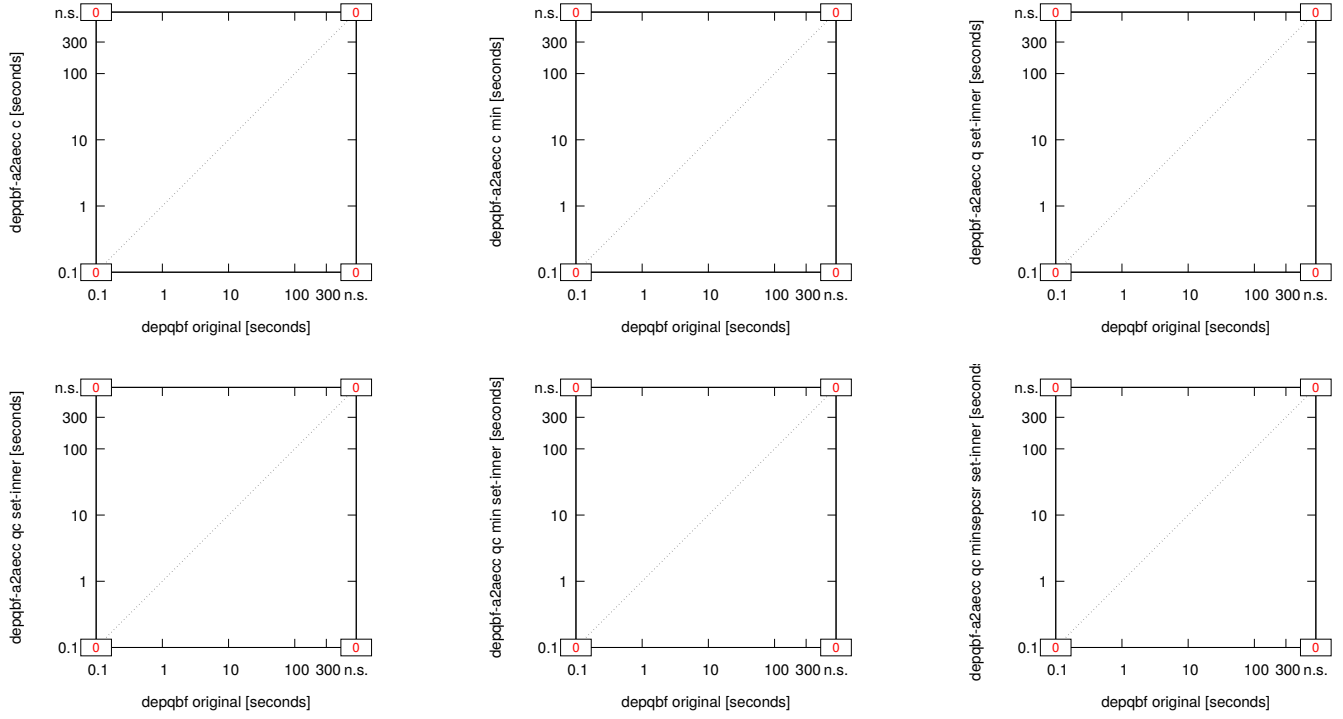


Fig. 915: Suite Messinger ($n = 0$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

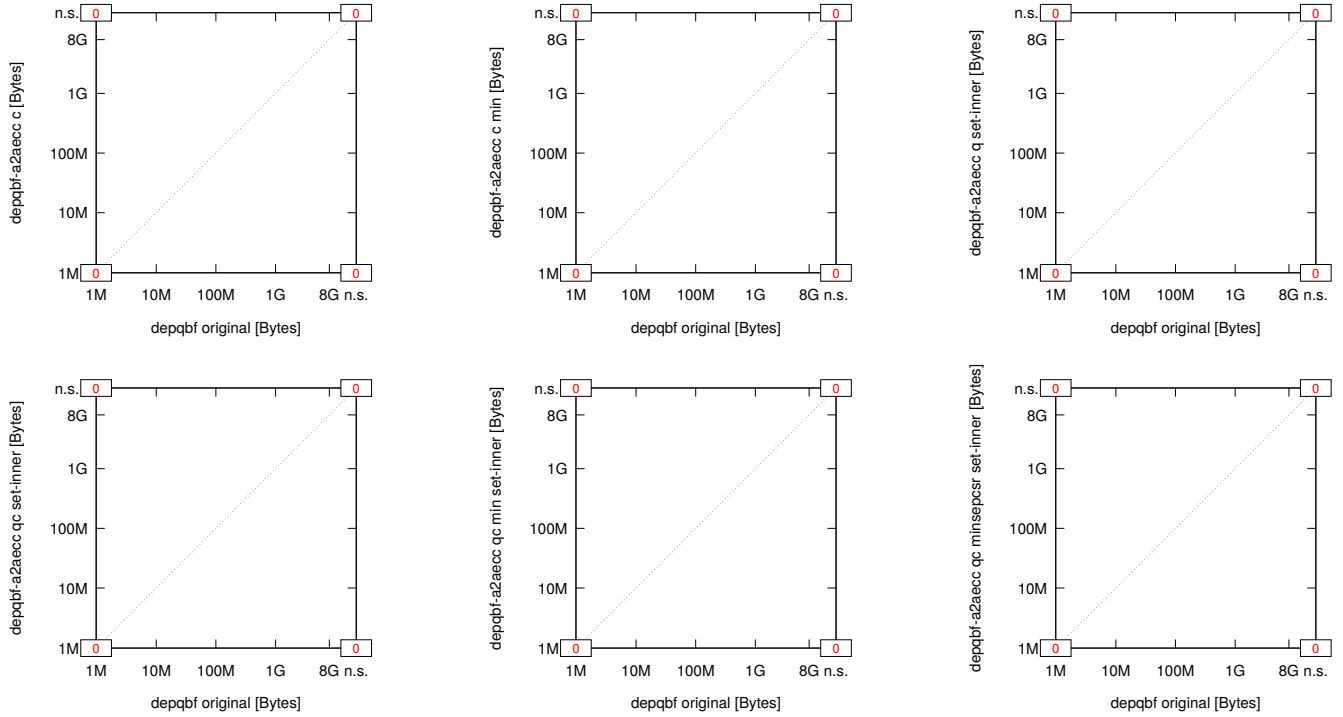


Fig. 916: Suite Messinger ($n = 0$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

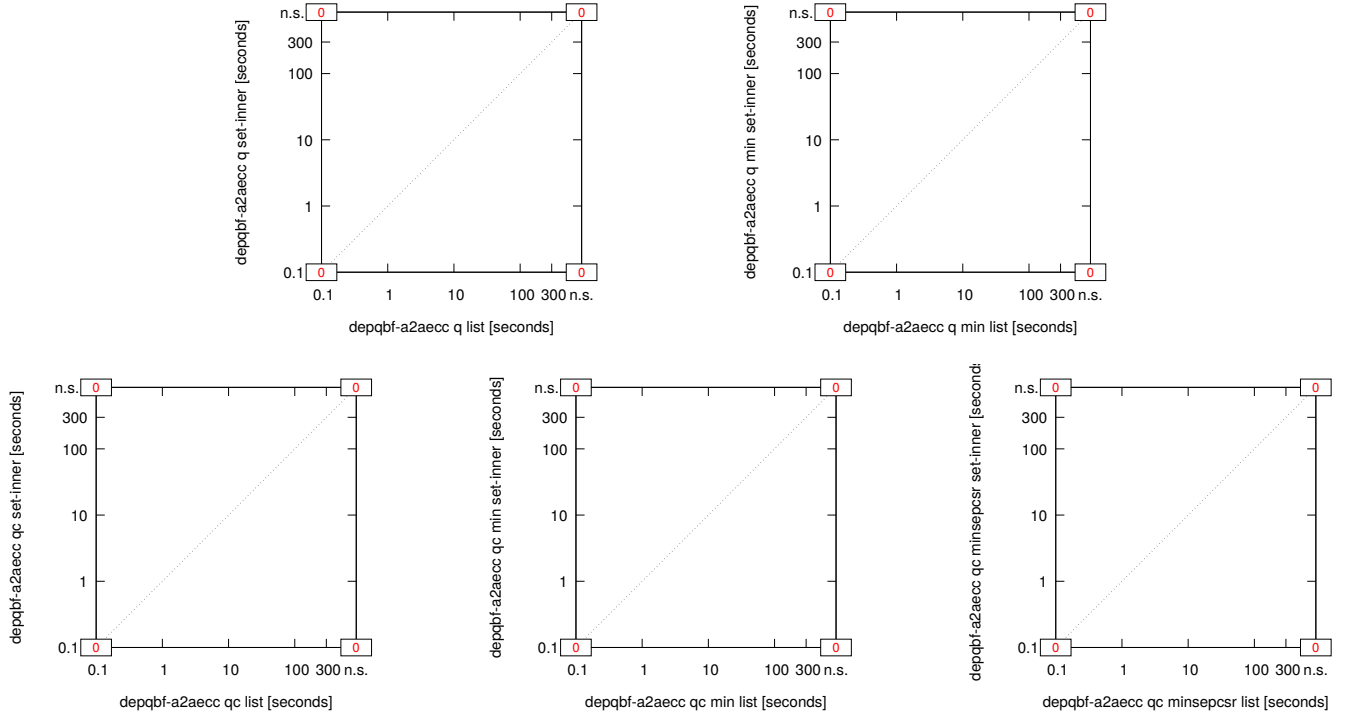


Fig. 917: Suite Messenger ($n = 0$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

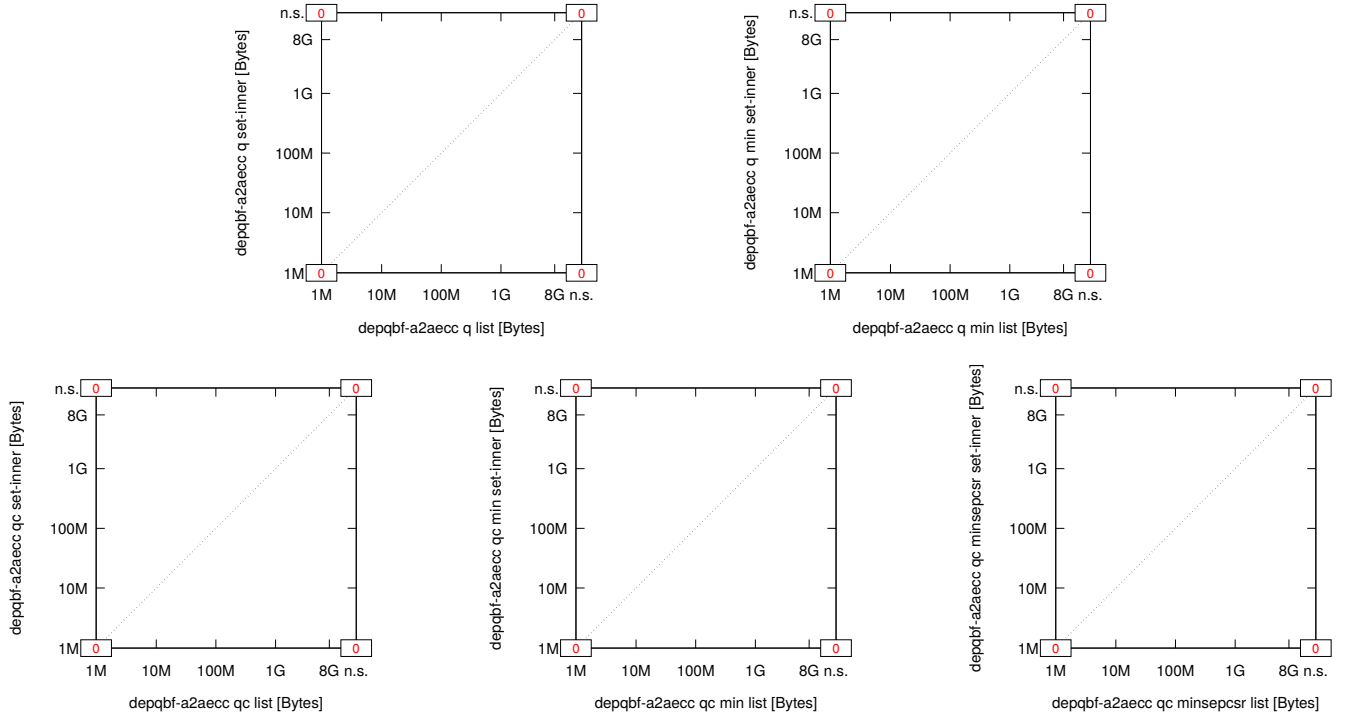


Fig. 918: Suite Messenger ($n = 0$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

30) Miller-Marin ($n = 189$):

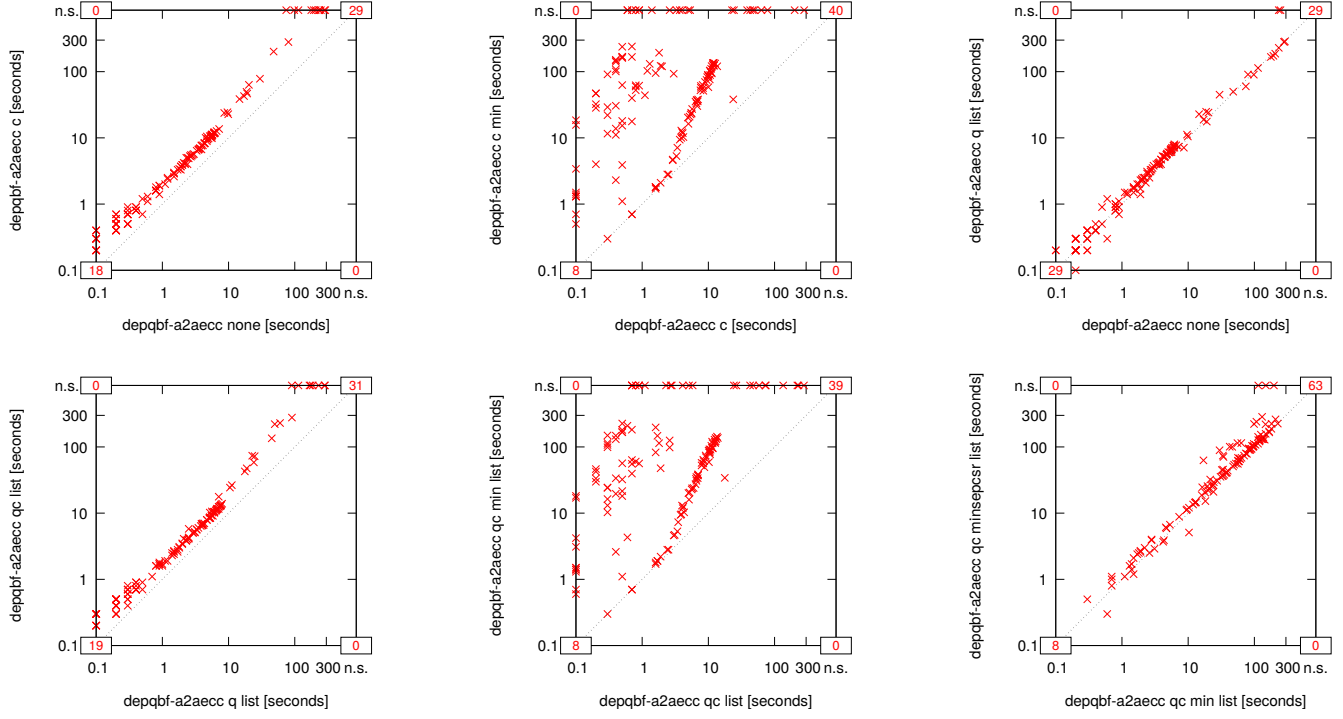


Fig. 919: Suite Miller-Marin ($n = 189$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

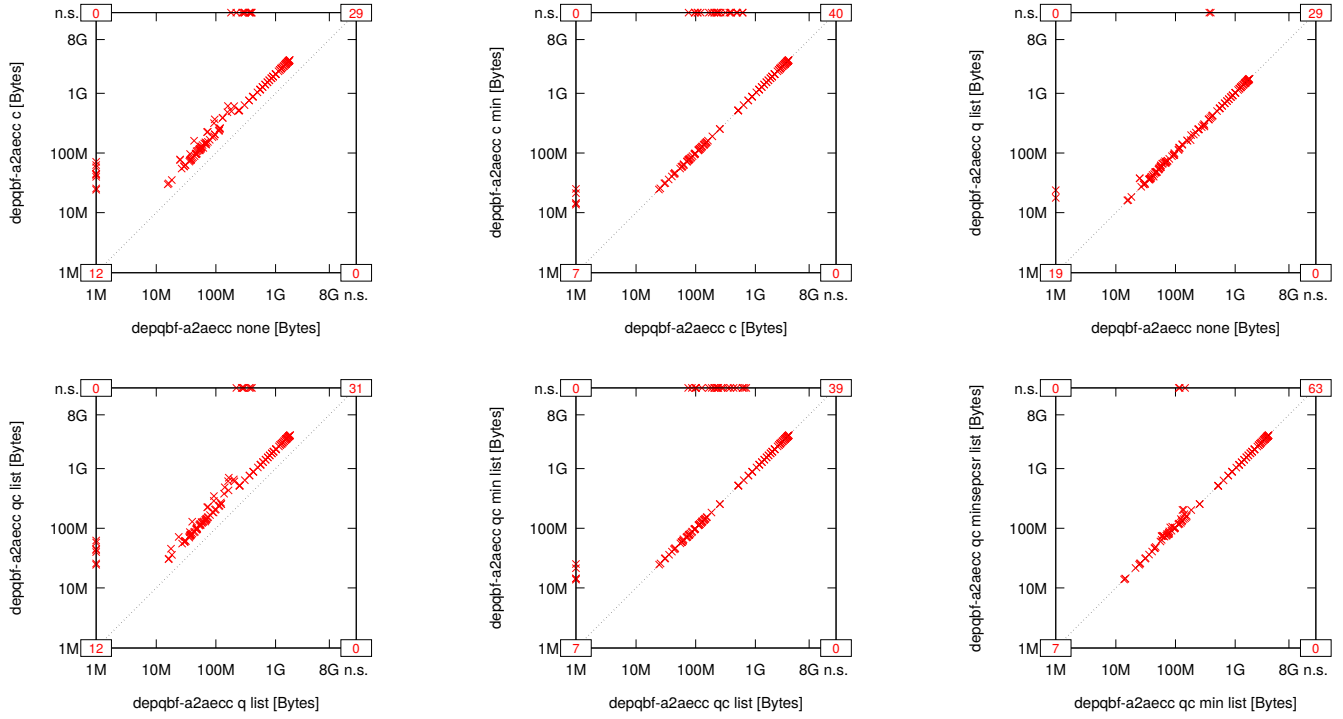


Fig. 920: Suite Miller-Marin ($n = 189$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

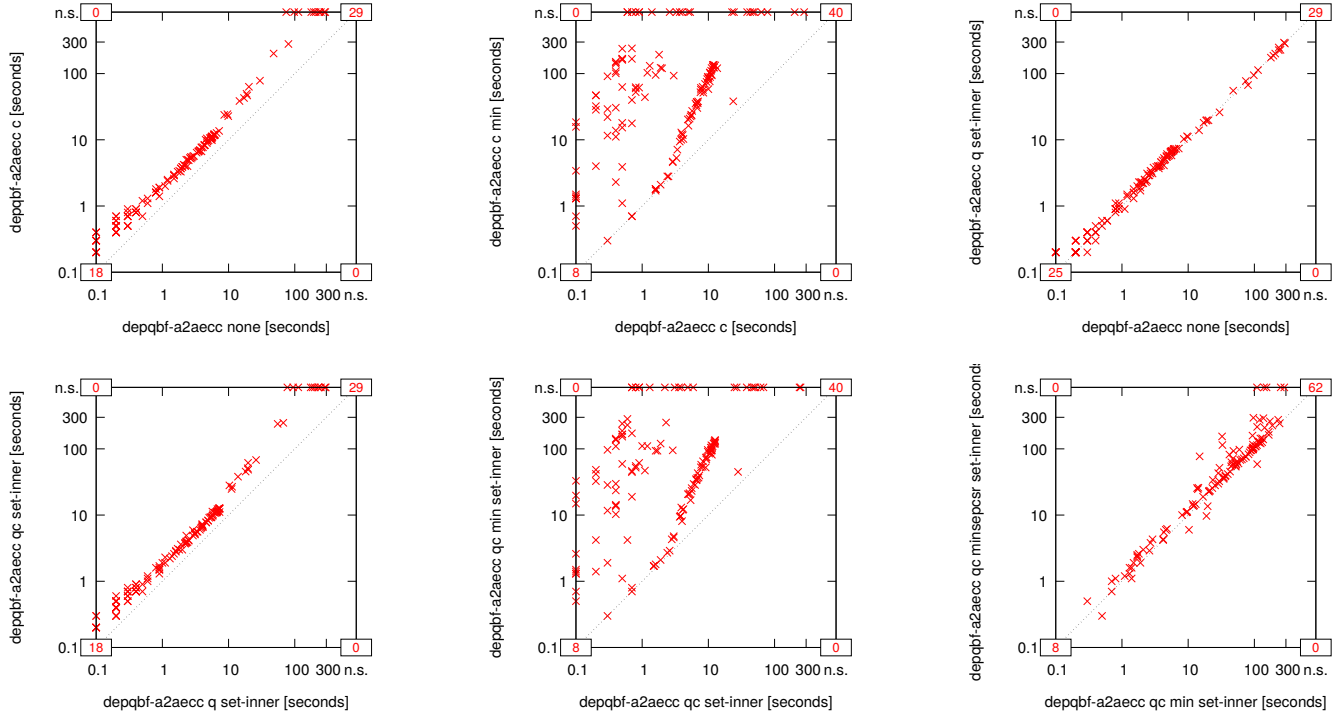


Fig. 921: Suite Miller-Marin ($n = 189$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

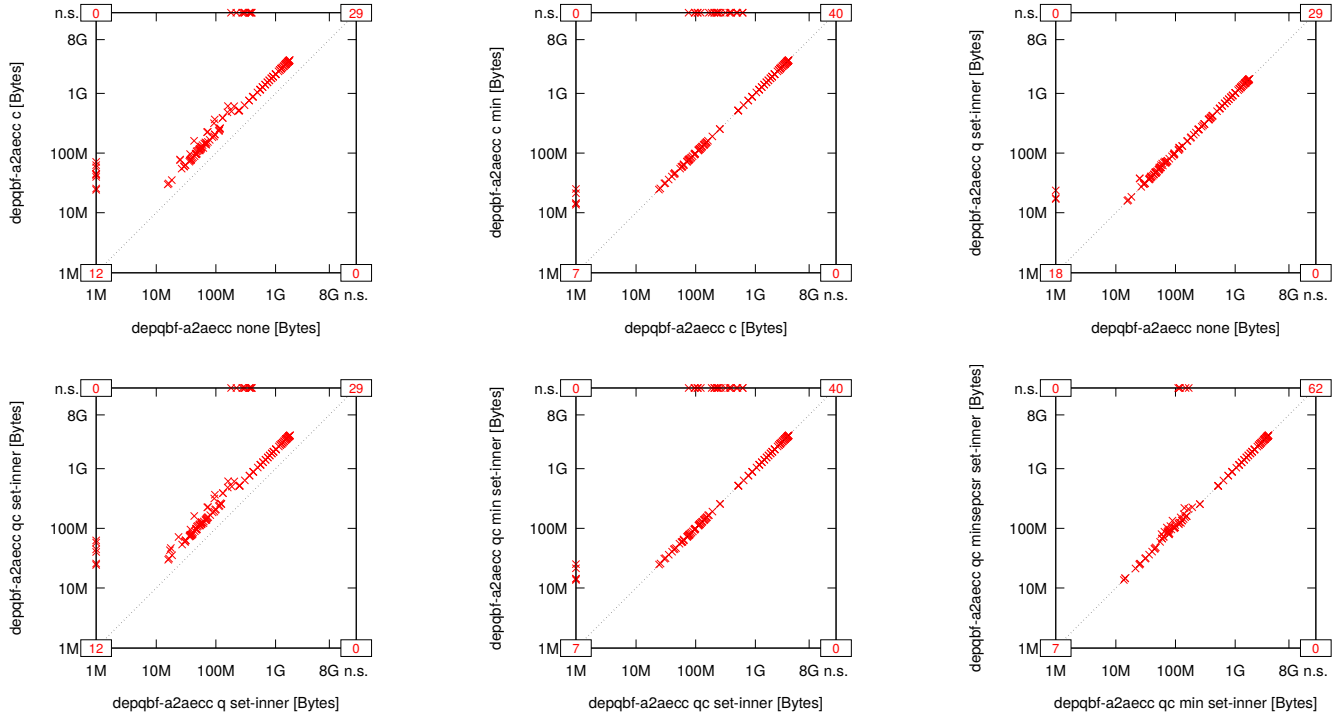


Fig. 922: Suite Miller-Marin ($n = 189$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

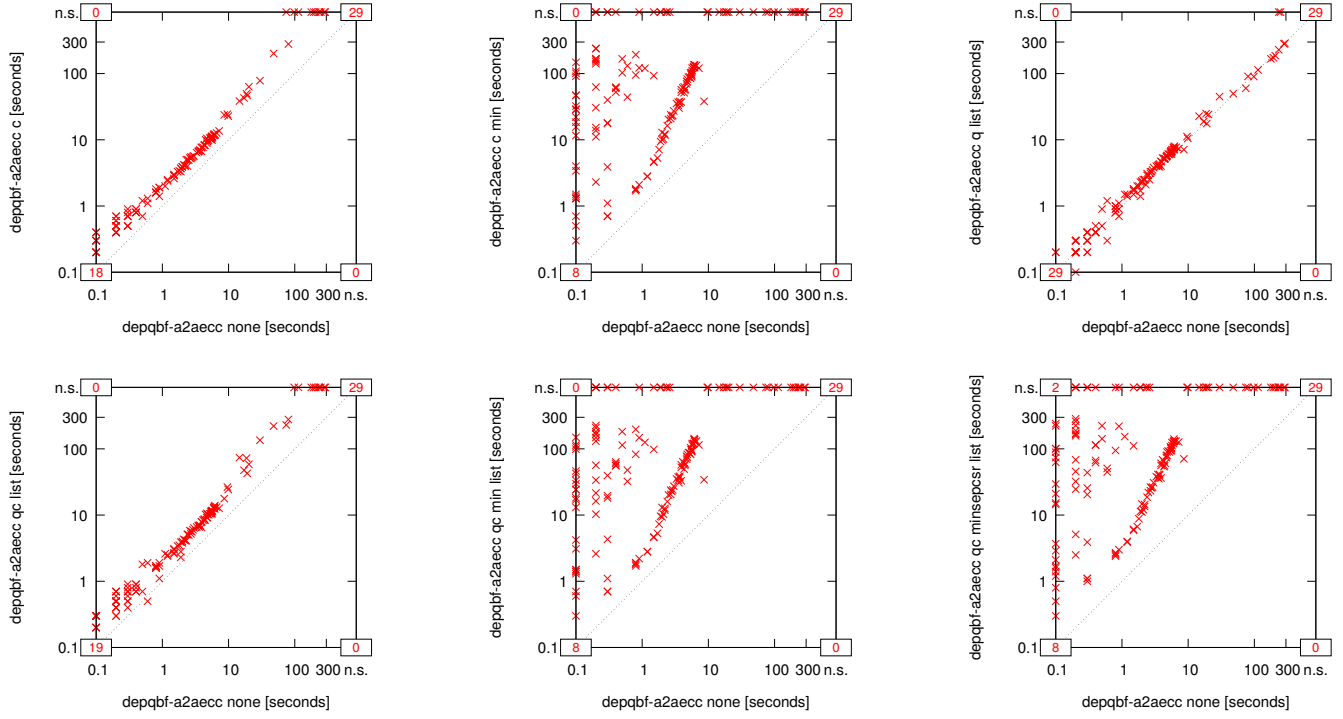


Fig. 923: Suite Miller-Marin ($n = 189$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

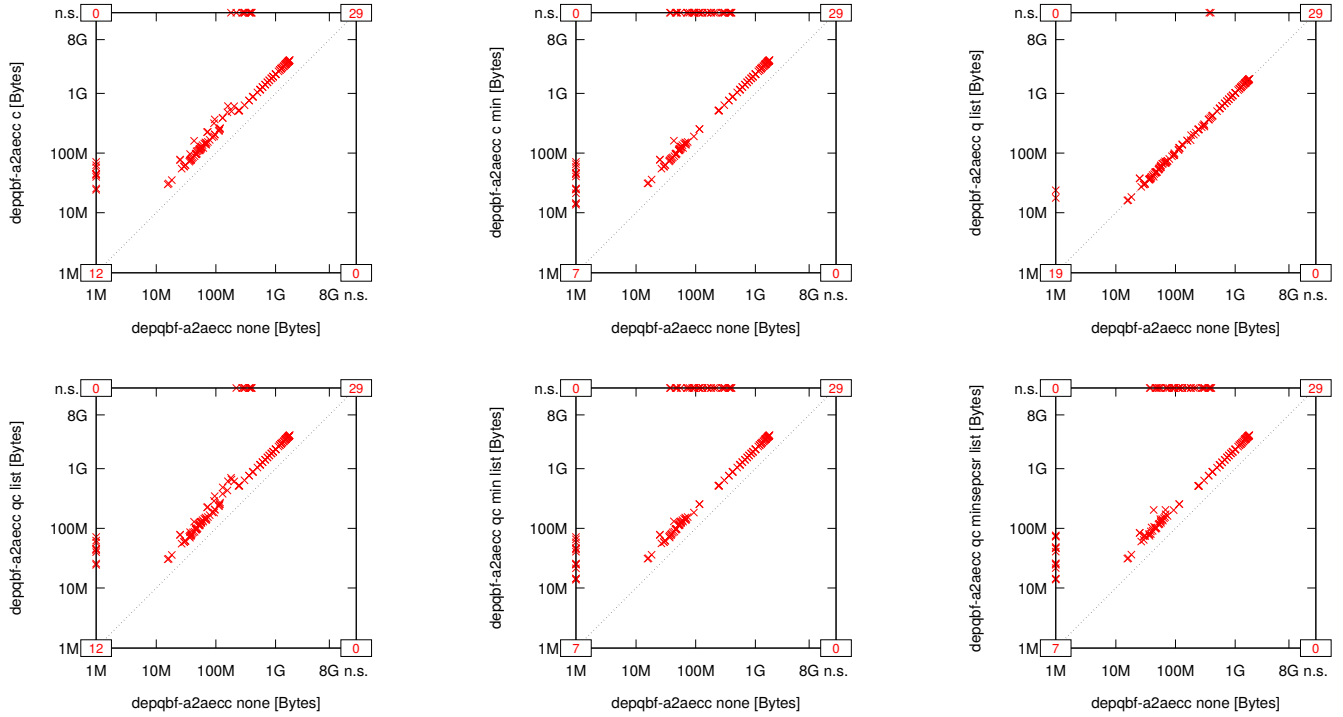


Fig. 924: Suite Miller-Marin ($n = 189$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

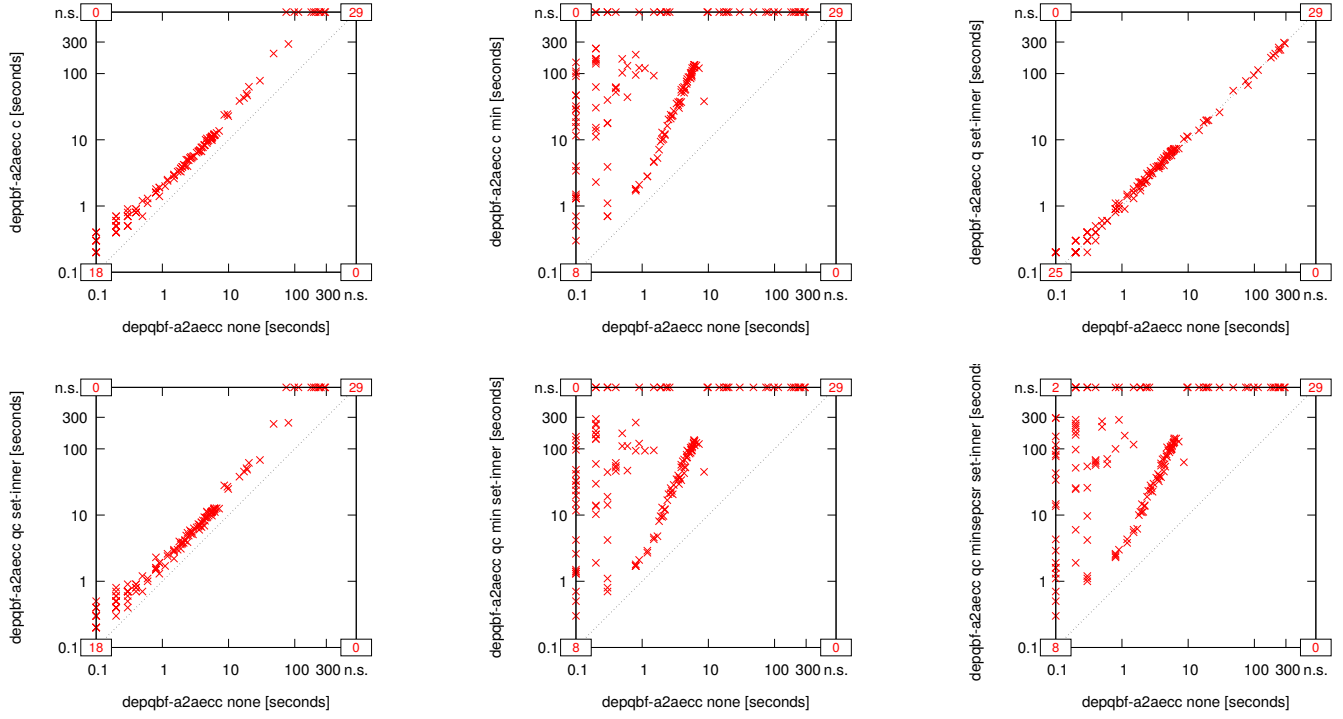


Fig. 925: Suite Miller-Marin ($n = 189$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

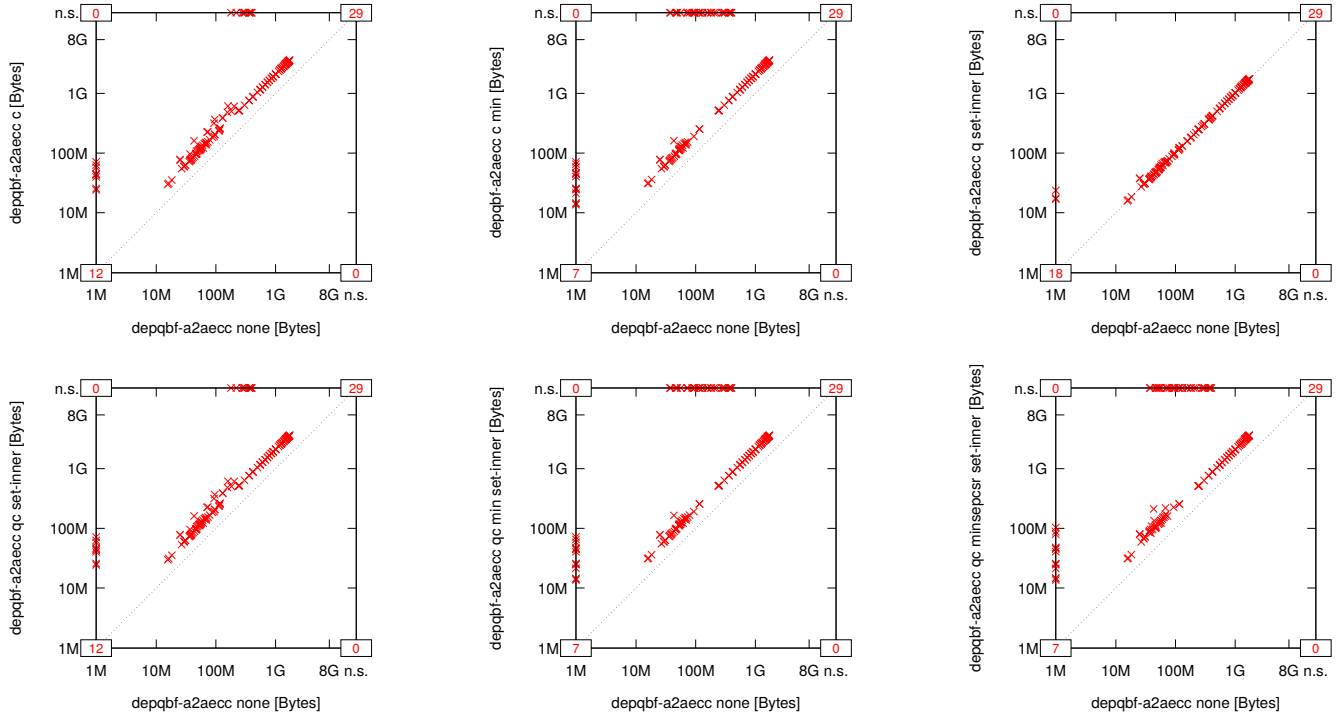


Fig. 926: Suite Miller-Marin ($n = 189$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

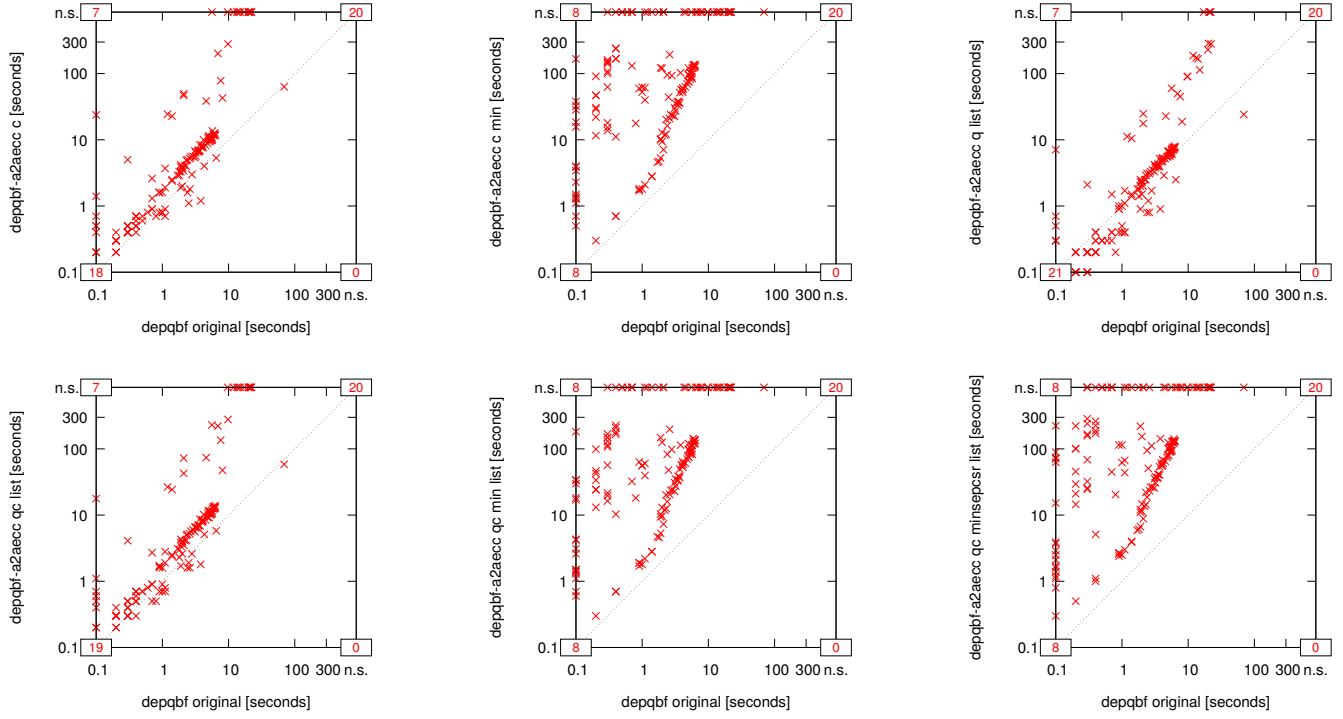


Fig. 927: Suite Miller-Marin ($n = 189$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

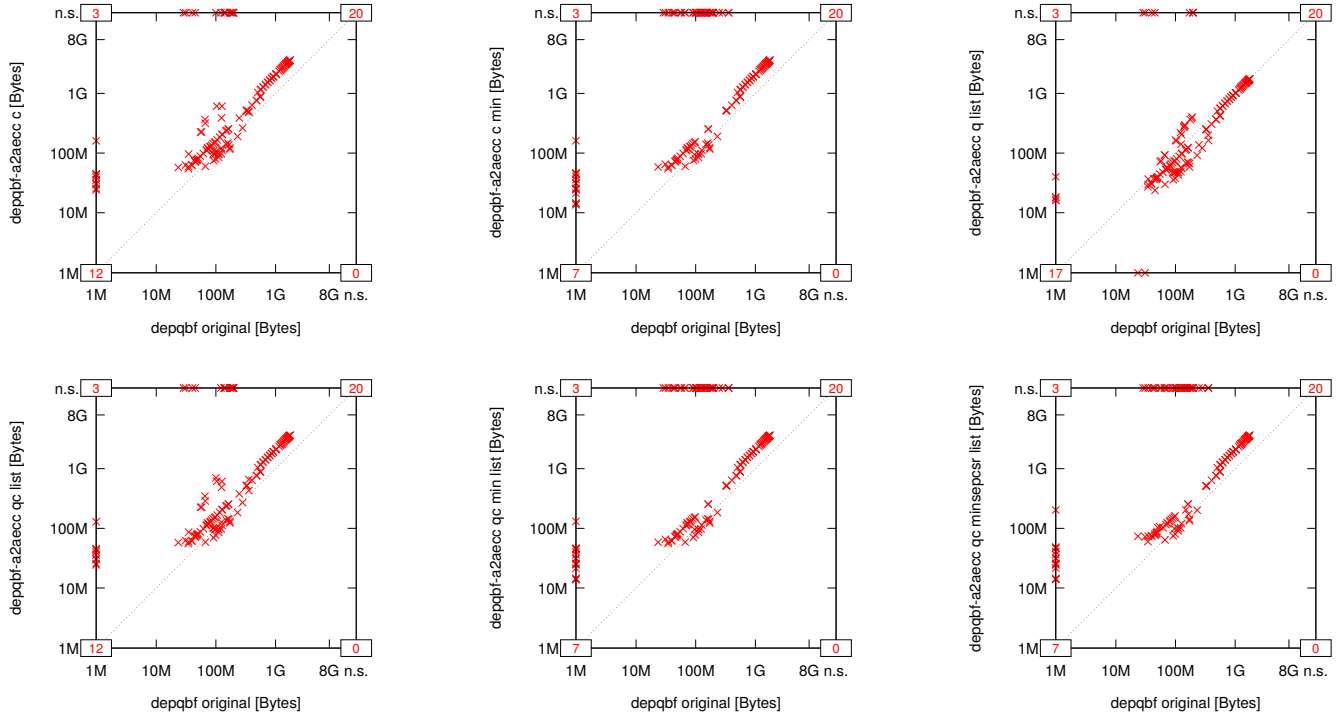


Fig. 928: Suite Miller-Marin ($n = 189$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

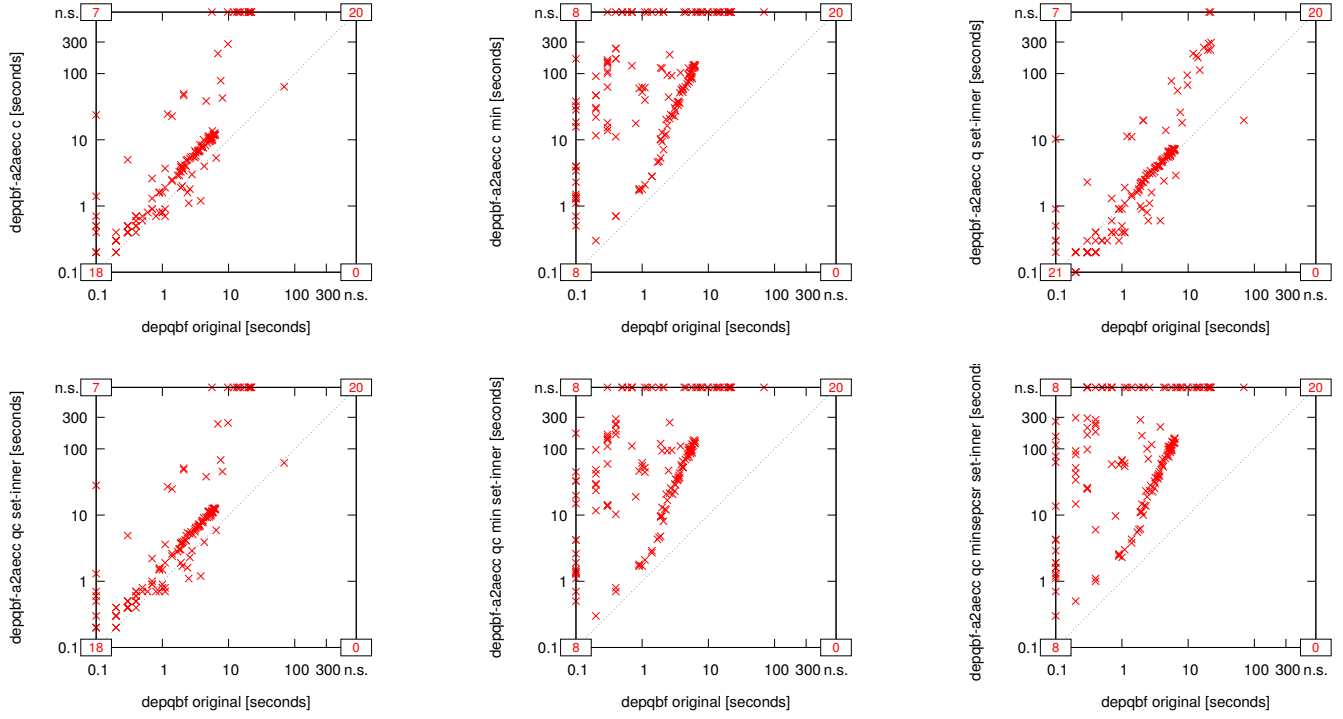


Fig. 929: Suite Miller-Marin ($n = 189$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

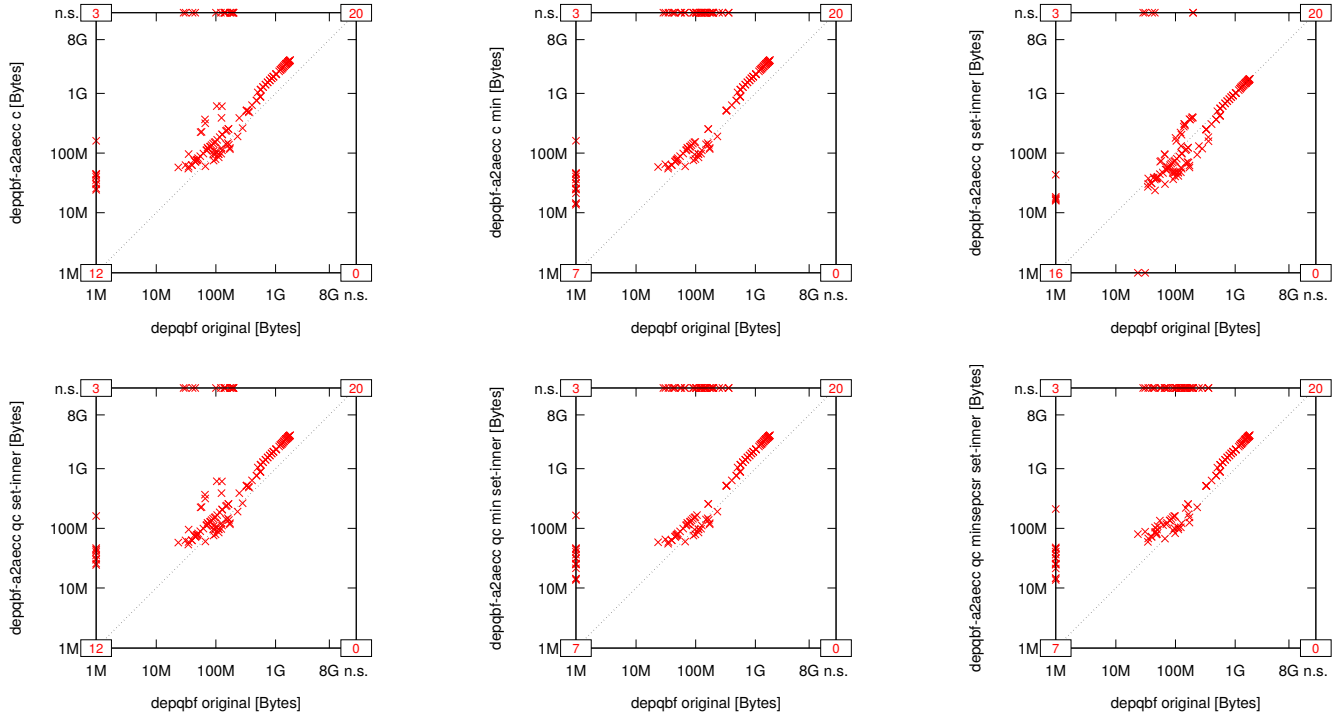


Fig. 930: Suite Miller-Marin ($n = 189$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

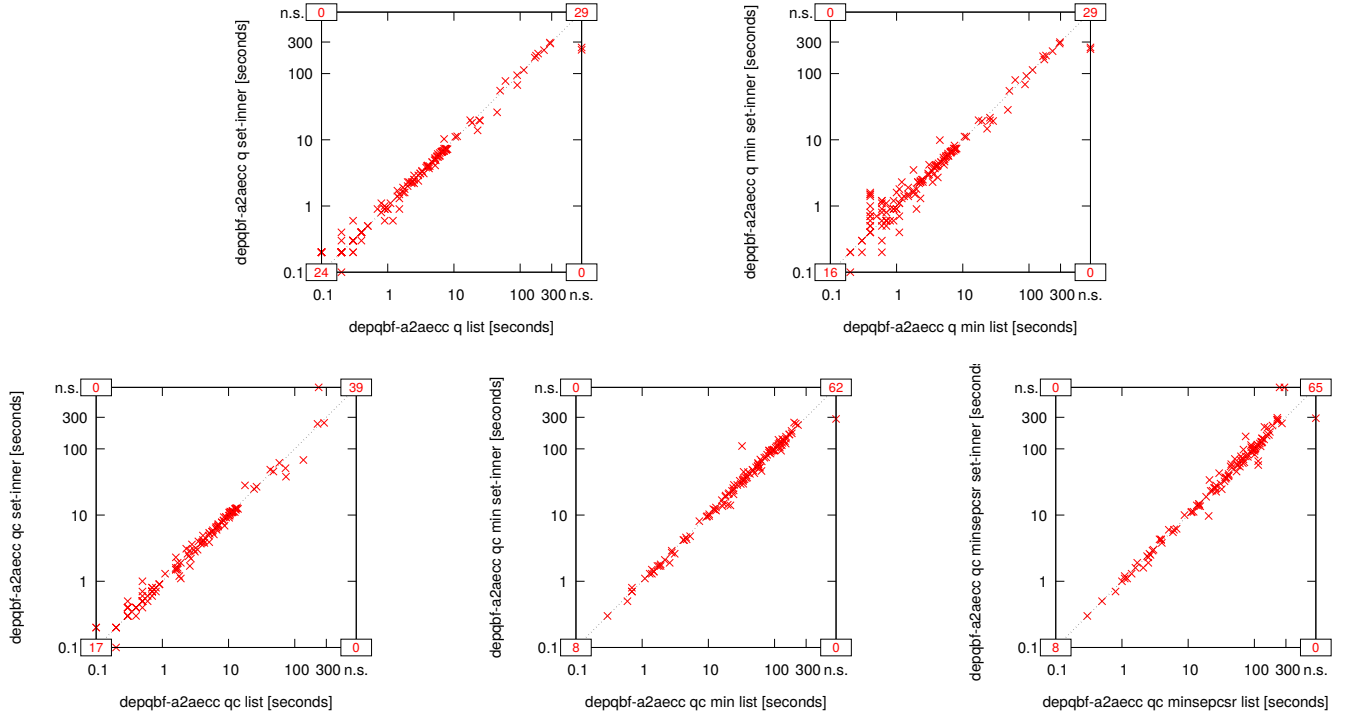


Fig. 931: Suite Miller-Marin ($n = 189$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

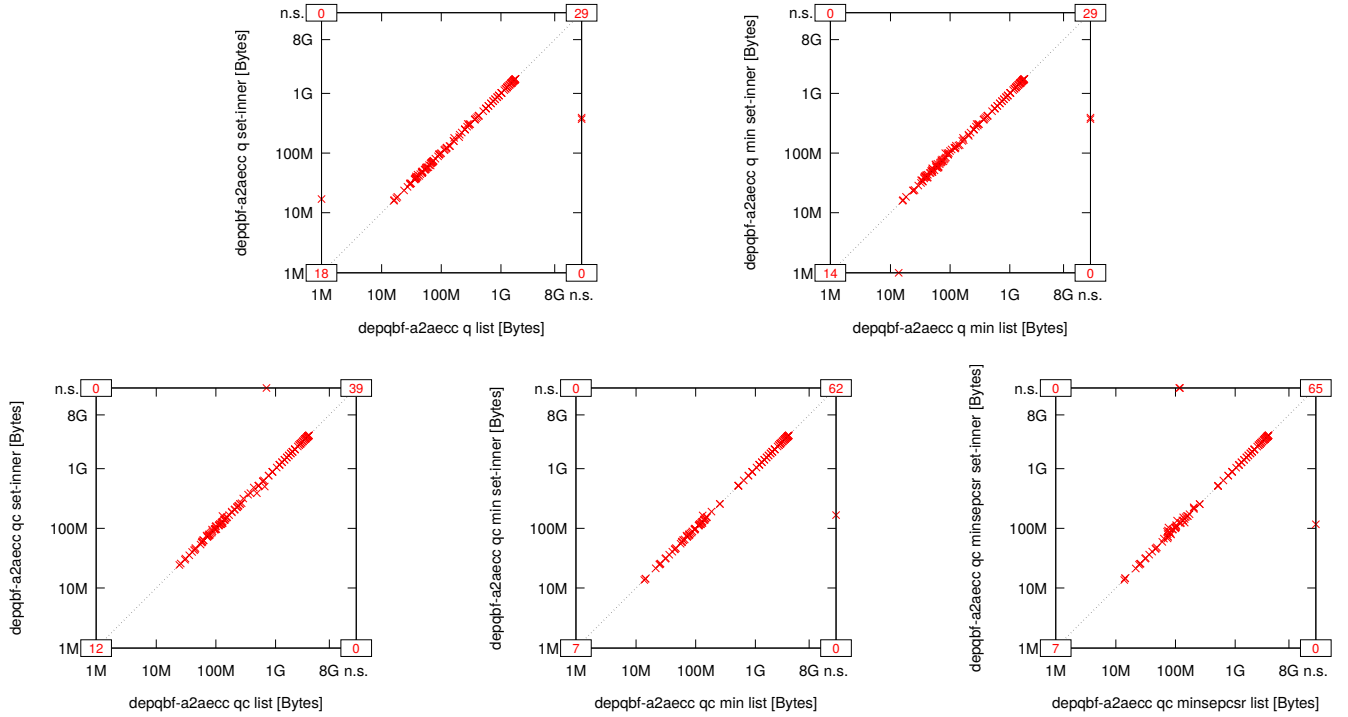


Fig. 932: Suite Miller-Marin ($n = 189$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

31) Miller-Scholl-Becker ($n = 160$):

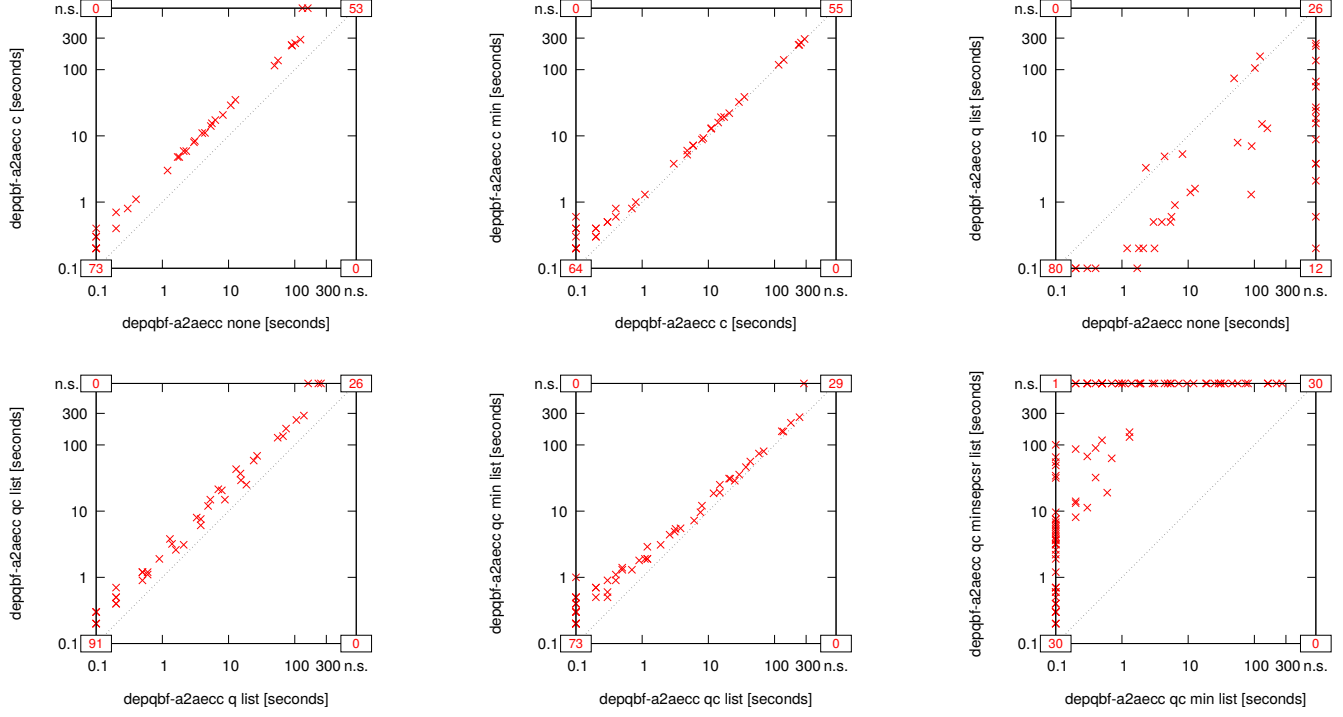


Fig. 933: Suite Miller-Scholl-Becker ($n = 160$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

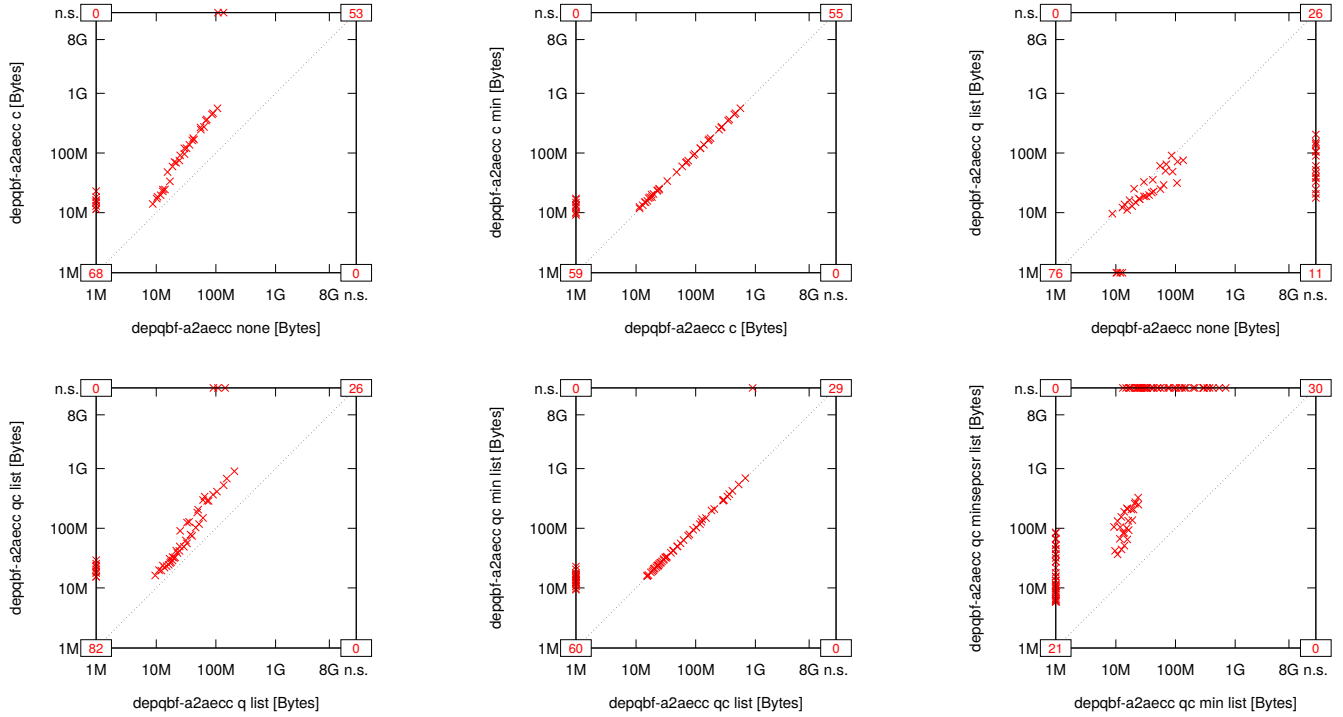


Fig. 934: Suite Miller-Scholl-Becker ($n = 160$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

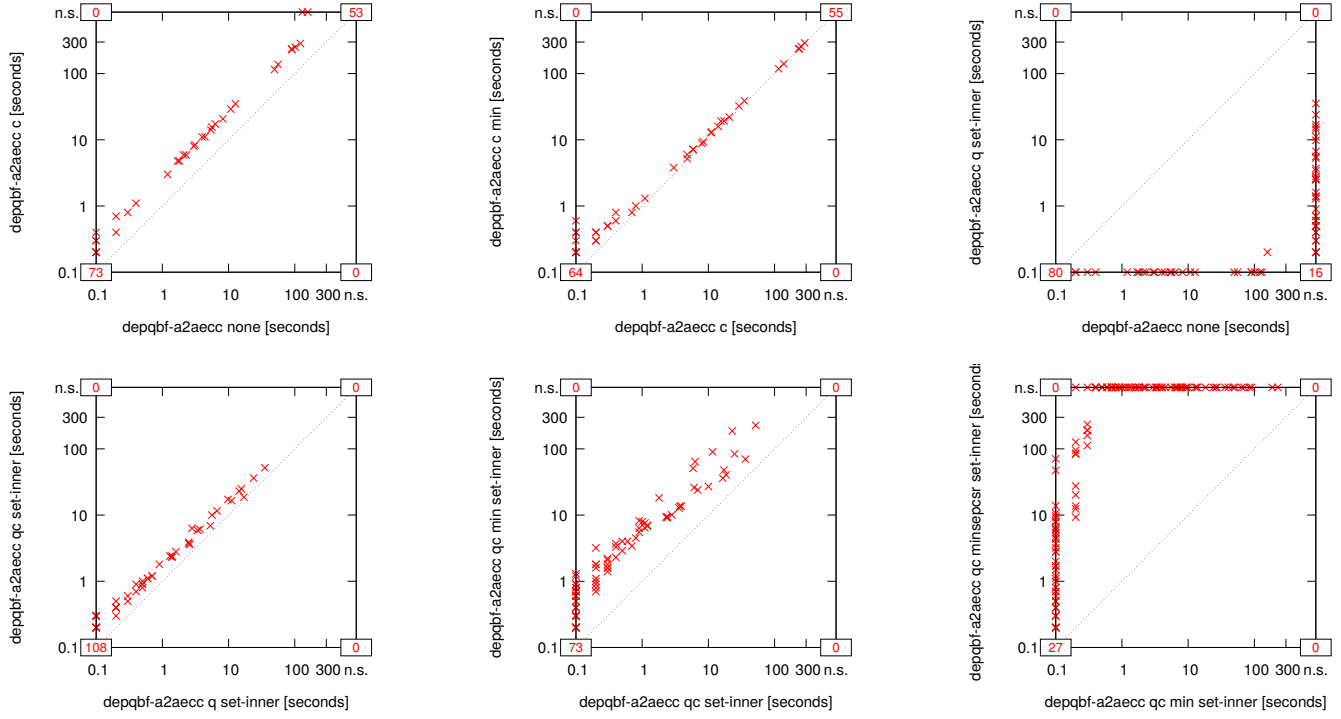


Fig. 935: Suite Miller-Scholl-Becker ($n = 160$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

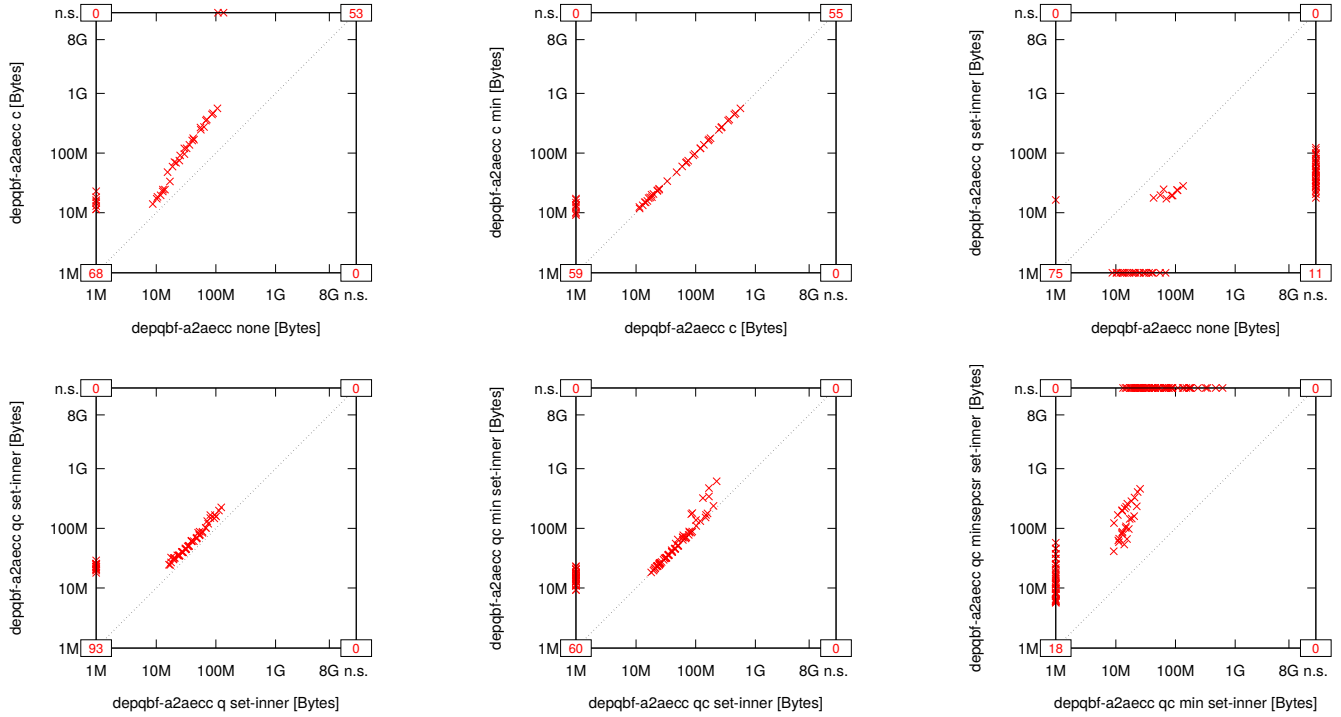


Fig. 936: Suite Miller-Scholl-Becker ($n = 160$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

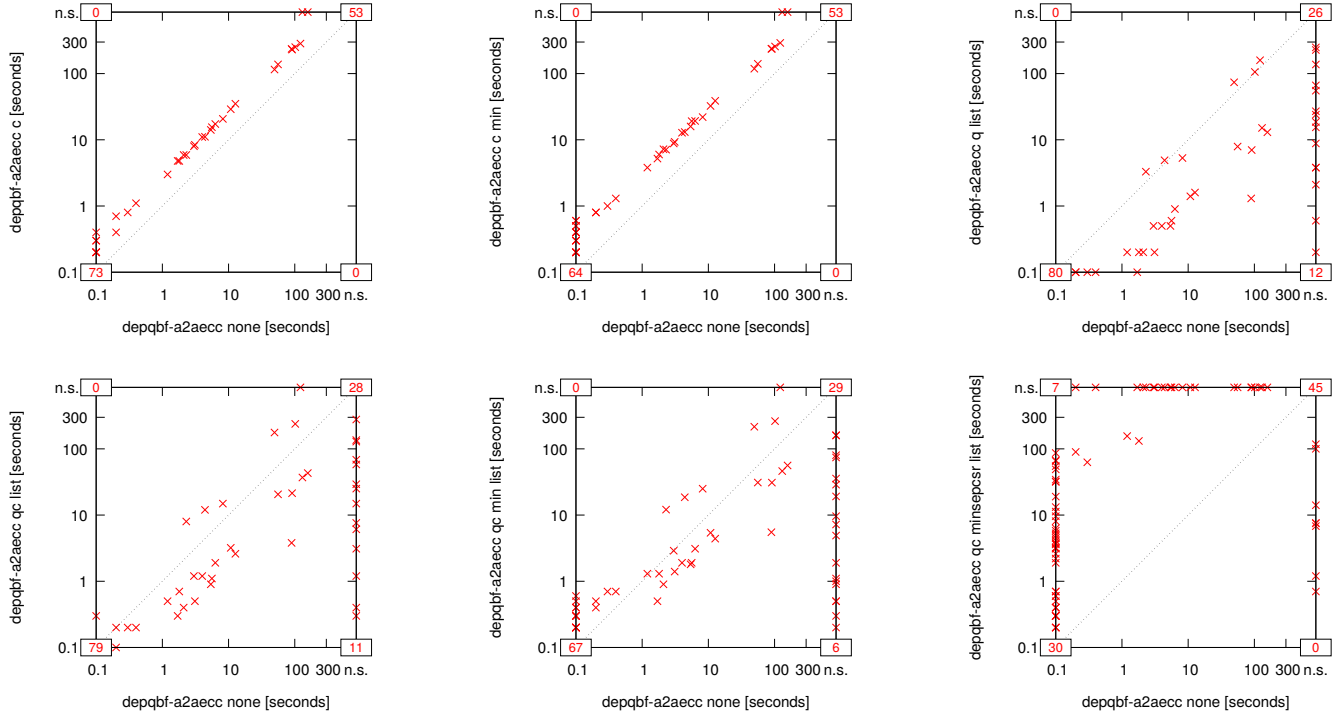


Fig. 937: Suite Miller-Scholl-Becker ($n = 160$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. Dep-QBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

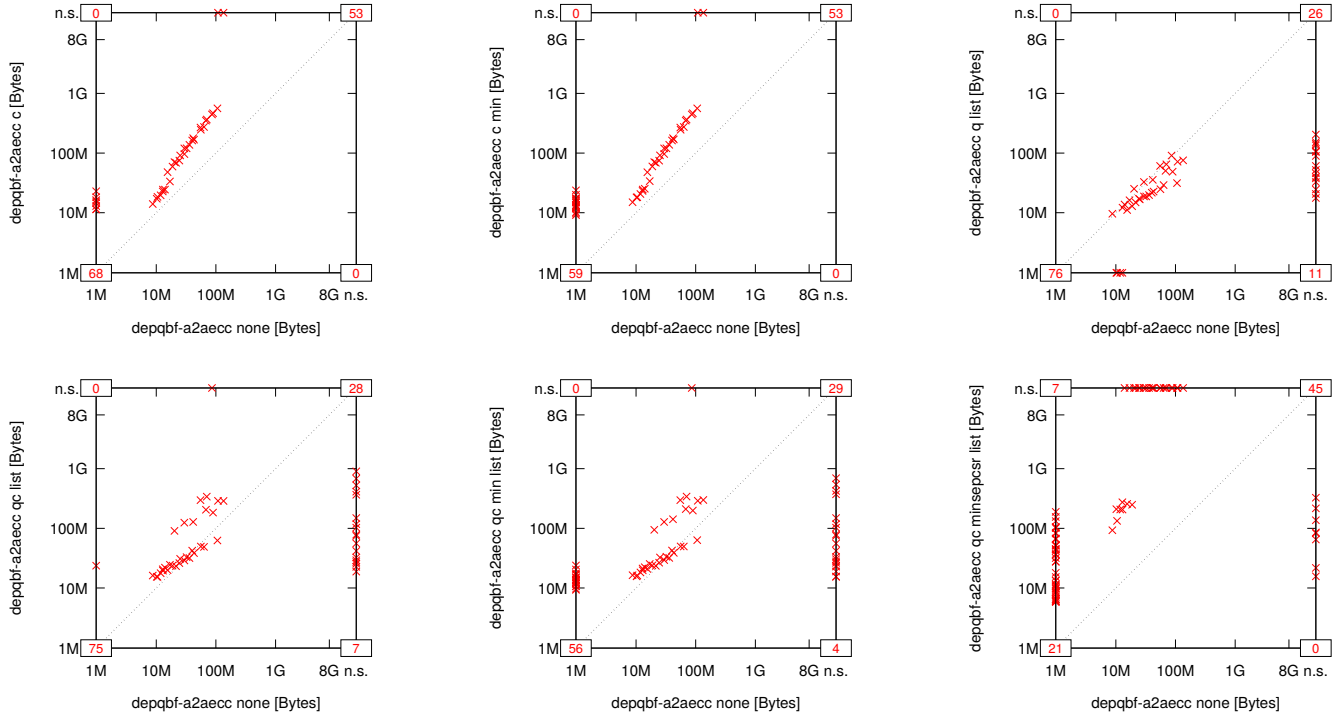


Fig. 938: Suite Miller-Scholl-Becker ($n = 160$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. Dep-QBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

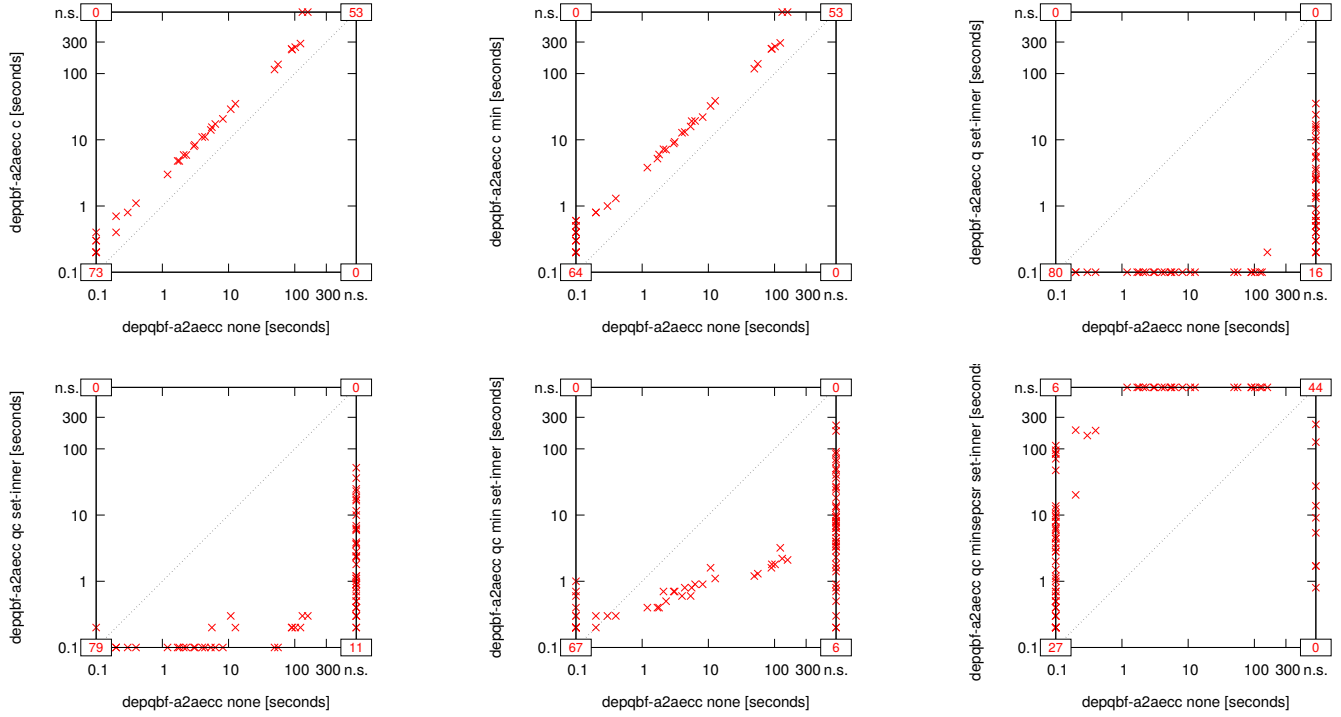


Fig. 939: Suite Miller-Scholl-Becker ($n = 160$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

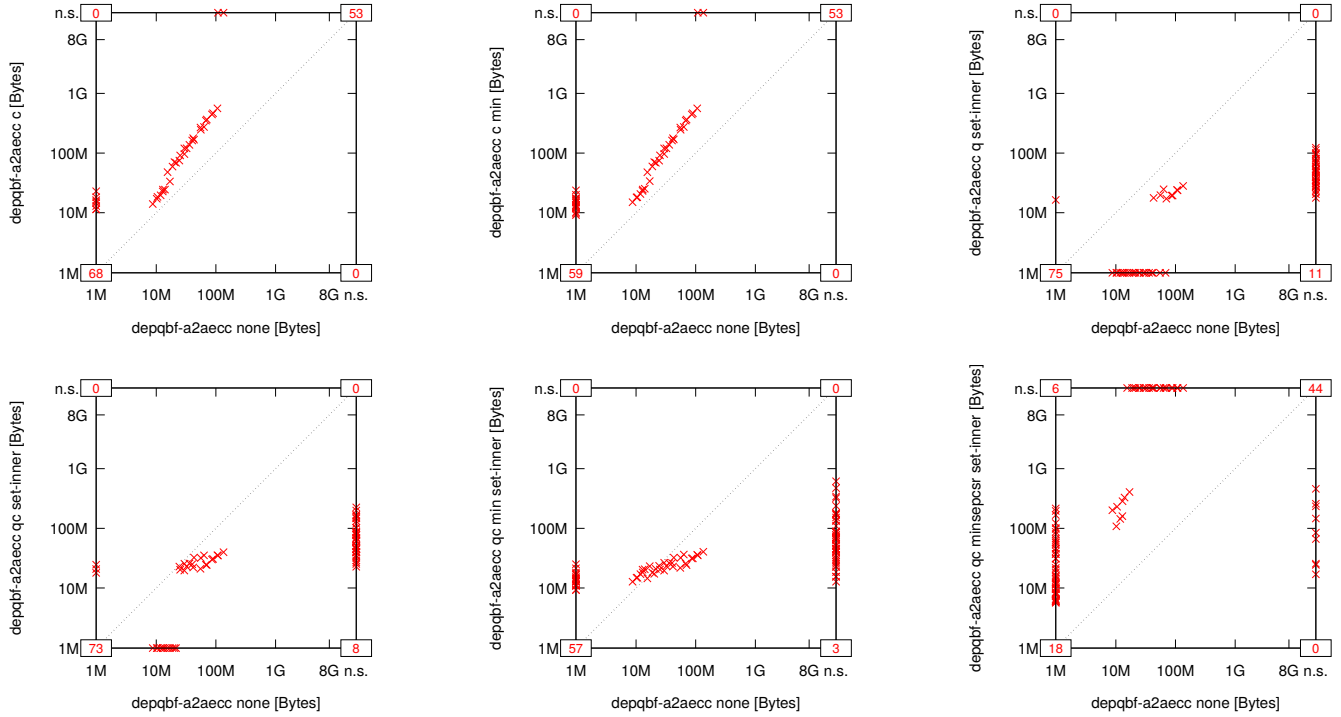


Fig. 940: Suite Miller-Scholl-Becker ($n = 160$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

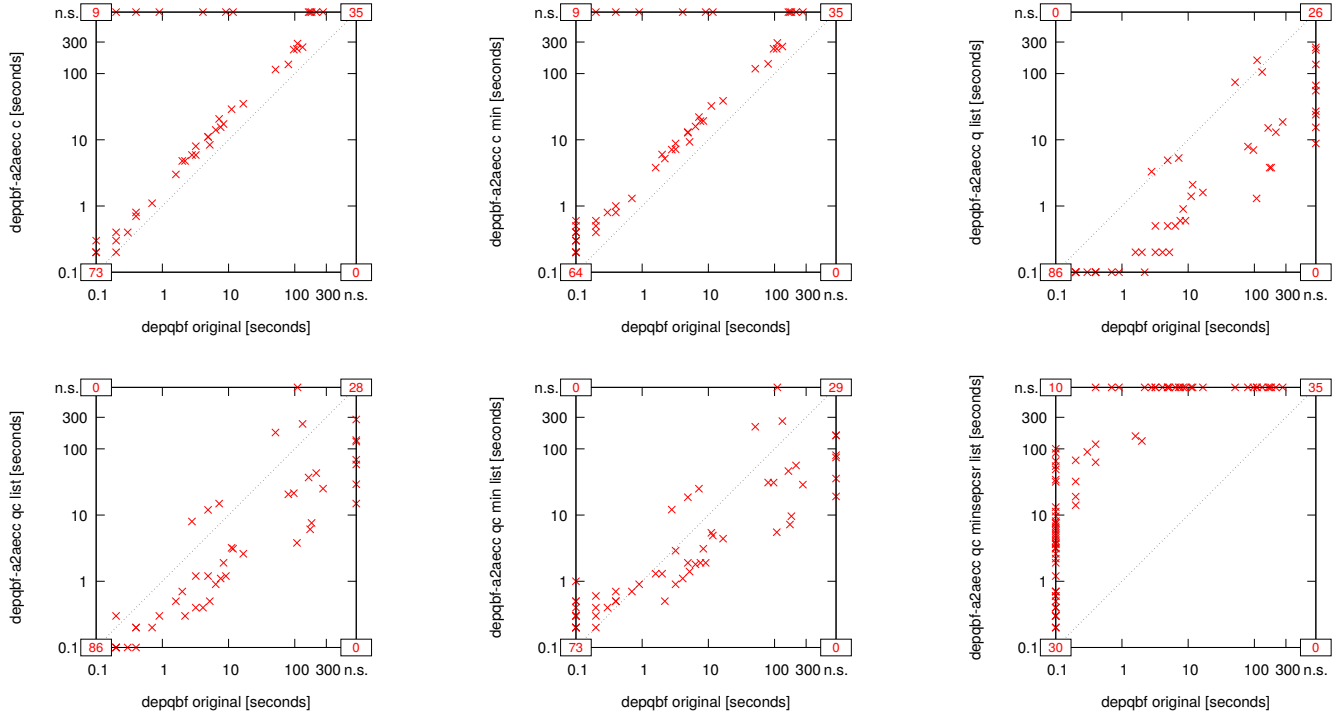


Fig. 941: Suite Miller-Scholl-Becker ($n = 160$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

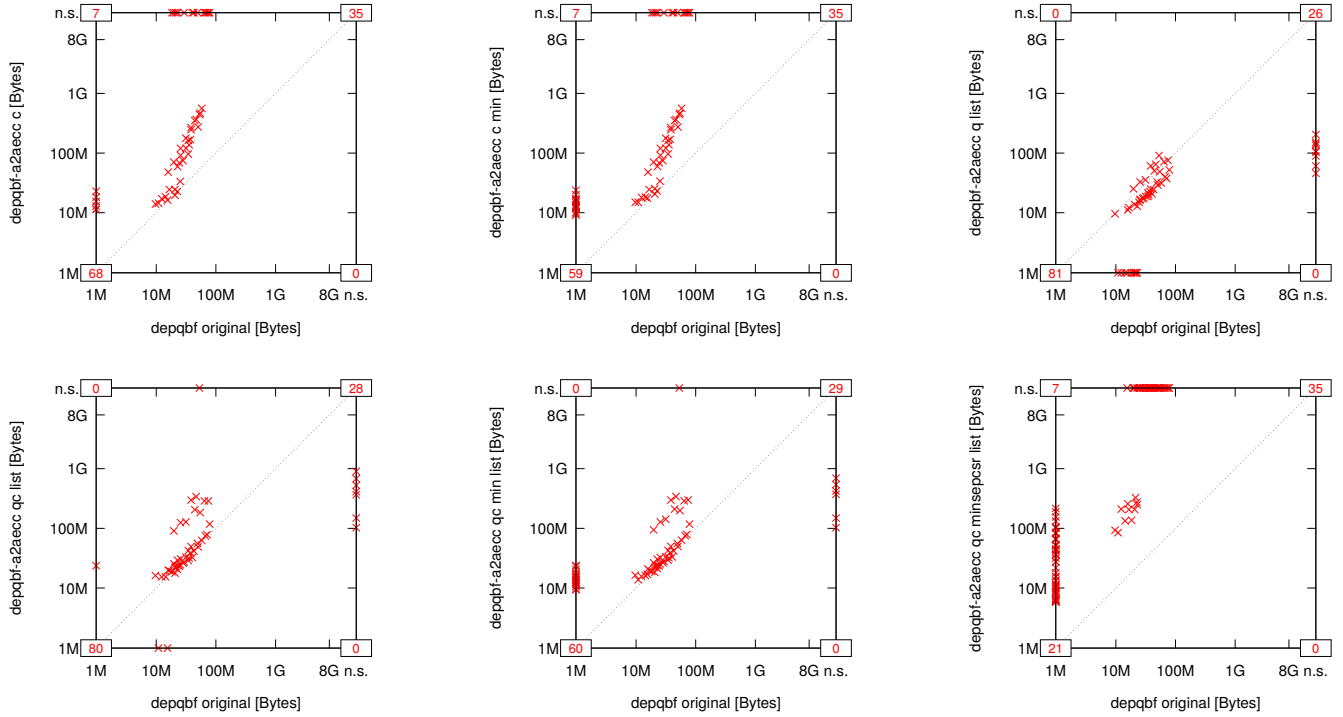


Fig. 942: Suite Miller-Scholl-Becker ($n = 160$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

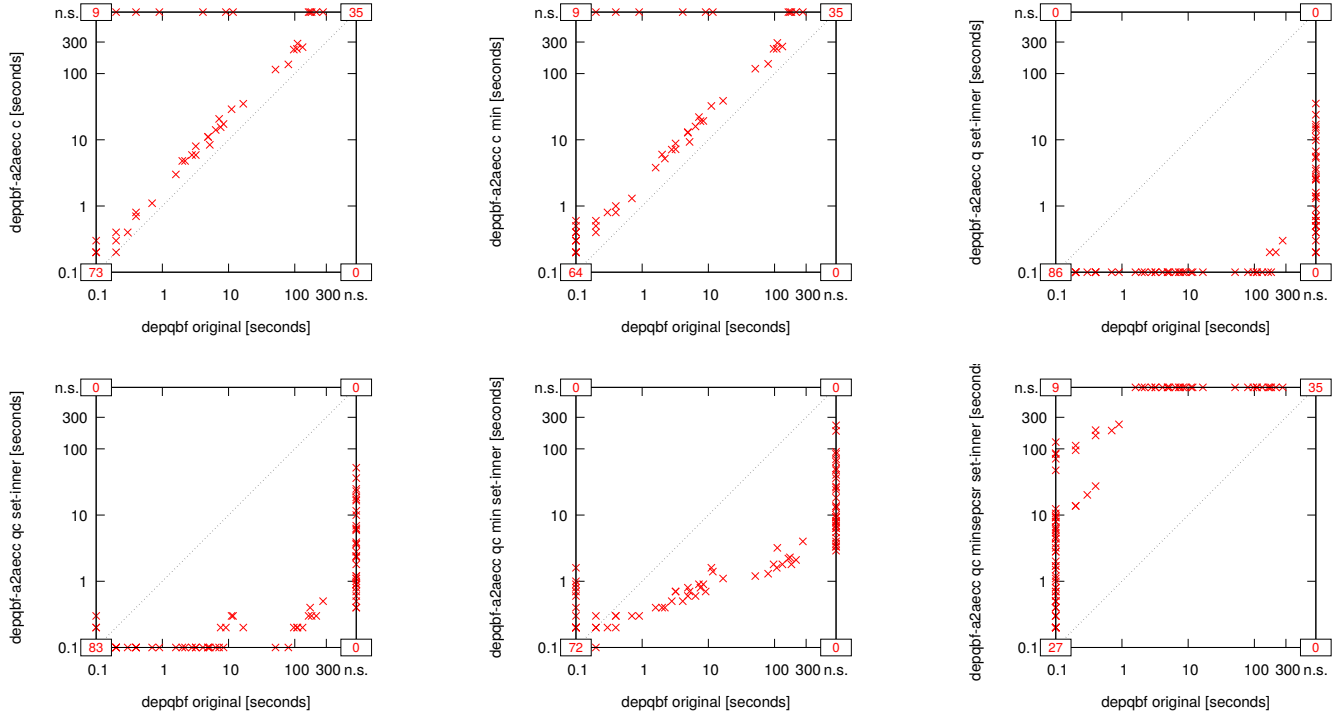


Fig. 943: Suite Miller-Scholl-Becker ($n = 160$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

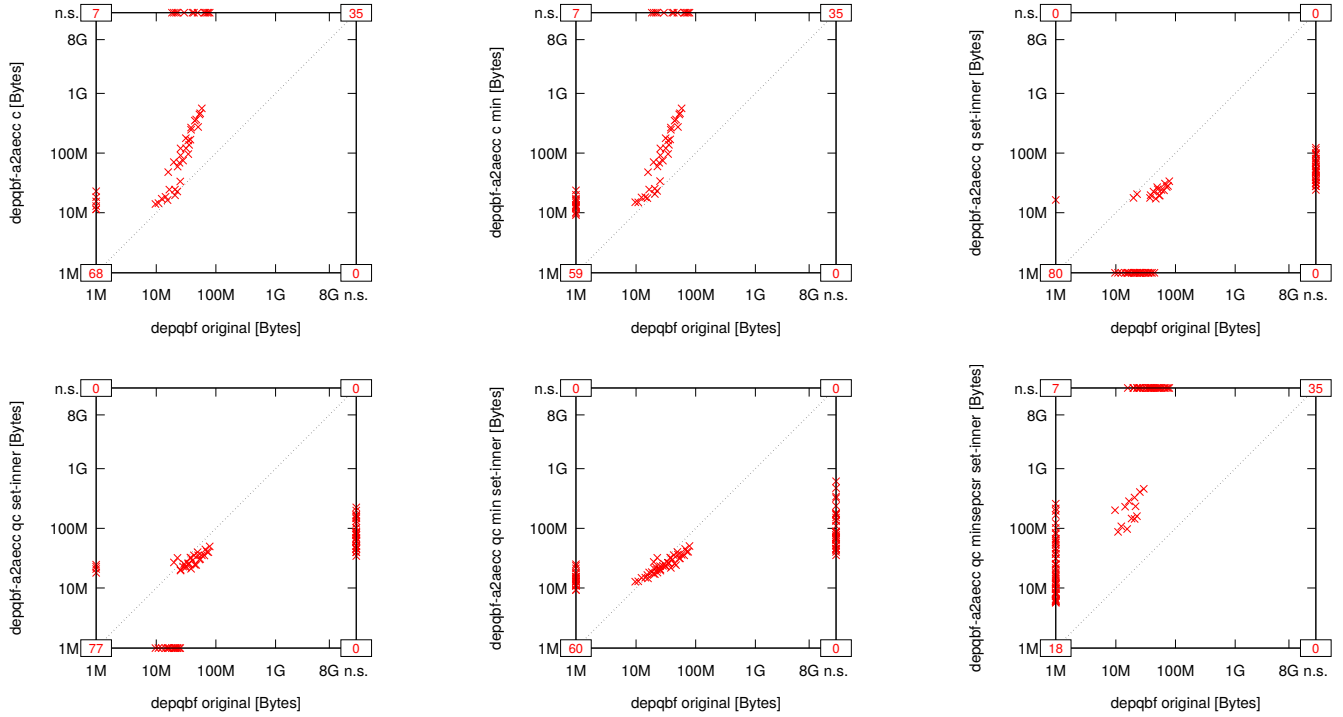


Fig. 944: Suite Miller-Scholl-Becker ($n = 160$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

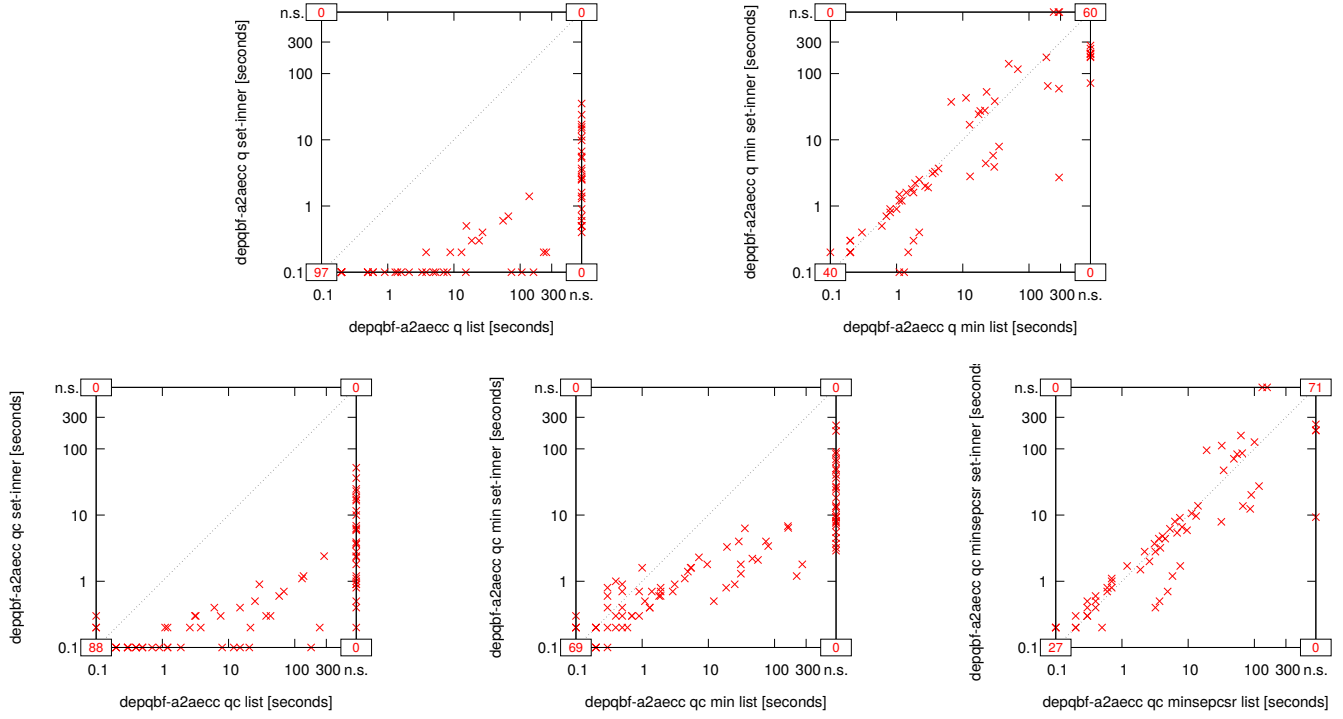


Fig. 945: Suite Miller-Scholl-Becker ($n = 160$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

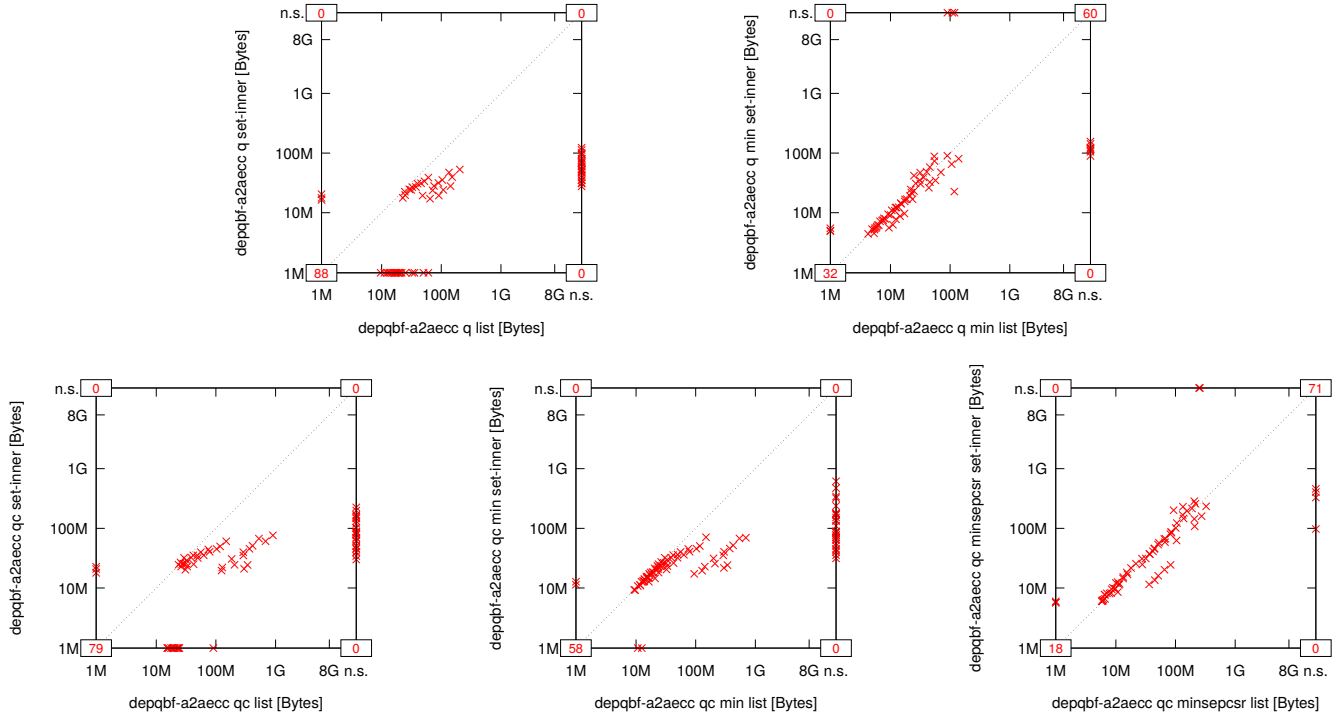


Fig. 946: Suite Miller-Scholl-Becker ($n = 160$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

32) Mneimneh-Sakallah ($n = 44$):

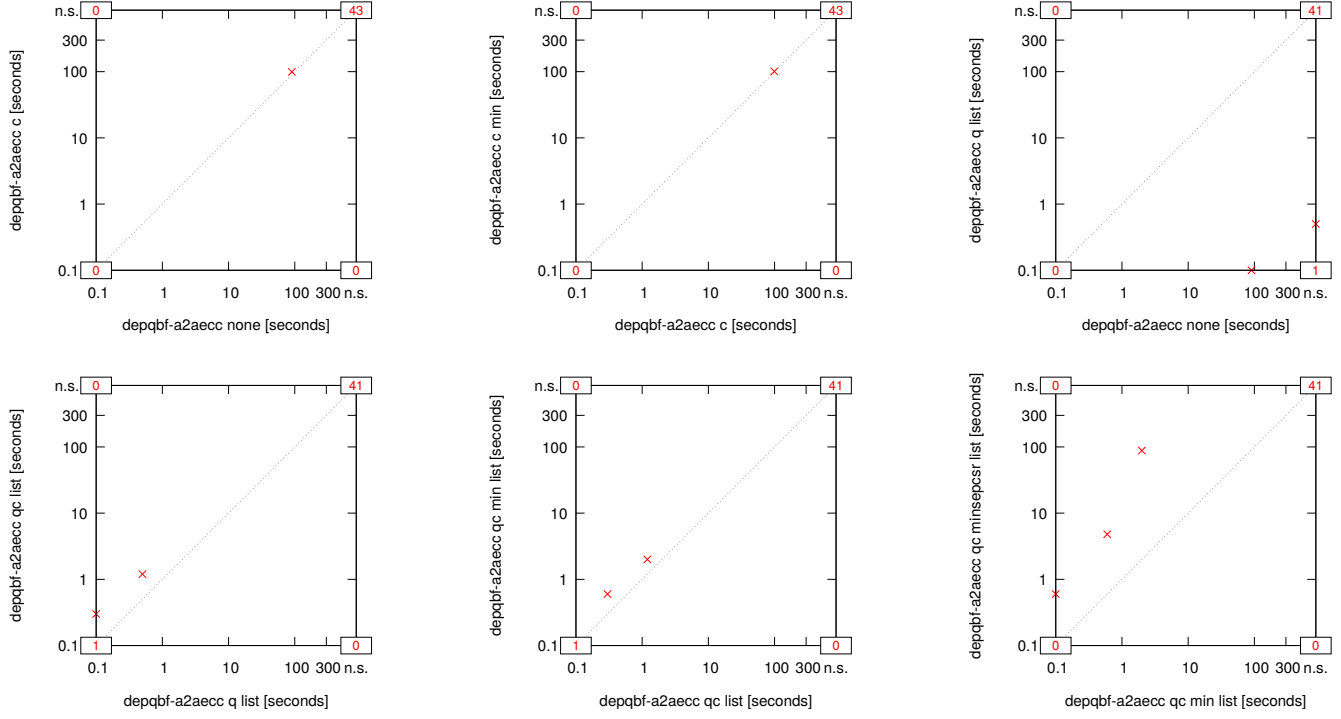


Fig. 947: Suite Mneimneh-Sakallah ($n = 44$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

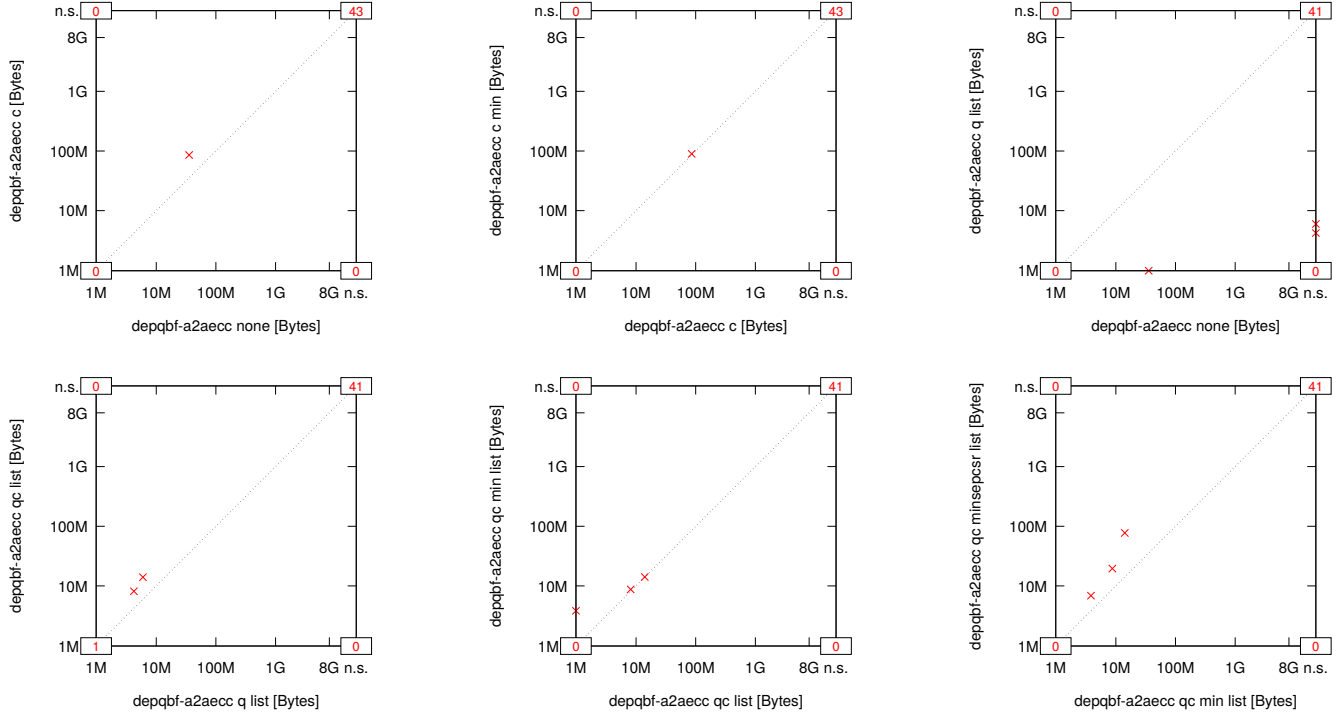


Fig. 948: Suite Mneimneh-Sakallah ($n = 44$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

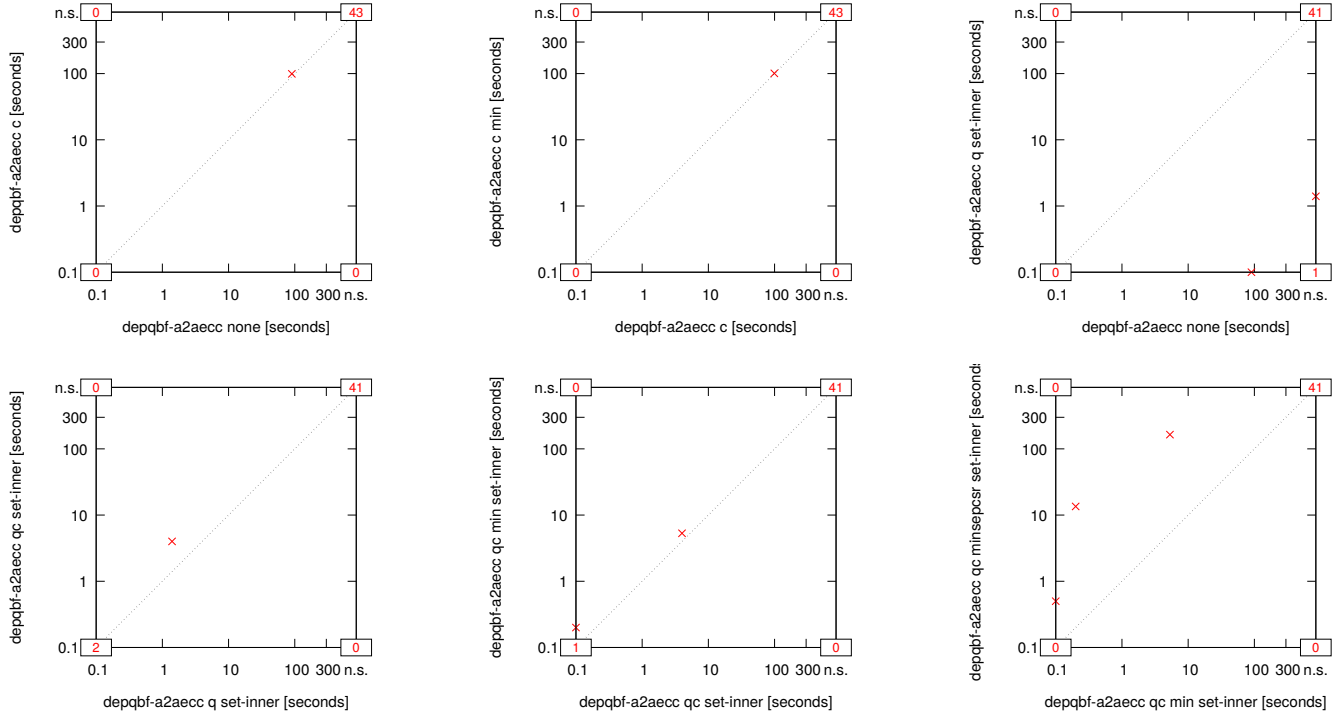


Fig. 949: Suite Mneimneh-Sakallah ($n = 44$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

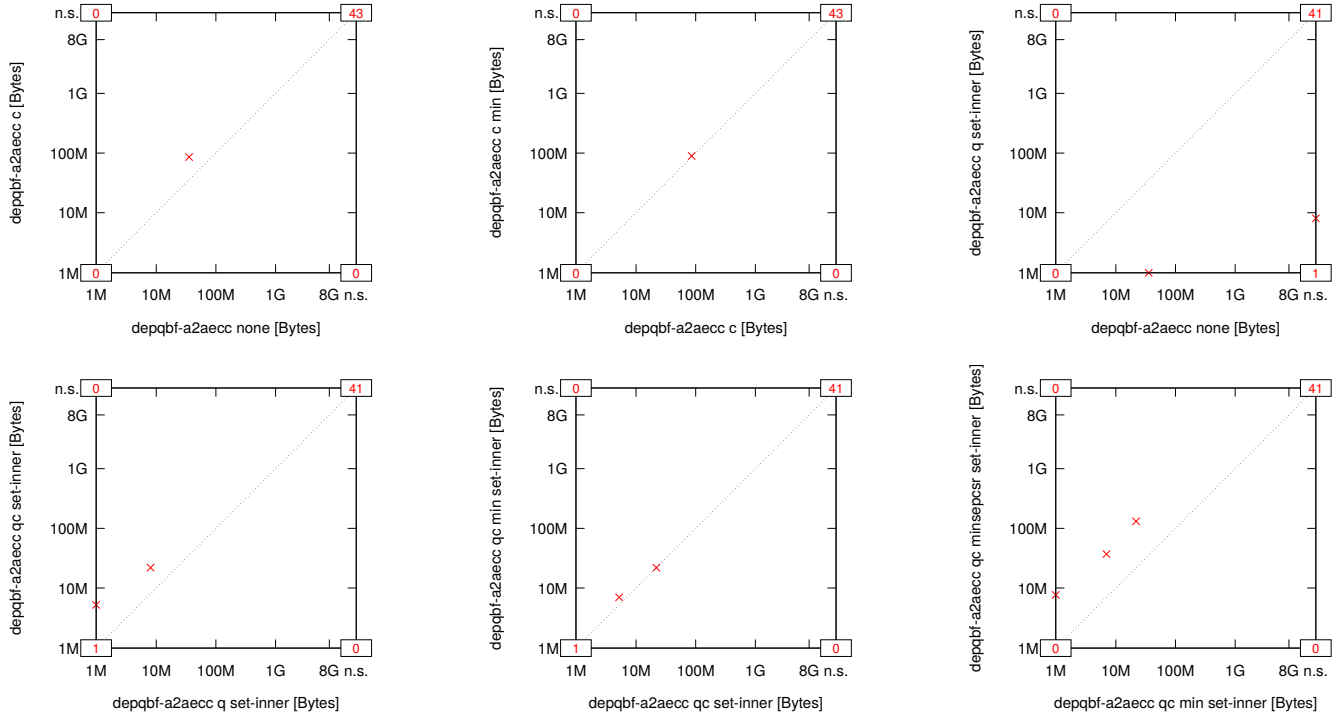


Fig. 950: Suite Mneimneh-Sakallah ($n = 44$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

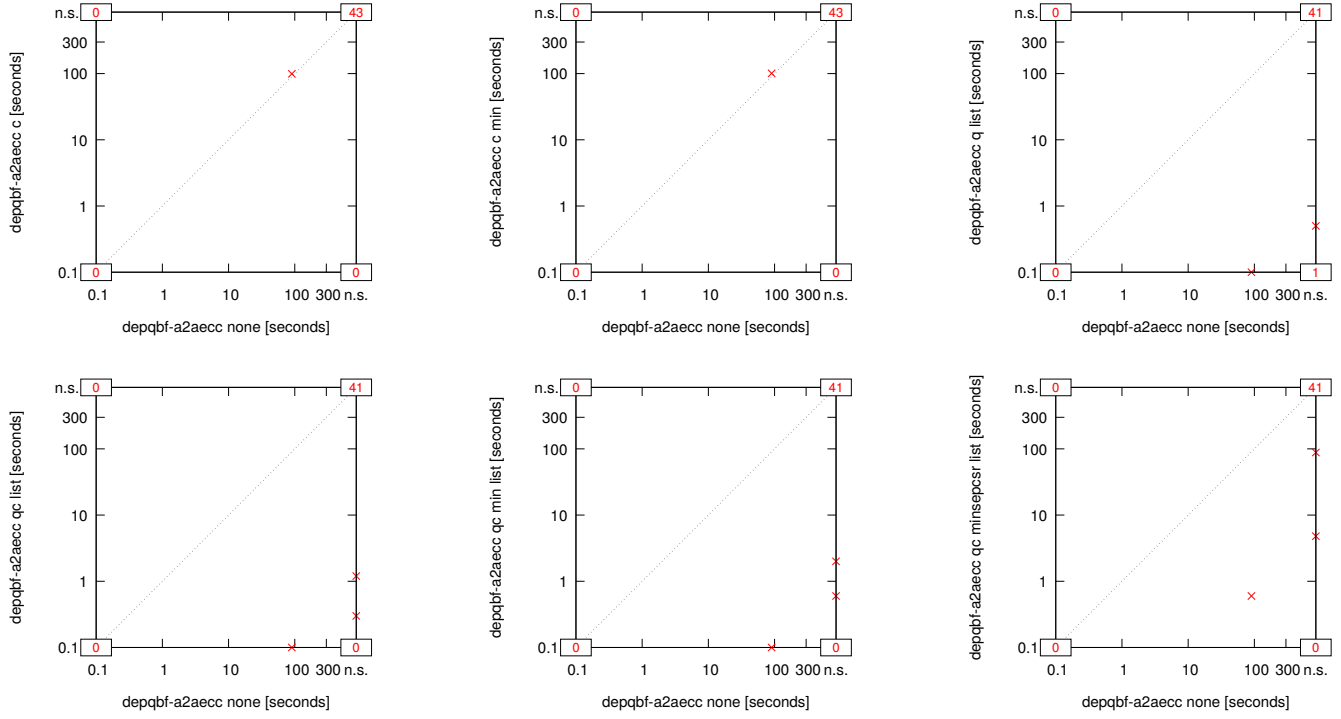


Fig. 951: Suite Mneimneh-Sakallah ($n = 44$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

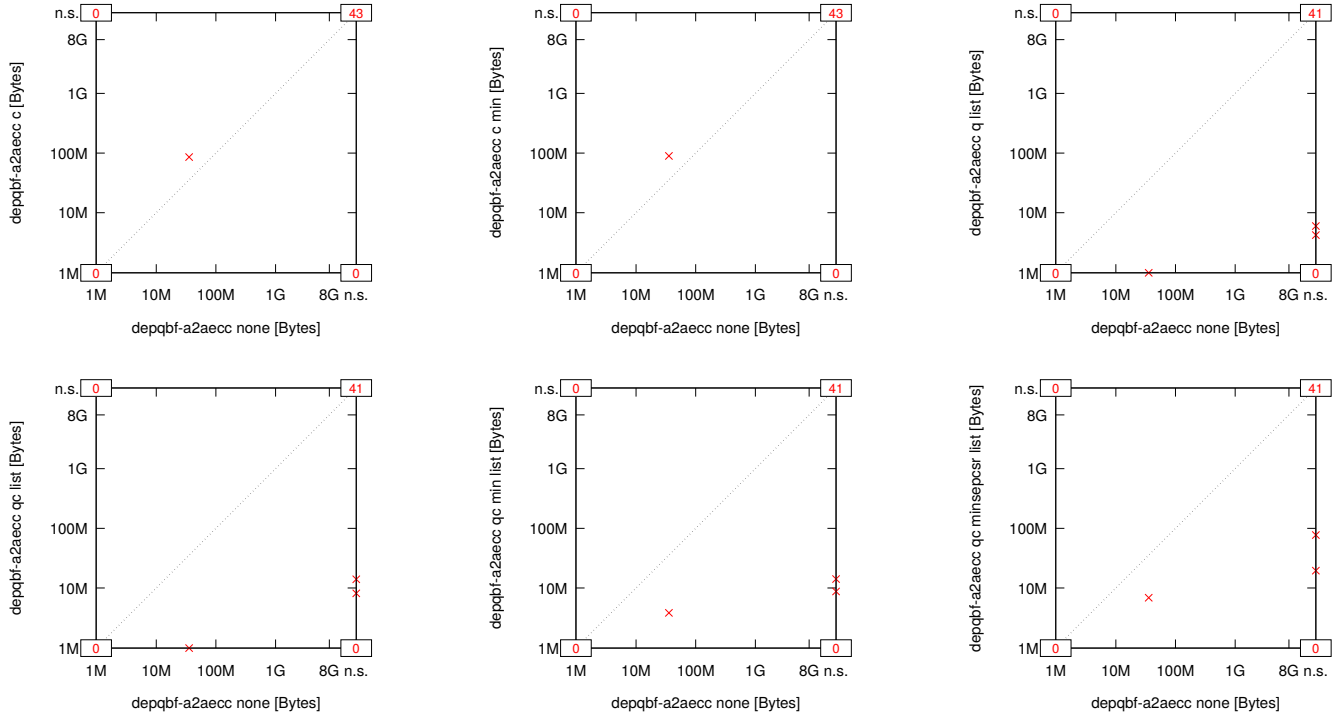


Fig. 952: Suite Mneimneh-Sakallah ($n = 44$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

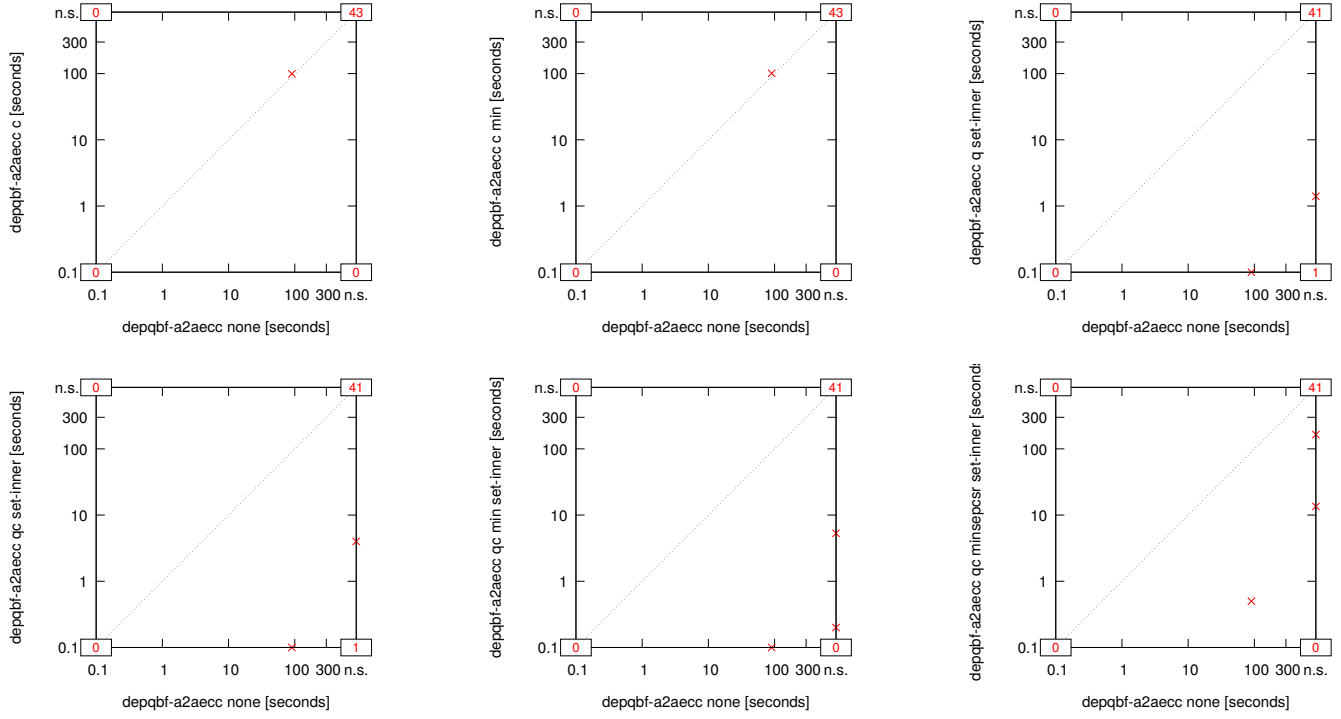


Fig. 953: Suite Mneimneh-Sakallah ($n = 44$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. Dep-QBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

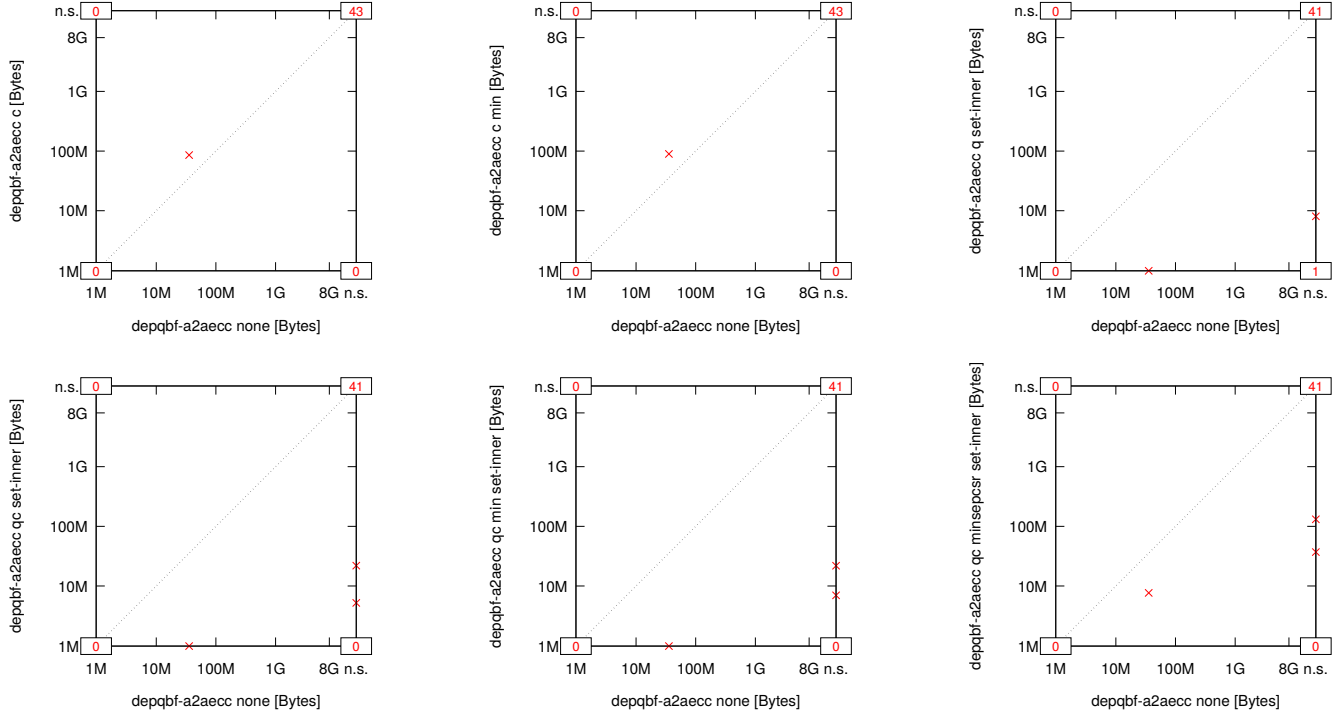


Fig. 954: Suite Mneimneh-Sakallah ($n = 44$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. Dep-QBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

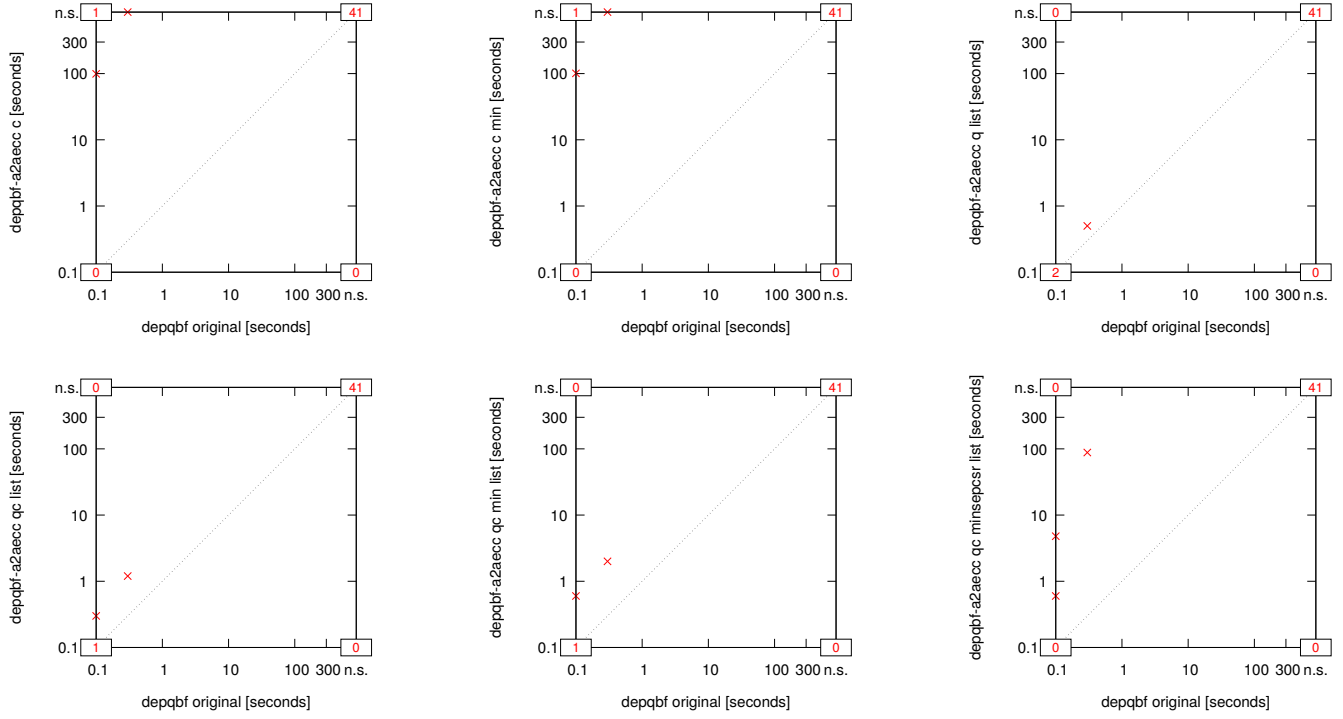


Fig. 955: Suite Mneimneh-Sakallah ($n = 44$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

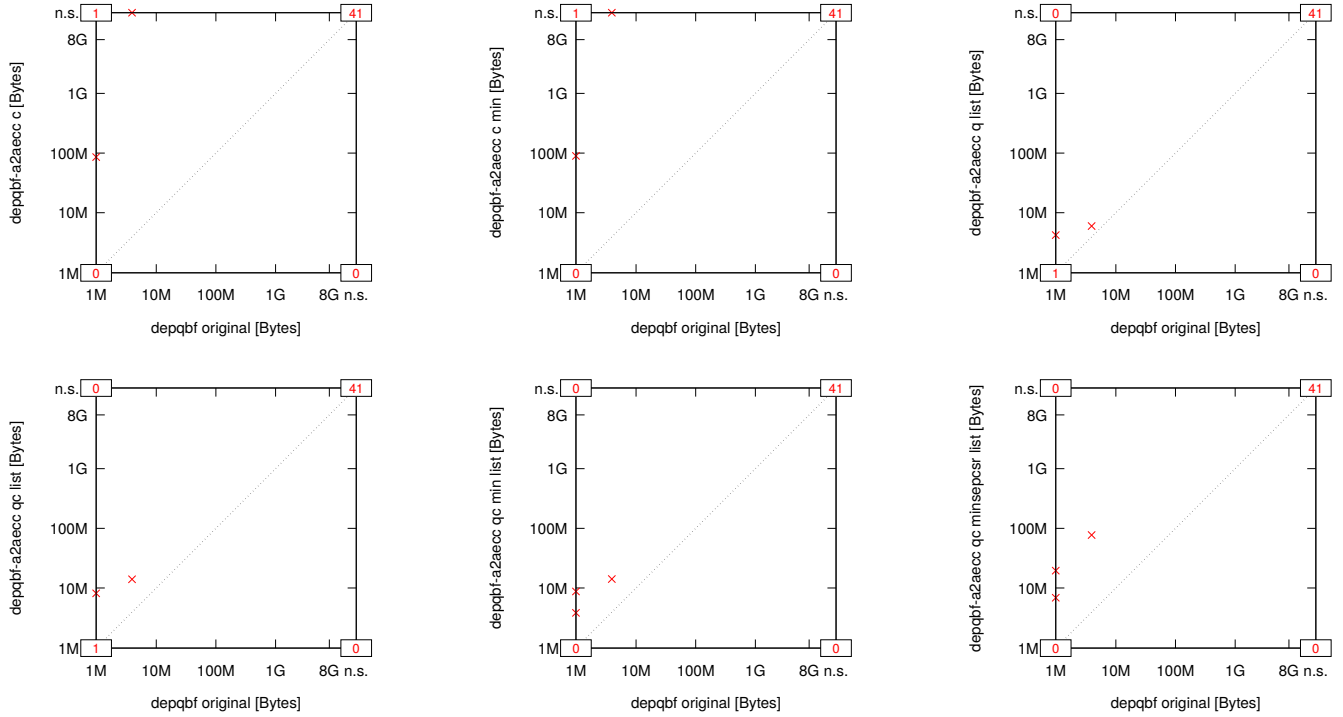


Fig. 956: Suite Mneimneh-Sakallah ($n = 44$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

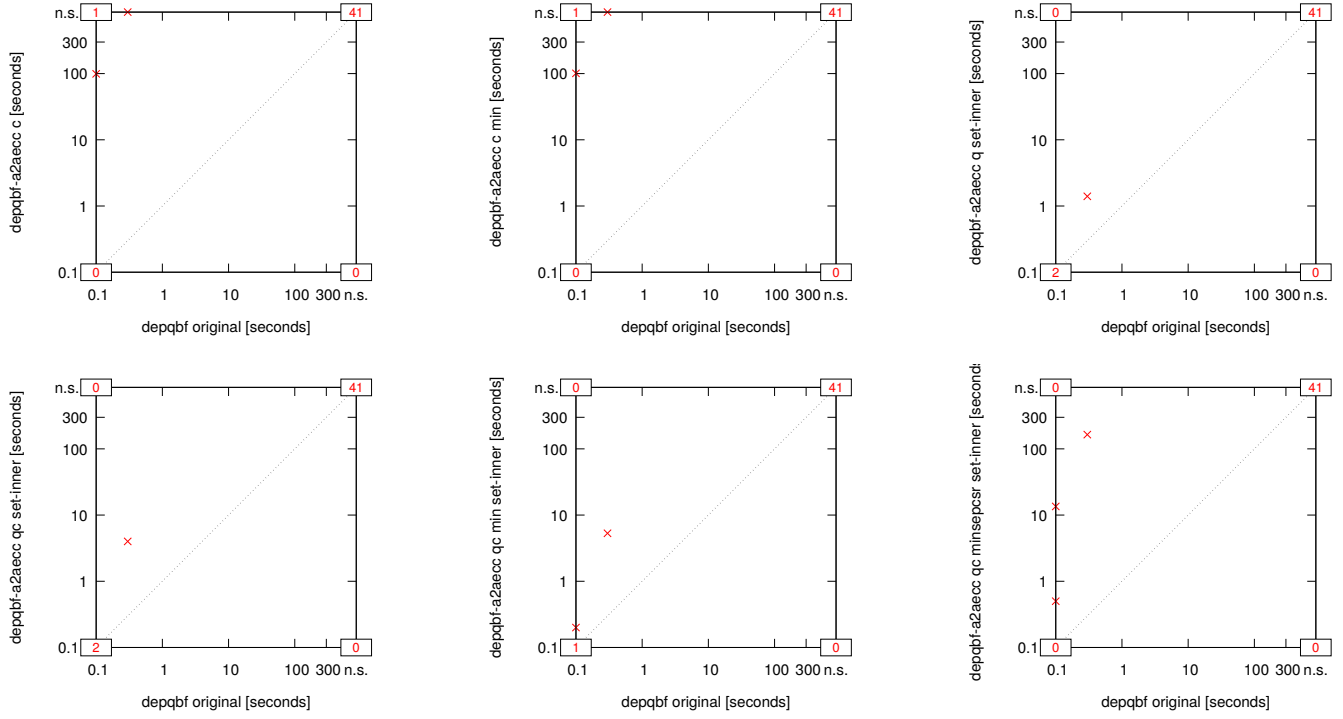


Fig. 957: Suite Mneimneh-Sakallah ($n = 44$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

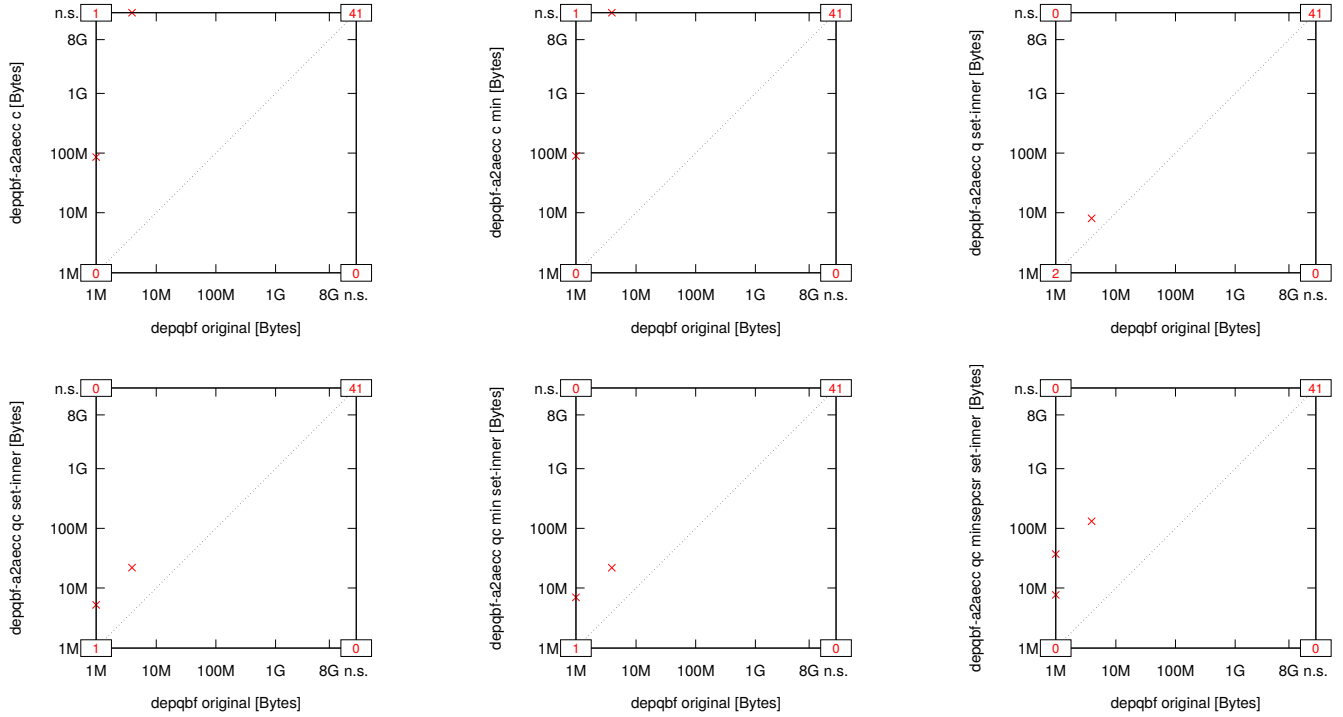


Fig. 958: Suite Mneimneh-Sakallah ($n = 44$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

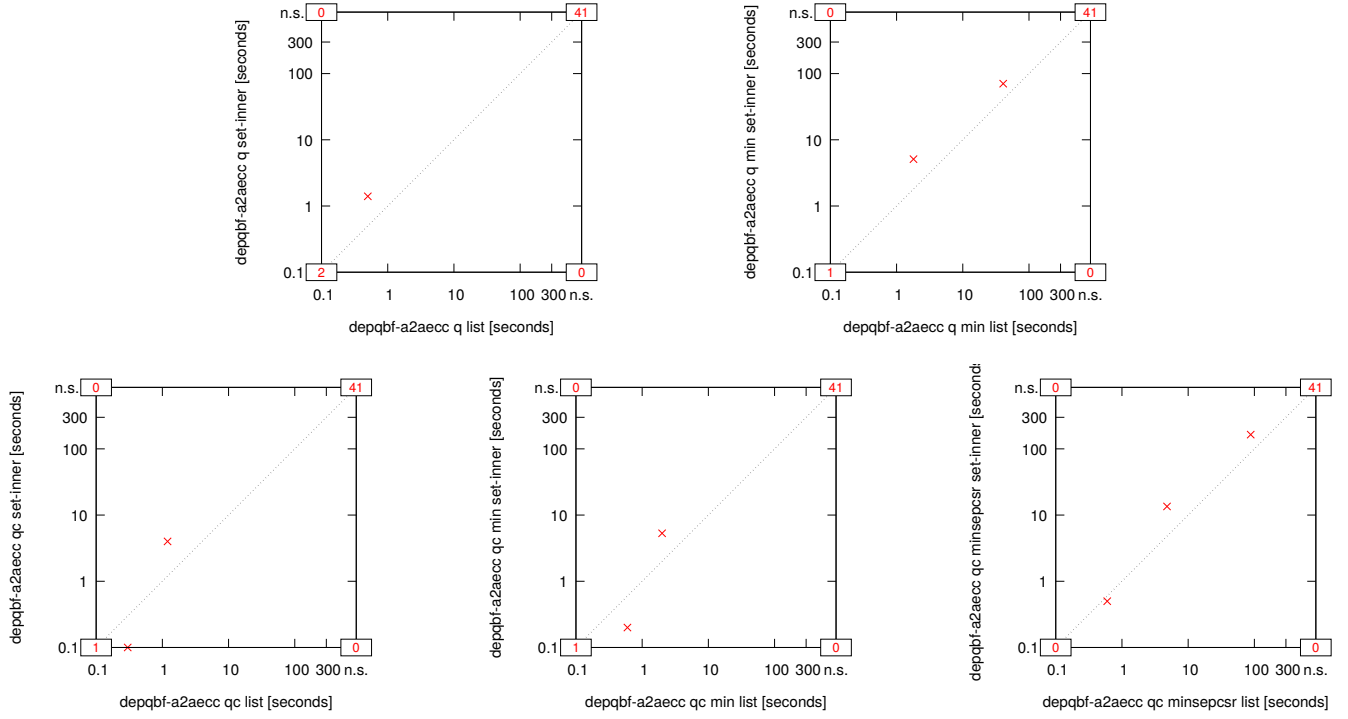


Fig. 959: Suite Mneimneh-Sakallah ($n = 44$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

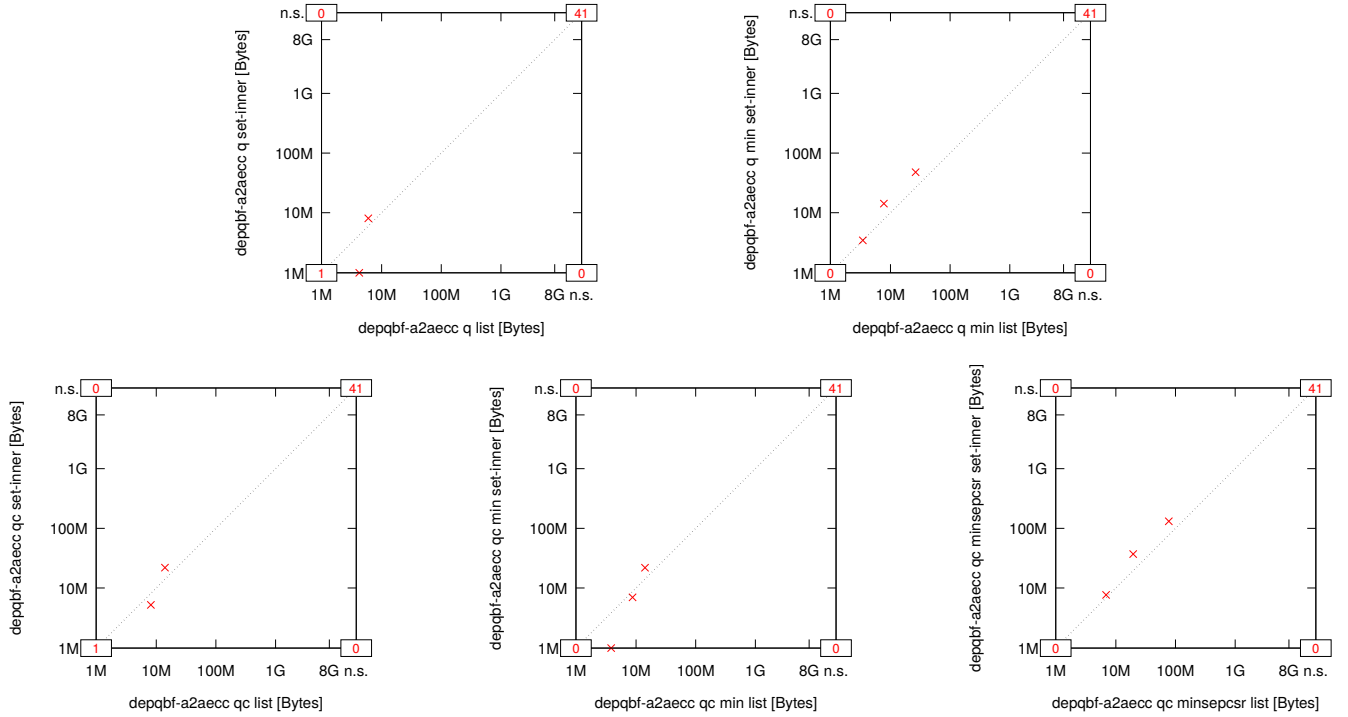


Fig. 960: Suite Mneimneh-Sakallah ($n = 44$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

33) Narizzano ($n = 78$):

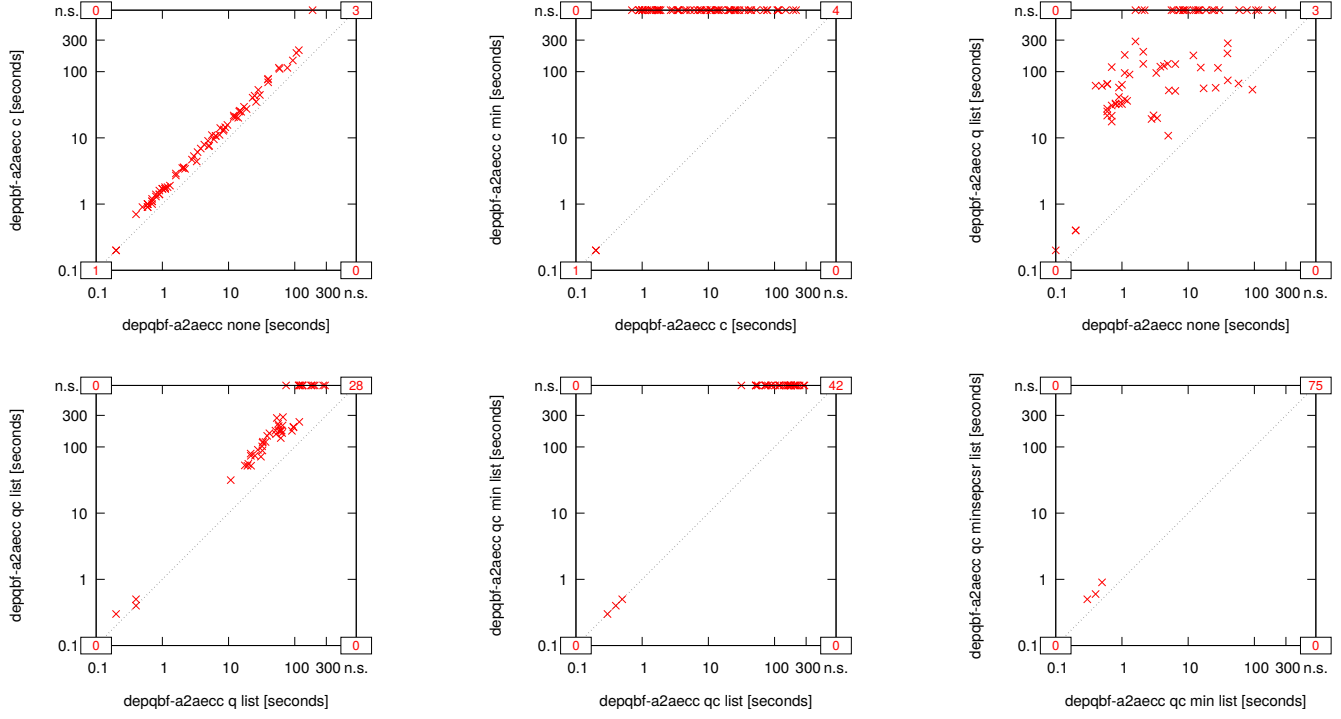


Fig. 961: Suite Narizzano ($n = 78$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

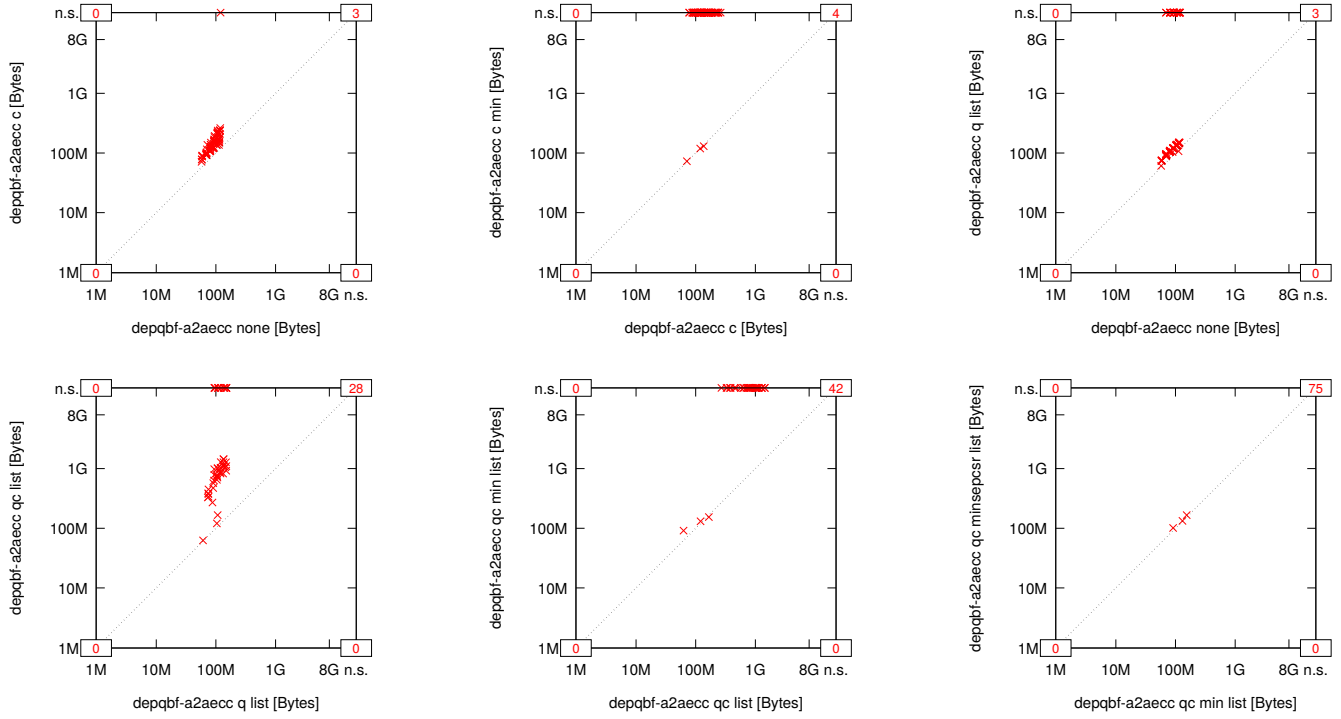


Fig. 962: Suite Narizzano ($n = 78$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

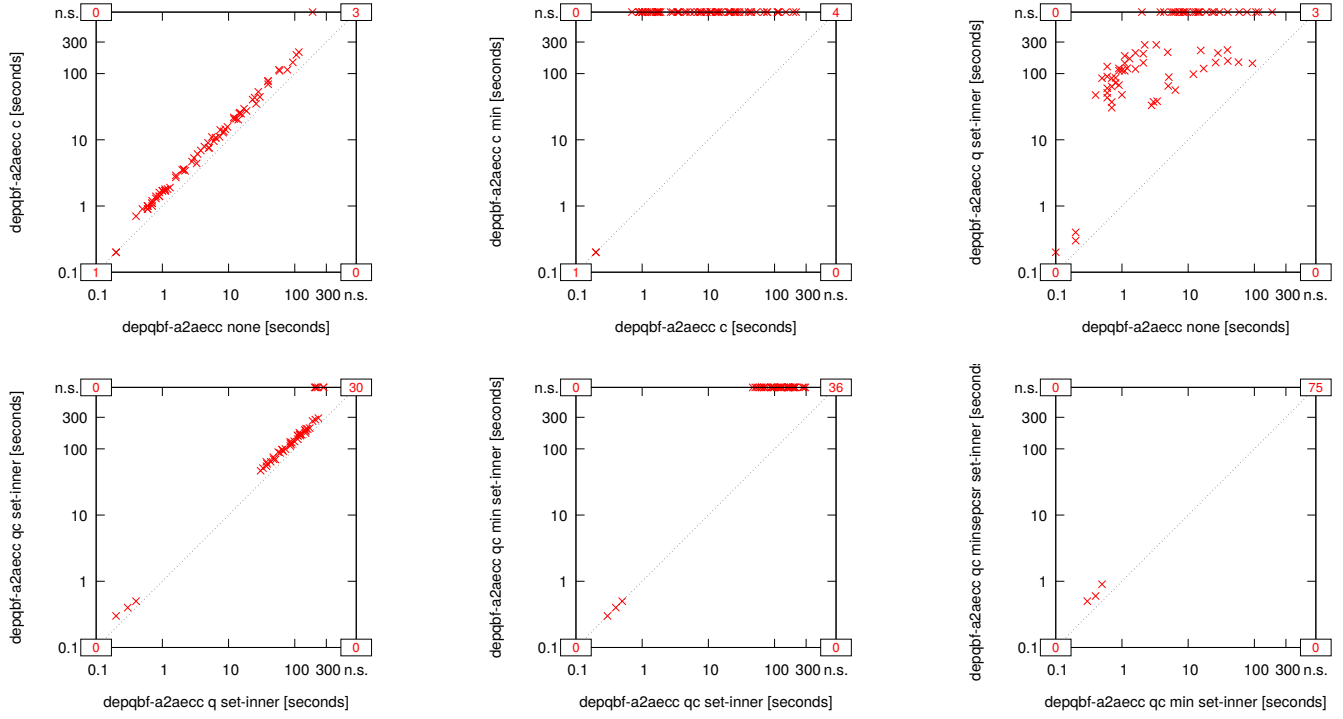


Fig. 963: Suite Narizzano ($n = 78$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

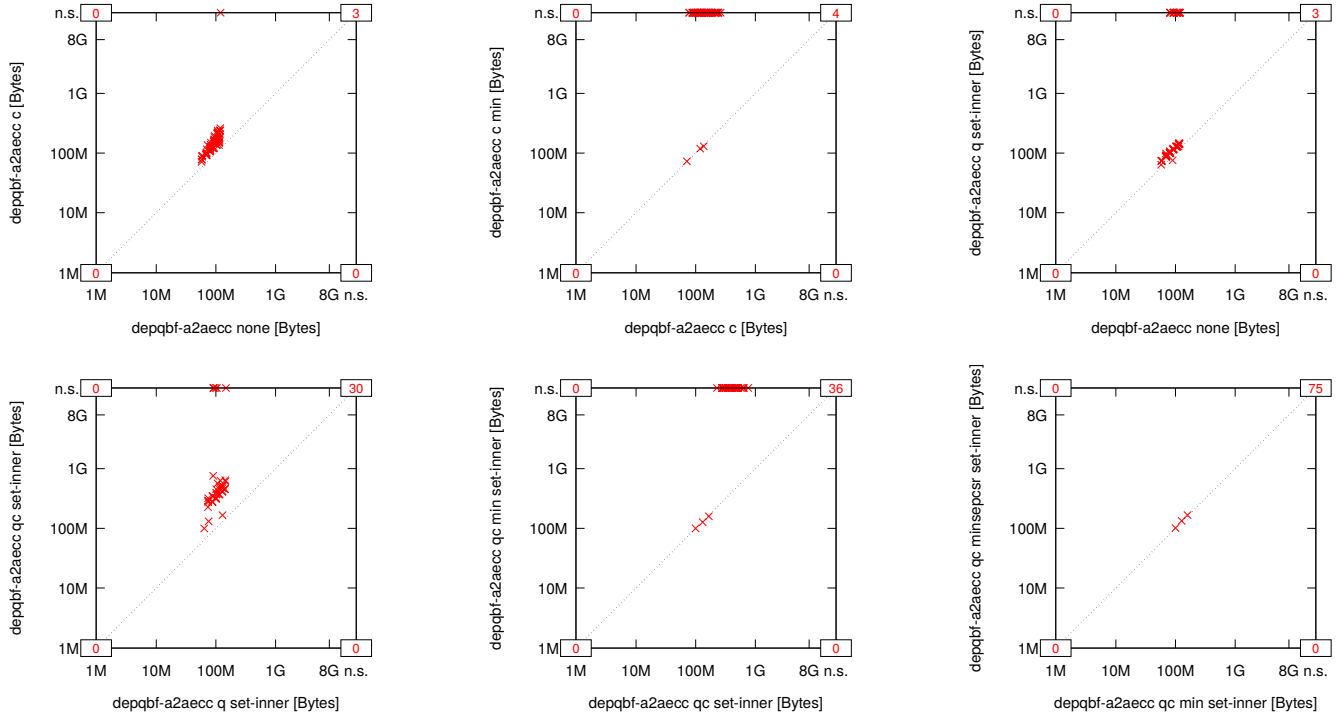


Fig. 964: Suite Narizzano ($n = 78$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

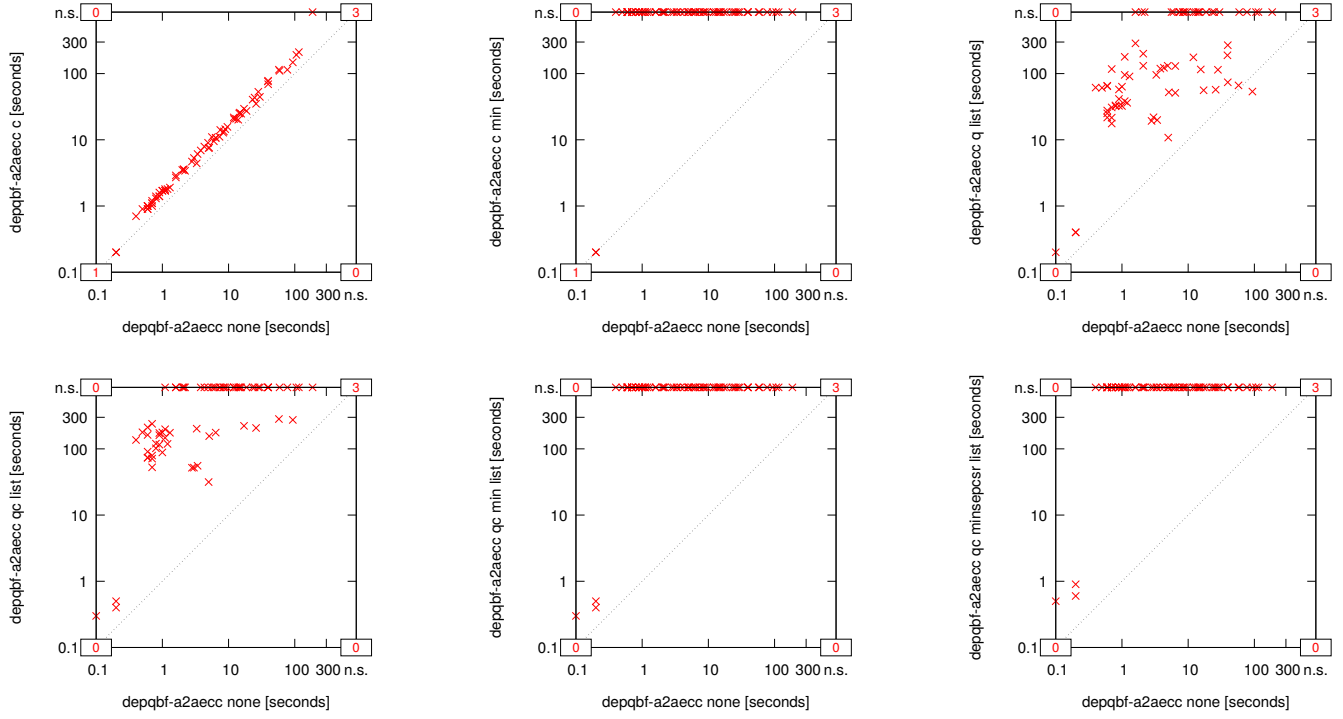


Fig. 965: Suite Narizzano ($n = 78$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

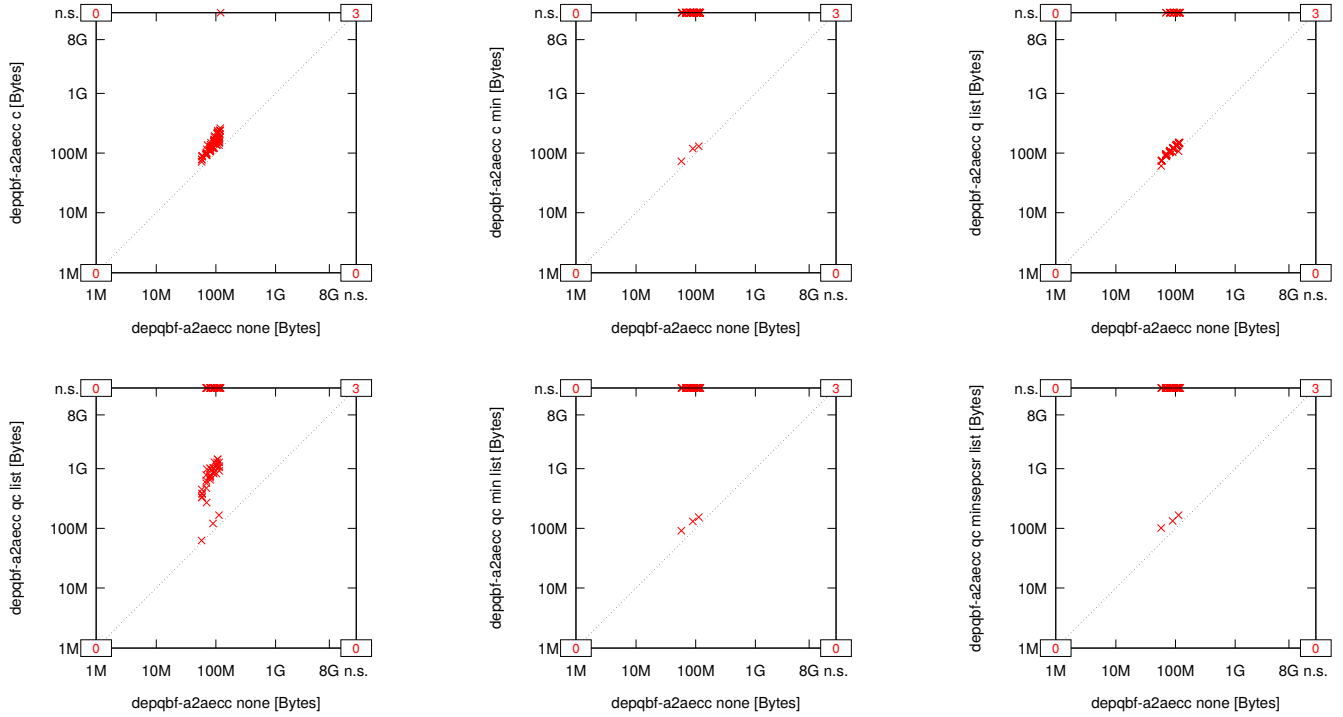


Fig. 966: Suite Narizzano ($n = 78$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

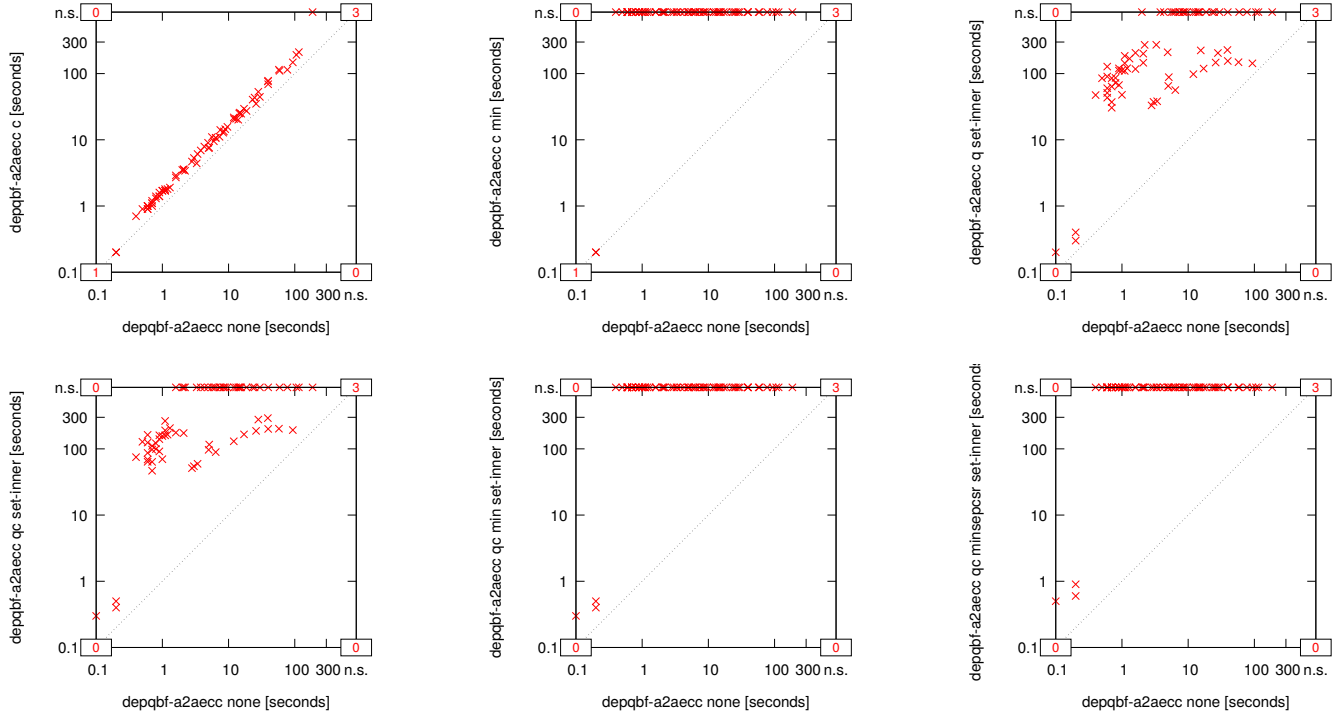


Fig. 967: Suite Narizzano ($n = 78$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

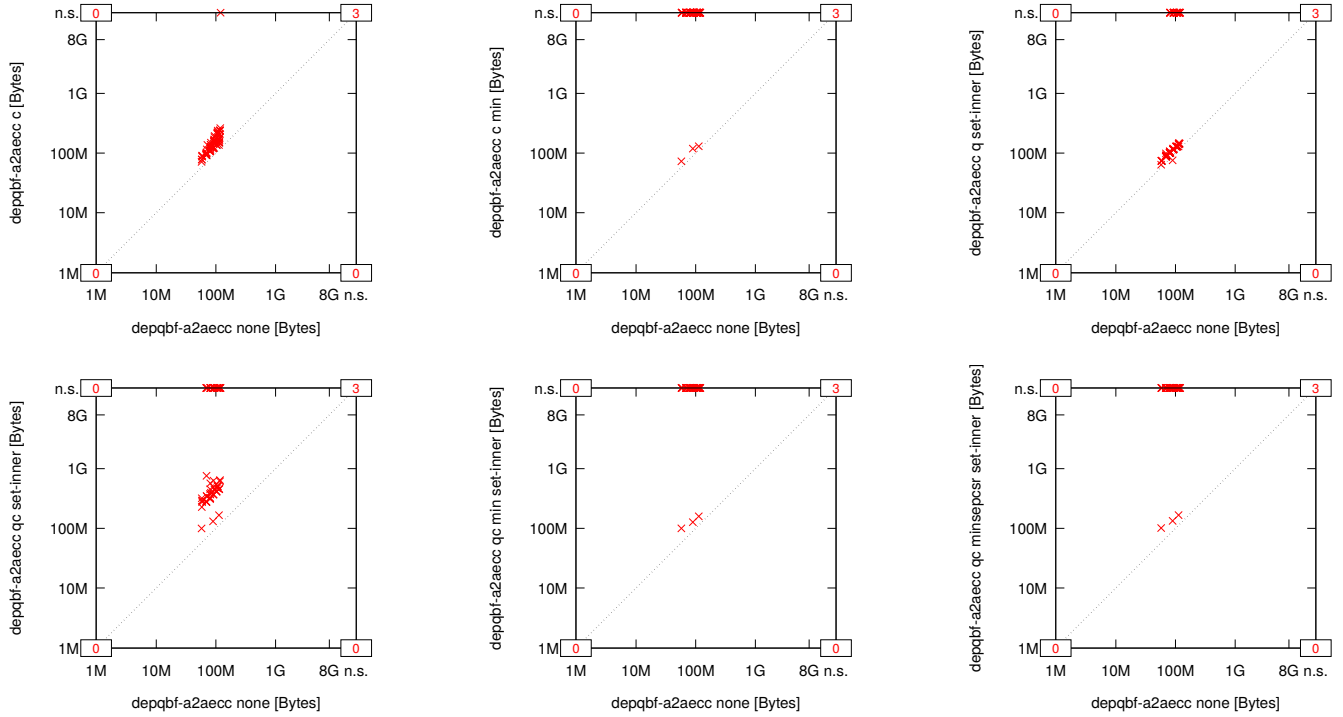


Fig. 968: Suite Narizzano ($n = 78$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

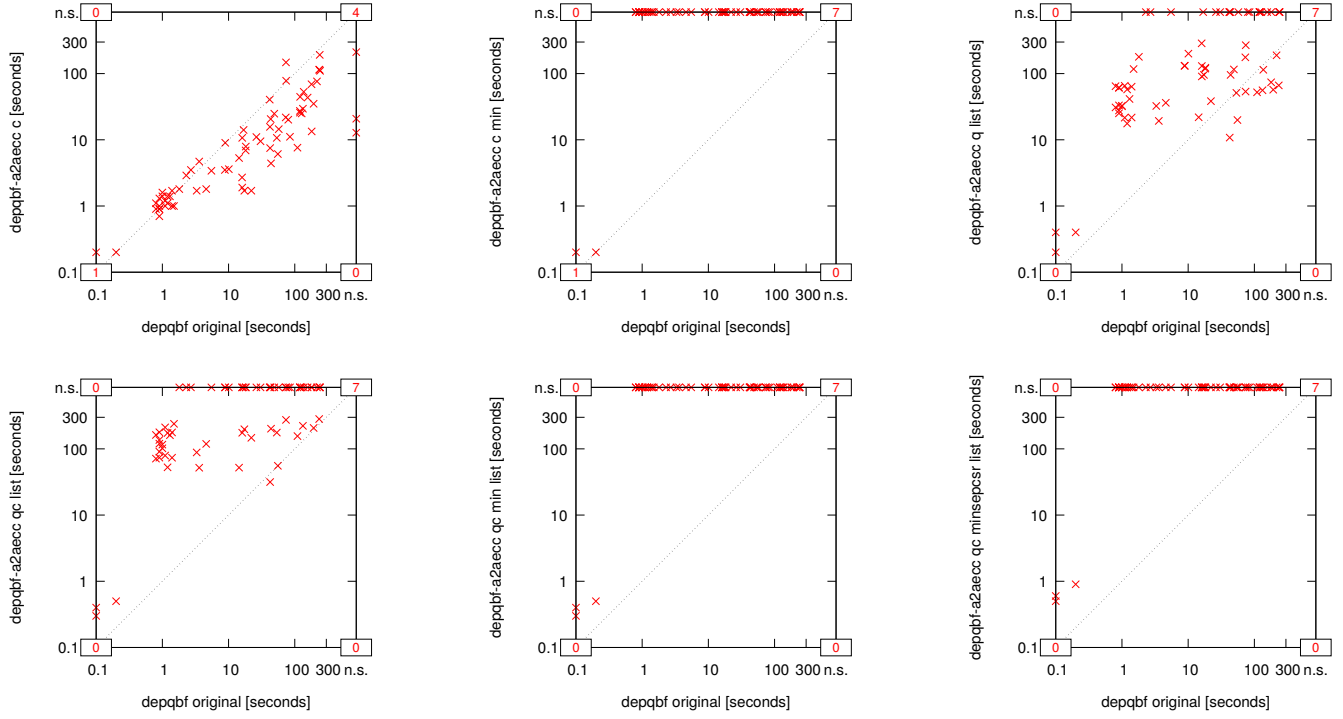


Fig. 969: Suite Narizzano ($n = 78$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

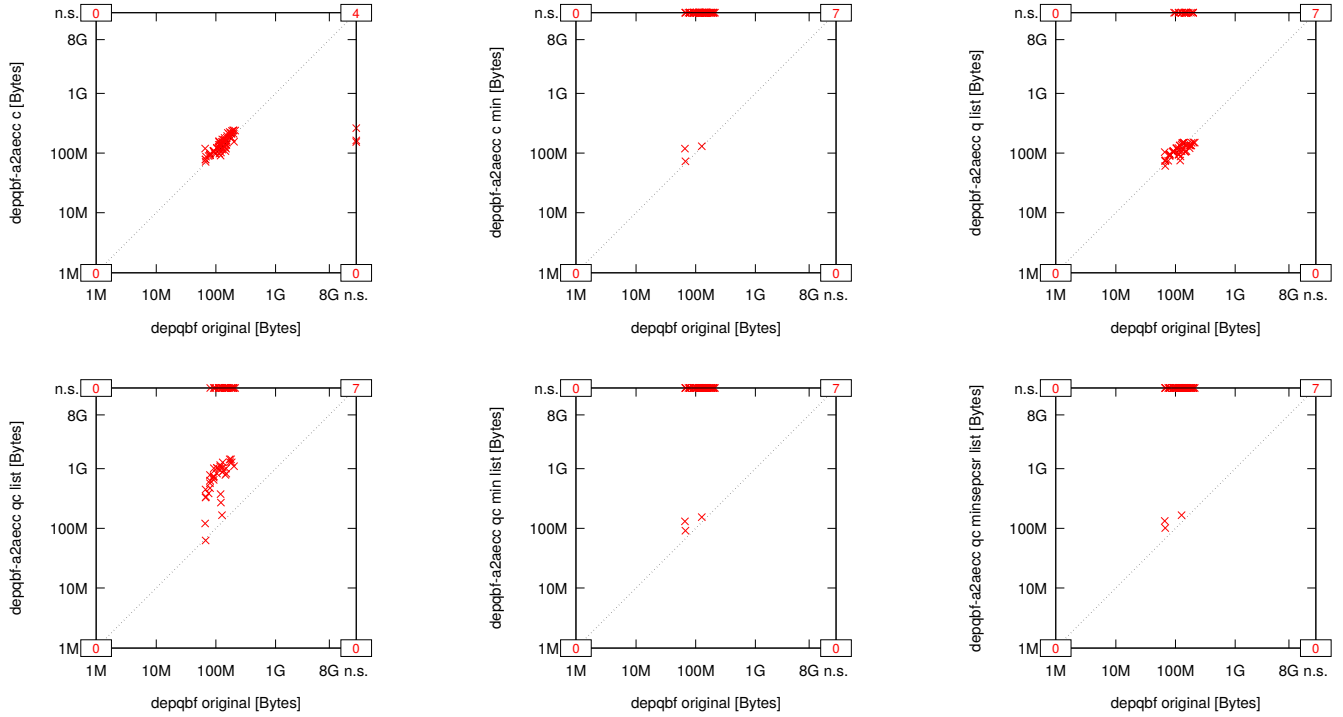


Fig. 970: Suite Narizzano ($n = 78$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

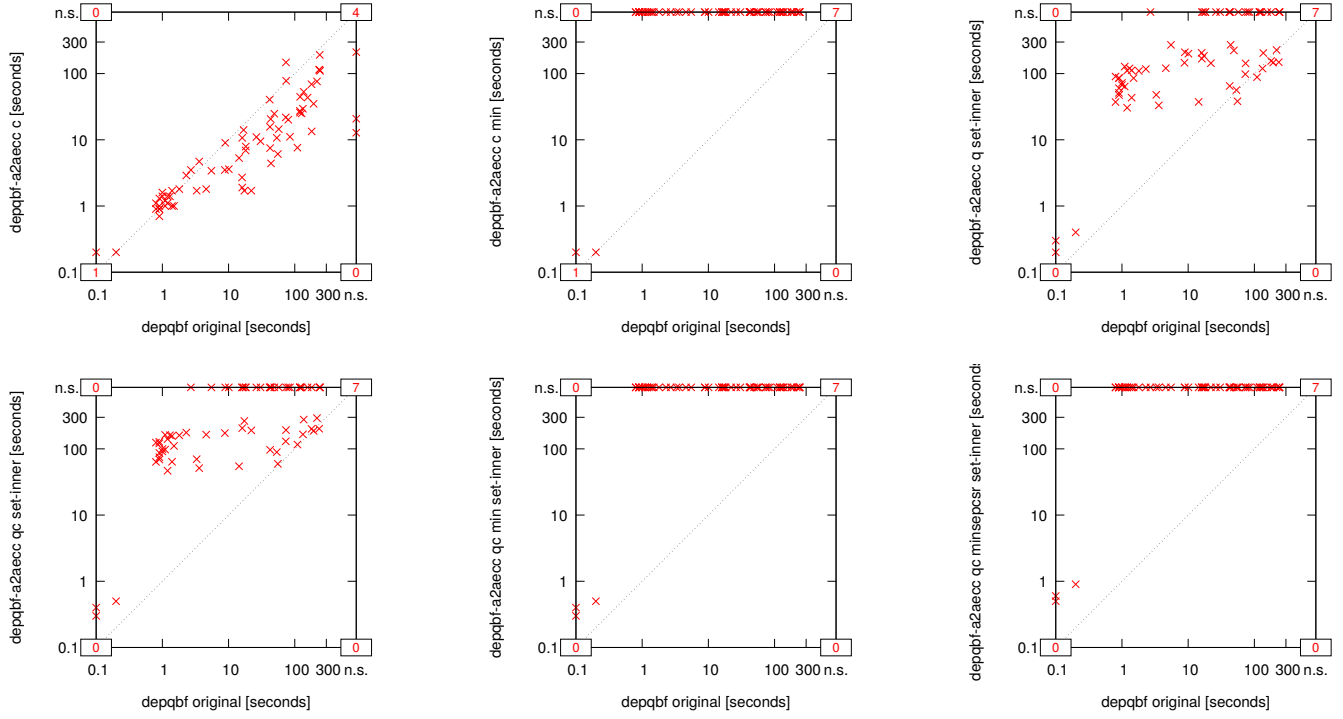


Fig. 971: Suite Narizzano ($n = 78$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

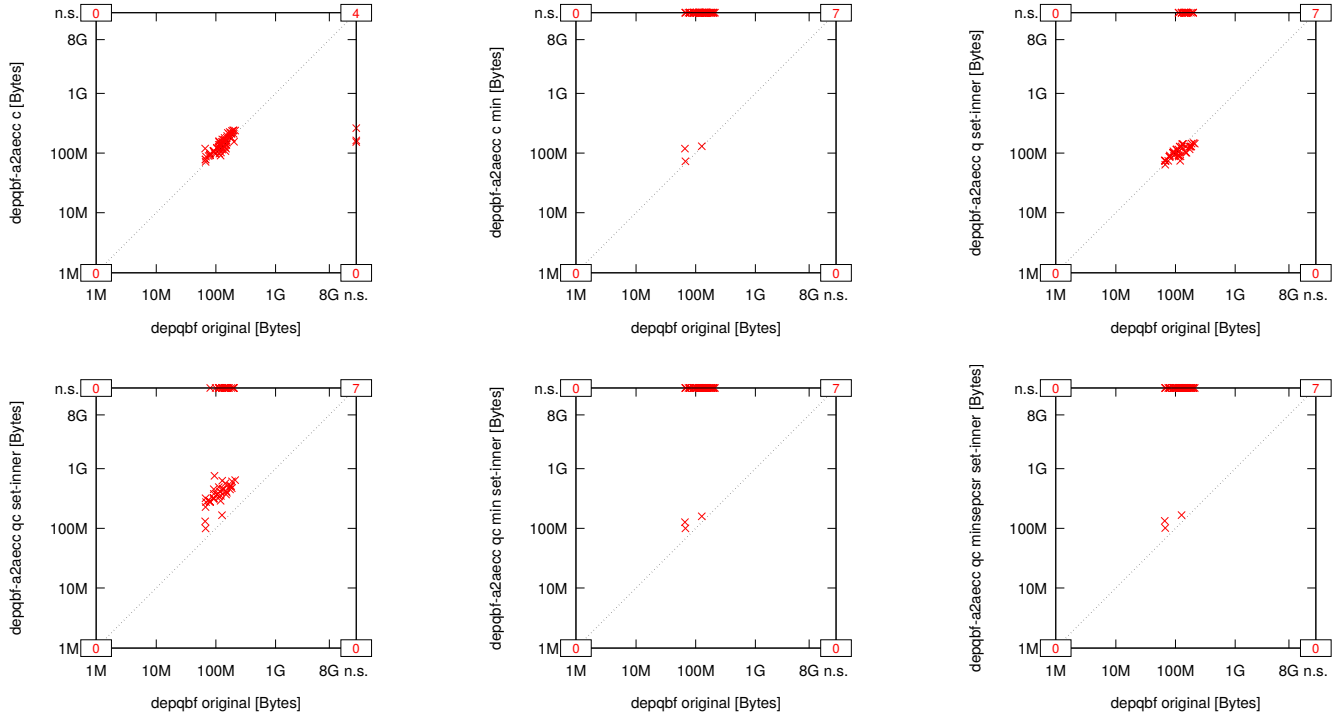


Fig. 972: Suite Narizzano ($n = 78$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

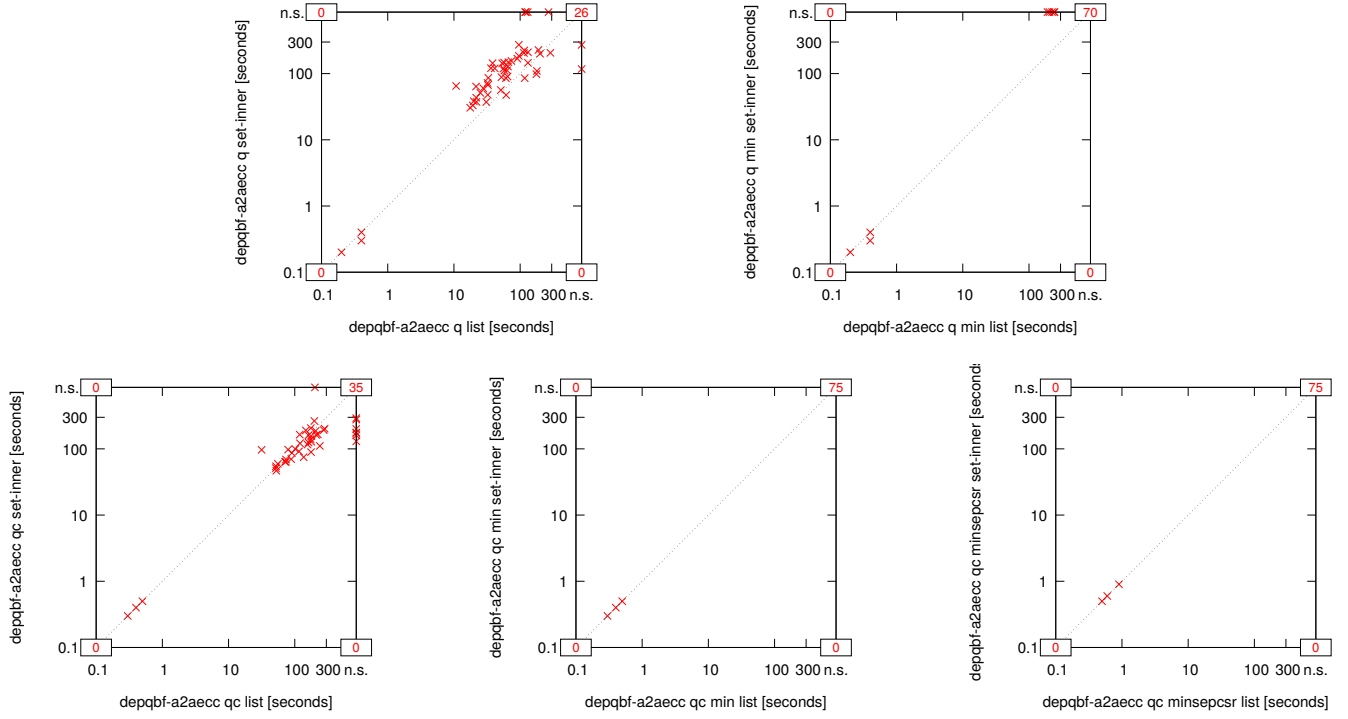


Fig. 973: Suite Narizzano ($n = 78$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

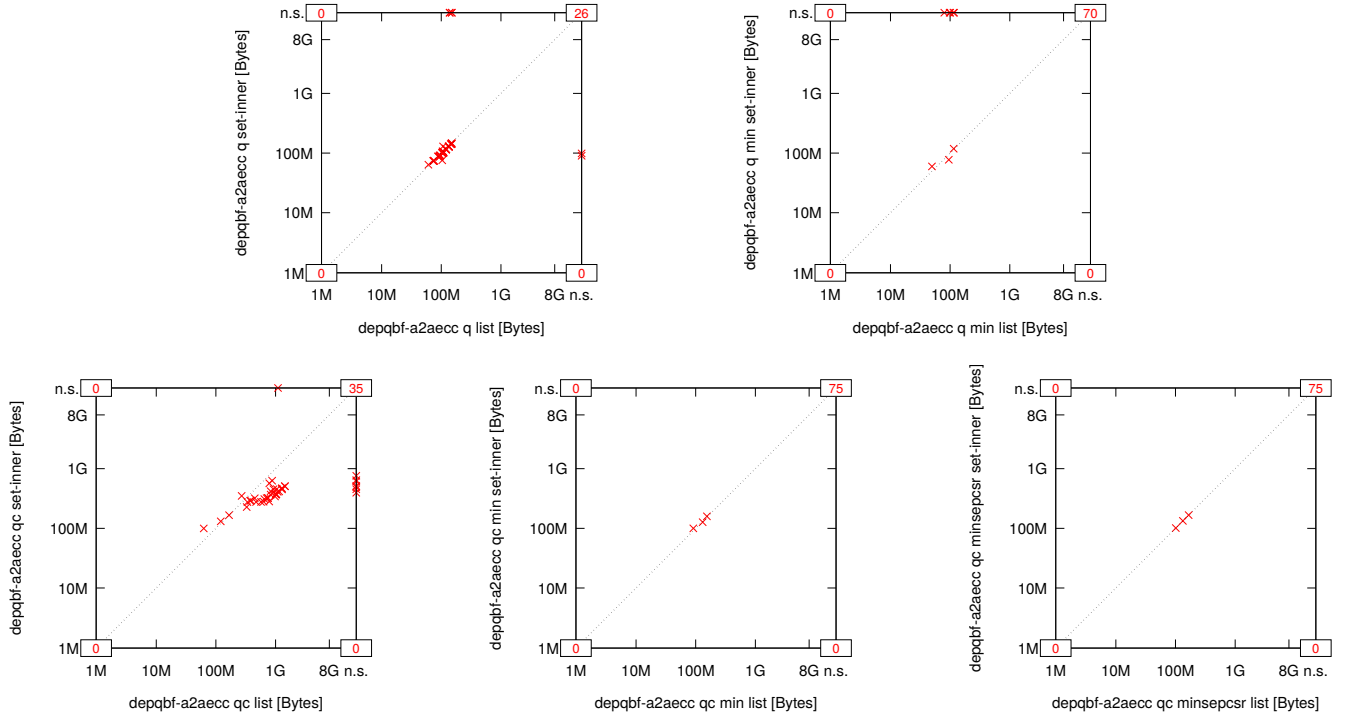


Fig. 974: Suite Narizzano ($n = 78$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

34) Palacios ($n = 9$):

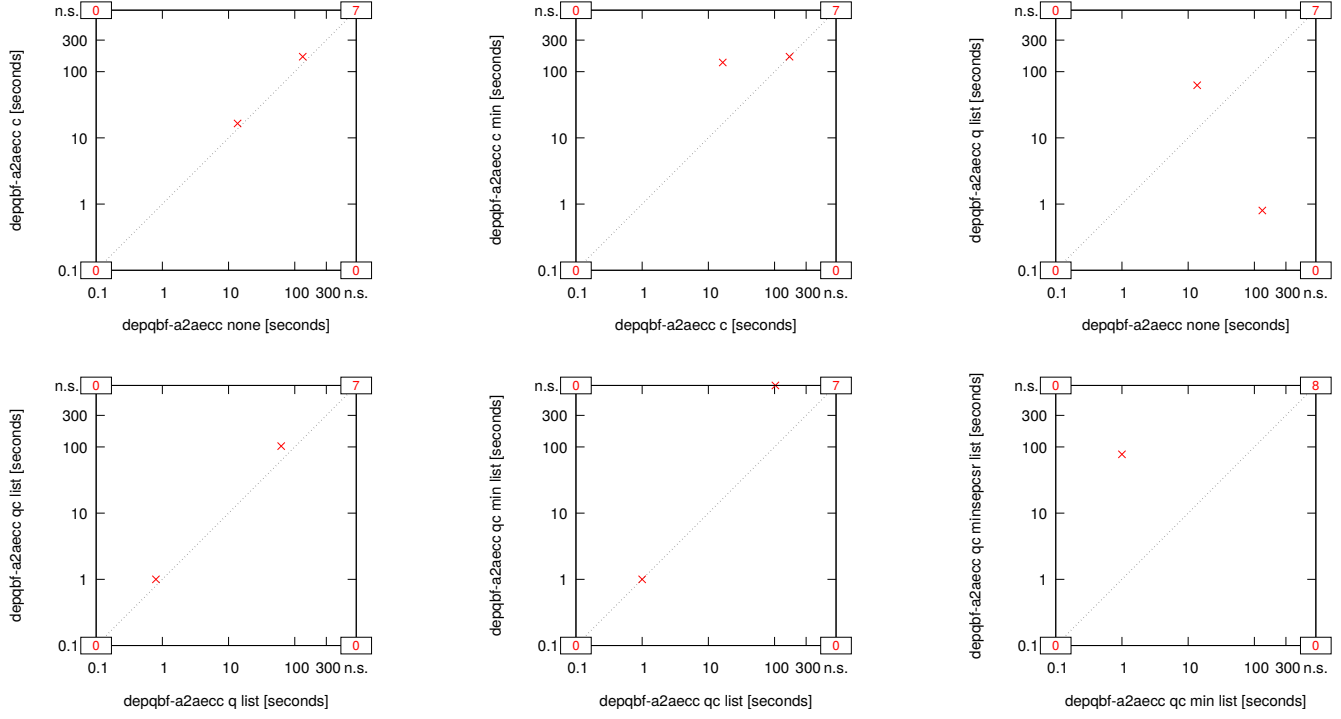


Fig. 975: Suite Palacios ($n = 9$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

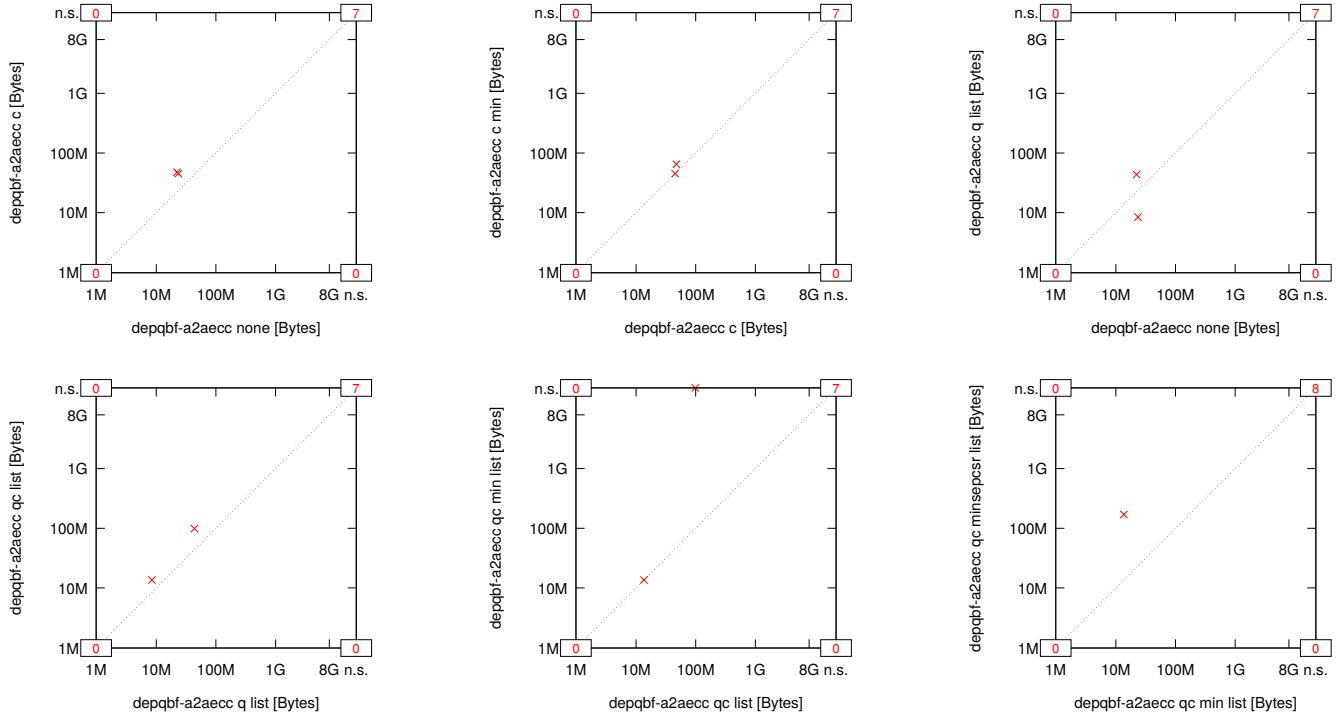


Fig. 976: Suite Palacios ($n = 9$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

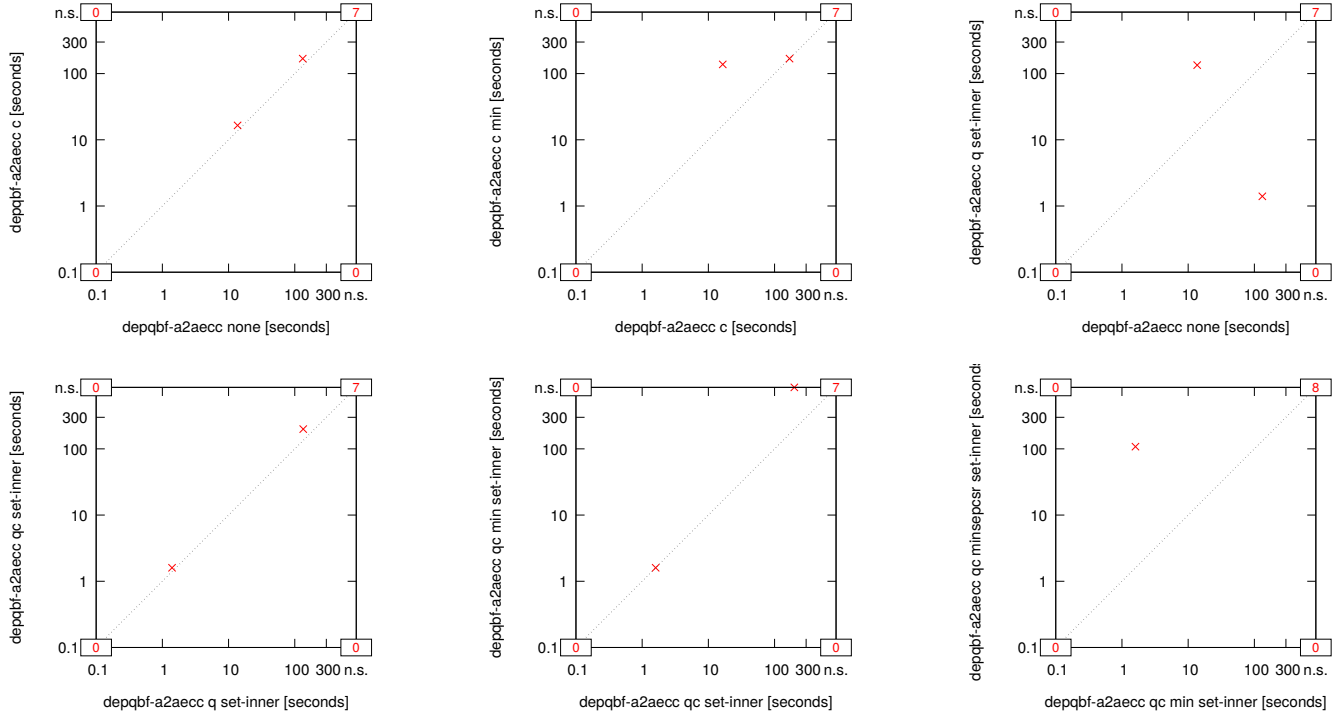


Fig. 977: Suite Palacios ($n = 9$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

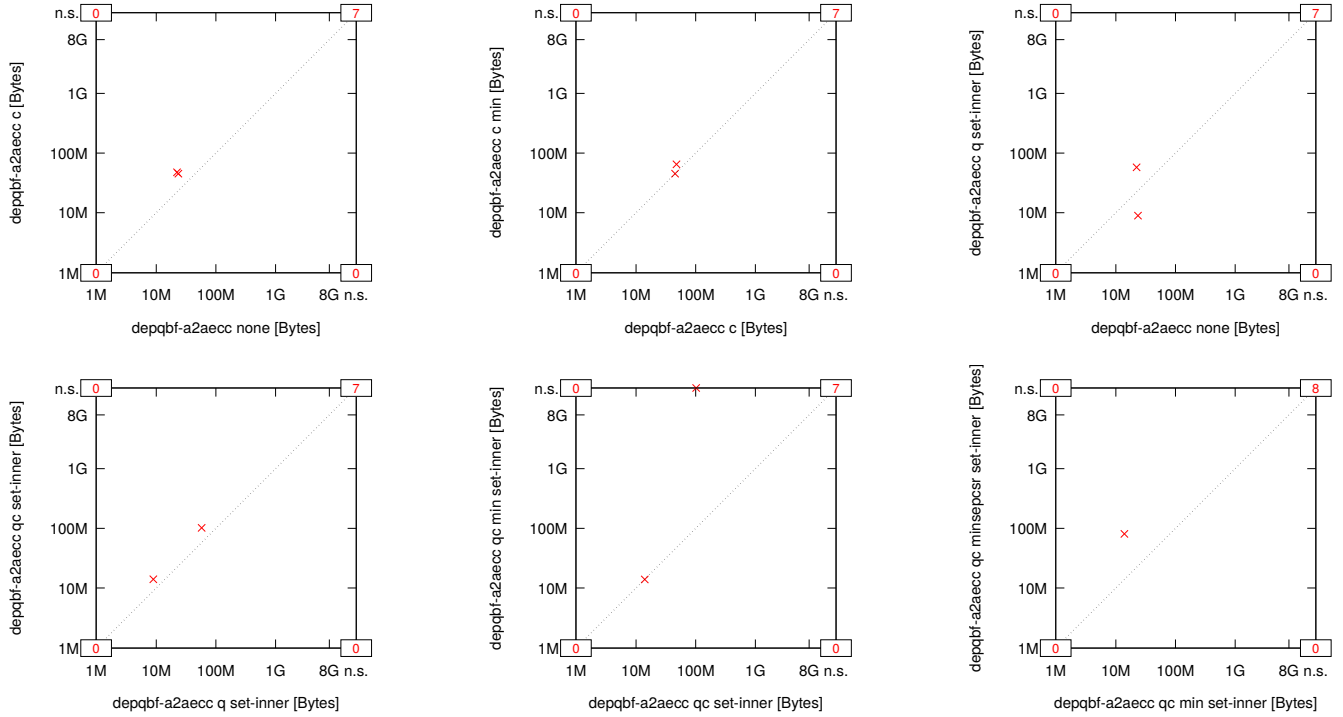


Fig. 978: Suite Palacios ($n = 9$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

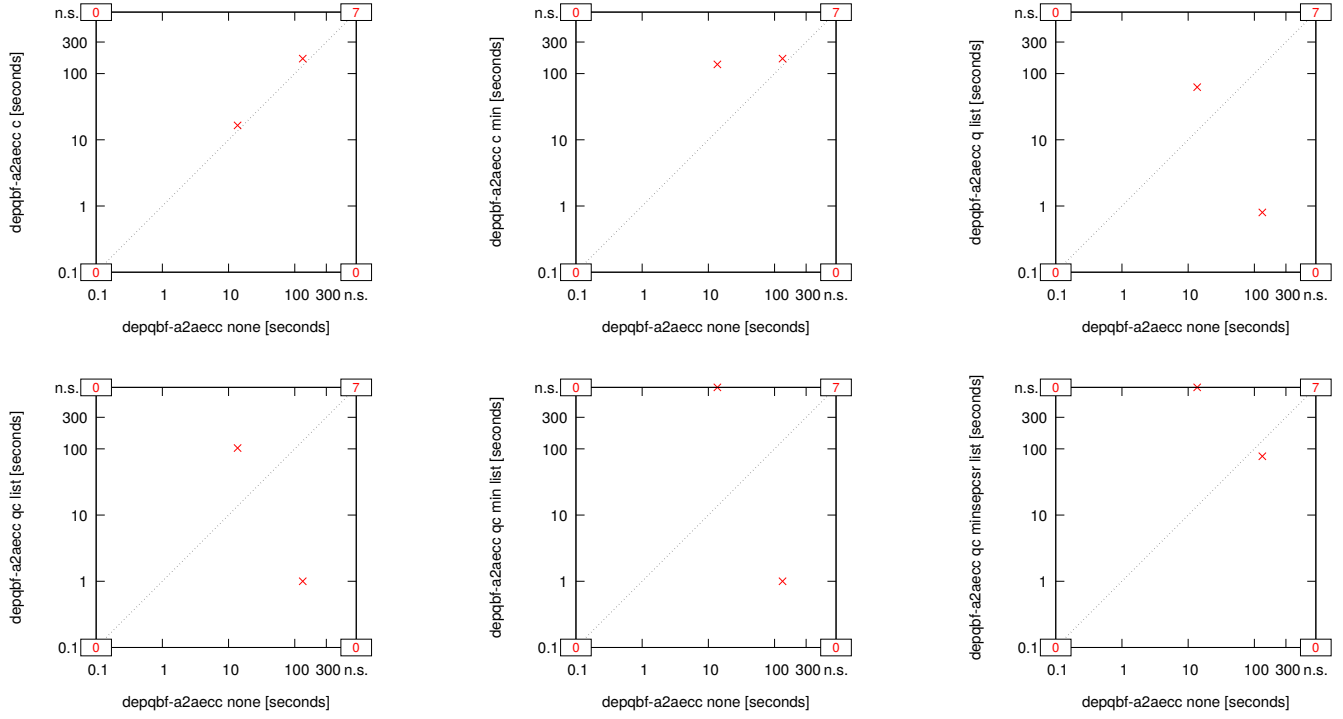


Fig. 979: Suite Palacios ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

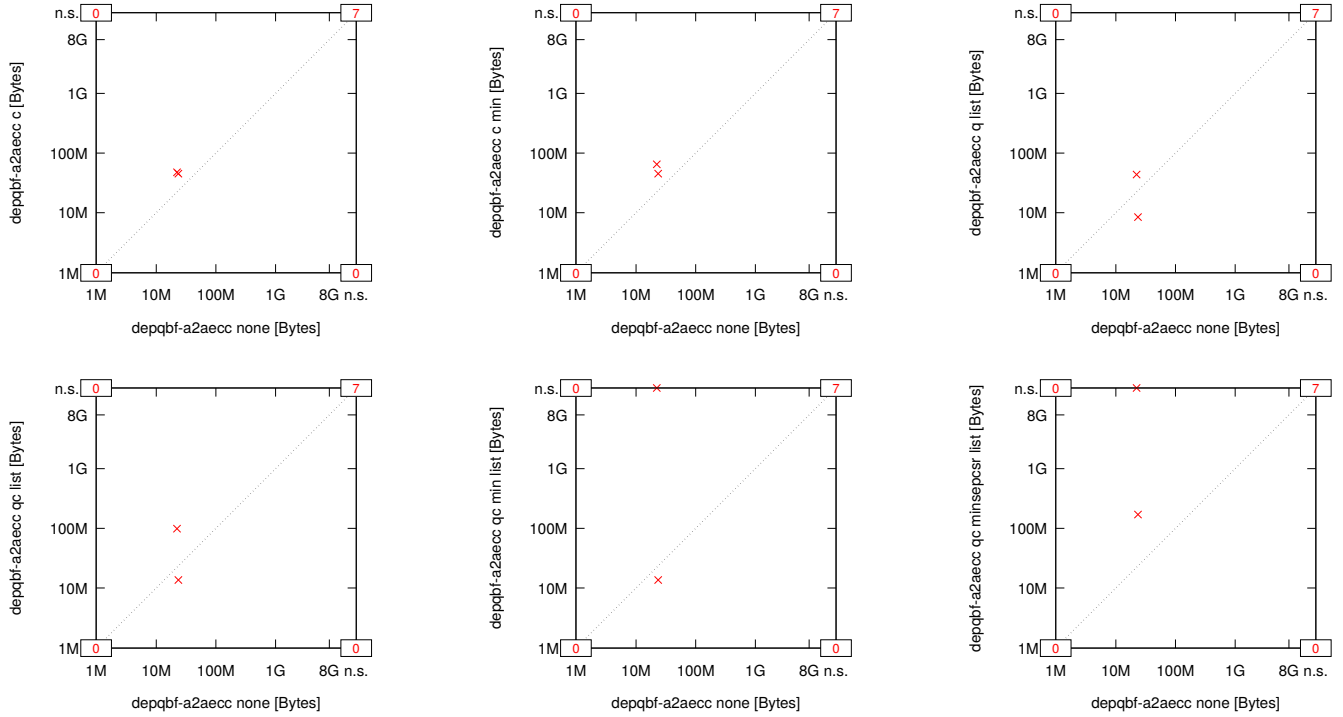


Fig. 980: Suite Palacios ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

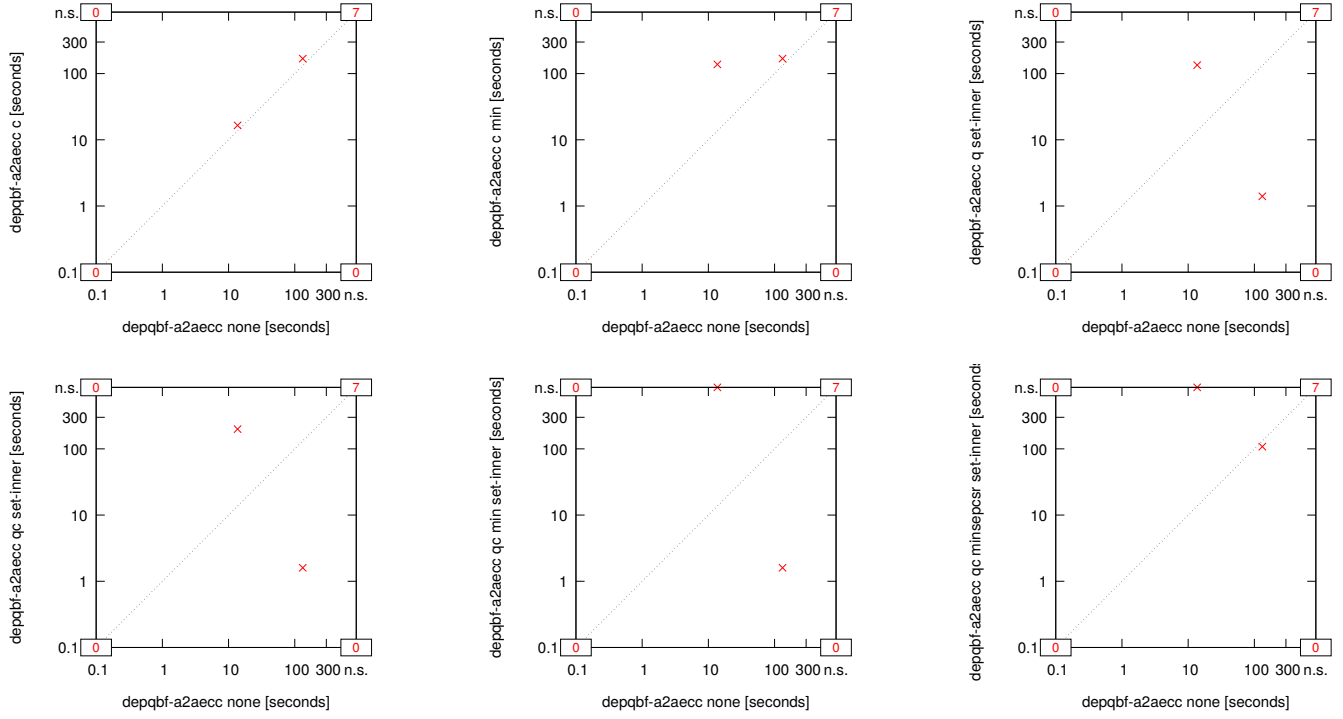


Fig. 981: Suite Palacios ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

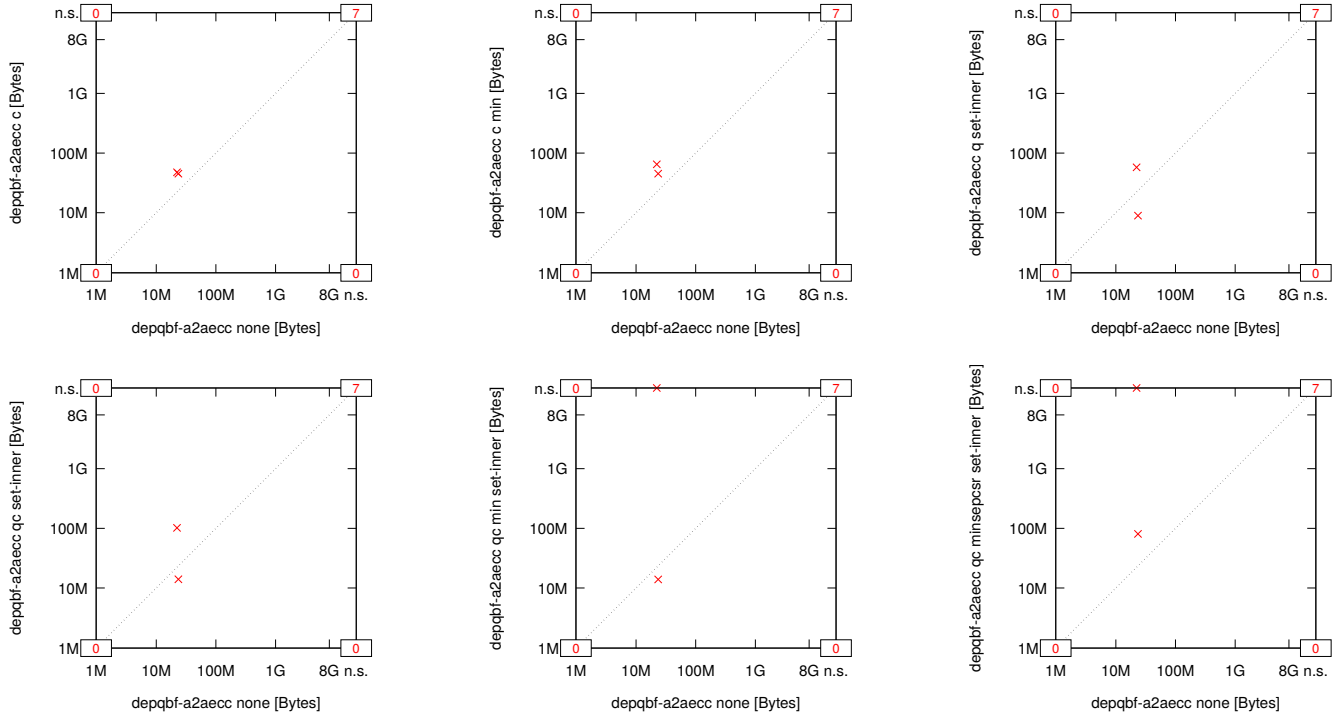


Fig. 982: Suite Palacios ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

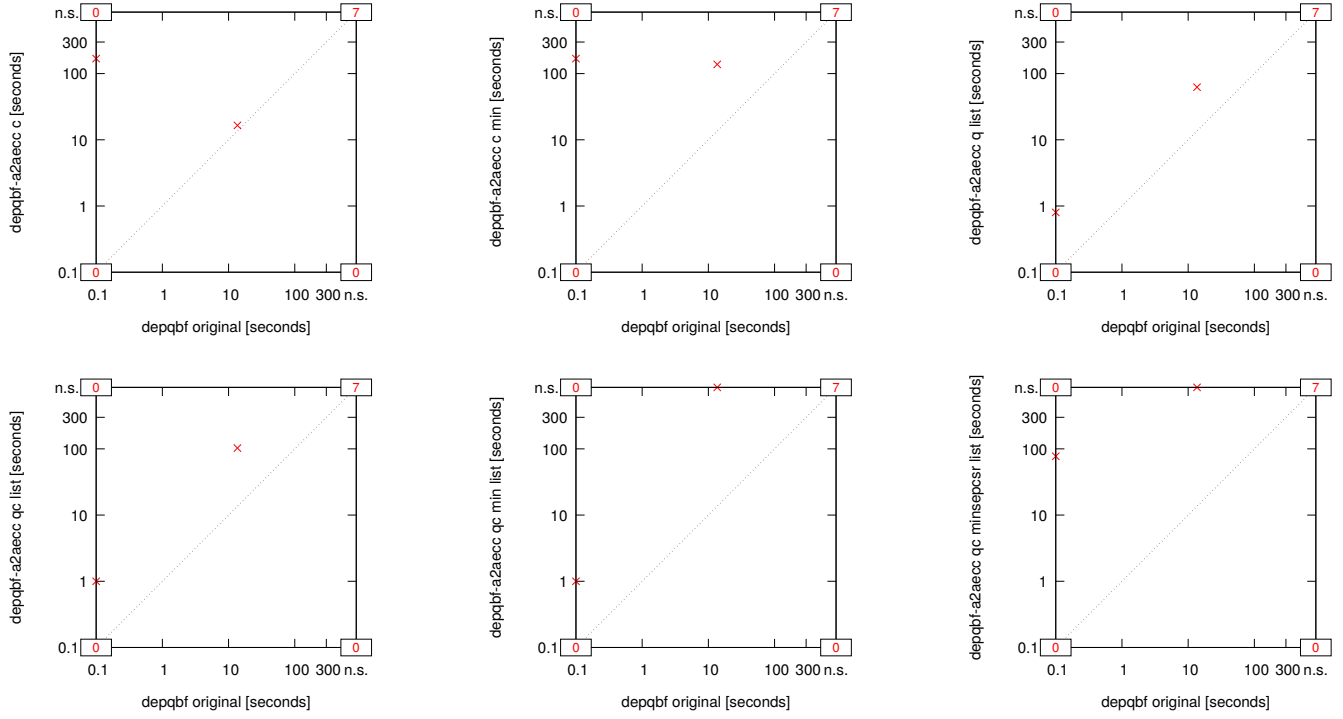


Fig. 983: Suite Palacios ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

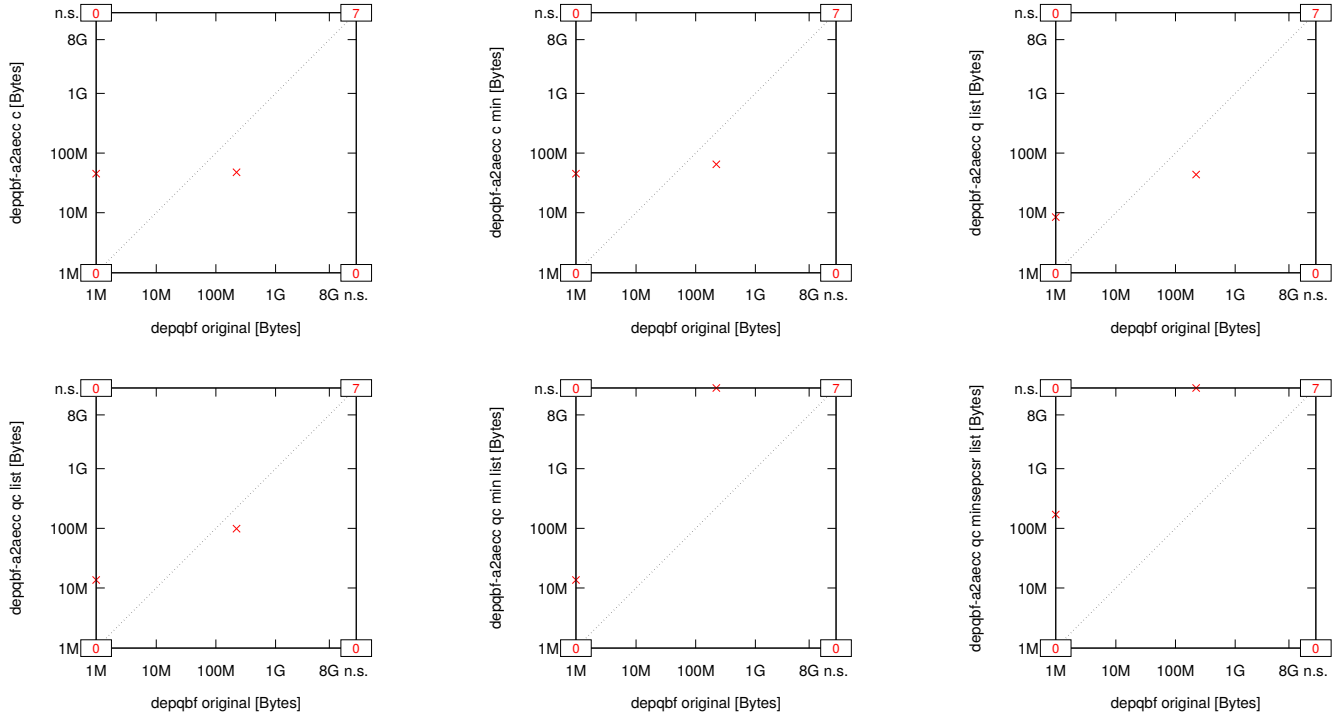


Fig. 984: Suite Palacios ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

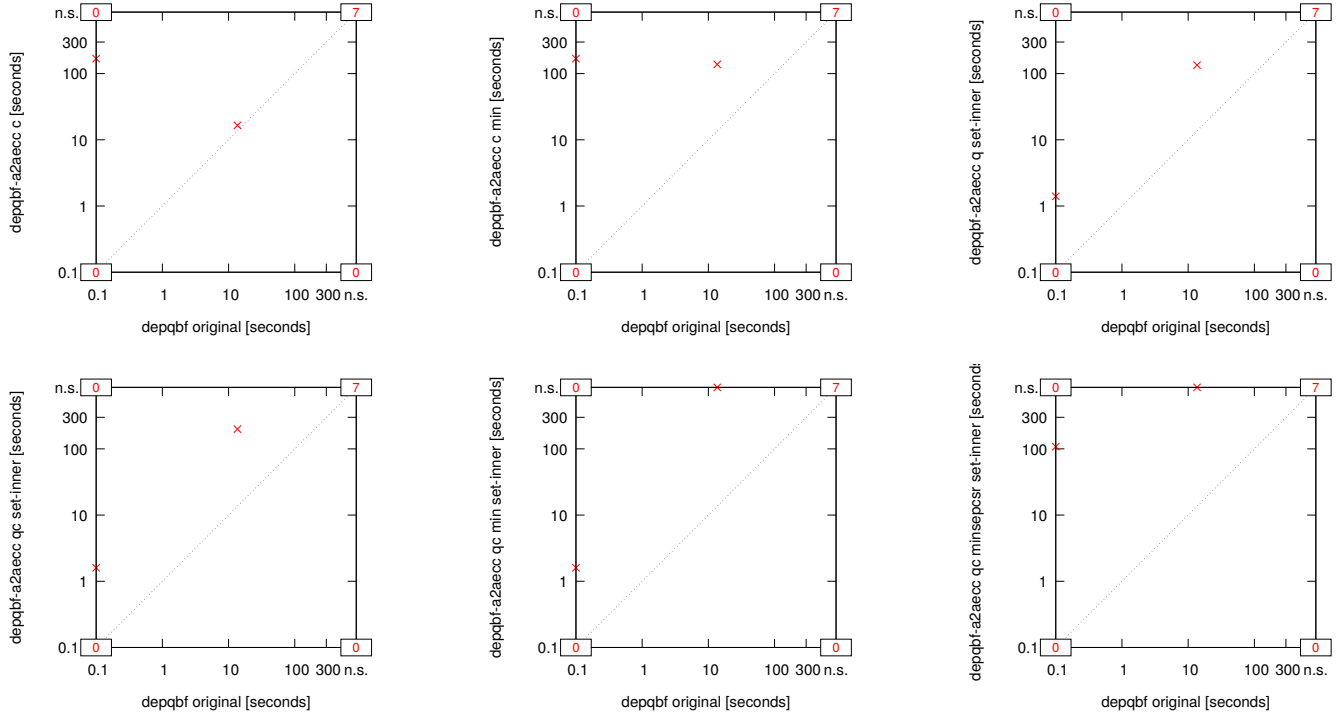


Fig. 985: Suite Palacios ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

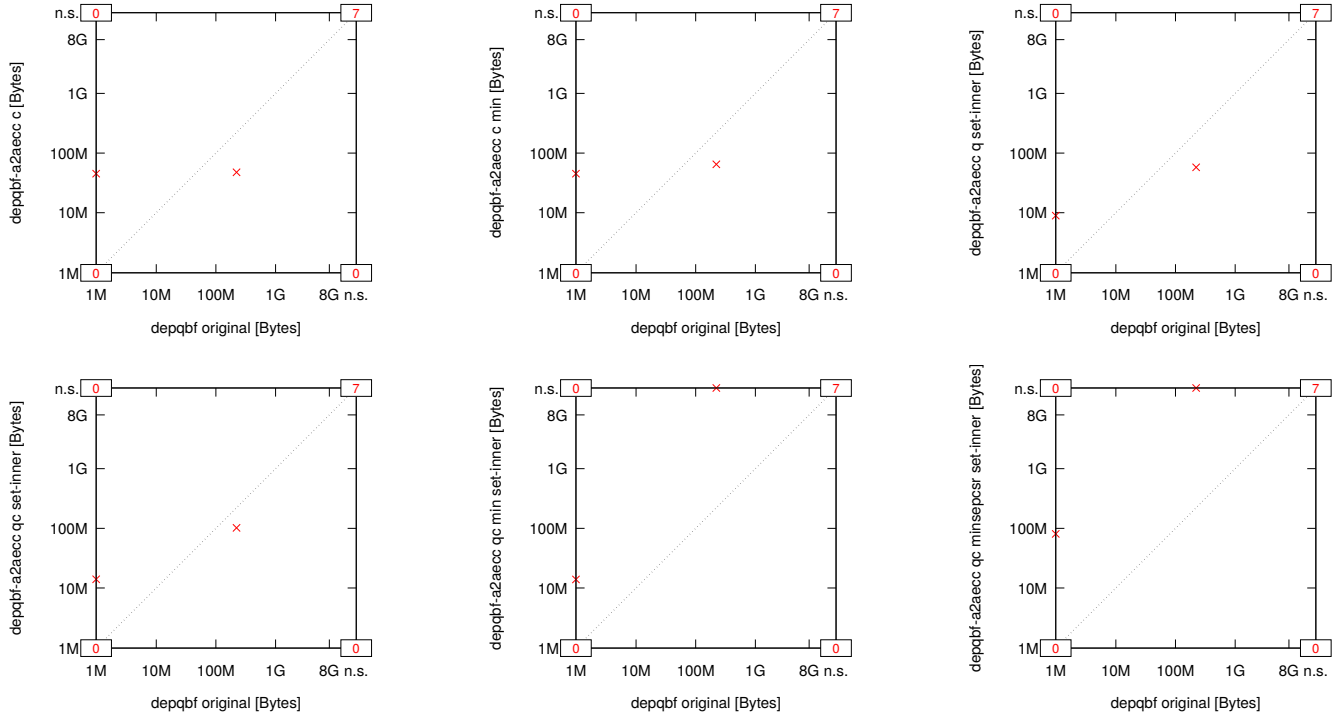


Fig. 986: Suite Palacios ($n = 9$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

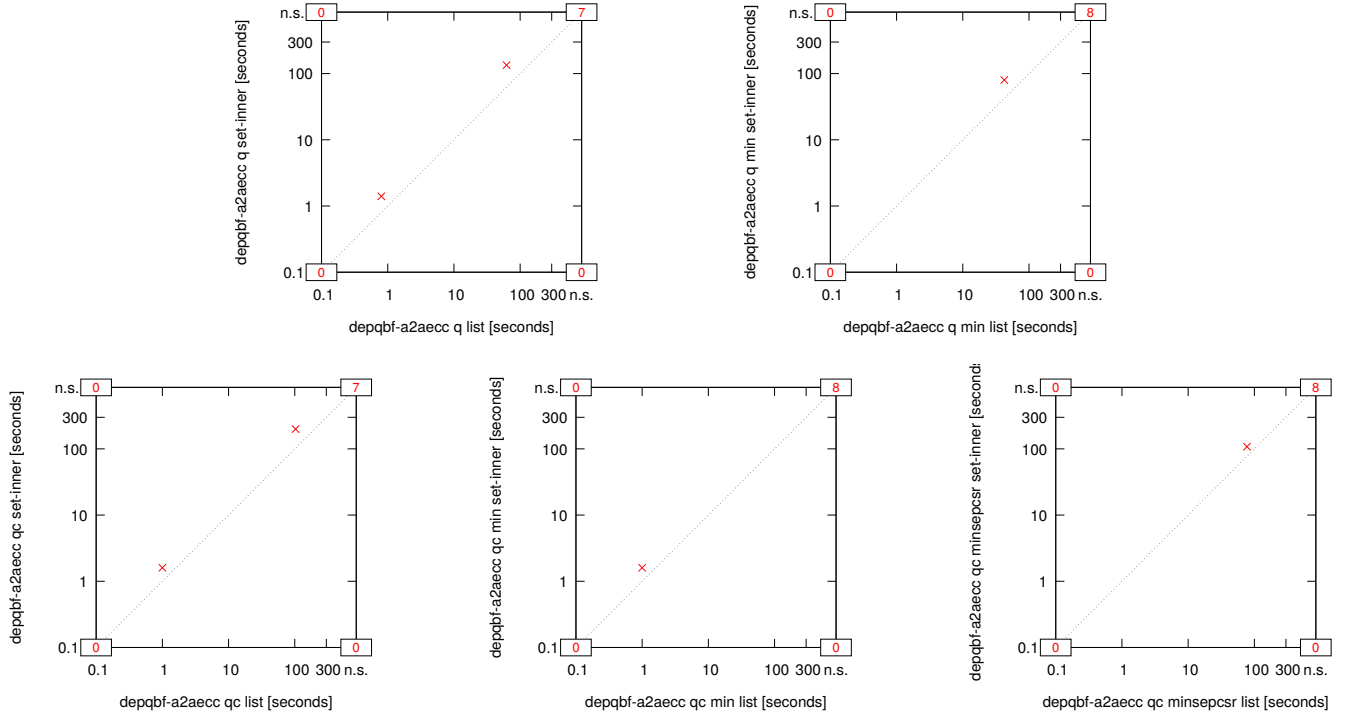


Fig. 987: Suite Palacios ($n = 9$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

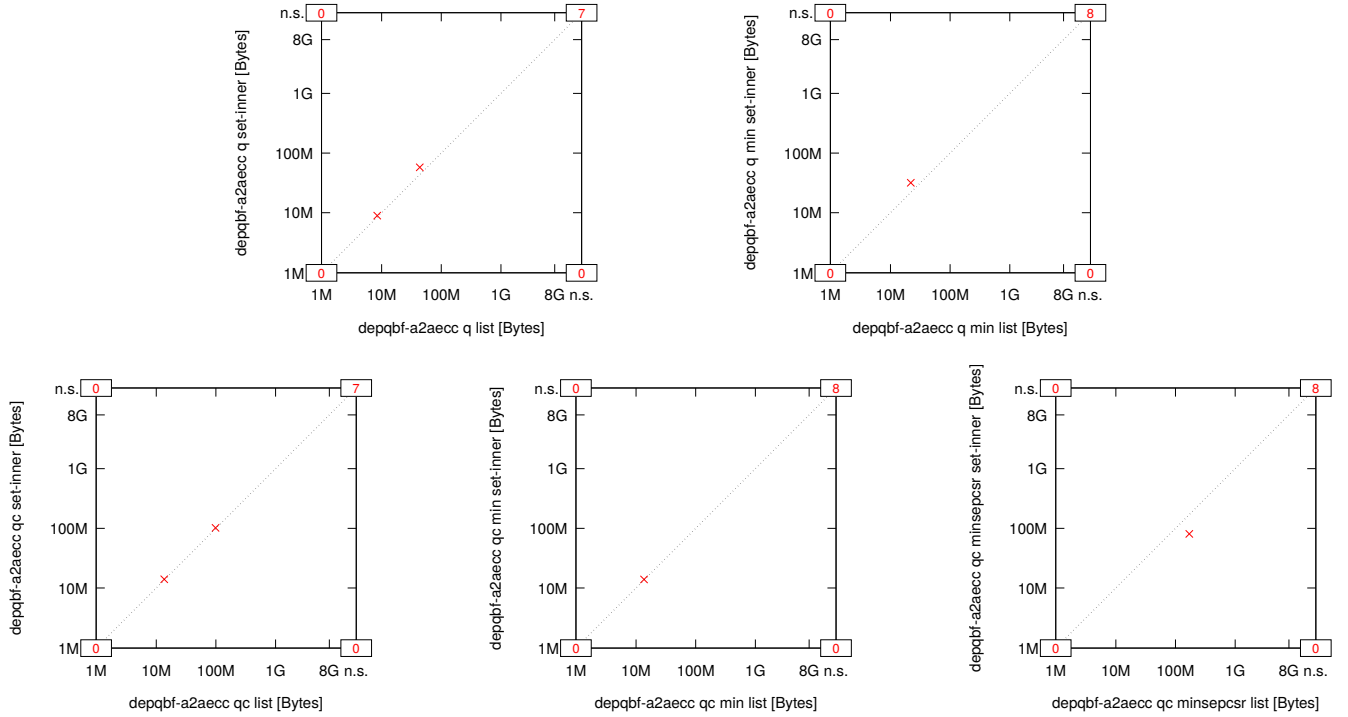


Fig. 988: Suite Palacios ($n = 9$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

35) *Pan* ($n = 89$):

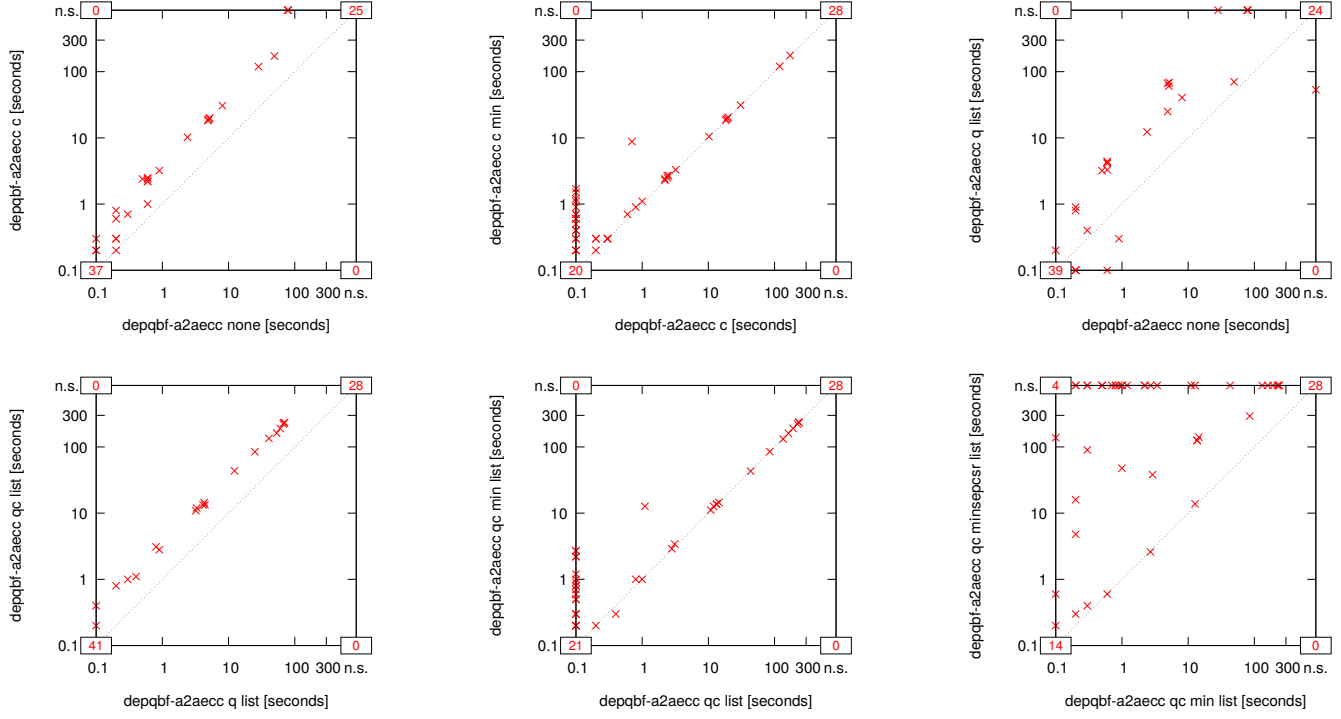


Fig. 989: Suite Pan ($n = 89$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

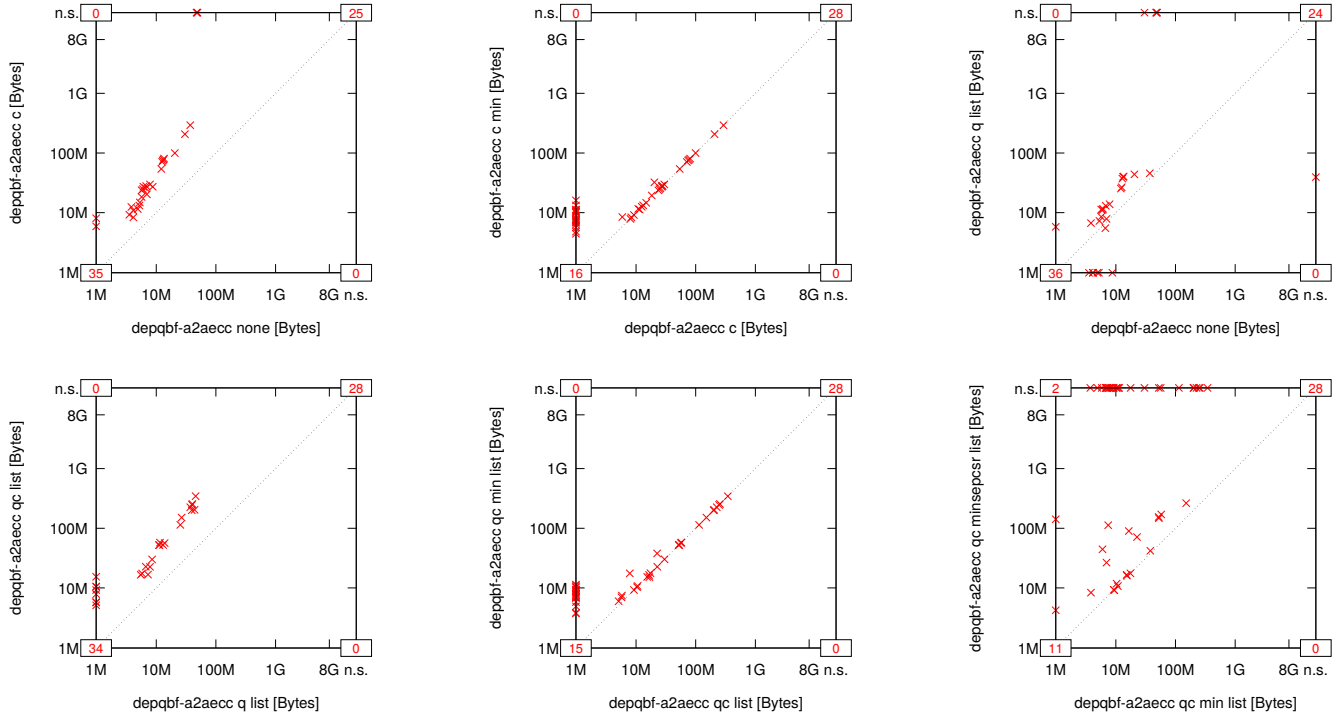


Fig. 990: Suite Pan ($n = 89$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

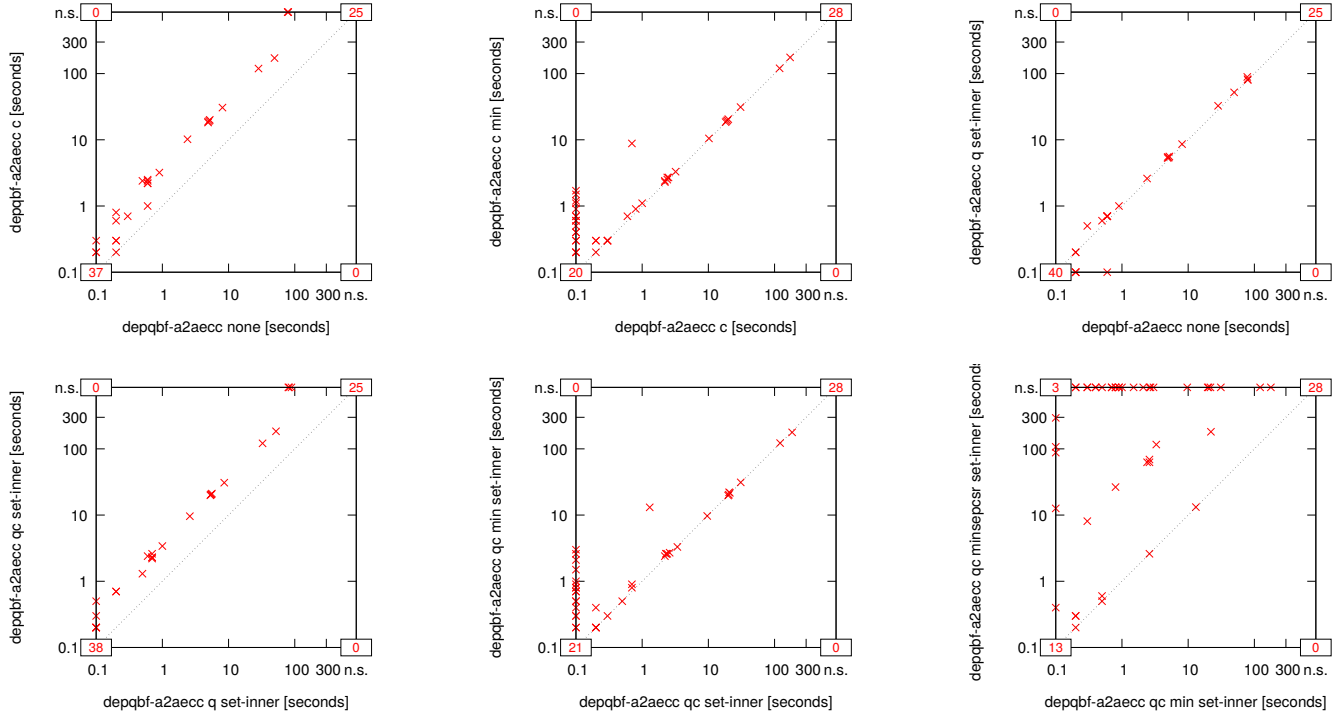


Fig. 991: Suite Pan ($n = 89$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

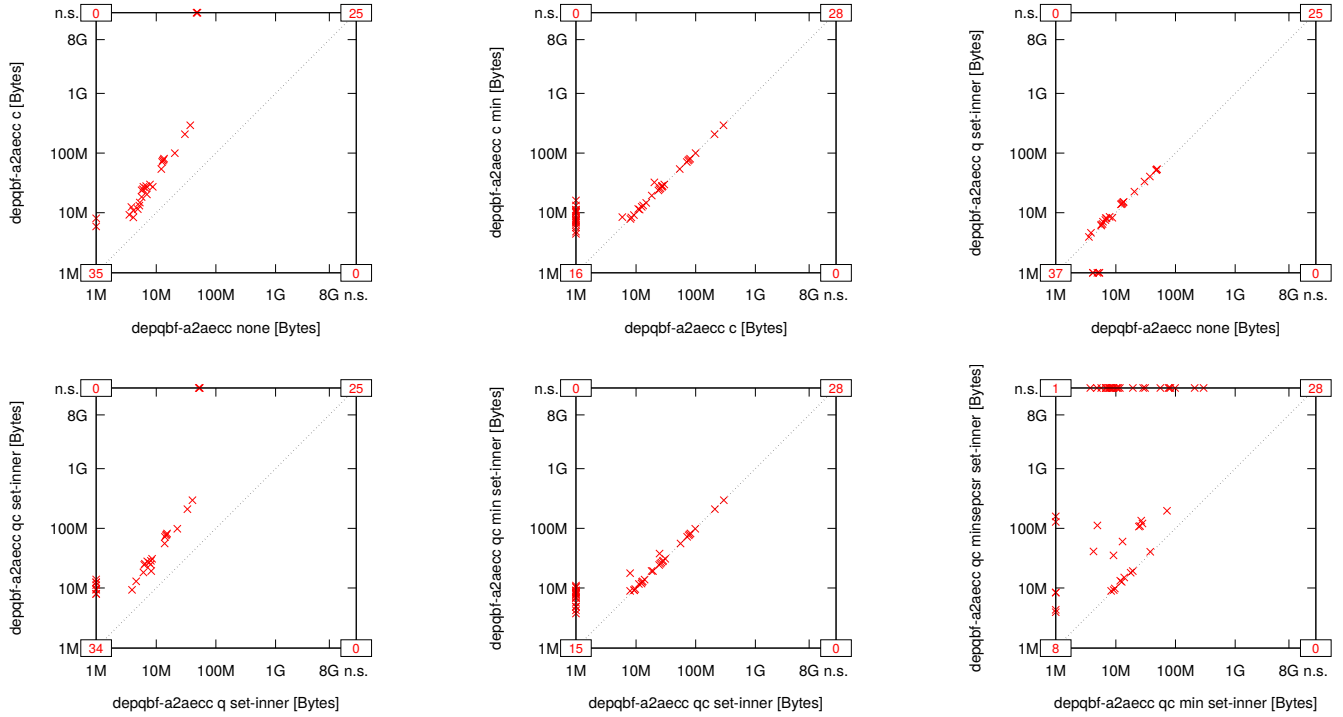


Fig. 992: Suite Pan ($n = 89$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

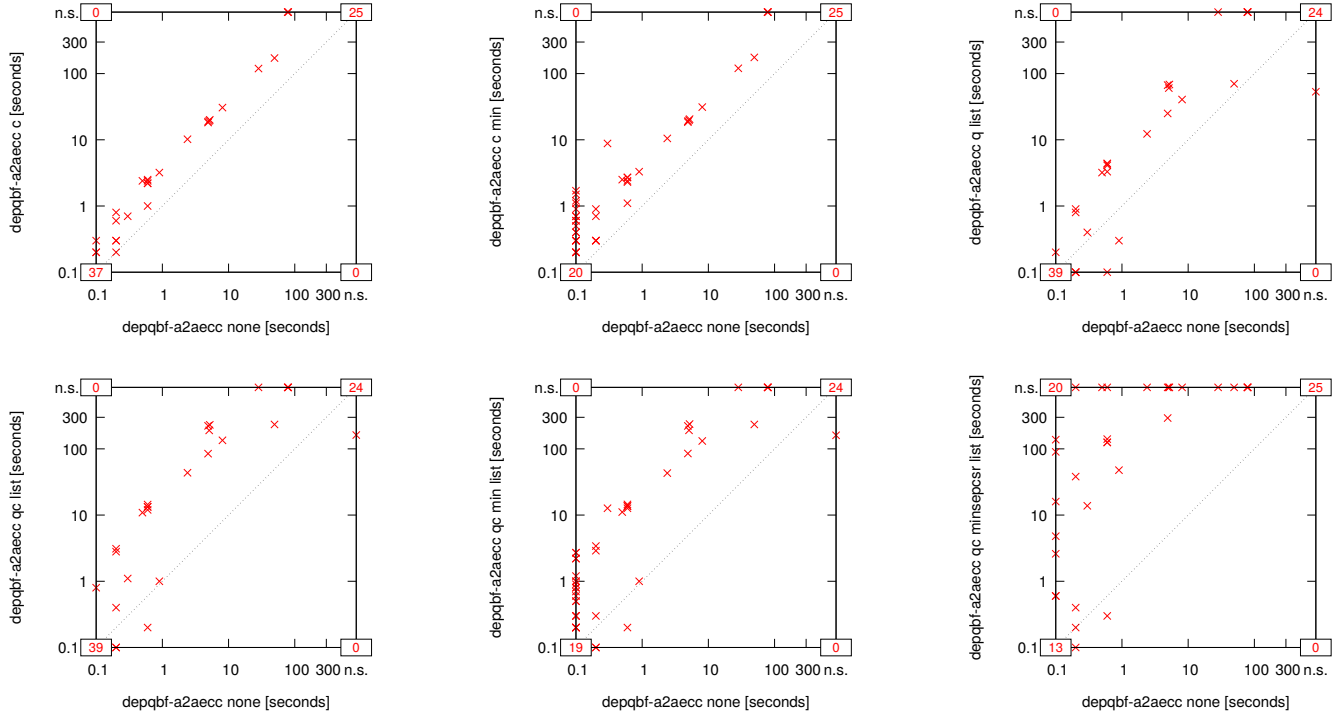


Fig. 993: Suite Pan ($n = 89$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

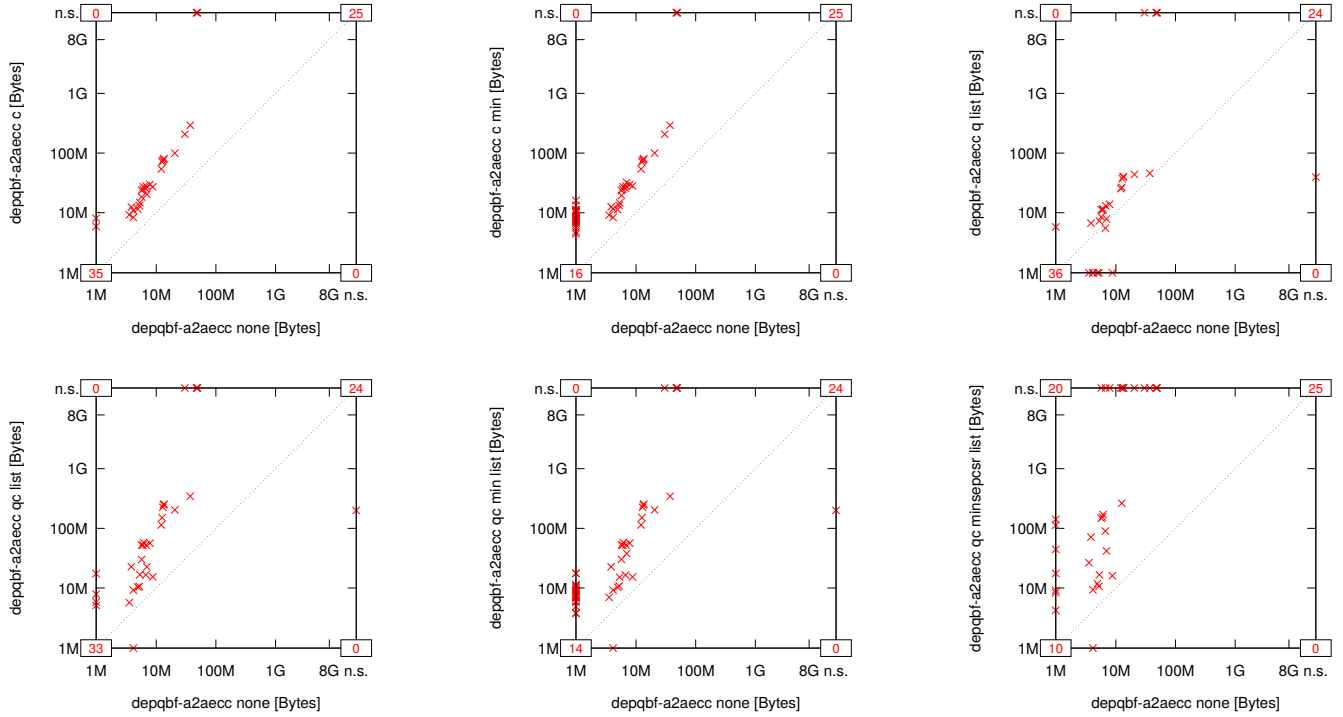


Fig. 994: Suite Pan ($n = 89$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

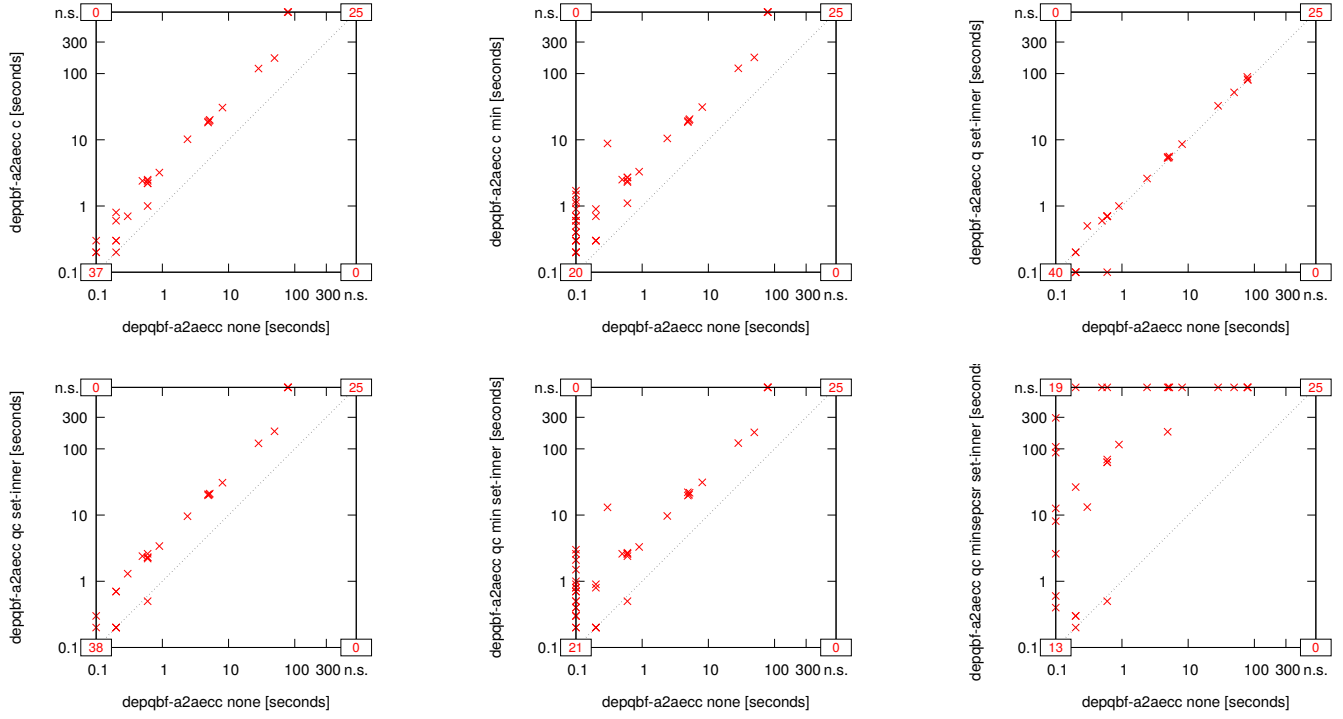


Fig. 995: Suite Pan ($n = 89$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

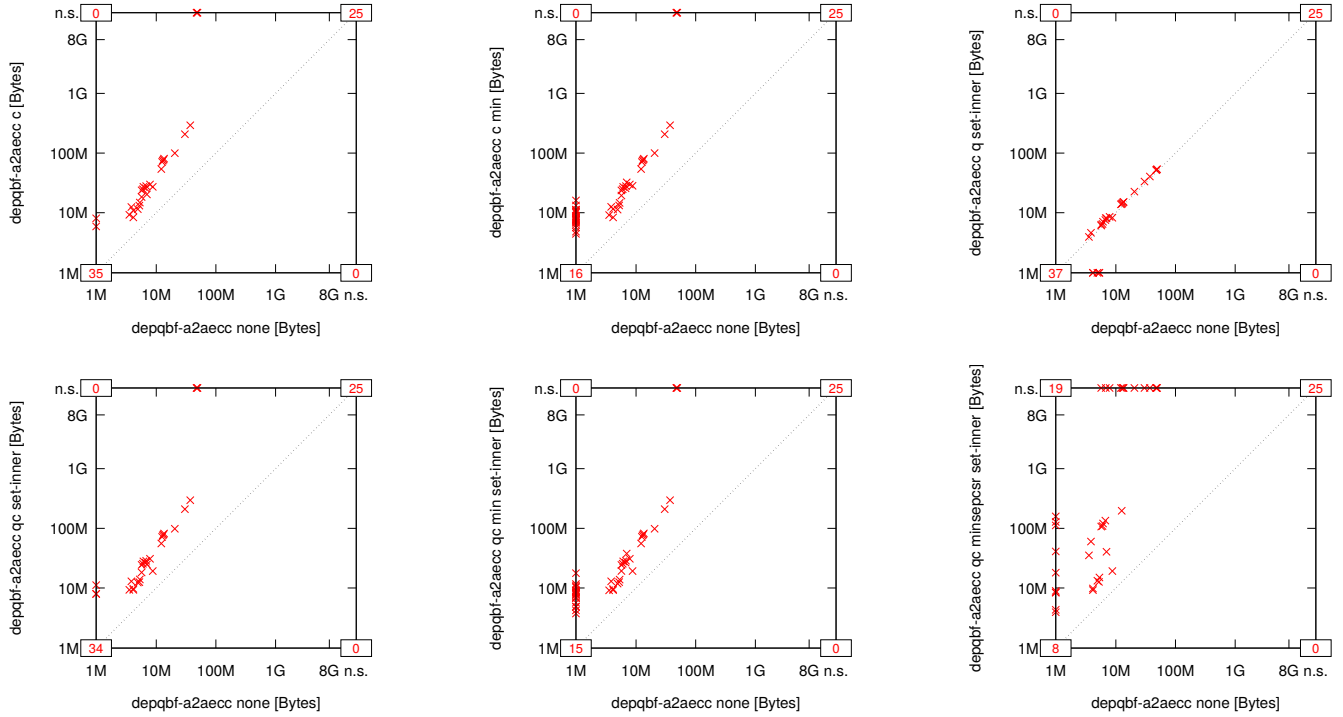


Fig. 996: Suite Pan ($n = 89$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

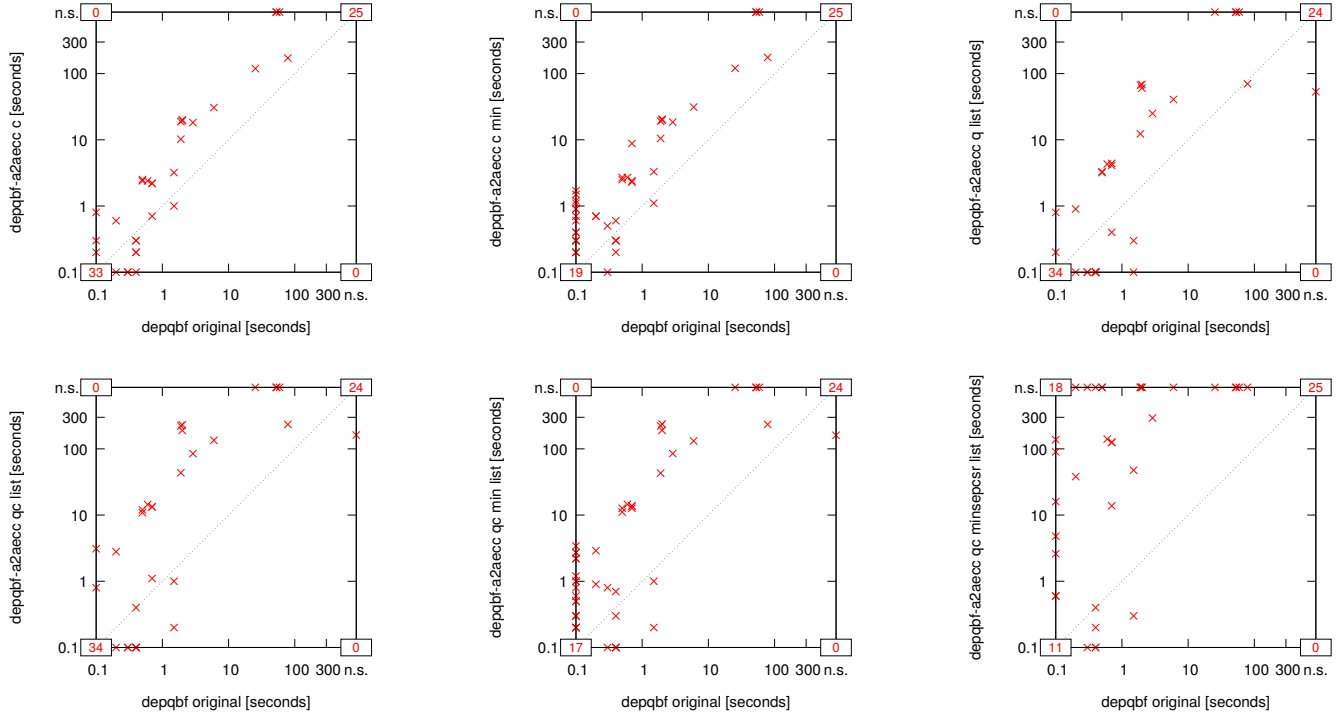


Fig. 997: Suite Pan ($n = 89$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

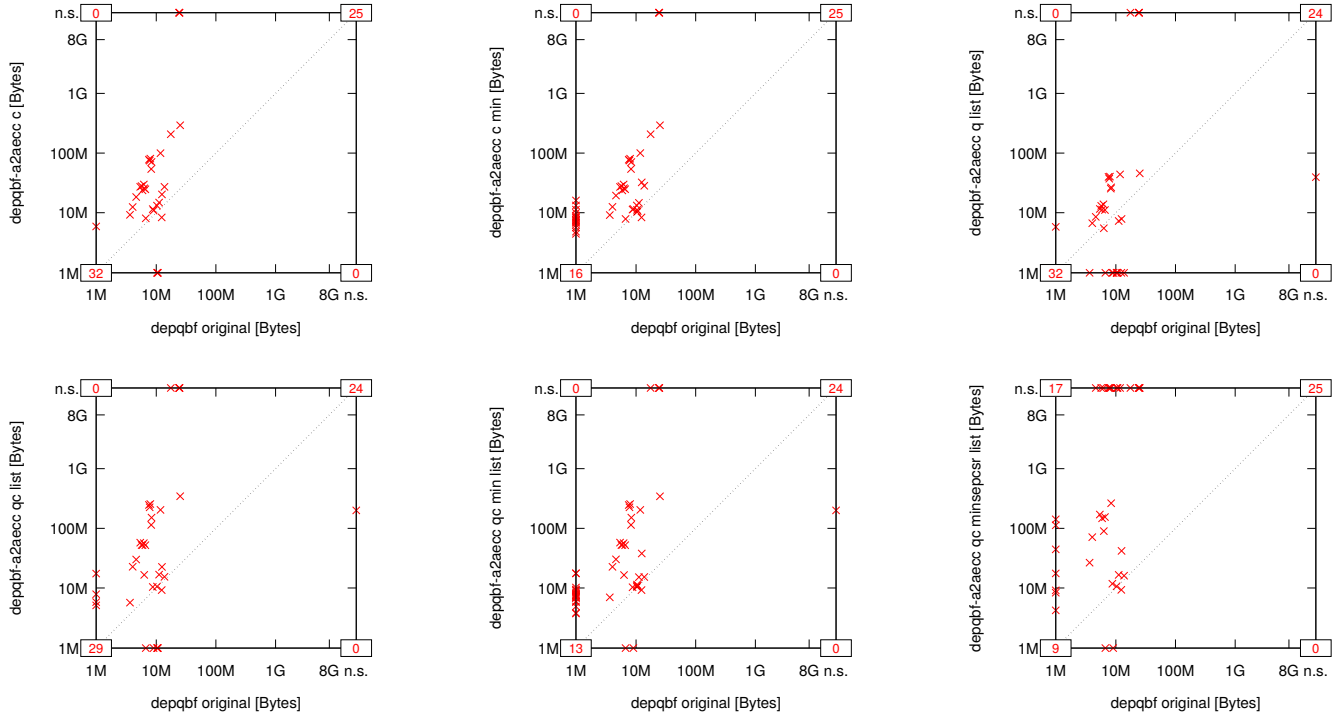


Fig. 998: Suite Pan ($n = 89$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

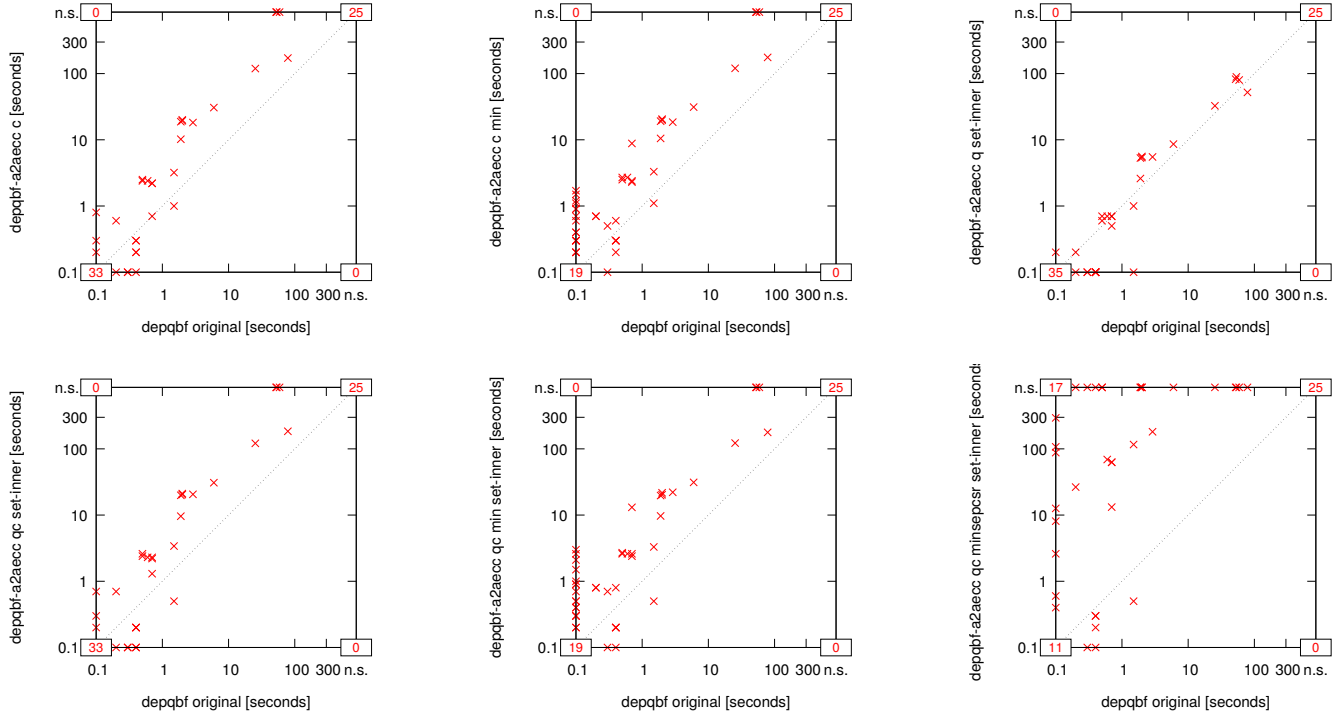


Fig. 999: Suite Pan ($n = 89$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

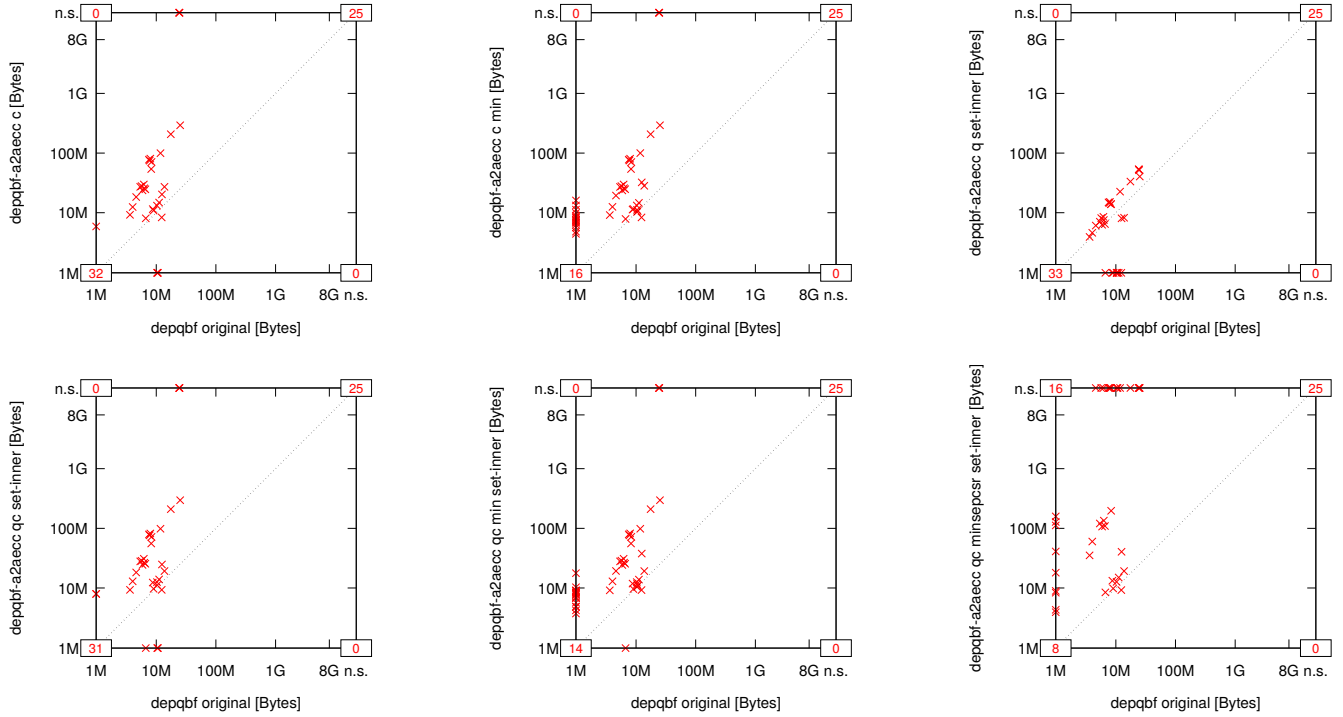


Fig. 1000: Suite Pan ($n = 89$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

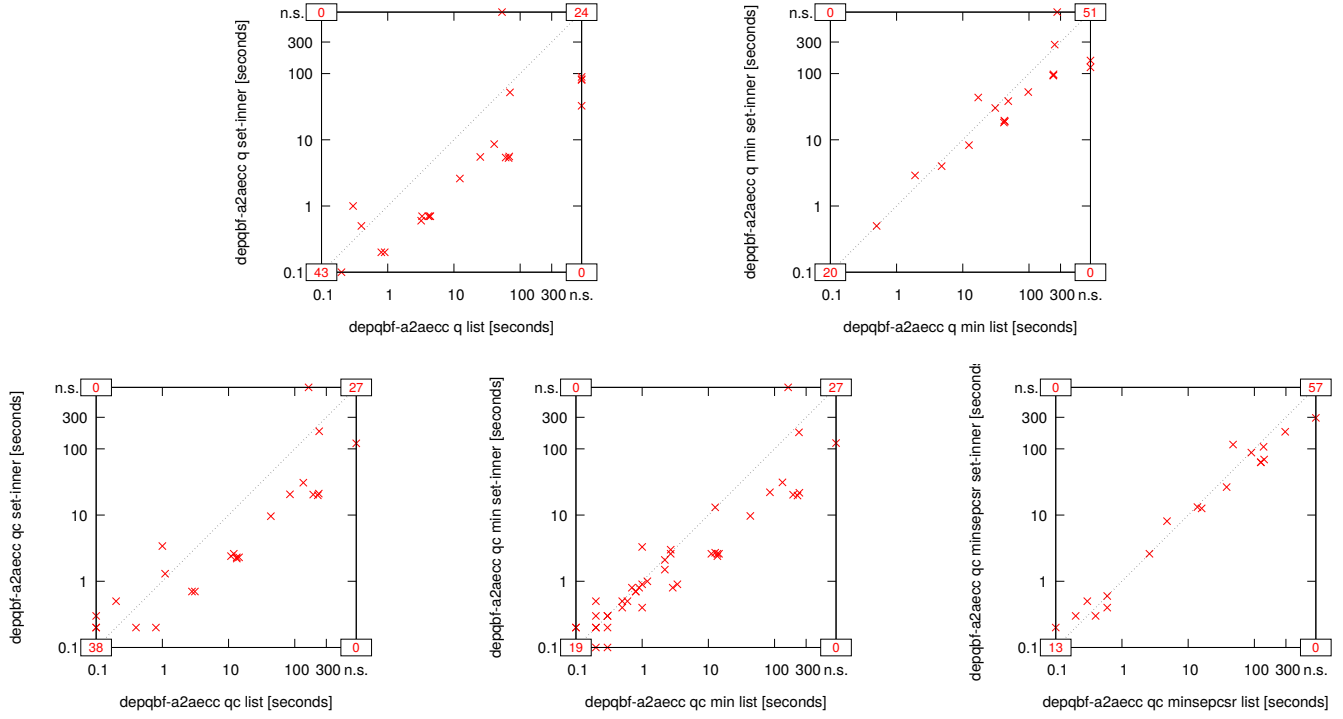


Fig. 1001: Suite Pan ($n = 89$): Extracting and minimizing unsatisfiable cores in `DepQBF-a2aecc` with set-inner versus list semantics (run time in seconds).

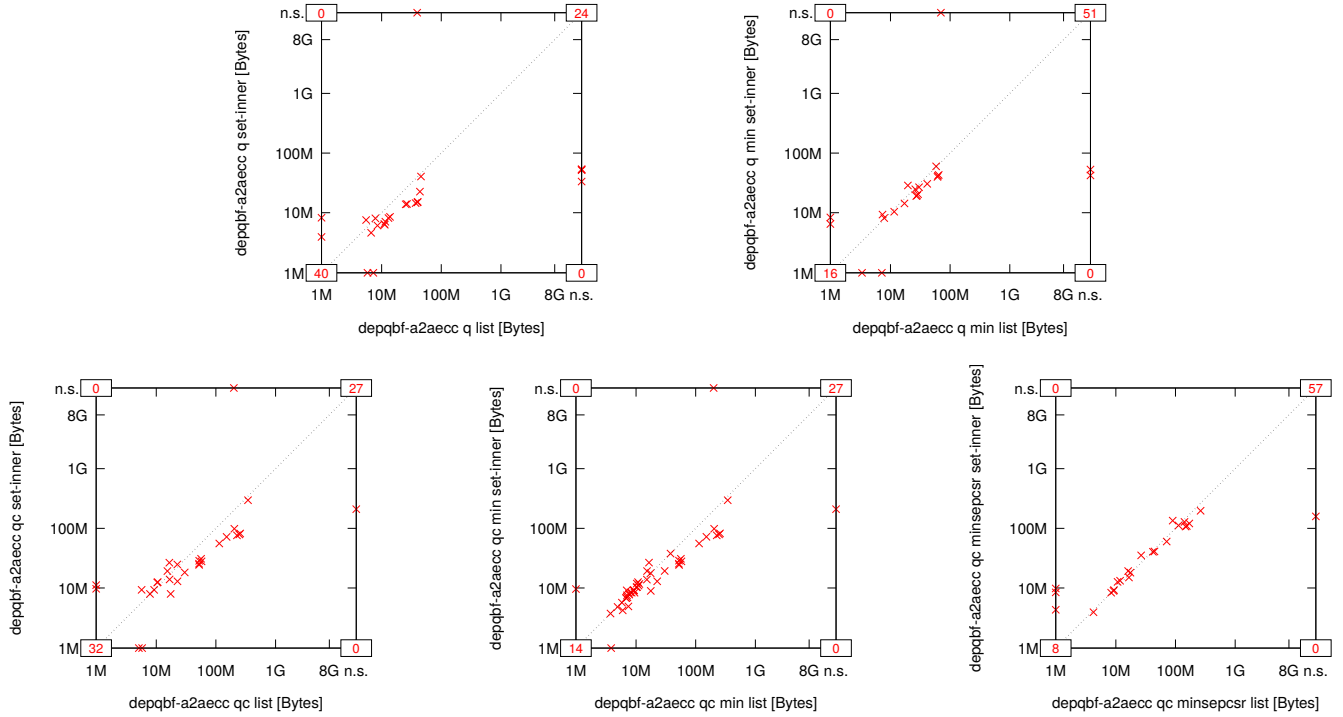


Fig. 1002: Suite Pan ($n = 89$): Extracting and minimizing unsatisfiable cores in `DepQBF-a2aecc` with set-inner versus list semantics (memory usage in Bytes).

36) *Peitl* ($n = 10$):

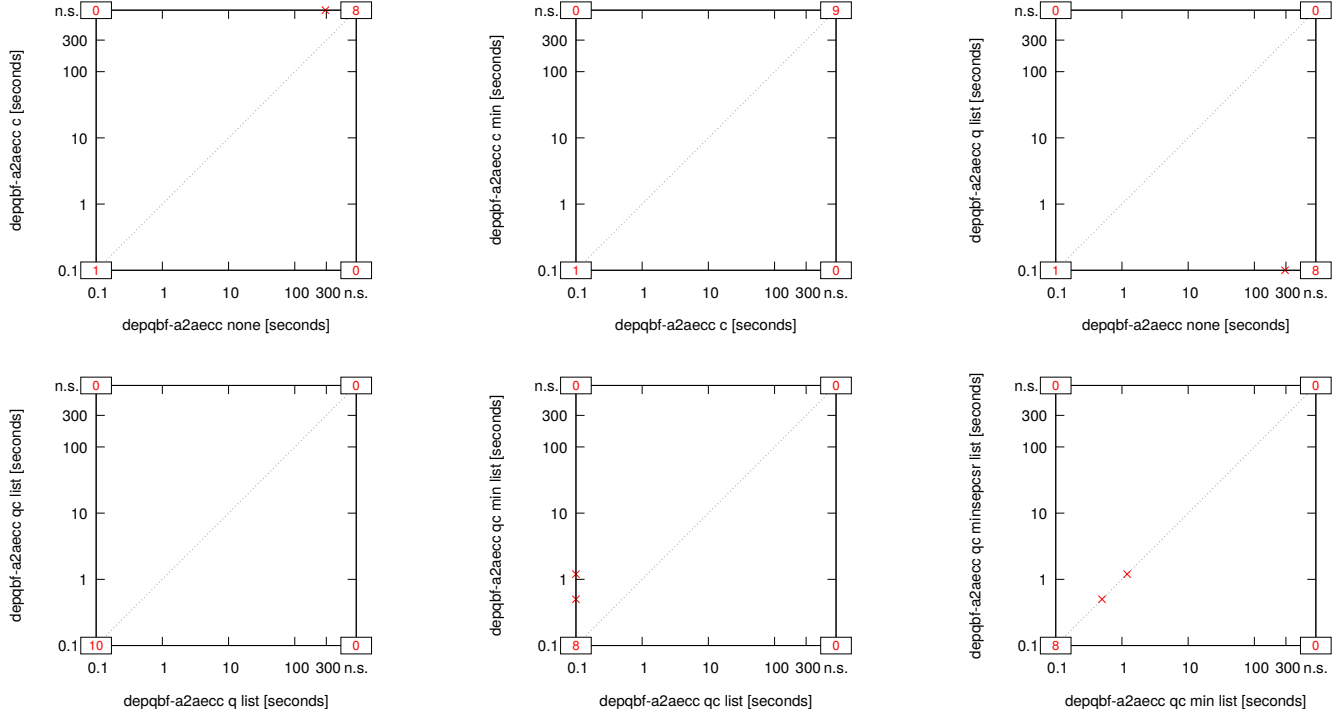


Fig. 1003: Suite Peitl ($n = 10$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

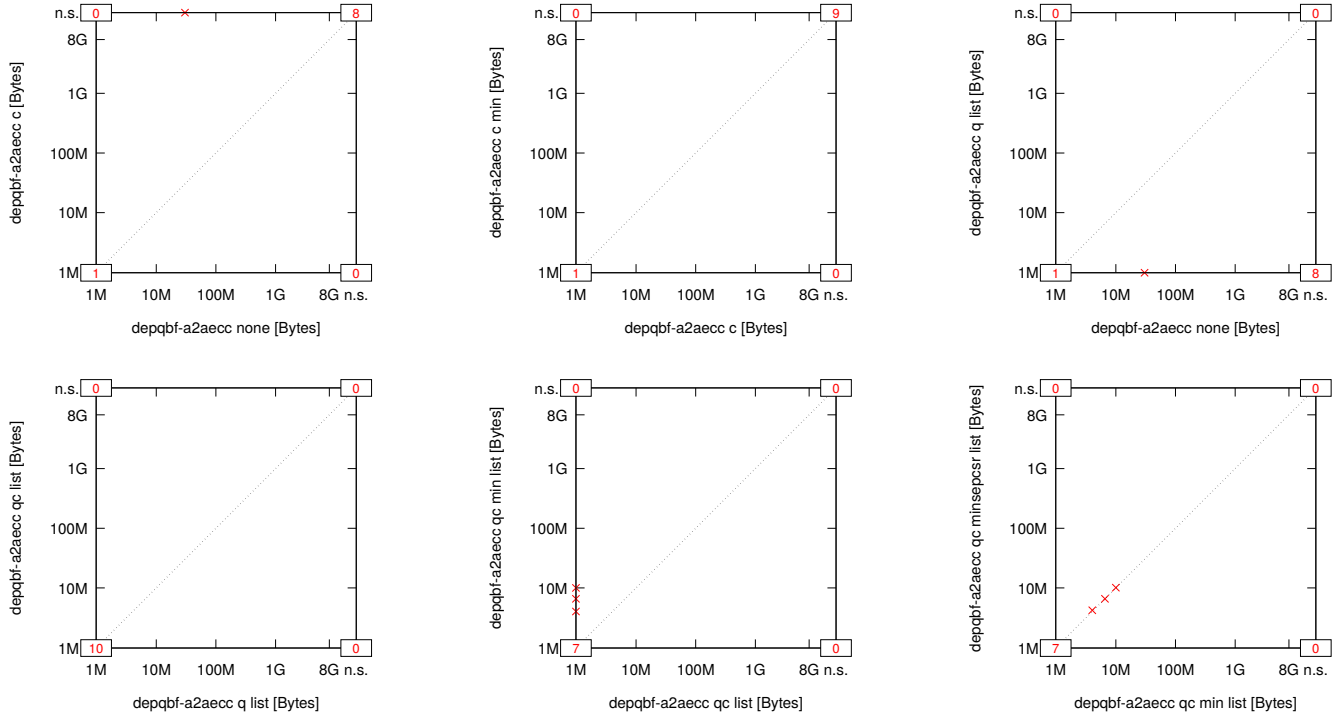


Fig. 1004: Suite Peitl ($n = 10$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

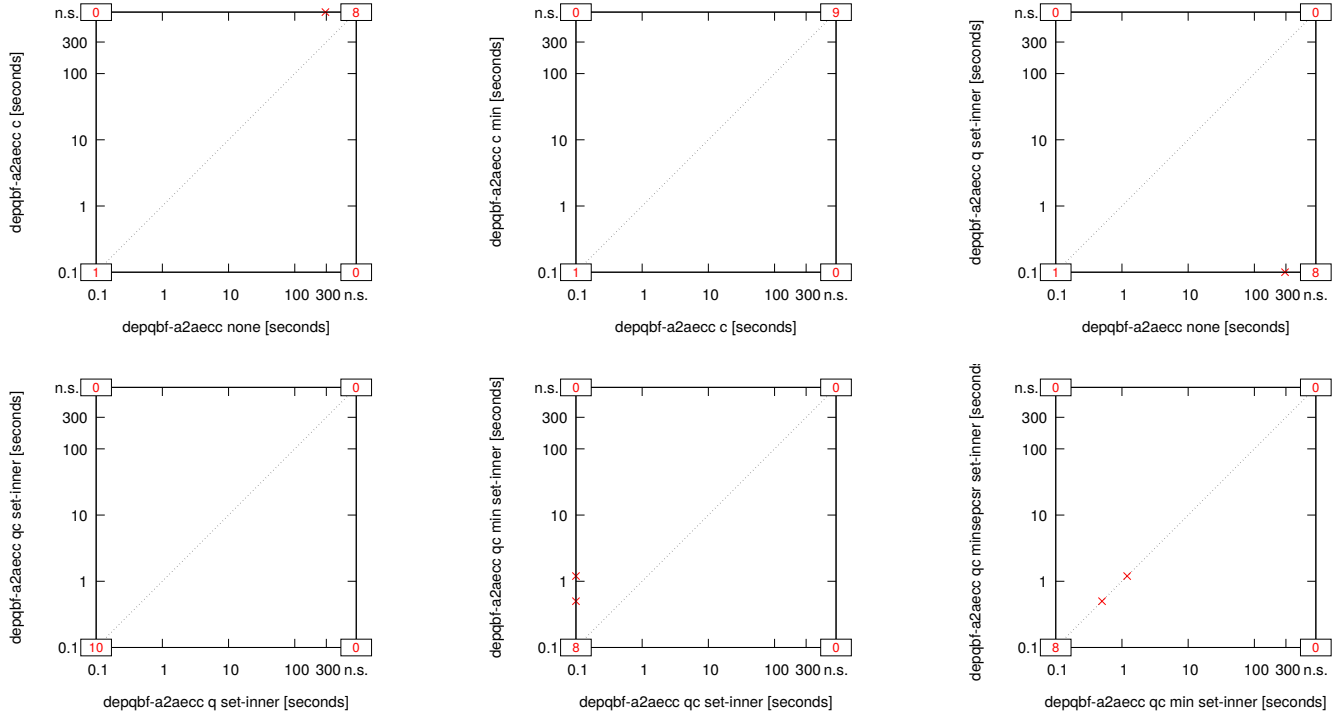


Fig. 1005: Suite Peitl ($n = 10$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

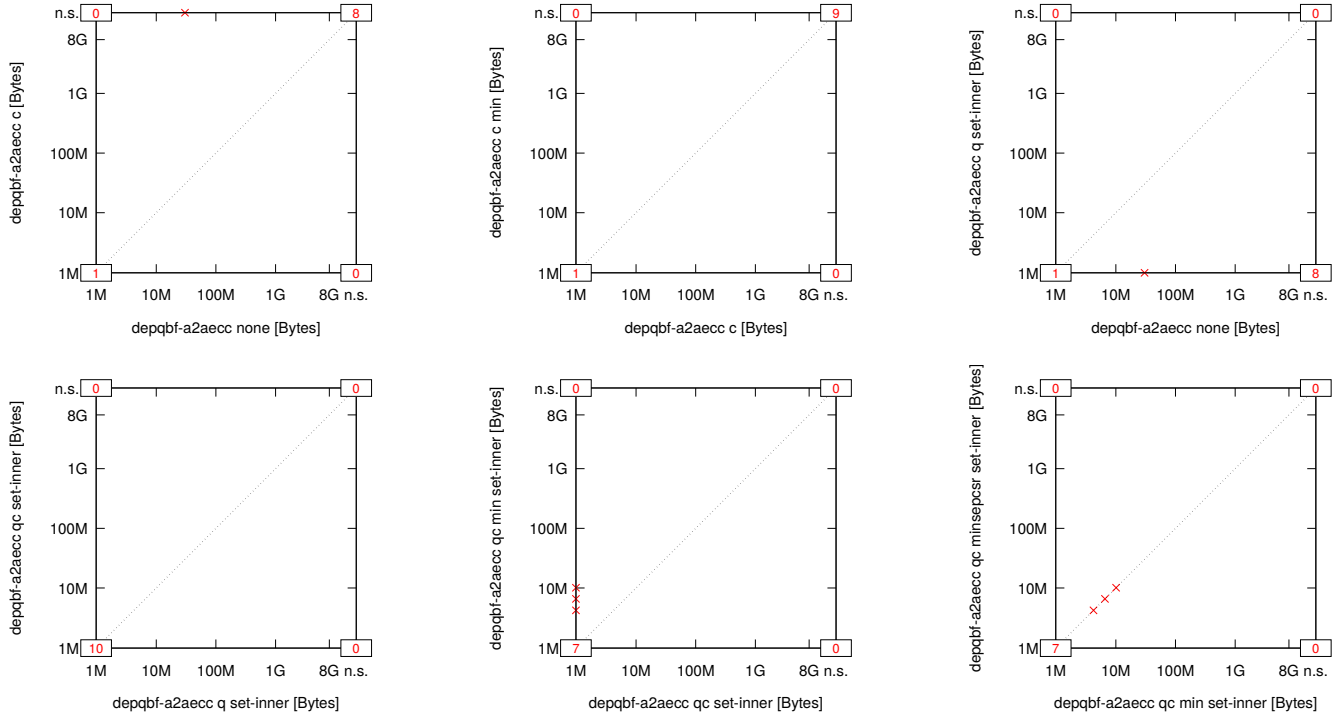


Fig. 1006: Suite Peitl ($n = 10$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

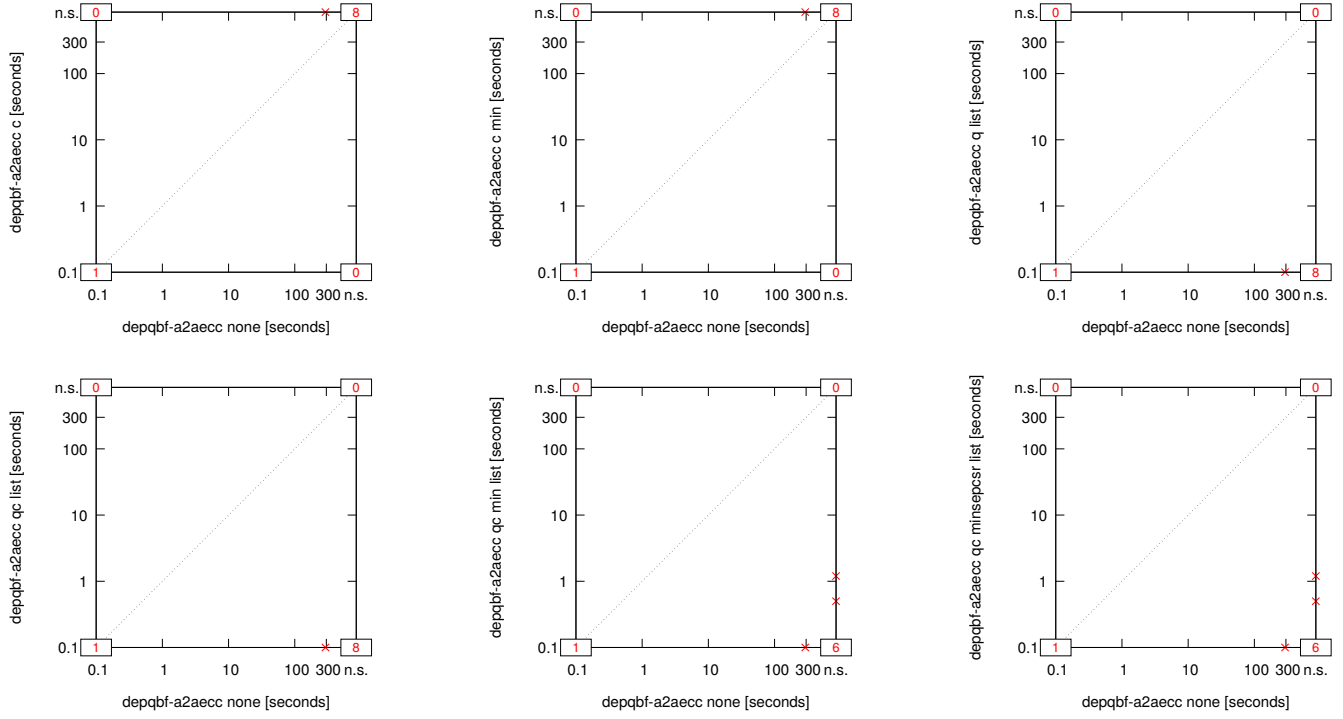


Fig. 1007: Suite Peitl ($n = 10$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

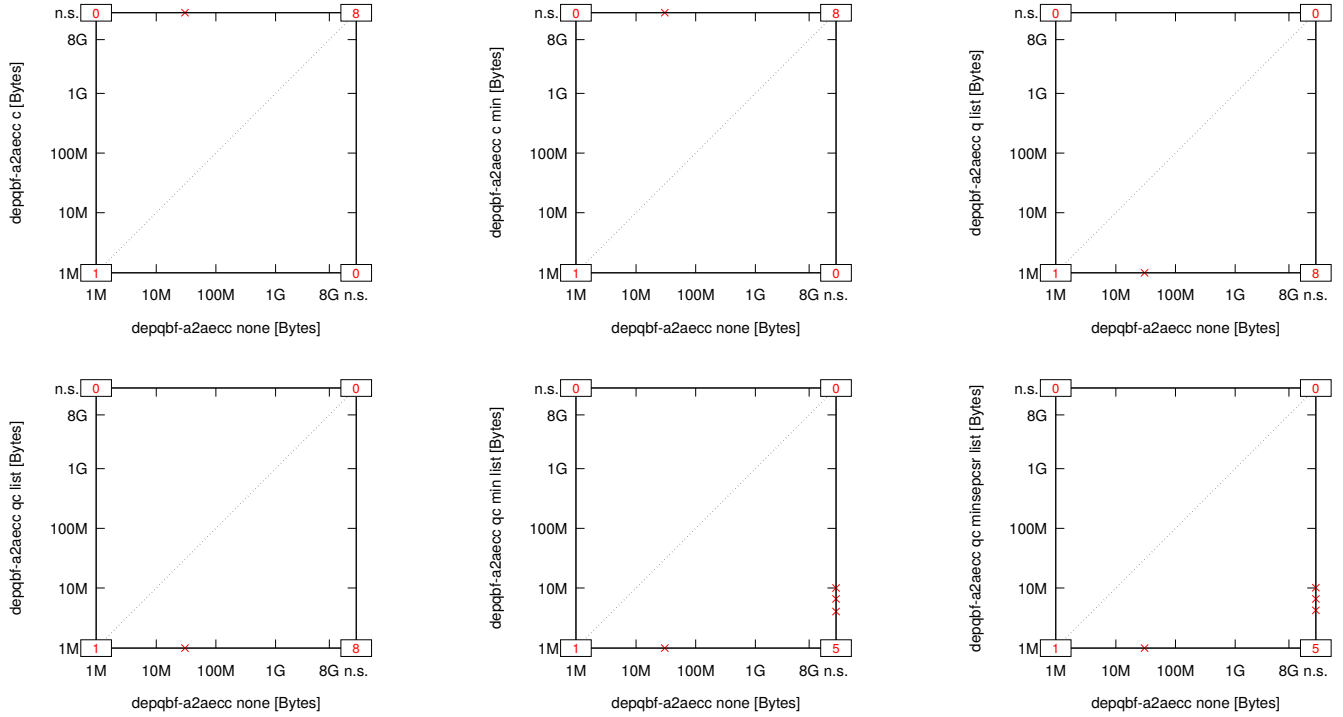


Fig. 1008: Suite Peitl ($n = 10$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

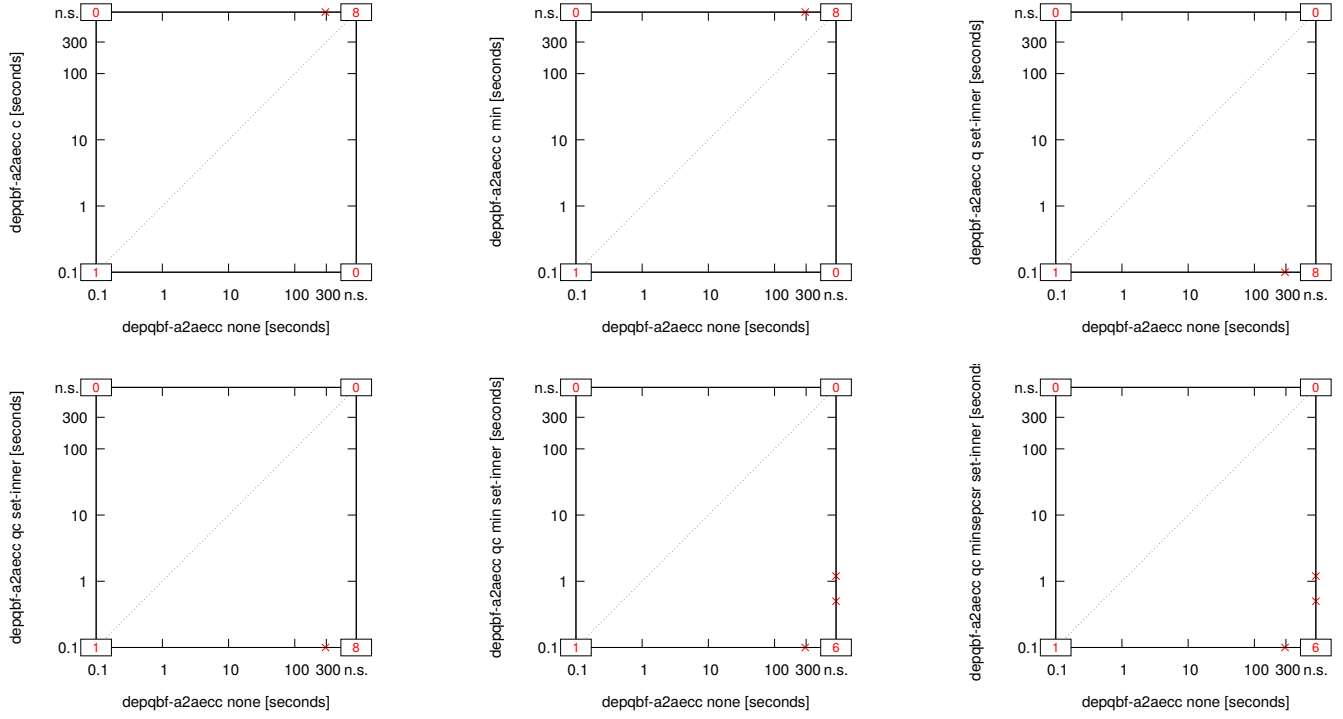


Fig. 1009: Suite Peitl ($n = 10$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

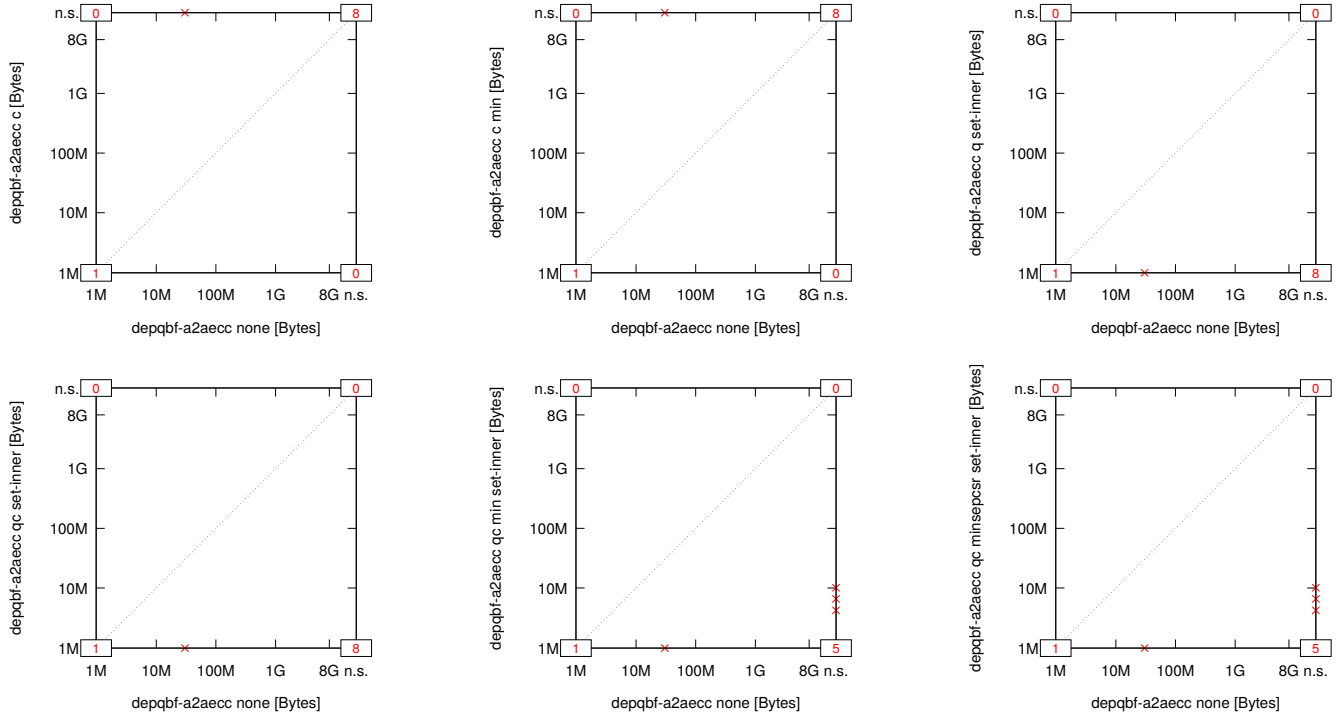


Fig. 1010: Suite Peitl ($n = 10$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

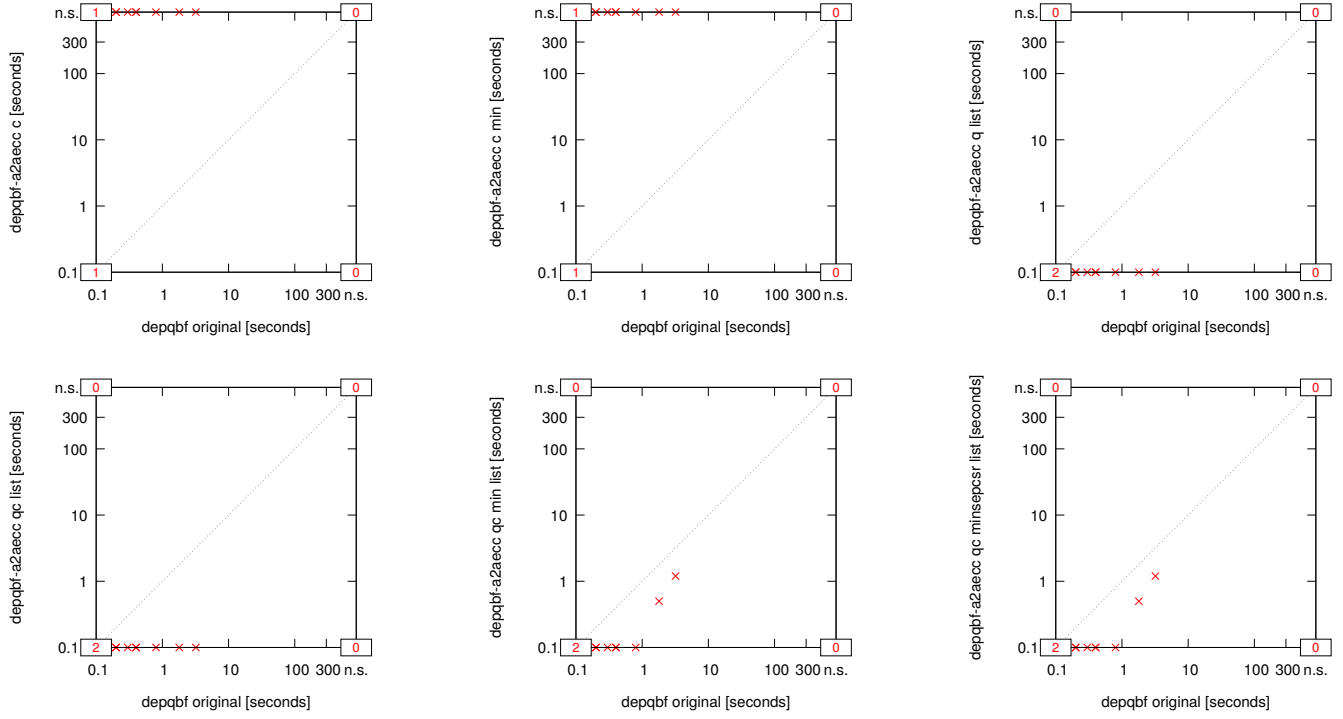


Fig. 1011: Suite Peitl ($n = 10$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

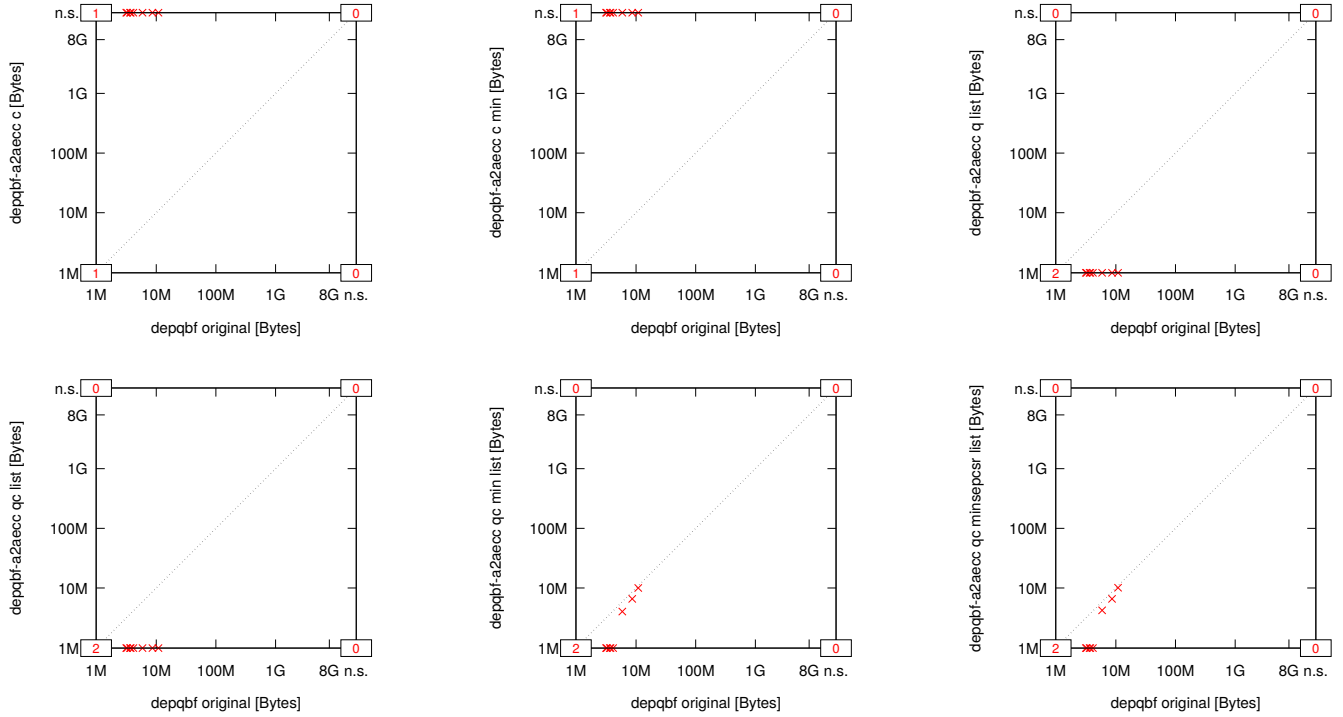


Fig. 1012: Suite Peitl ($n = 10$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

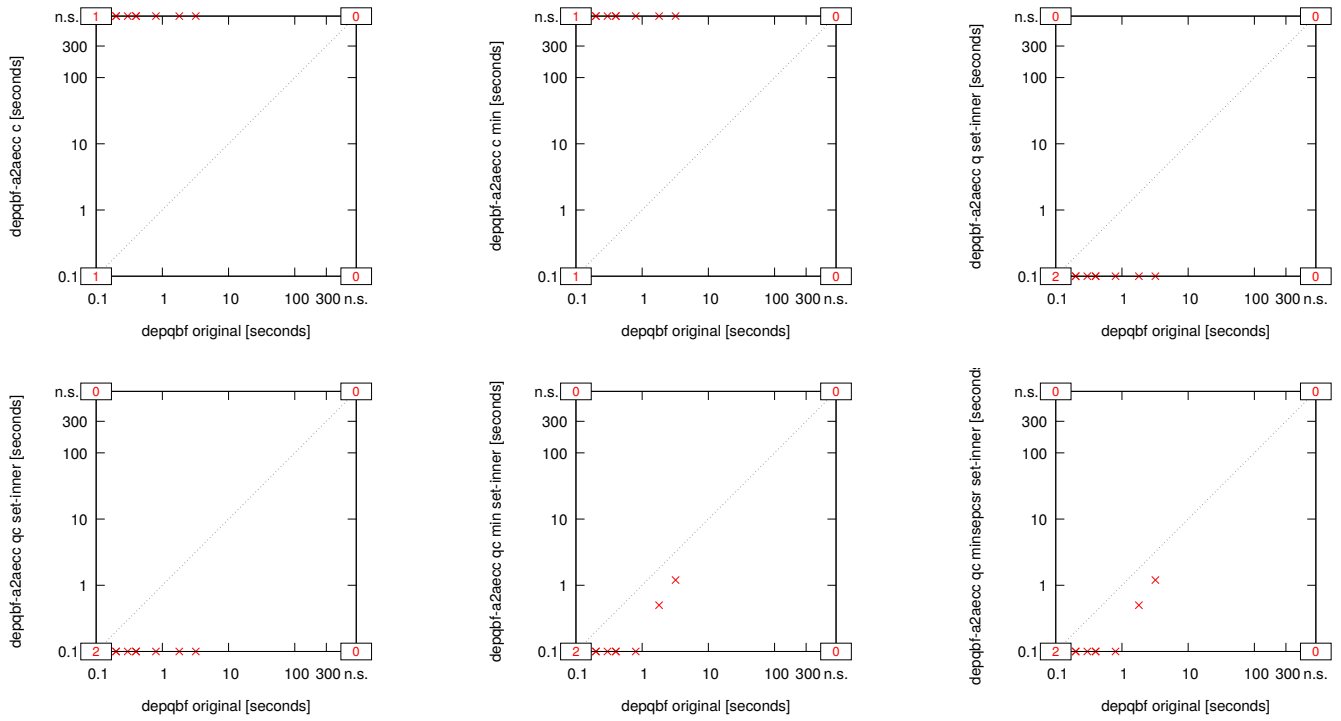


Fig. 1013: Suite Peitl ($n = 10$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

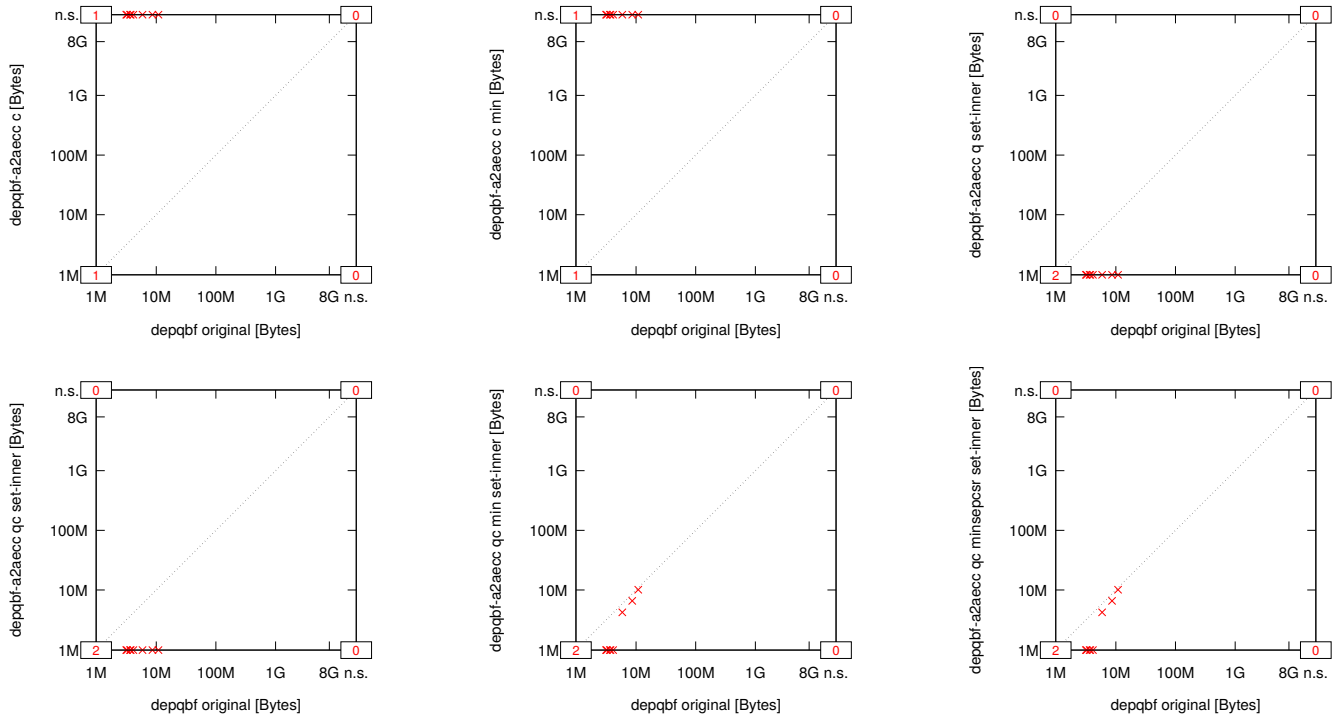


Fig. 1014: Suite Peitl ($n = 10$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

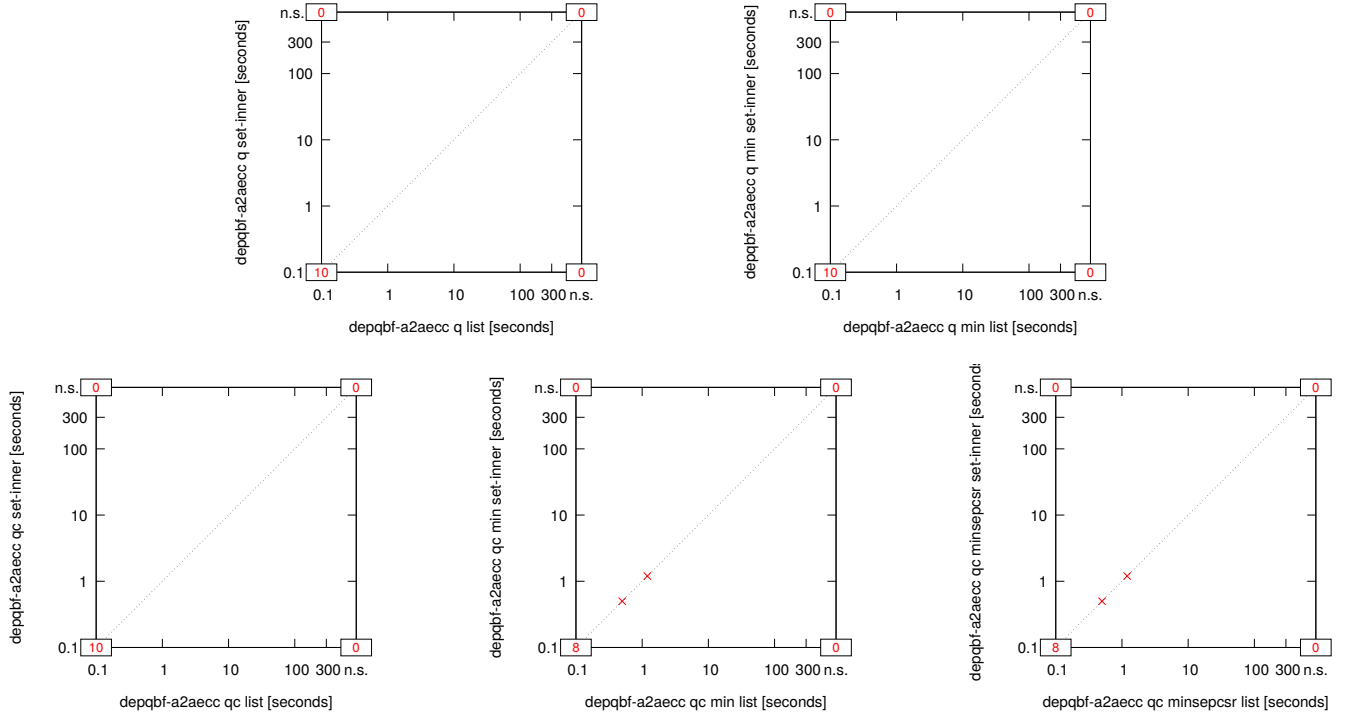


Fig. 1015: Suite Peitl ($n = 10$): Extracting and minimizing unsatisfiable cores in `DepQBF-a2aecc` with set-inner versus list semantics (run time in seconds).

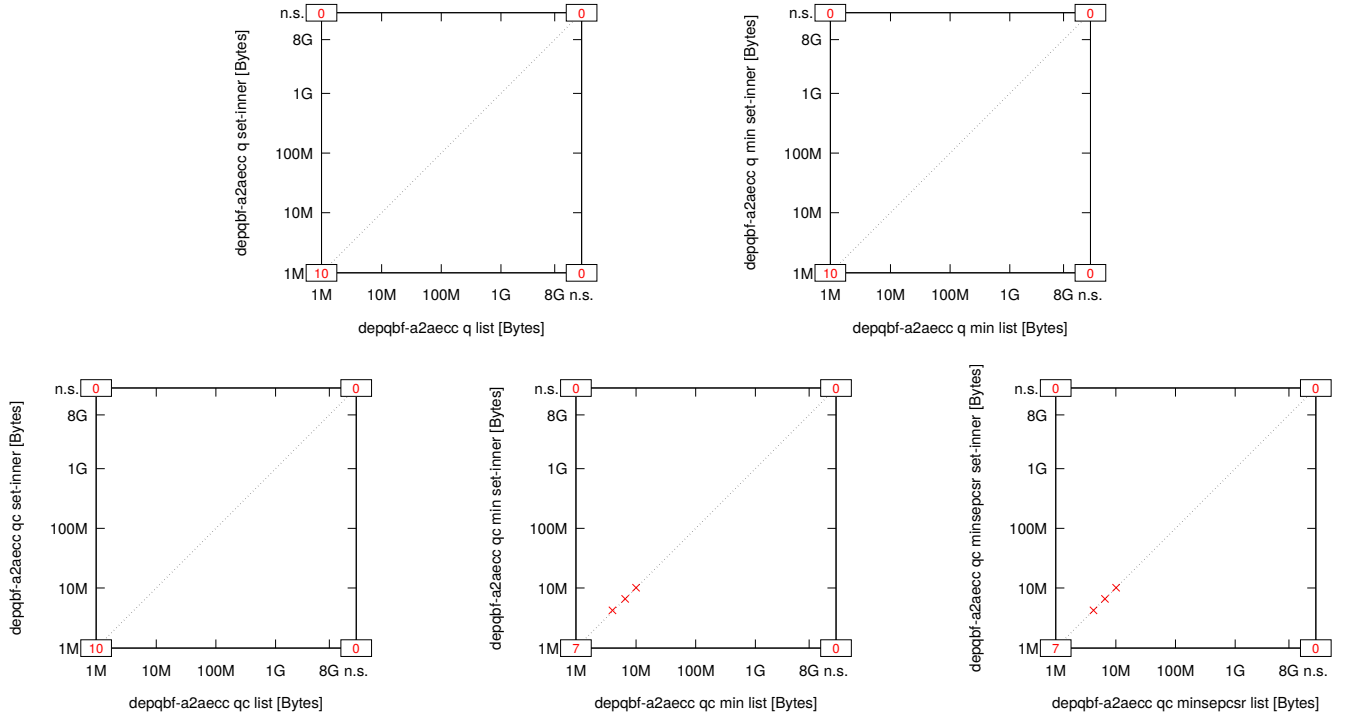


Fig. 1016: Suite Peitl ($n = 10$): Extracting and minimizing unsatisfiable cores in `DepQBF-a2aecc` with set-inner versus list semantics (memory usage in Bytes).

37) Preusser ($n = 0$):

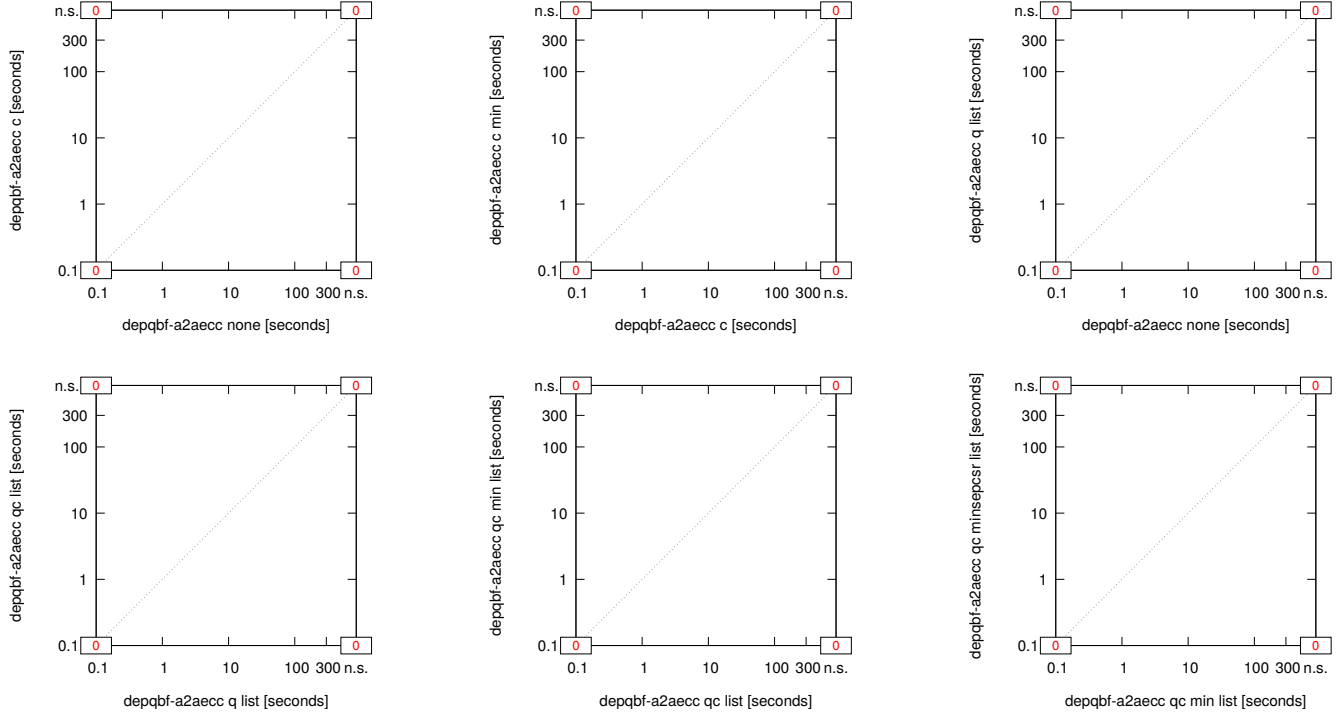


Fig. 1017: Suite Preusser ($n = 0$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

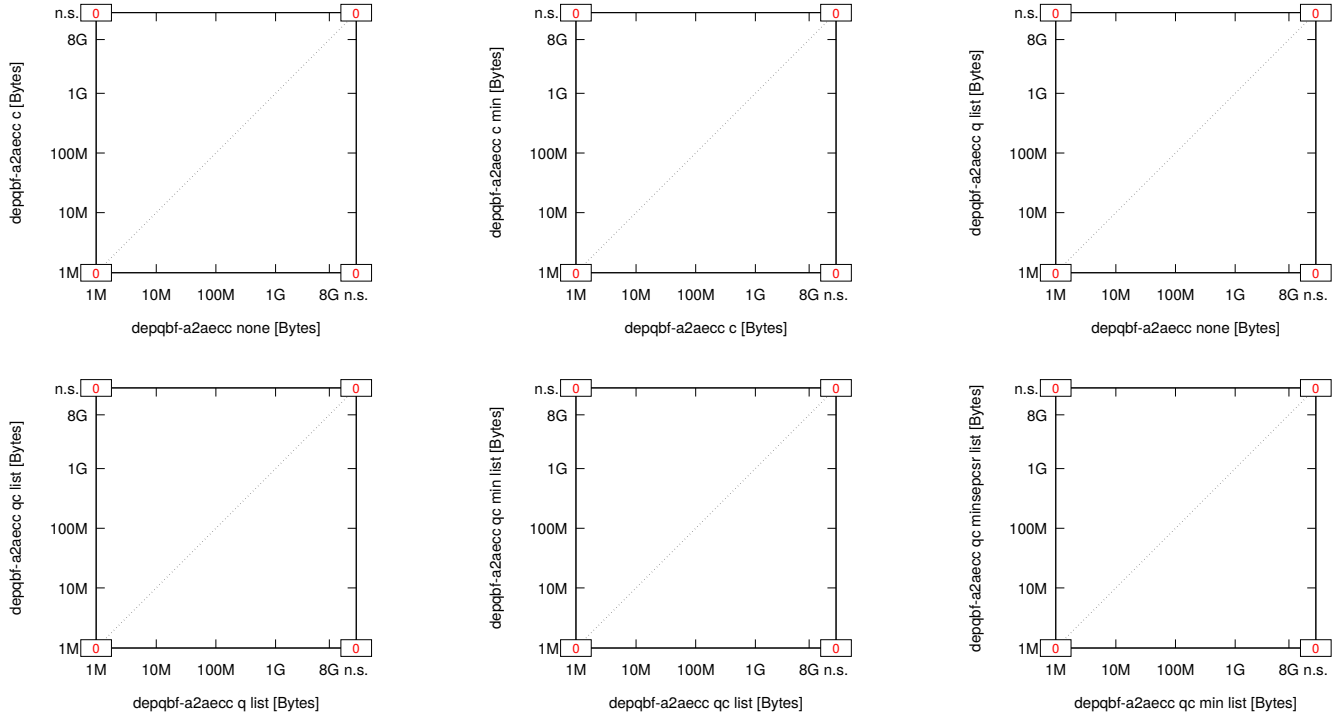


Fig. 1018: Suite Preusser ($n = 0$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

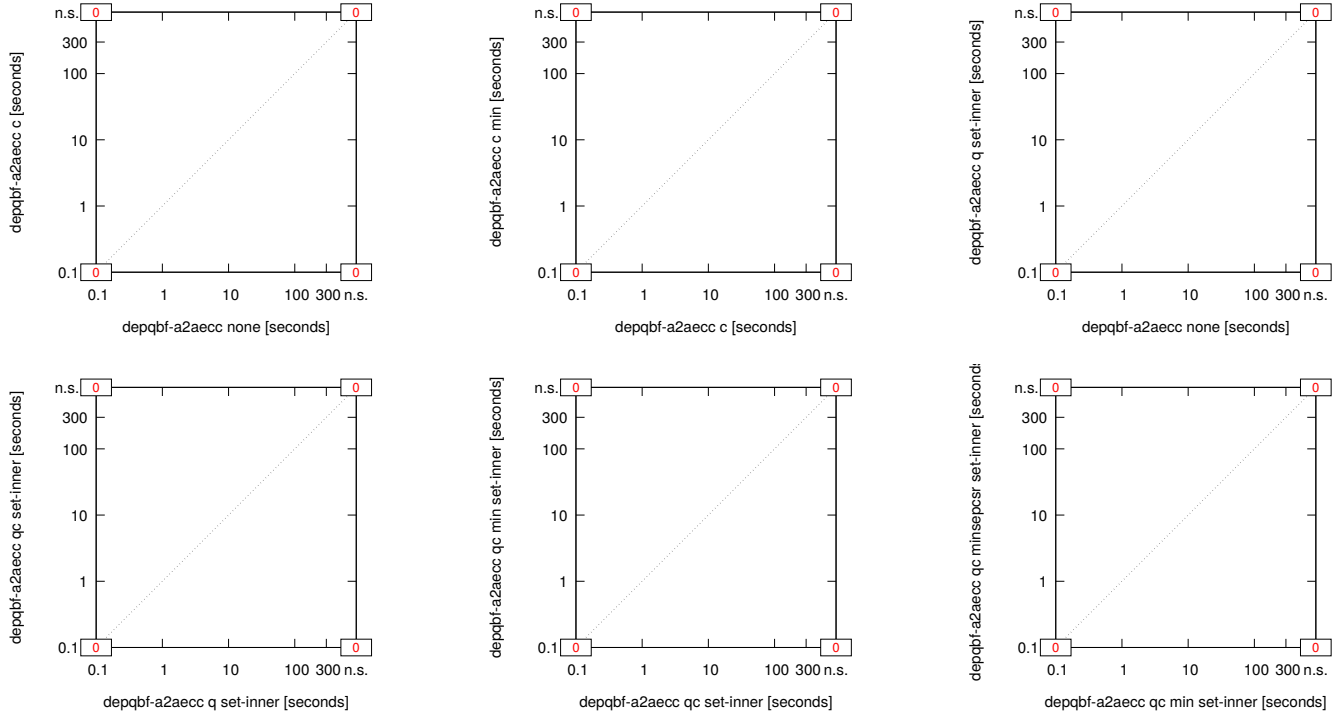


Fig. 1019: Suite Preusser ($n = 0$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

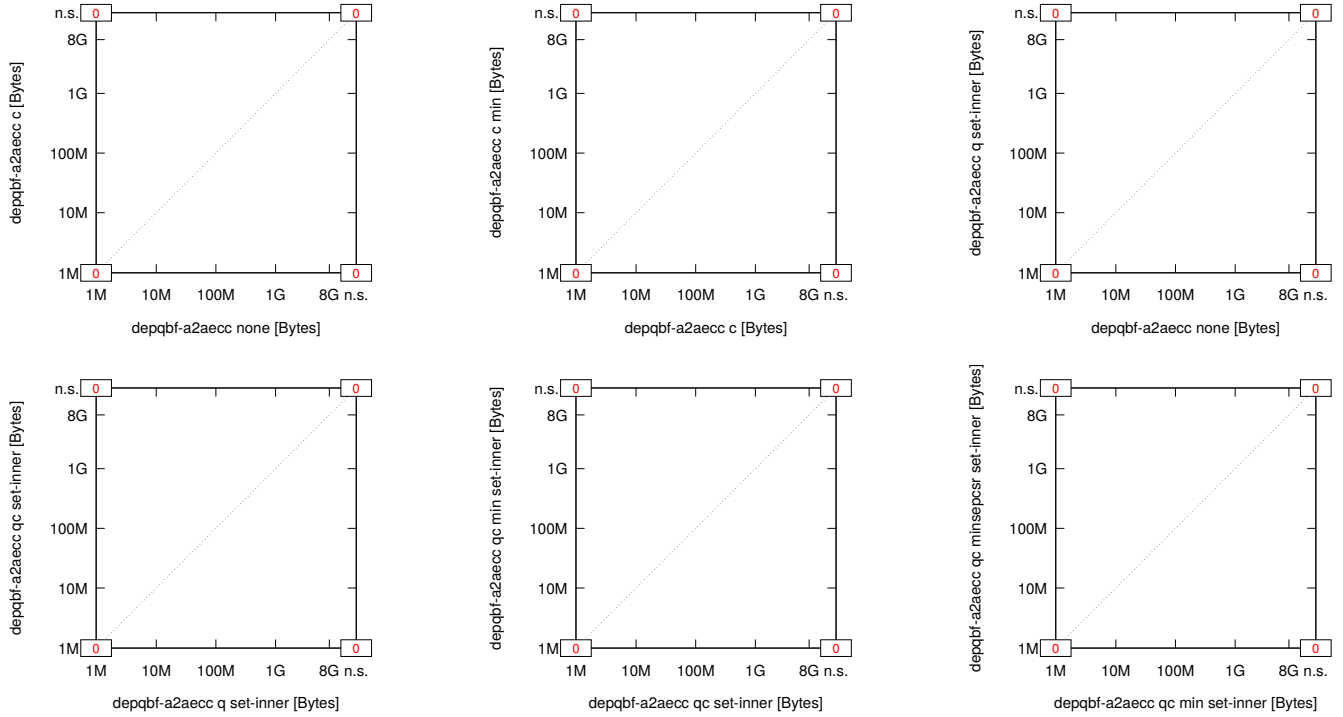


Fig. 1020: Suite Preusser ($n = 0$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

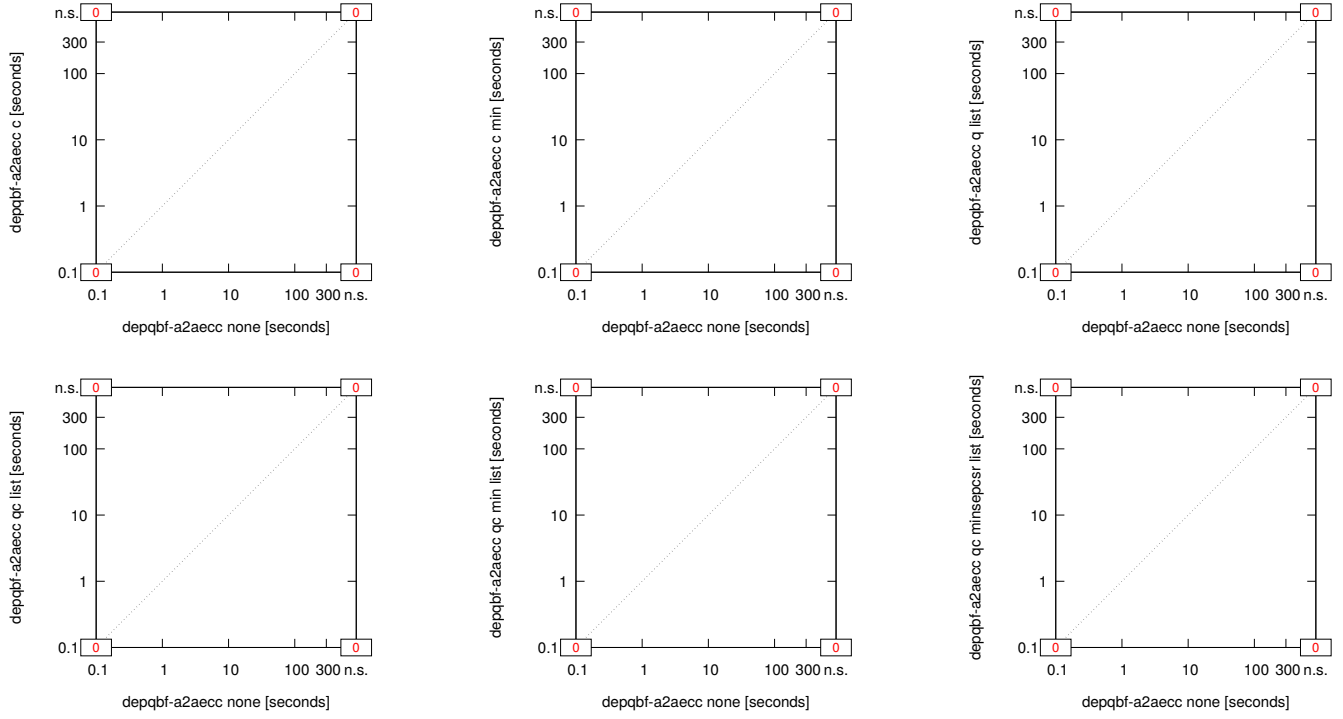


Fig. 1021: Suite Preusser ($n = 0$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

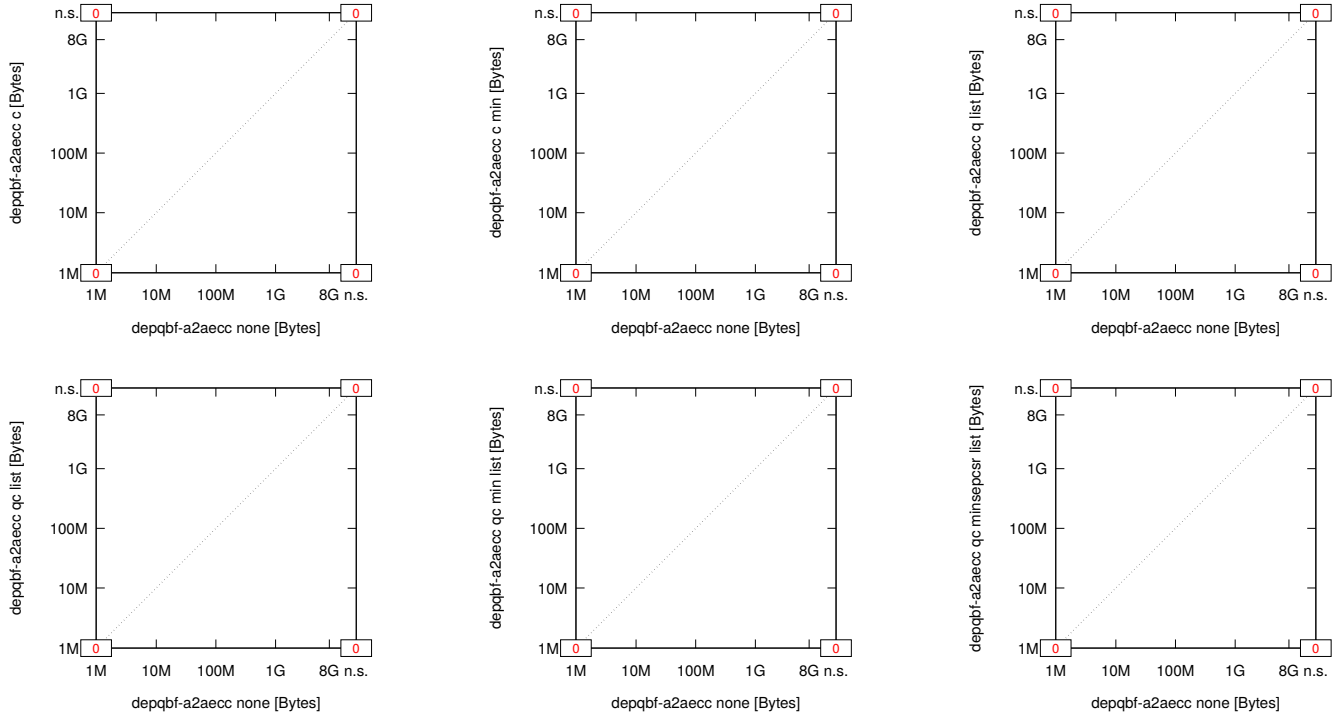


Fig. 1022: Suite Preusser ($n = 0$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

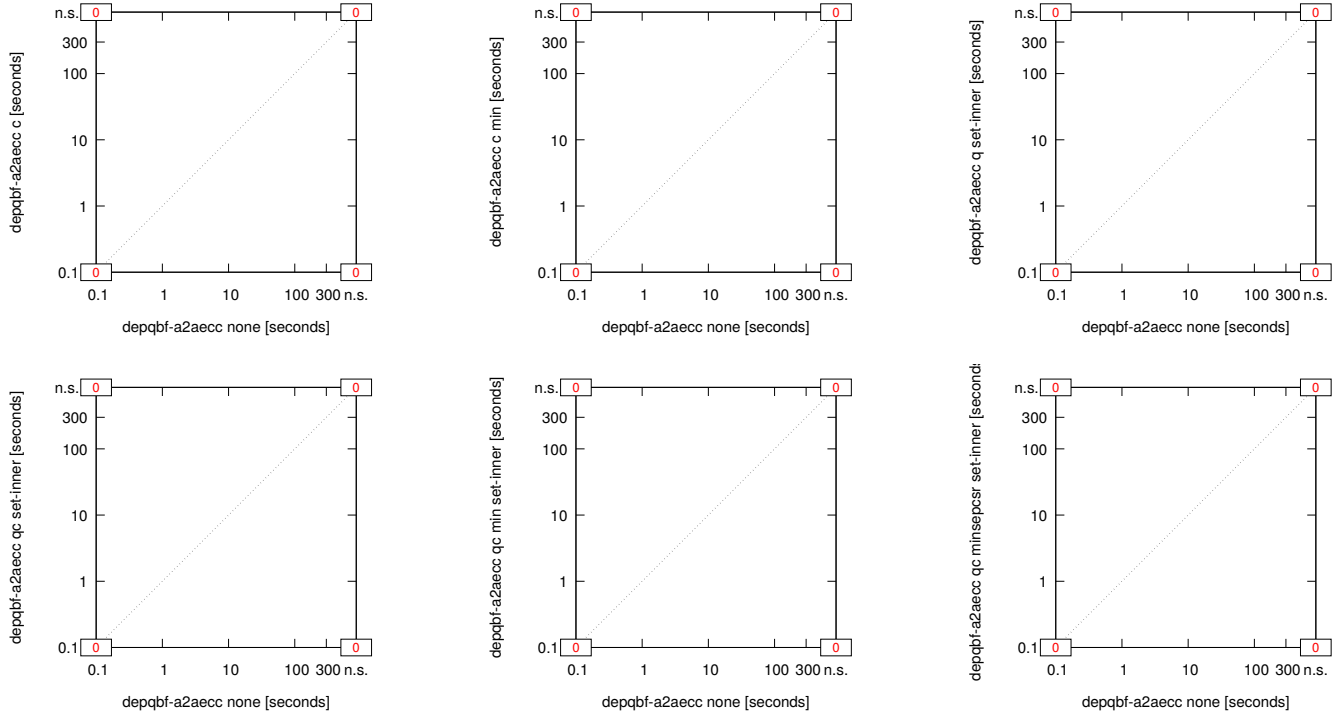


Fig. 1023: Suite Preusser ($n = 0$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

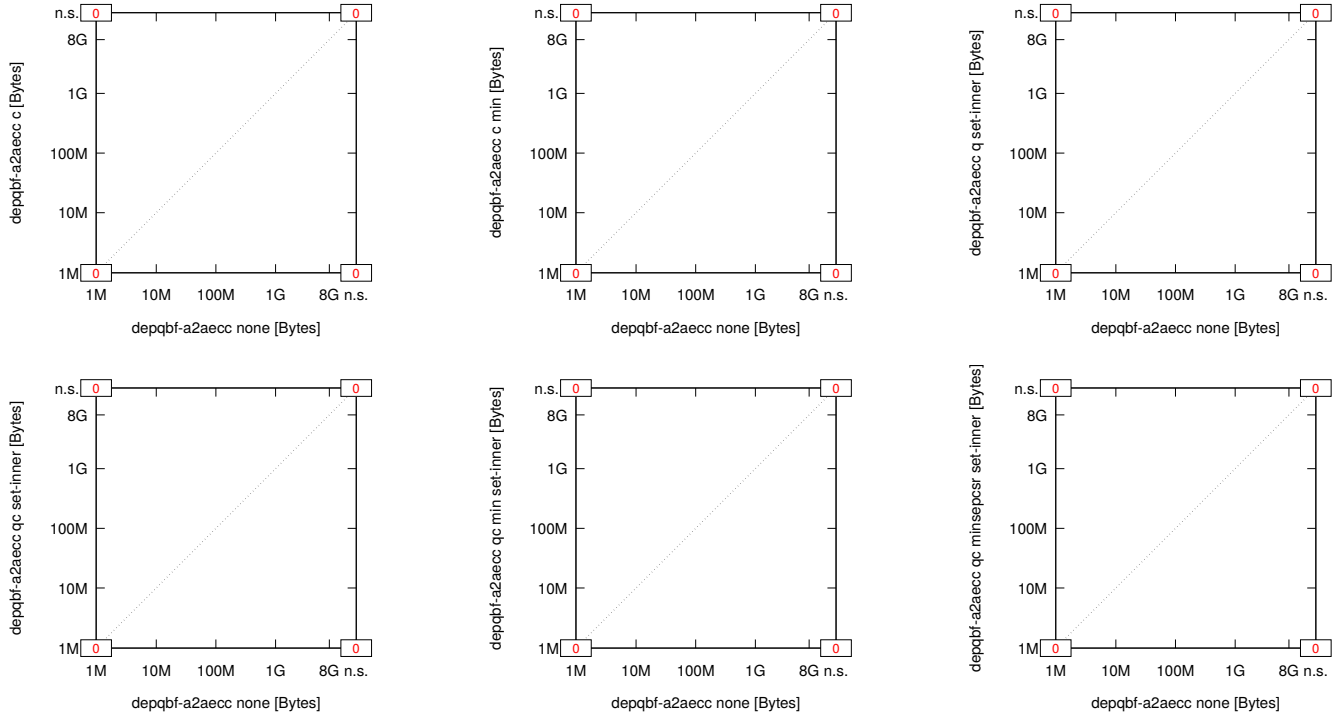


Fig. 1024: Suite Preusser ($n = 0$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

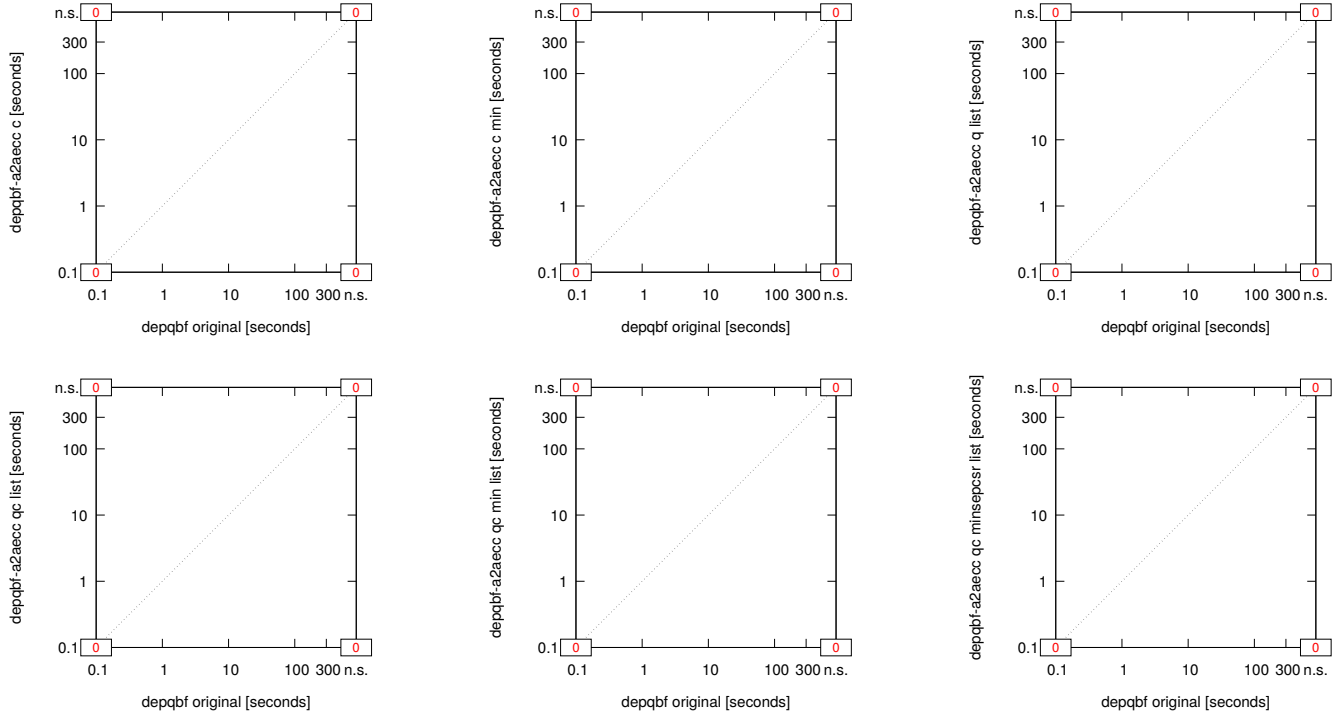


Fig. 1025: Suite Preusser ($n = 0$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

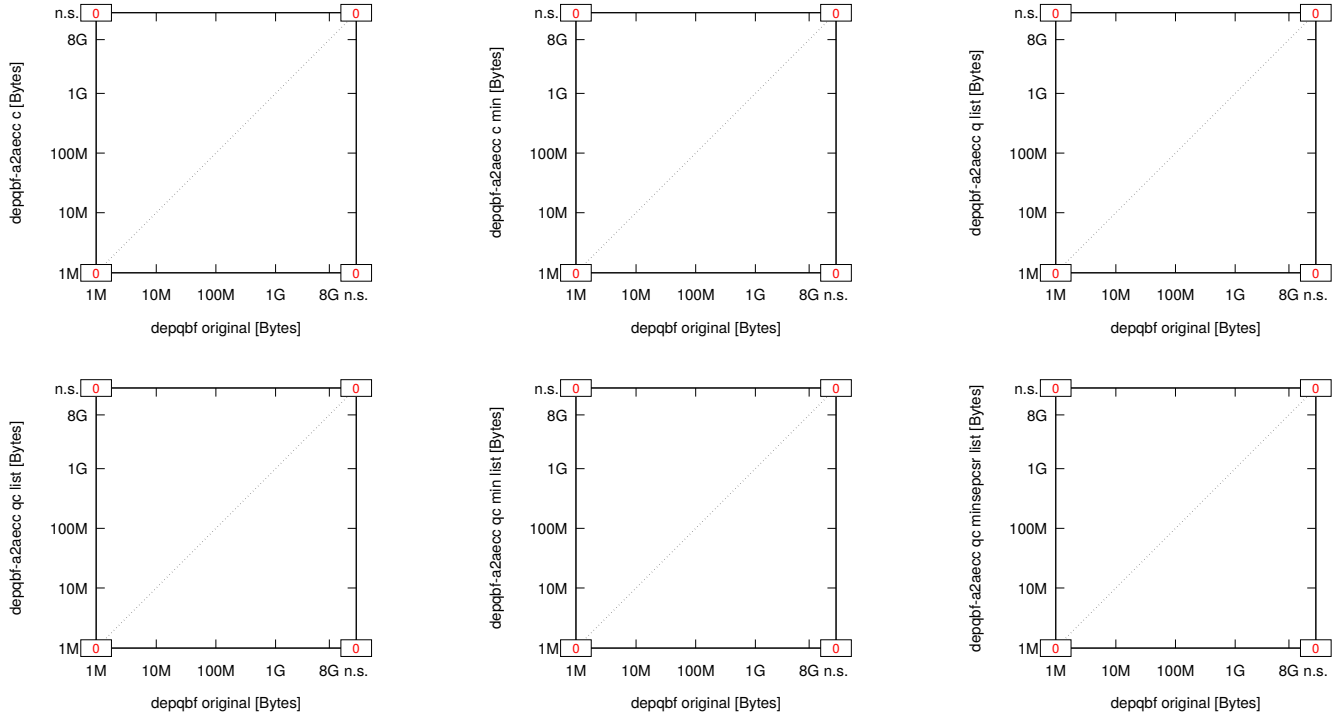


Fig. 1026: Suite Preusser ($n = 0$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

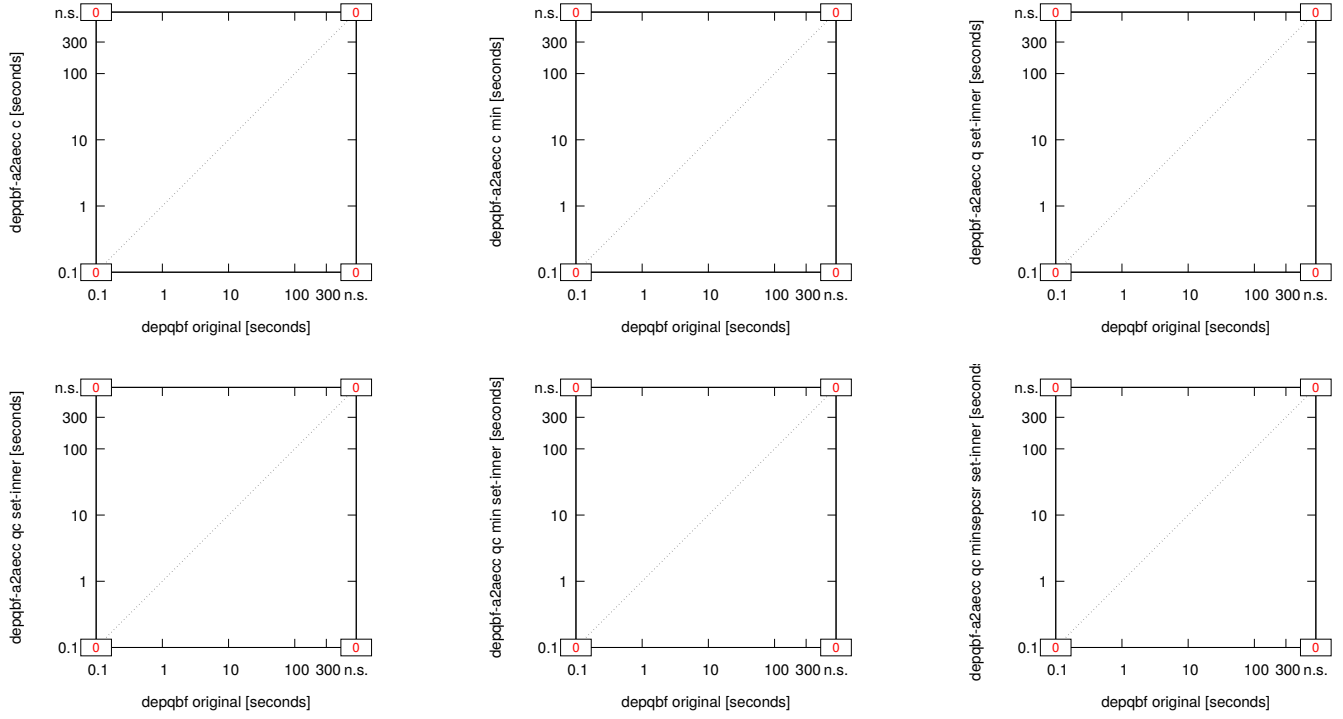


Fig. 1027: Suite Preusser ($n = 0$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

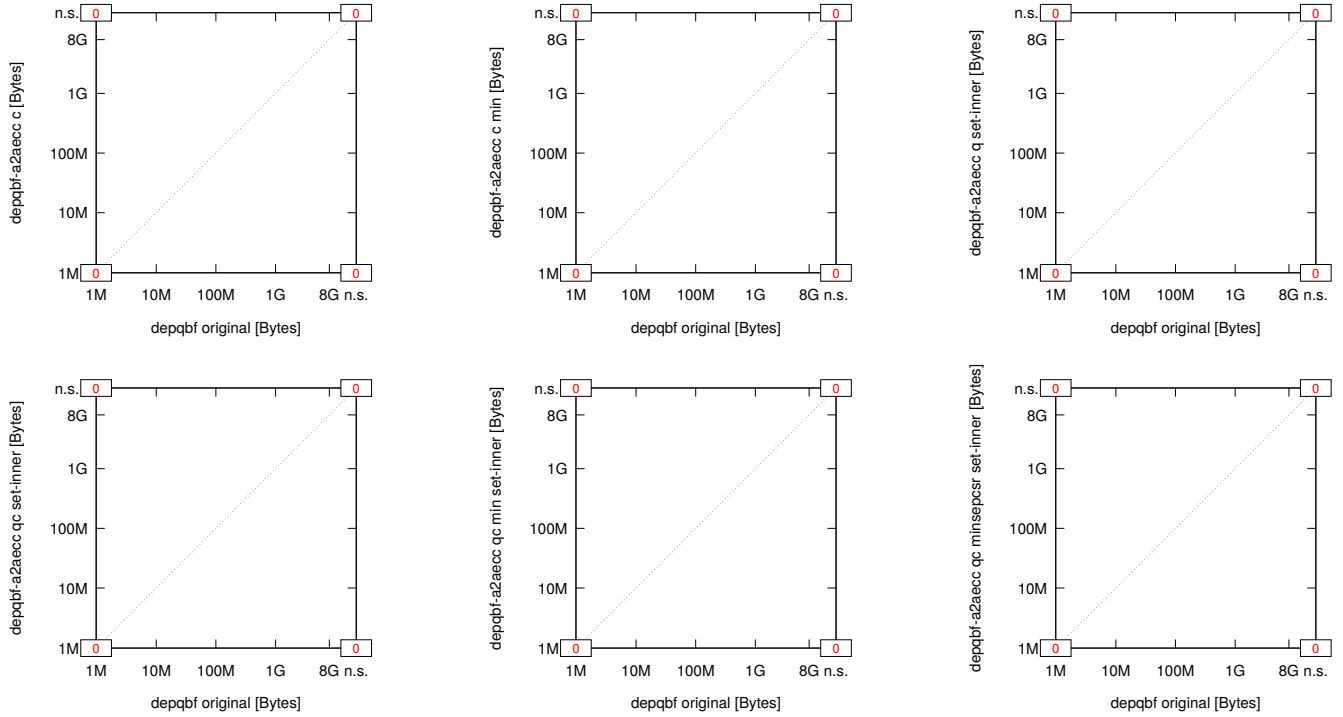


Fig. 1028: Suite Preusser ($n = 0$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

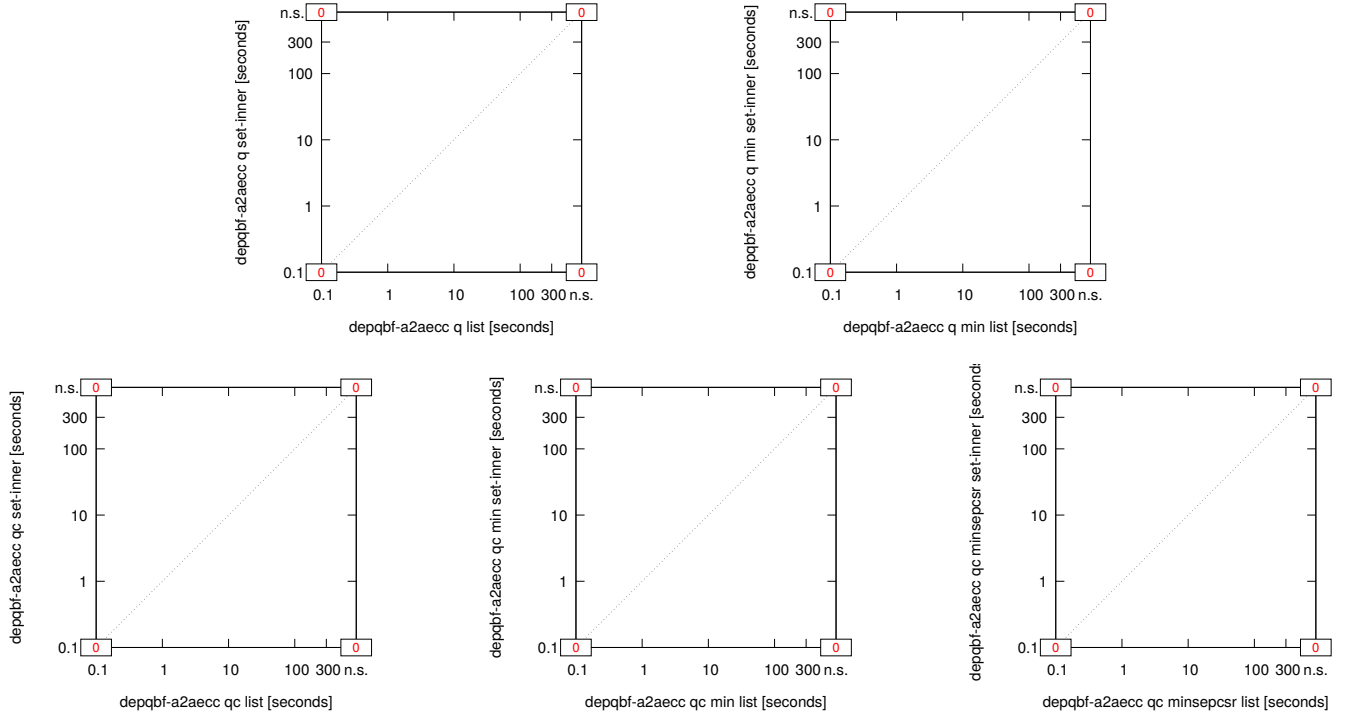


Fig. 1029: Suite Preusser ($n = 0$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

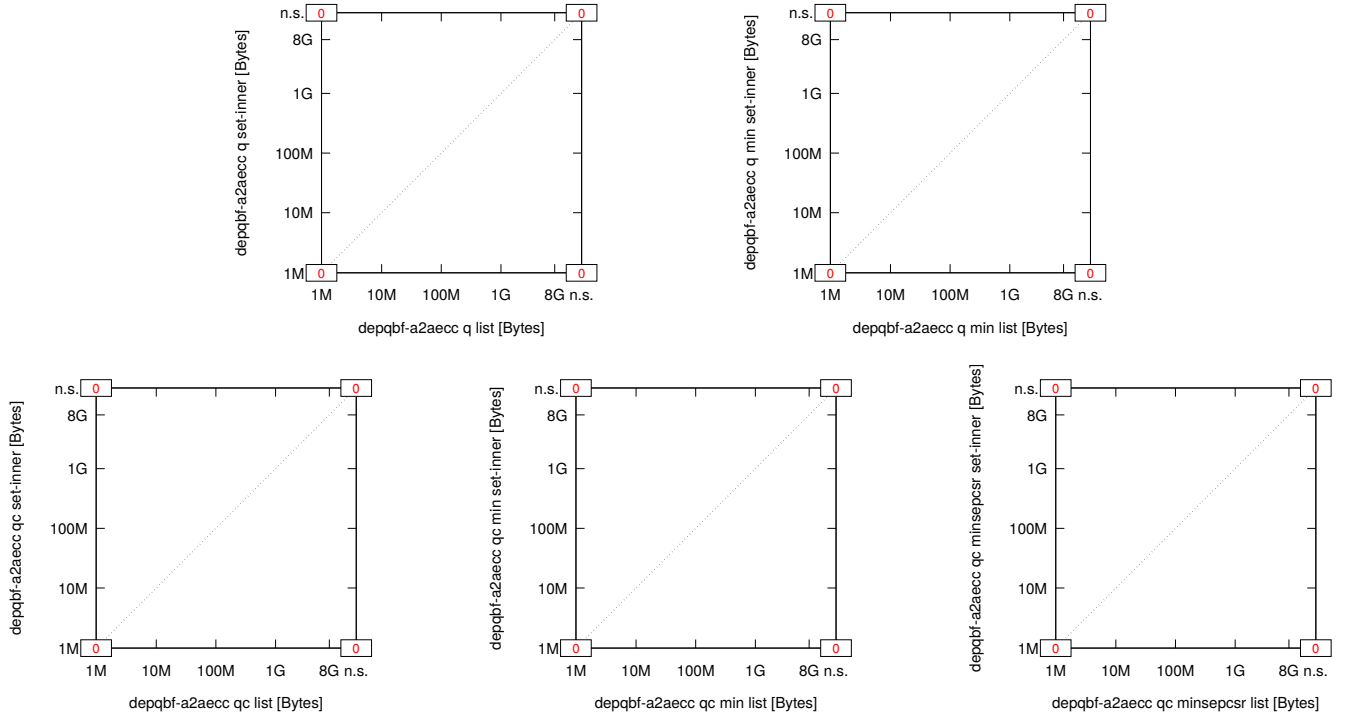


Fig. 1030: Suite Preusser ($n = 0$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

38) *qbfeval12* ($n = 8$):

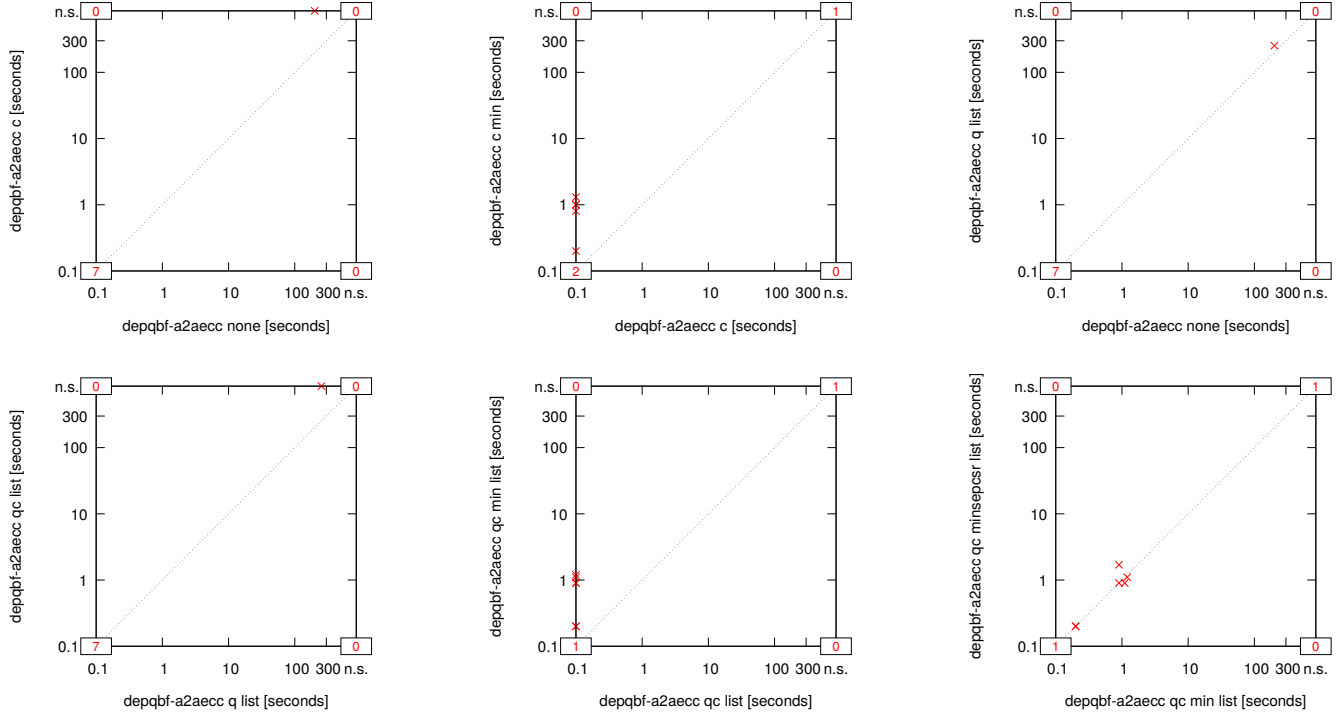


Fig. 1031: Suite *qbfeval12* ($n = 8$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

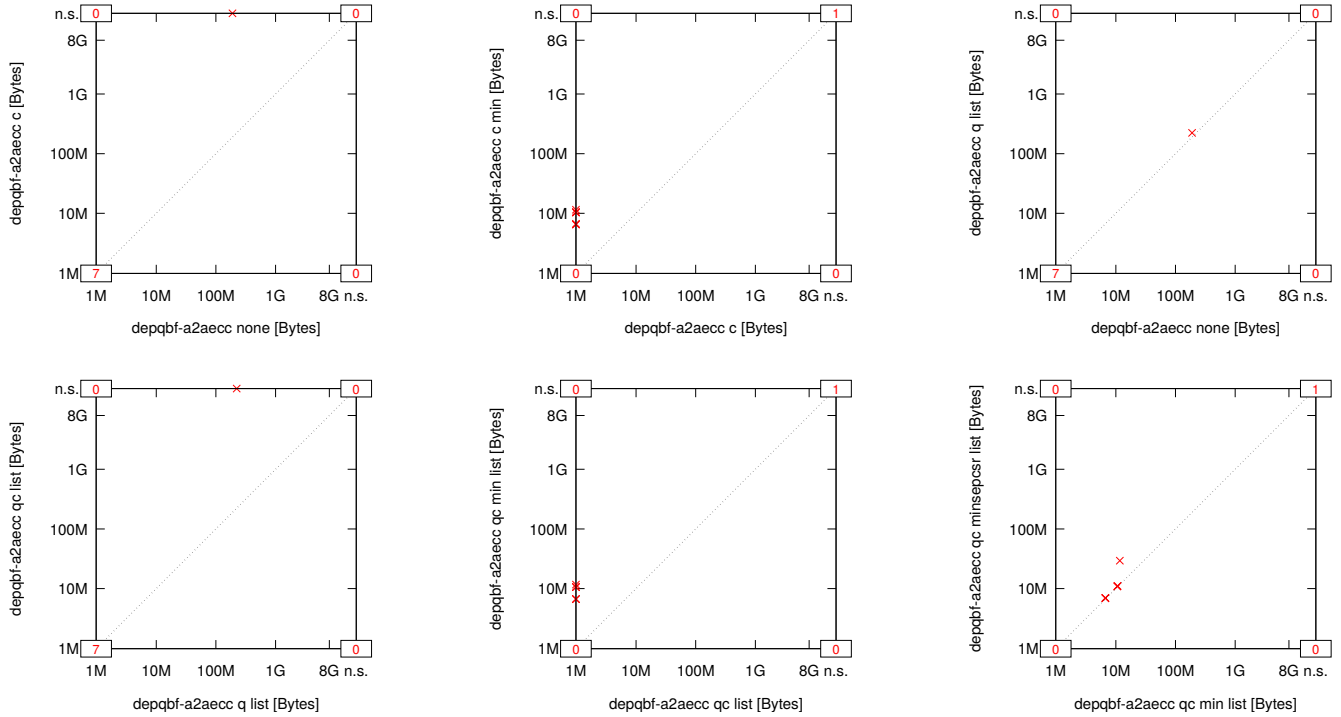


Fig. 1032: Suite *qbfeval12* ($n = 8$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

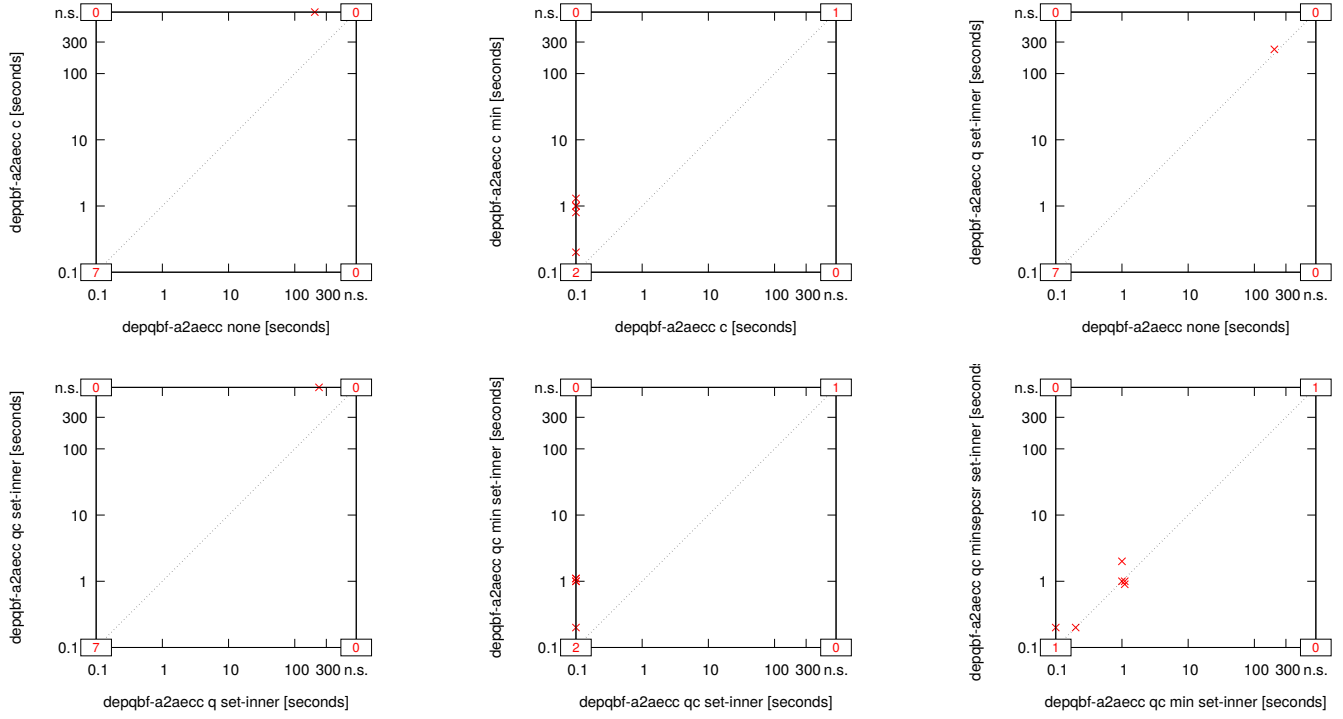


Fig. 1033: Suite qbfeval12 ($n = 8$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

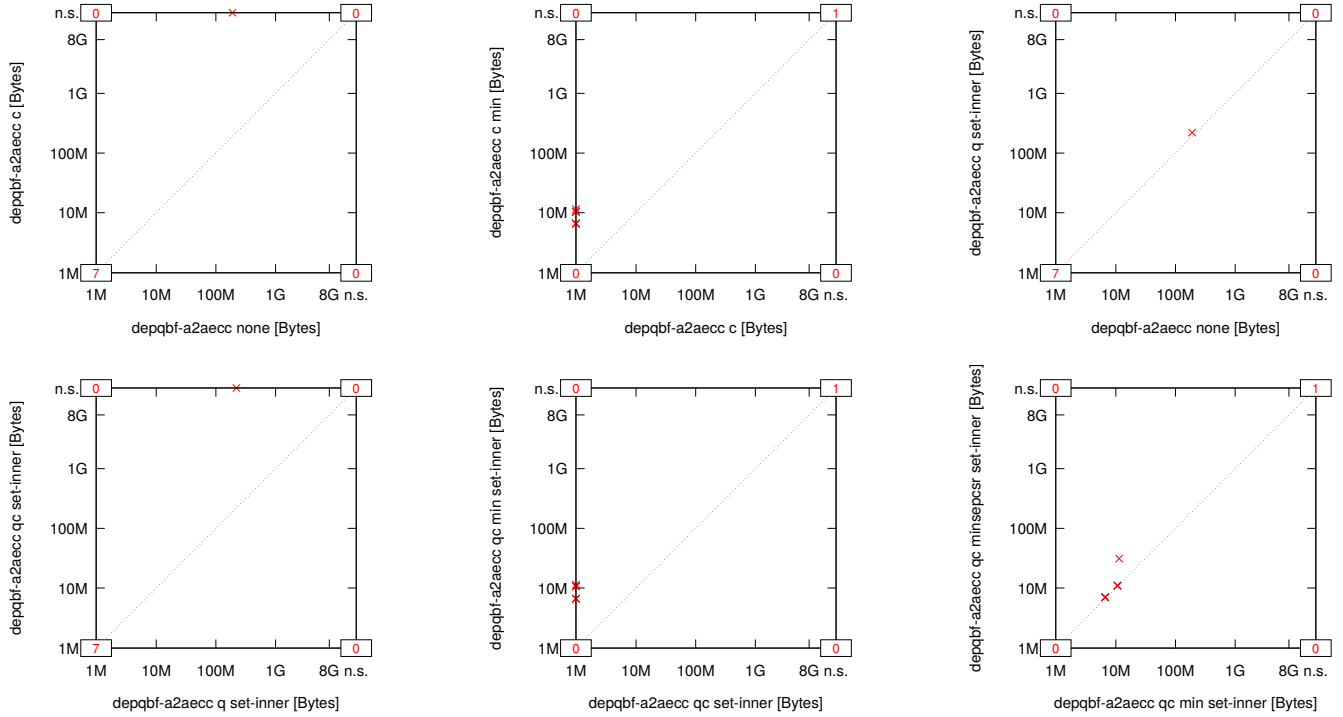


Fig. 1034: Suite qbfeval12 ($n = 8$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

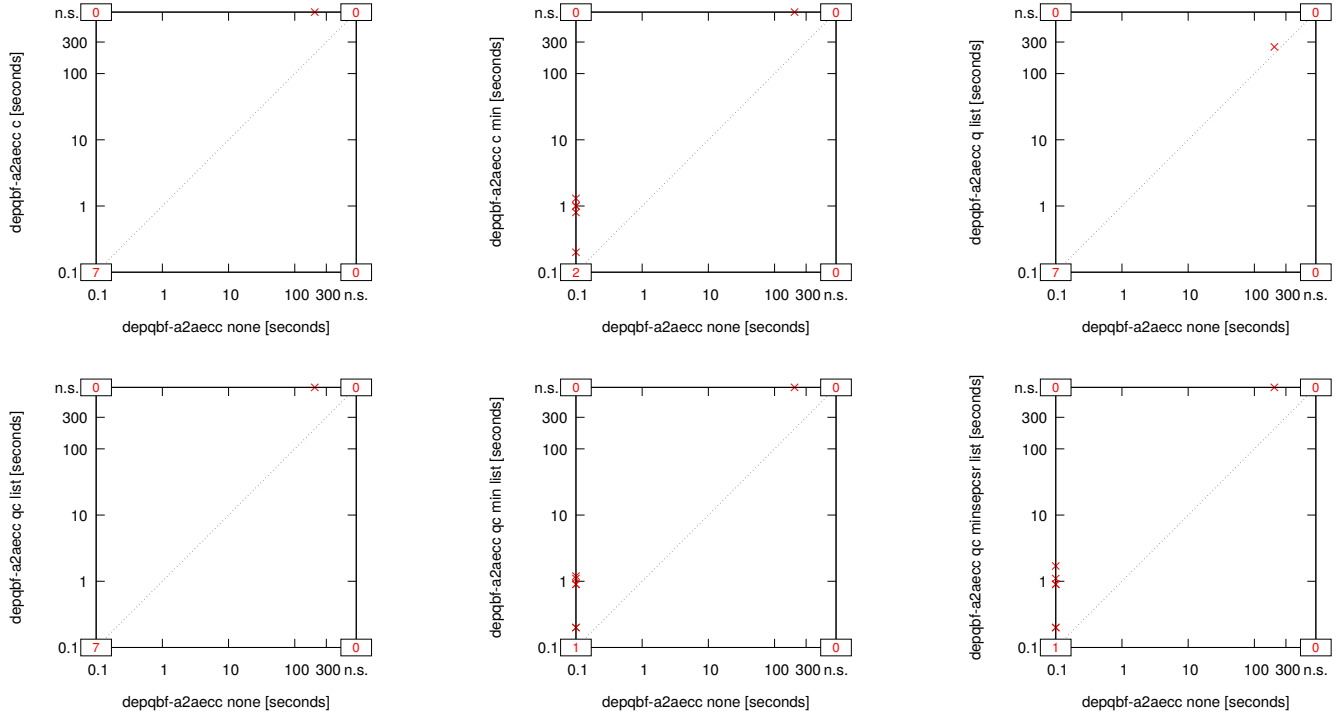


Fig. 1035: Suite qbfeval12 ($n = 8$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

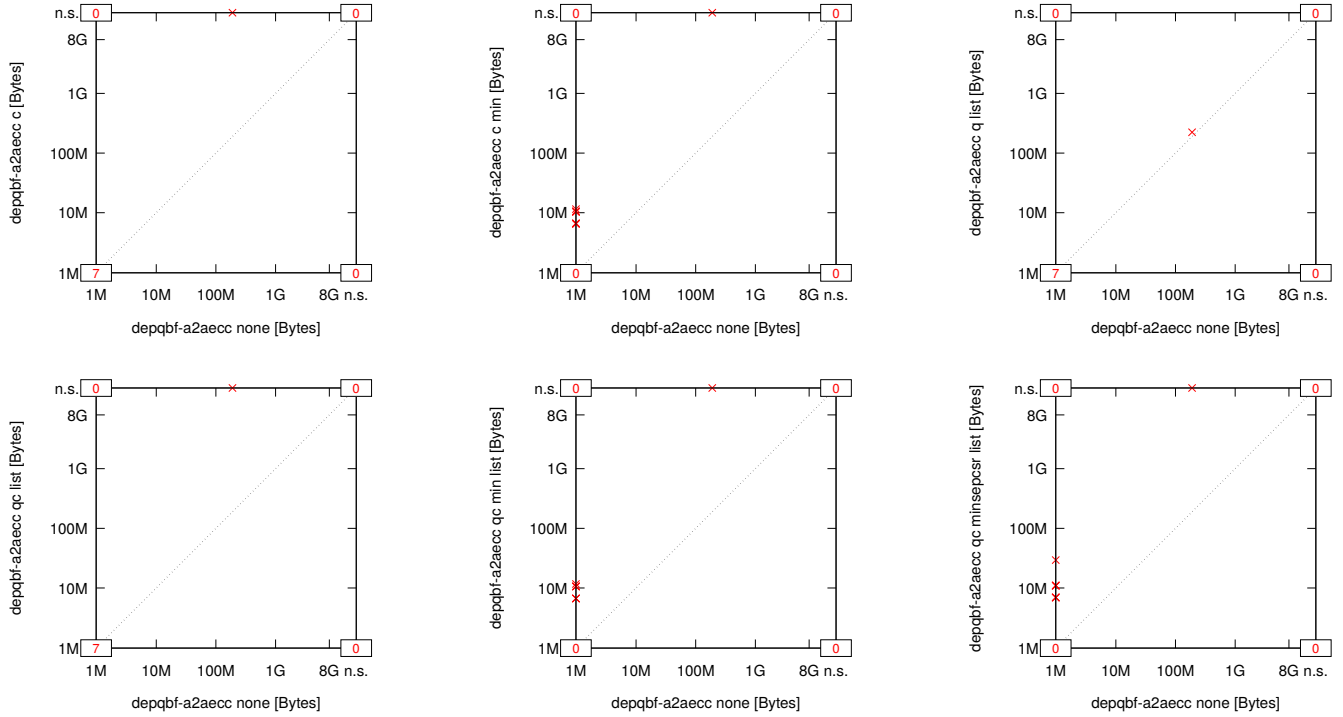


Fig. 1036: Suite qbfeval12 ($n = 8$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

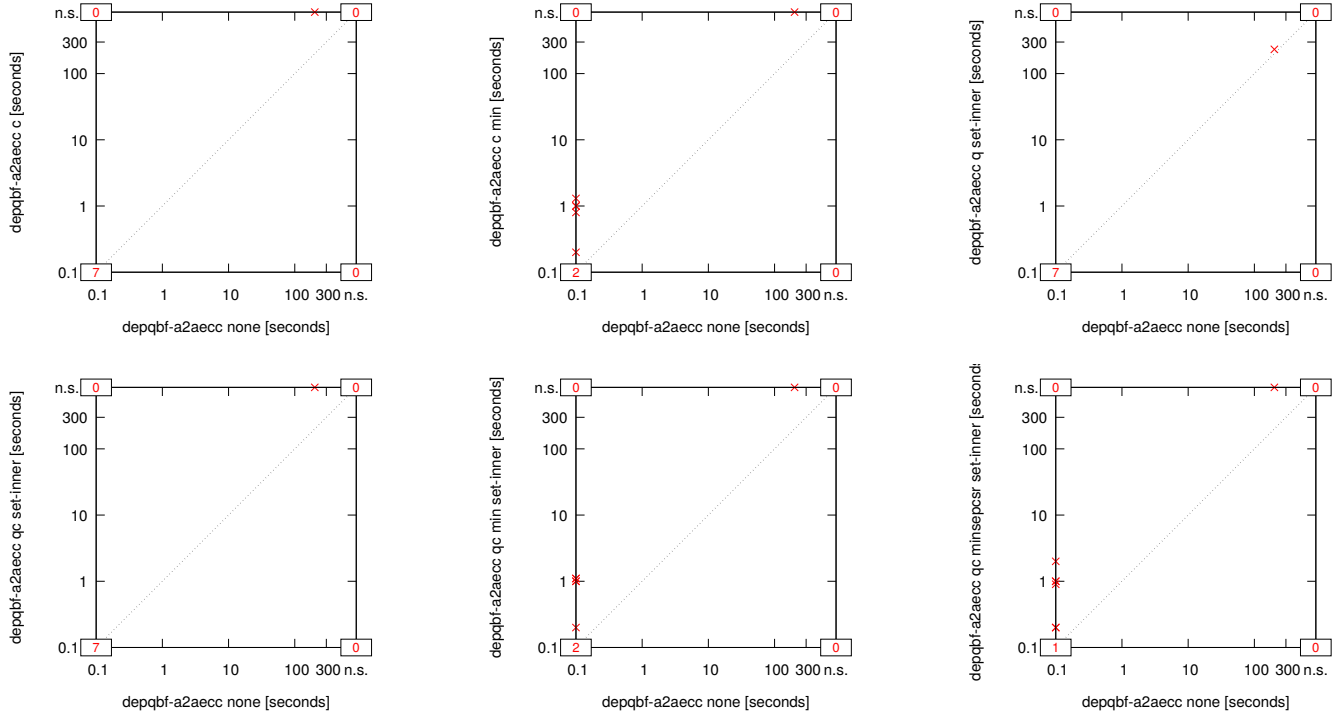


Fig. 1037: Suite qbfeval12 ($n = 8$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

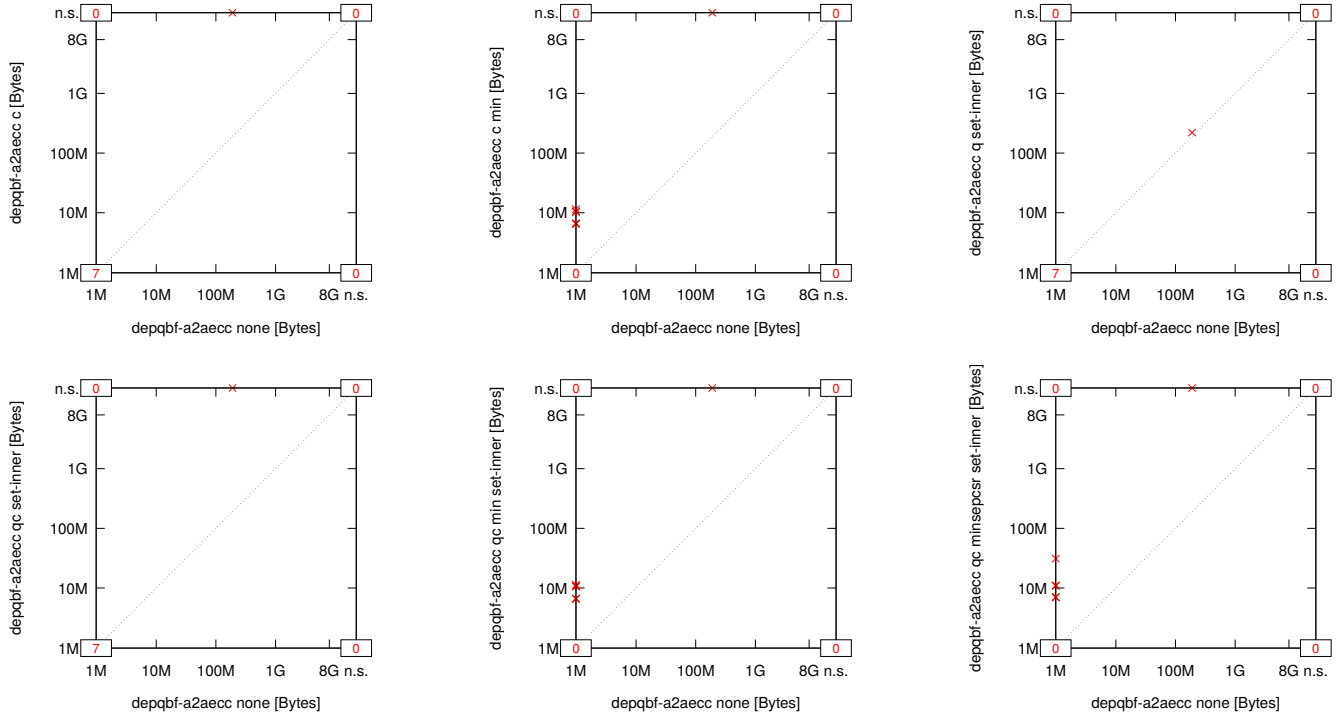


Fig. 1038: Suite qbfeval12 ($n = 8$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

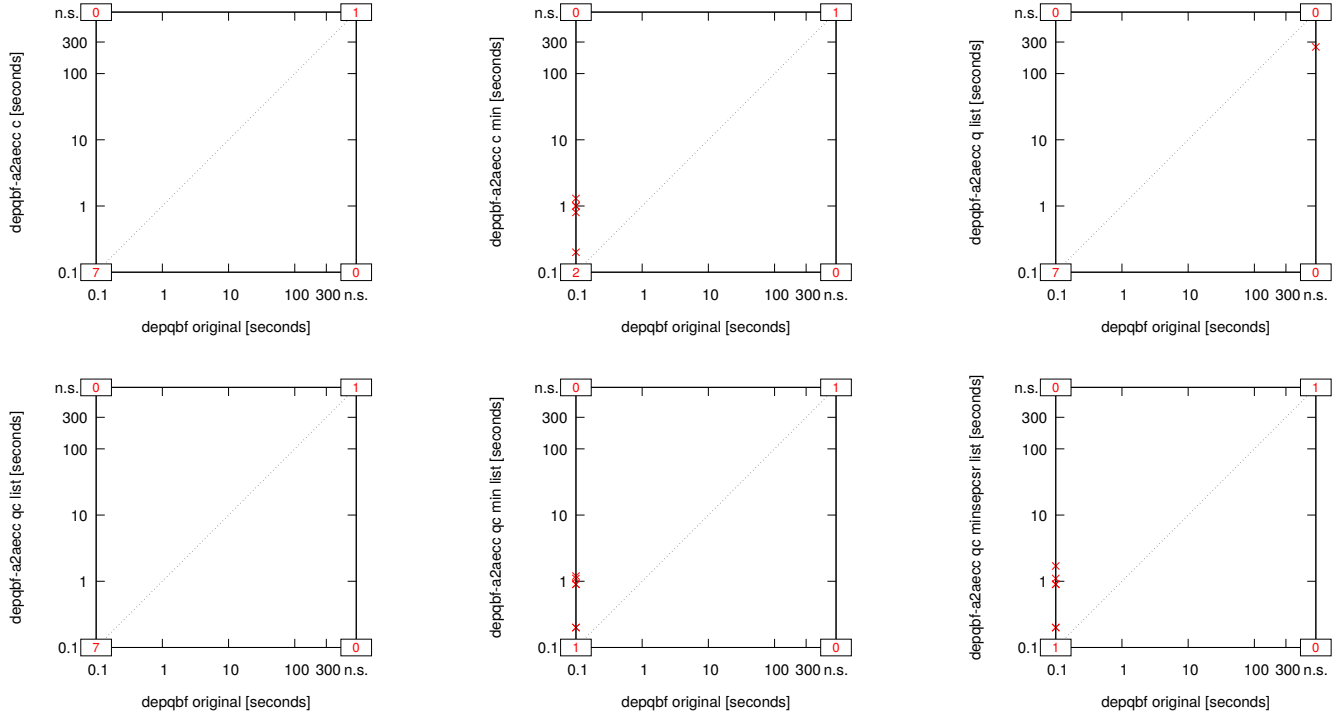


Fig. 1039: Suite qbfeval12 ($n = 8$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

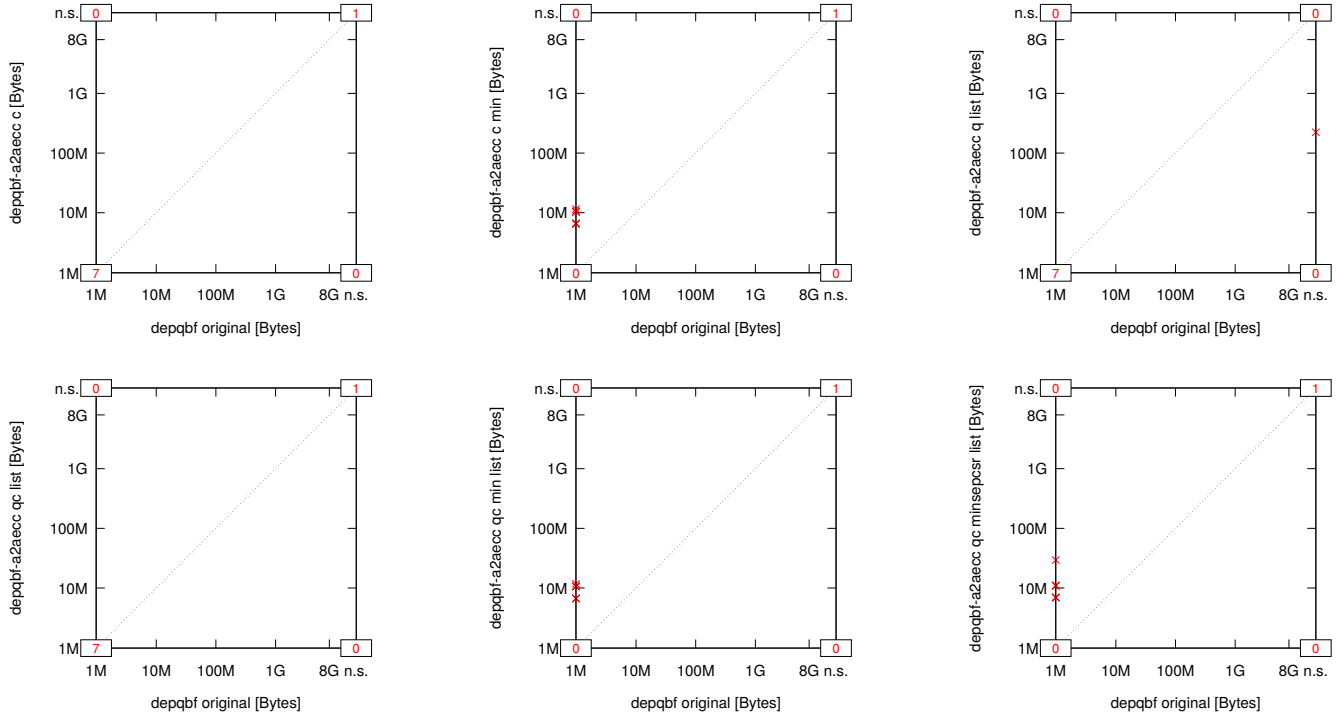


Fig. 1040: Suite qbfeval12 ($n = 8$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

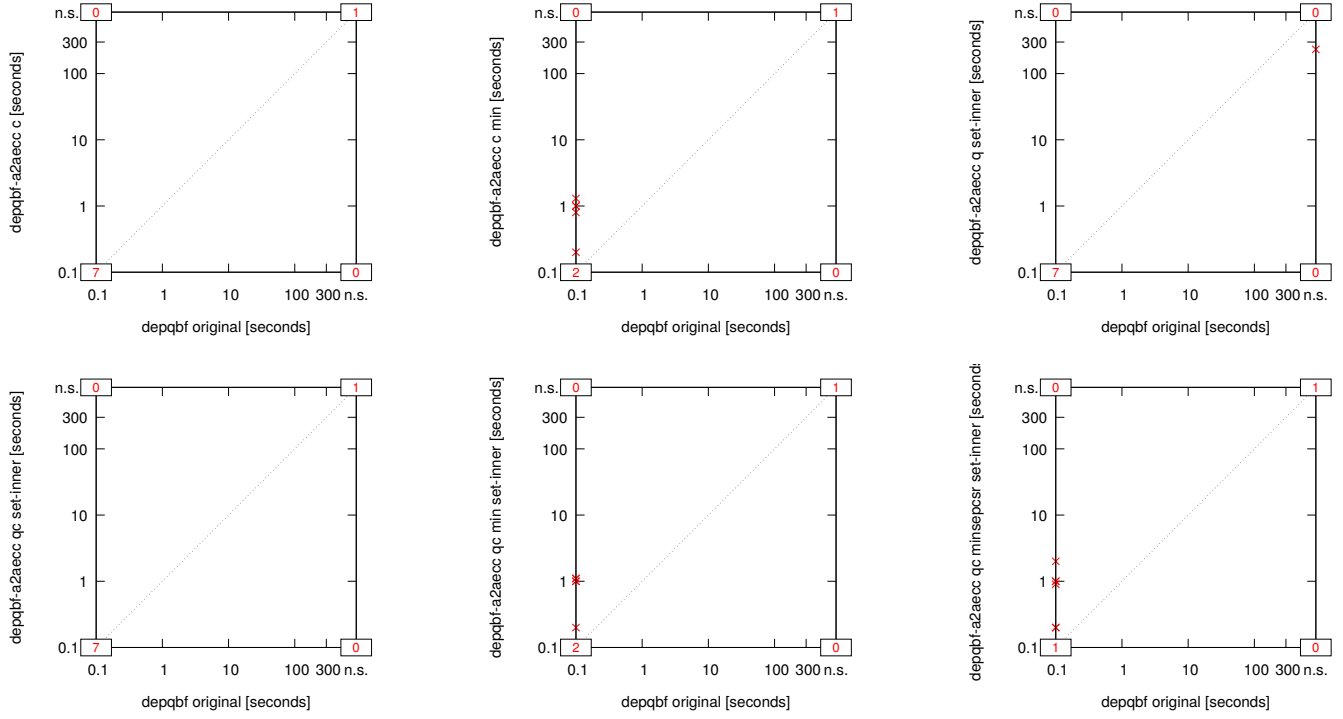


Fig. 1041: Suite qbfeval12 ($n = 8$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

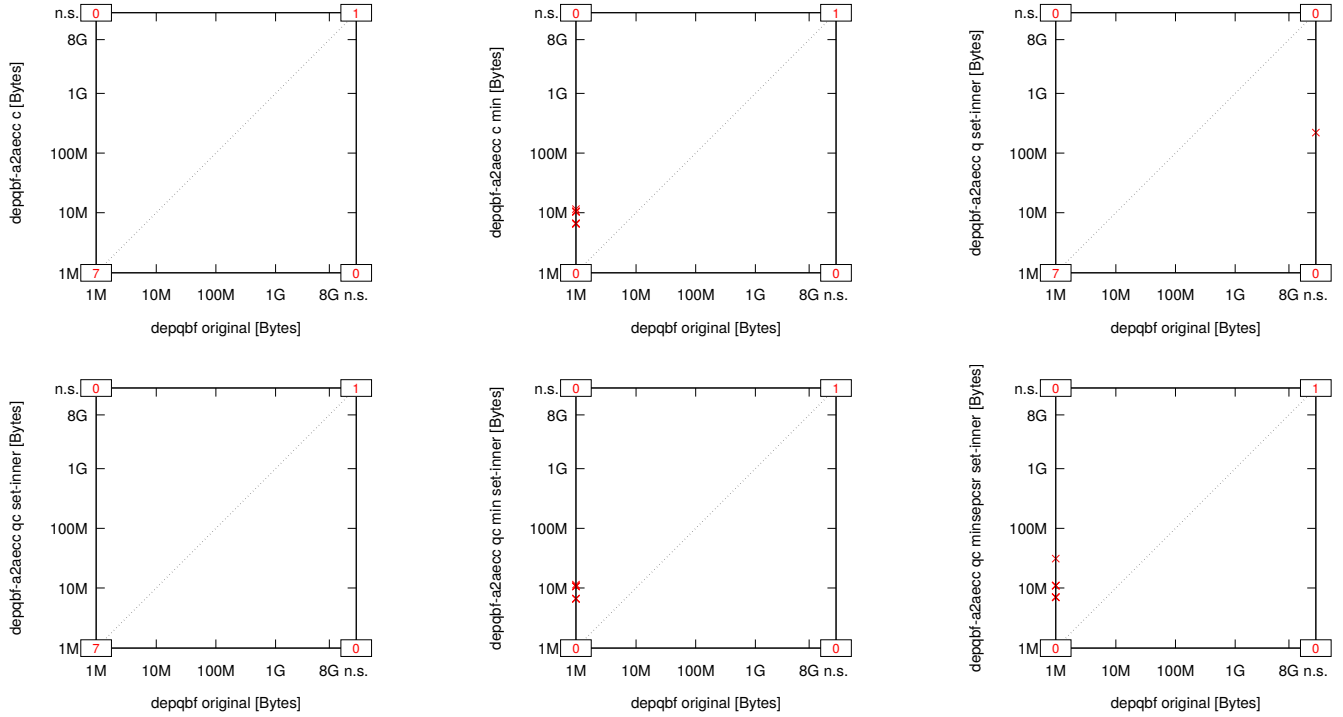


Fig. 1042: Suite qbfeval12 ($n = 8$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

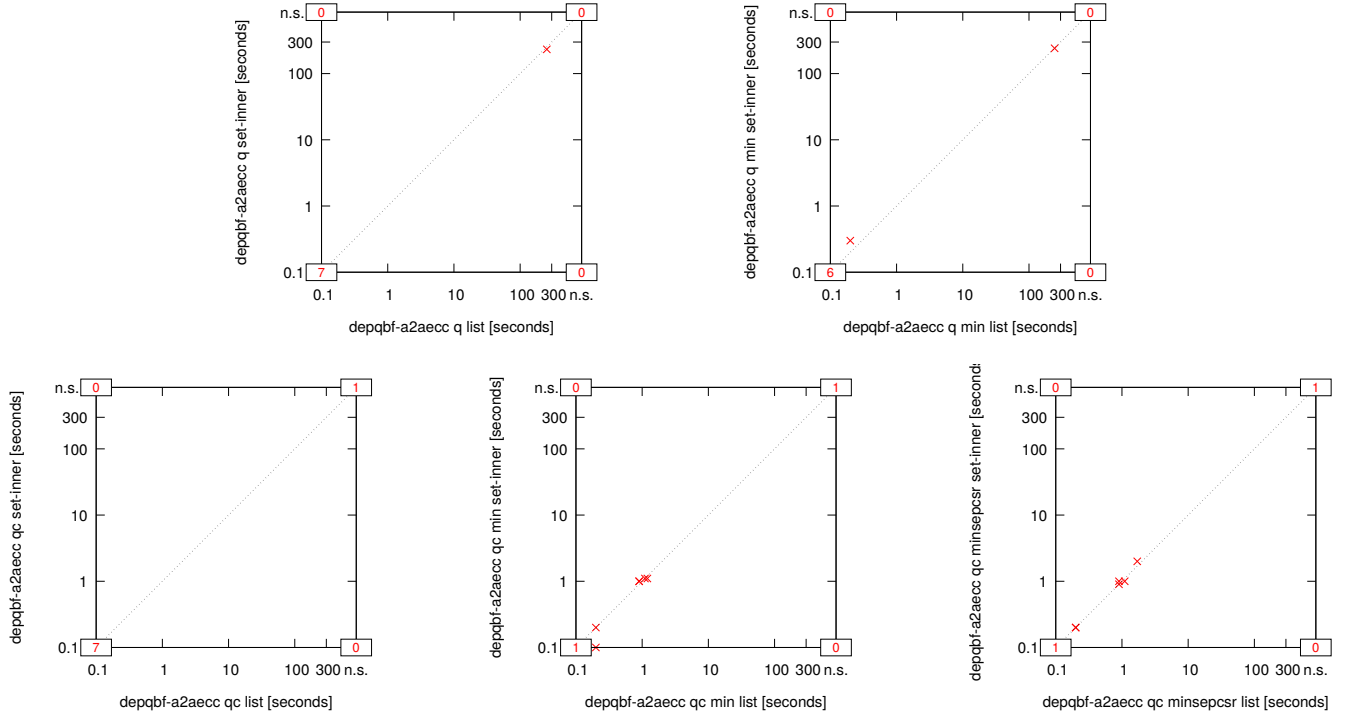


Fig. 1043: Suite qbfeval12 ($n = 8$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

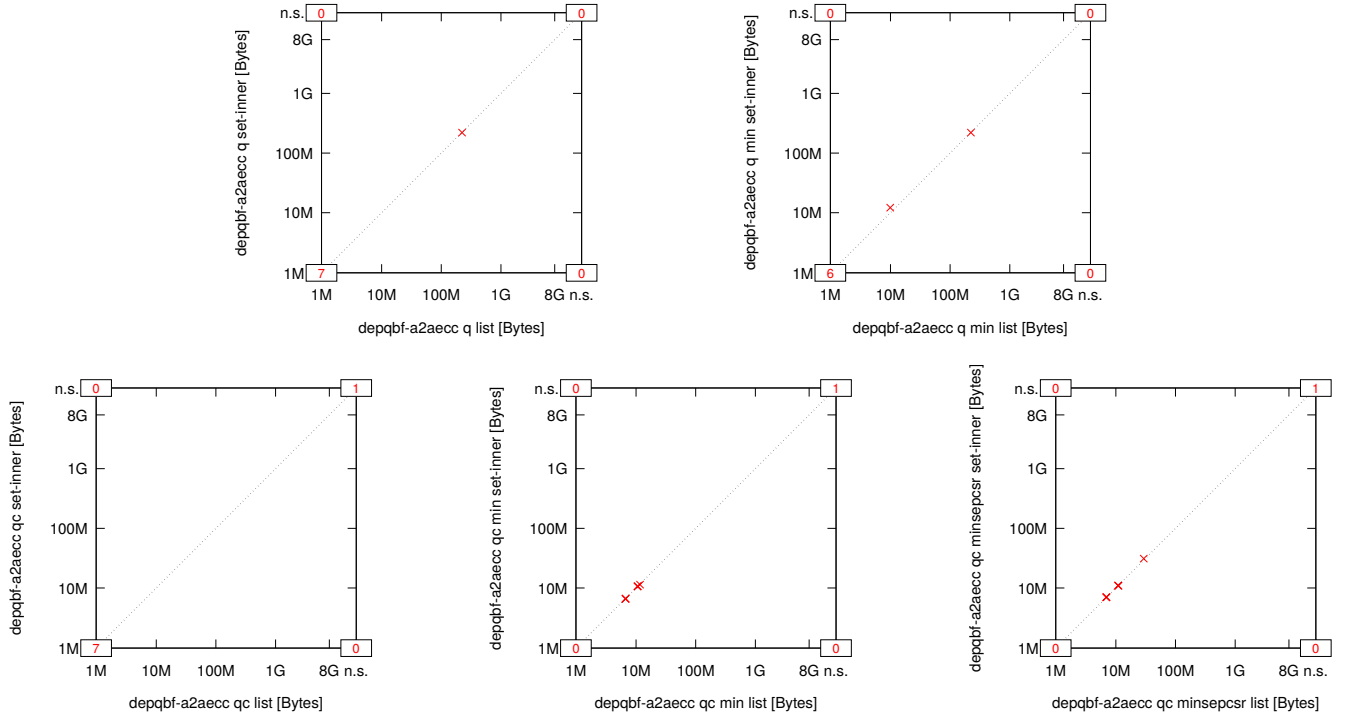


Fig. 1044: Suite qbfeval12 ($n = 8$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

39) Rabe ($n = 3$):

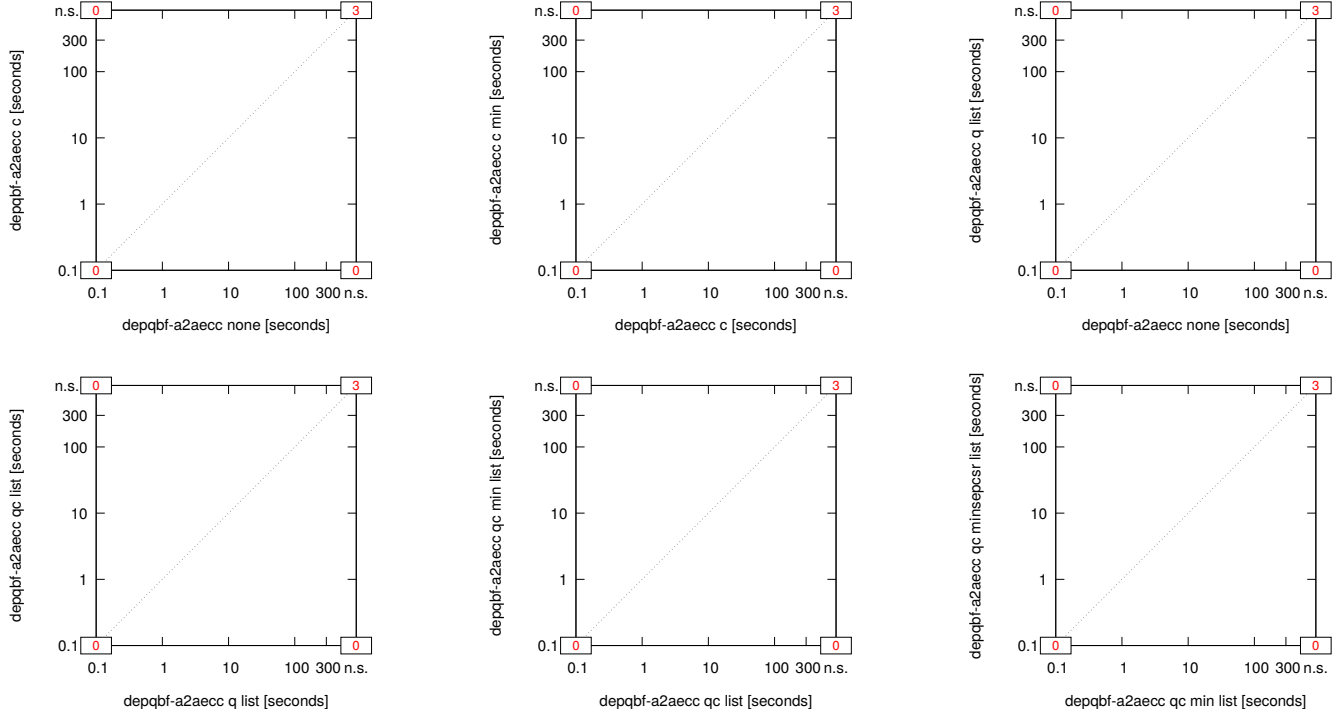


Fig. 1045: Suite Rabe ($n = 3$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

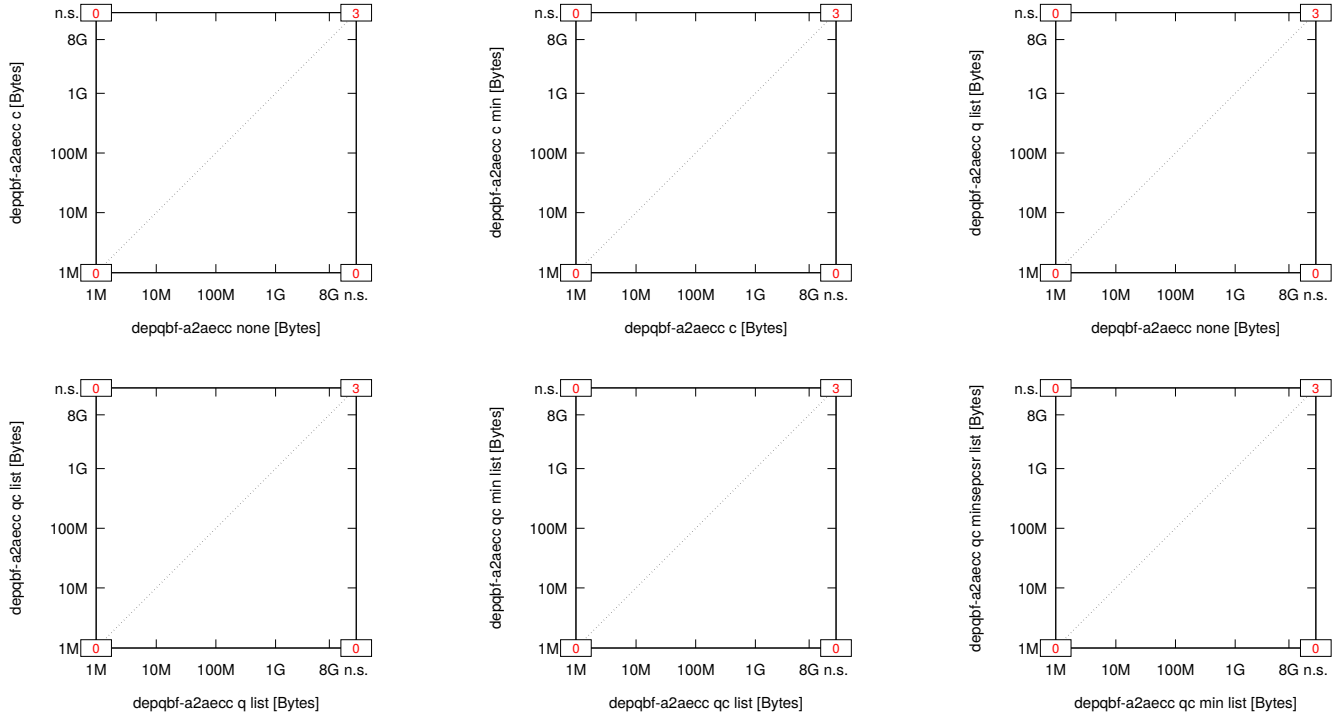


Fig. 1046: Suite Rabe ($n = 3$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

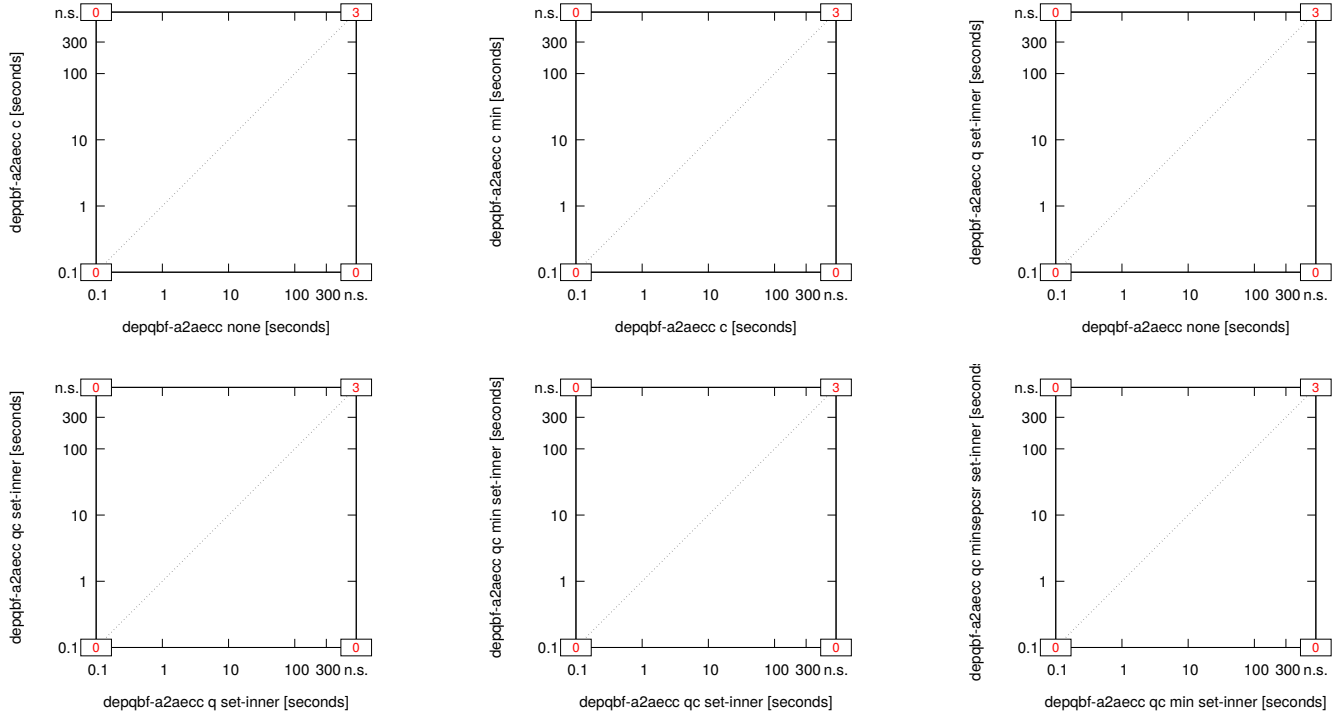


Fig. 1047: Suite Rabe ($n = 3$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

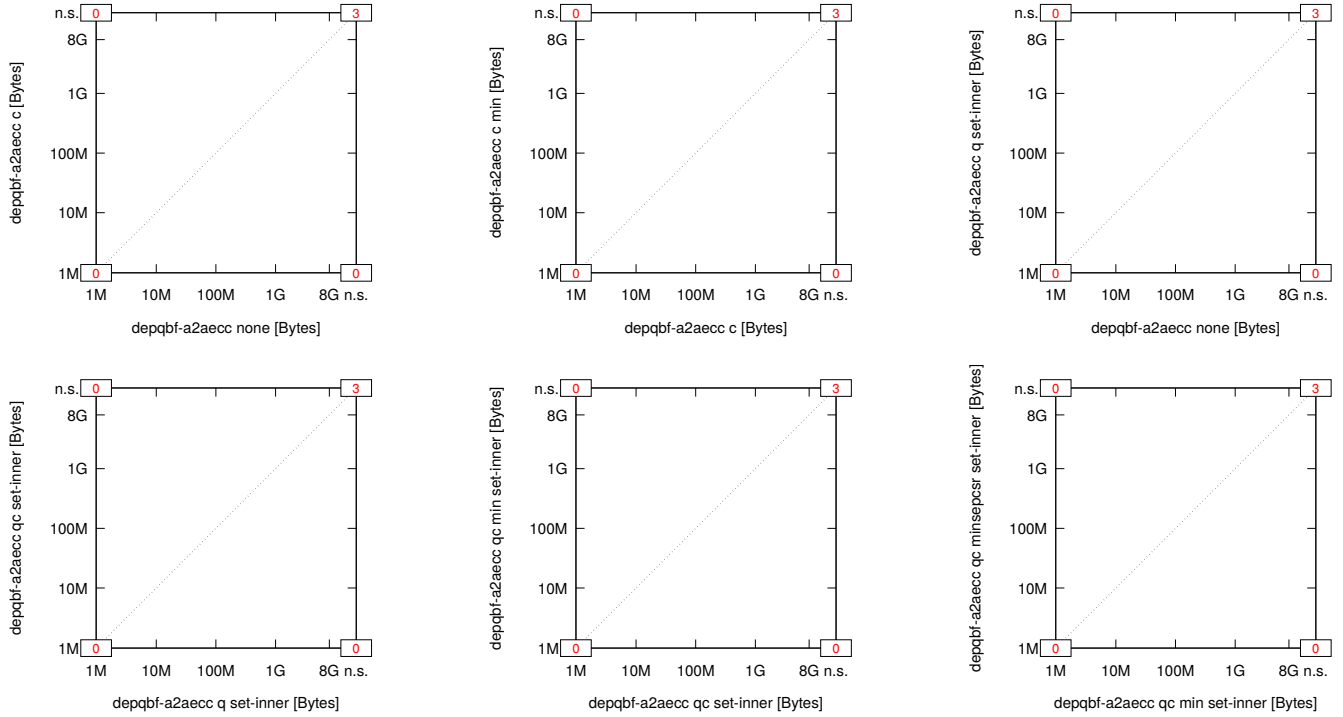


Fig. 1048: Suite Rabe ($n = 3$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

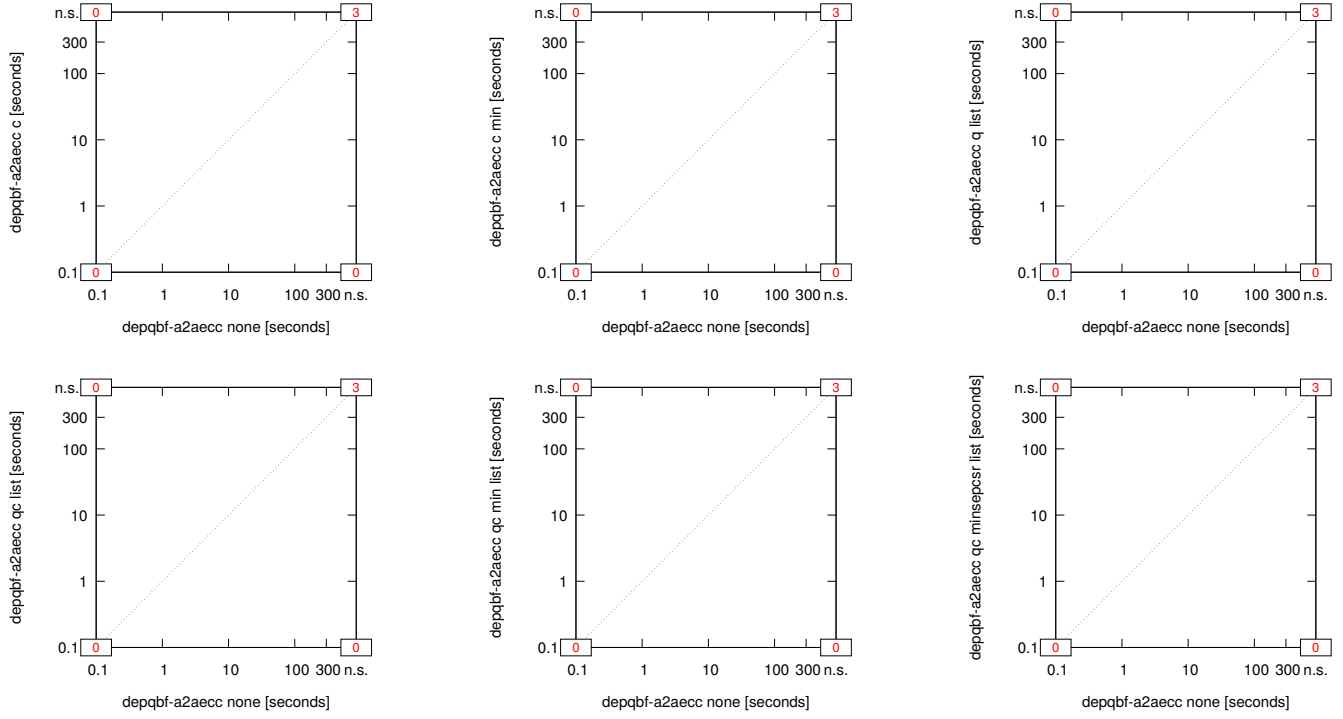


Fig. 1049: Suite Rabe ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

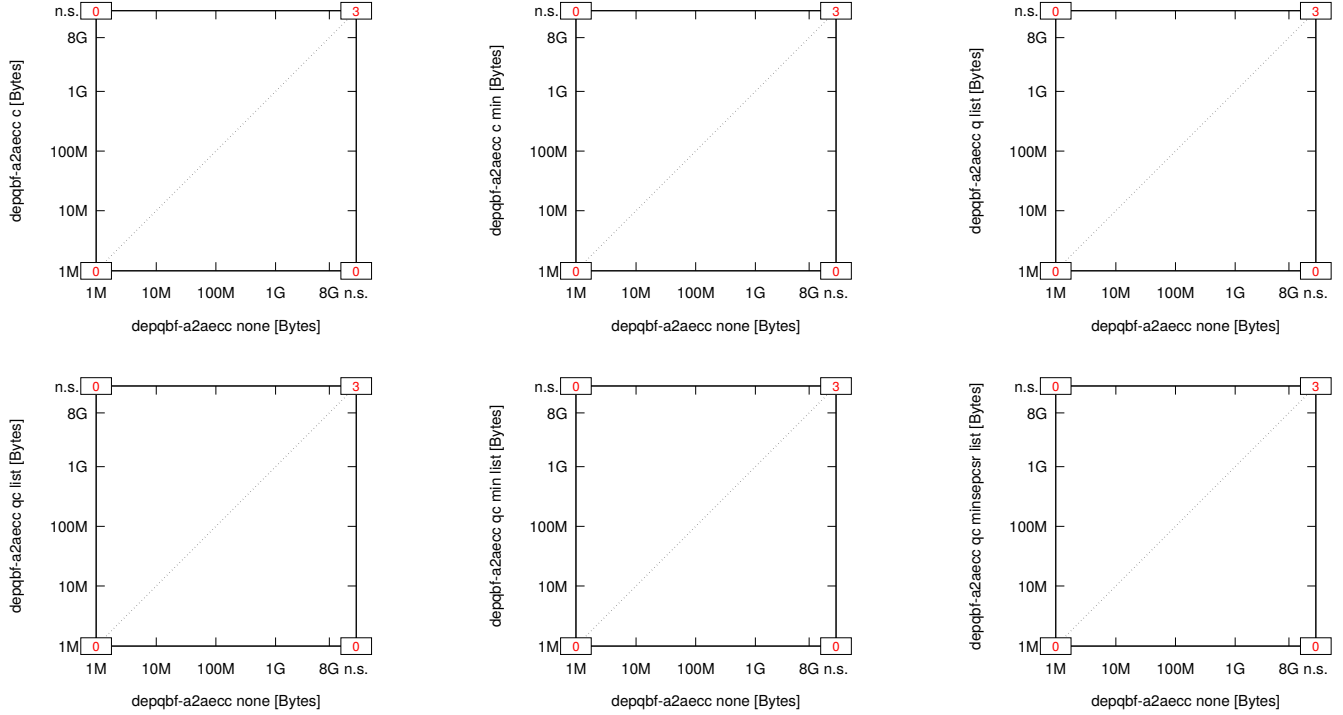


Fig. 1050: Suite Rabe ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

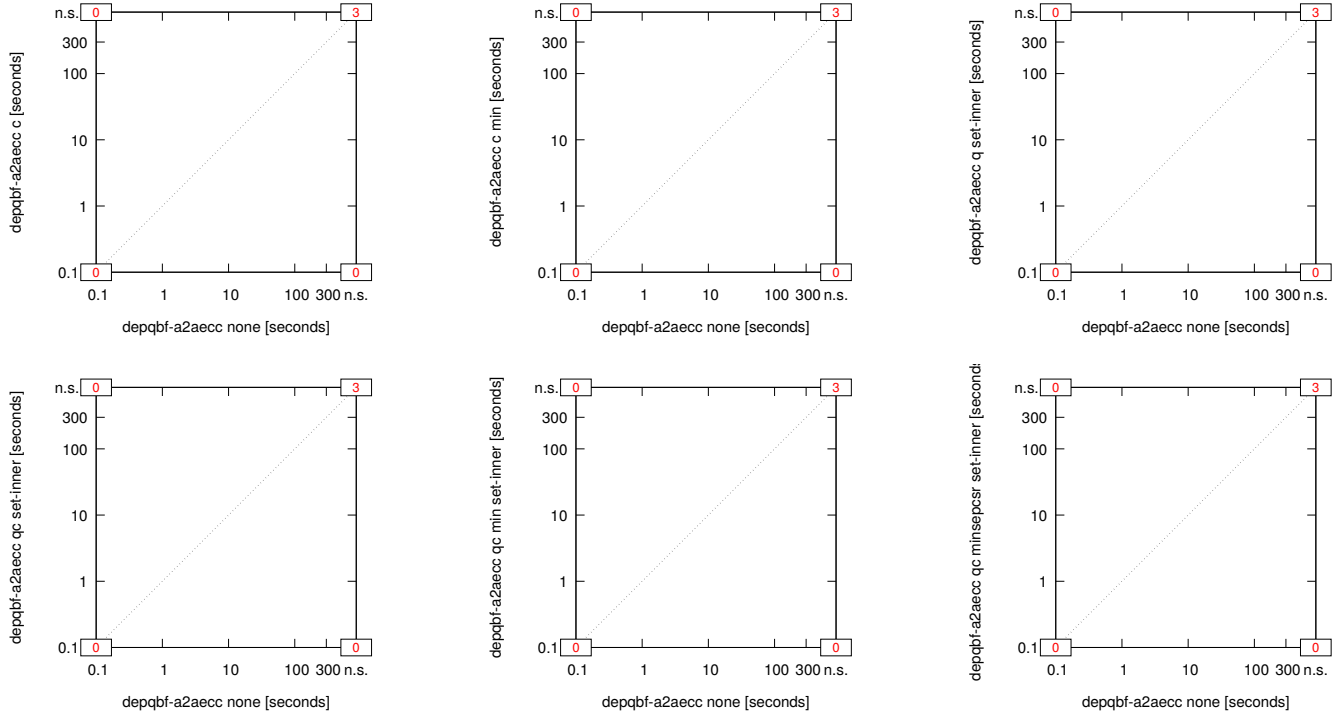


Fig. 1051: Suite Rabe ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

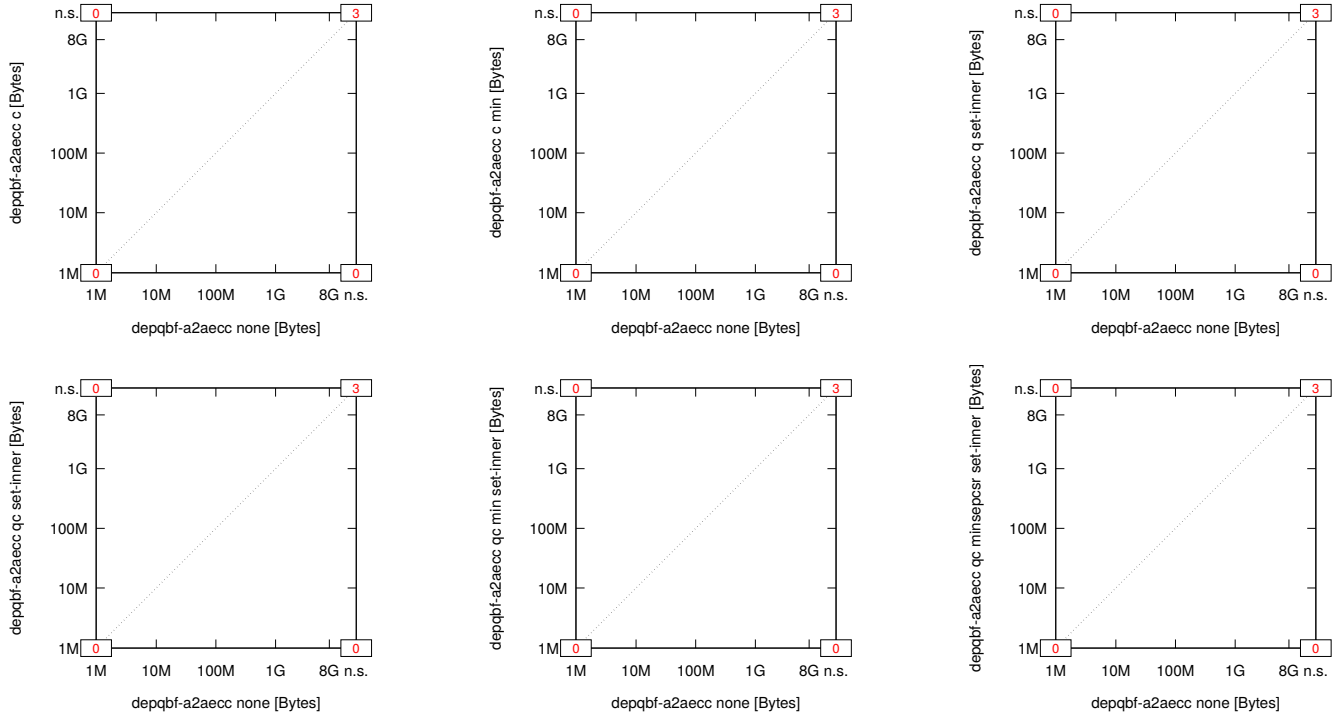


Fig. 1052: Suite Rabe ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

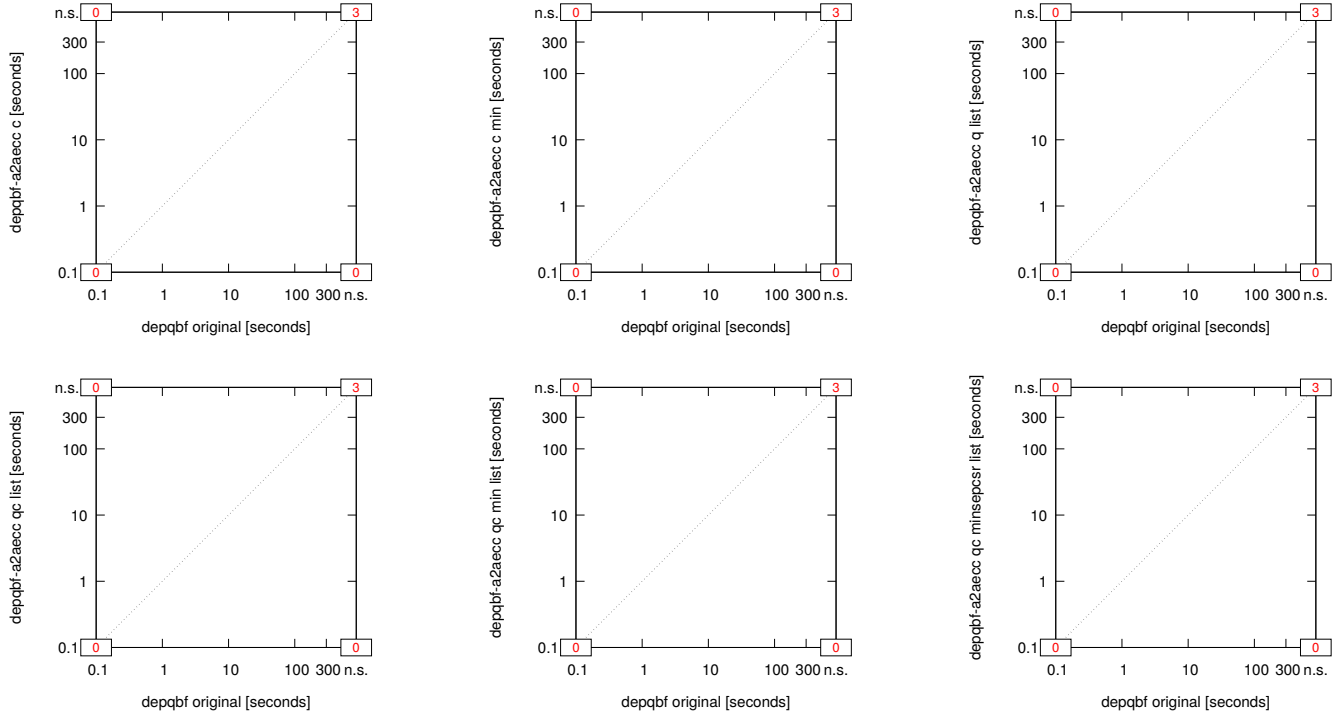


Fig. 1053: Suite Rabe ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

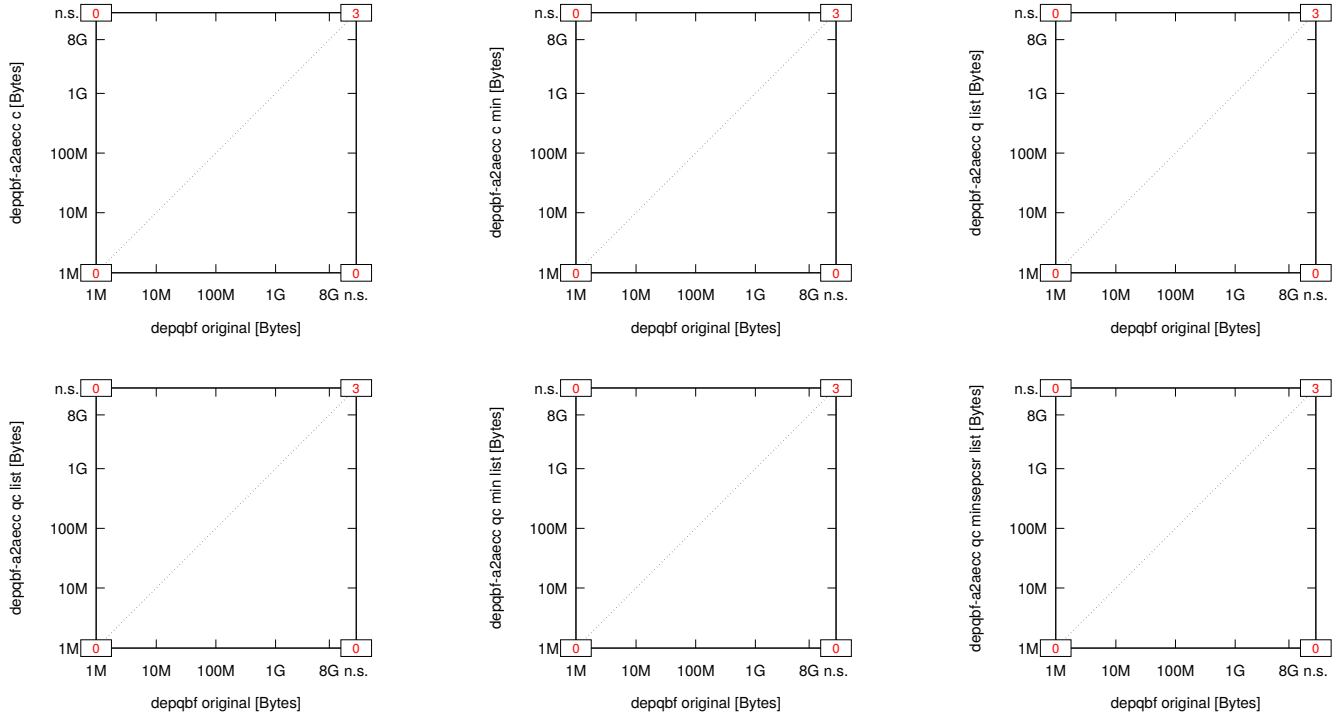


Fig. 1054: Suite Rabe ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

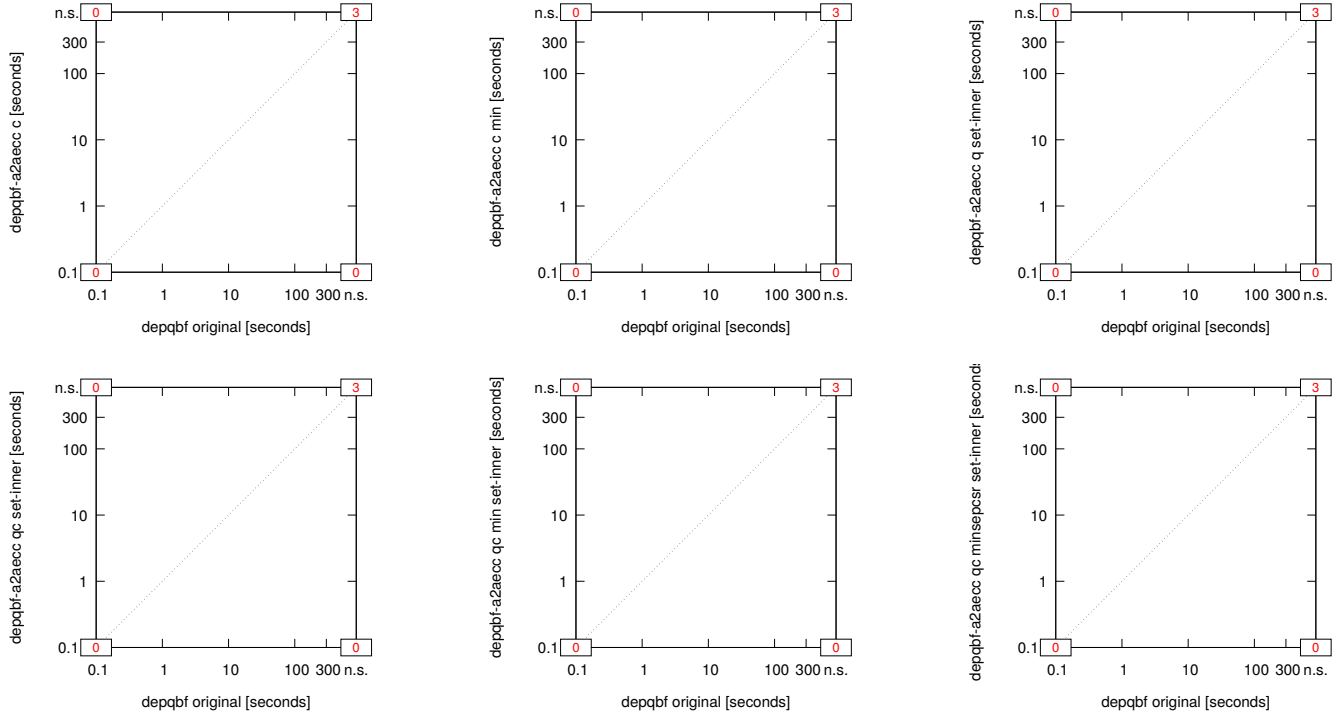


Fig. 1055: Suite Rabe ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

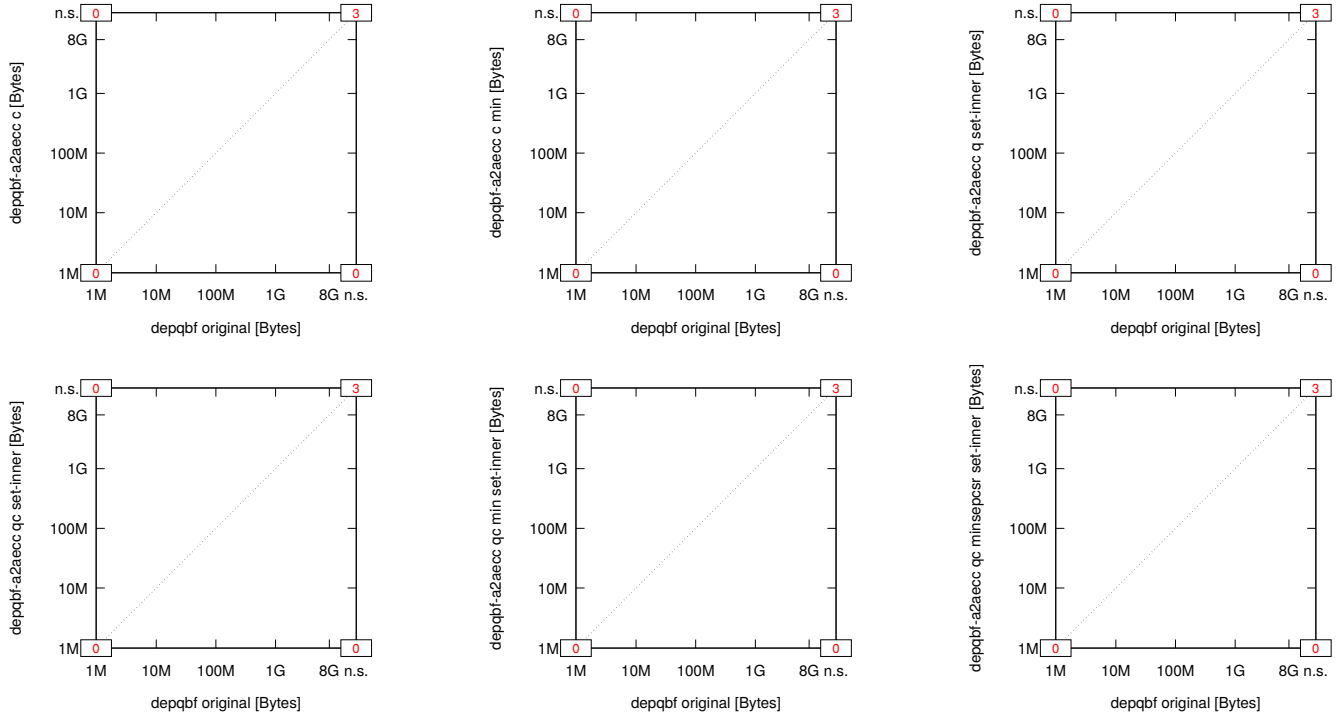


Fig. 1056: Suite Rabe ($n = 3$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

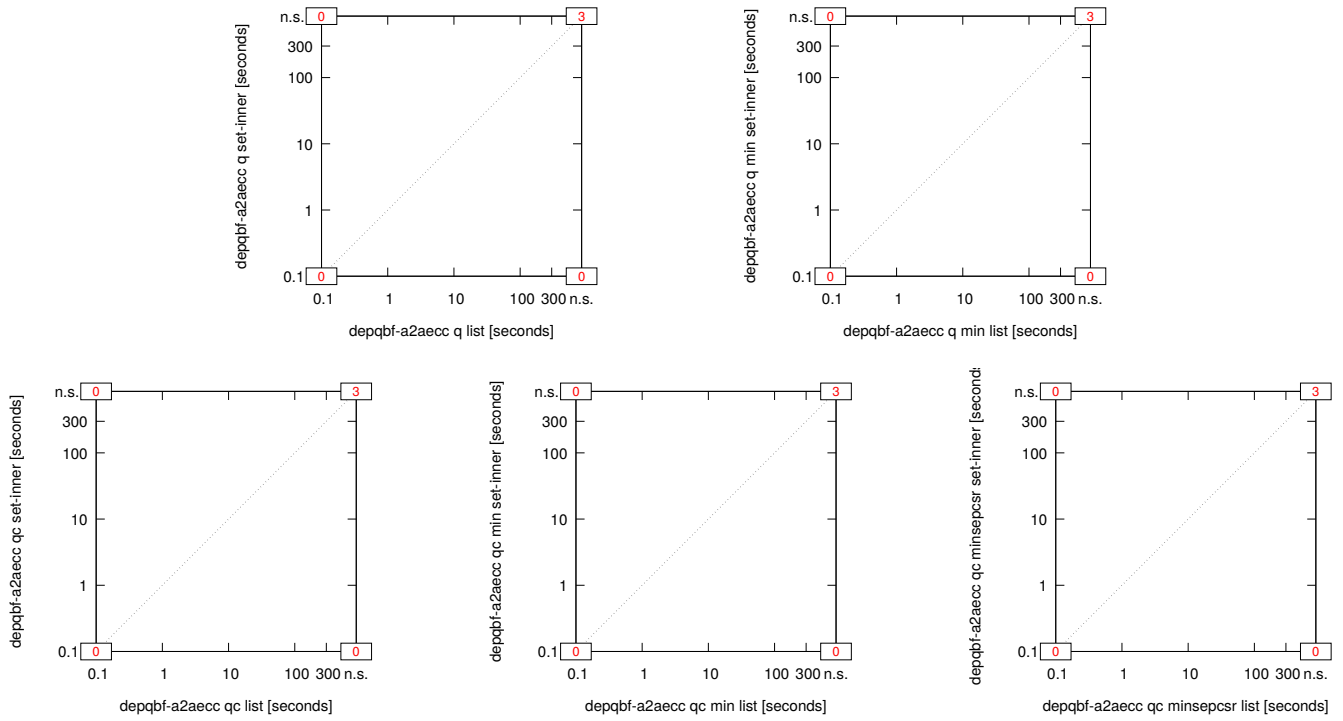


Fig. 1057: Suite Rabe ($n = 3$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

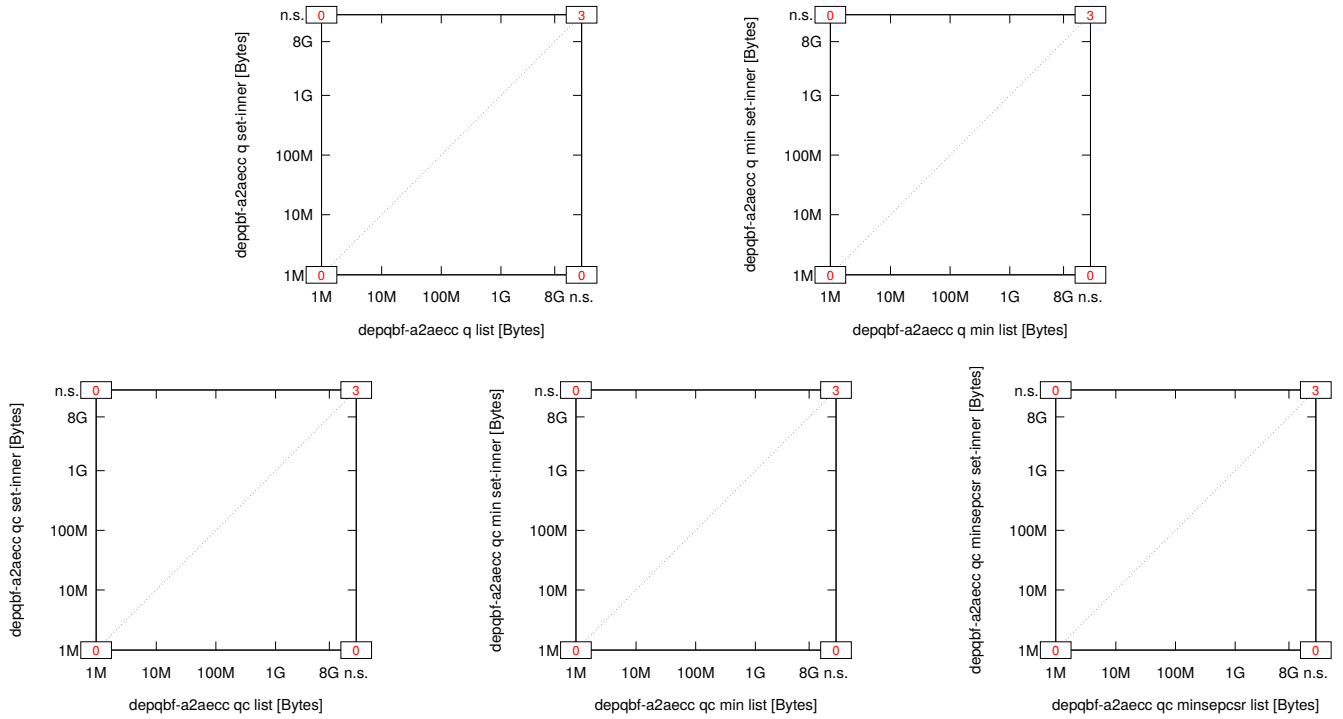


Fig. 1058: Suite Rabe ($n = 3$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

40) Rintanen ($n = 55$):

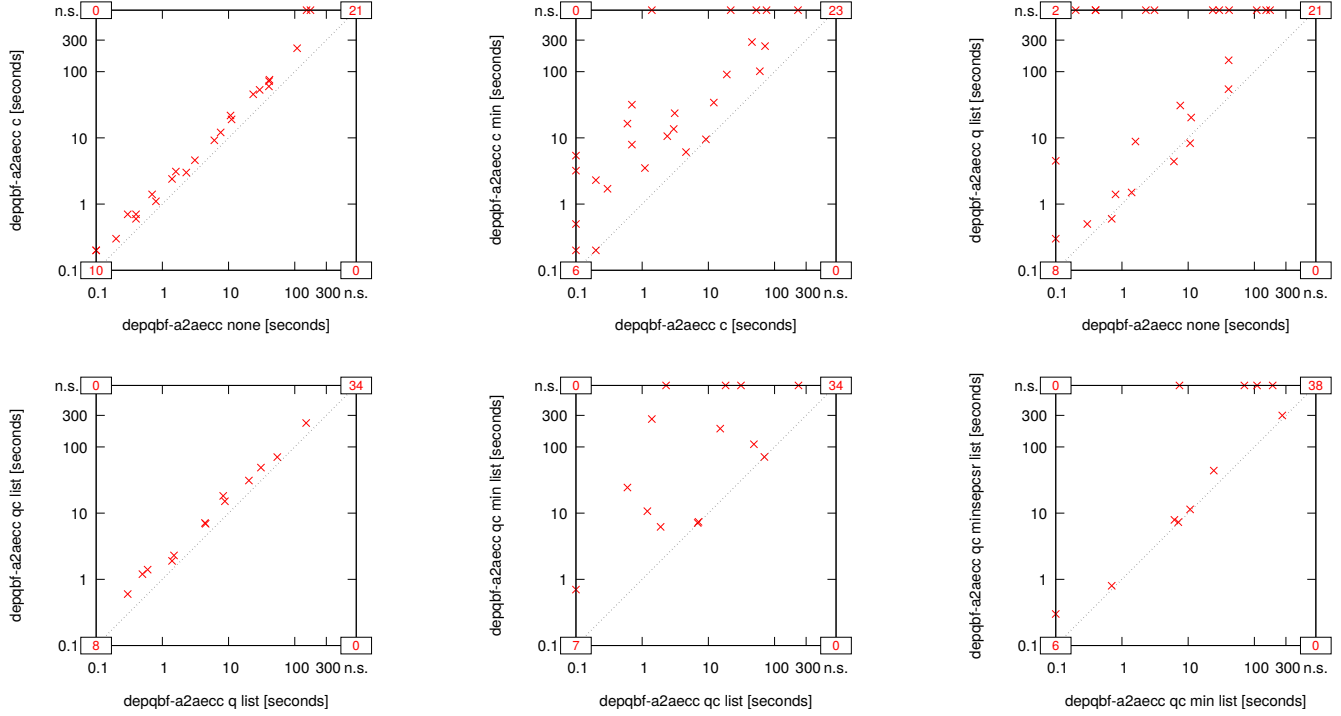


Fig. 1059: Suite Rintanen ($n = 55$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

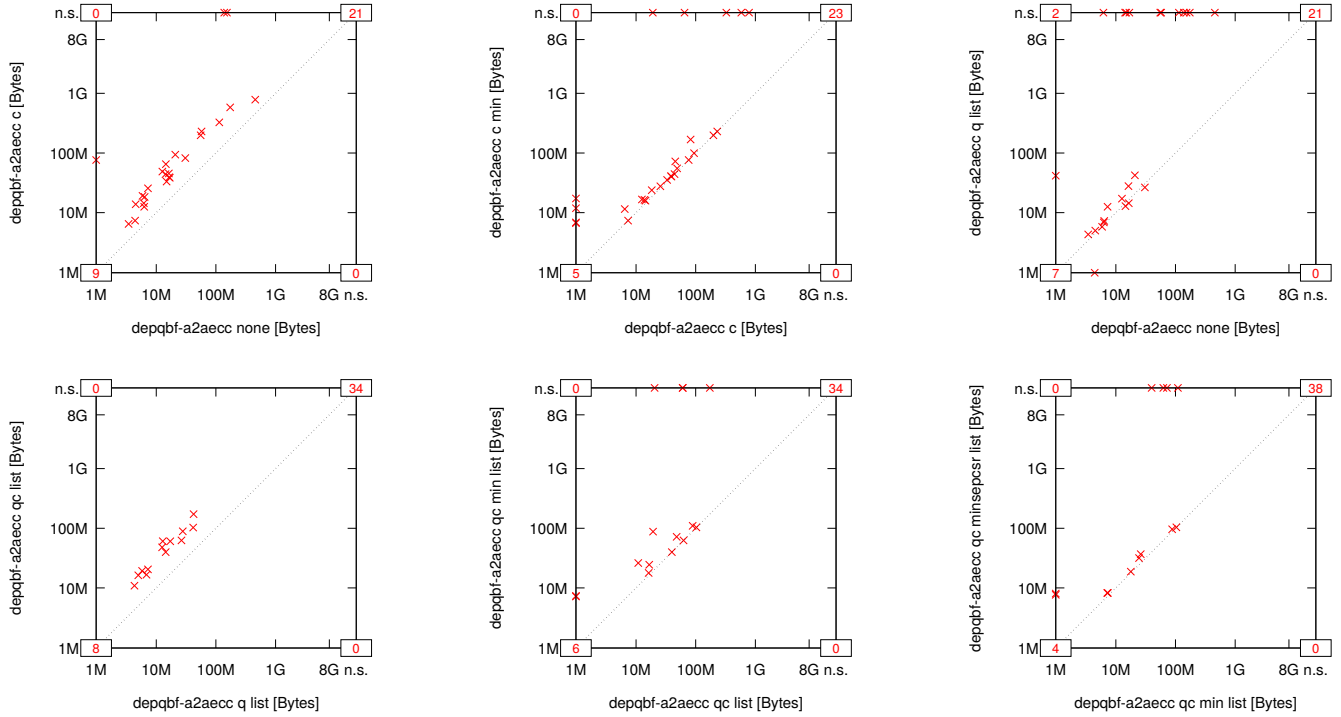


Fig. 1060: Suite Rintanen ($n = 55$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

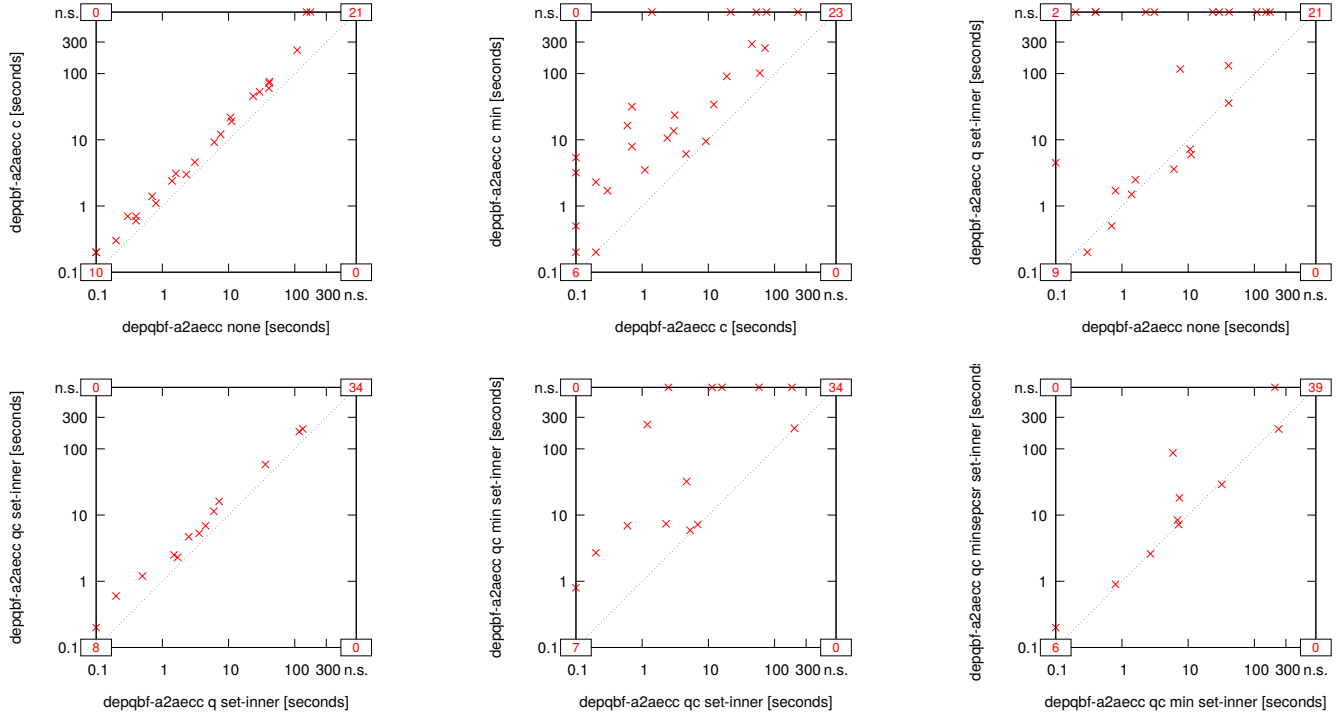


Fig. 1061: Suite Rintanen ($n = 55$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

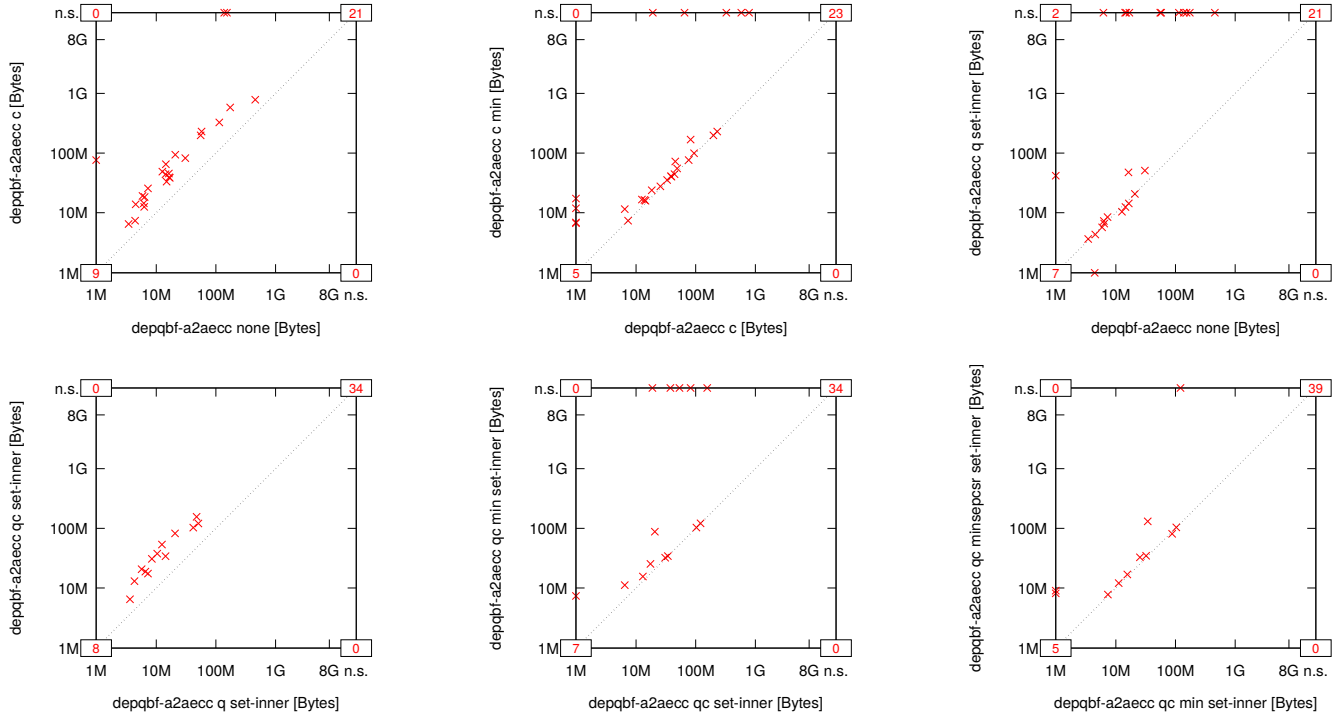


Fig. 1062: Suite Rintanen ($n = 55$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

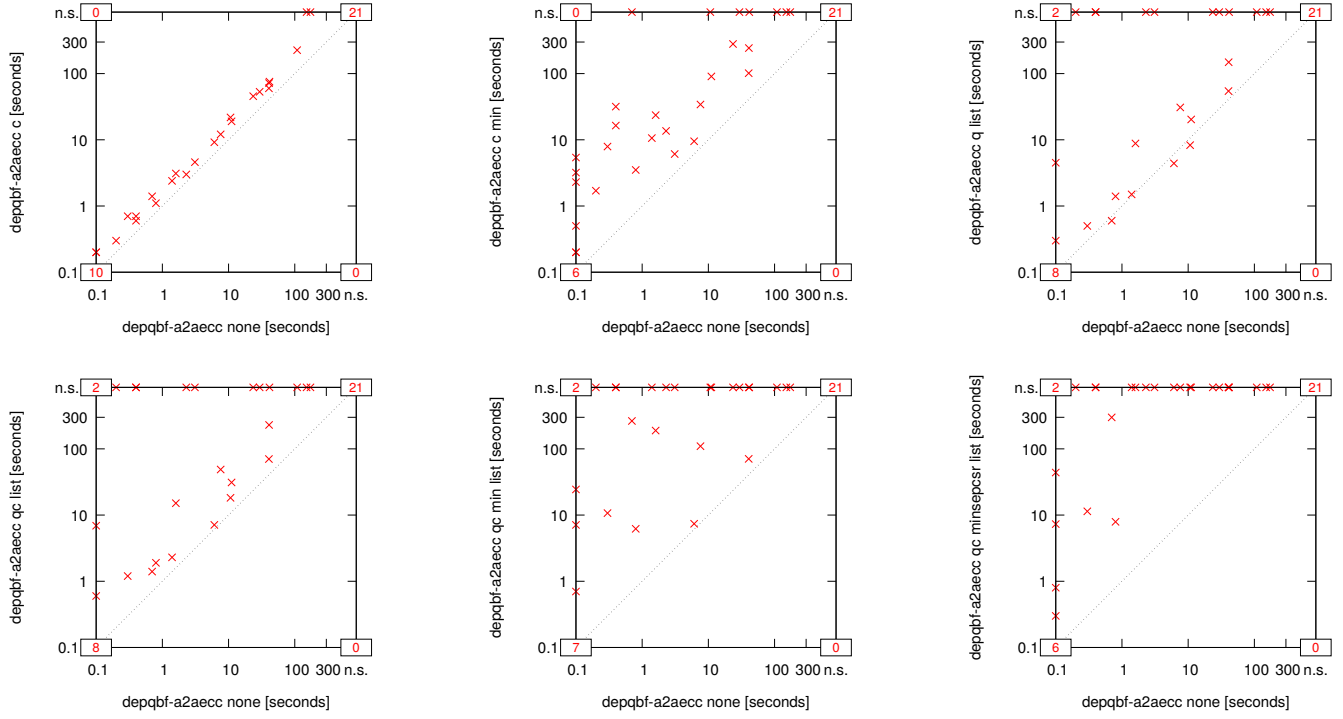


Fig. 1063: Suite Rintanen ($n = 55$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

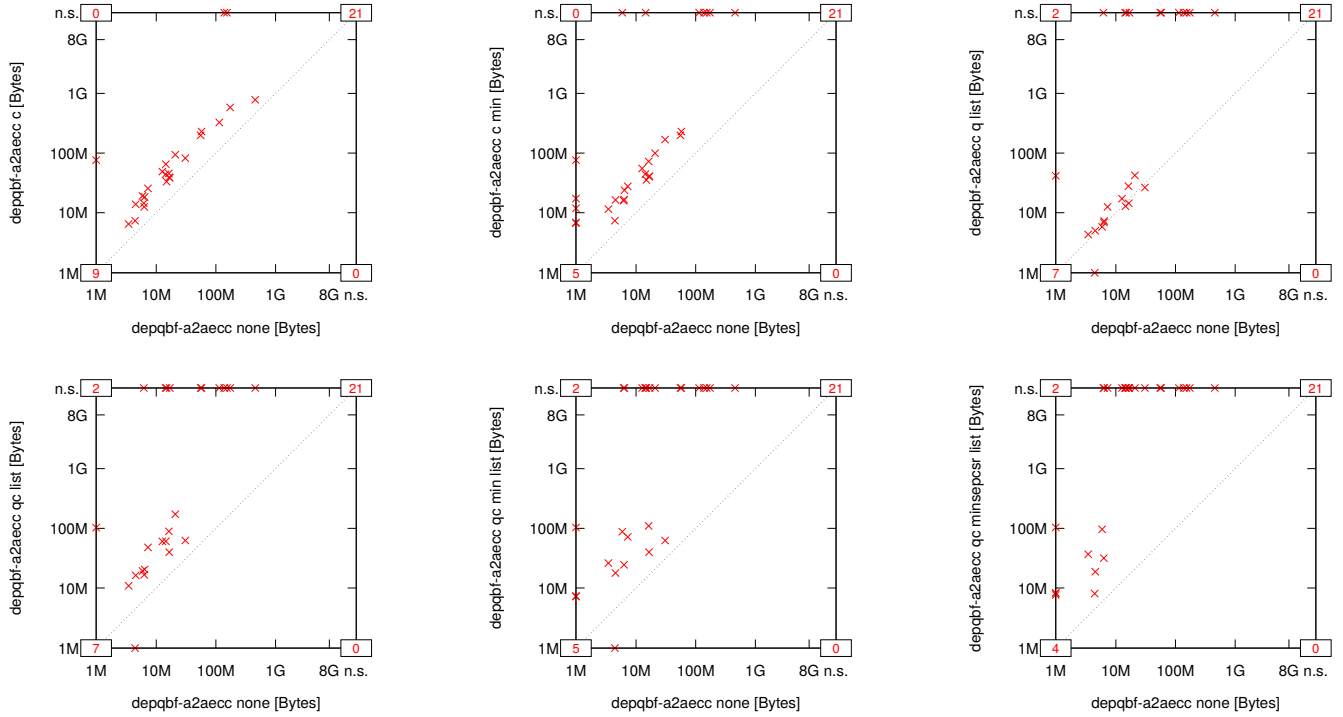


Fig. 1064: Suite Rintanen ($n = 55$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

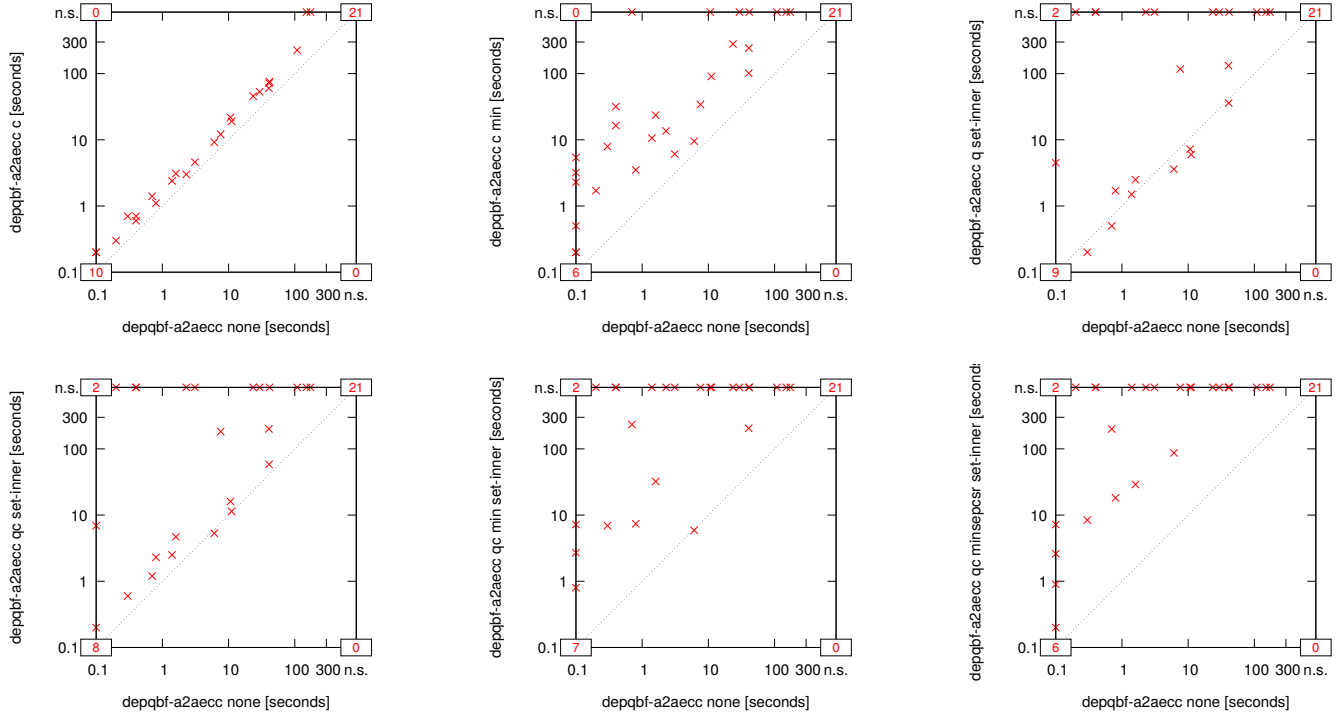


Fig. 1065: Suite Rintanen ($n = 55$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

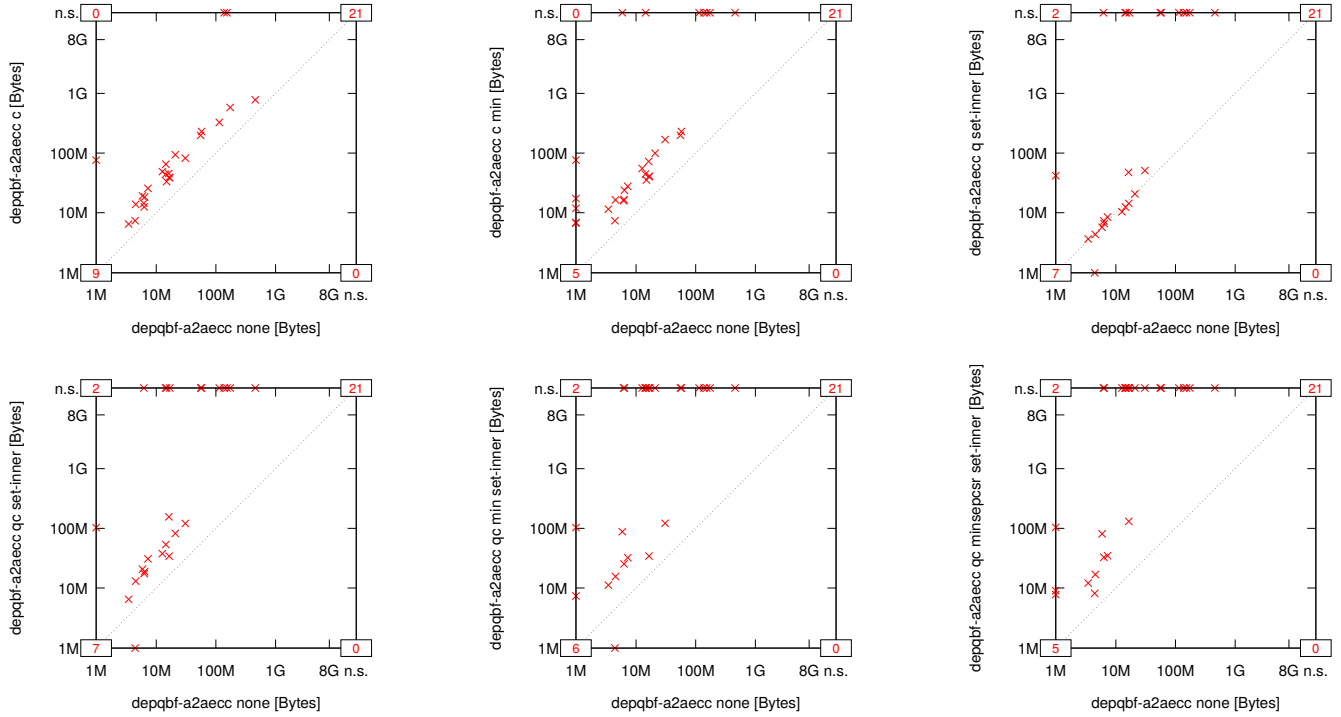


Fig. 1066: Suite Rintanen ($n = 55$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

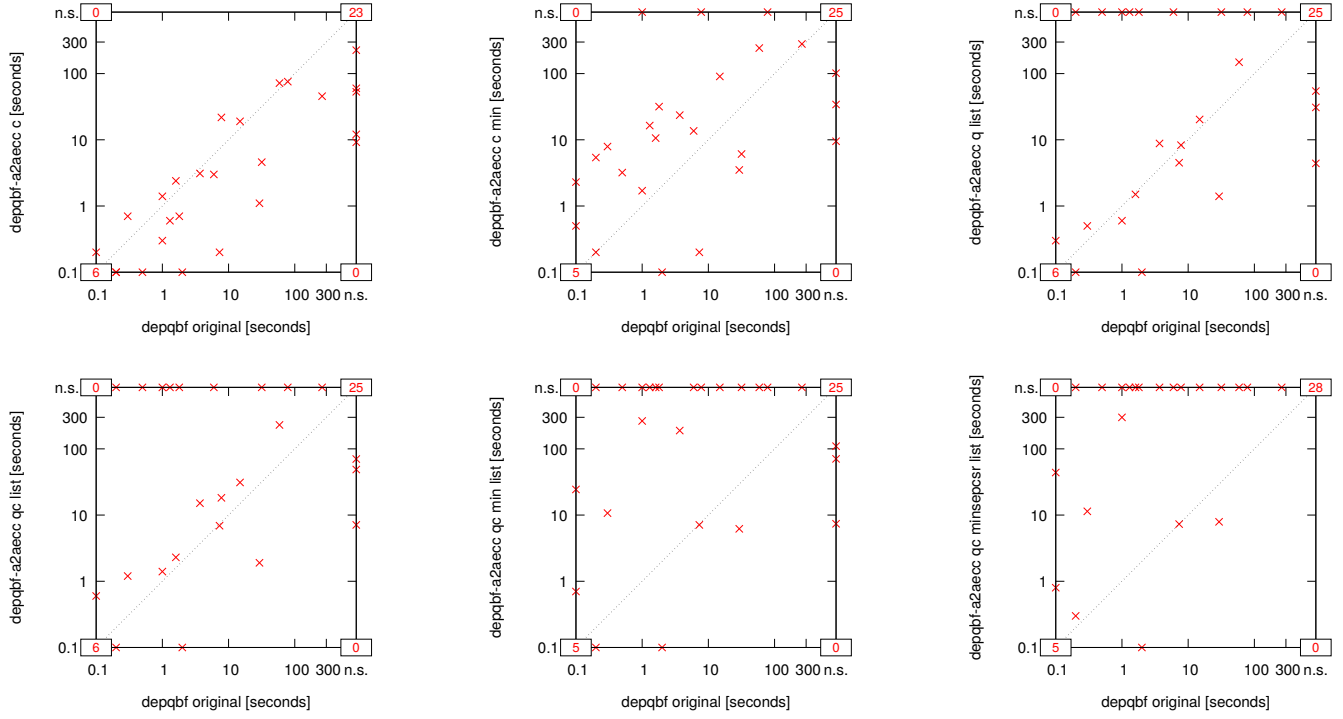


Fig. 1067: Suite Rintanen ($n = 55$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

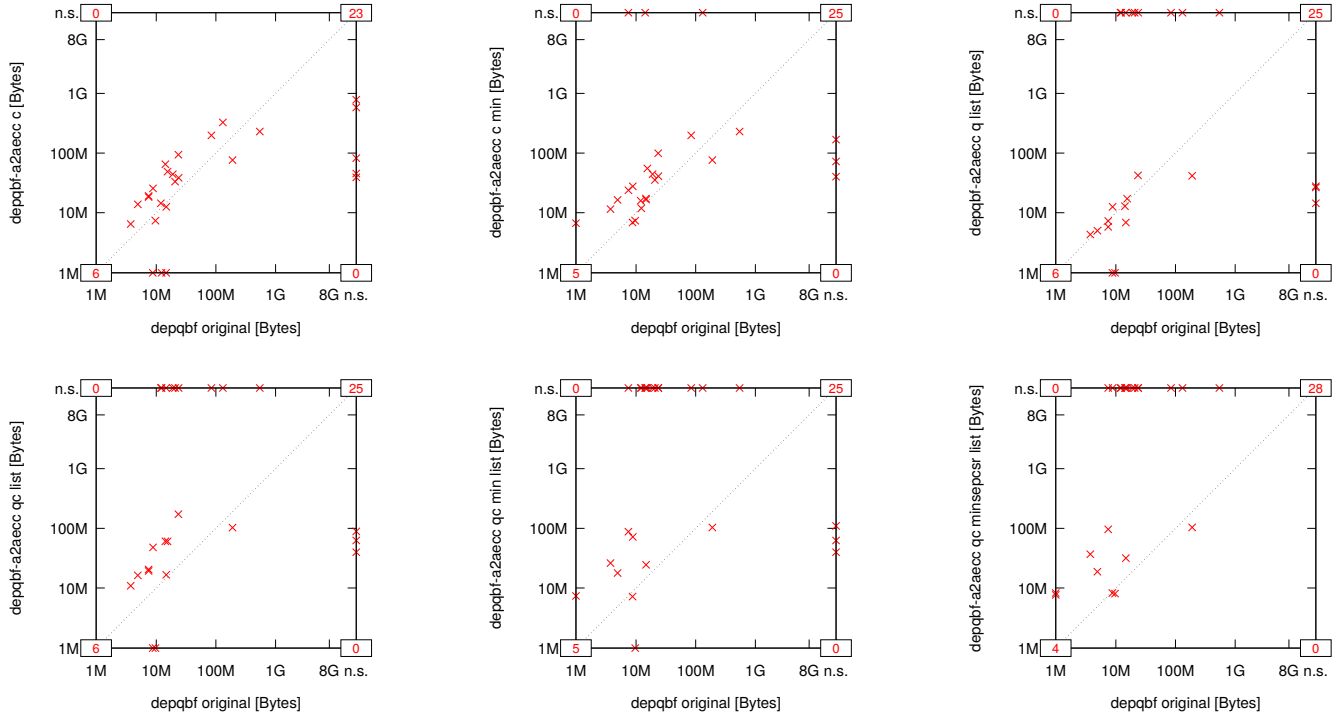


Fig. 1068: Suite Rintanen ($n = 55$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

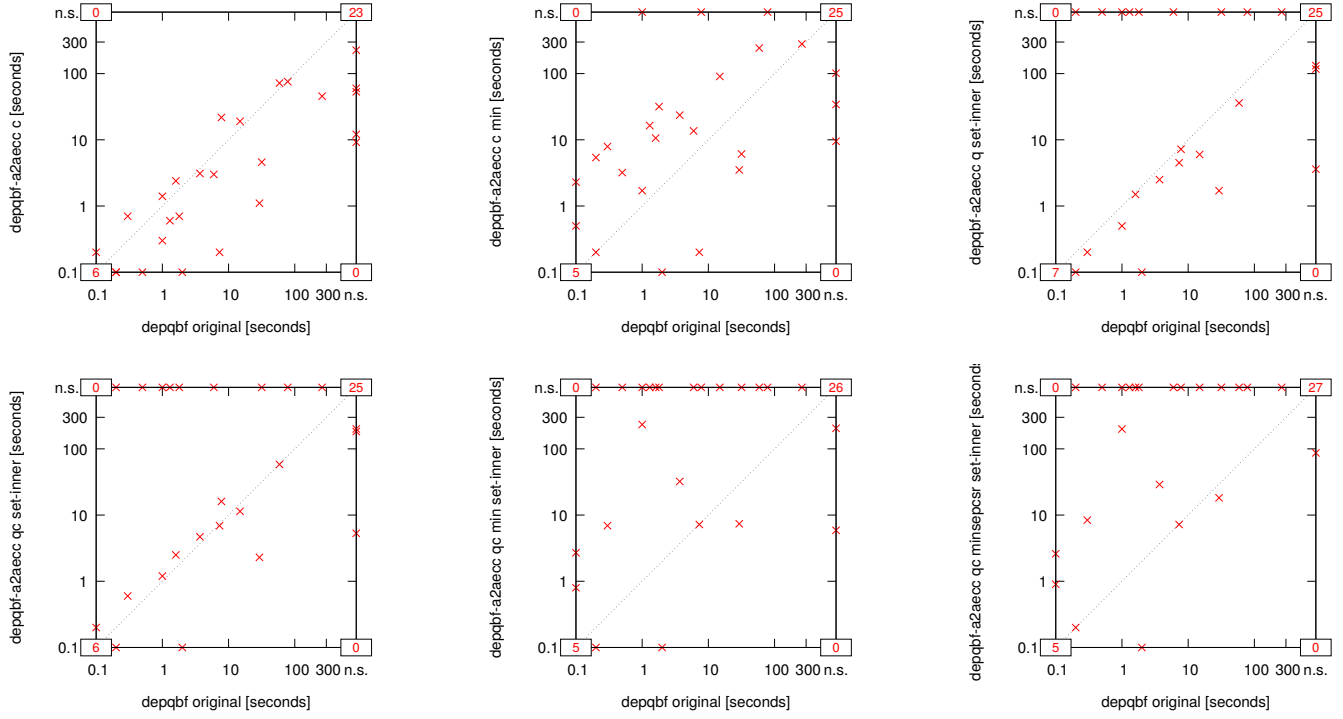


Fig. 1069: Suite Rintanen ($n = 55$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

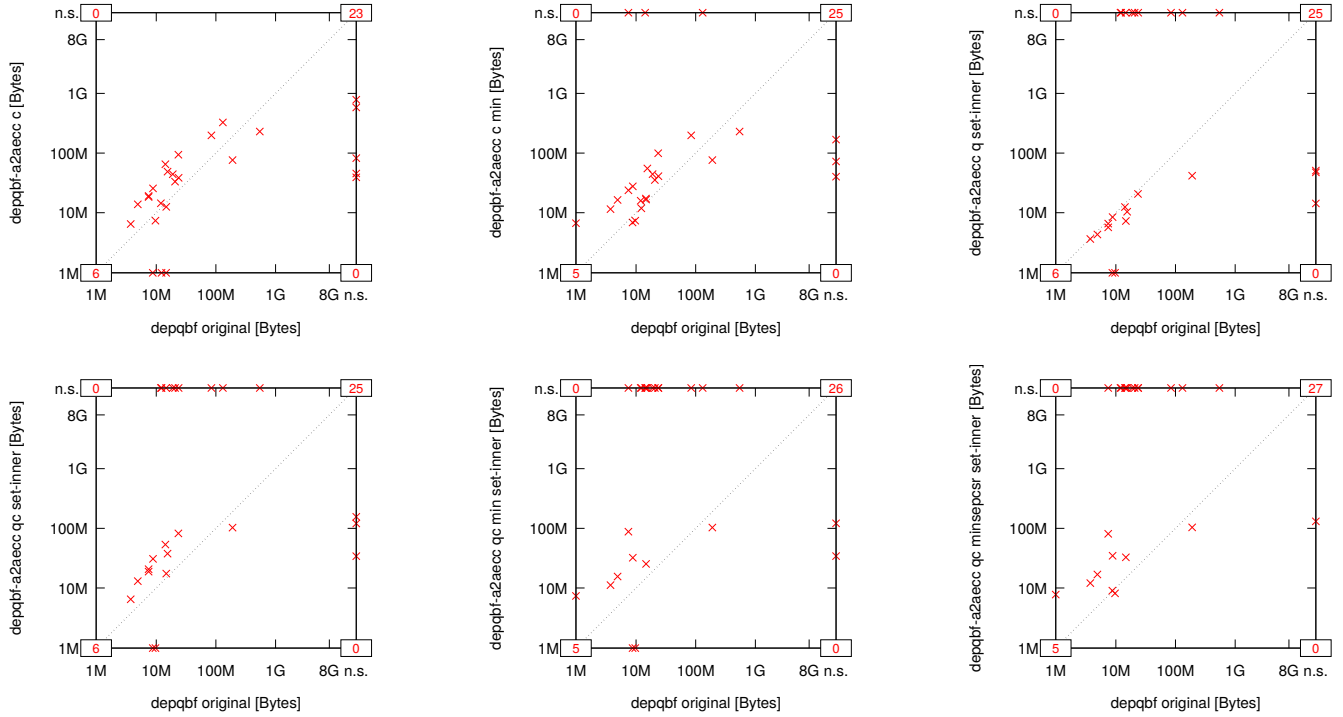


Fig. 1070: Suite Rintanen ($n = 55$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

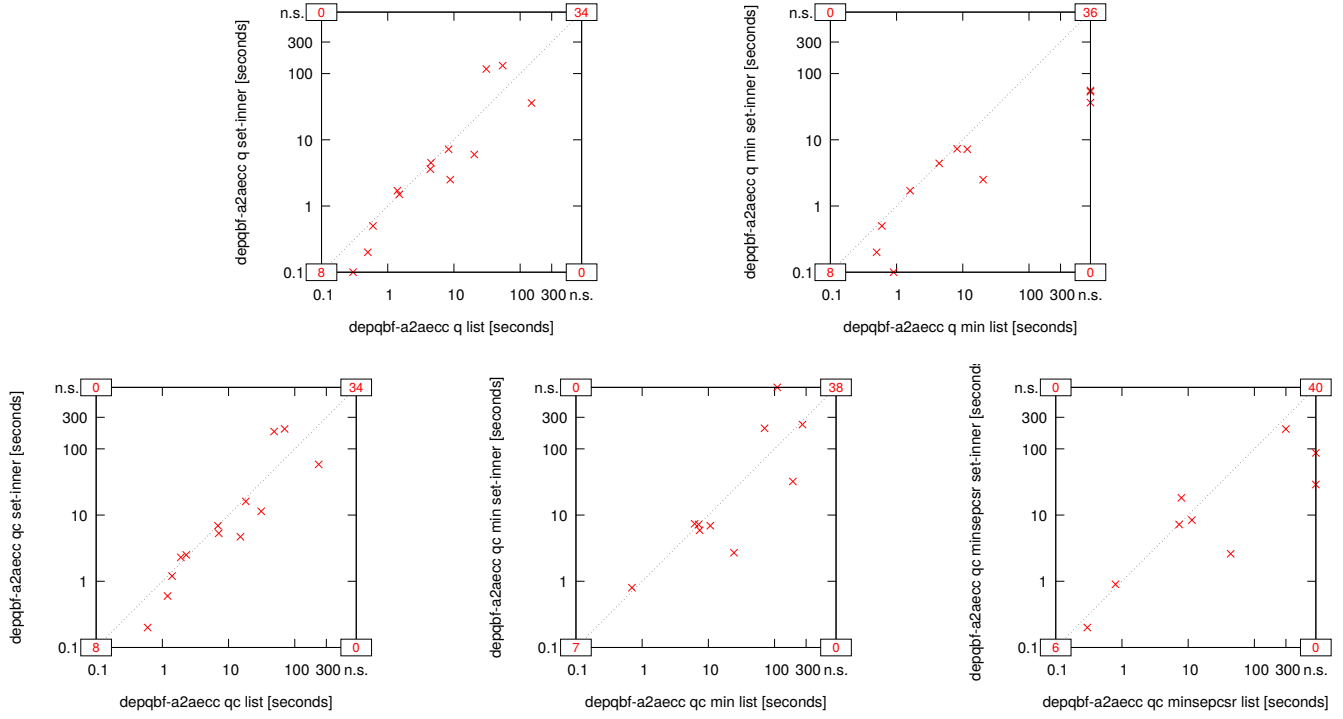


Fig. 1071: Suite Rintanen ($n = 55$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

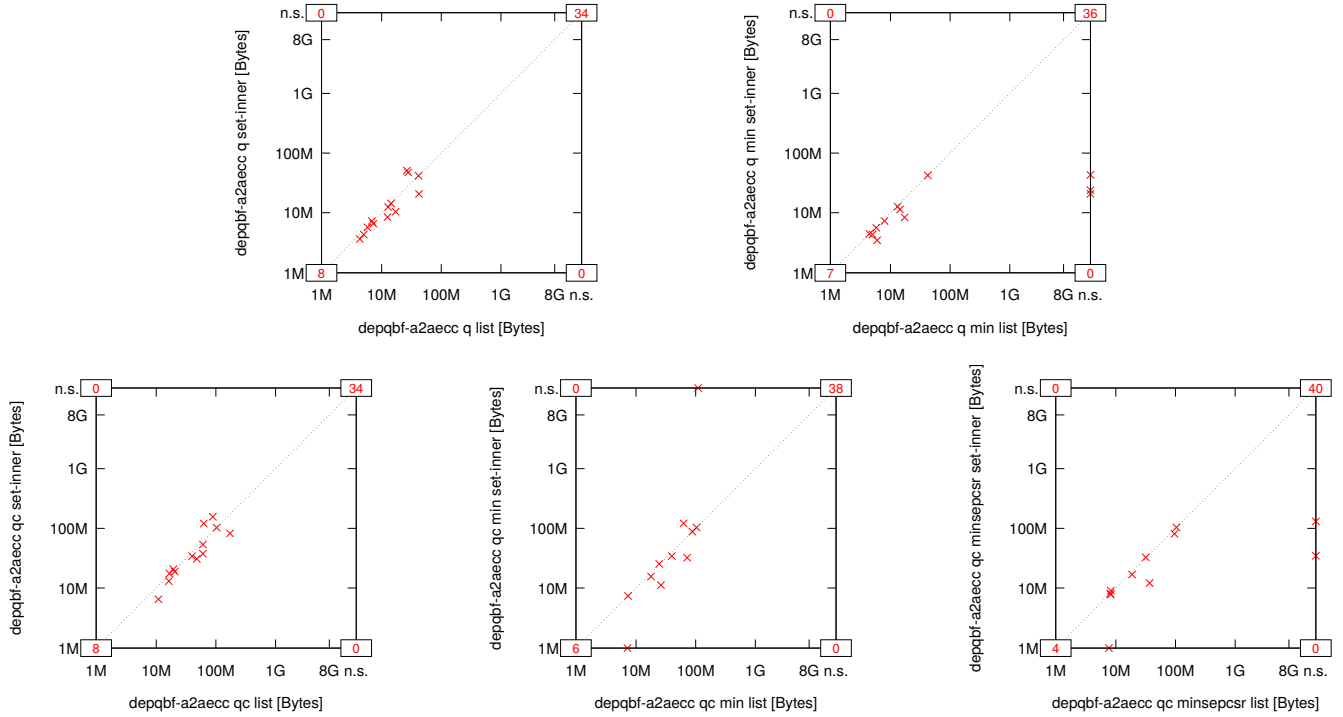


Fig. 1072: Suite Rintanen ($n = 55$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

41) Sauer-Reimer ($n = 42$):

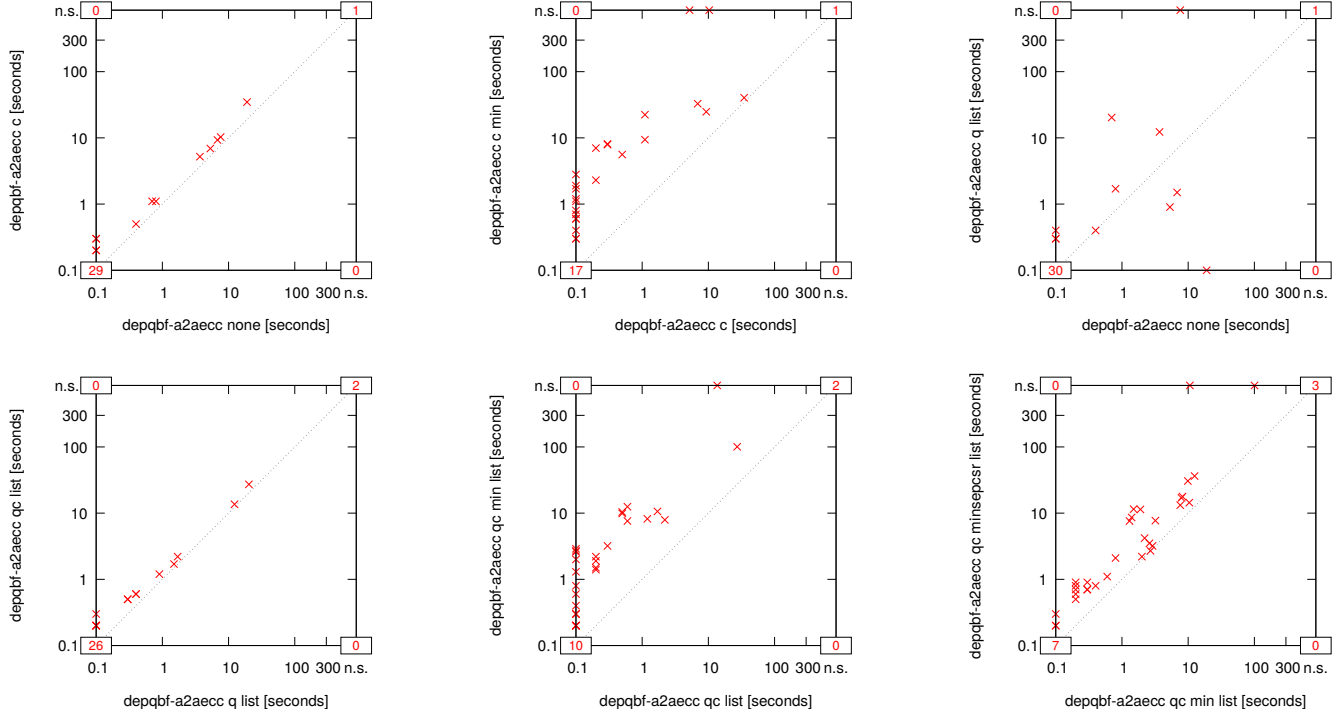


Fig. 1073: Suite Sauer-Reimer ($n = 42$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

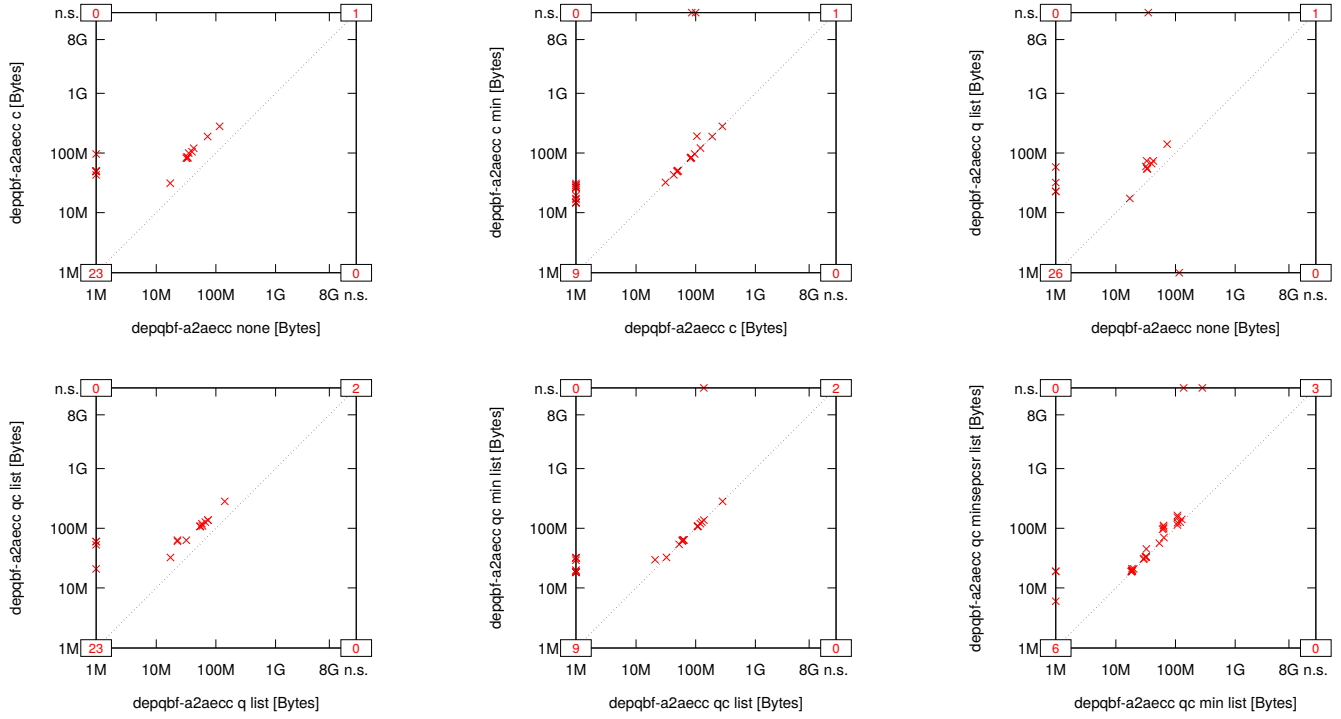


Fig. 1074: Suite Sauer-Reimer ($n = 42$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

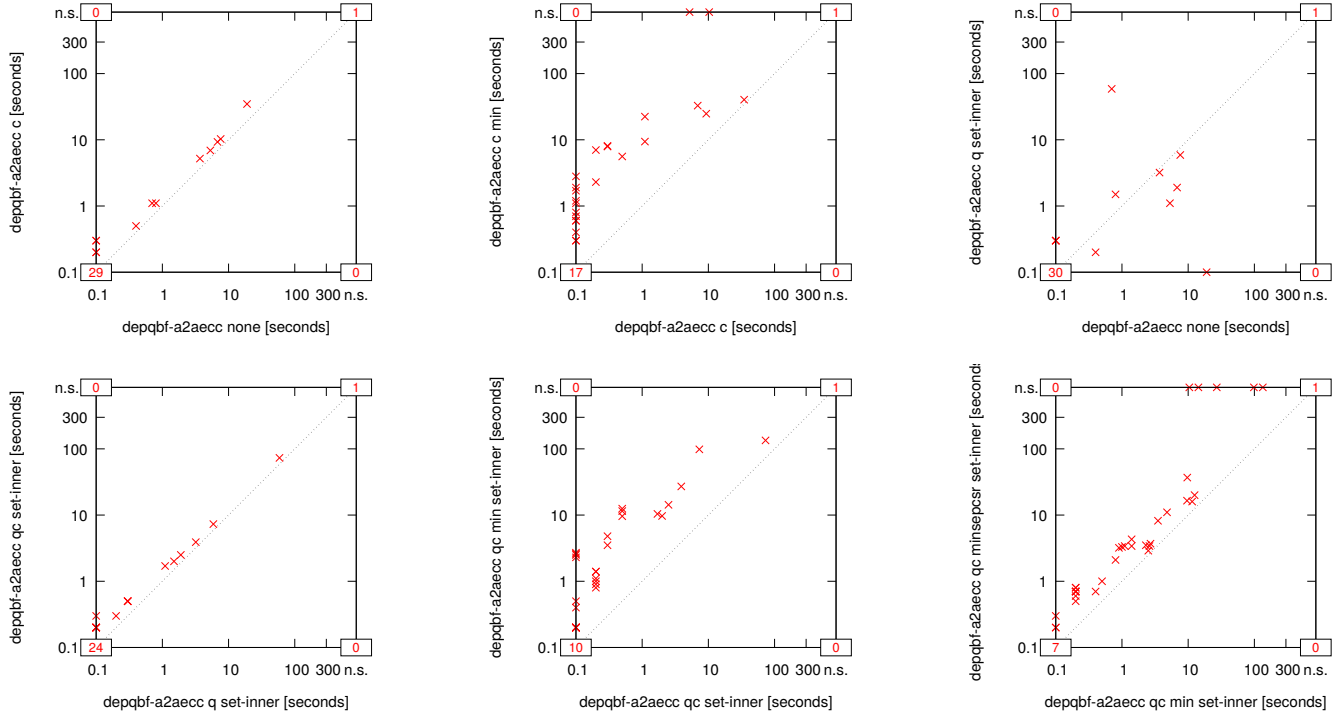


Fig. 1075: Suite Sauer-Reimer ($n = 42$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

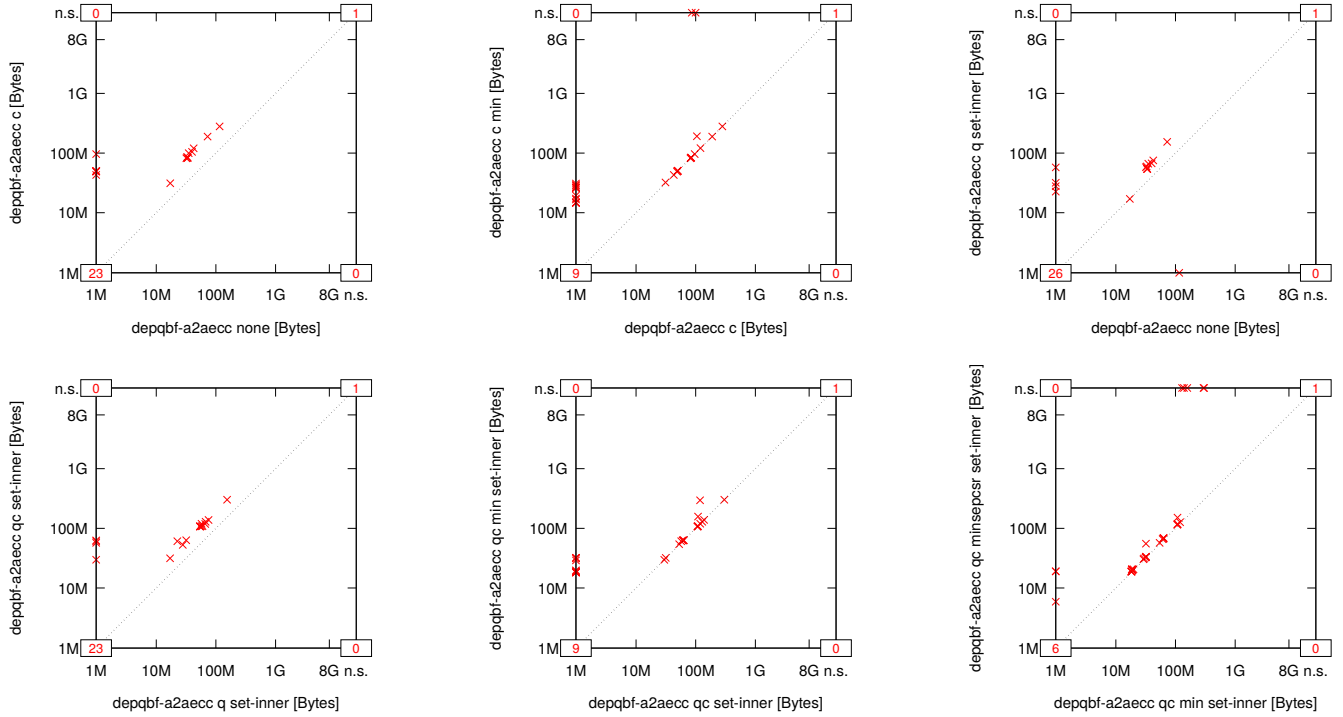


Fig. 1076: Suite Sauer-Reimer ($n = 42$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

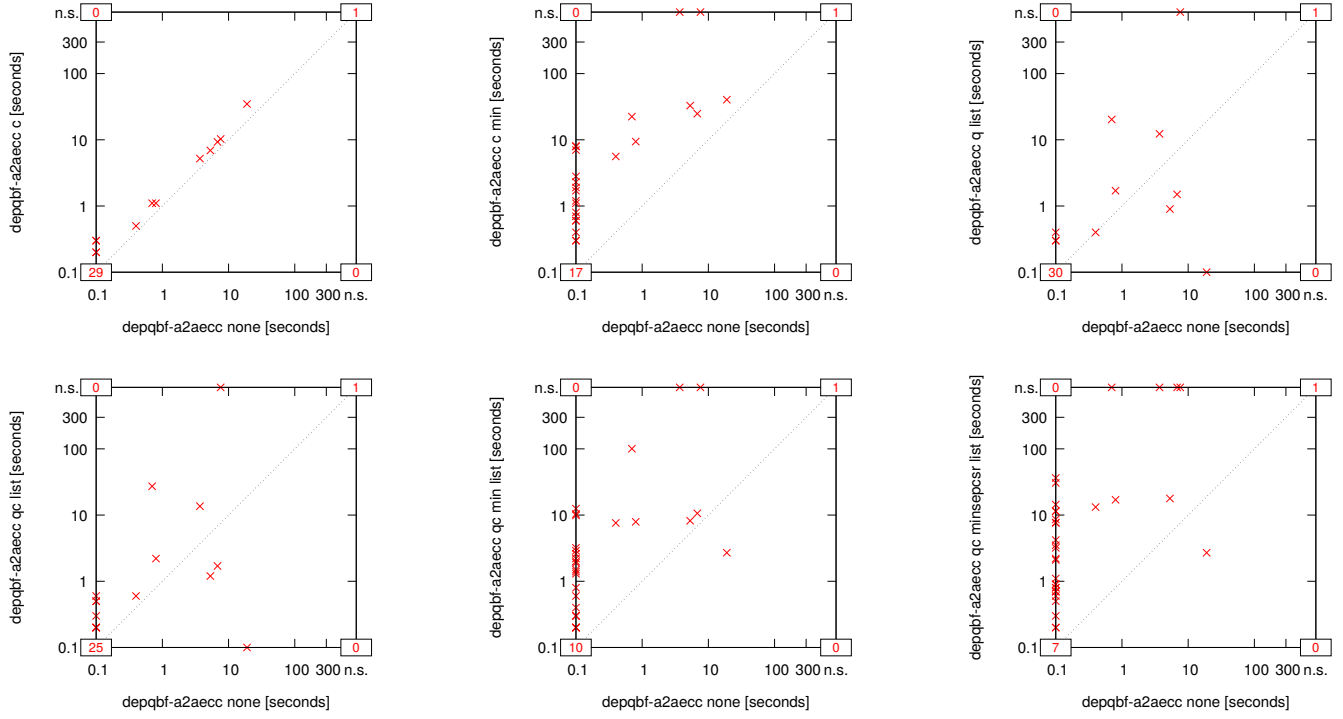


Fig. 1077: Suite Sauer-Reimer ($n = 42$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

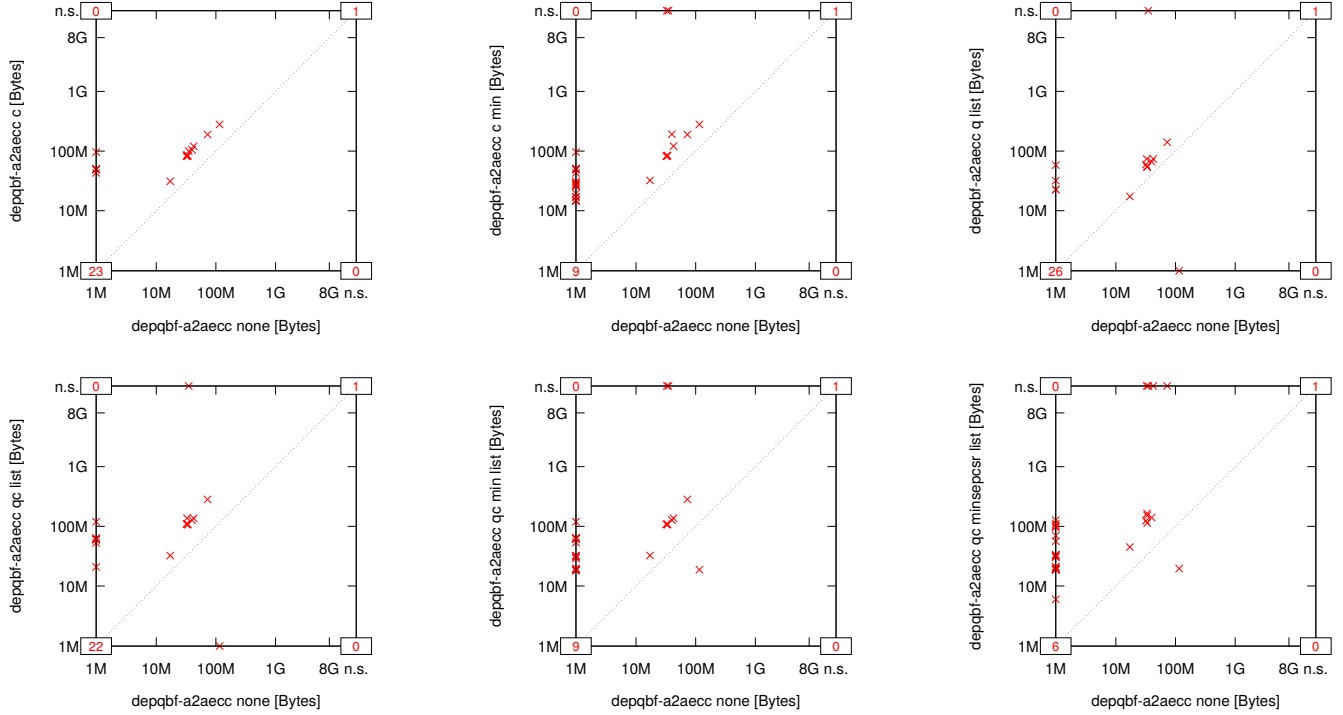


Fig. 1078: Suite Sauer-Reimer ($n = 42$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

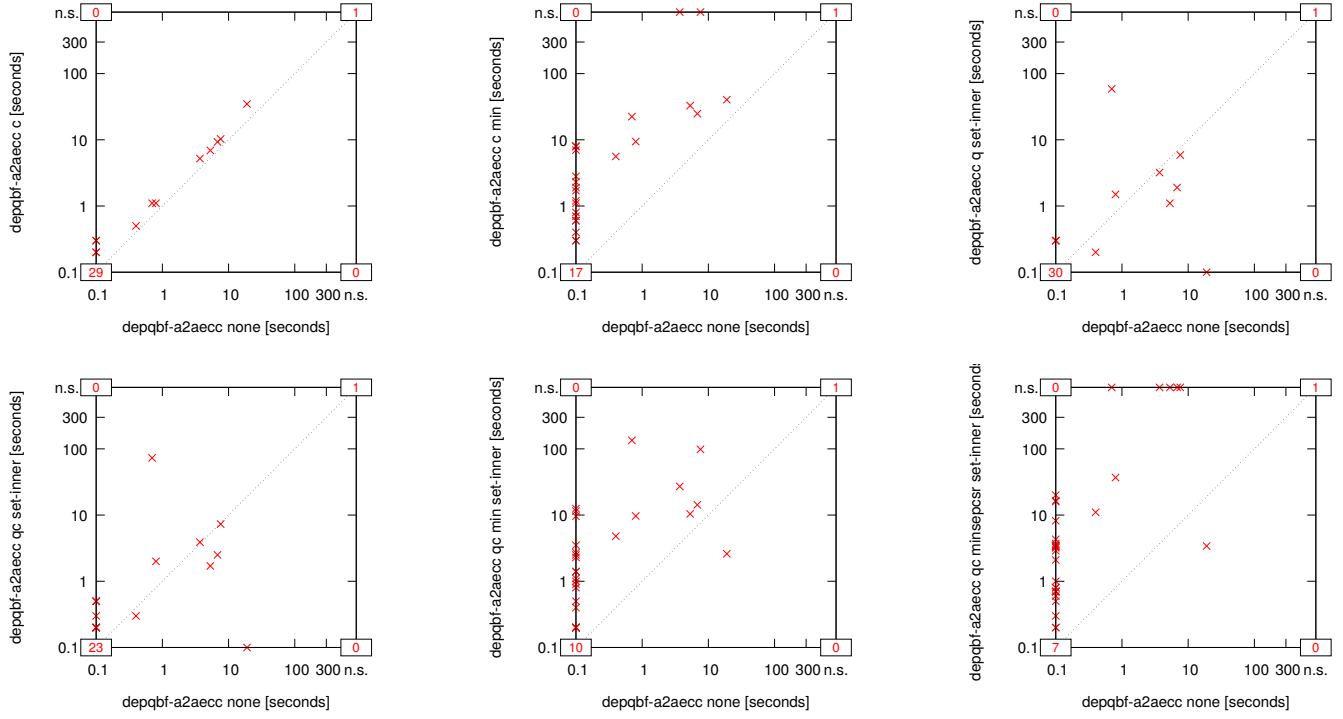


Fig. 1079: Suite Sauer-Reimer ($n = 42$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. Dep-QBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

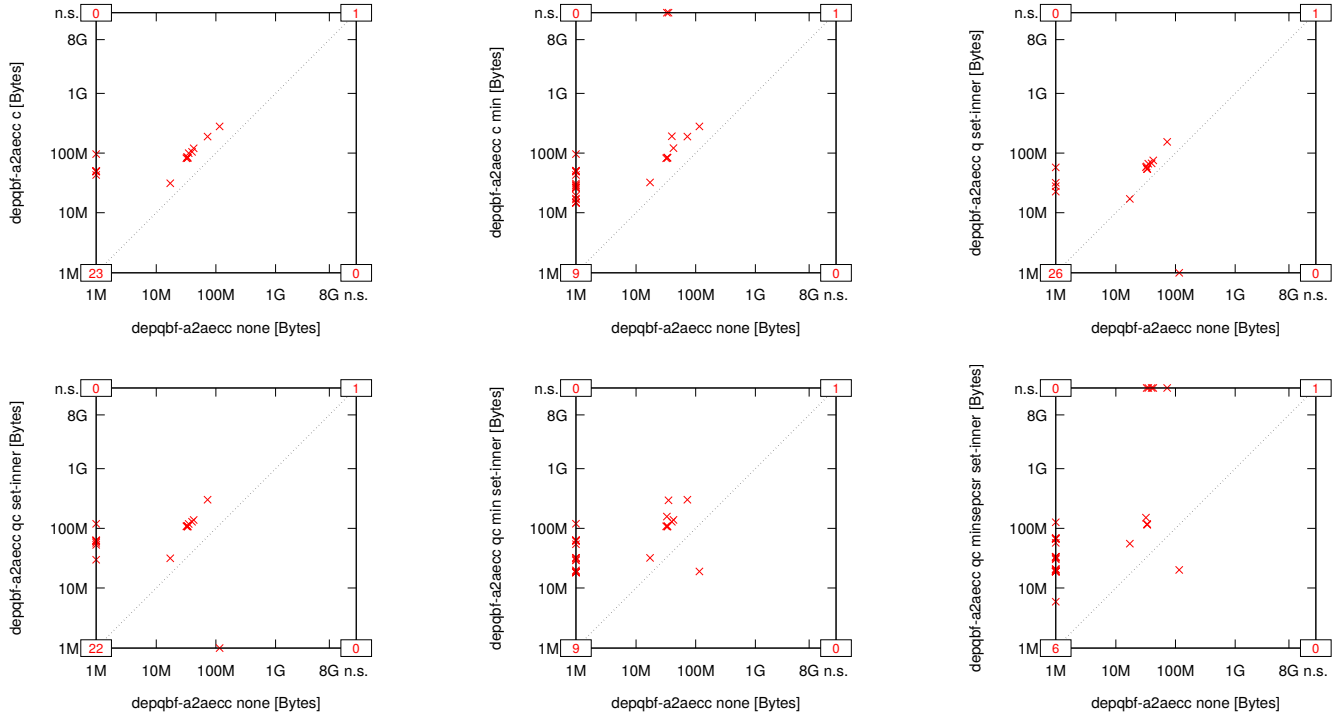


Fig. 1080: Suite Sauer-Reimer ($n = 42$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. Dep-QBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

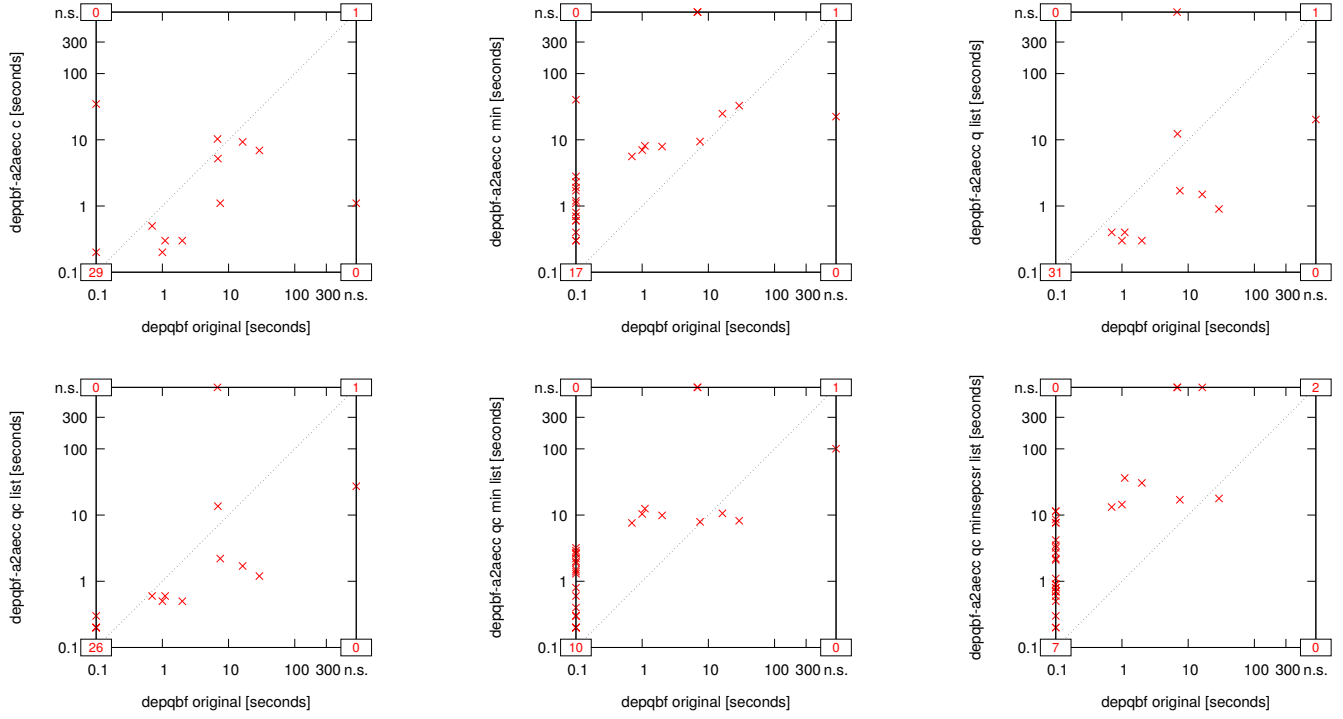


Fig. 1081: Suite Sauer-Reimer ($n = 42$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

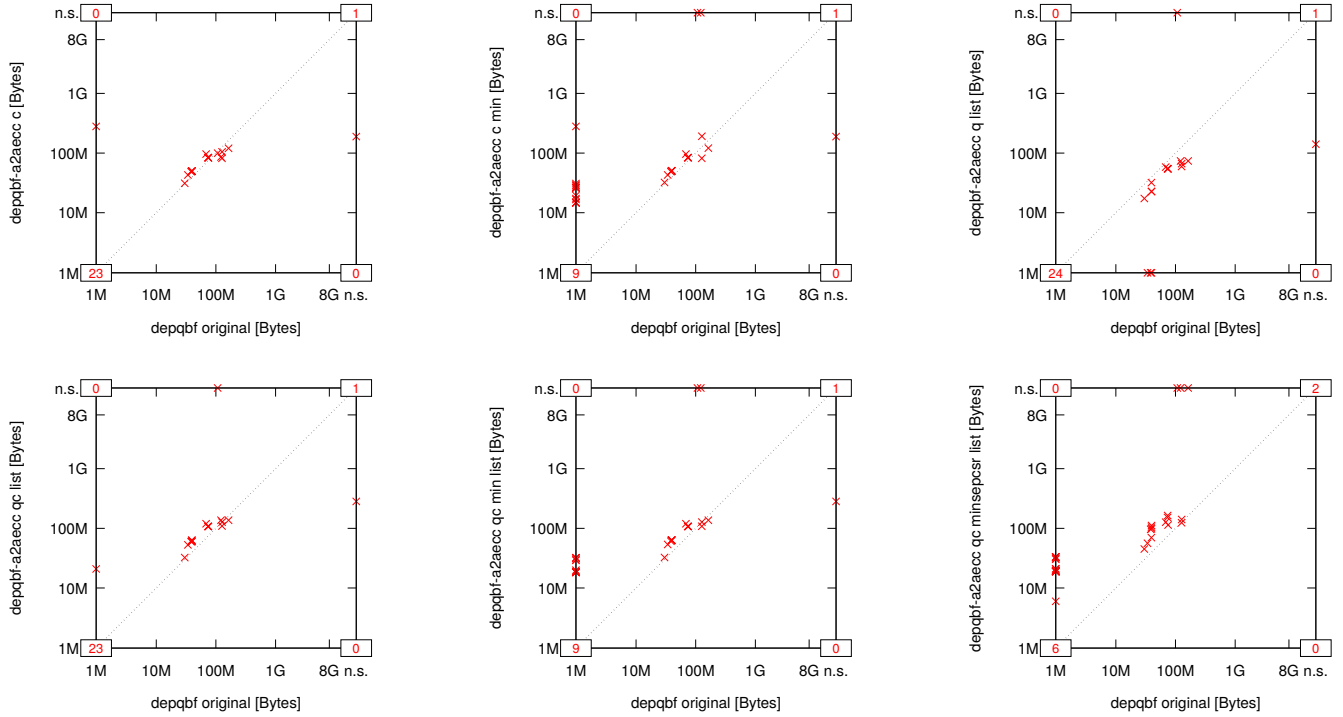


Fig. 1082: Suite Sauer-Reimer ($n = 42$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

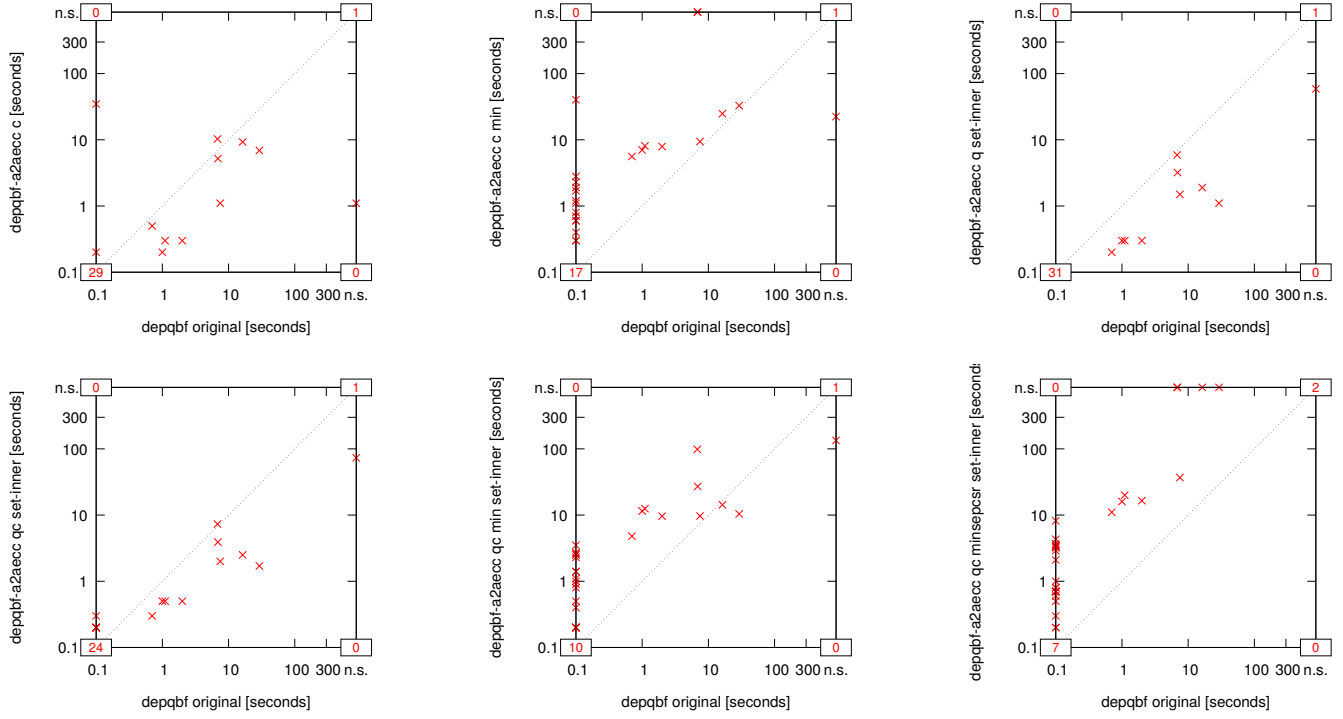


Fig. 1083: Suite Sauer-Reimer ($n = 42$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

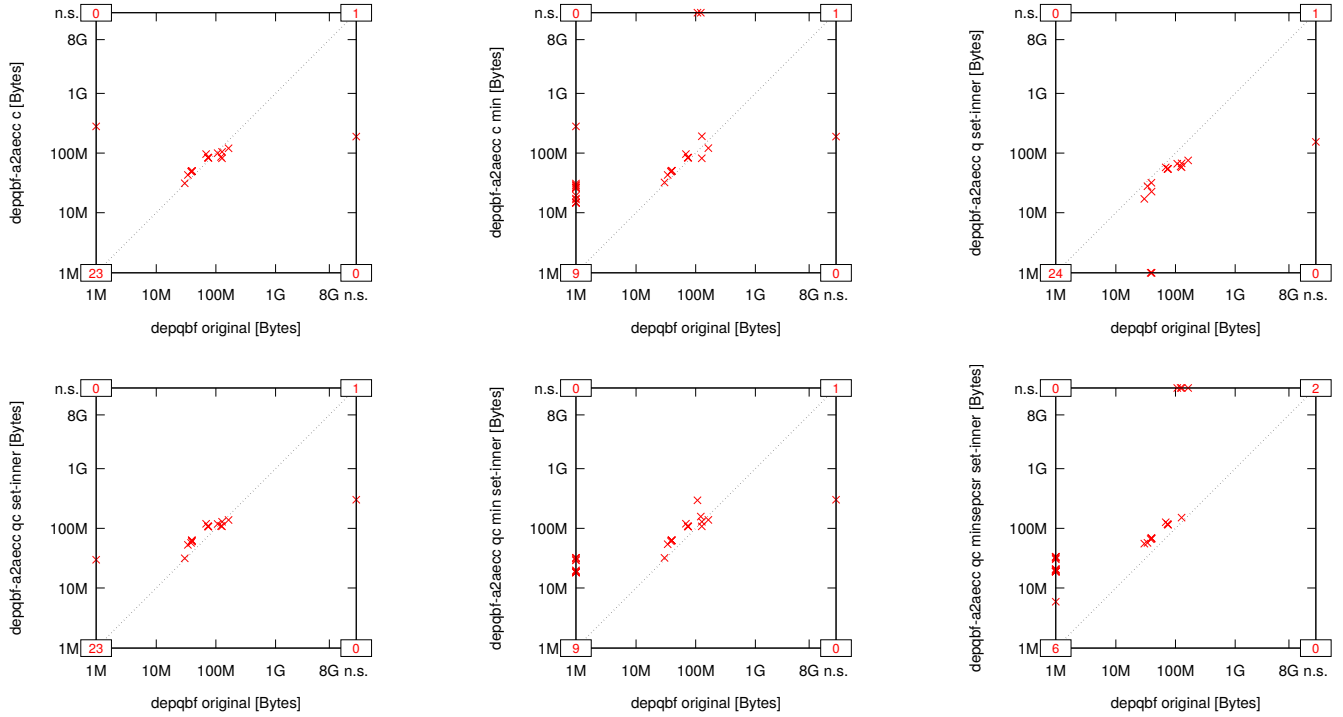


Fig. 1084: Suite Sauer-Reimer ($n = 42$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

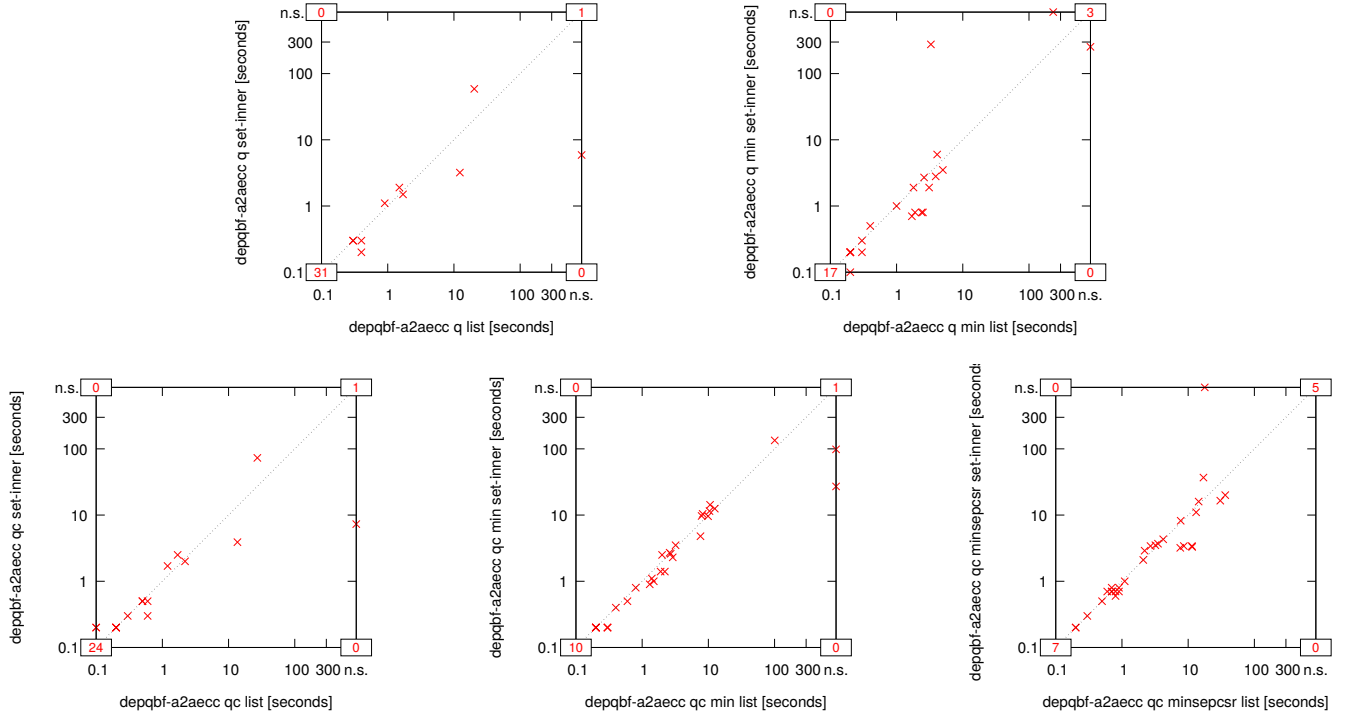


Fig. 1085: Suite Sauer-Reimer ($n = 42$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

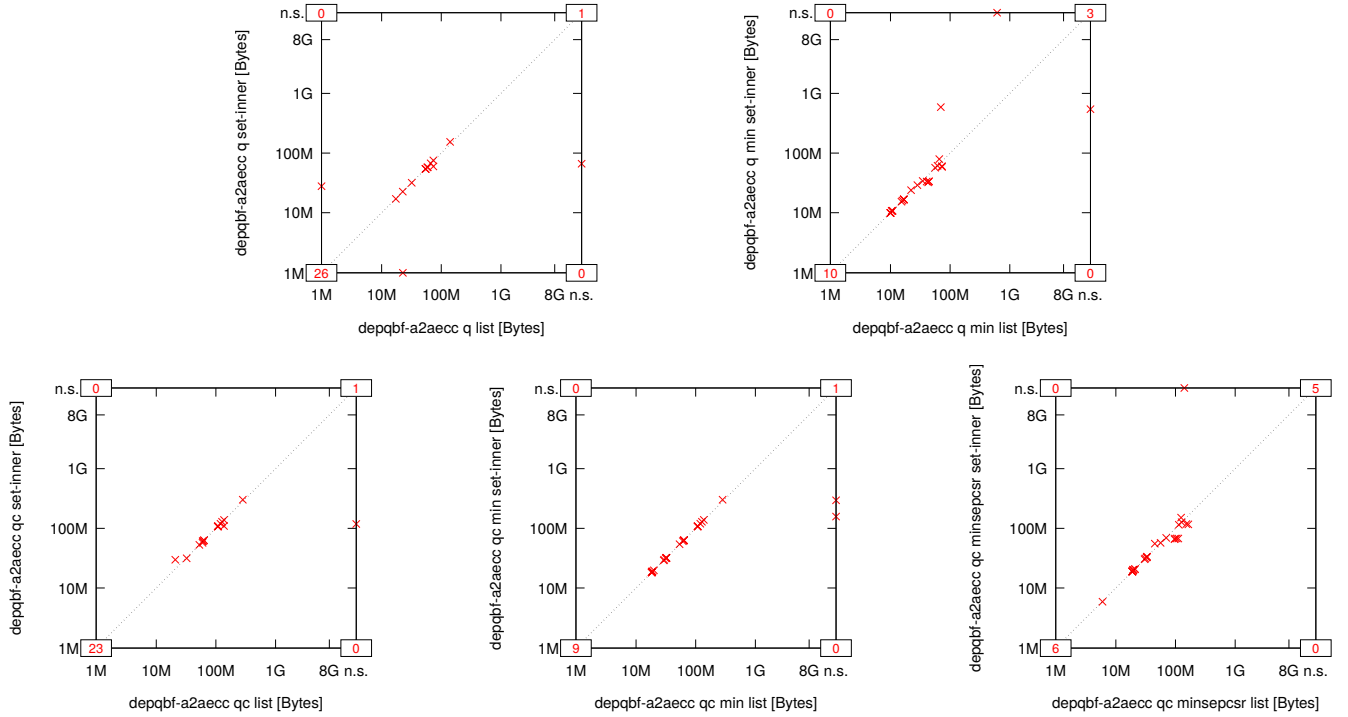


Fig. 1086: Suite Sauer-Reimer ($n = 42$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

42) Scholl-Becker ($n = 30$):

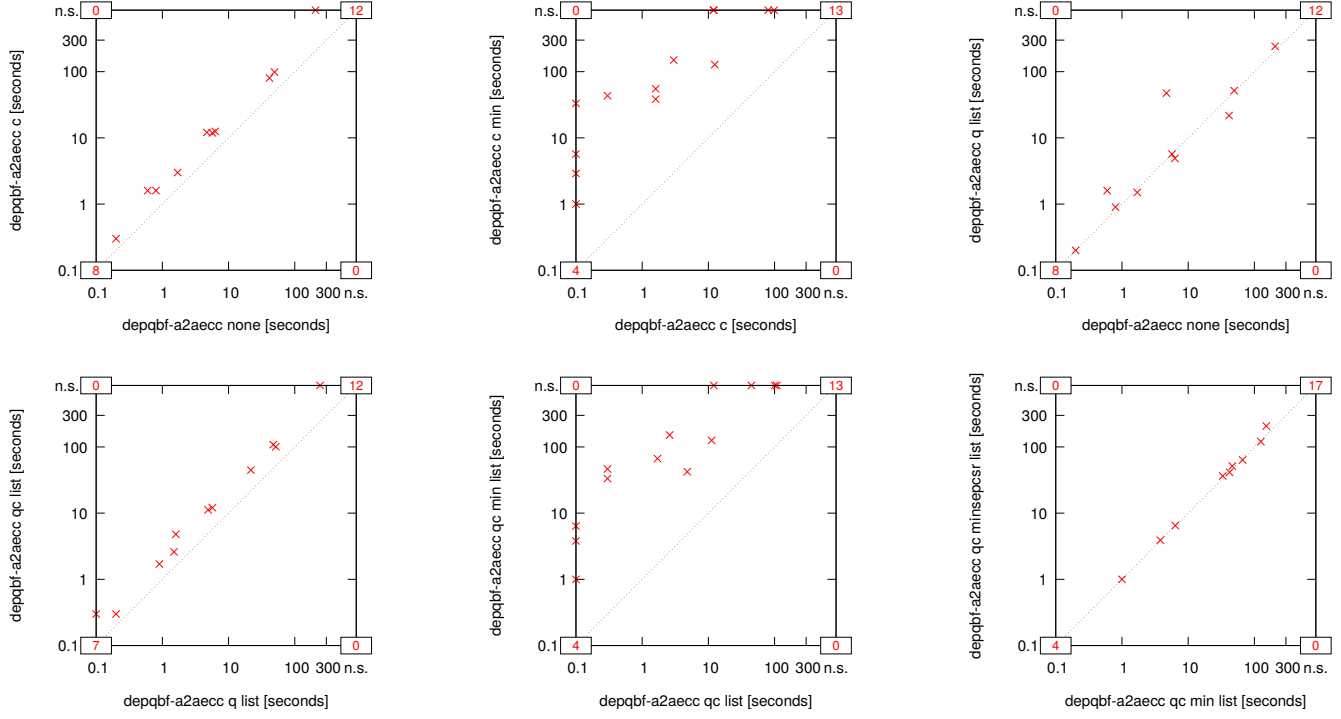


Fig. 1087: Suite Scholl-Becker ($n = 30$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

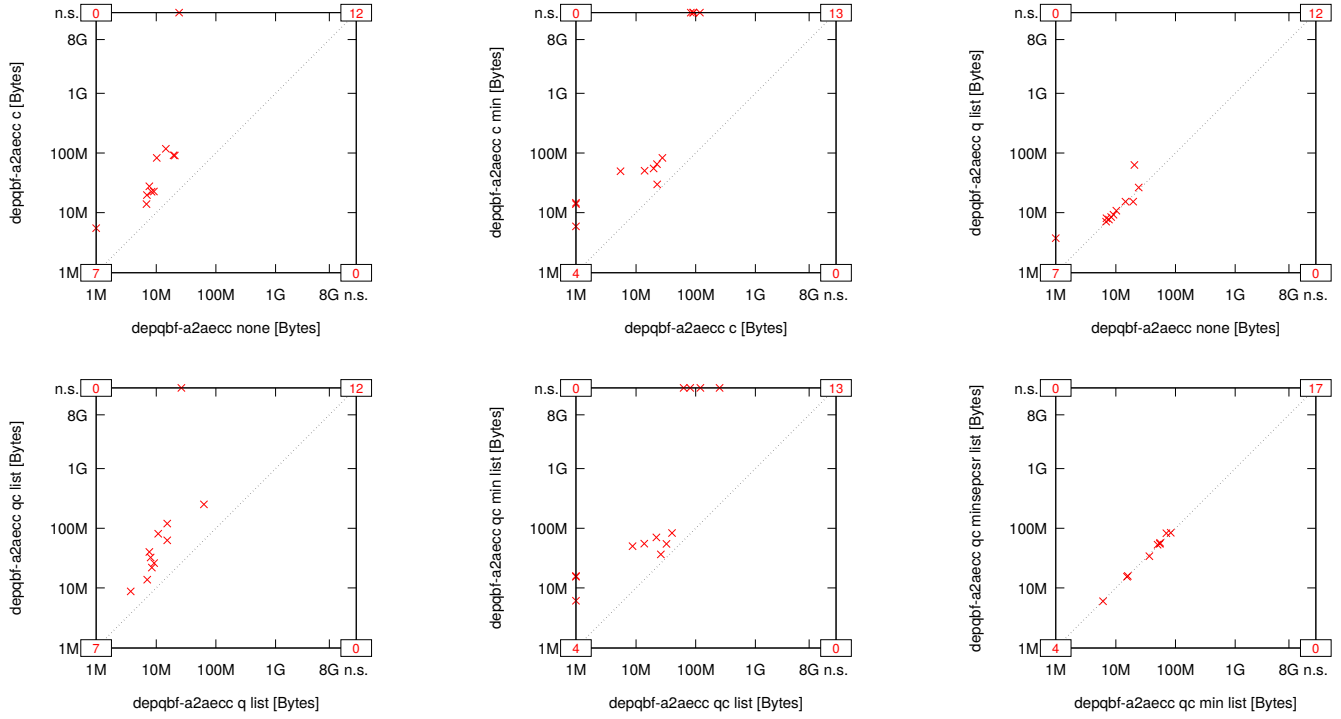


Fig. 1088: Suite Scholl-Becker ($n = 30$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

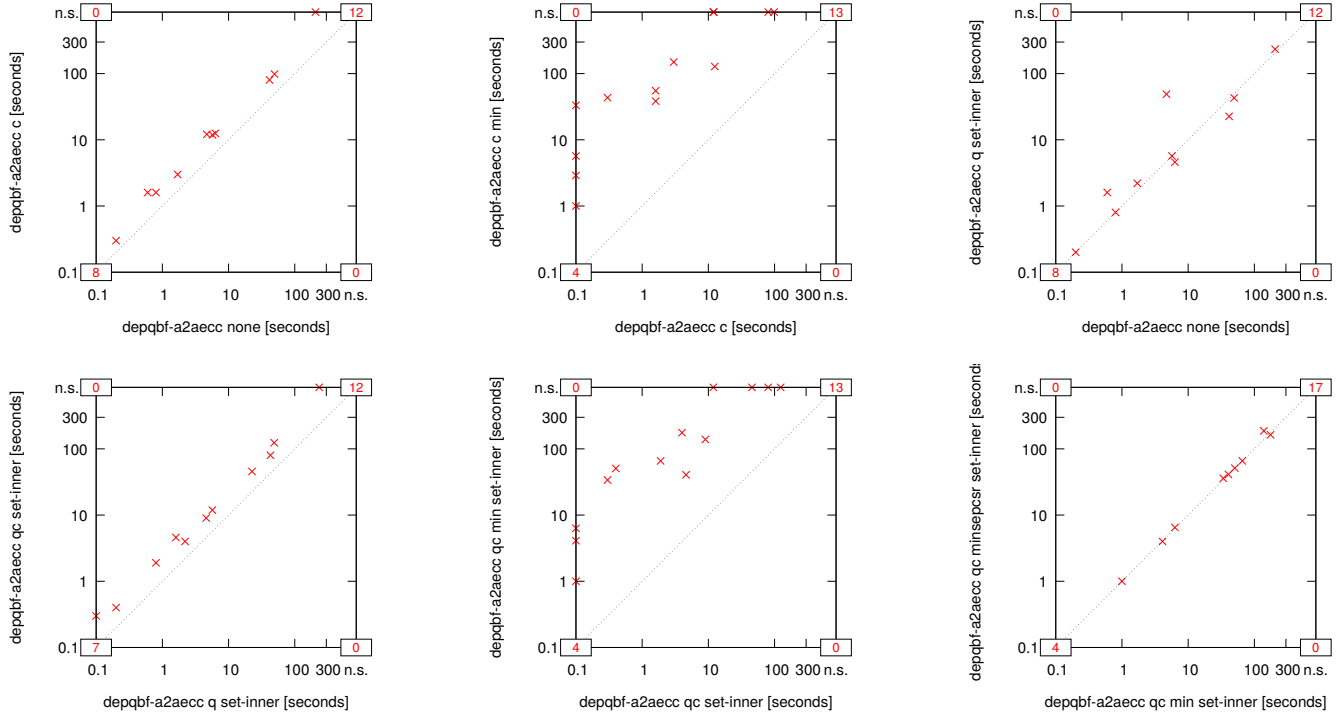


Fig. 1089: Suite Scholl-Becker ($n = 30$): Comparing run times for extracting different unsatisfiable cores in `DepQBF-a2aecc` with `set-inner` semantics (run time in seconds).

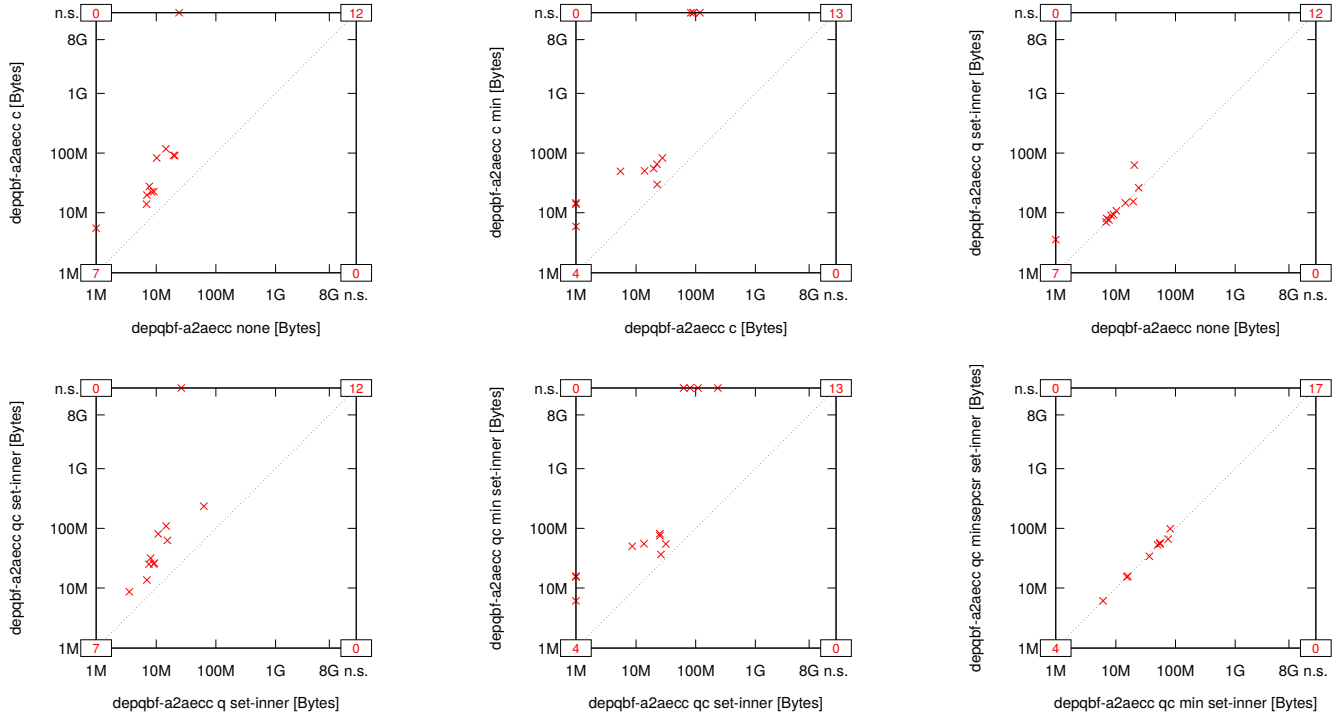


Fig. 1090: Suite Scholl-Becker ($n = 30$): Comparing memory usage for extracting different unsatisfiable cores in `DepQBF-a2aecc` with `set-inner` semantics (memory usage in Bytes).

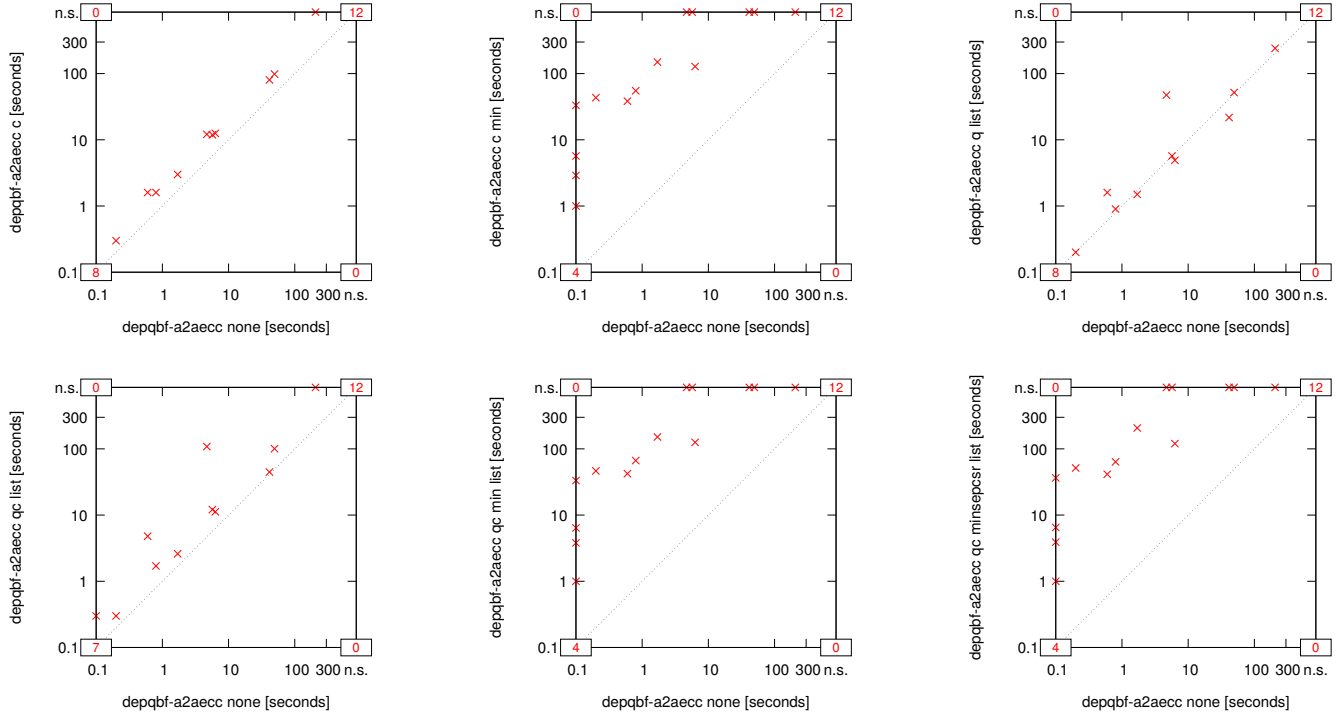


Fig. 1091: Suite Scholl-Becker ($n = 30$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

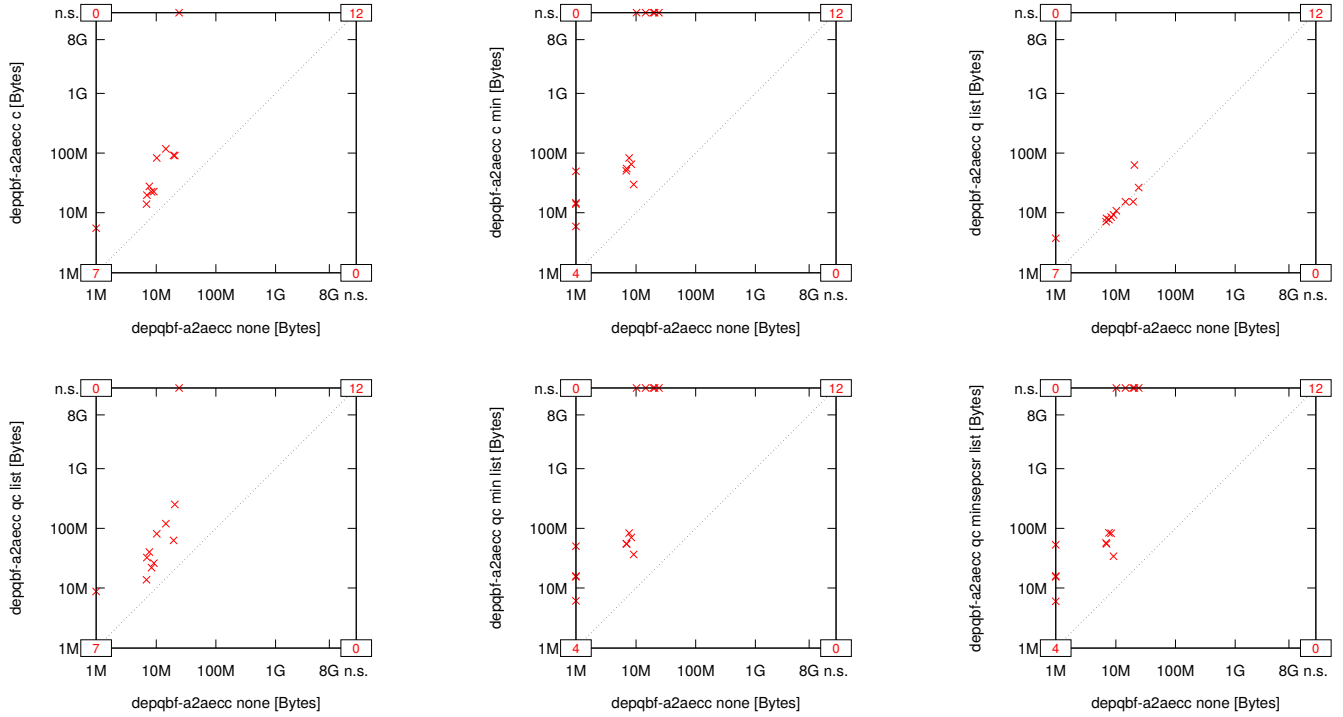


Fig. 1092: Suite Scholl-Becker ($n = 30$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

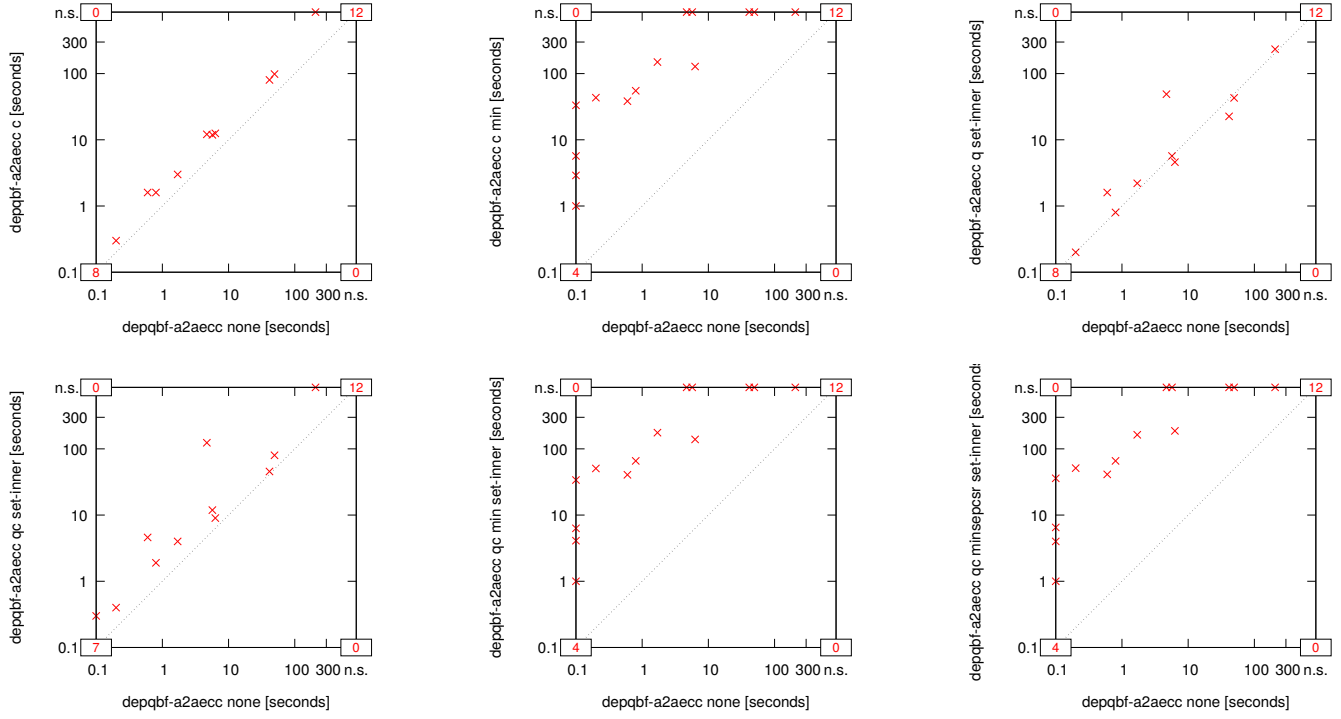


Fig. 1093: Suite Scholl-Becker ($n = 30$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. Dep-QBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

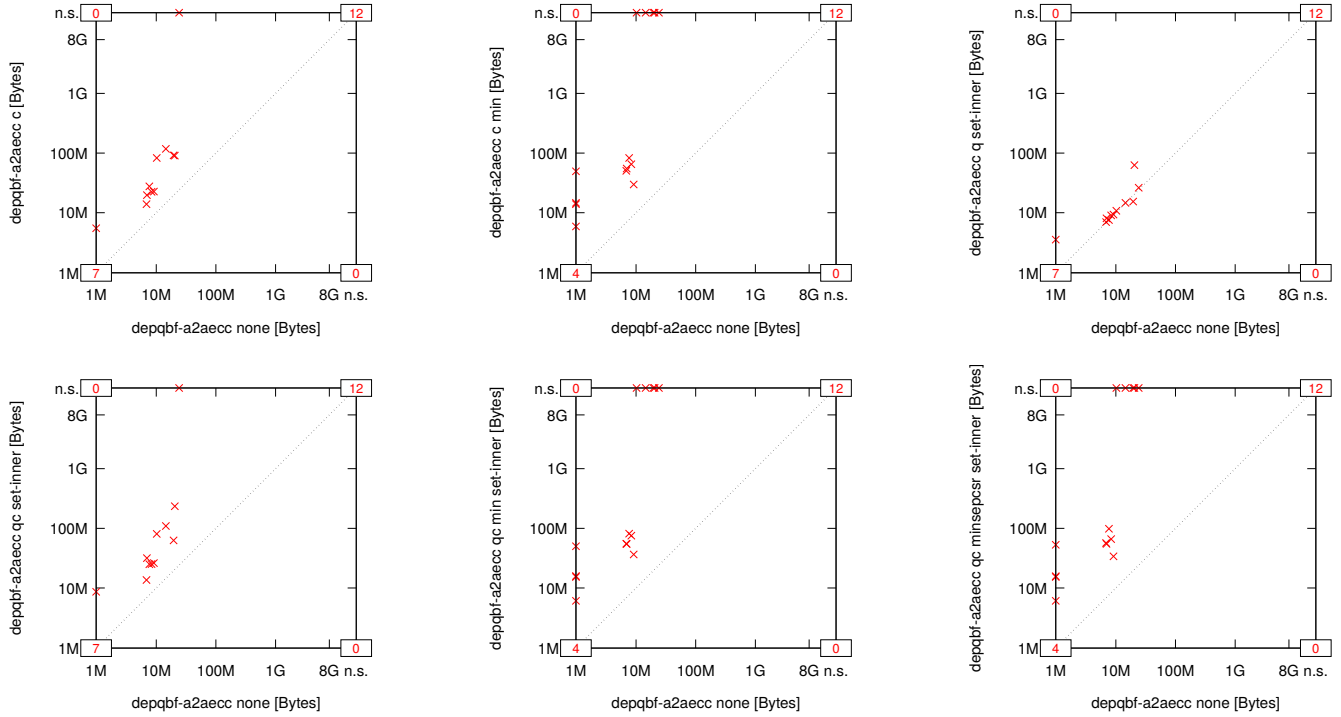


Fig. 1094: Suite Scholl-Becker ($n = 30$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. Dep-QBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

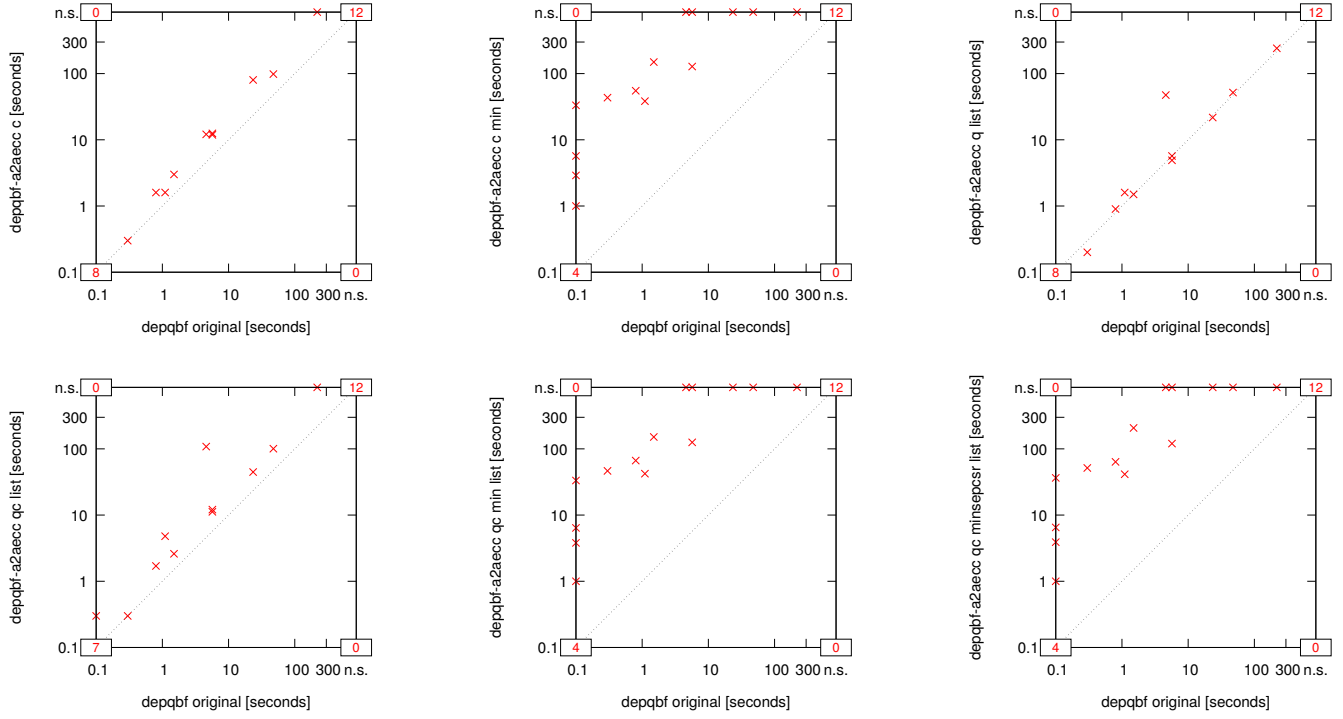


Fig. 1095: Suite Scholl-Becker ($n = 30$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

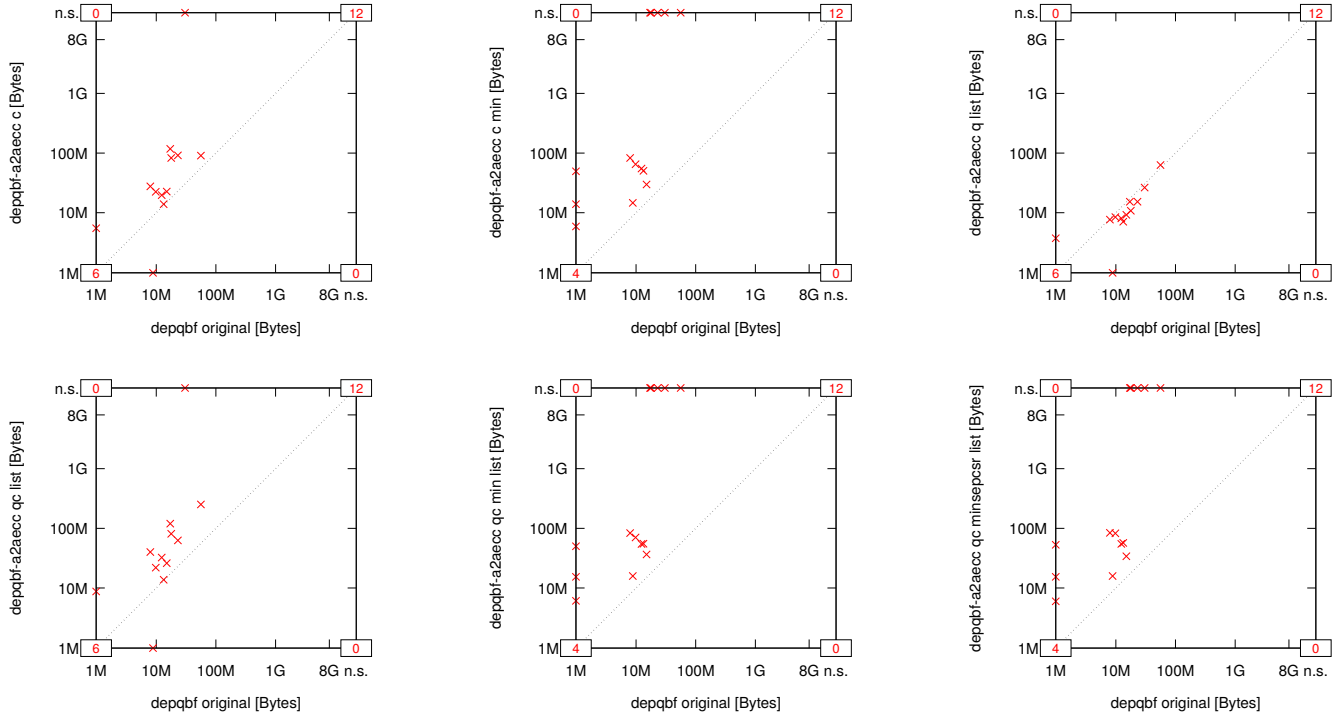


Fig. 1096: Suite Scholl-Becker ($n = 30$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

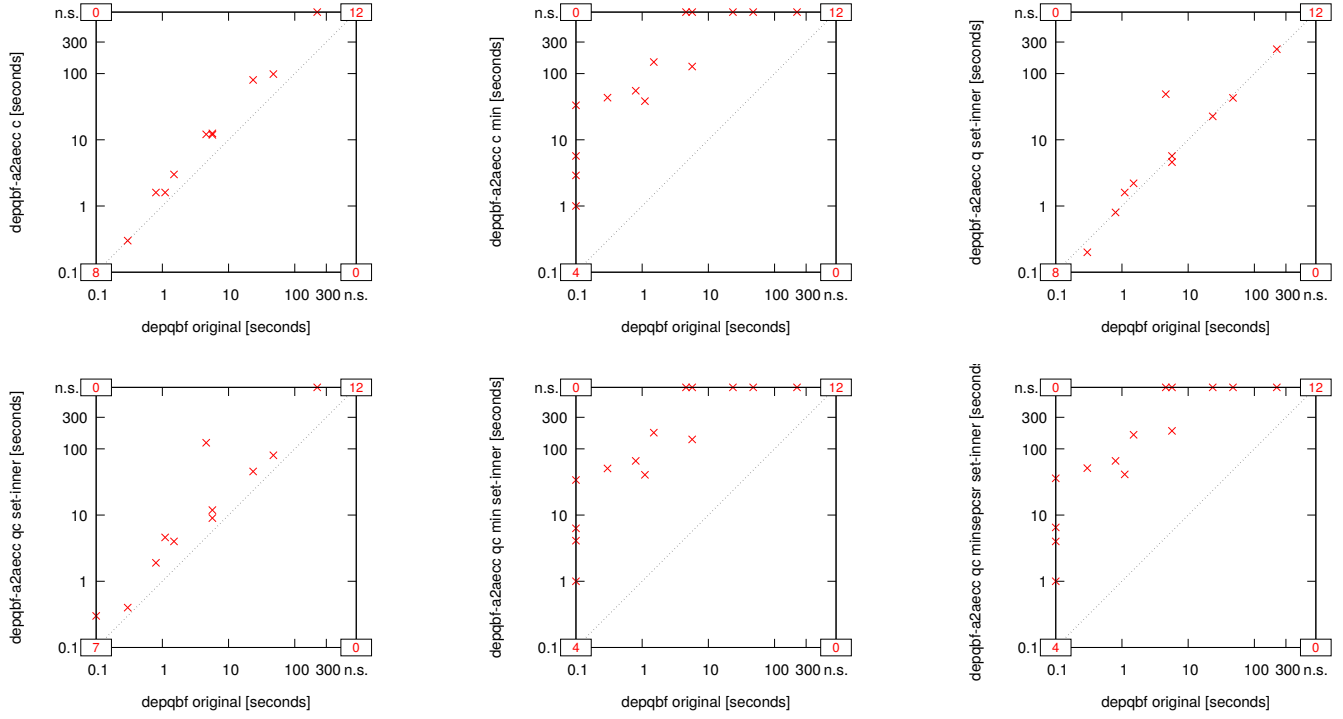


Fig. 1097: Suite Scholl-Becker ($n = 30$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

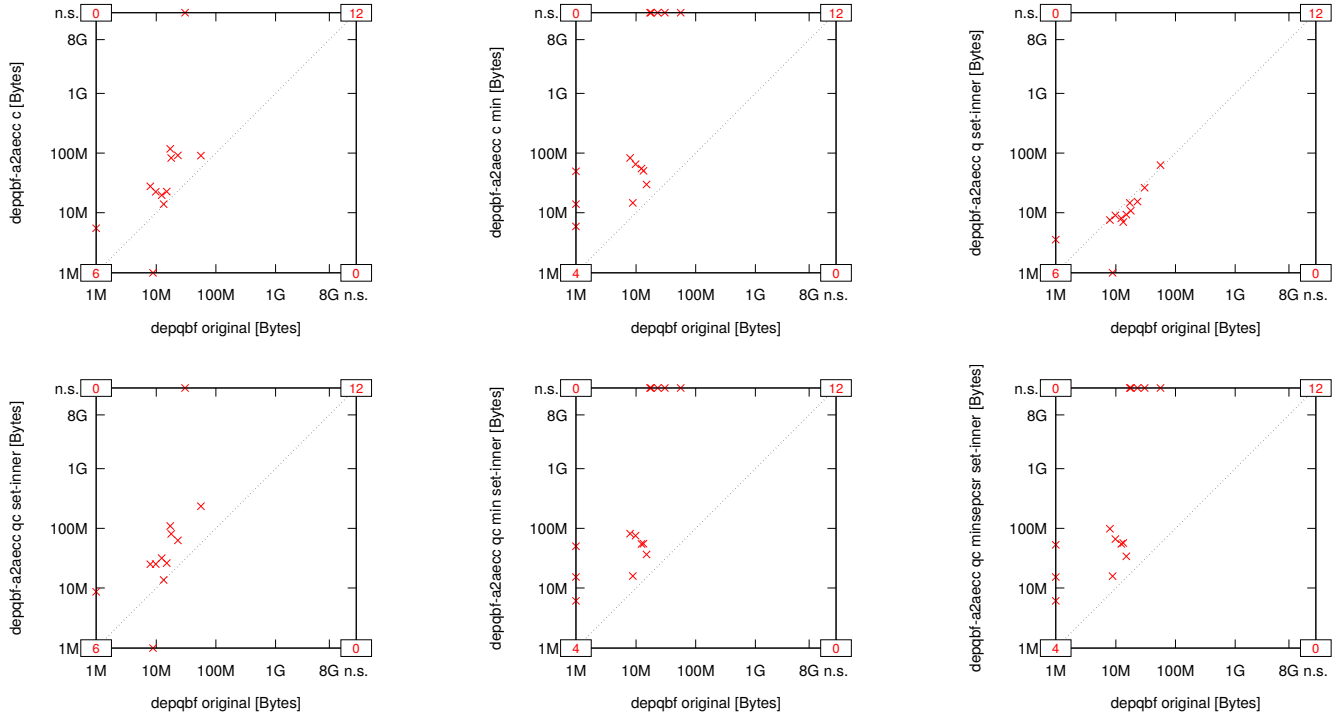


Fig. 1098: Suite Scholl-Becker ($n = 30$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

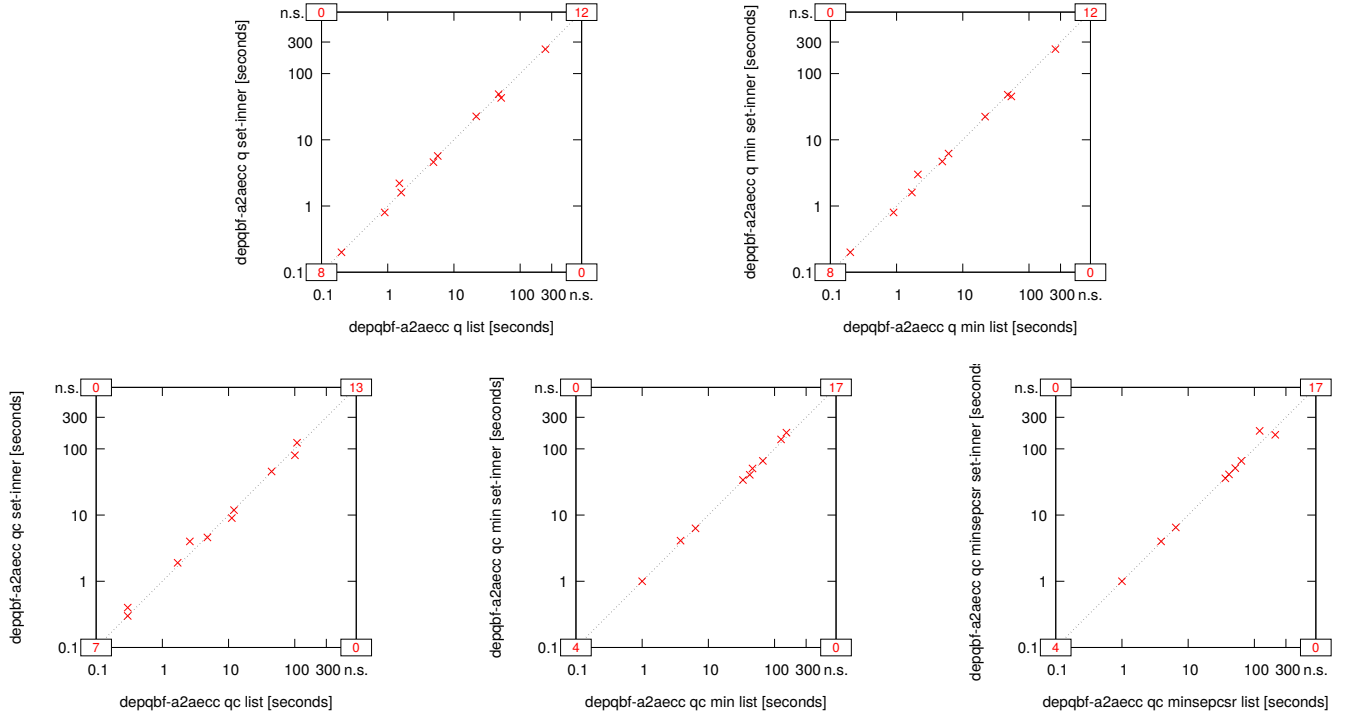


Fig. 1099: Suite Scholl-Becker ($n = 30$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

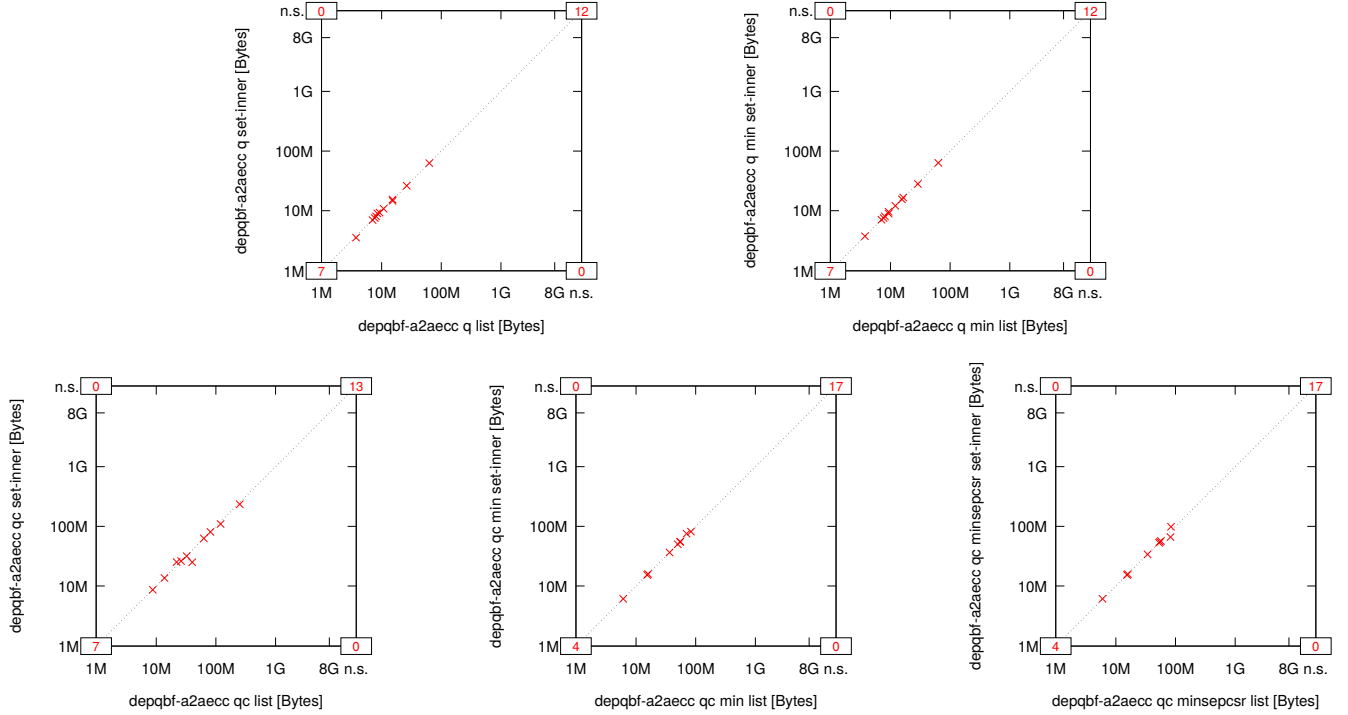


Fig. 1100: Suite Scholl-Becker ($n = 30$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

43) Seidl ($n = 194$):

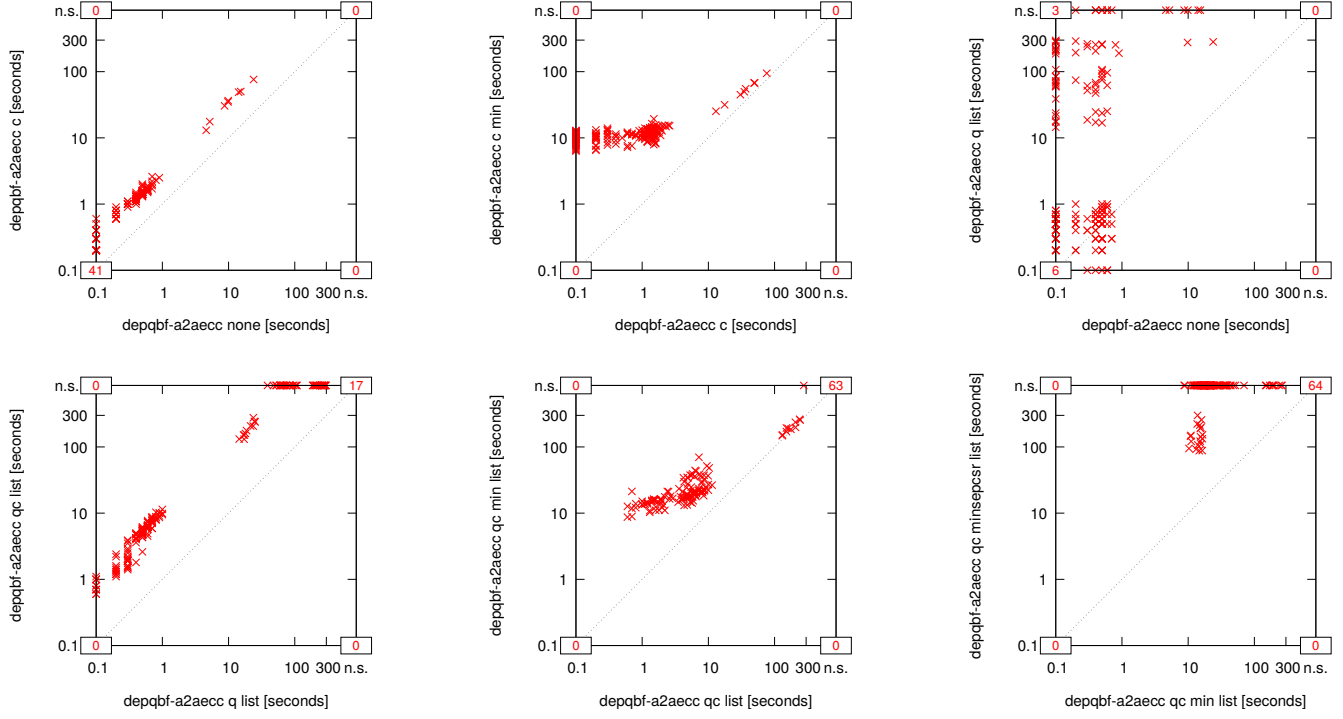


Fig. 1101: Suite Seidl ($n = 194$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

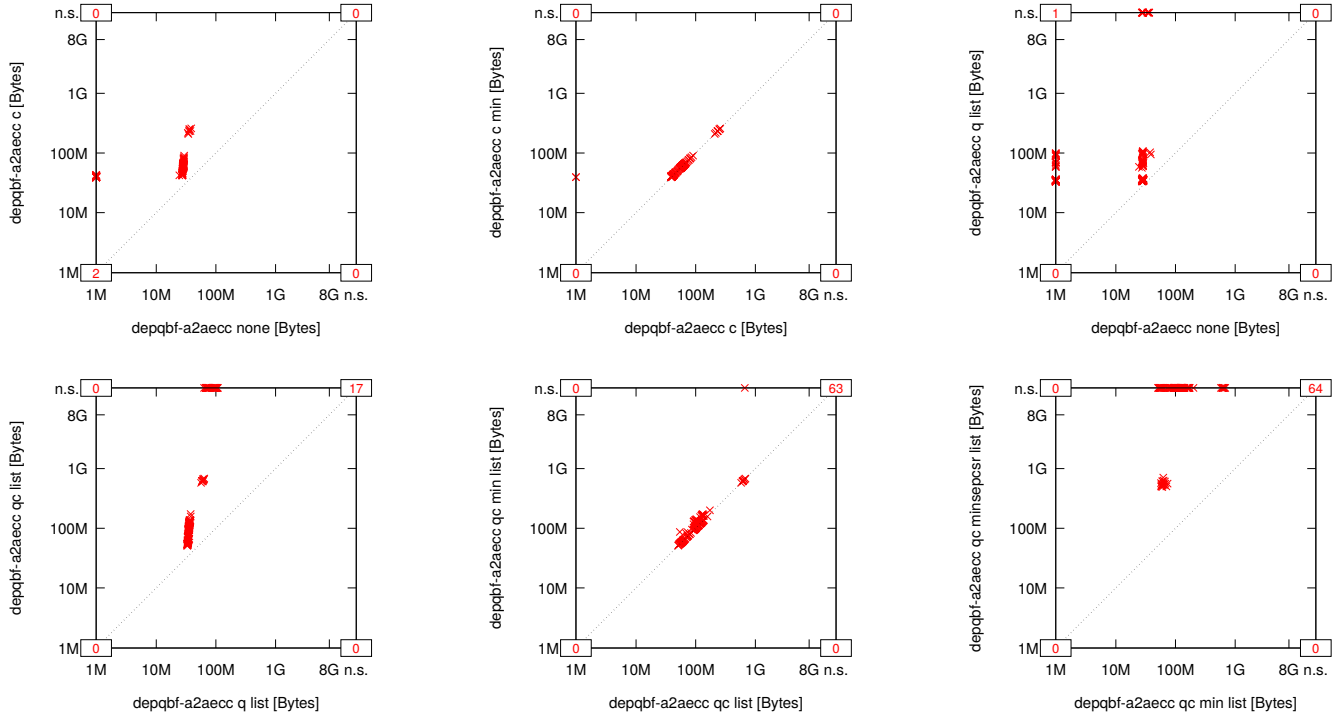


Fig. 1102: Suite Seidl ($n = 194$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

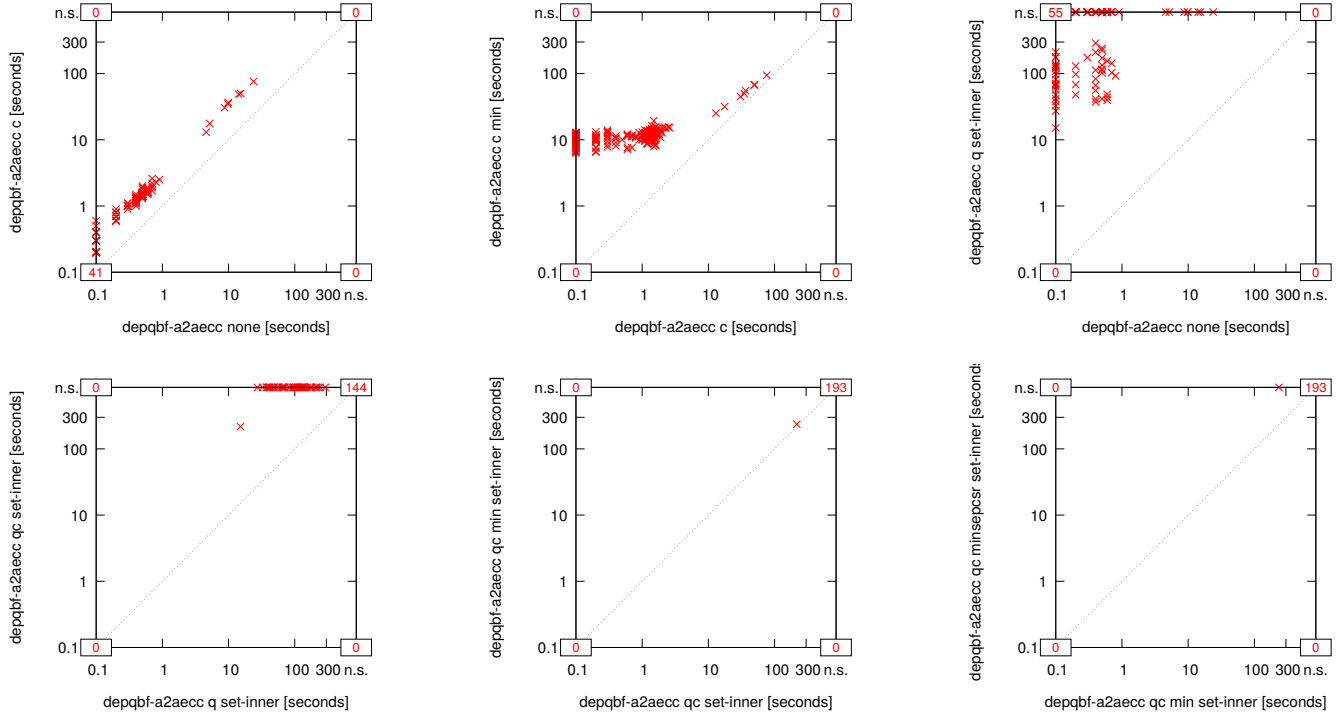


Fig. 1103: Suite Seidl ($n = 194$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

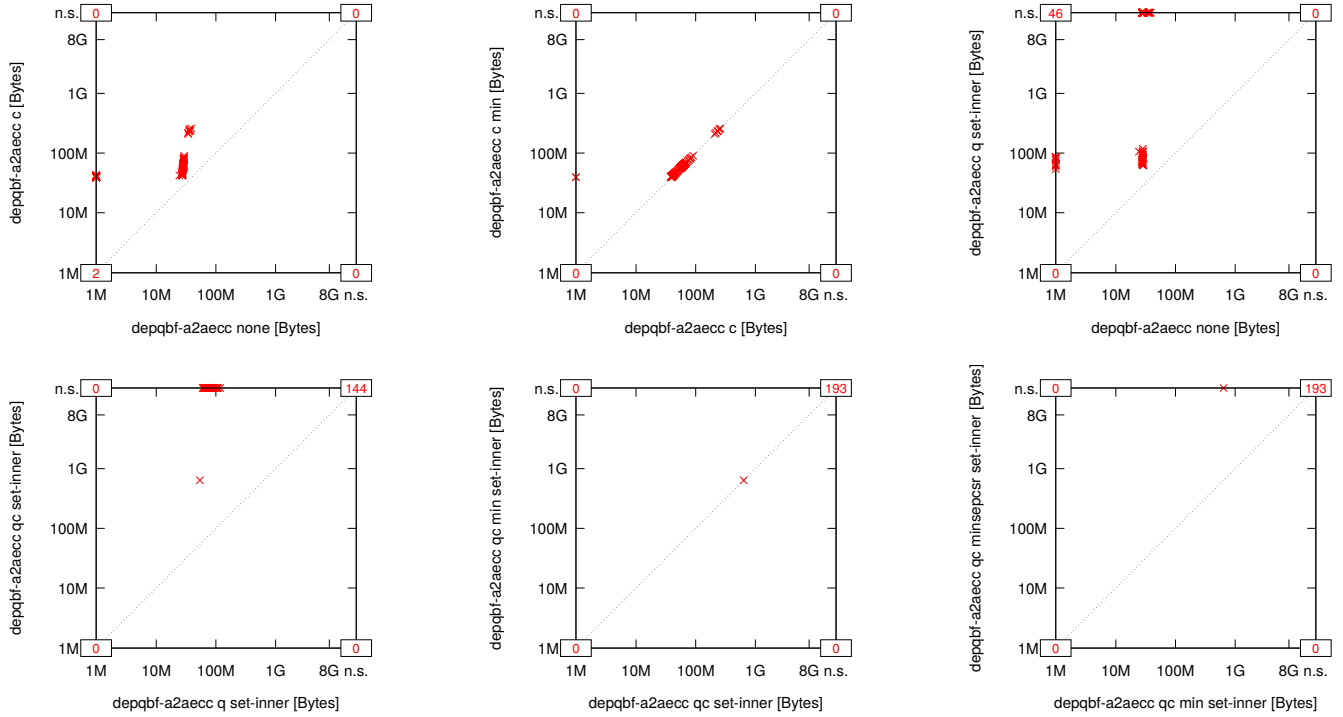


Fig. 1104: Suite Seidl ($n = 194$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

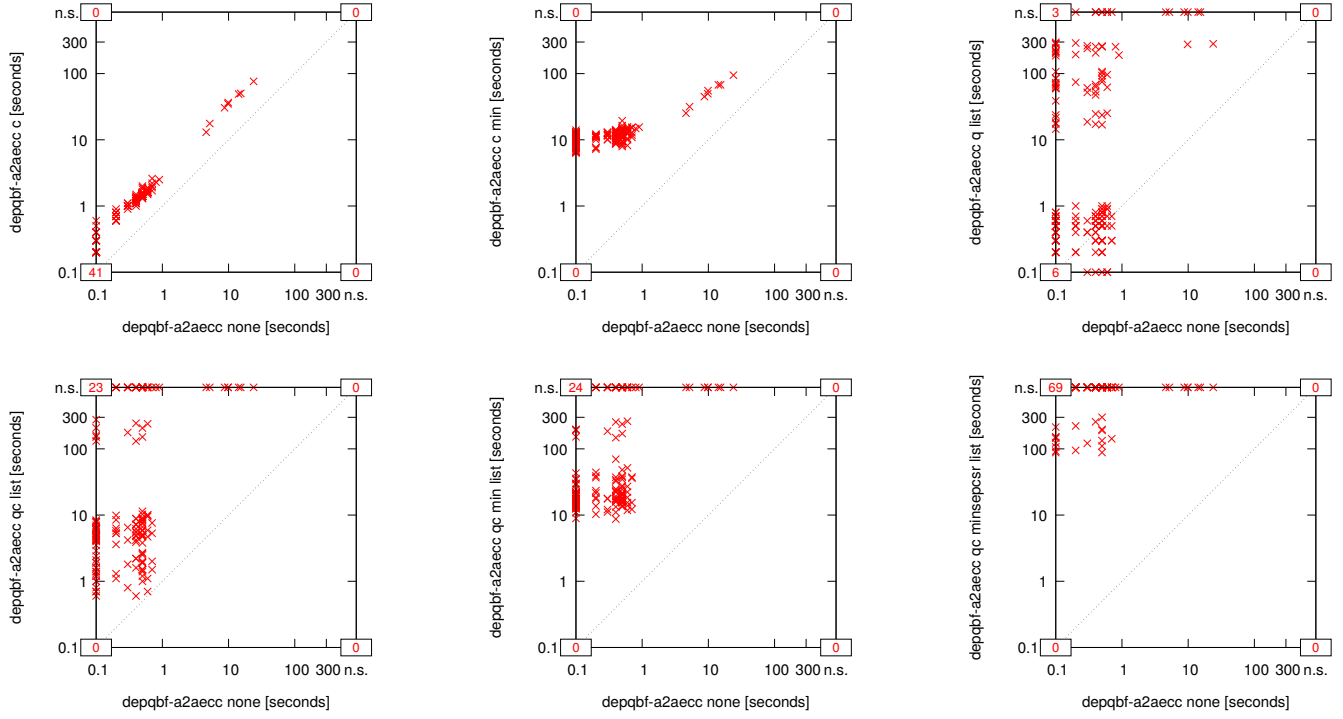


Fig. 1105: Suite Seidl ($n = 194$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

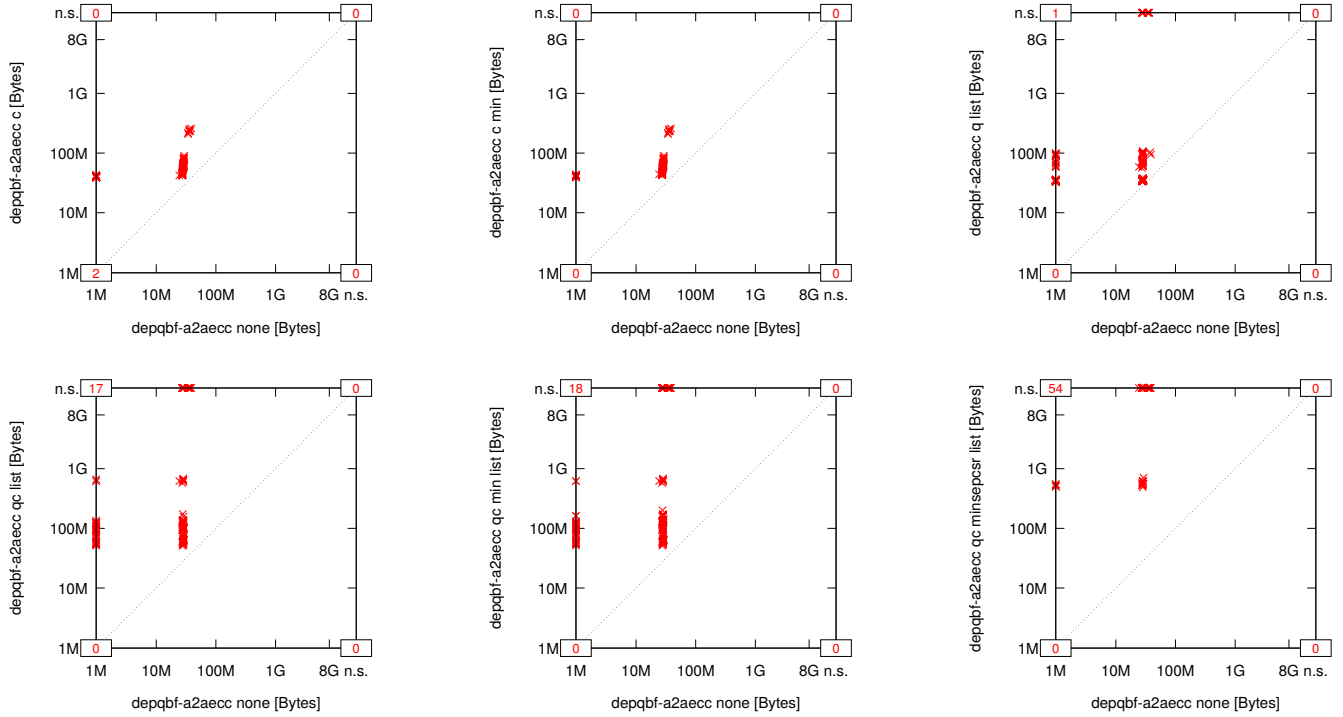


Fig. 1106: Suite Seidl ($n = 194$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

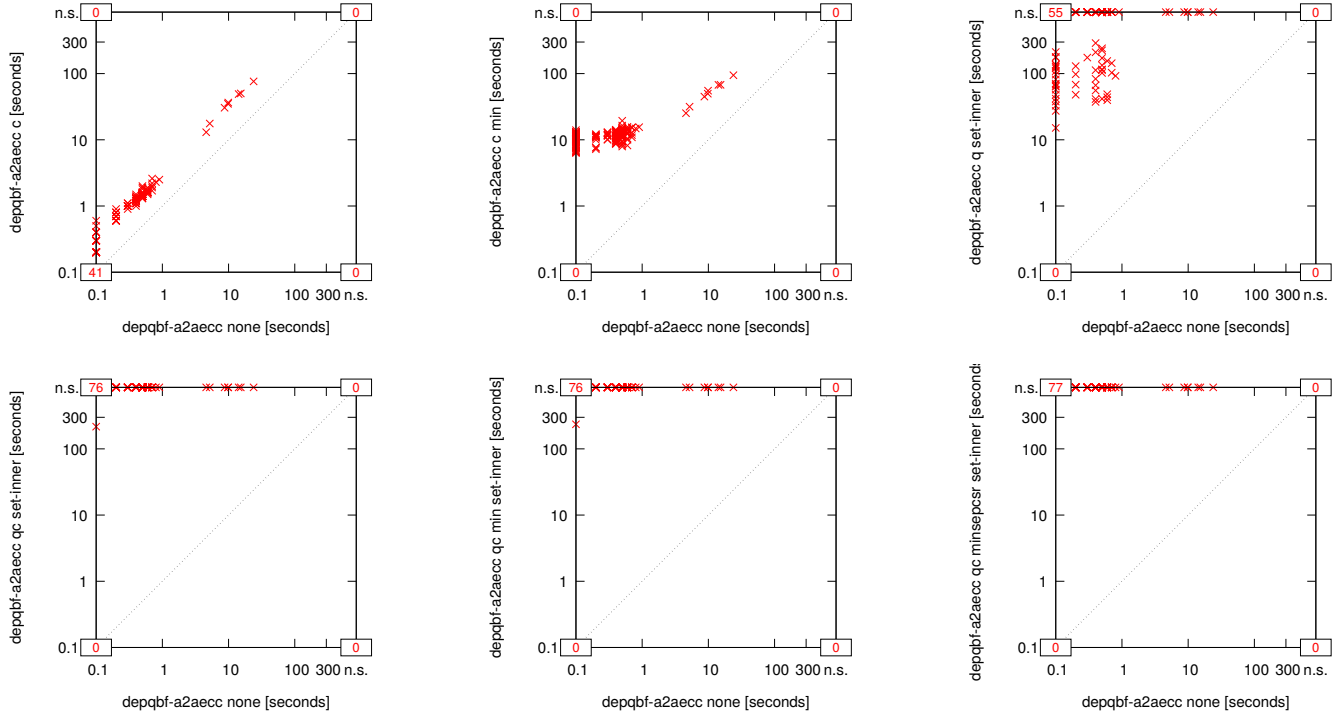


Fig. 1107: Suite Seidl ($n = 194$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

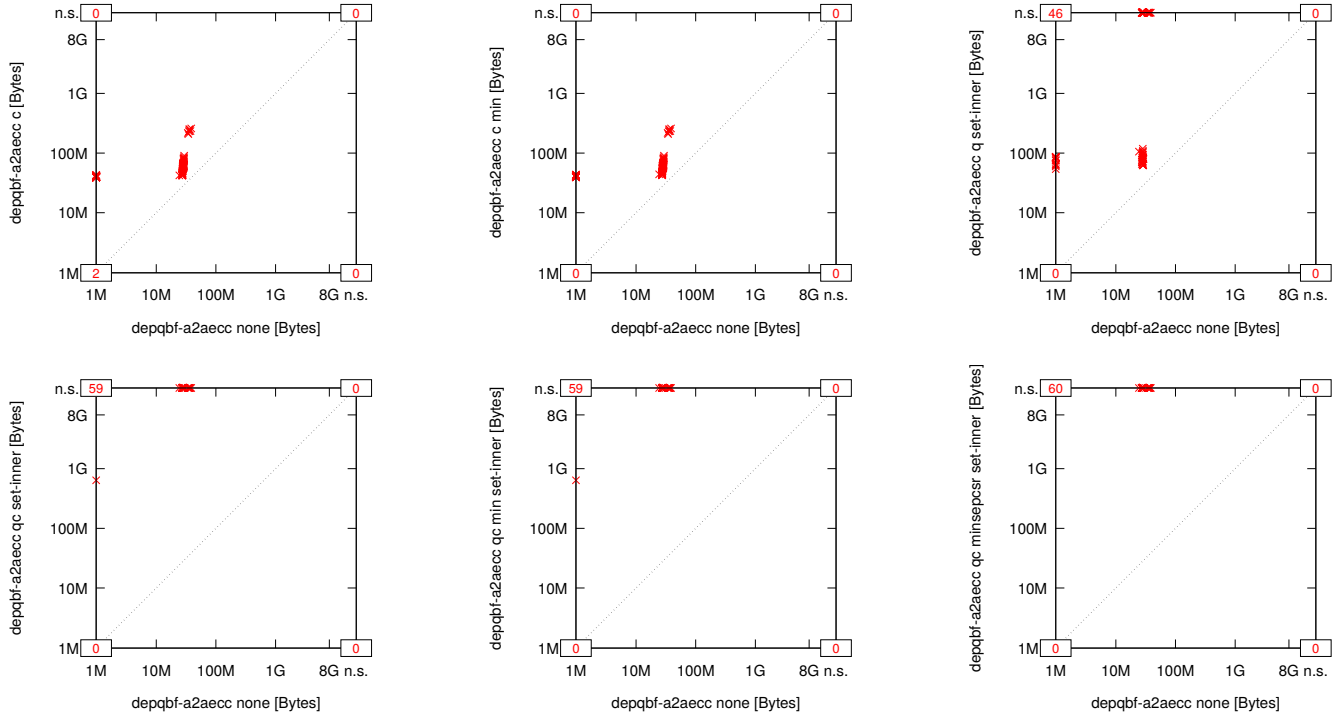


Fig. 1108: Suite Seidl ($n = 194$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

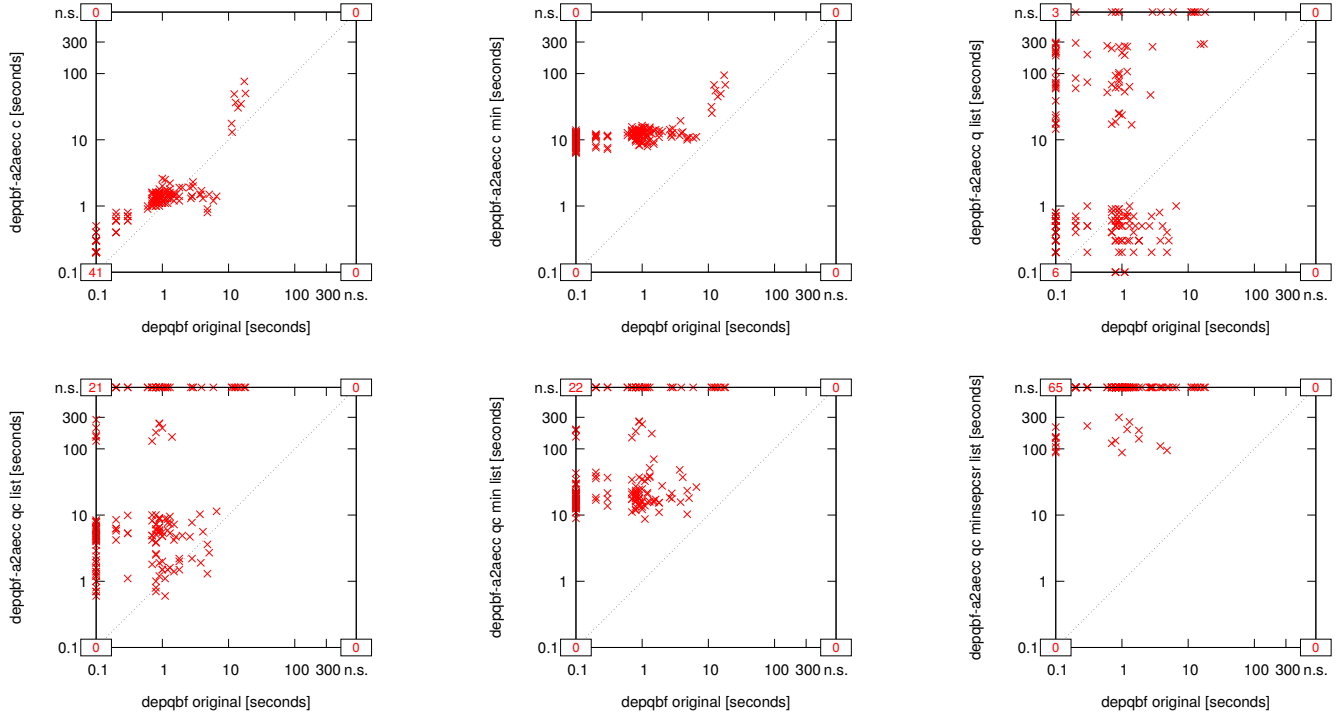


Fig. 1109: Suite Seidl ($n = 194$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

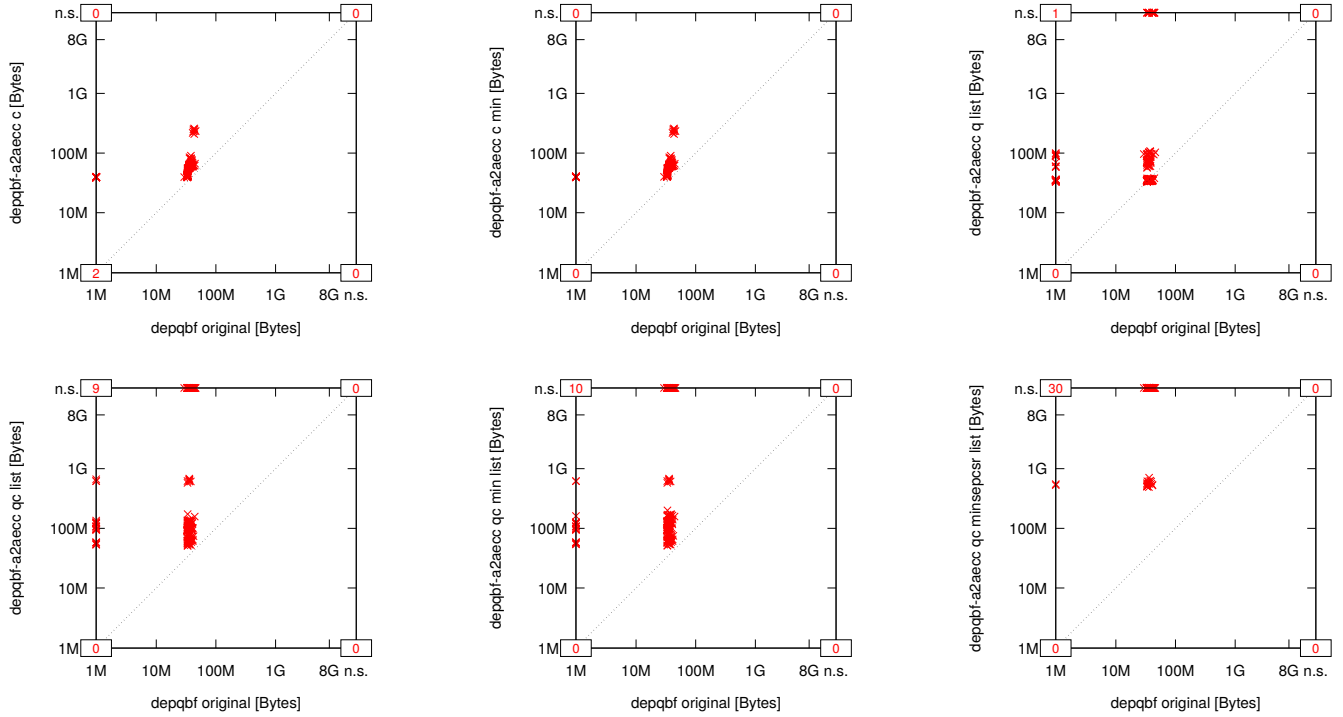


Fig. 1110: Suite Seidl ($n = 194$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

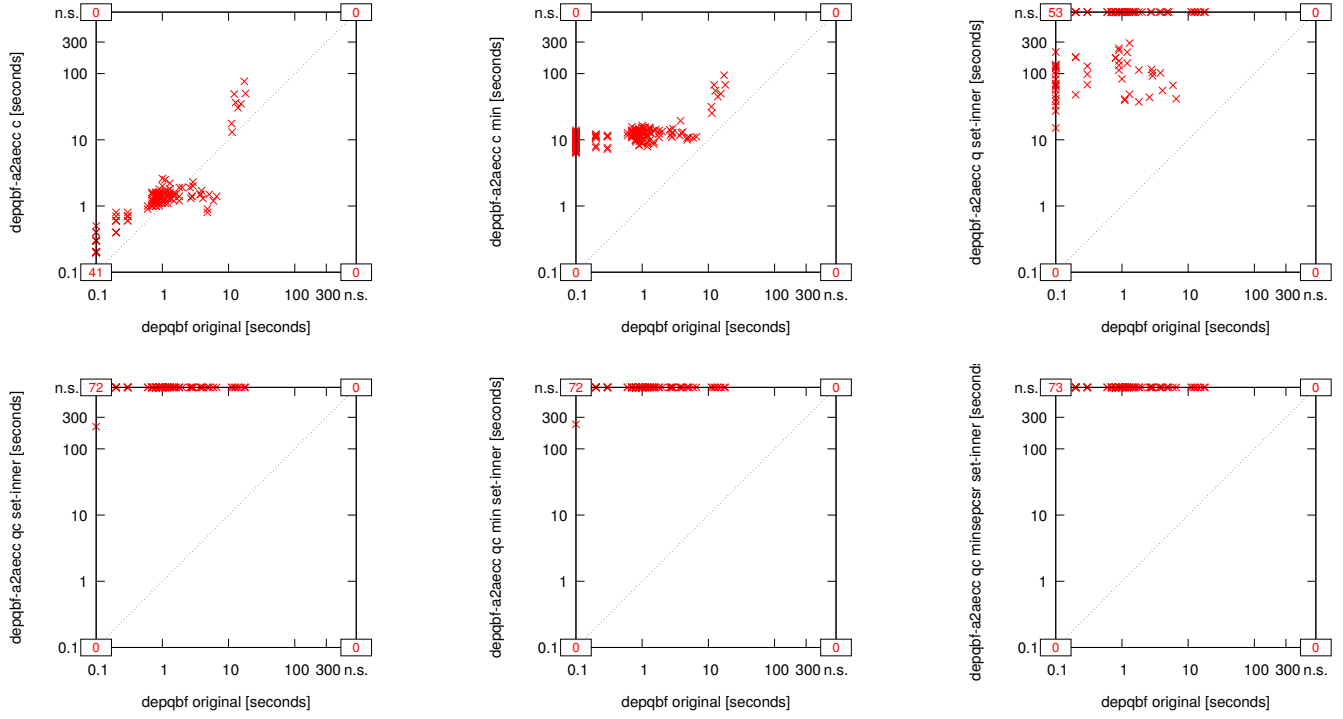


Fig. 1111: Suite Seidl ($n = 194$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

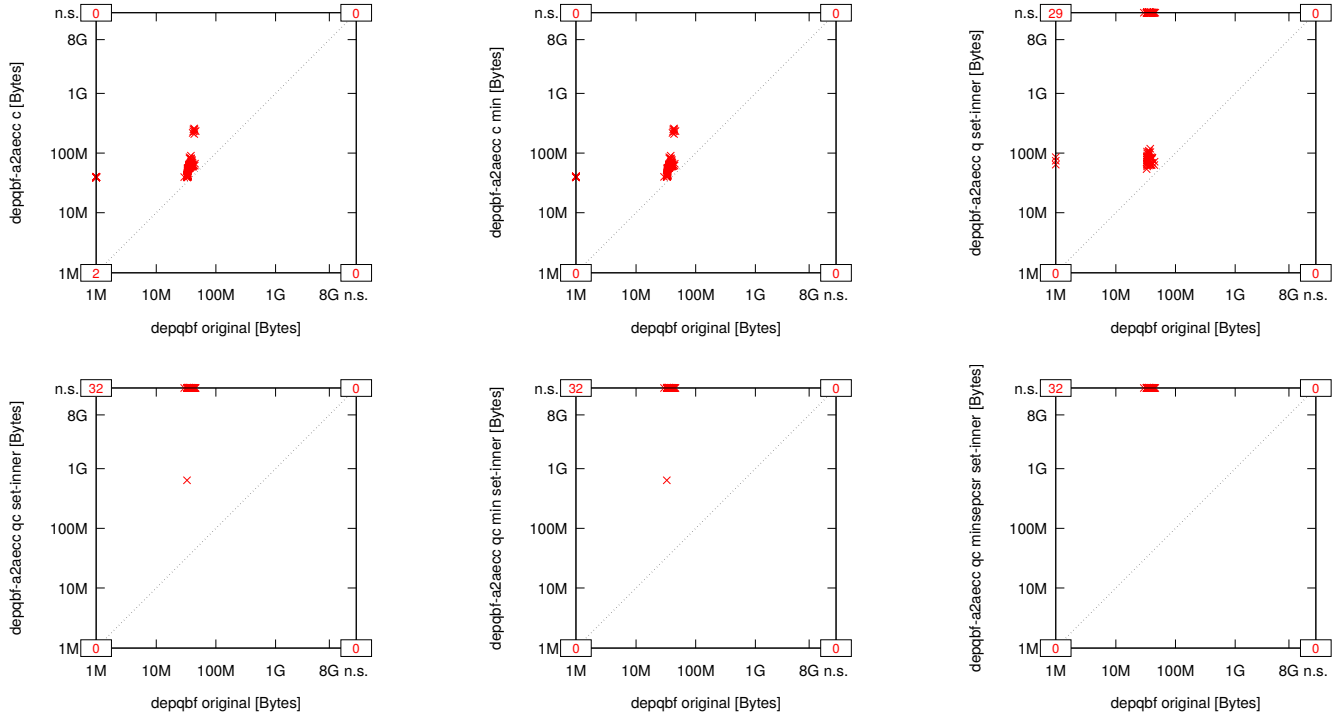


Fig. 1112: Suite Seidl ($n = 194$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

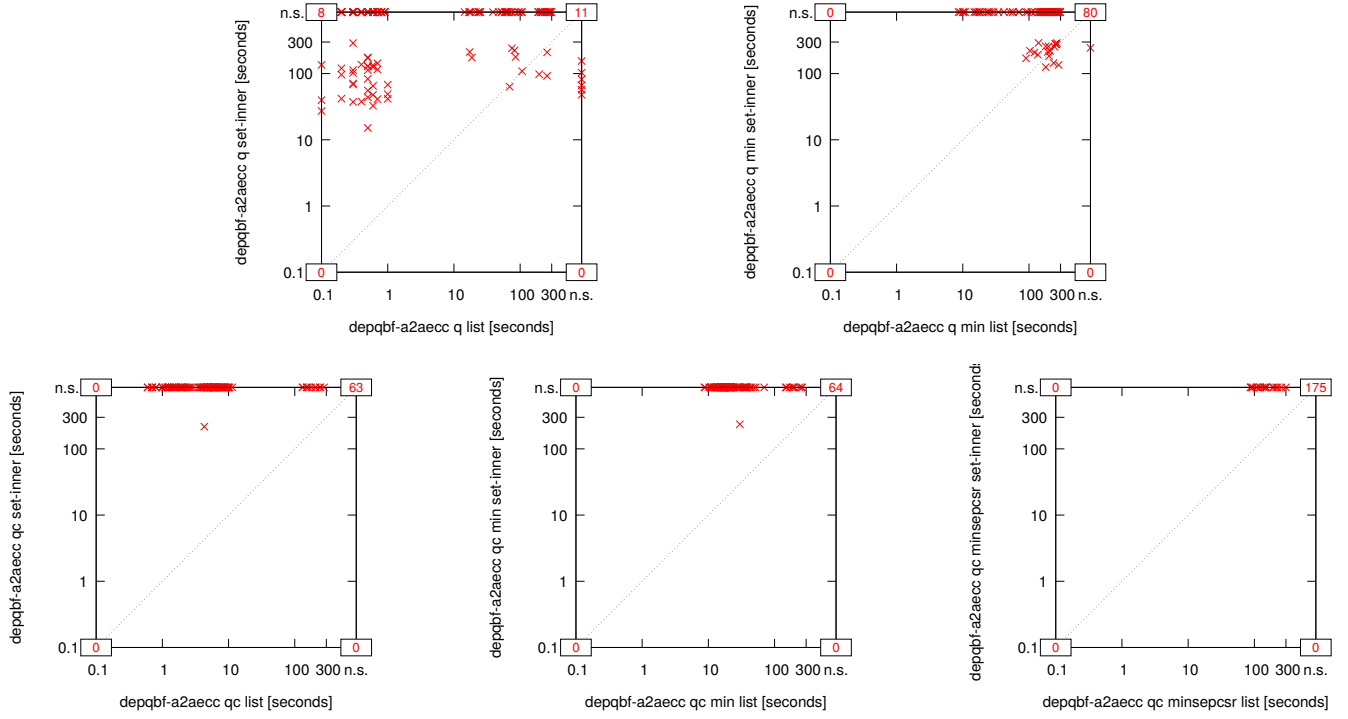


Fig. 1113: Suite Seidl ($n = 194$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

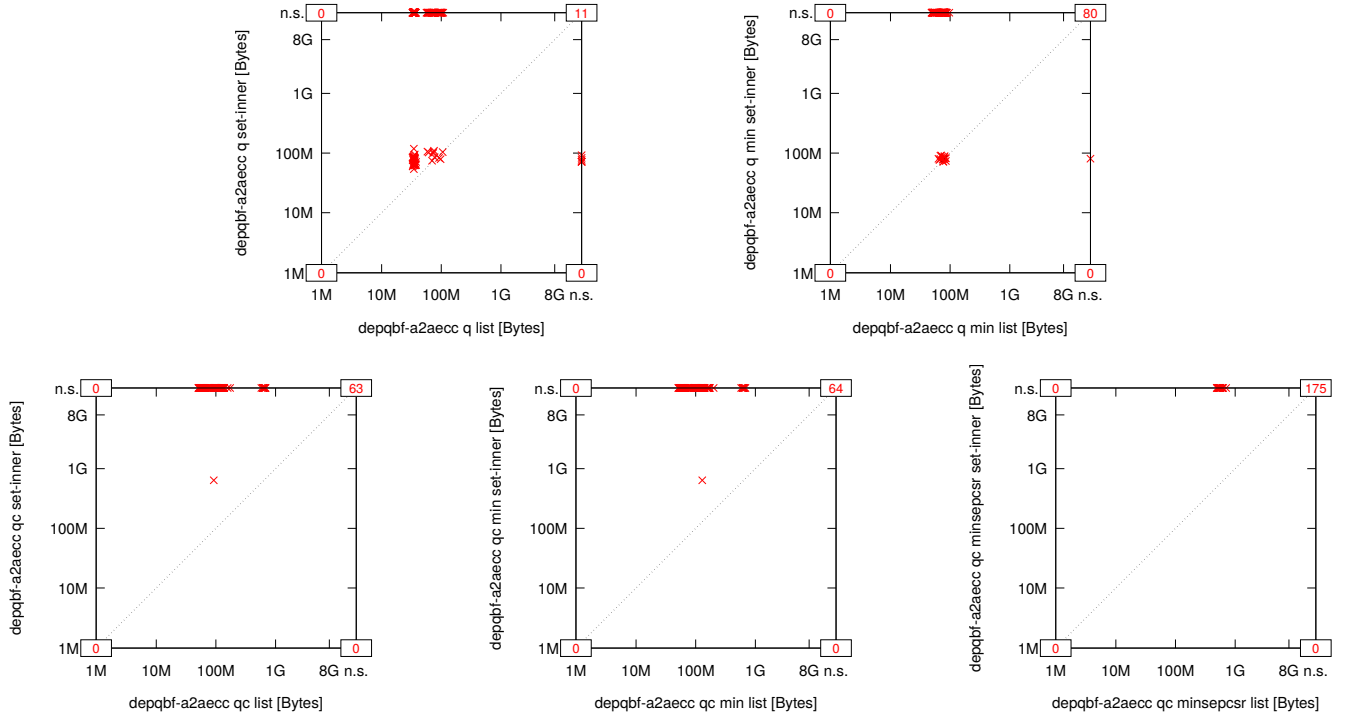


Fig. 1114: Suite Seidl ($n = 194$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

44) Tacchella ($n = 122$):

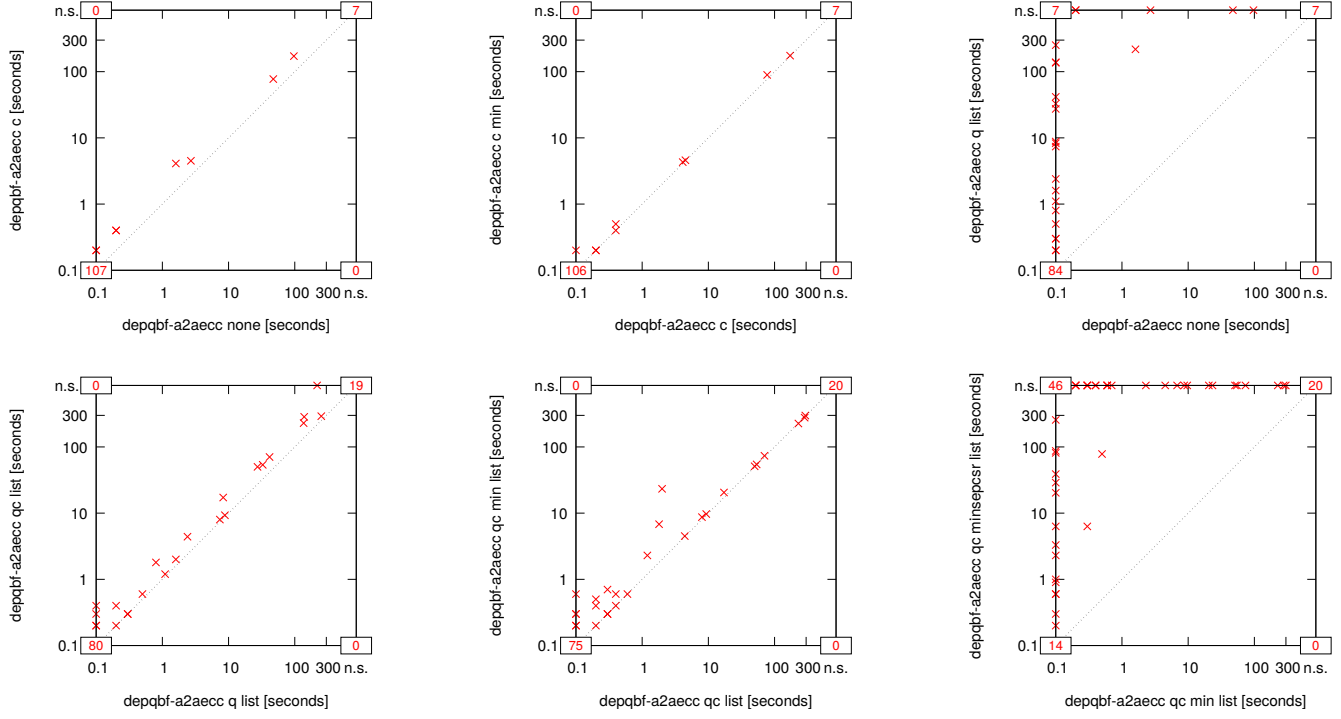


Fig. 1115: Suite Tacchella ($n = 122$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

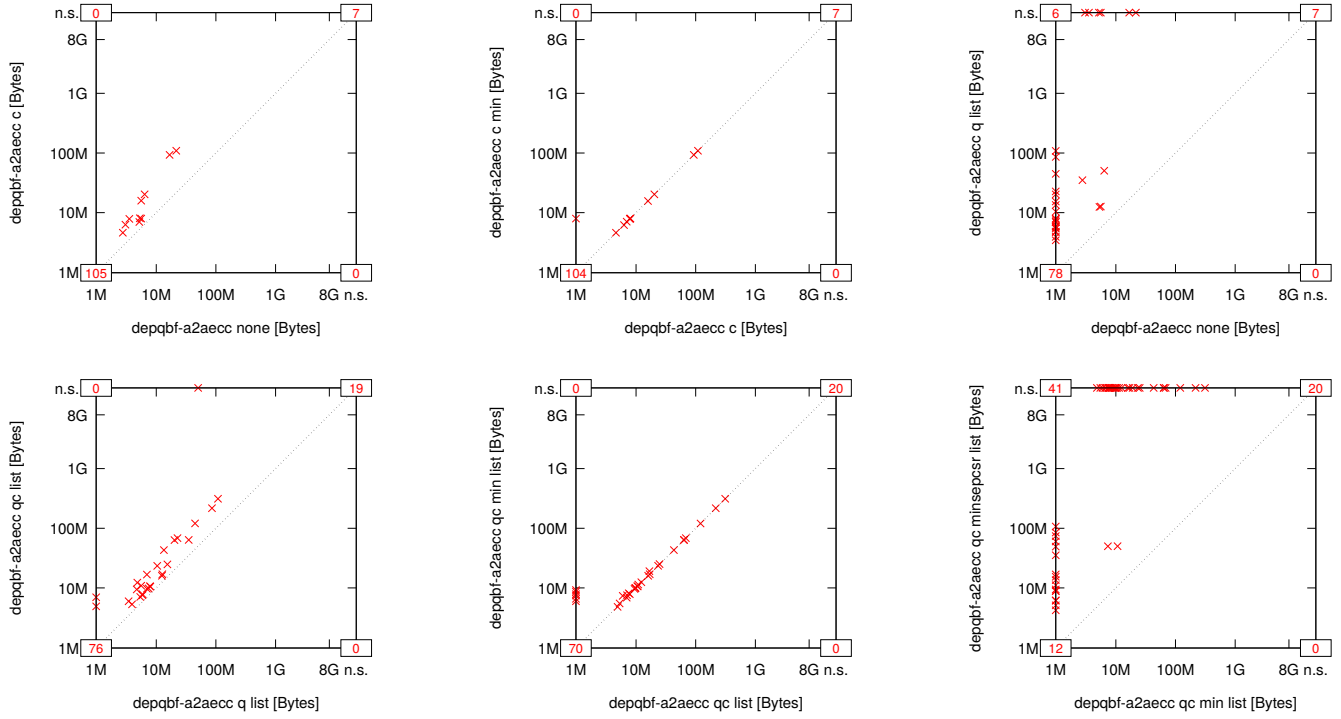


Fig. 1116: Suite Tacchella ($n = 122$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

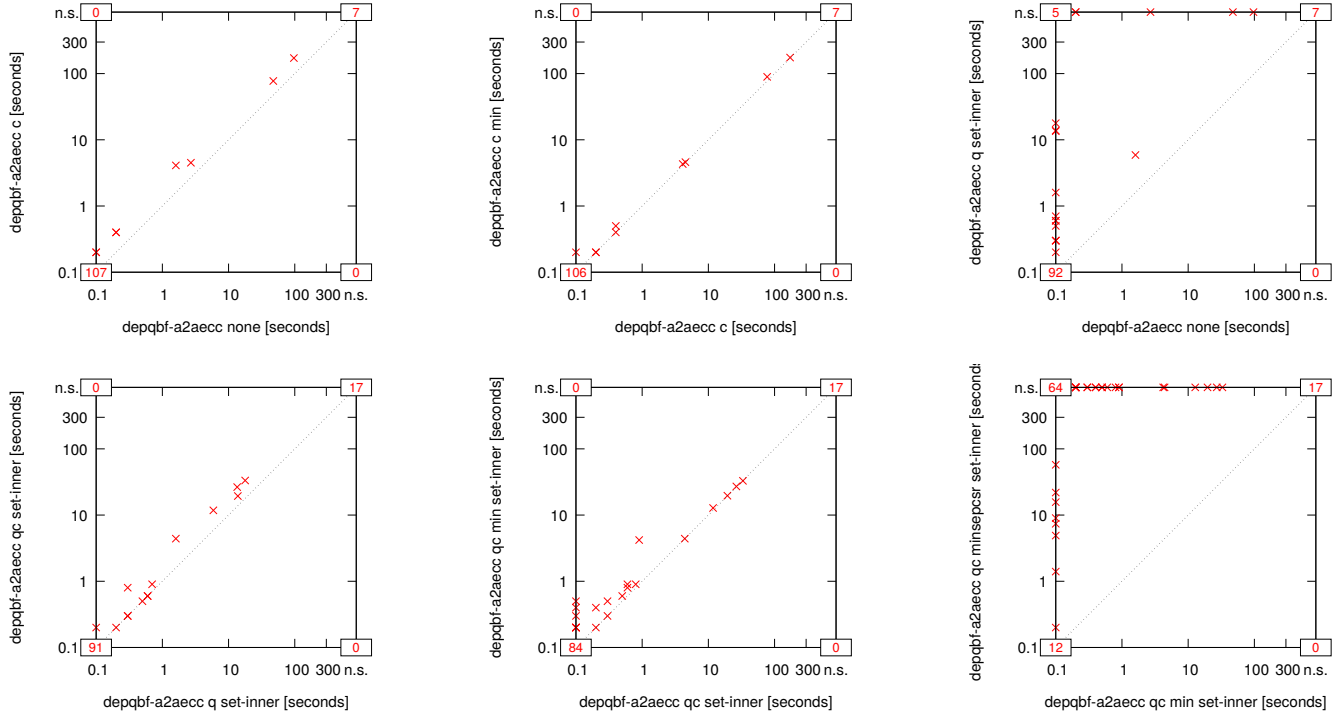


Fig. 1117: Suite Tacchella ($n = 122$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

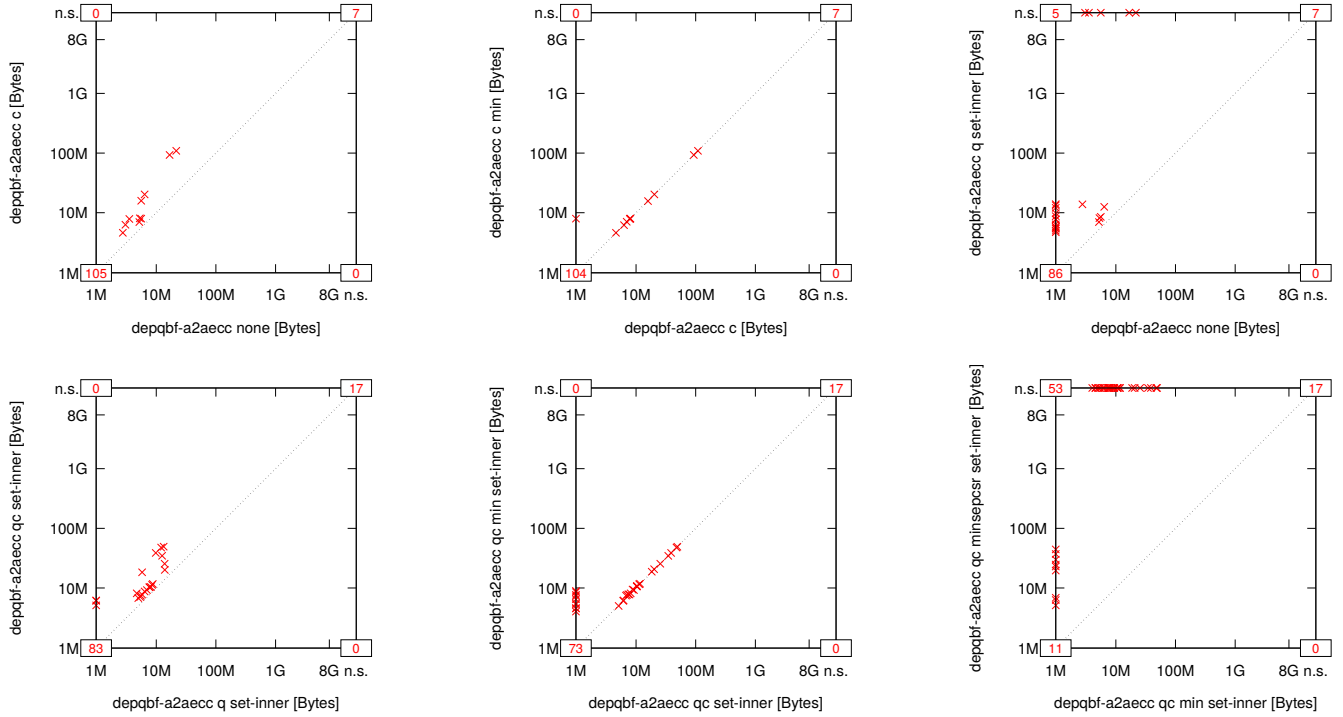


Fig. 1118: Suite Tacchella ($n = 122$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

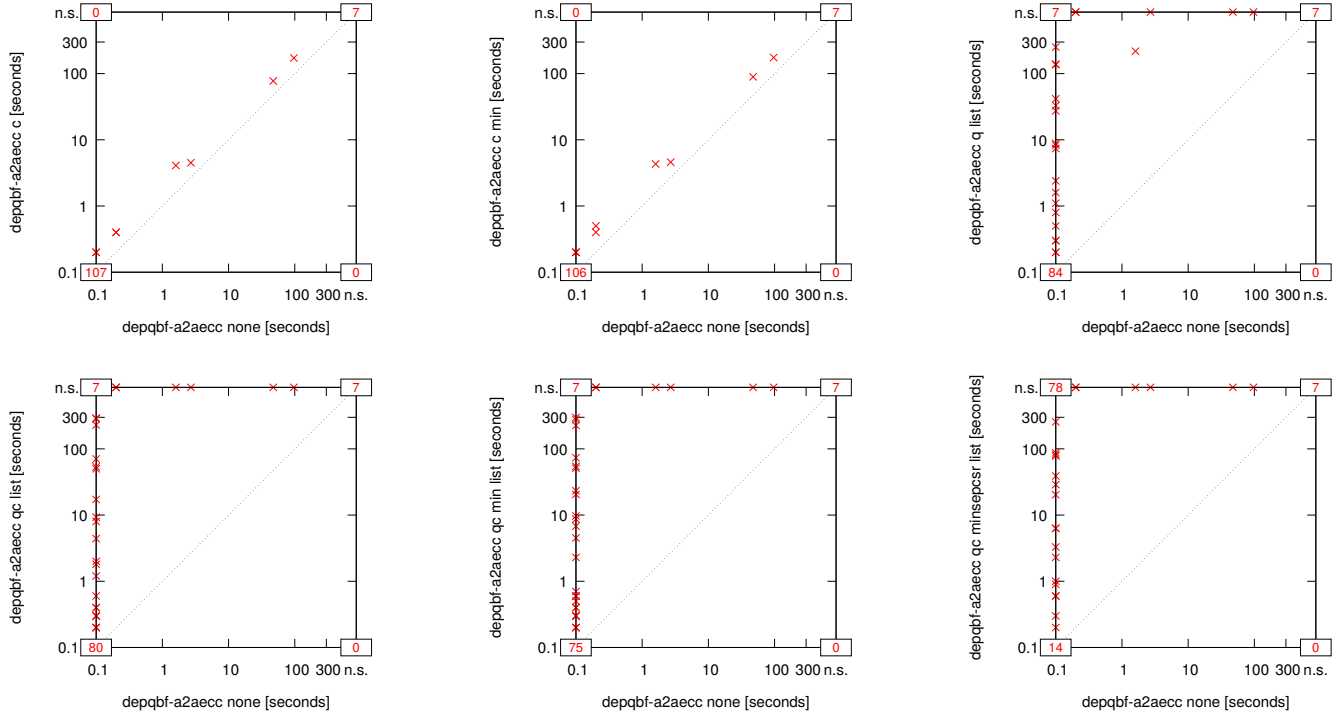


Fig. 1119: Suite Tacchella ($n = 122$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

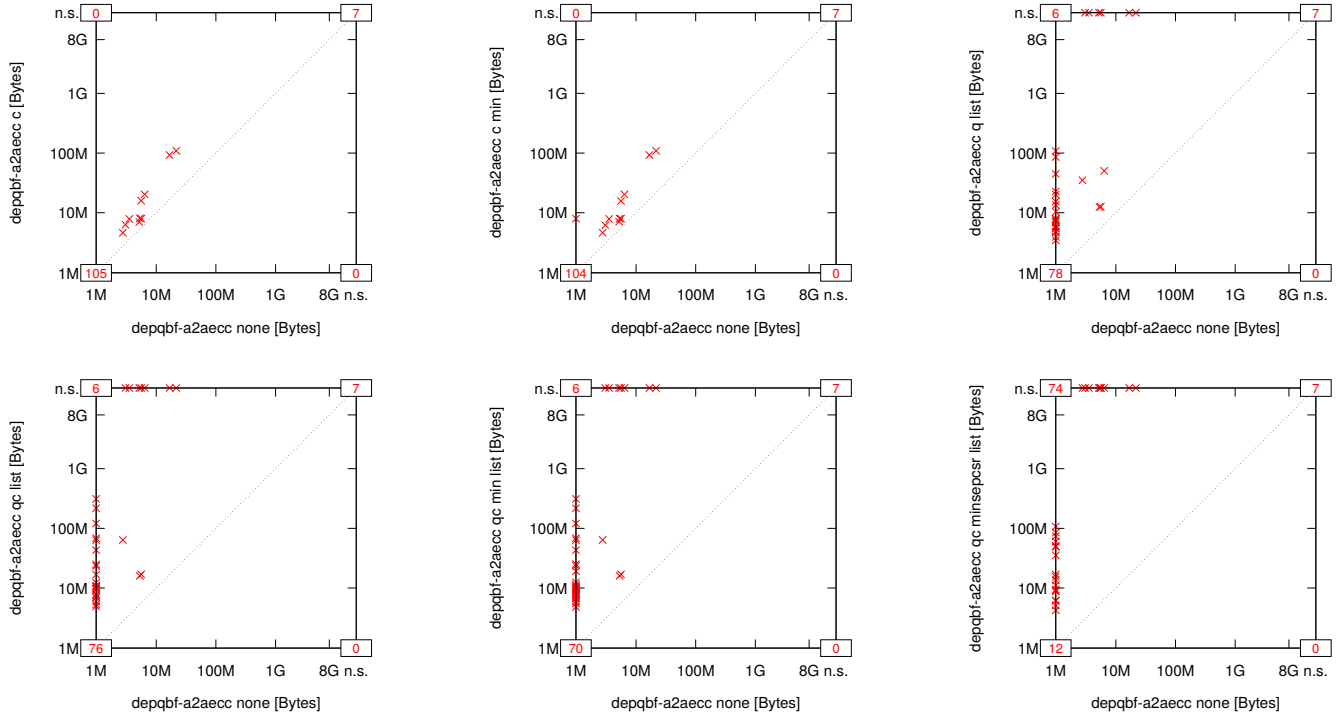


Fig. 1120: Suite Tacchella ($n = 122$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

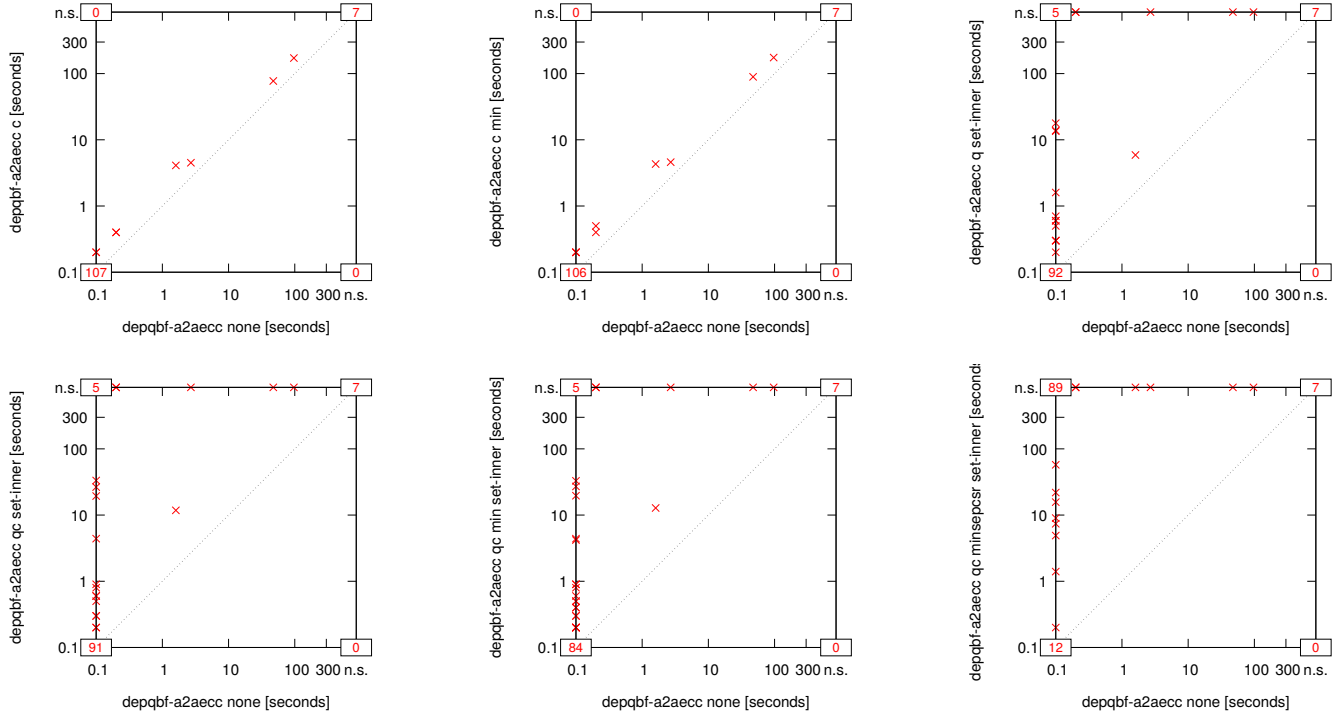


Fig. 1121: Suite Tacchella ($n = 122$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

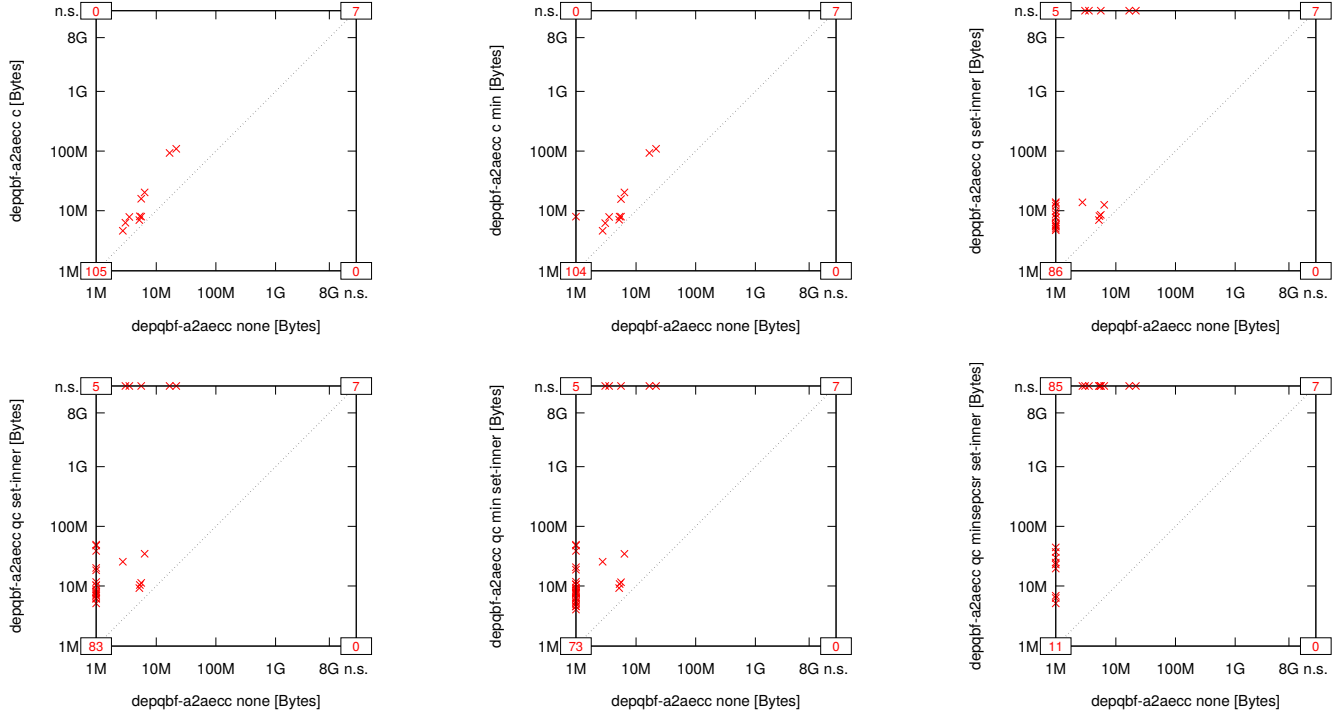


Fig. 1122: Suite Tacchella ($n = 122$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

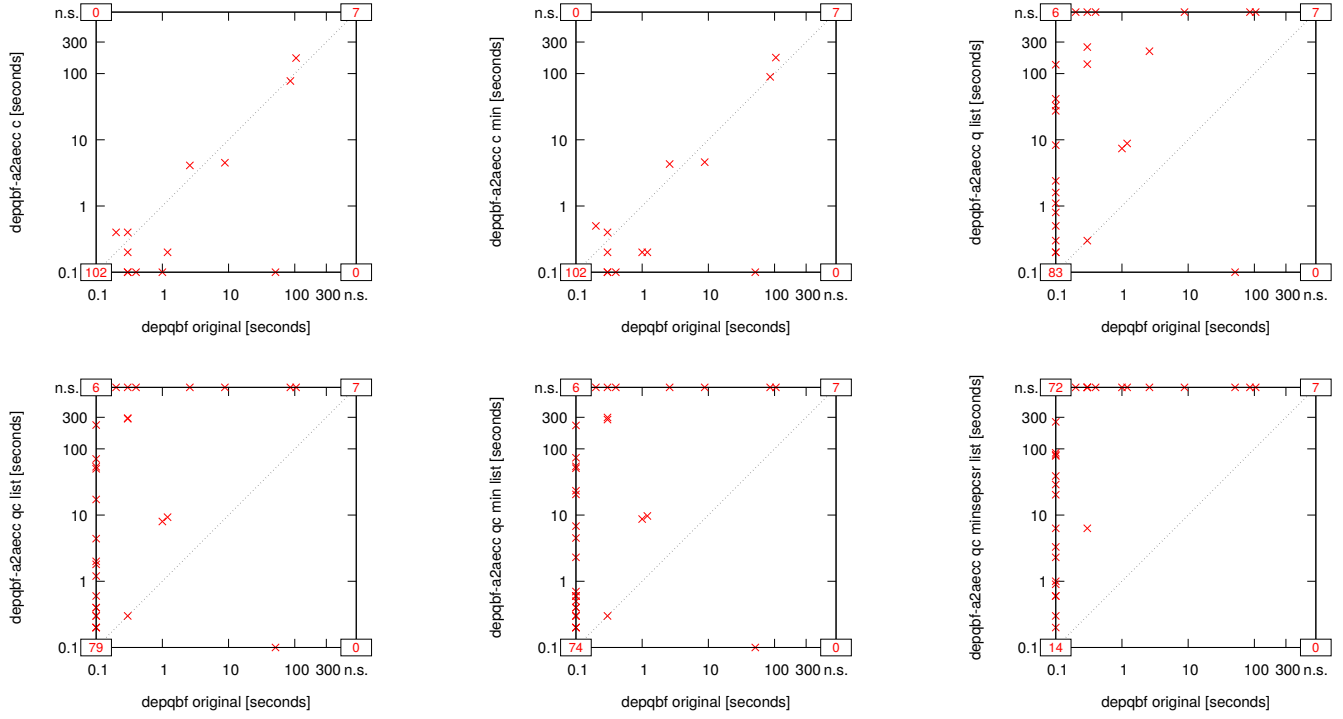


Fig. 1123: Suite Tacchella ($n = 122$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

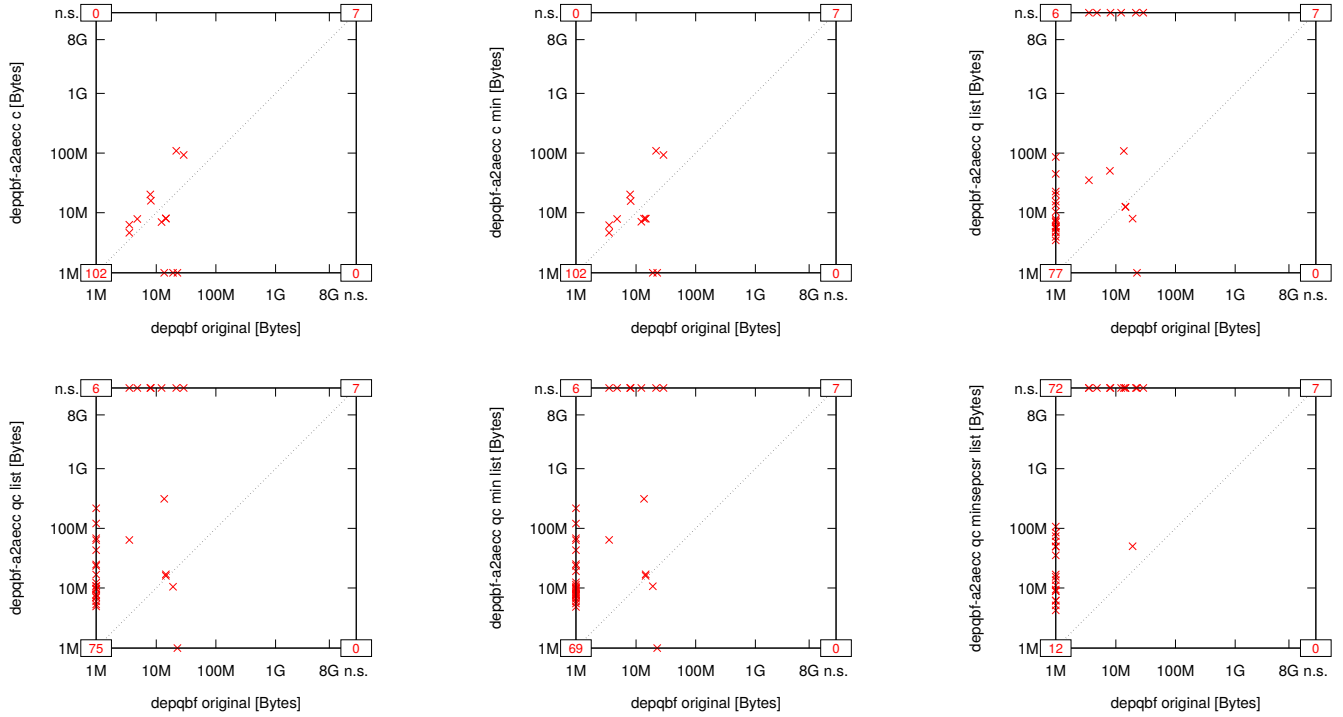


Fig. 1124: Suite Tacchella ($n = 122$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

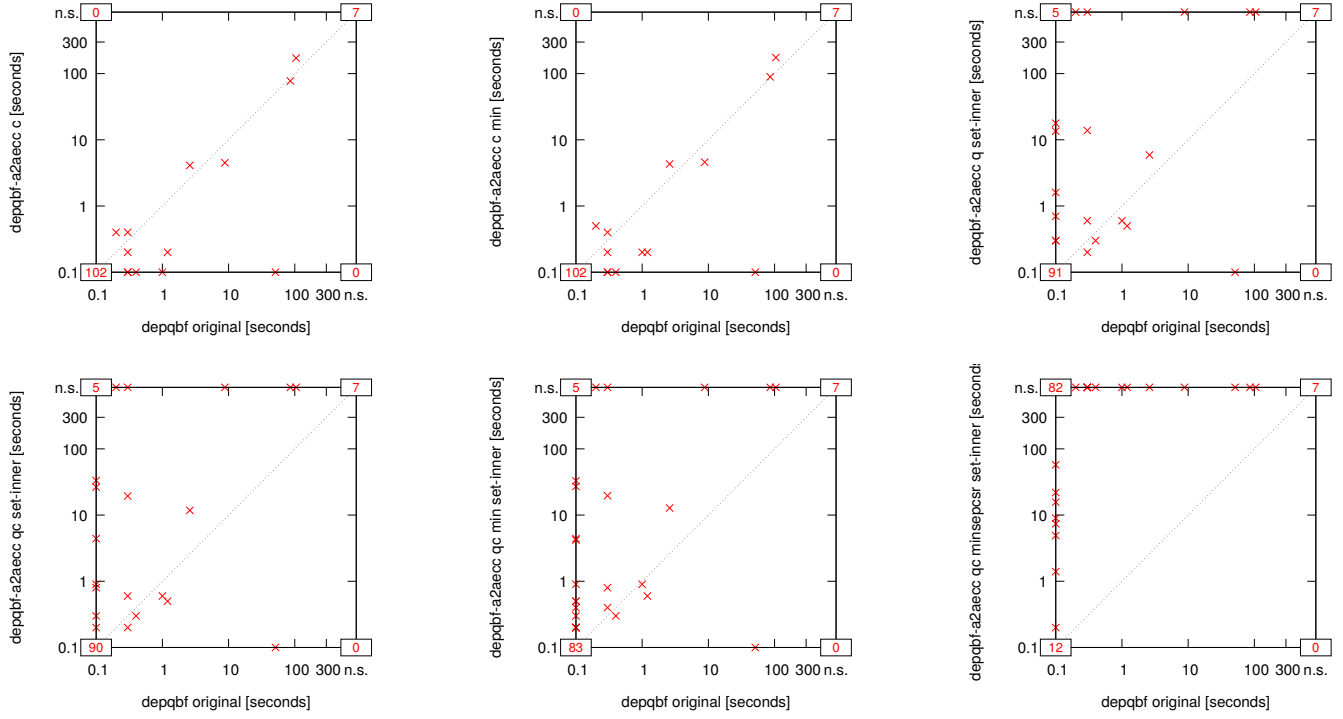


Fig. 1125: Suite Tacchella ($n = 122$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

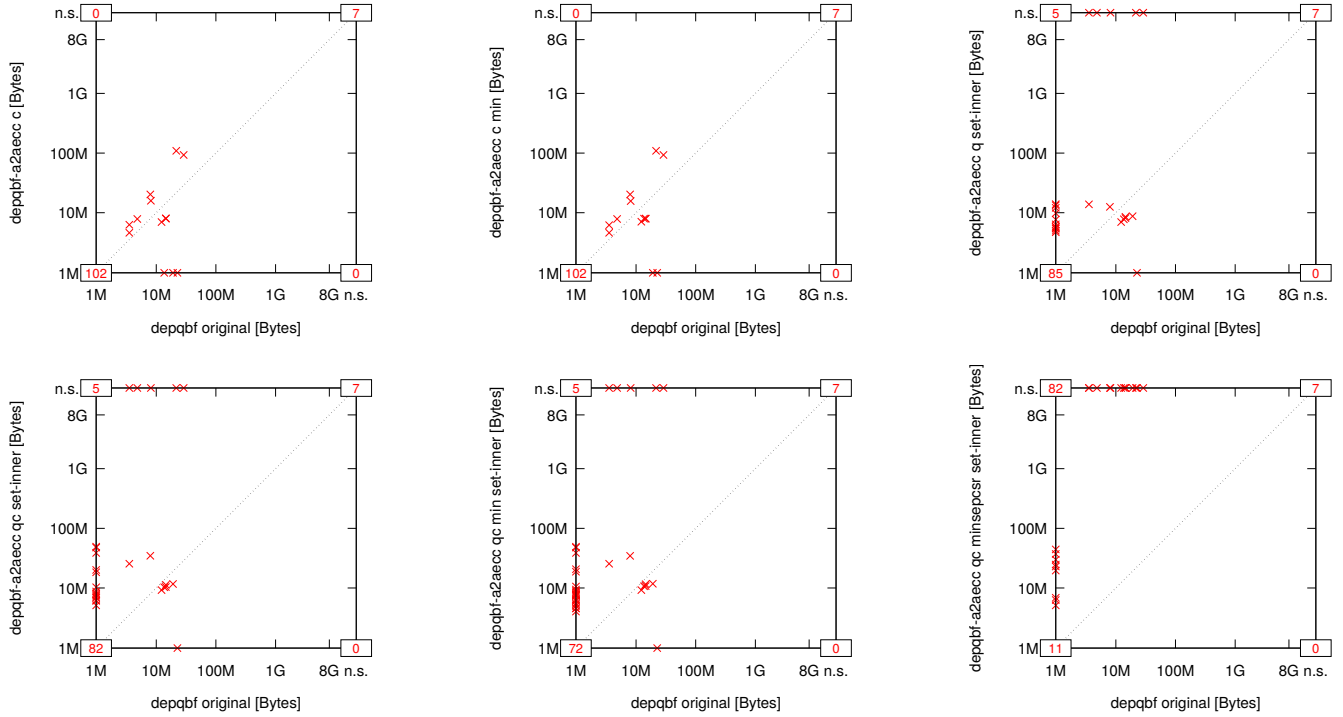


Fig. 1126: Suite Tacchella ($n = 122$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

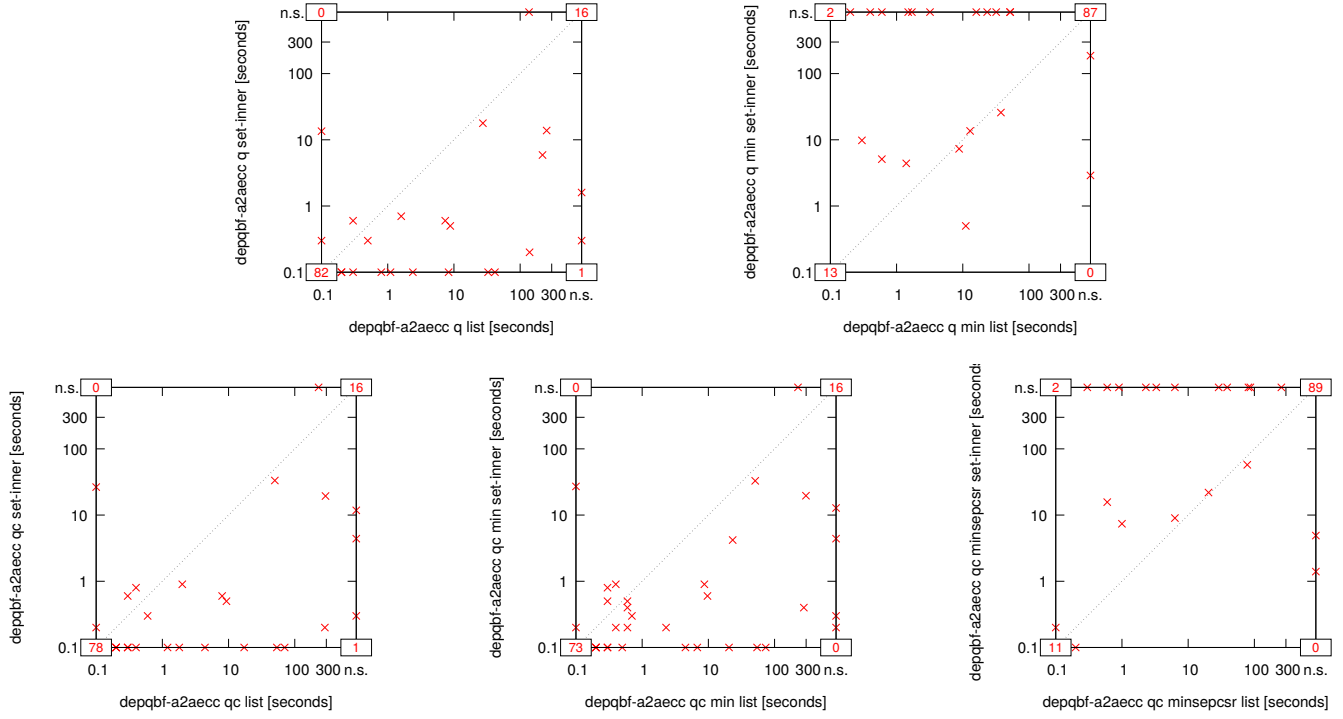


Fig. 1127: Suite Tacchella ($n = 122$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

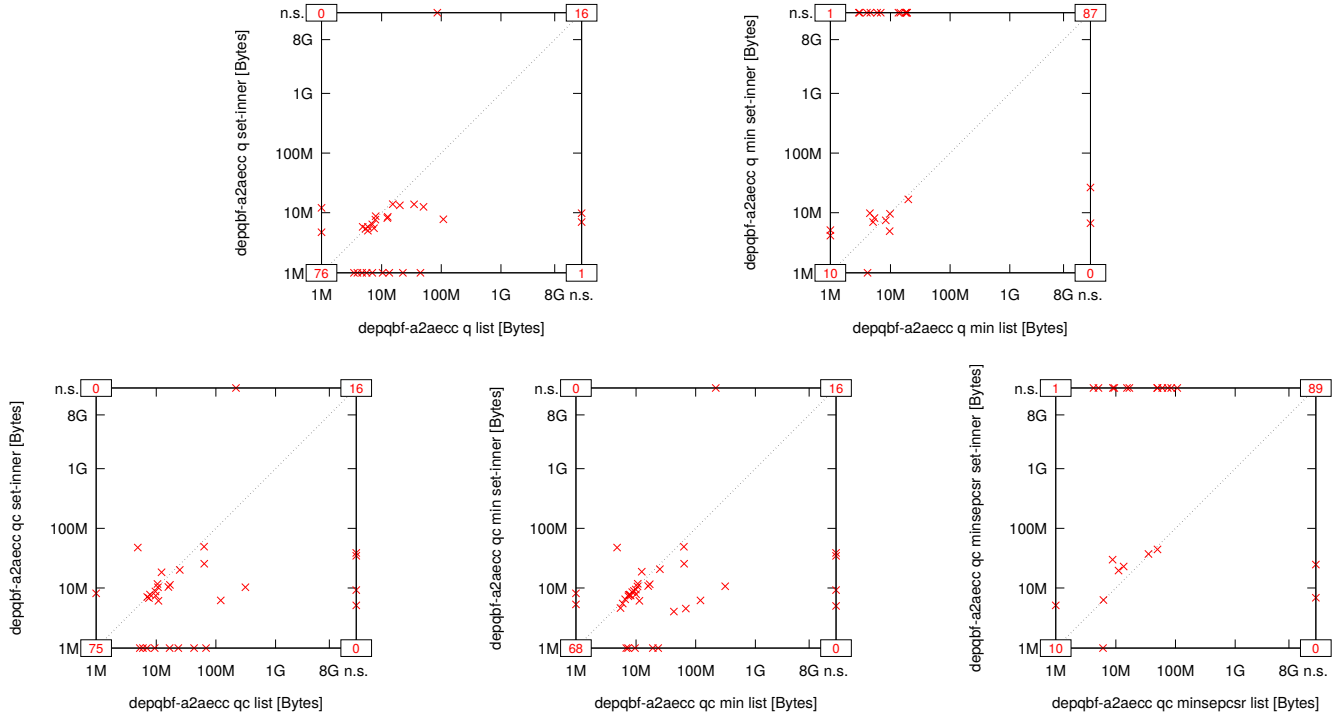


Fig. 1128: Suite Tacchella ($n = 122$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

45) *Tentrup* ($n = 17$):

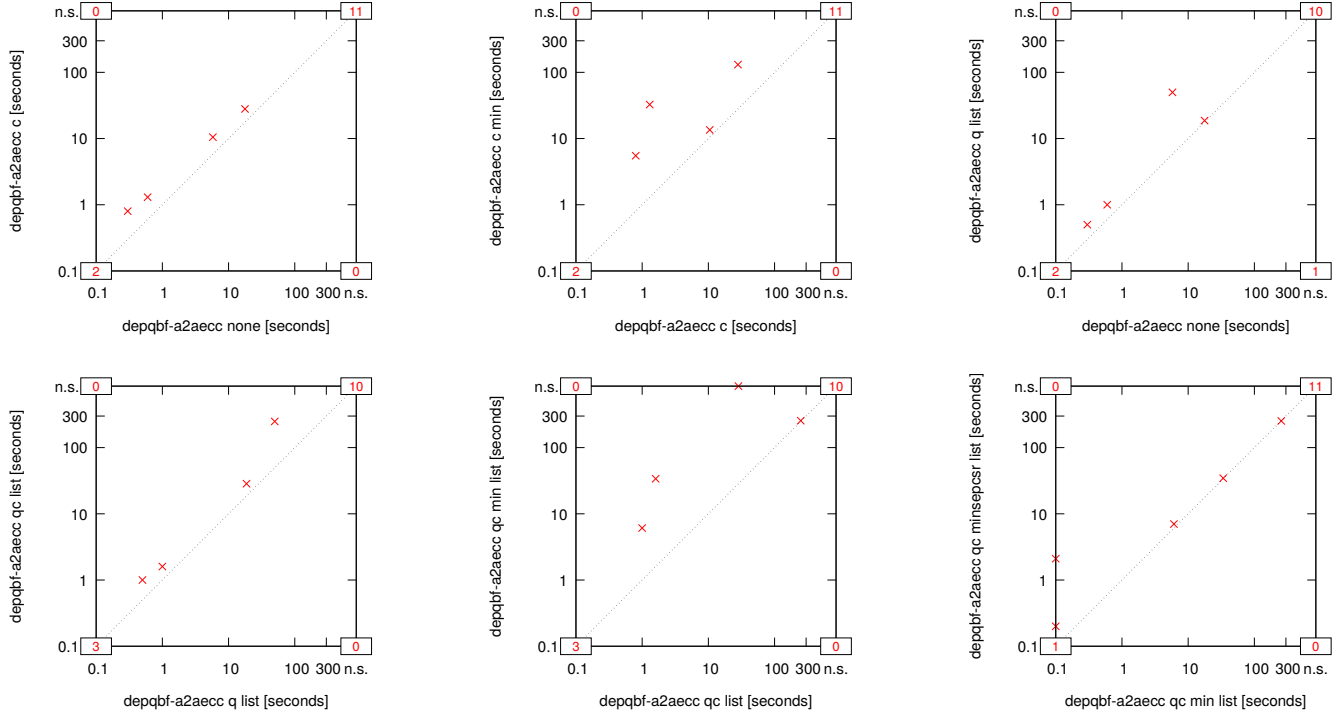


Fig. 1129: Suite Tentrup ($n = 17$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

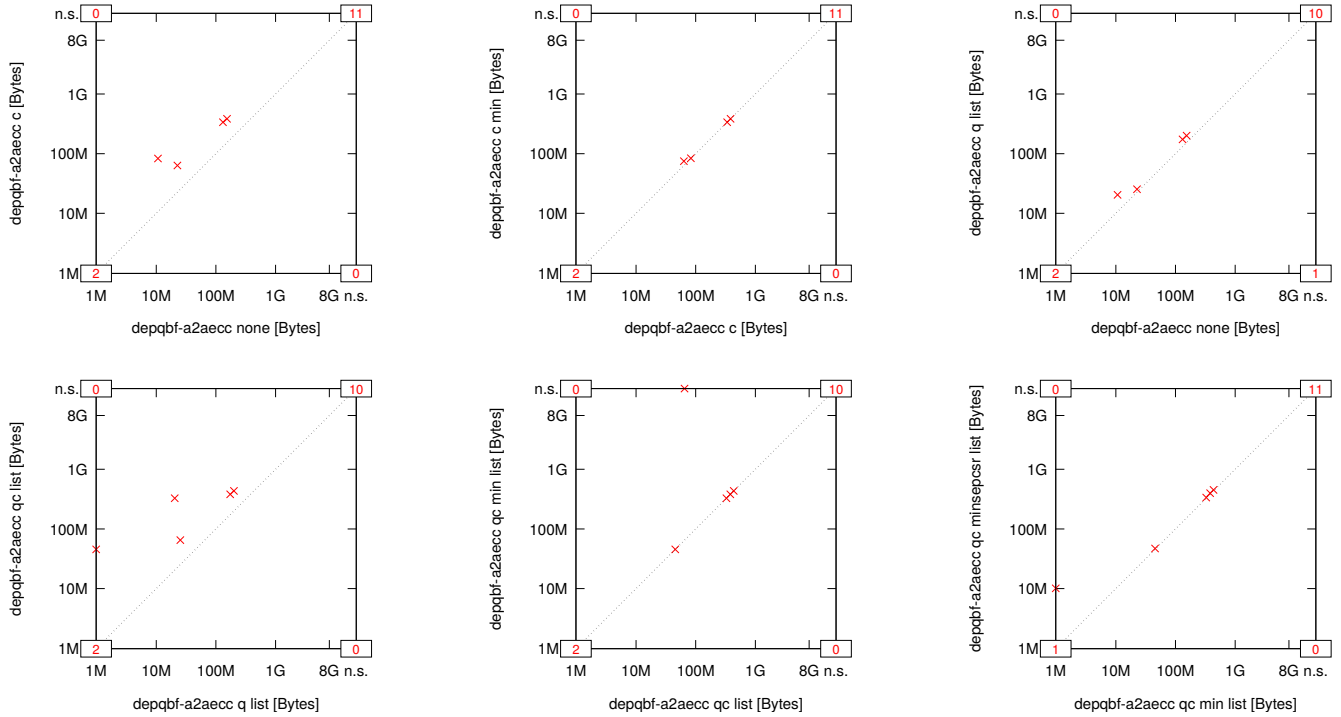


Fig. 1130: Suite Tentrup ($n = 17$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

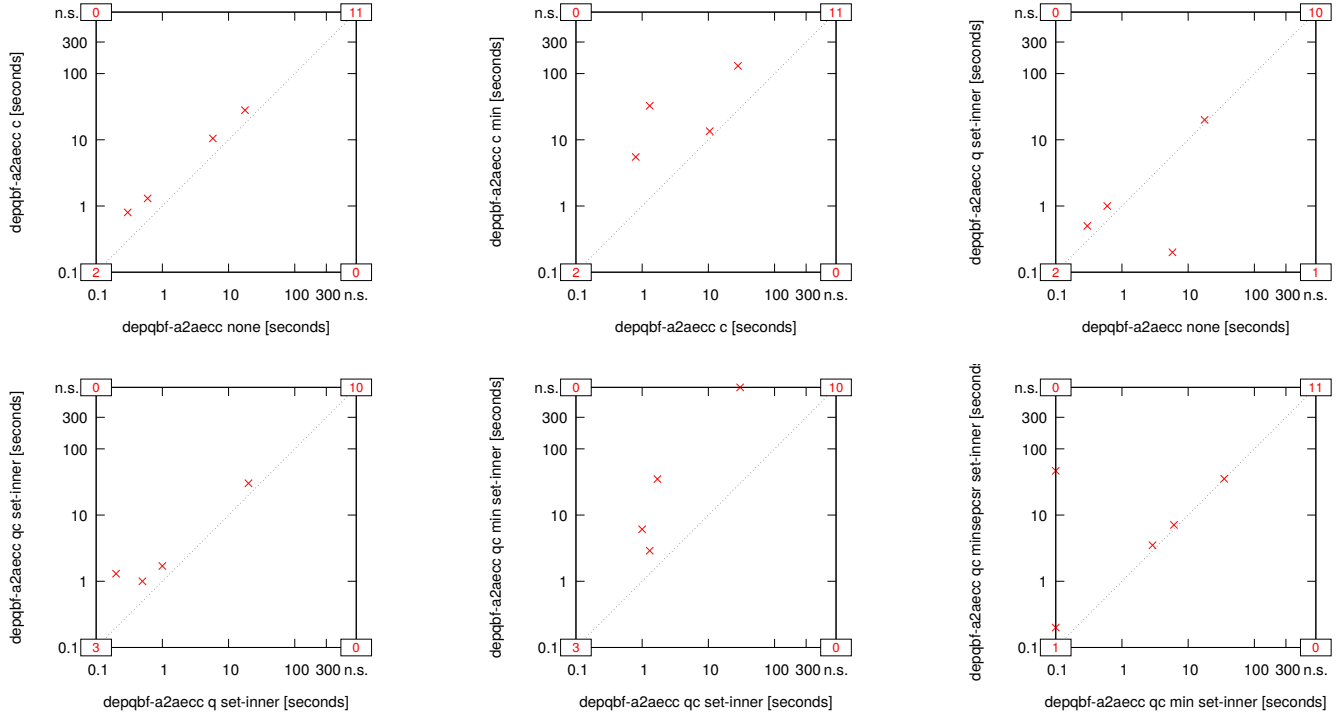


Fig. 1131: Suite Tentrup ($n = 17$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

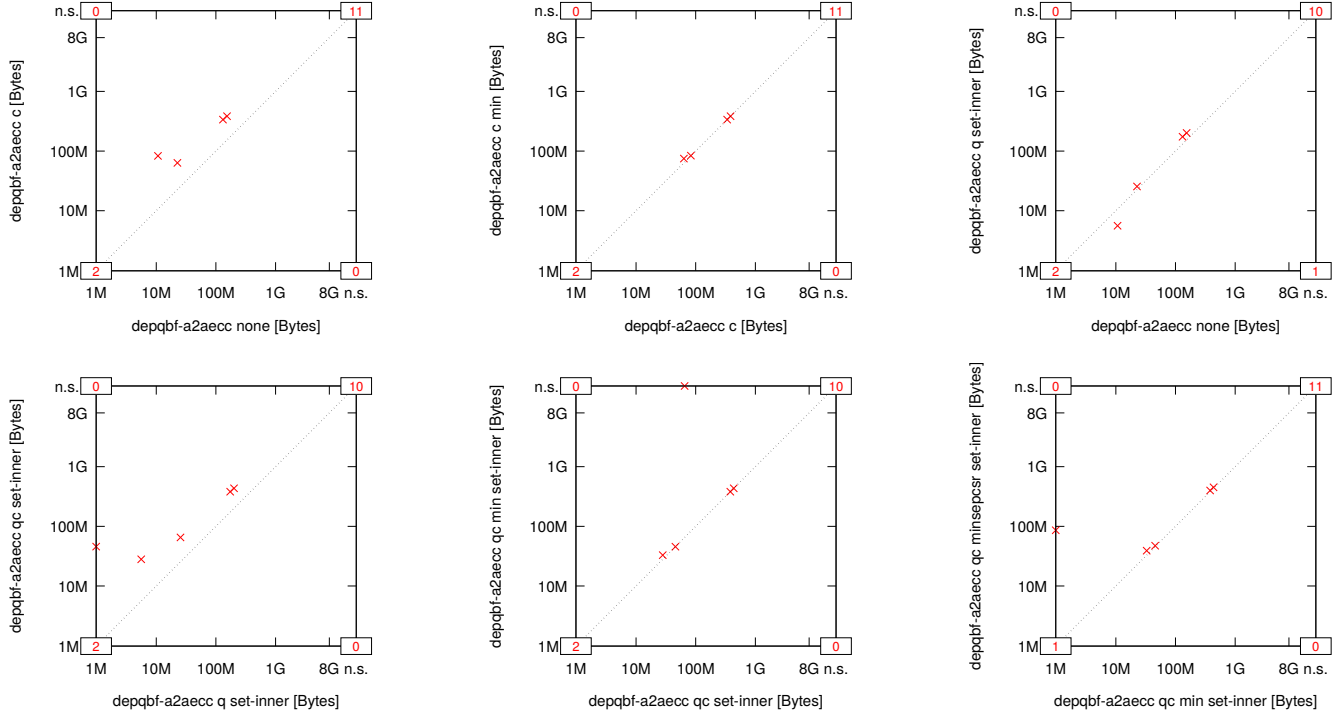


Fig. 1132: Suite Tentrup ($n = 17$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

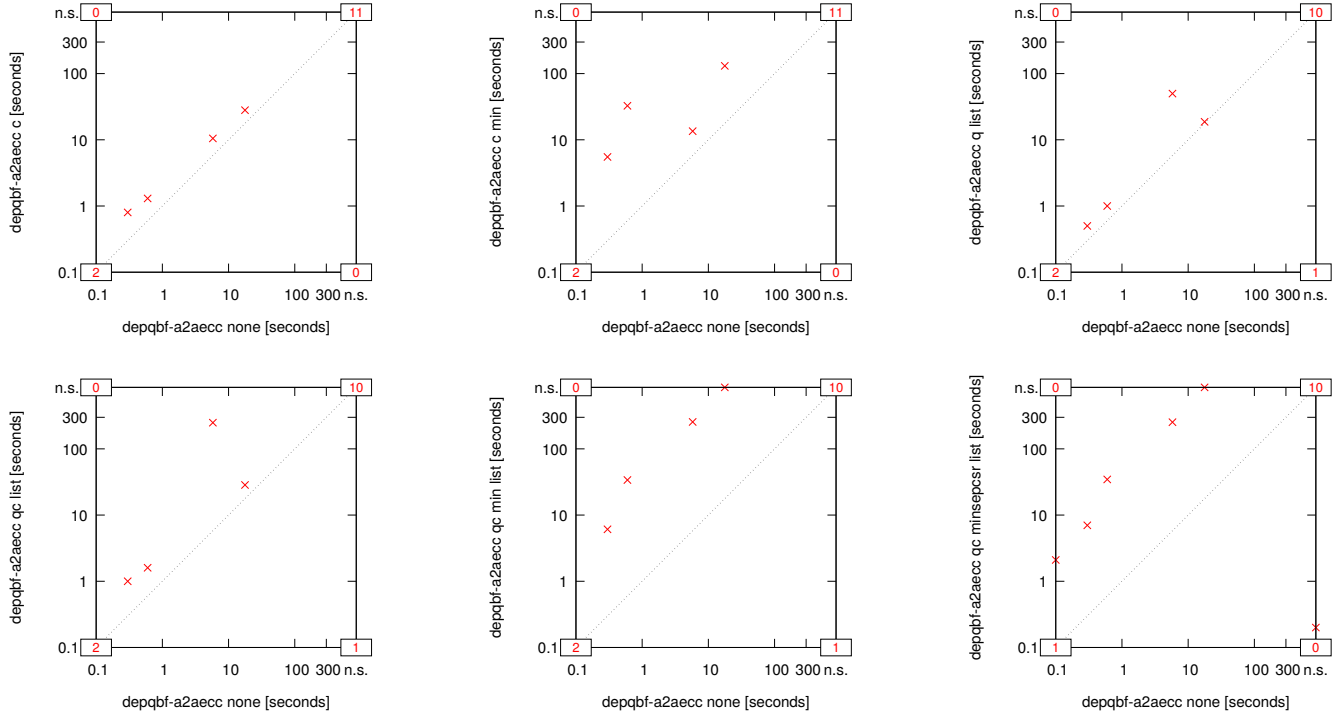


Fig. 1133: Suite Tentrup ($n = 17$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

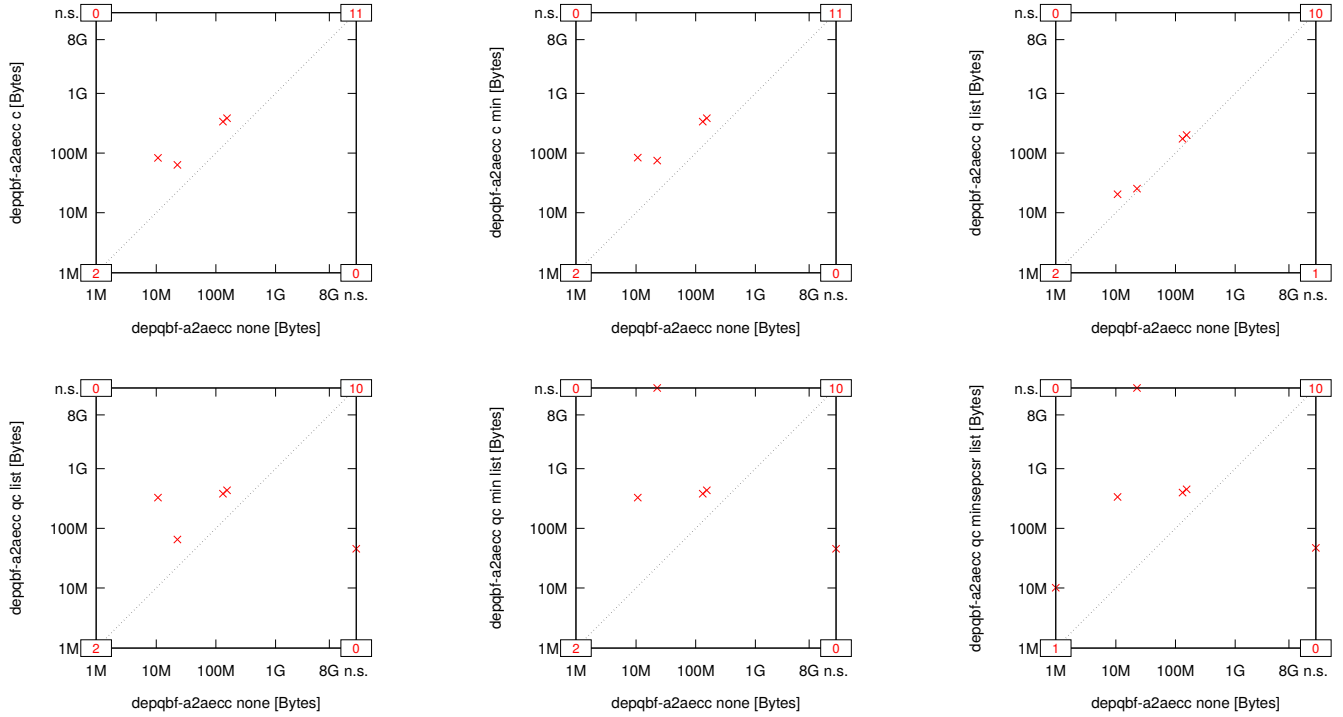


Fig. 1134: Suite Tentrup ($n = 17$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

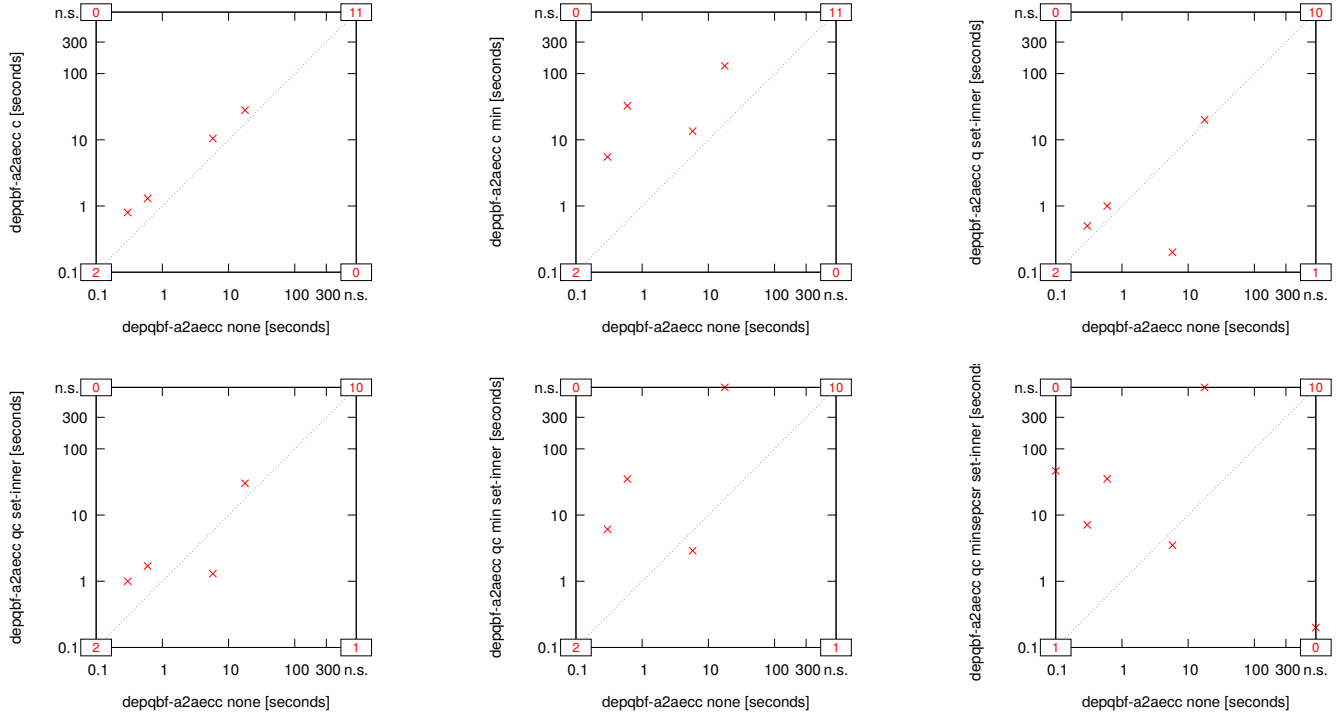


Fig. 1135: Suite Tentrup ($n = 17$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

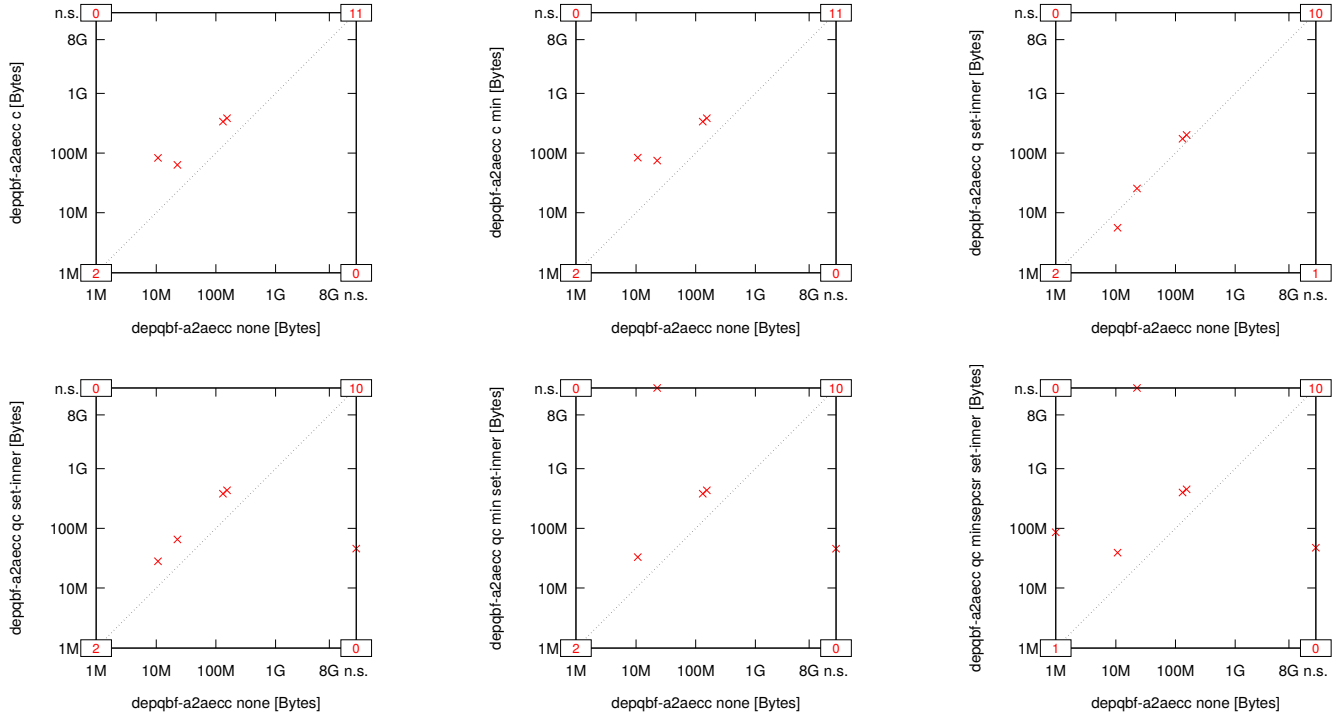


Fig. 1136: Suite Tentrup ($n = 17$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

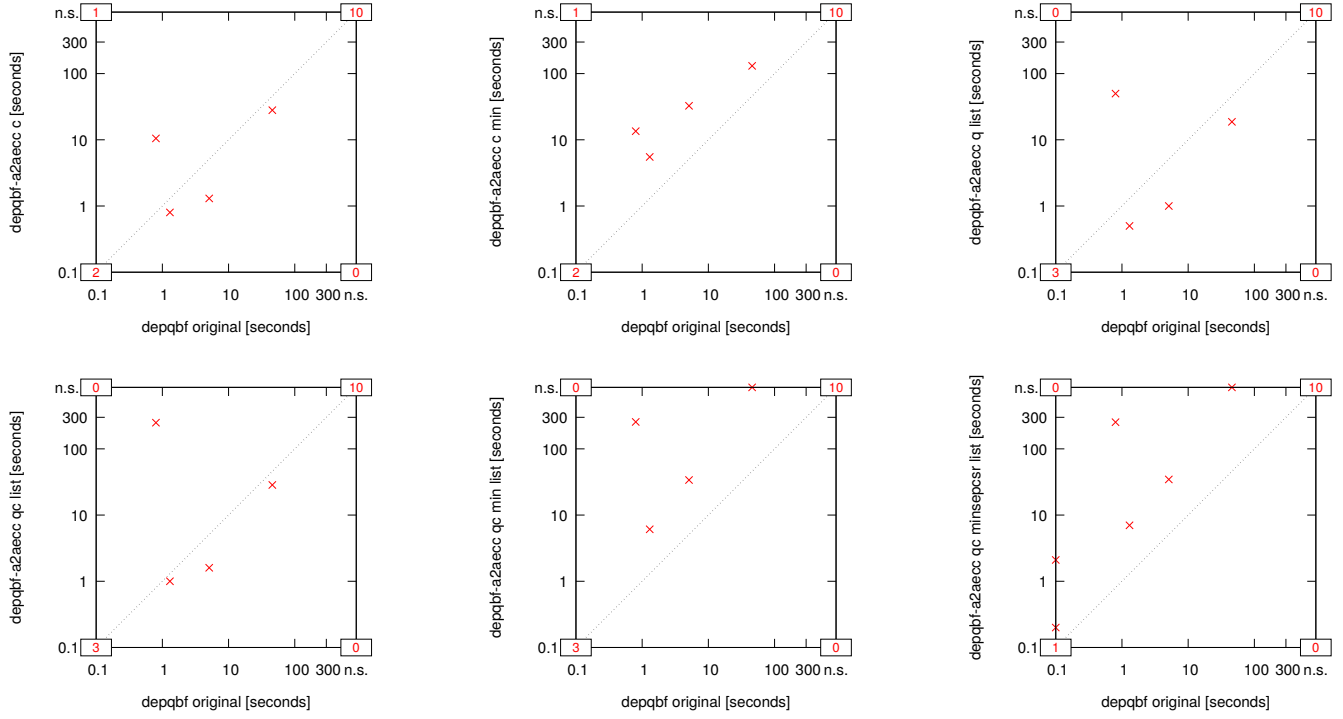


Fig. 1137: Suite Tentrup ($n = 17$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

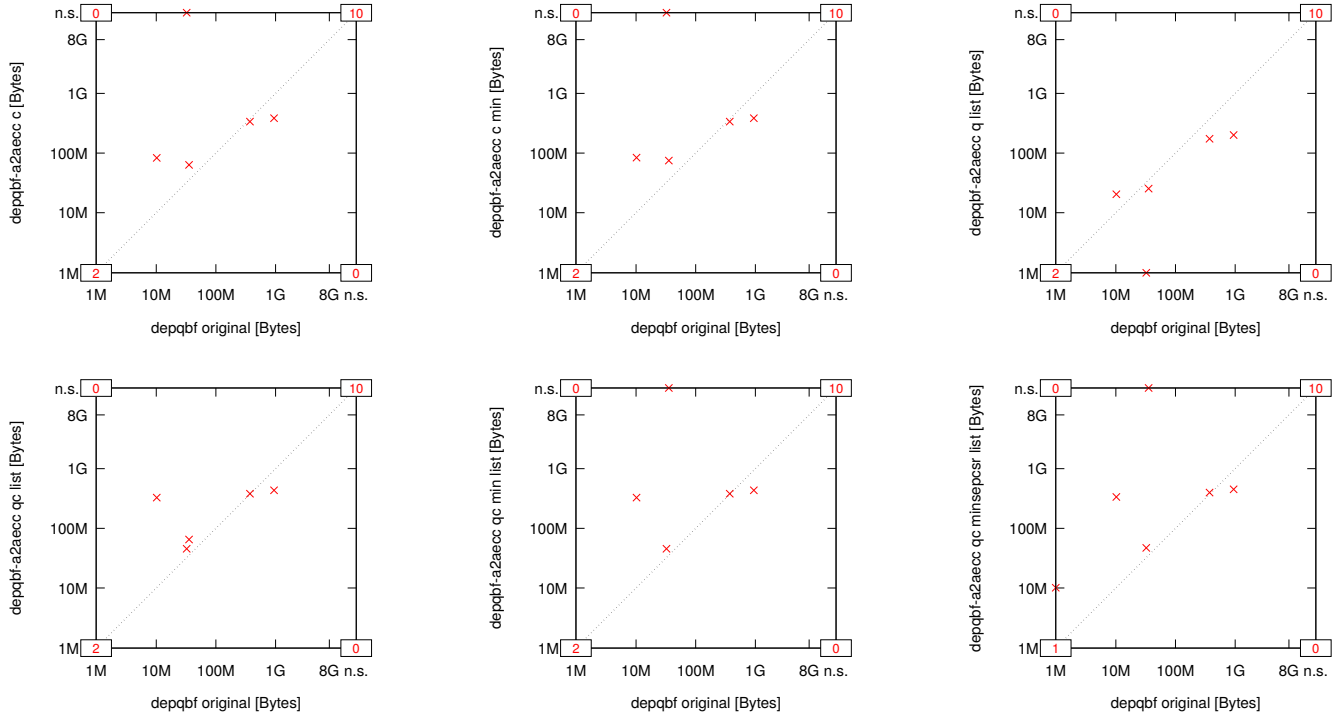


Fig. 1138: Suite Tentrup ($n = 17$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

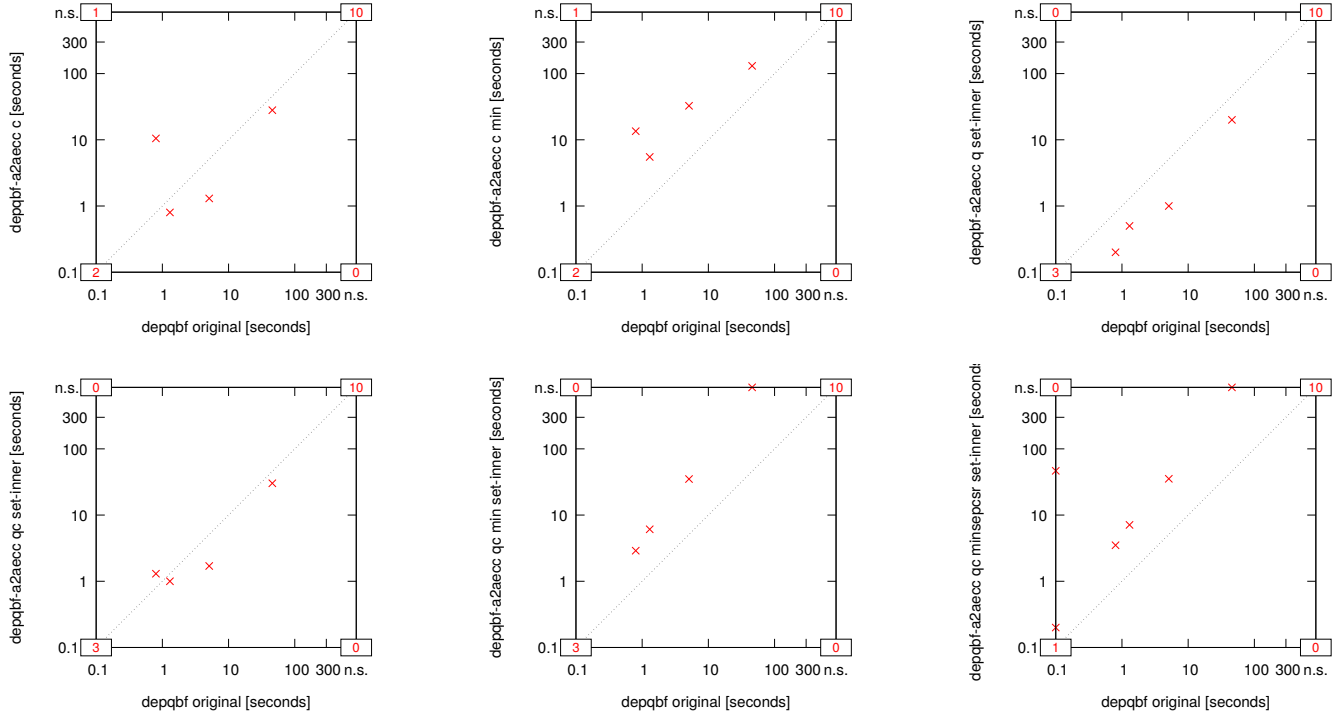


Fig. 1139: Suite Tentrup ($n = 17$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

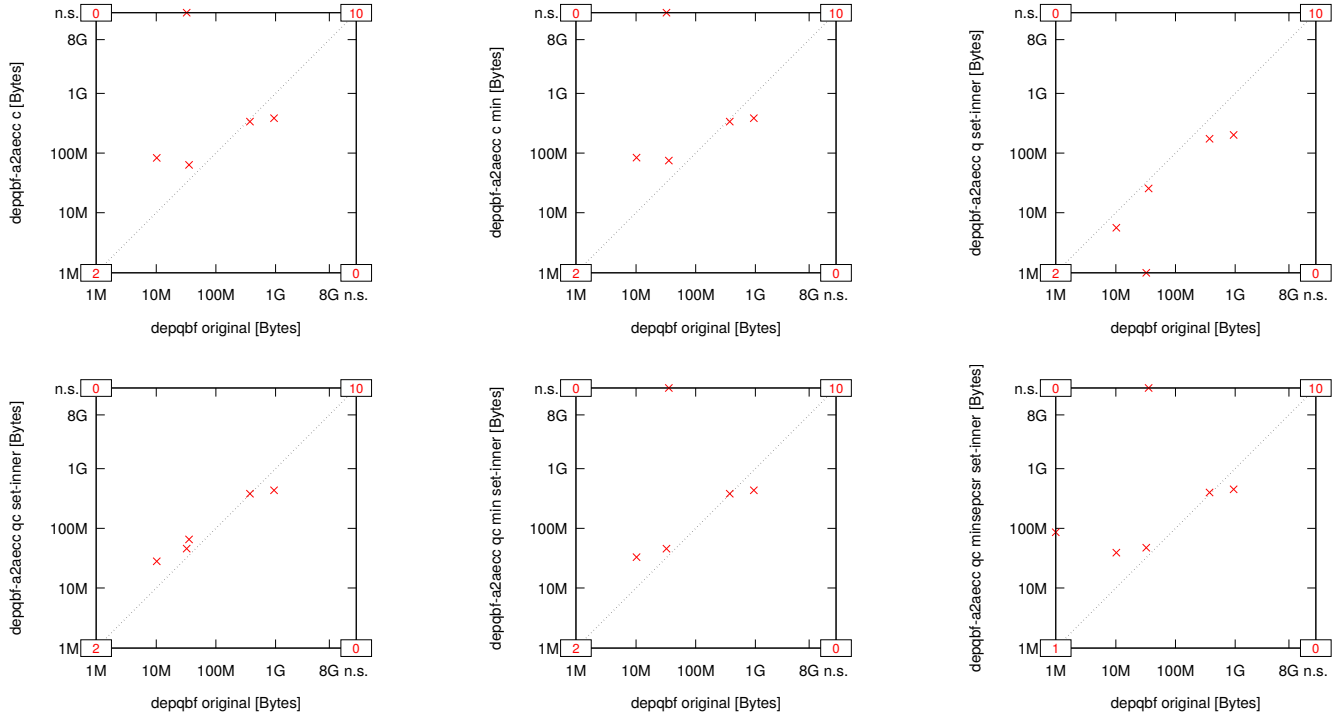


Fig. 1140: Suite Tentrup ($n = 17$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

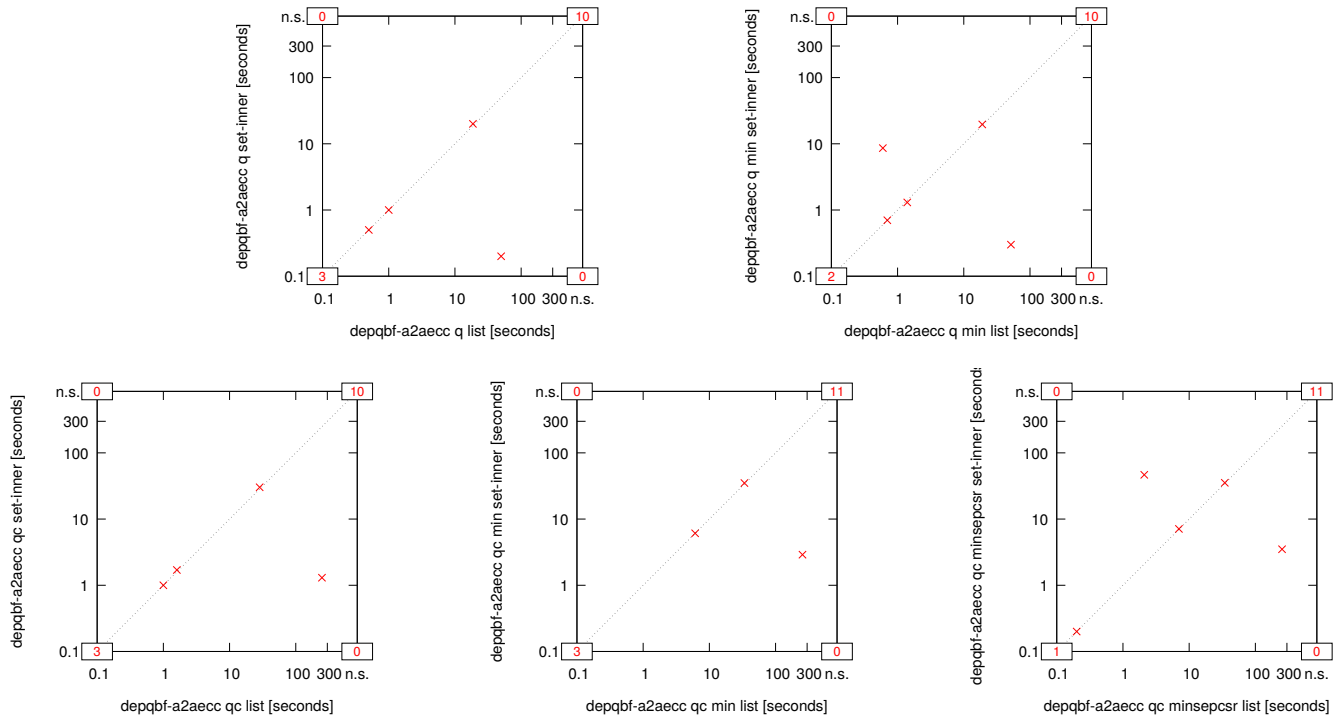


Fig. 1141: Suite Tentrup ($n = 17$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

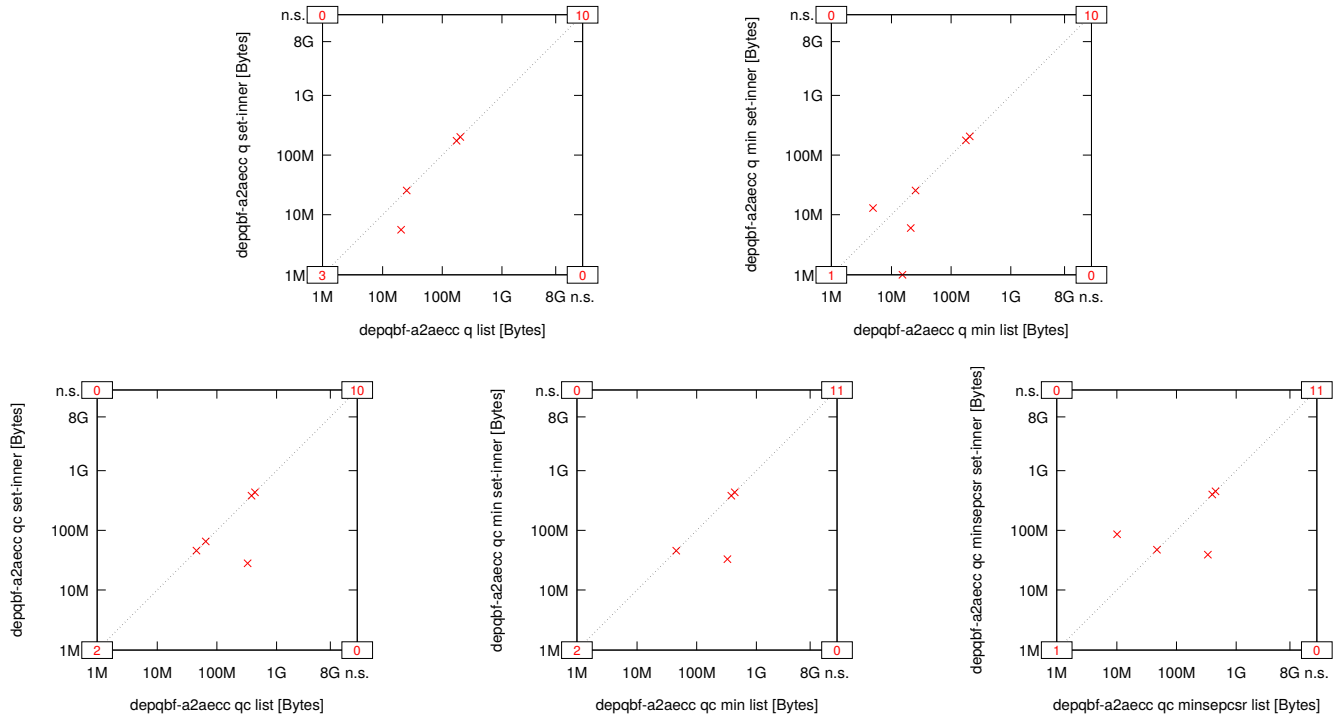


Fig. 1142: Suite Tentrup ($n = 17$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

46) Wintersteiger ($n = 38$):

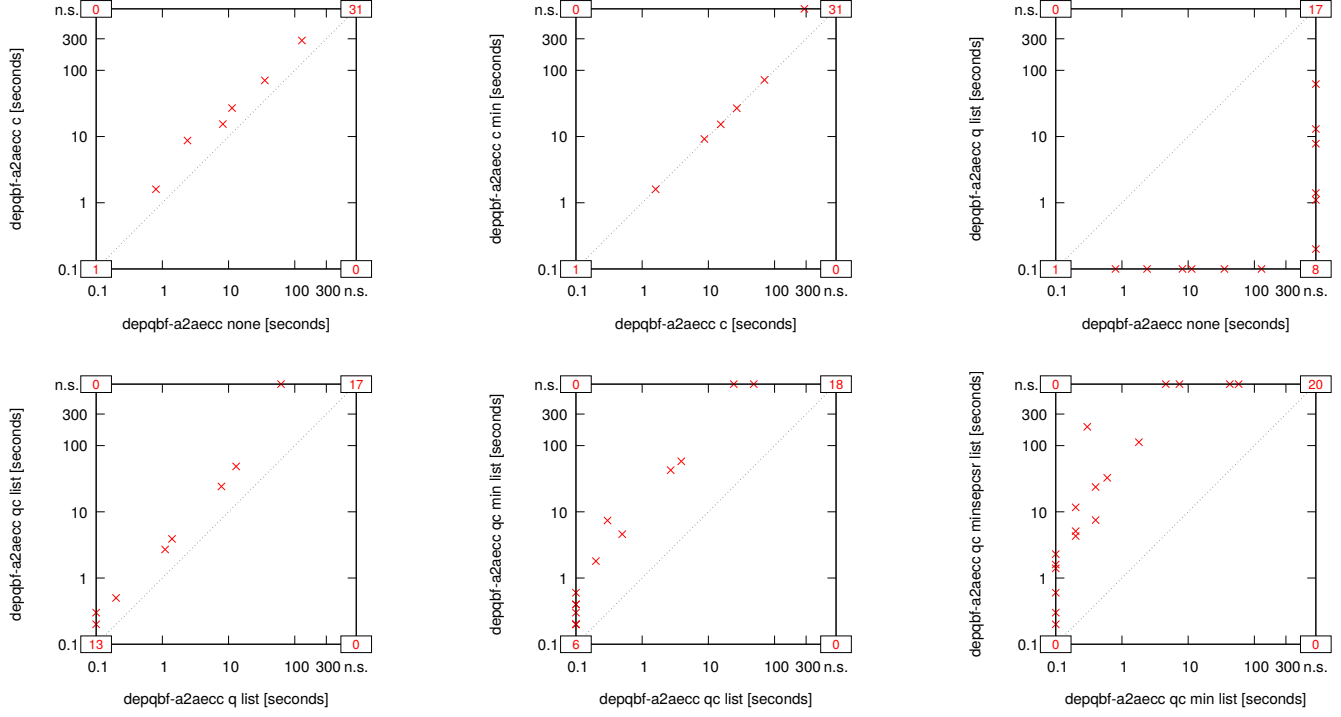


Fig. 1143: Suite Wintersteiger ($n = 38$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (run time in seconds).

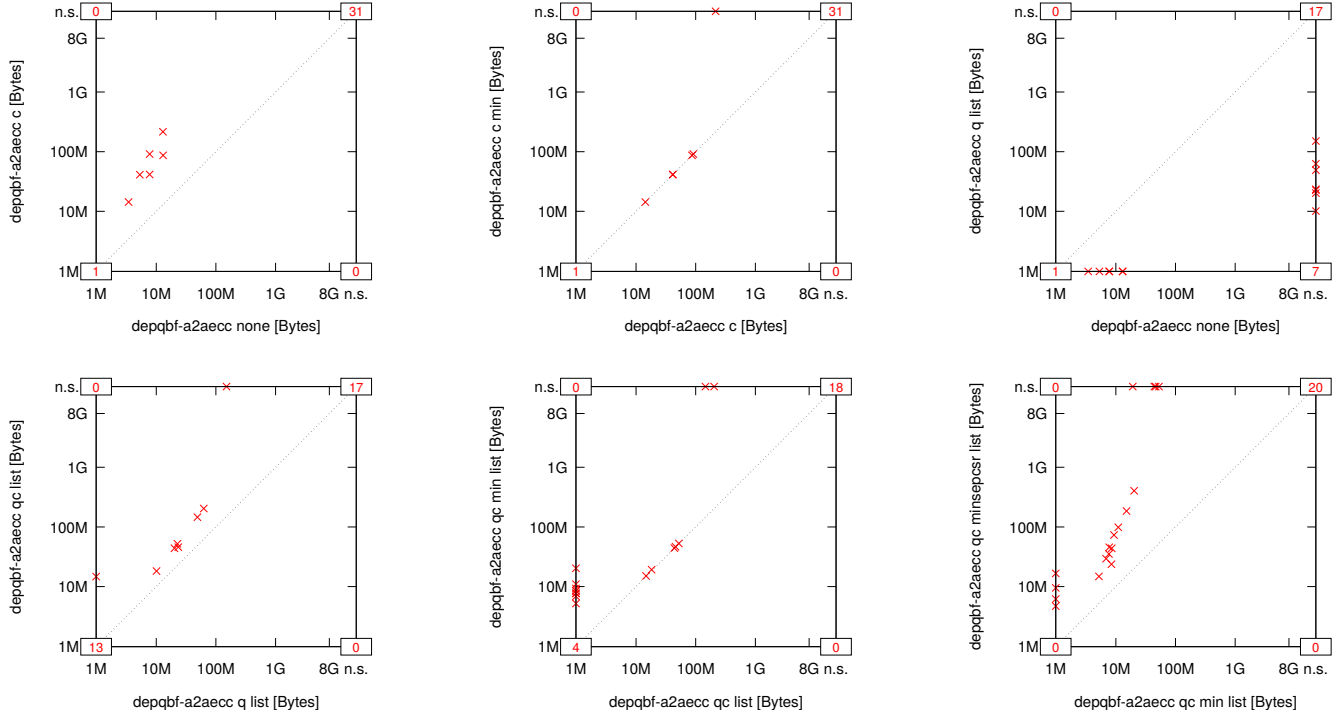


Fig. 1144: Suite Wintersteiger ($n = 38$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with list semantics (memory usage in Bytes).

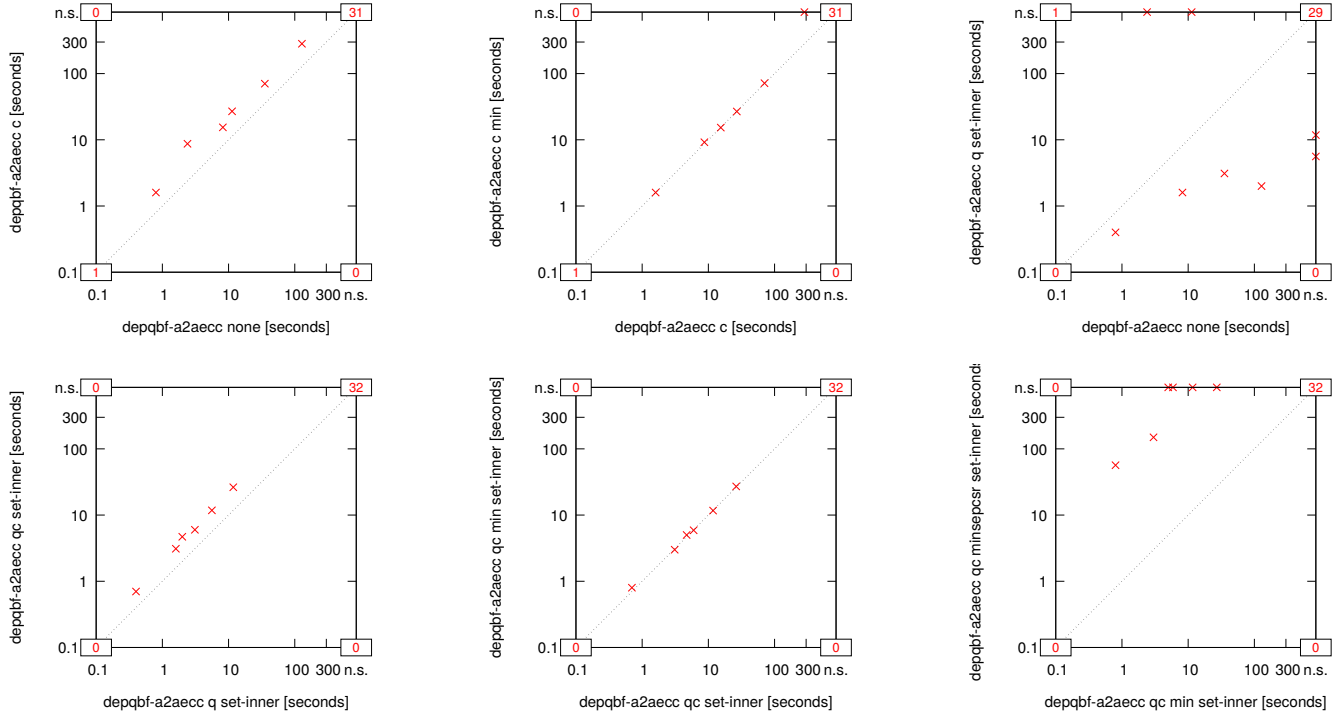


Fig. 1145: Suite Wintersteiger ($n = 38$): Comparing run times for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (run time in seconds).

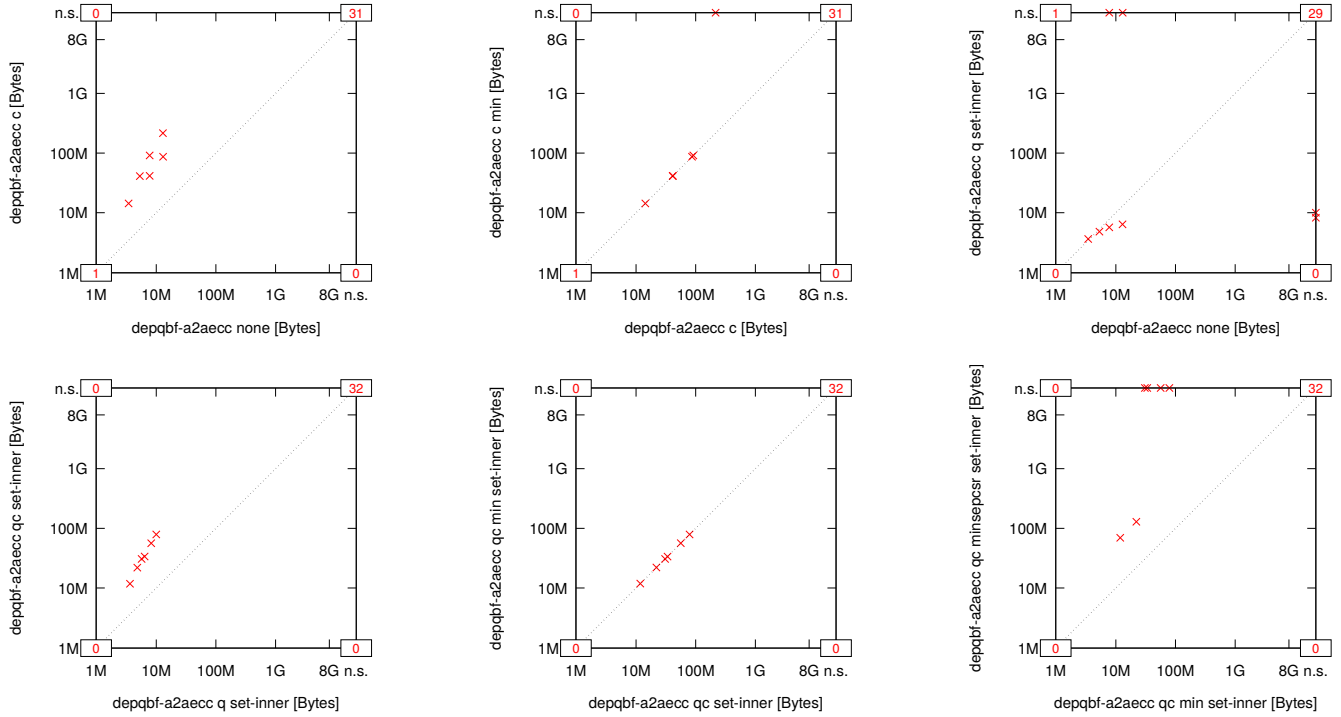


Fig. 1146: Suite Wintersteiger ($n = 38$): Comparing memory usage for extracting different unsatisfiable cores in DepQBF-a2aecc with set-inner semantics (memory usage in Bytes).

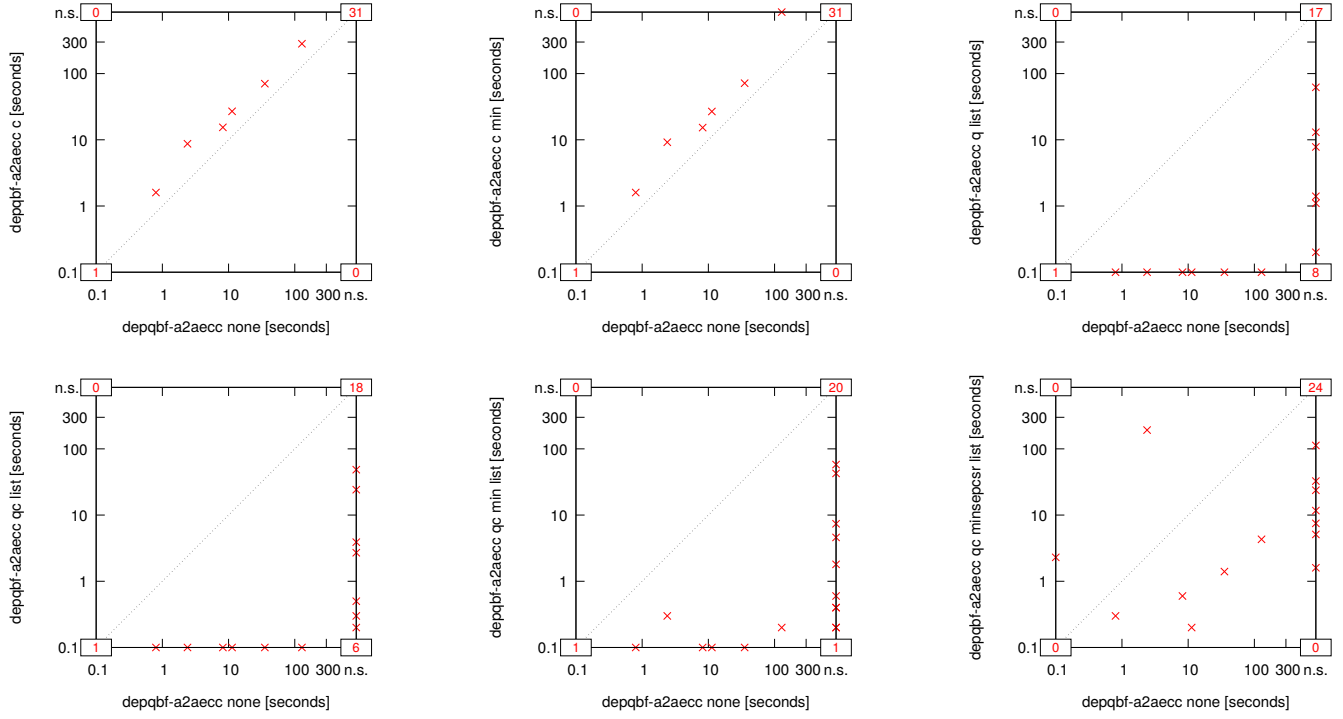


Fig. 1147: Suite Wintersteiger ($n = 38$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

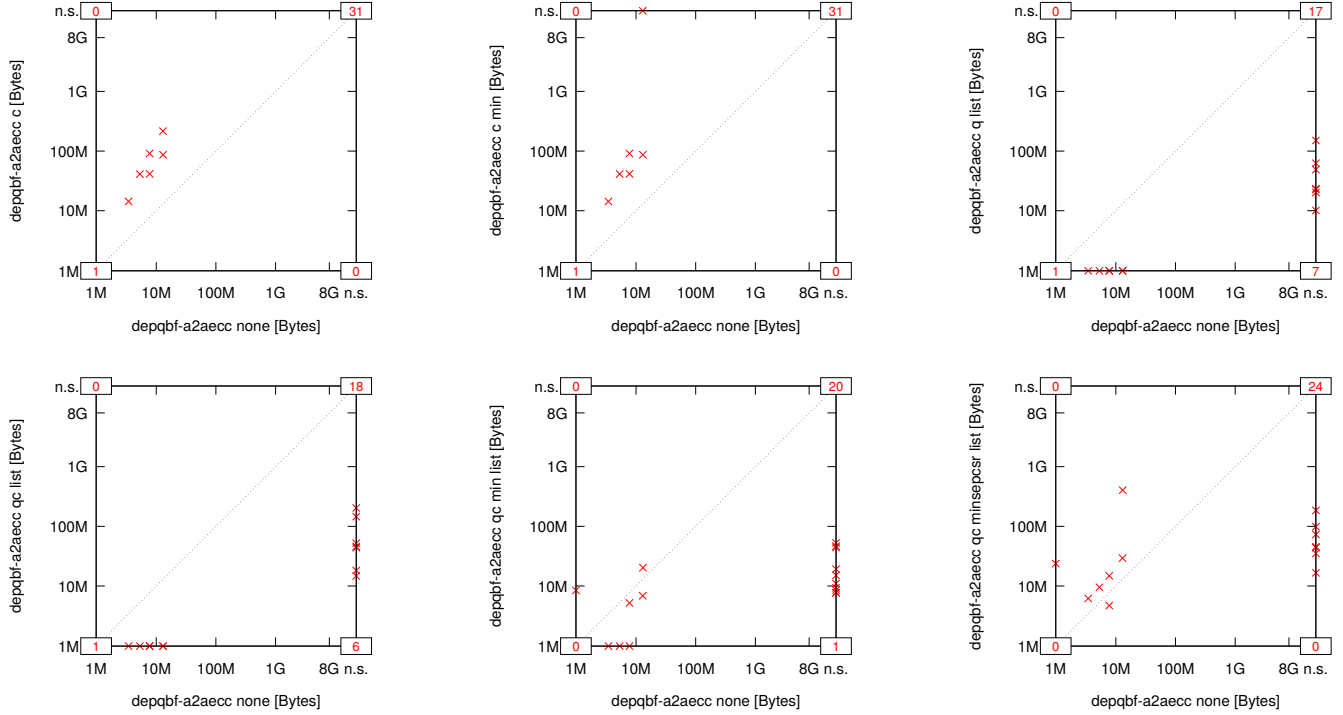


Fig. 1148: Suite Wintersteiger ($n = 38$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. DepQBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

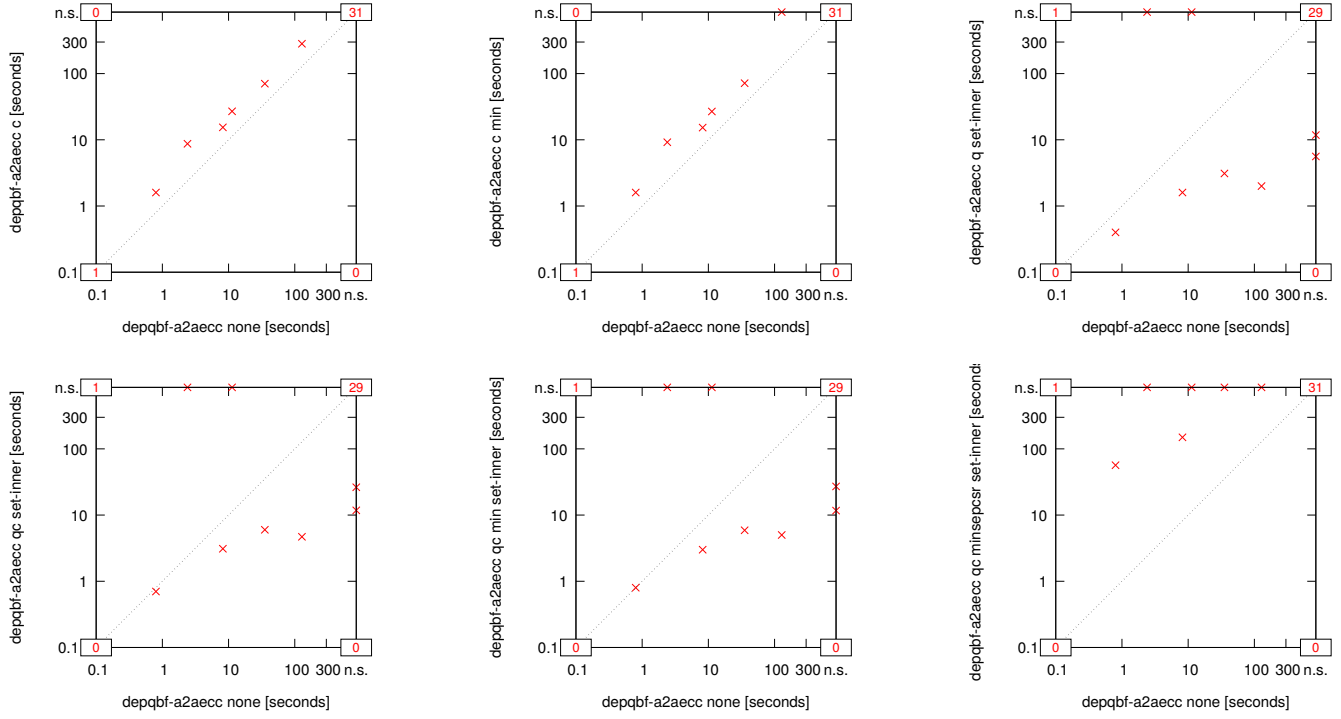


Fig. 1149: Suite Wintersteiger ($n = 38$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. Dep-QBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (run time in seconds).

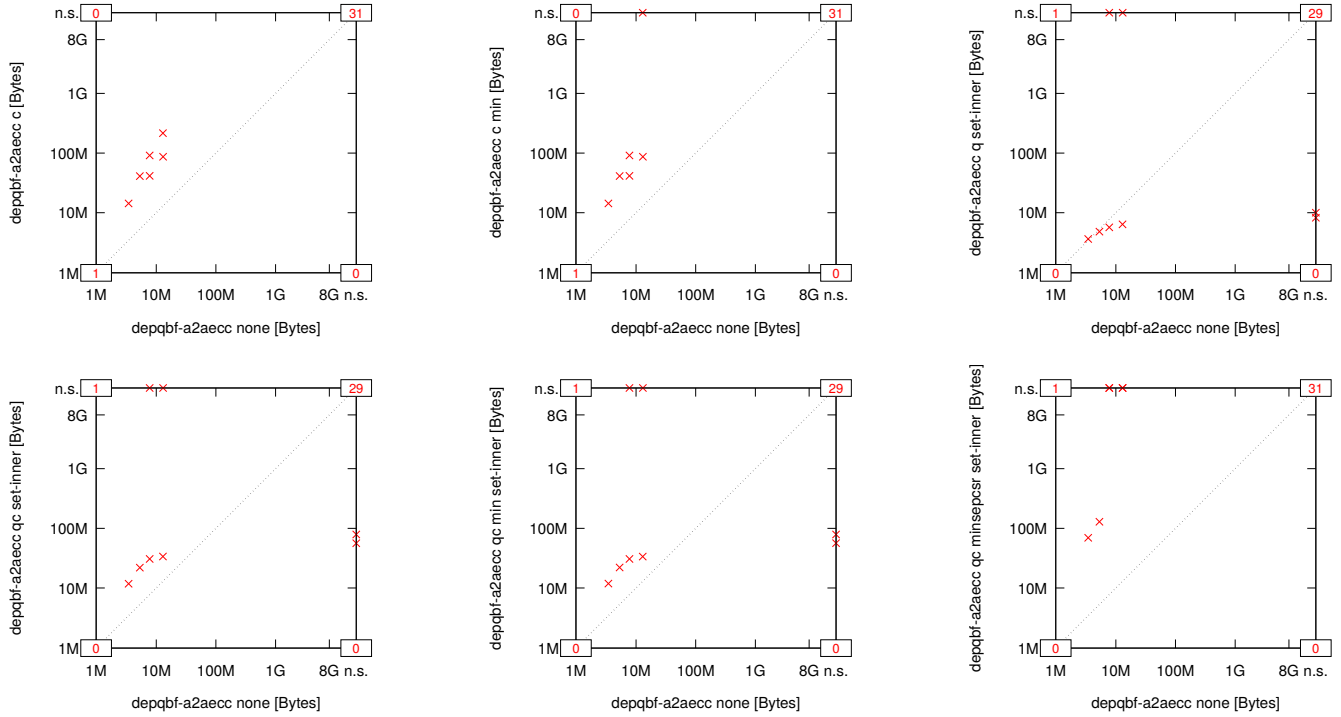


Fig. 1150: Suite Wintersteiger ($n = 38$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. Dep-QBF-a2aecc with incremental solving enabled but without unsatisfiable core extraction as the baseline (memory usage in Bytes).

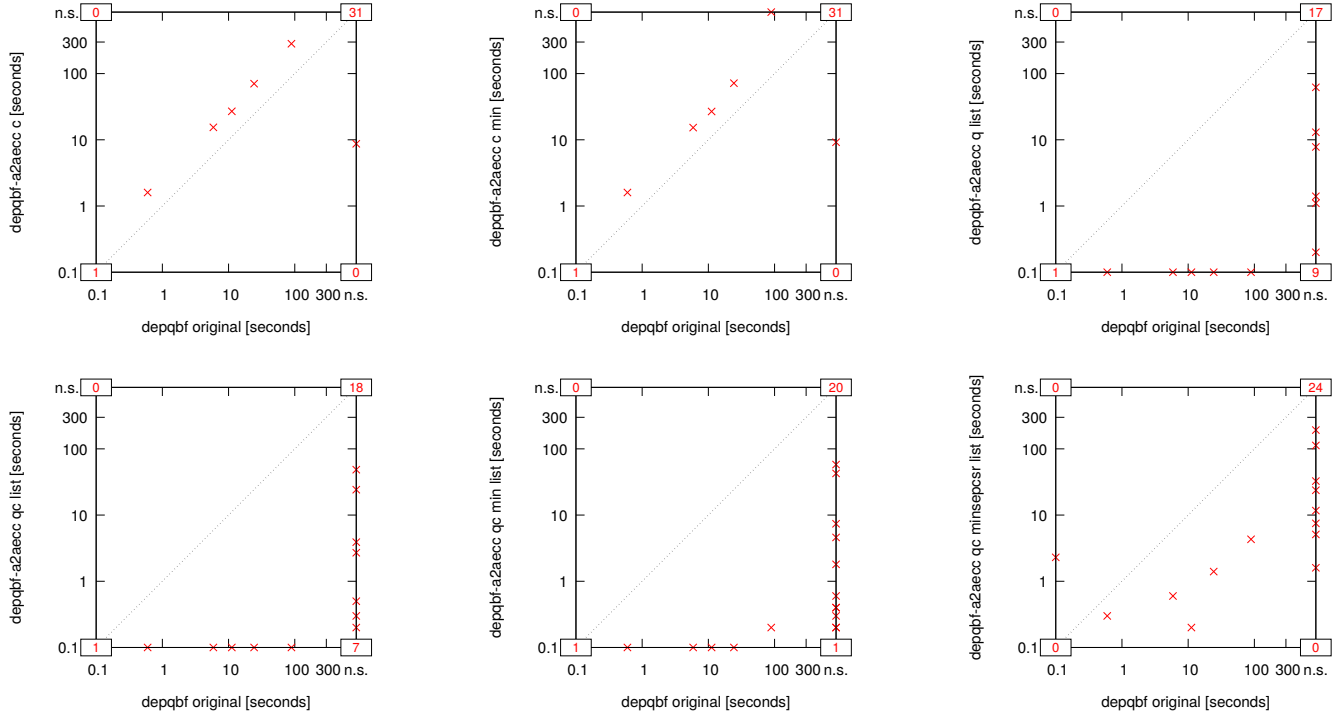


Fig. 1151: Suite Wintersteiger ($n = 38$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

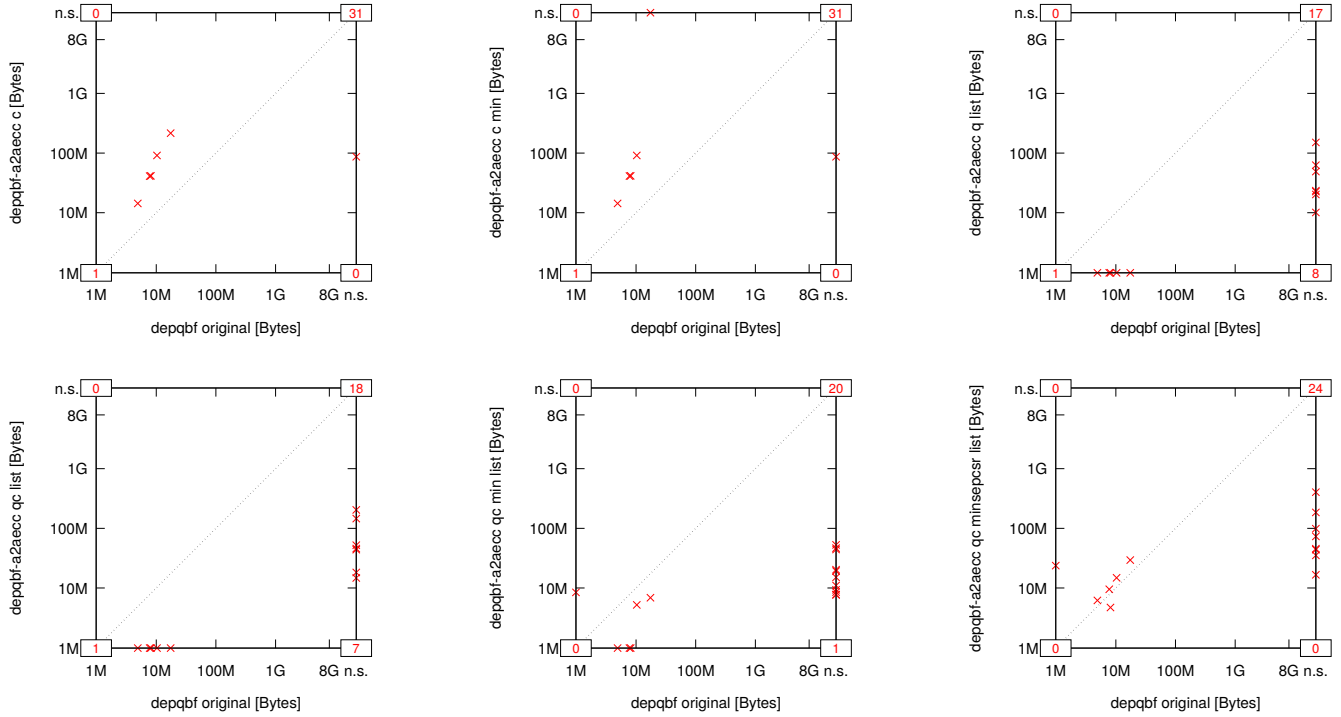


Fig. 1152: Suite Wintersteiger ($n = 38$): Overhead of extracting and minimizing unsatisfiable cores with list semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

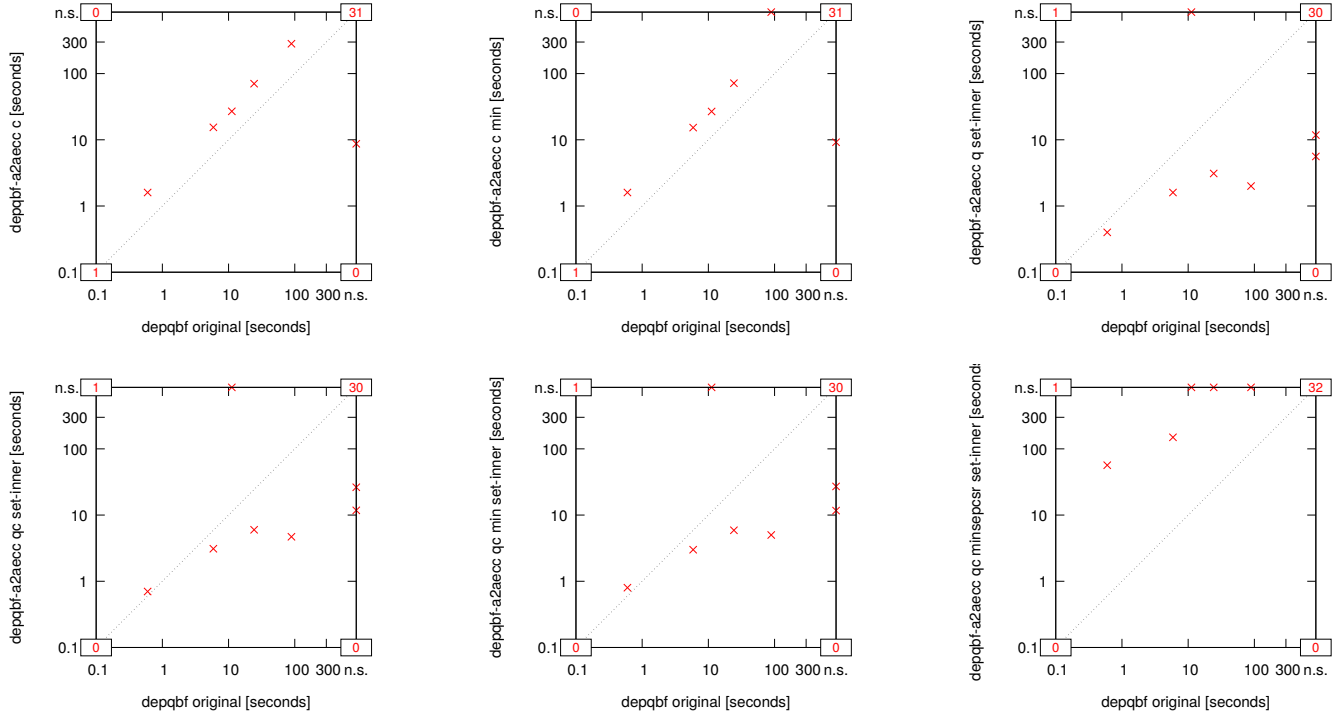


Fig. 1153: Suite Wintersteiger ($n = 38$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (run time in seconds).

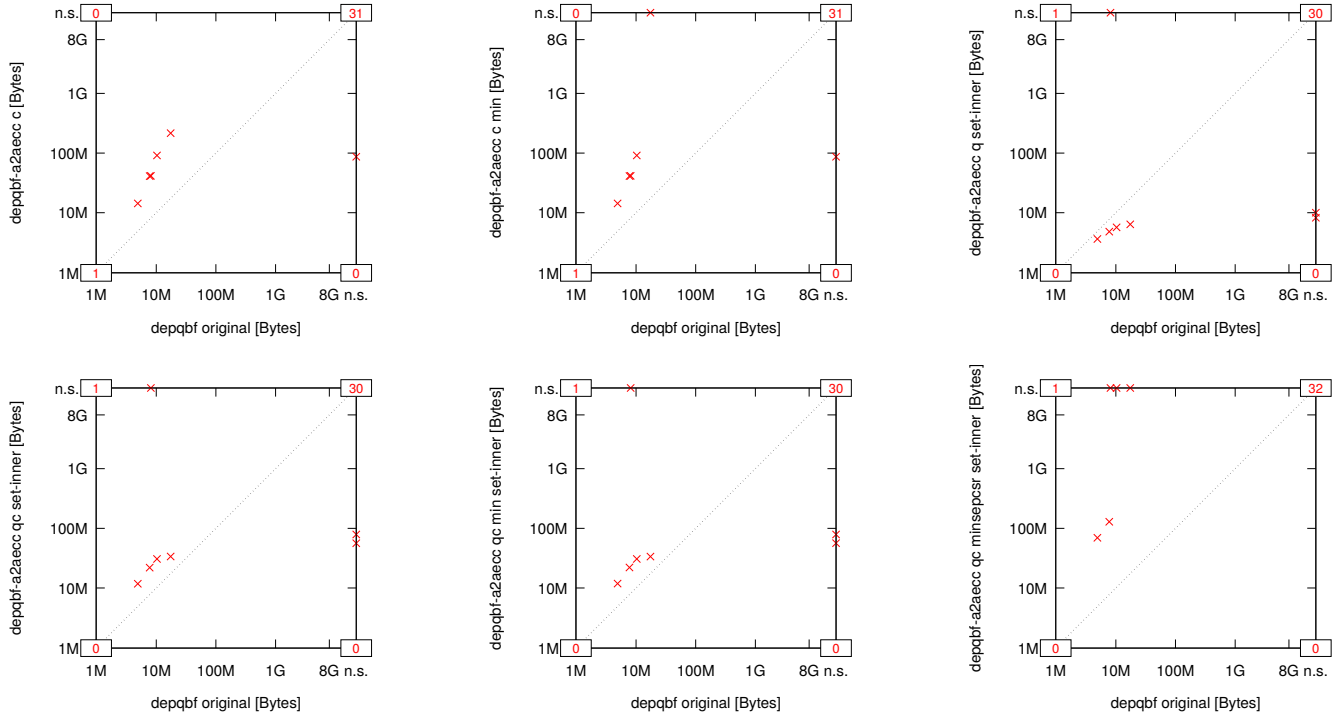


Fig. 1154: Suite Wintersteiger ($n = 38$): Overhead of extracting and minimizing unsatisfiable cores with set-inner semantics w.r.t. the original version of DepQBF without incremental solving as the baseline (memory usage in Bytes).

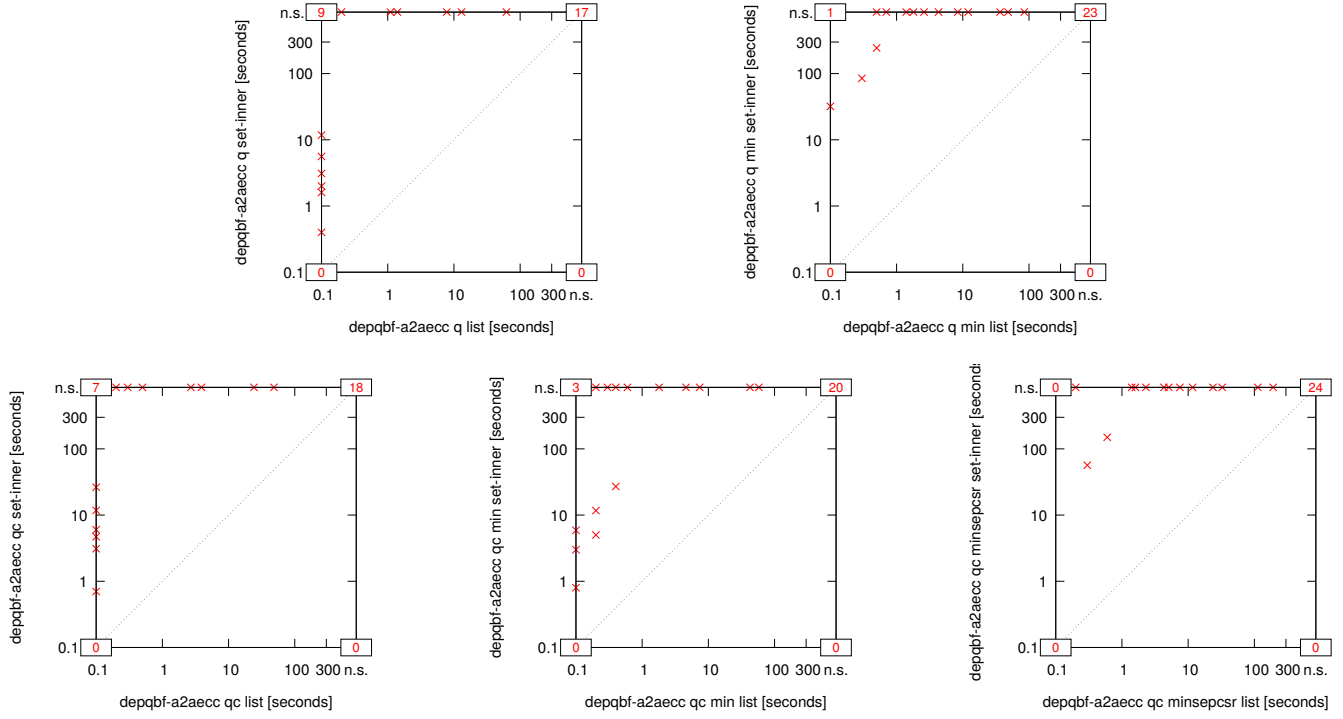


Fig. 1155: Suite Wintersteiger ($n = 38$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (run time in seconds).

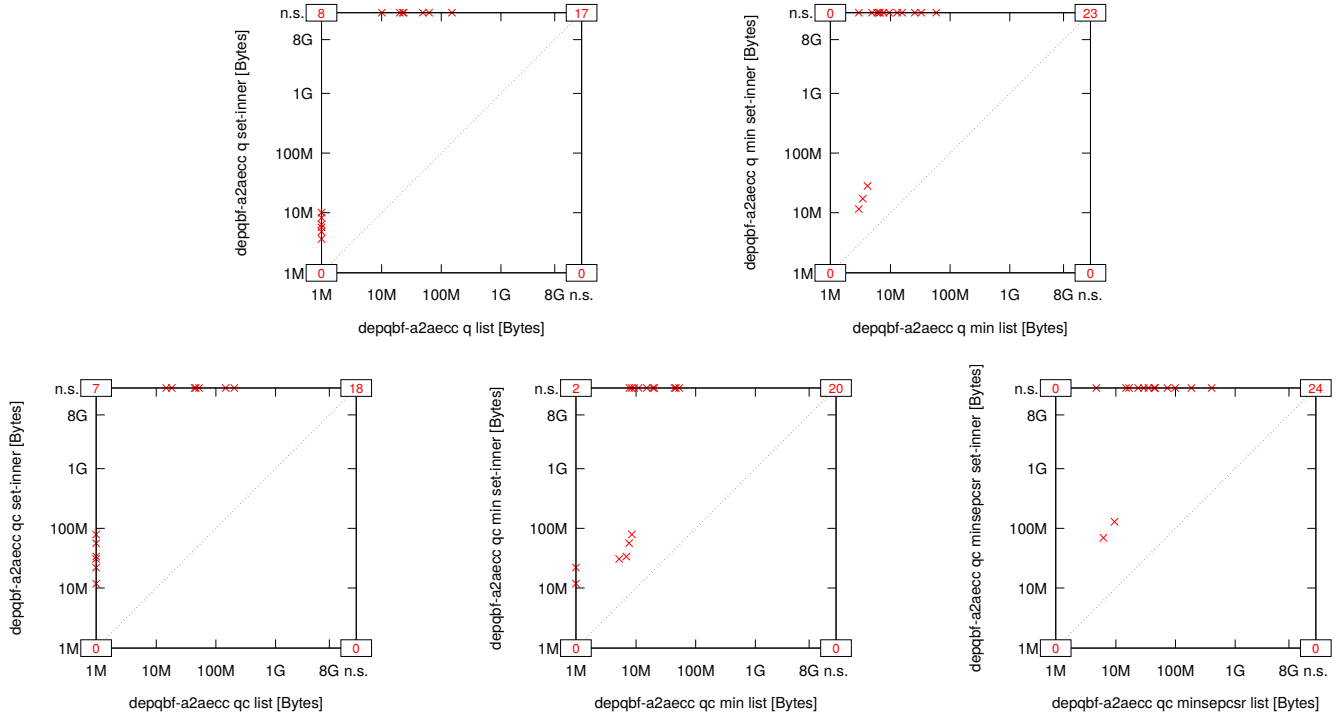


Fig. 1156: Suite Wintersteiger ($n = 38$): Extracting and minimizing unsatisfiable cores in DepQBF-a2aecc with set-inner versus list semantics (memory usage in Bytes).

B. Partitioned by Number of \forall Quantified Variables

This subsection shows figures of the overhead of extracting and minimizing unsatisfiable cores with subfigures for partitions of the benchmarks according to the number of \forall quantified variables.

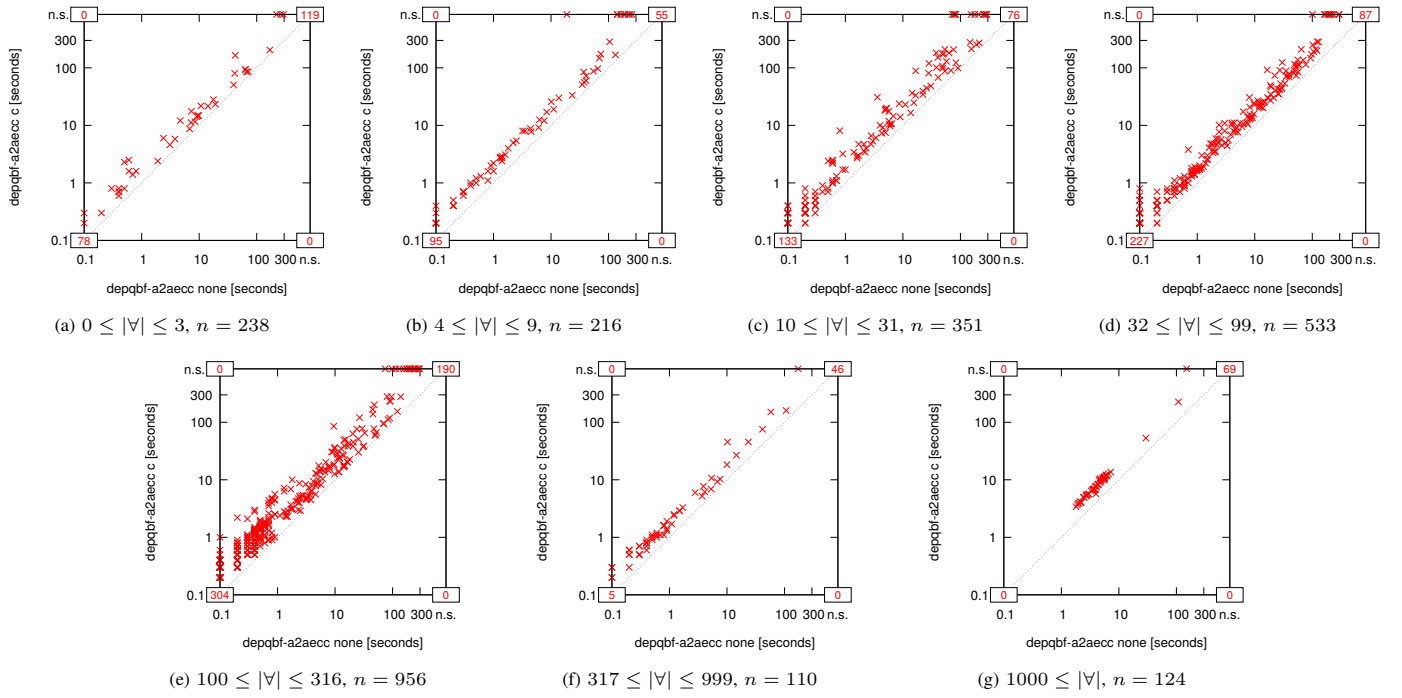


Fig. 1157: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c versus mode none partitioned by number of \forall quantified variables (run time in seconds).

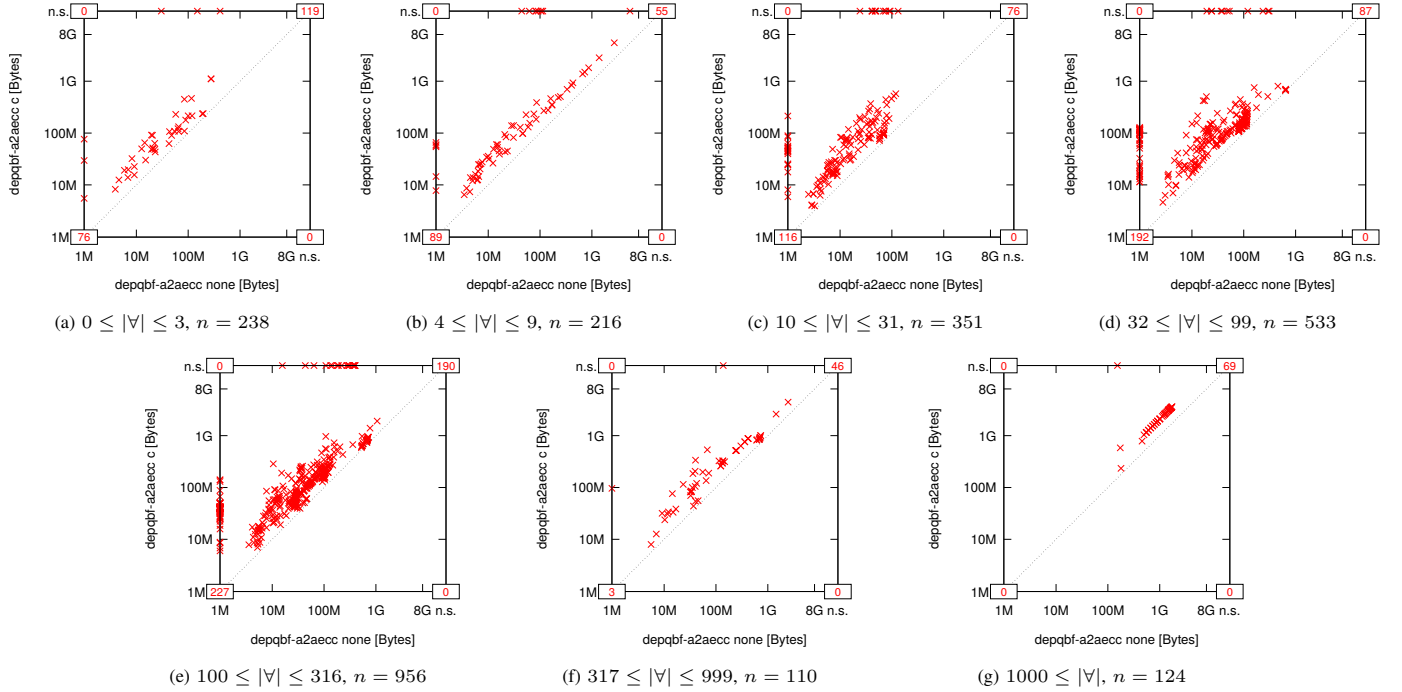


Fig. 1158: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c versus mode none partitioned by number of \forall quantified variables (memory usage in Bytes).

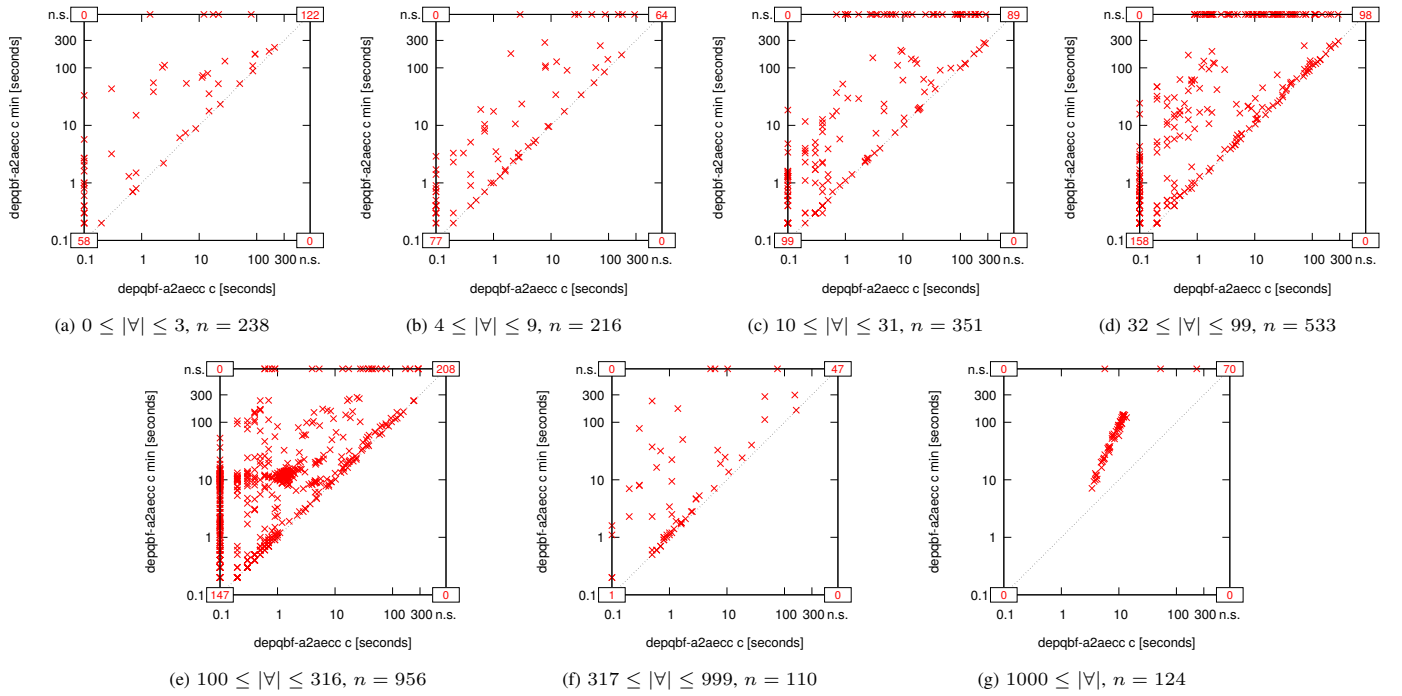


Fig. 1159: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode c partitioned by number of \forall quantified variables (run time in seconds).

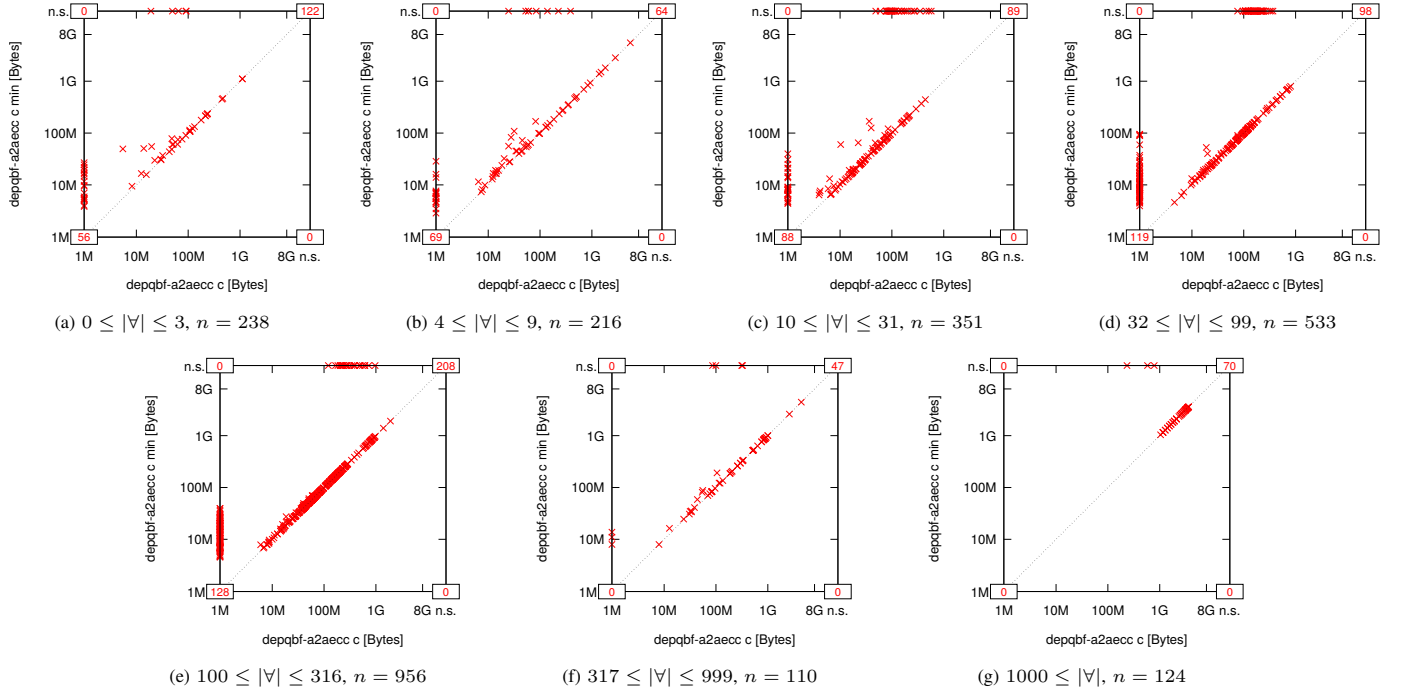


Fig. 1160: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode c partitioned by number of \forall quantified variables (memory usage in Bytes).

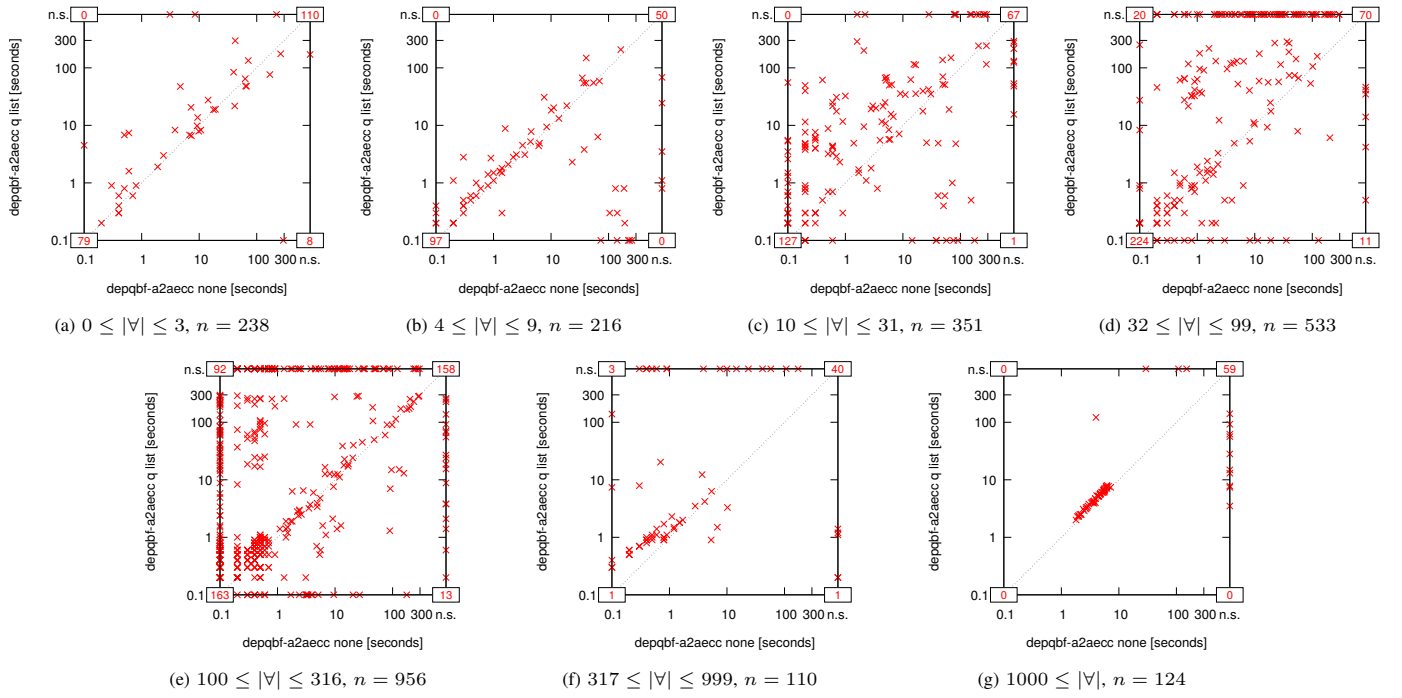


Fig. 1161: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode none partitioned by number of \forall quantified variables (run time in seconds).

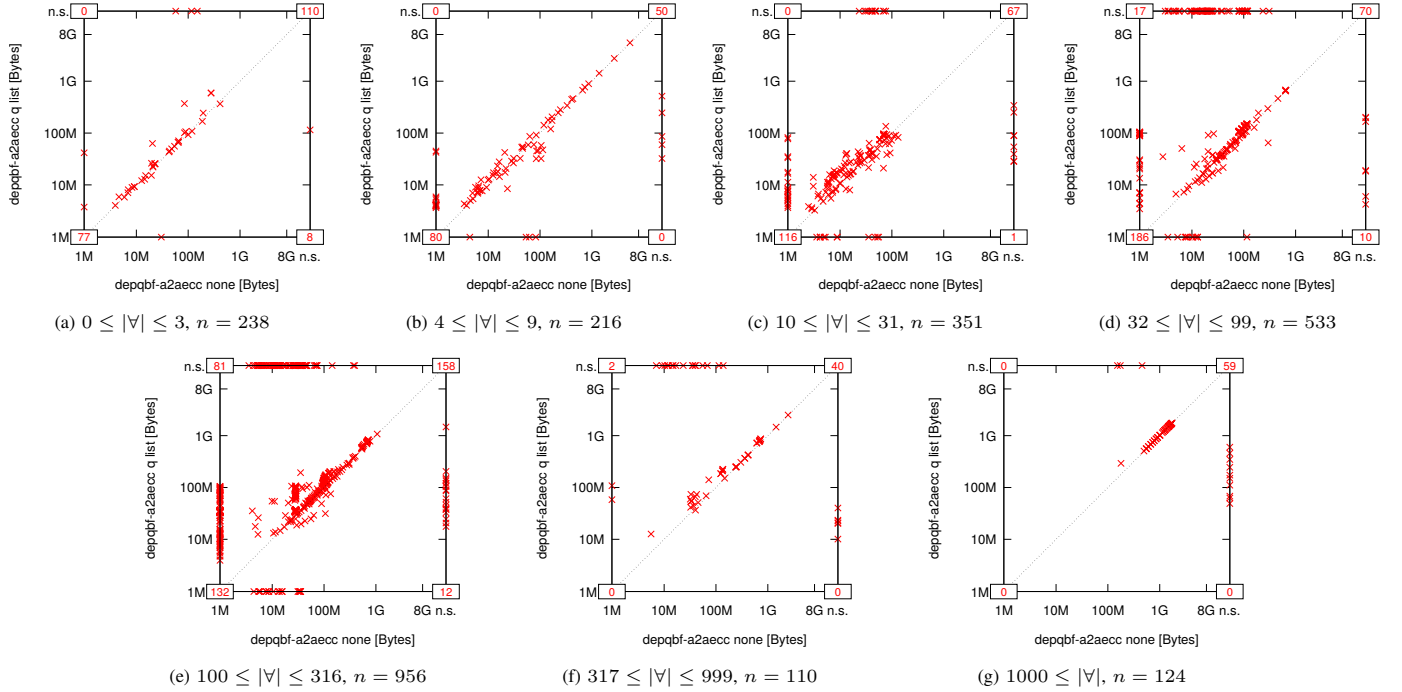


Fig. 1162: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode none partitioned by number of \forall quantified variables (memory usage in Bytes).

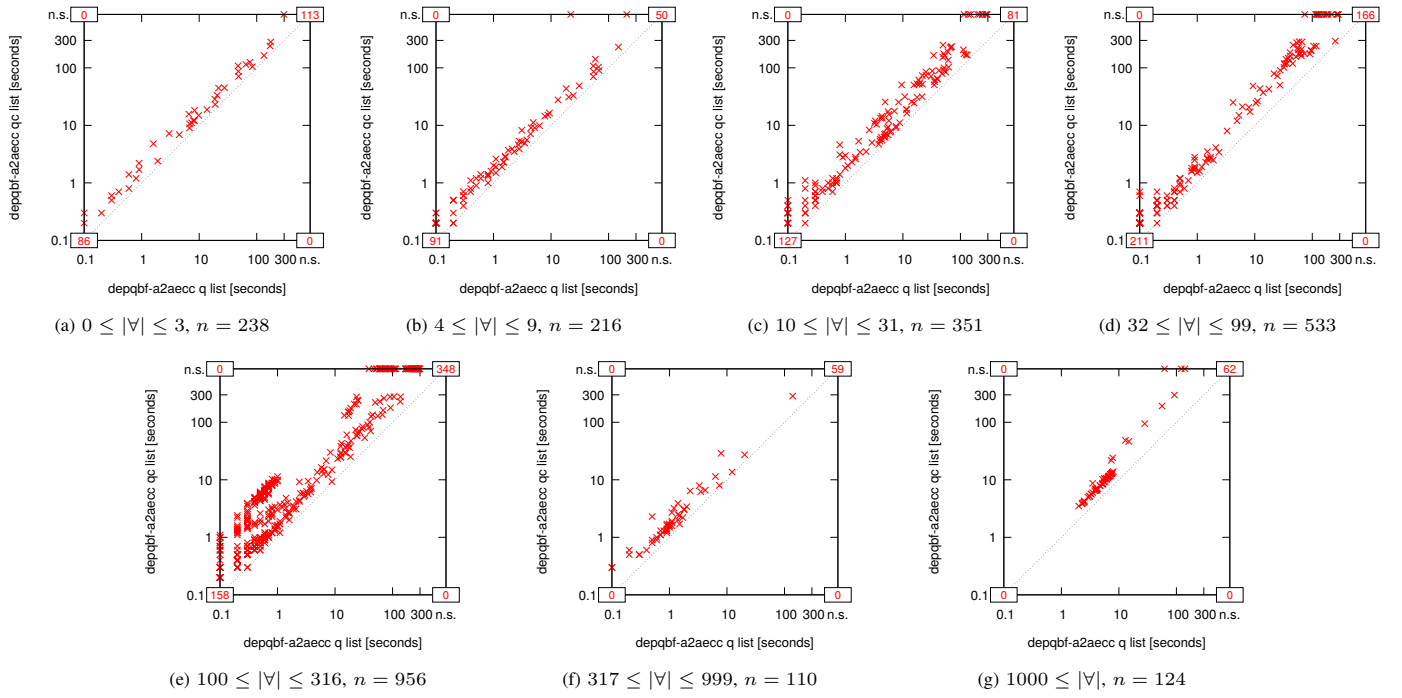


Fig. 1163: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode q list partitioned by number of \forall quantified variables (run time in seconds).

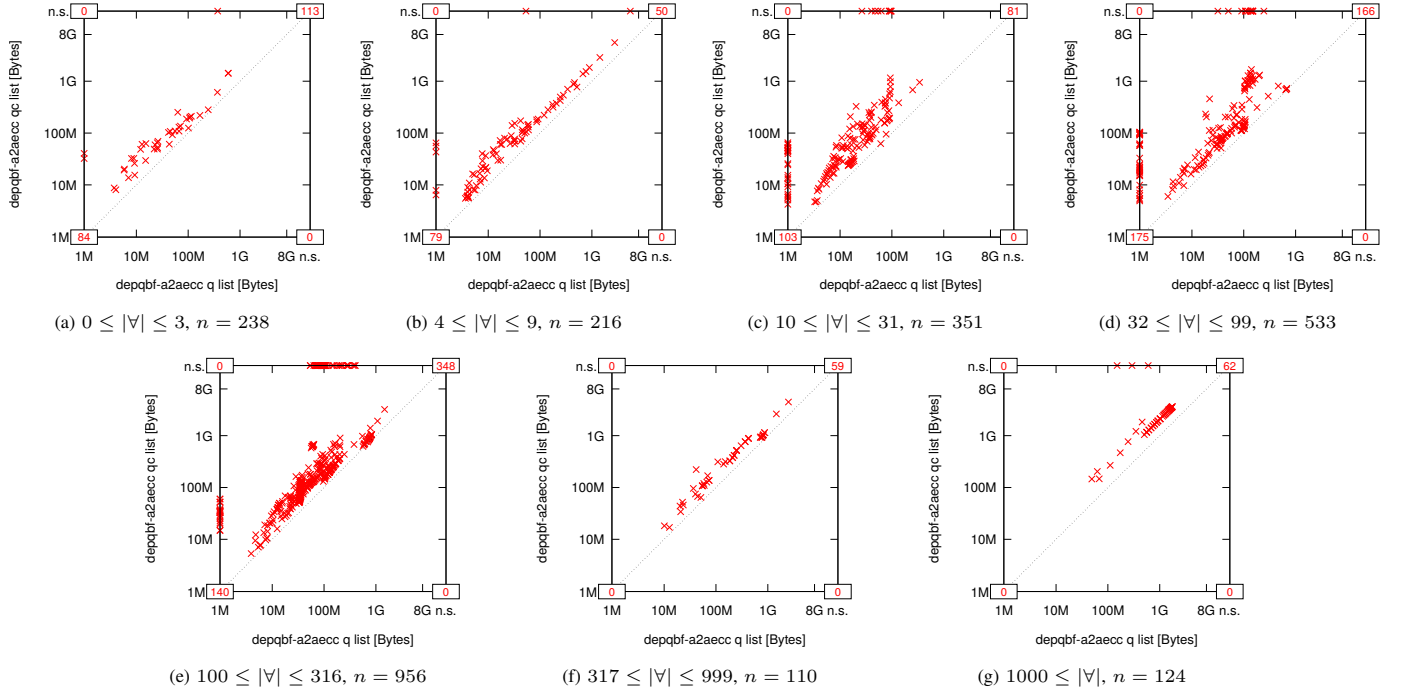


Fig. 1164: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode q list partitioned by number of \forall quantified variables (memory usage in Bytes).

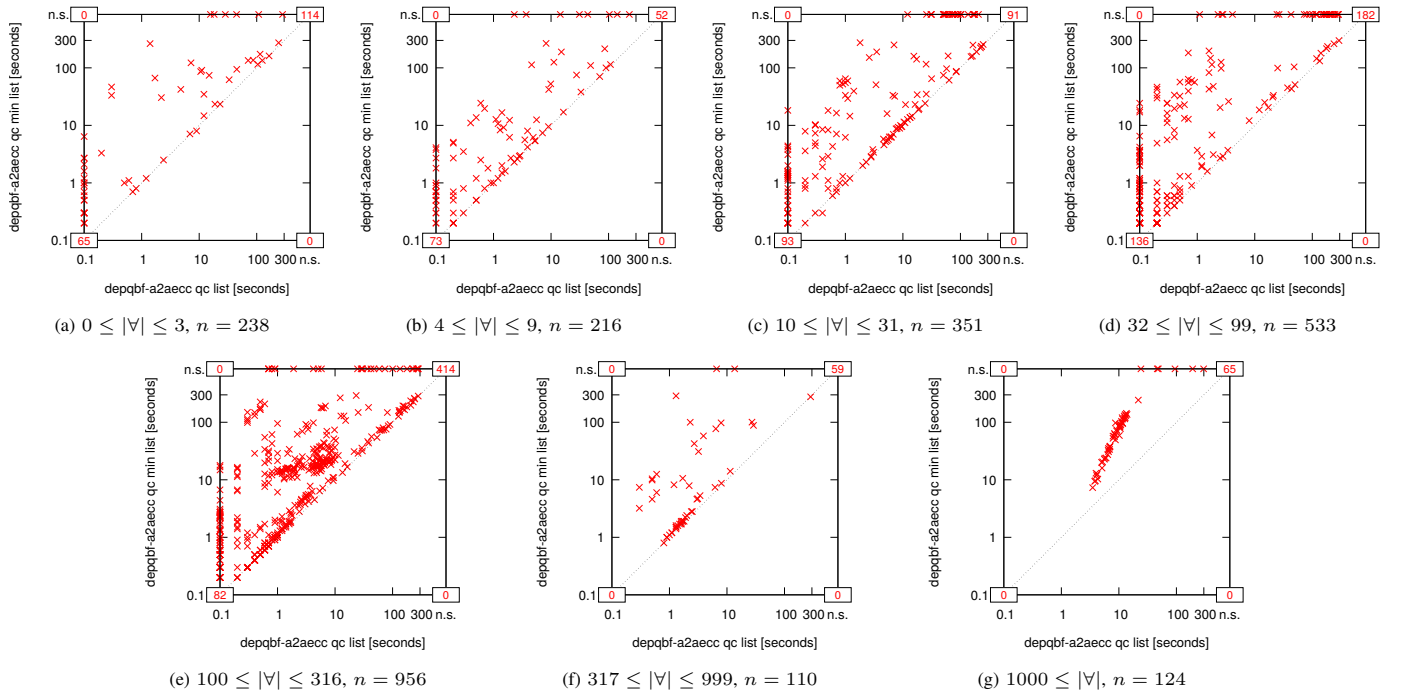


Fig. 1165: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode qc list partitioned by number of \forall quantified variables (run time in seconds).

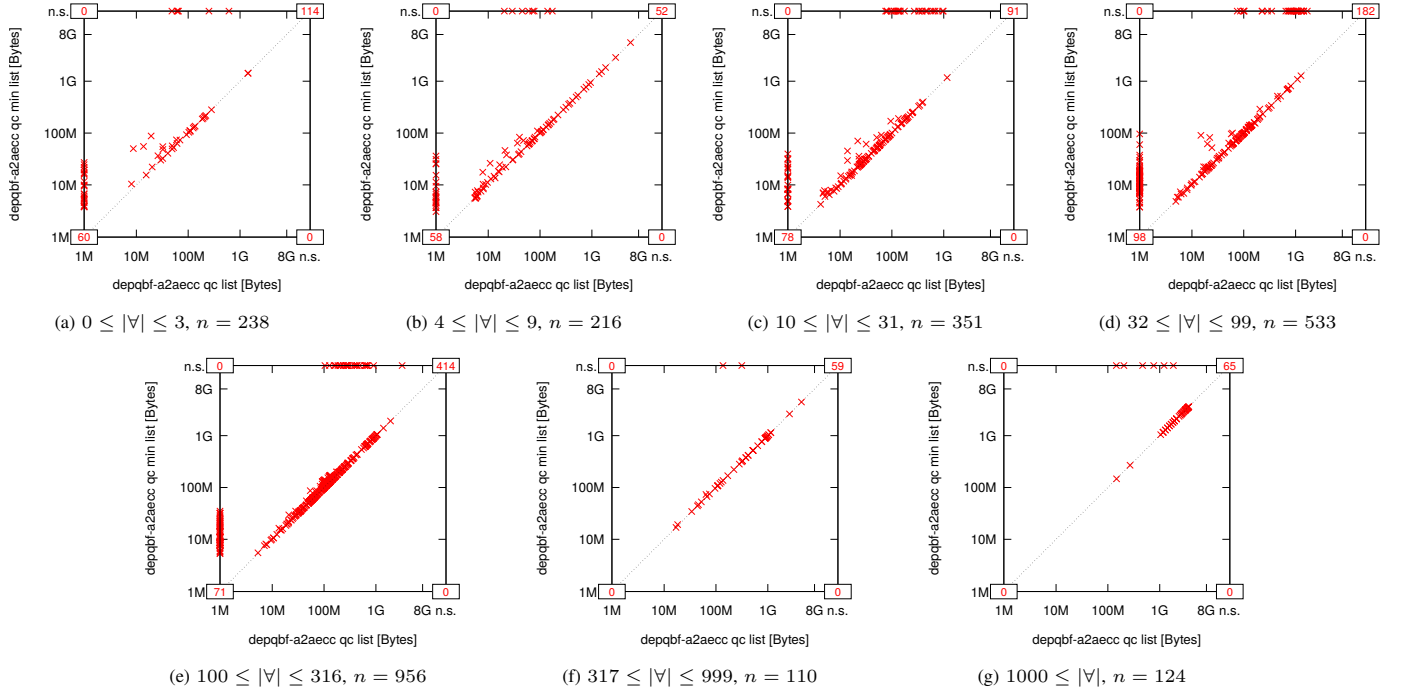


Fig. 1166: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode qc list partitioned by number of \forall quantified variables (memory usage in Bytes).

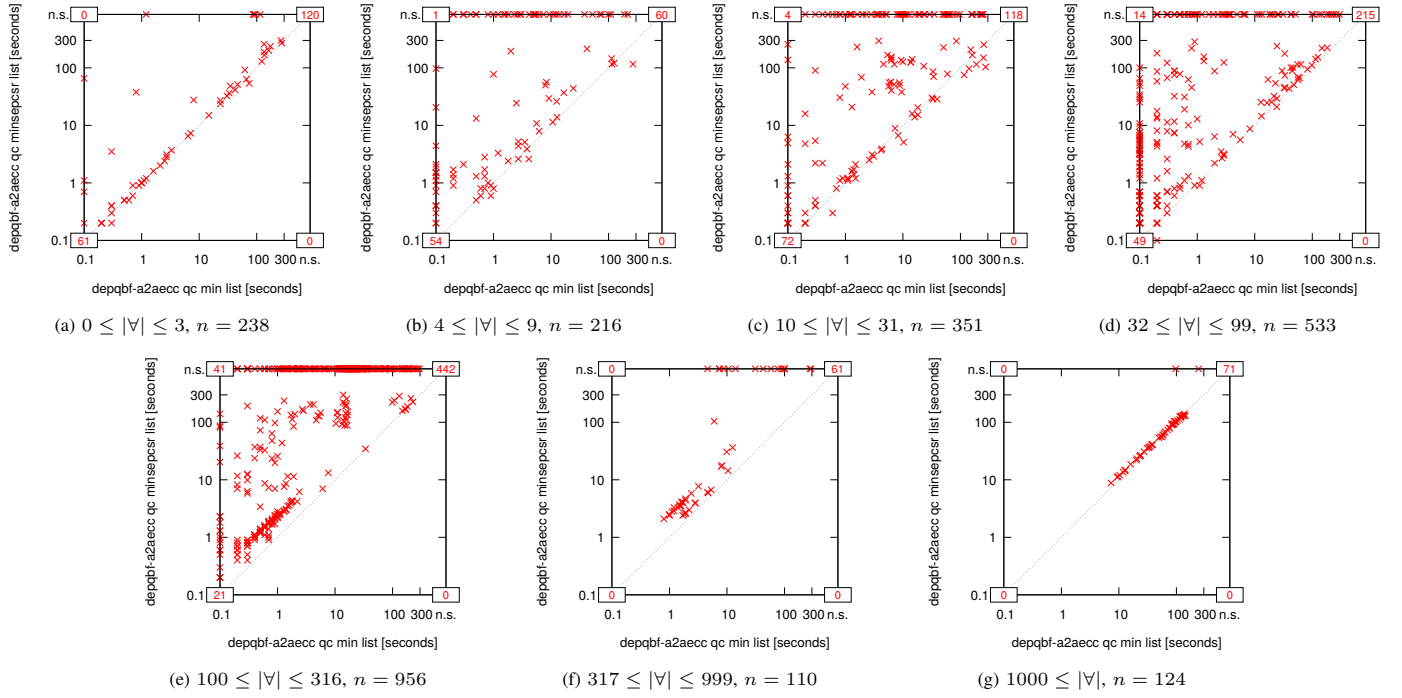


Fig. 1167: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode qc min list partitioned by number of \forall quantified variables (run time in seconds).

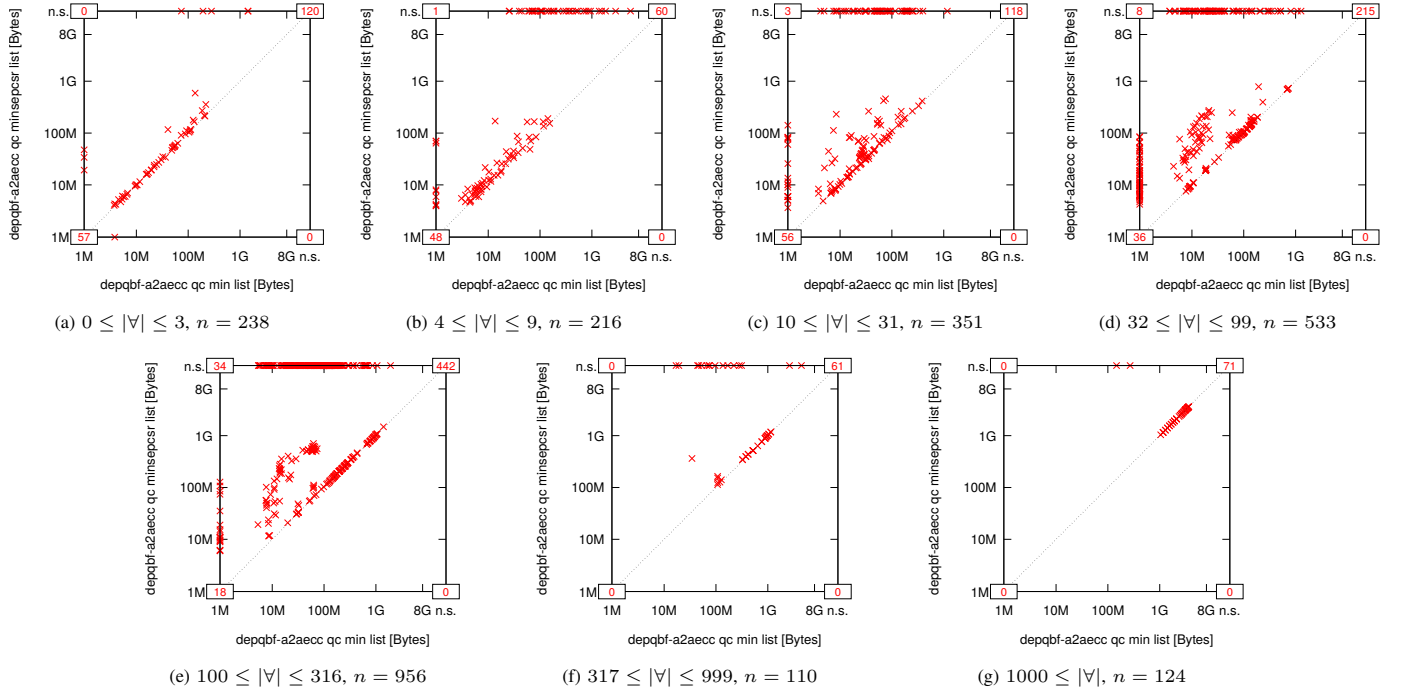


Fig. 1168: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode qc min list partitioned by number of \forall quantified variables (memory usage in Bytes).

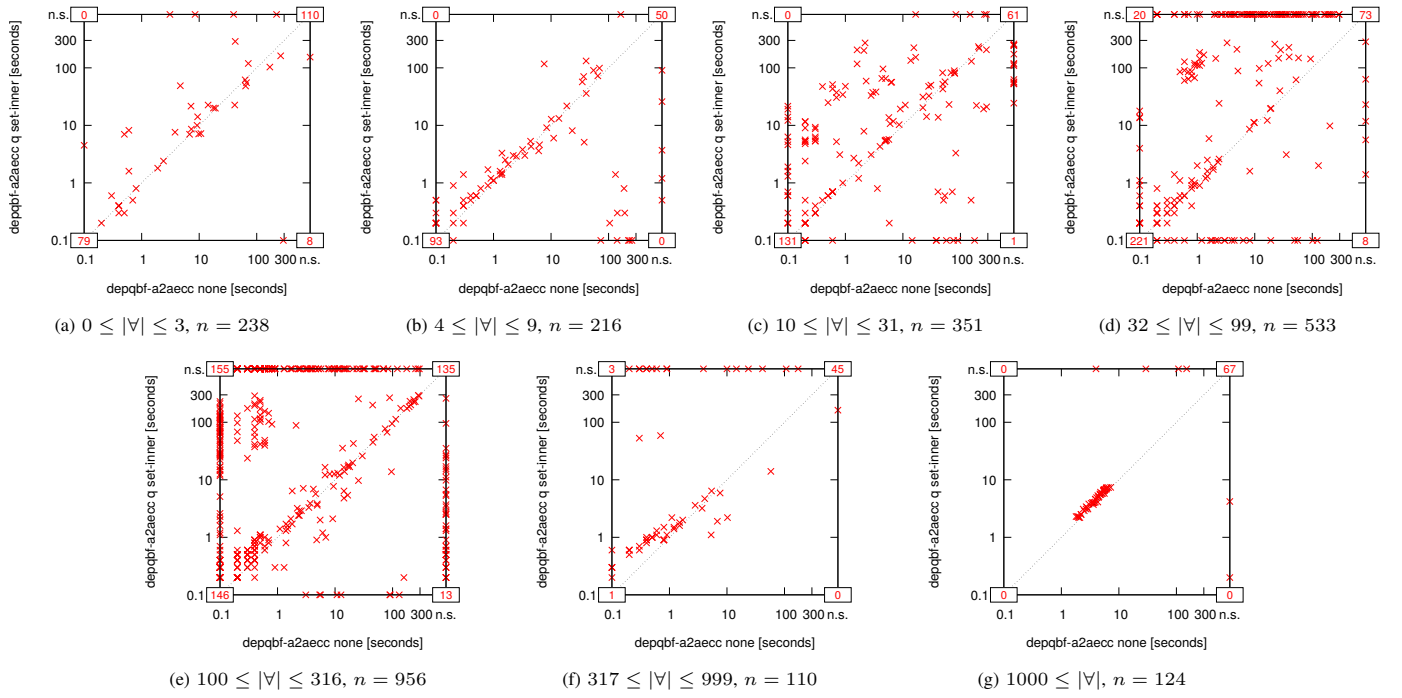


Fig. 1169: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode none partitioned by number of \forall quantified variables (run time in seconds).

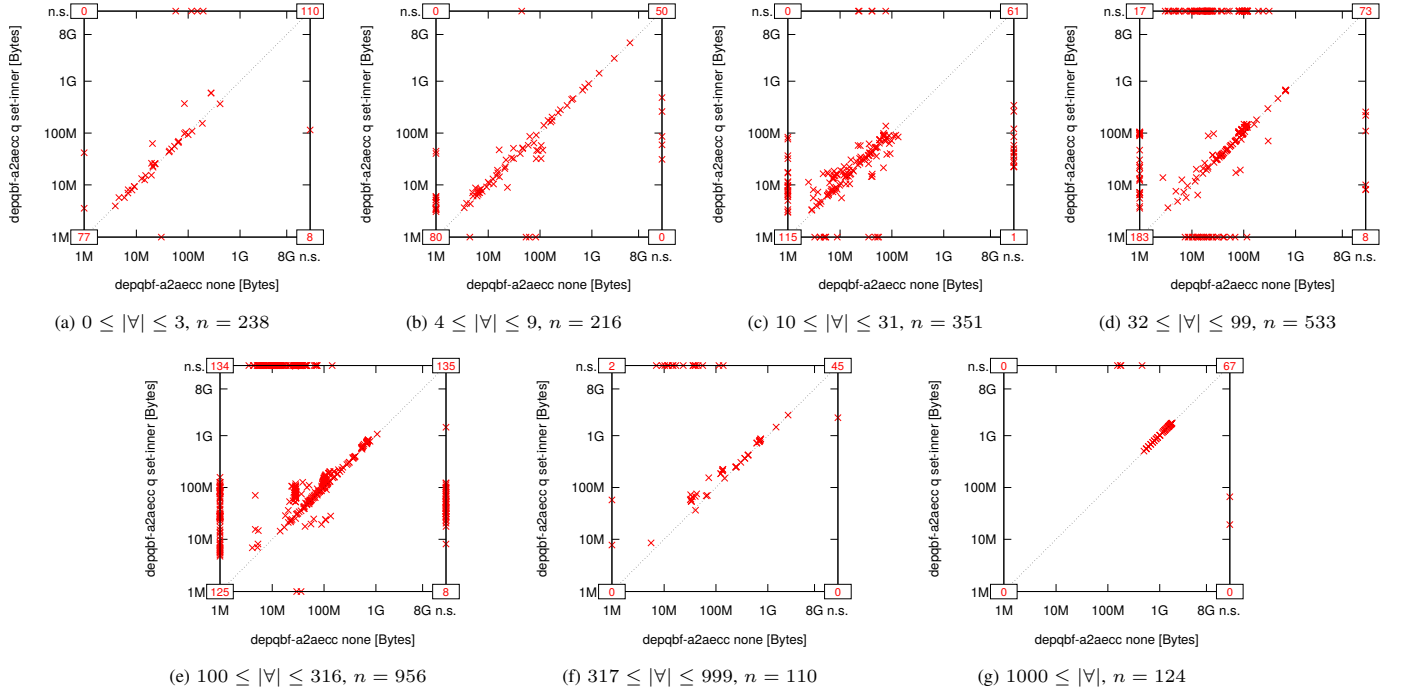


Fig. 1170: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode none partitioned by number of \forall quantified variables (memory usage in Bytes).

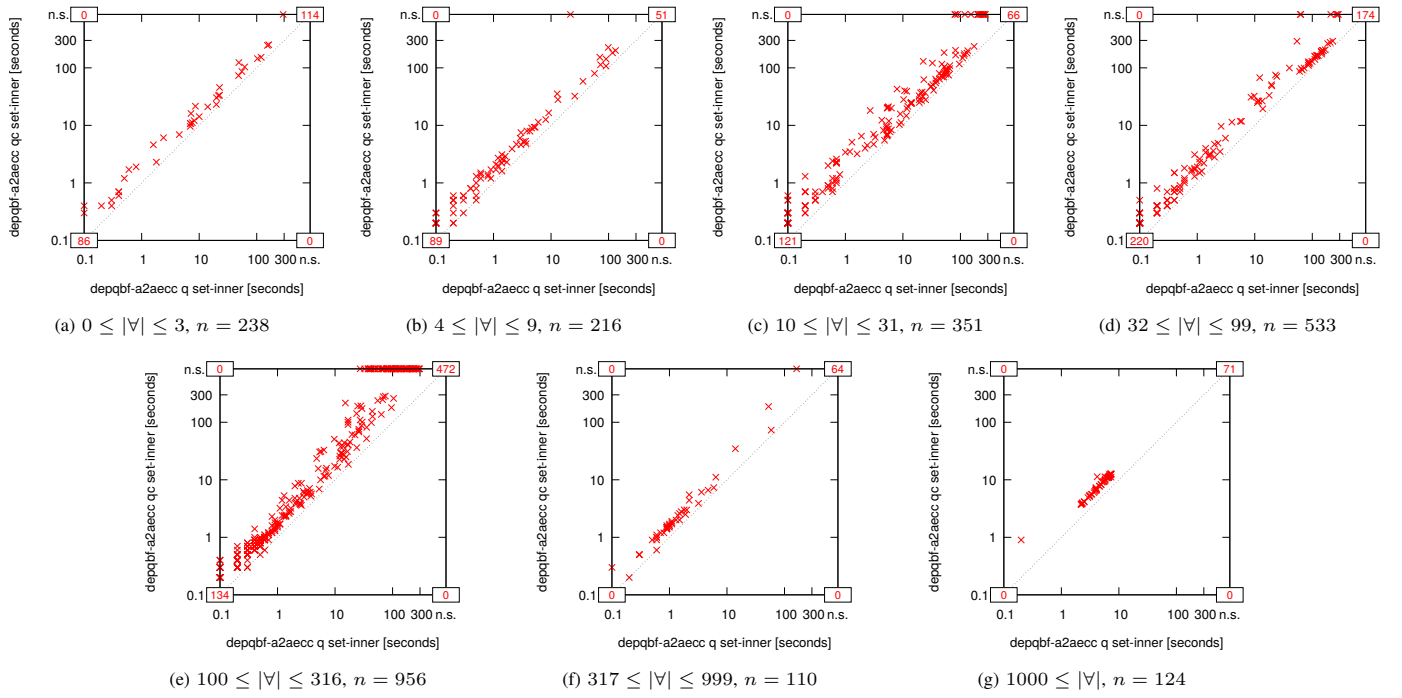


Fig. 1171: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode q set-inner partitioned by number of \forall quantified variables (run time in seconds).

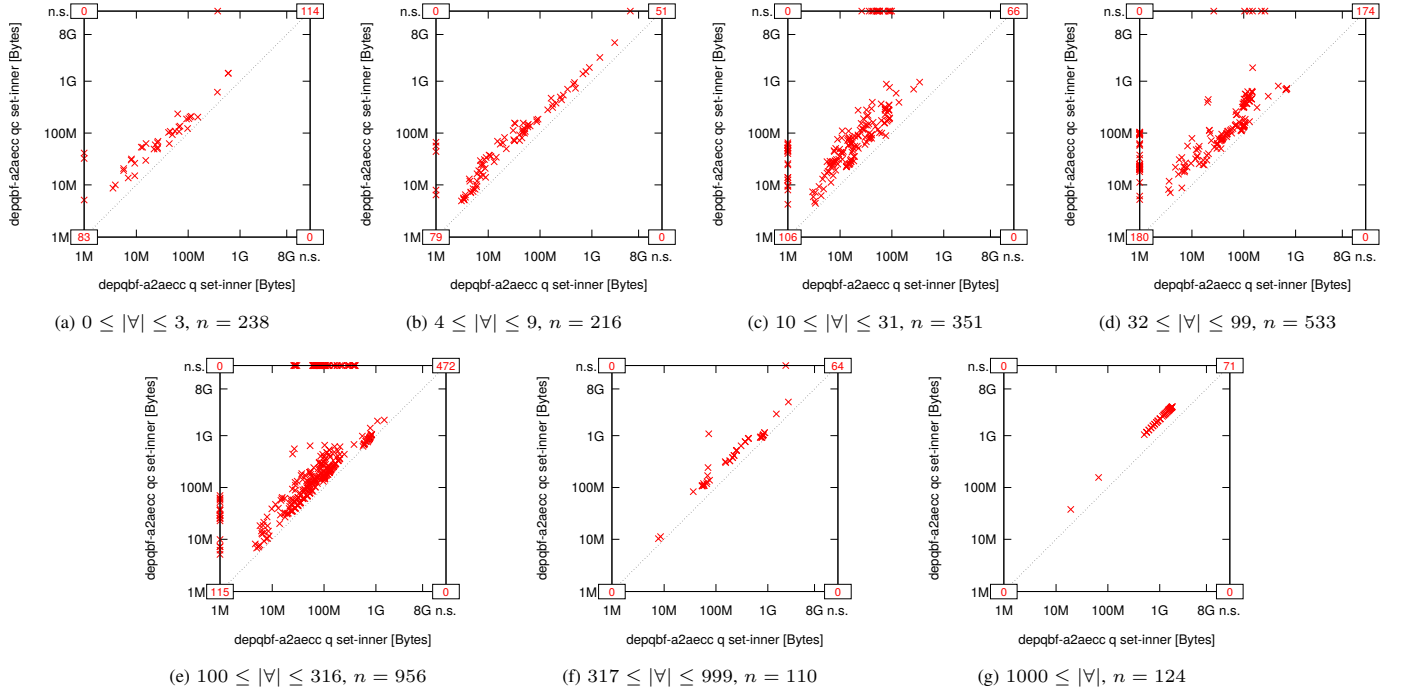


Fig. 1172: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode q set-inner partitioned by number of \forall quantified variables (memory usage in Bytes).

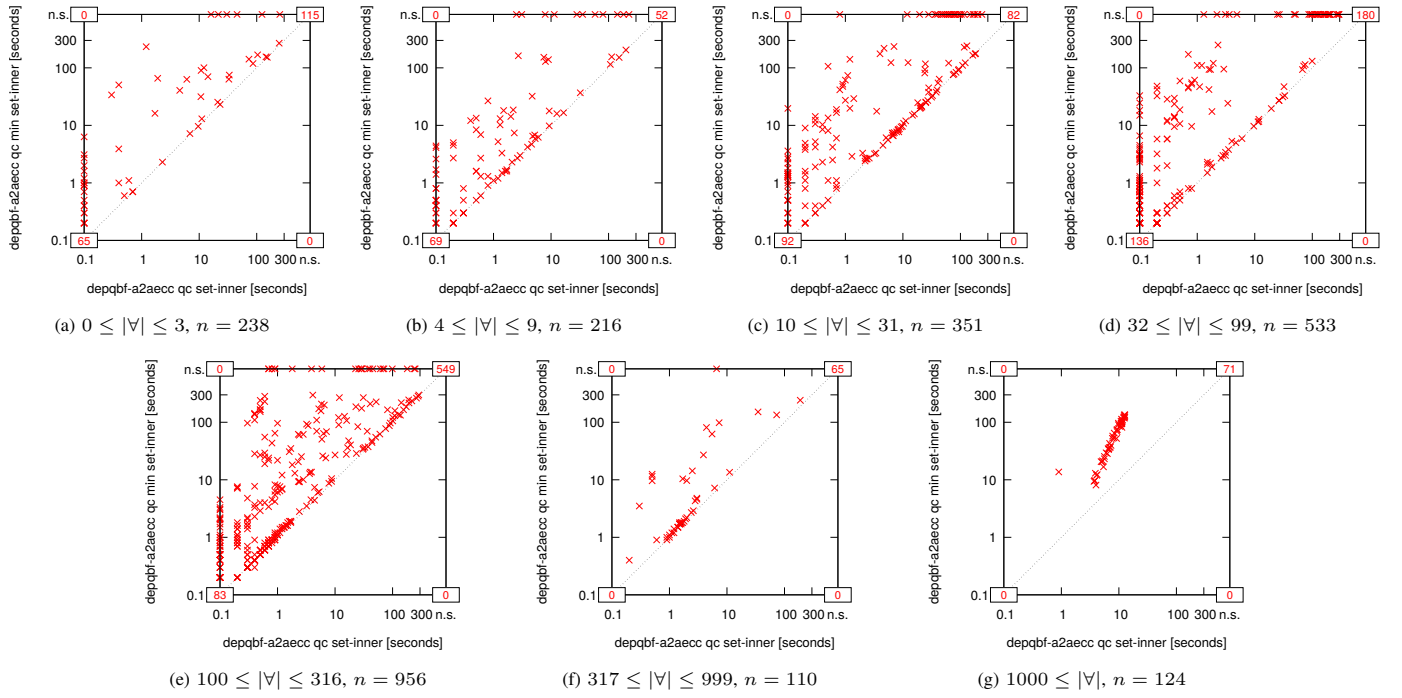


Fig. 1173: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode qc set-inner partitioned by number of \forall quantified variables (run time in seconds).

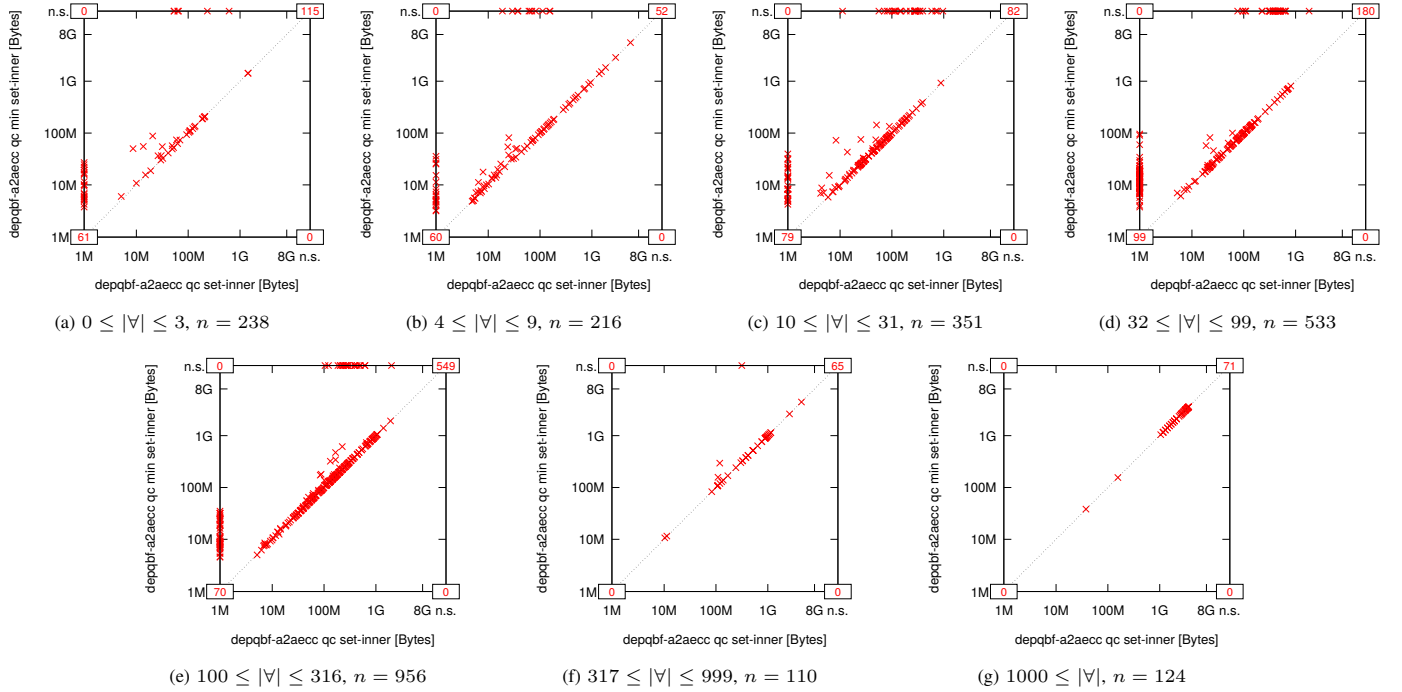


Fig. 1174: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode qc set-inner partitioned by number of \forall quantified variables (memory usage in Bytes).

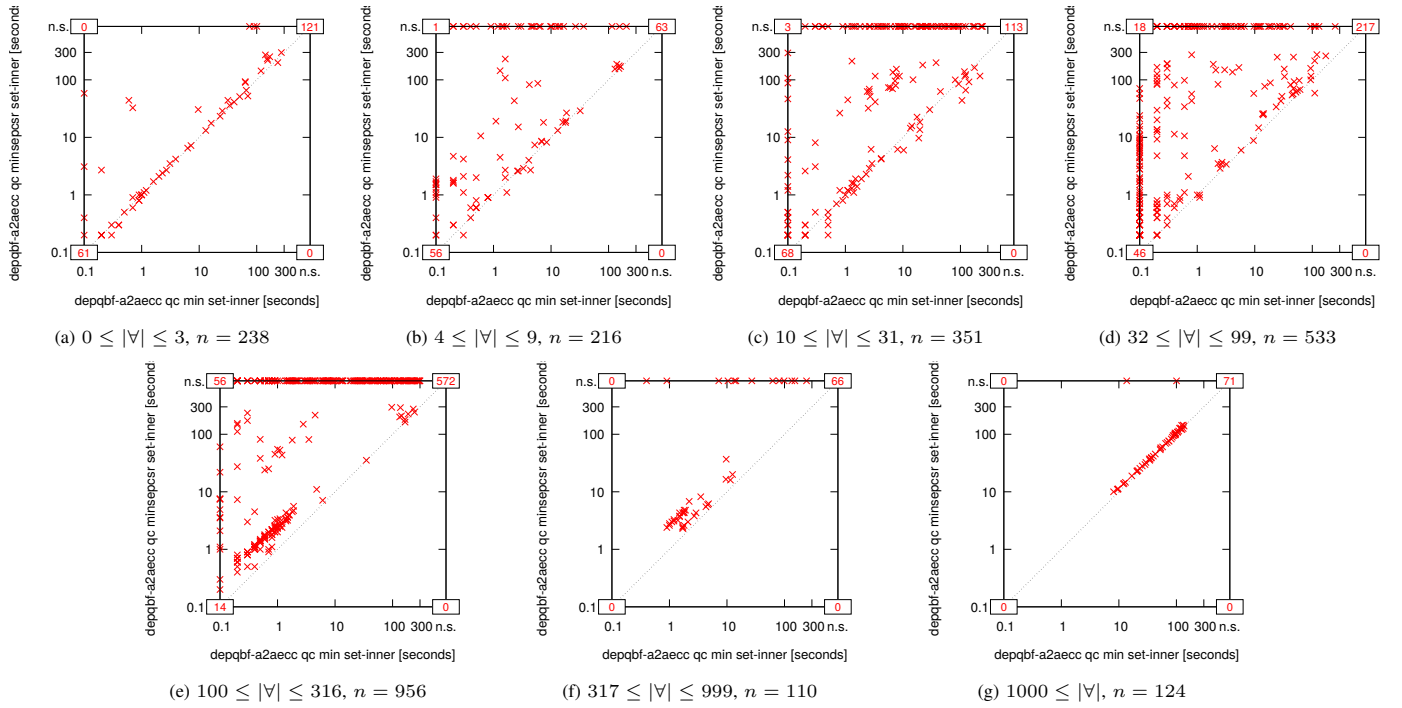


Fig. 1175: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode qc min set-inner partitioned by number of \forall quantified variables (run time in seconds).

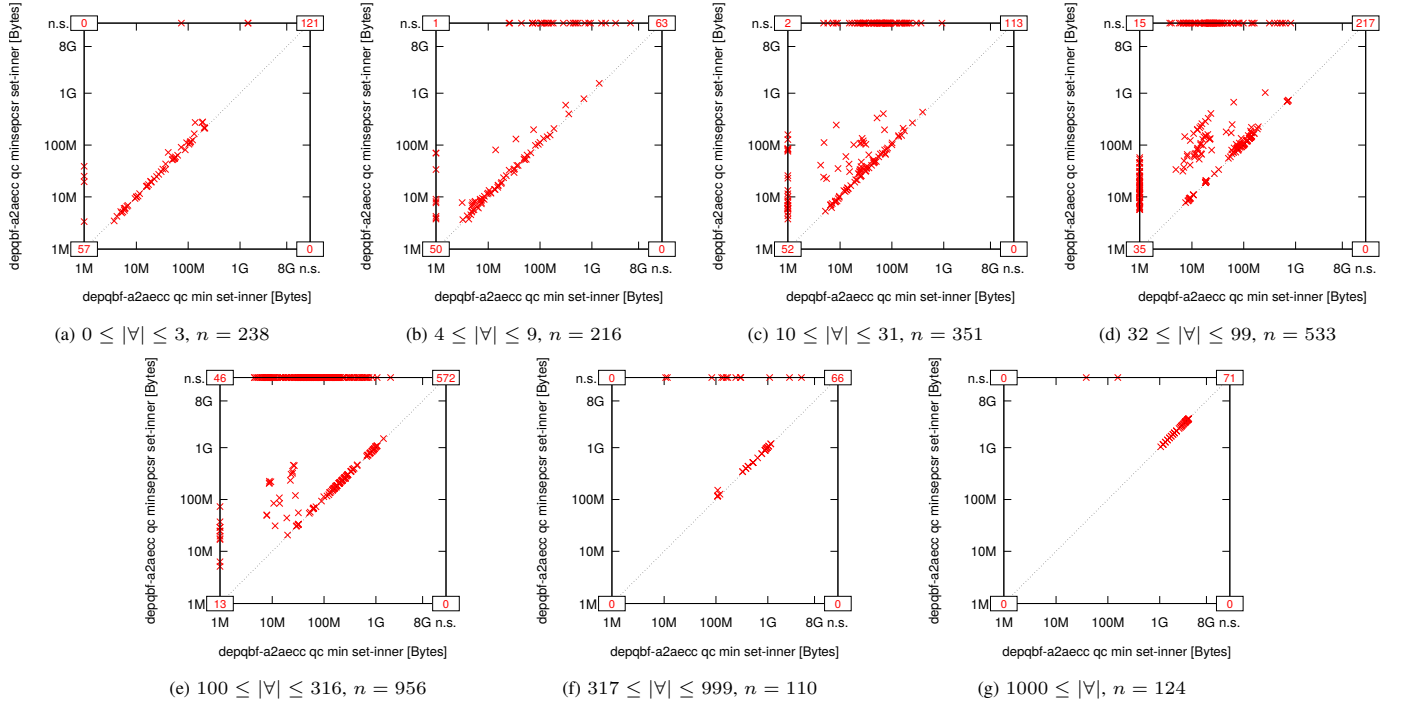


Fig. 1176: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode qc min set-inner partitioned by number of \forall quantified variables (memory usage in Bytes).

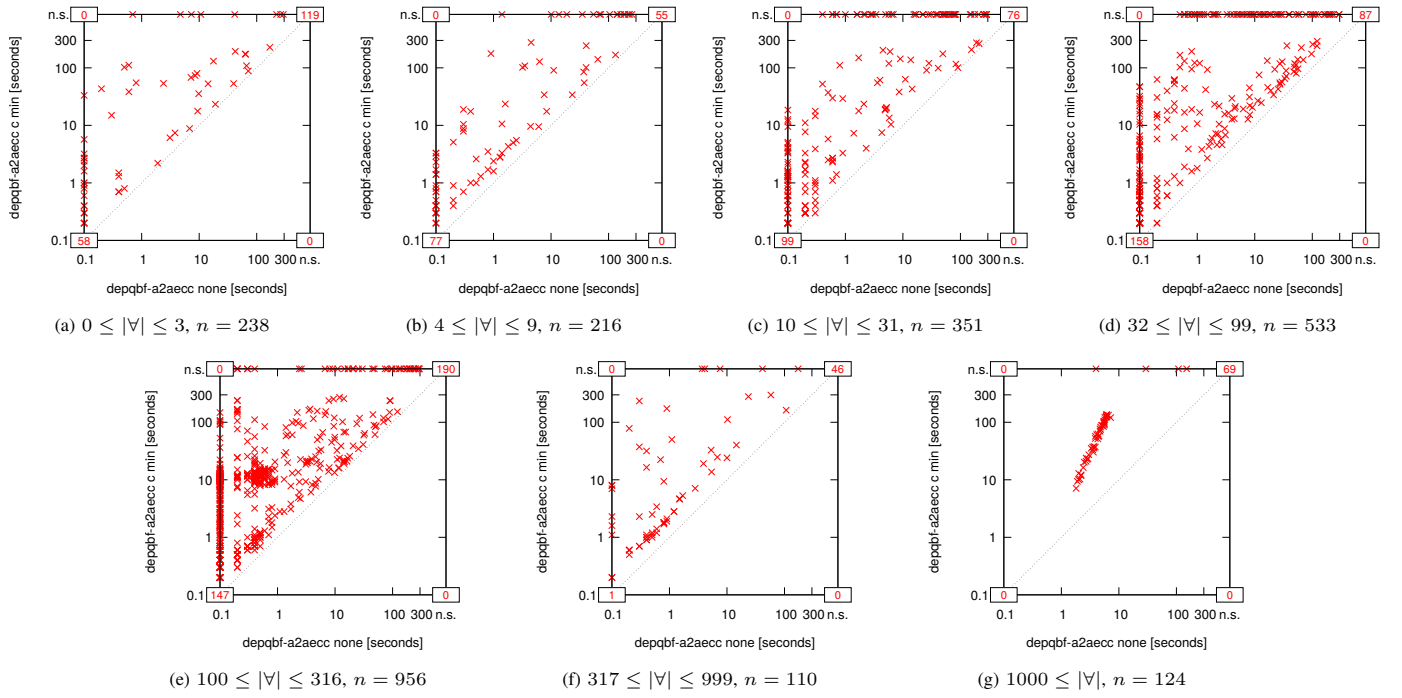


Fig. 1177: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode none partitioned by number of \forall quantified variables (run time in seconds).

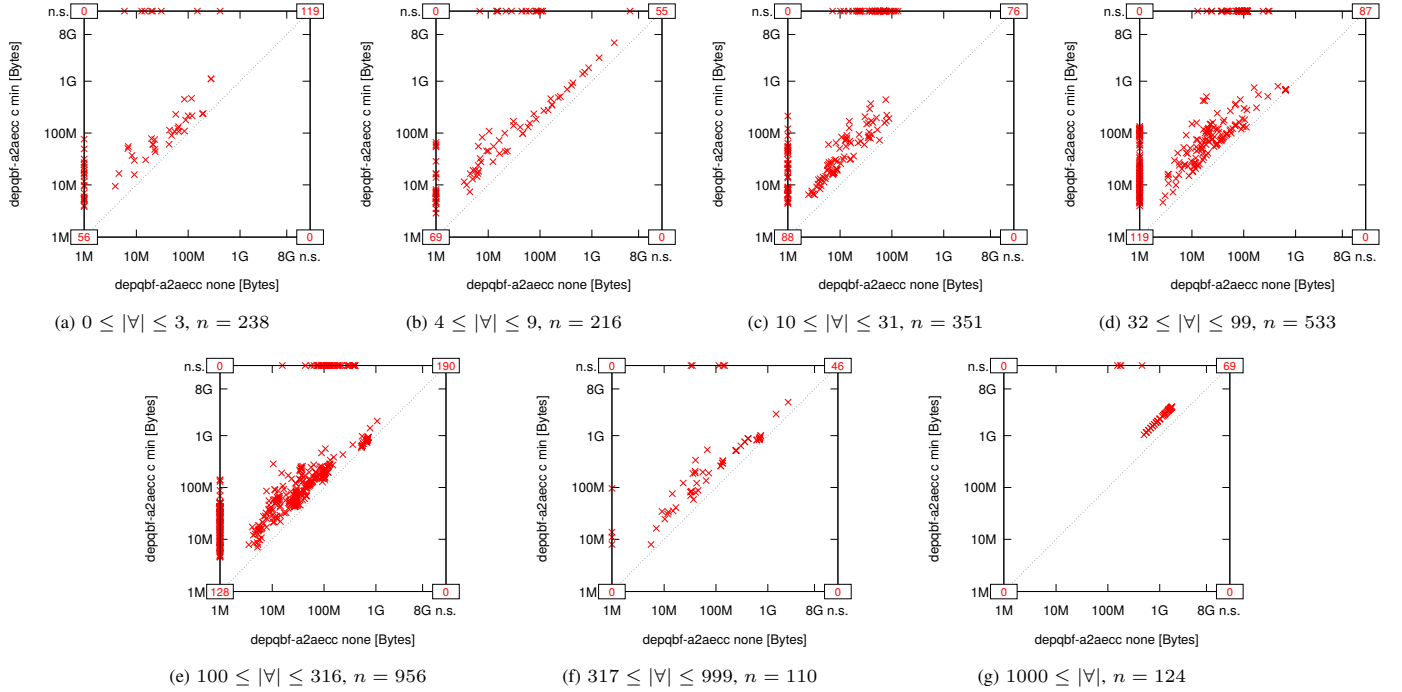


Fig. 1178: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode none partitioned by number of \forall quantified variables (memory usage in Bytes).

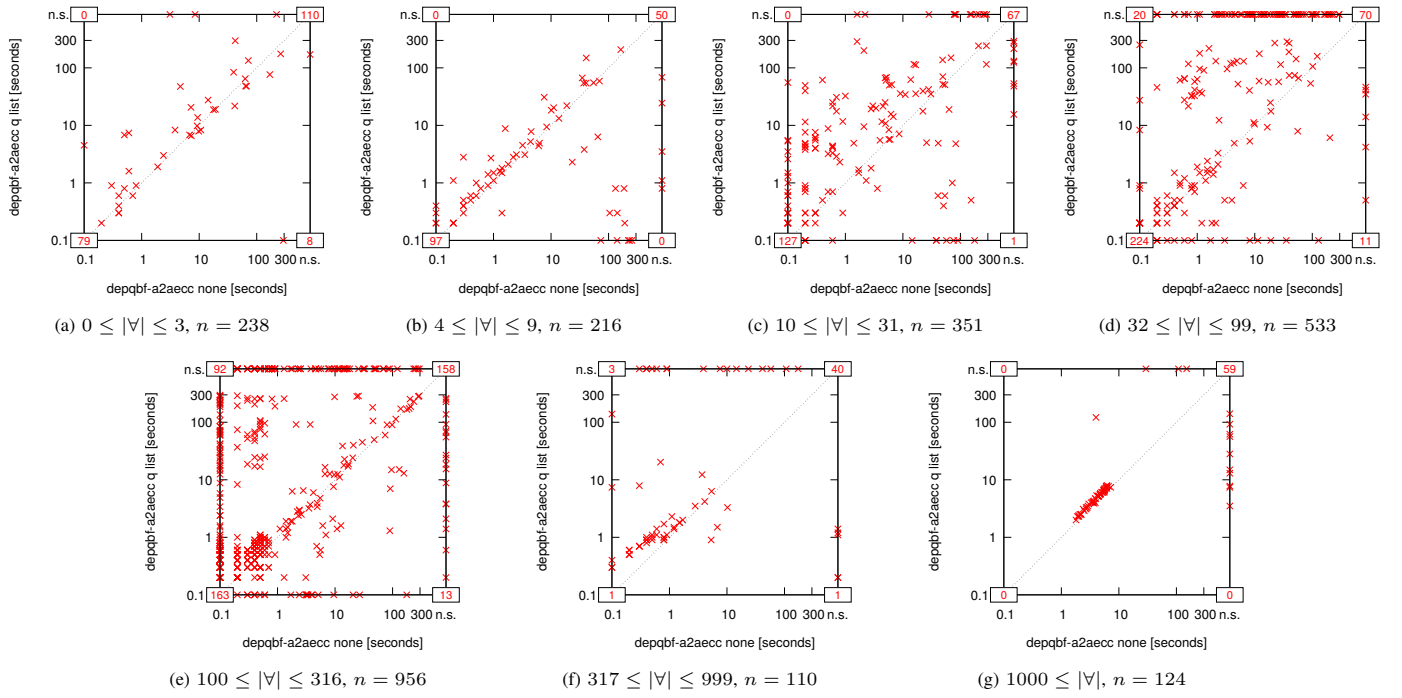


Fig. 1179: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode none partitioned by number of \forall quantified variables (run time in seconds).

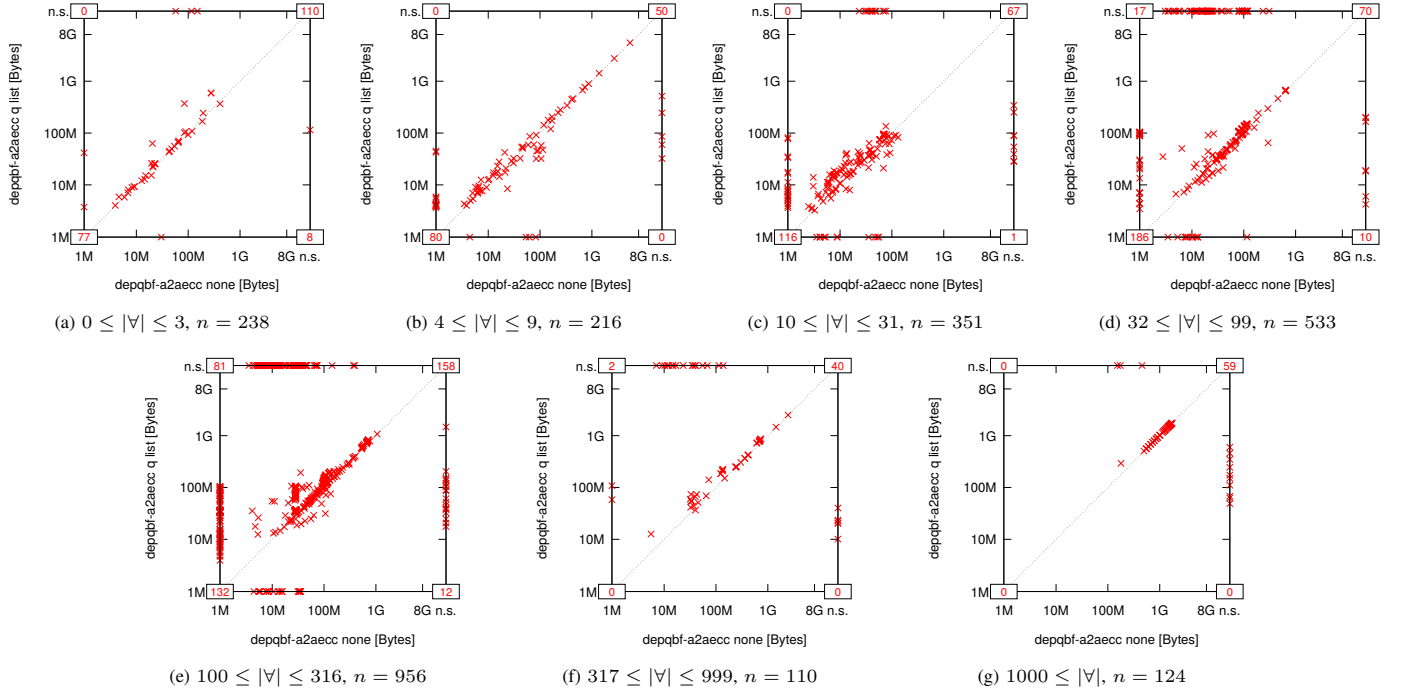


Fig. 1180: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode none partitioned by number of \forall quantified variables (memory usage in Bytes).

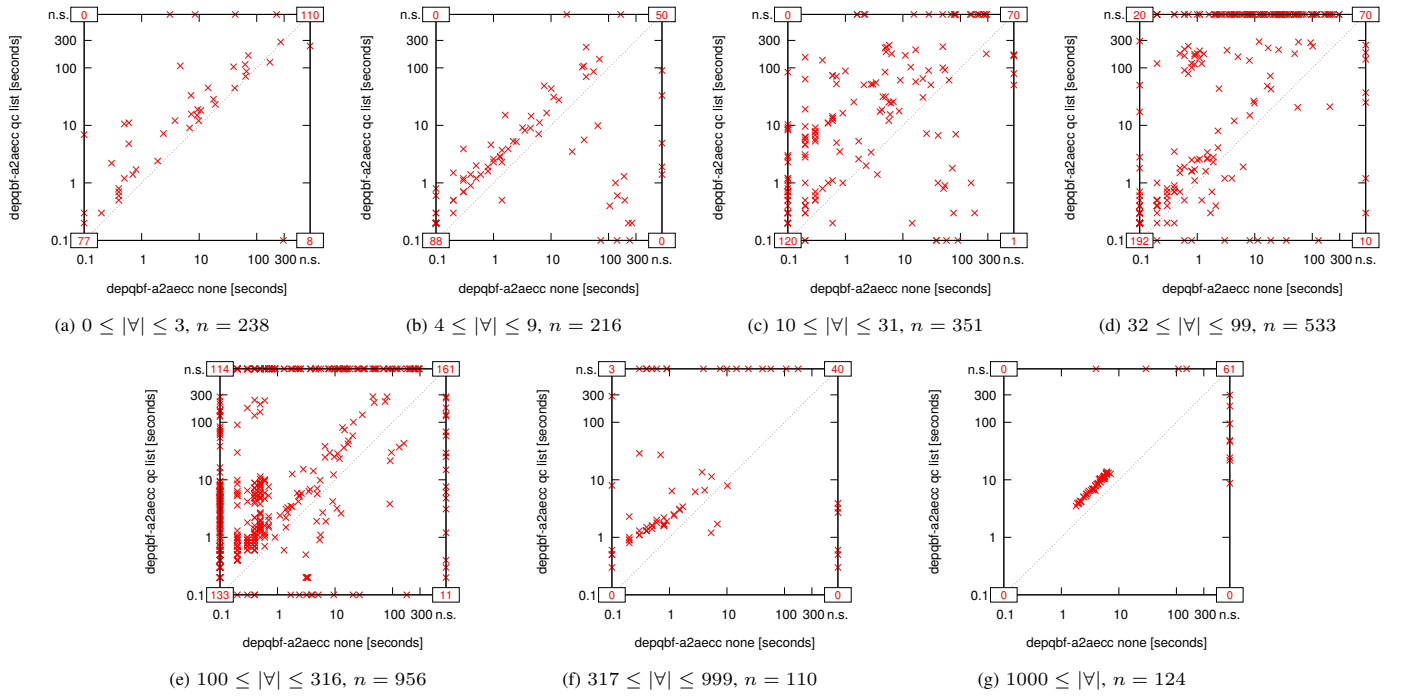


Fig. 1181: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode none partitioned by number of \forall quantified variables (run time in seconds).

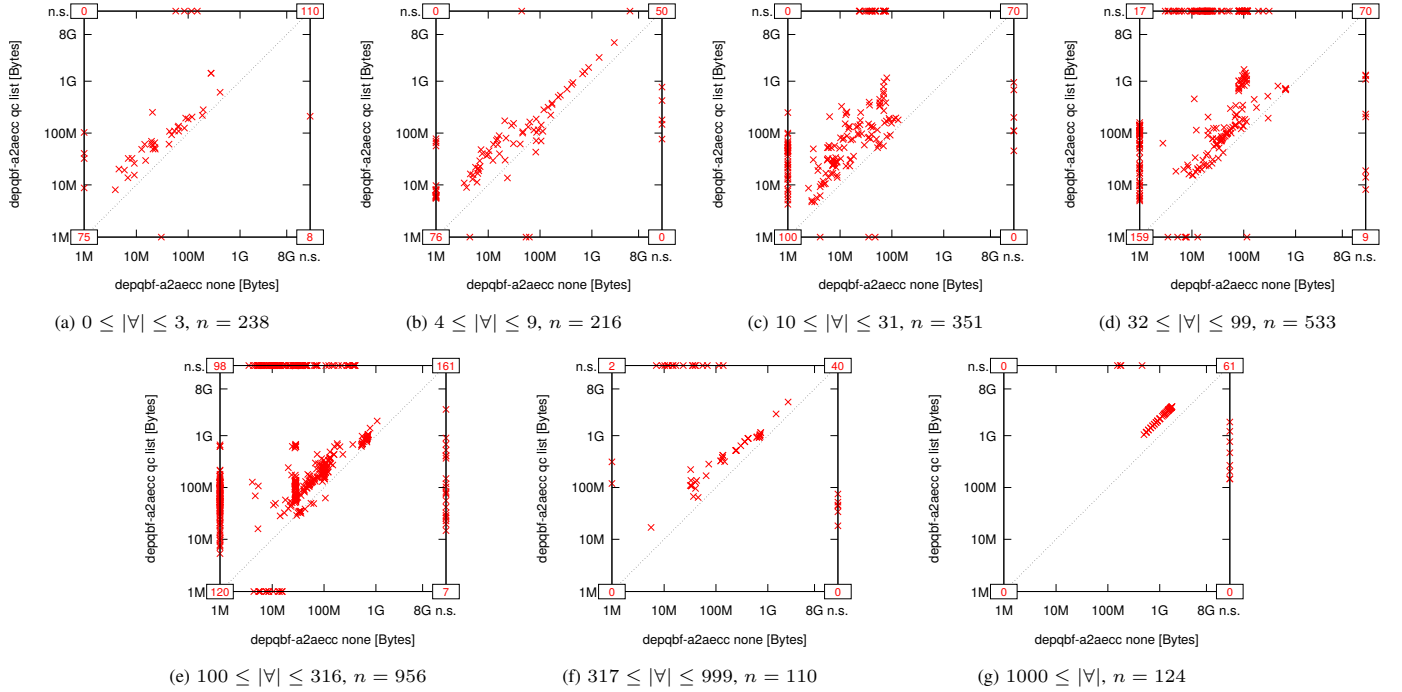


Fig. 1182: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode none partitioned by number of \forall quantified variables (memory usage in Bytes).

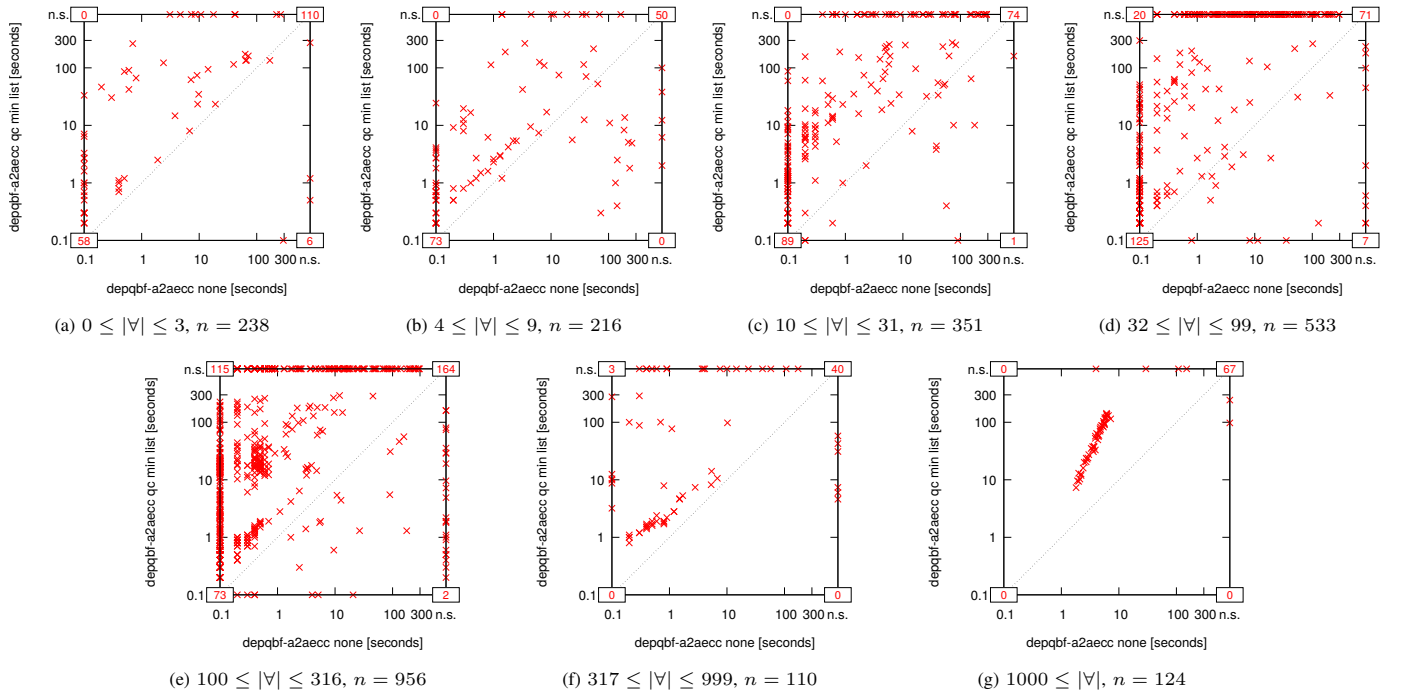


Fig. 1183: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode none partitioned by number of \forall quantified variables (run time in seconds).

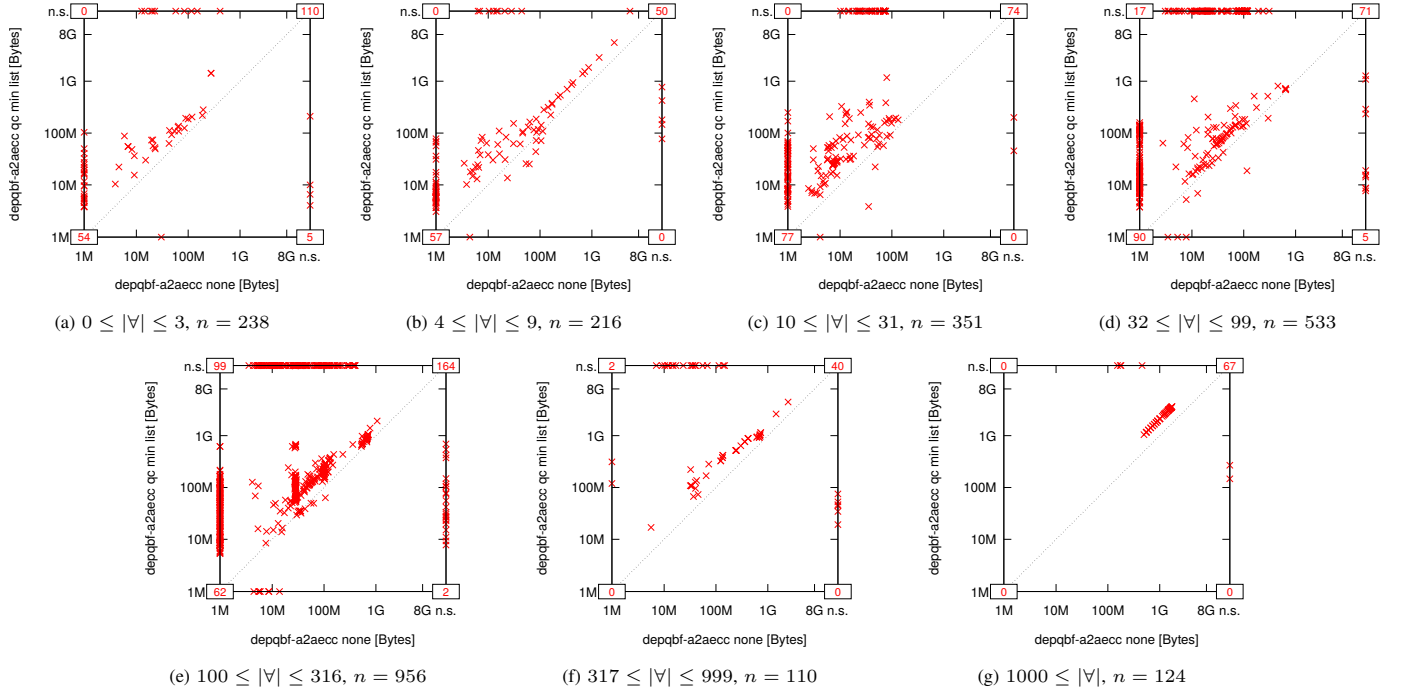


Fig. 1184: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode none partitioned by number of \forall quantified variables (memory usage in Bytes).

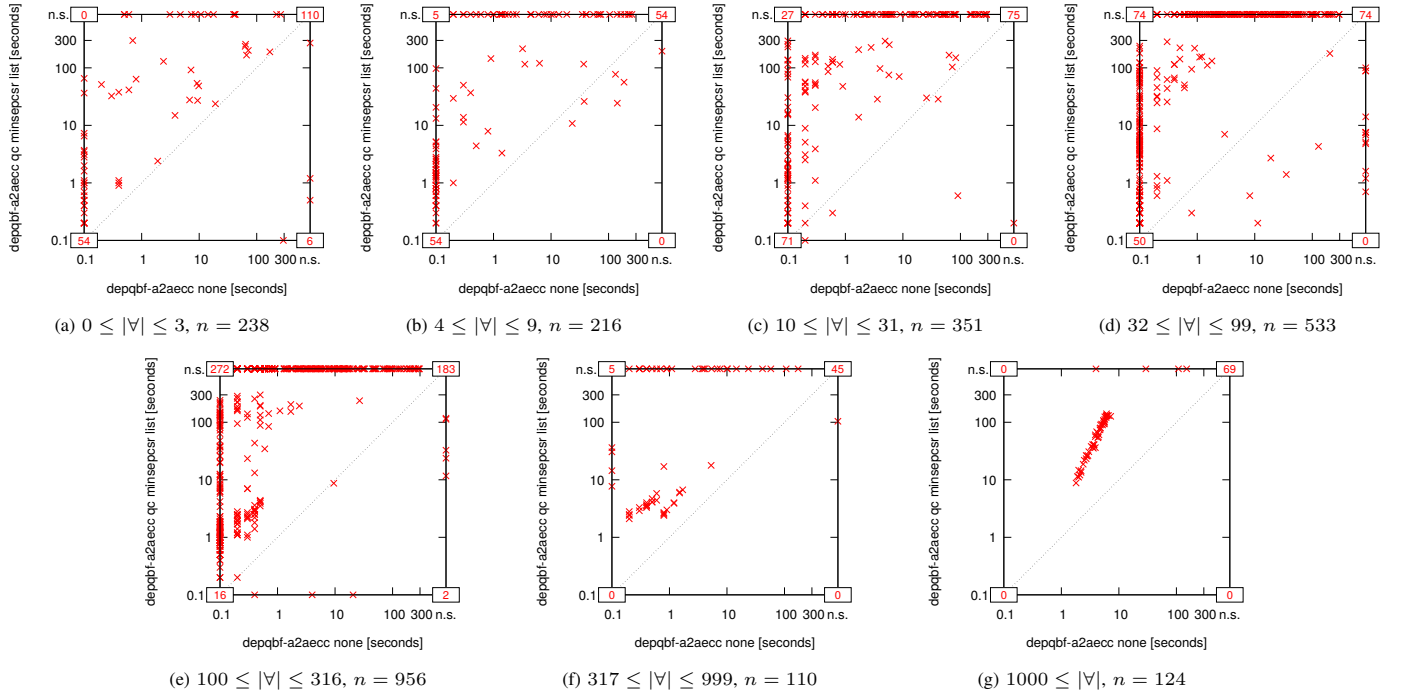


Fig. 1185: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode none partitioned by number of \forall quantified variables (run time in seconds).

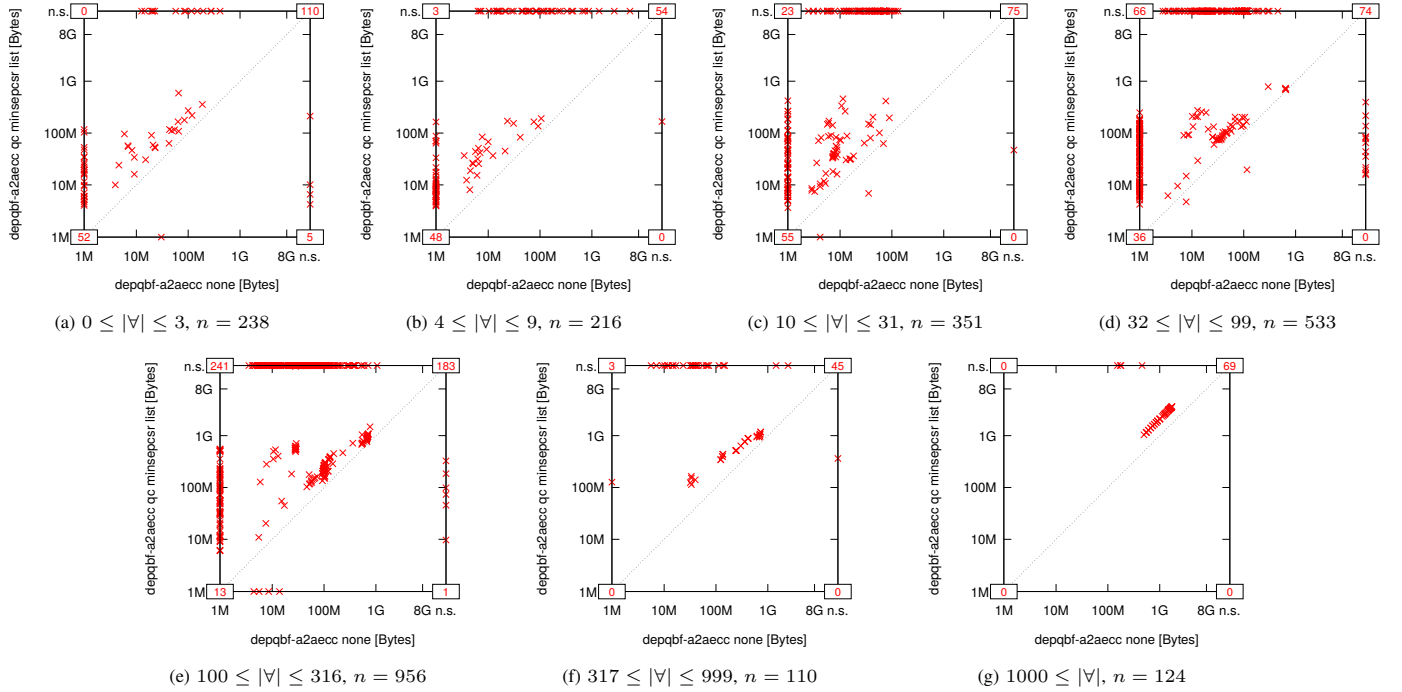


Fig. 1186: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode none partitioned by number of \forall quantified variables (memory usage in Bytes).

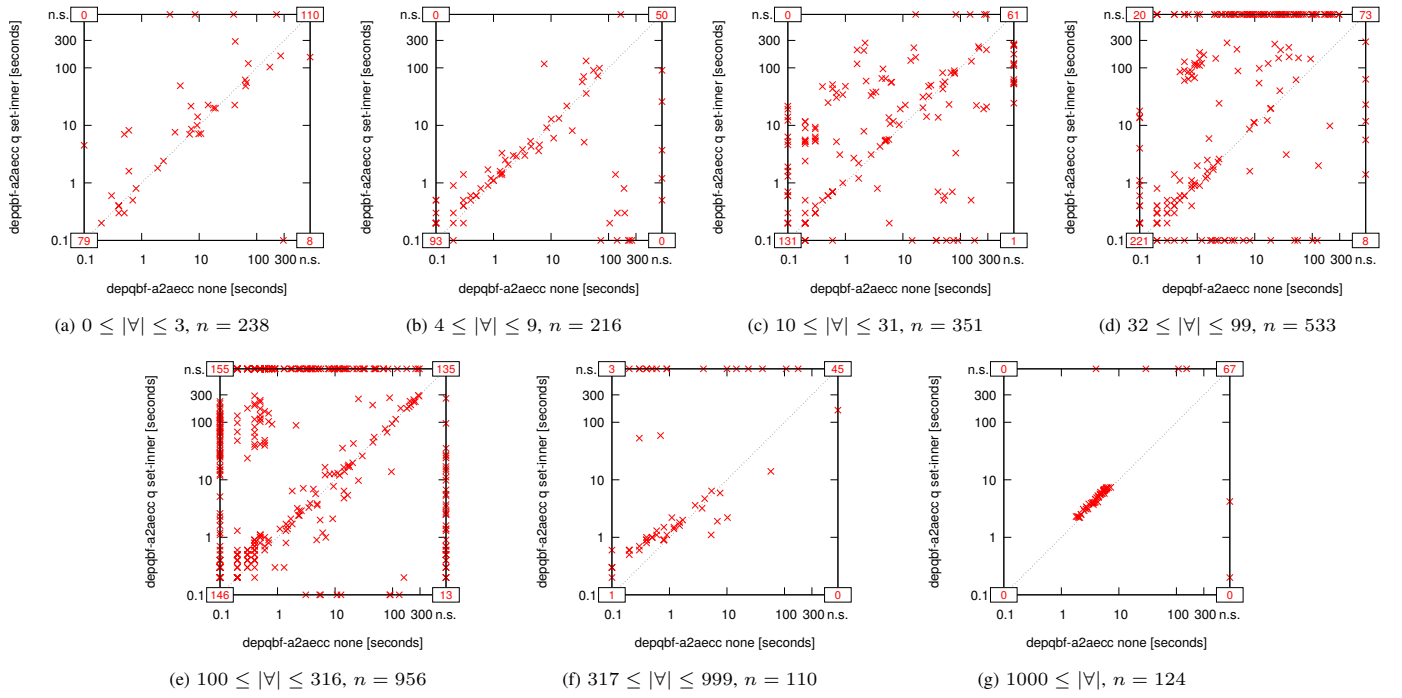


Fig. 1187: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode none partitioned by number of \forall quantified variables (run time in seconds).

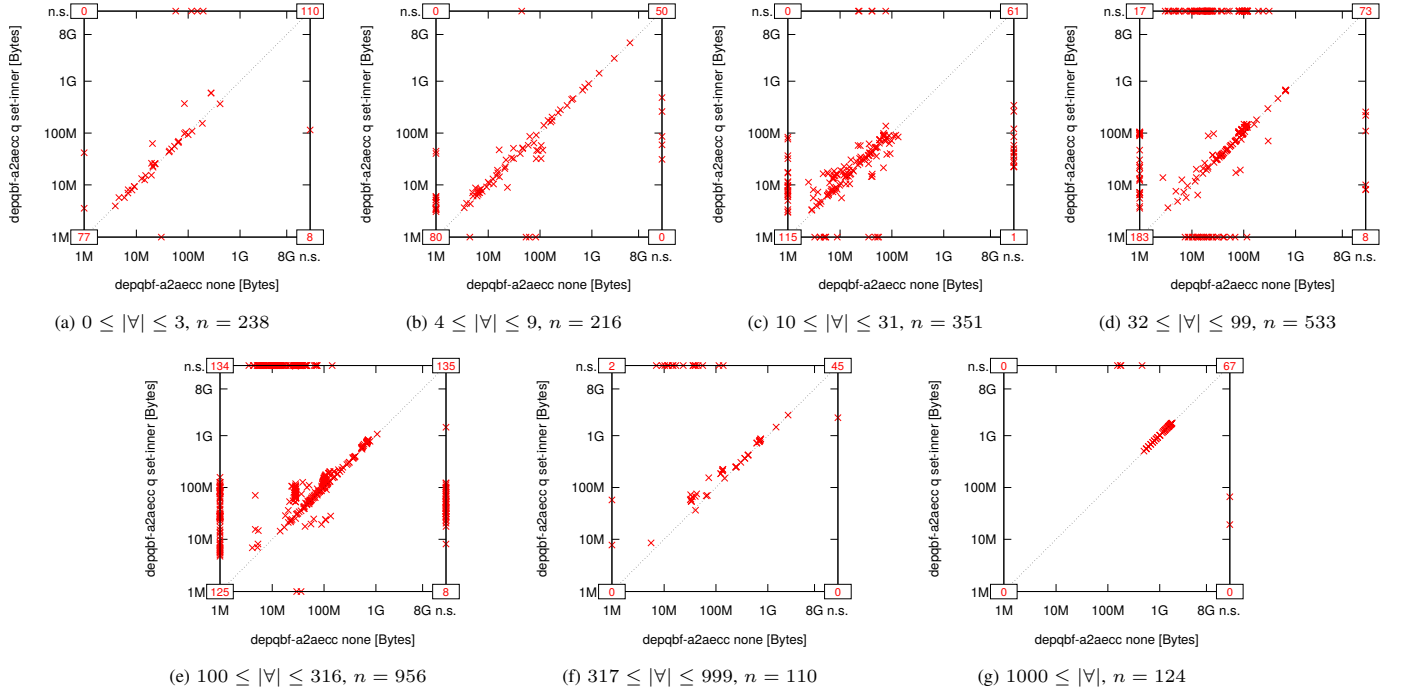


Fig. 1188: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode none partitioned by number of \forall quantified variables (memory usage in Bytes).

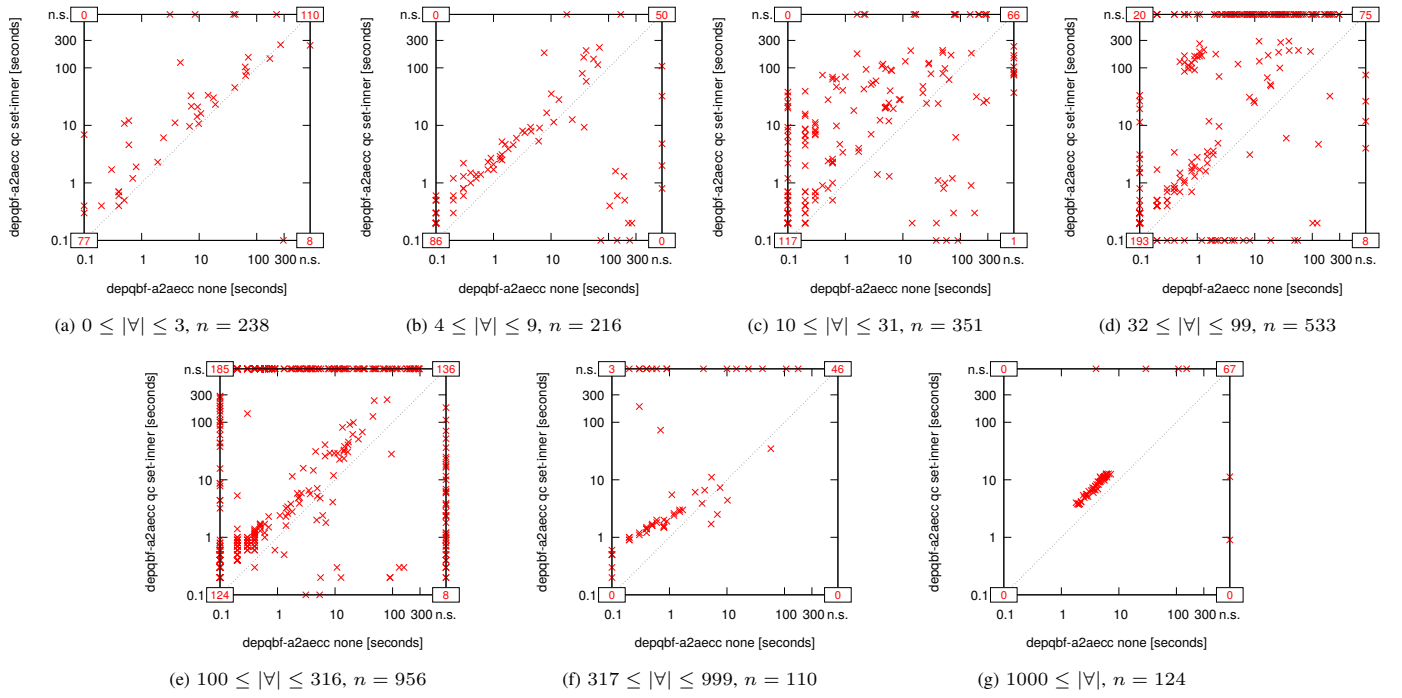


Fig. 1189: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode none partitioned by number of \forall quantified variables (run time in seconds).

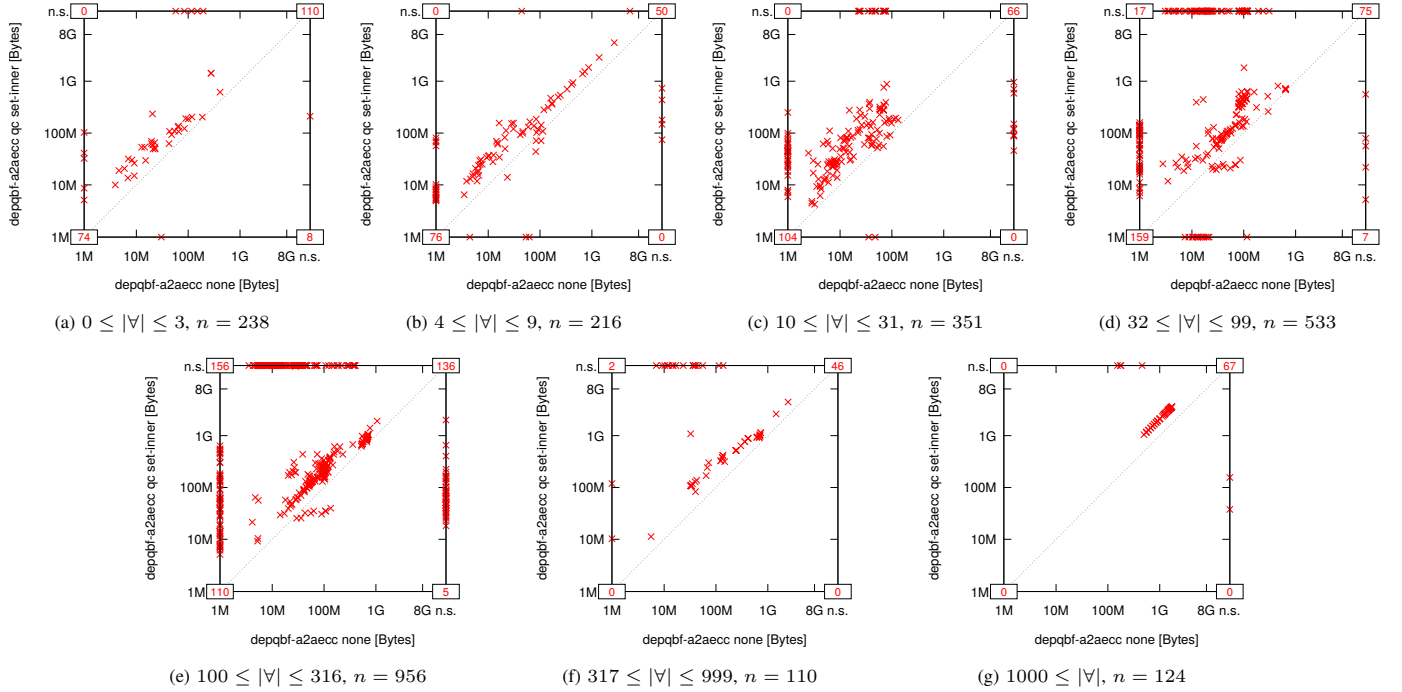


Fig. 1190: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode none partitioned by number of \forall quantified variables (memory usage in Bytes).

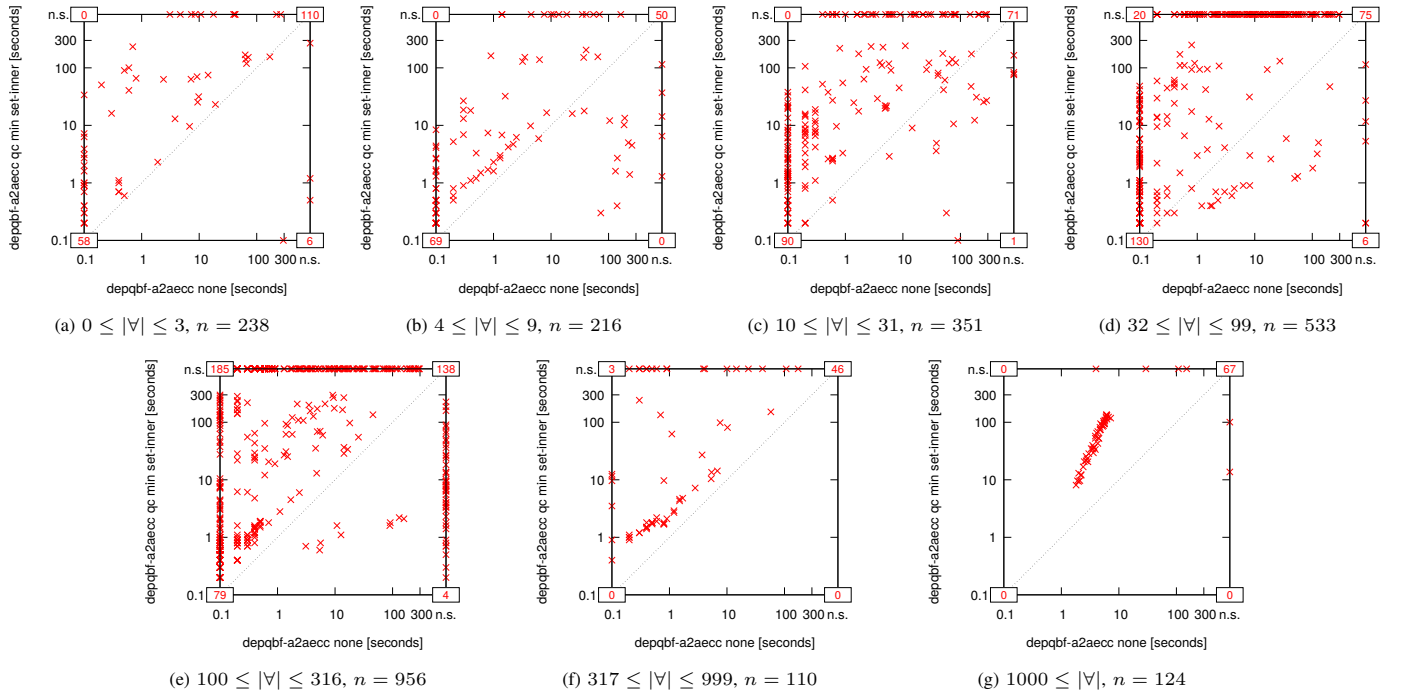


Fig. 1191: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode none partitioned by number of \forall quantified variables (run time in seconds).

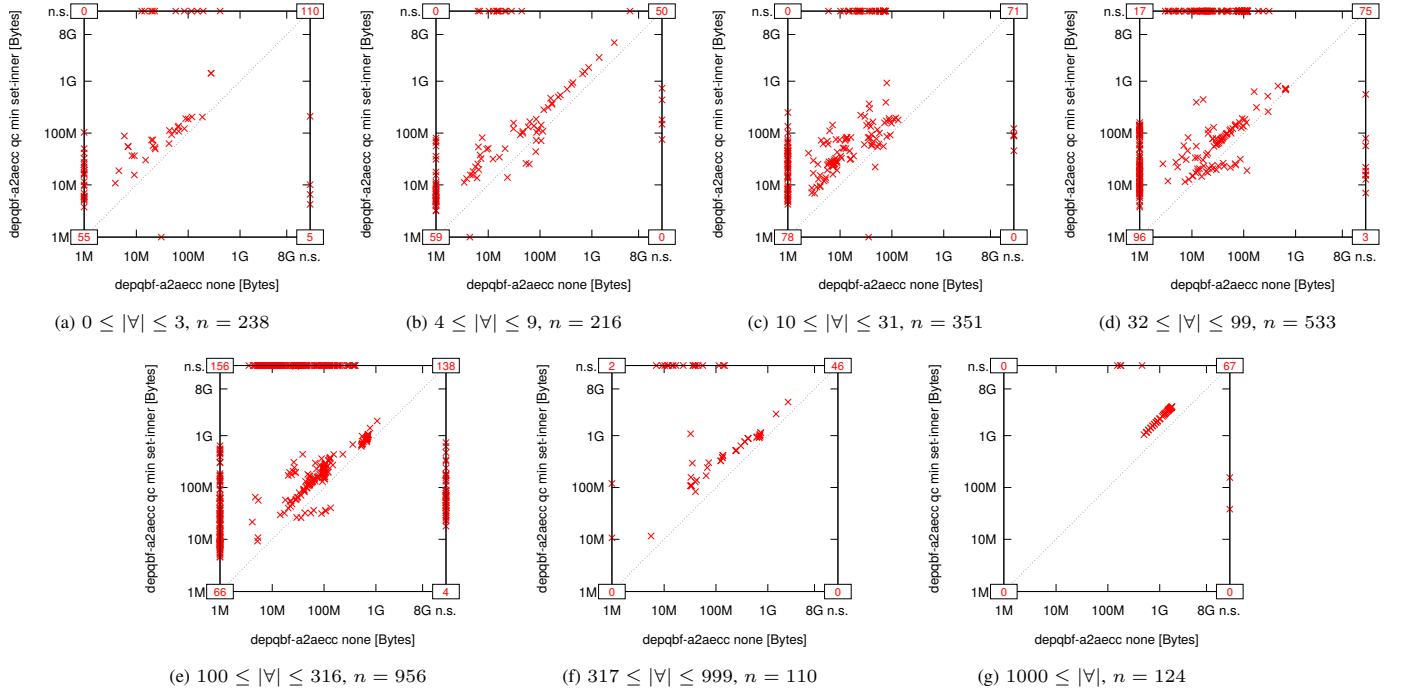


Fig. 1192: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode none partitioned by number of \forall quantified variables (memory usage in Bytes).

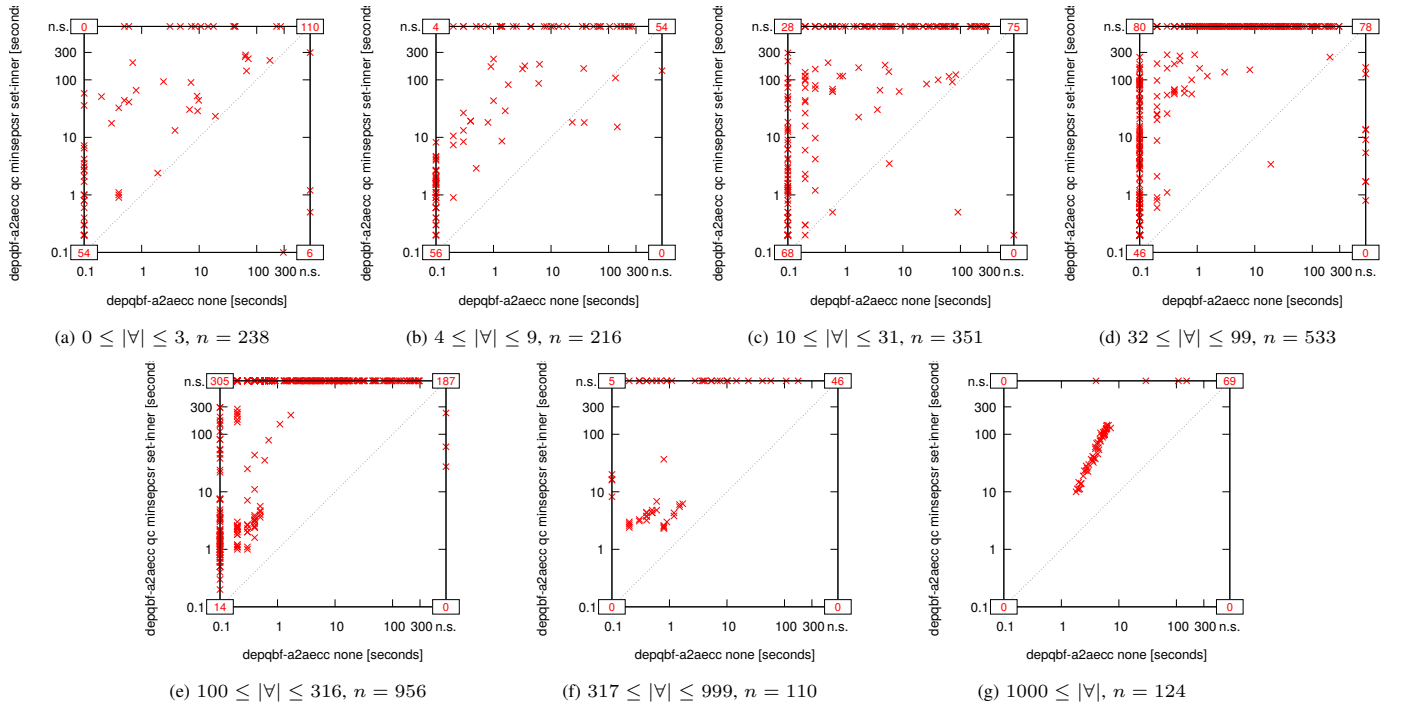


Fig. 1193: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode none partitioned by number of \forall quantified variables (run time in seconds).

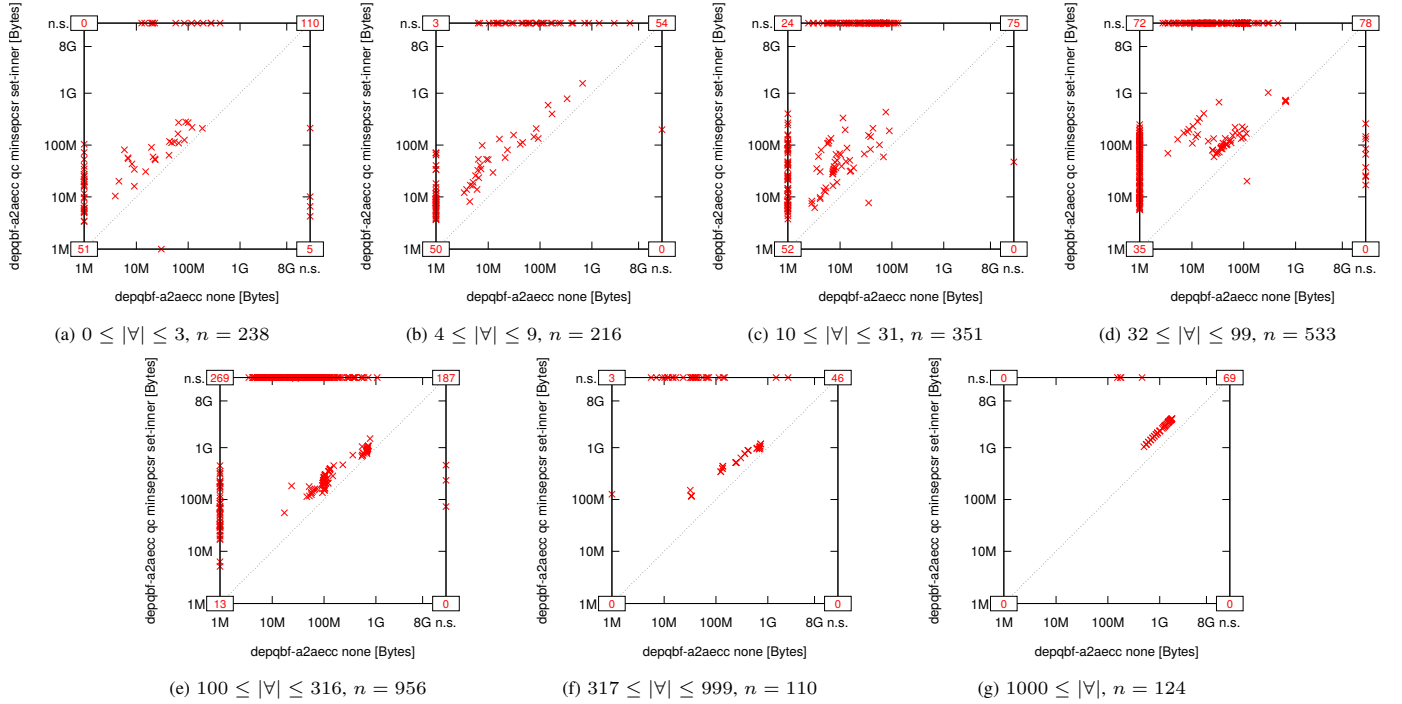


Fig. 1194: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode none partitioned by number of \forall quantified variables (memory usage in Bytes).

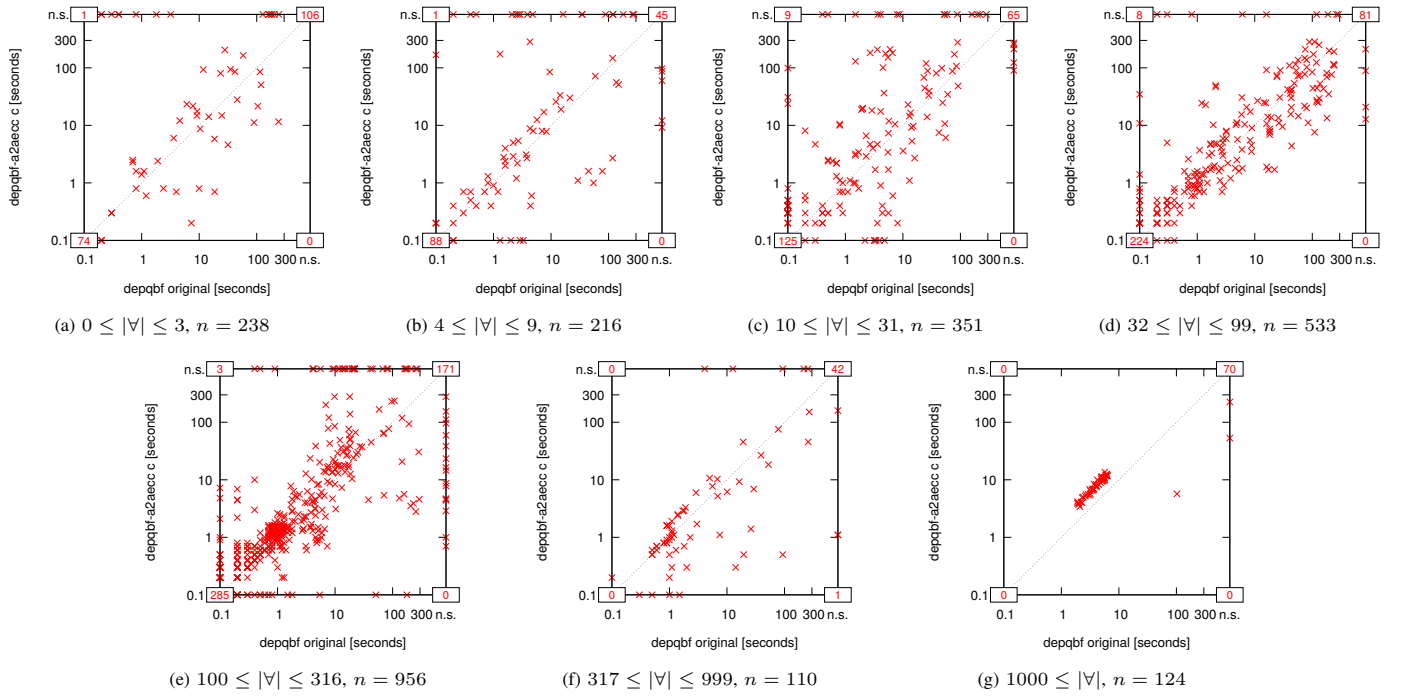


Fig. 1195: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c versus mode original partitioned by number of \forall quantified variables (run time in seconds).

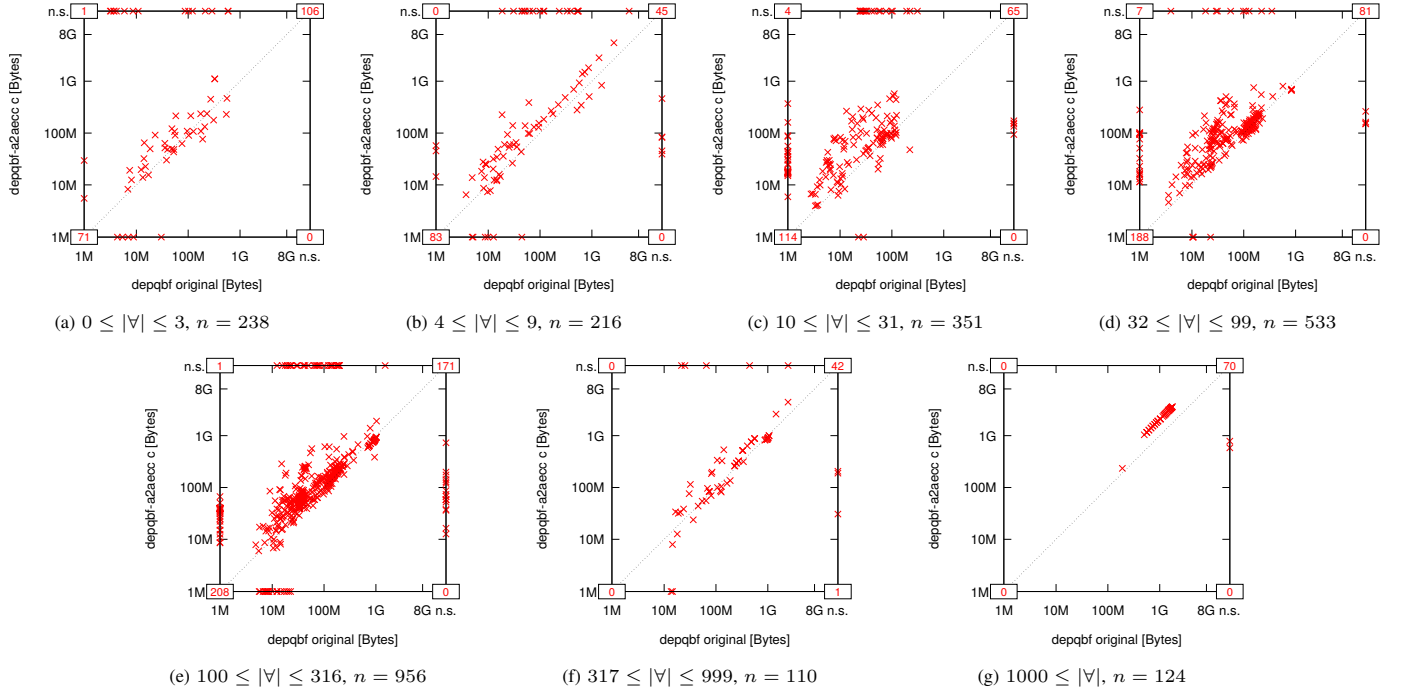


Fig. 1196: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c versus mode original partitioned by number of \forall quantified variables (memory usage in Bytes).

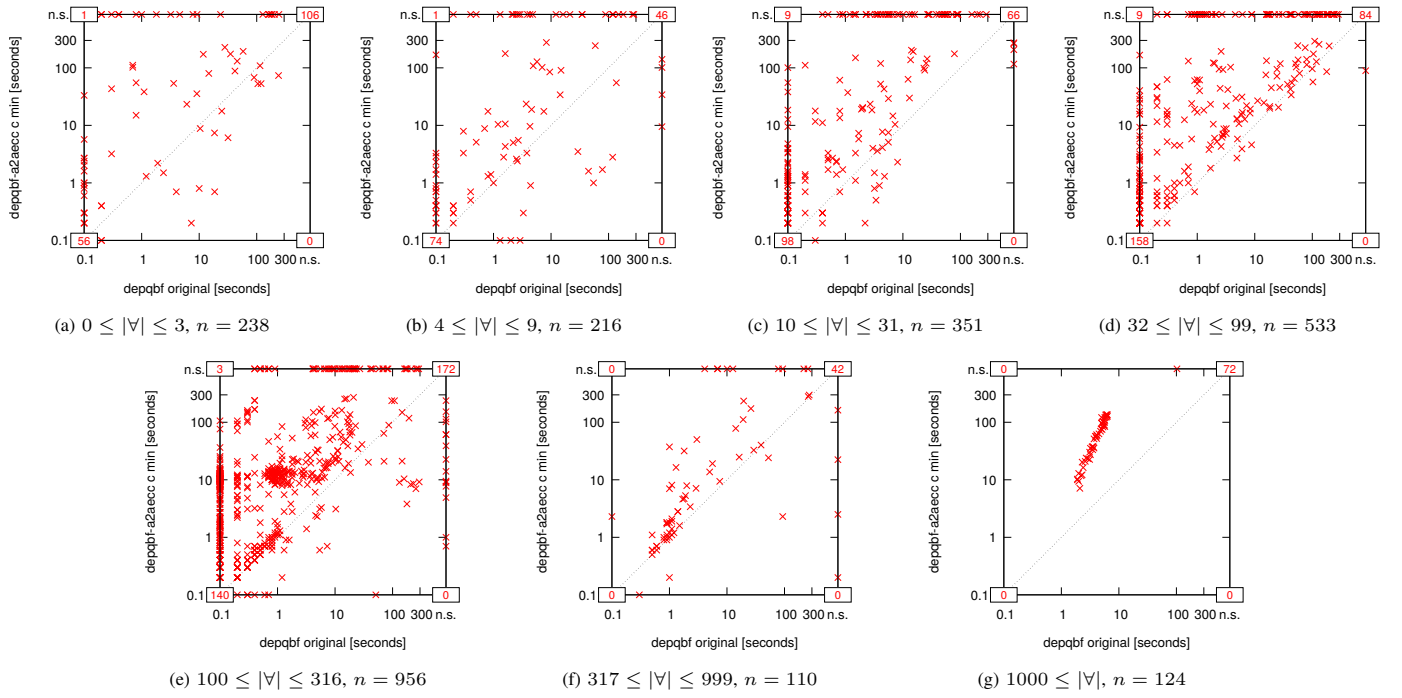


Fig. 1197: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode original partitioned by number of \forall quantified variables (run time in seconds).

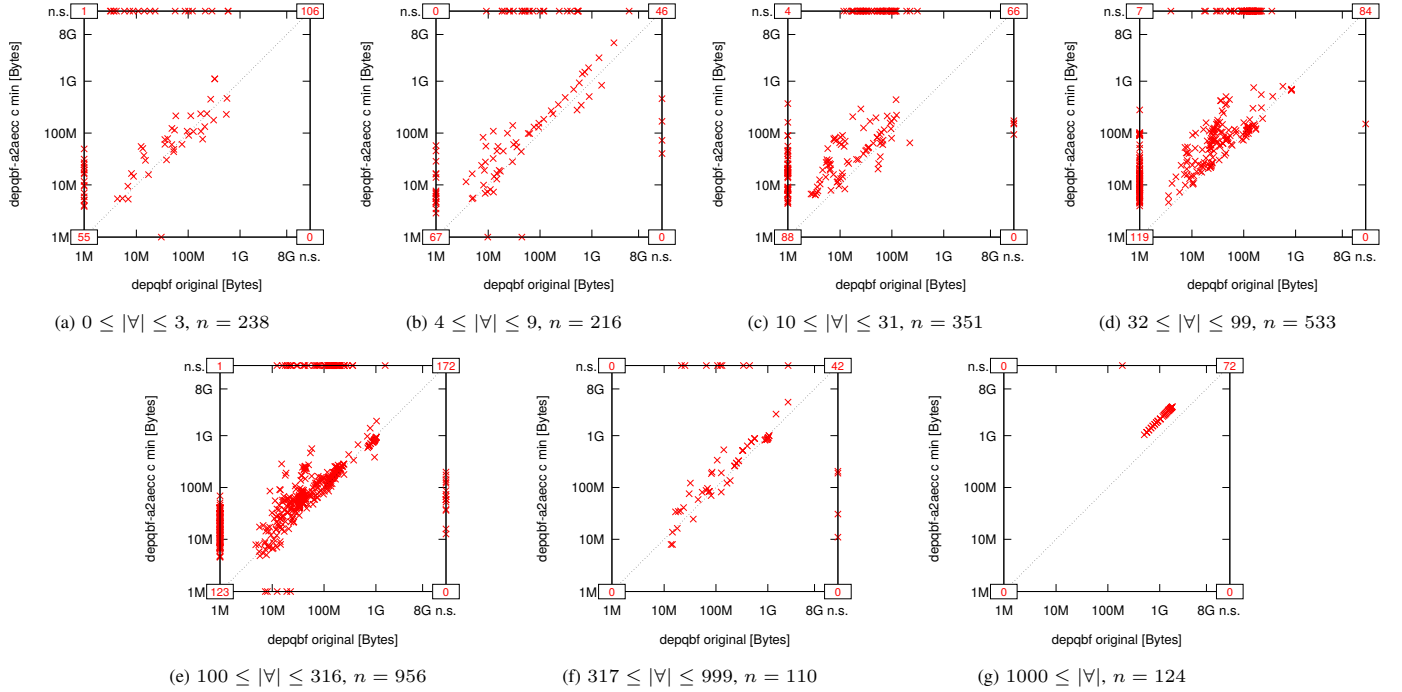


Fig. 1198: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode original partitioned by number of \forall quantified variables (memory usage in Bytes).

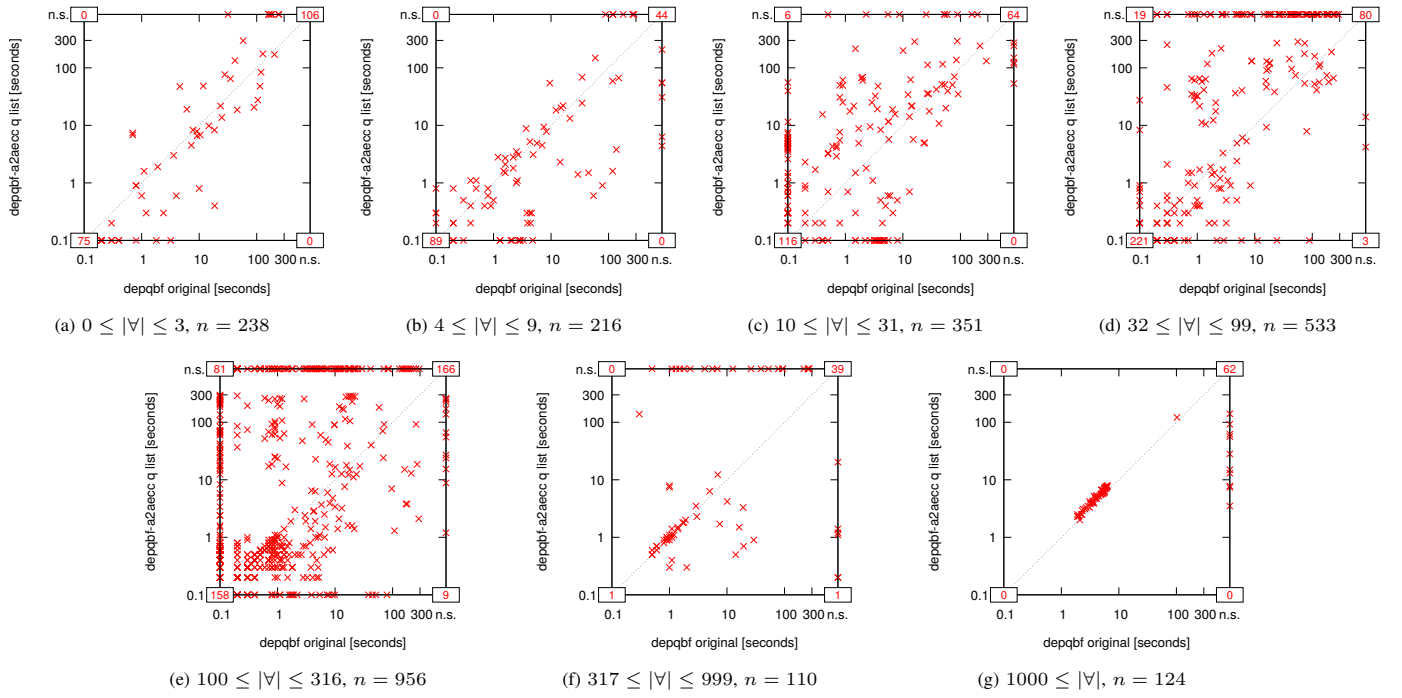


Fig. 1199: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode original partitioned by number of \forall quantified variables (run time in seconds).

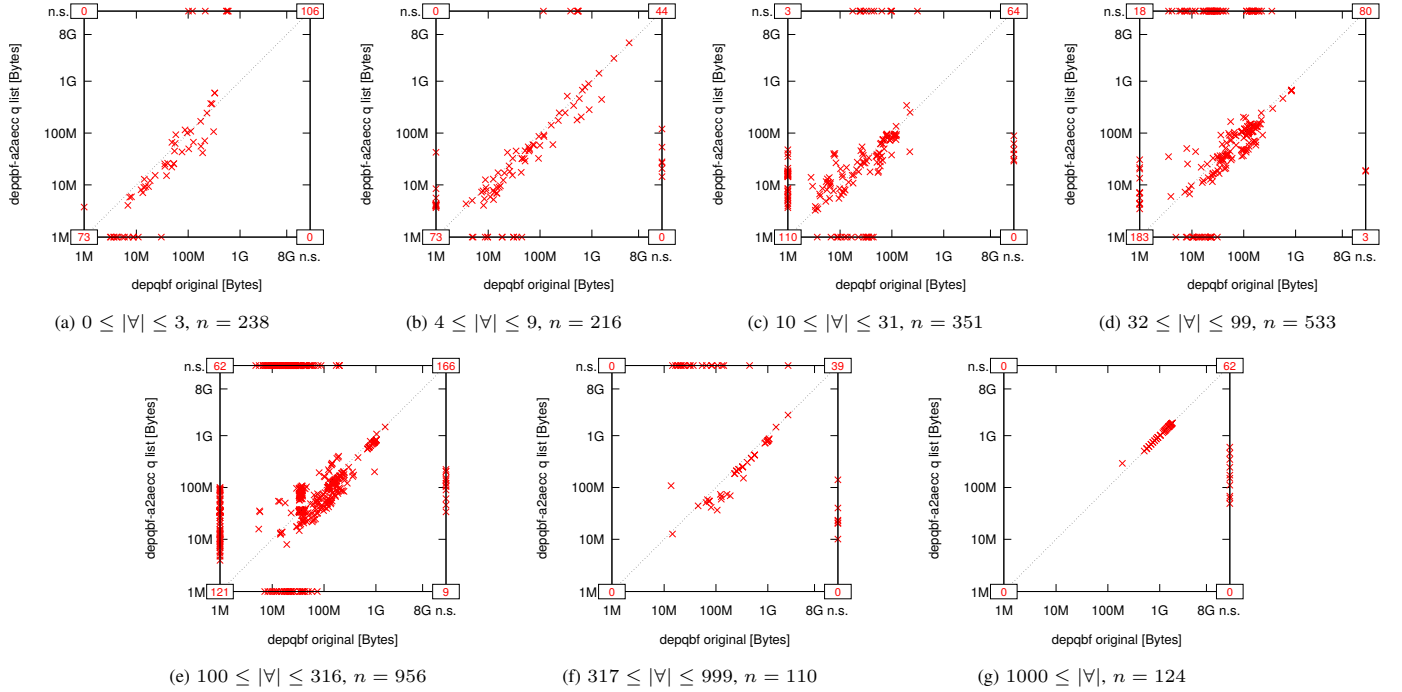


Fig. 1200: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode original partitioned by number of \forall quantified variables (memory usage in Bytes).

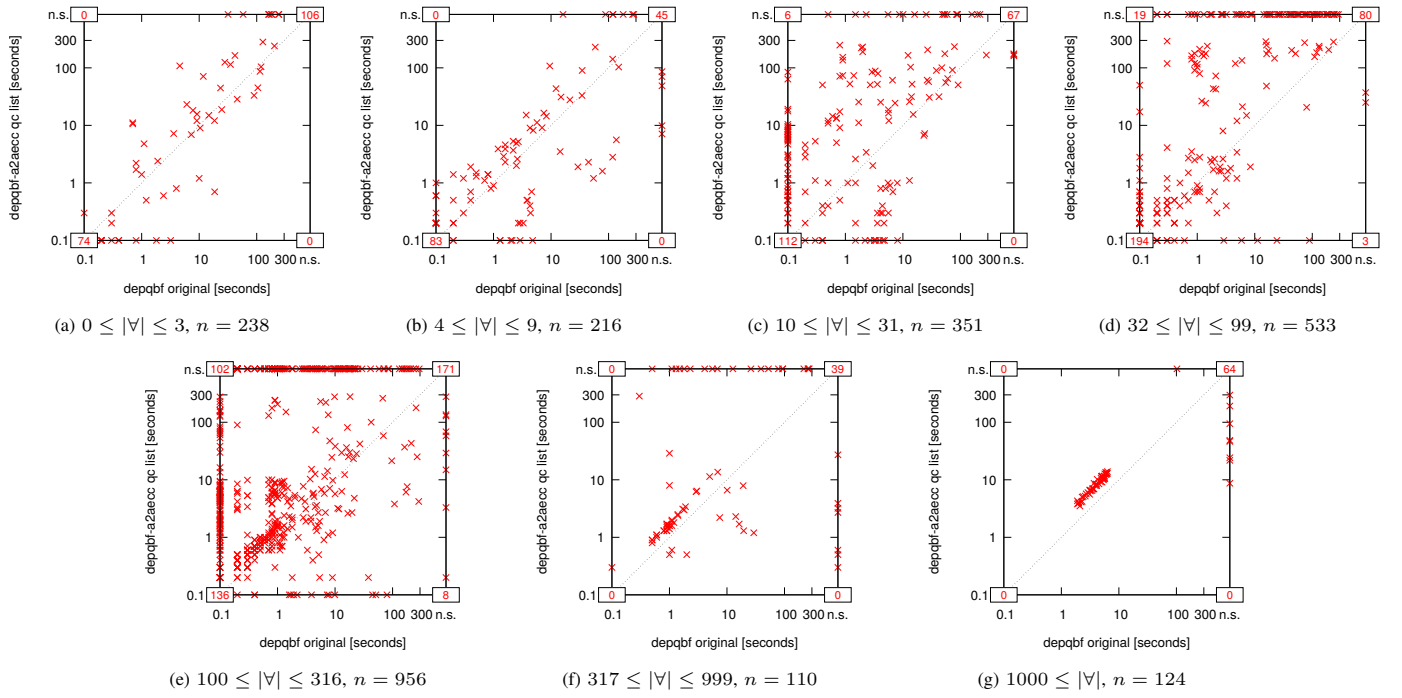


Fig. 1201: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode original partitioned by number of \forall quantified variables (run time in seconds).

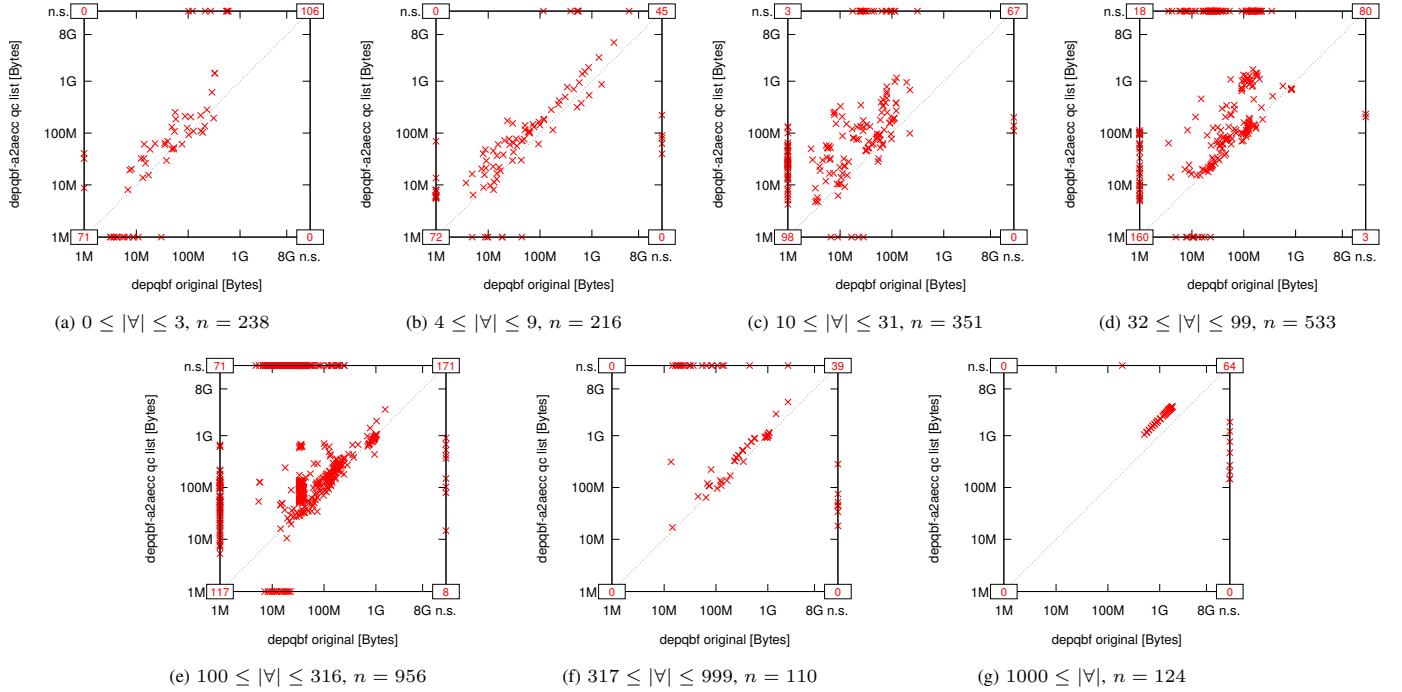


Fig. 1202: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode original partitioned by number of \forall quantified variables (memory usage in Bytes).

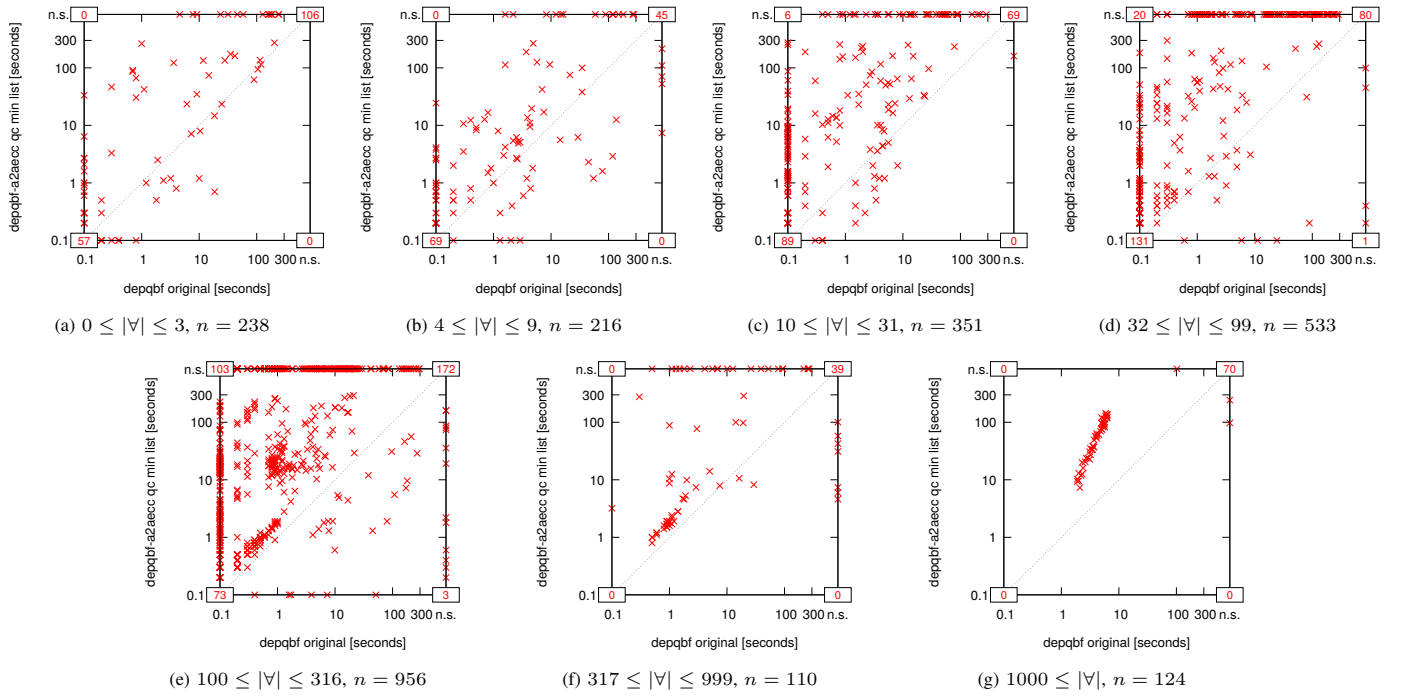


Fig. 1203: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode original partitioned by number of \forall quantified variables (run time in seconds).

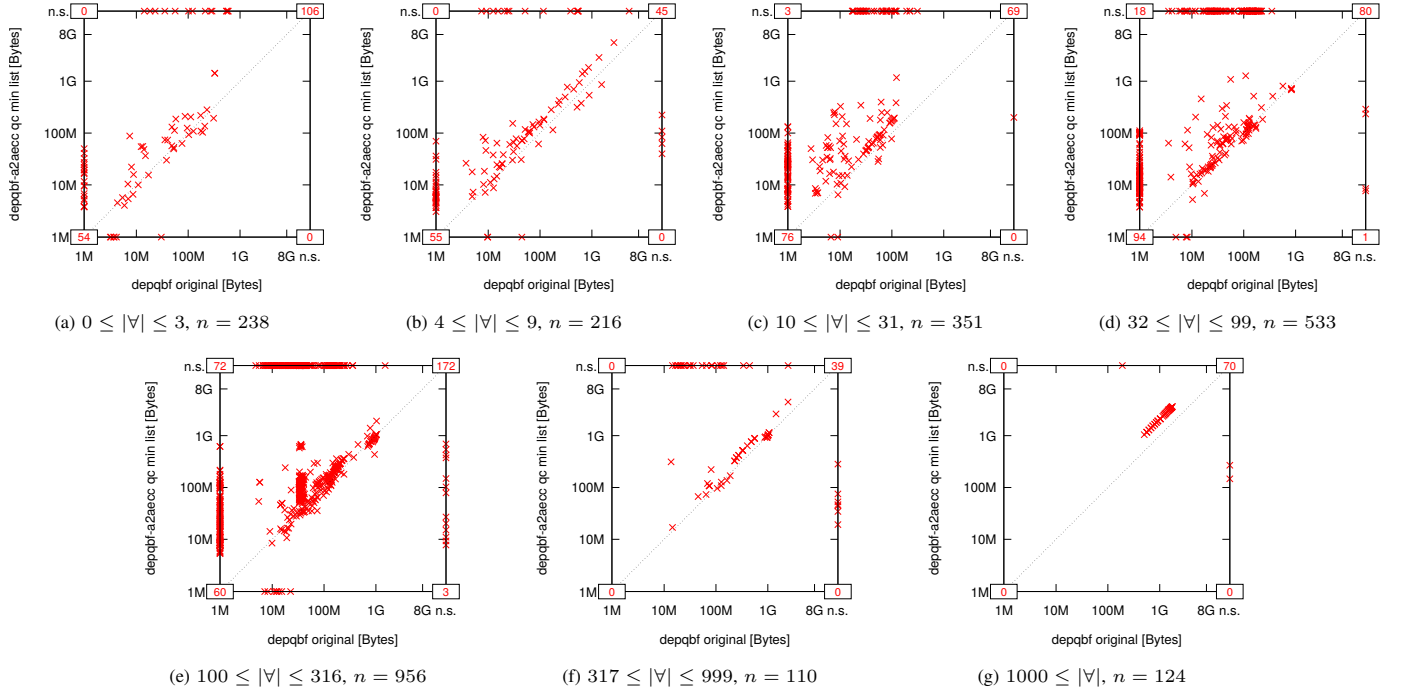


Fig. 1204: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode original partitioned by number of \forall quantified variables (memory usage in Bytes).

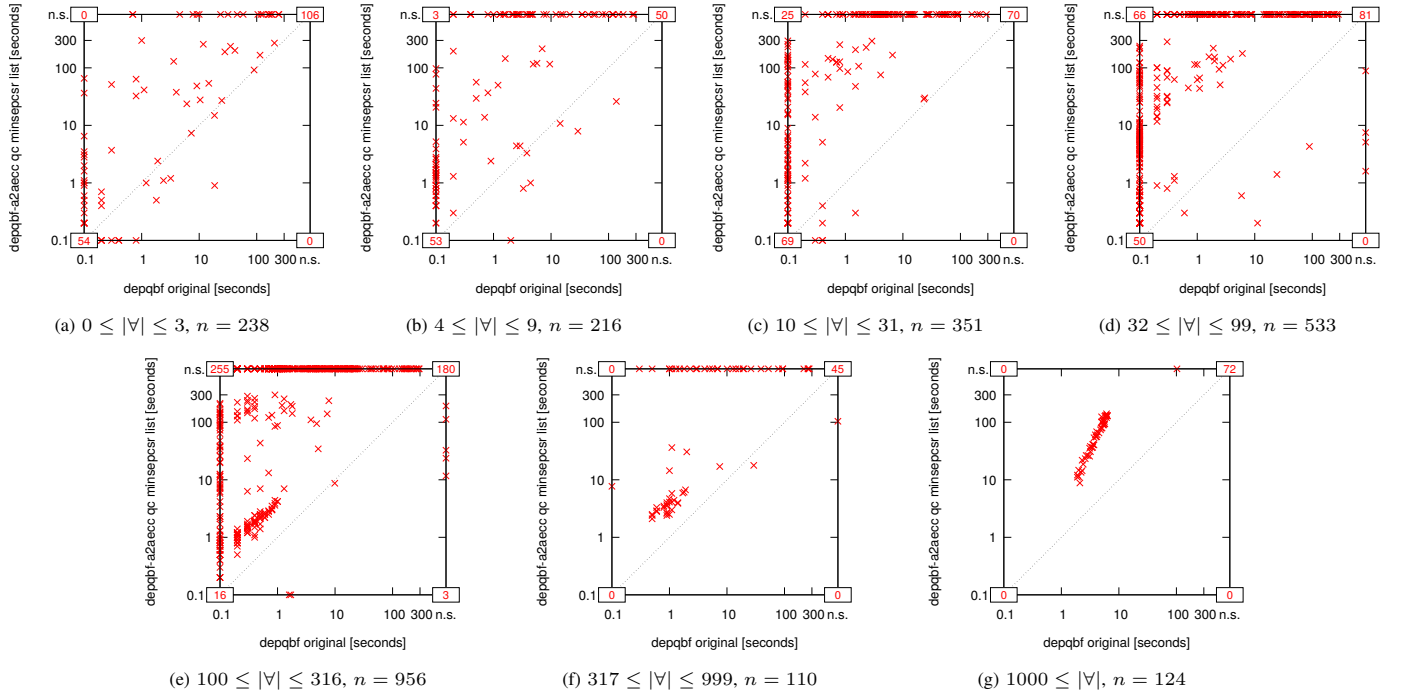


Fig. 1205: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode original partitioned by number of \forall quantified variables (run time in seconds).

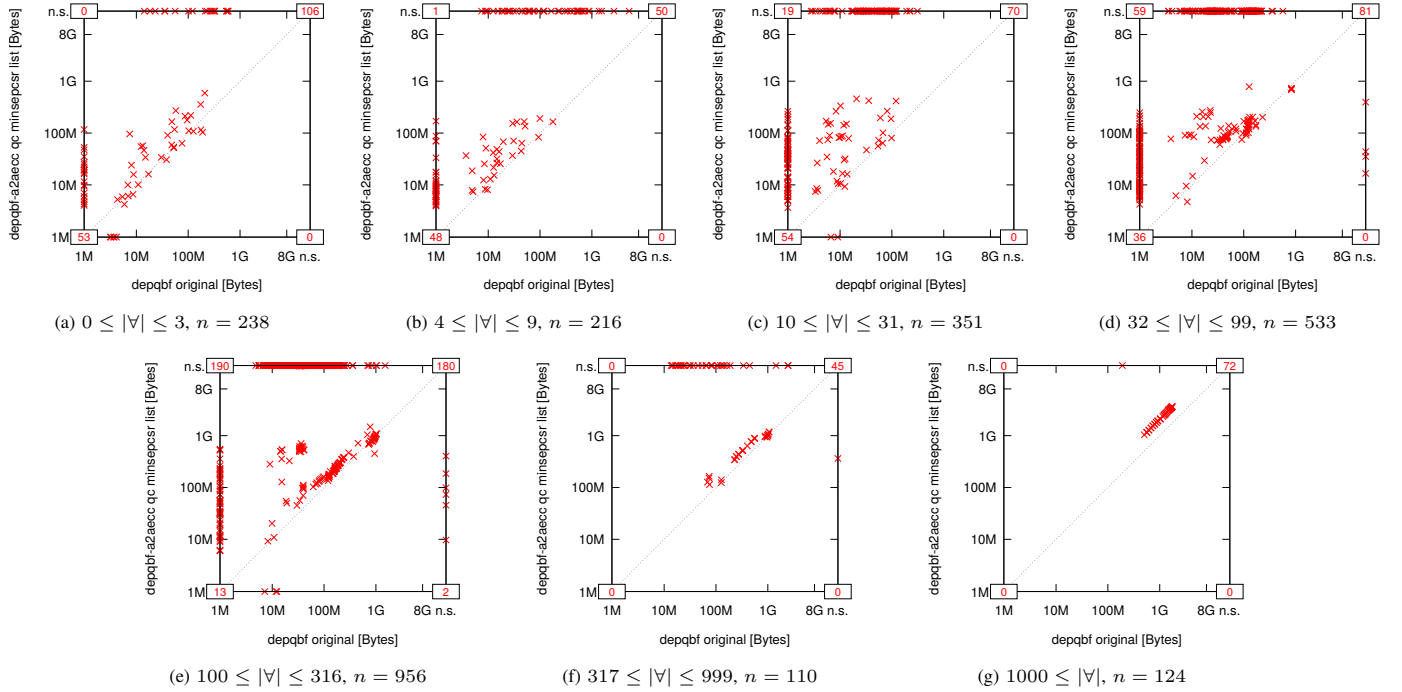


Fig. 1206: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode original partitioned by number of \forall quantified variables (memory usage in Bytes).

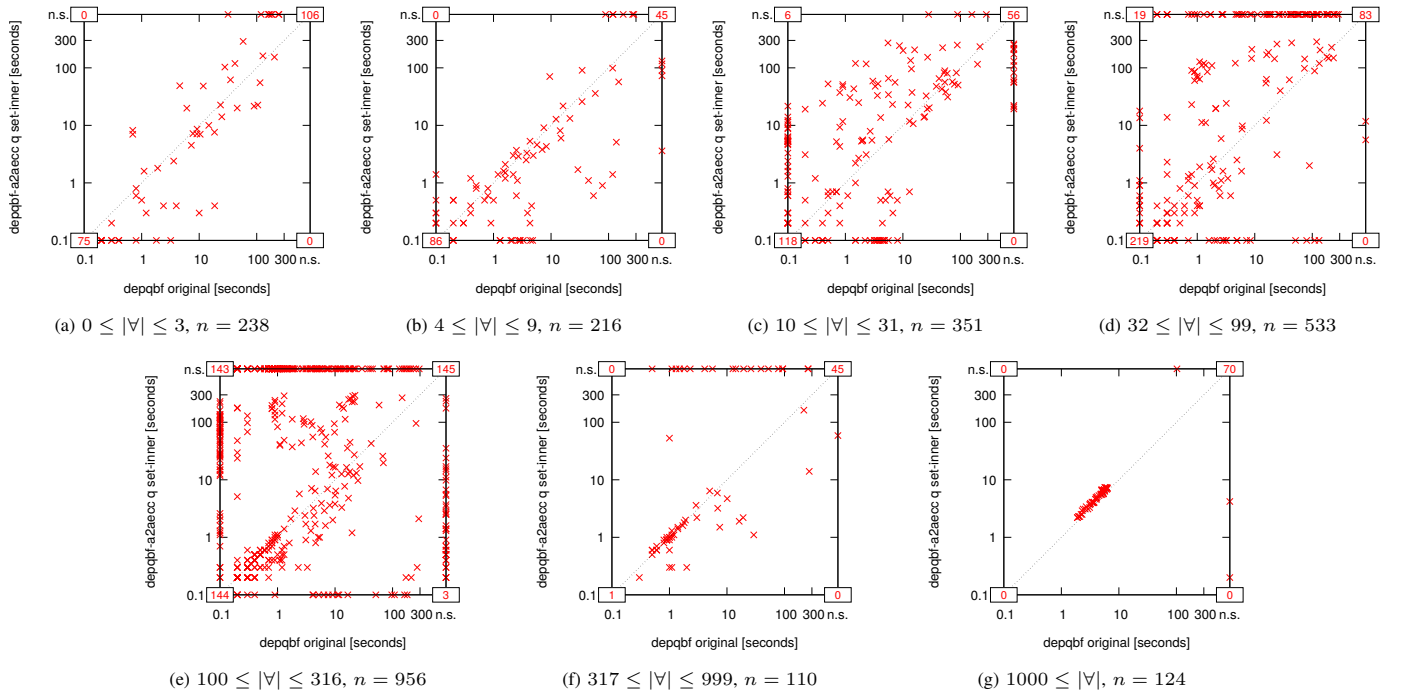


Fig. 1207: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode original partitioned by number of \forall quantified variables (run time in seconds).

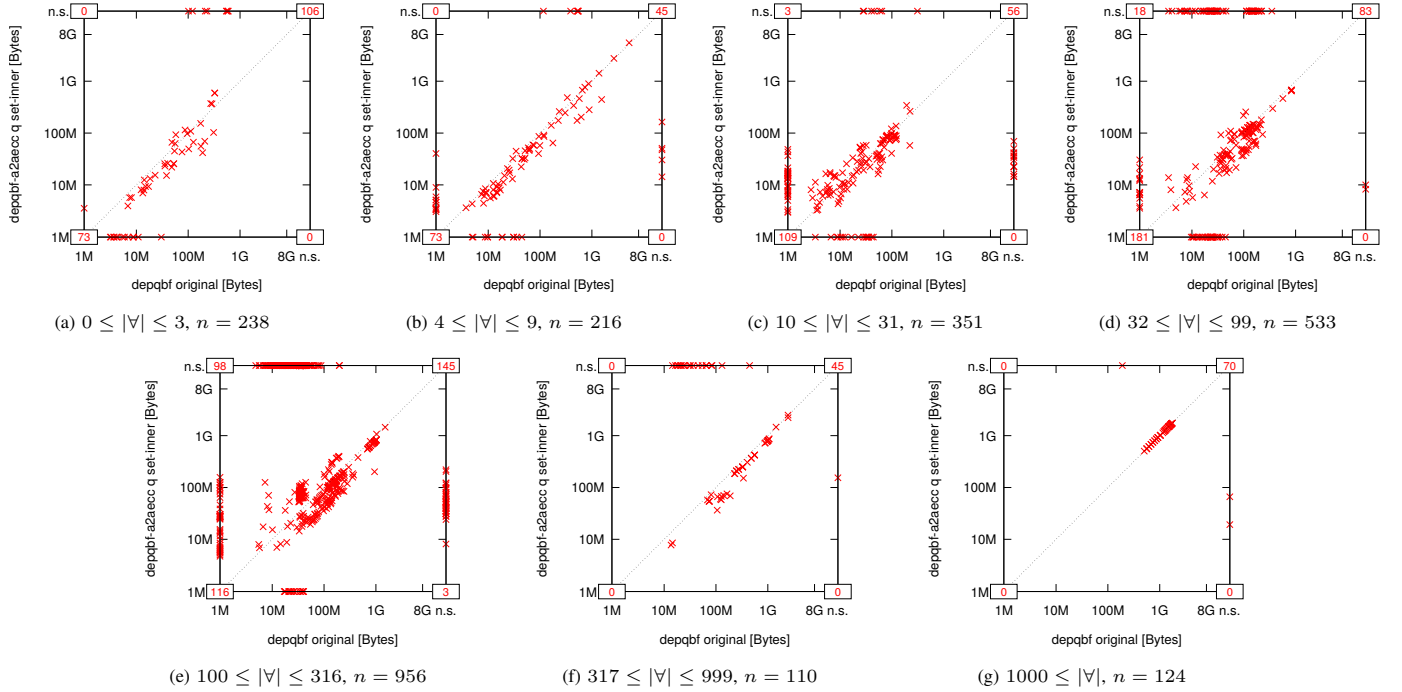


Fig. 1208: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode original partitioned by number of \forall quantified variables (memory usage in Bytes).

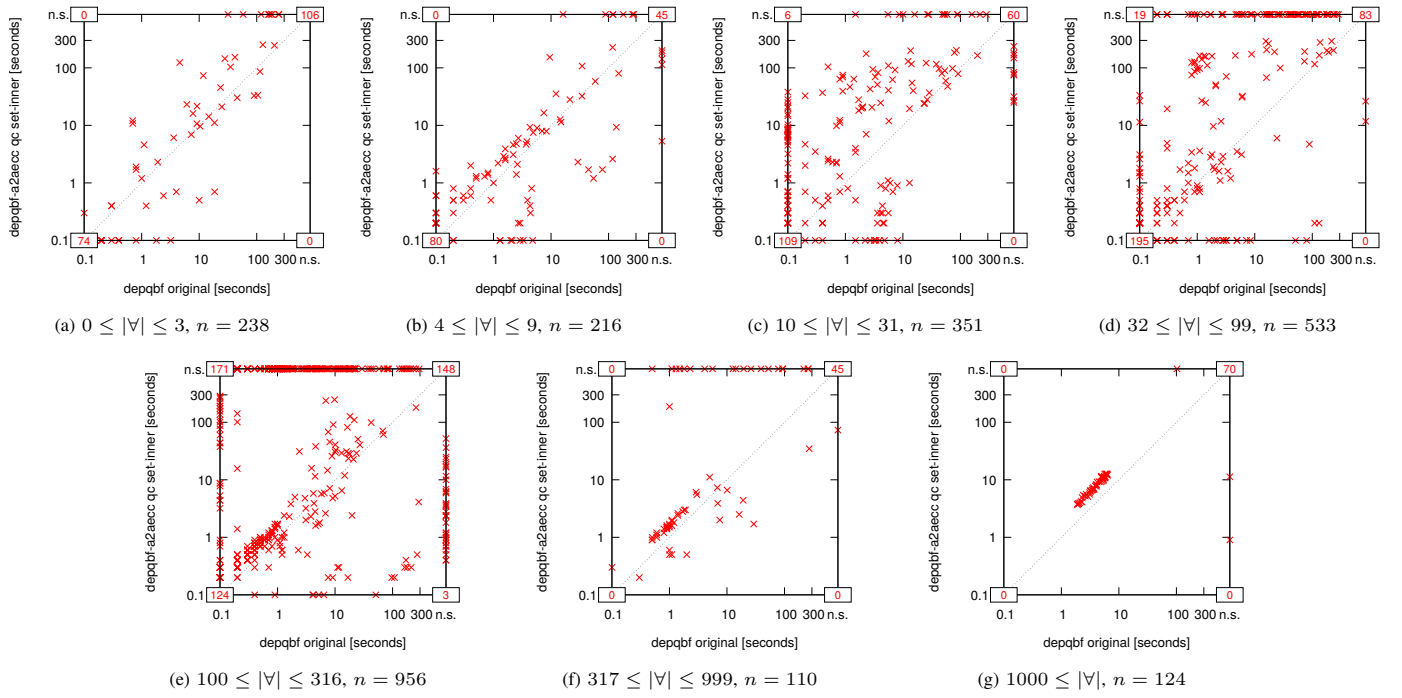


Fig. 1209: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode original partitioned by number of \forall quantified variables (run time in seconds).

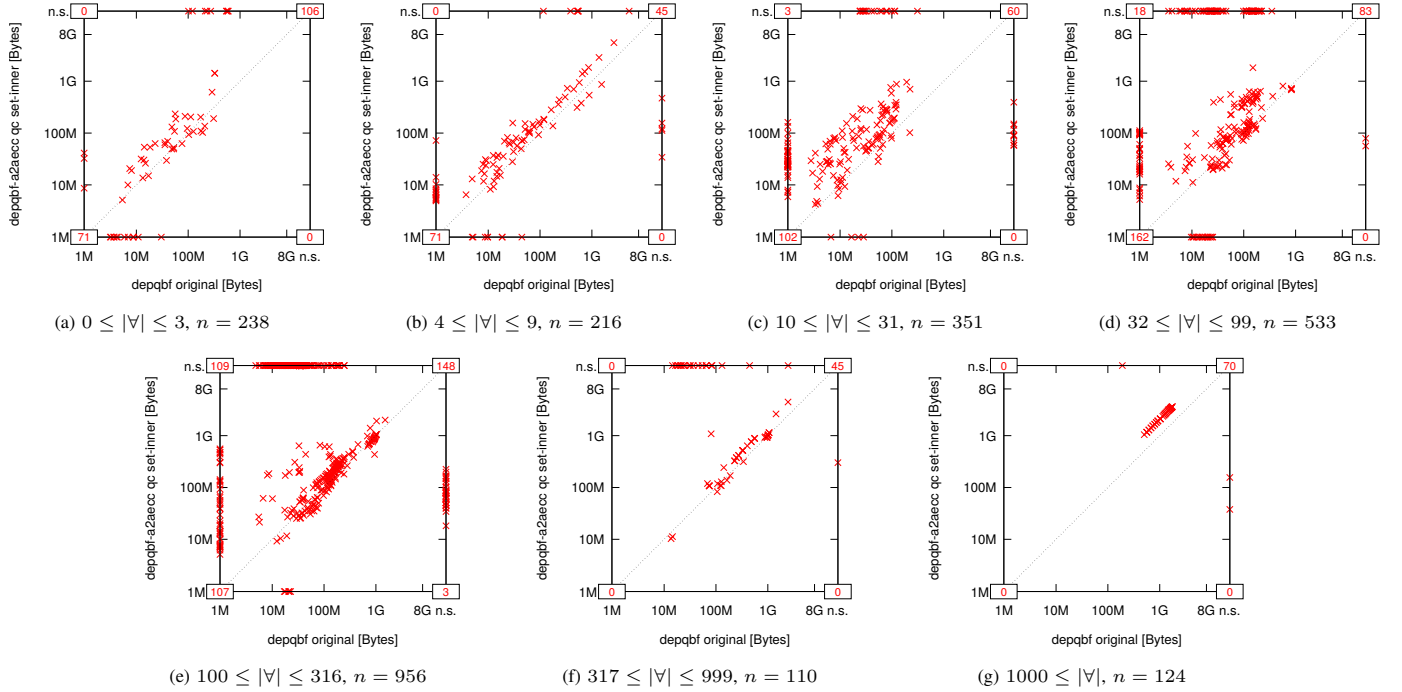


Fig. 1210: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode original partitioned by number of \forall quantified variables (memory usage in Bytes).

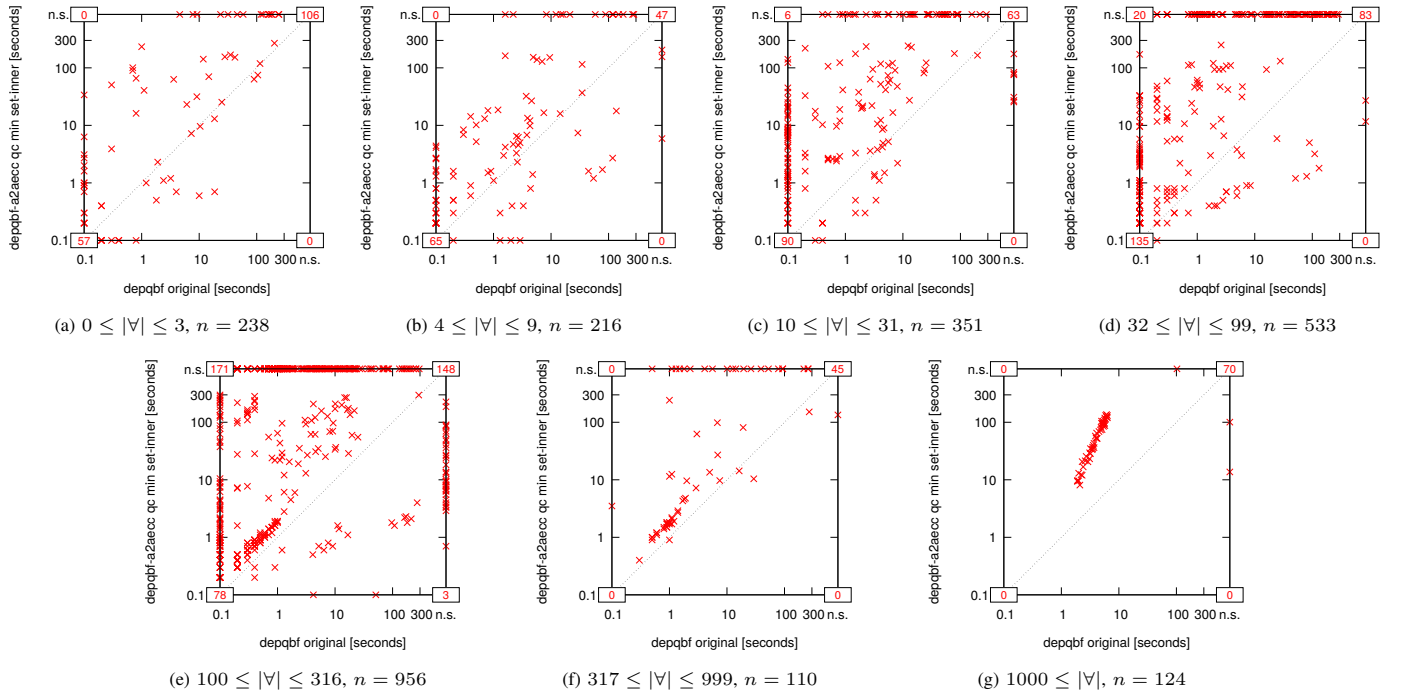


Fig. 1211: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode original partitioned by number of \forall quantified variables (run time in seconds).

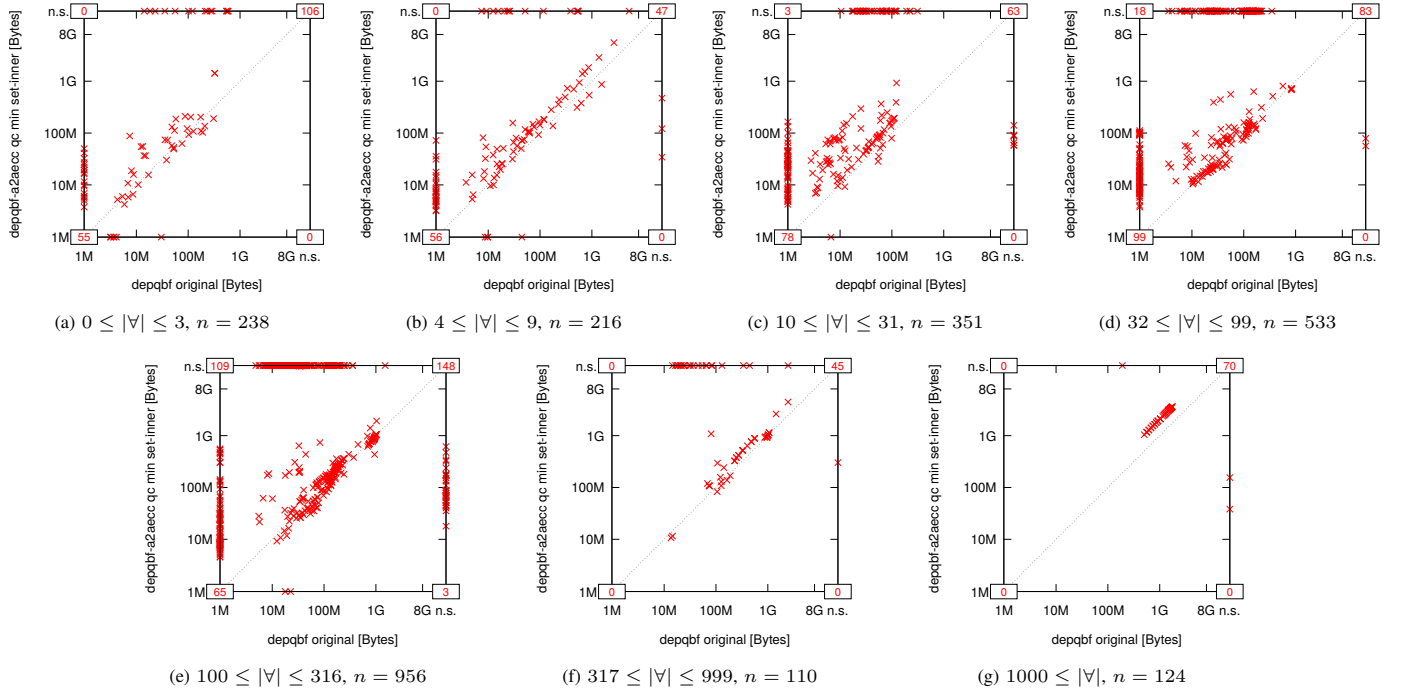


Fig. 1212: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode original partitioned by number of \forall quantified variables (memory usage in Bytes).

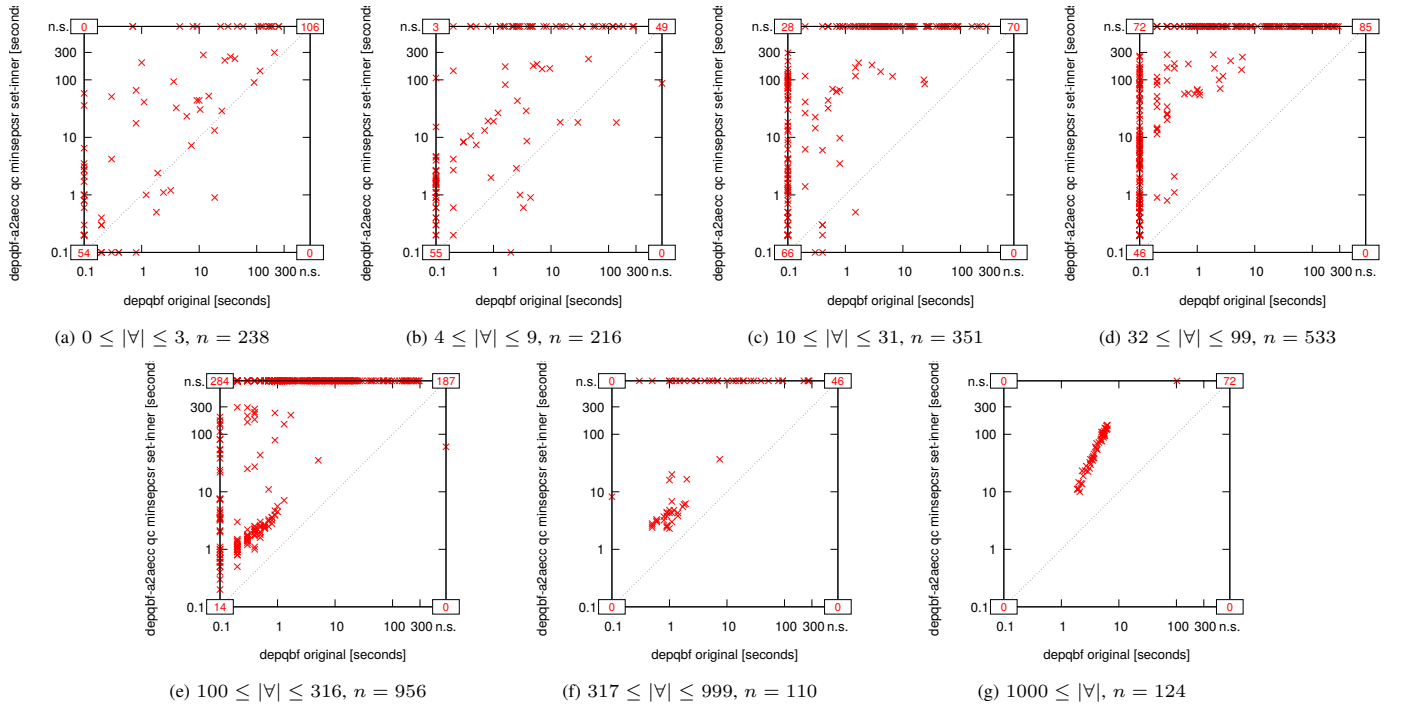


Fig. 1213: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode original partitioned by number of \forall quantified variables (run time in seconds).

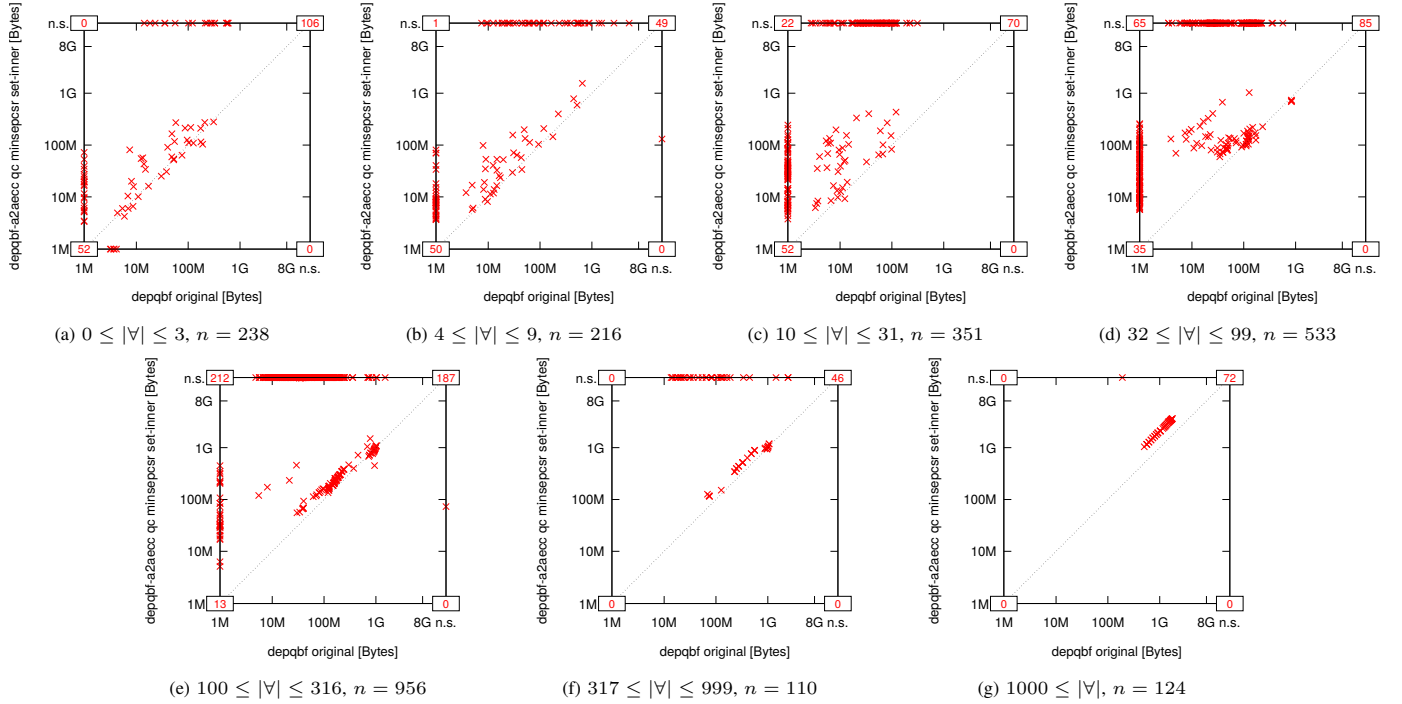


Fig. 1214: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode original partitioned by number of \forall quantified variables (memory usage in Bytes).

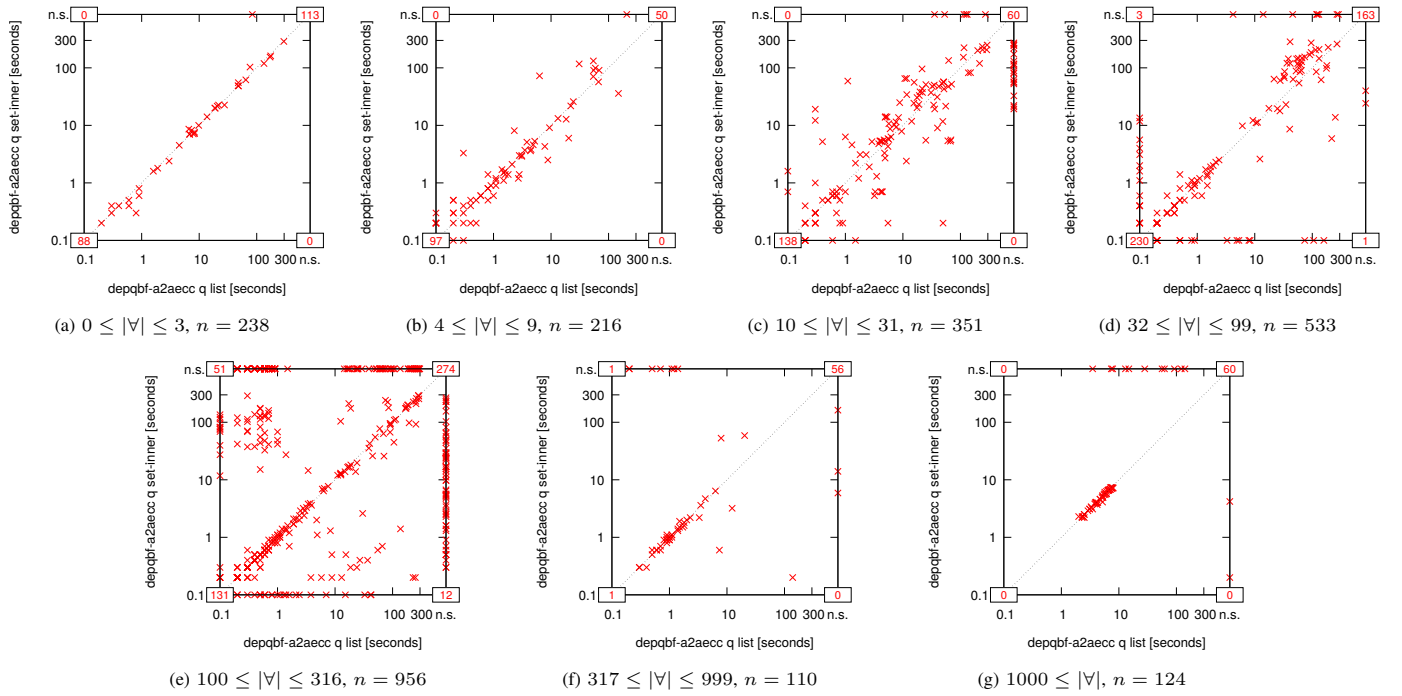


Fig. 1215: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q with set-inner versus list semantics partitioned by number of \forall quantified variables (run time in seconds).

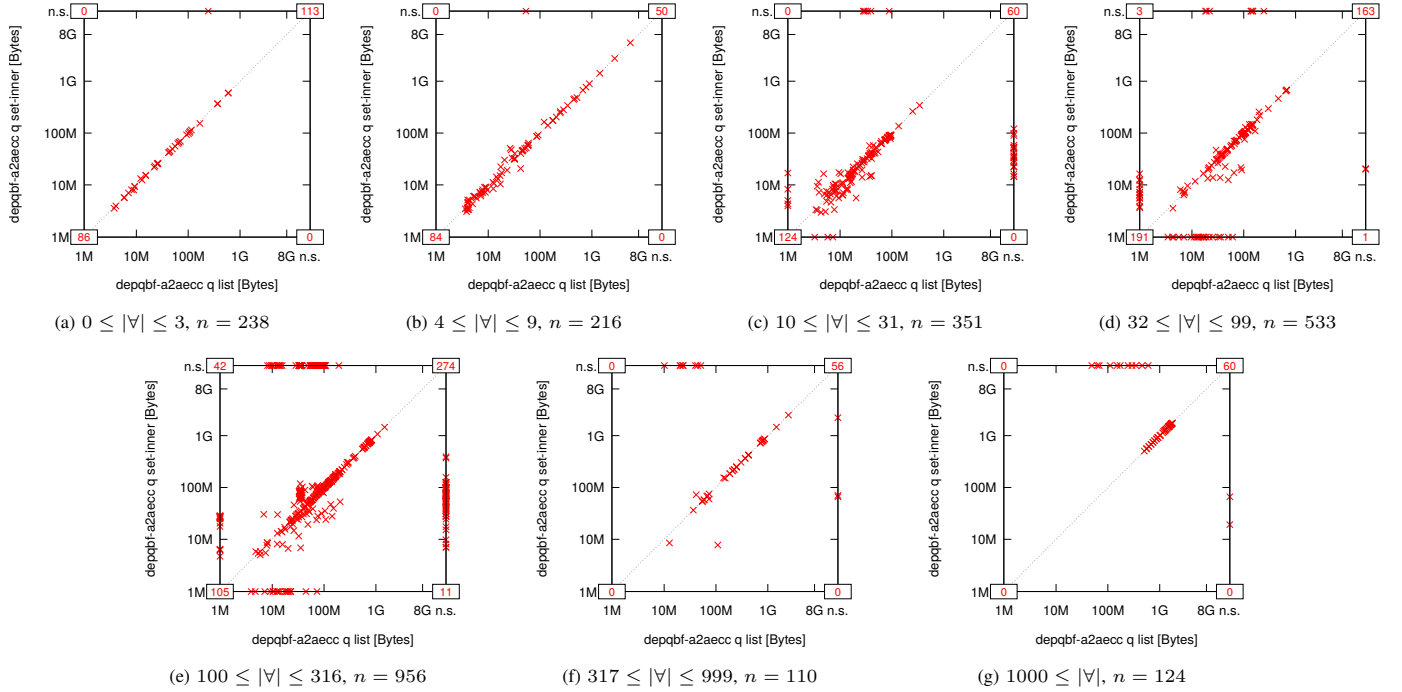


Fig. 1216: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q with set-inner versus list semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

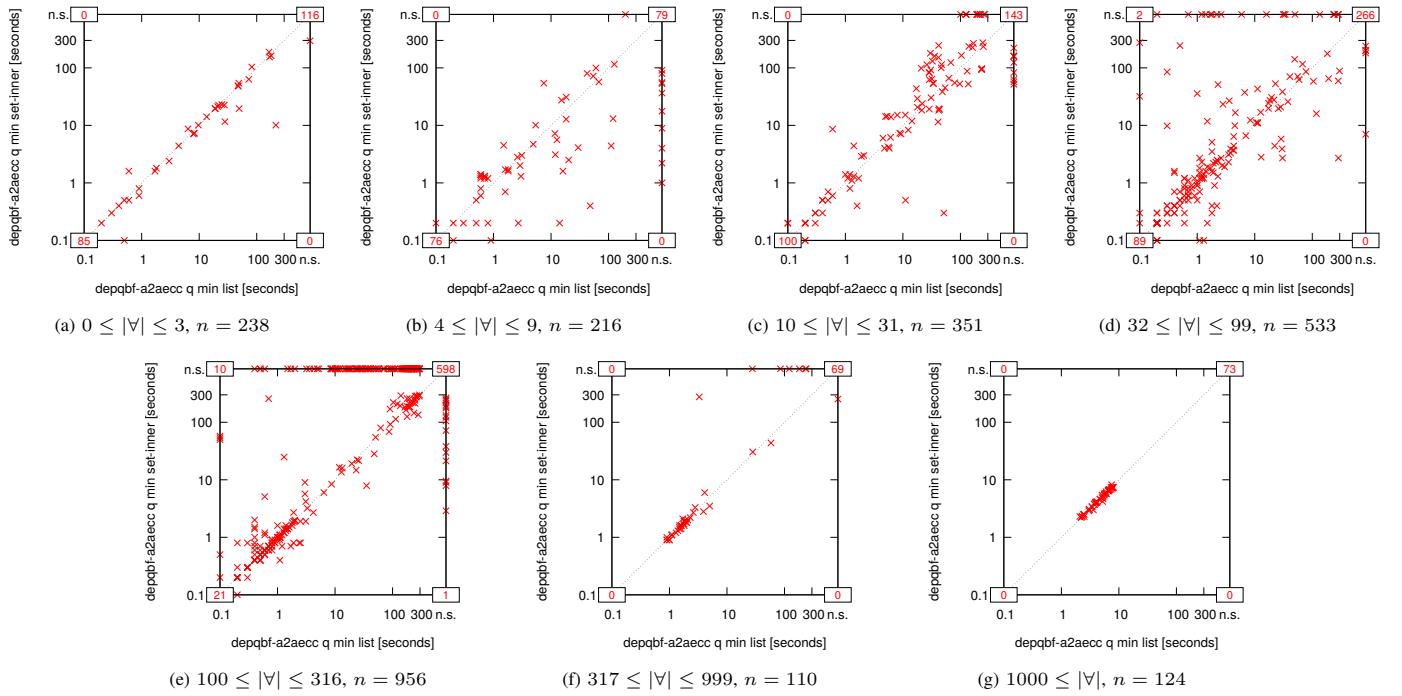


Fig. 1217: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q min with set-inner versus list semantics partitioned by number of \forall quantified variables (run time in seconds).

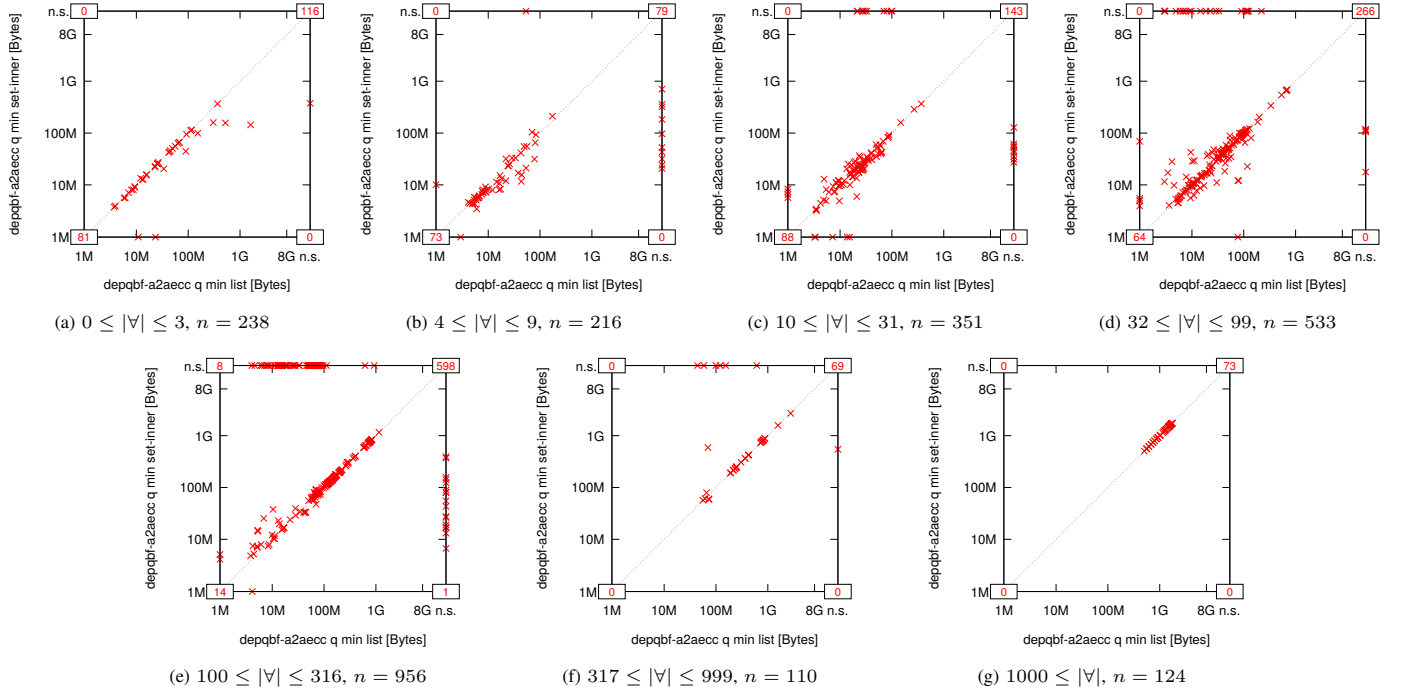


Fig. 1218: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q min with set-inner versus list semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

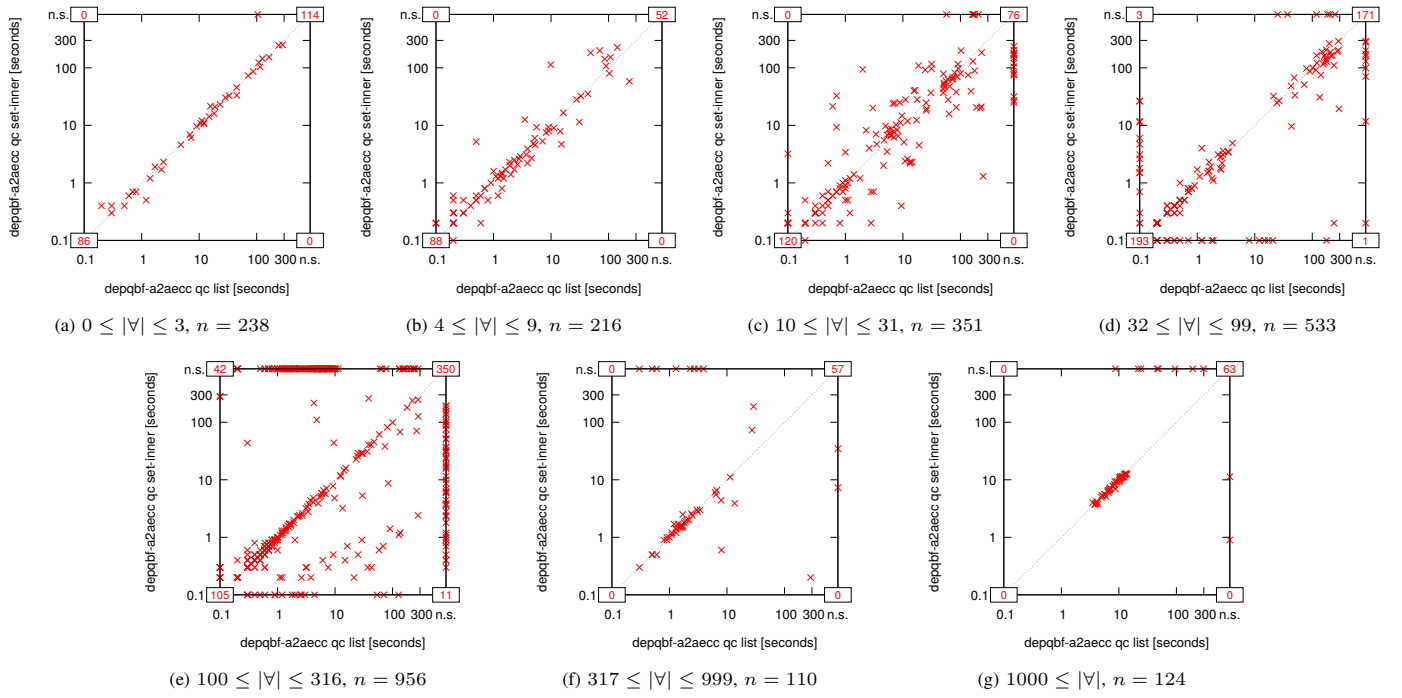


Fig. 1219: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc with set-inner versus list semantics partitioned by number of \forall quantified variables (run time in seconds).

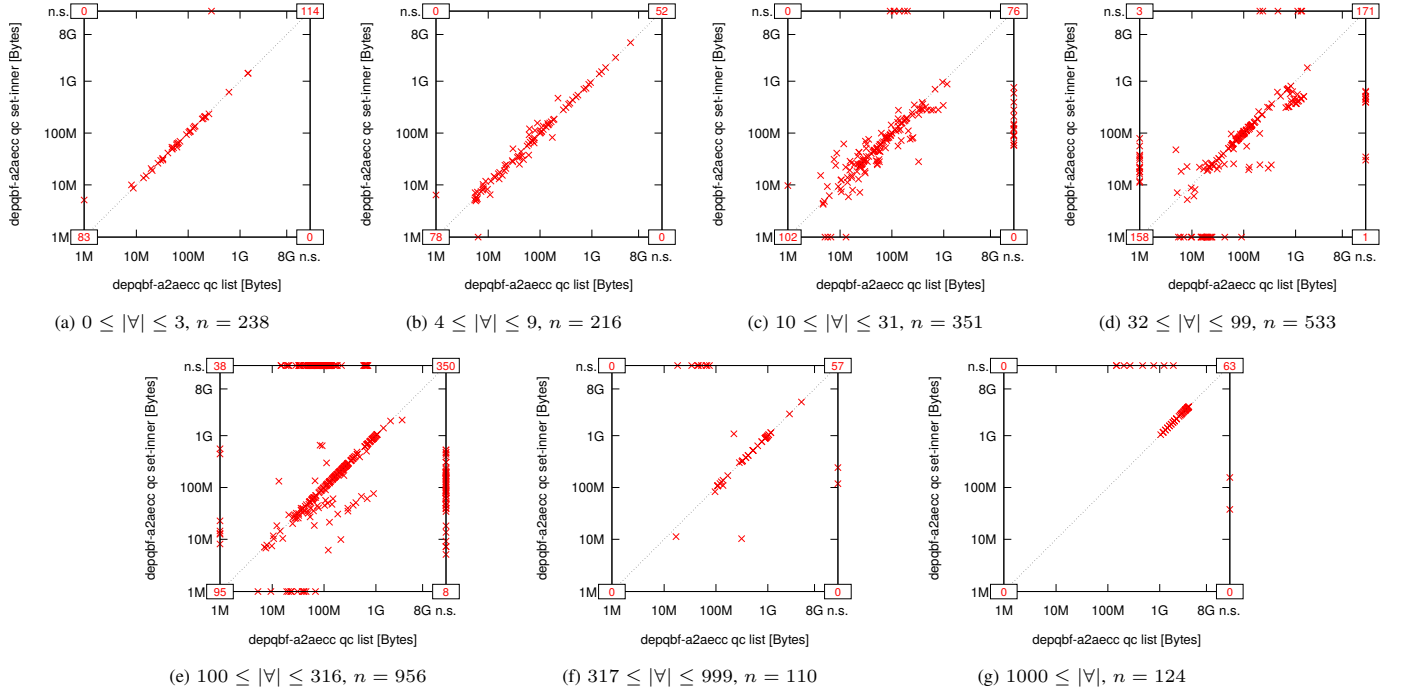


Fig. 1220: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc with set-inner versus list semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

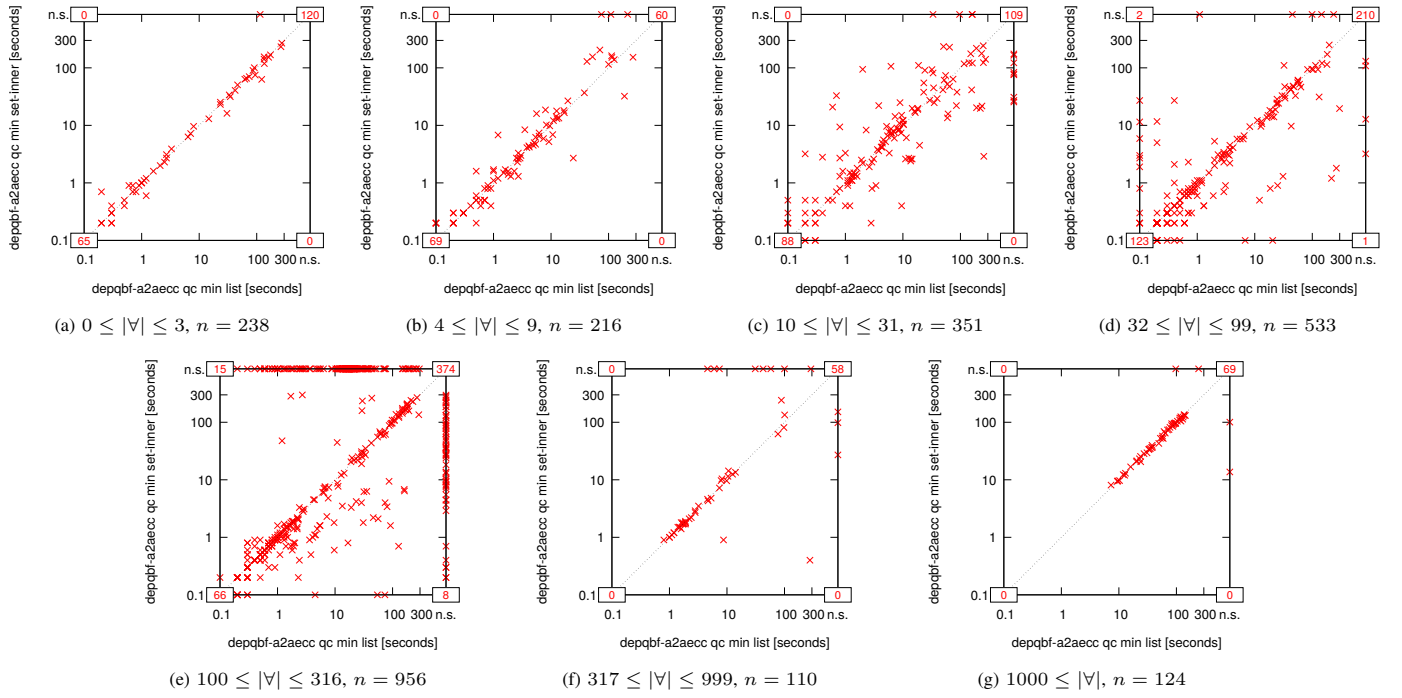


Fig. 1221: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min with set-inner versus list semantics partitioned by number of \forall quantified variables (run time in seconds).

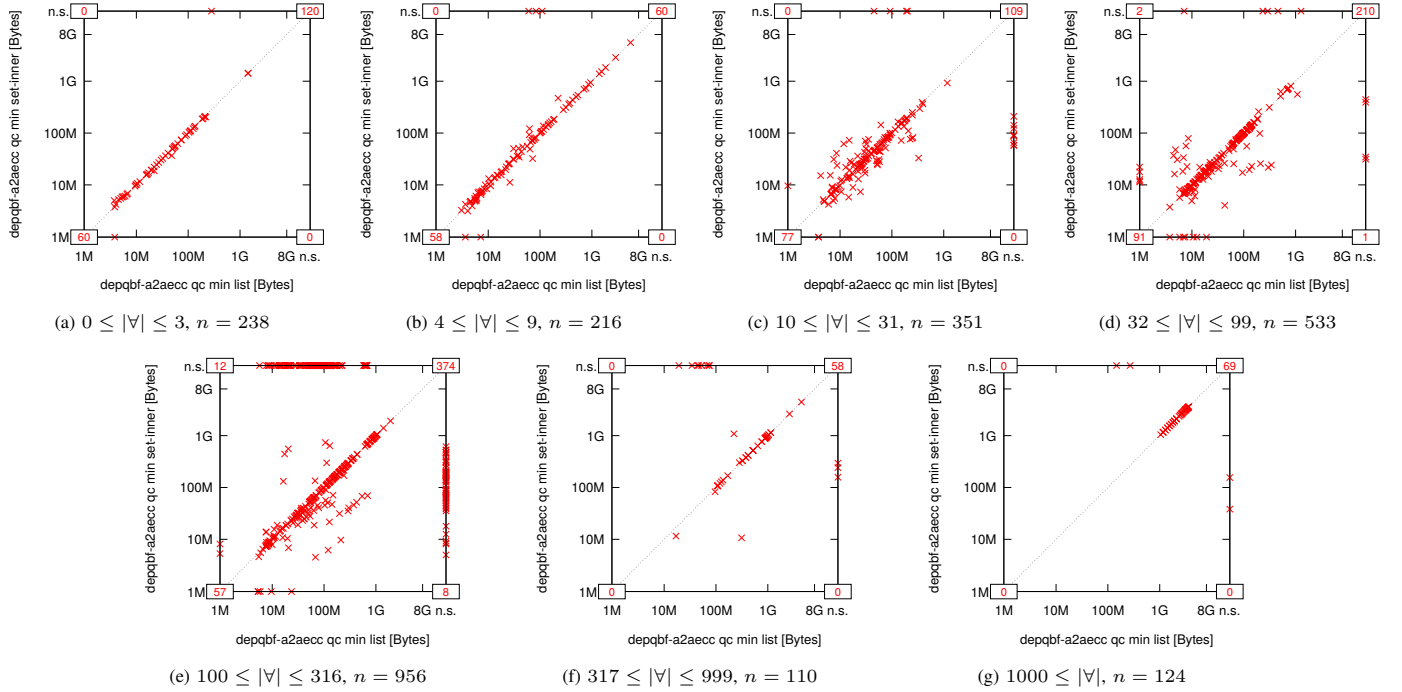


Fig. 1222: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min with set-inner versus list semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

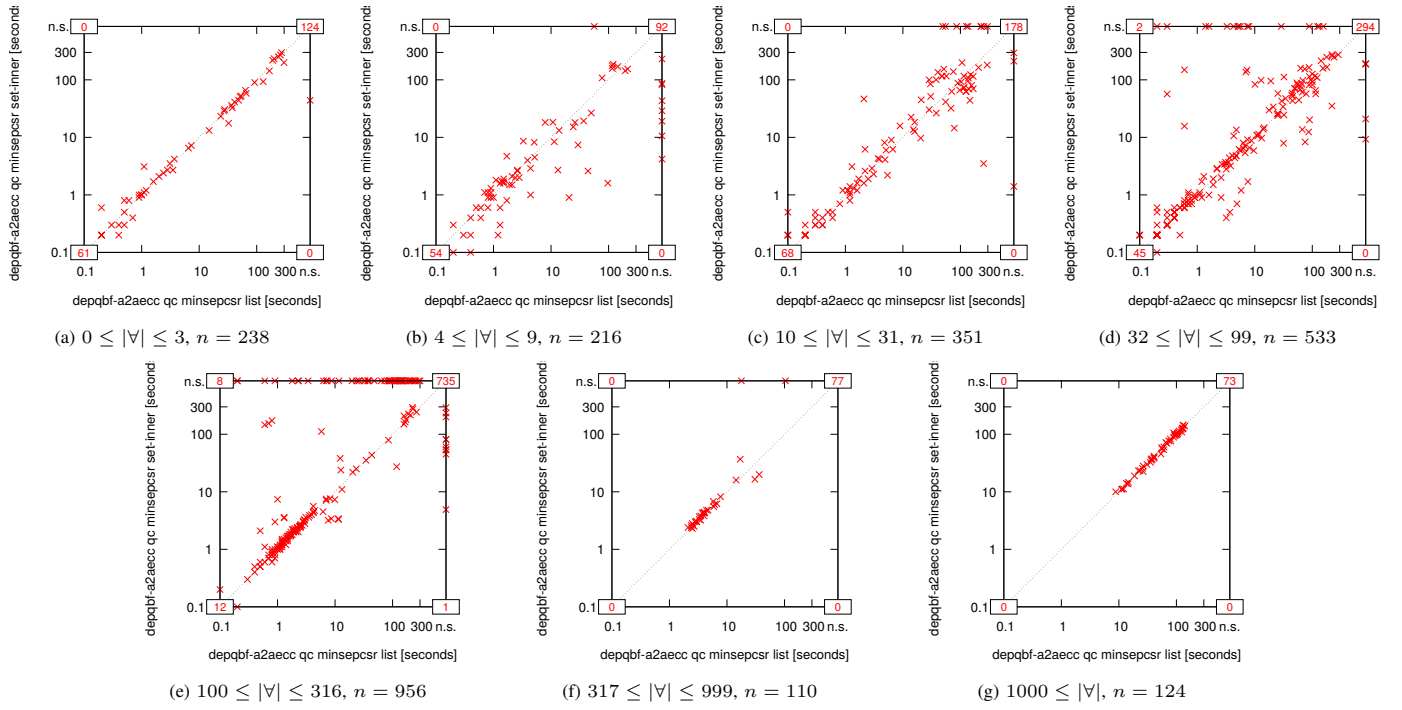


Fig. 1223: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr with set-inner versus list semantics partitioned by number of \forall quantified variables (run time in seconds).

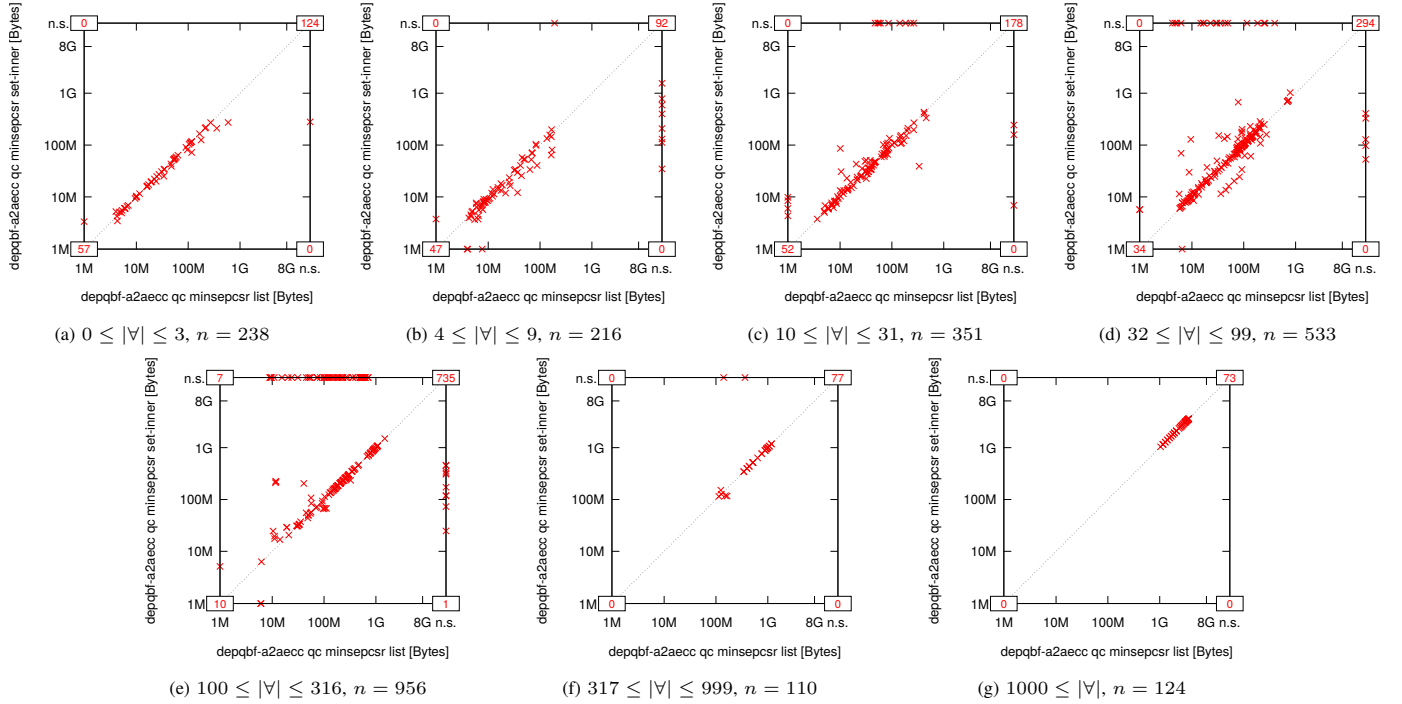


Fig. 1224: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr with set-inner versus list semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

C. Partitioned by Alternation Depth

This subsection shows figures of the overhead of extracting and minimizing unsatisfiable cores with subfigures for partitions of the benchmarks according to their alternation depths.

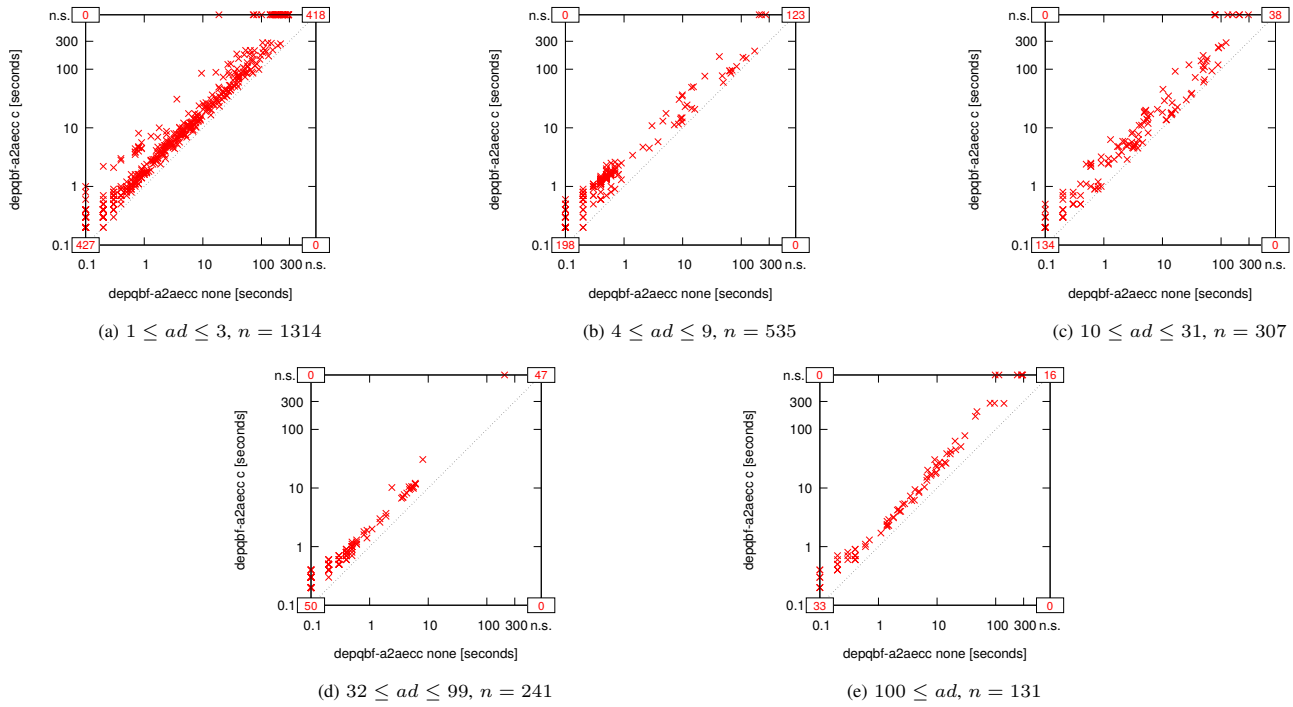


Fig. 1225: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c versus mode none partitioned by alternation depth (run time in seconds).

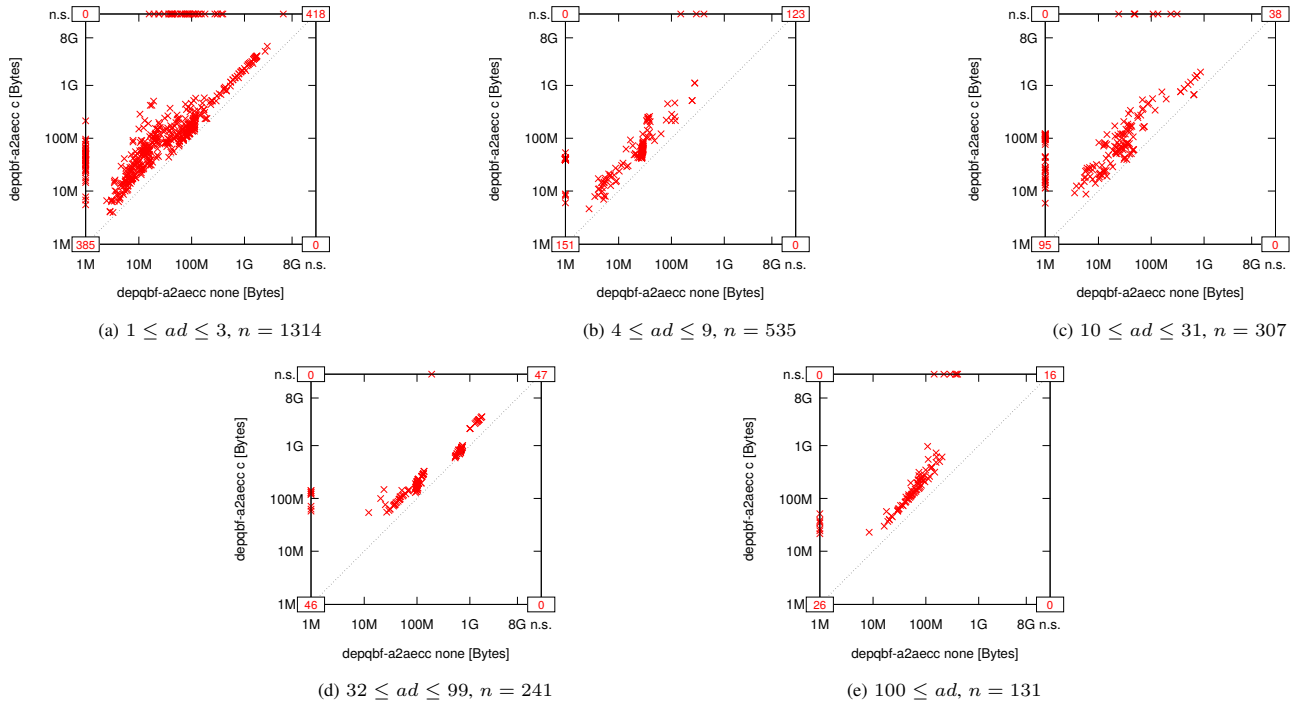


Fig. 1226: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c versus mode none partitioned by alternation depth (memory usage in Bytes).

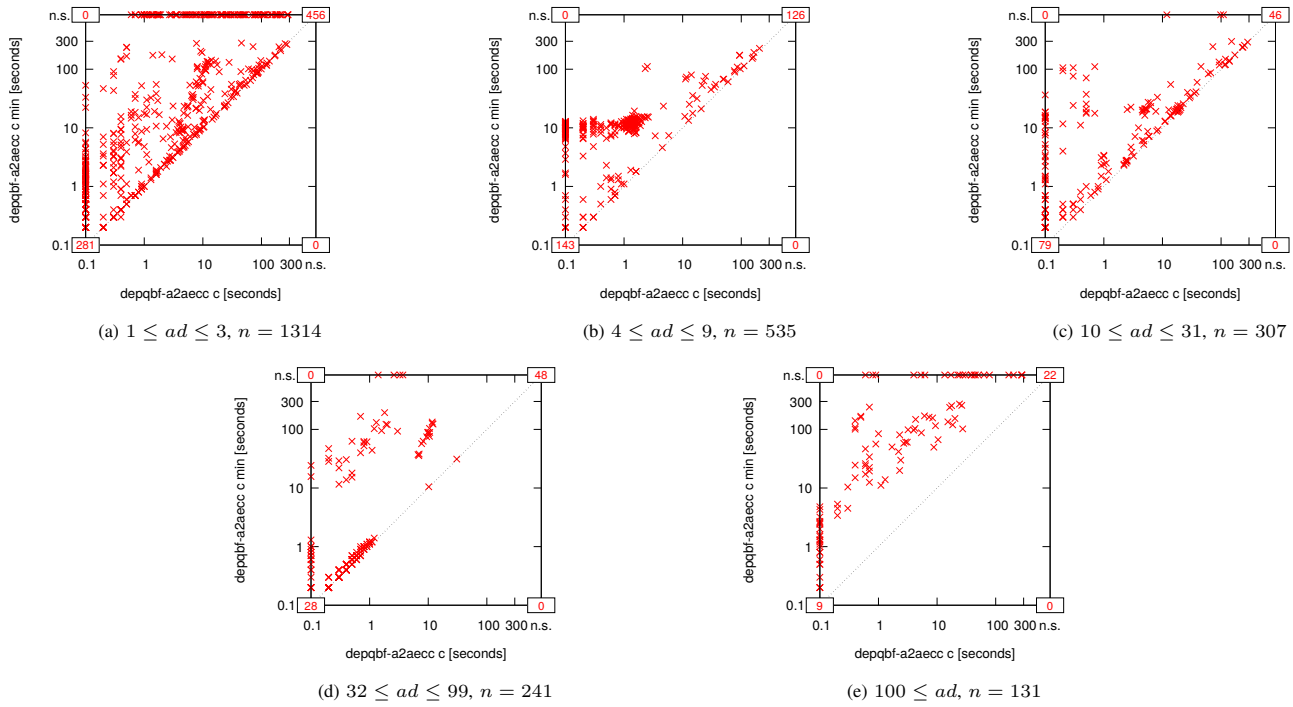


Fig. 1227: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode c partitioned by alternation depth (run time in seconds).

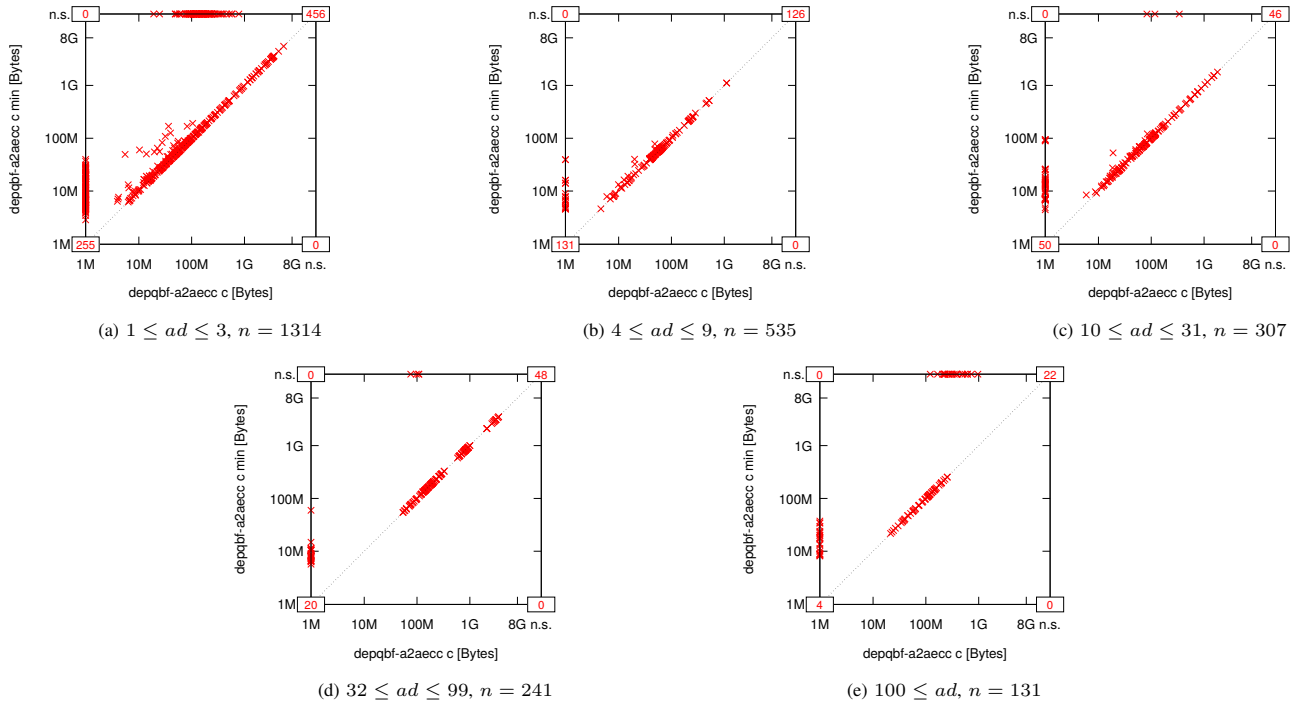


Fig. 1228: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode c partitioned by alternation depth (memory usage in Bytes).

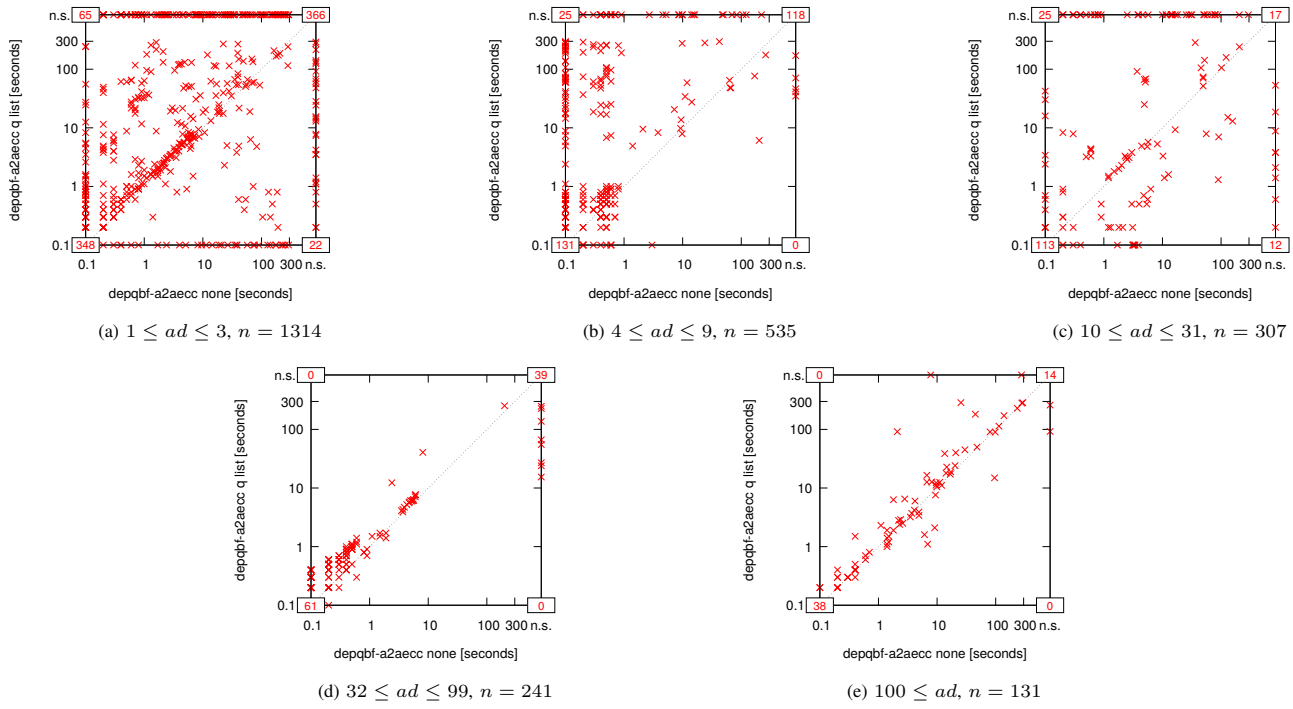


Fig. 1229: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode none partitioned by alternation depth (run time in seconds).

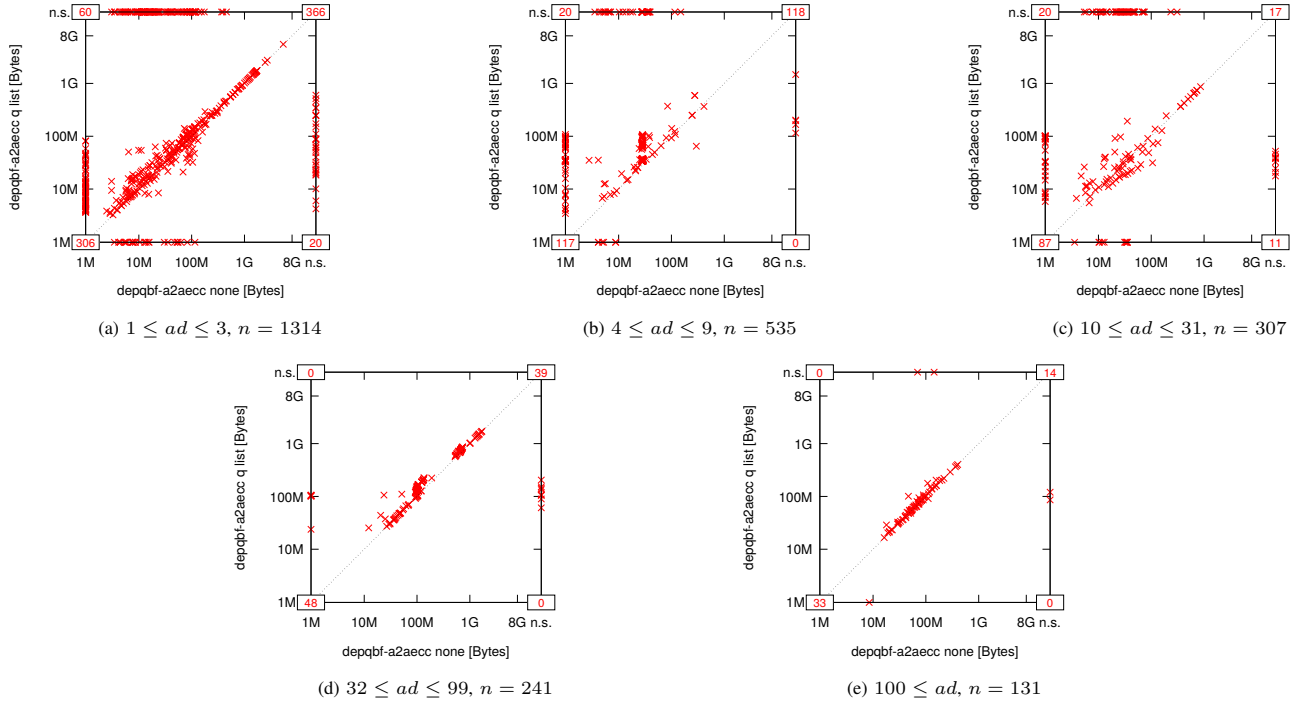


Fig. 1230: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode none partitioned by alternation depth (memory usage in Bytes).

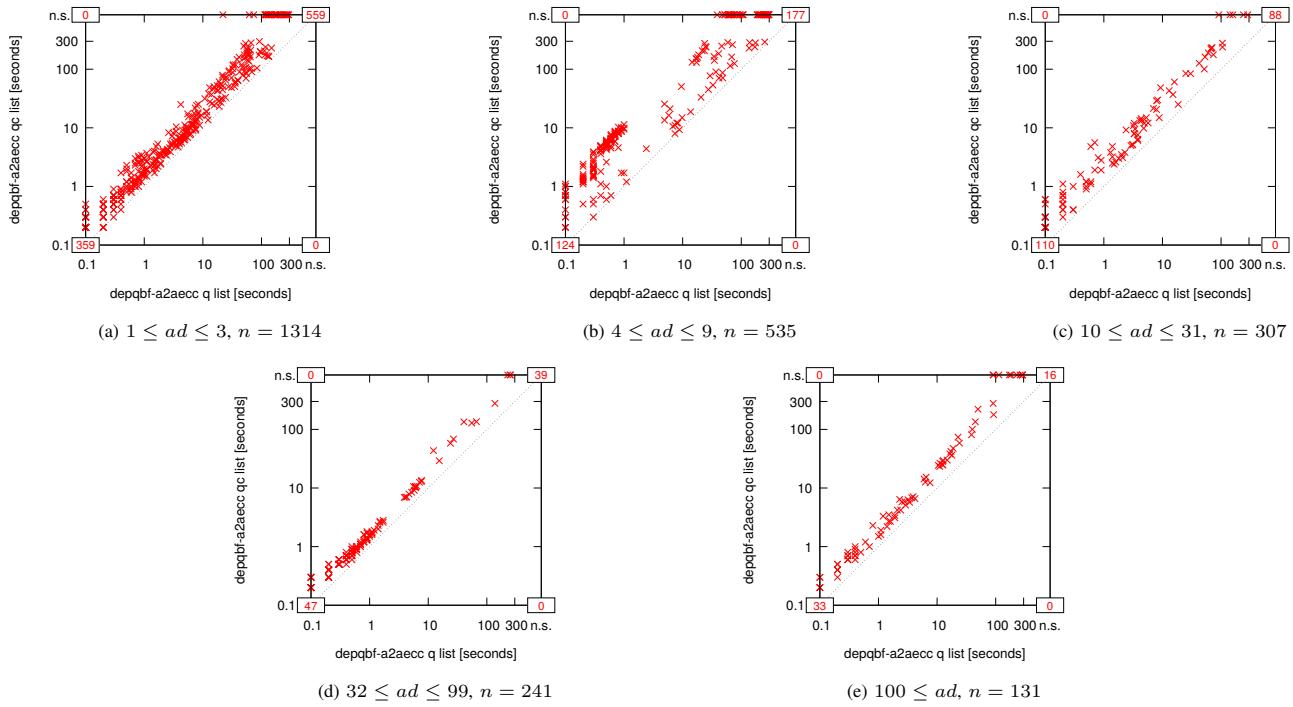


Fig. 1231: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode q list partitioned by alternation depth (run time in seconds).

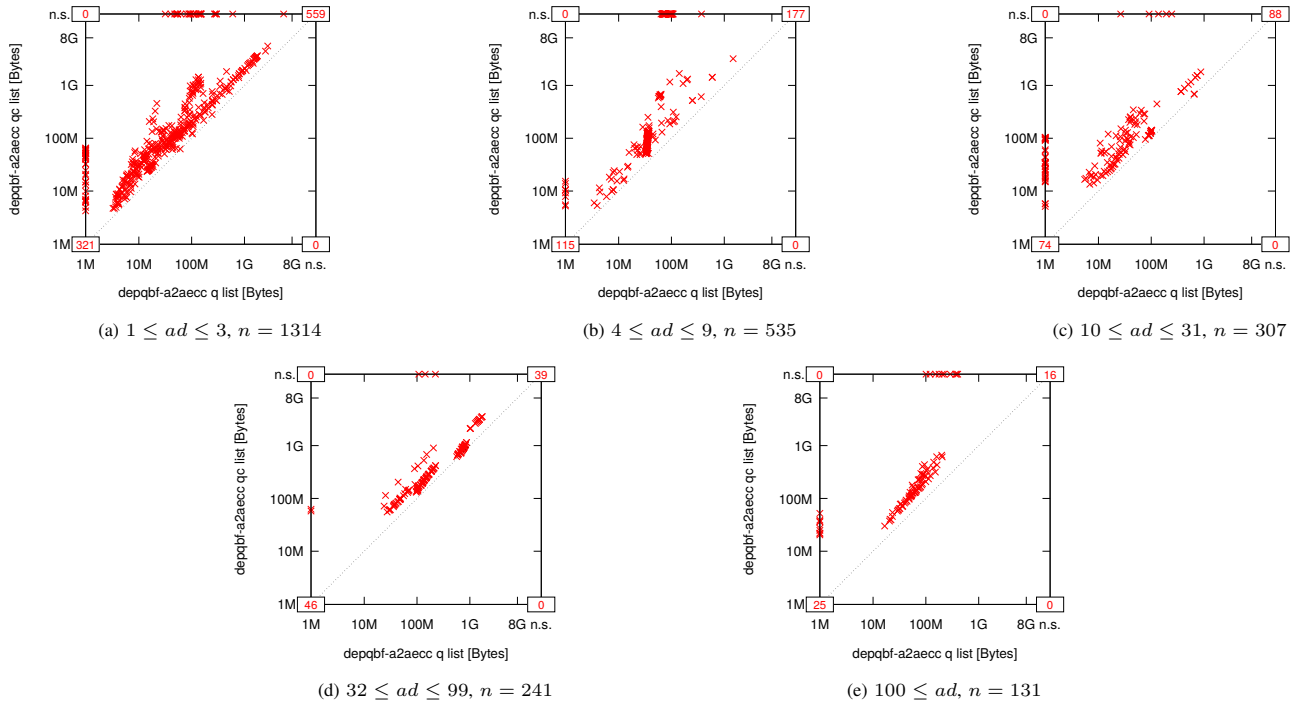


Fig. 1232: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode q list partitioned by alternation depth (memory usage in Bytes).

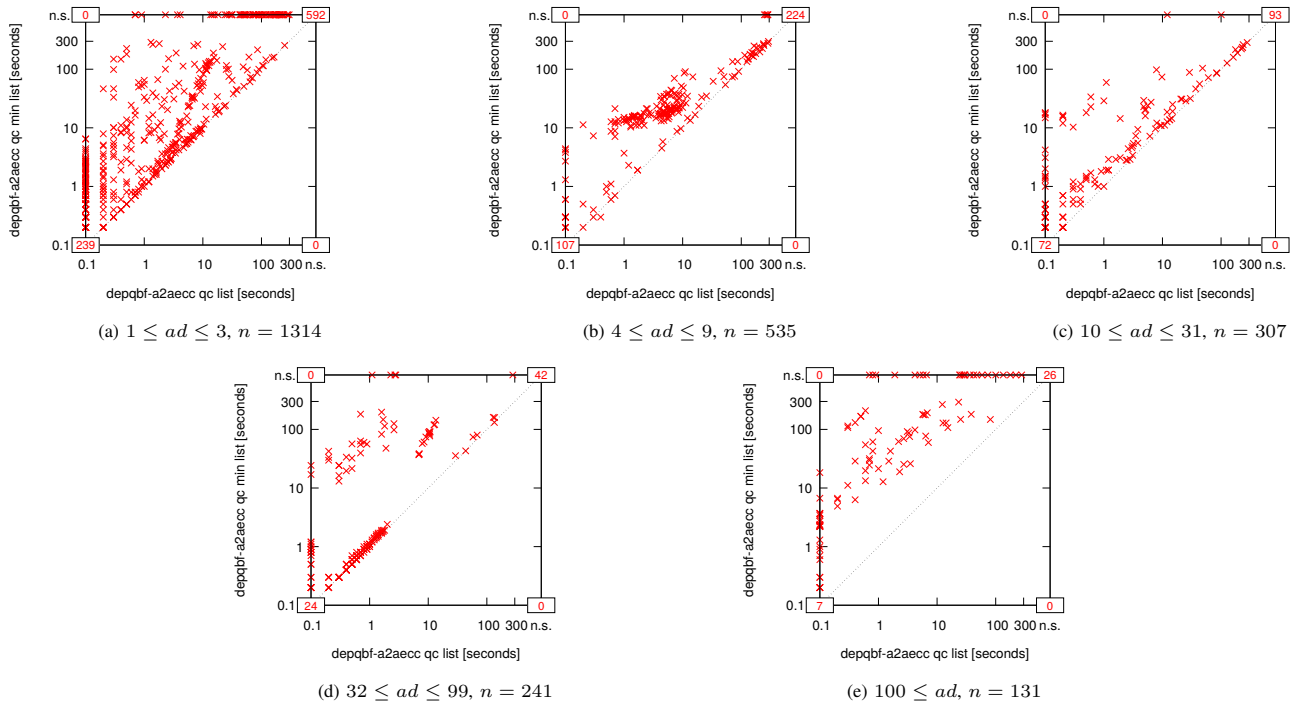


Fig. 1233: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode qc list partitioned by alternation depth (run time in seconds).

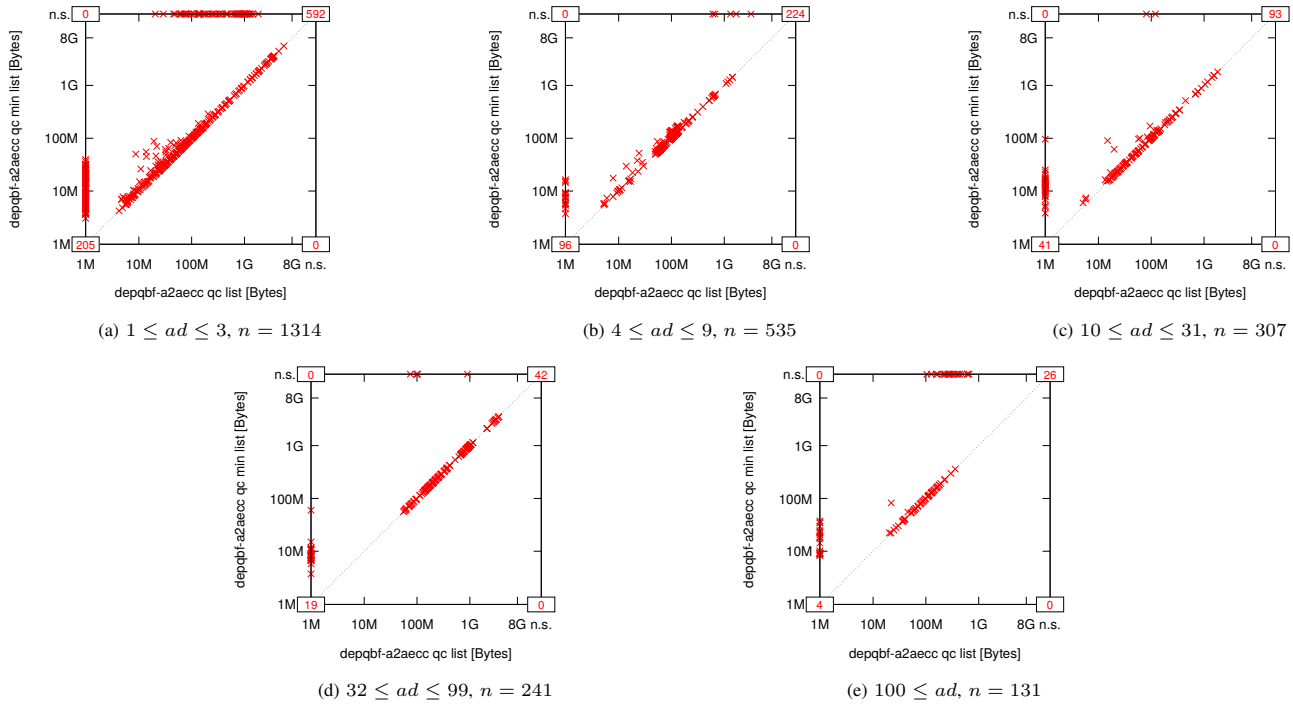


Fig. 1234: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode qc list partitioned by alternation depth (memory usage in Bytes).

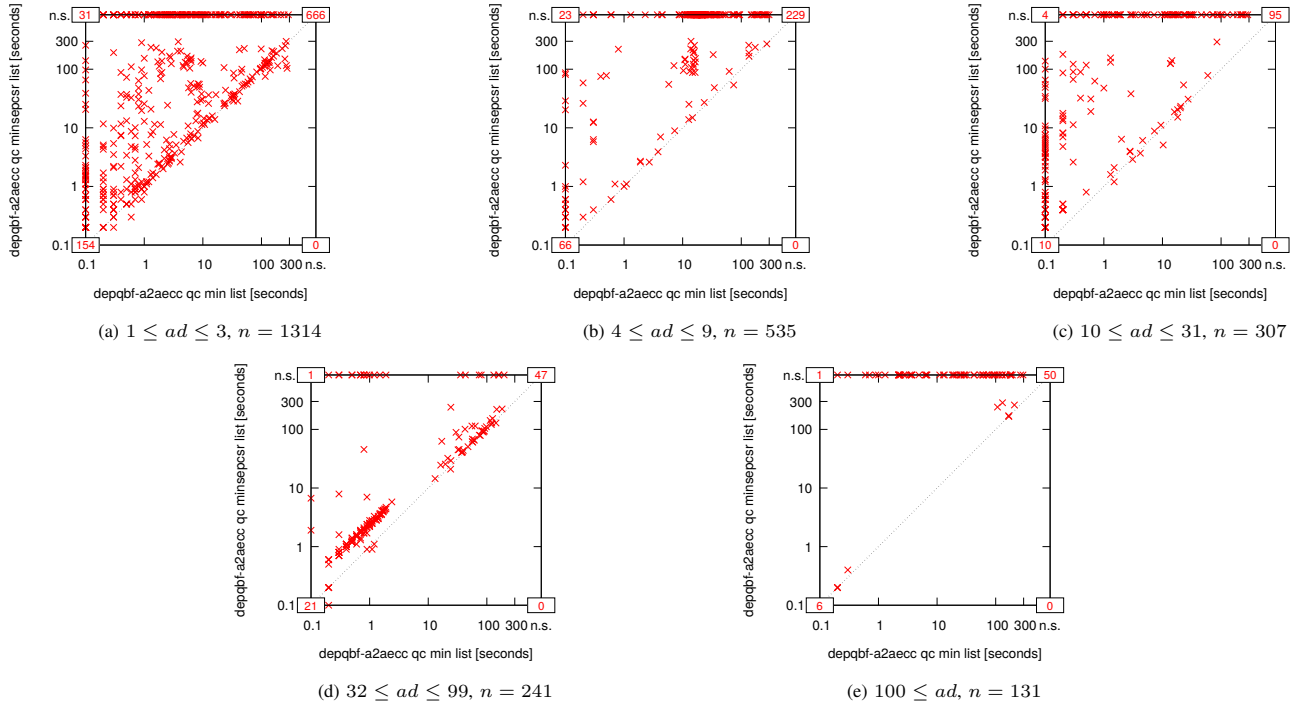


Fig. 1235: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode qc min list partitioned by alternation depth (run time in seconds).

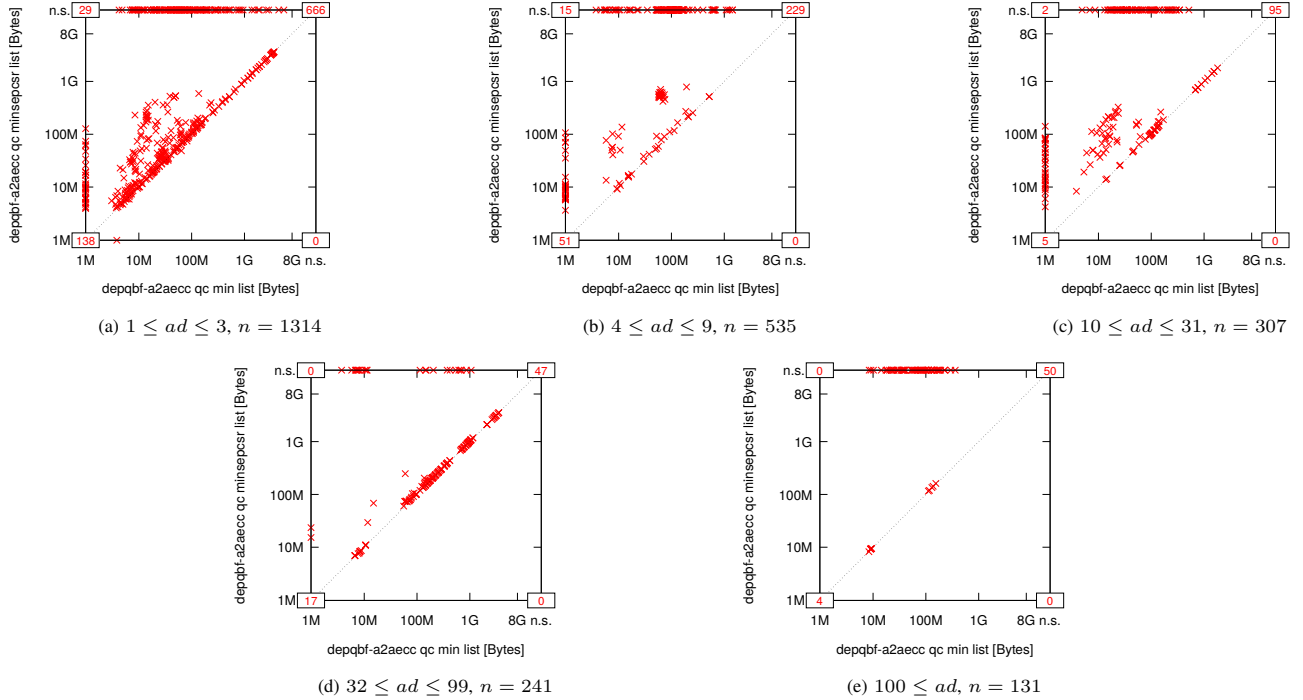


Fig. 1236: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode qc min list partitioned by alternation depth (memory usage in Bytes).

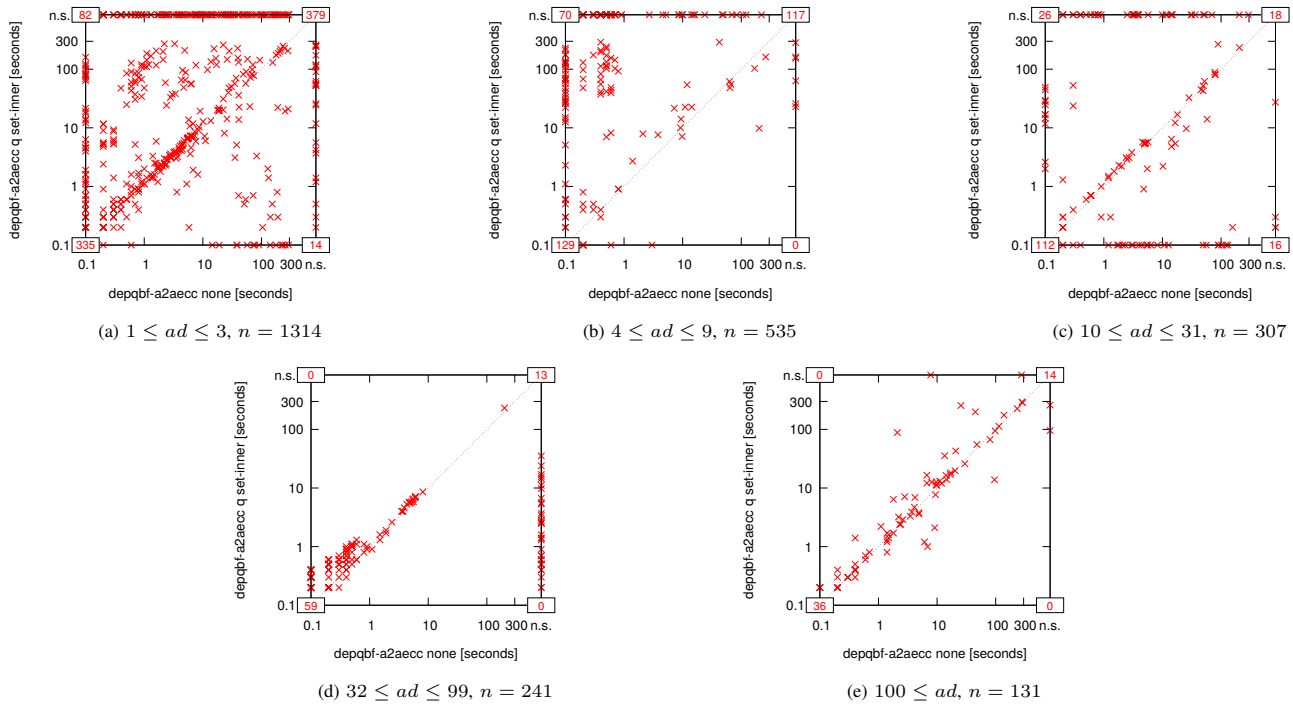


Fig. 1237: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode none partitioned by alternation depth (run time in seconds).

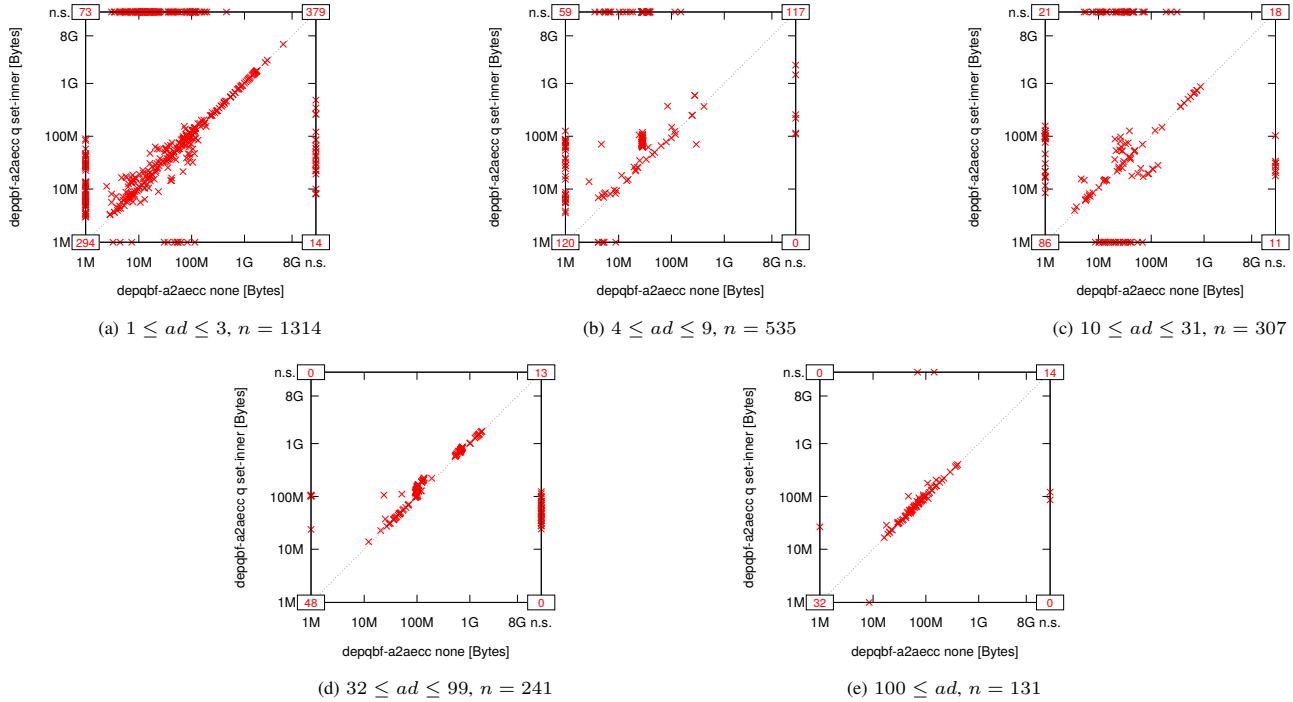


Fig. 1238: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode none partitioned by alternation depth (memory usage in Bytes).

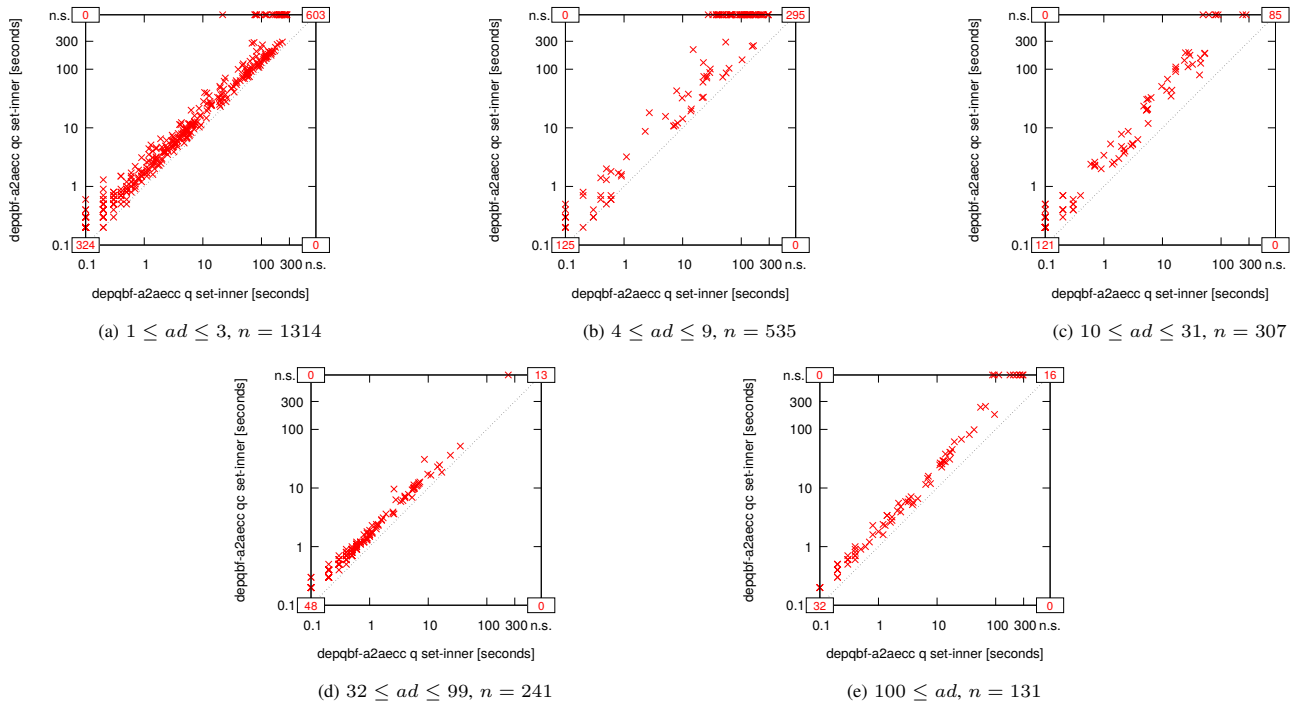


Fig. 1239: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode q set-inner partitioned by alternation depth (run time in seconds).

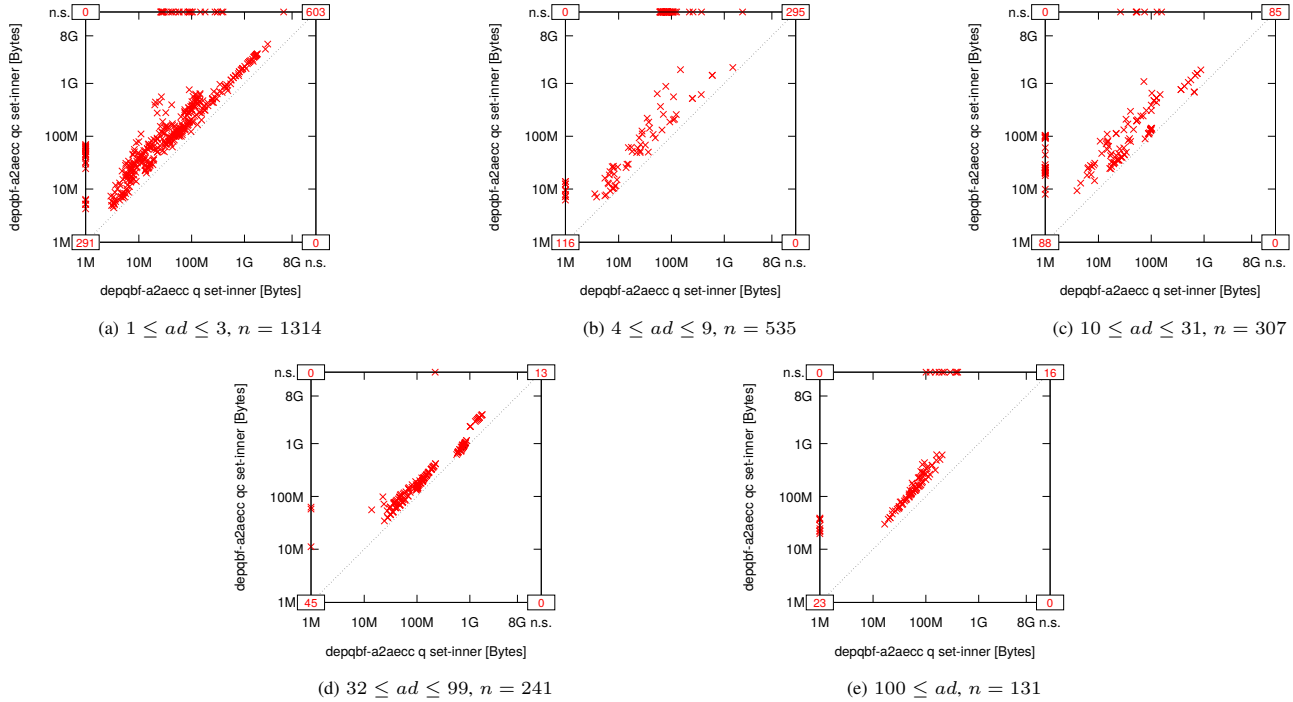


Fig. 1240: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode q set-inner partitioned by alternation depth (memory usage in Bytes).

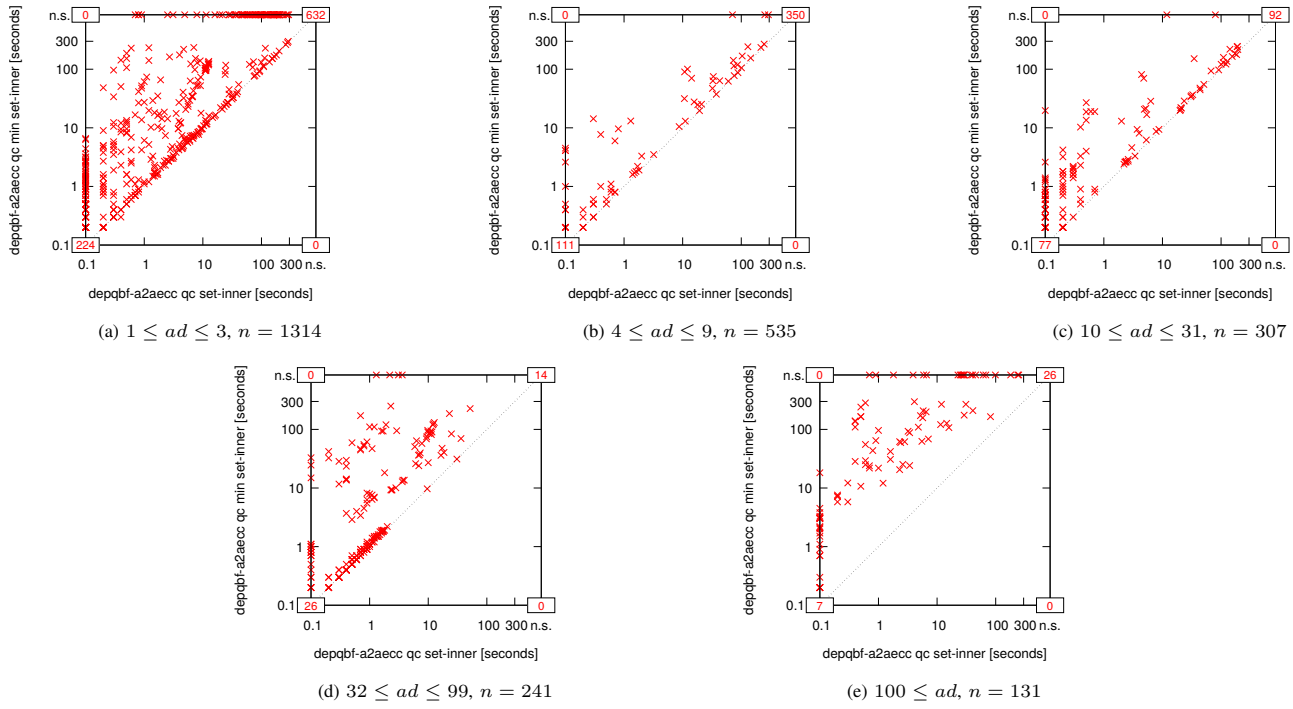


Fig. 1241: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode qc set-inner partitioned by alternation depth (run time in seconds).

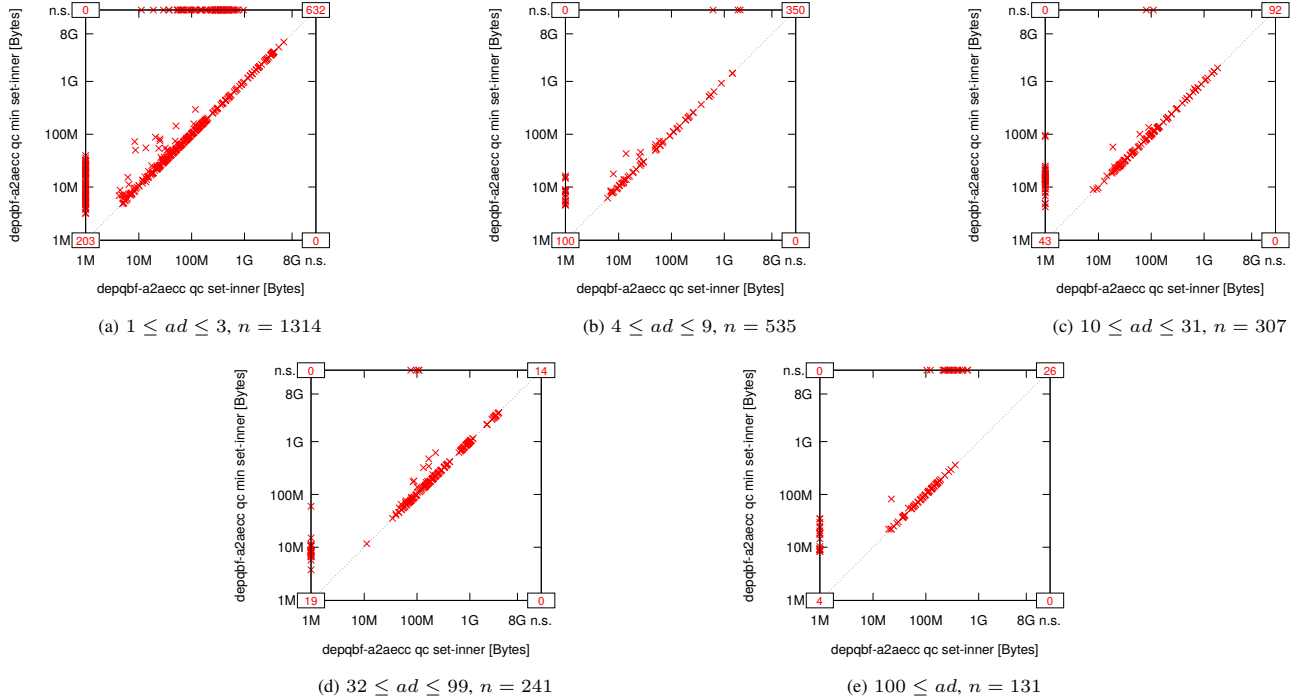


Fig. 1242: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode qc set-inner partitioned by alternation depth (memory usage in Bytes).

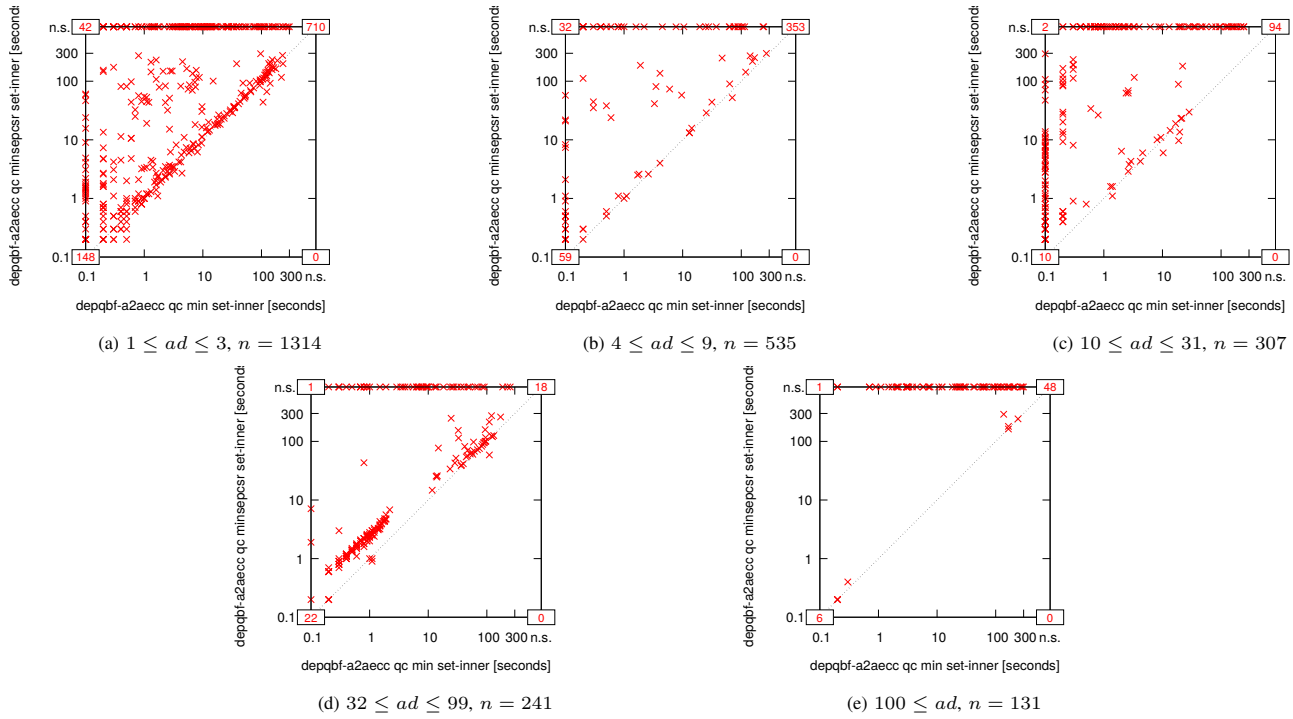


Fig. 1243: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode qc min set-inner partitioned by alternation depth (run time in seconds).

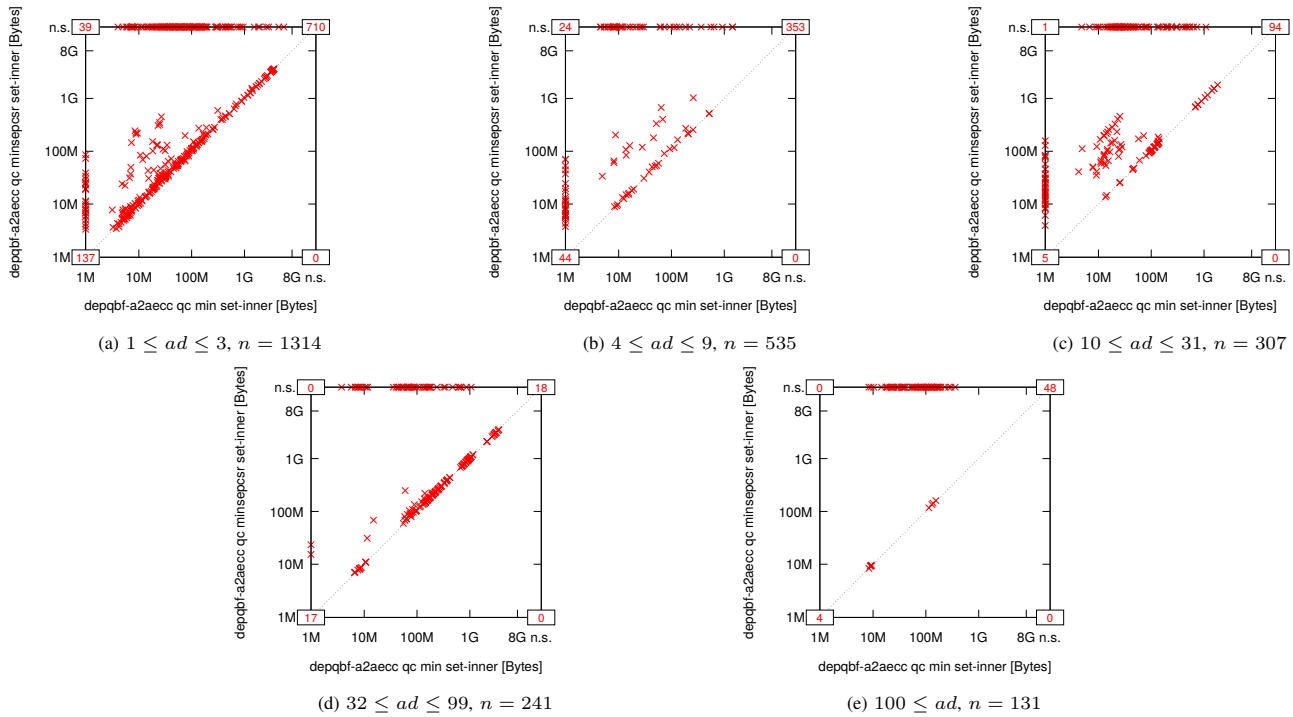


Fig. 1244: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode qc min set-inner partitioned by alternation depth (memory usage in Bytes).

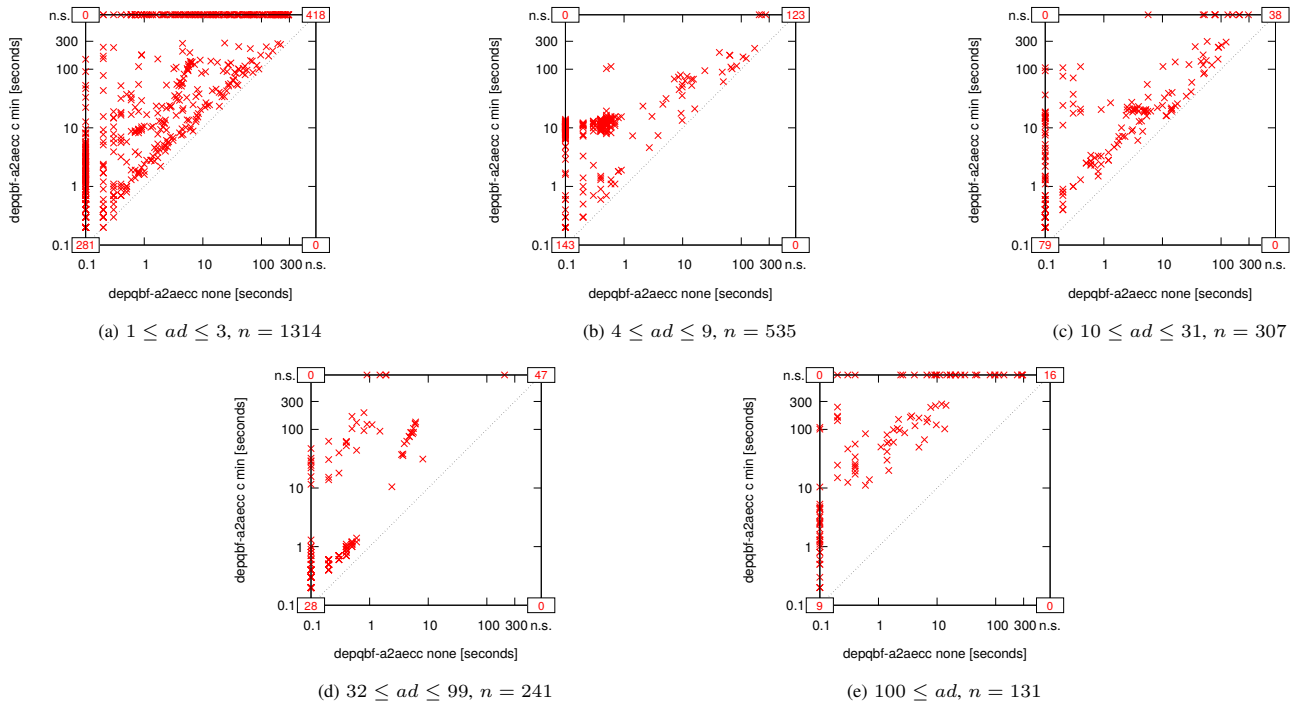


Fig. 1245: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode none partitioned by alternation depth (run time in seconds).

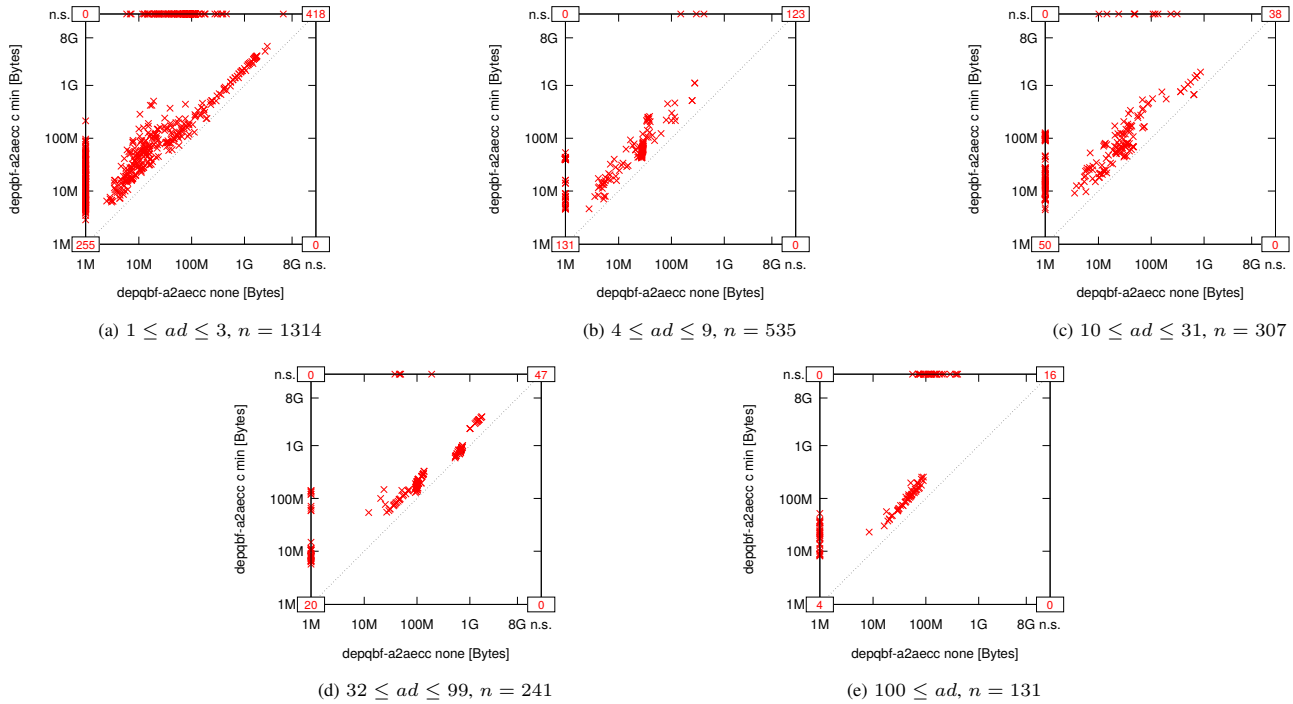


Fig. 1246: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode none partitioned by alternation depth (memory usage in Bytes).

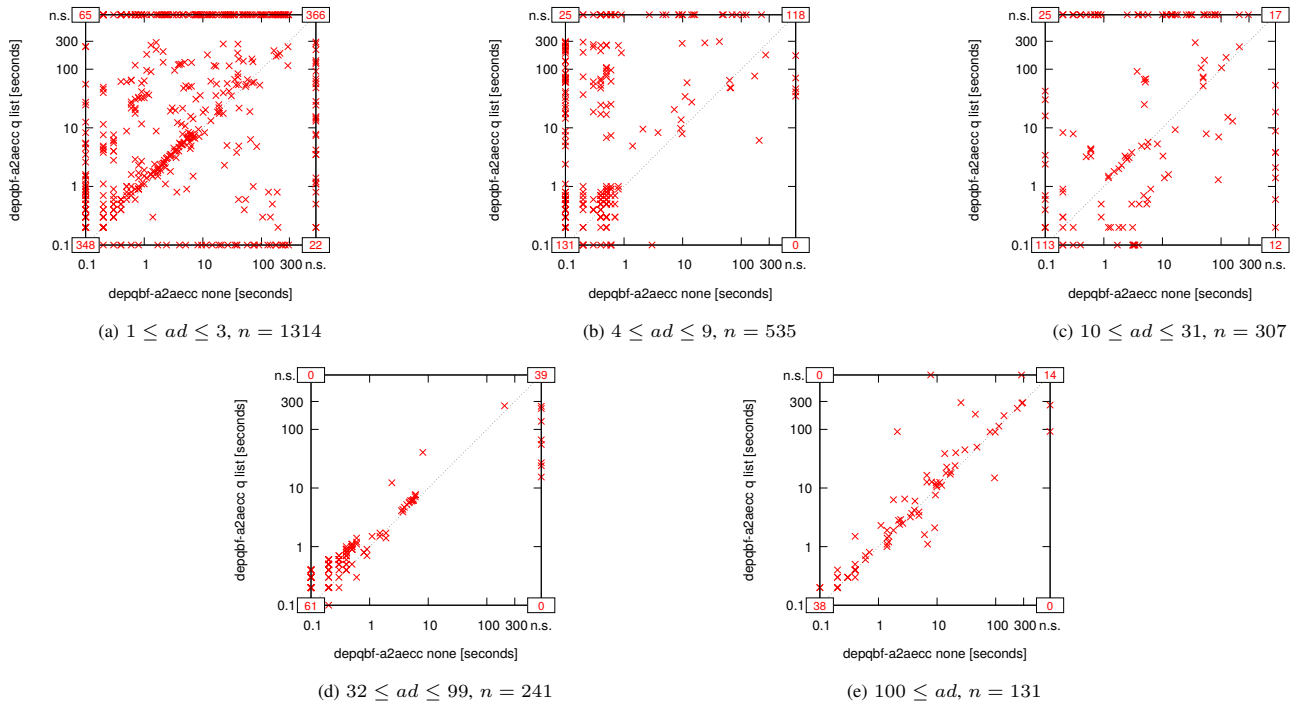


Fig. 1247: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode none partitioned by alternation depth (run time in seconds).

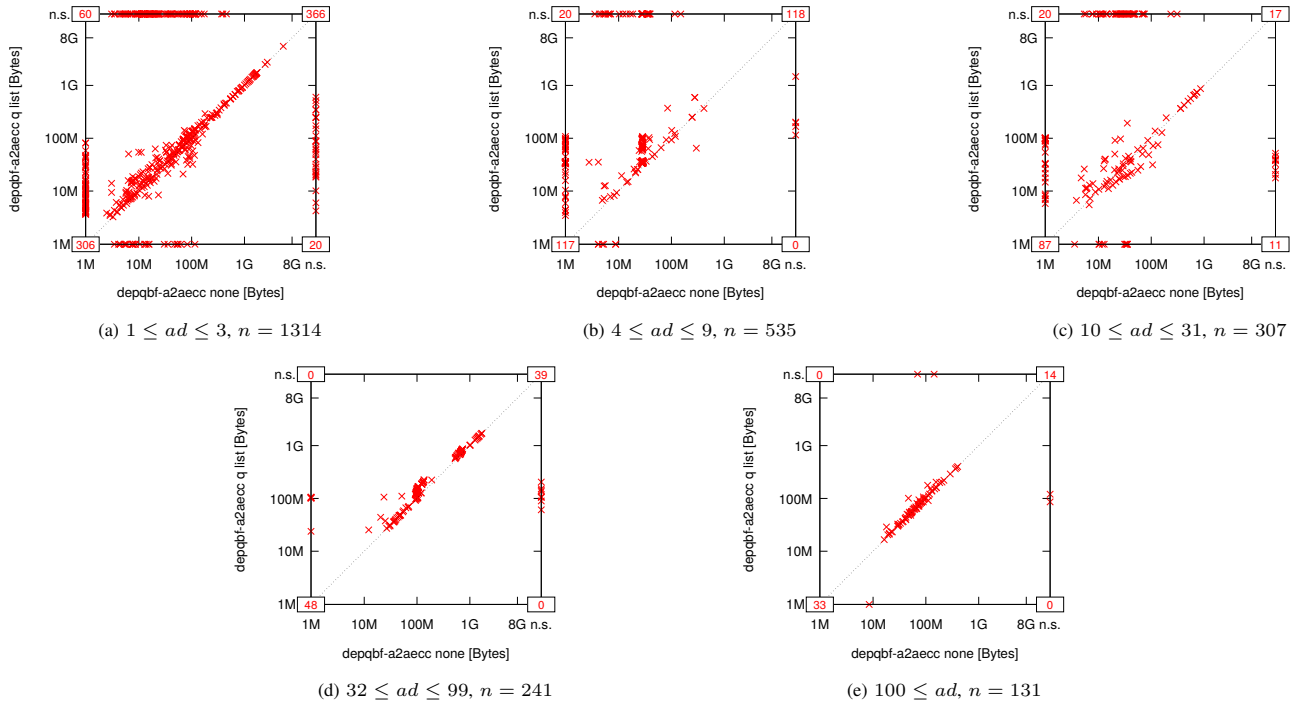


Fig. 1248: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode none partitioned by alternation depth (memory usage in Bytes).

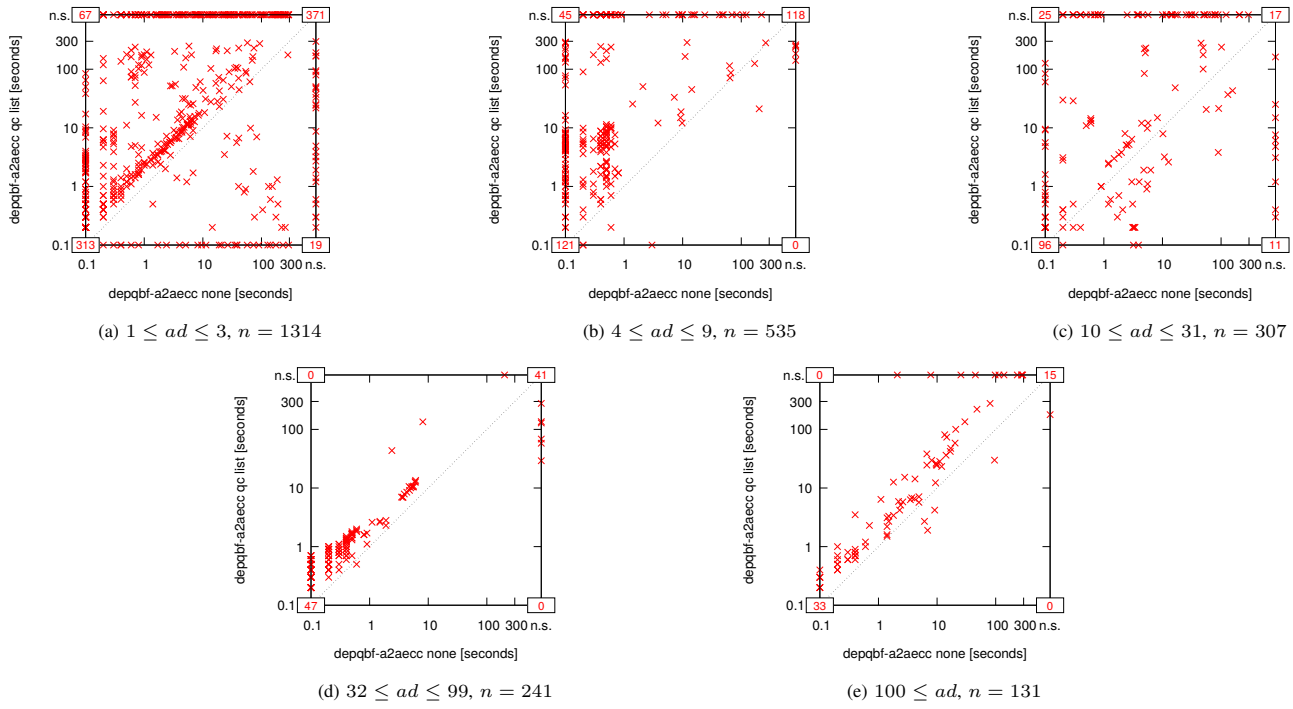


Fig. 1249: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode none partitioned by alternation depth (run time in seconds).

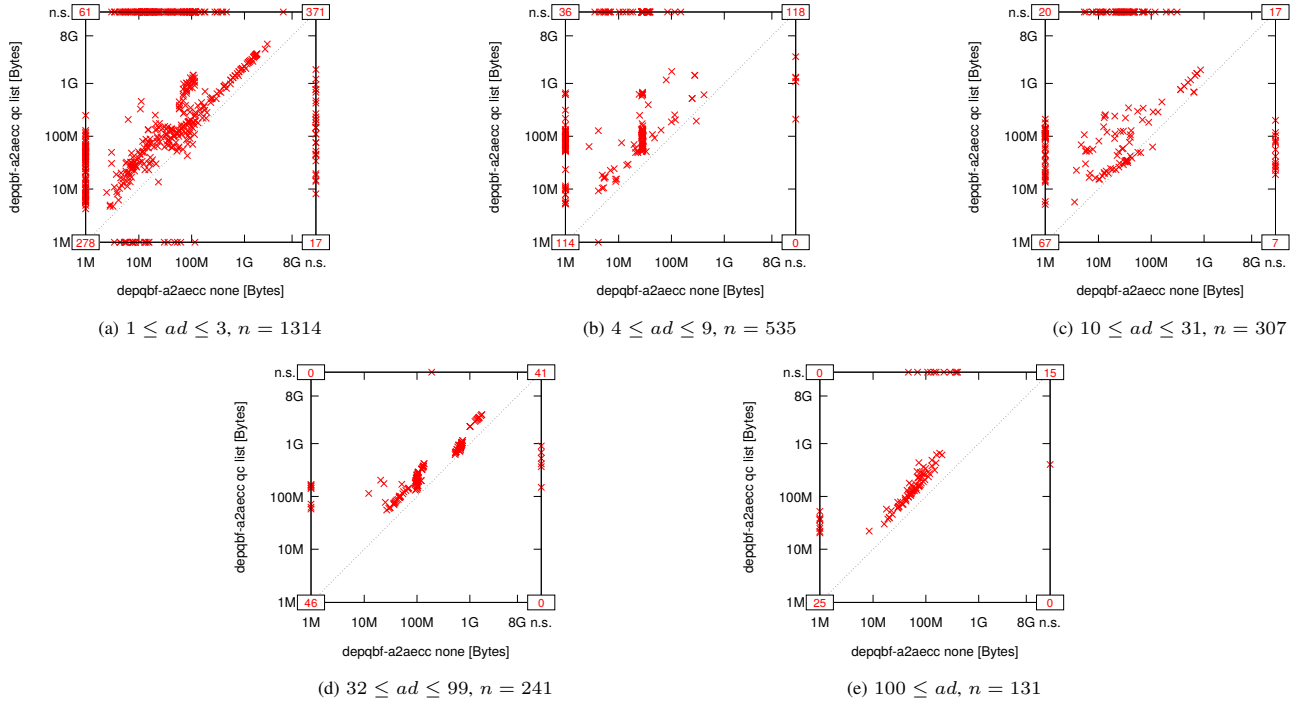


Fig. 1250: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode none partitioned by alternation depth (memory usage in Bytes).

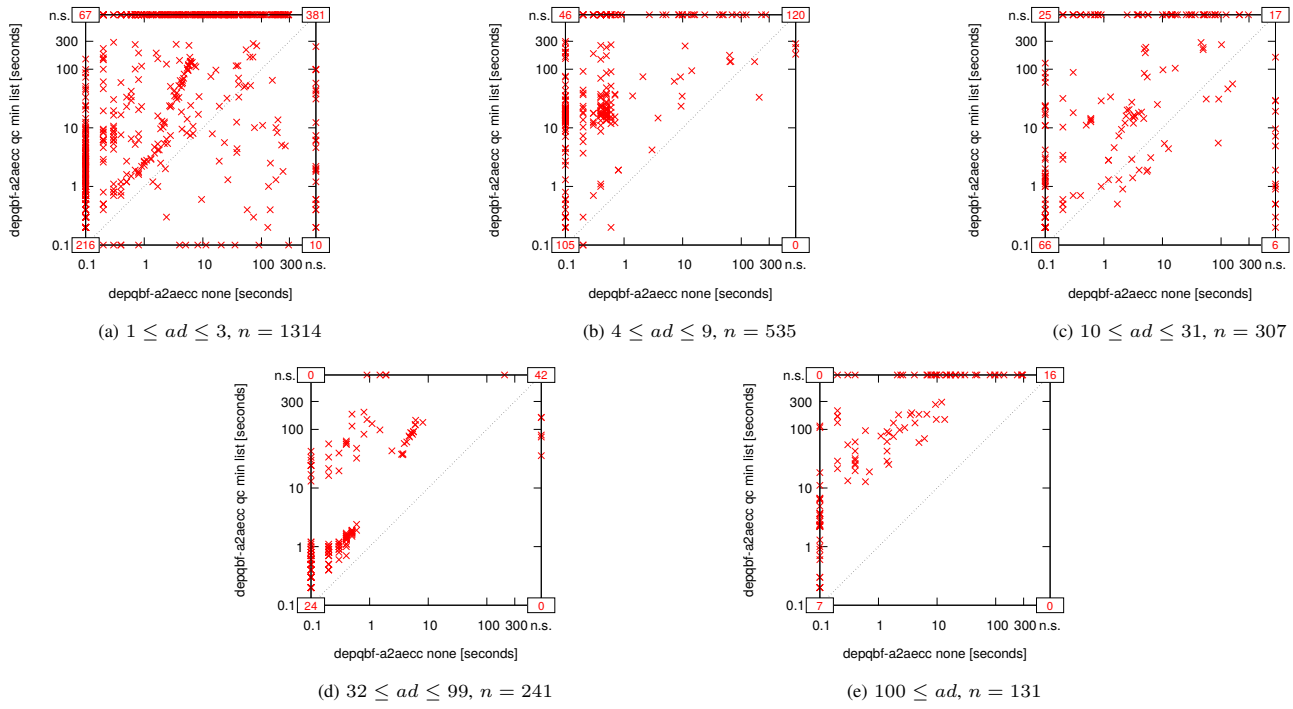


Fig. 1251: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode none partitioned by alternation depth (run time in seconds).

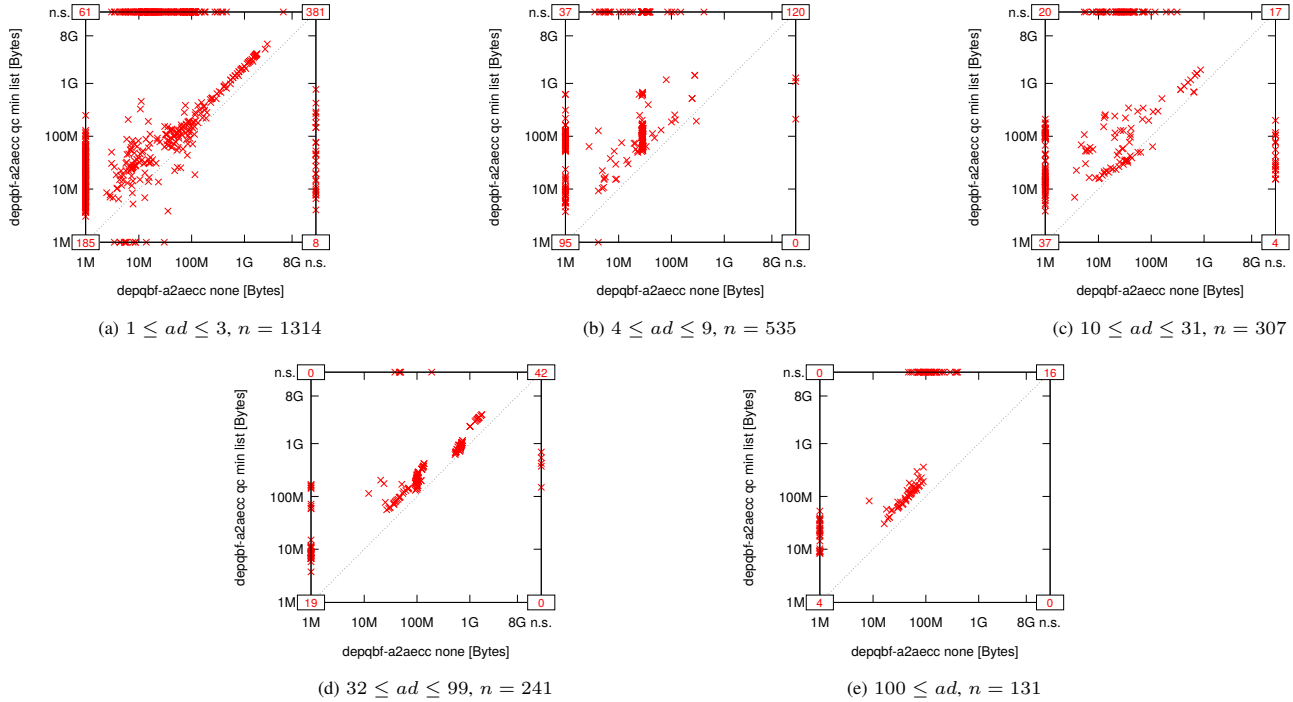


Fig. 1252: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode none partitioned by alternation depth (memory usage in Bytes).

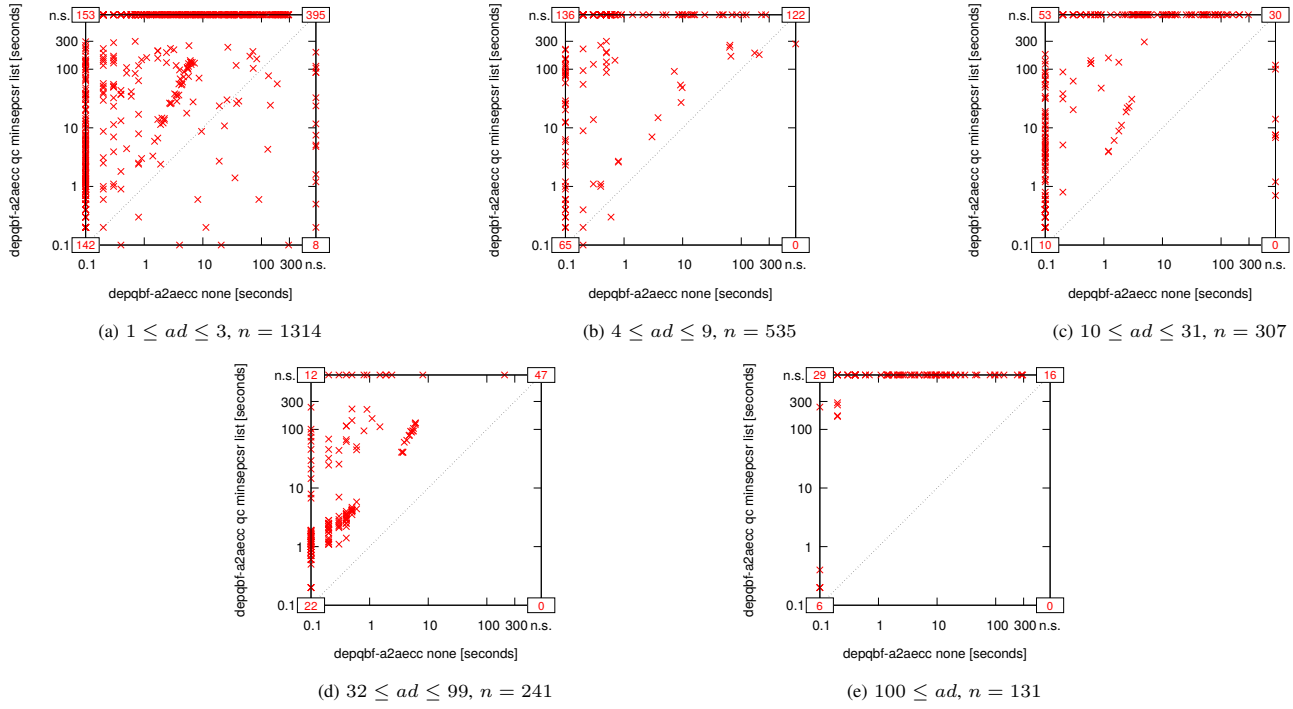


Fig. 1253: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode none partitioned by alternation depth (run time in seconds).

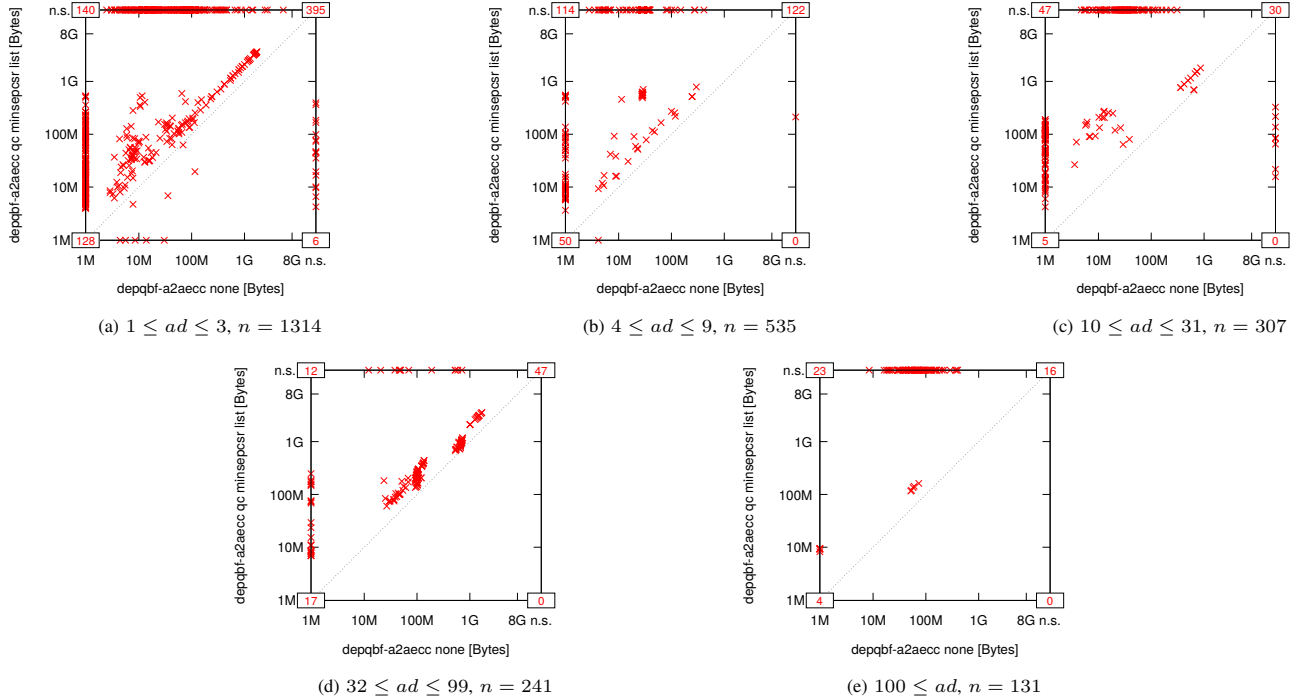


Fig. 1254: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode none partitioned by alternation depth (memory usage in Bytes).

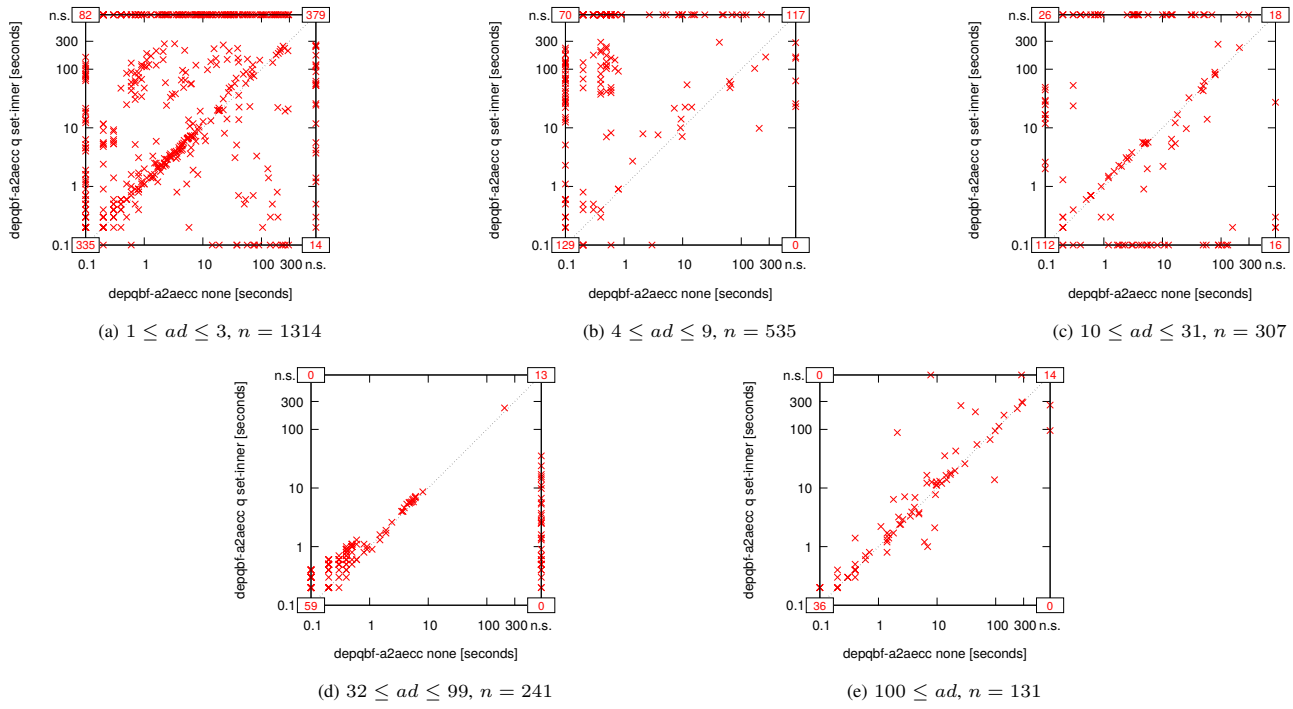


Fig. 1255: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode none partitioned by alternation depth (run time in seconds).

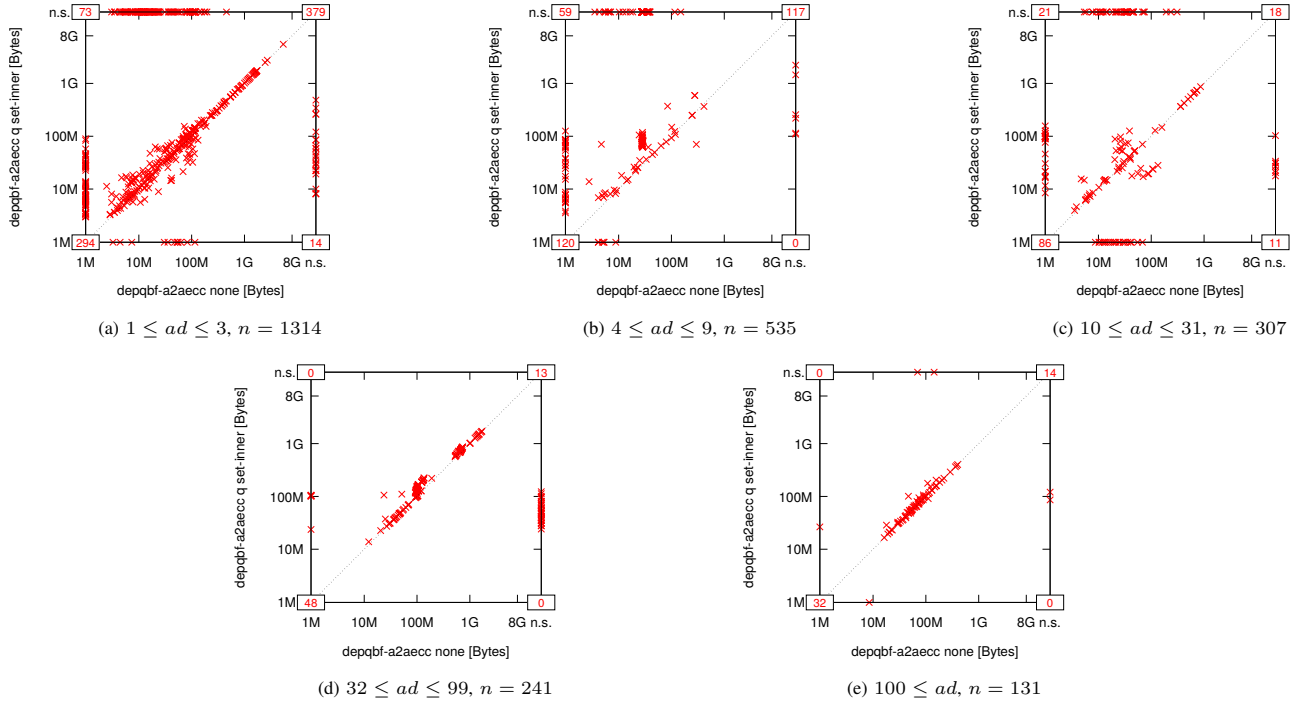


Fig. 1256: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode none partitioned by alternation depth (memory usage in Bytes).

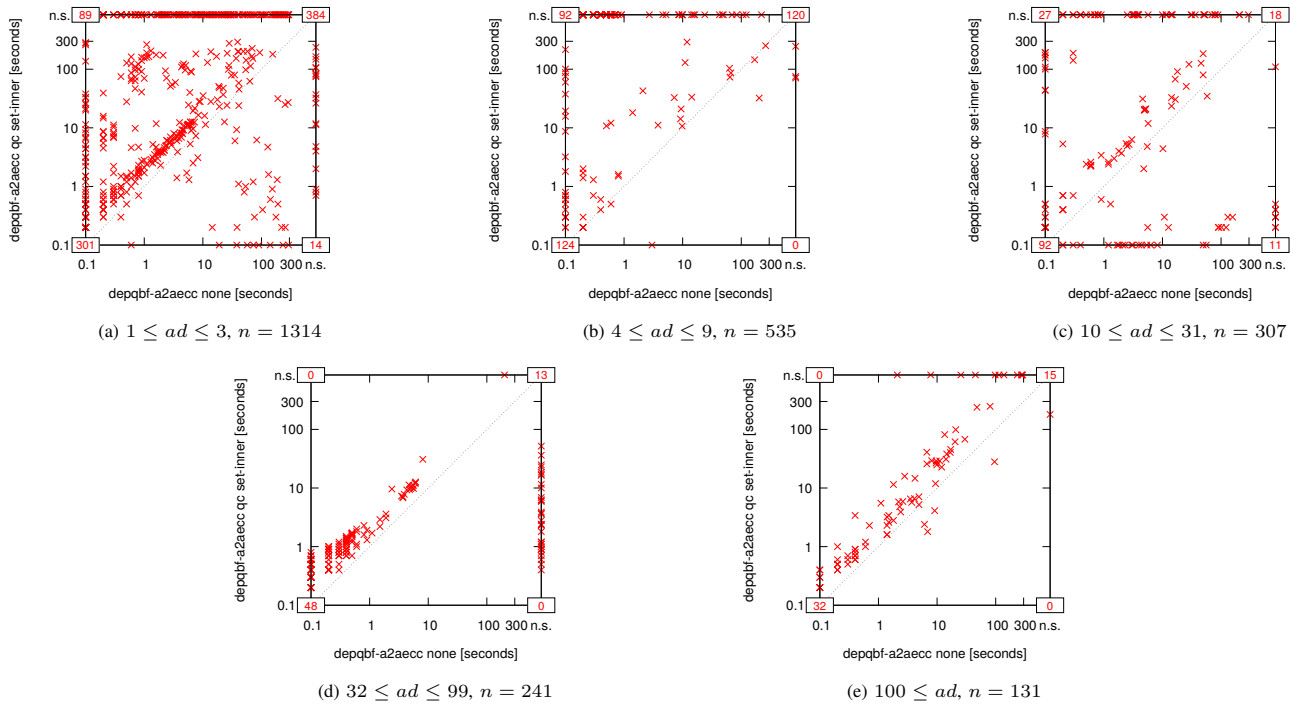


Fig. 1257: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode none partitioned by alternation depth (run time in seconds).

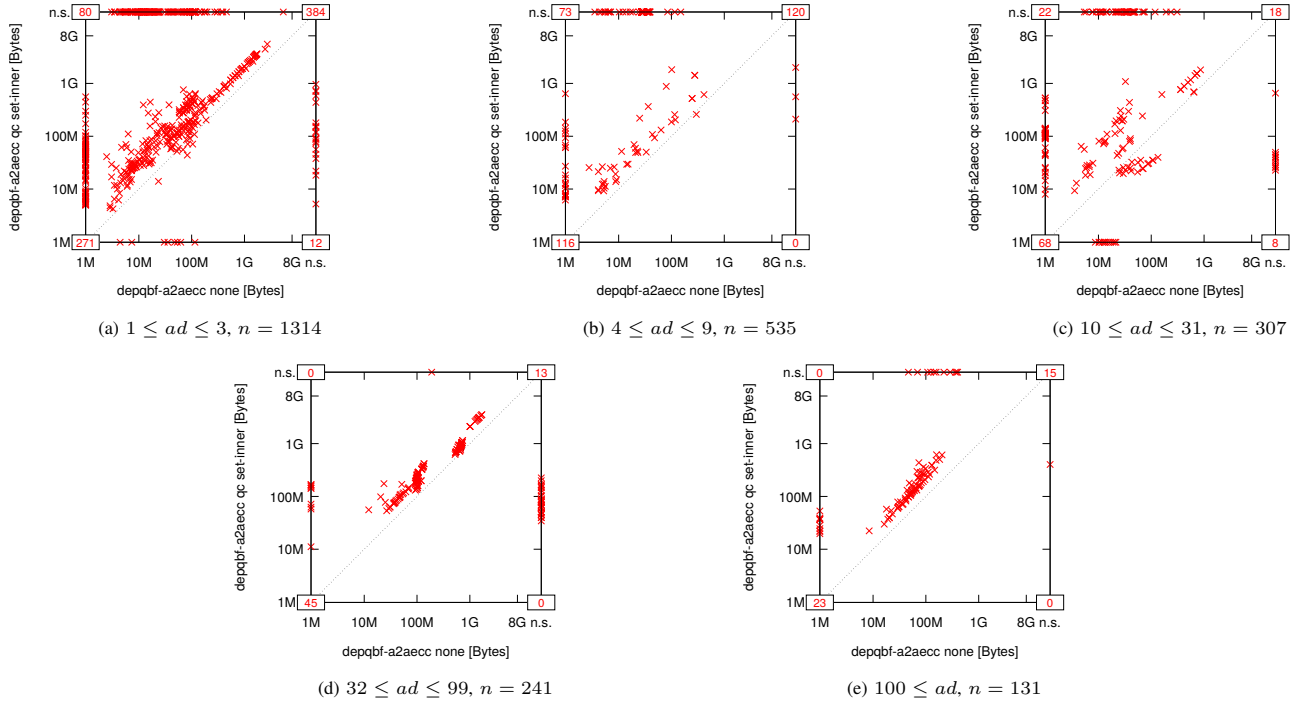


Fig. 1258: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode none partitioned by alternation depth (memory usage in Bytes).

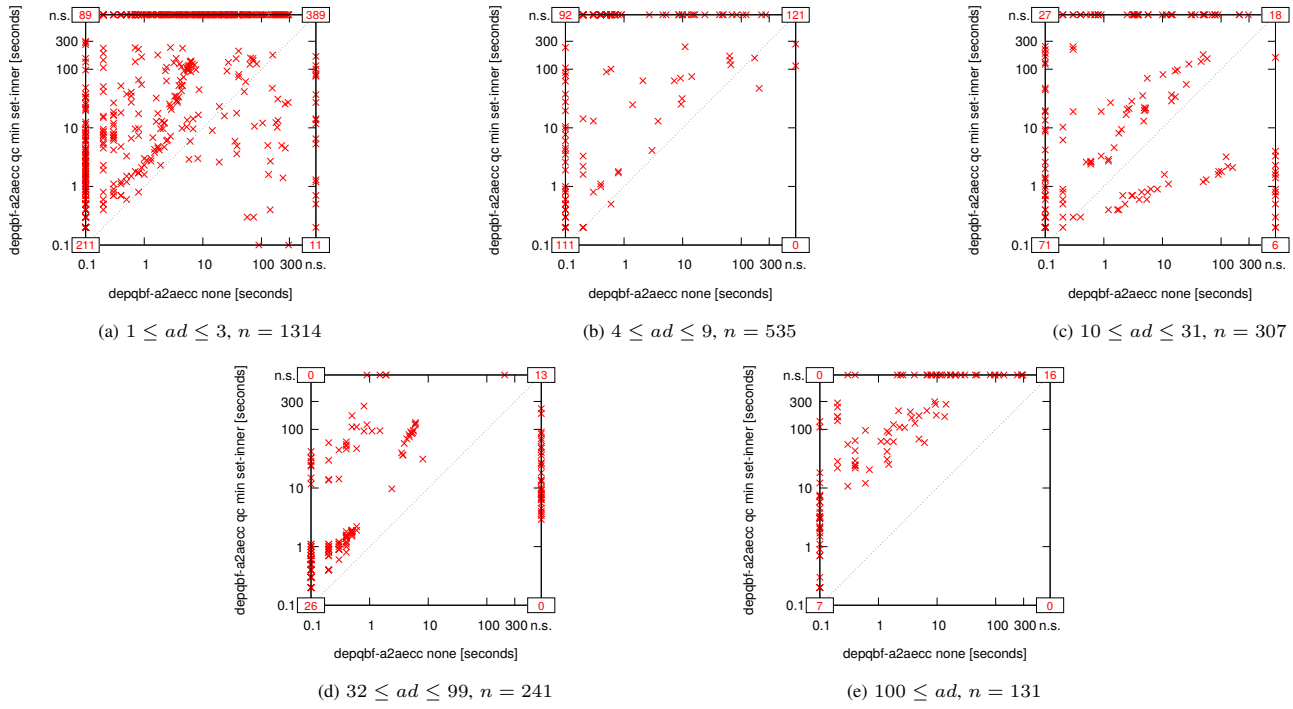


Fig. 1259: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode none partitioned by alternation depth (run time in seconds).

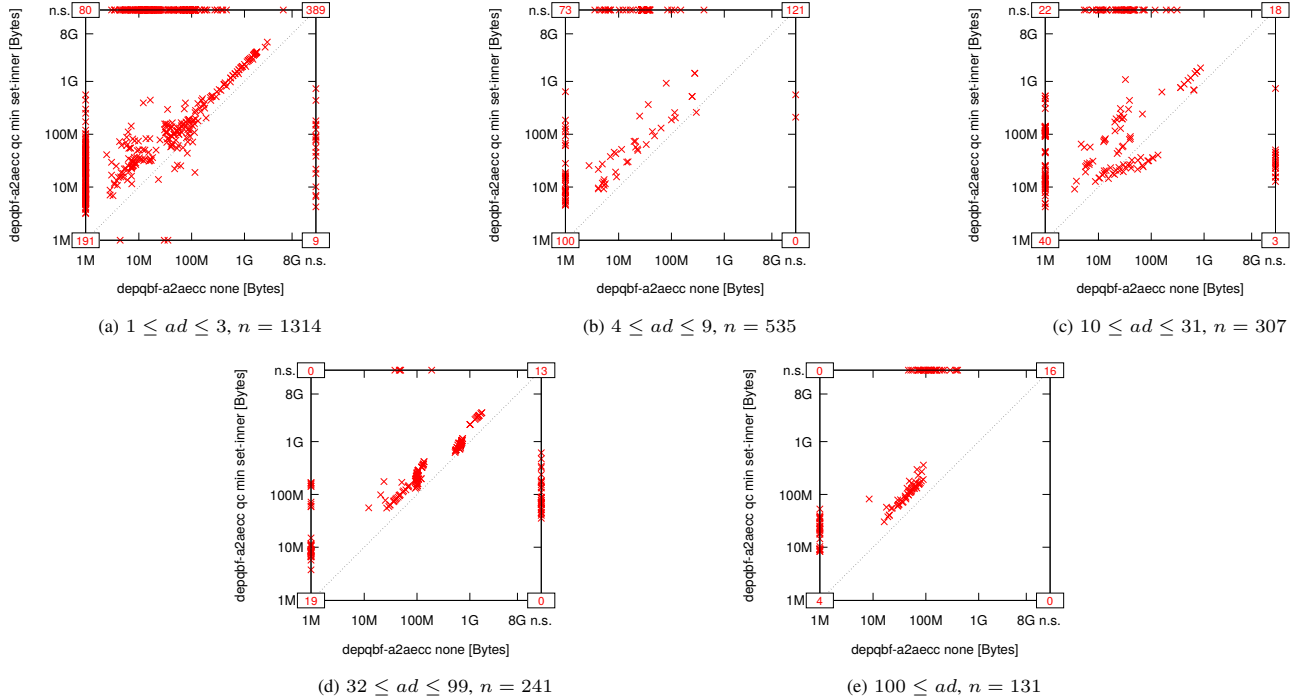


Fig. 1260: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode none partitioned by alternation depth (memory usage in Bytes).

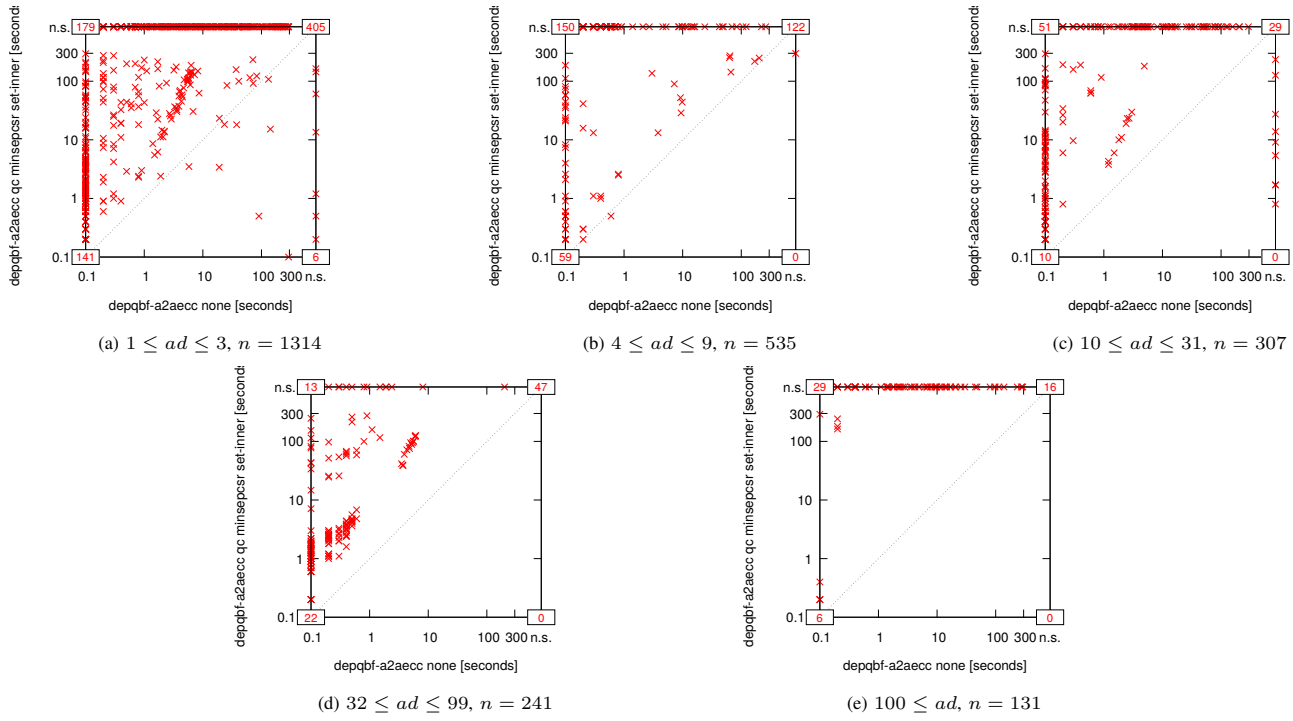


Fig. 1261: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode none partitioned by alternation depth (run time in seconds).

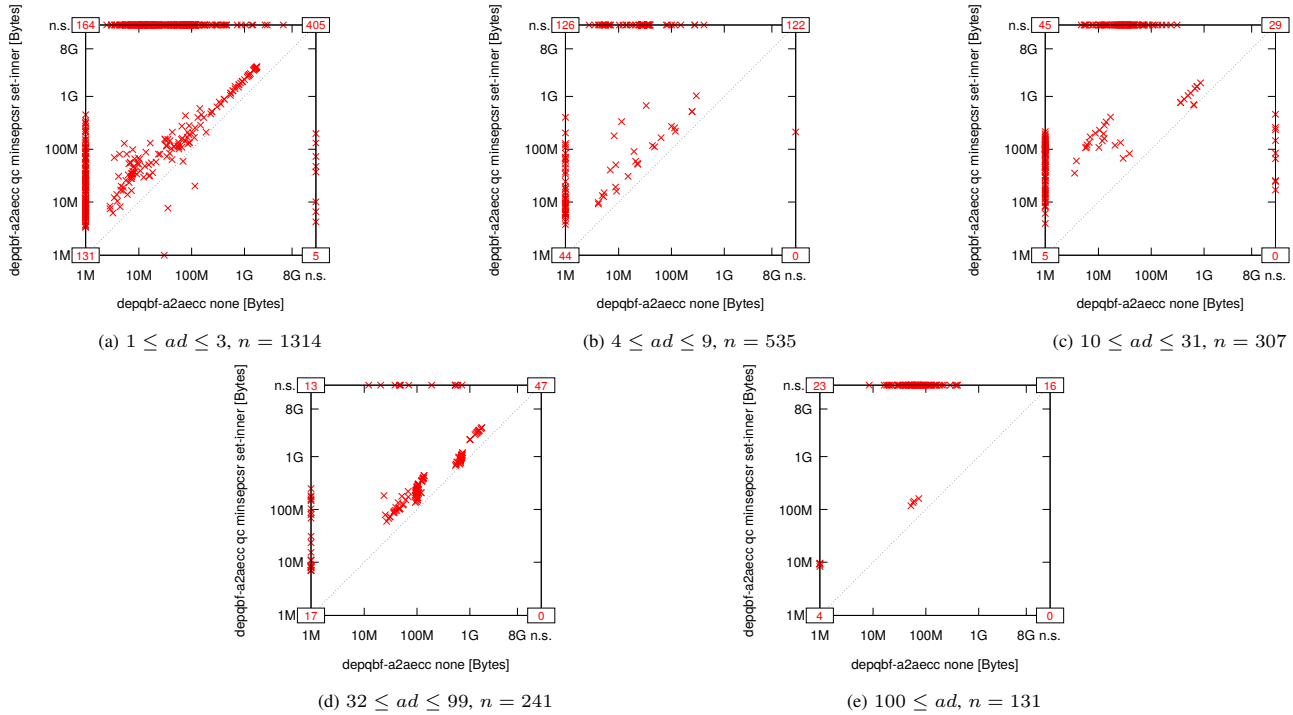


Fig. 1262: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode none partitioned by alternation depth (memory usage in Bytes).

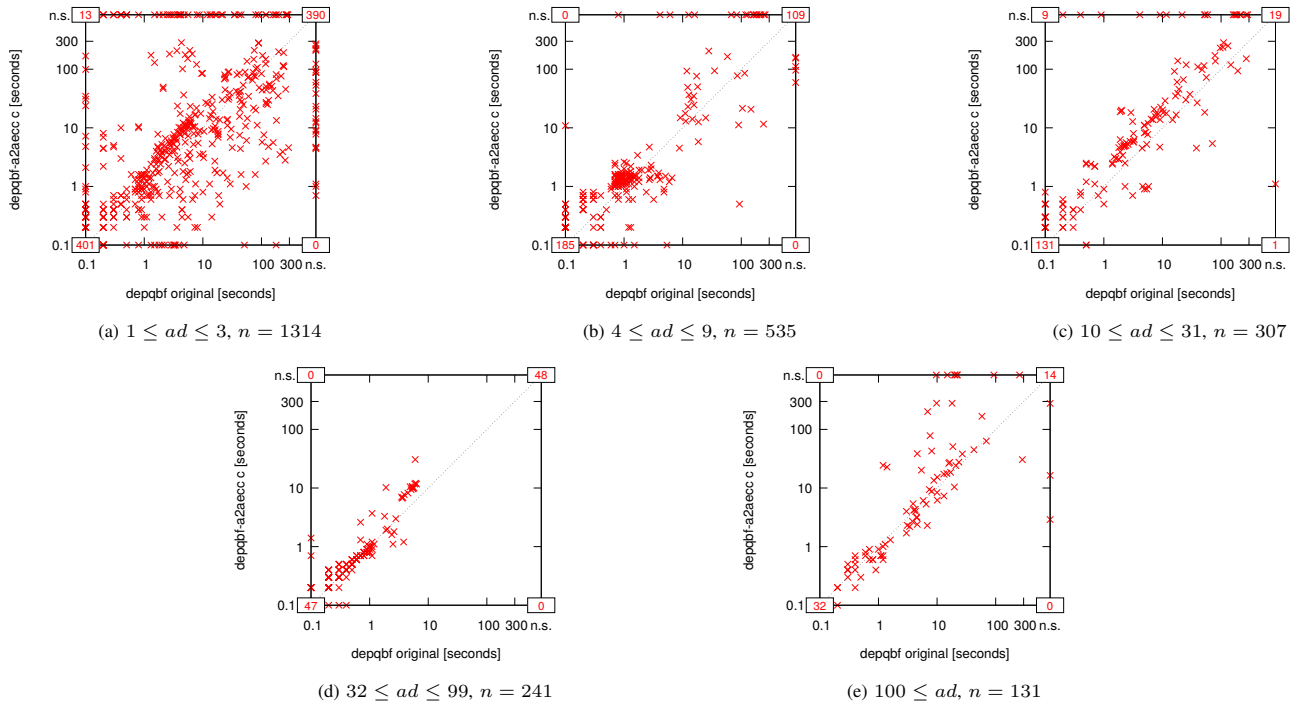


Fig. 1263: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c versus mode original partitioned by alternation depth (run time in seconds).

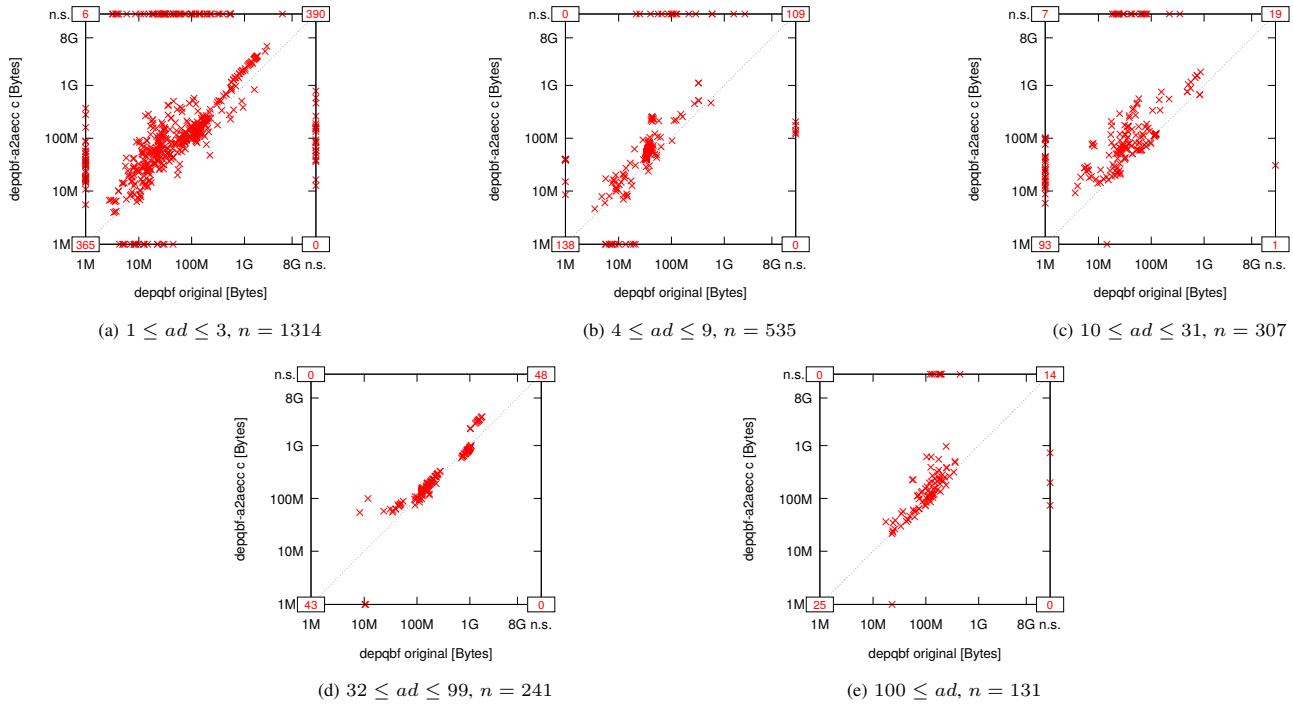


Fig. 1264: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c versus mode original partitioned by alternation depth (memory usage in Bytes).

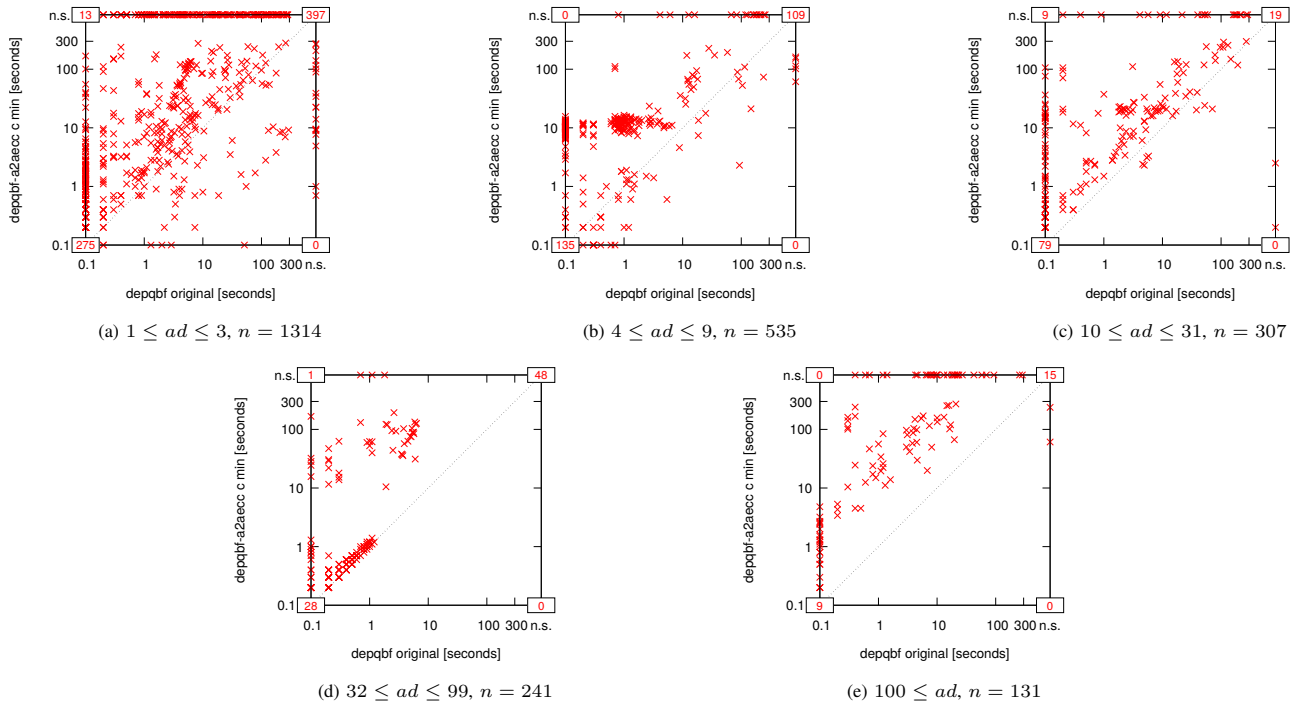


Fig. 1265: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode original partitioned by alternation depth (run time in seconds).

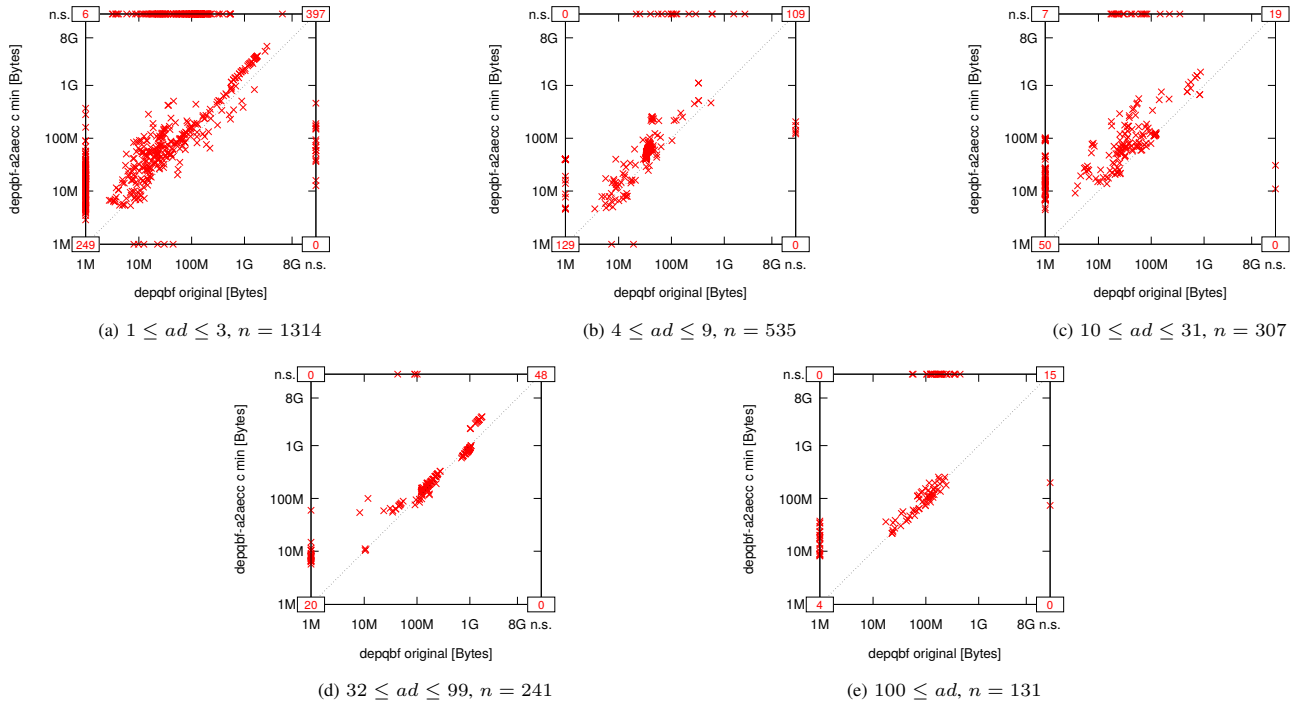


Fig. 1266: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode original partitioned by alternation depth (memory usage in Bytes).

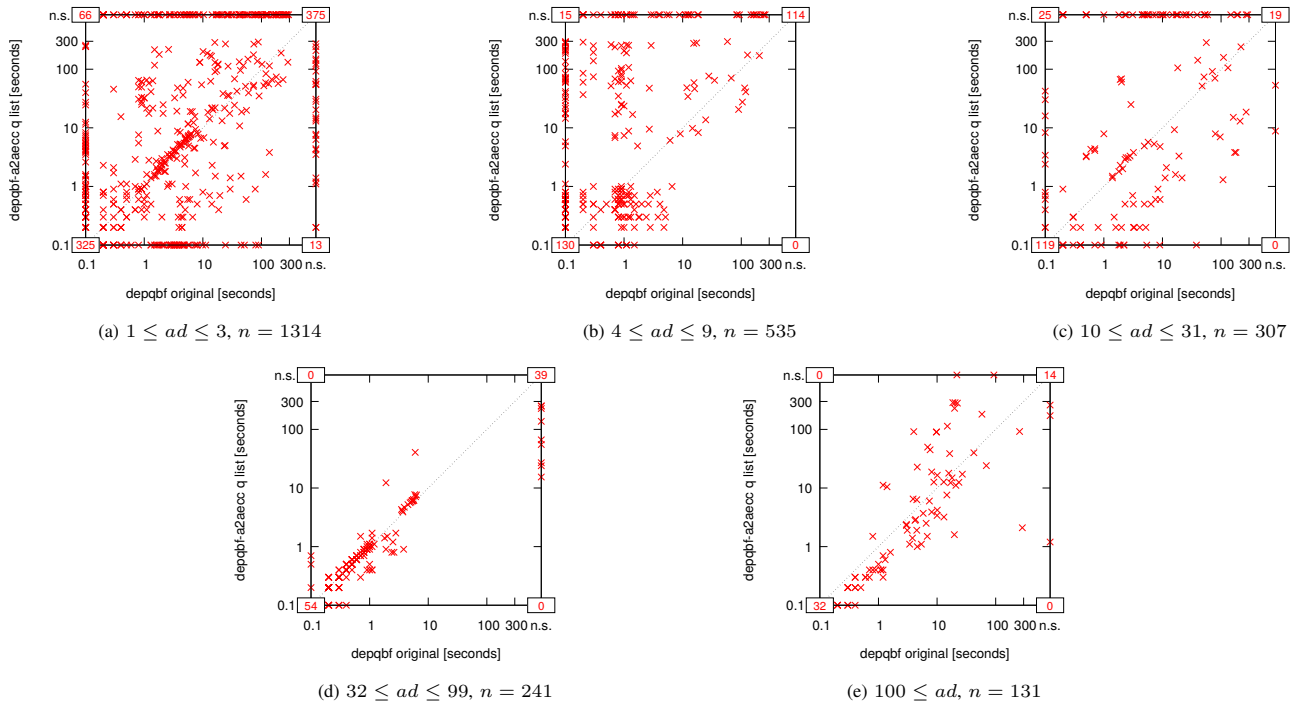


Fig. 1267: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode original partitioned by alternation depth (run time in seconds).

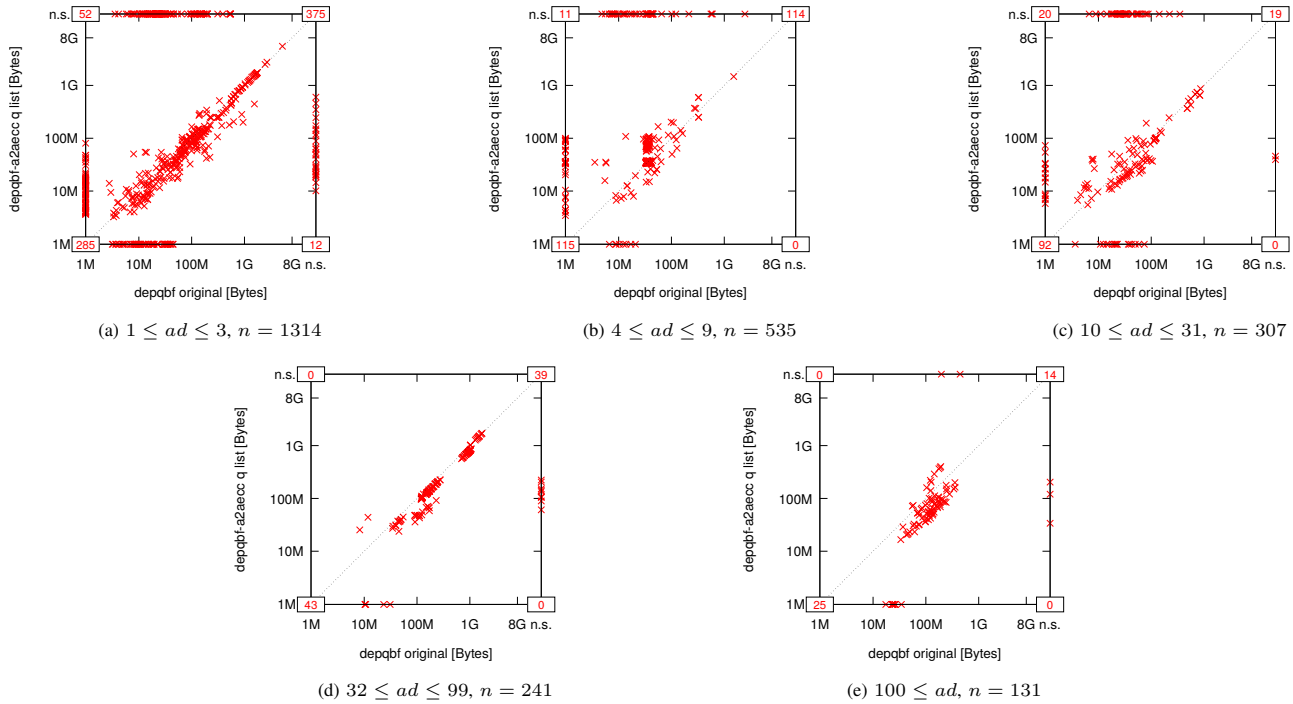


Fig. 1268: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode original partitioned by alternation depth (memory usage in Bytes).

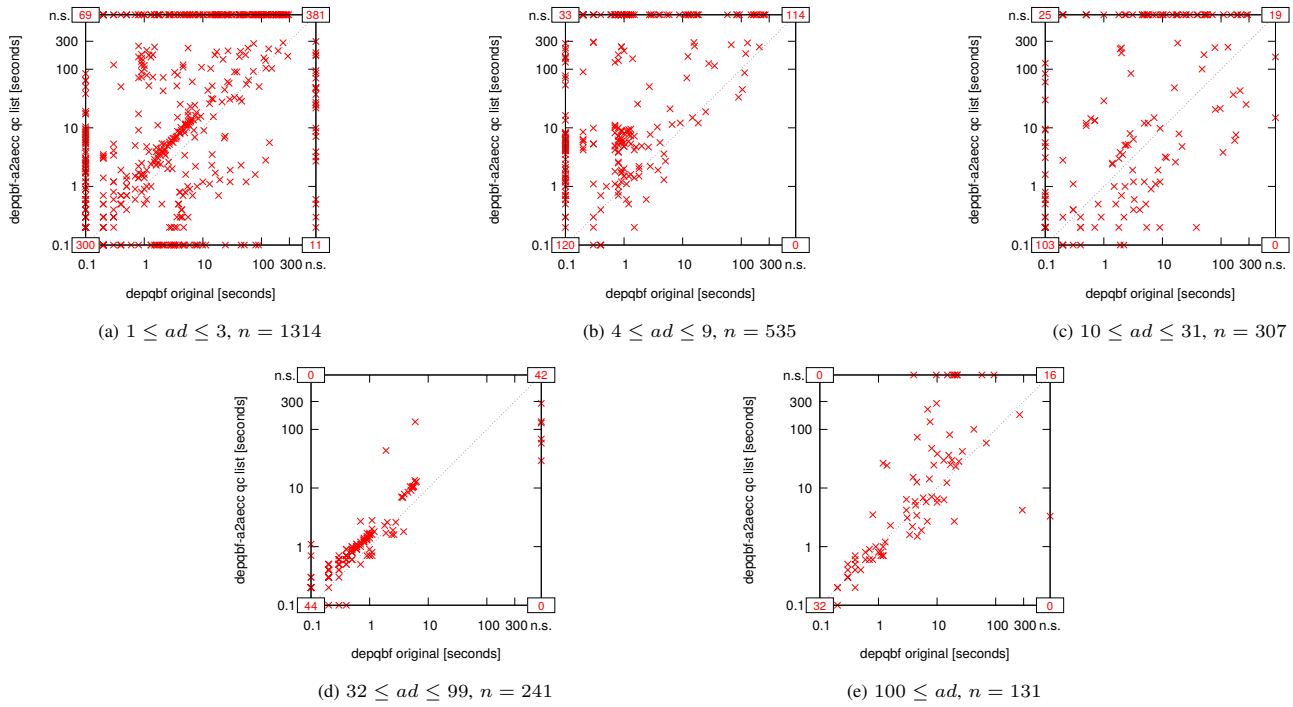


Fig. 1269: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode original partitioned by alternation depth (run time in seconds).

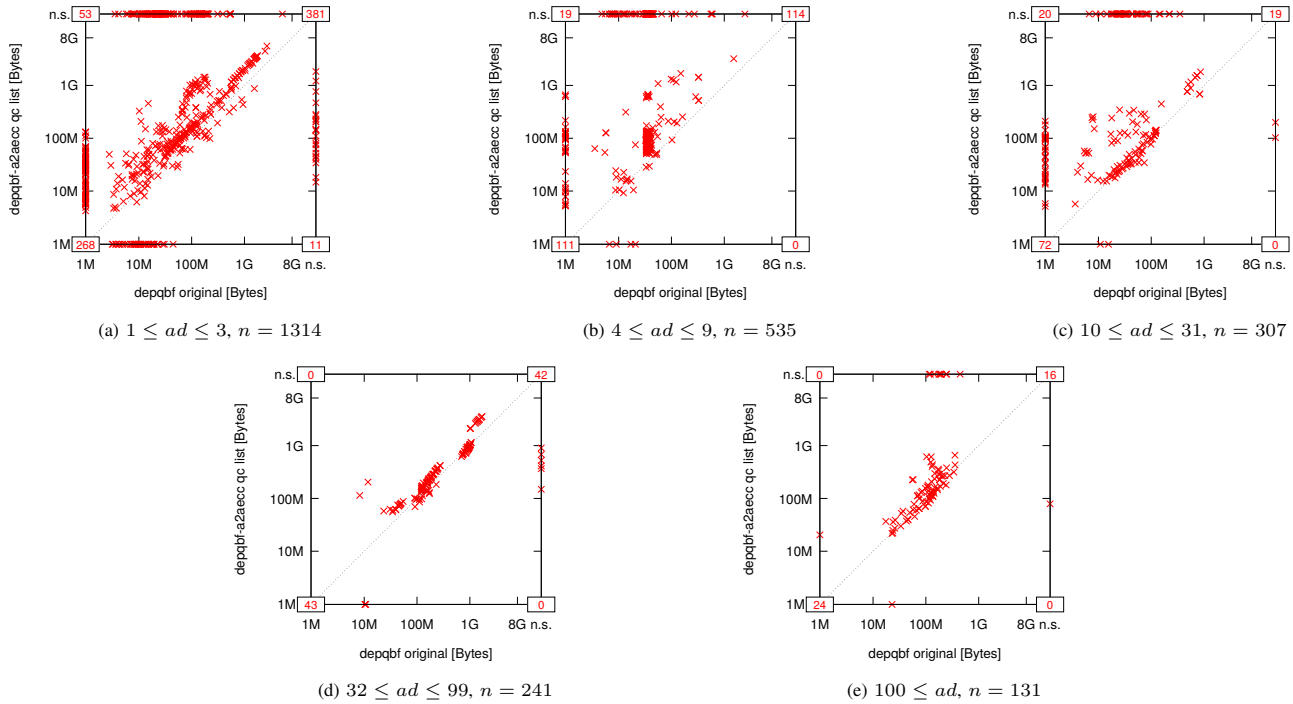


Fig. 1270: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode original partitioned by alternation depth (memory usage in Bytes).

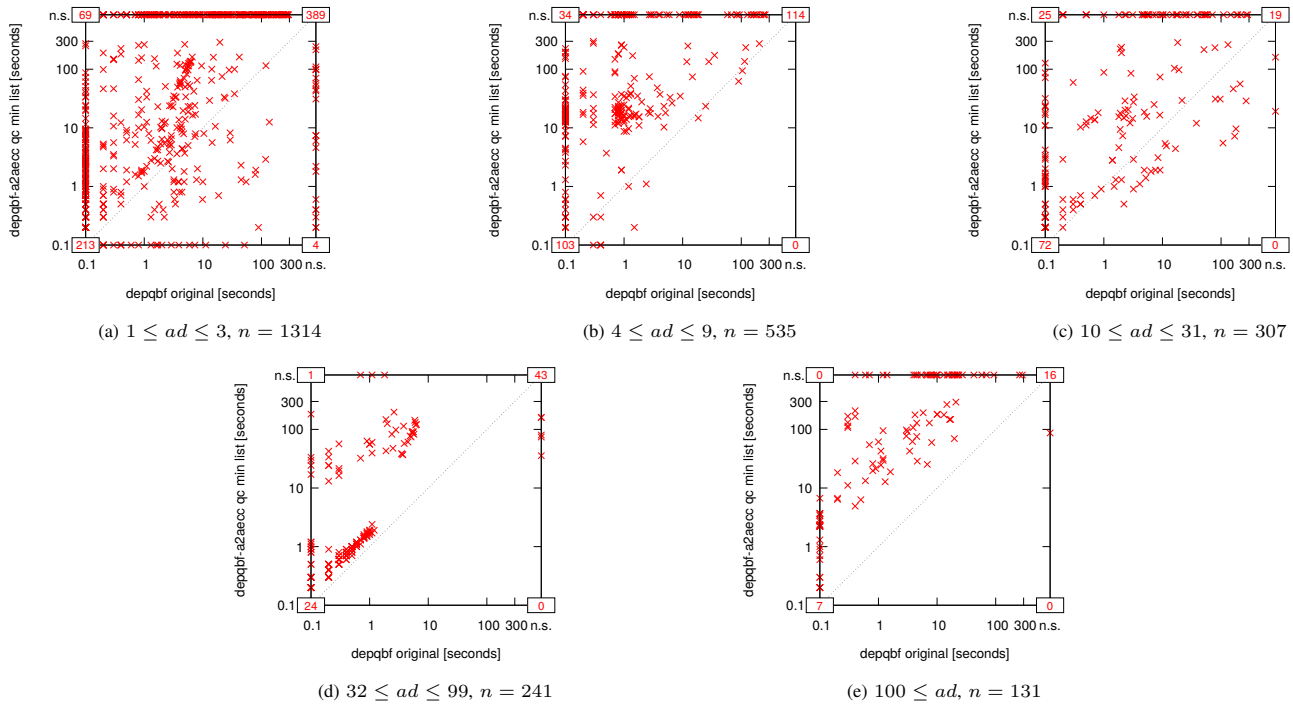


Fig. 1271: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode original partitioned by alternation depth (run time in seconds).

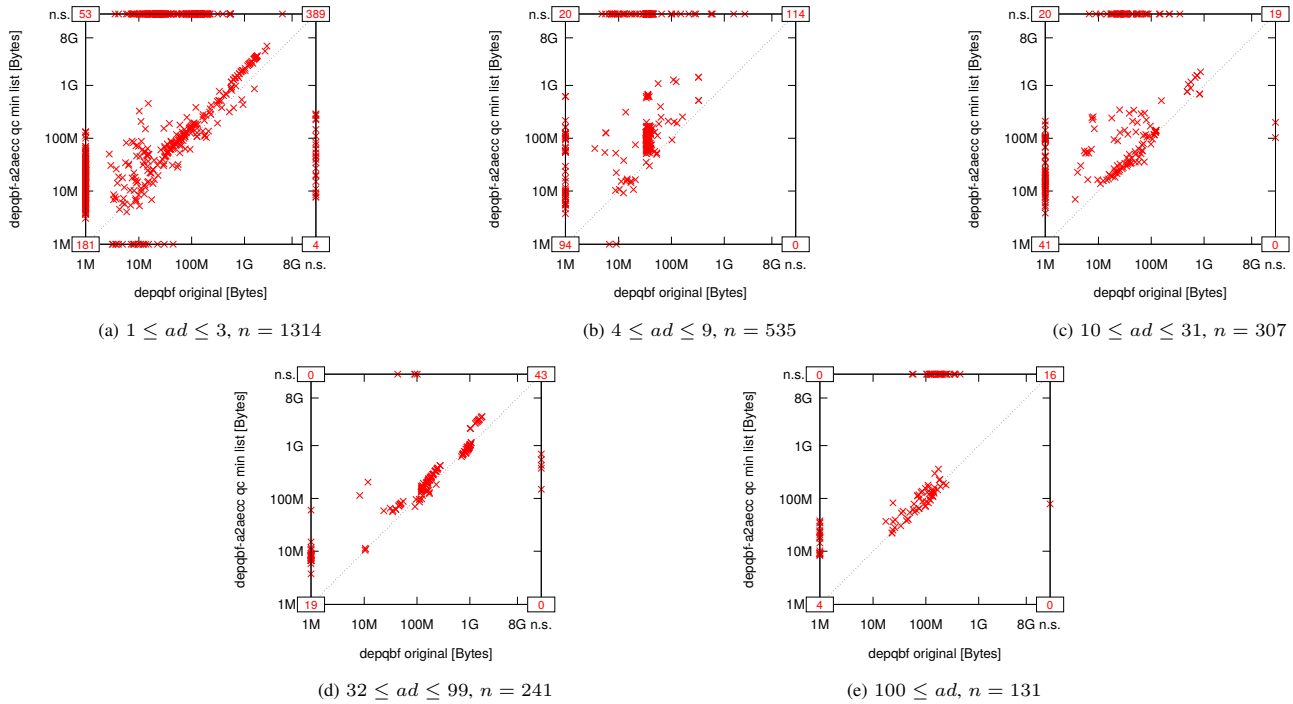


Fig. 1272: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode original partitioned by alternation depth (memory usage in Bytes).

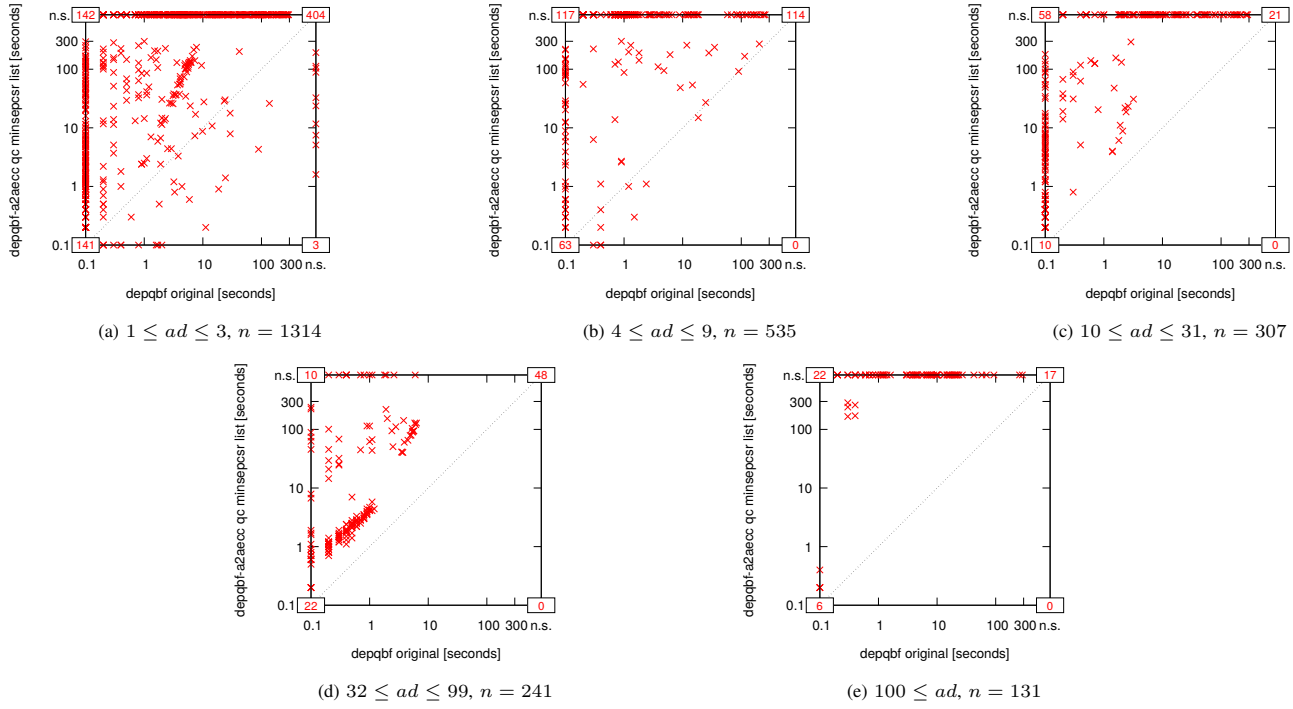


Fig. 1273: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode original partitioned by alternation depth (run time in seconds).

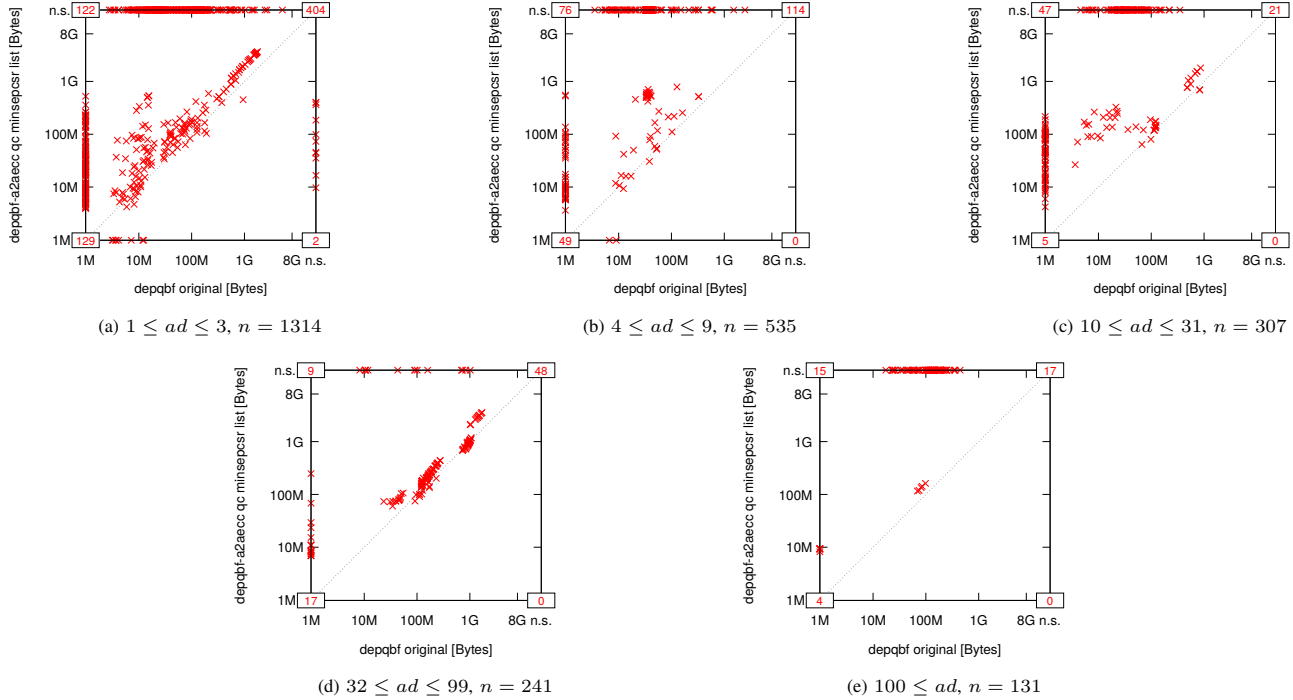


Fig. 1274: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode original partitioned by alternation depth (memory usage in Bytes).

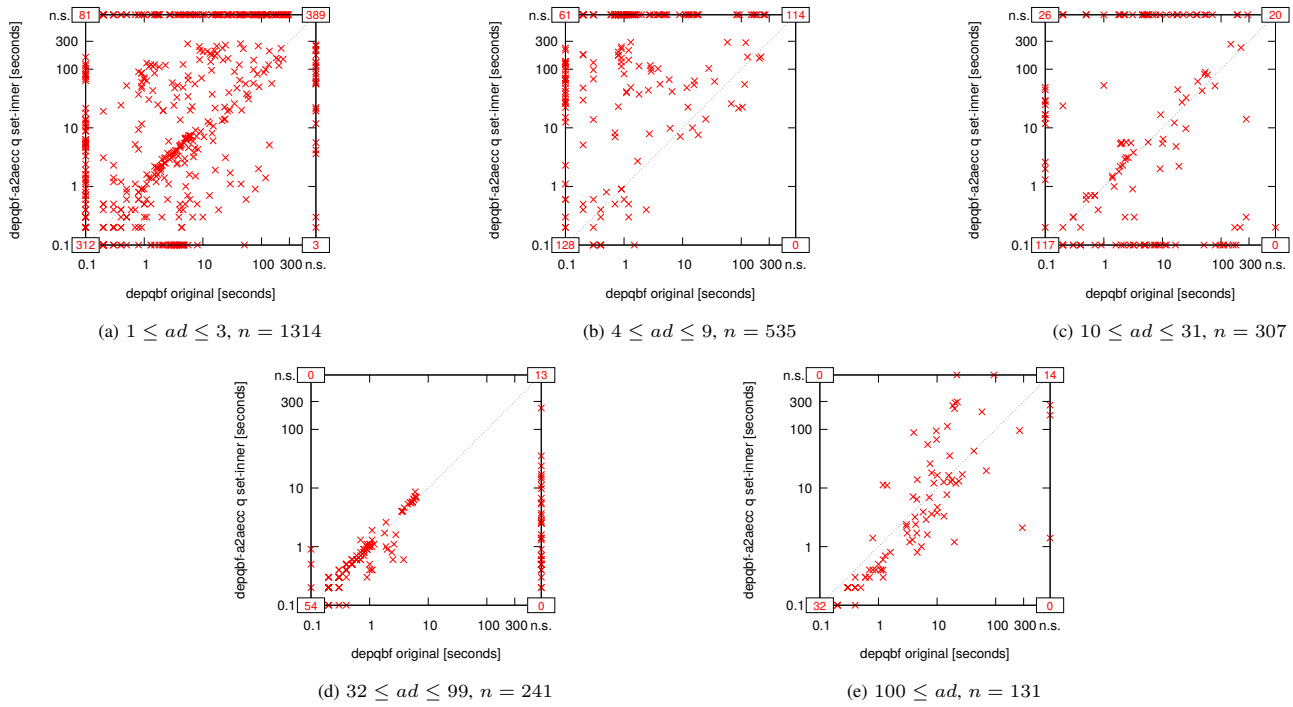


Fig. 1275: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode original partitioned by alternation depth (run time in seconds).

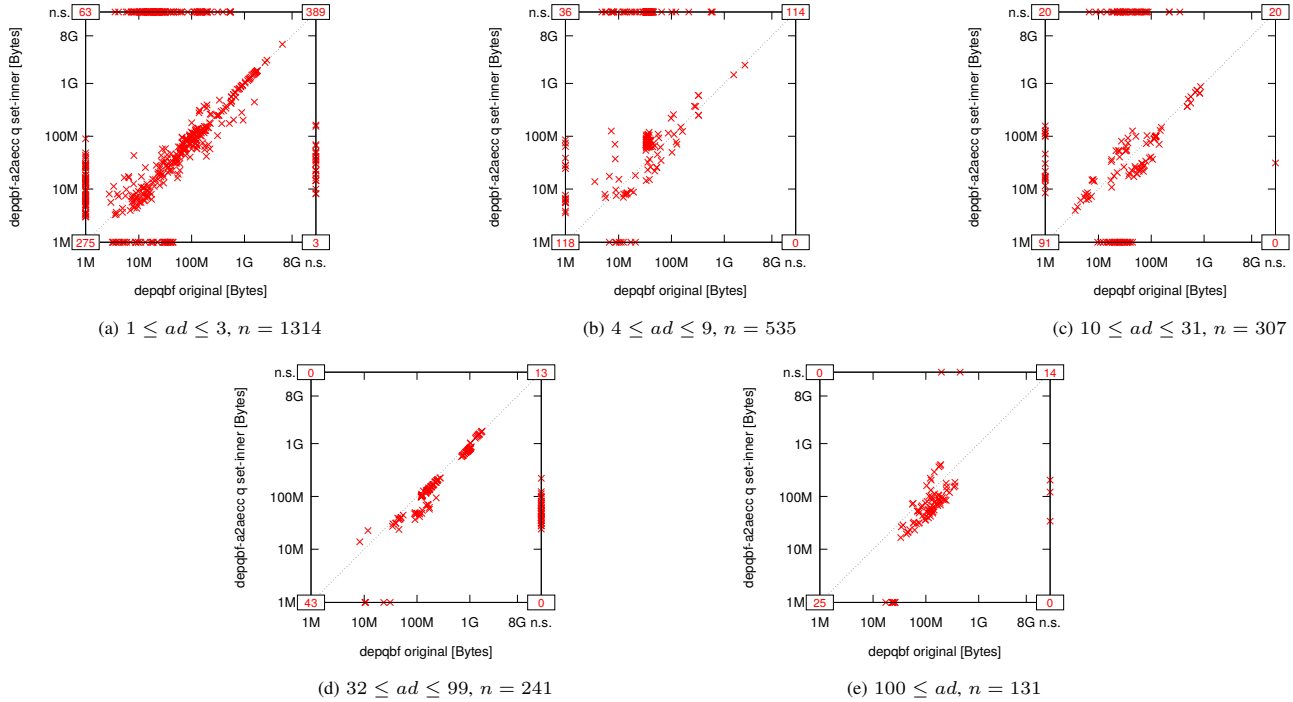


Fig. 1276: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode original partitioned by alternation depth (memory usage in Bytes).

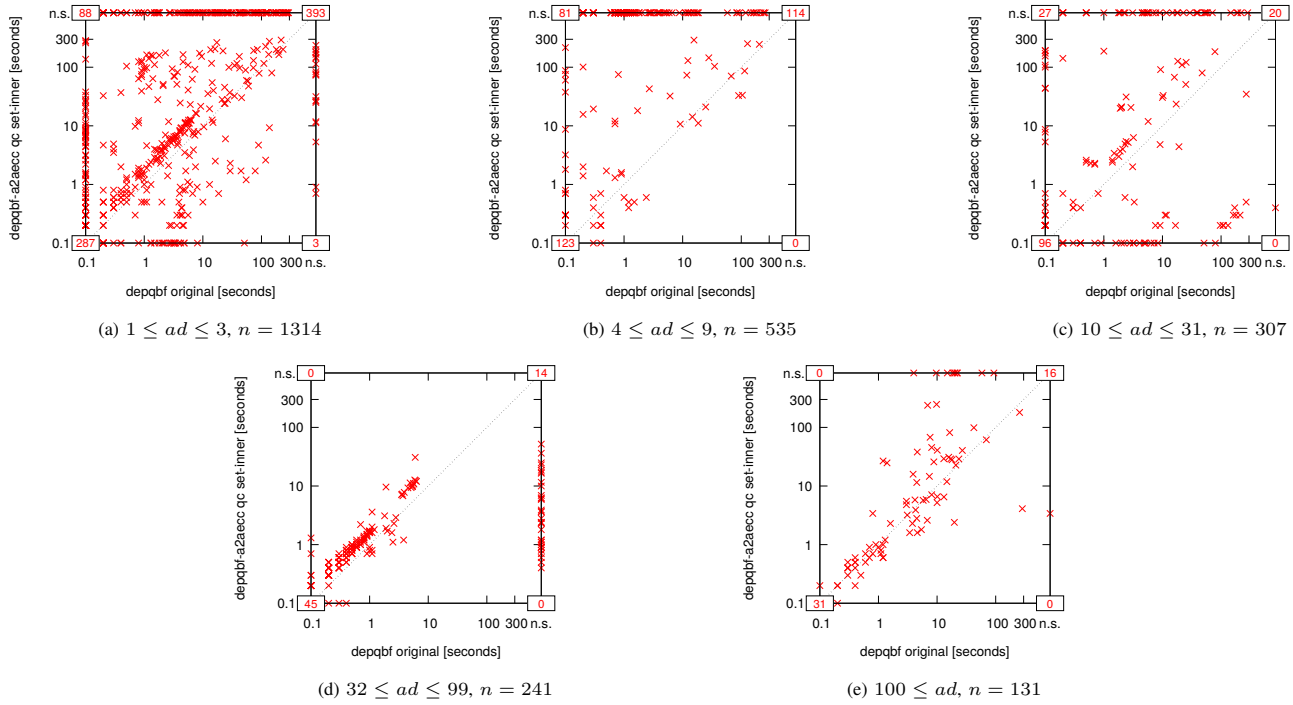


Fig. 1277: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode original partitioned by alternation depth (run time in seconds).

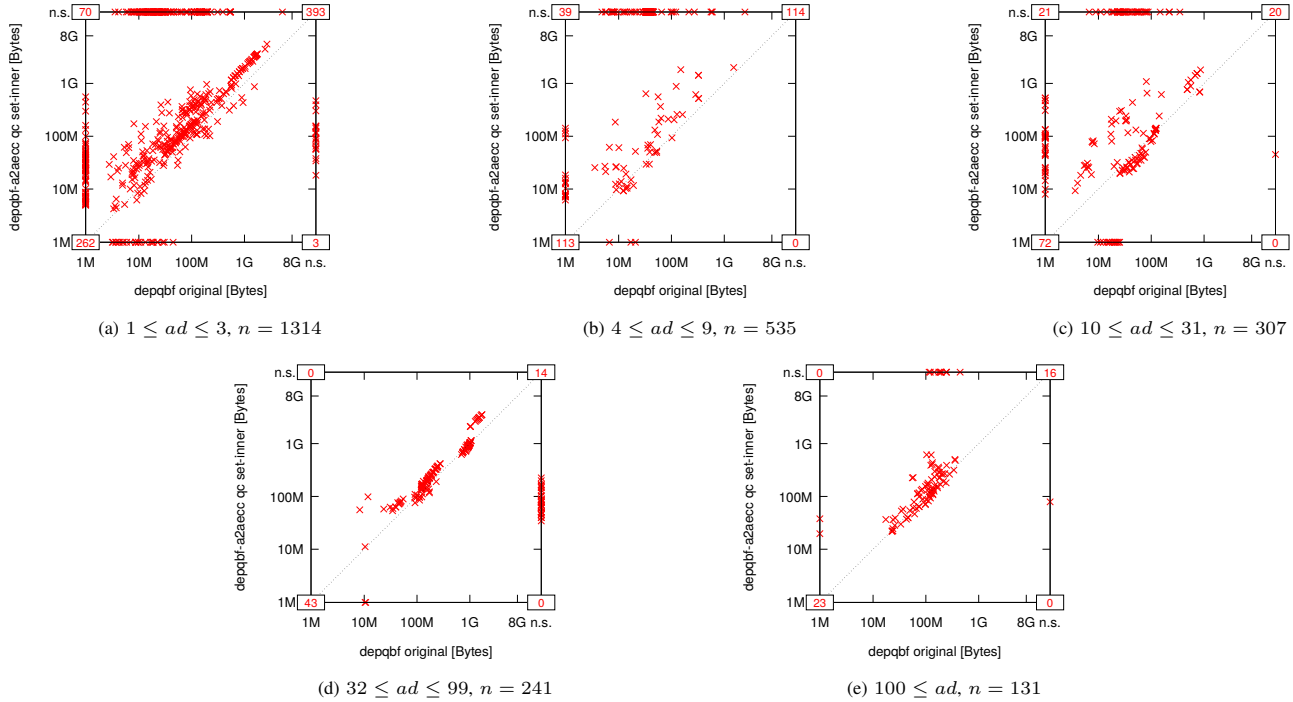


Fig. 1278: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode original partitioned by alternation depth (memory usage in Bytes).

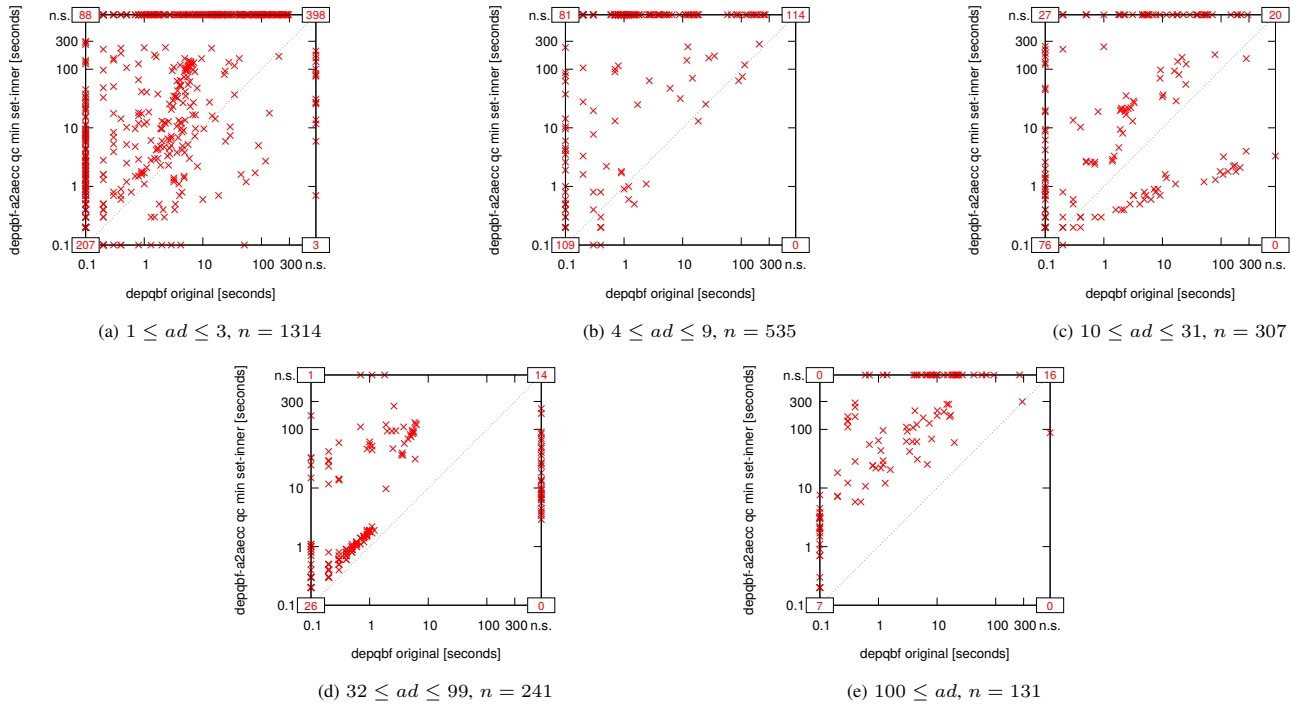


Fig. 1279: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode original partitioned by alternation depth (run time in seconds).

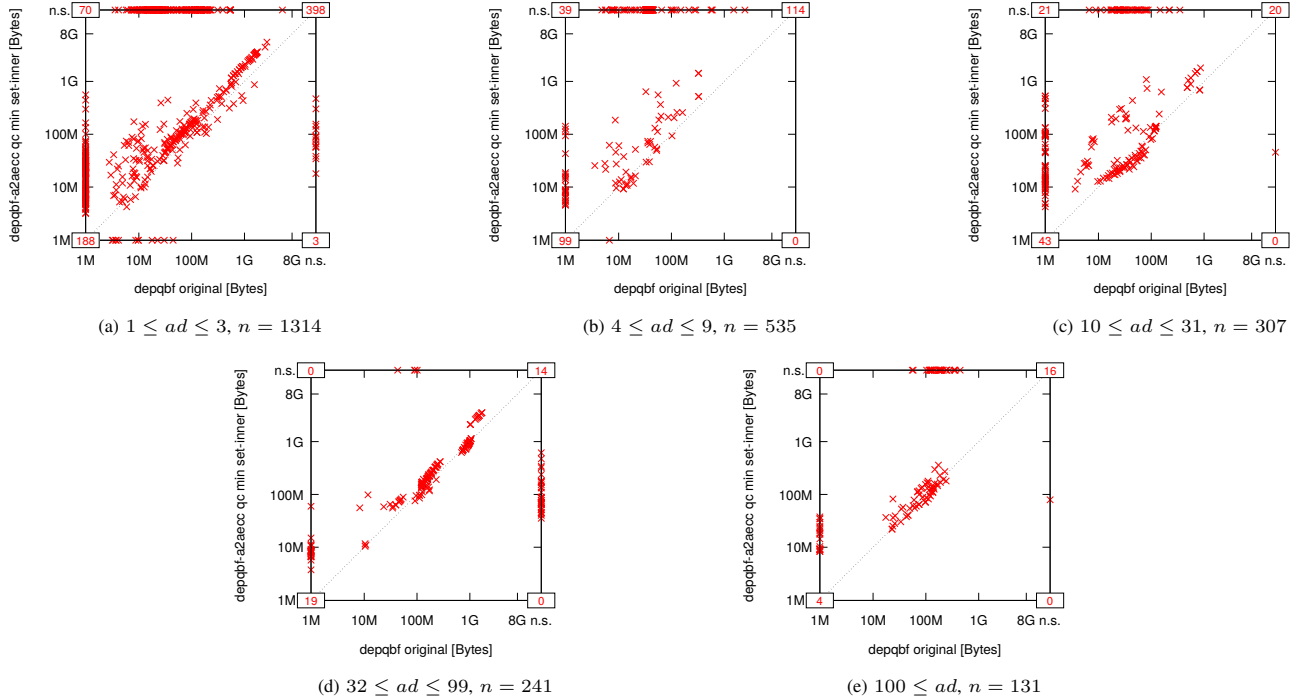


Fig. 1280: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode original partitioned by alternation depth (memory usage in Bytes).

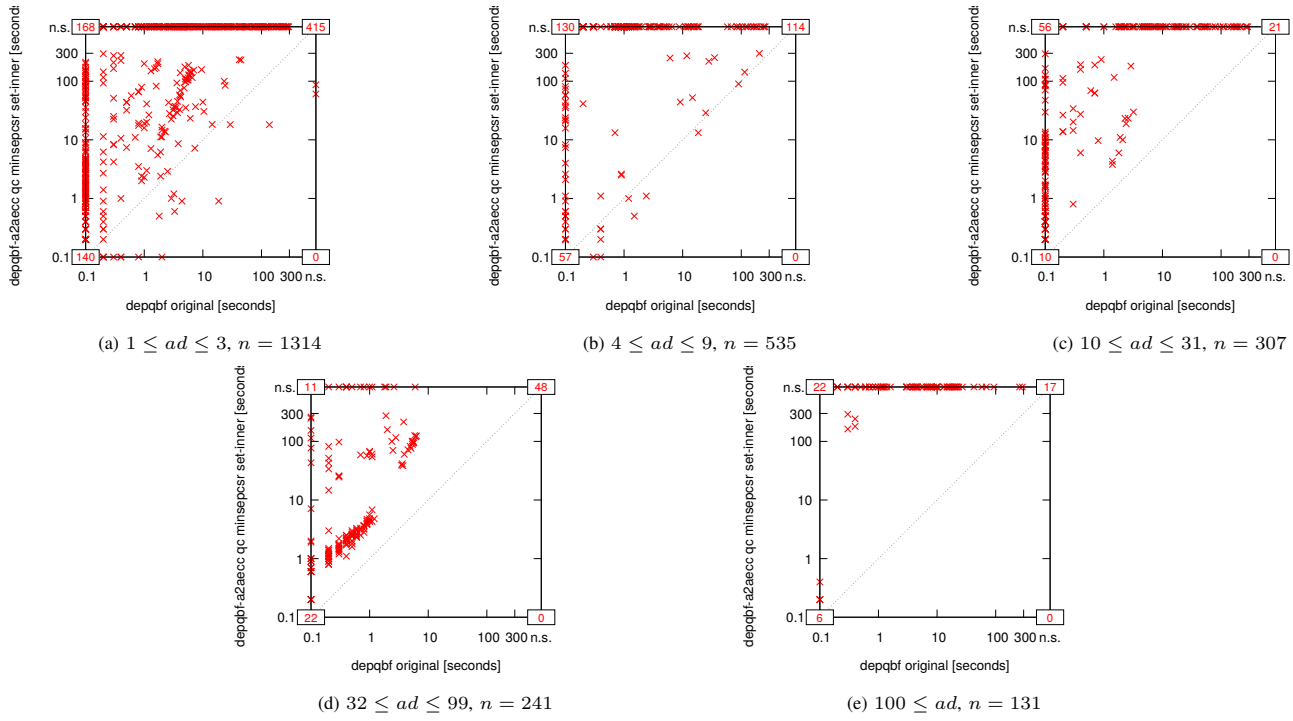


Fig. 1281: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode original partitioned by alternation depth (run time in seconds).

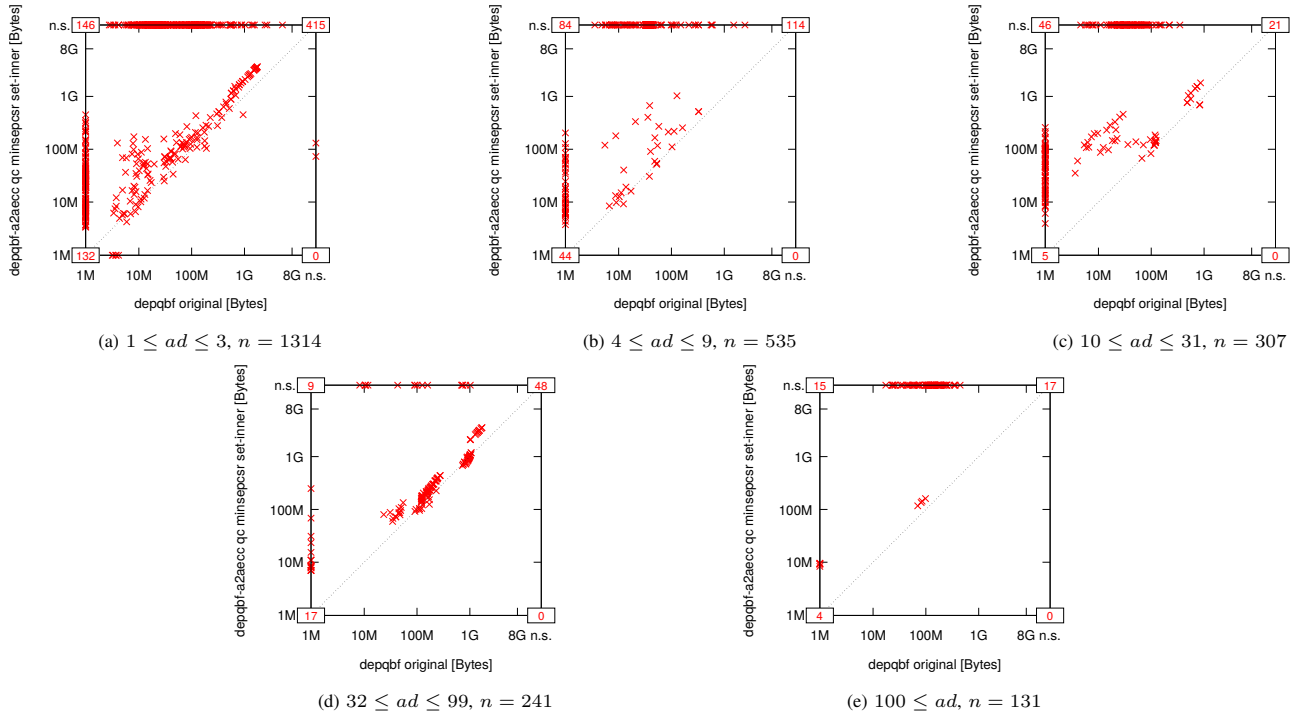


Fig. 1282: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode original partitioned by alternation depth (memory usage in Bytes).

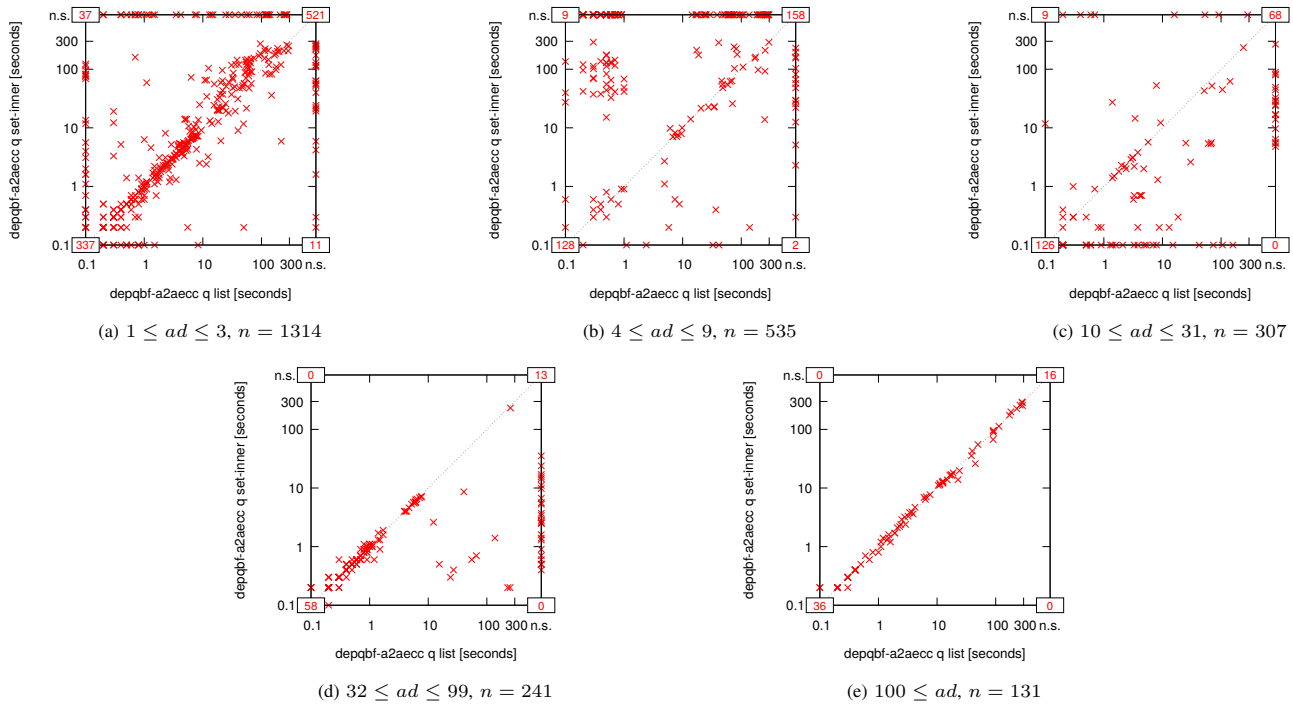


Fig. 1283: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q with set-inner versus list semantics partitioned by alternation depth (run time in seconds).

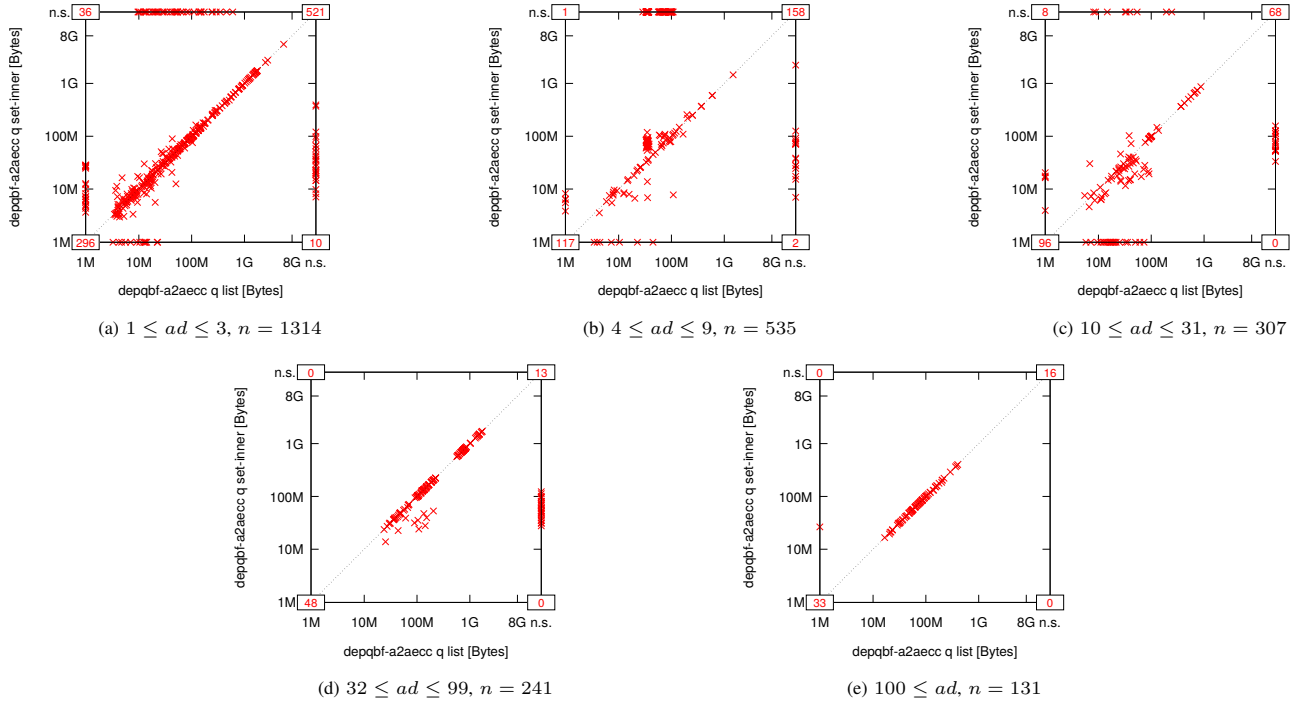


Fig. 1284: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q with set-inner versus list semantics partitioned by alternation depth (memory usage in Bytes).

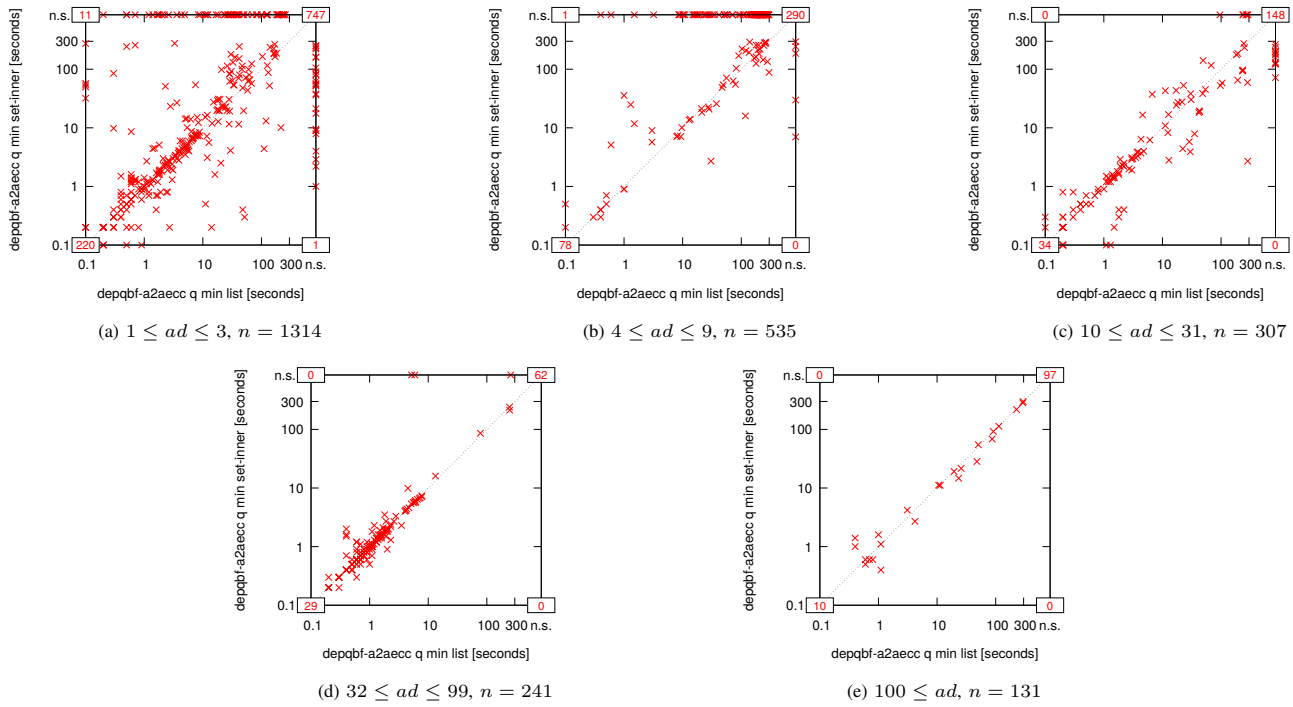


Fig. 1285: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q min with set-inner versus list semantics partitioned by alternation depth (run time in seconds).

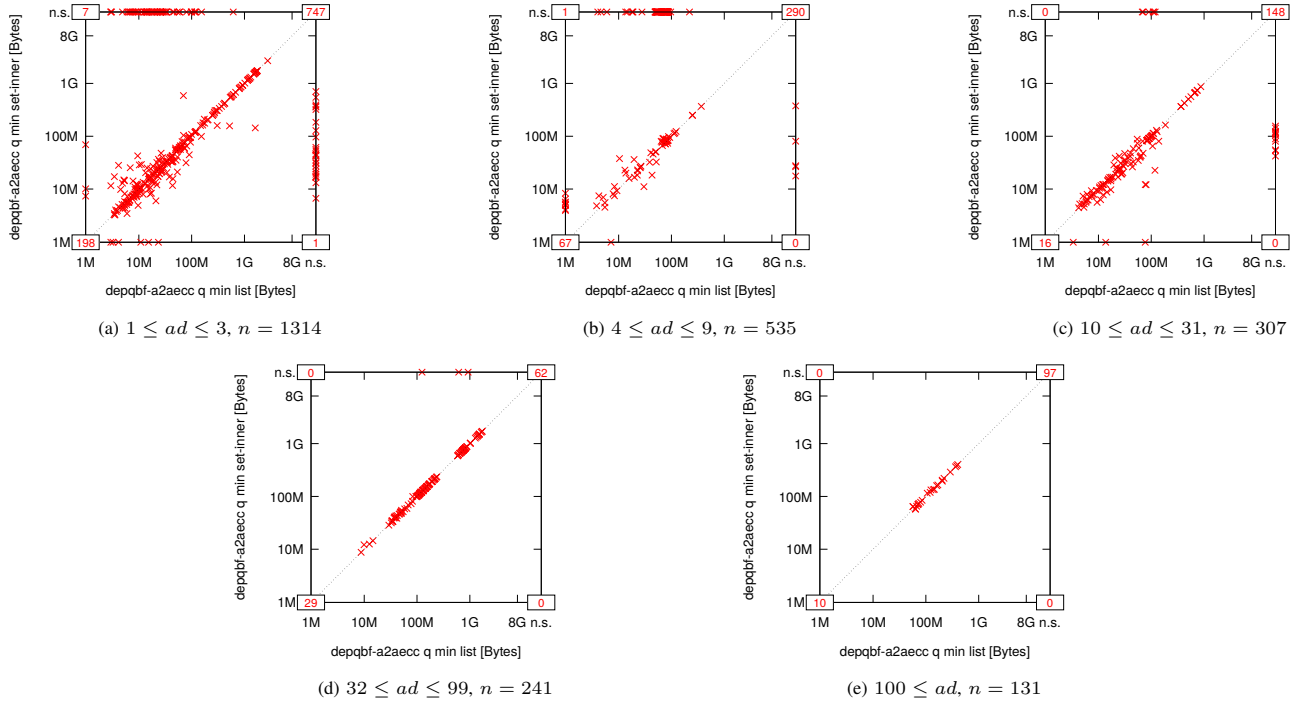


Fig. 1286: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q min with set-inner versus list semantics partitioned by alternation depth (memory usage in Bytes).

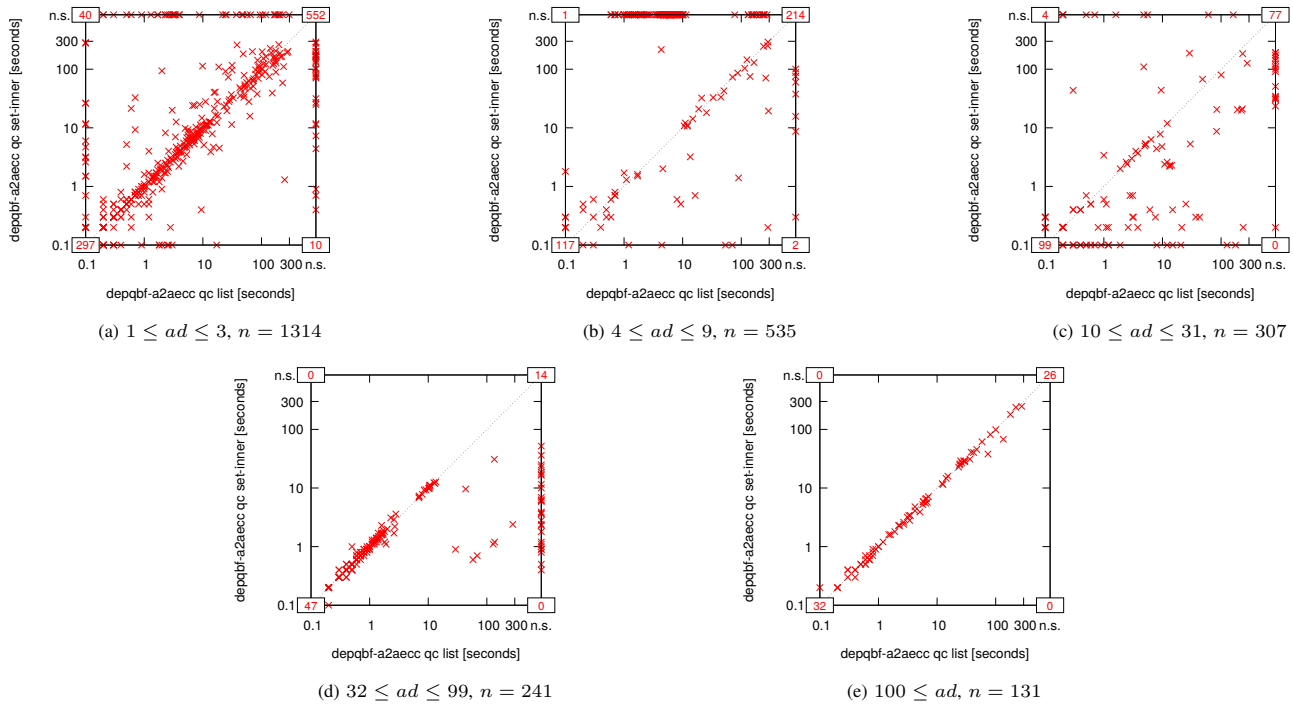


Fig. 1287: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc with set-inner versus list semantics partitioned by alternation depth (run time in seconds).

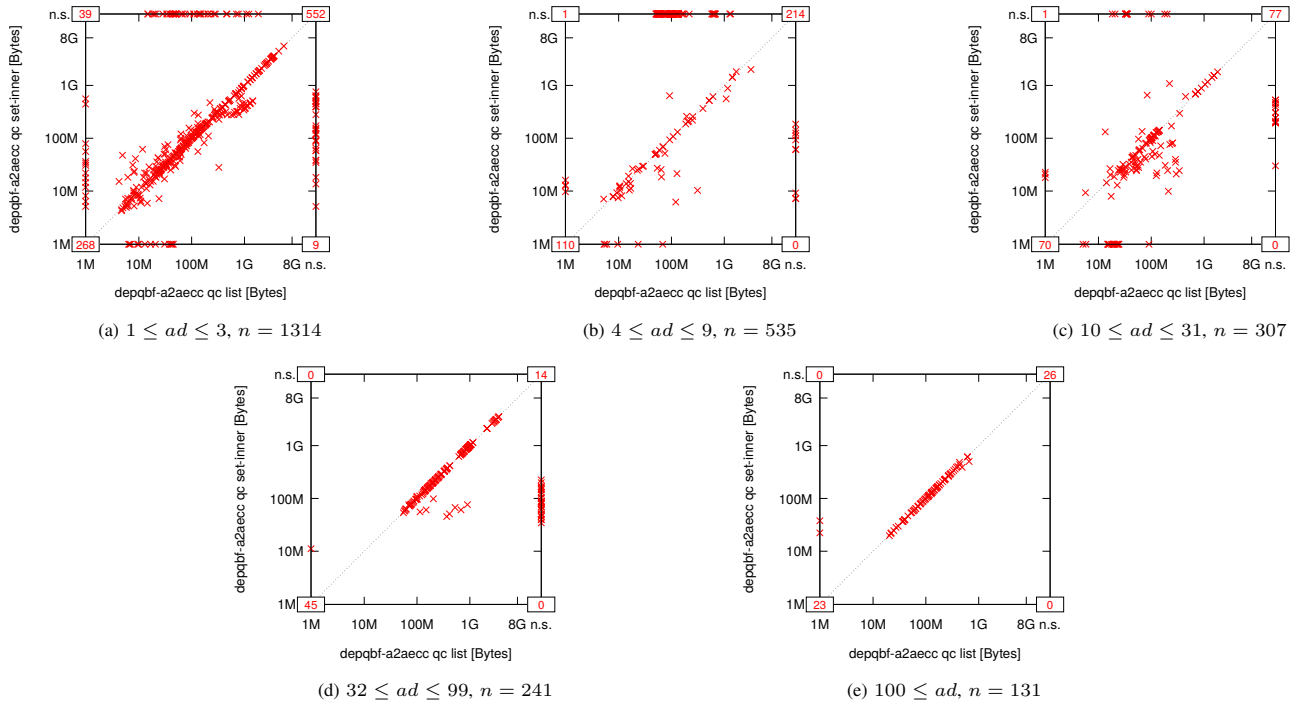


Fig. 1288: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc with set-inner versus list semantics partitioned by alternation depth (memory usage in Bytes).

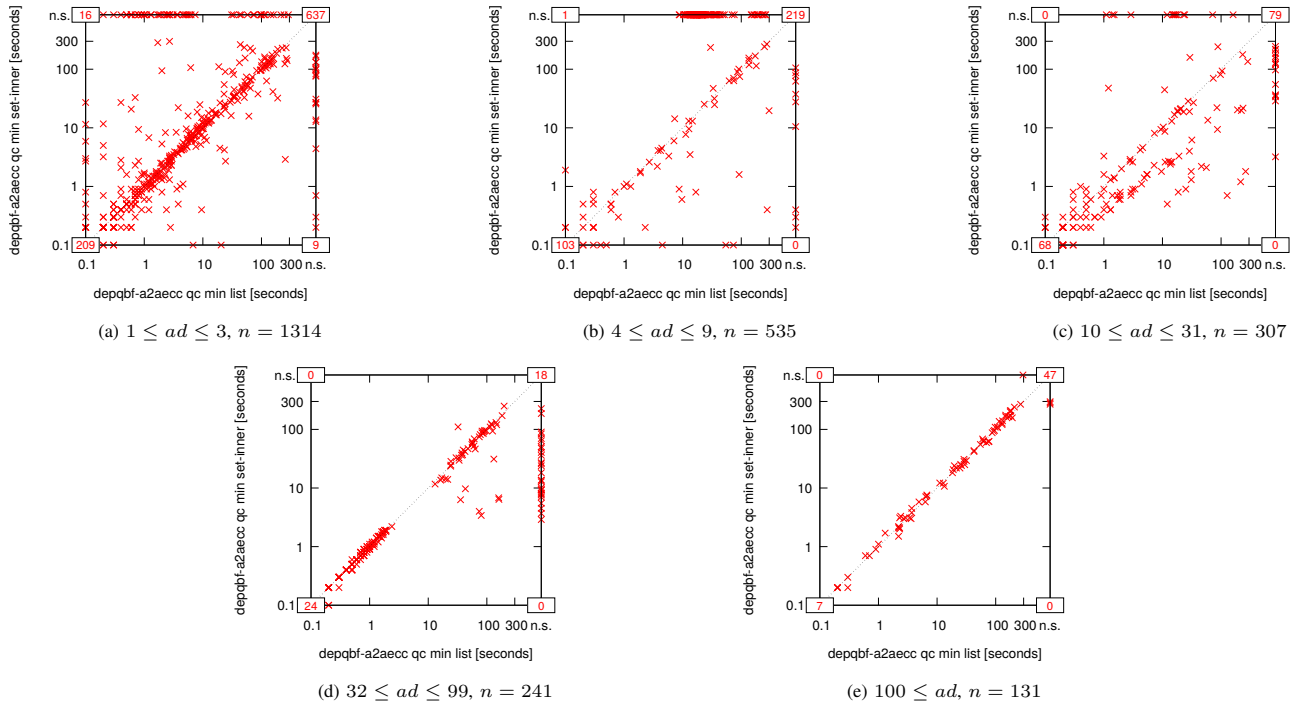


Fig. 1289: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min with set-inner versus list semantics partitioned by alternation depth (run time in seconds).

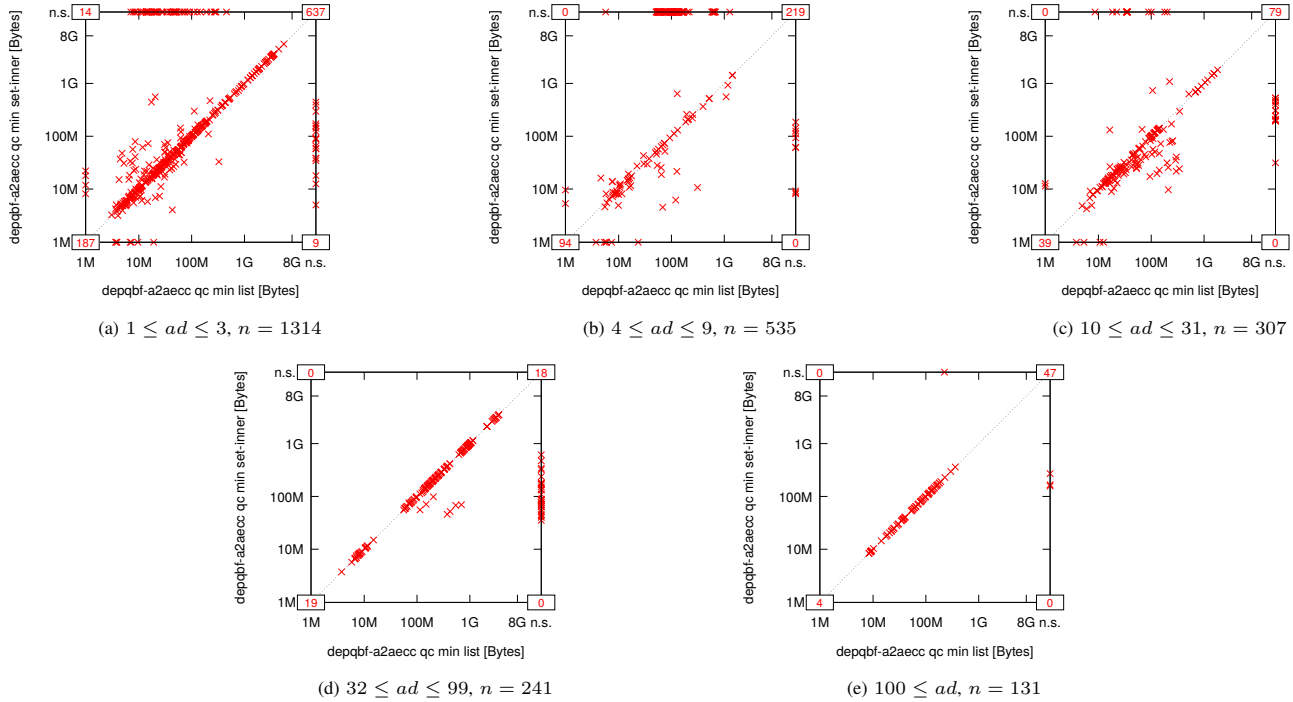


Fig. 1290: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min with set-inner versus list semantics partitioned by alternation depth (memory usage in Bytes).

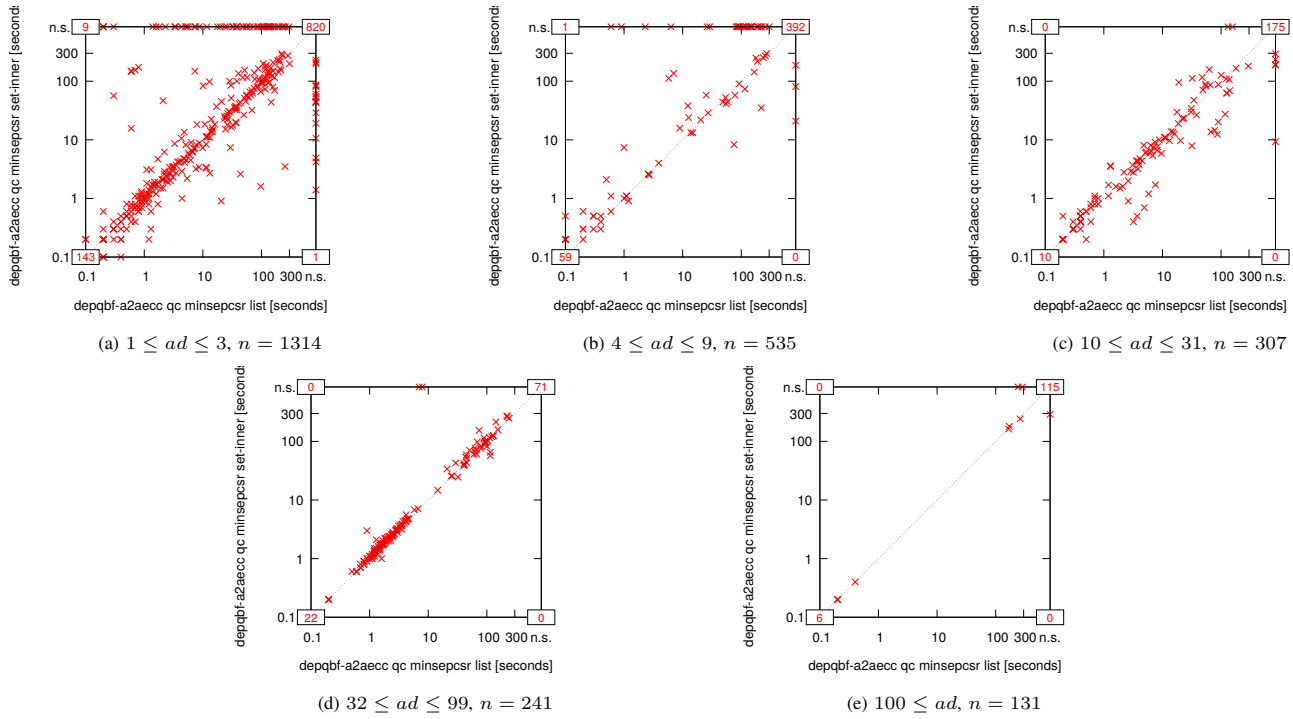


Fig. 1291: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr with set-inner versus list semantics partitioned by alternation depth (run time in seconds).

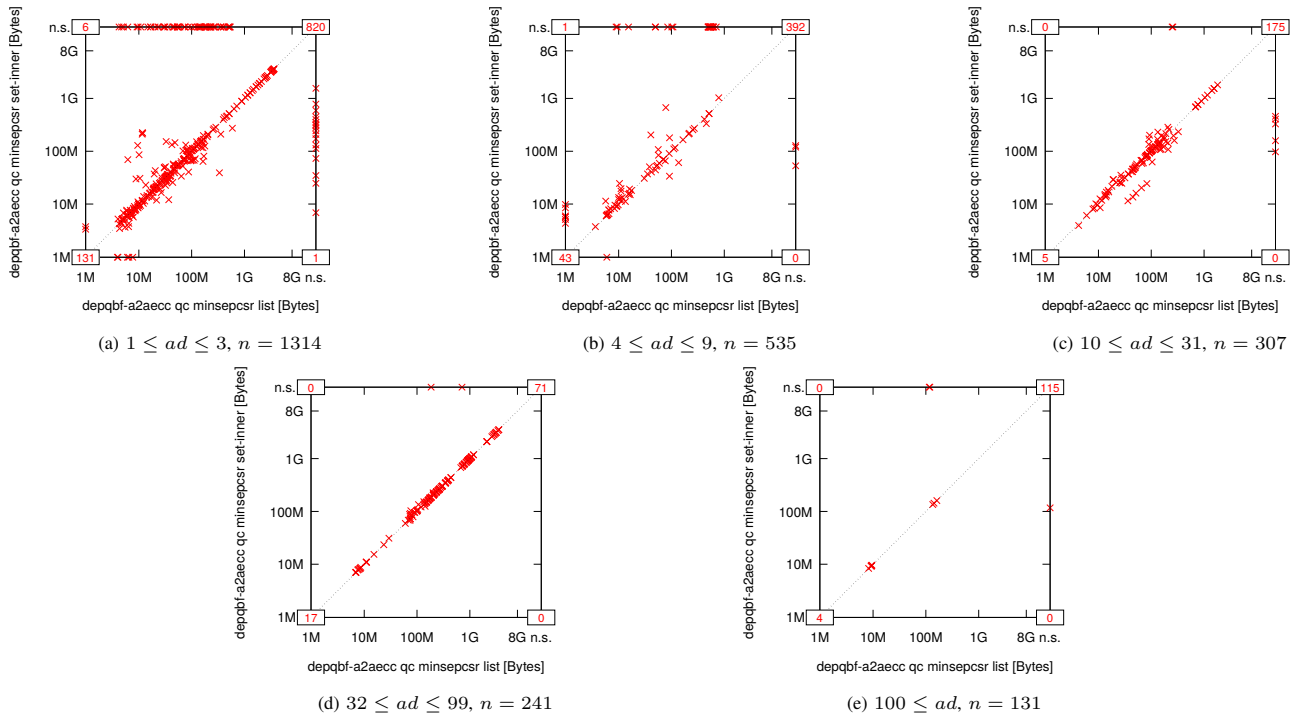


Fig. 1292: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr with set-inner versus list semantics partitioned by alternation depth (memory usage in Bytes).

D. Partitioned by Number of Clauses

This subsection shows figures of the overhead of extracting and minimizing unsatisfiable cores with subfigures for partitions of the benchmarks according to their number of clauses.

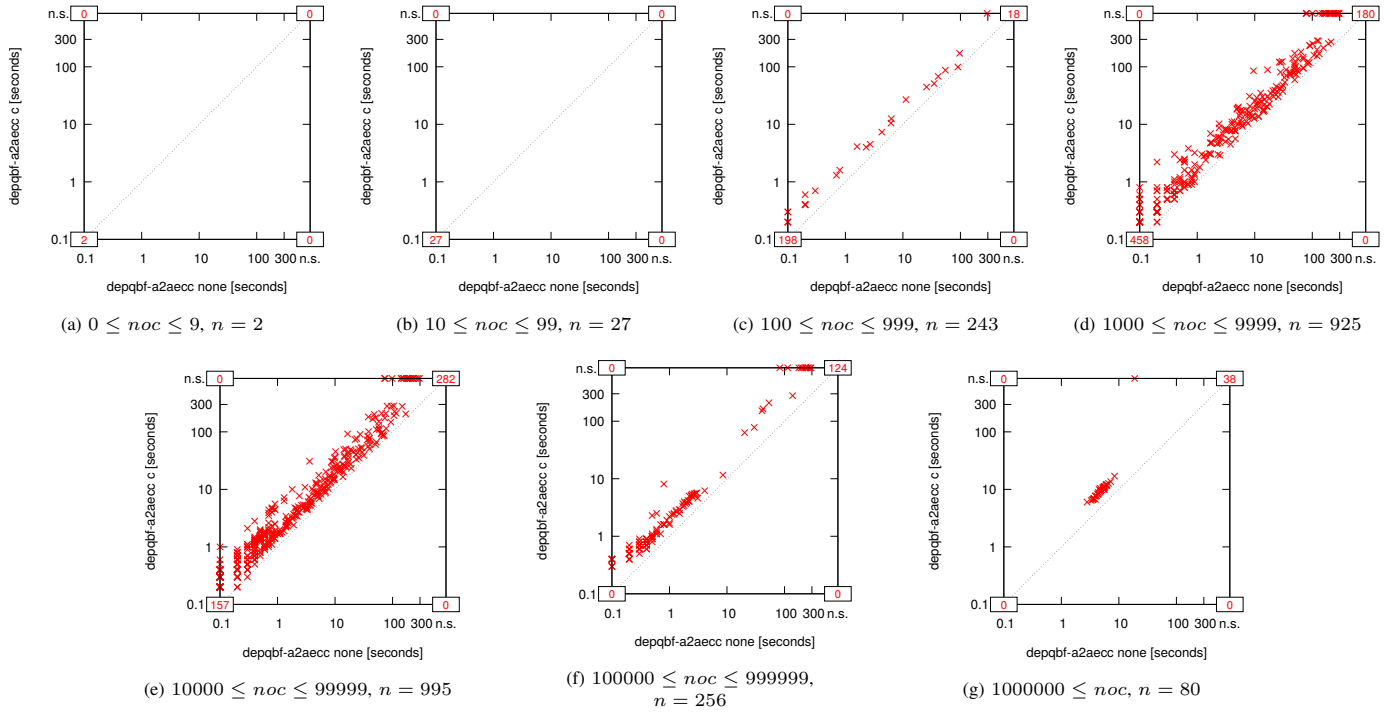


Fig. 1293: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c versus mode none partitioned by number of clauses (run time in seconds).

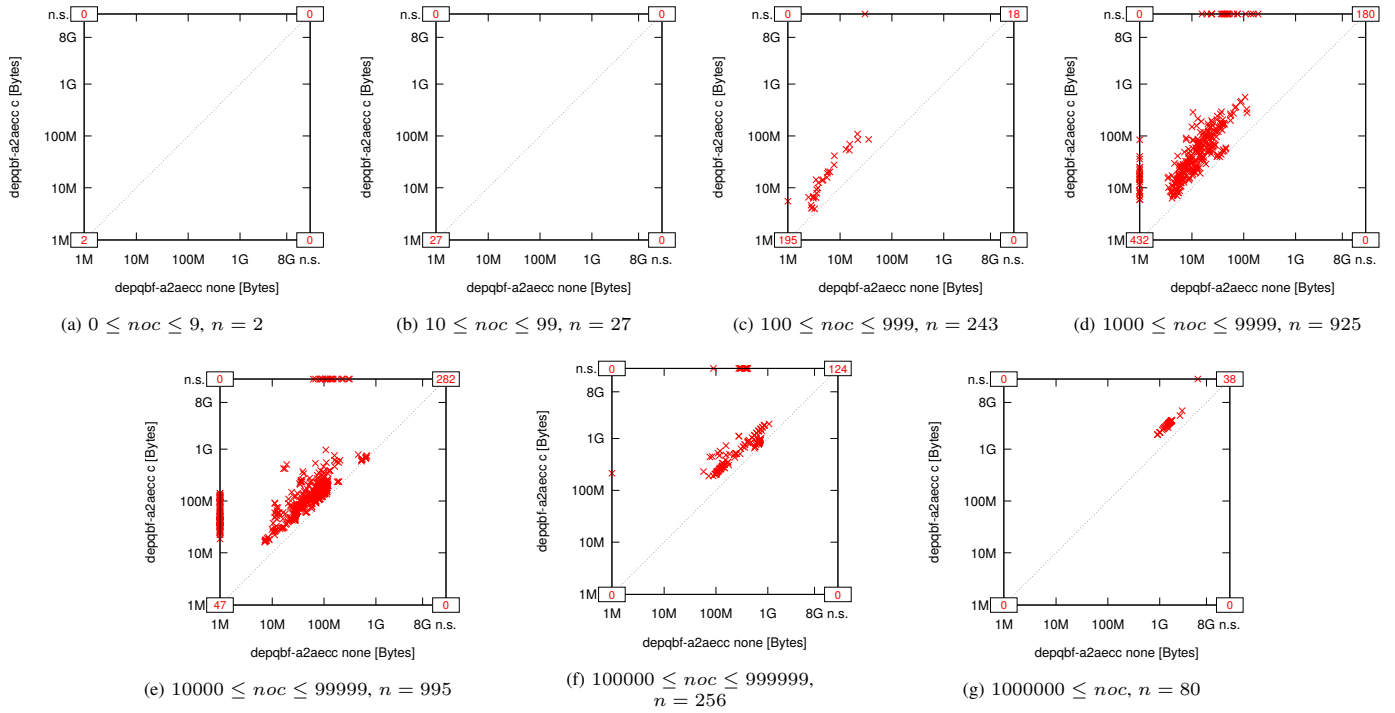


Fig. 1294: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c versus mode none partitioned by number of clauses (memory usage in Bytes).

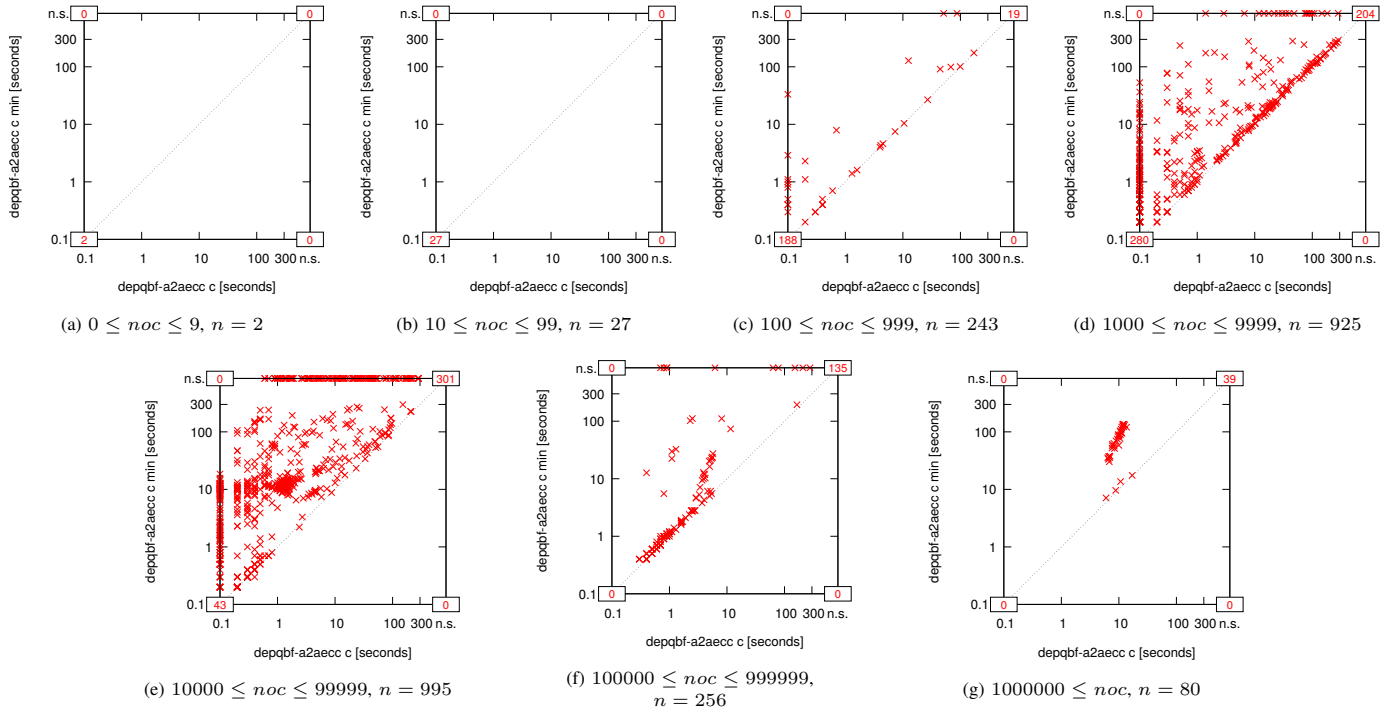


Fig. 1295: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode c partitioned by number of clauses (run time in seconds).

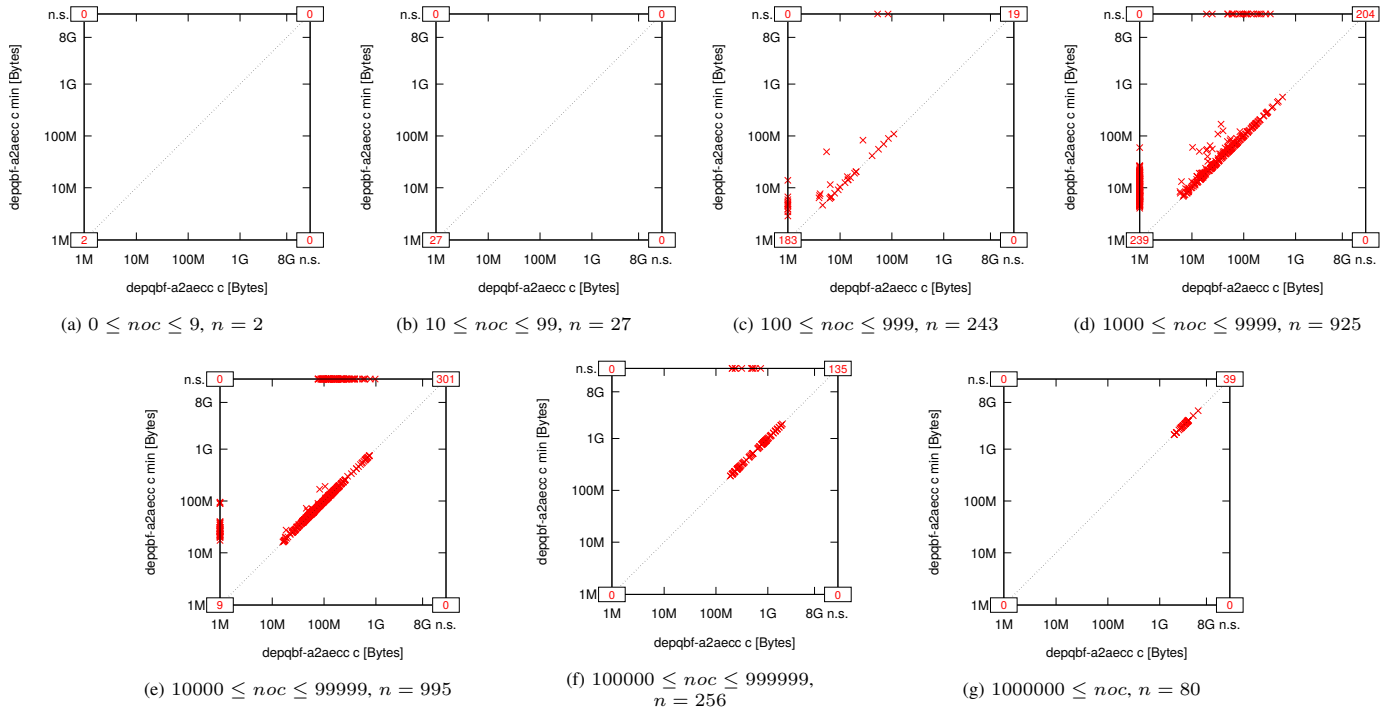


Fig. 1296: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode c partitioned by number of clauses (memory usage in Bytes).

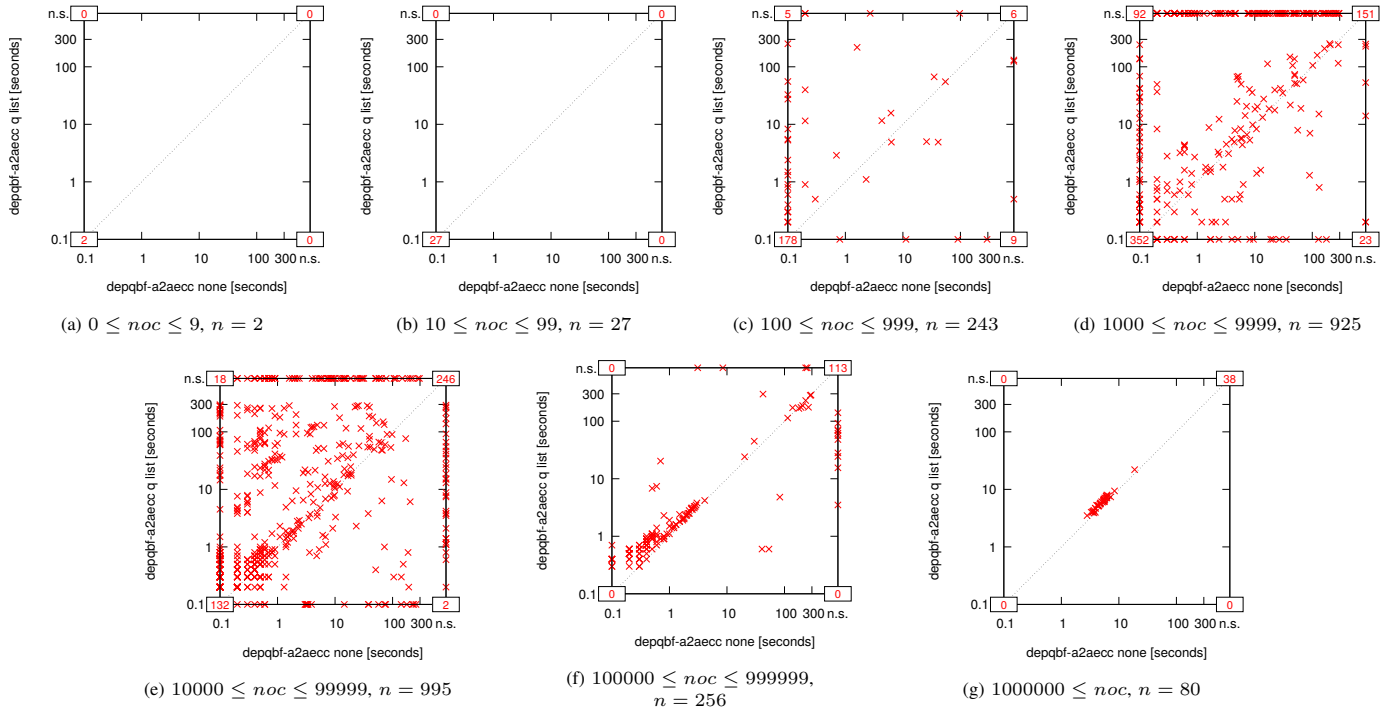


Fig. 1297: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode none partitioned by number of clauses (run time in seconds).

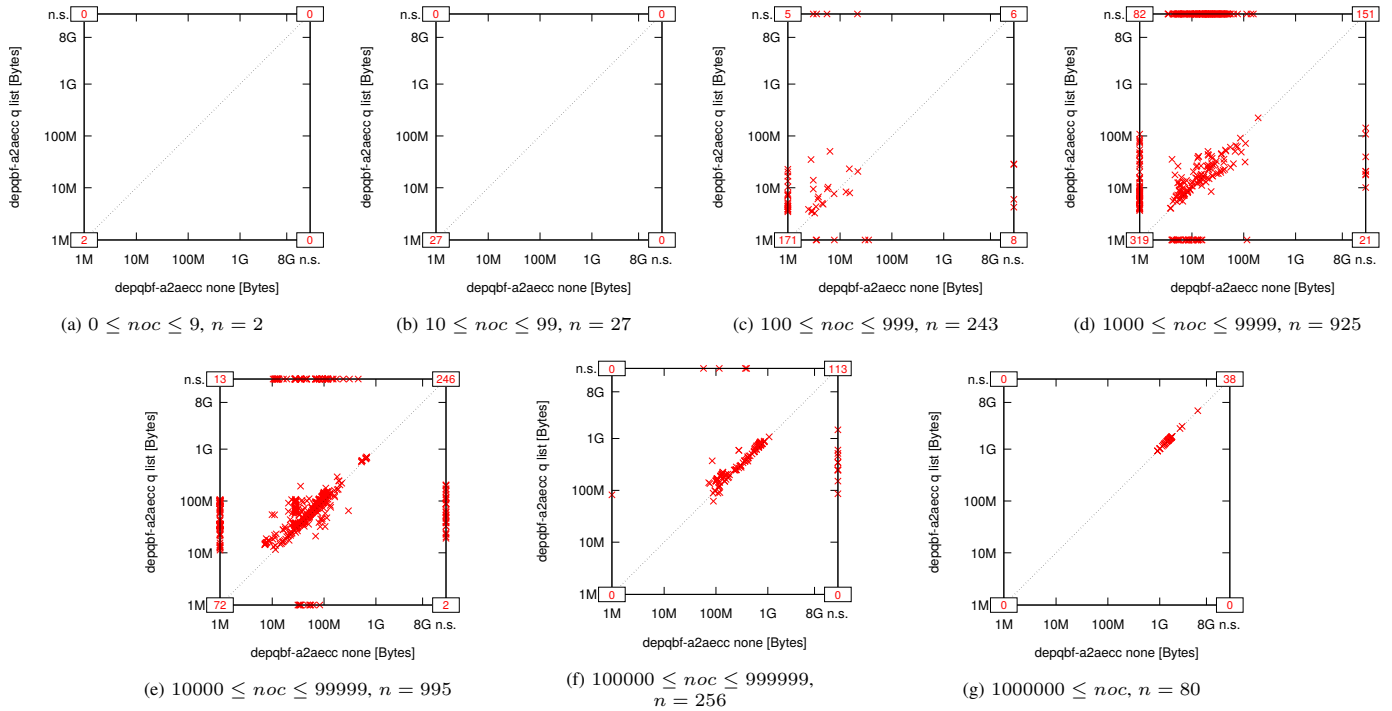


Fig. 1298: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode none partitioned by number of clauses (memory usage in Bytes).

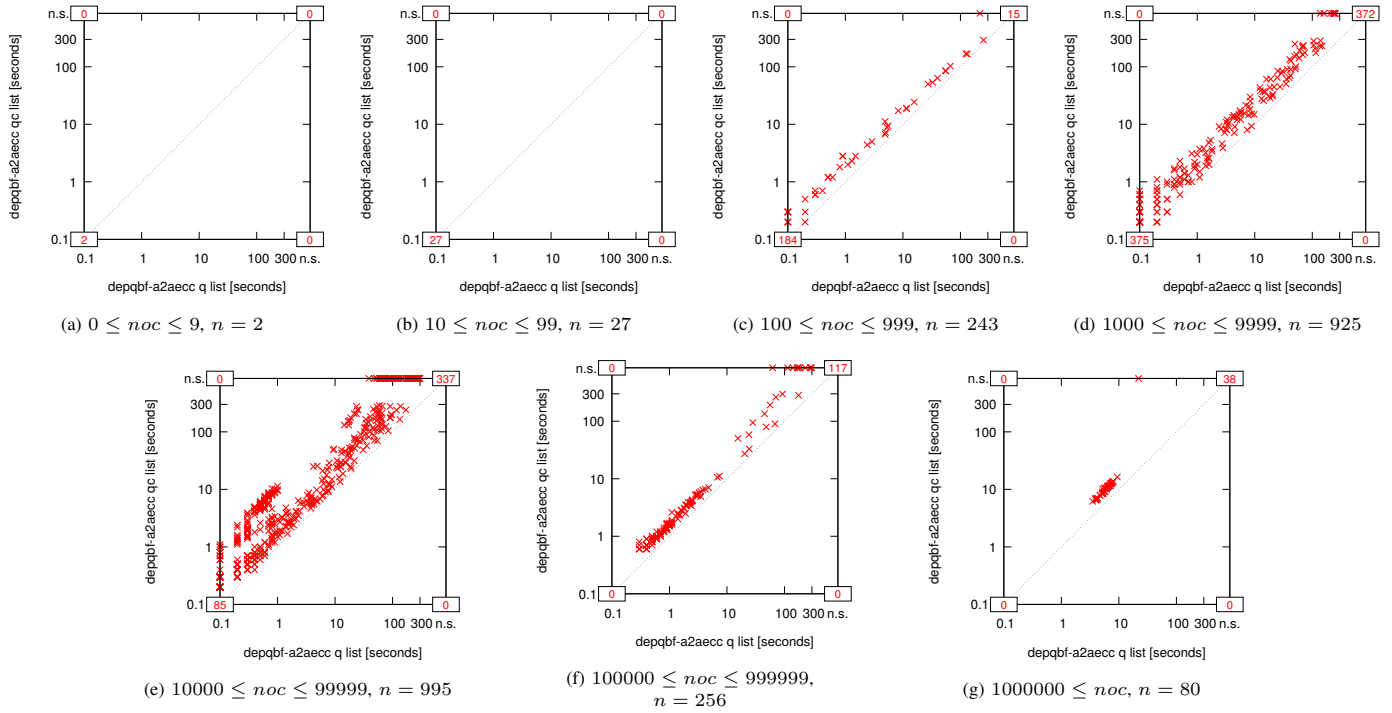


Fig. 1299: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode q list partitioned by number of clauses (run time in seconds).

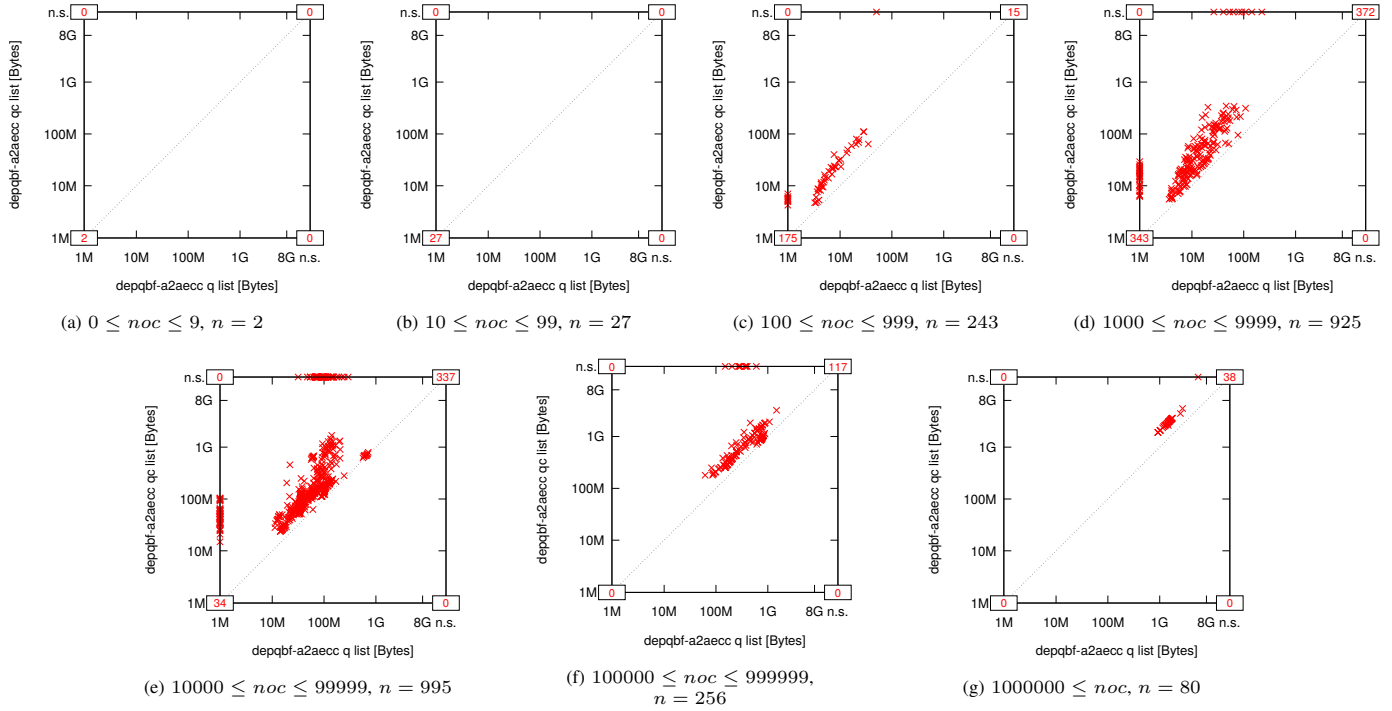


Fig. 1300: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode q list partitioned by number of clauses (memory usage in Bytes).

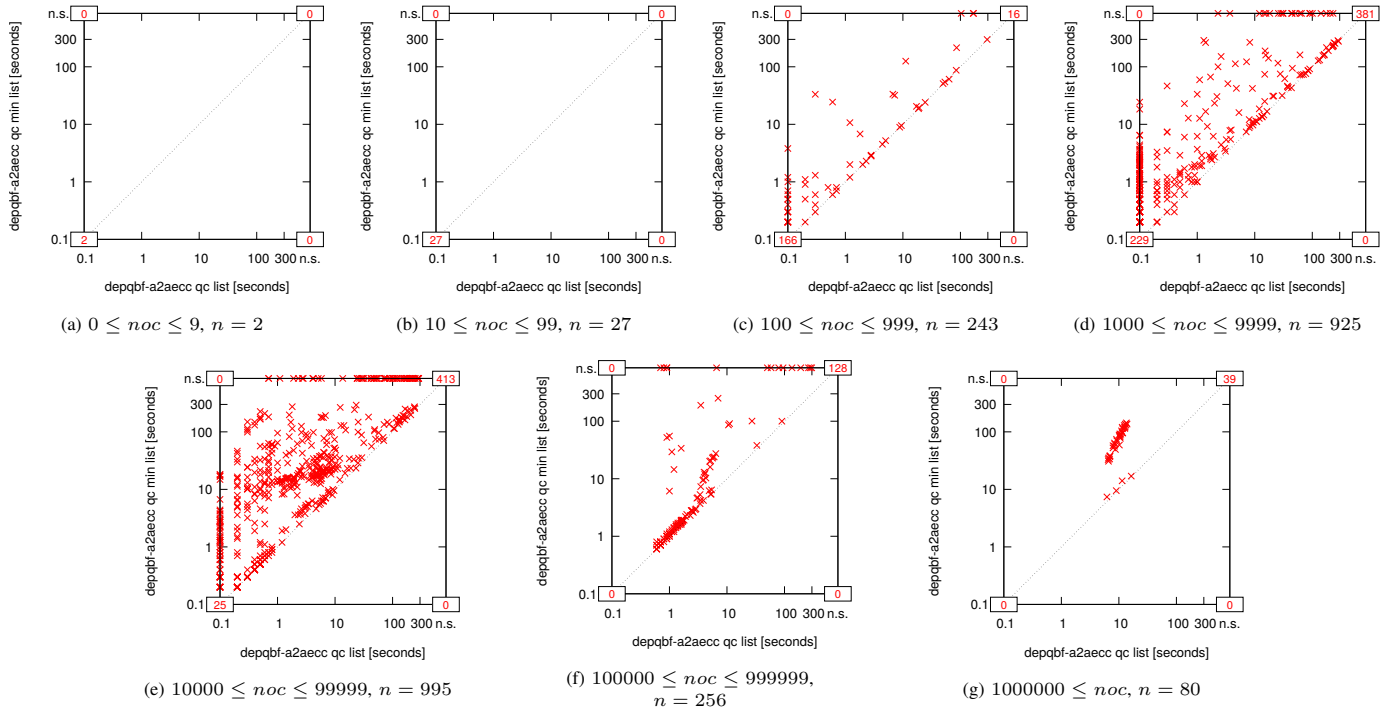


Fig. 1301: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode qc list partitioned by number of clauses (run time in seconds).

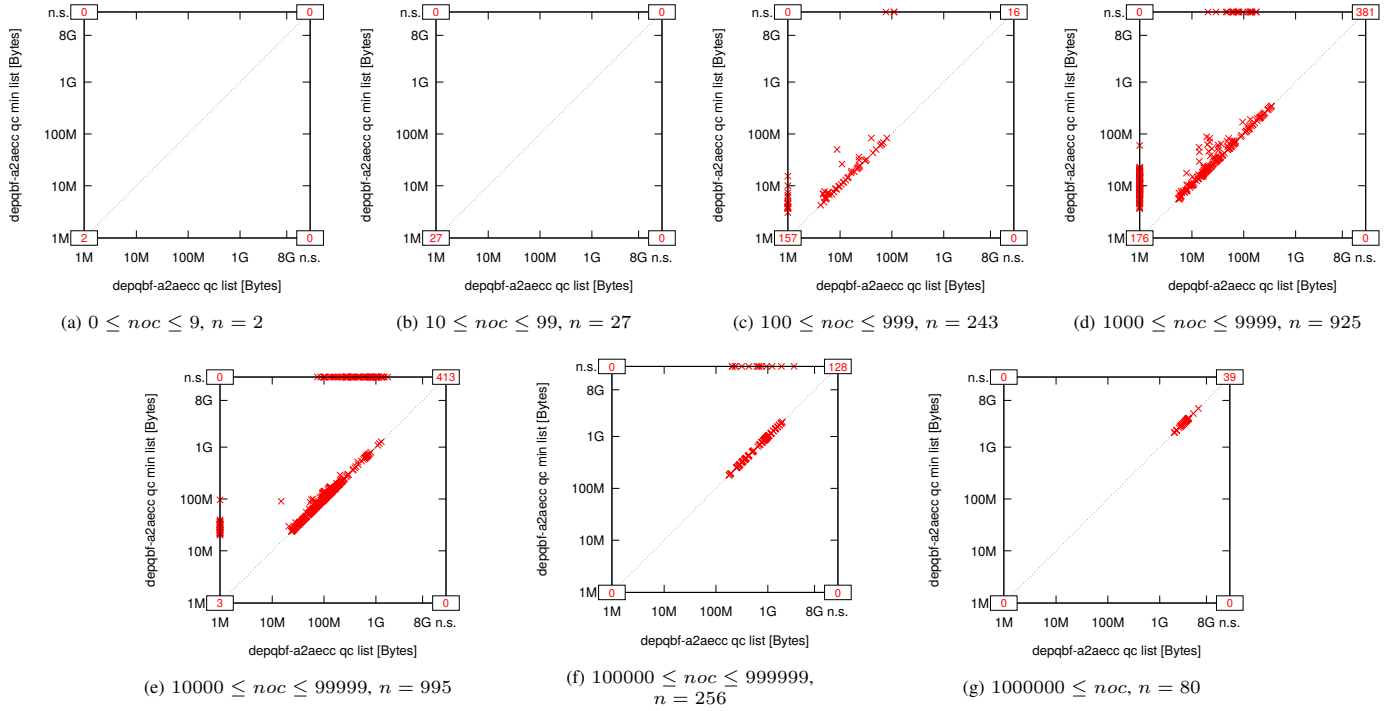


Fig. 1302: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode qc list partitioned by number of clauses (memory usage in Bytes).

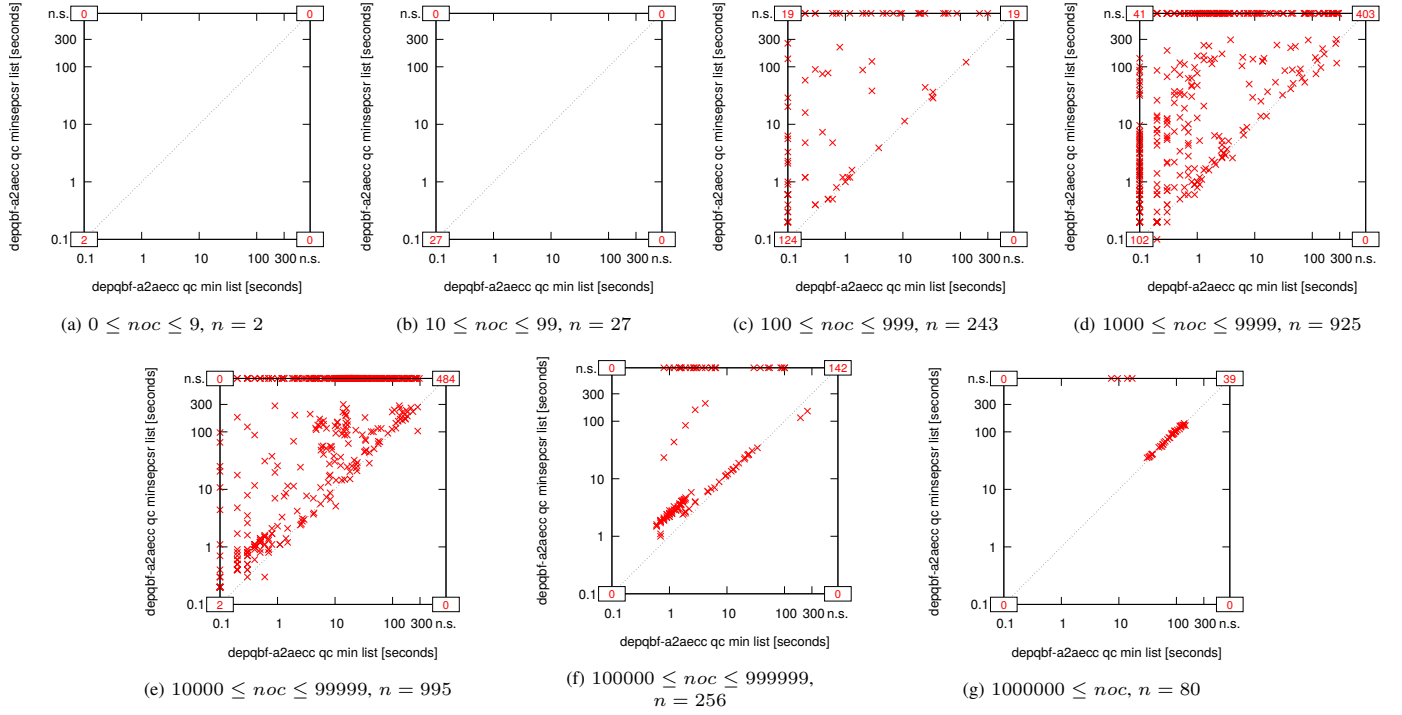


Fig. 1303: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode qc min list partitioned by number of clauses (run time in seconds).

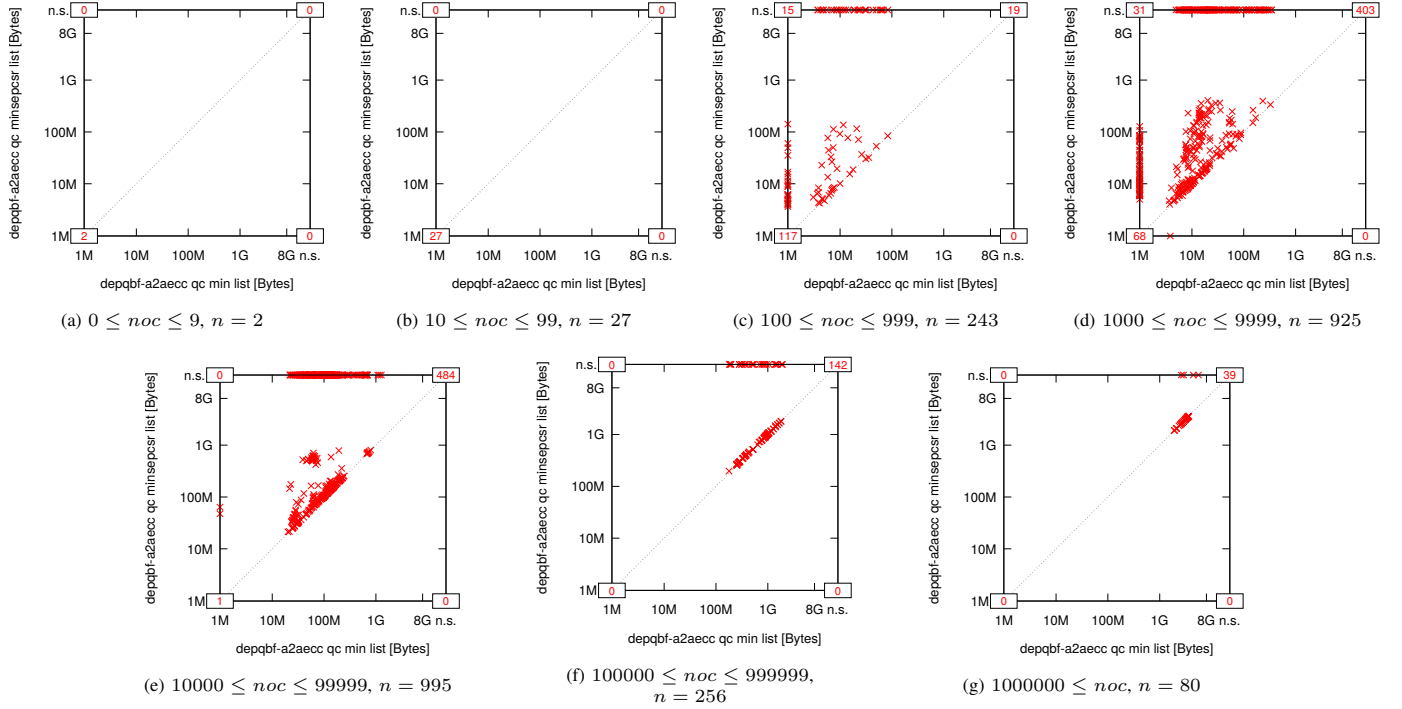


Fig. 1304: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode qc min list partitioned by number of clauses (memory usage in Bytes).

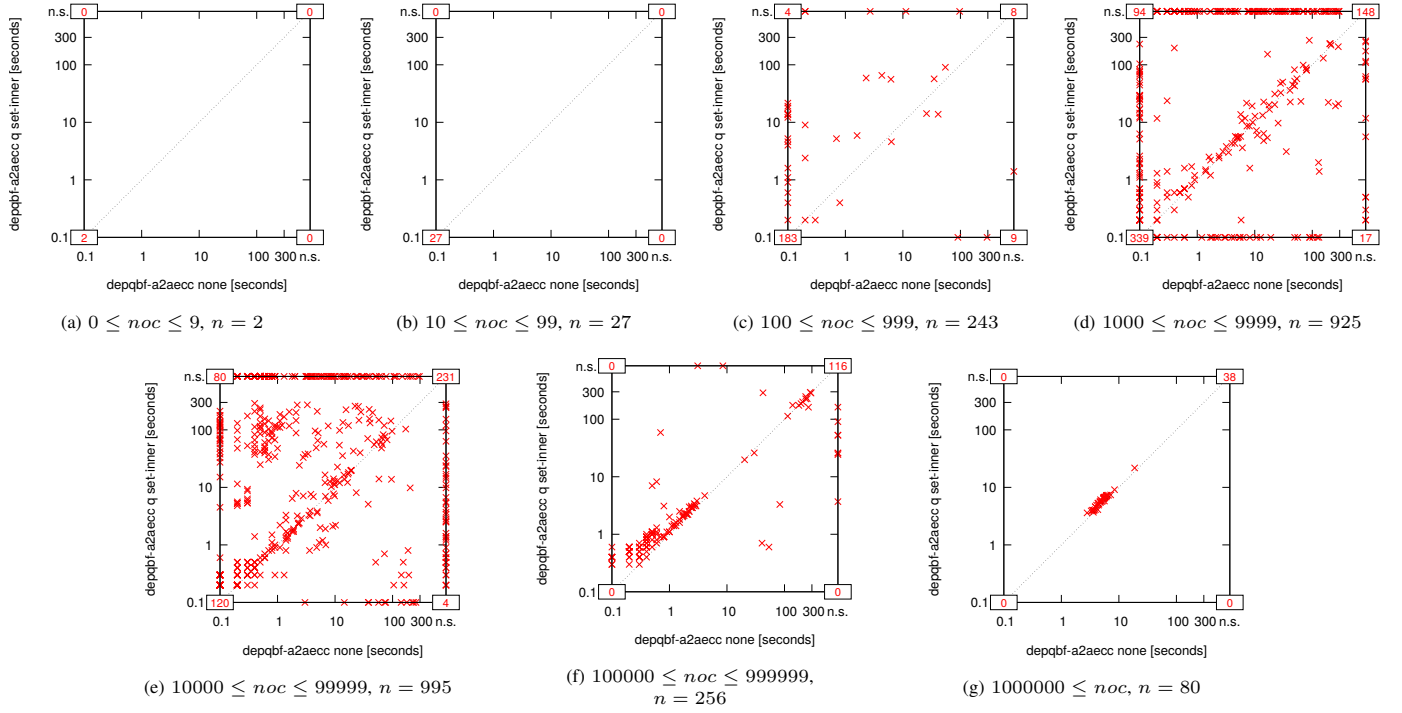


Fig. 1305: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode none partitioned by number of clauses (run time in seconds).

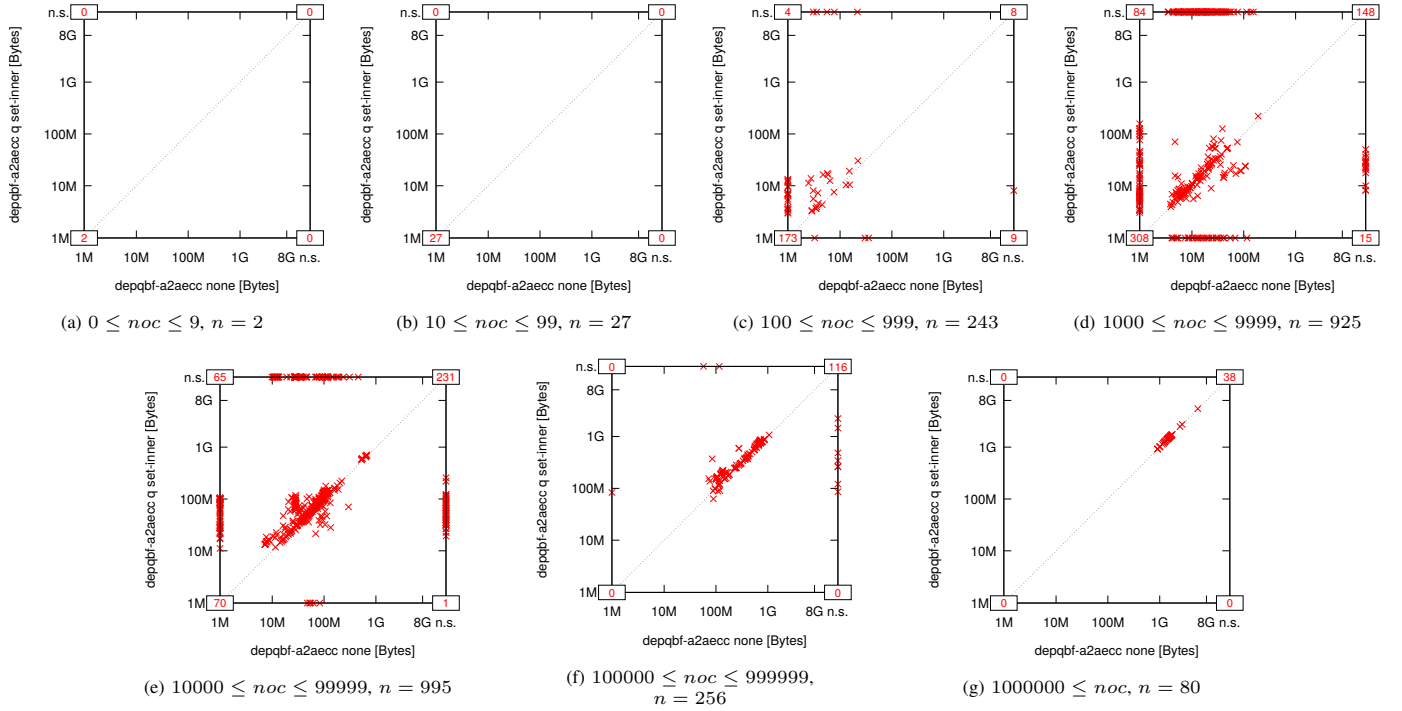


Fig. 1306: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode none partitioned by number of clauses (memory usage in Bytes).

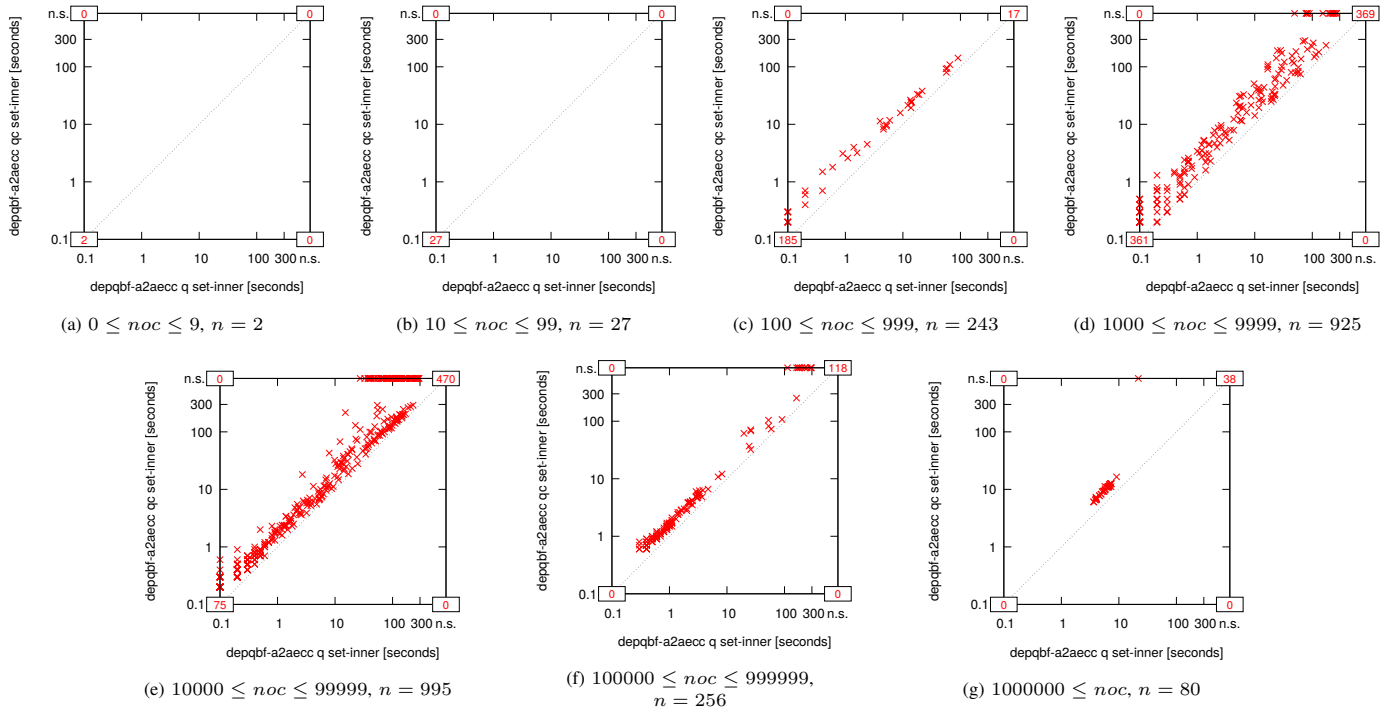


Fig. 1307: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode q set-inner partitioned by number of clauses (run time in seconds).

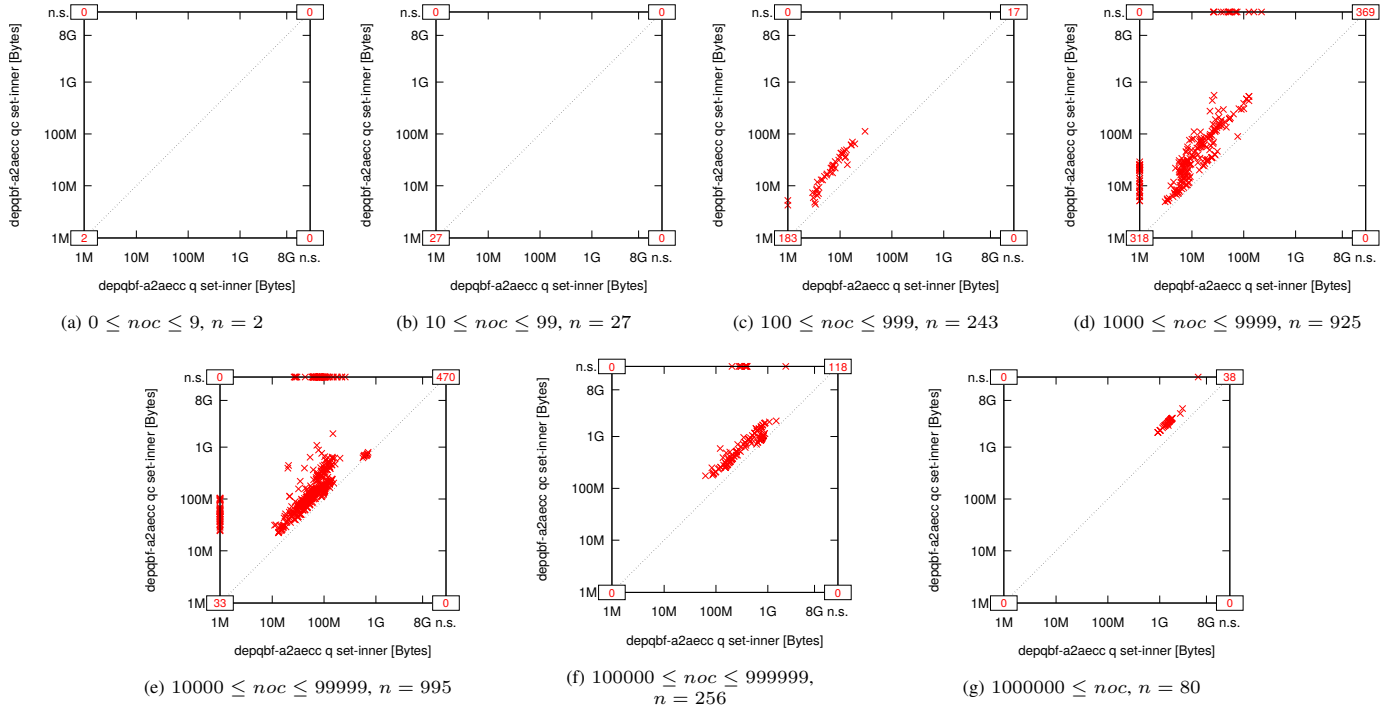


Fig. 1308: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode q set-inner partitioned by number of clauses (memory usage in Bytes).

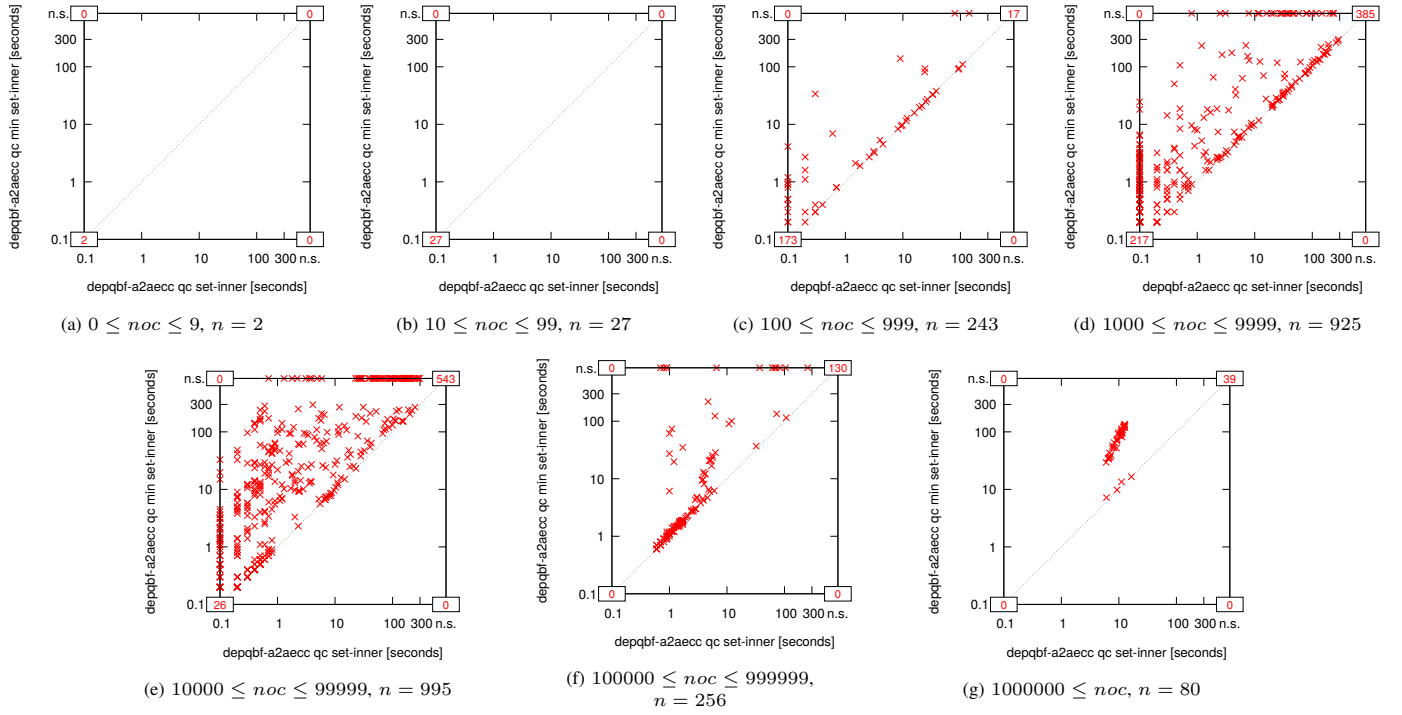


Fig. 1309: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode qc set-inner partitioned by number of clauses (run time in seconds).

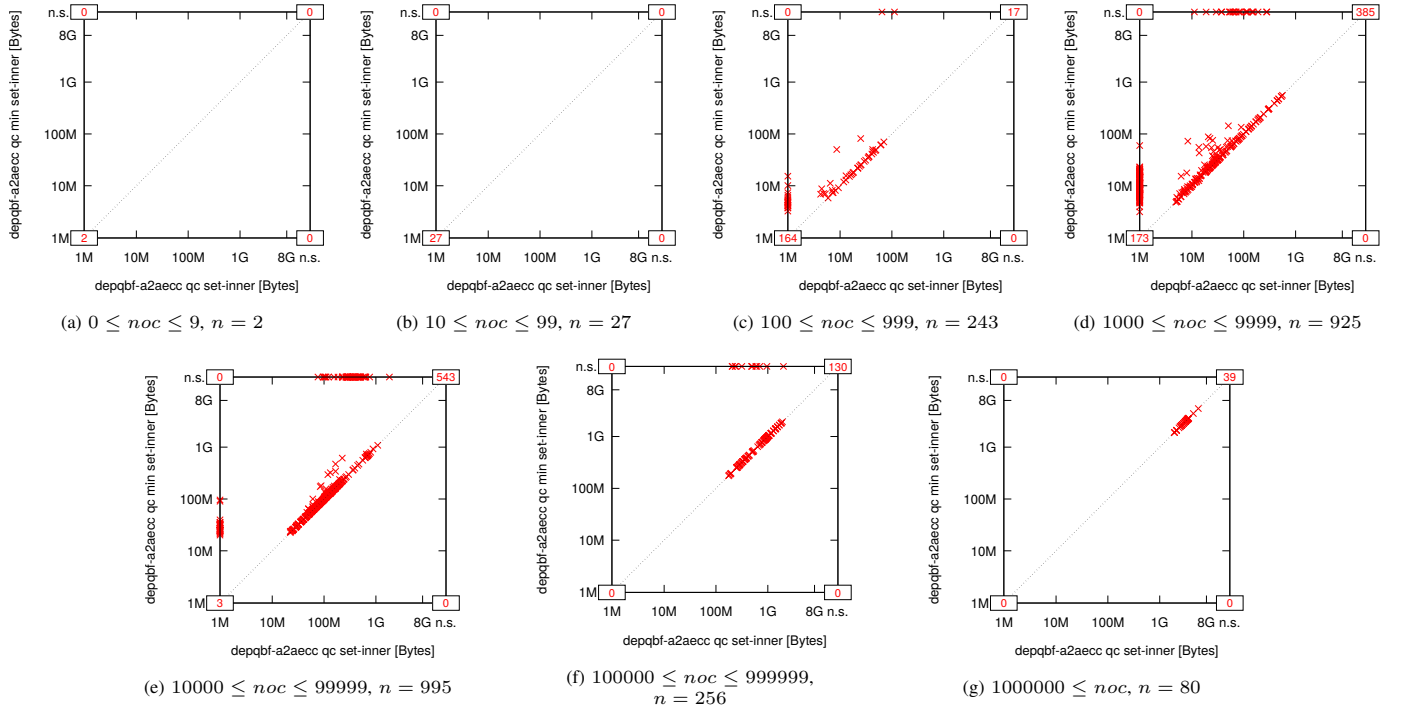


Fig. 1310: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode qc set-inner partitioned by number of clauses (memory usage in Bytes).

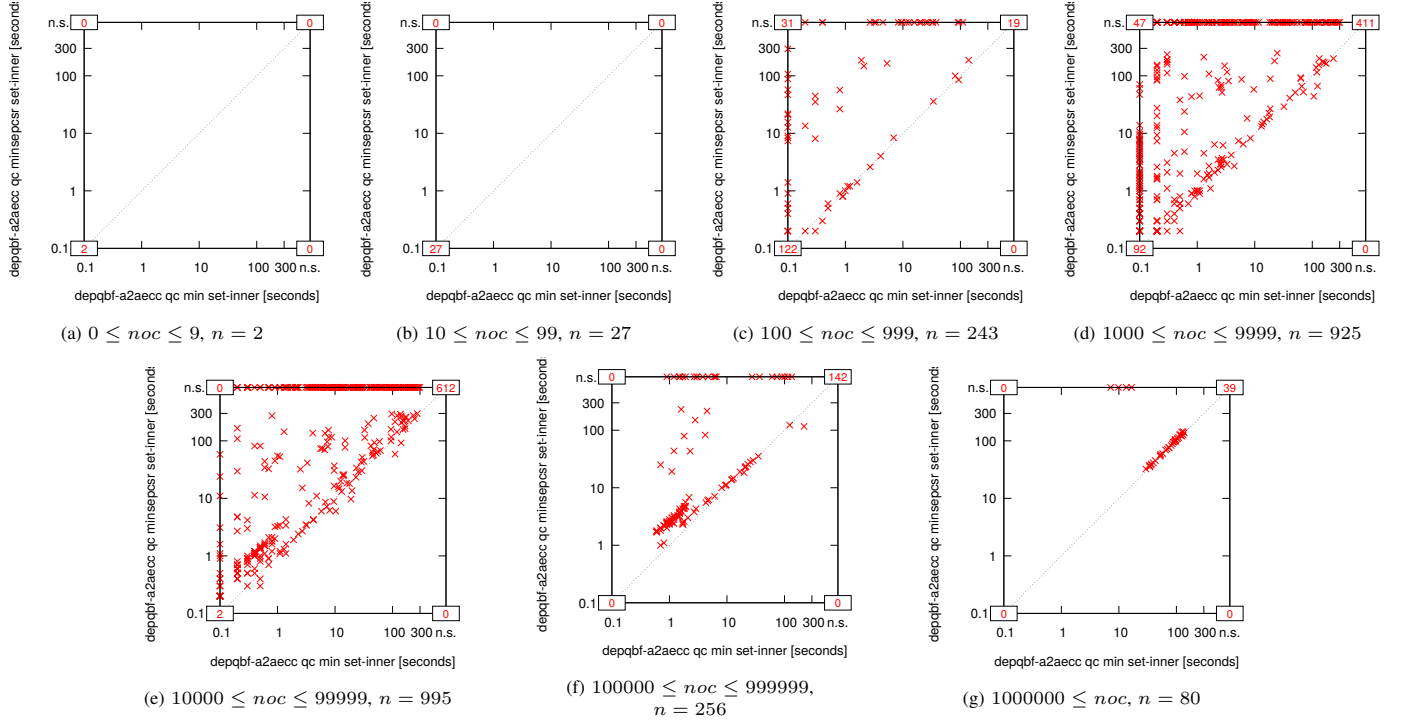


Fig. 1311: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode qc min set-inner partitioned by number of clauses (run time in seconds).

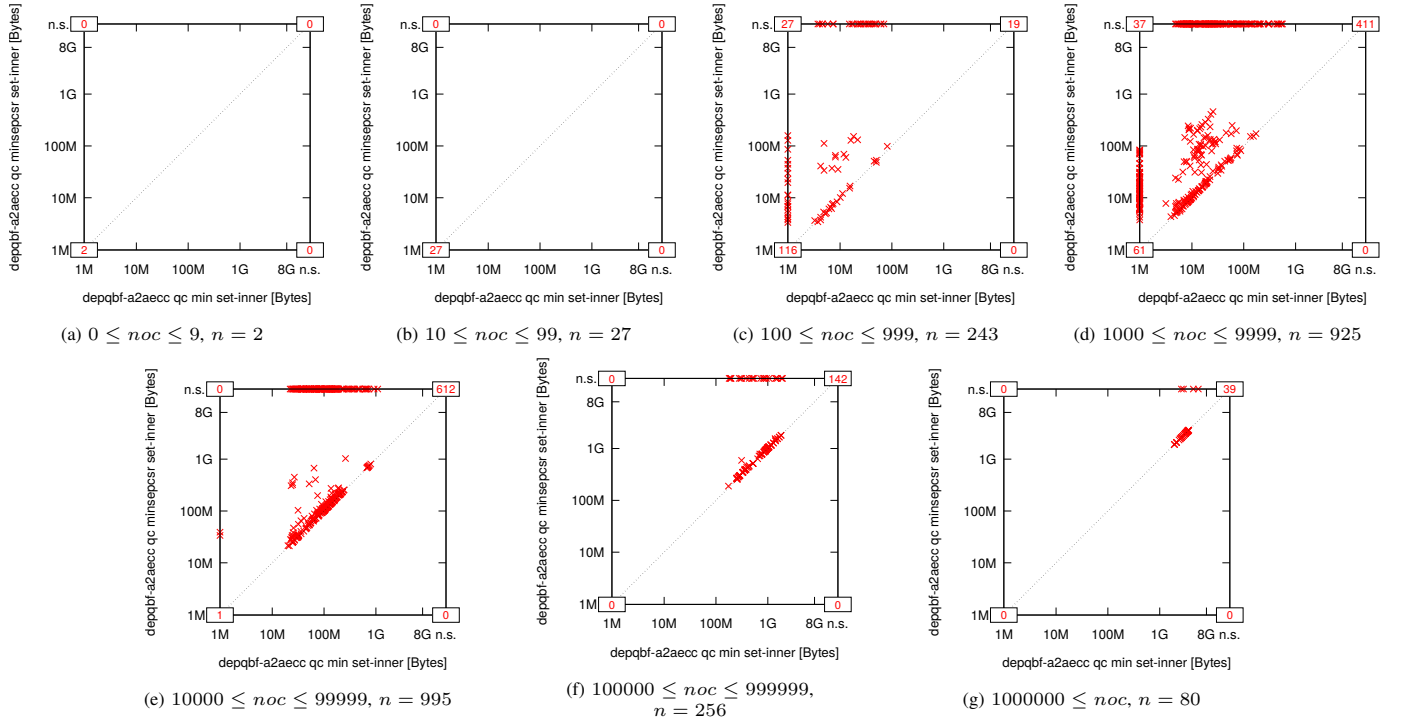


Fig. 1312: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode qc min set-inner partitioned by number of clauses (memory usage in Bytes).

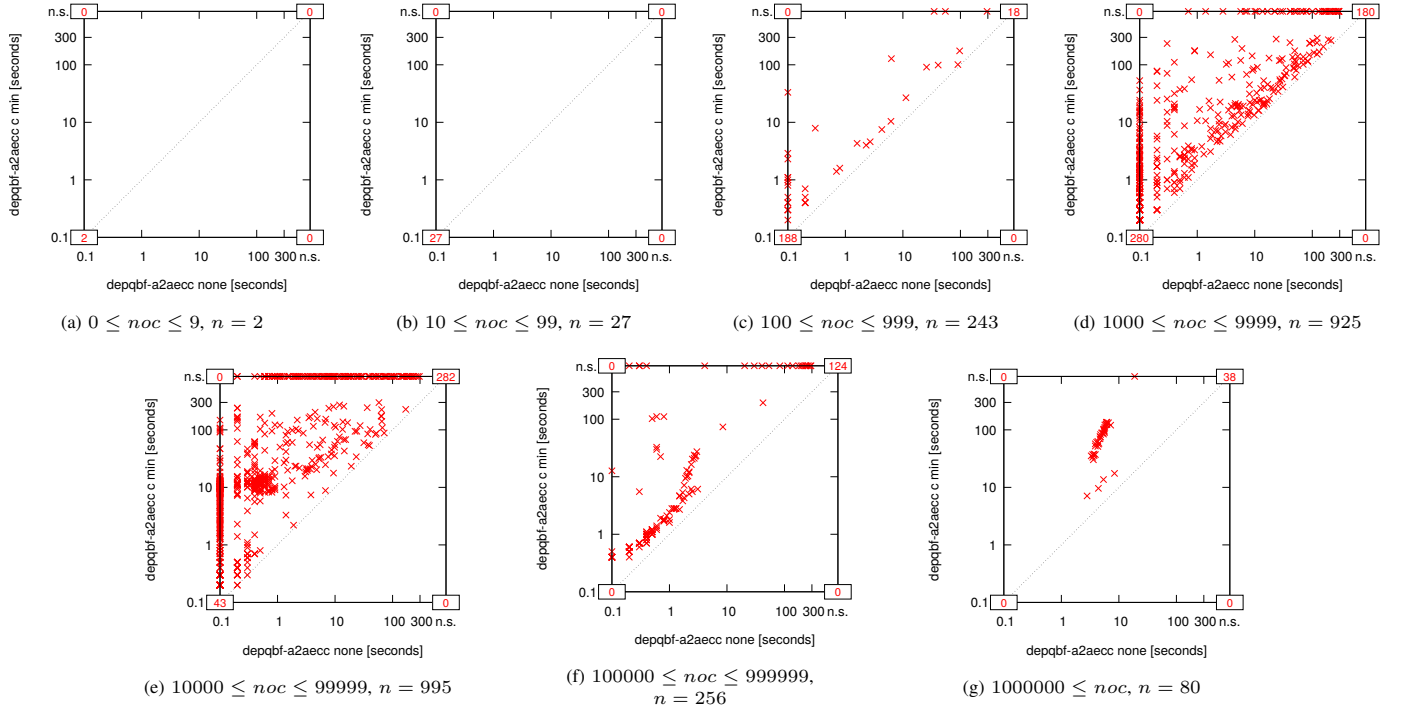


Fig. 1313: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode none partitioned by number of clauses (run time in seconds).

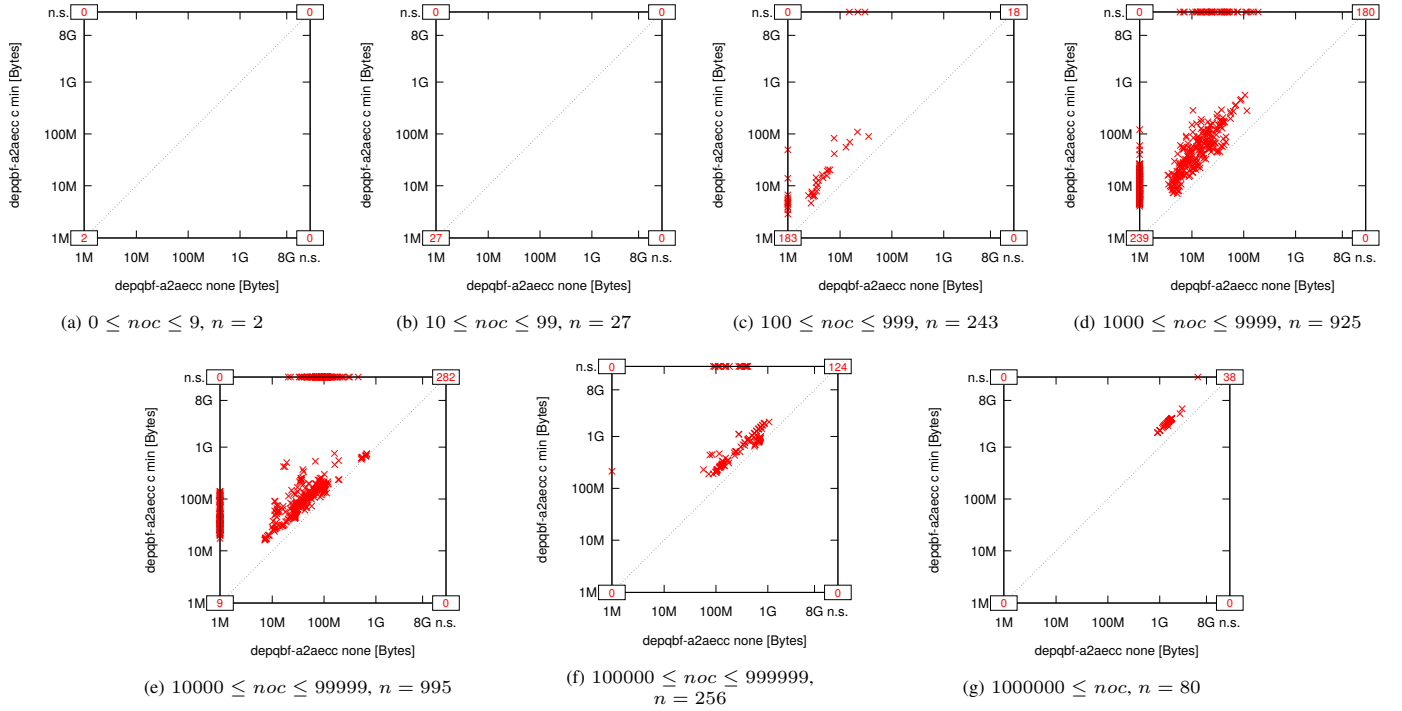


Fig. 1314: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode none partitioned by number of clauses (memory usage in Bytes).

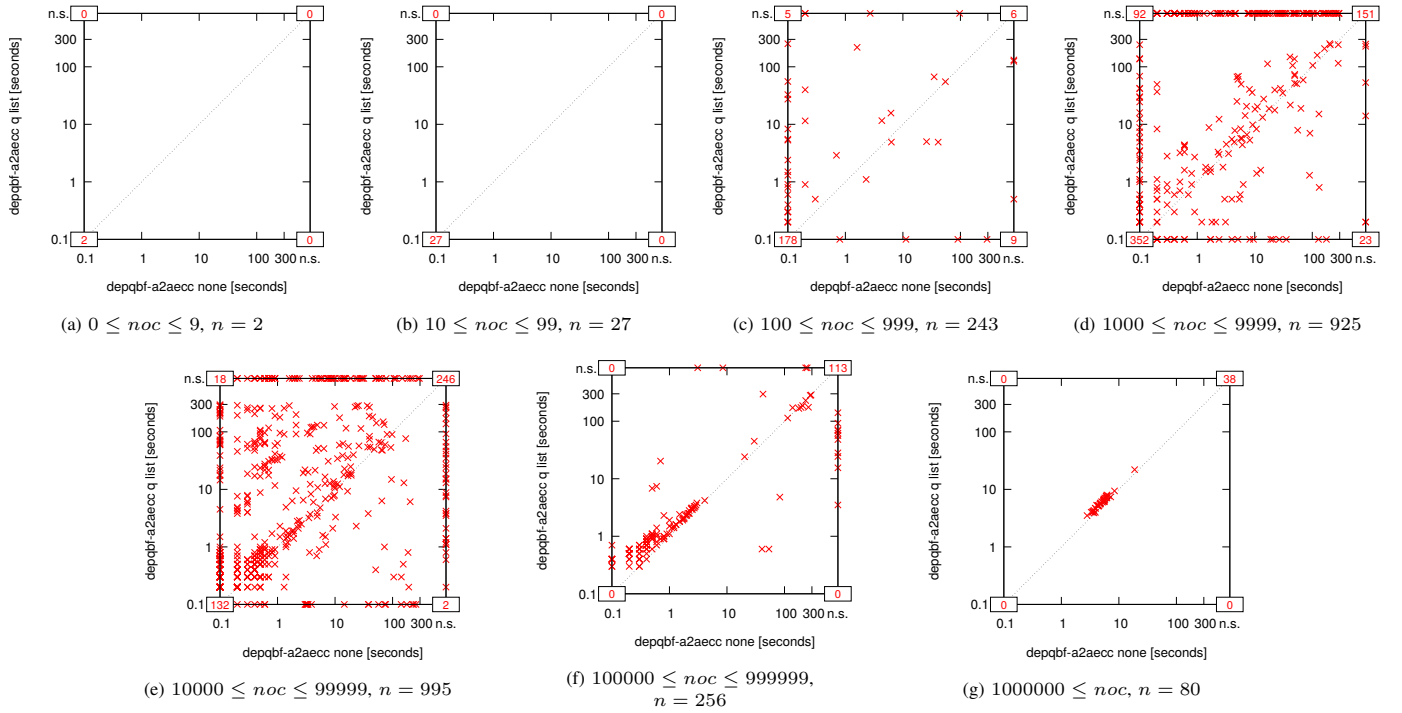


Fig. 1315: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode none partitioned by number of clauses (run time in seconds).

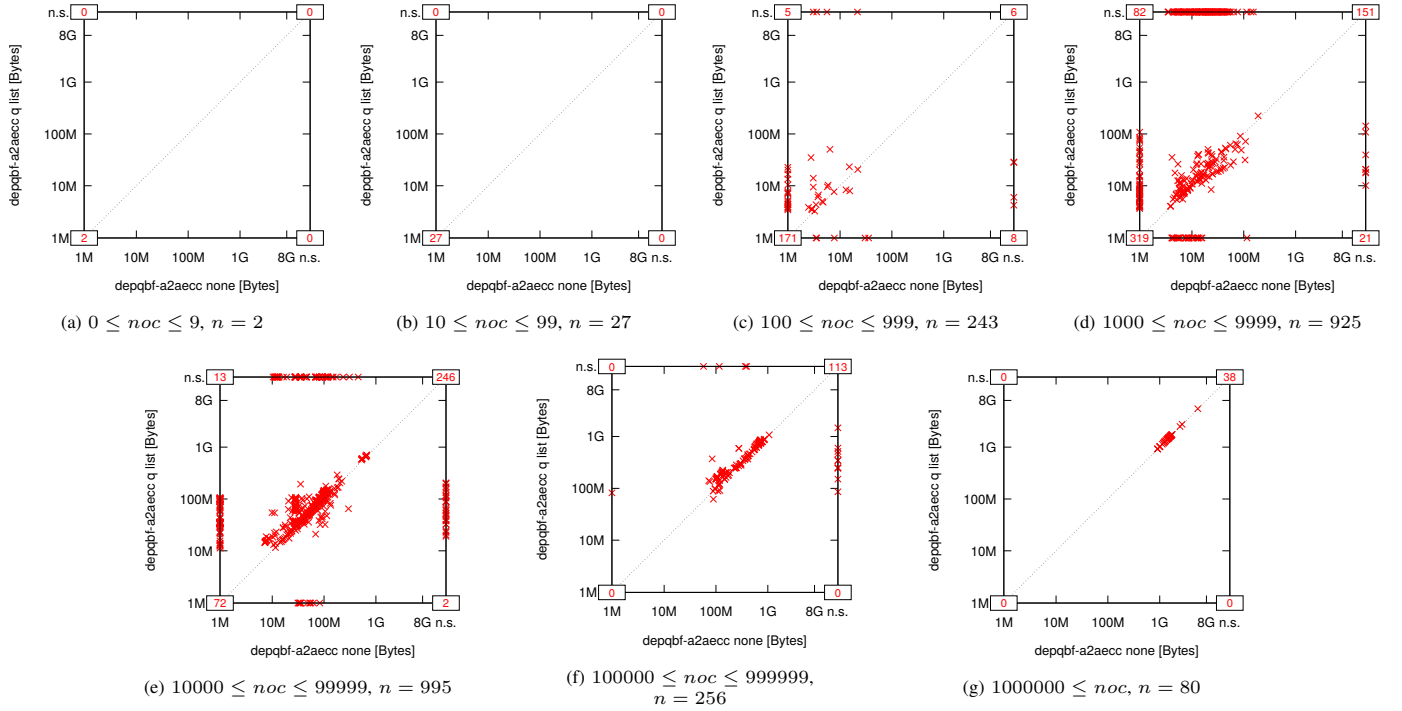


Fig. 1316: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode none partitioned by number of clauses (memory usage in Bytes).

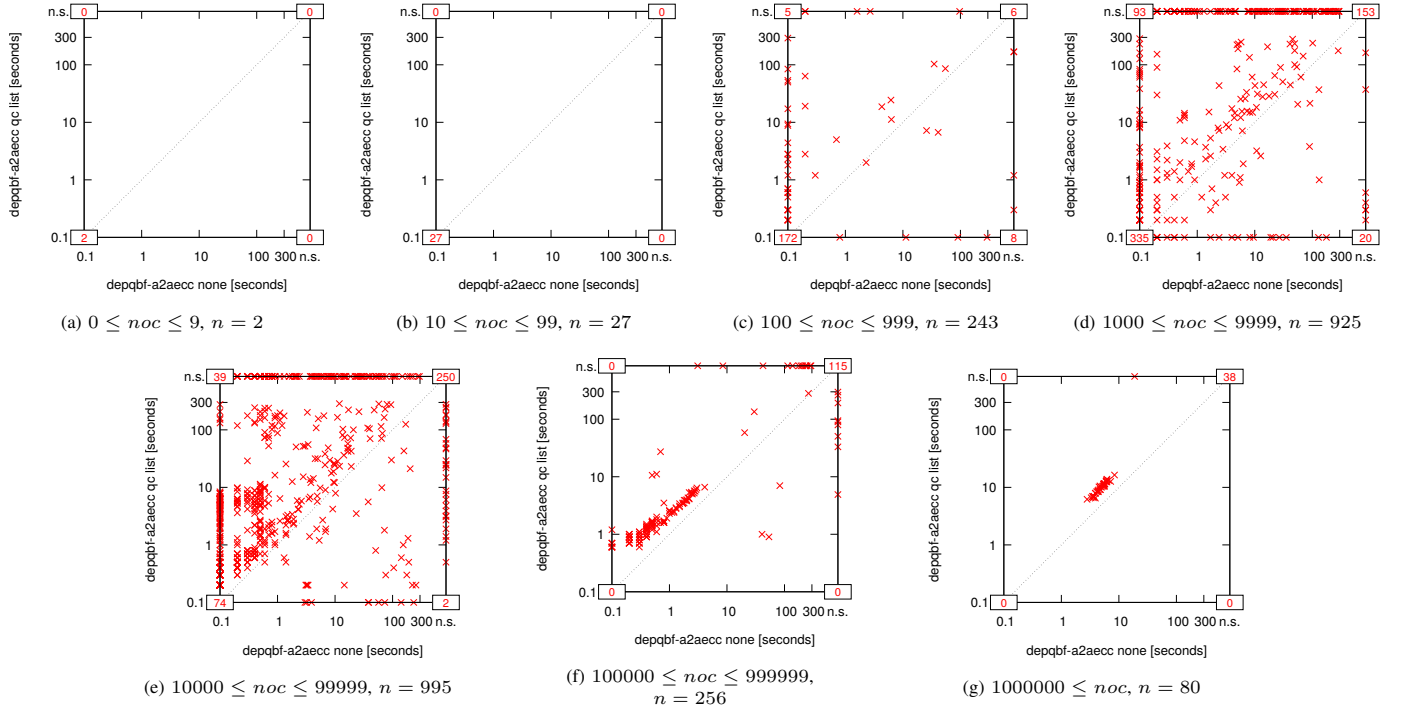


Fig. 1317: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode none partitioned by number of clauses (run time in seconds).

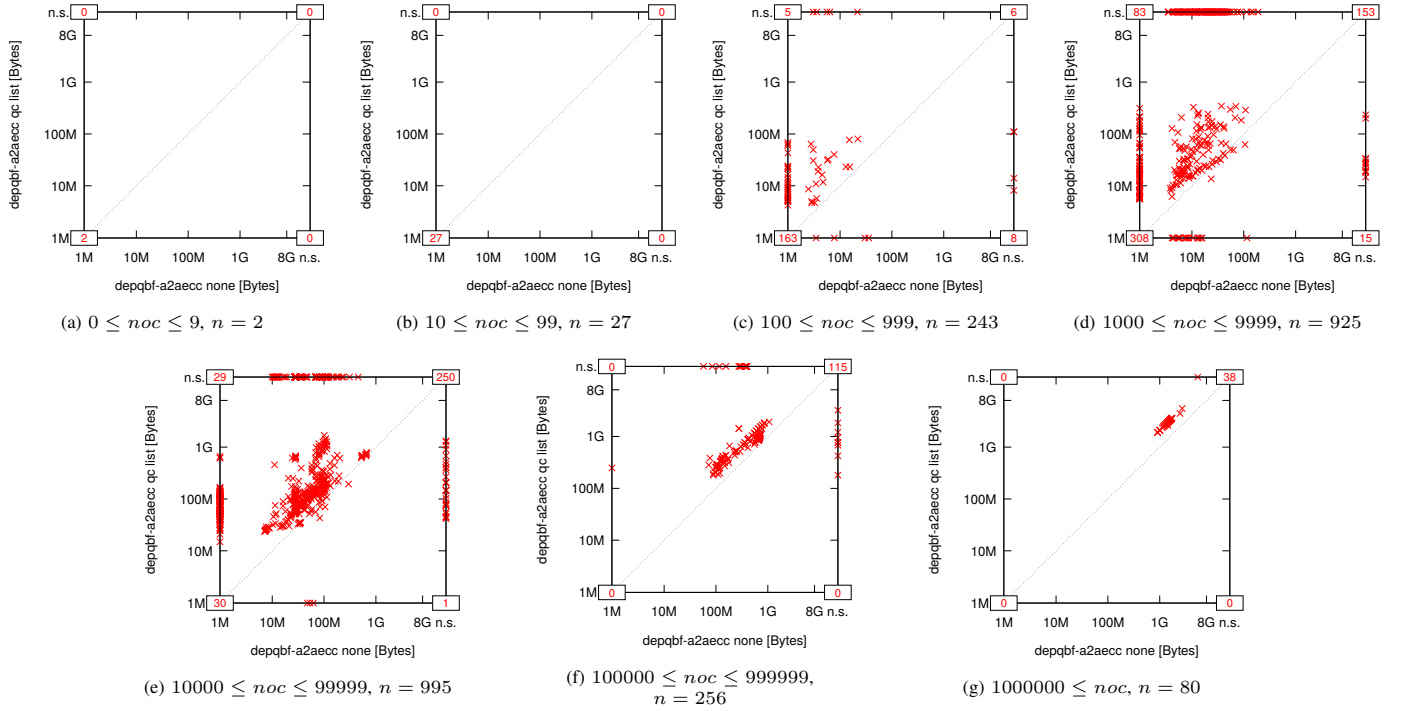


Fig. 1318: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode none partitioned by number of clauses (memory usage in Bytes).

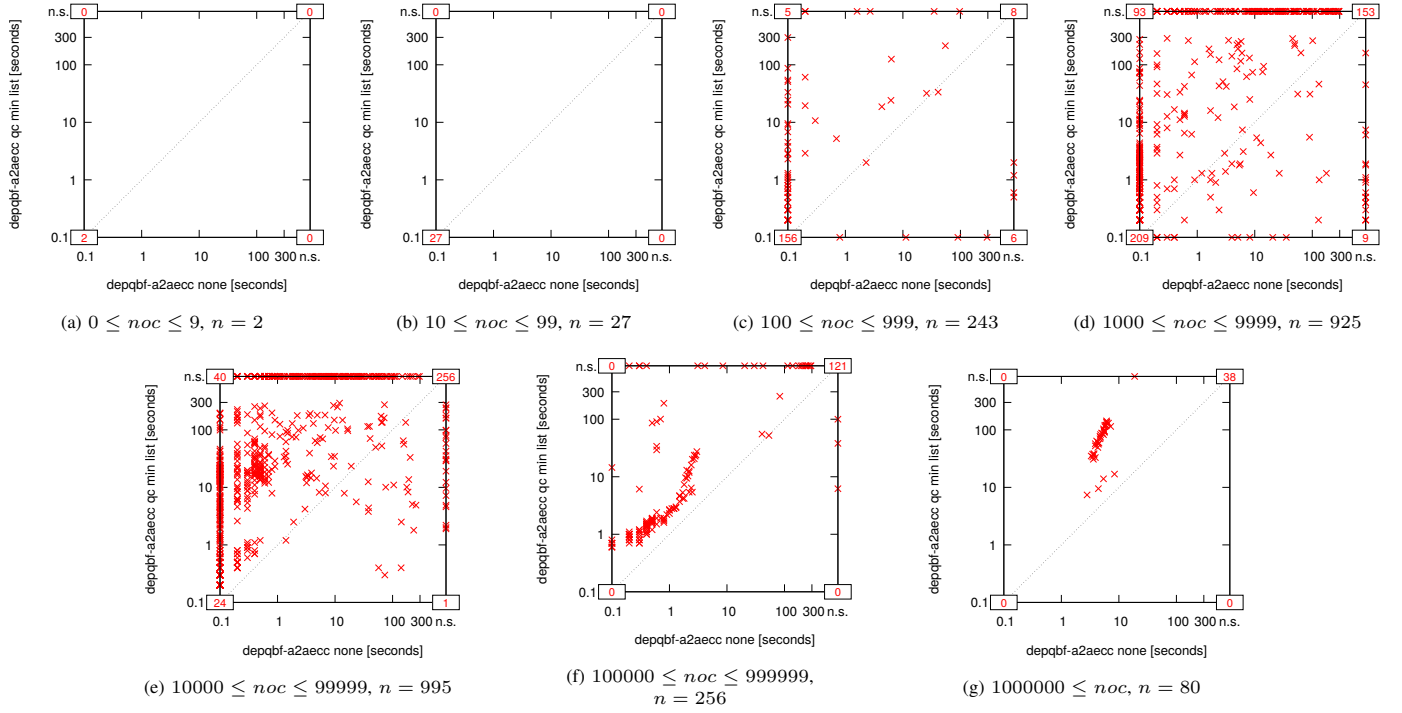


Fig. 1319: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode none partitioned by number of clauses (run time in seconds).

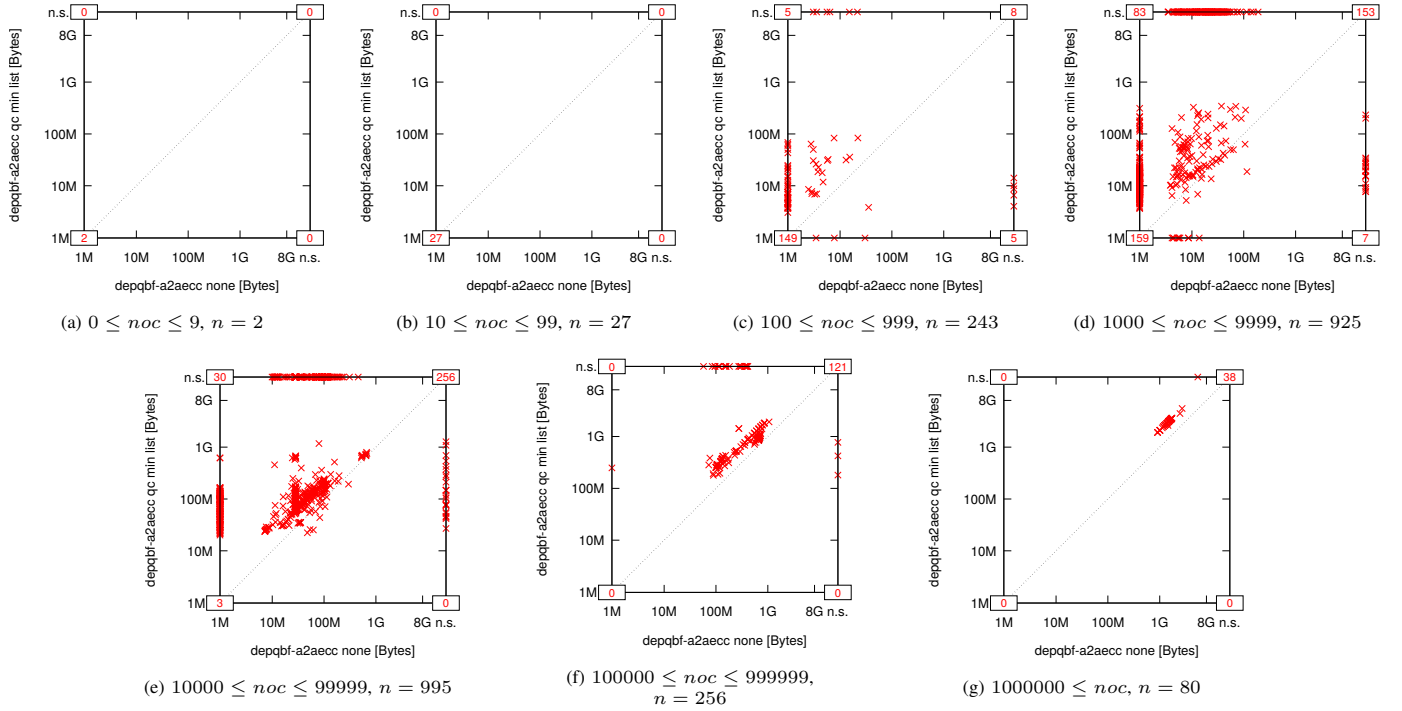


Fig. 1320: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode none partitioned by number of clauses (memory usage in Bytes).

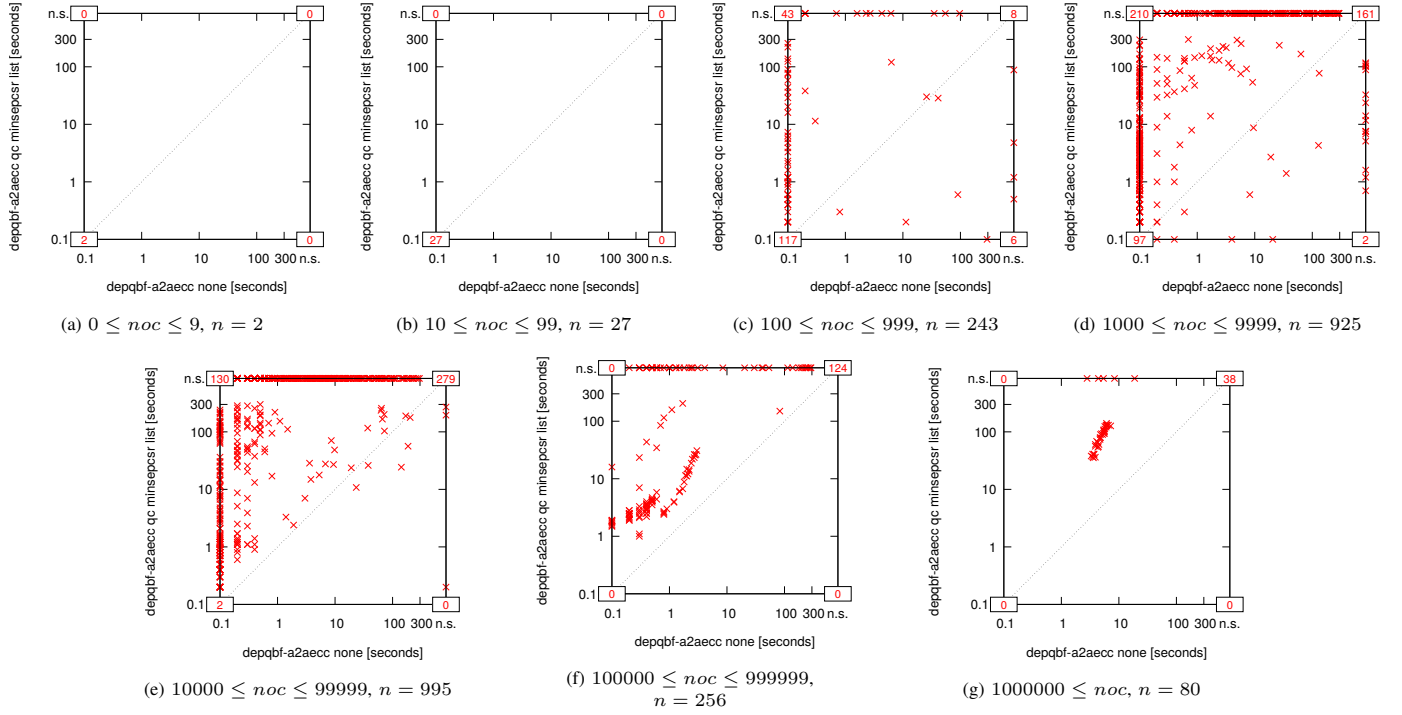


Fig. 1321: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode none partitioned by number of clauses (run time in seconds).

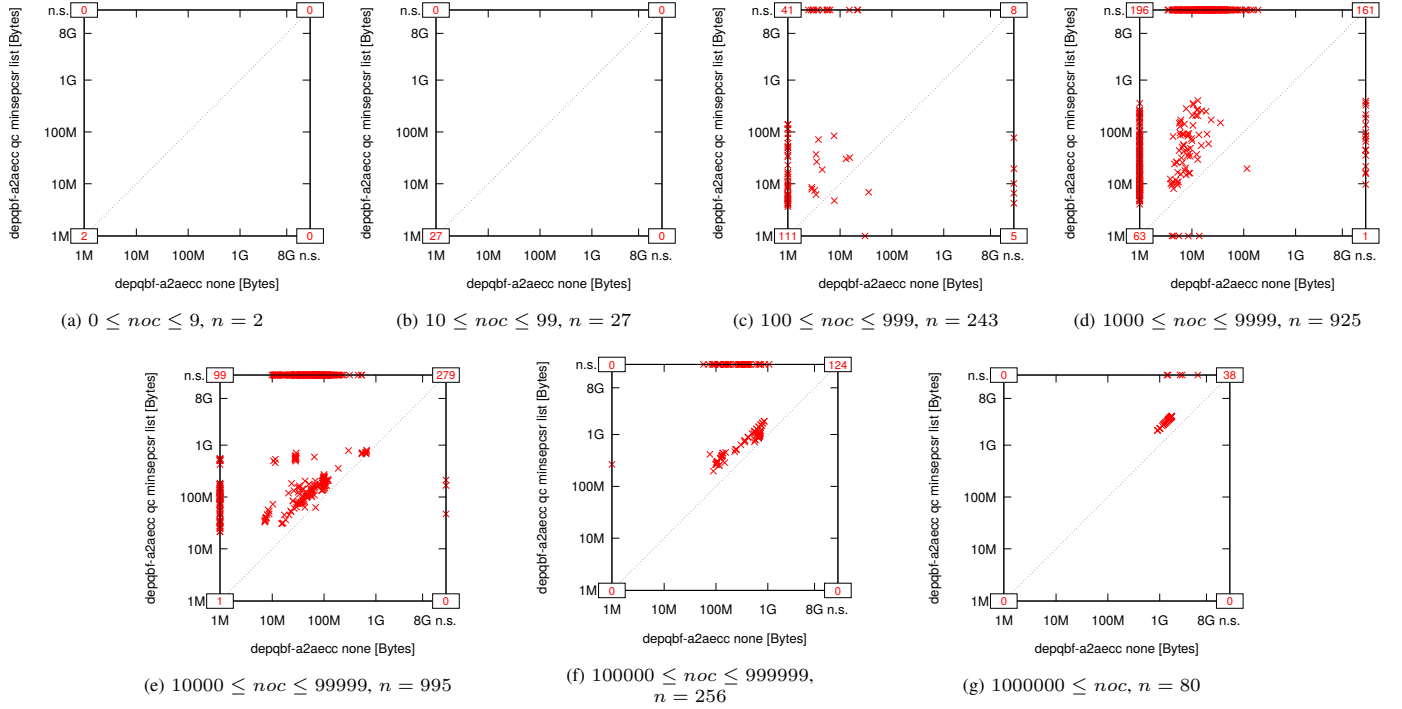


Fig. 1322: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode none partitioned by number of clauses (memory usage in Bytes).

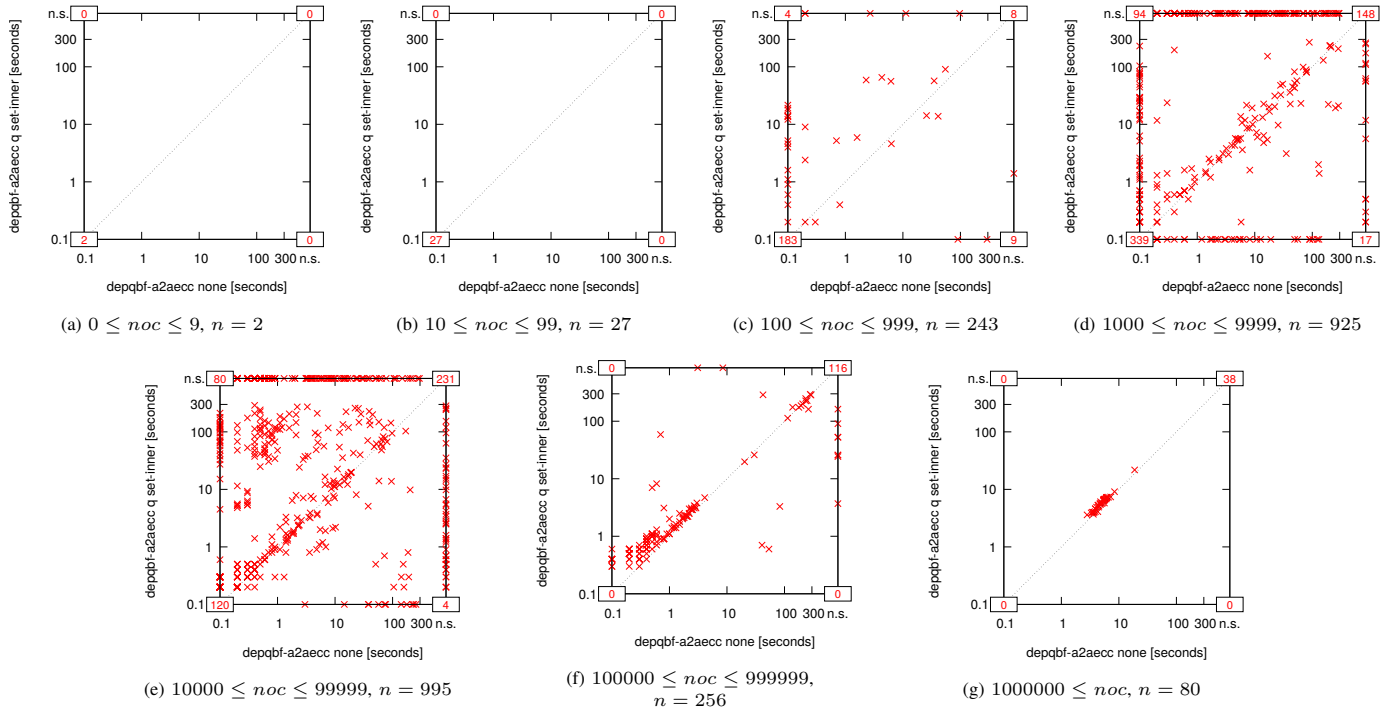


Fig. 1323: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode none partitioned by number of clauses (run time in seconds).

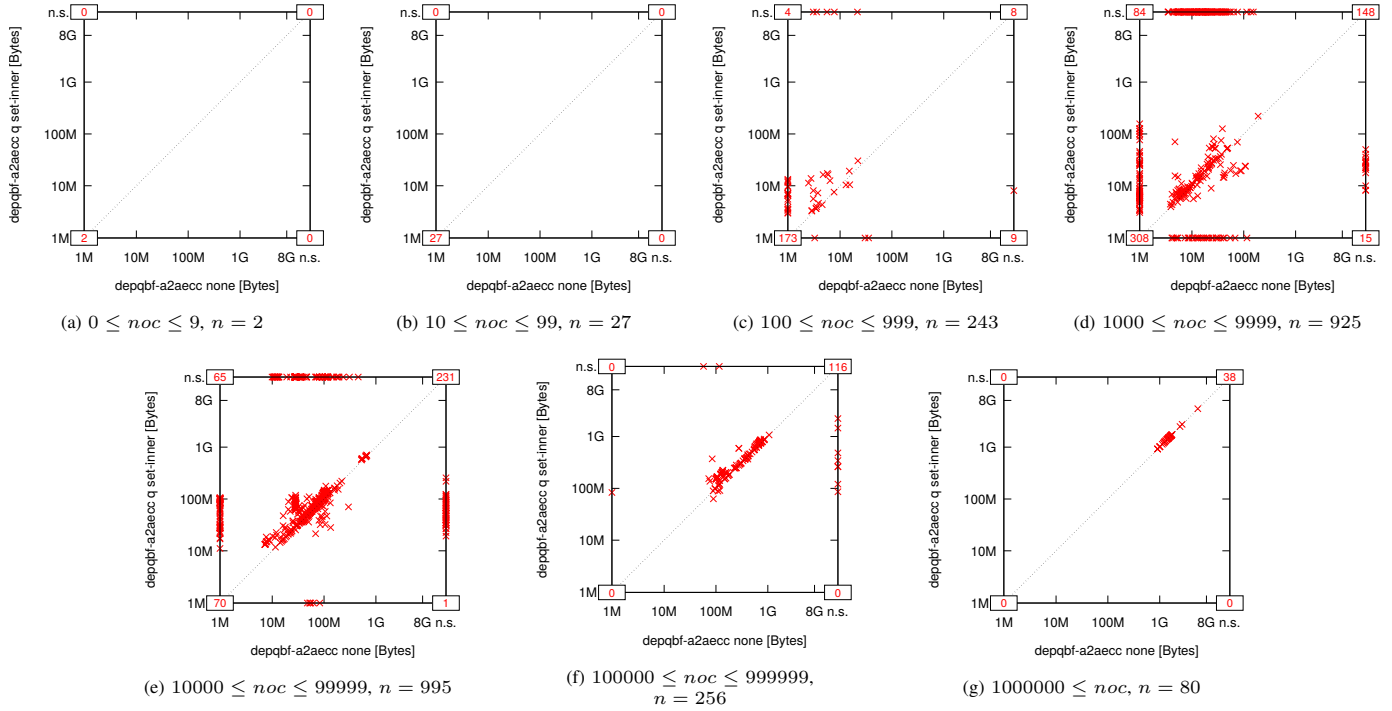


Fig. 1324: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode none partitioned by number of clauses (memory usage in Bytes).

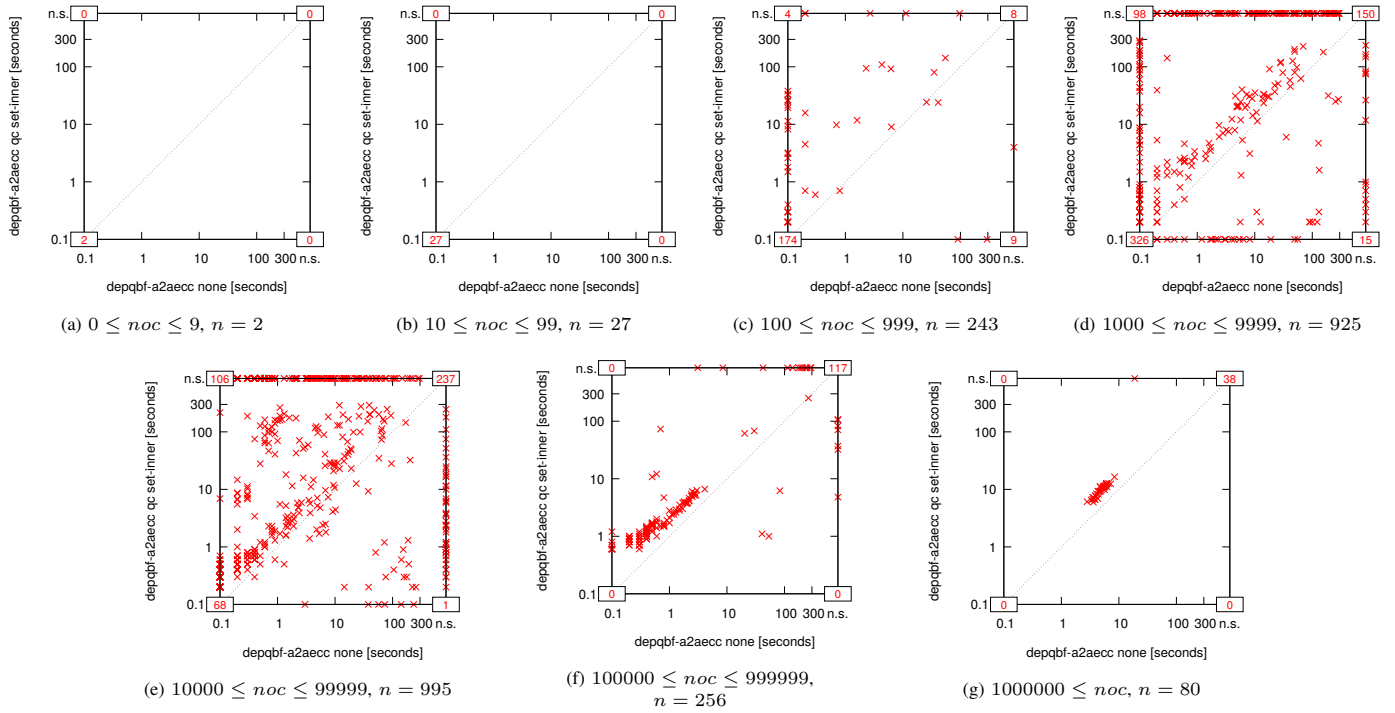


Fig. 1325: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode none partitioned by number of clauses (run time in seconds).

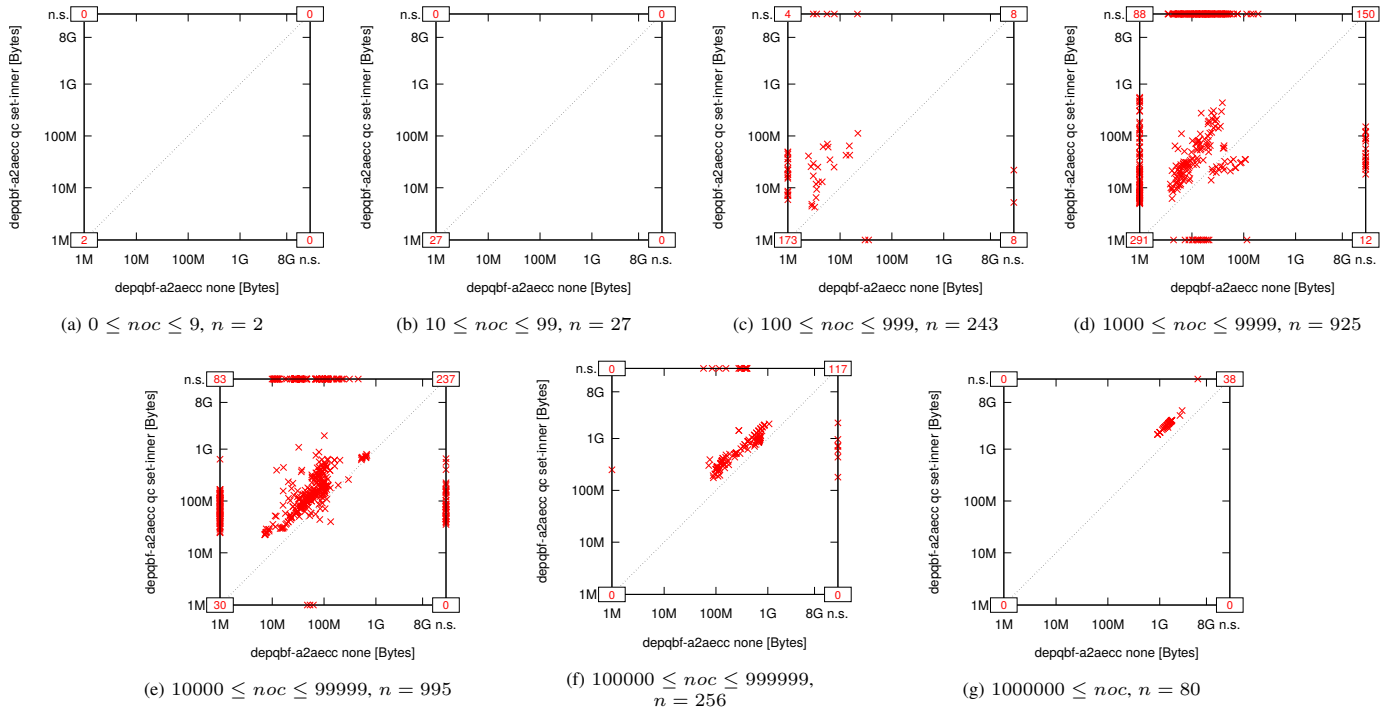


Fig. 1326: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode none partitioned by number of clauses (memory usage in Bytes).

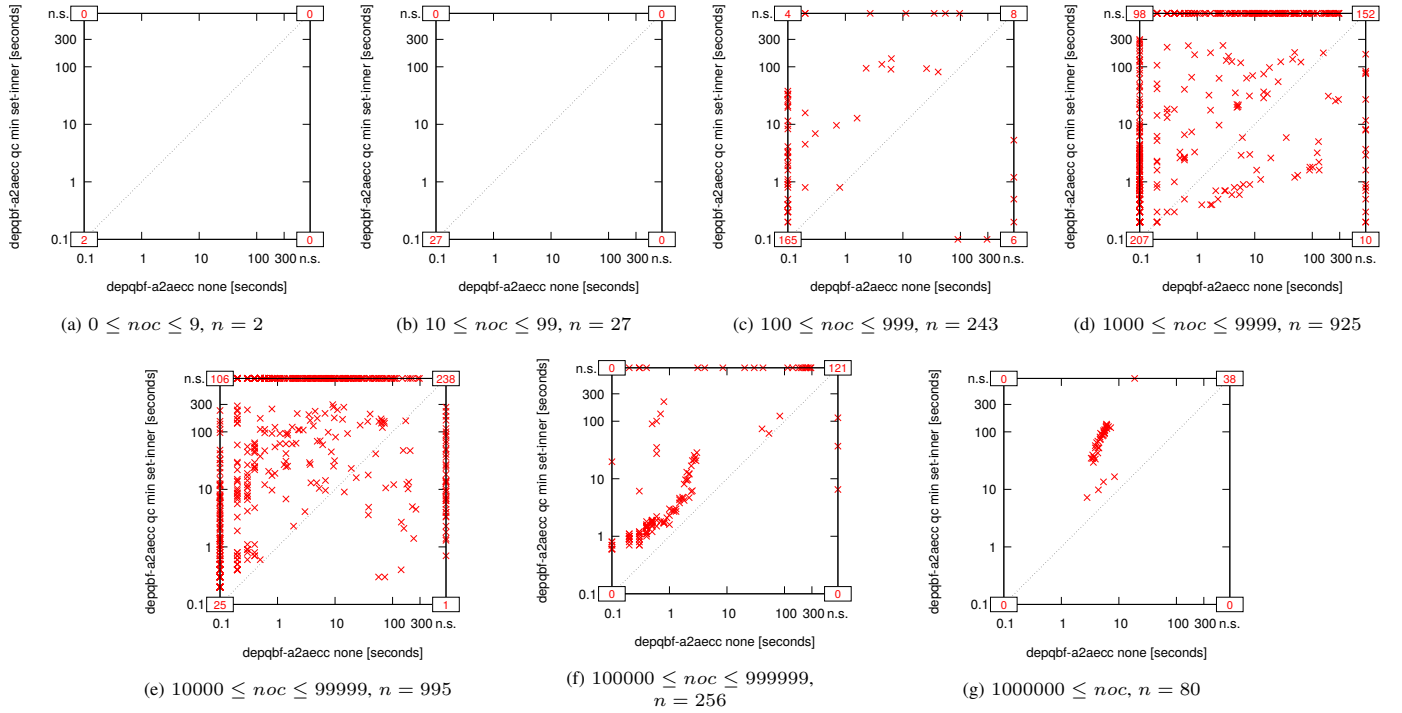


Fig. 1327: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode none partitioned by number of clauses (run time in seconds).

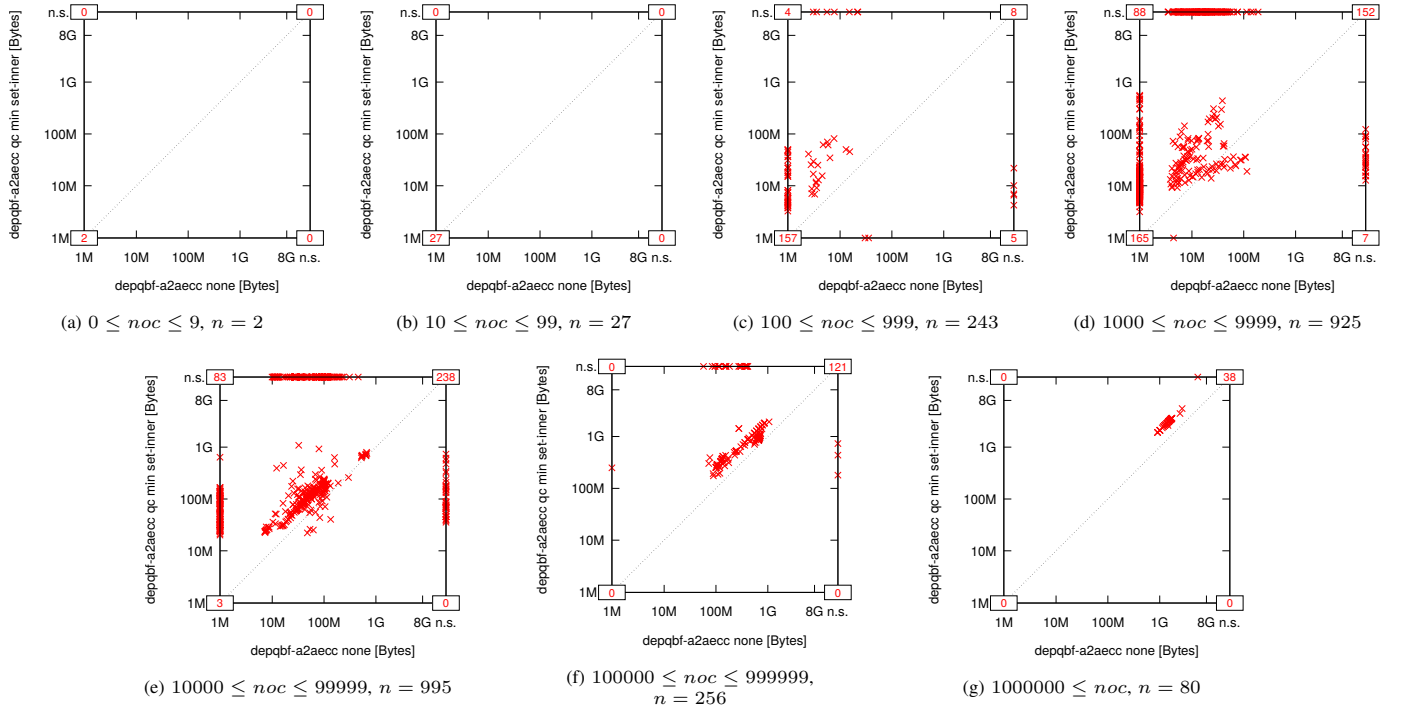


Fig. 1328: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode none partitioned by number of clauses (memory usage in Bytes).

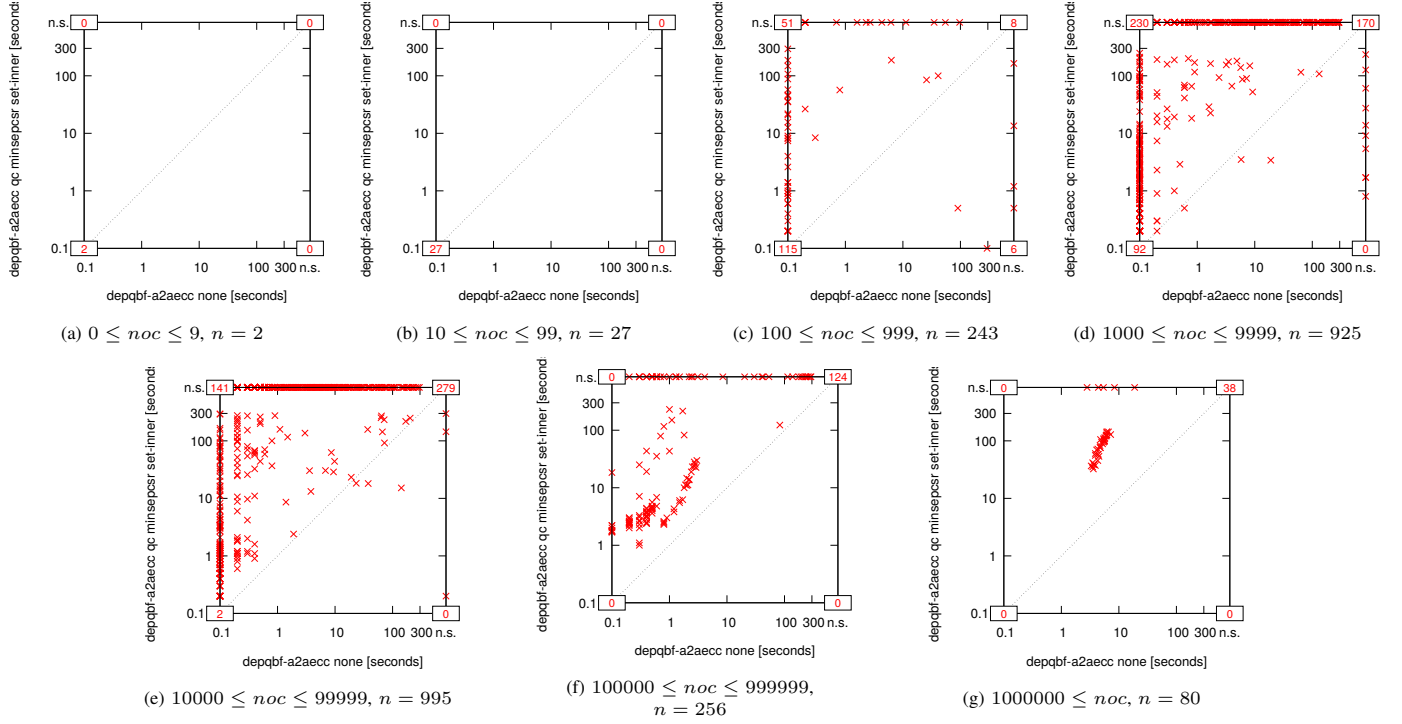


Fig. 1329: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode none partitioned by number of clauses (run time in seconds).

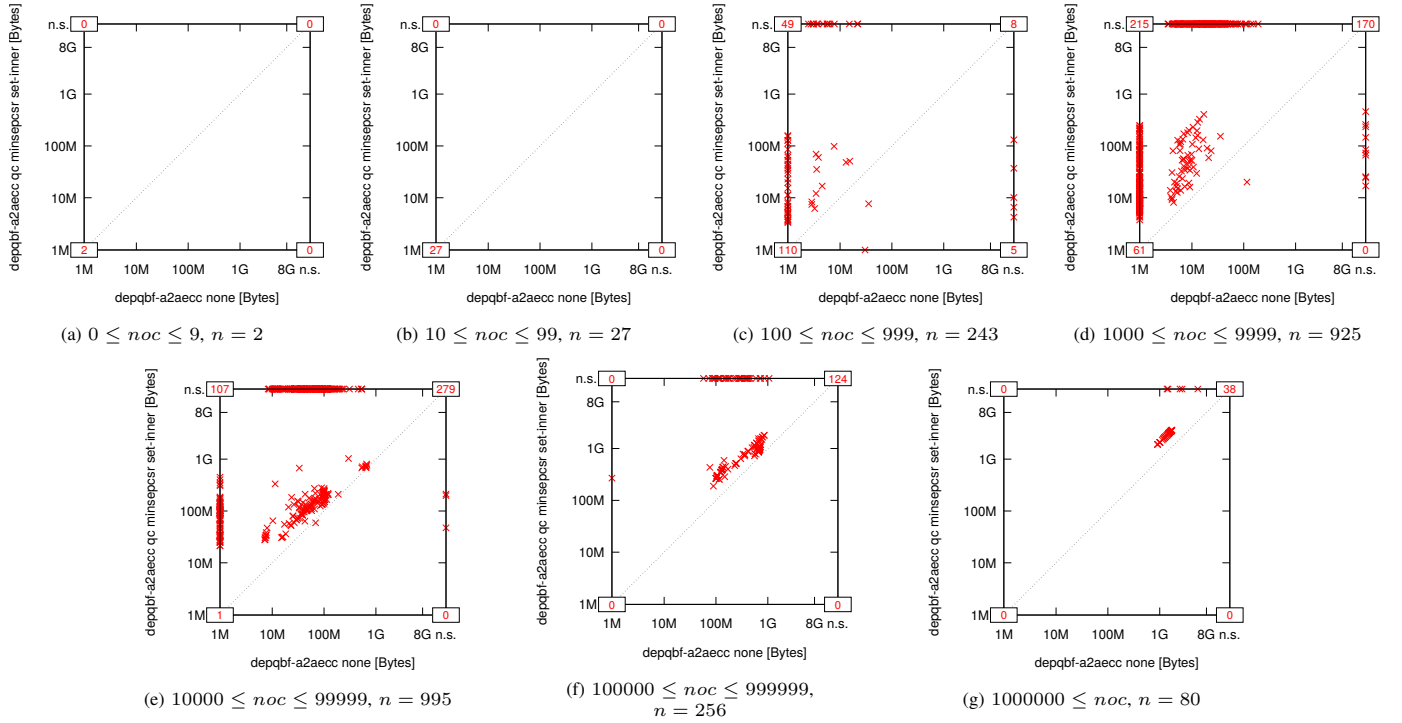


Fig. 1330: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode none partitioned by number of clauses (memory usage in Bytes).

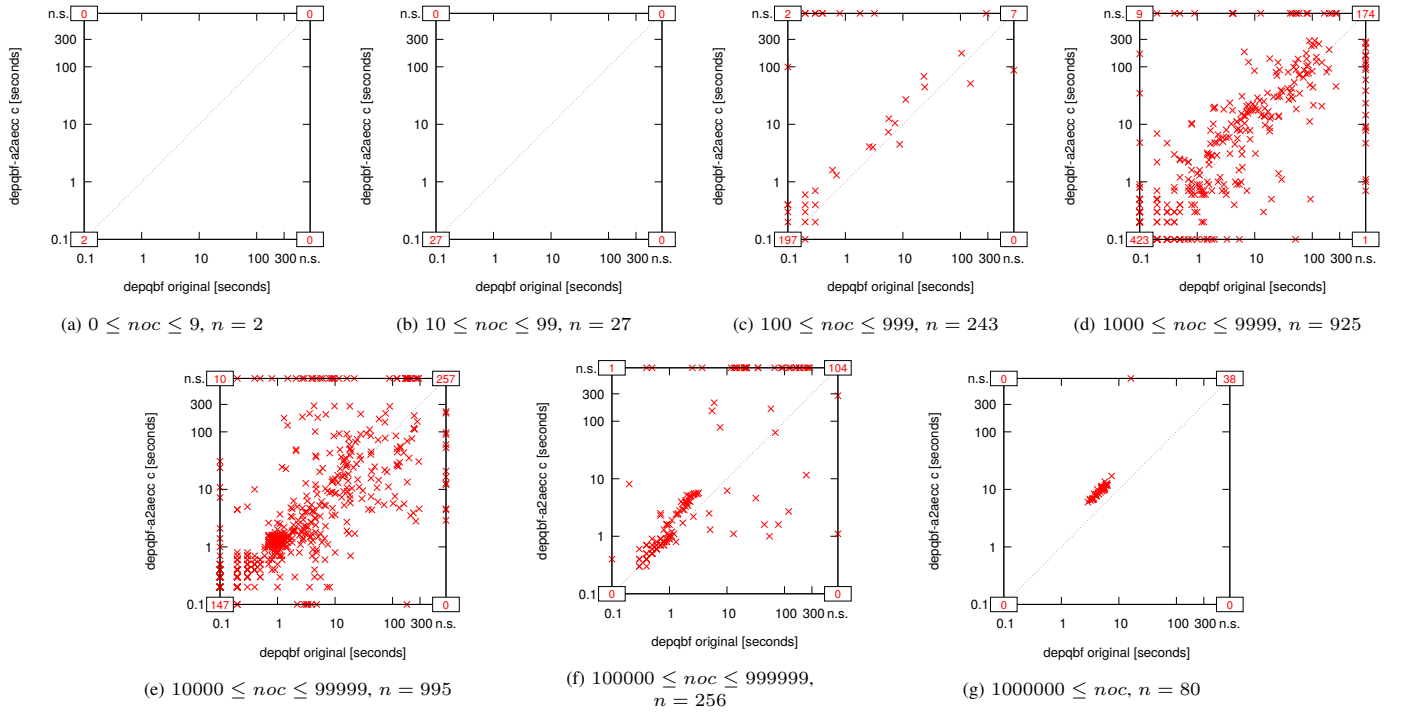


Fig. 1331: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c versus mode original partitioned by number of clauses (run time in seconds).

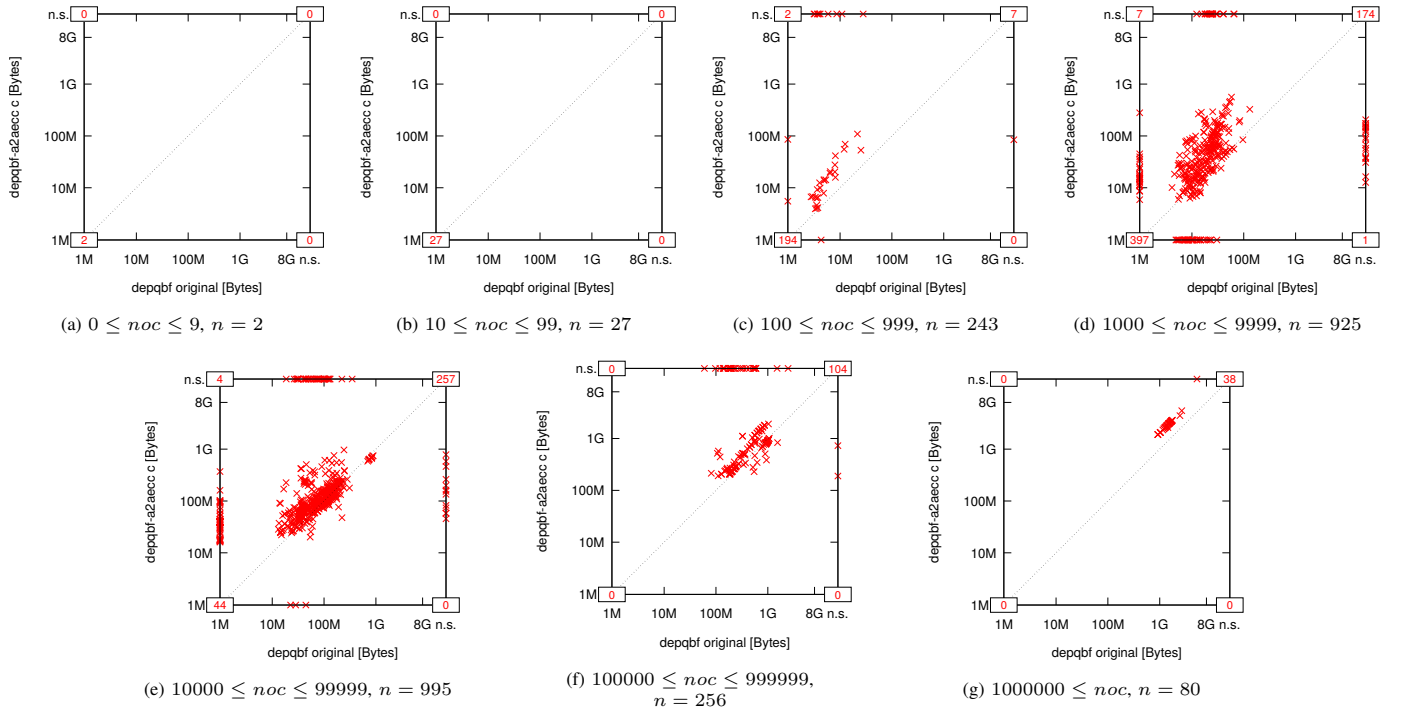


Fig. 1332: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c versus mode original partitioned by number of clauses (memory usage in Bytes).

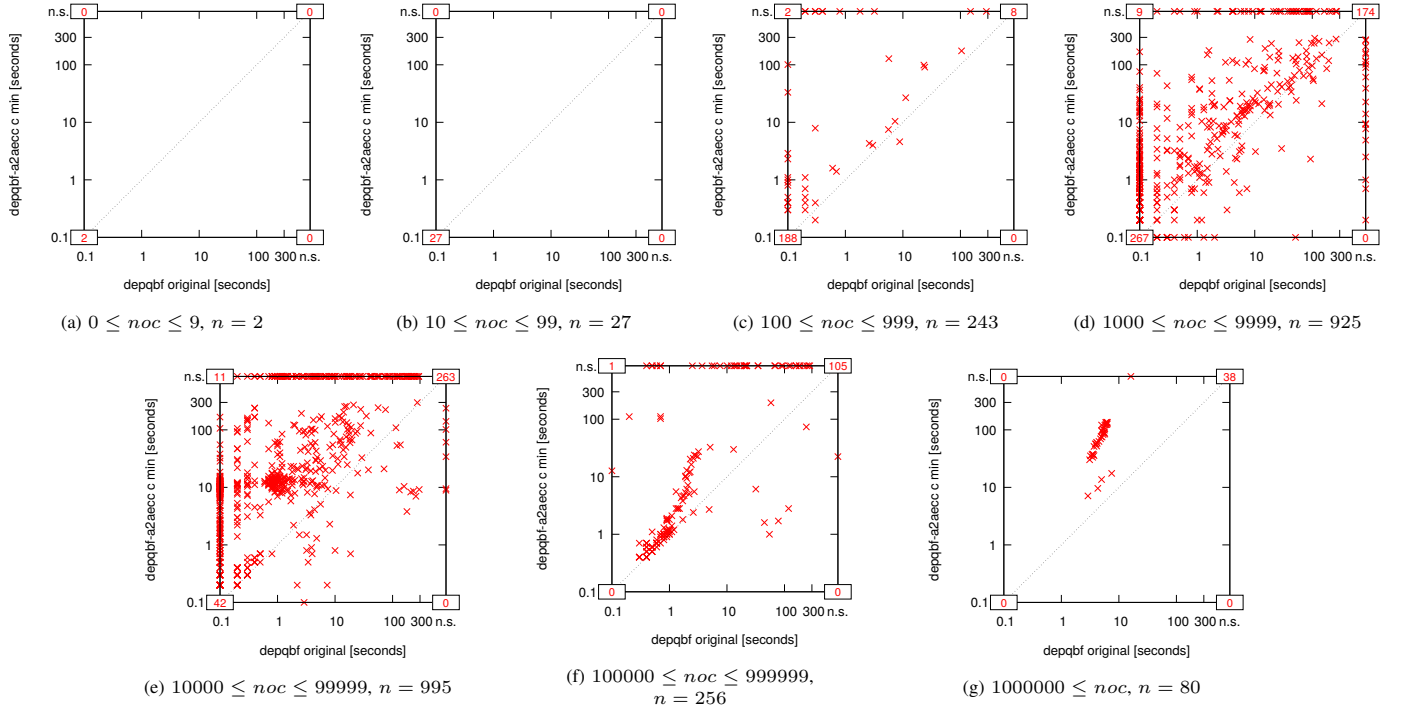


Fig. 1333: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode original partitioned by number of clauses (run time in seconds).

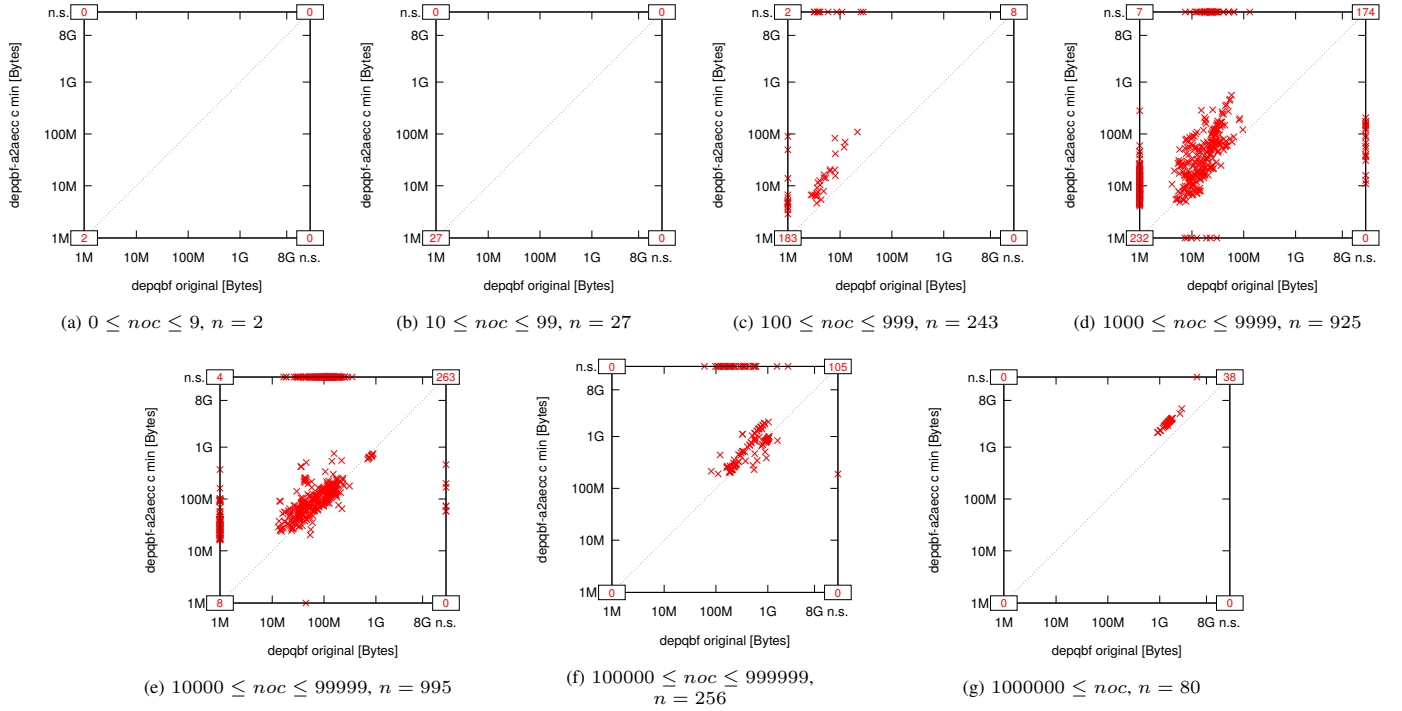


Fig. 1334: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode original partitioned by number of clauses (memory usage in Bytes).

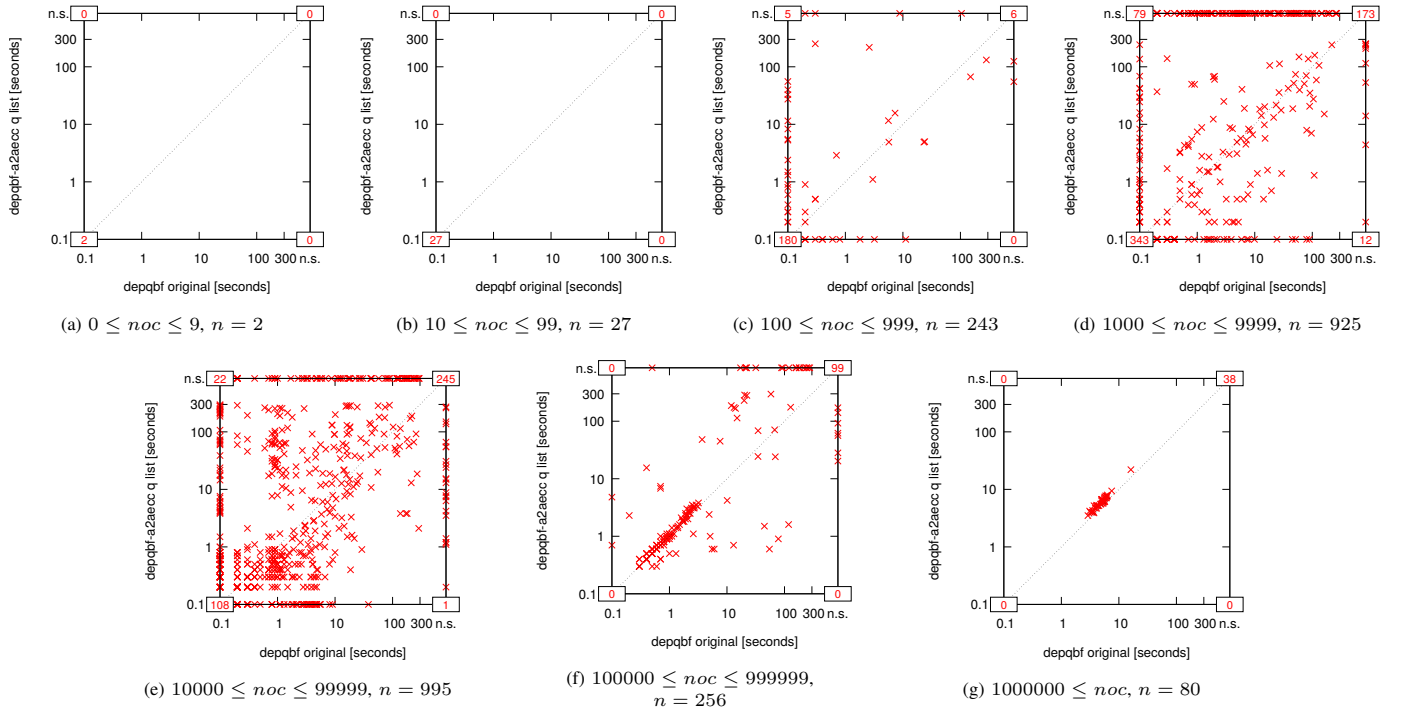


Fig. 1335: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode original partitioned by number of clauses (run time in seconds).

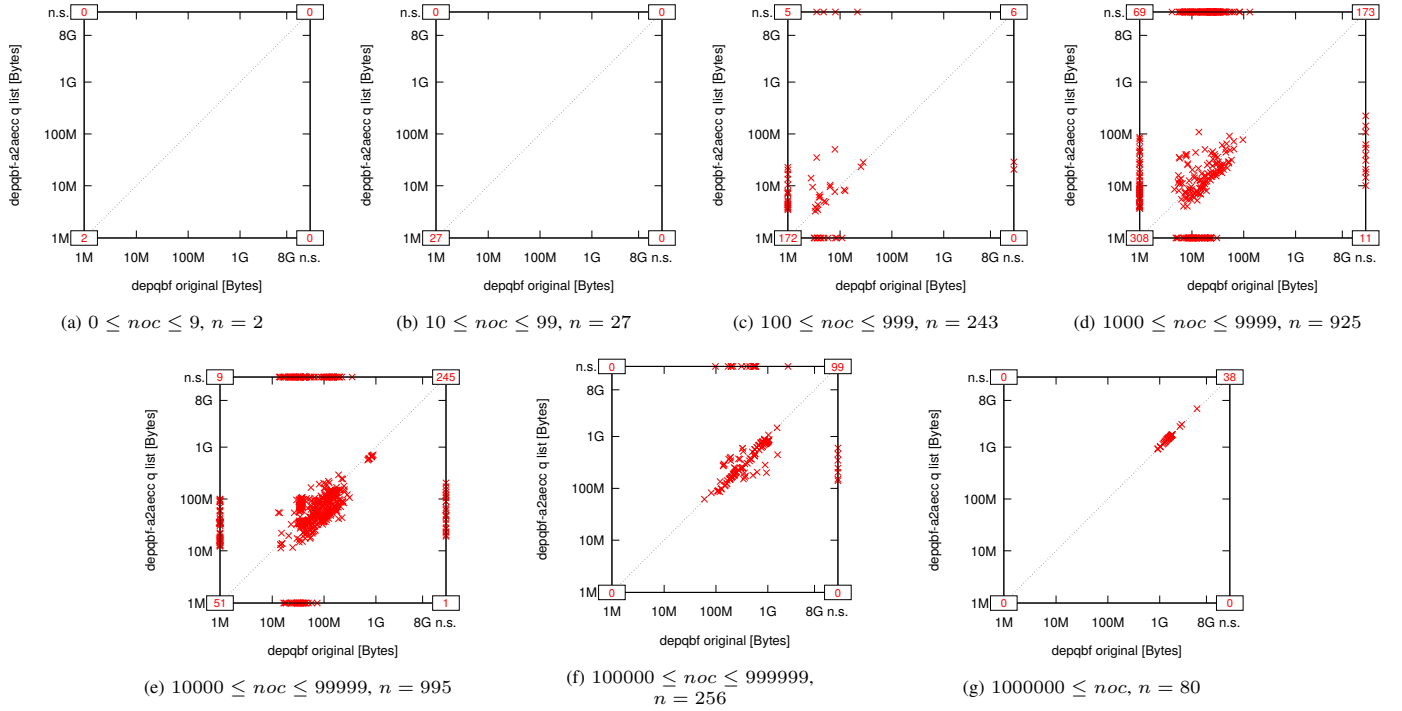


Fig. 1336: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode original partitioned by number of clauses (memory usage in Bytes).

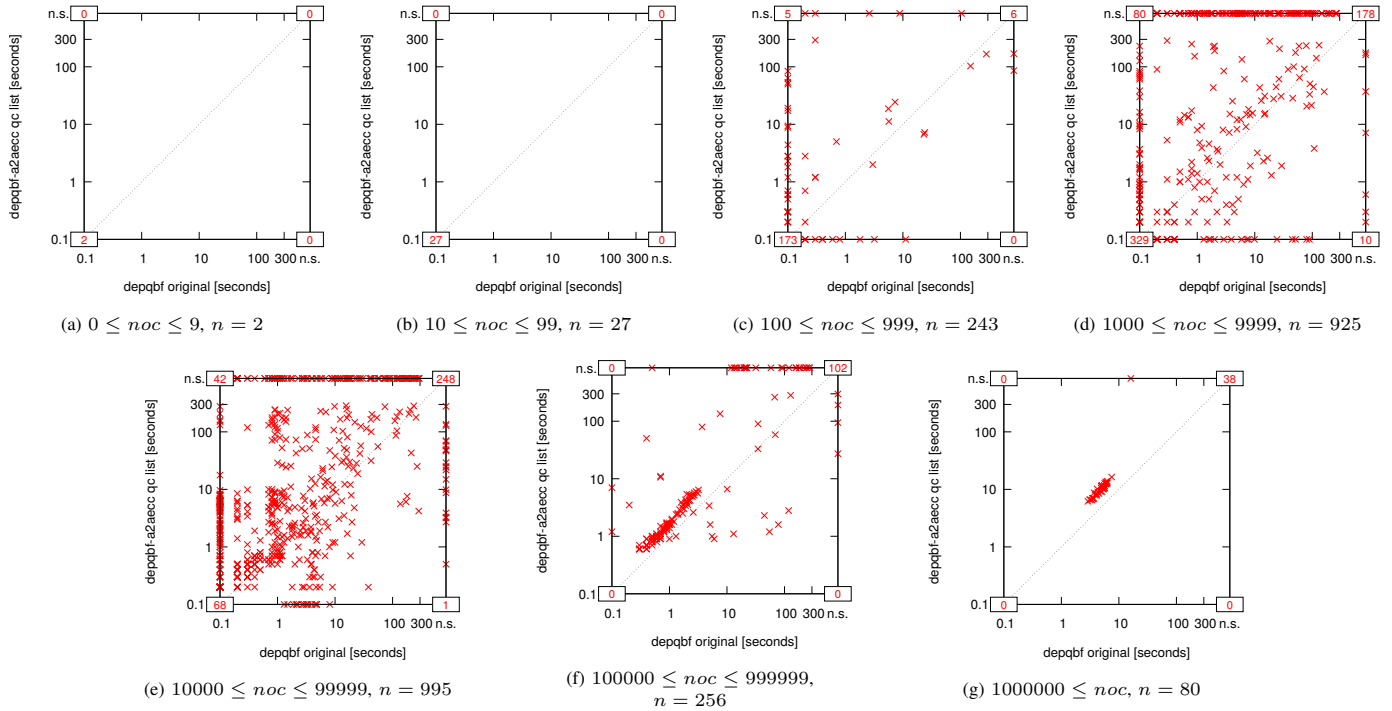


Fig. 1337: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode original partitioned by number of clauses (run time in seconds).

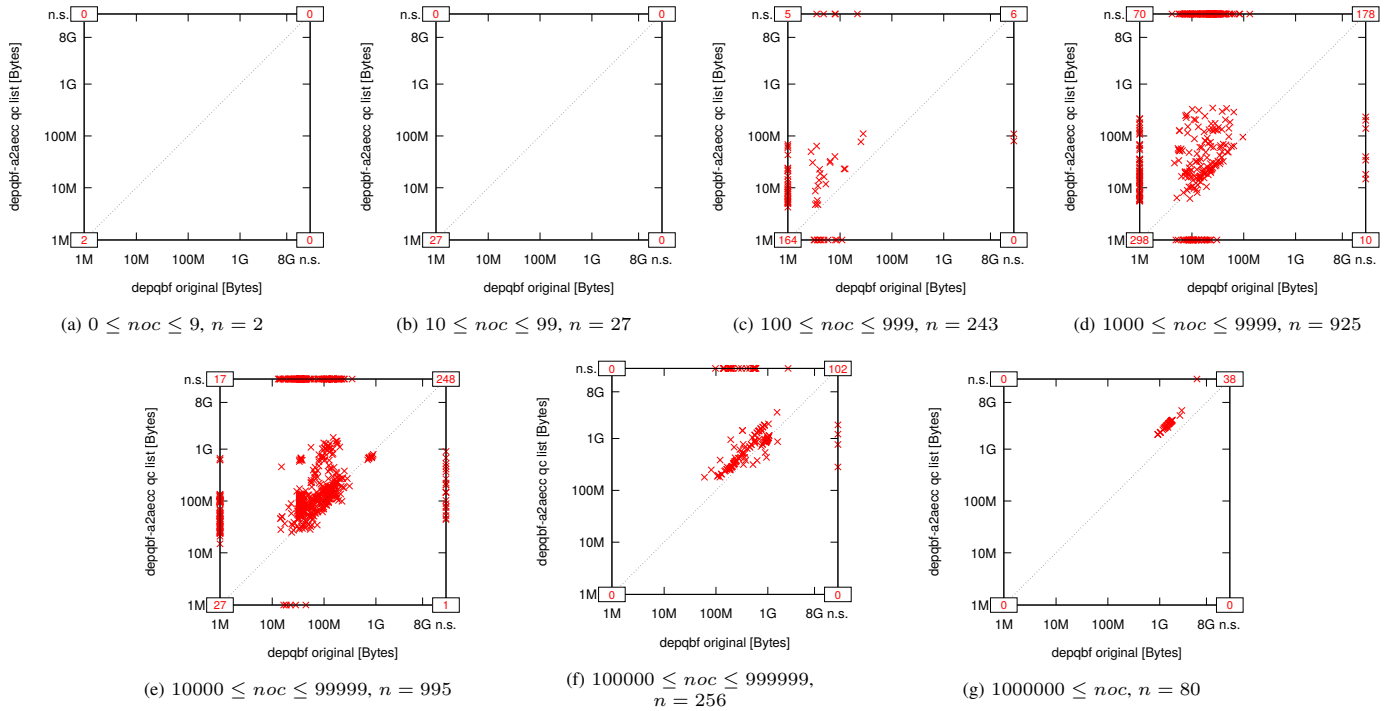


Fig. 1338: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode original partitioned by number of clauses (memory usage in Bytes).

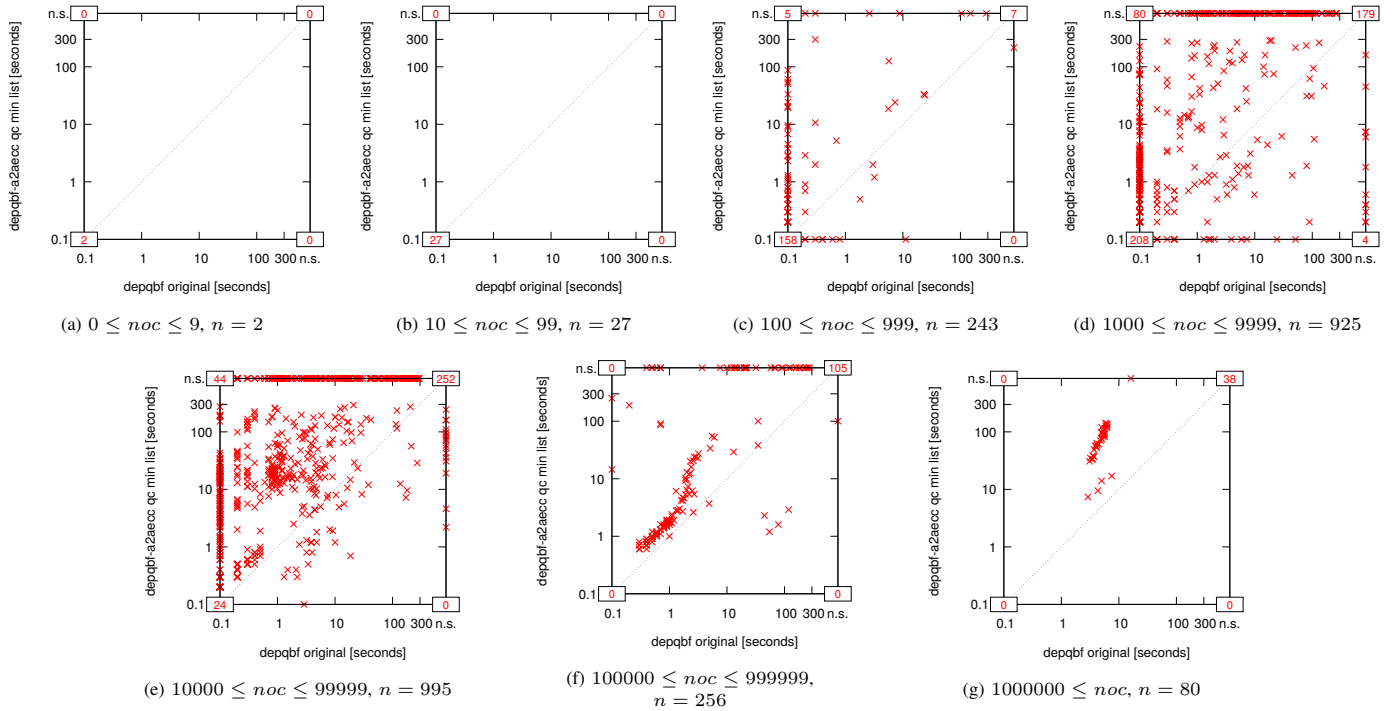


Fig. 1339: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode original partitioned by number of clauses (run time in seconds).

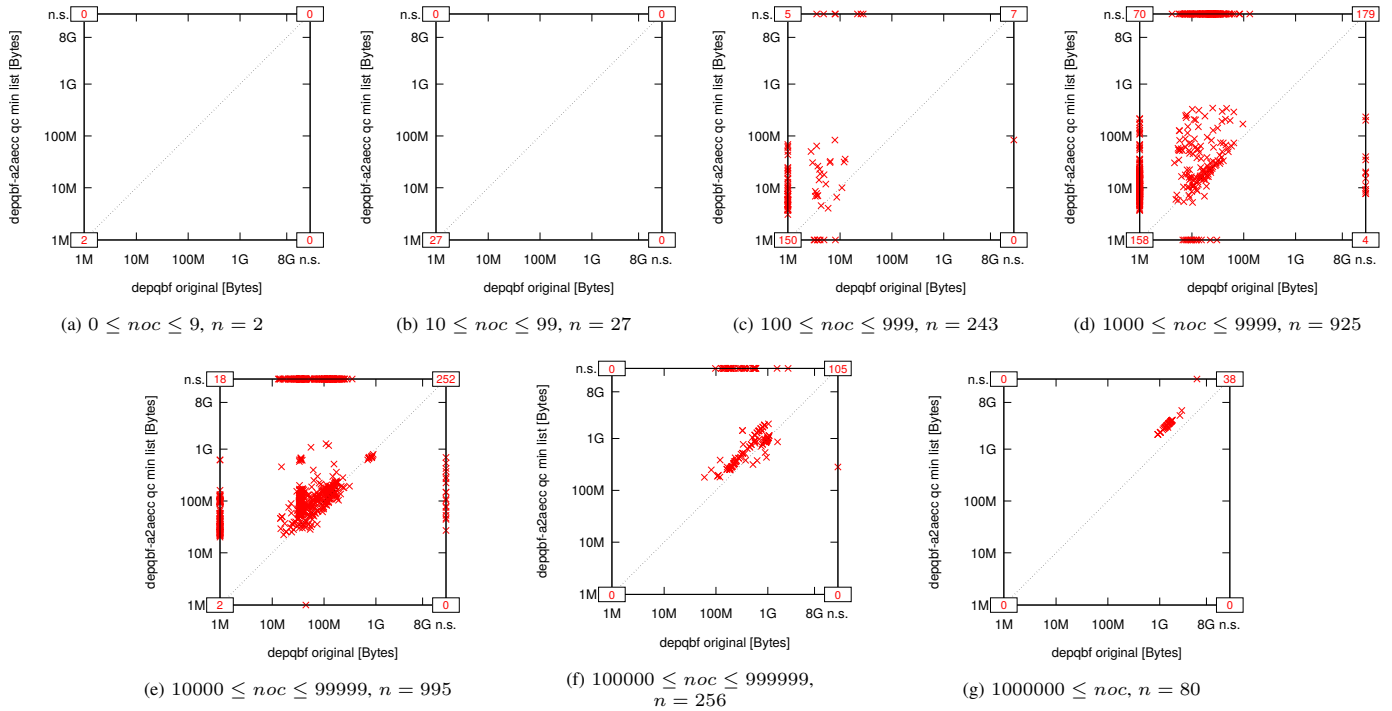


Fig. 1340: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode original partitioned by number of clauses (memory usage in Bytes).

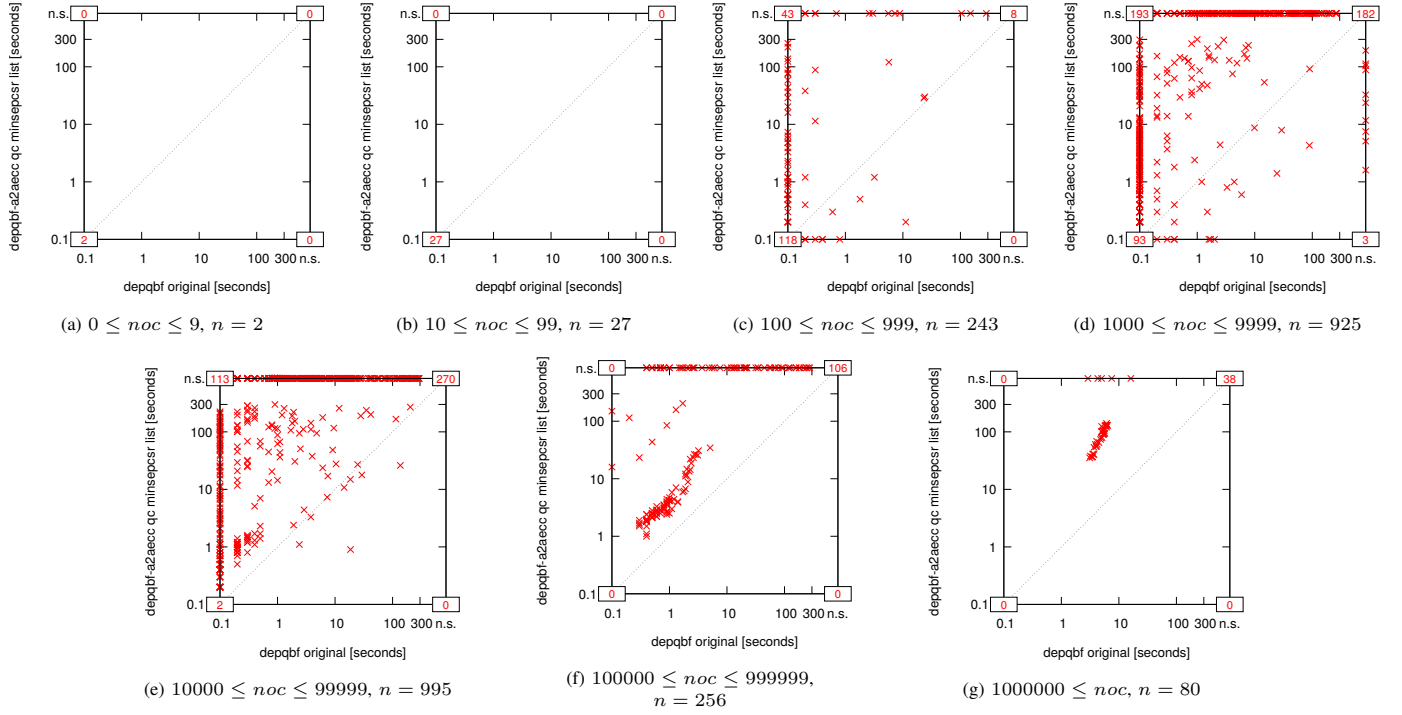


Fig. 1341: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode original partitioned by number of clauses (run time in seconds).

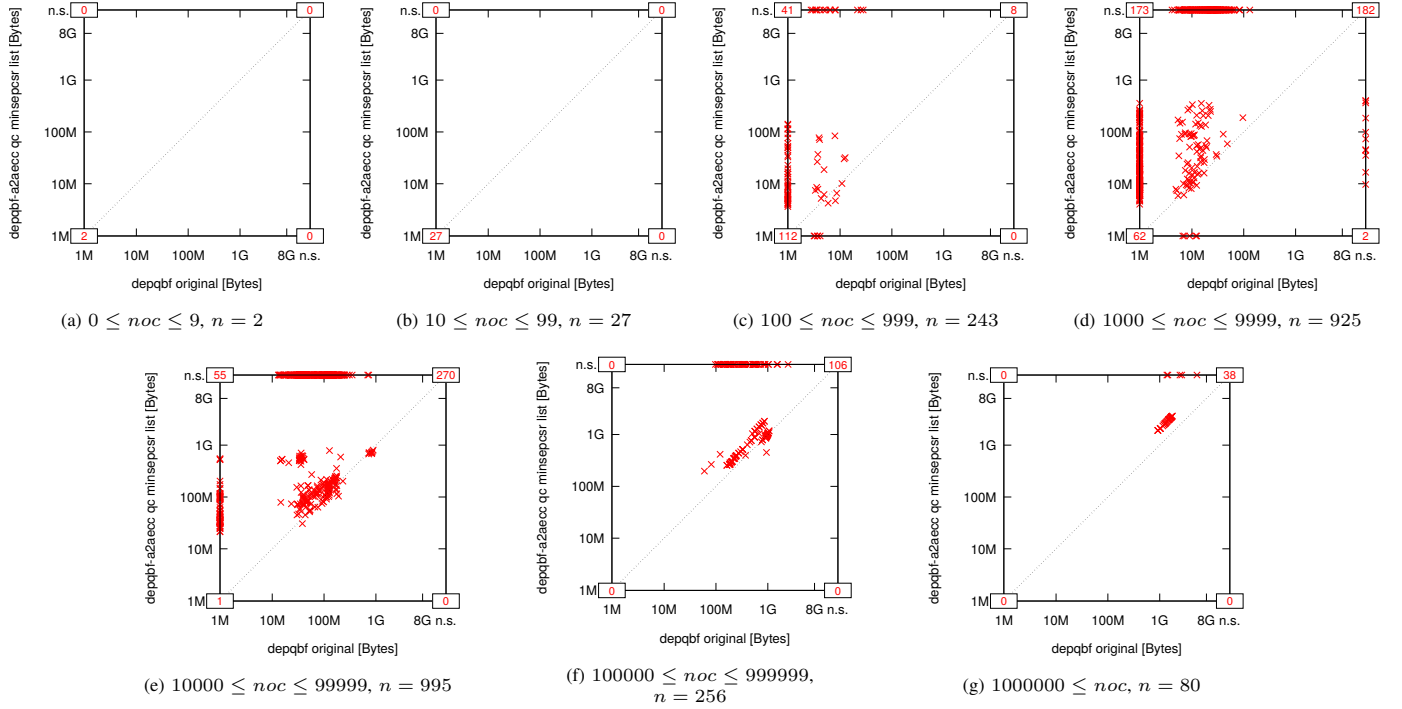


Fig. 1342: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode original partitioned by number of clauses (memory usage in Bytes).

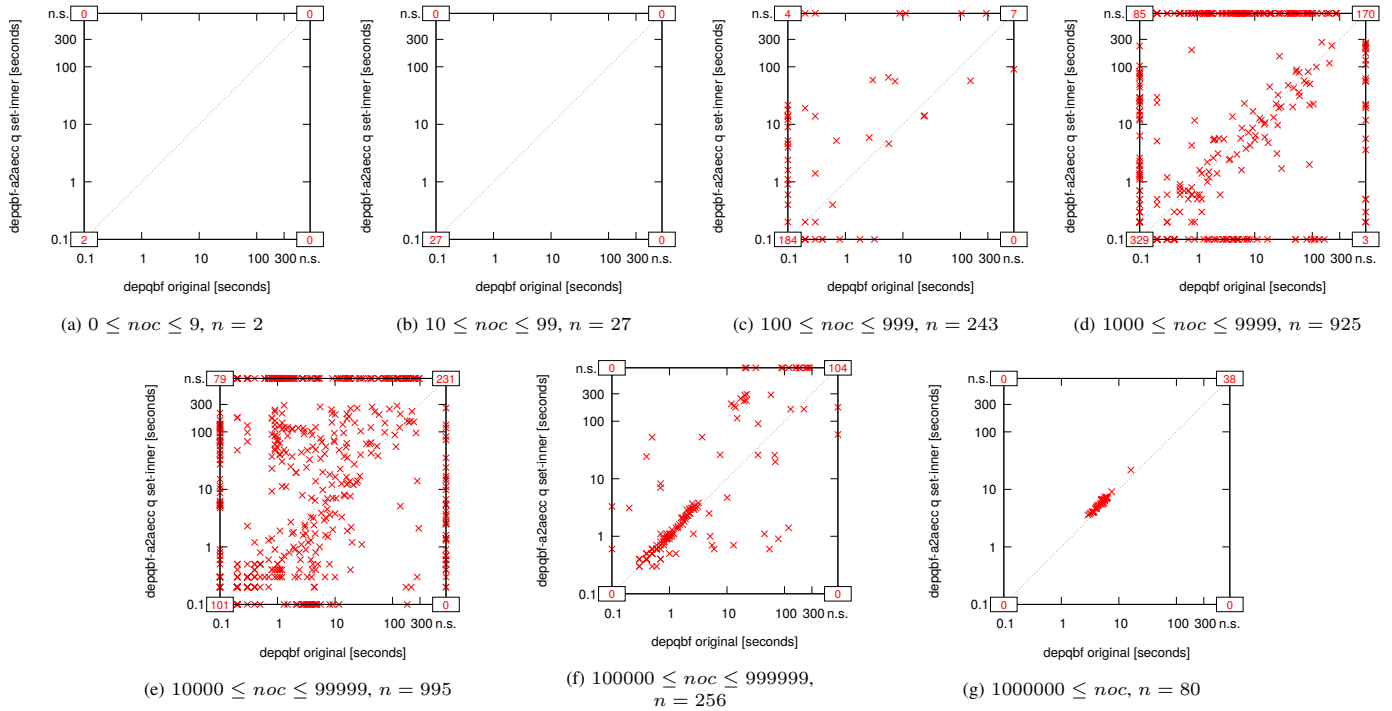


Fig. 1343: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode original partitioned by number of clauses (run time in seconds).

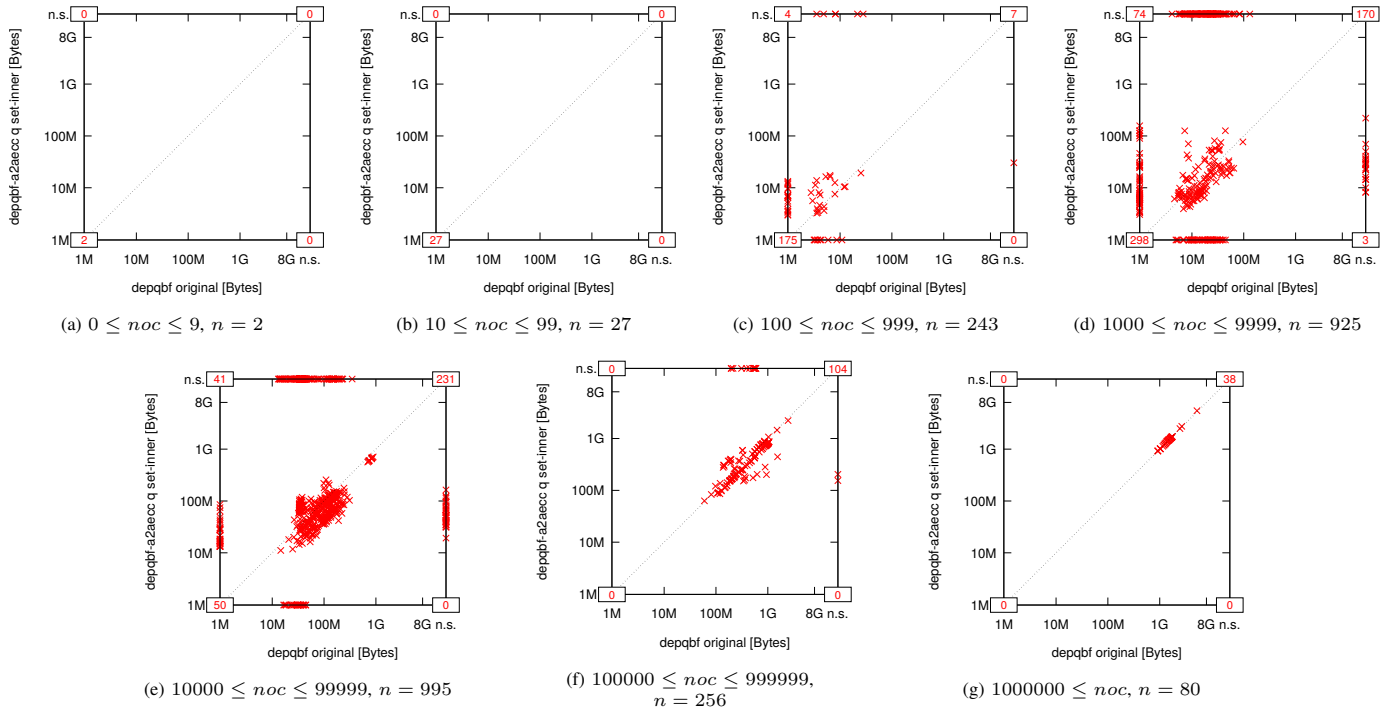


Fig. 1344: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode original partitioned by number of clauses (memory usage in Bytes).

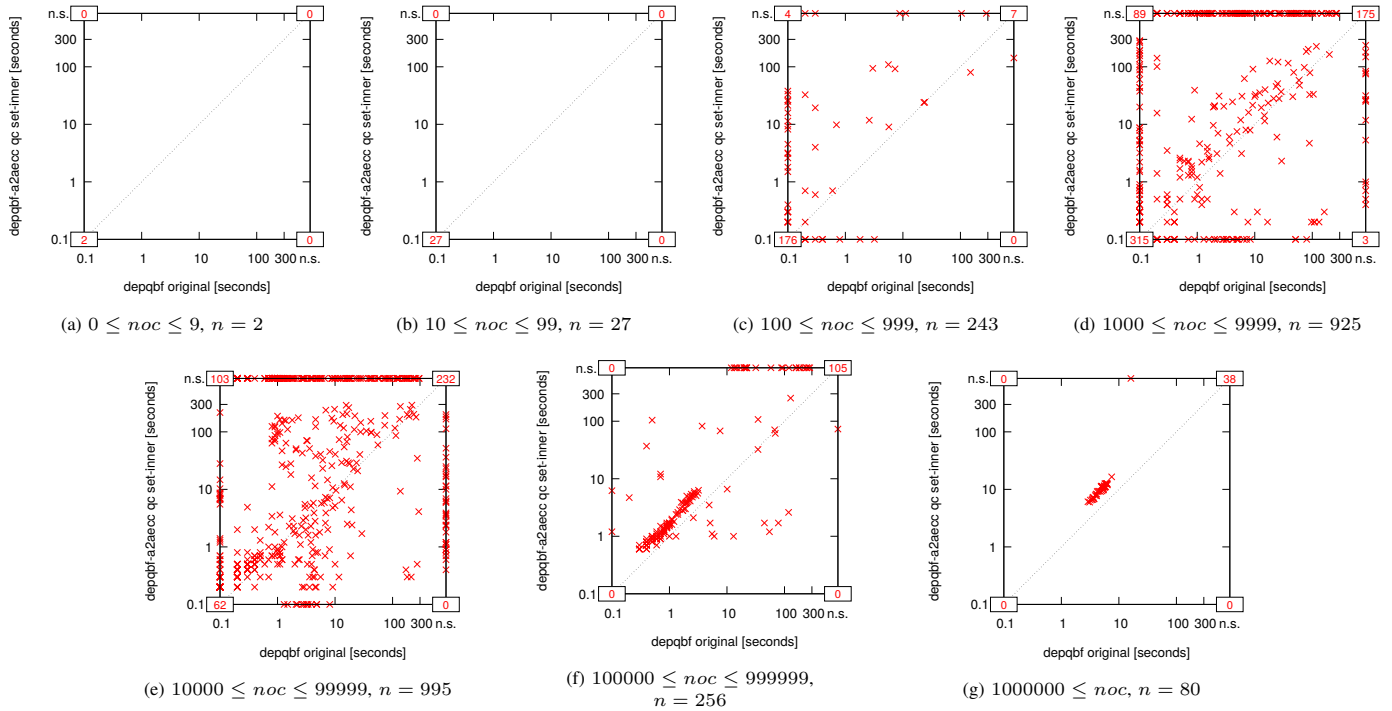


Fig. 1345: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode original partitioned by number of clauses (run time in seconds).

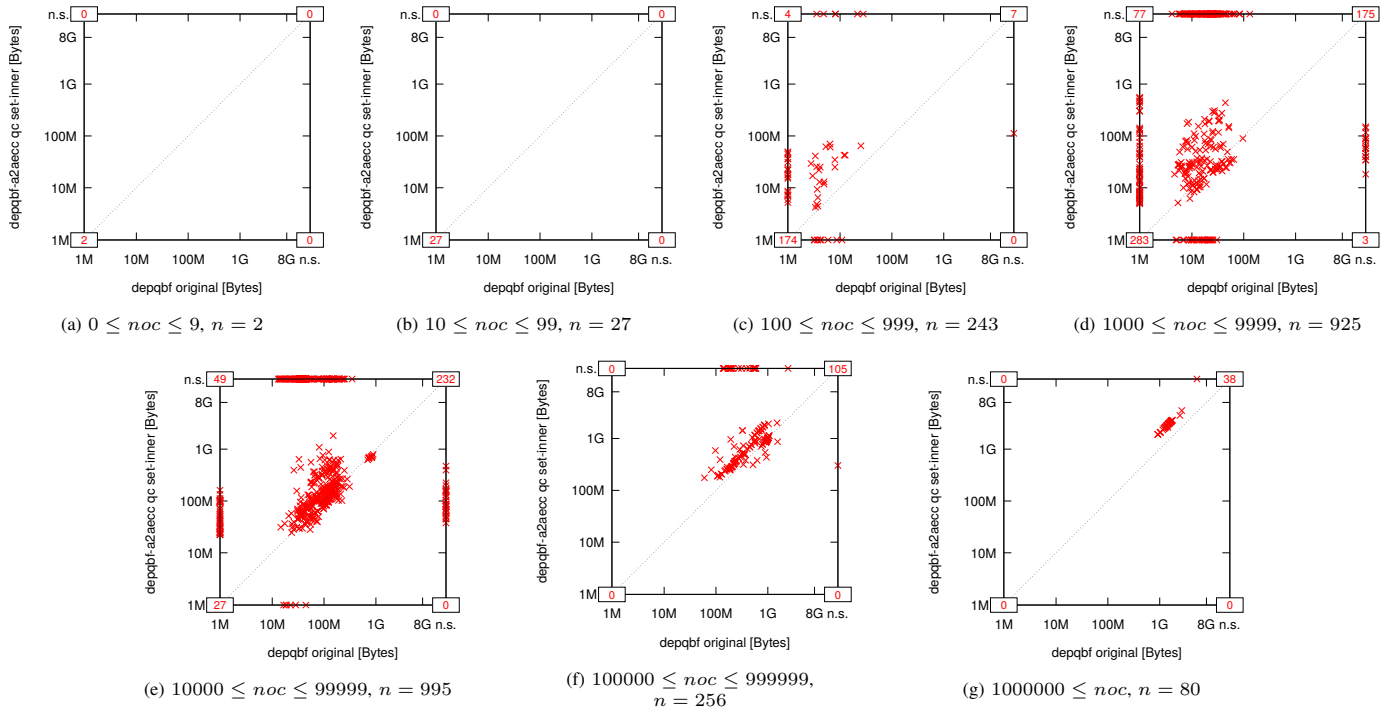


Fig. 1346: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode original partitioned by number of clauses (memory usage in Bytes).

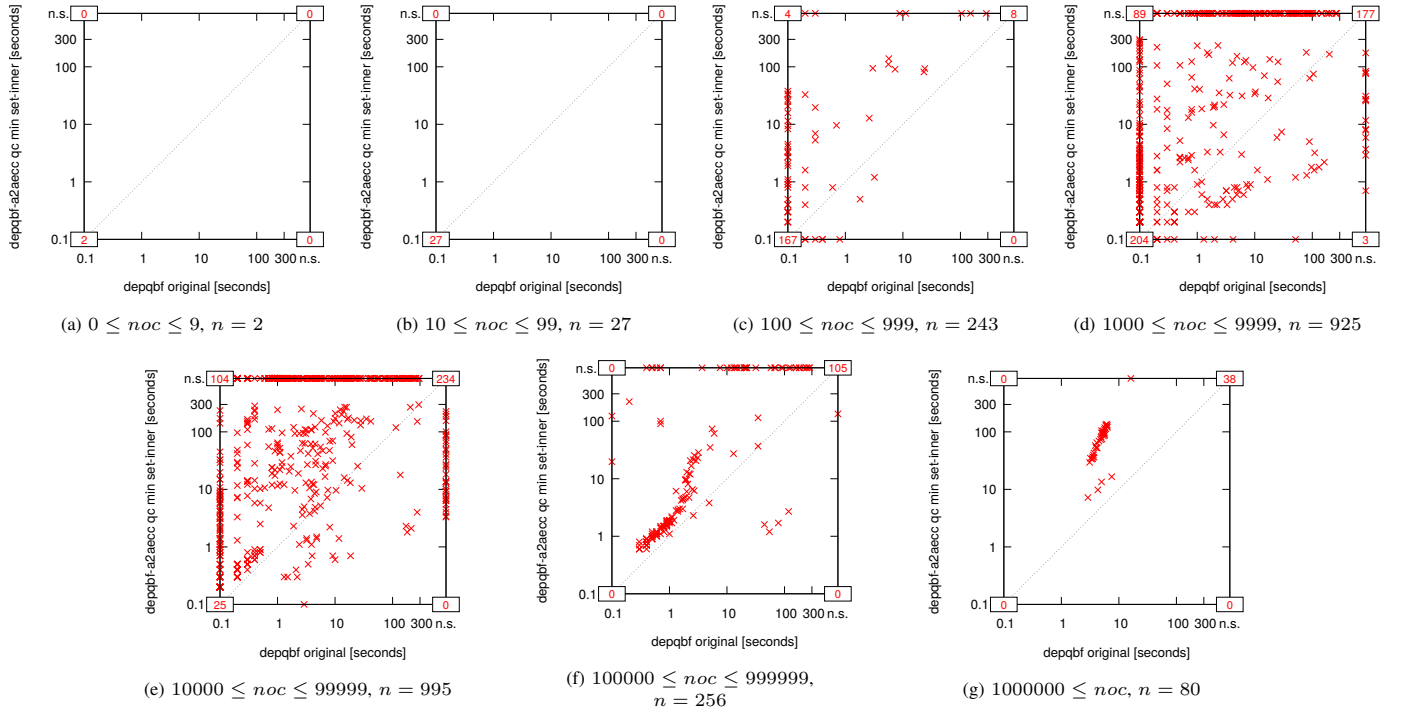


Fig. 1347: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode original partitioned by number of clauses (run time in seconds).

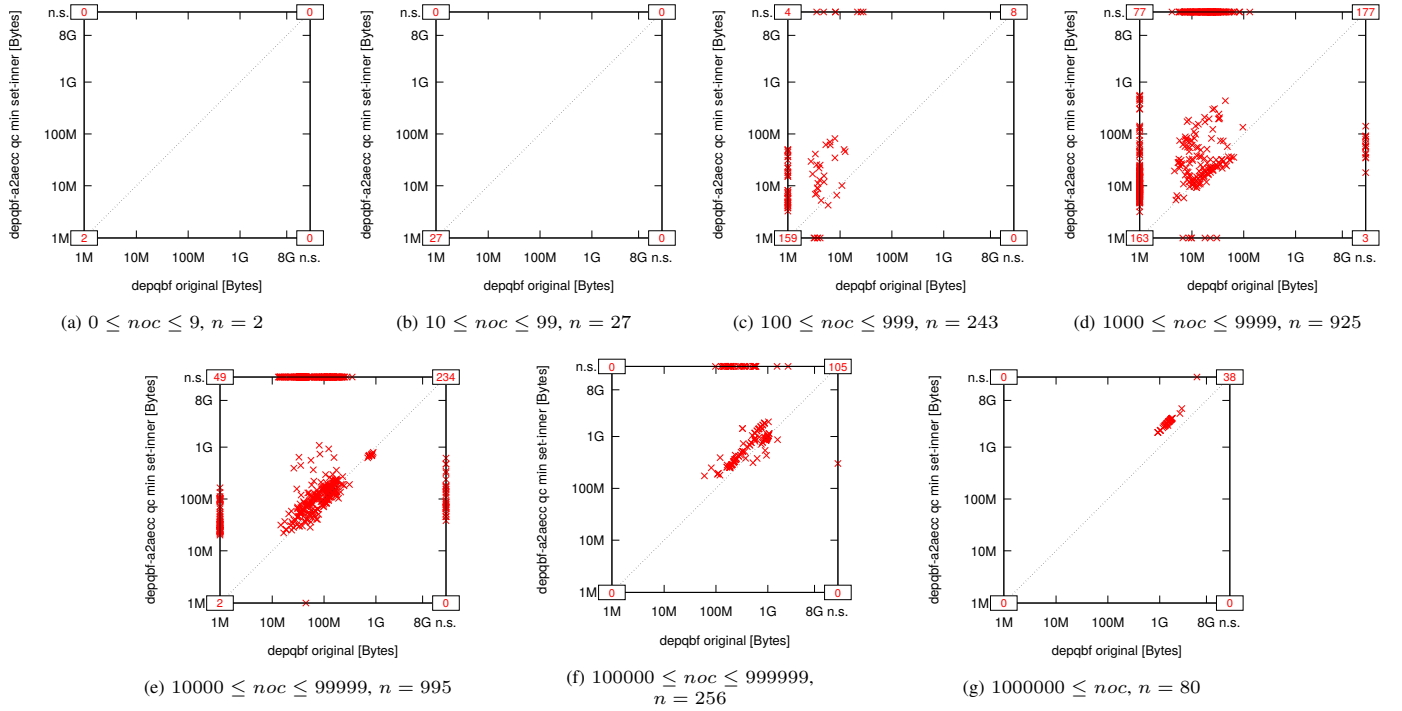


Fig. 1348: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode original partitioned by number of clauses (memory usage in Bytes).

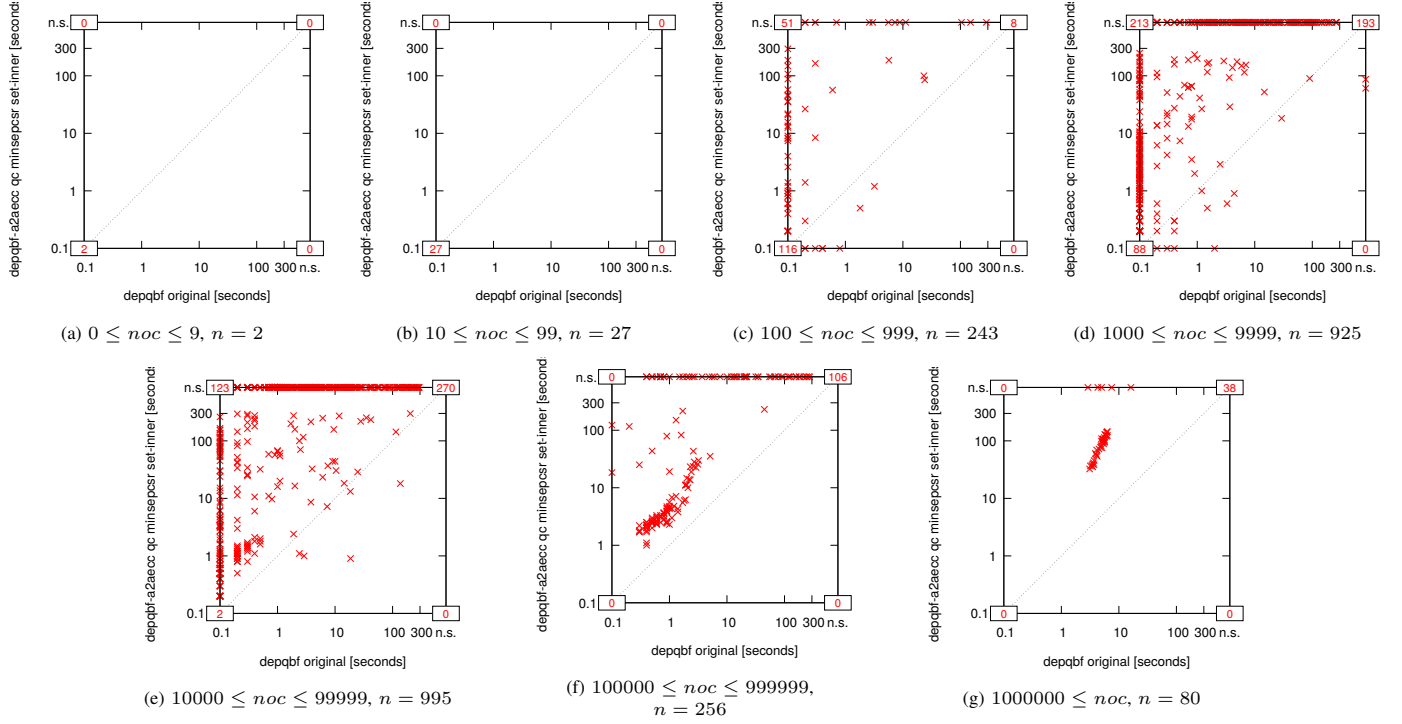


Fig. 1349: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode original partitioned by number of clauses (run time in seconds).

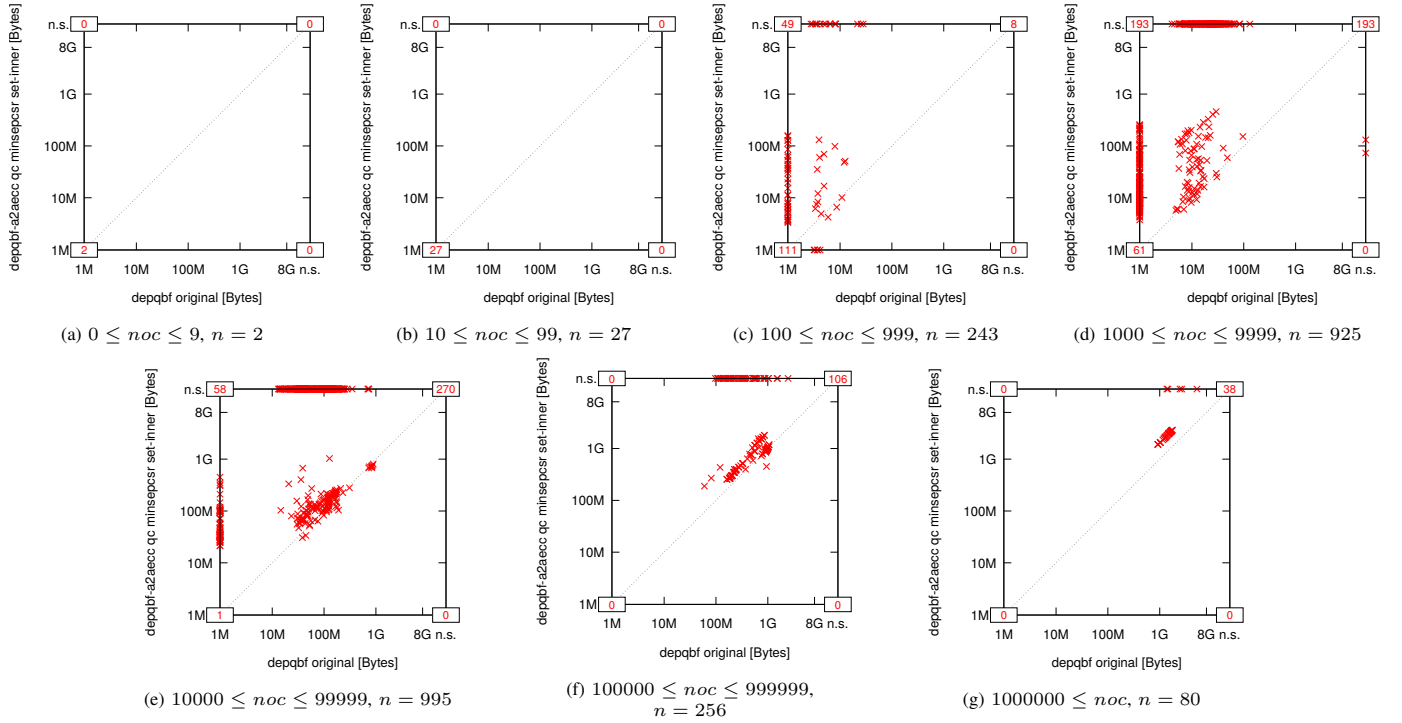


Fig. 1350: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode original partitioned by number of clauses (memory usage in Bytes).

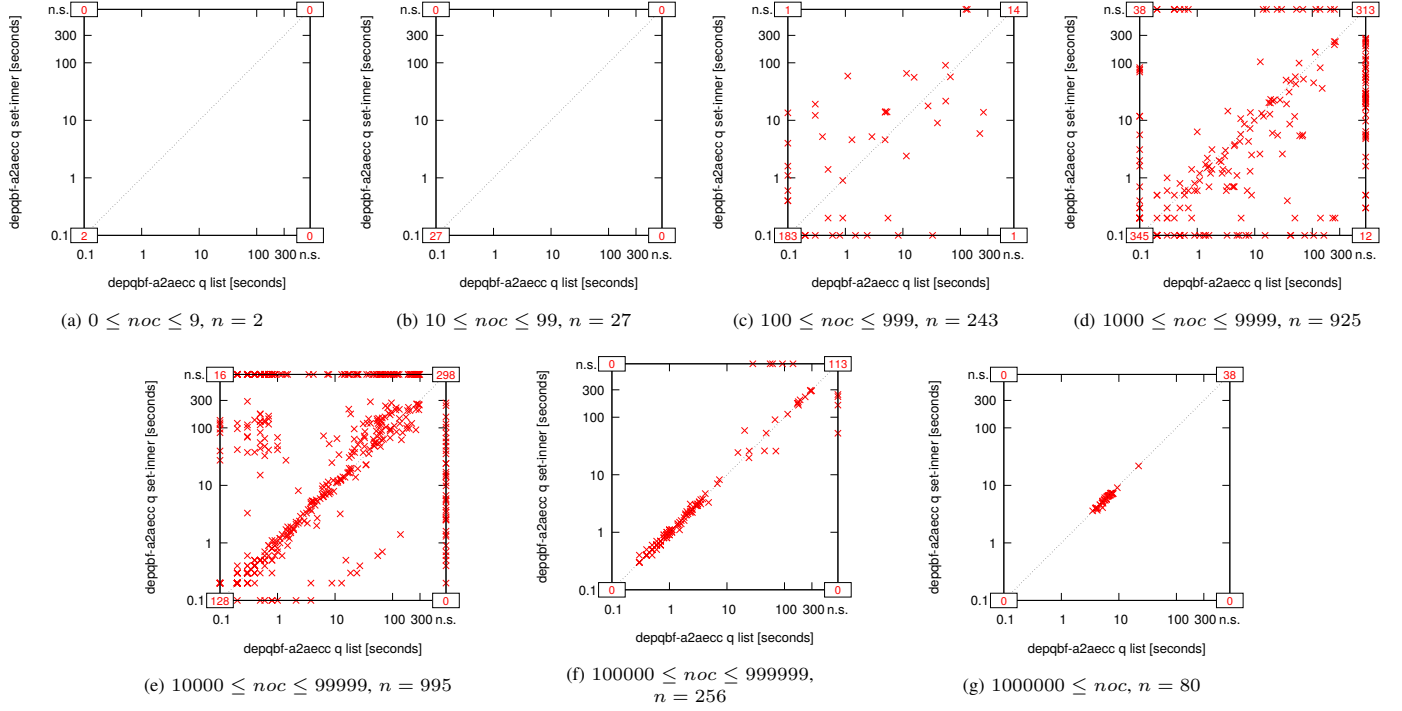


Fig. 1351: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q with set-inner versus list semantics partitioned by number of clauses (run time in seconds).

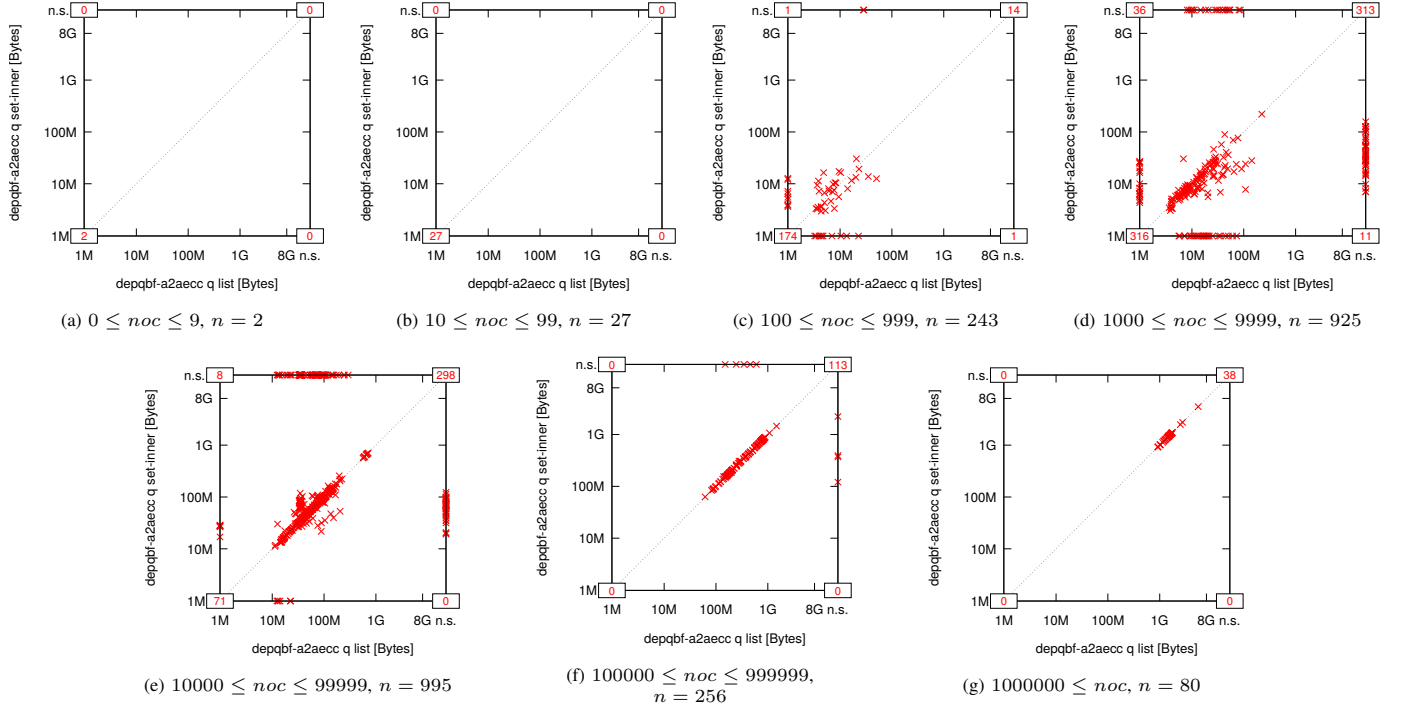


Fig. 1352: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q with set-inner versus list semantics partitioned by number of clauses (memory usage in Bytes).

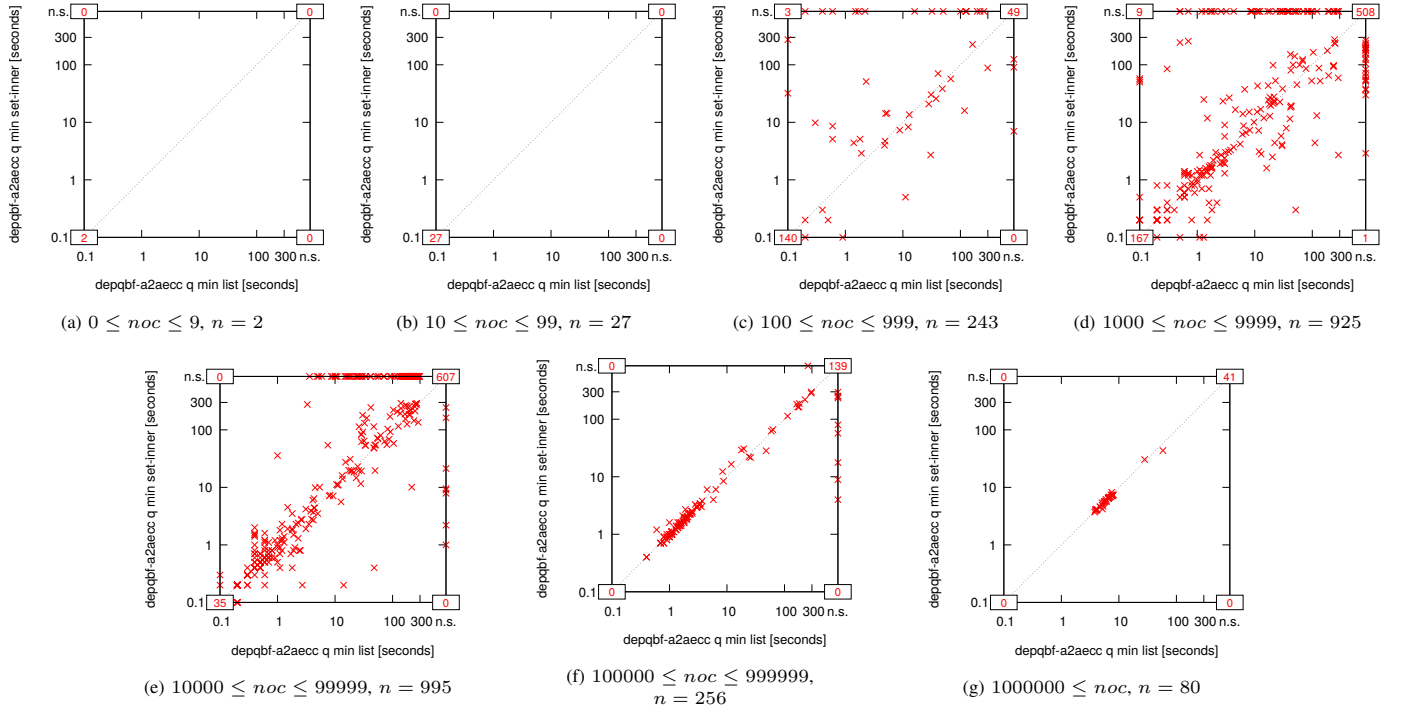


Fig. 1353: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q min with set-inner versus list semantics partitioned by number of clauses (run time in seconds).

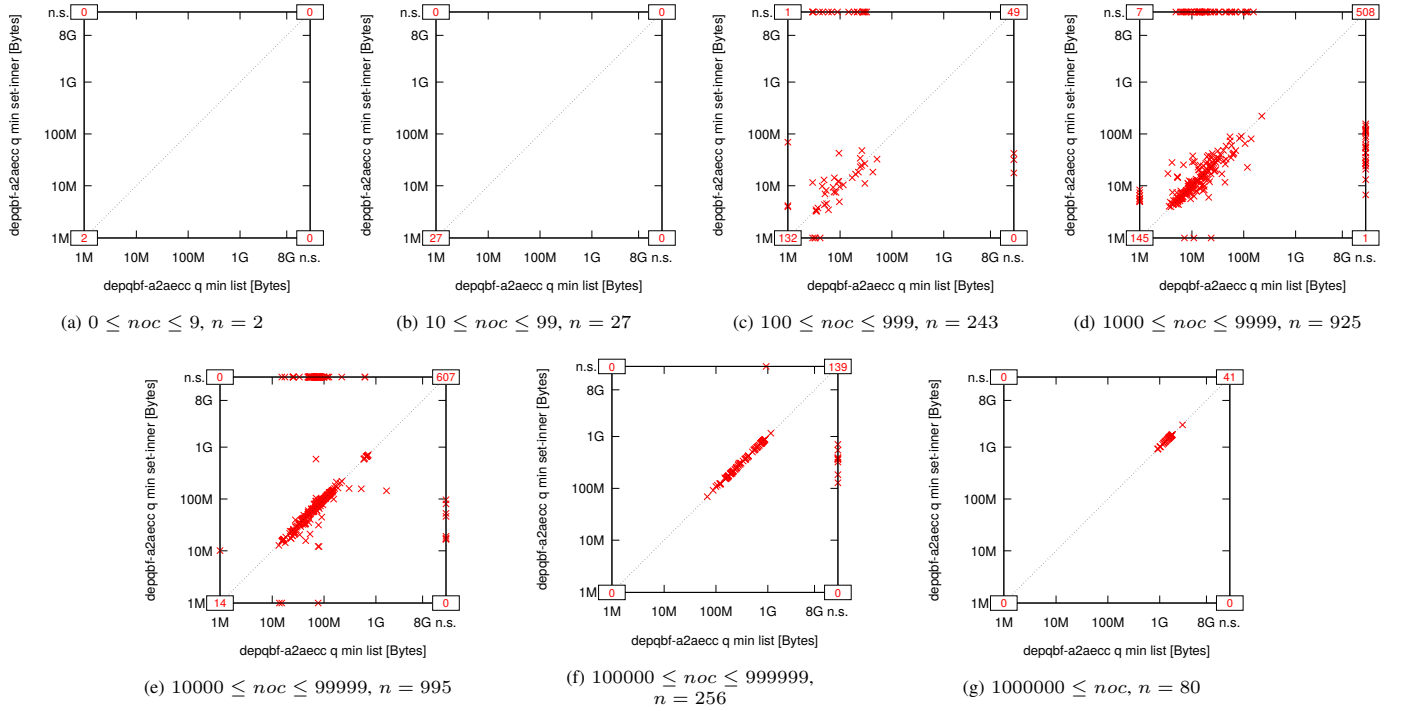


Fig. 1354: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q min with set-inner versus list semantics partitioned by number of clauses (memory usage in Bytes).

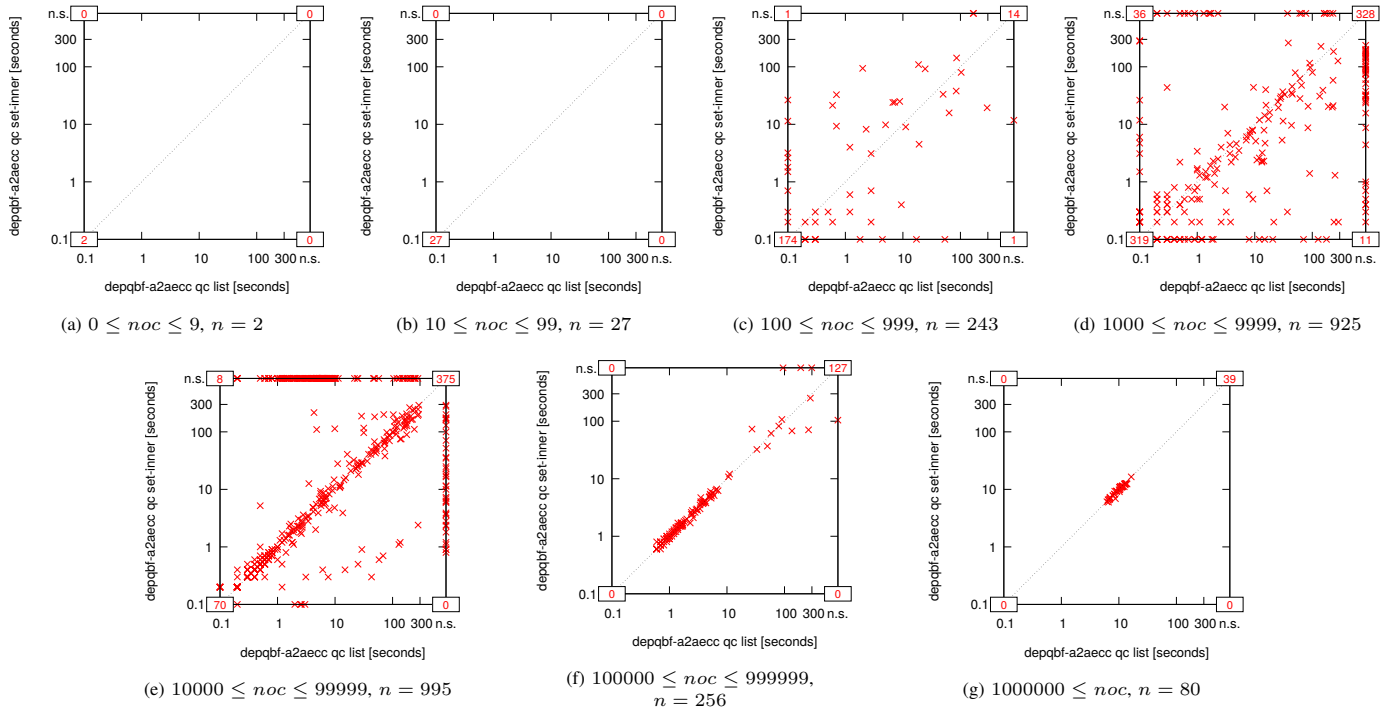


Fig. 1355: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc with set-inner versus list semantics partitioned by number of clauses (run time in seconds).

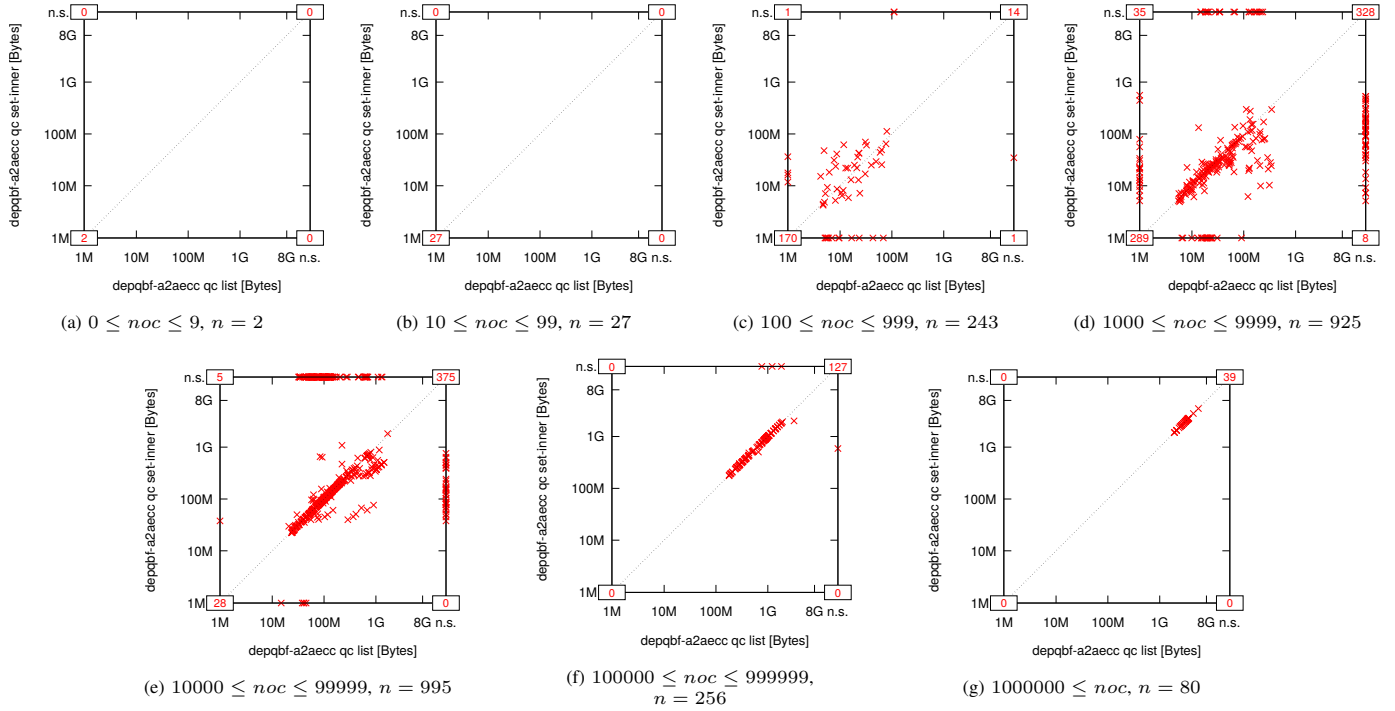


Fig. 1356: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc with set-inner versus list semantics partitioned by number of clauses (memory usage in Bytes).

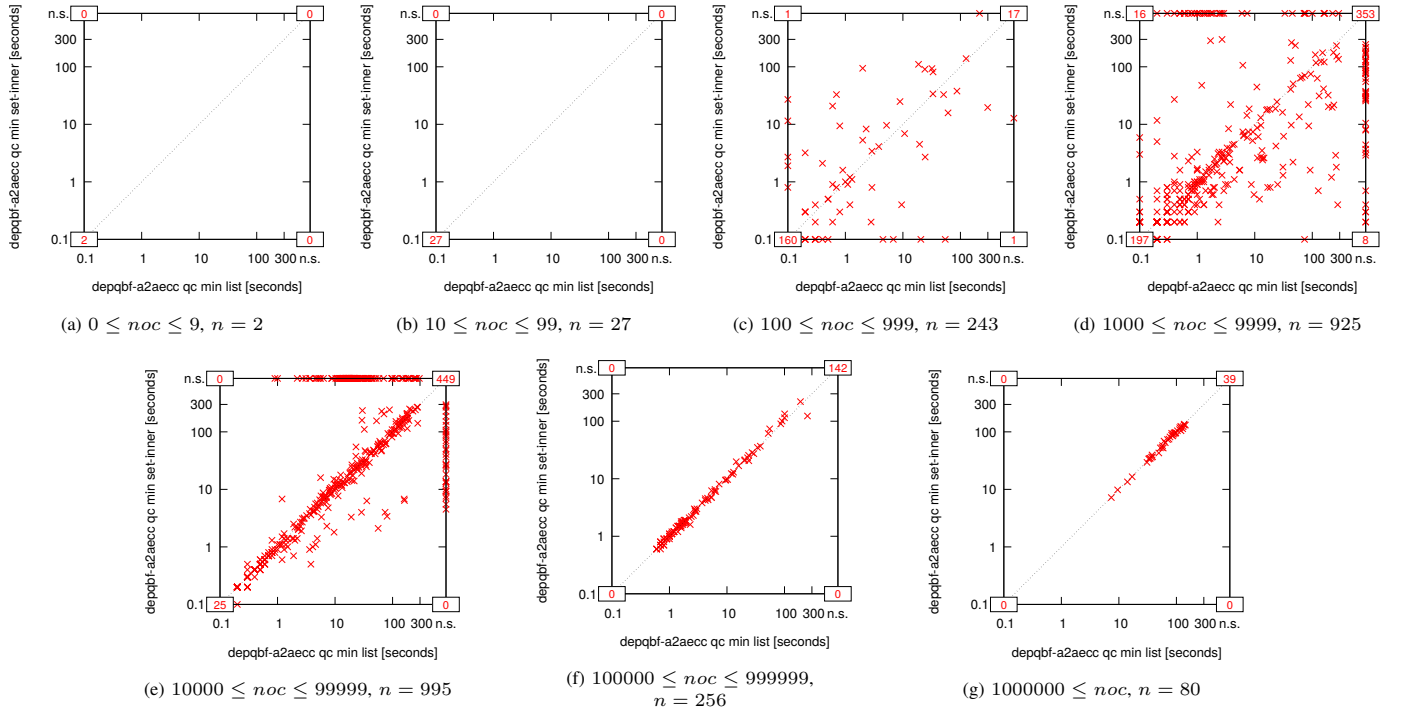


Fig. 1357: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min with set-inner versus list semantics partitioned by number of clauses (run time in seconds).

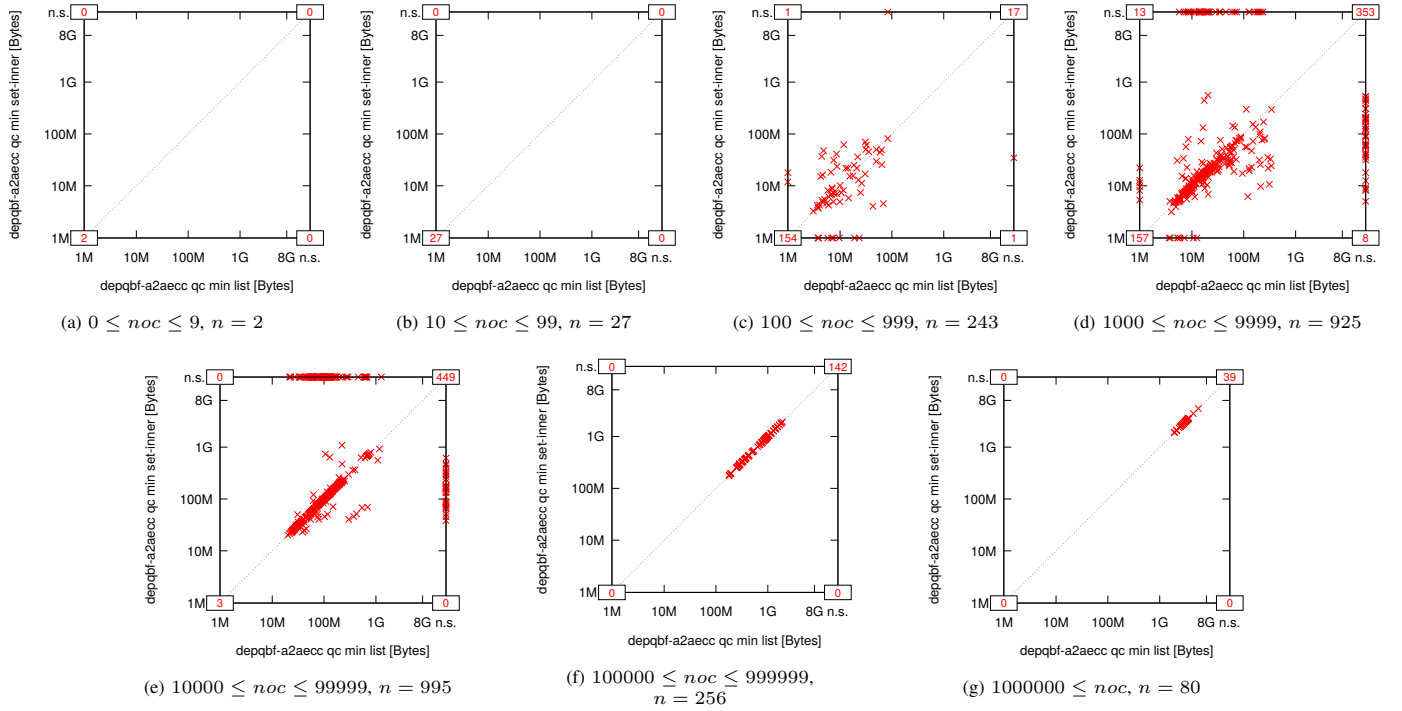


Fig. 1358: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min with set-inner versus list semantics partitioned by number of clauses (memory usage in Bytes).

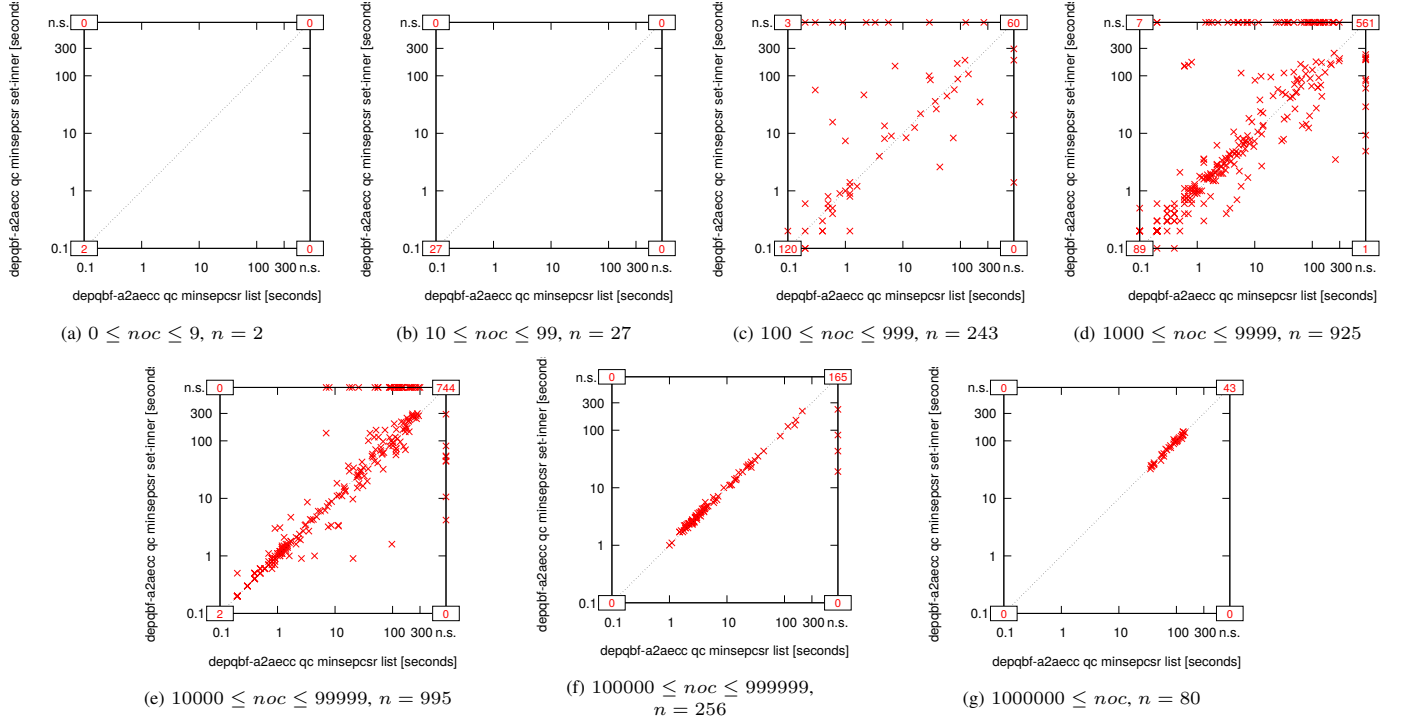


Fig. 1359: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr with set-inner versus list semantics partitioned by number of clauses (run time in seconds).

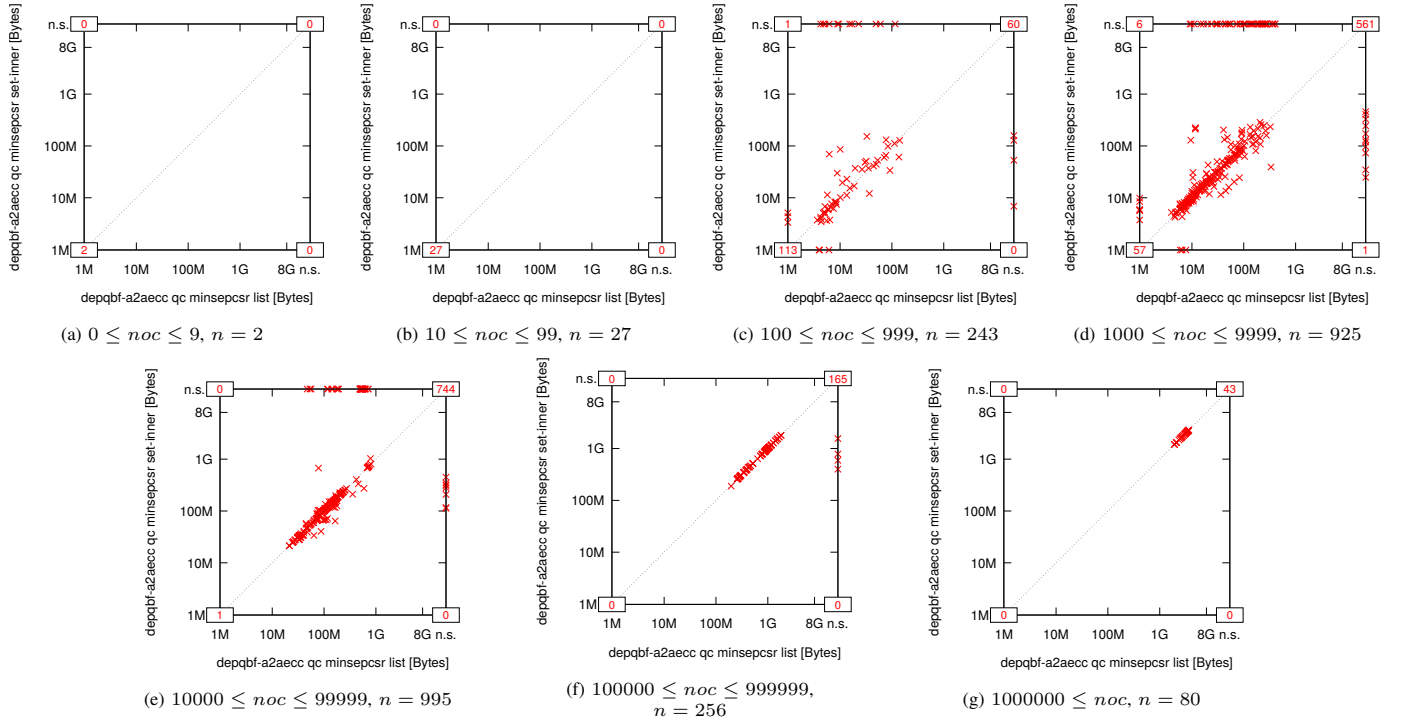


Fig. 1360: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr with set-inner versus list semantics partitioned by number of clauses (memory usage in Bytes).

E. Partitioned by Maximum Variable Index

This subsection shows figures of the overhead of extracting and minimizing unsatisfiable cores with subfigures for partitions of the benchmarks according to their maximum variable index.

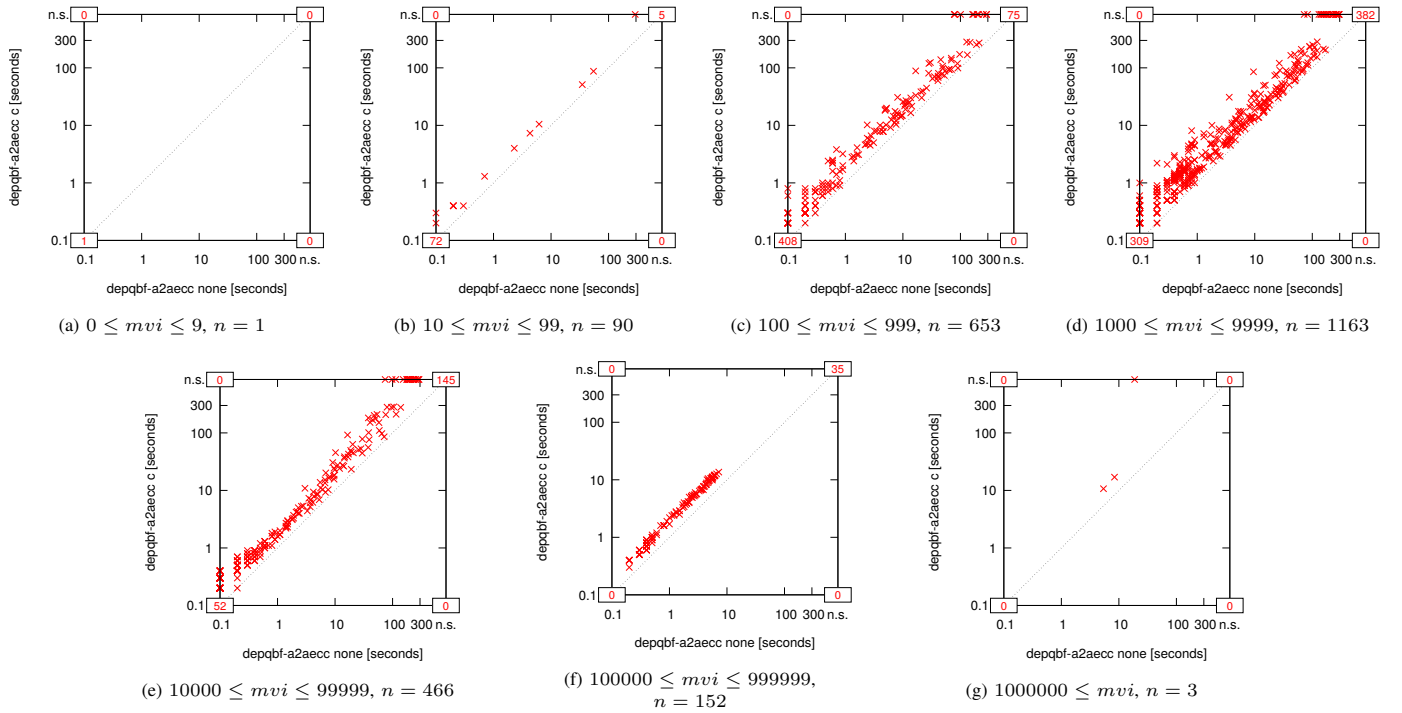


Fig. 1361: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c versus mode none partitioned by maximum variable index (run time in seconds).

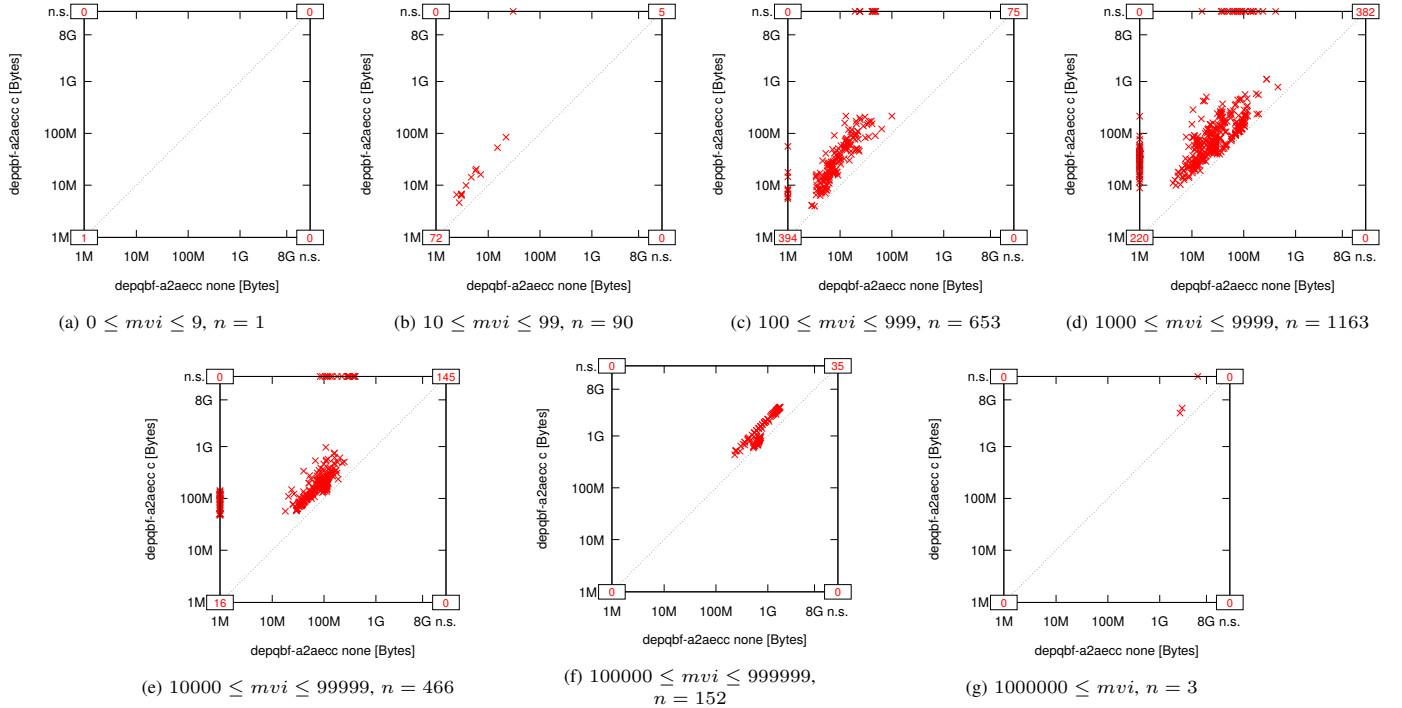


Fig. 1362: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c versus mode none partitioned by maximum variable index (memory usage in Bytes).

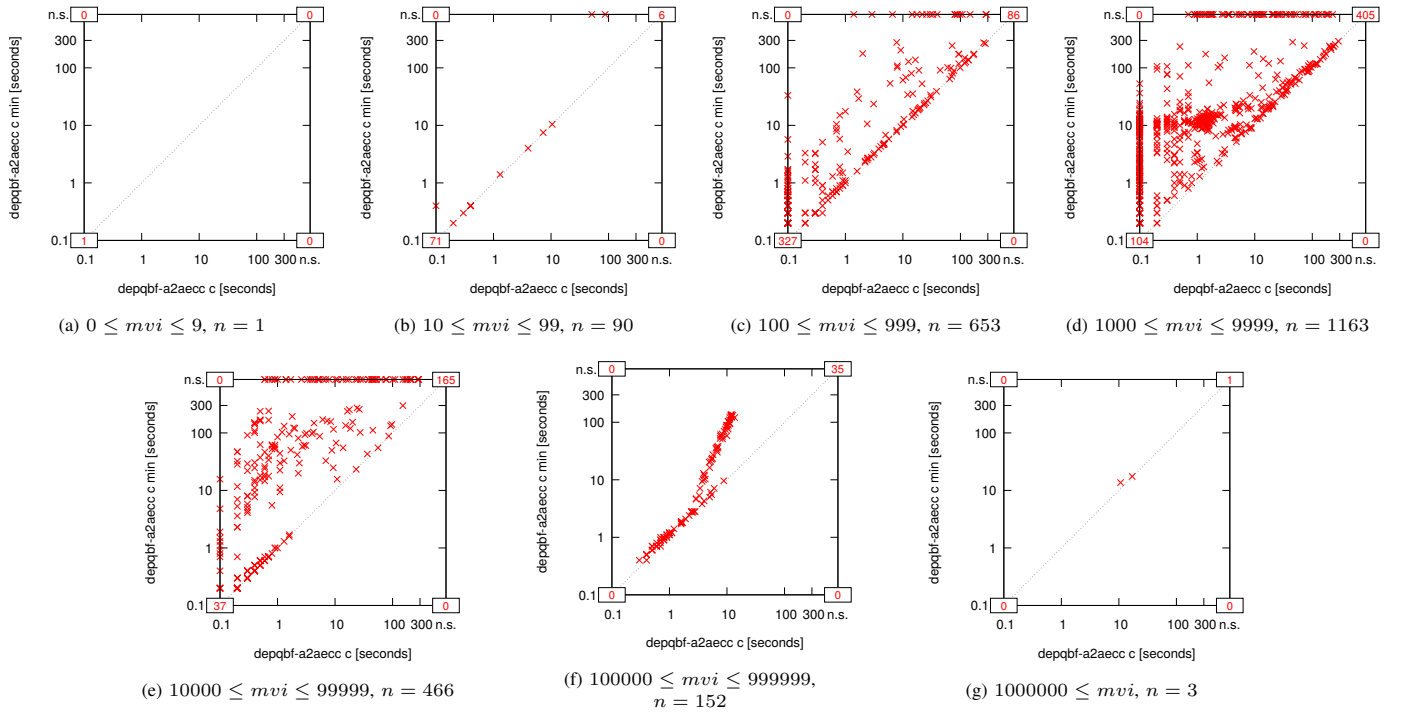


Fig. 1363: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode c partitioned by maximum variable index (run time in seconds).

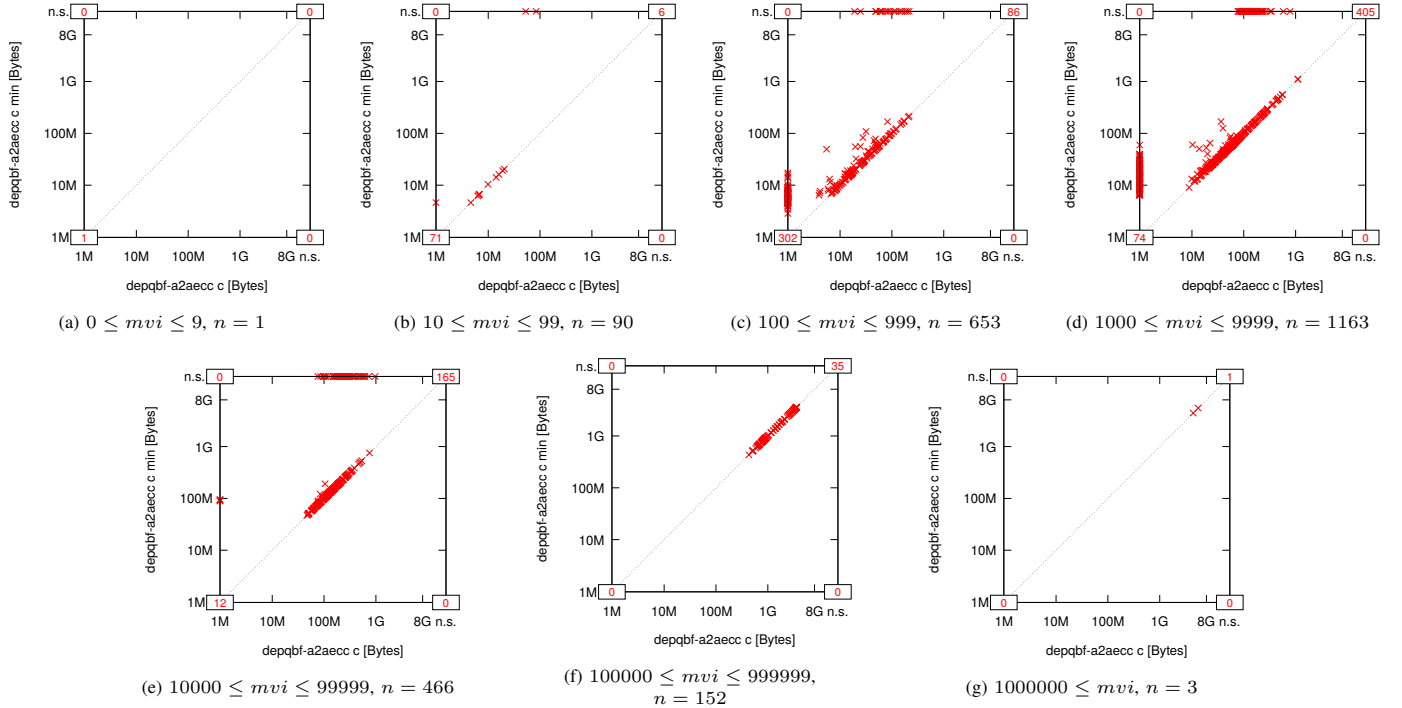


Fig. 1364: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode c partitioned by maximum variable index (memory usage in Bytes).

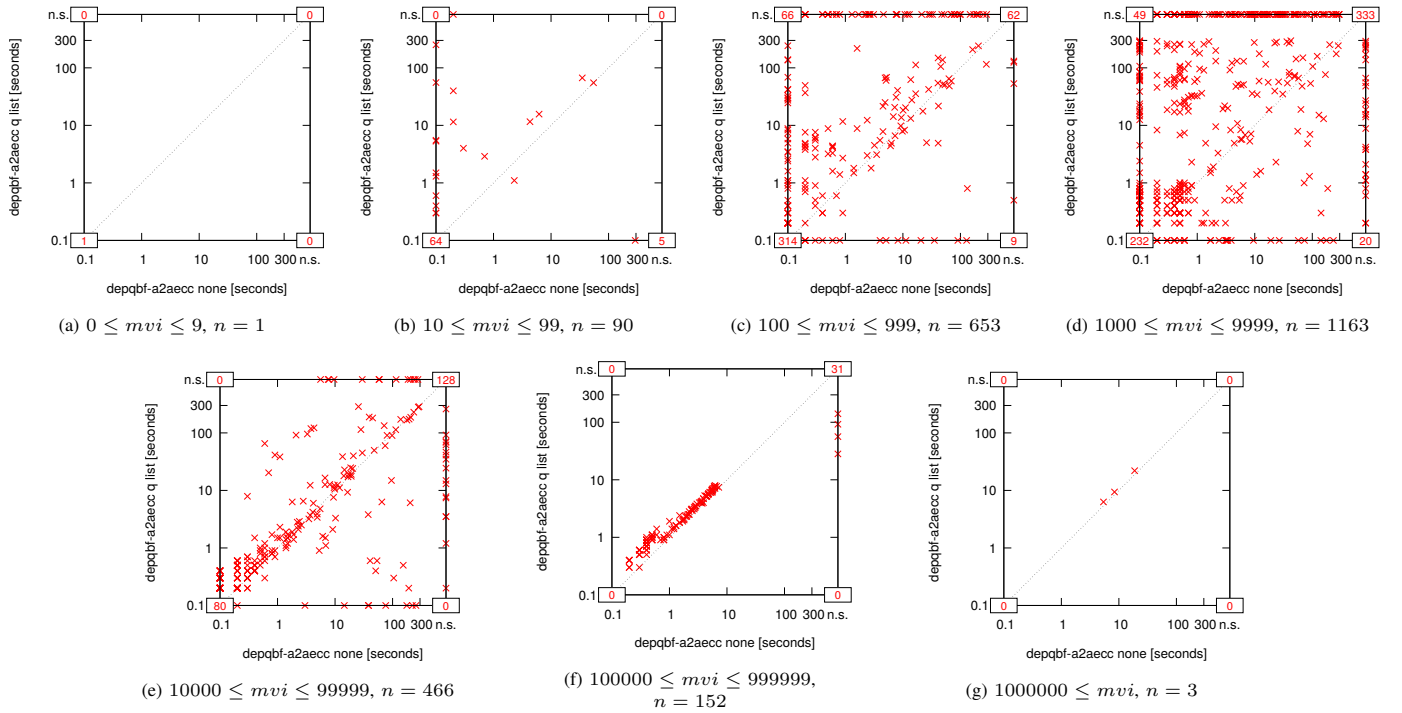


Fig. 1365: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode none partitioned by maximum variable index (run time in seconds).

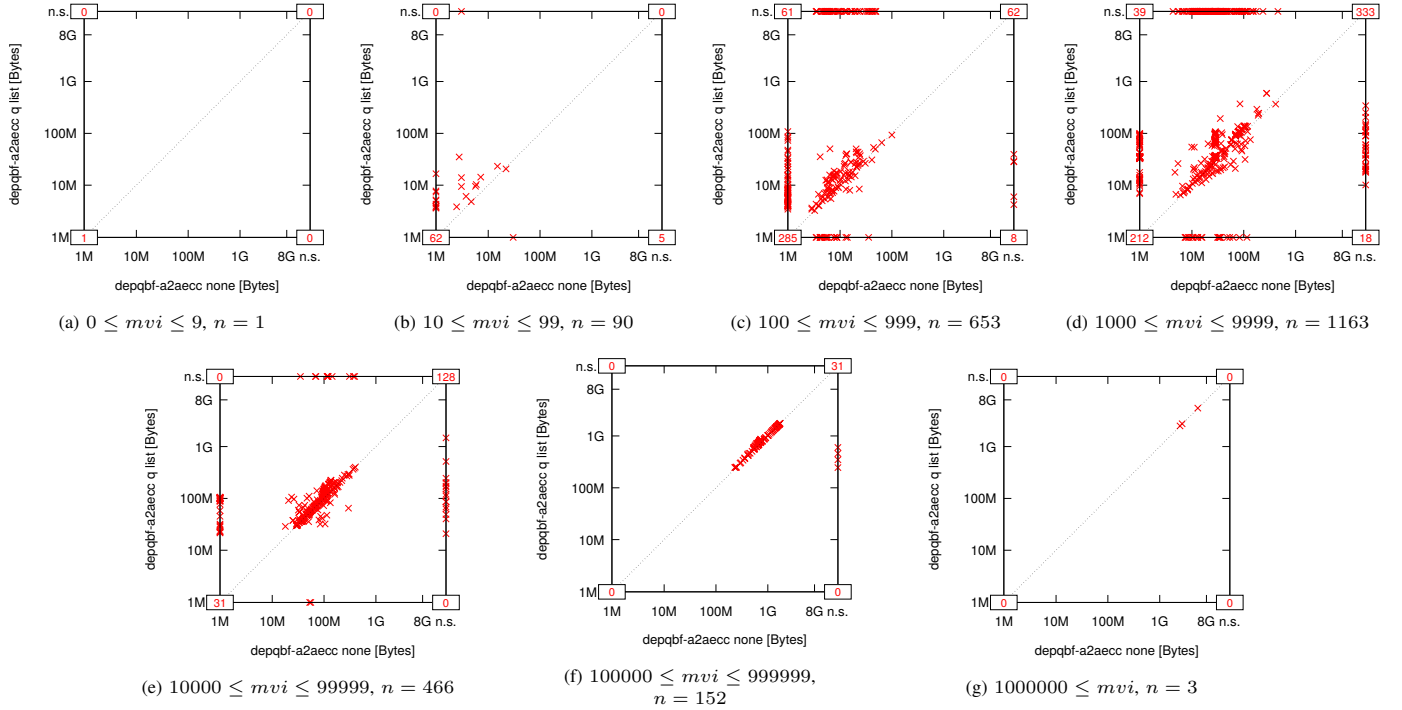


Fig. 1366: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode none partitioned by maximum variable index (memory usage in Bytes).

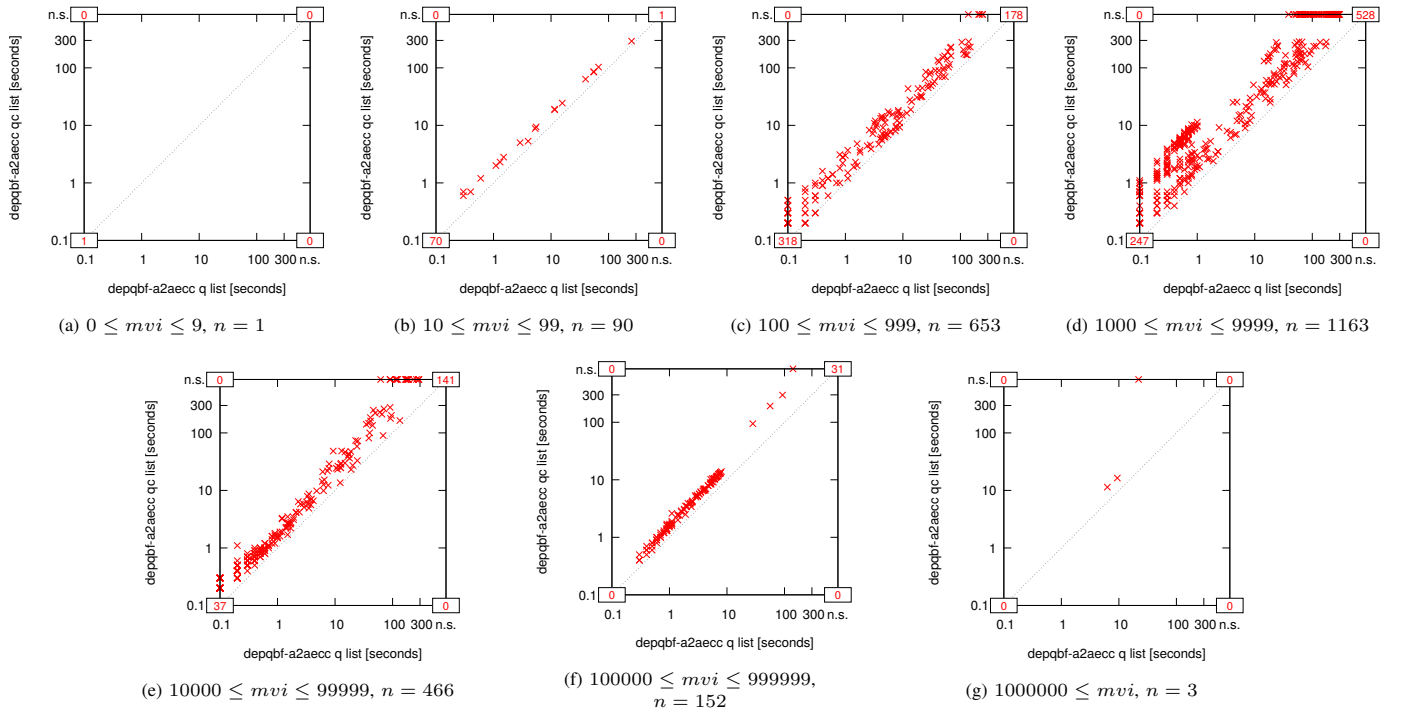


Fig. 1367: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode q list partitioned by maximum variable index (run time in seconds).

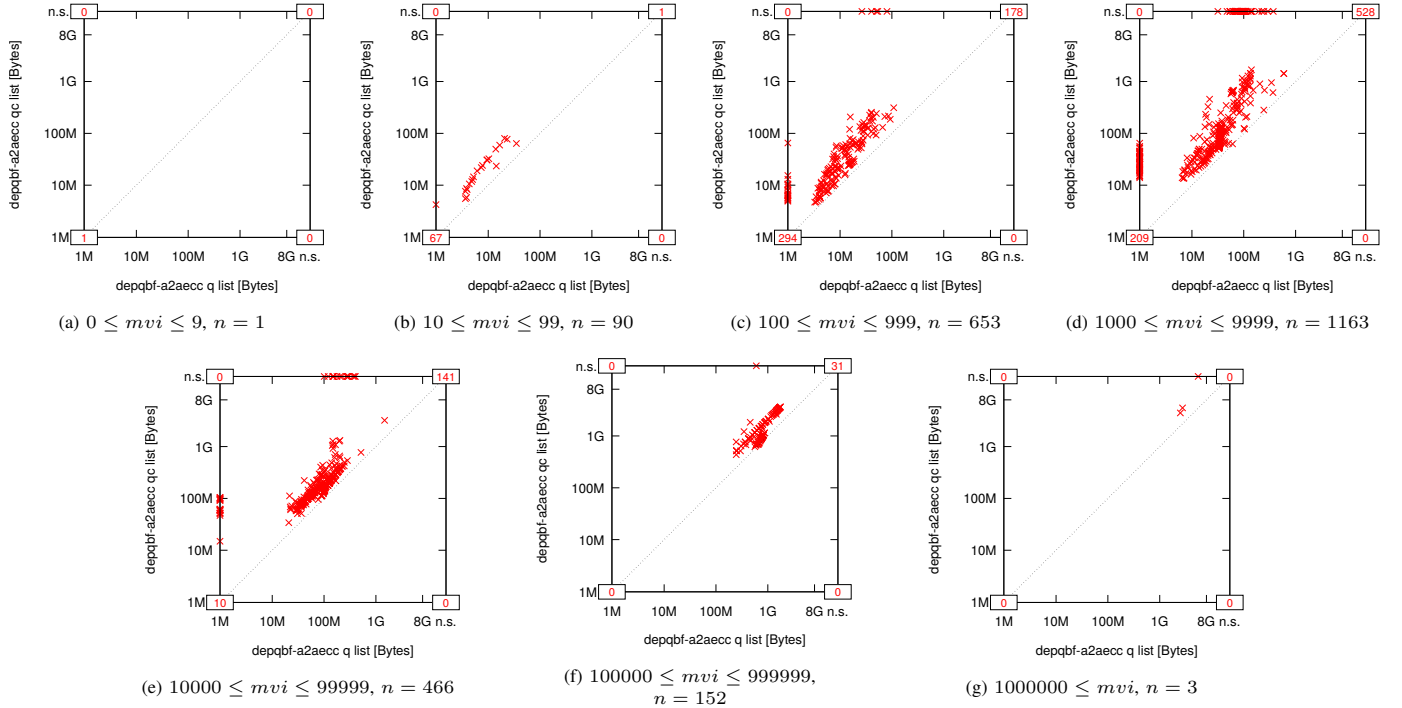


Fig. 1368: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode q list partitioned by maximum variable index (memory usage in Bytes).

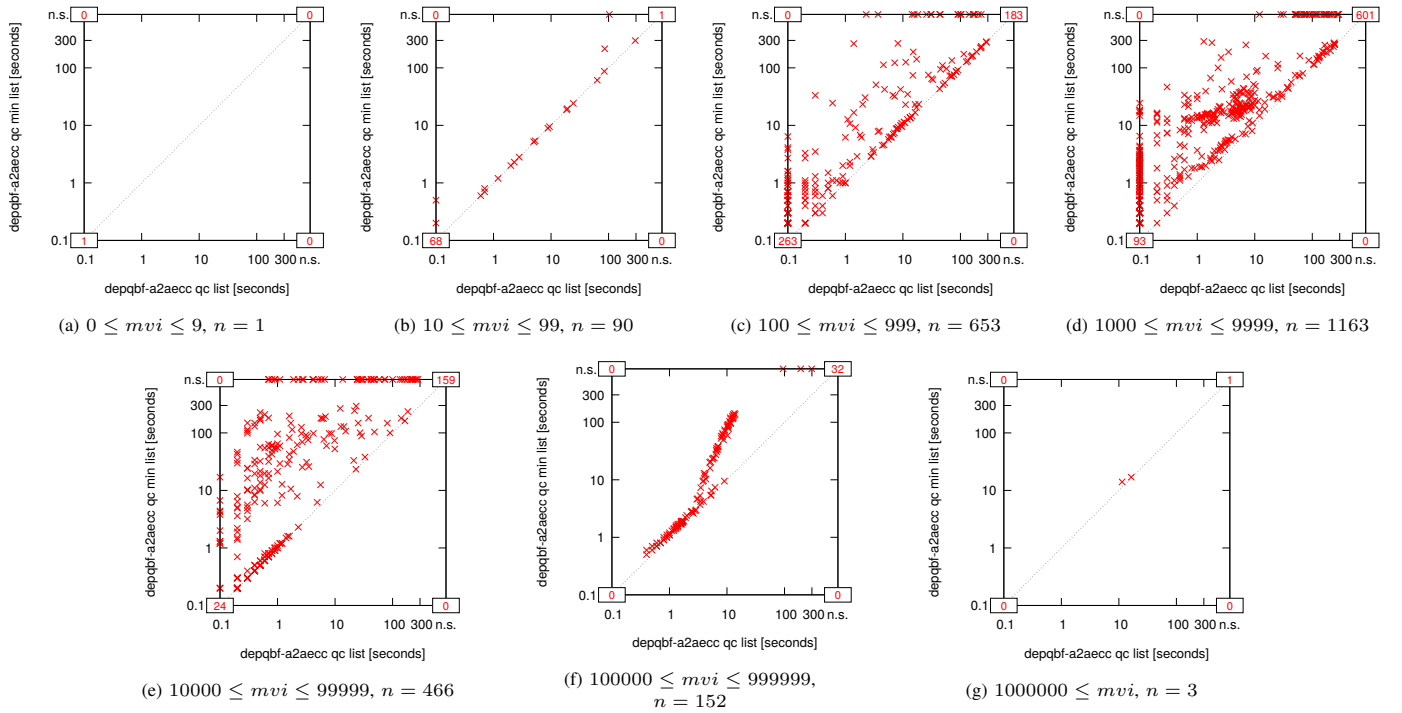


Fig. 1369: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode qc list partitioned by maximum variable index (run time in seconds).

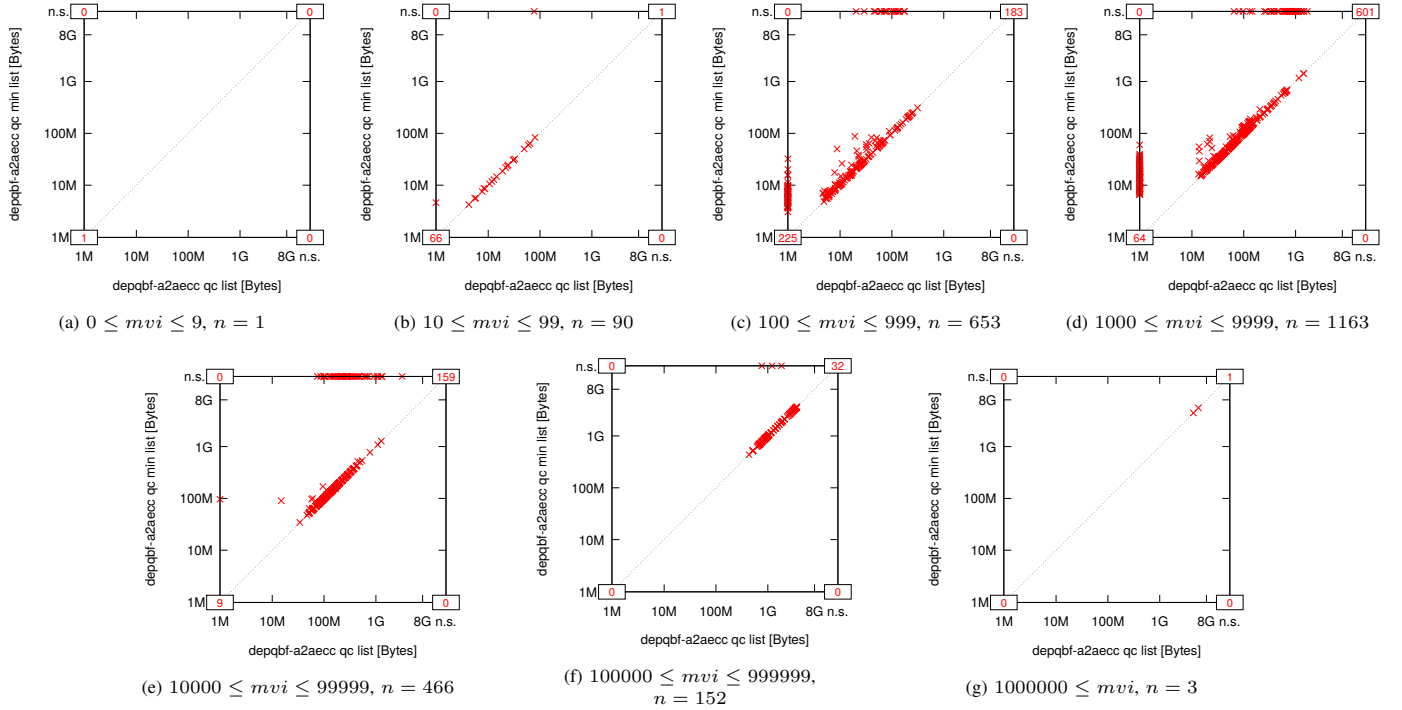


Fig. 1370: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode qc list partitioned by maximum variable index (memory usage in Bytes).

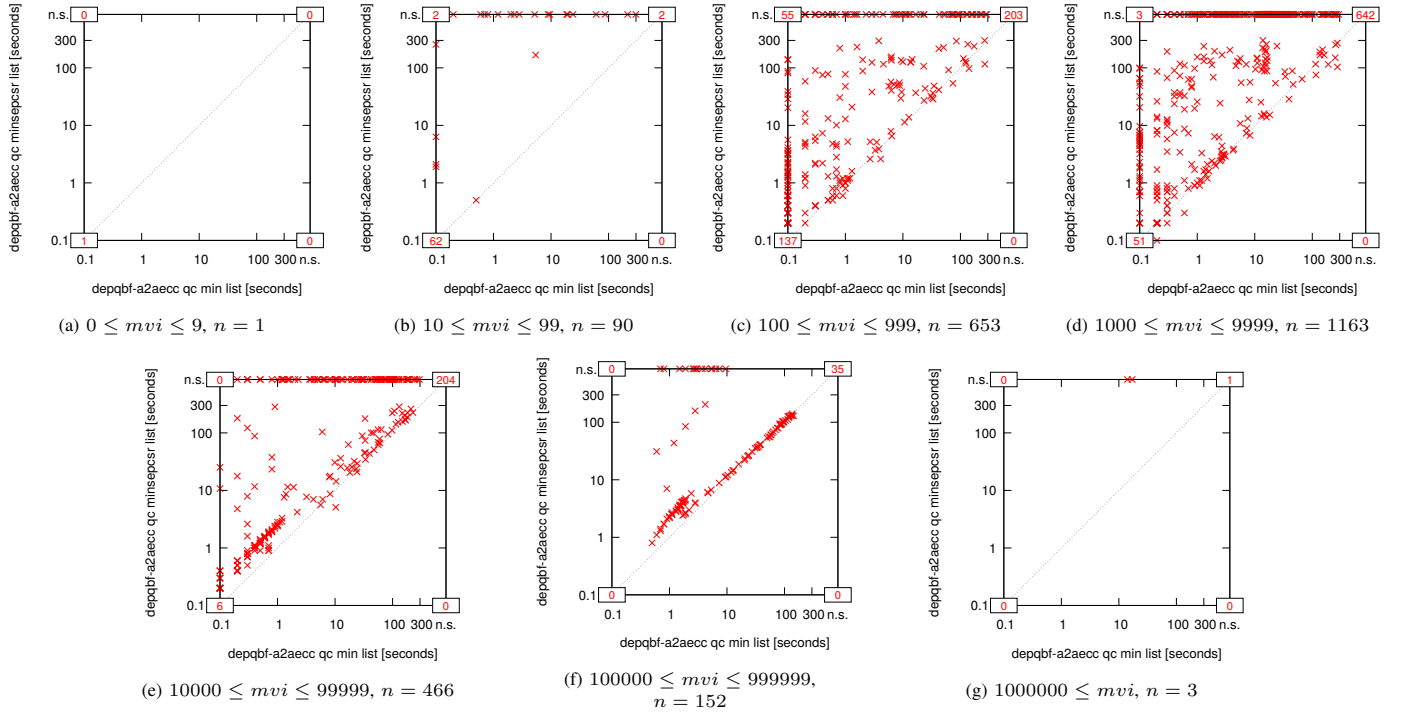


Fig. 1371: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode qc min list partitioned by maximum variable index (run time in seconds).

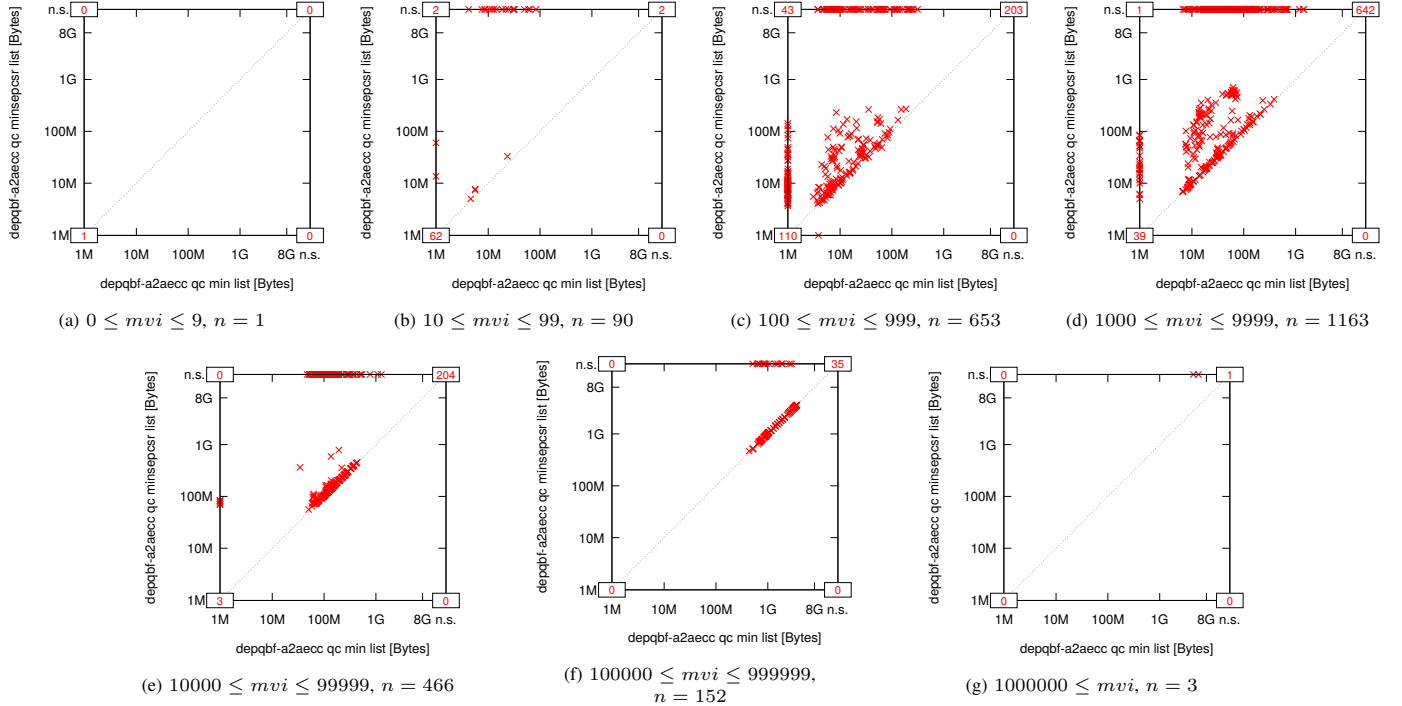


Fig. 1372: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode qc min list partitioned by maximum variable index (memory usage in Bytes).

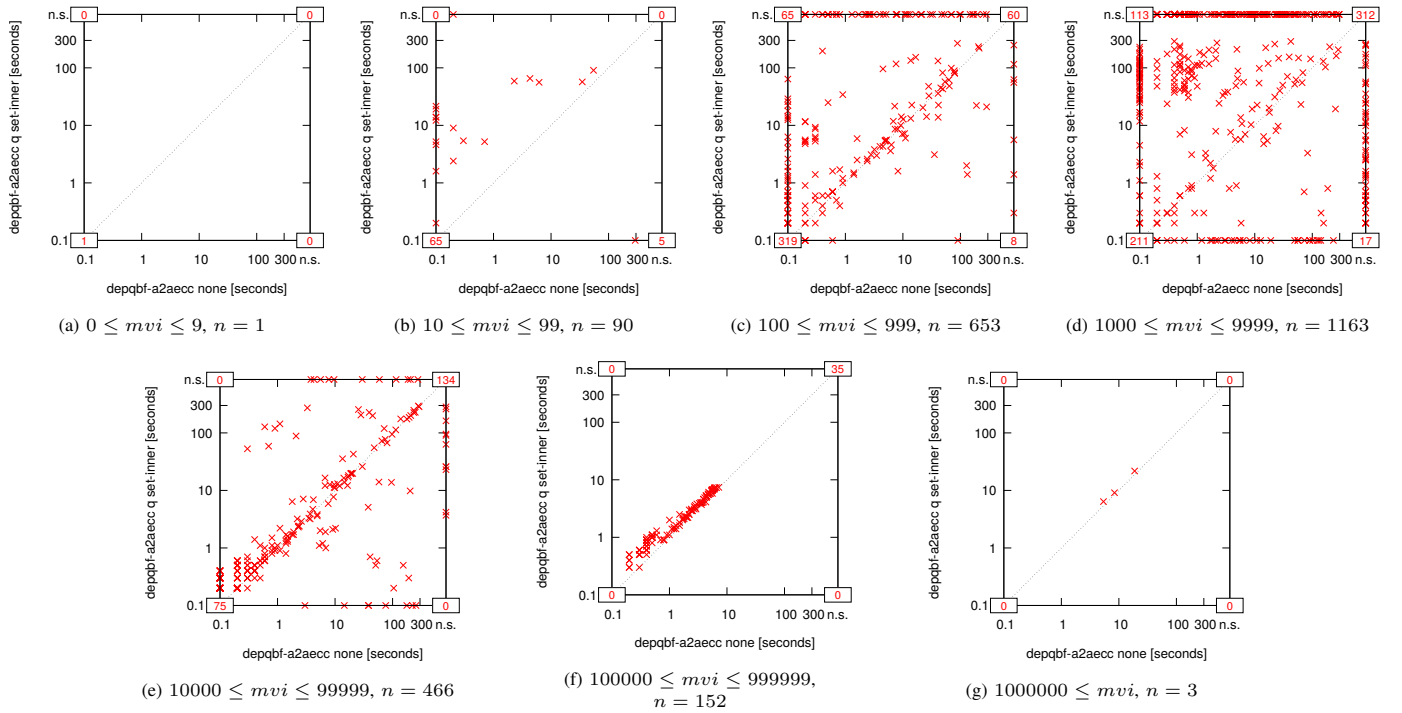


Fig. 1373: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode none partitioned by maximum variable index (run time in seconds).

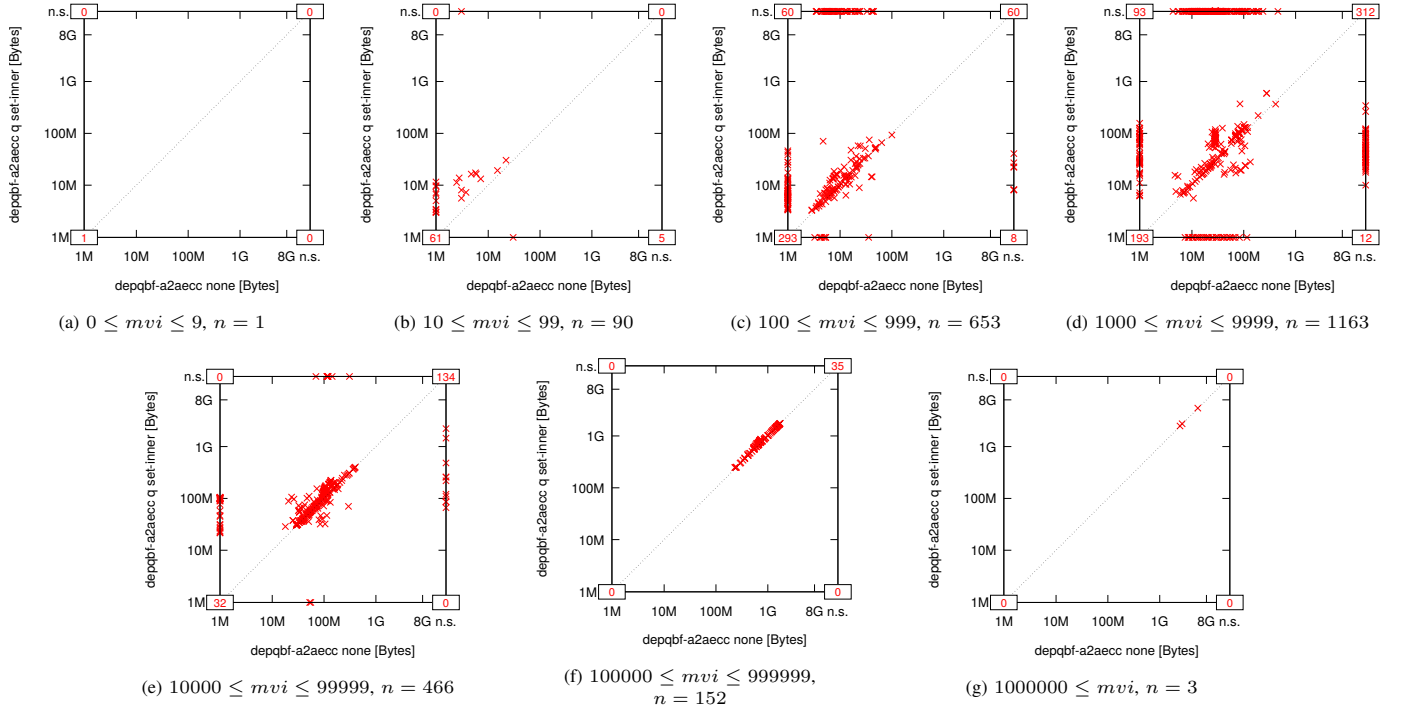


Fig. 1374: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode none partitioned by maximum variable index (memory usage in Bytes).

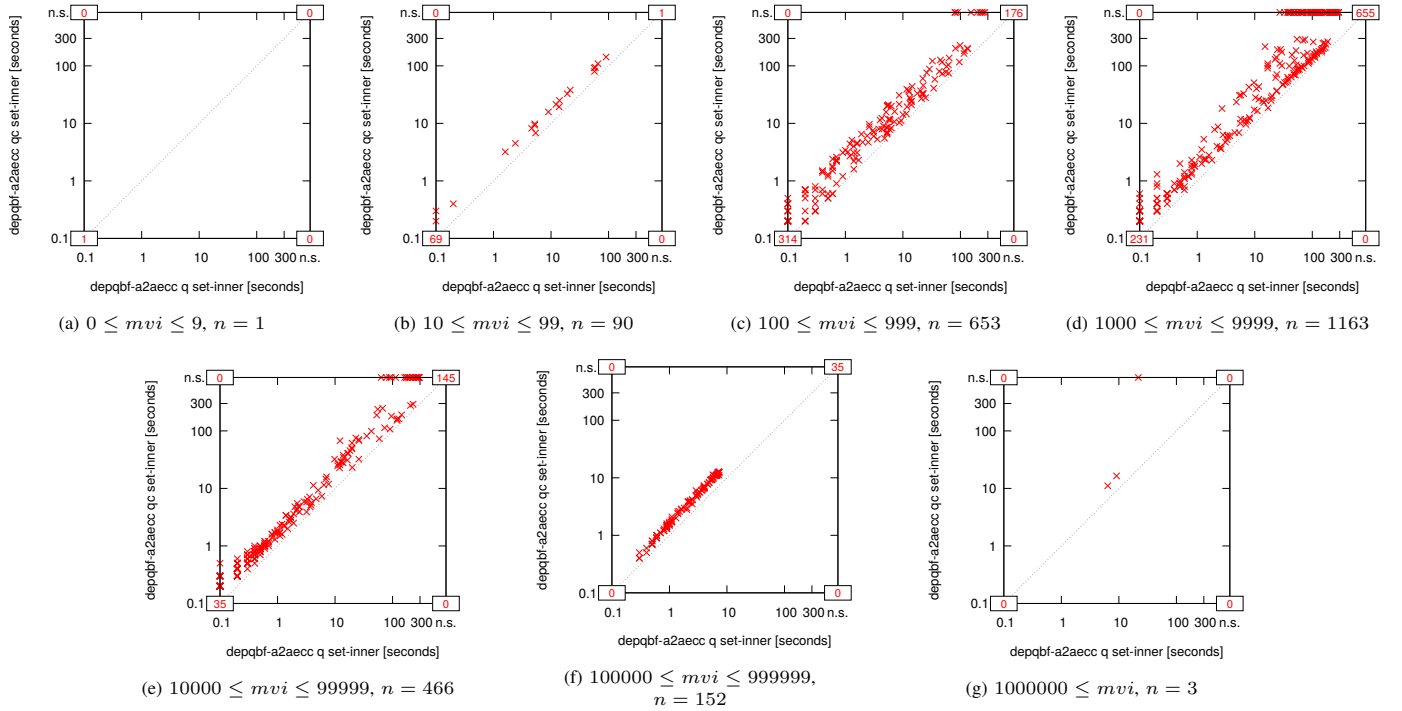


Fig. 1375: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode q set-inner partitioned by maximum variable index (run time in seconds).

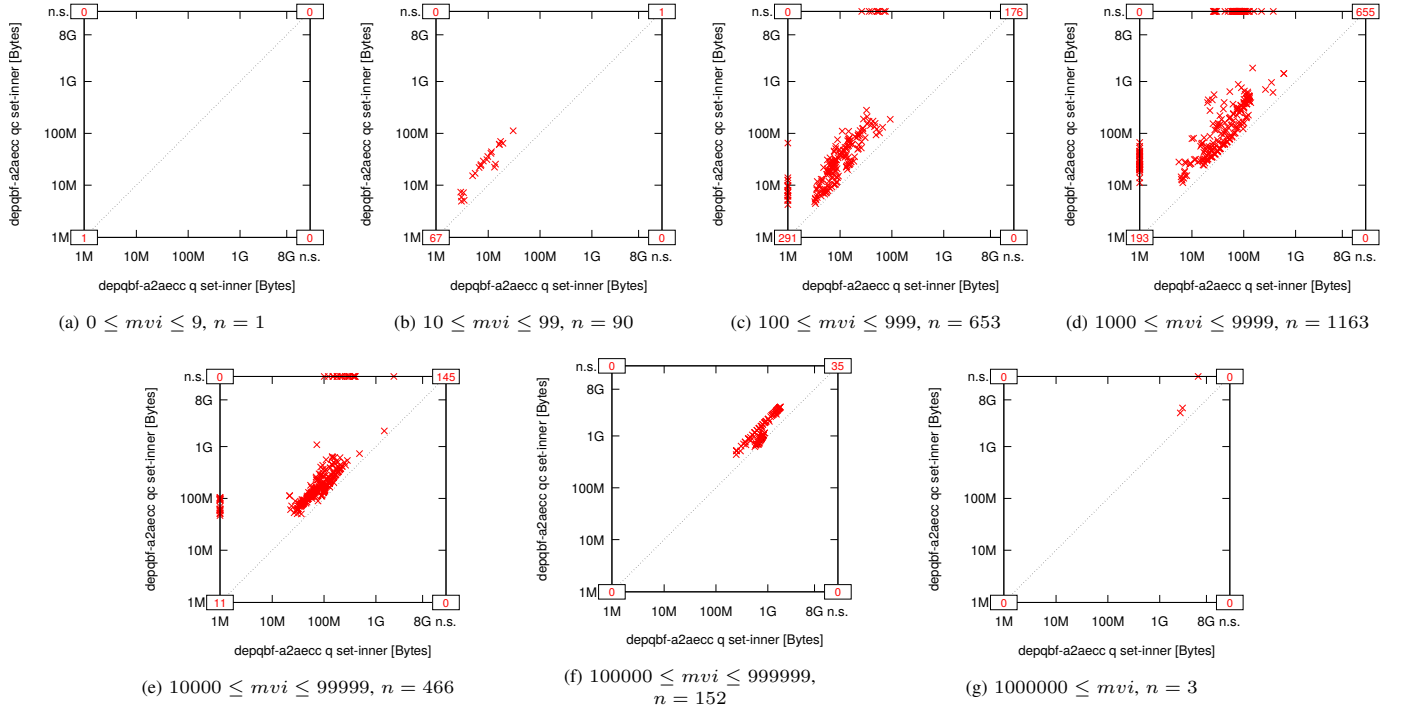


Fig. 1376: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode q set-inner partitioned by maximum variable index (memory usage in Bytes).

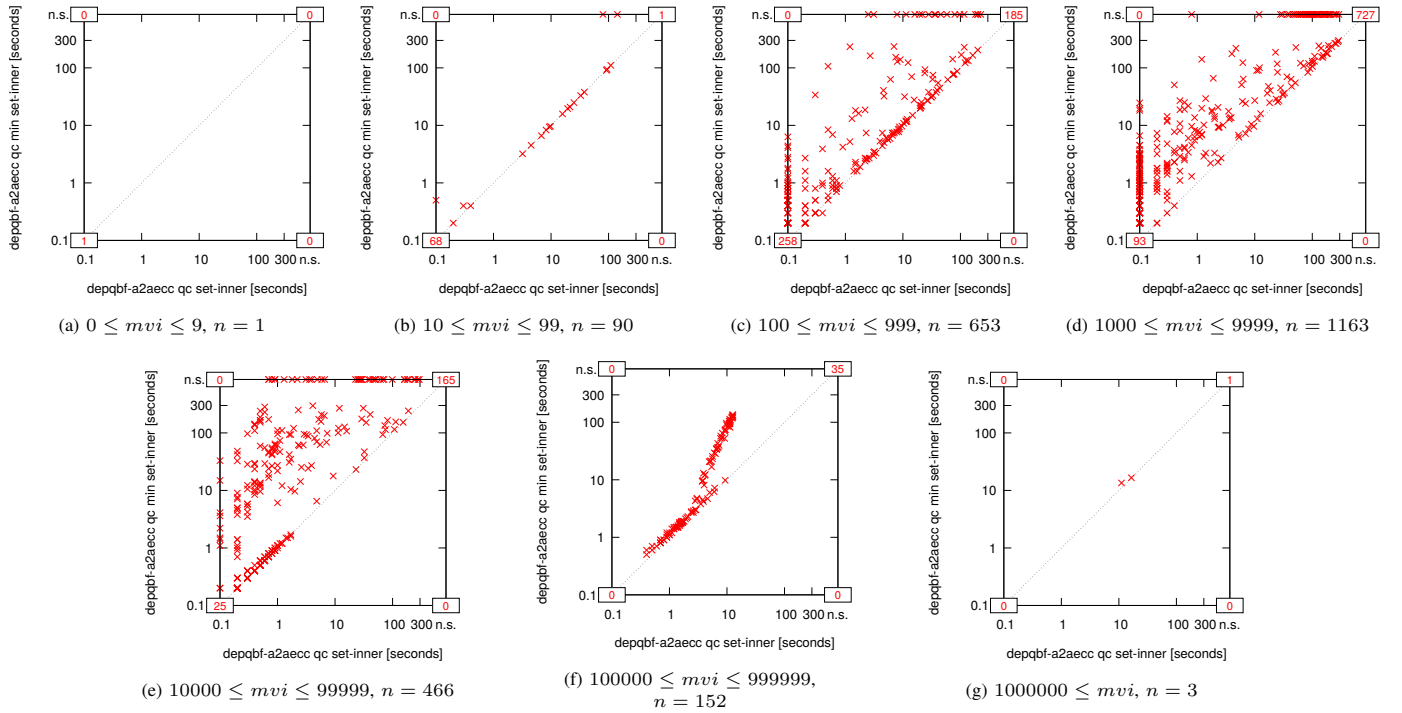


Fig. 1377: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode qc set-inner partitioned by maximum variable index (run time in seconds).

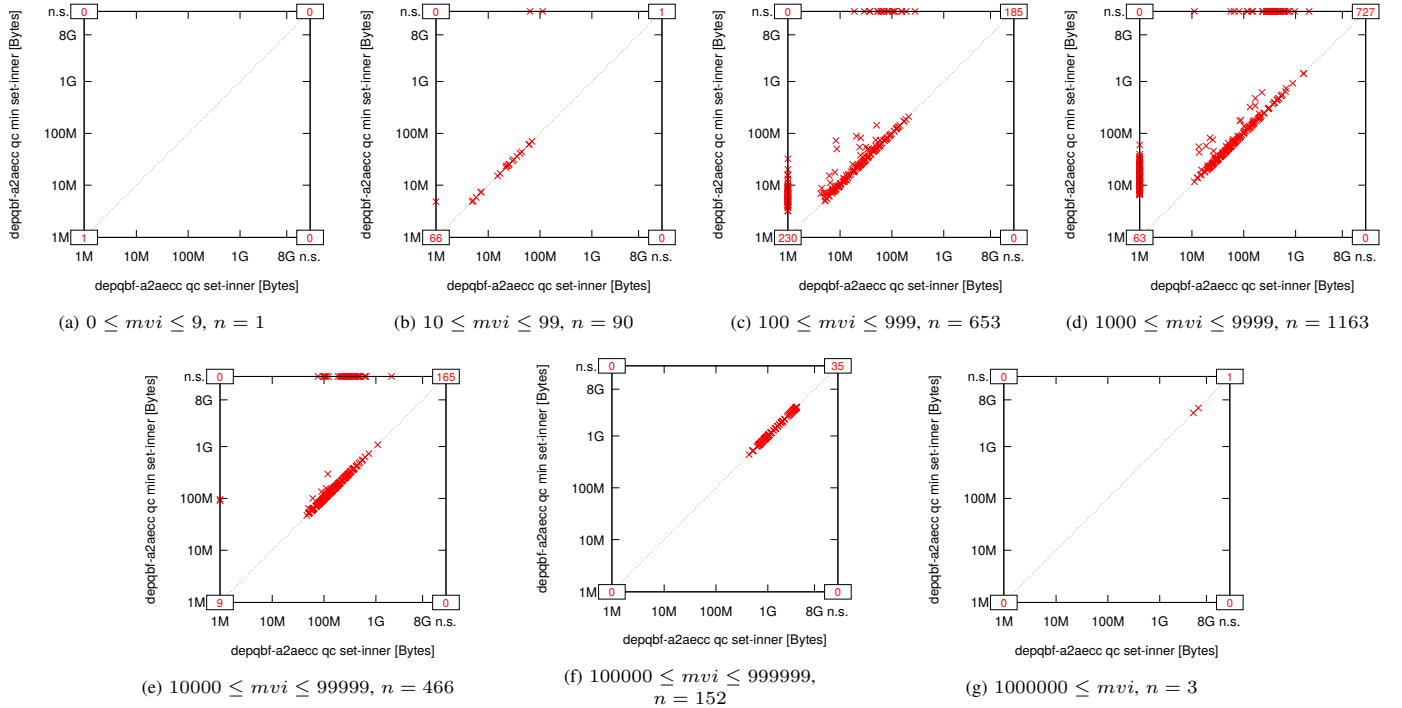


Fig. 1378: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode qc set-inner partitioned by maximum variable index (memory usage in Bytes).

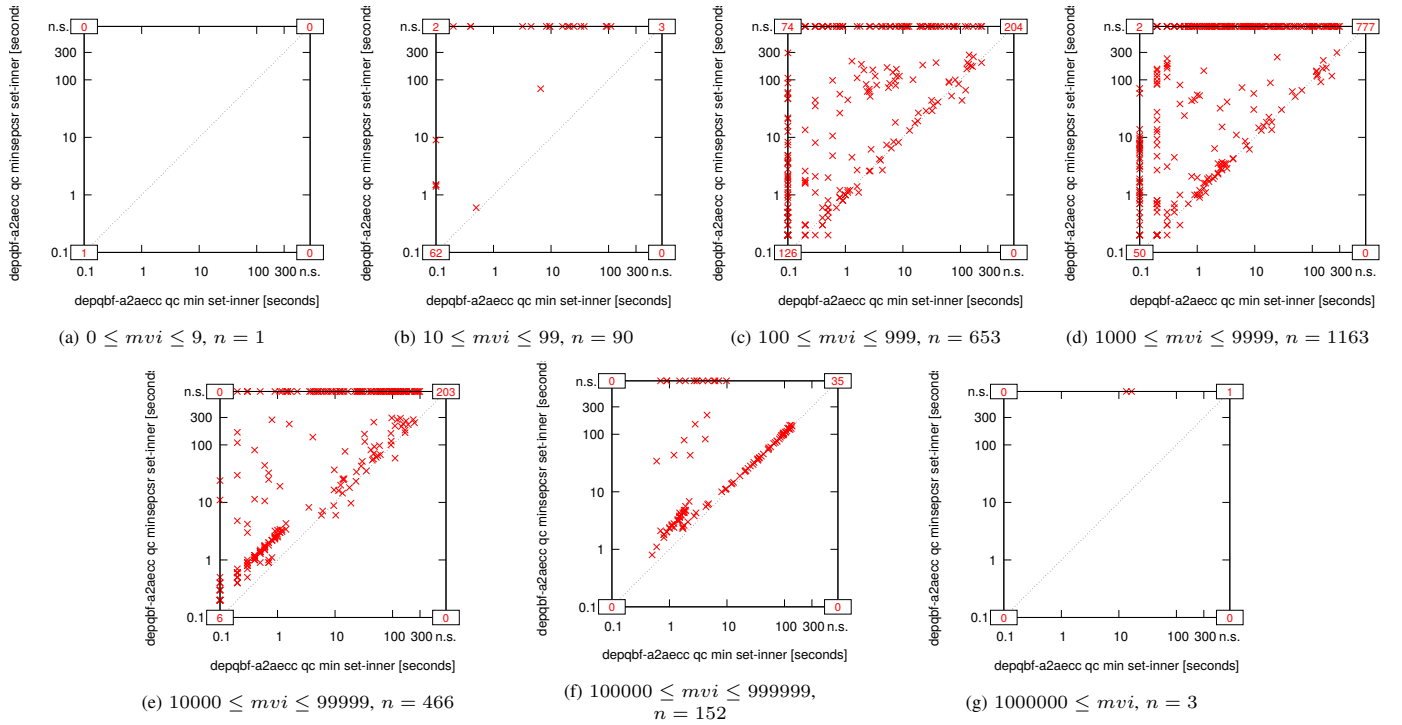


Fig. 1379: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode qc min set-inner partitioned by maximum variable index (run time in seconds).

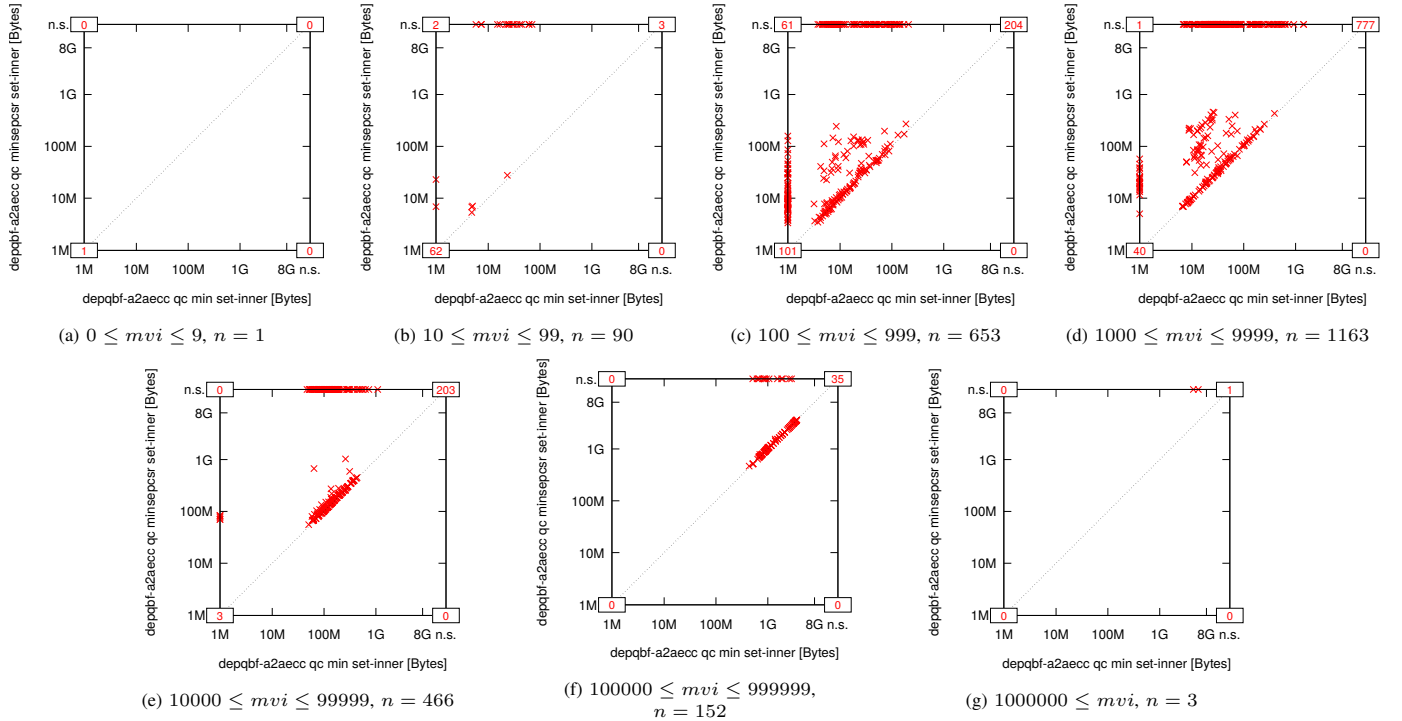


Fig. 1380: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode qc min set-inner partitioned by maximum variable index (memory usage in Bytes).

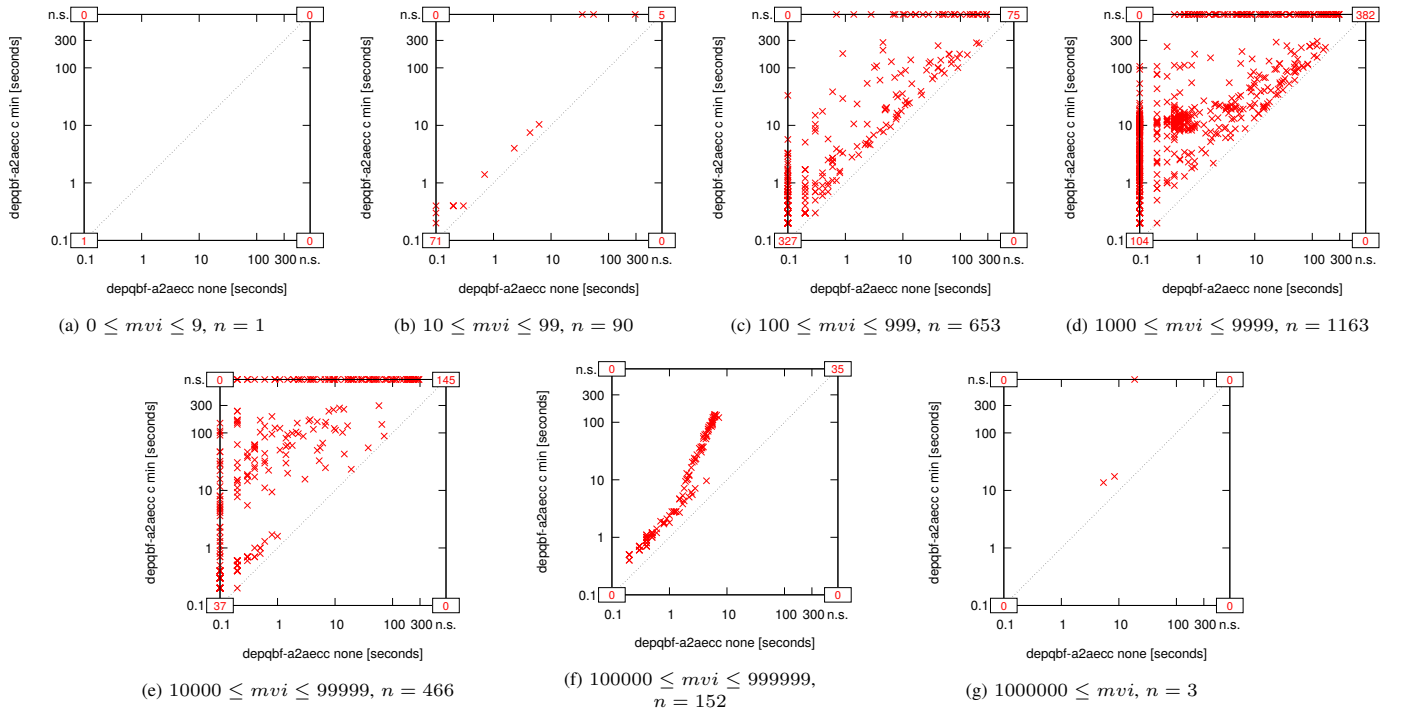


Fig. 1381: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode none partitioned by maximum variable index (run time in seconds).

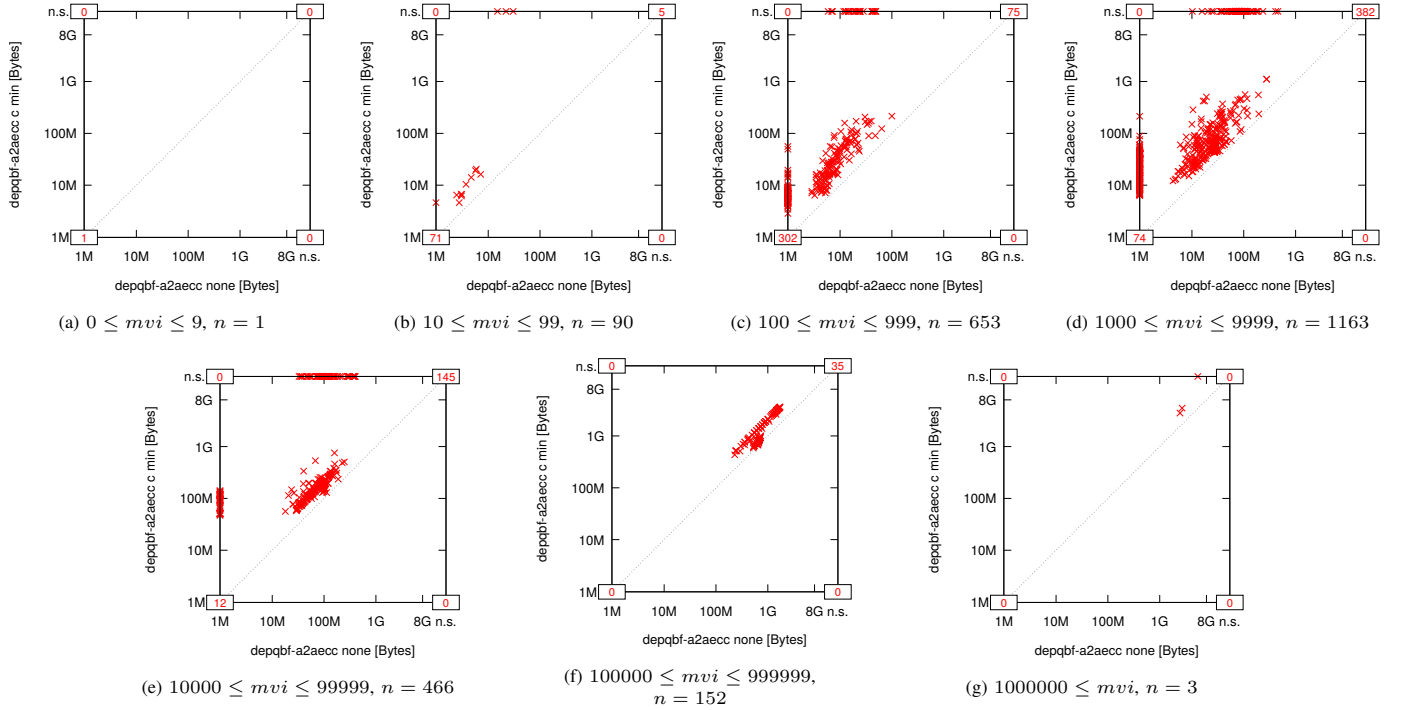


Fig. 1382: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode none partitioned by maximum variable index (memory usage in Bytes).

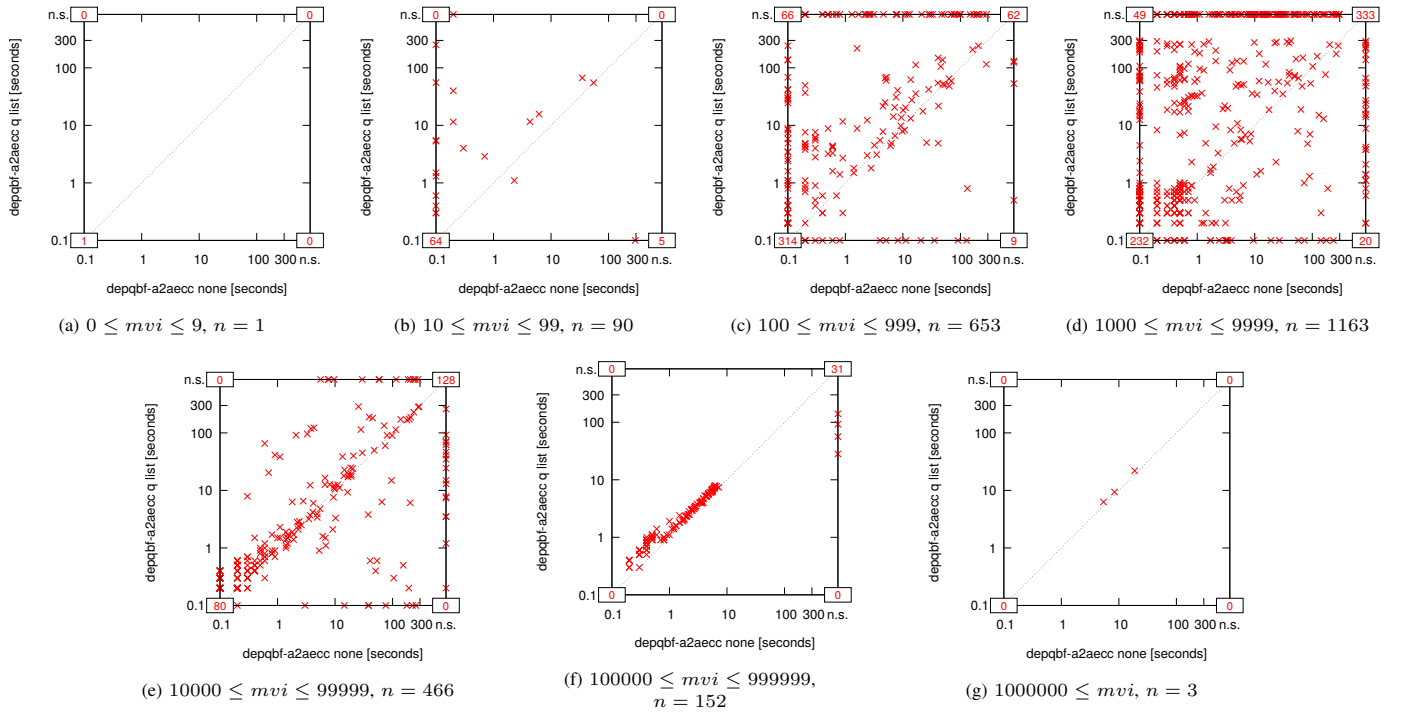


Fig. 1383: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode none partitioned by maximum variable index (run time in seconds).

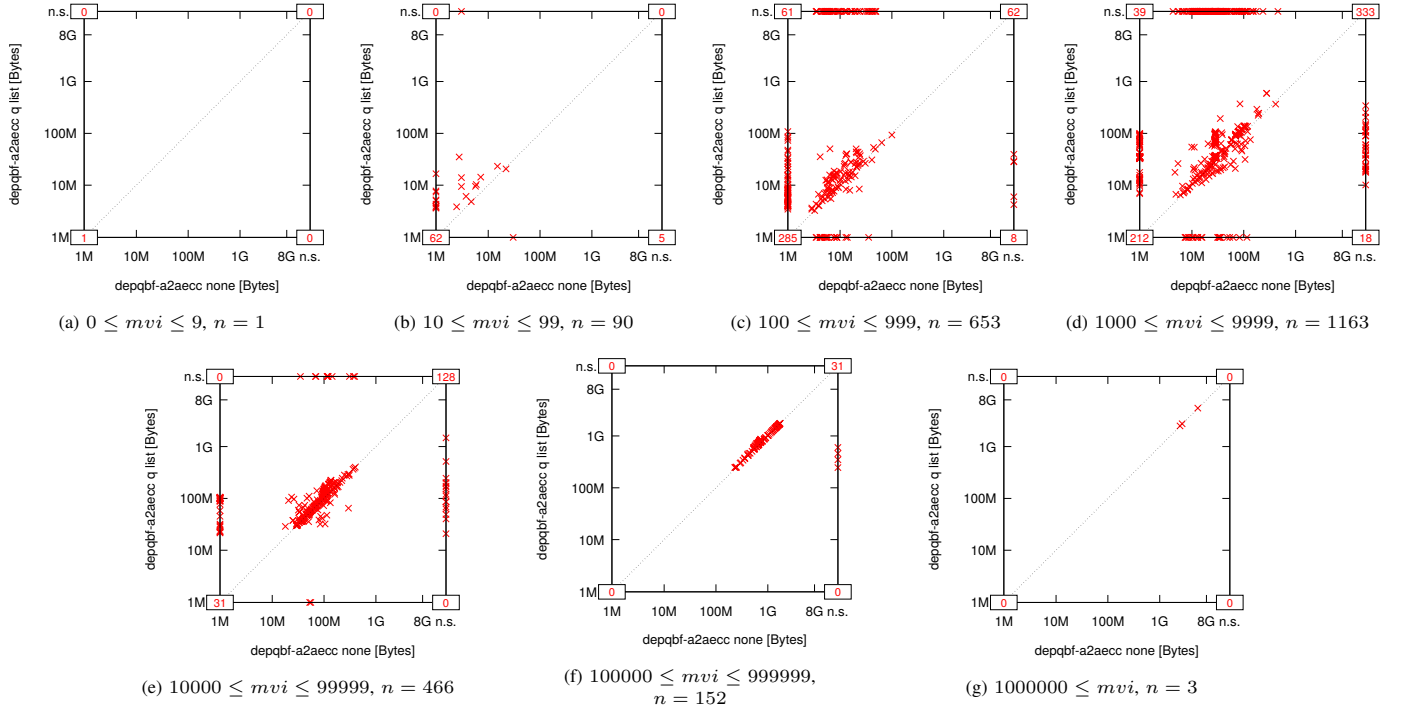


Fig. 1384: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode none partitioned by maximum variable index (memory usage in Bytes).

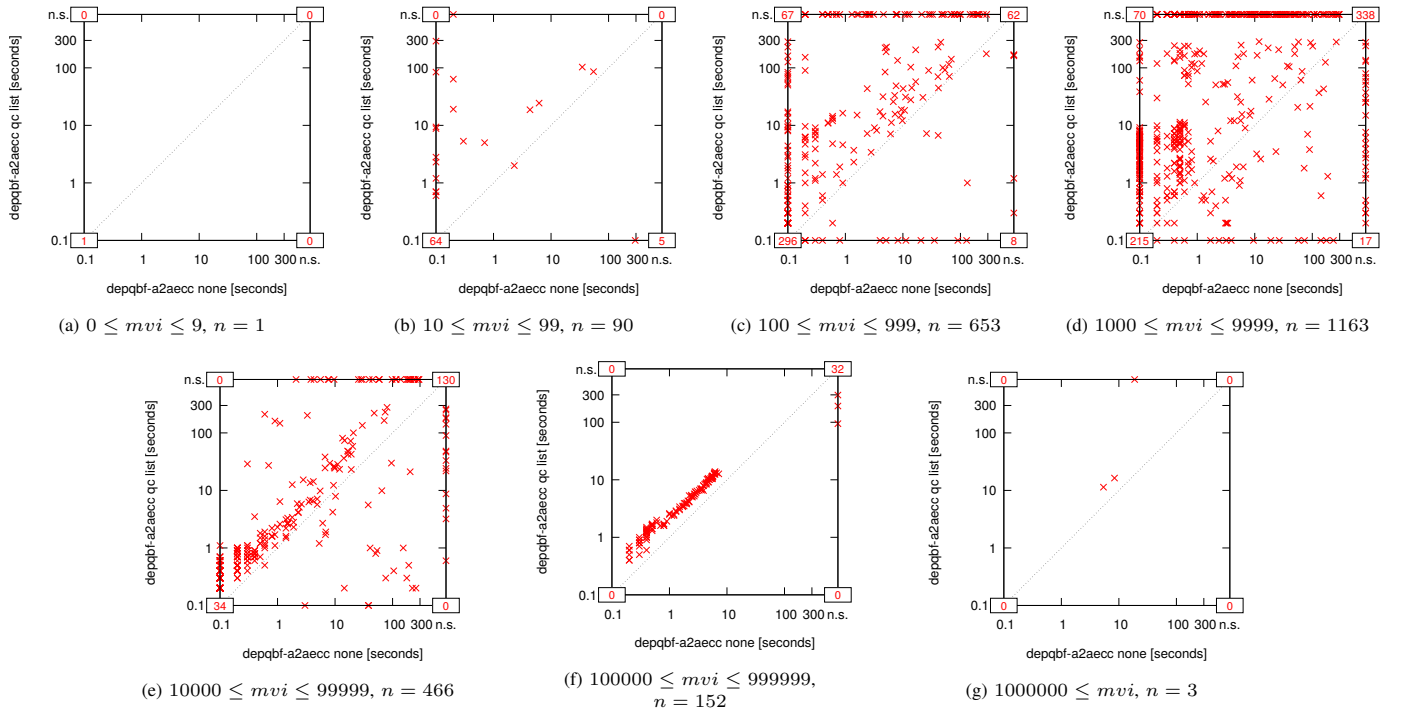


Fig. 1385: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode none partitioned by maximum variable index (run time in seconds).

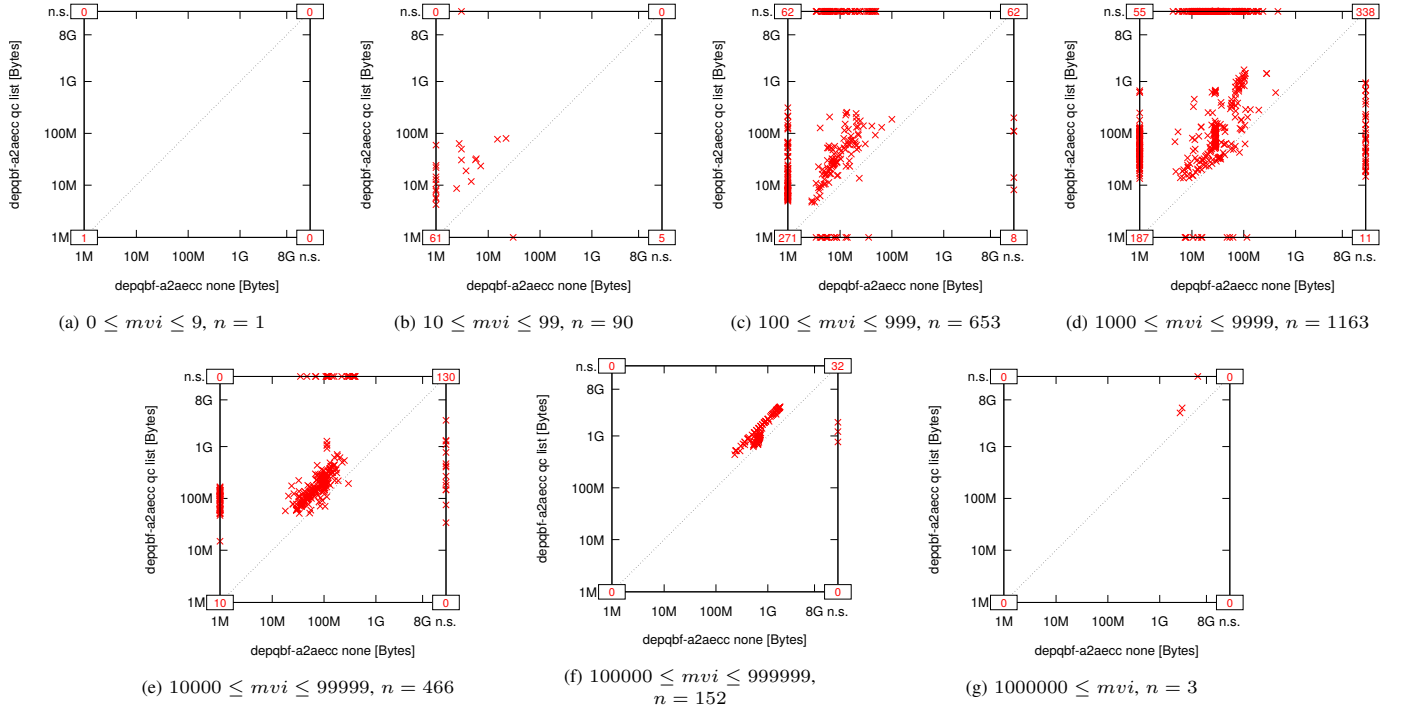


Fig. 1386: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode none partitioned by maximum variable index (memory usage in Bytes).

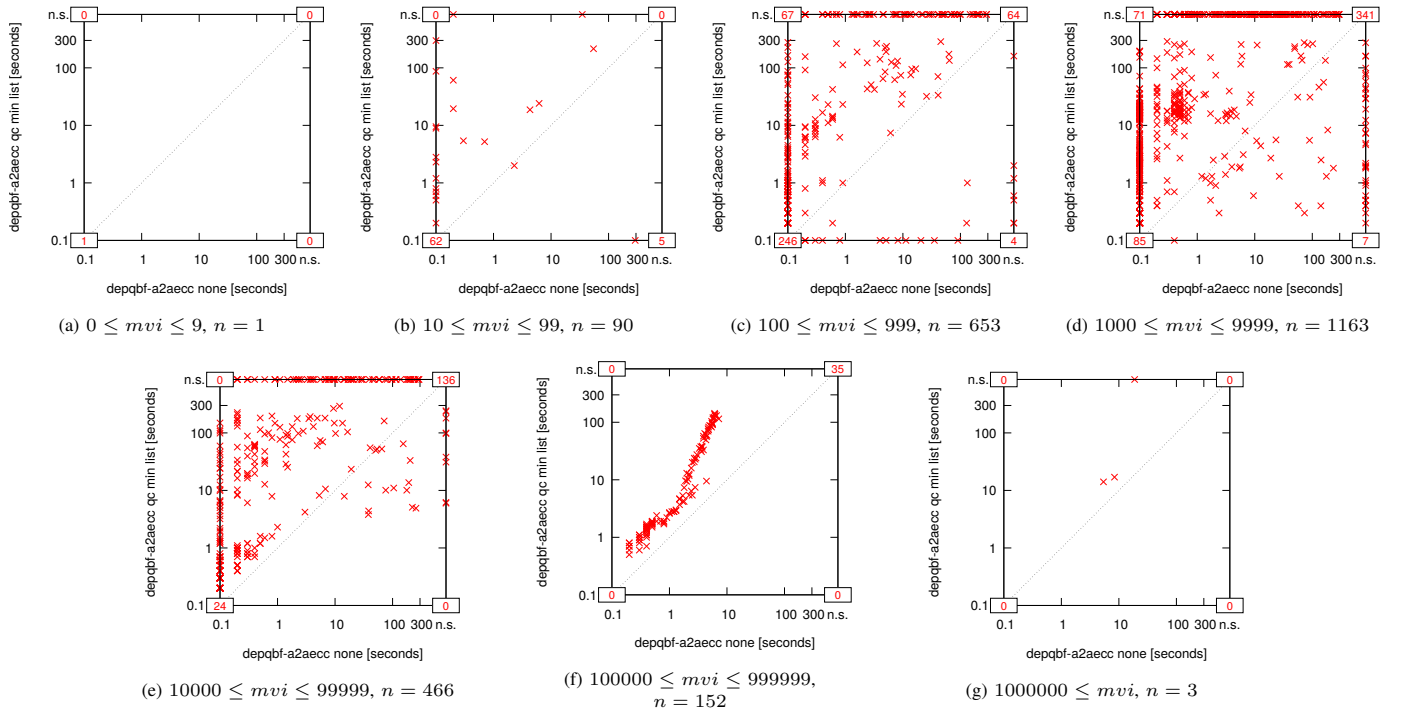


Fig. 1387: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode none partitioned by maximum variable index (run time in seconds).

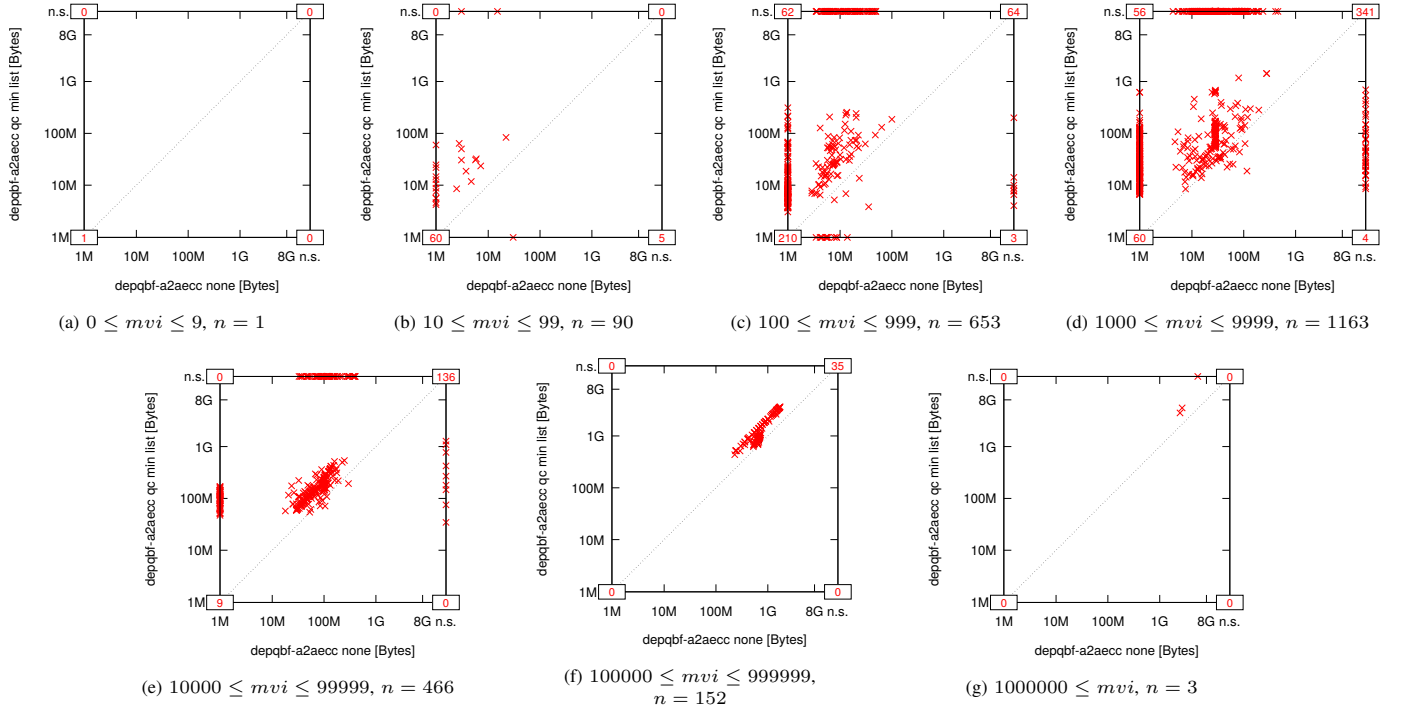


Fig. 1388: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode none partitioned by maximum variable index (memory usage in Bytes).

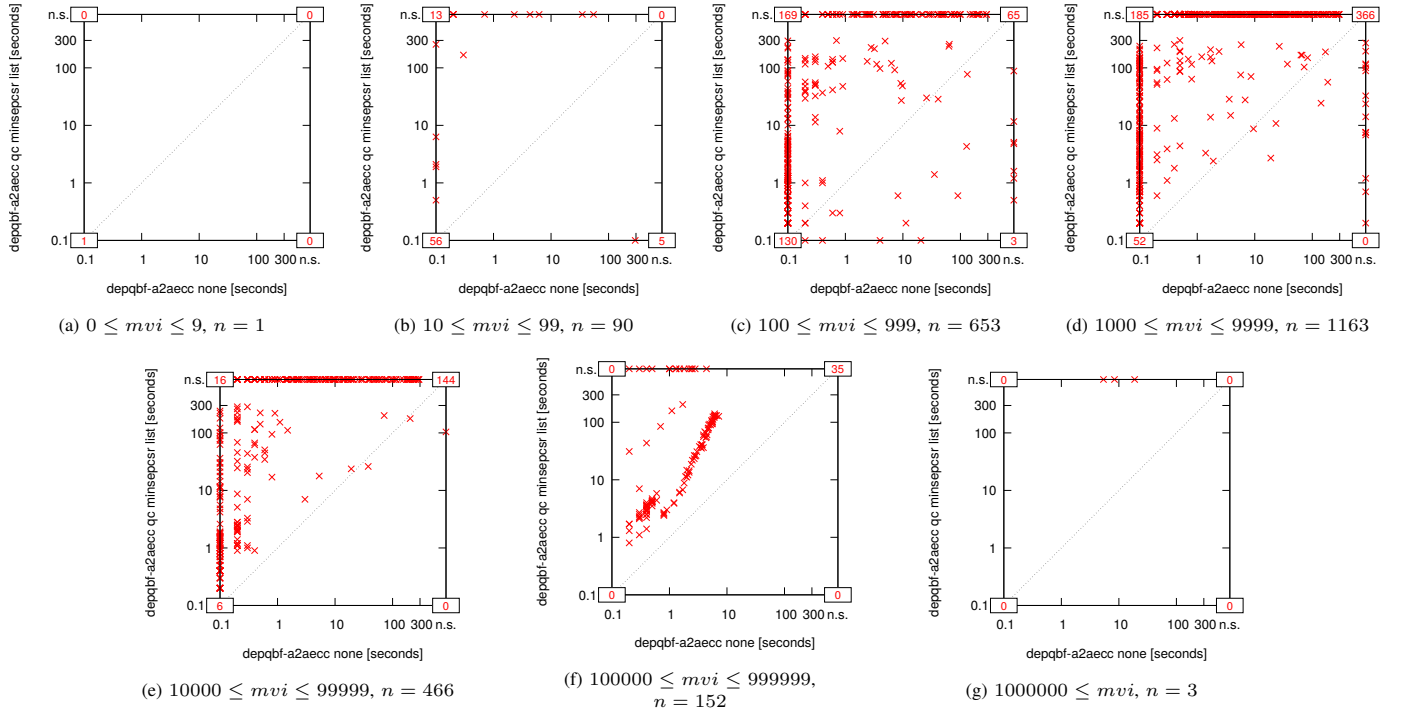


Fig. 1389: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode none partitioned by maximum variable index (run time in seconds).

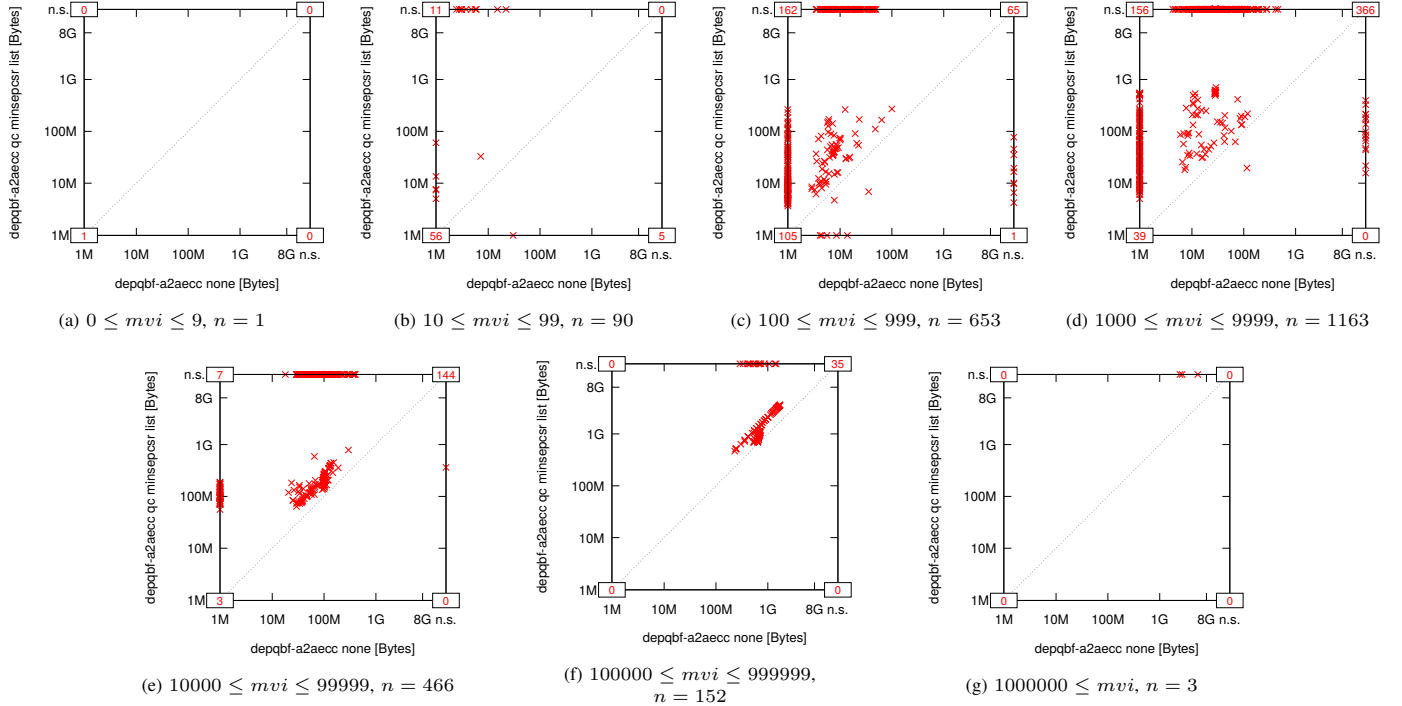


Fig. 1390: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode none partitioned by maximum variable index (memory usage in Bytes).

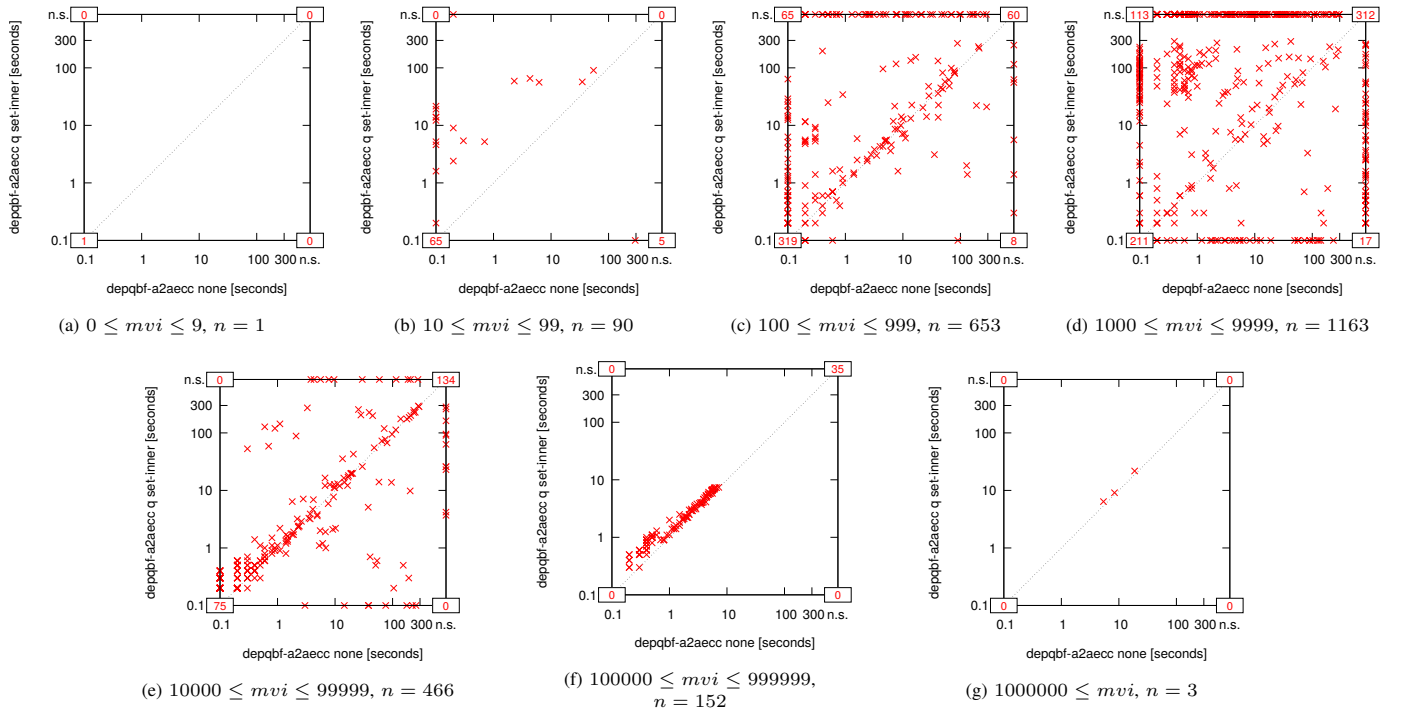


Fig. 1391: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode none partitioned by maximum variable index (run time in seconds).

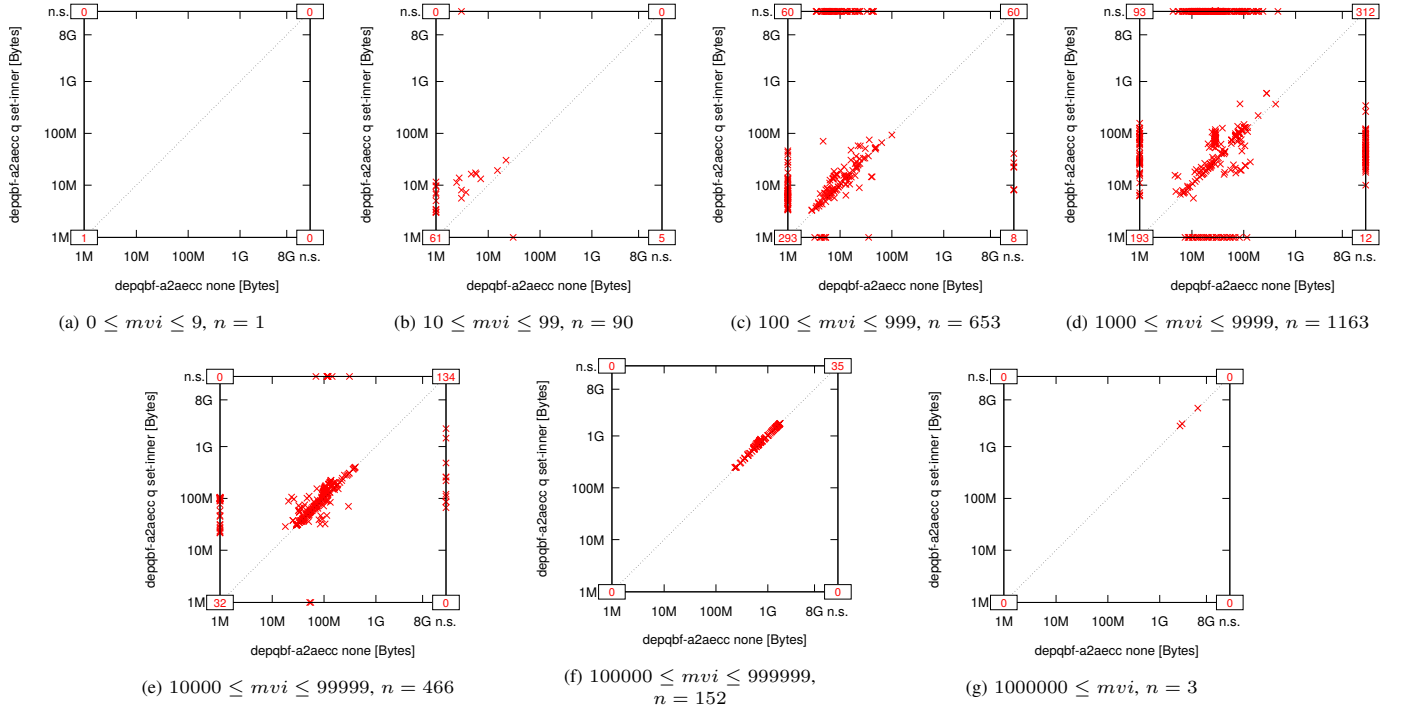


Fig. 1392: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode none partitioned by maximum variable index (memory usage in Bytes).

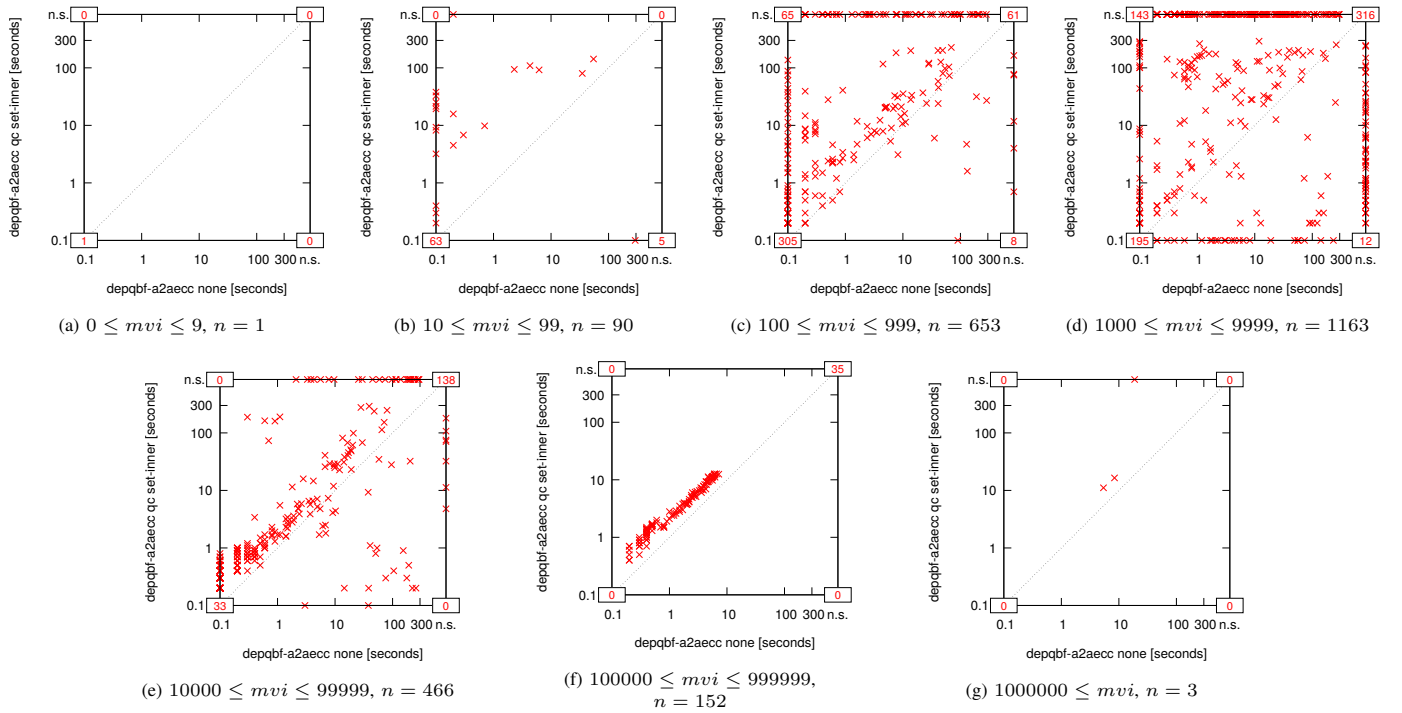


Fig. 1393: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode none partitioned by maximum variable index (run time in seconds).

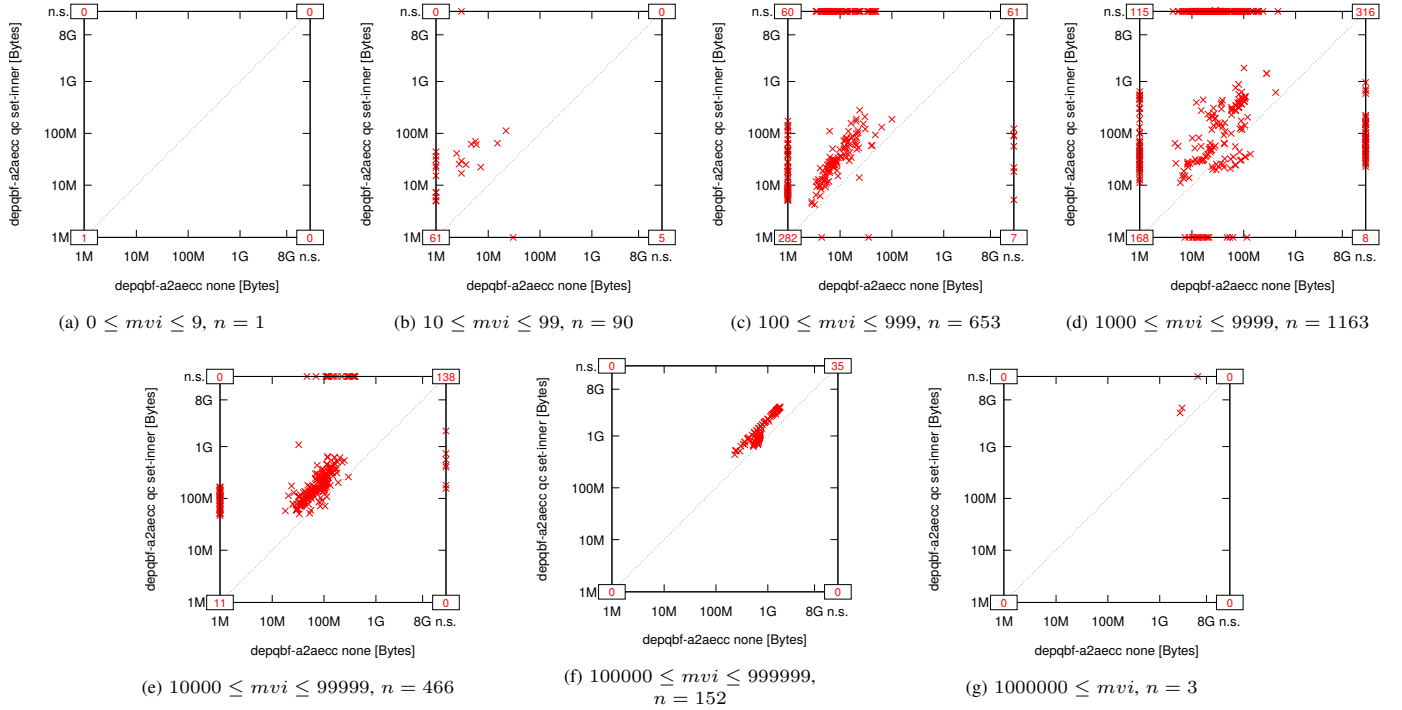


Fig. 1394: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode none partitioned by maximum variable index (memory usage in Bytes).

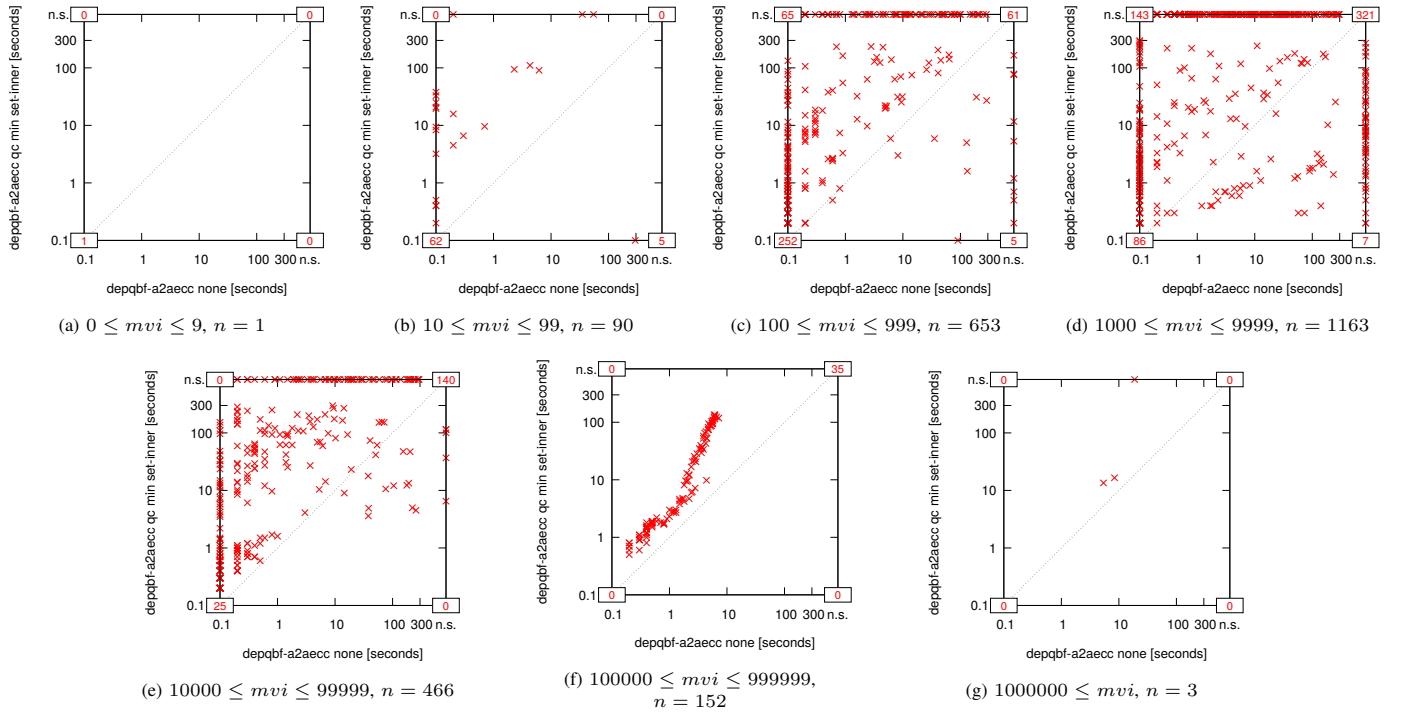


Fig. 1395: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode none partitioned by maximum variable index (run time in seconds).

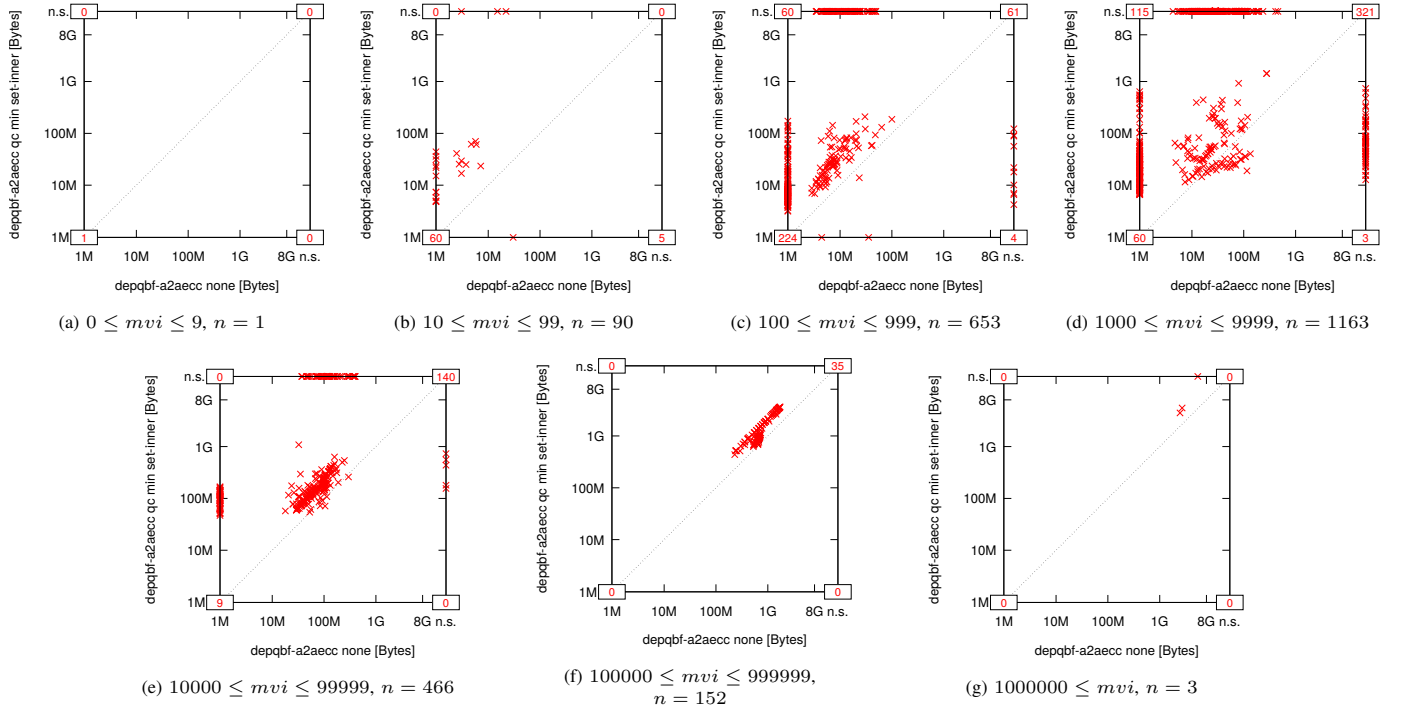


Fig. 1396: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode none partitioned by maximum variable index (memory usage in Bytes).

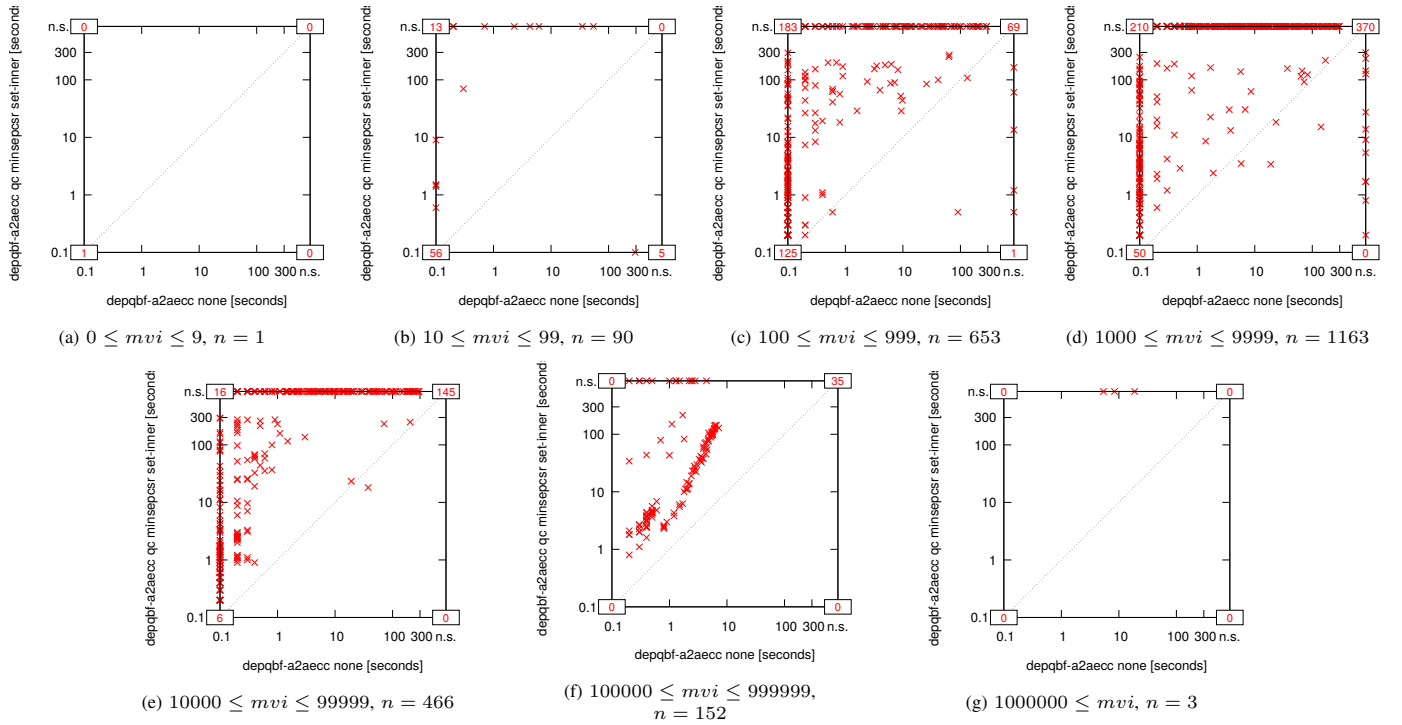


Fig. 1397: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode none partitioned by maximum variable index (run time in seconds).

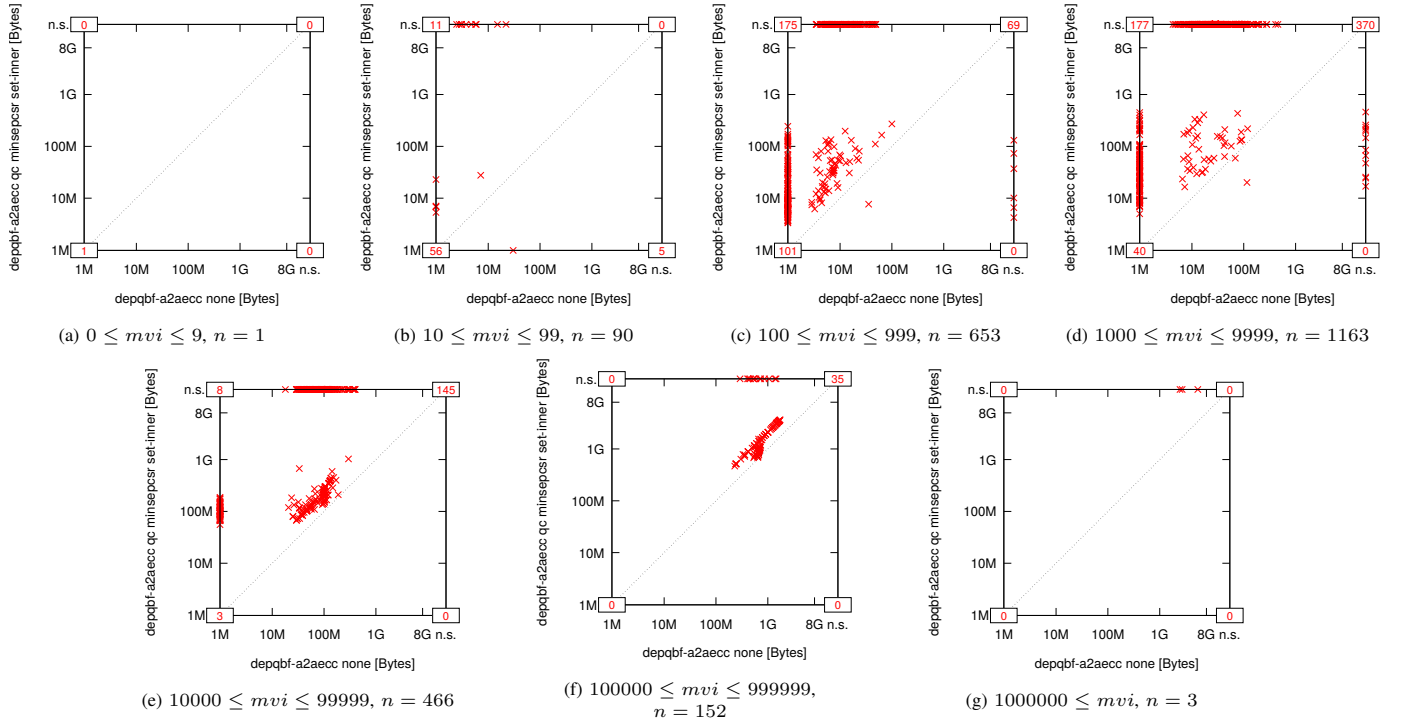


Fig. 1398: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode none partitioned by maximum variable index (memory usage in Bytes).

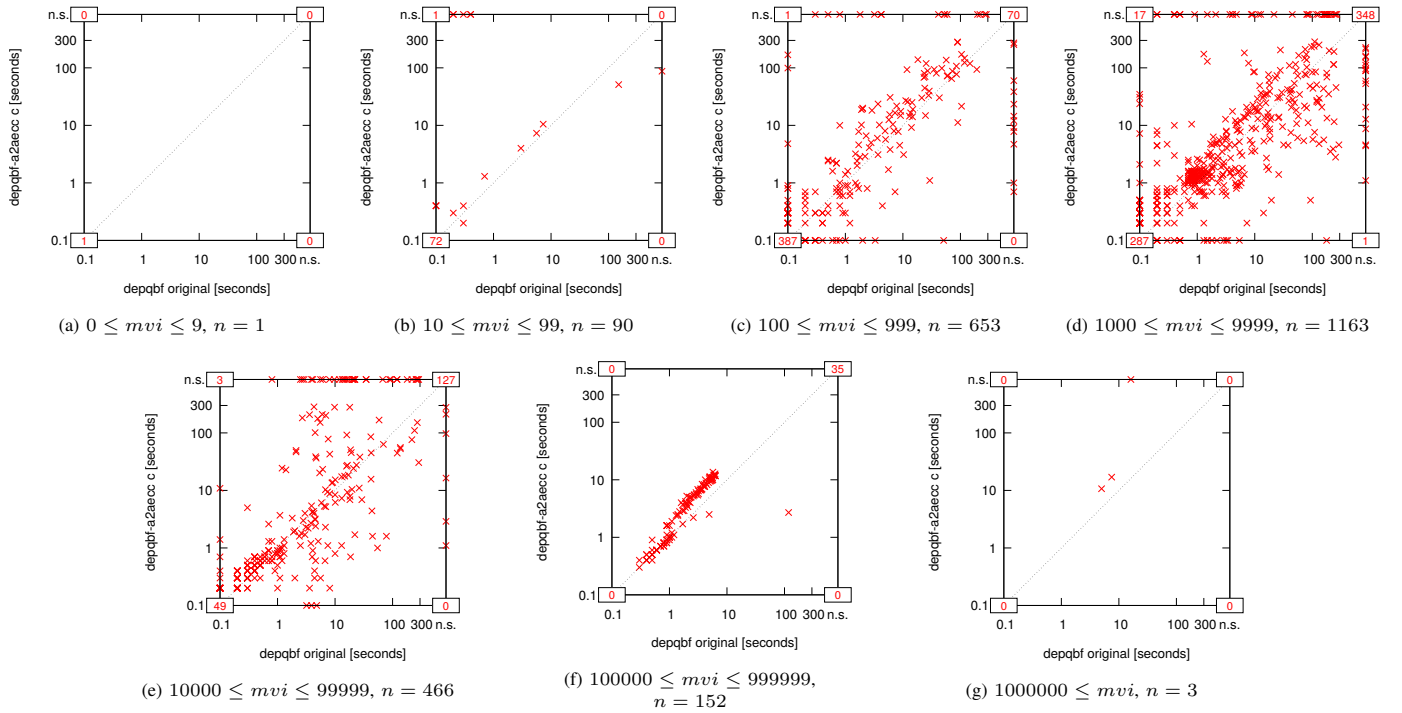


Fig. 1399: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c versus mode original partitioned by maximum variable index (run time in seconds).

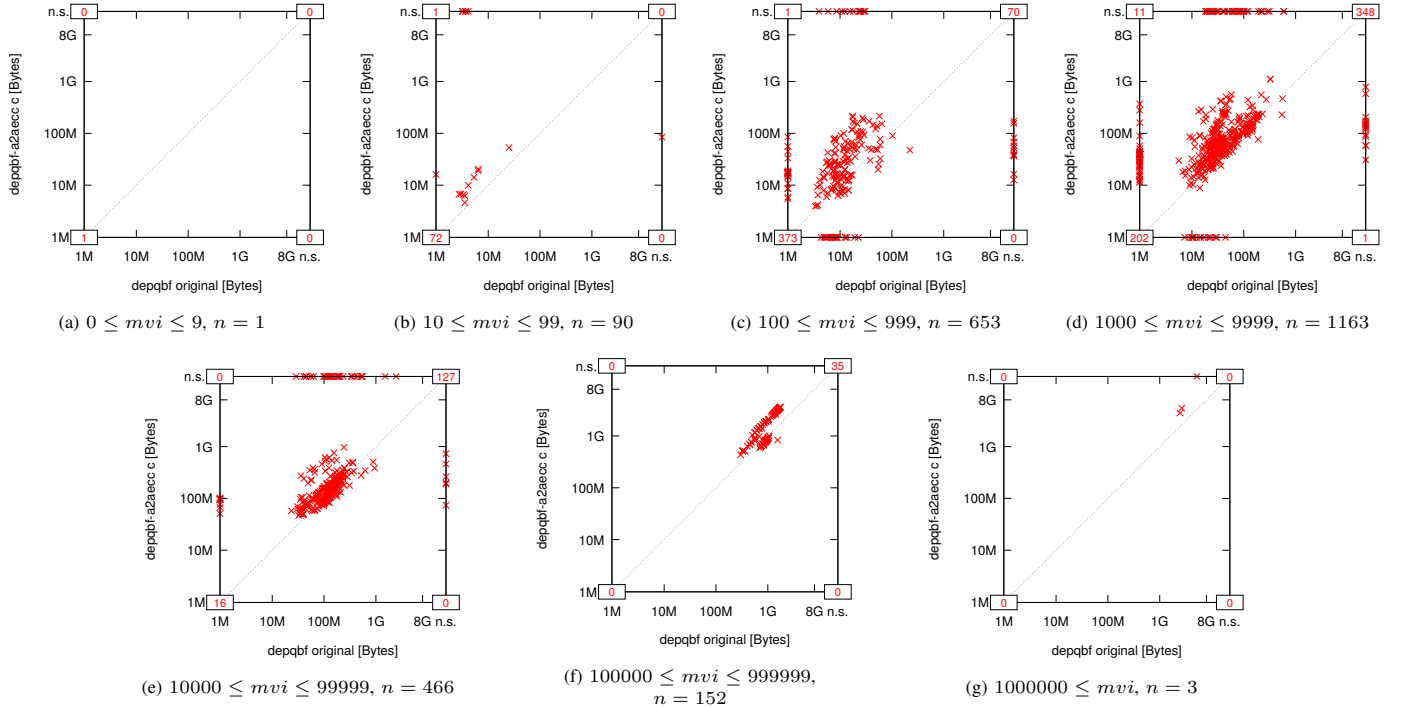


Fig. 1400: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c versus mode original partitioned by maximum variable index (memory usage in Bytes).

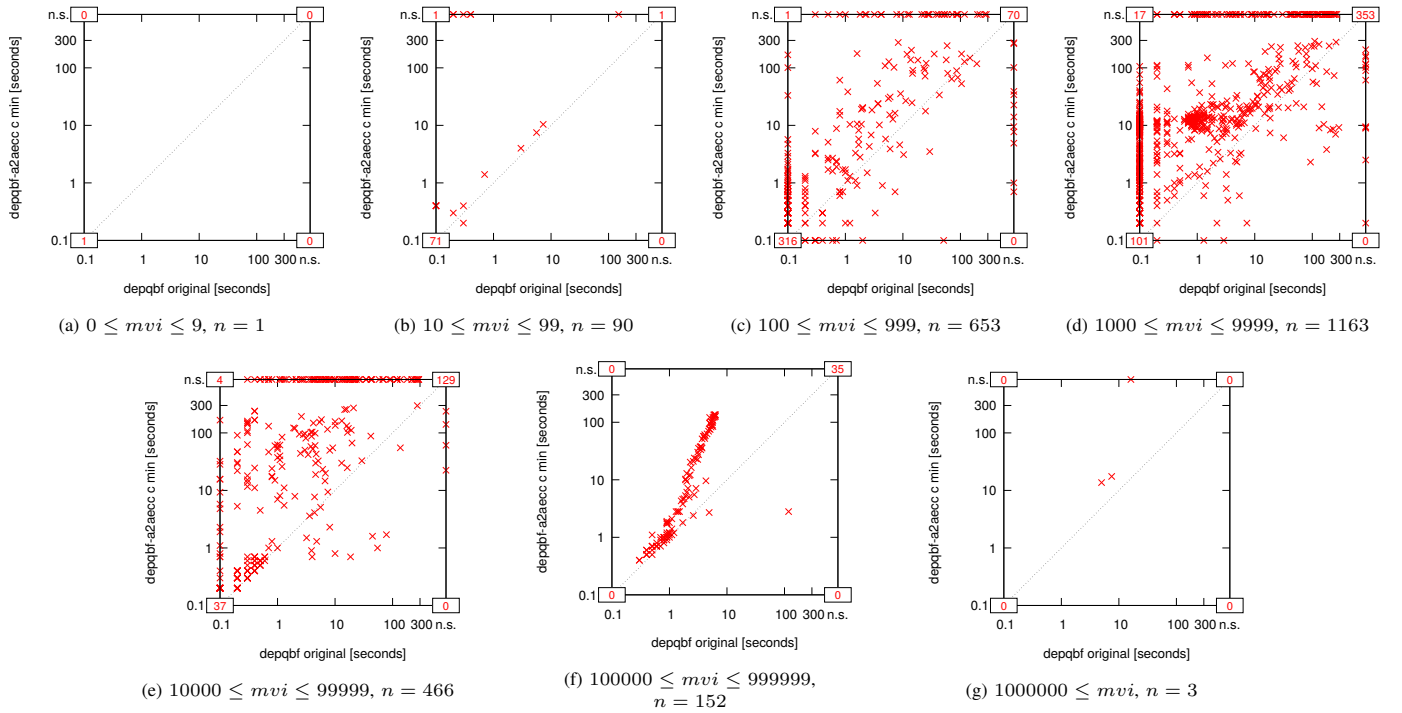


Fig. 1401: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode original partitioned by maximum variable index (run time in seconds).

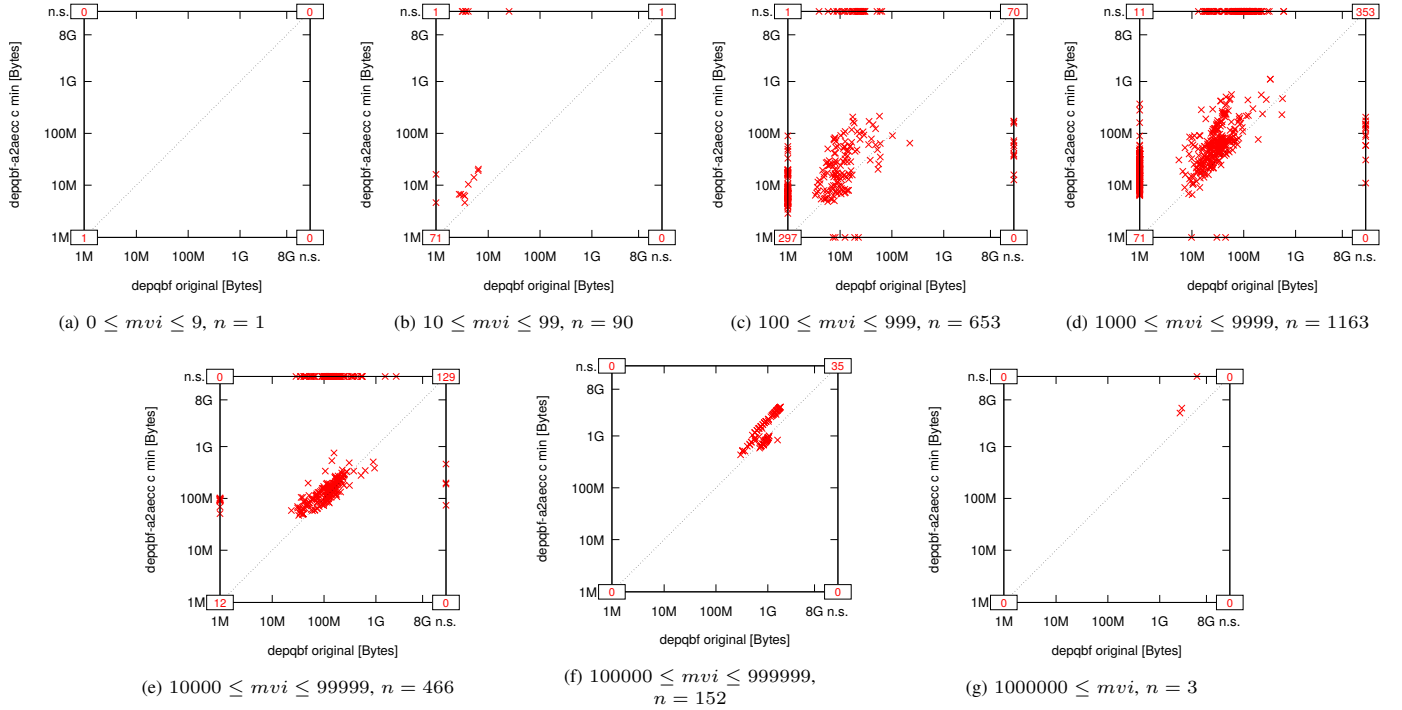


Fig. 1402: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode c min versus mode original partitioned by maximum variable index (memory usage in Bytes).

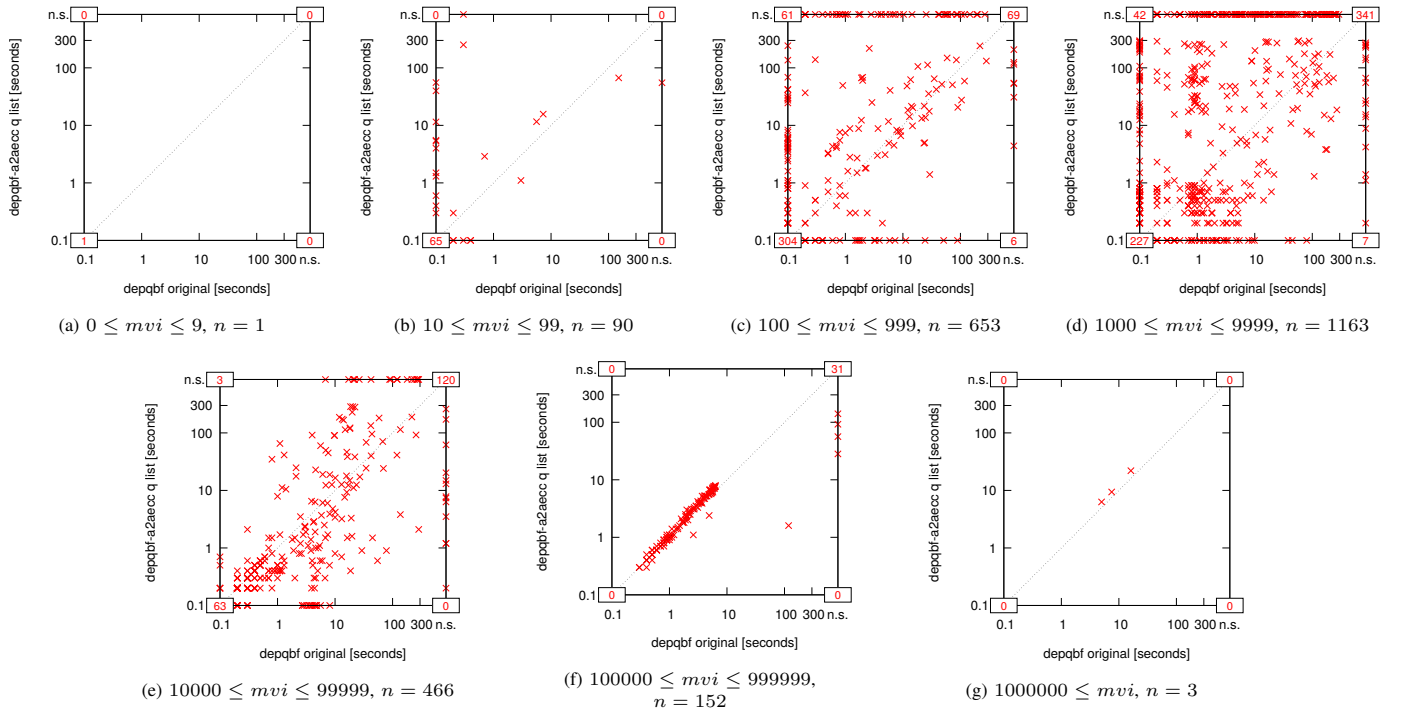


Fig. 1403: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode original partitioned by maximum variable index (run time in seconds).

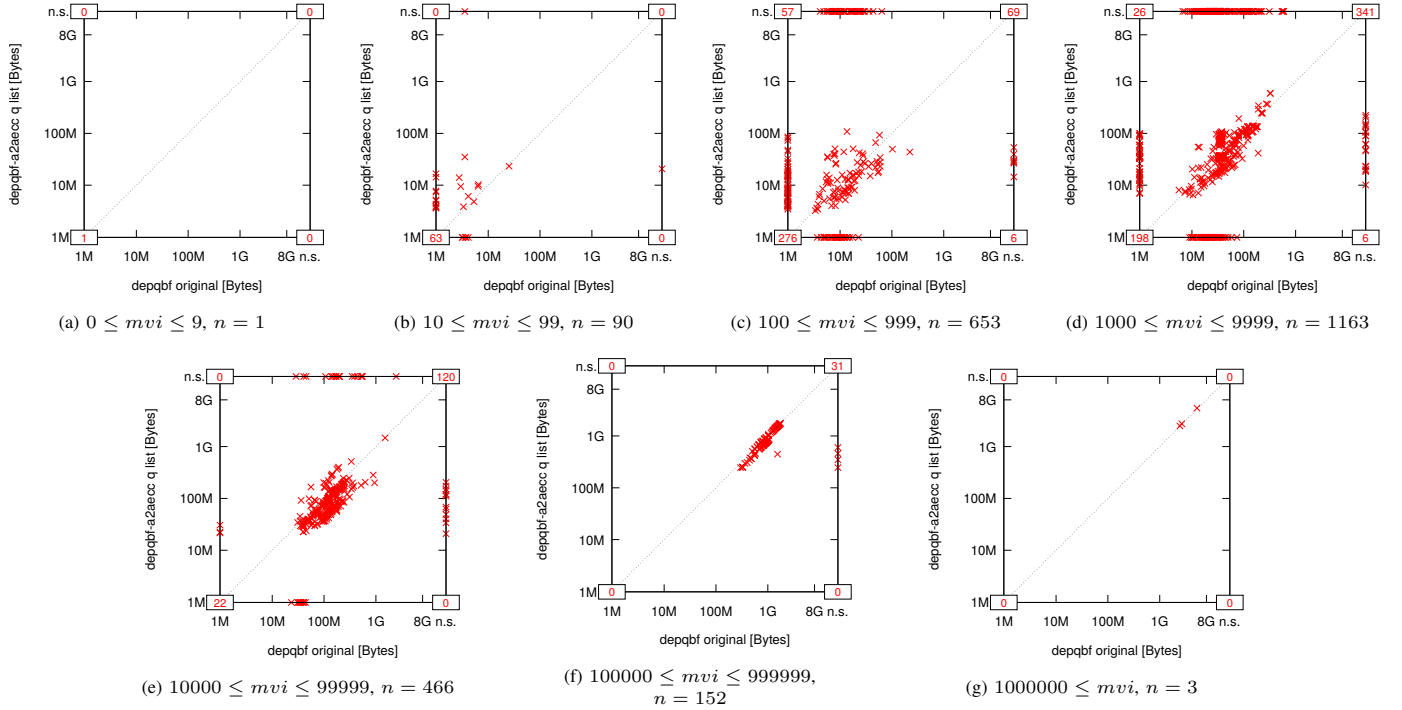


Fig. 1404: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q list versus mode original partitioned by maximum variable index (memory usage in Bytes).

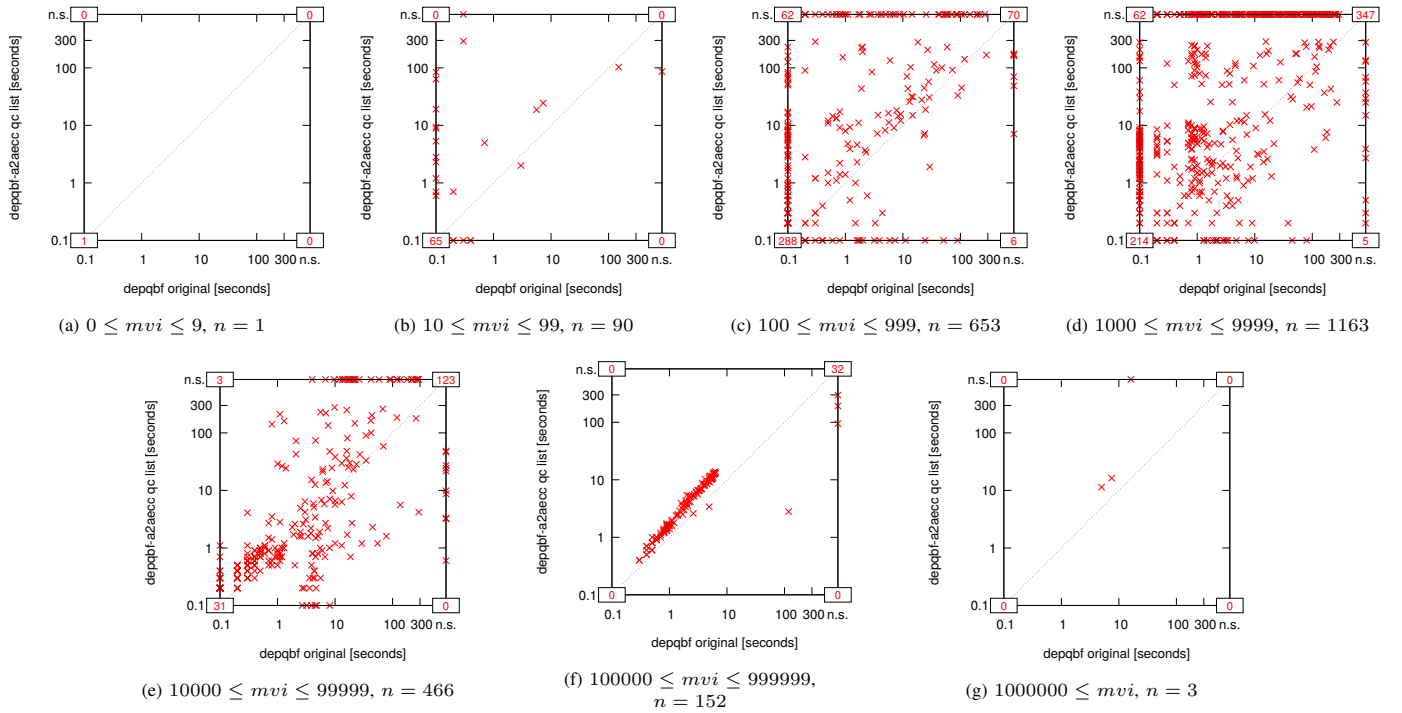


Fig. 1405: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode original partitioned by maximum variable index (run time in seconds).

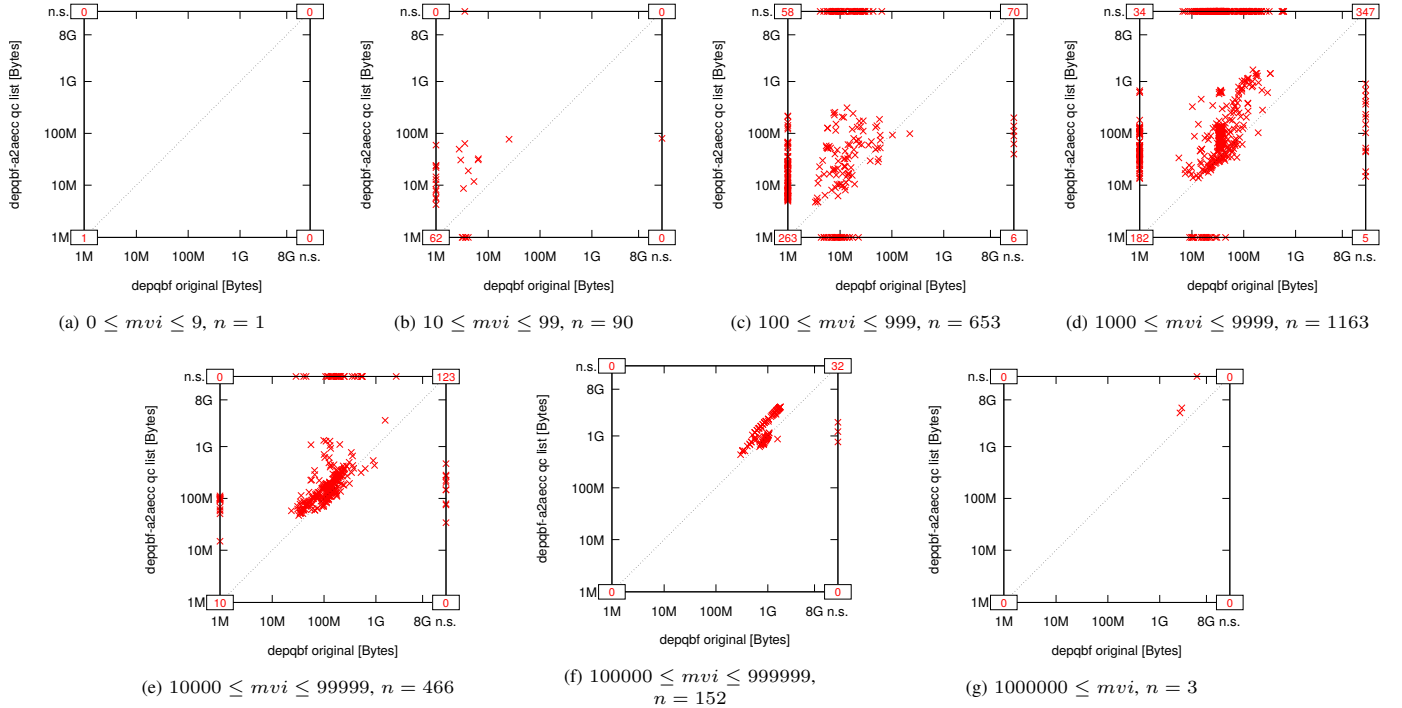


Fig. 1406: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc list versus mode original partitioned by maximum variable index (memory usage in Bytes).

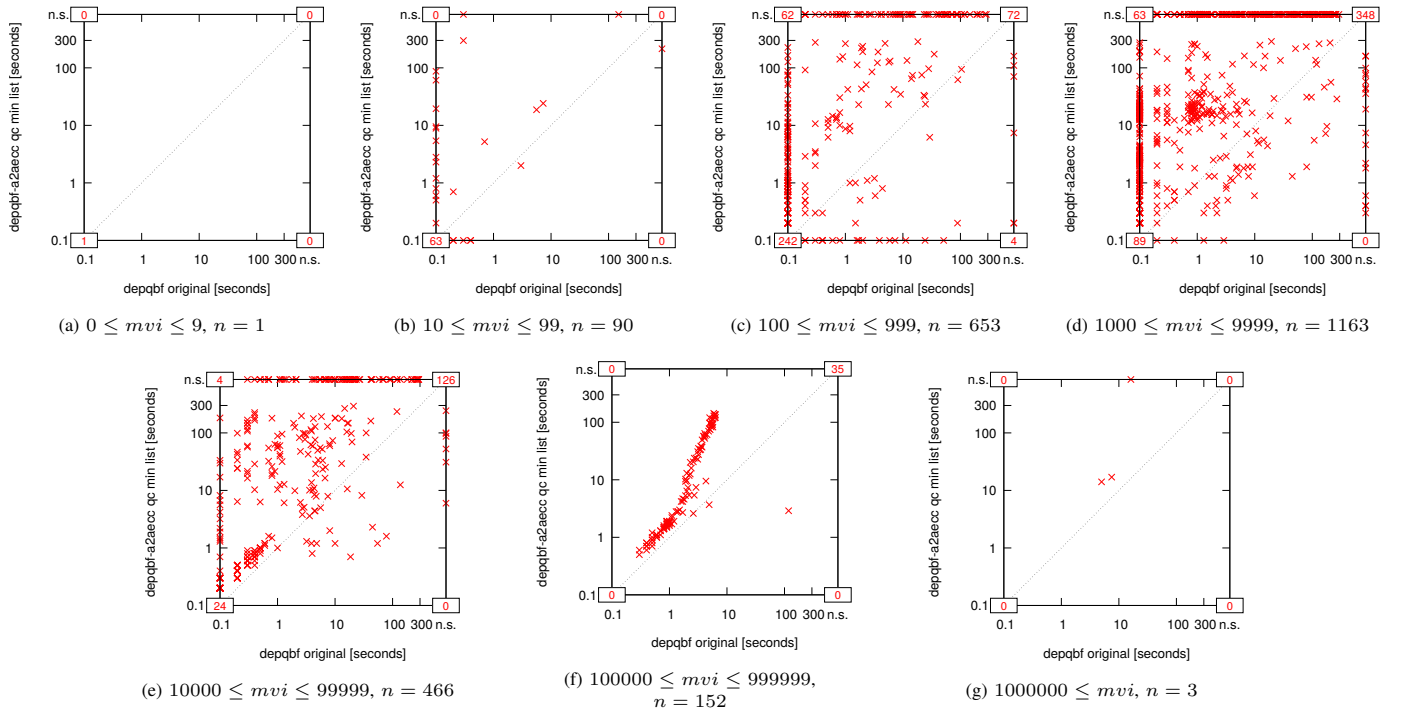


Fig. 1407: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode original partitioned by maximum variable index (run time in seconds).

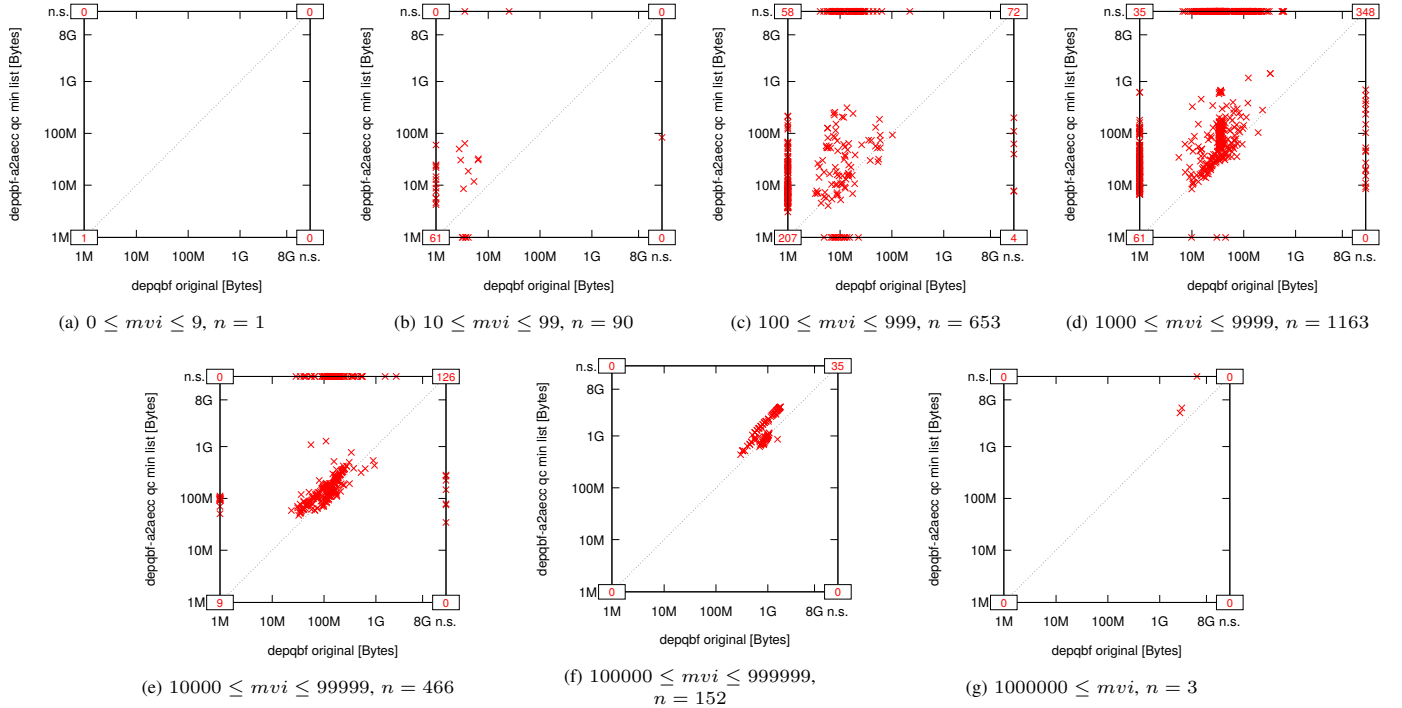


Fig. 1408: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min list versus mode original partitioned by maximum variable index (memory usage in Bytes).

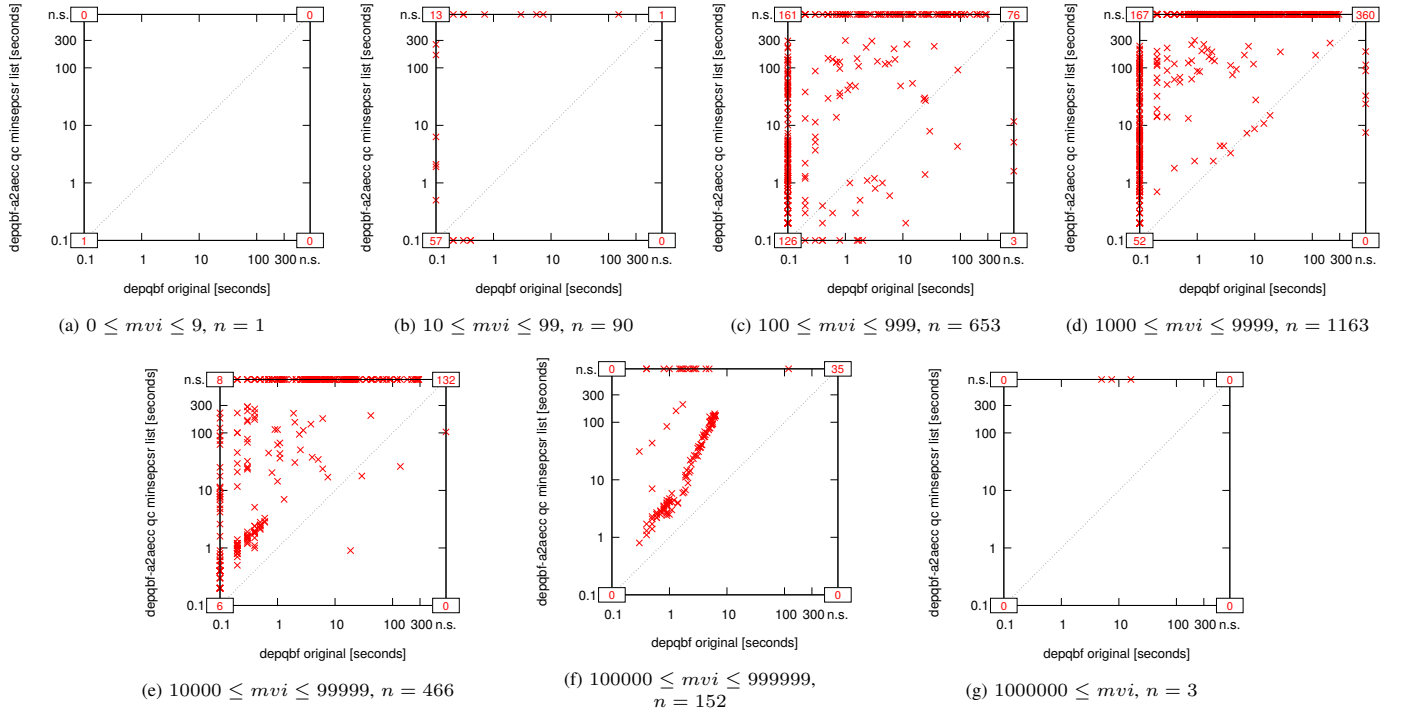


Fig. 1409: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode original partitioned by maximum variable index (run time in seconds).

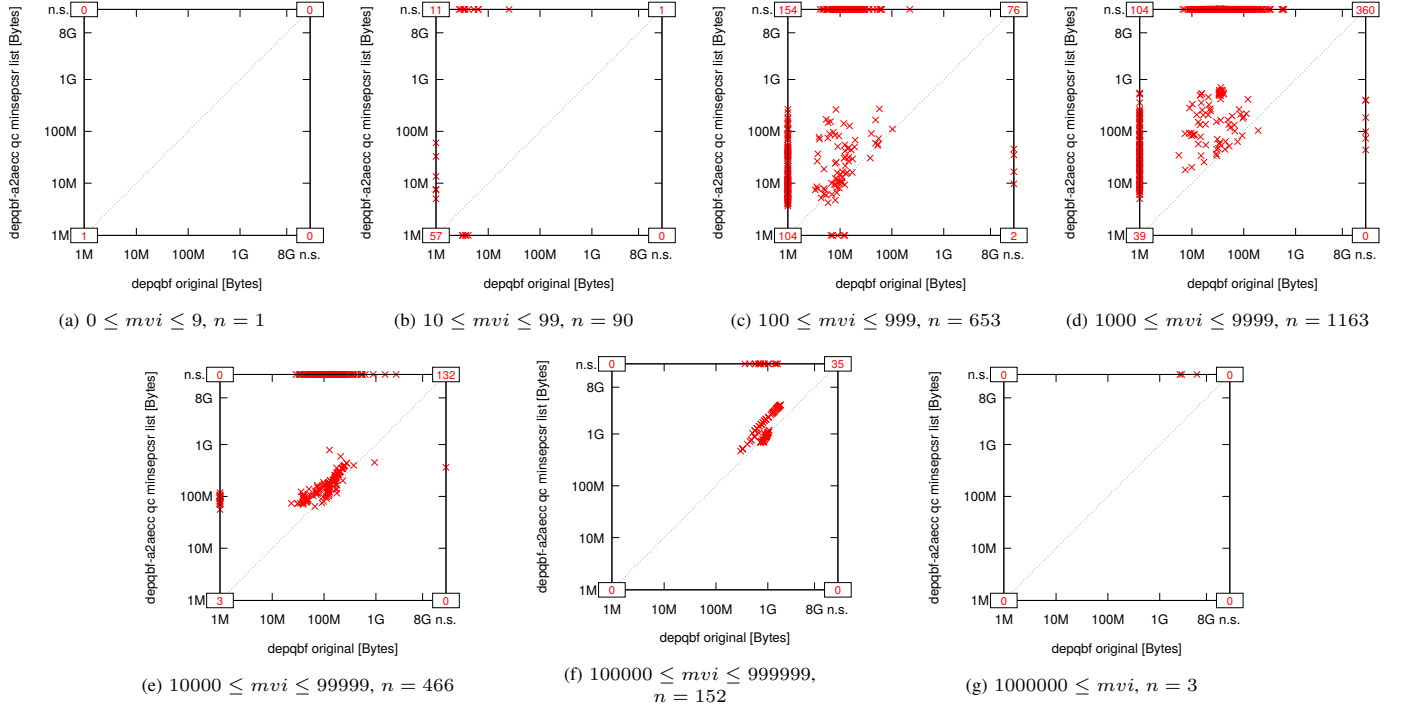


Fig. 1410: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr list versus mode original partitioned by maximum variable index (memory usage in Bytes).

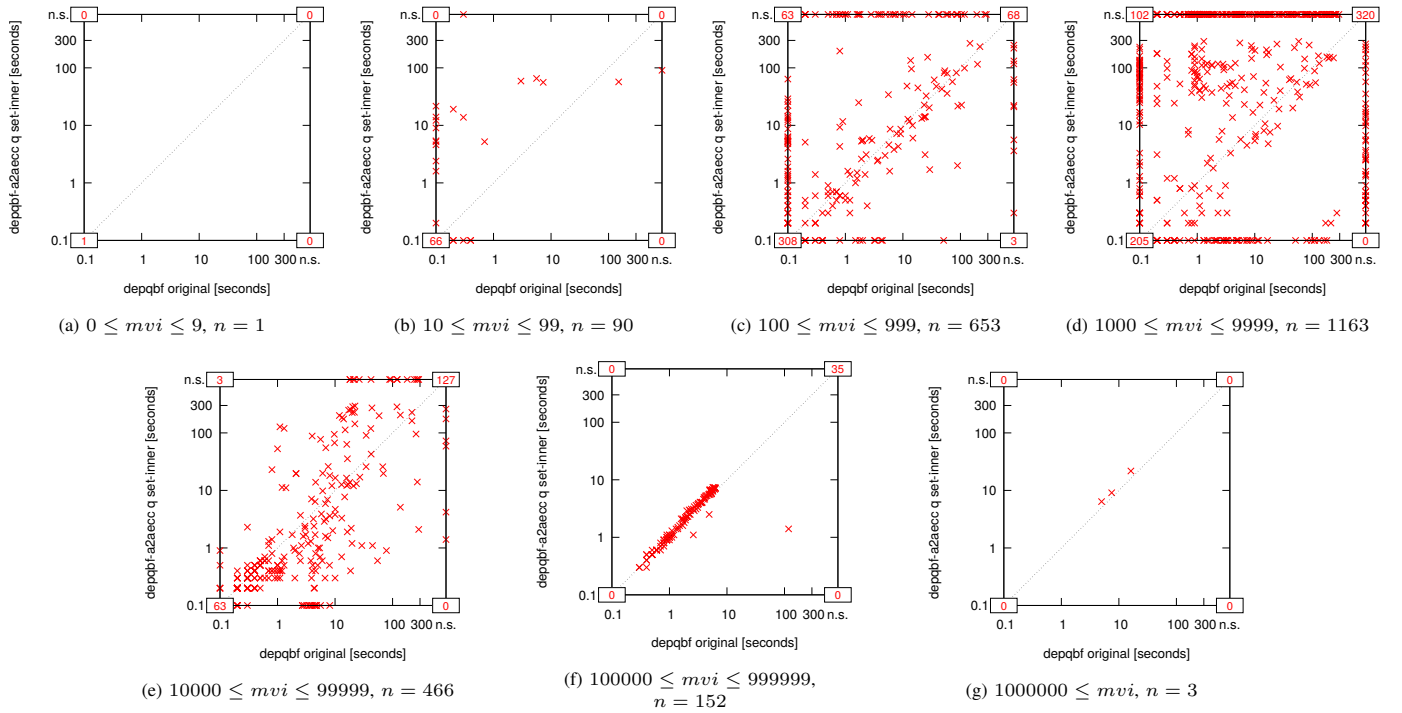


Fig. 1411: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode original partitioned by maximum variable index (run time in seconds).

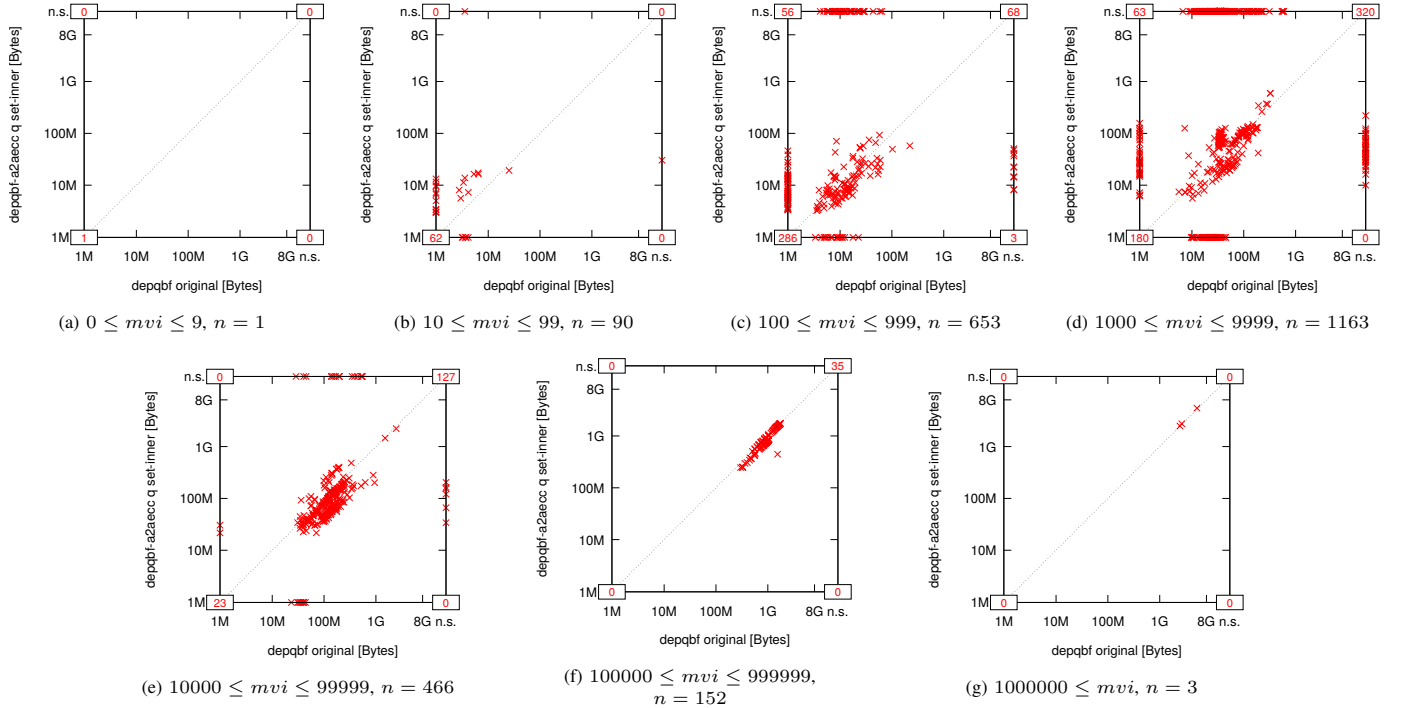


Fig. 1412: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q set-inner versus mode original partitioned by maximum variable index (memory usage in Bytes).

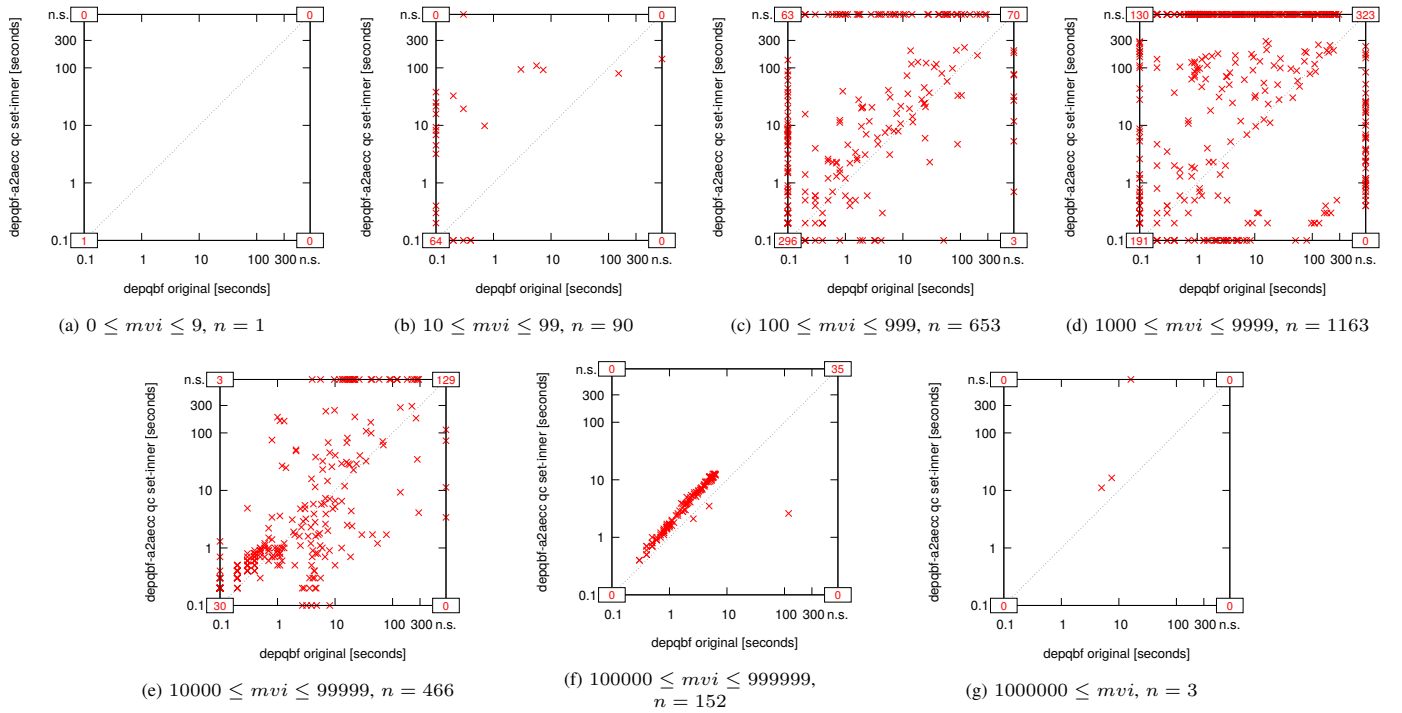


Fig. 1413: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode original partitioned by maximum variable index (run time in seconds).

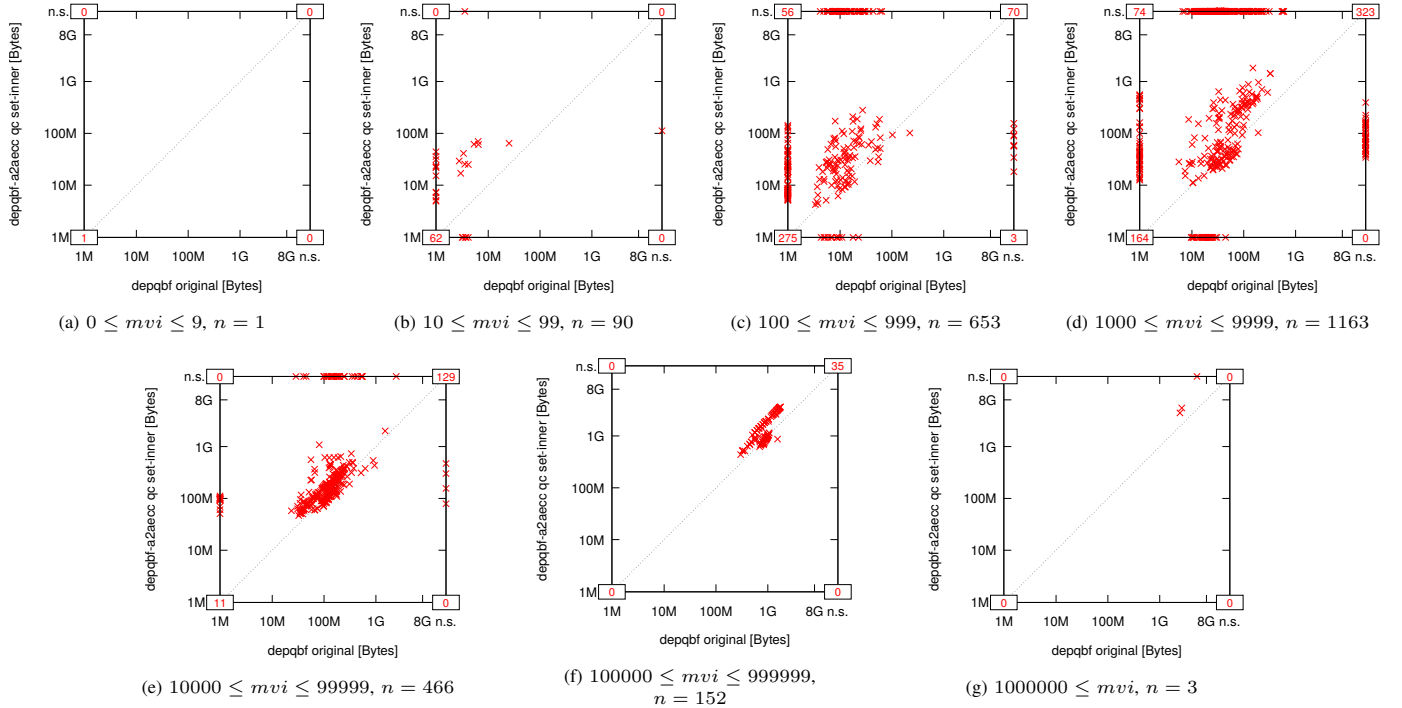


Fig. 1414: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc set-inner versus mode original partitioned by maximum variable index (memory usage in Bytes).

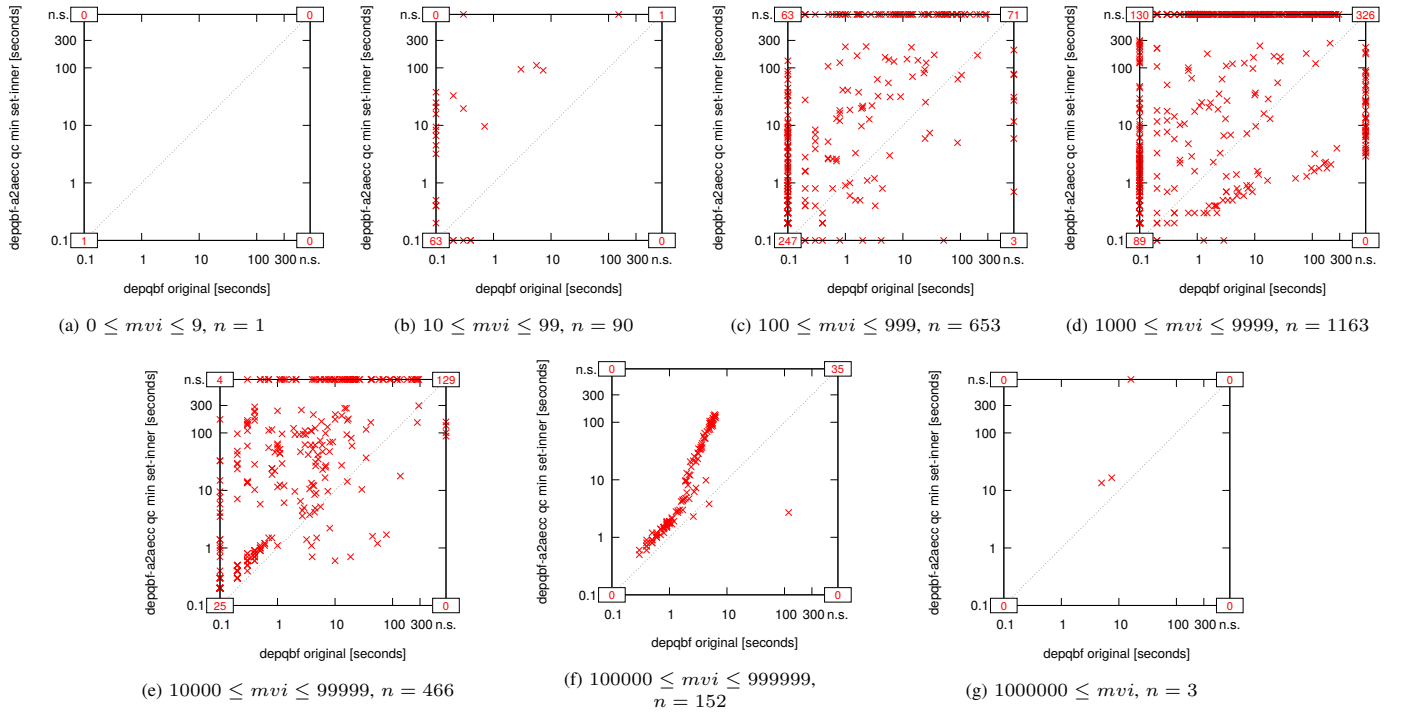


Fig. 1415: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode original partitioned by maximum variable index (run time in seconds).

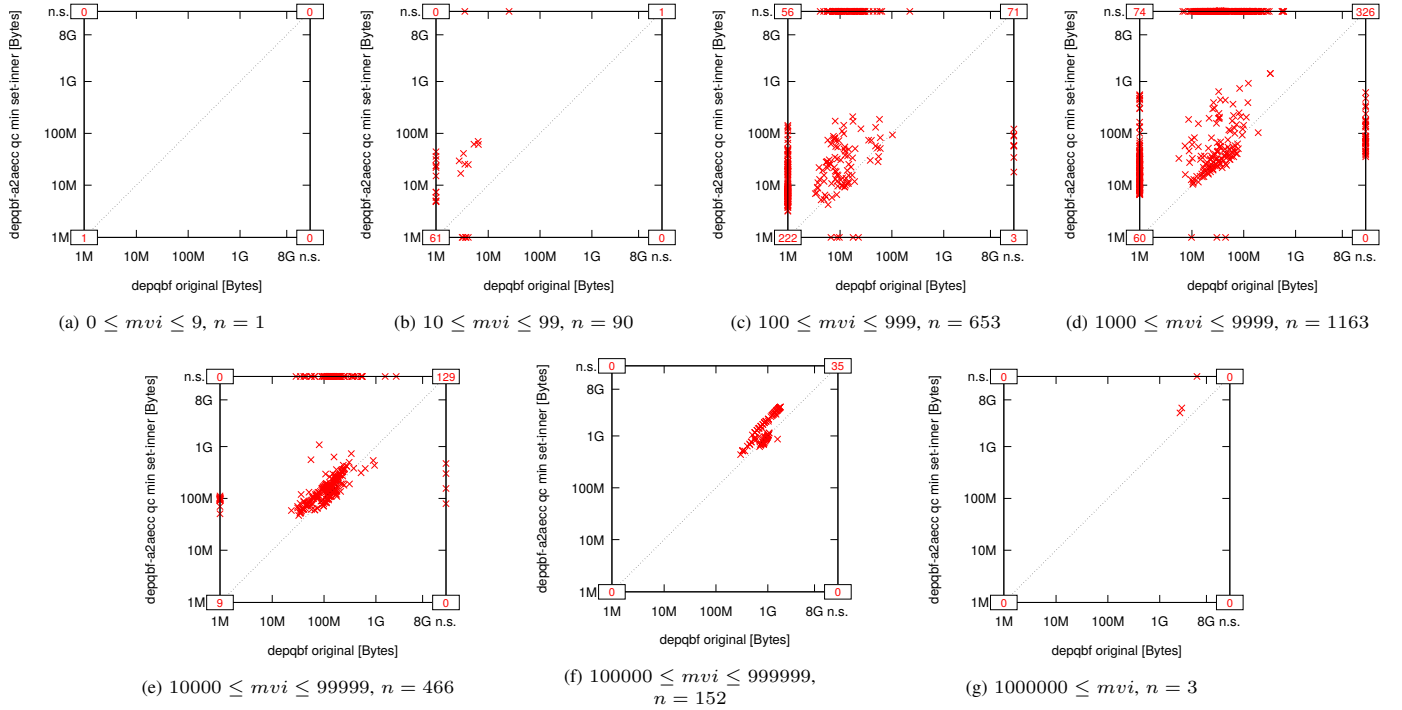


Fig. 1416: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min set-inner versus mode original partitioned by maximum variable index (memory usage in Bytes).

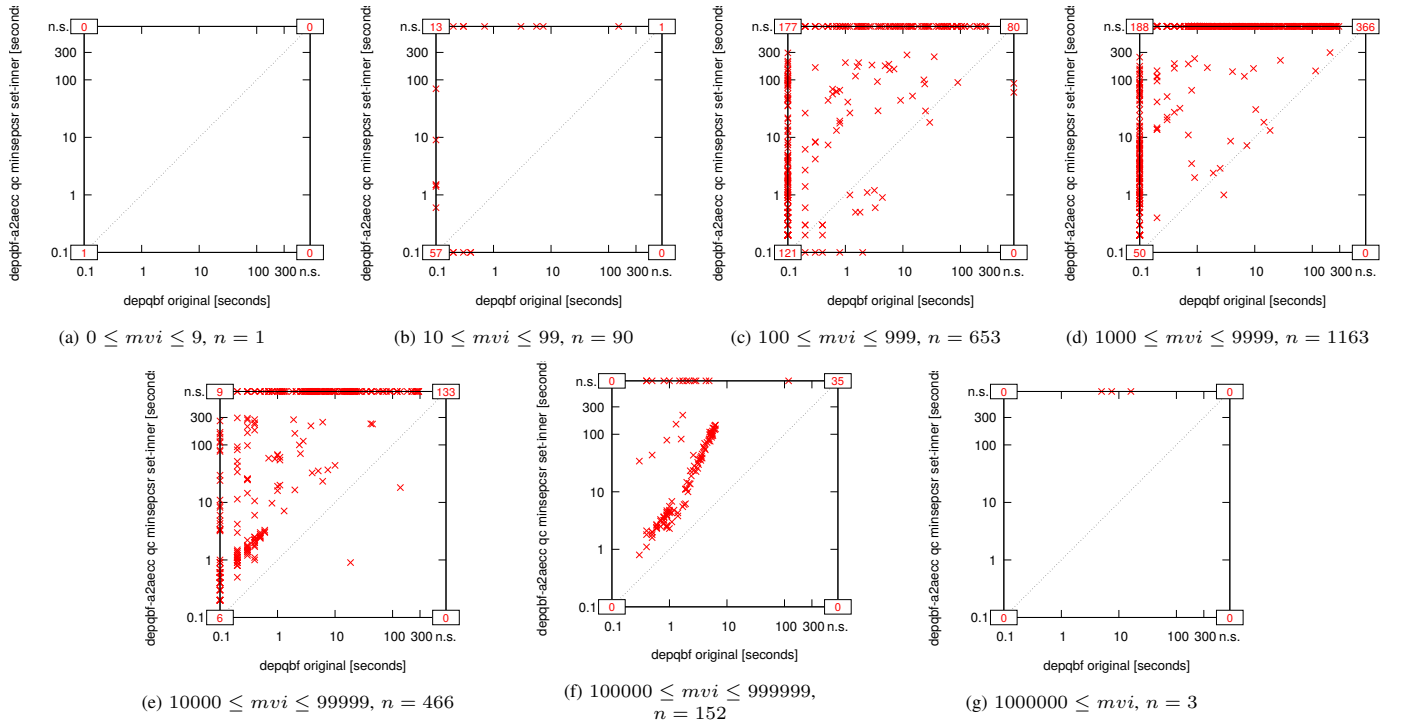


Fig. 1417: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode original partitioned by maximum variable index (run time in seconds).

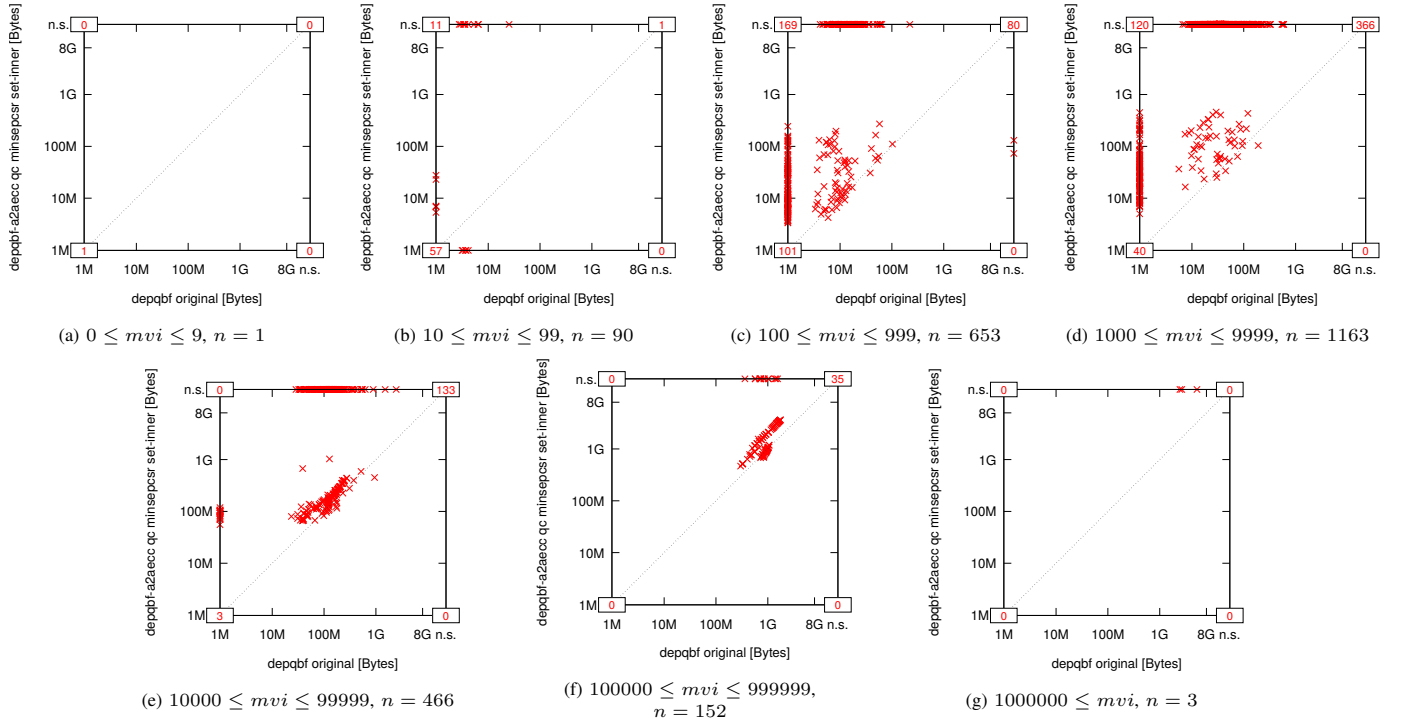


Fig. 1418: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr set-inner versus mode original partitioned by maximum variable index (memory usage in Bytes).

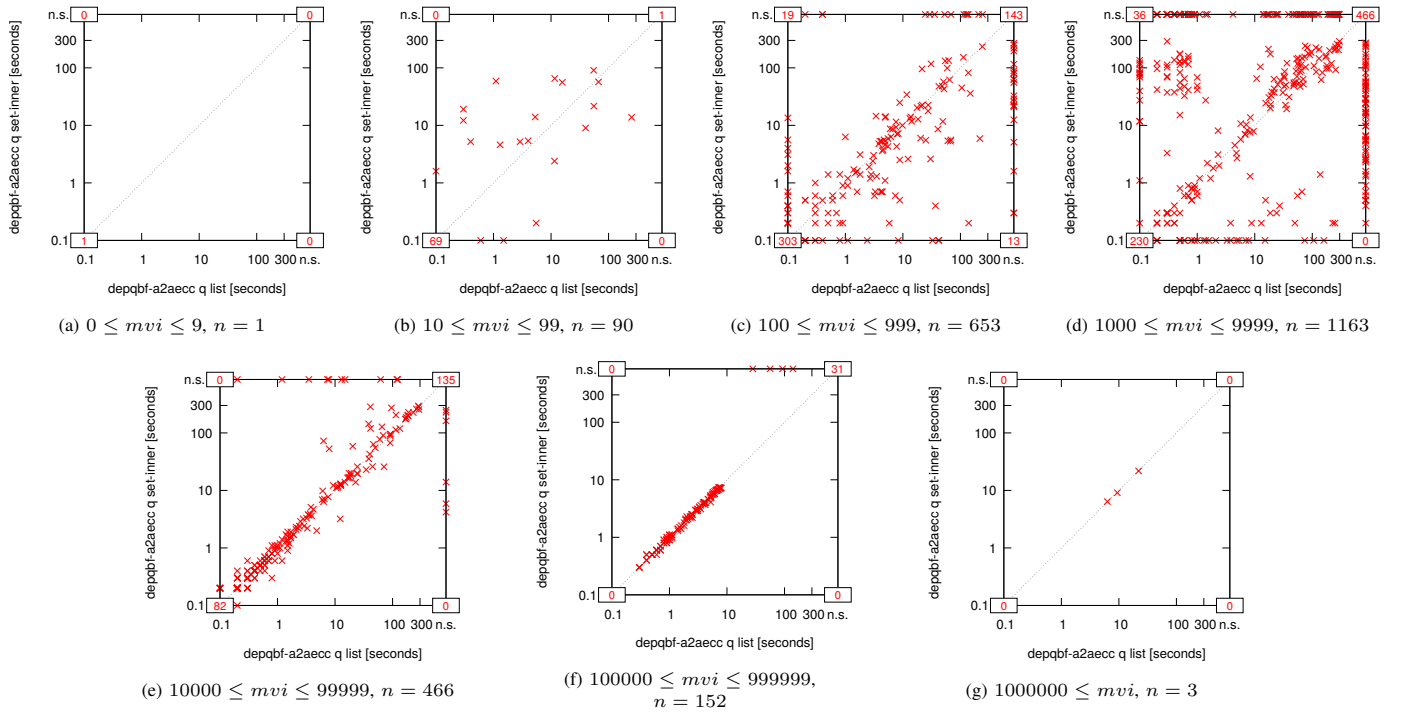


Fig. 1419: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q with set-inner versus list semantics partitioned by maximum variable index (run time in seconds).

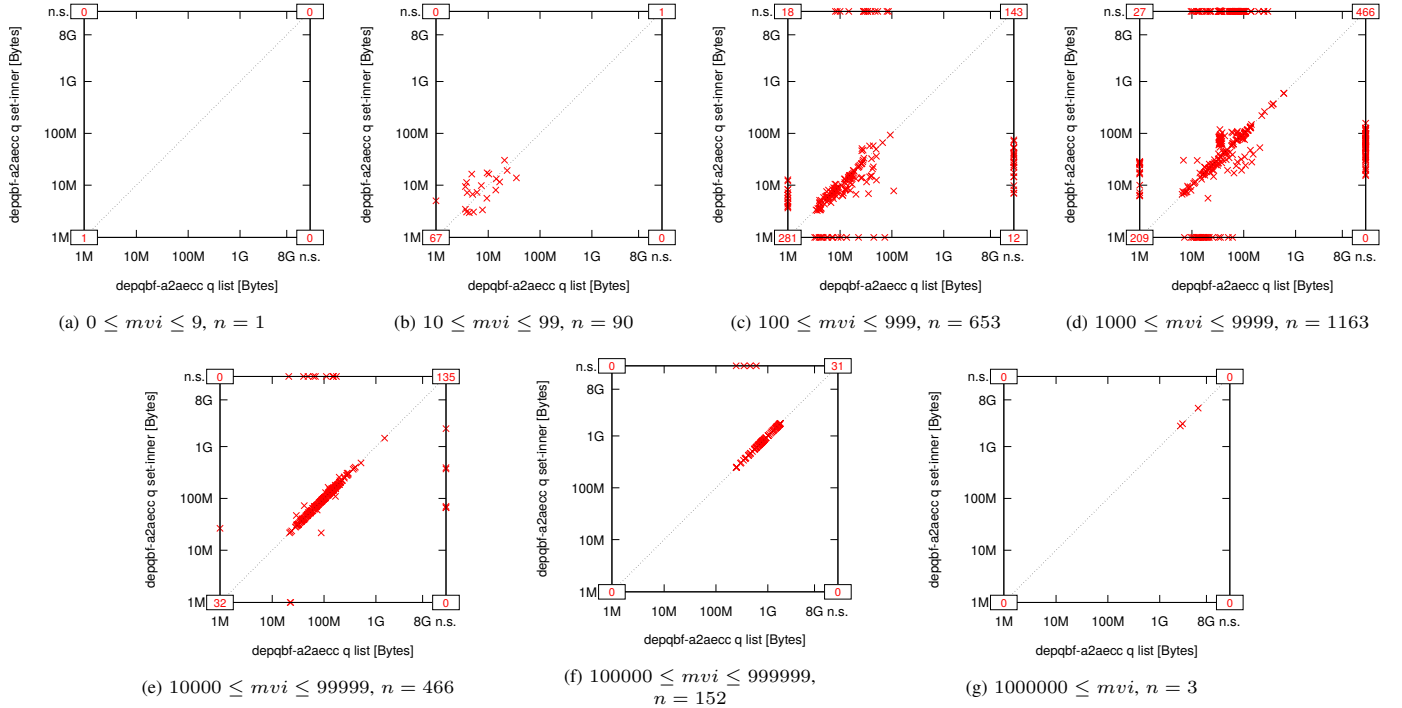


Fig. 1420: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q with set-inner versus list semantics partitioned by maximum variable index (memory usage in Bytes).

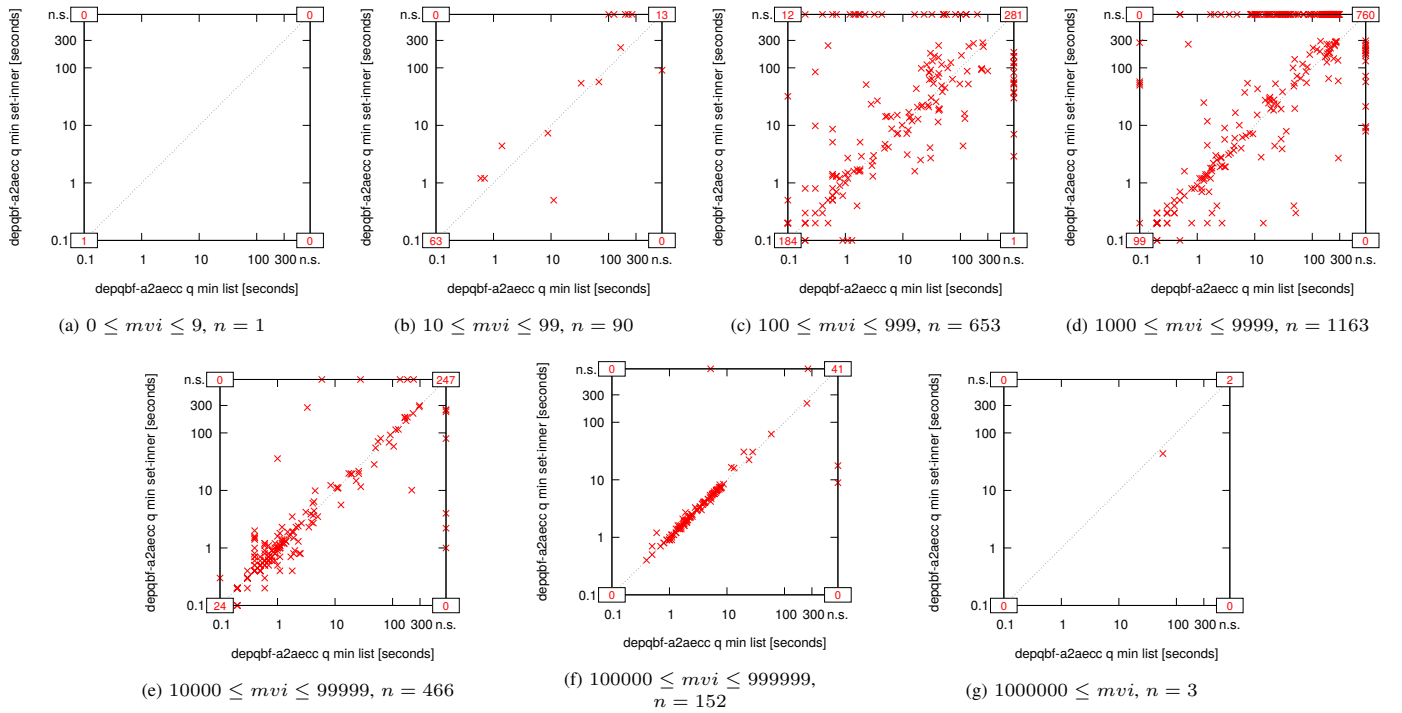


Fig. 1421: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q min with set-inner versus list semantics partitioned by maximum variable index (run time in seconds).

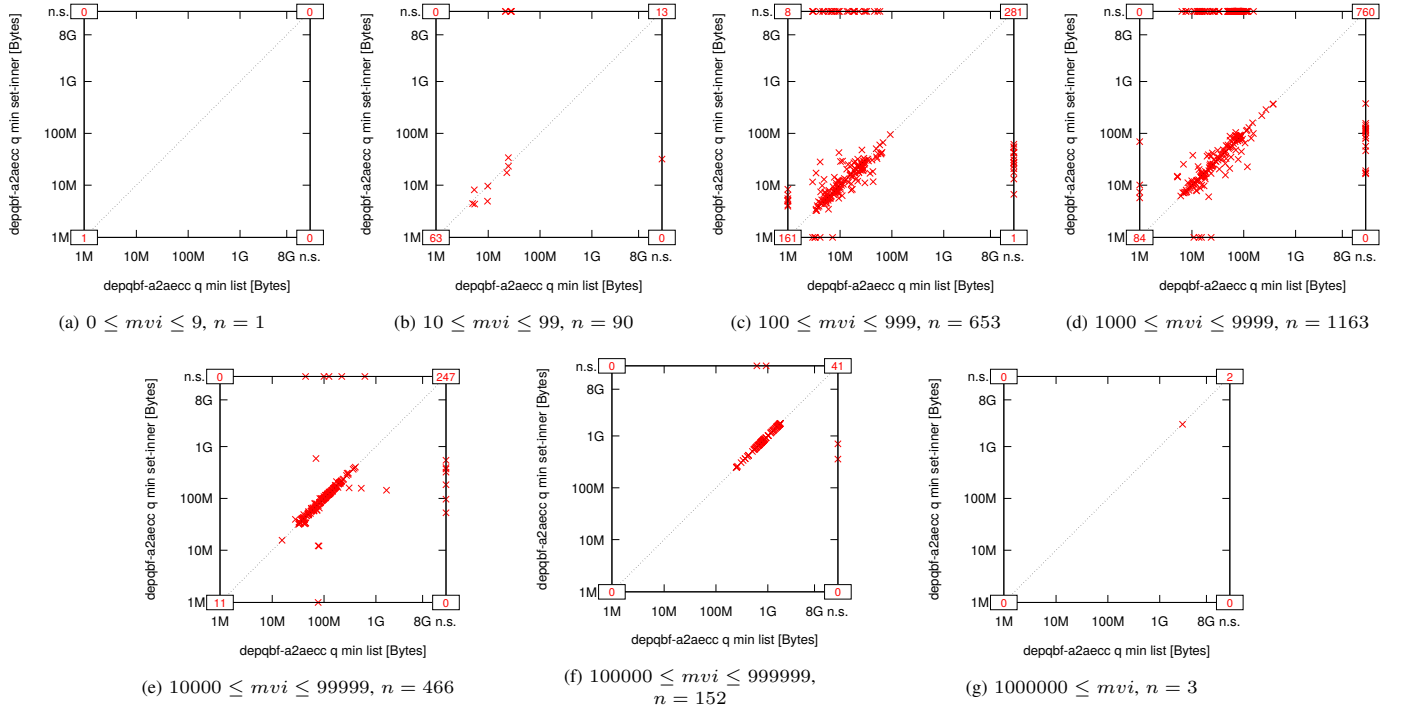


Fig. 1422: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode q min with set-inner versus list semantics partitioned by maximum variable index (memory usage in Bytes).

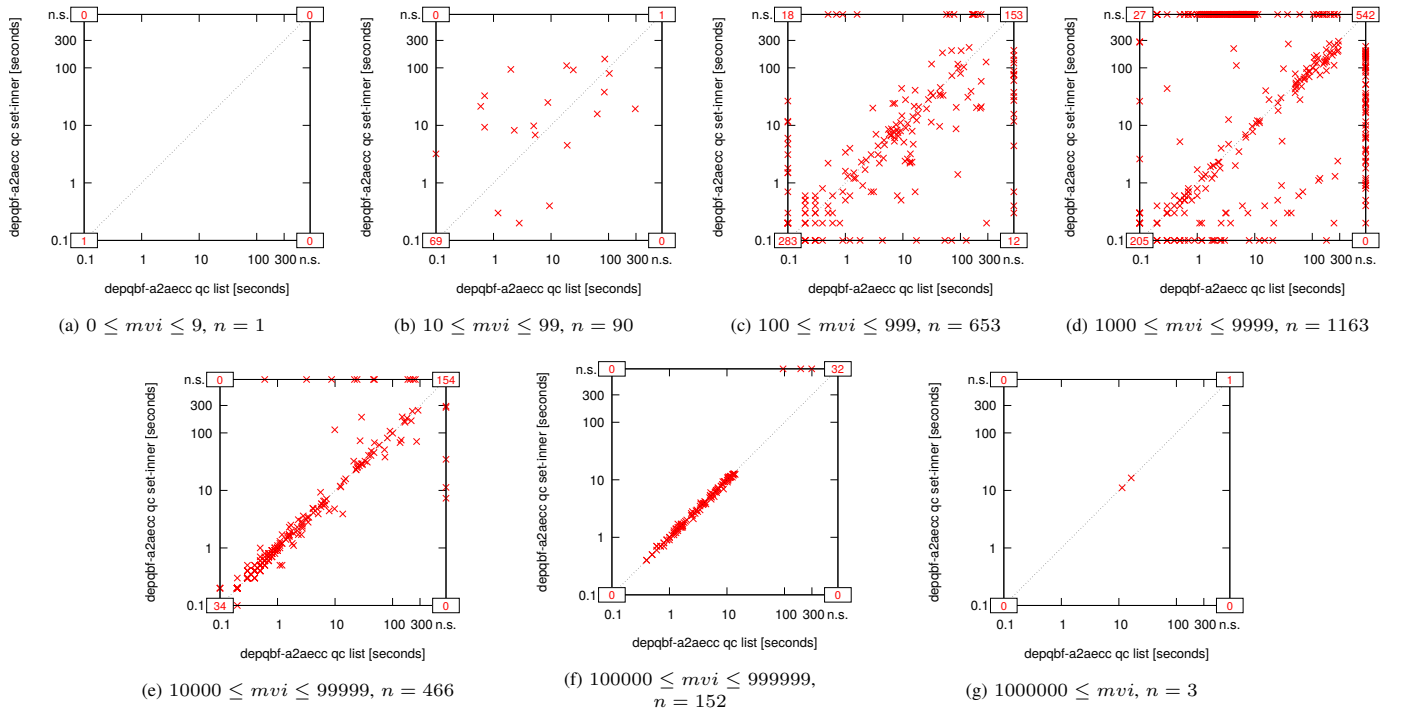


Fig. 1423: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc with set-inner versus list semantics partitioned by maximum variable index (run time in seconds).

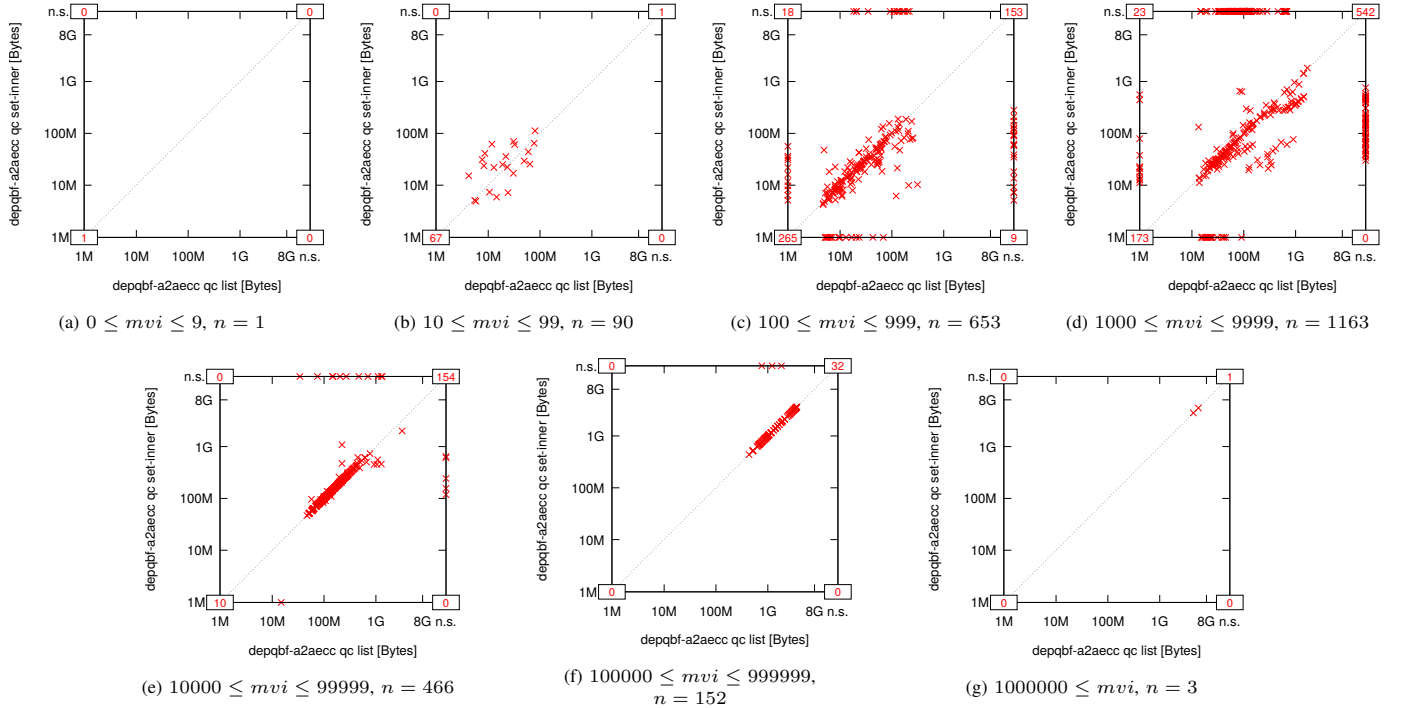


Fig. 1424: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc with set-inner versus list semantics partitioned by maximum variable index (memory usage in Bytes).

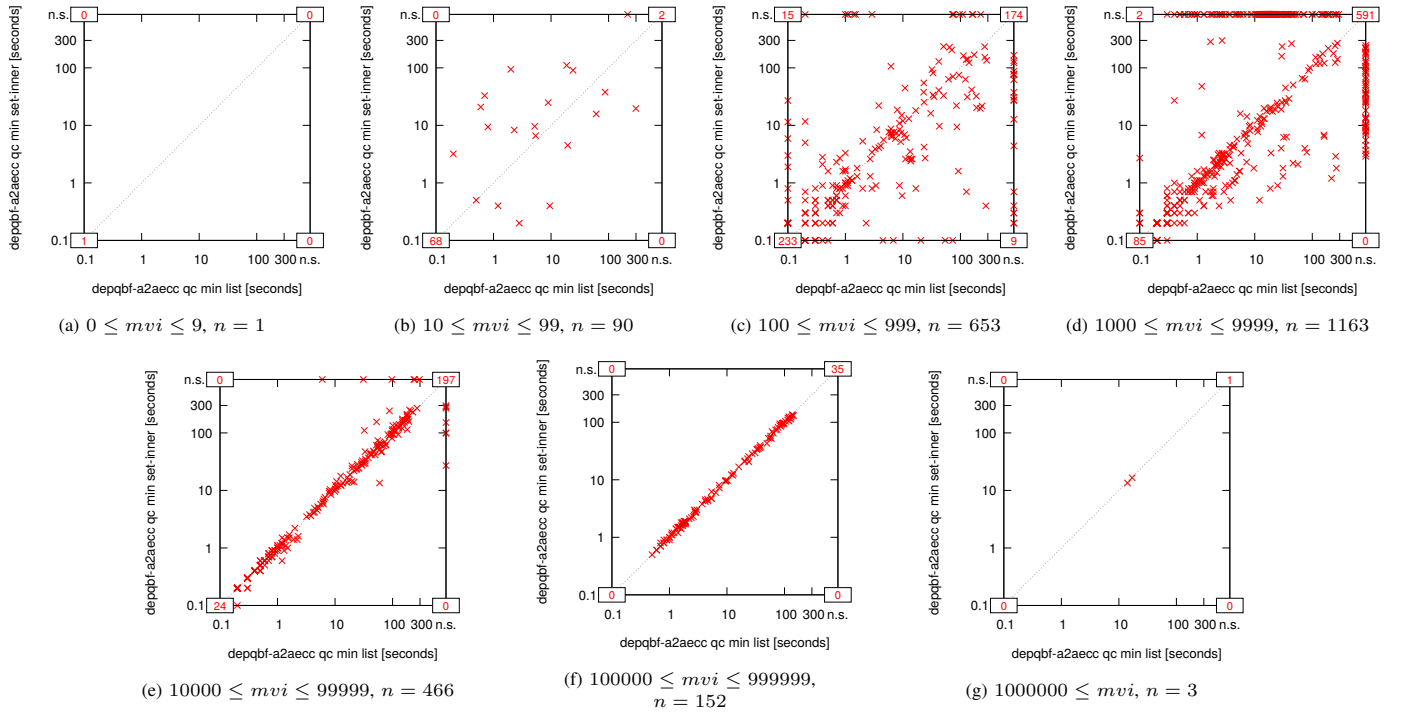


Fig. 1425: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min with set-inner versus list semantics partitioned by maximum variable index (run time in seconds).

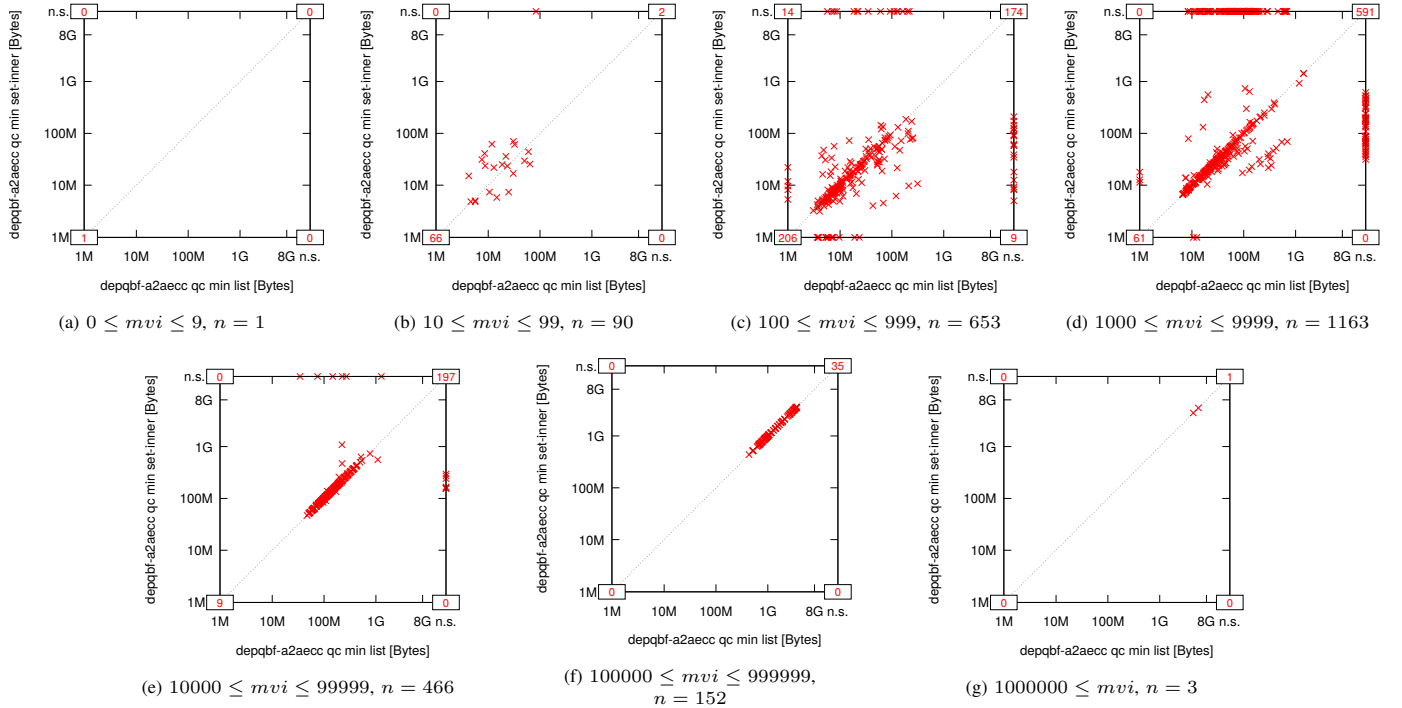


Fig. 1426: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc min with set-inner versus list semantics partitioned by maximum variable index (memory usage in Bytes).

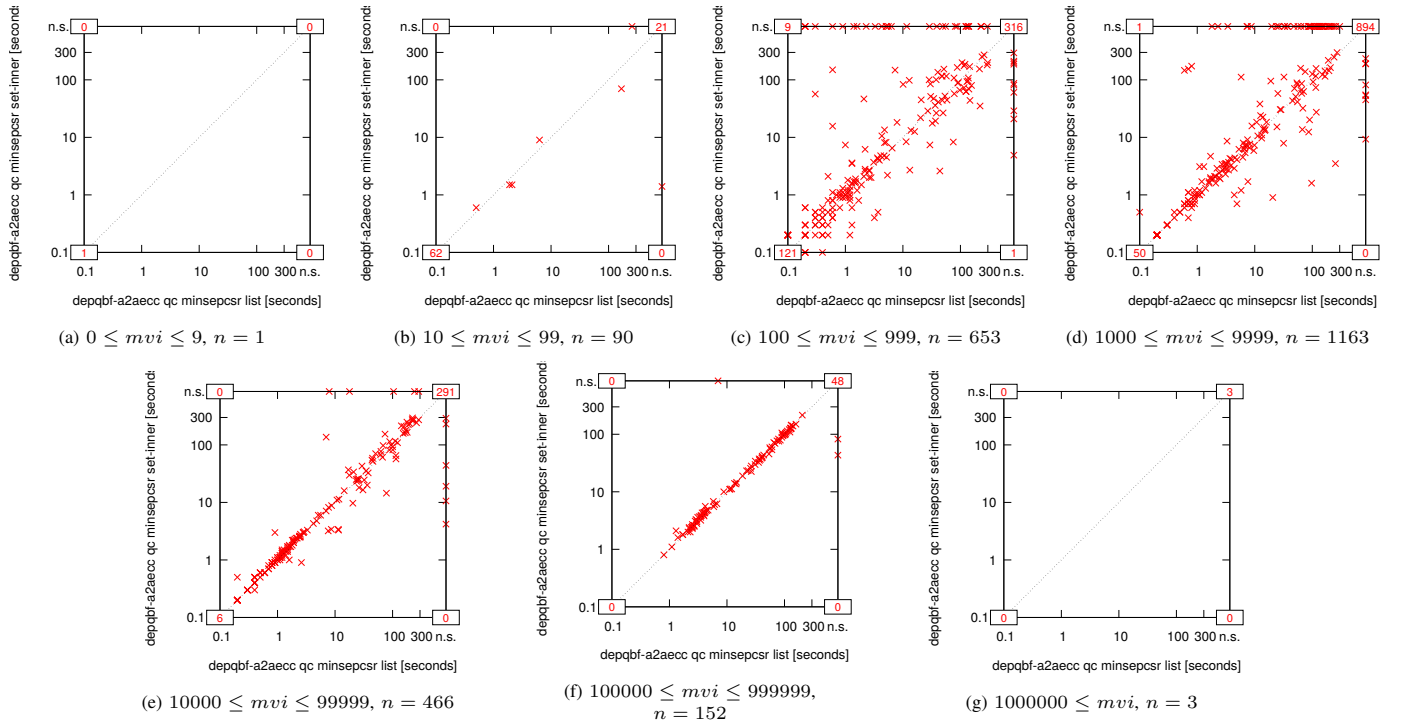


Fig. 1427: Comparing run times for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr with set-inner versus list semantics partitioned by maximum variable index (run time in seconds).

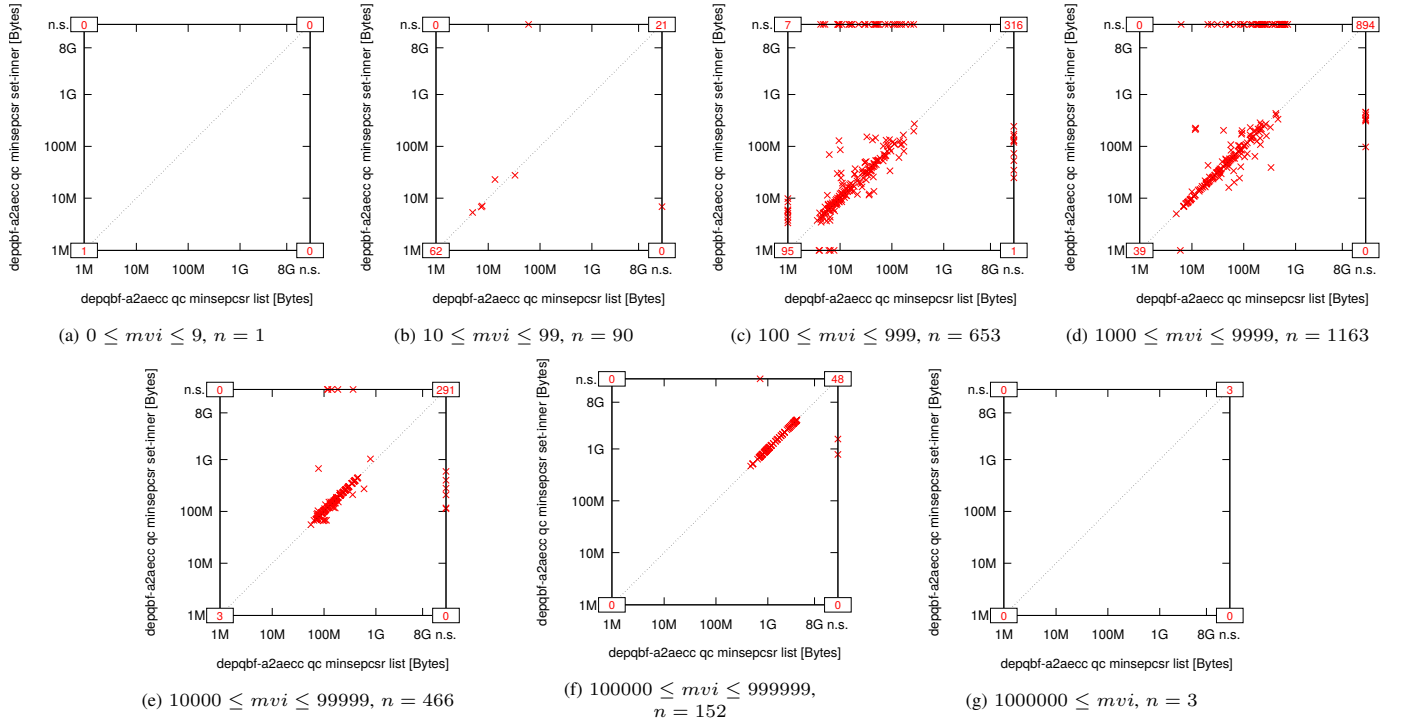


Fig. 1428: Comparing memory usage for extracting and minimizing unsatisfiable cores in DepQBF-a2aecc in mode qc minsepcsr with set-inner versus list semantics partitioned by maximum variable index (memory usage in Bytes).

APPENDIX F

SOLVING TRANSFORMED VERSUS ORIGINAL INSTANCES IN DIFFERENT QBF SOLVERS

In this appendix we show additional plots along the lines of Fig. 6 that partition the set of benchmark instances by

- benchmark suite,
- number of \forall quantified variables,
- alternation depth,
- number of clauses, and
- maximum variable index.

When we provide a number of benchmarks for a plot or a number of plots ($n = \dots$), then in this appendix this includes all benchmarks relevant to the plot including ones unsolved by any solver in our experiments.

A. *Partitioned by Benchmark Suite*

In this subsection there are 6 figures for each benchmark suite with subfigures for each solver:

- 1) Comparing run times for solving the transformed versus the original instances with list semantics.
- 2) Memory usage for 1.
- 3) As 1. but with set-inner semantics.
- 4) Memory usage for 3.
- 5) A direct comparison of solving the transformed instances with set-inner versus list semantics.
- 6) Memory usage for 5.

1) All ($n = 5342$):

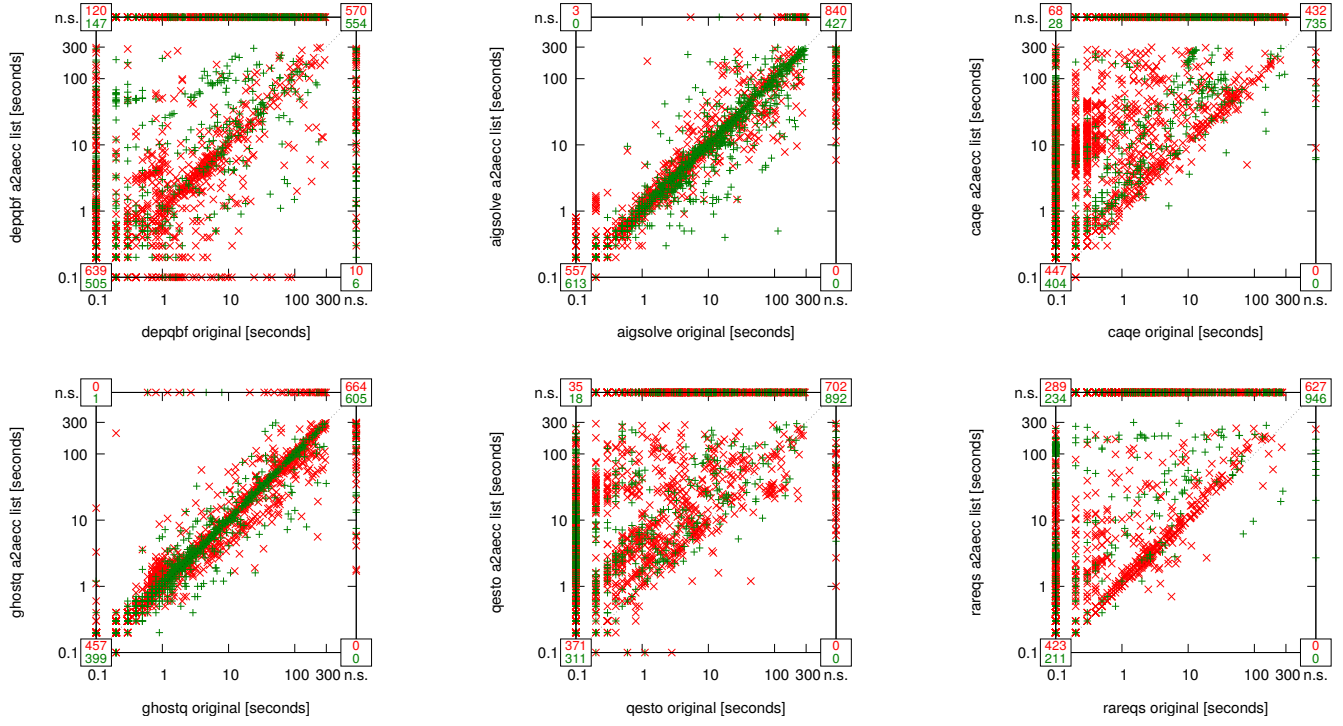


Fig. 1429: Suite All ($n = 5342$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

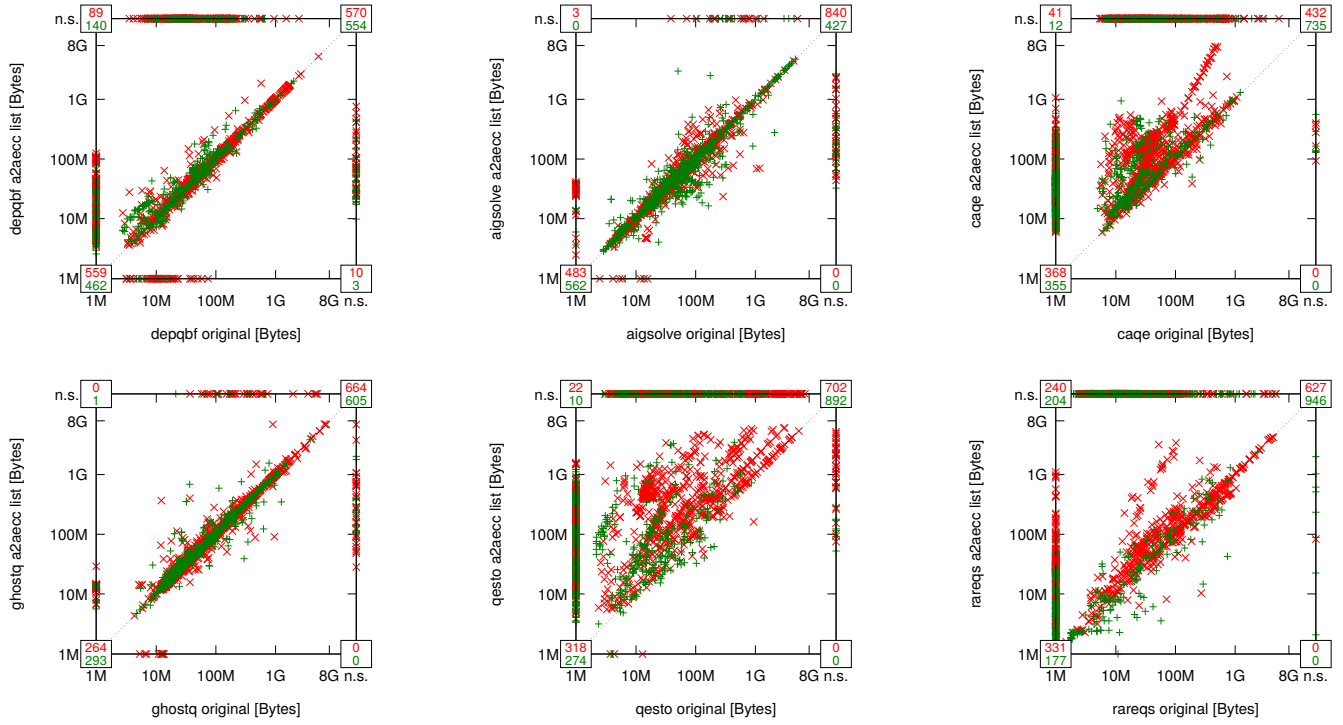


Fig. 1430: Suite All ($n = 5342$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

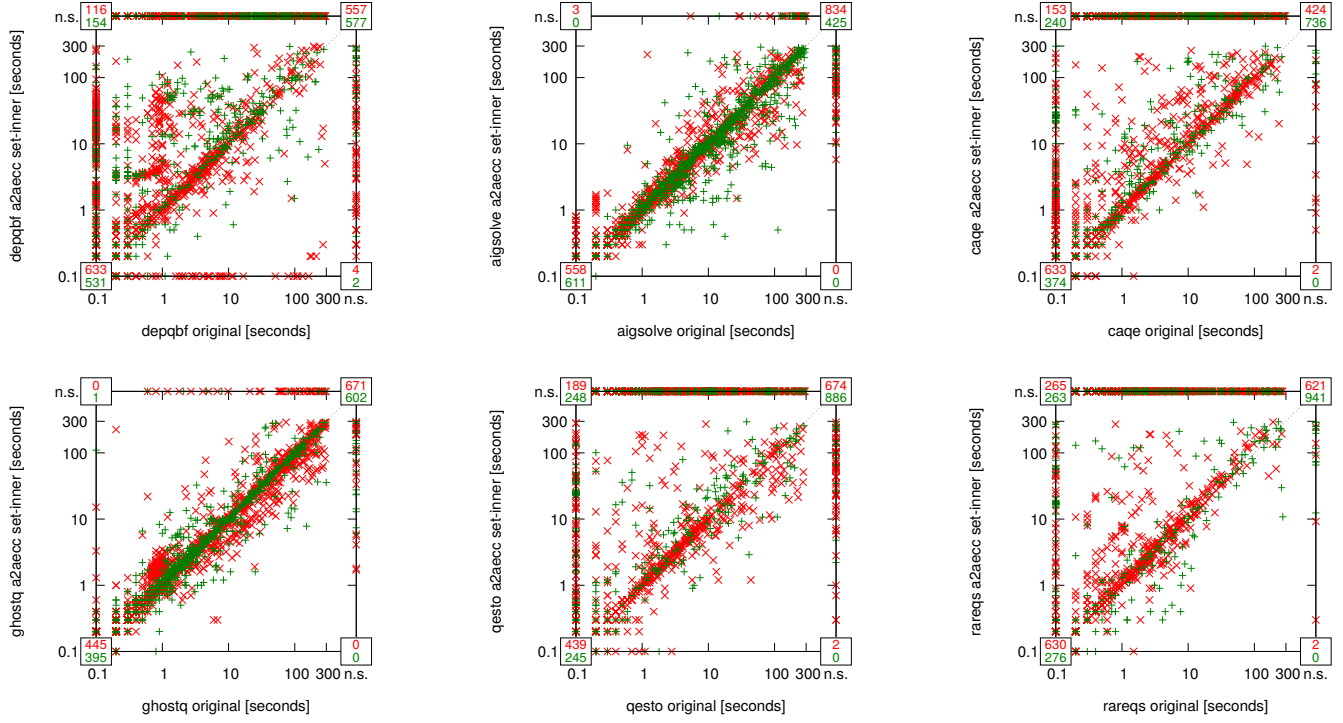


Fig. 1431: Suite All ($n = 5342$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

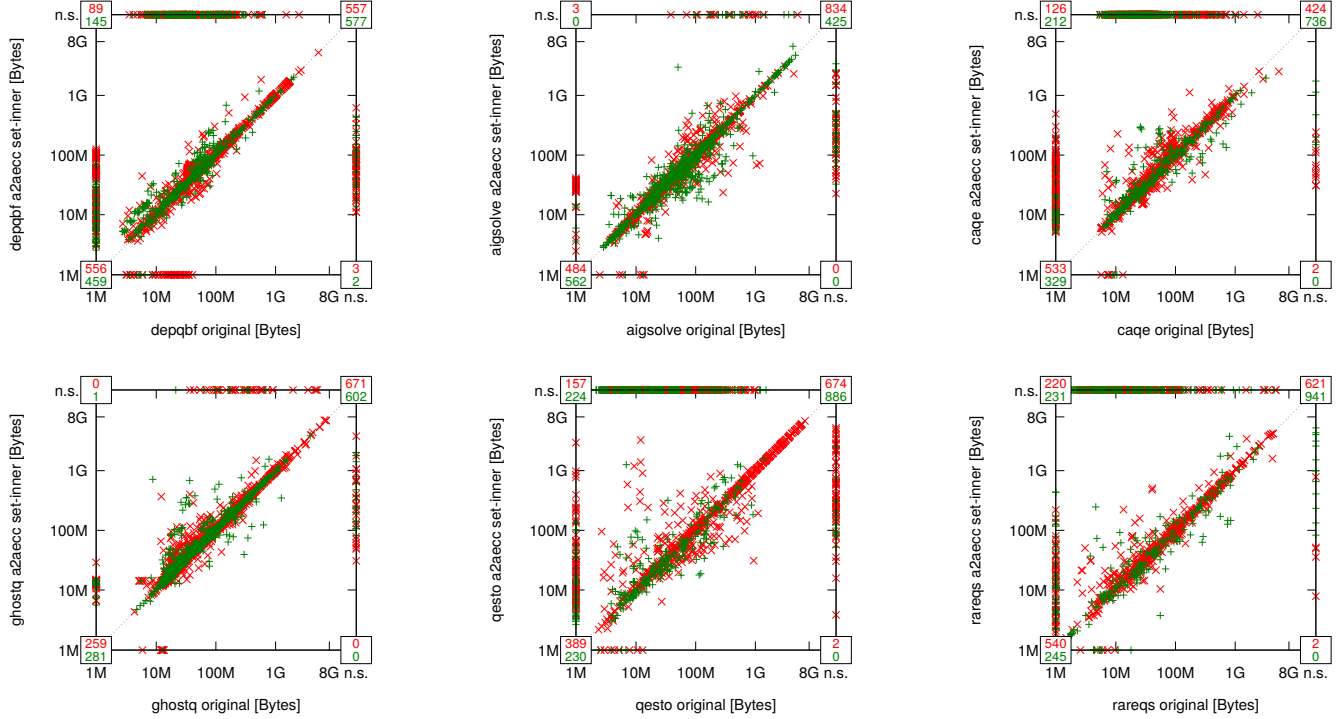


Fig. 1432: Suite All ($n = 5342$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

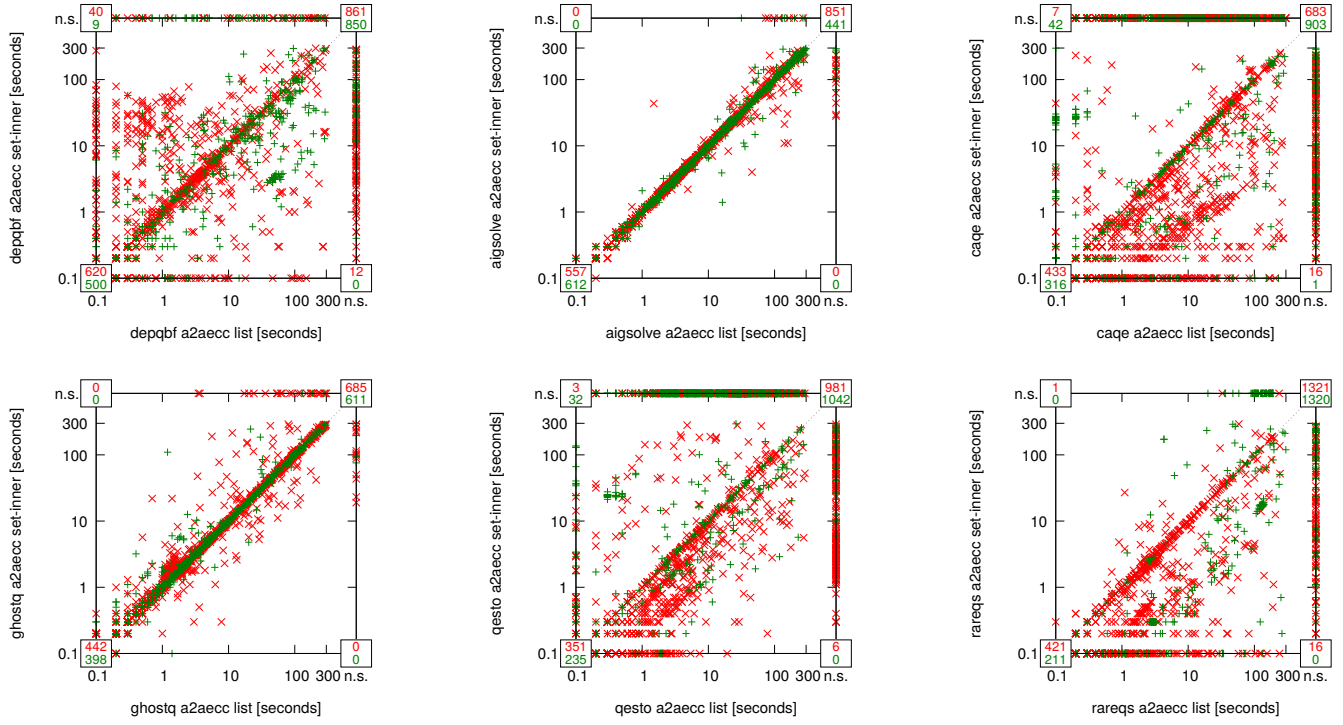


Fig. 1433: Suite All ($n = 5342$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

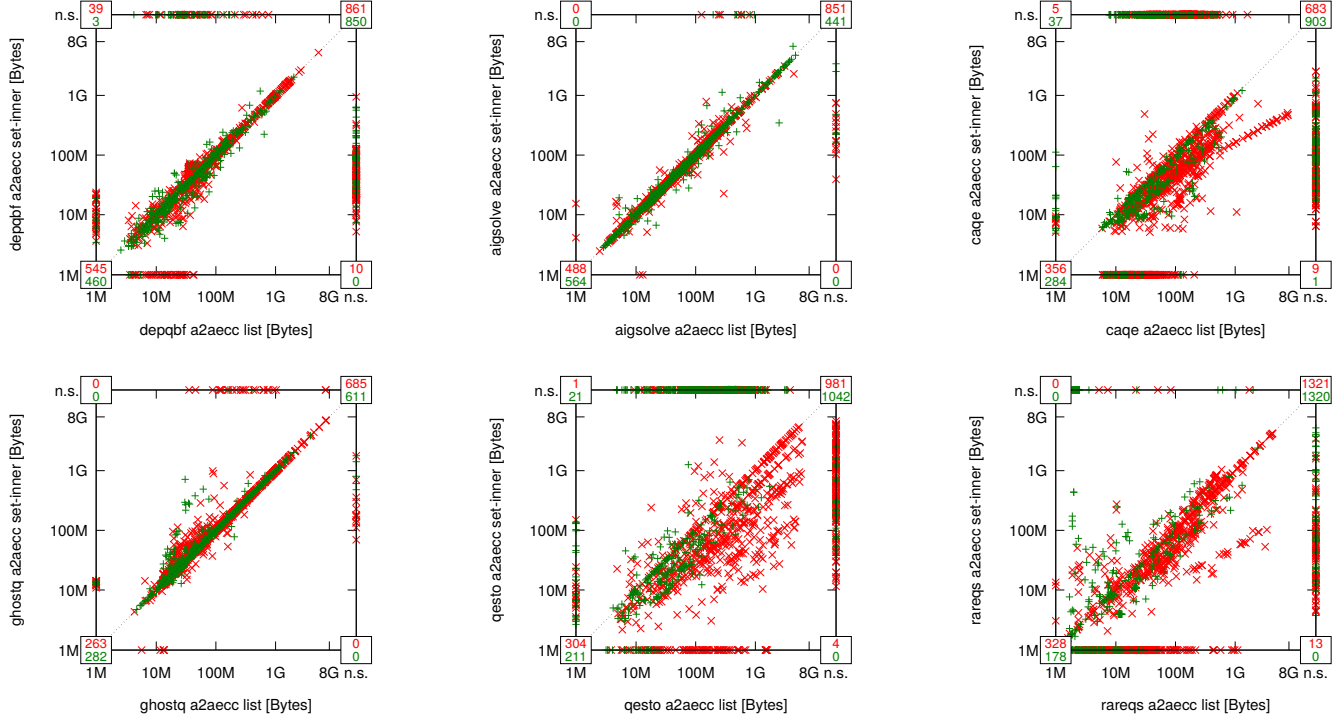


Fig. 1434: Suite All ($n = 5342$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

2) Akshay-Chakraborty-John-Shah-Rabe ($n = 20$):

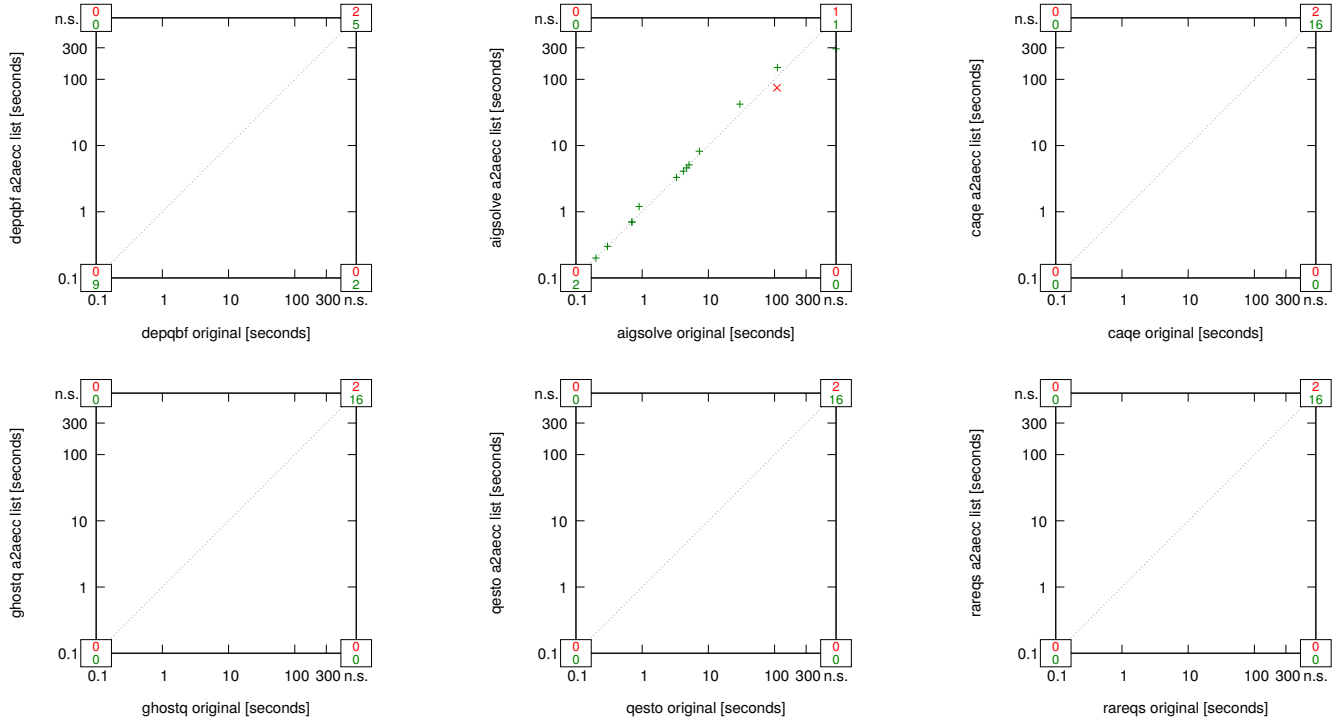


Fig. 1435: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 20$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

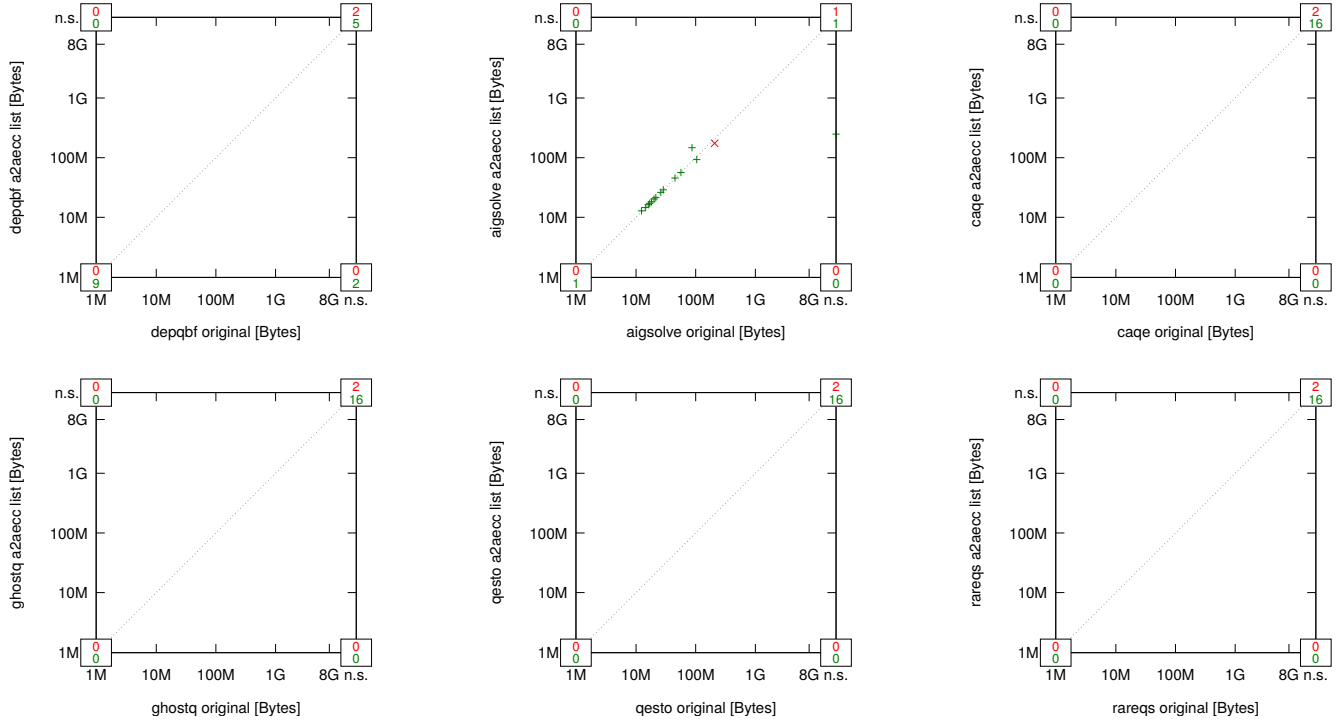


Fig. 1436: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 20$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

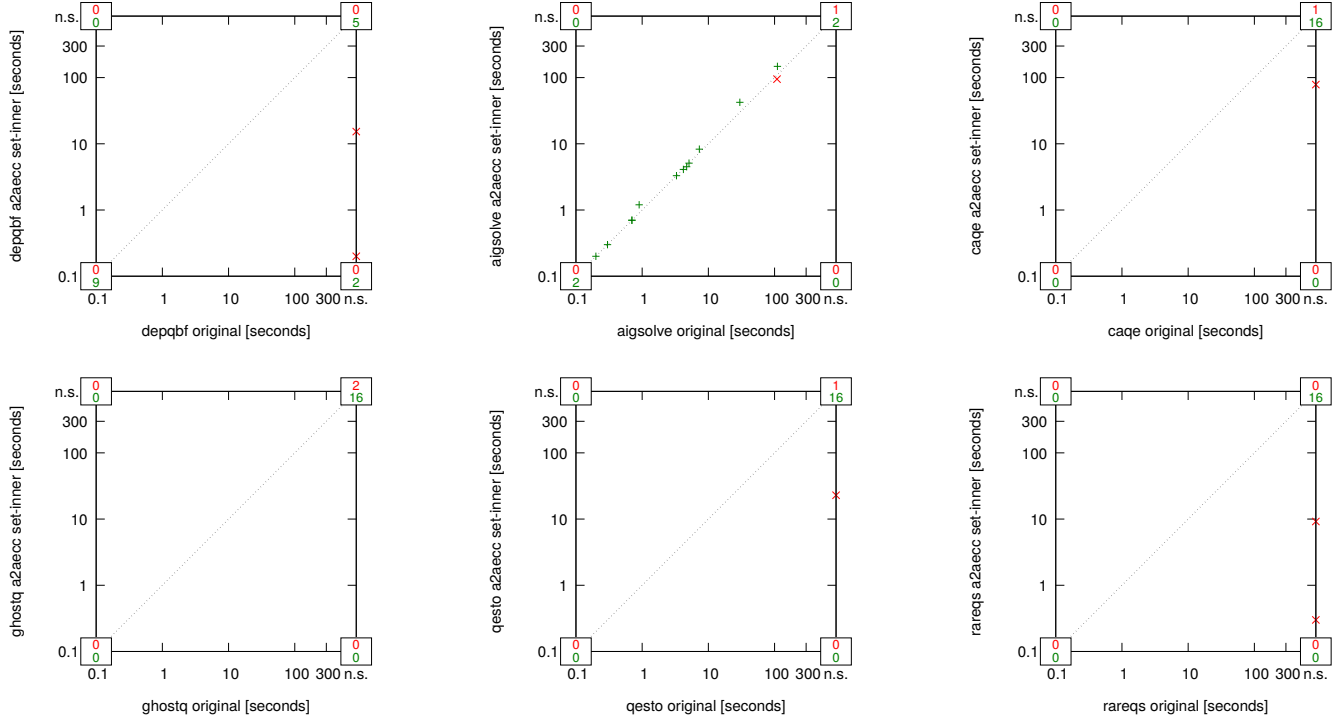


Fig. 1437: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 20$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

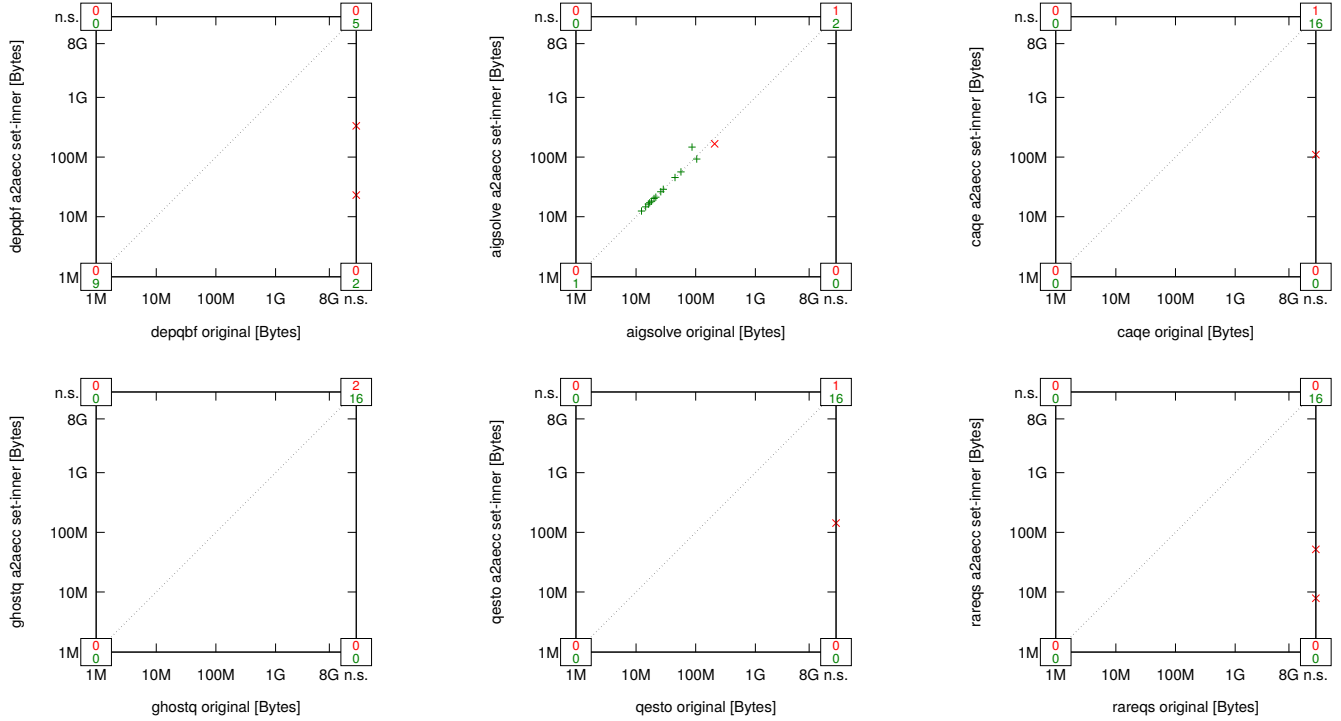


Fig. 1438: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 20$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

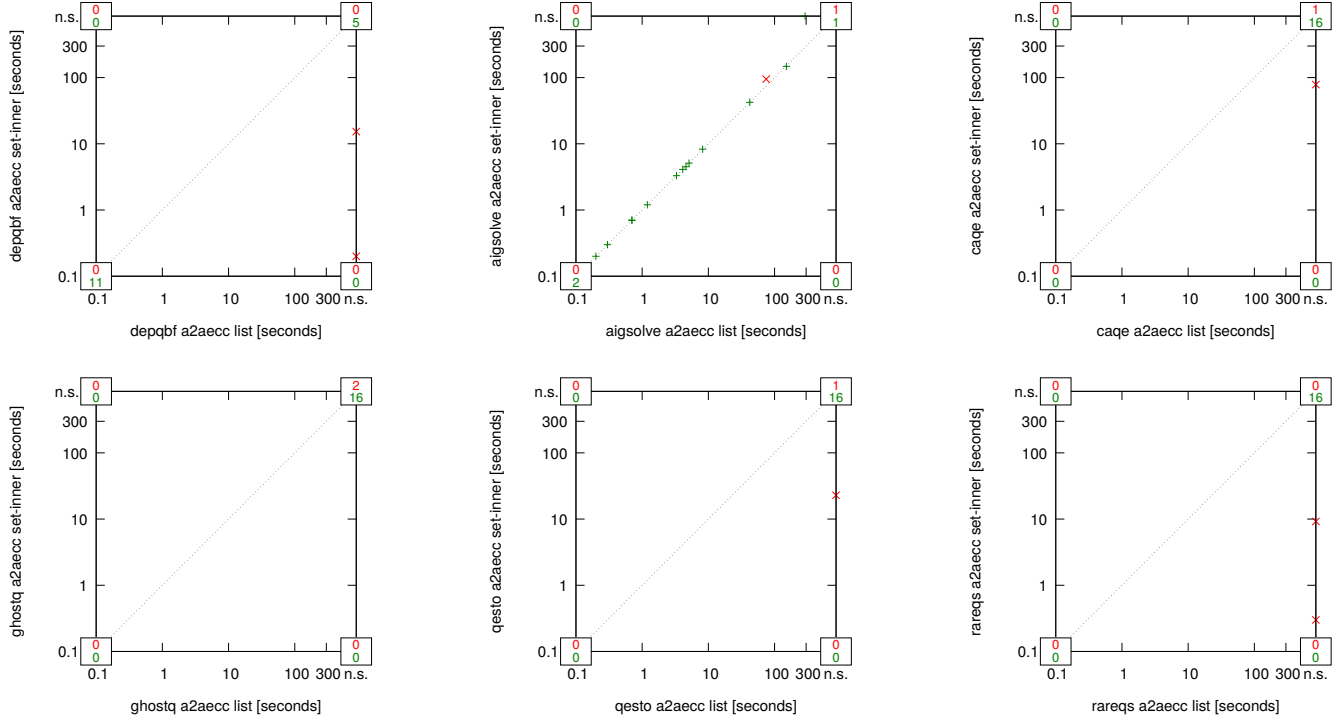


Fig. 1439: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 20$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

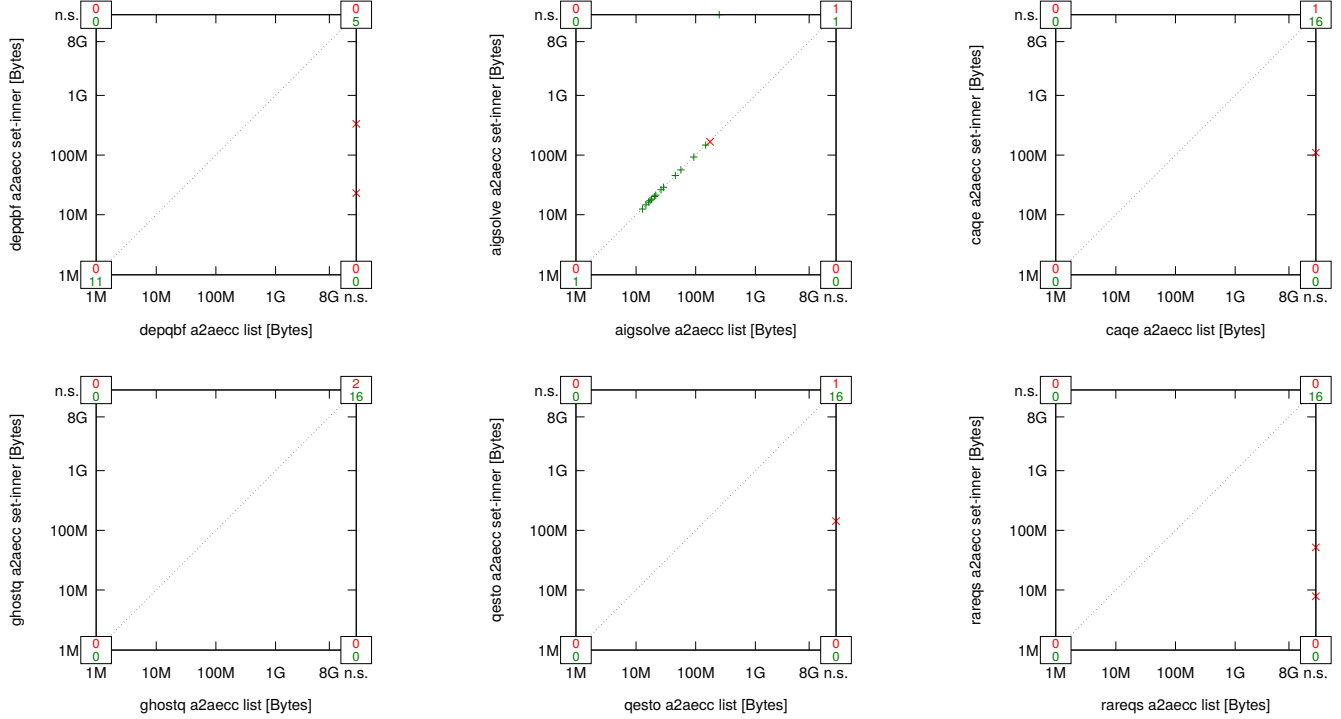


Fig. 1440: Suite Akshay-Chakraborty-John-Shah-Rabe ($n = 20$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

3) Amendola-Ricca-Truszczyński ($n = 112$):

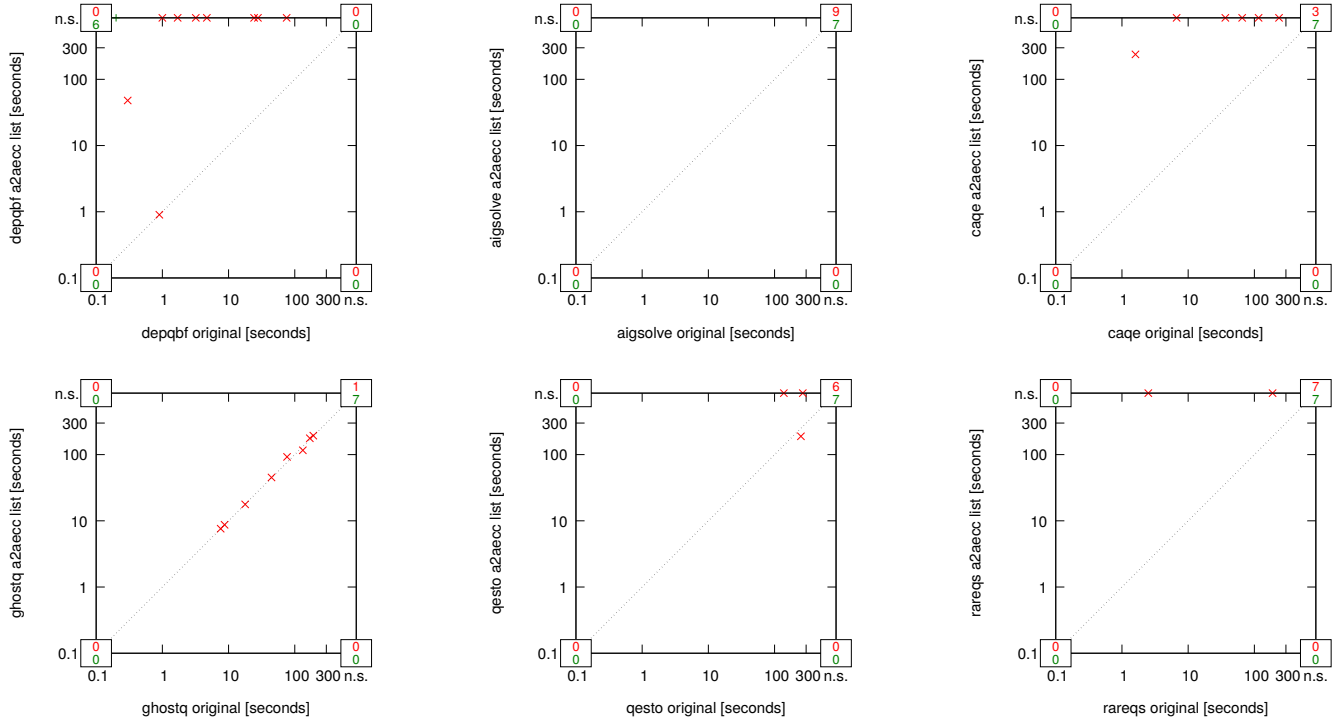


Fig. 1441: Suite Amendola-Ricca-Truszczyński ($n = 112$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

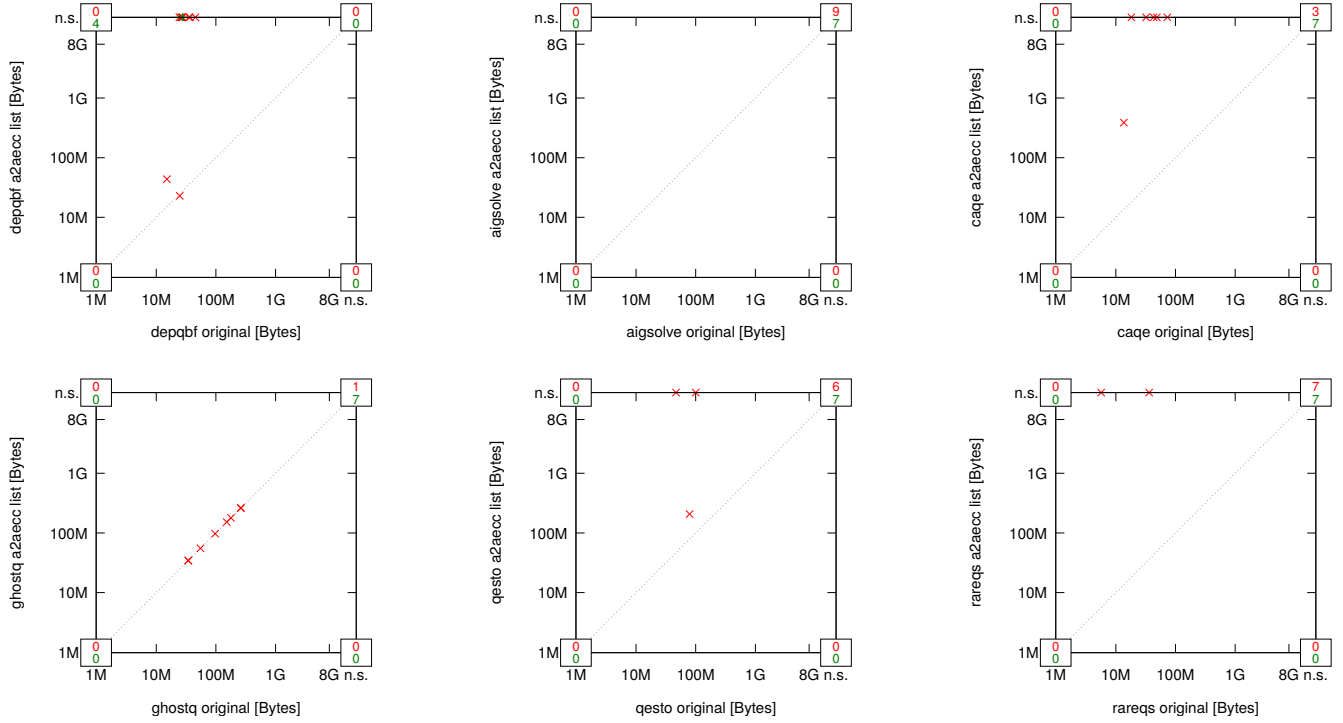


Fig. 1442: Suite Amendola-Ricca-Truszczyński ($n = 112$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

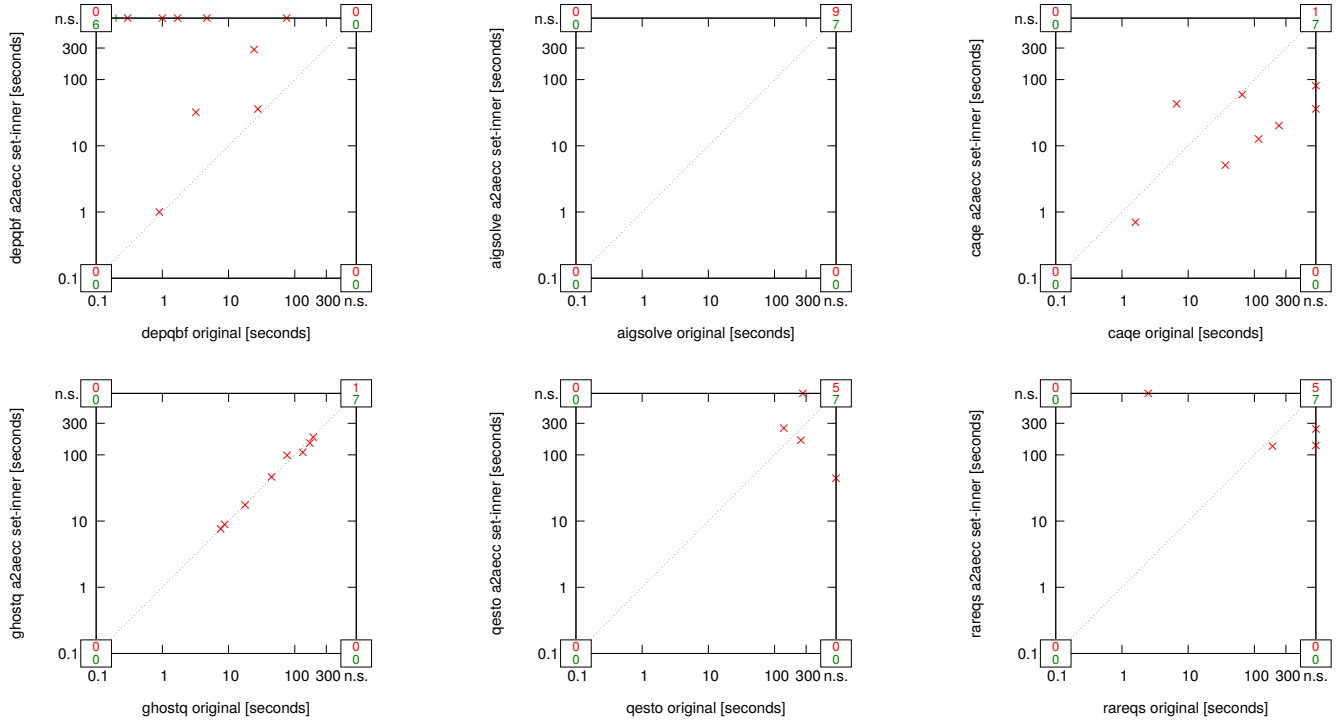


Fig. 1443: Suite Amendola-Ricca-Truszczyński ($n = 112$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

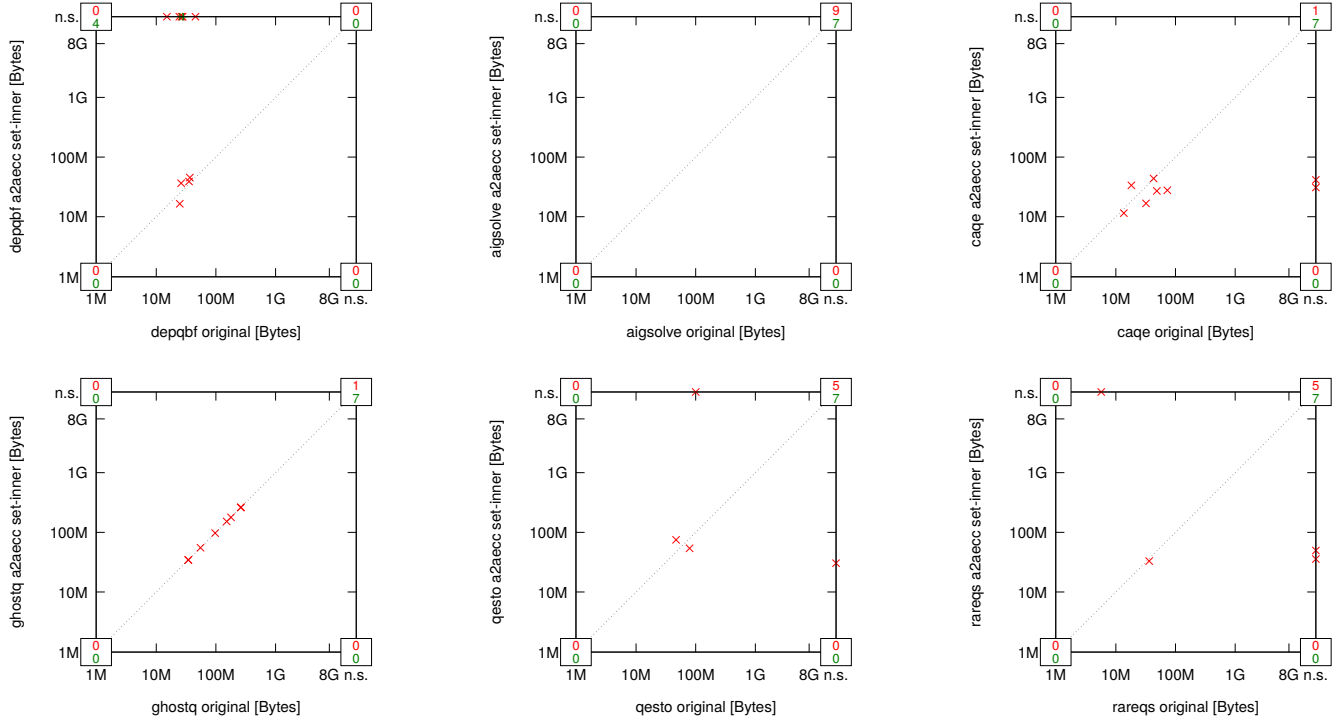


Fig. 1444: Suite Amendola-Ricca-Truszczyński ($n = 112$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

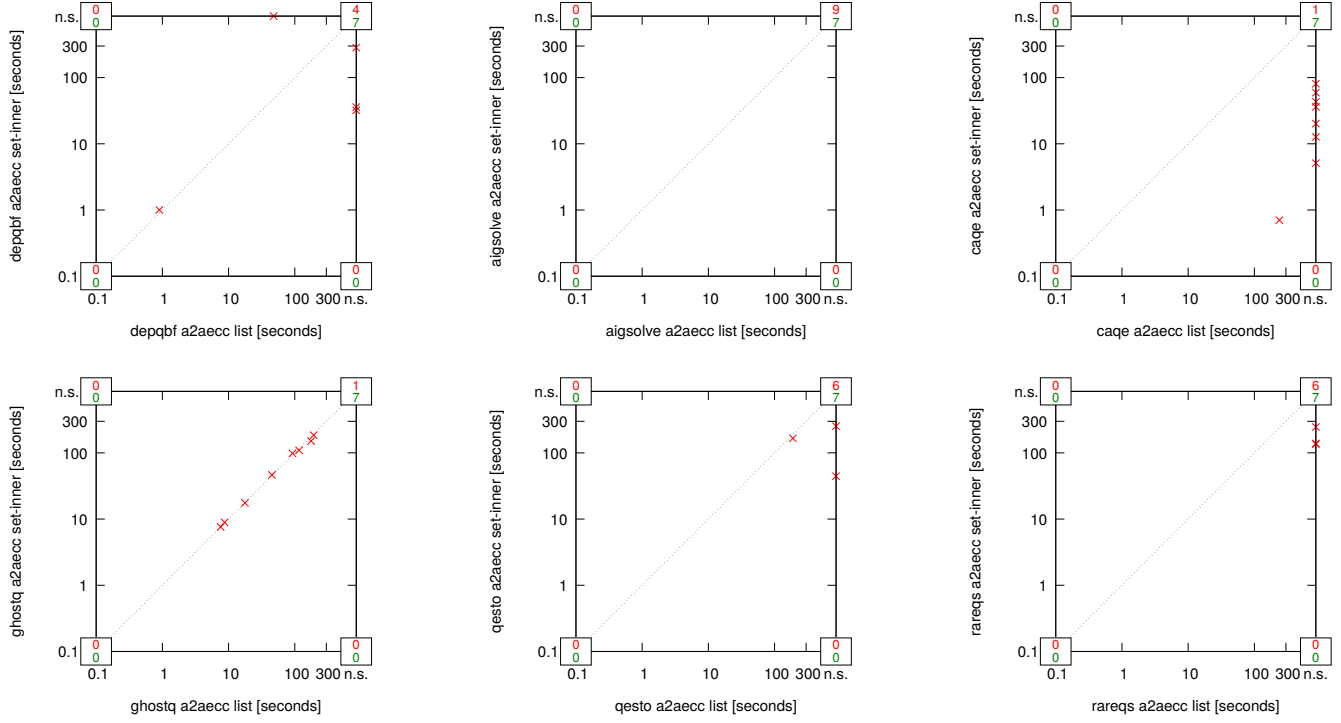


Fig. 1445: Suite Amendola-Ricca-Truszczyński ($n = 112$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

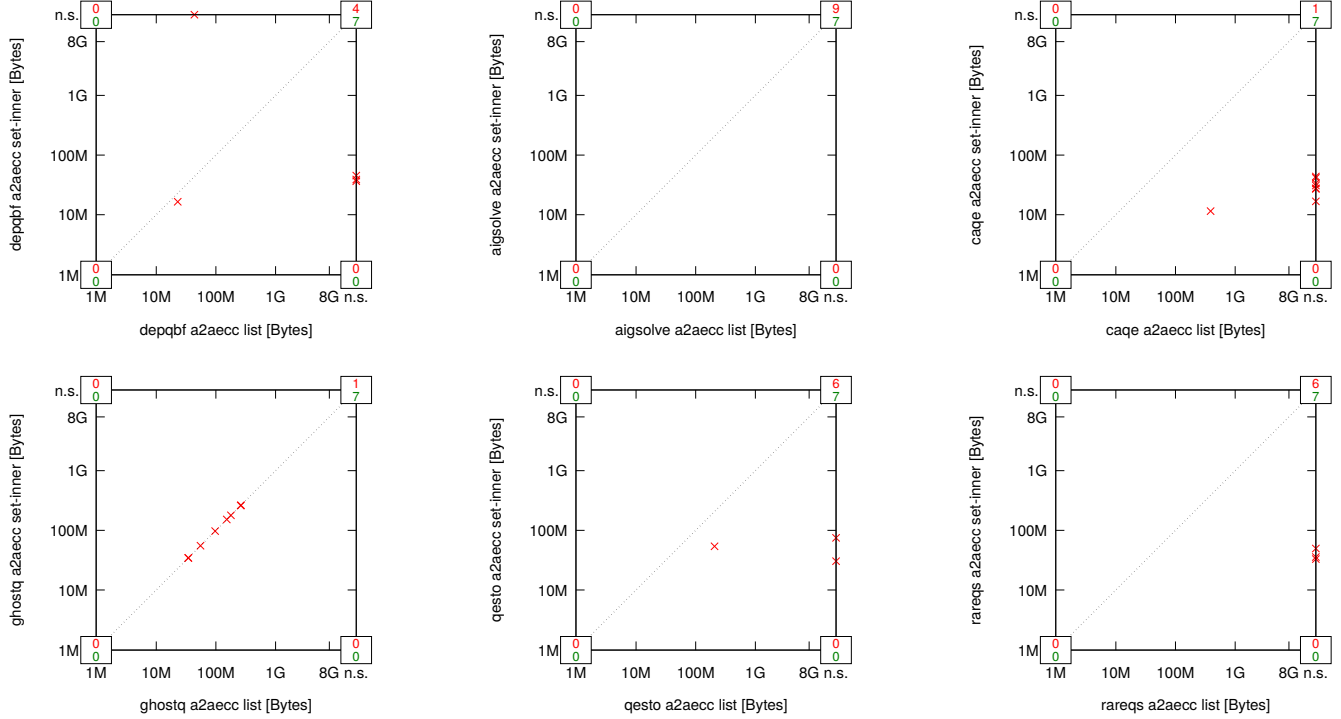


Fig. 1446: Suite Amendola-Ricca-Truszczyński ($n = 112$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

4) Ansotegui ($n = 38$):

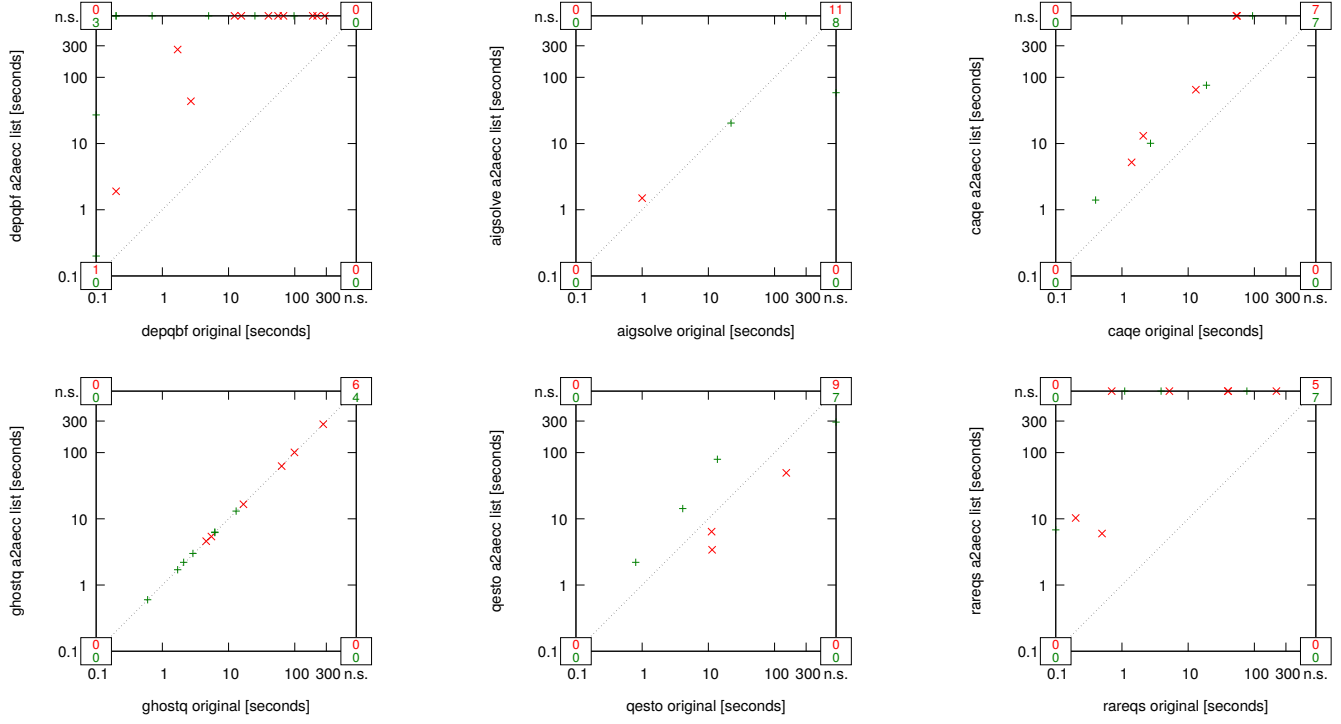


Fig. 1447: Suite Ansotegui ($n = 38$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

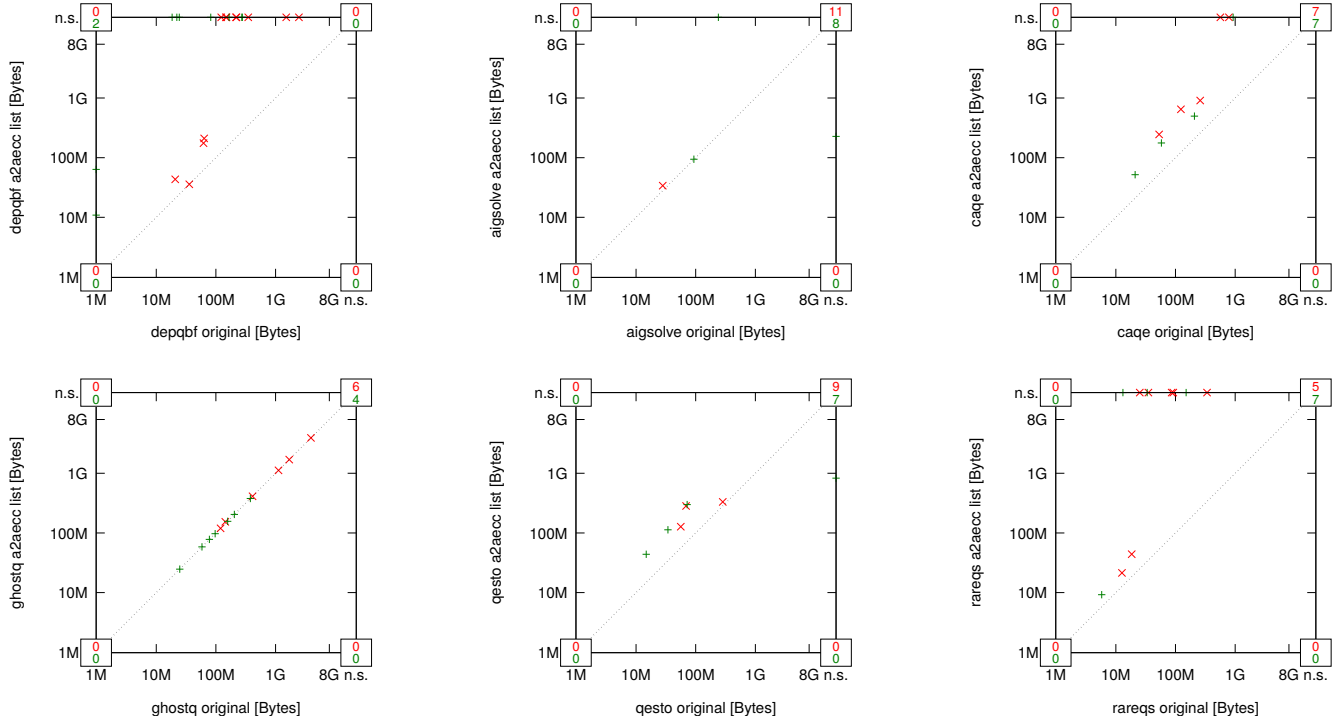


Fig. 1448: Suite Ansotegui ($n = 38$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

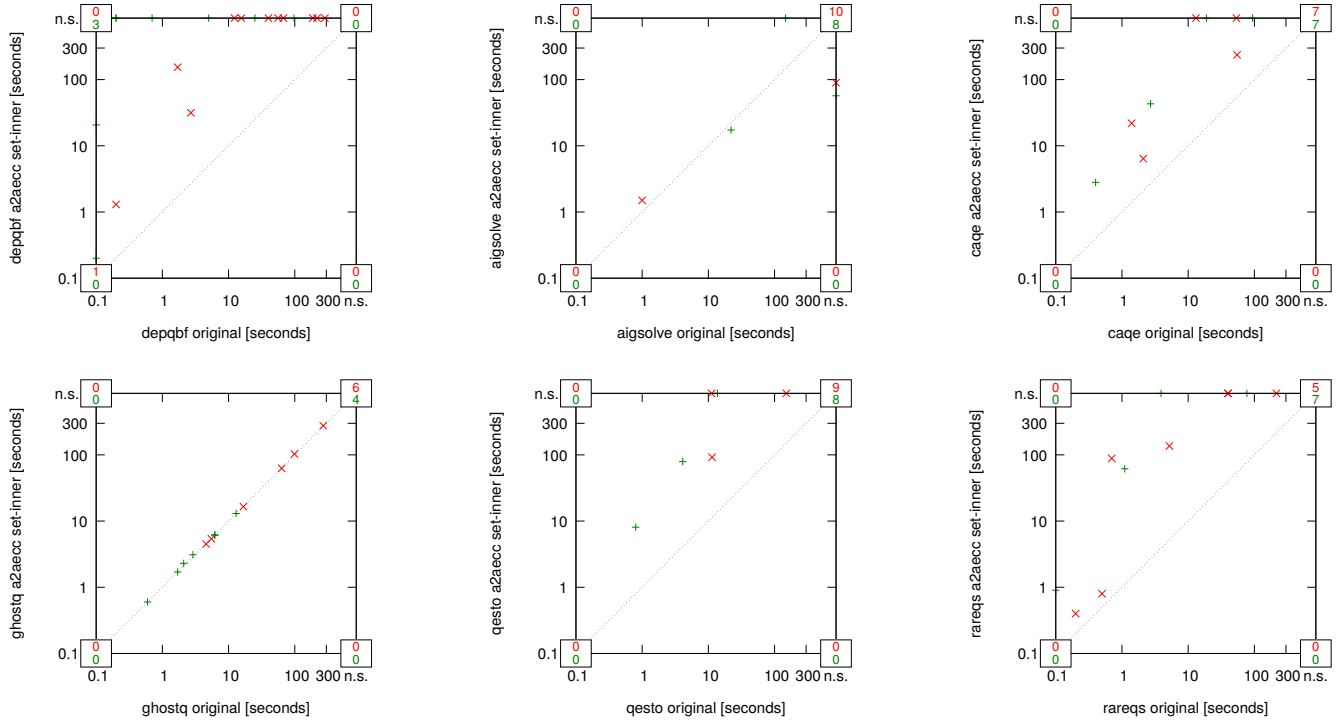


Fig. 1449: Suite Ansotegui ($n = 38$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

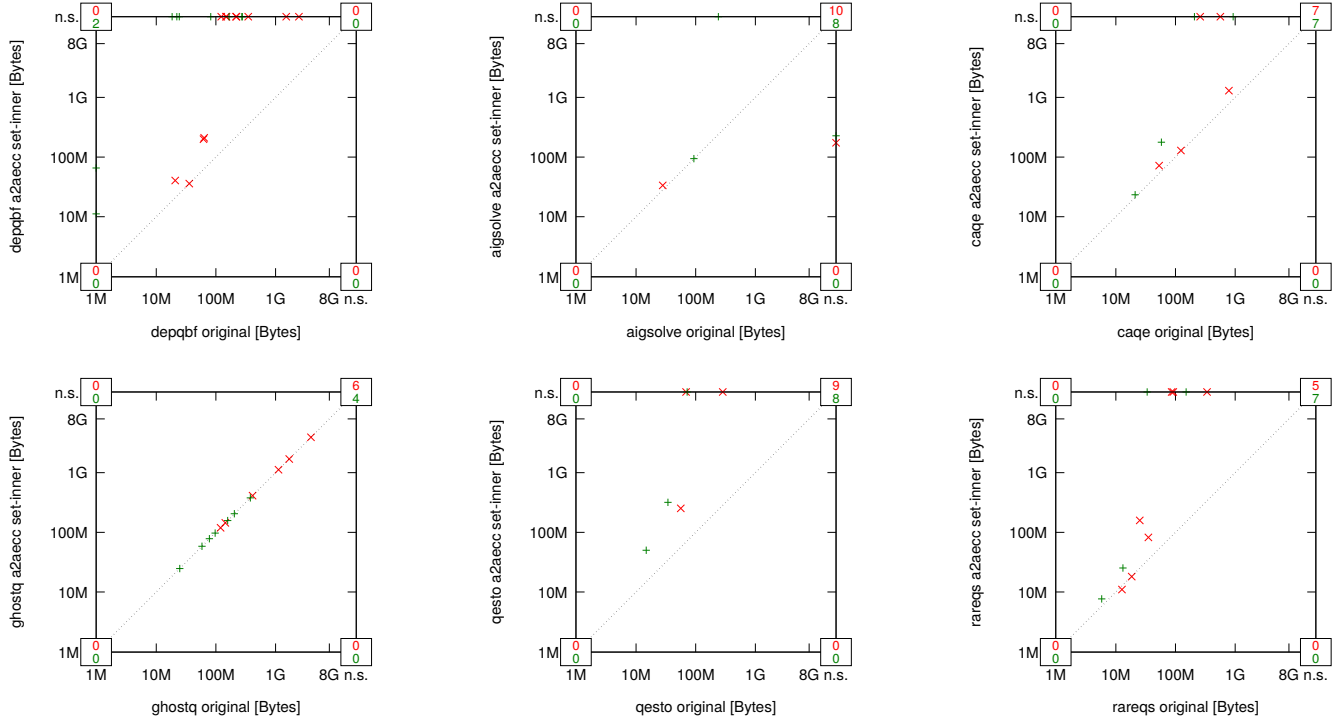


Fig. 1450: Suite Ansotegui ($n = 38$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

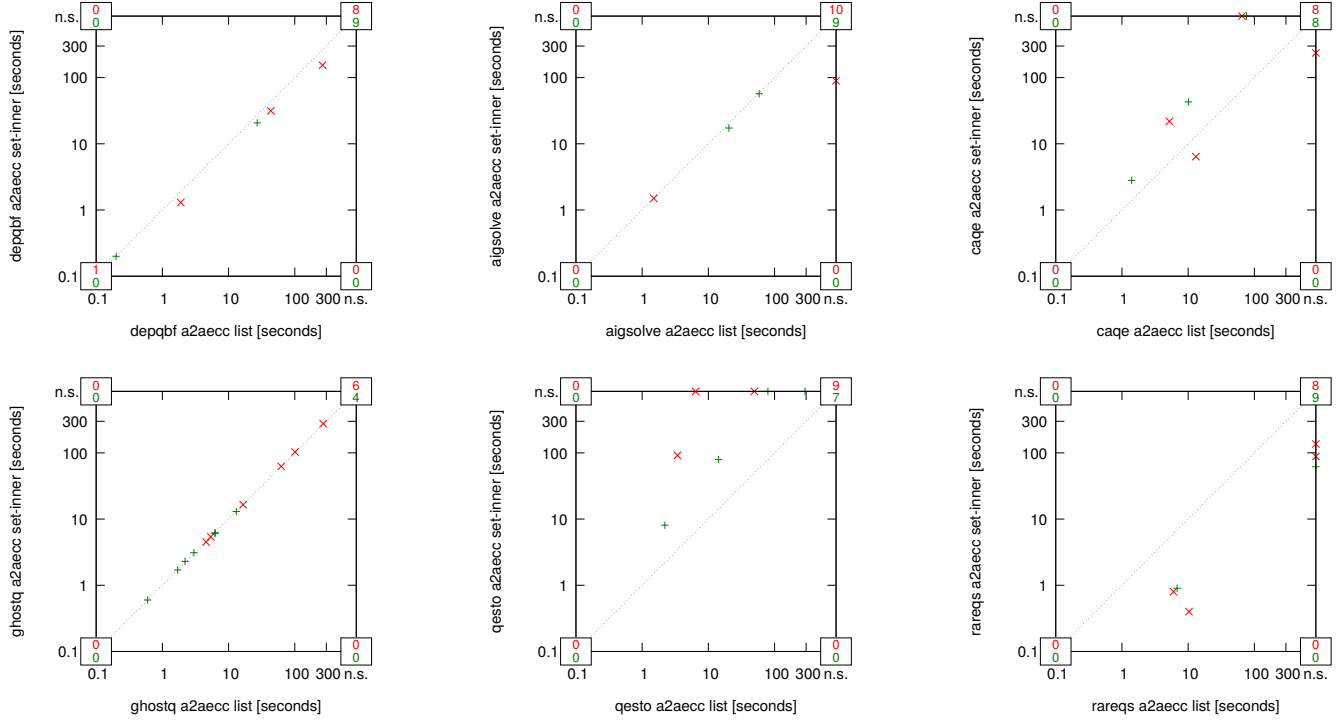


Fig. 1451: Suite Ansotegui ($n = 38$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

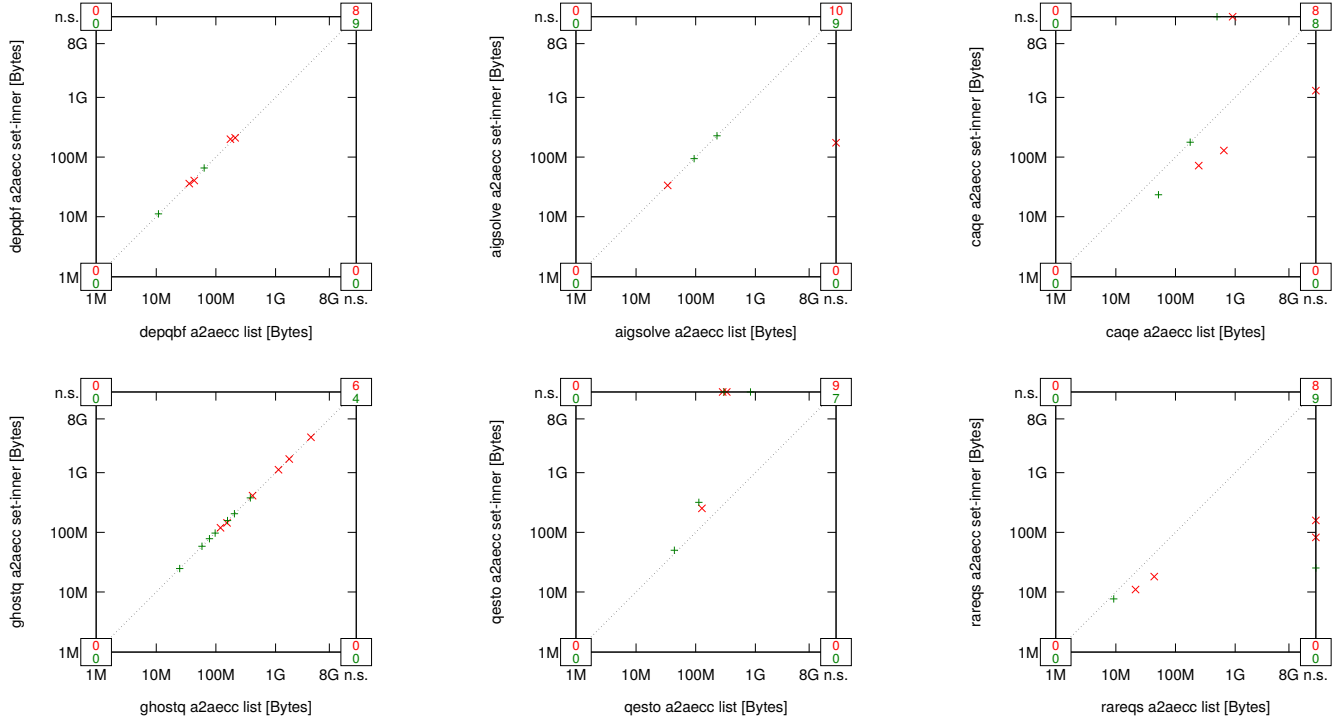


Fig. 1452: Suite Ansotegui ($n = 38$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

5) *Ayari* ($n = 71$):

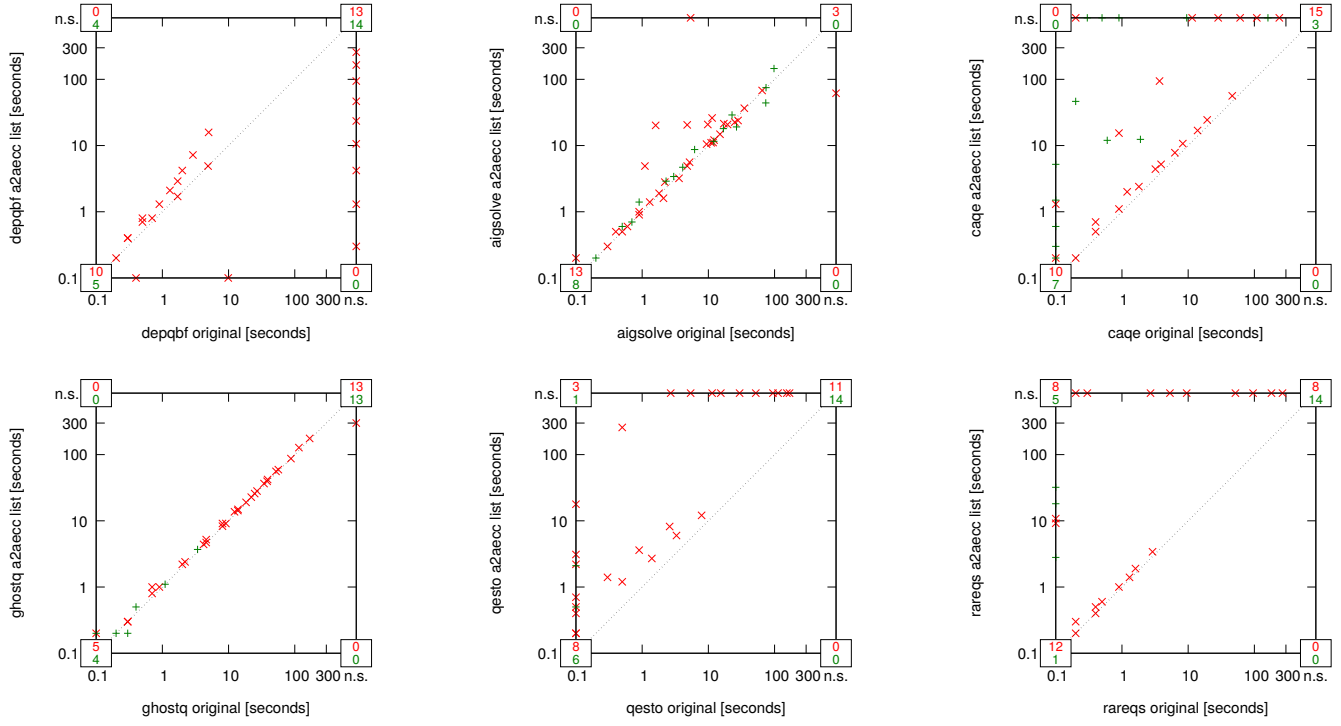


Fig. 1453: Suite Ayari ($n = 71$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

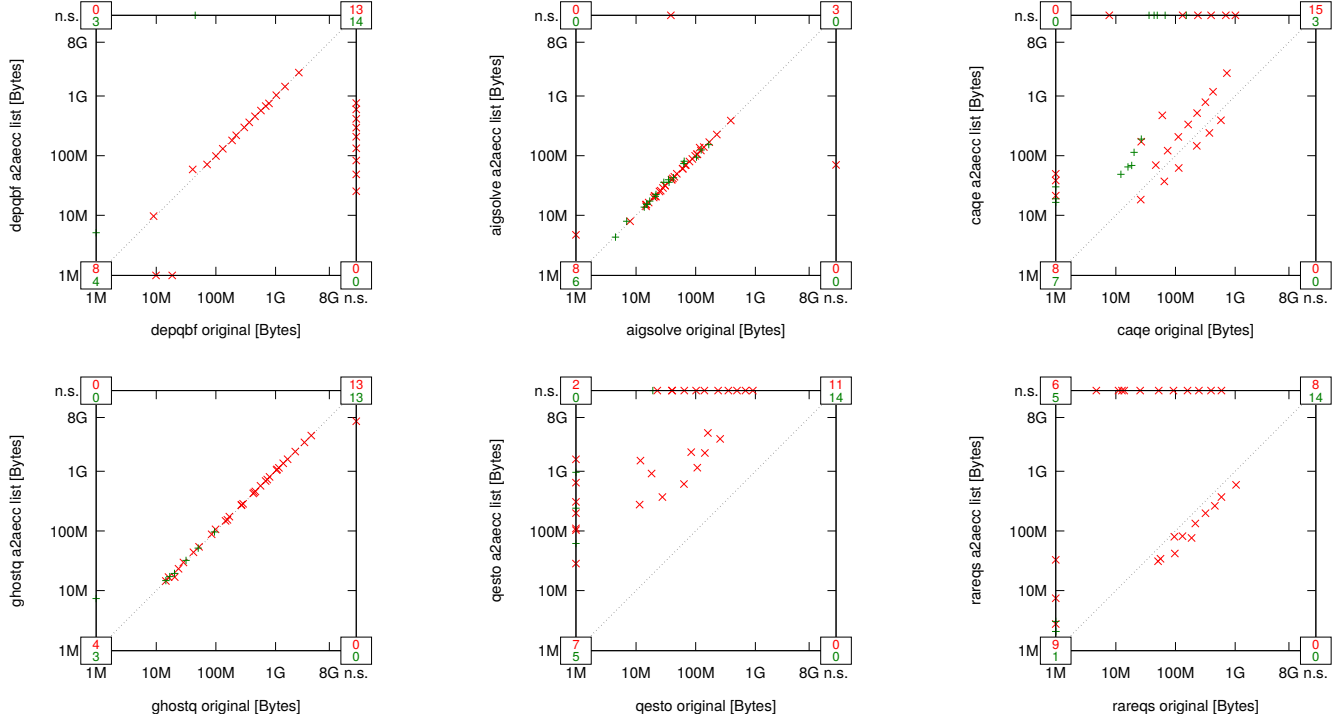


Fig. 1454: Suite Ayari ($n = 71$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

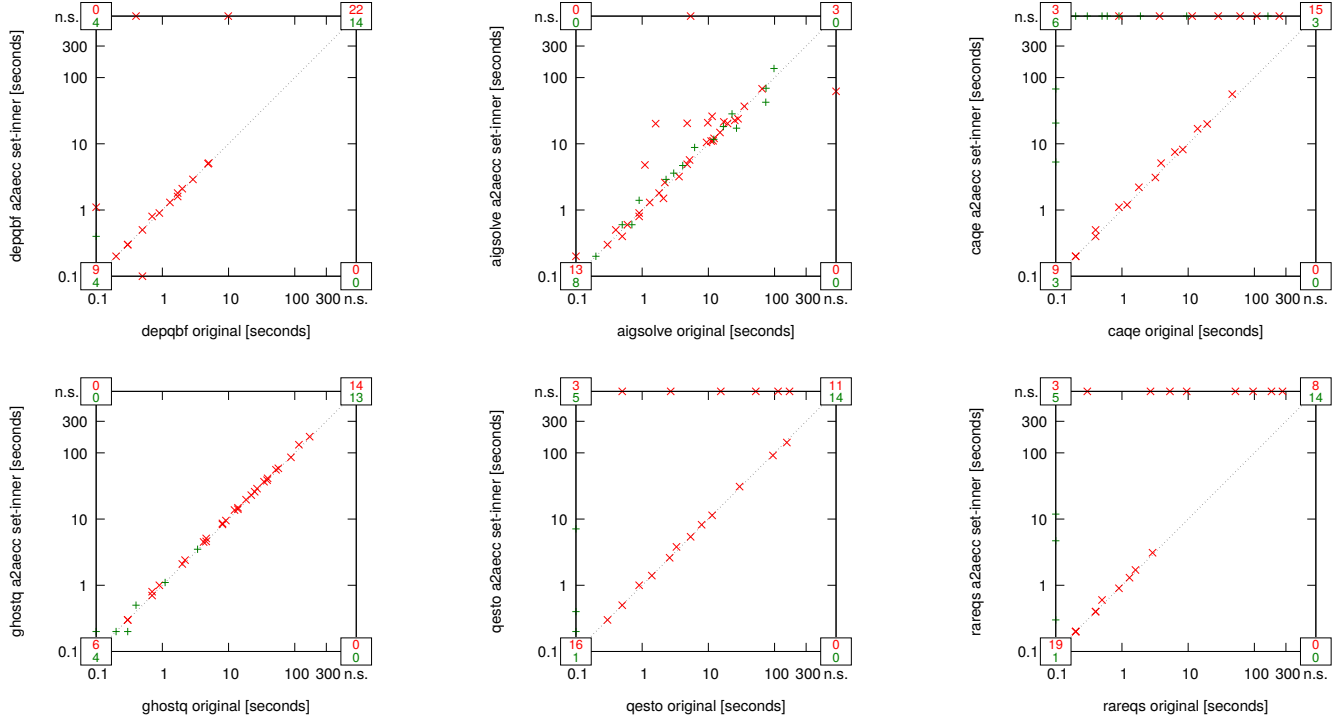


Fig. 1455: Suite Ayari ($n = 71$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

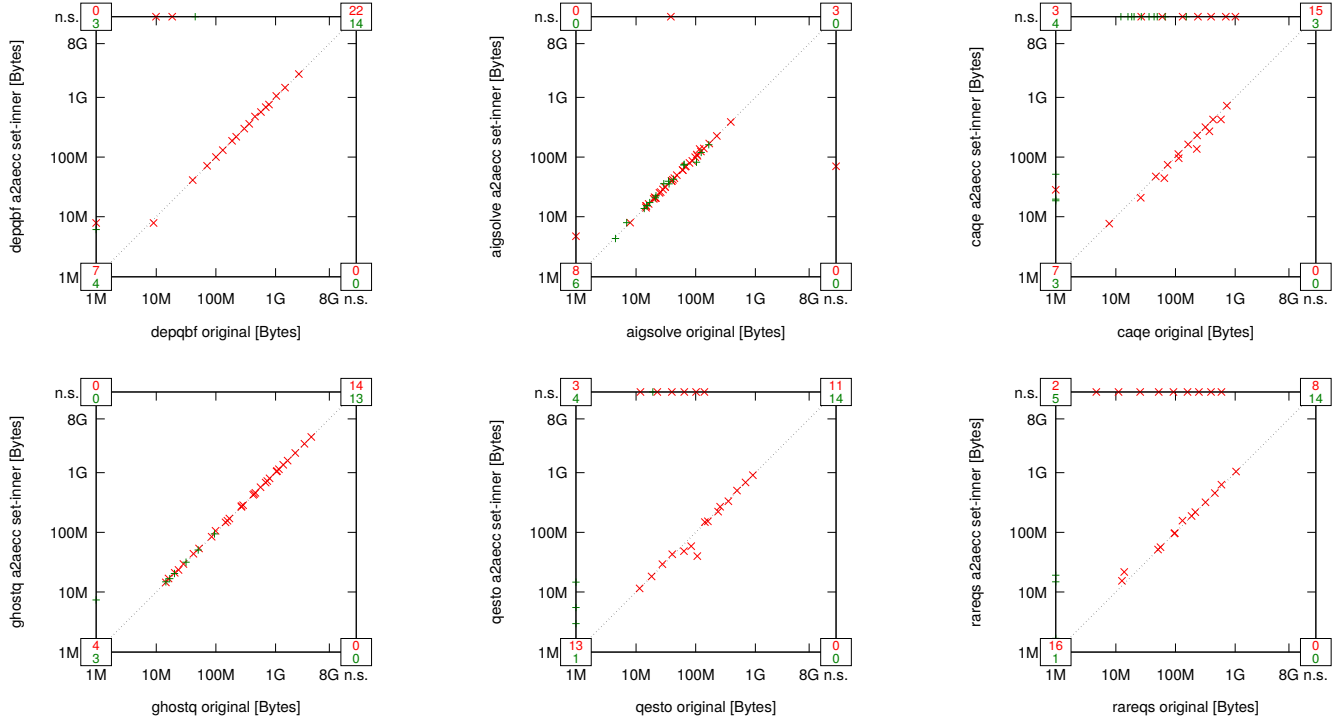


Fig. 1456: Suite Ayari ($n = 71$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

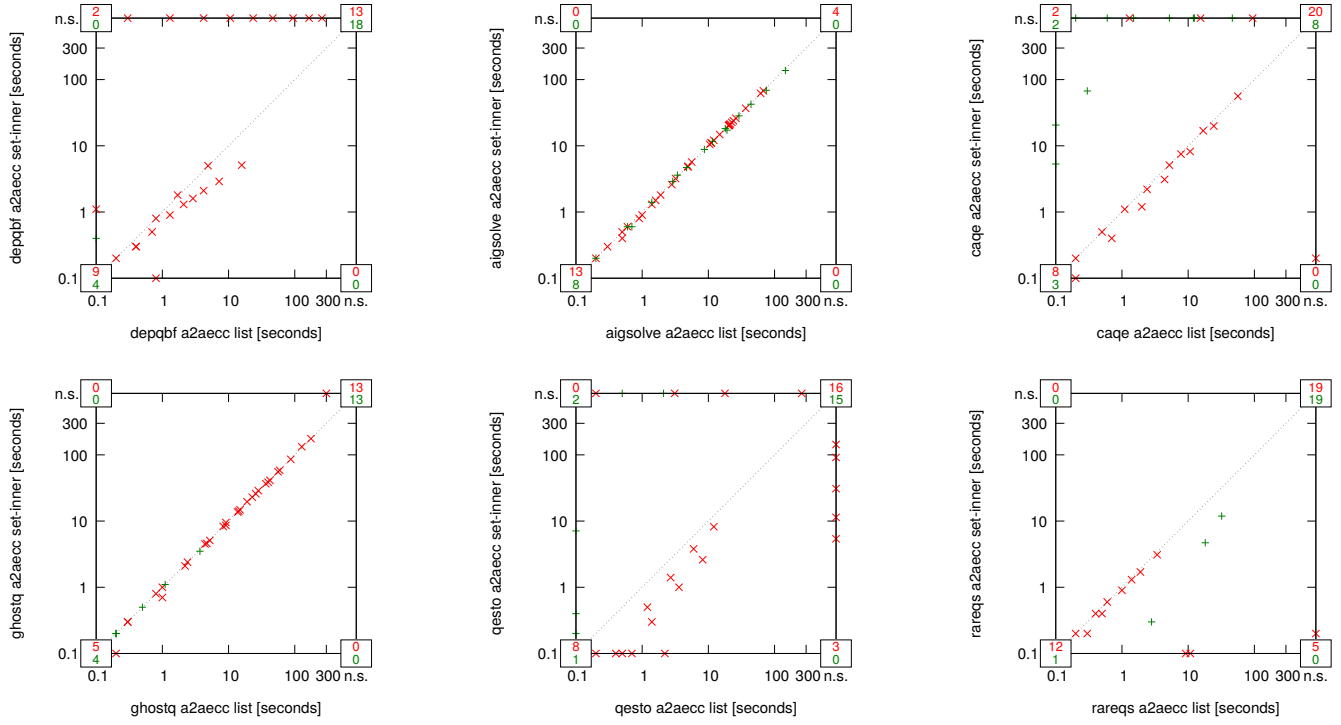


Fig. 1457: Suite Ayari ($n = 71$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

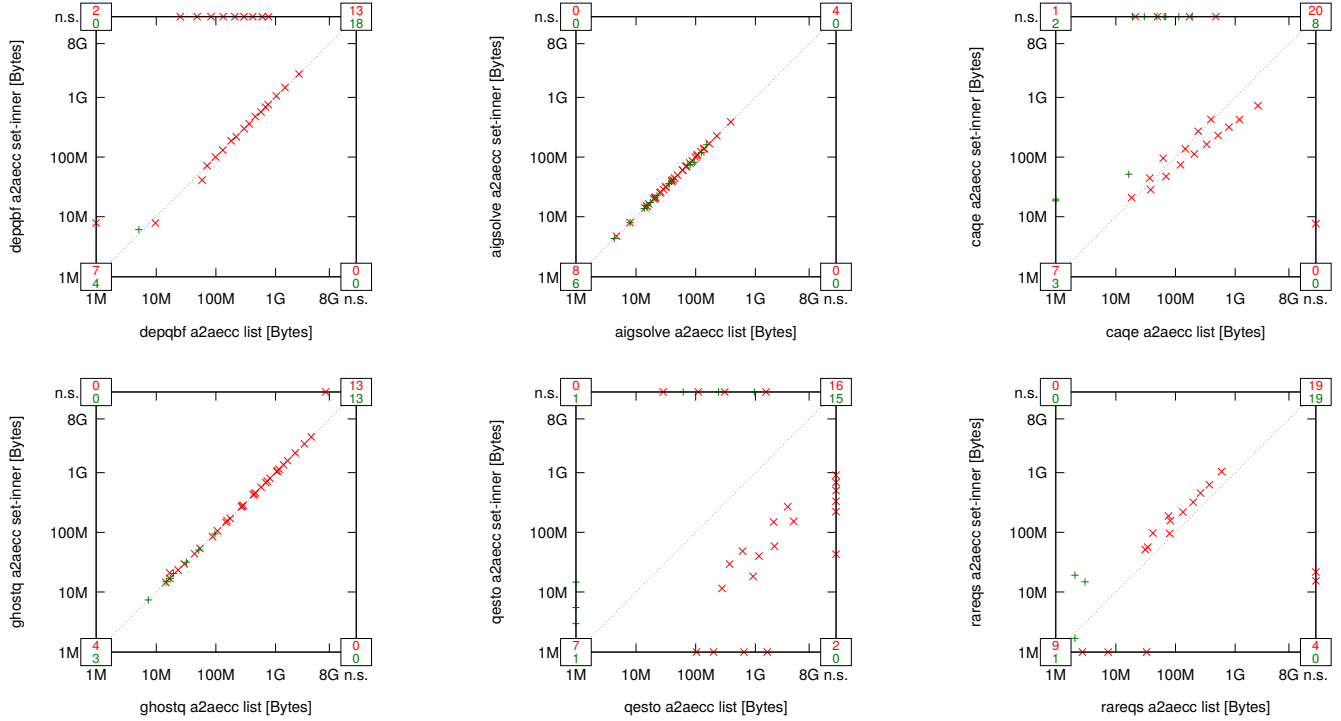


Fig. 1458: Suite Ayari ($n = 71$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

6) Basler ($n = 193$):

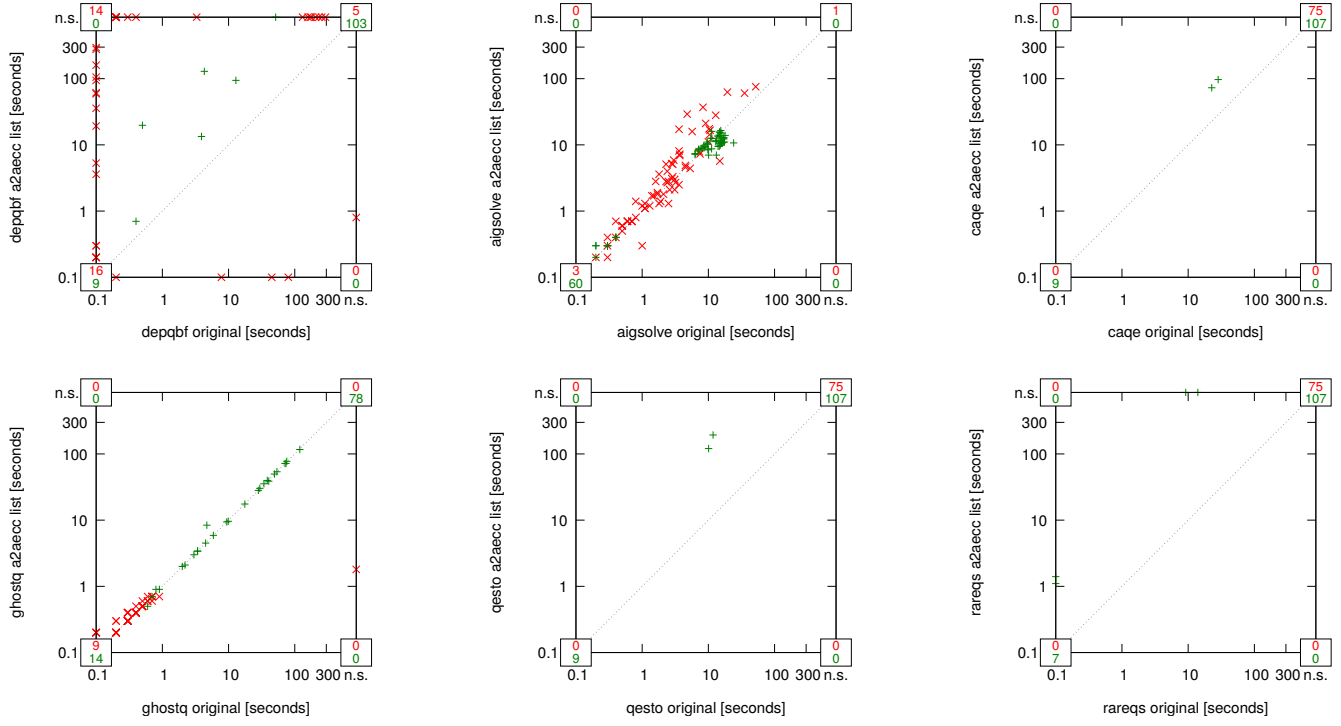


Fig. 1459: Suite Basler ($n = 193$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

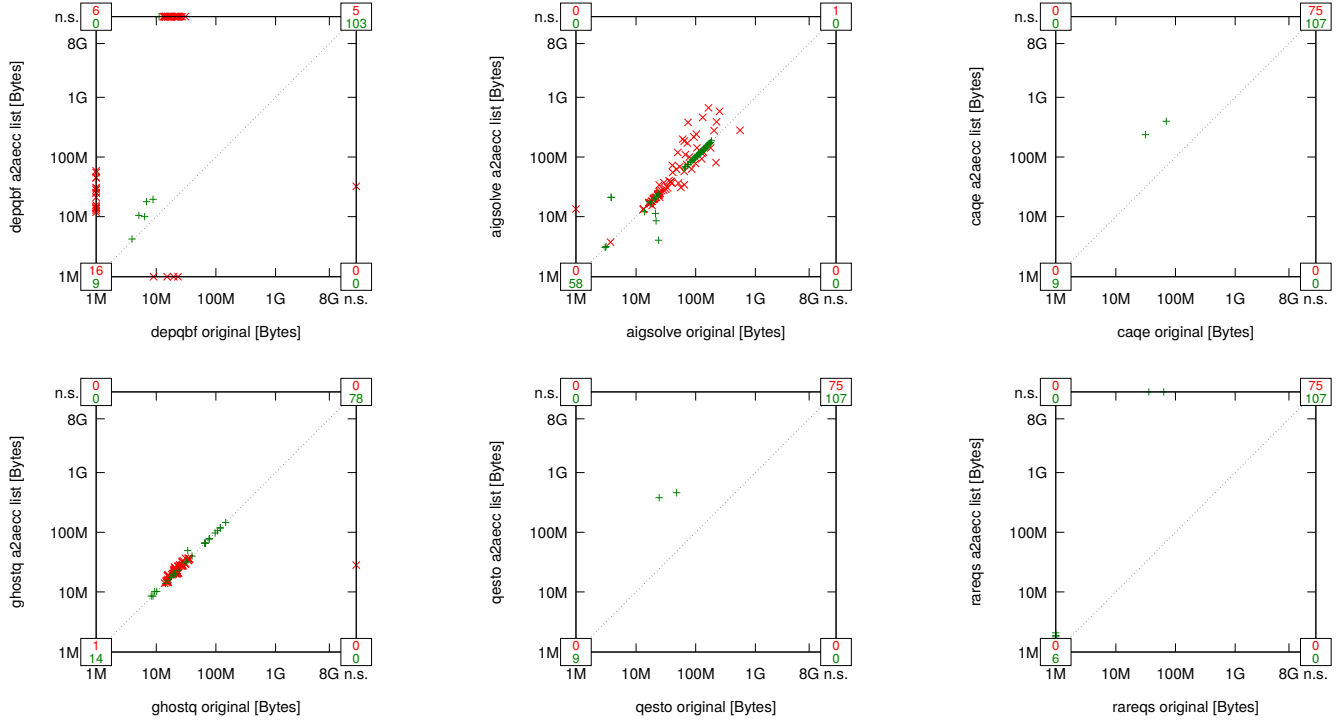


Fig. 1460: Suite Basler ($n = 193$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

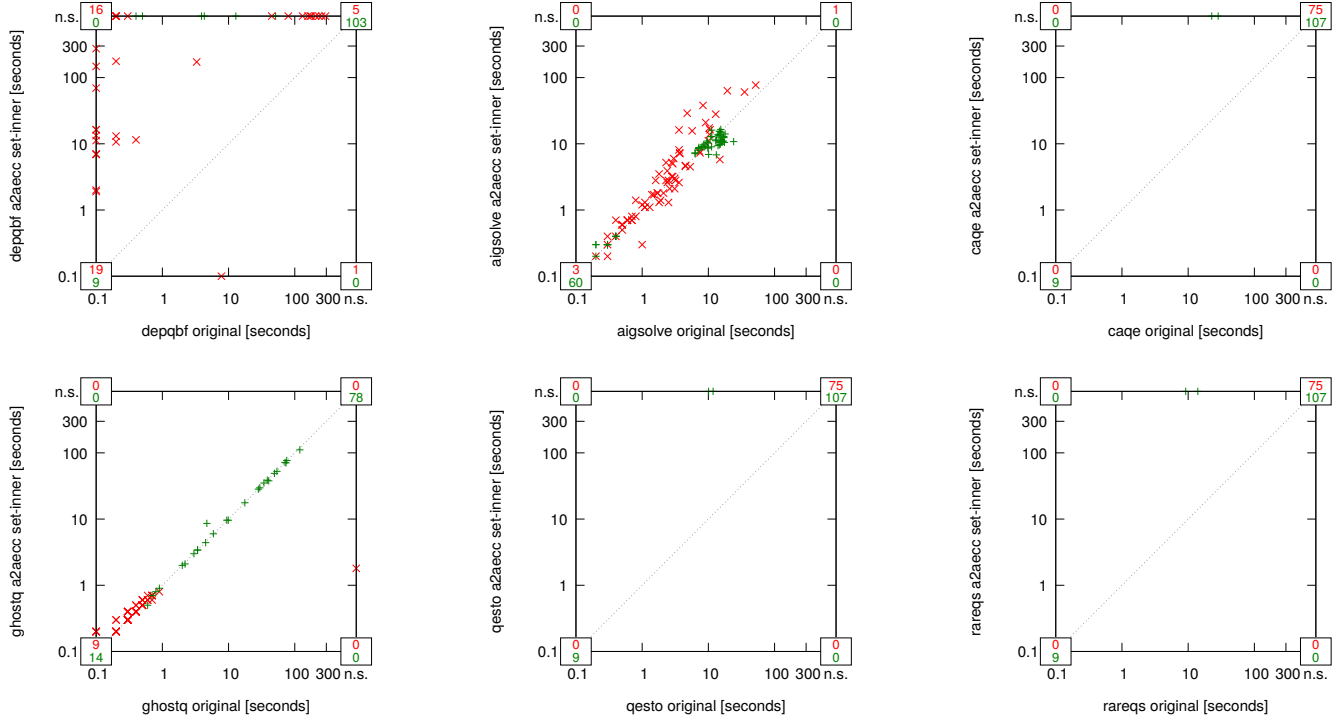


Fig. 1461: Suite Basler ($n = 193$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

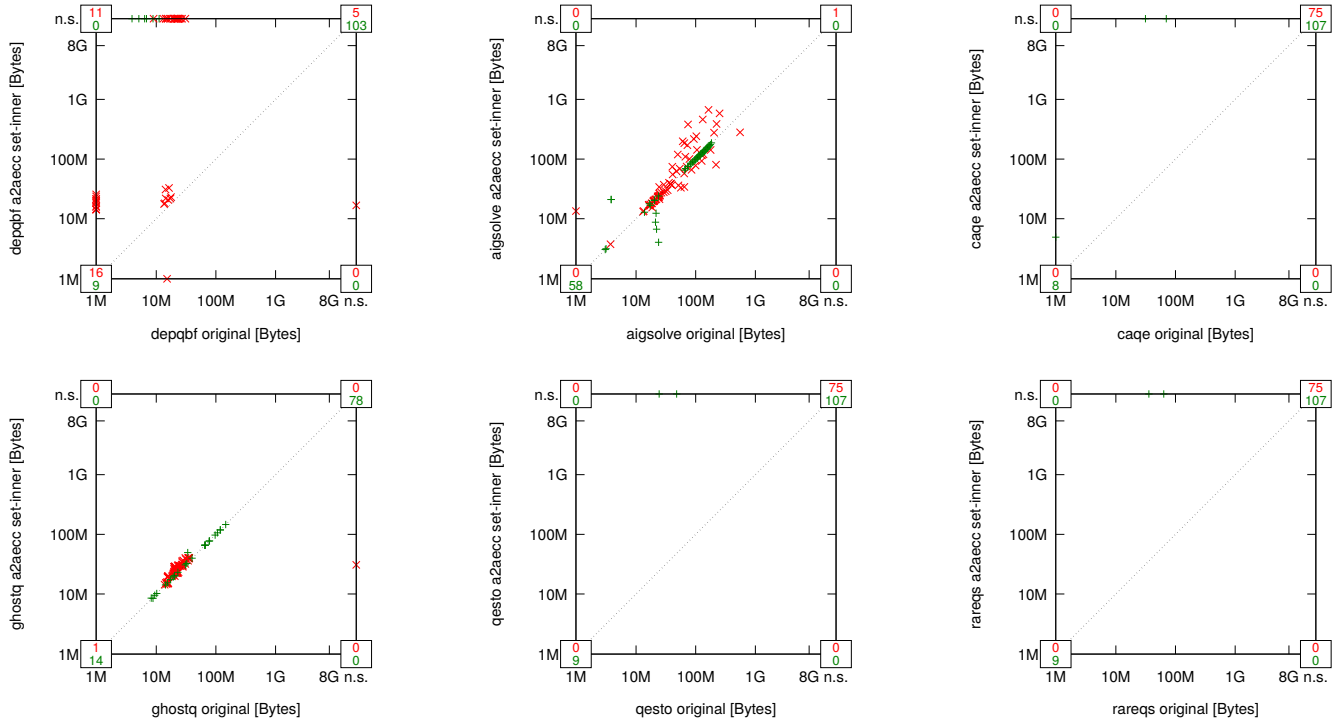


Fig. 1462: Suite Basler ($n = 193$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

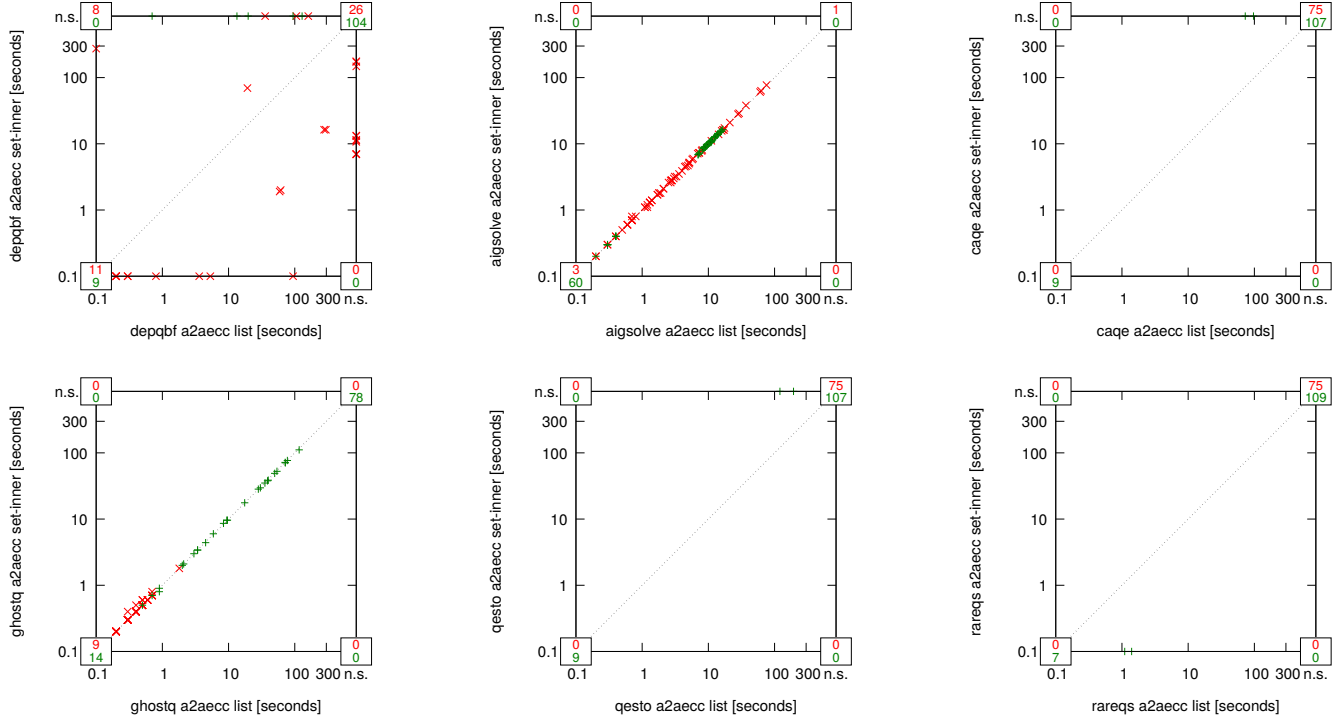


Fig. 1463: Suite Basler ($n = 193$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

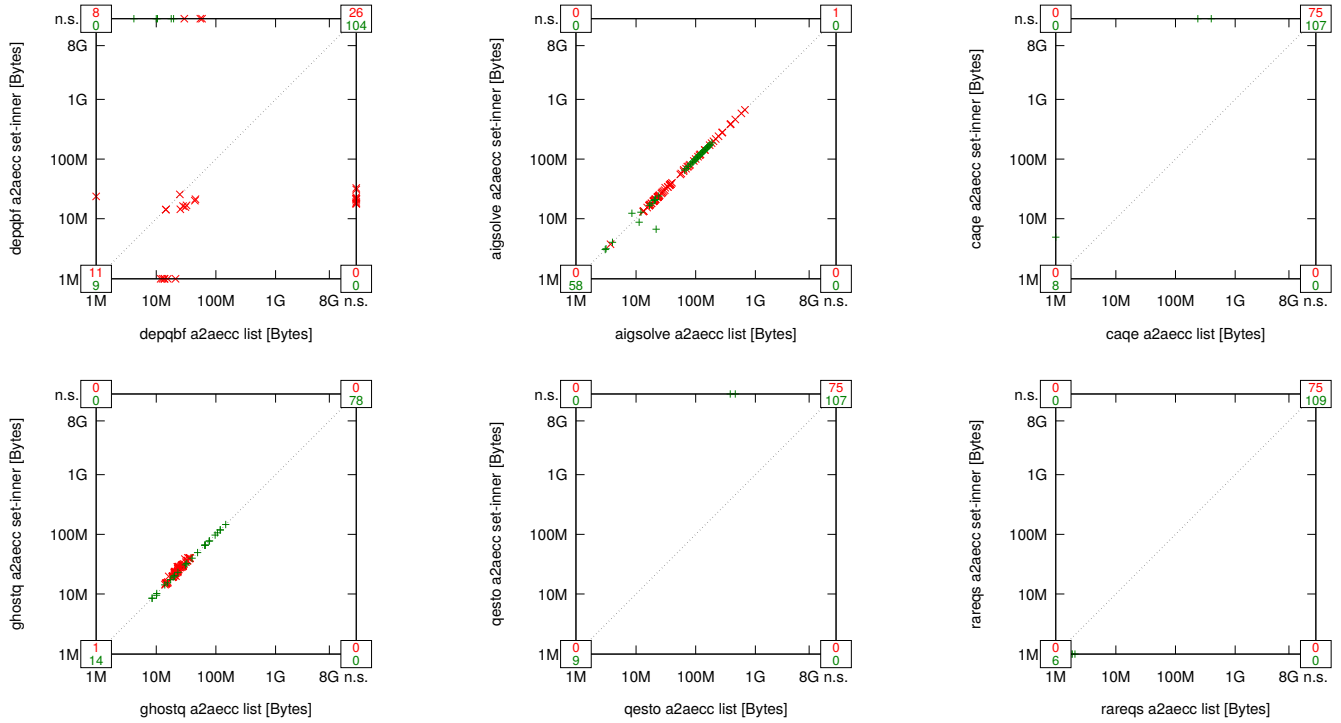


Fig. 1464: Suite Basler ($n = 193$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

7) *Biere* ($n = 194$):

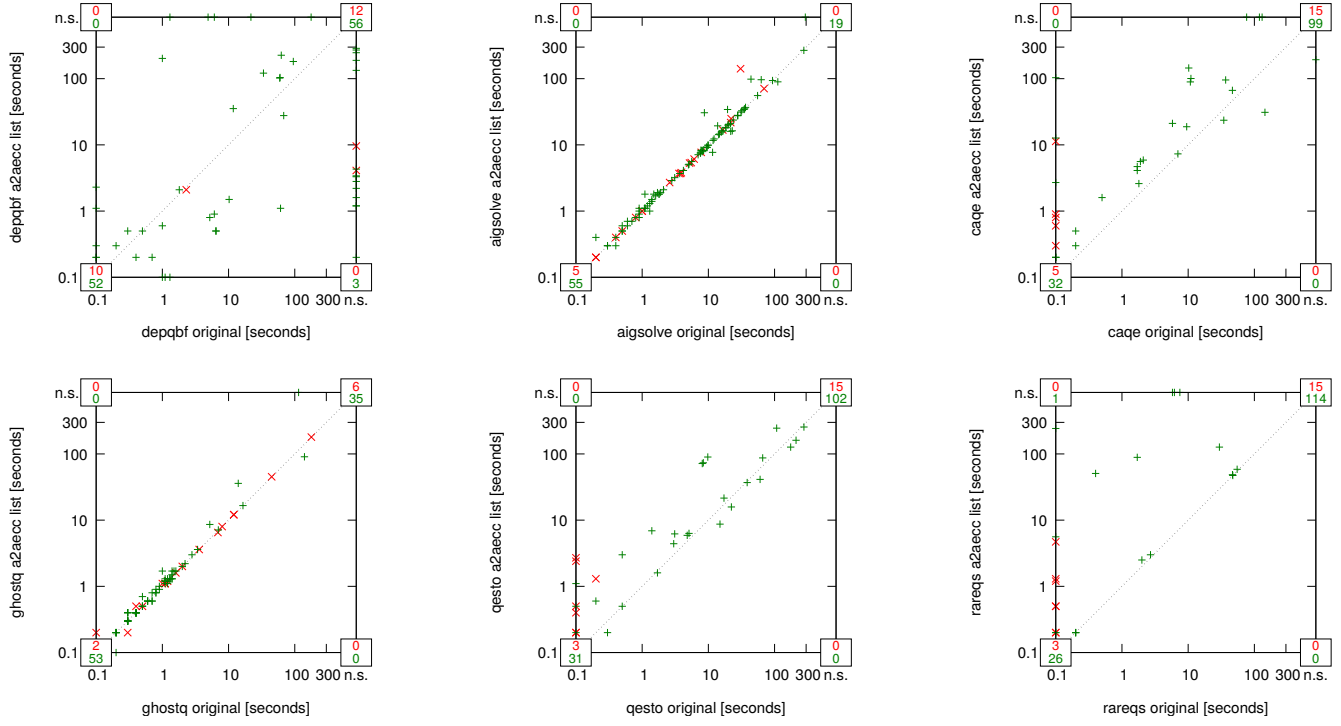


Fig. 1465: Suite Biere ($n = 194$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

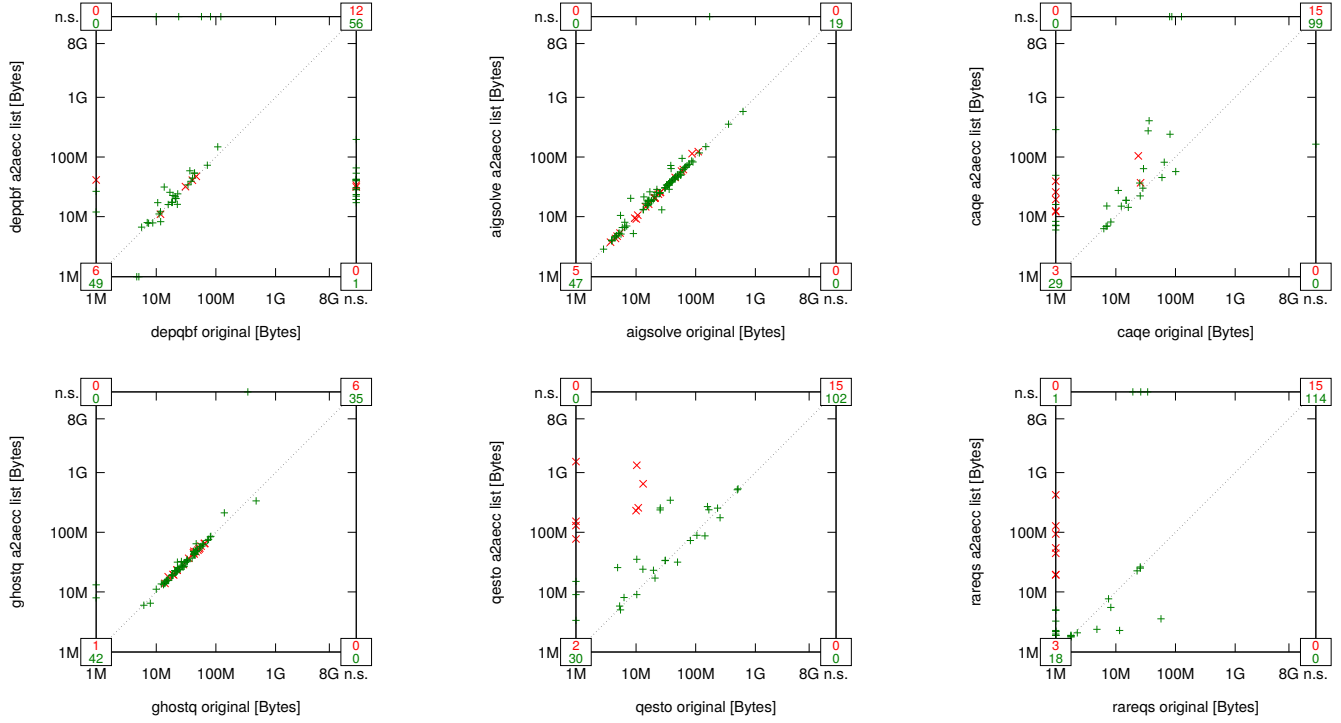


Fig. 1466: Suite Biere ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

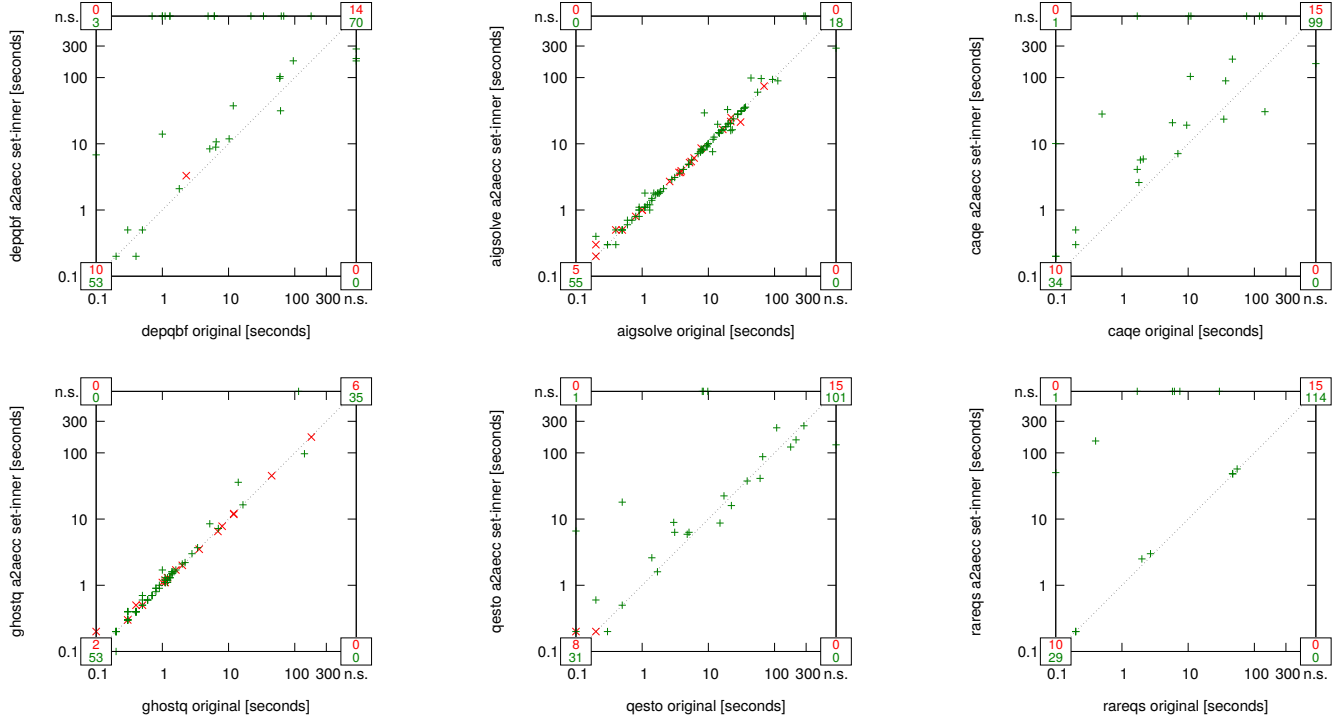


Fig. 1467: Suite Biere ($n = 194$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

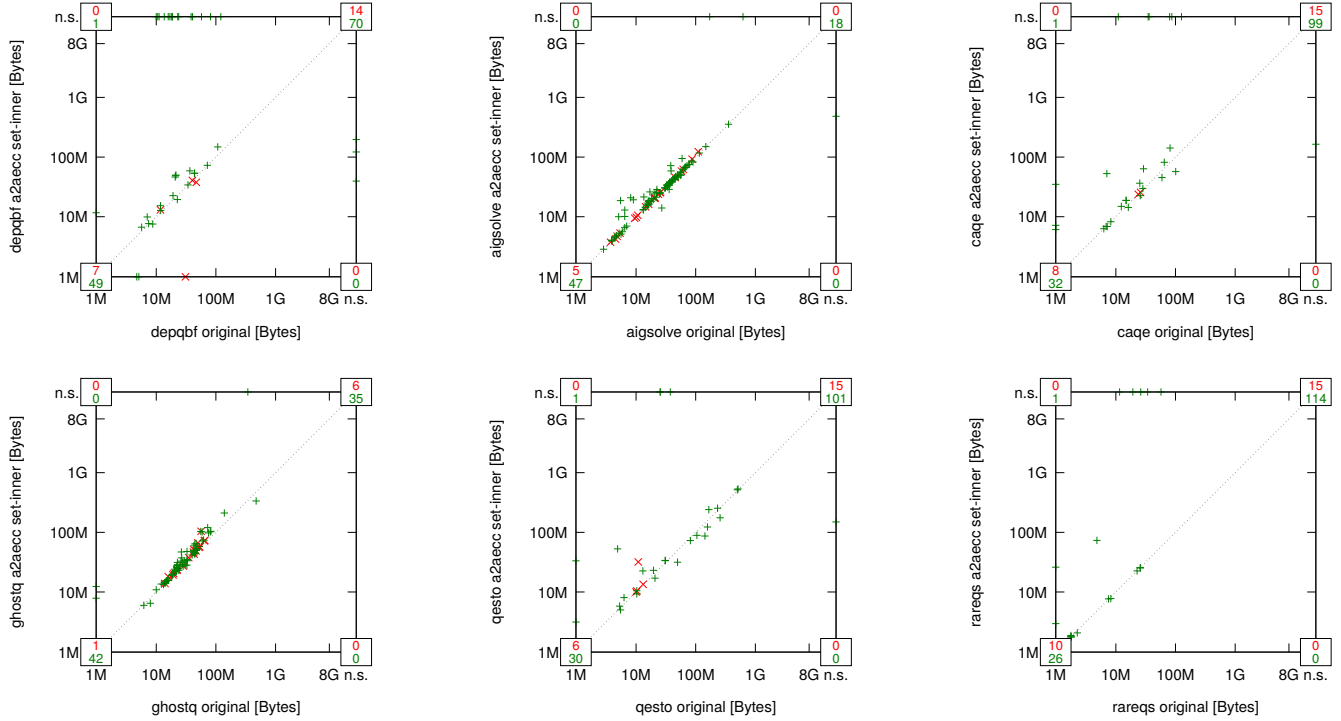


Fig. 1468: Suite Biere ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

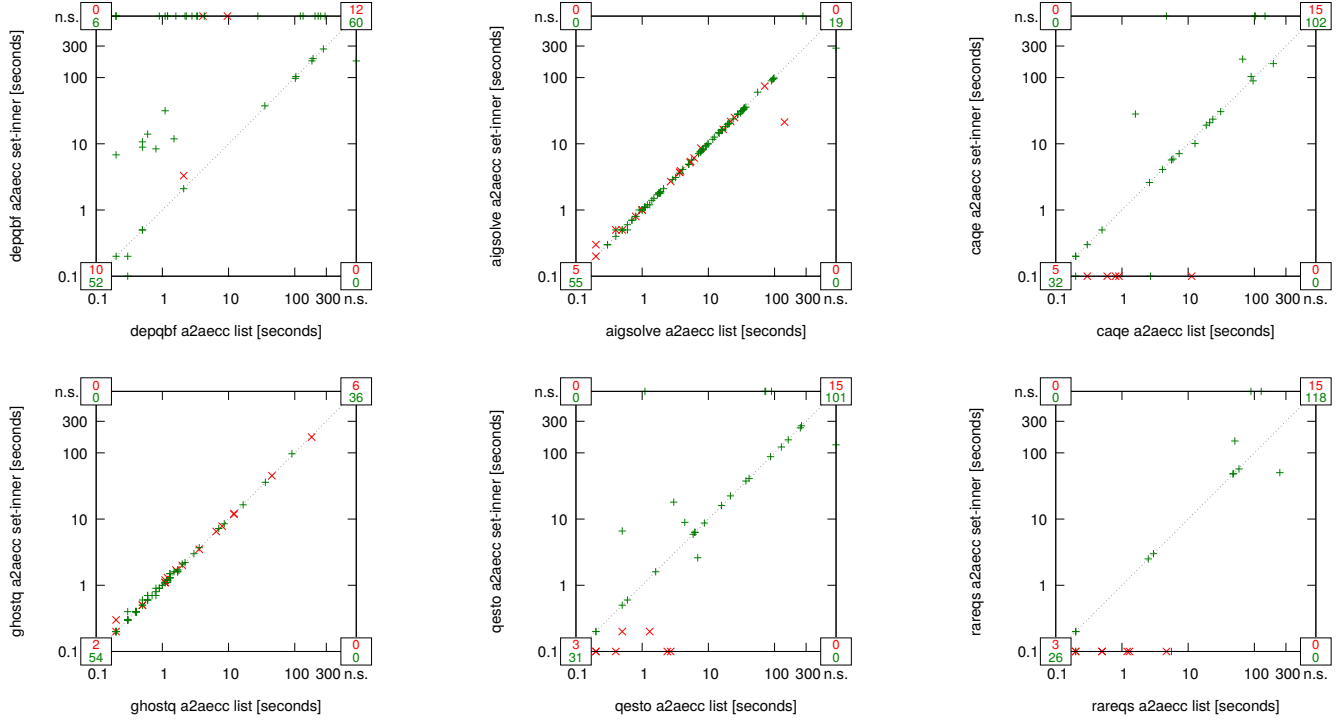


Fig. 1469: Suite Biere ($n = 194$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

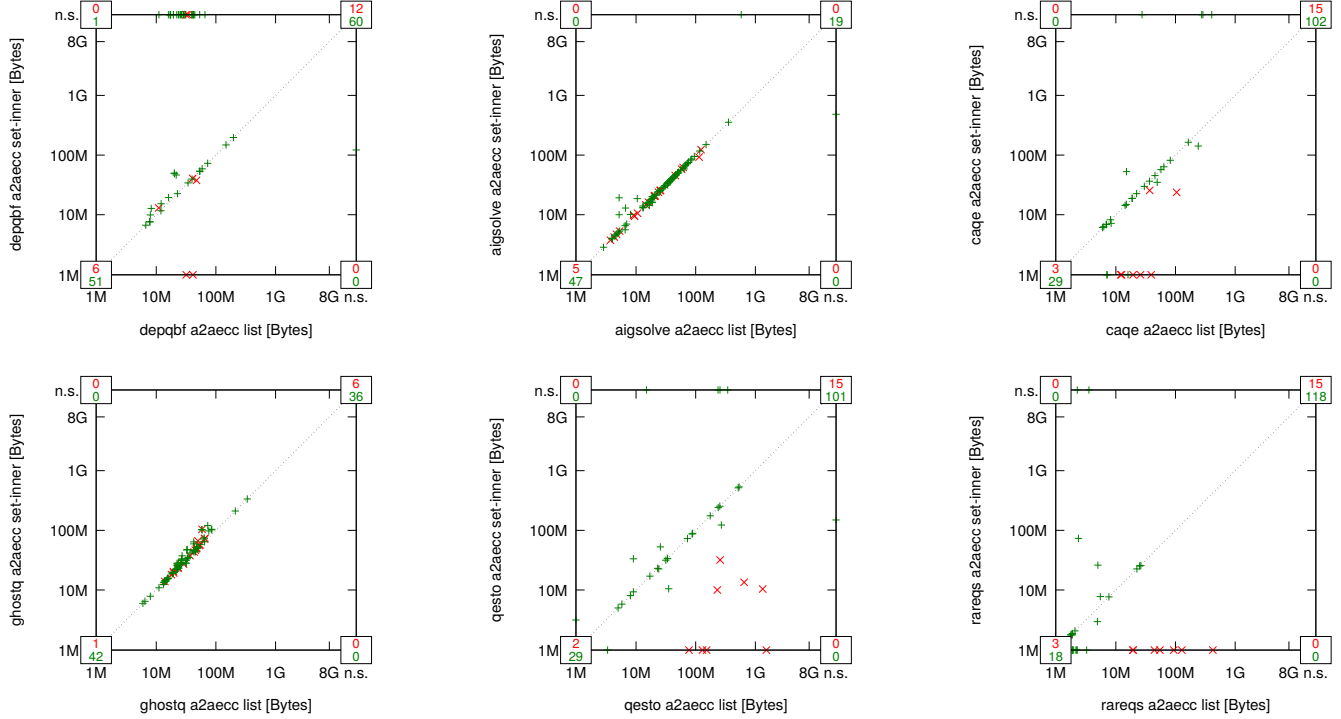


Fig. 1470: Suite Biere ($n = 194$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

8) Cashmore-Fox-Giunchiglia ($n = 150$):

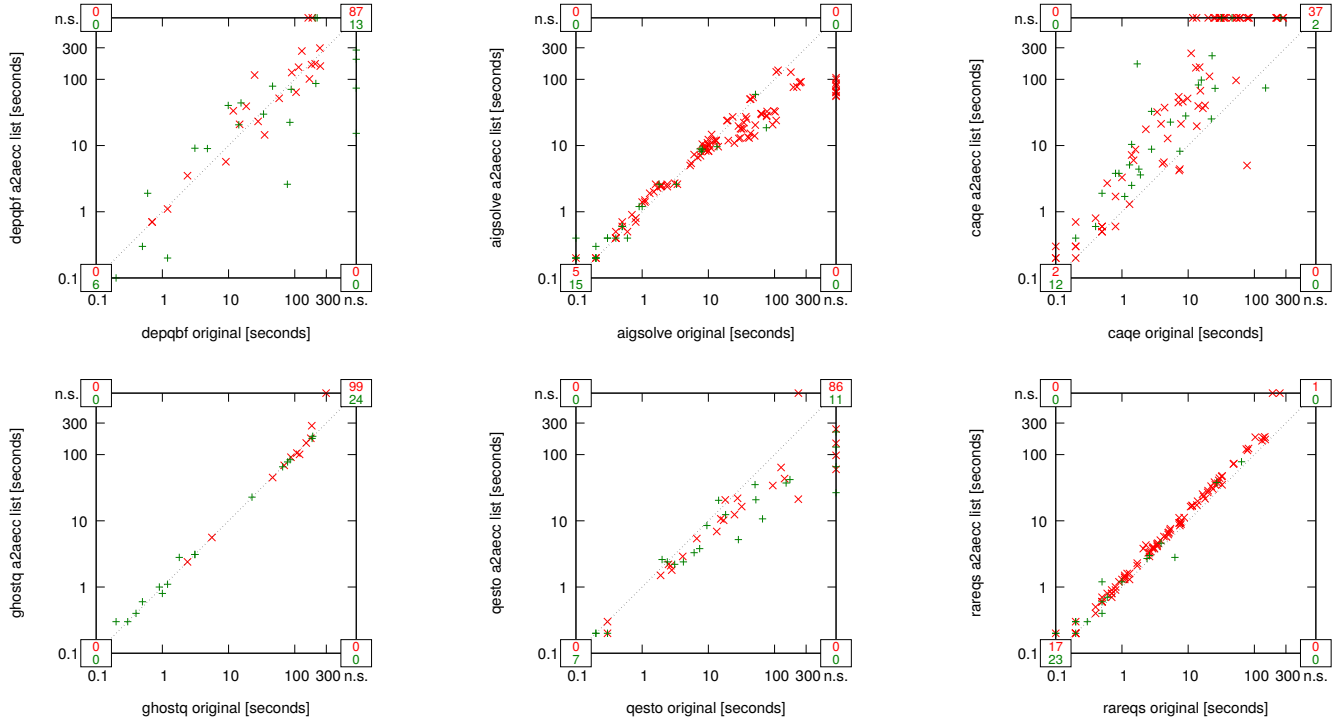


Fig. 1471: Suite Cashmore-Fox-Giunchiglia ($n = 150$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

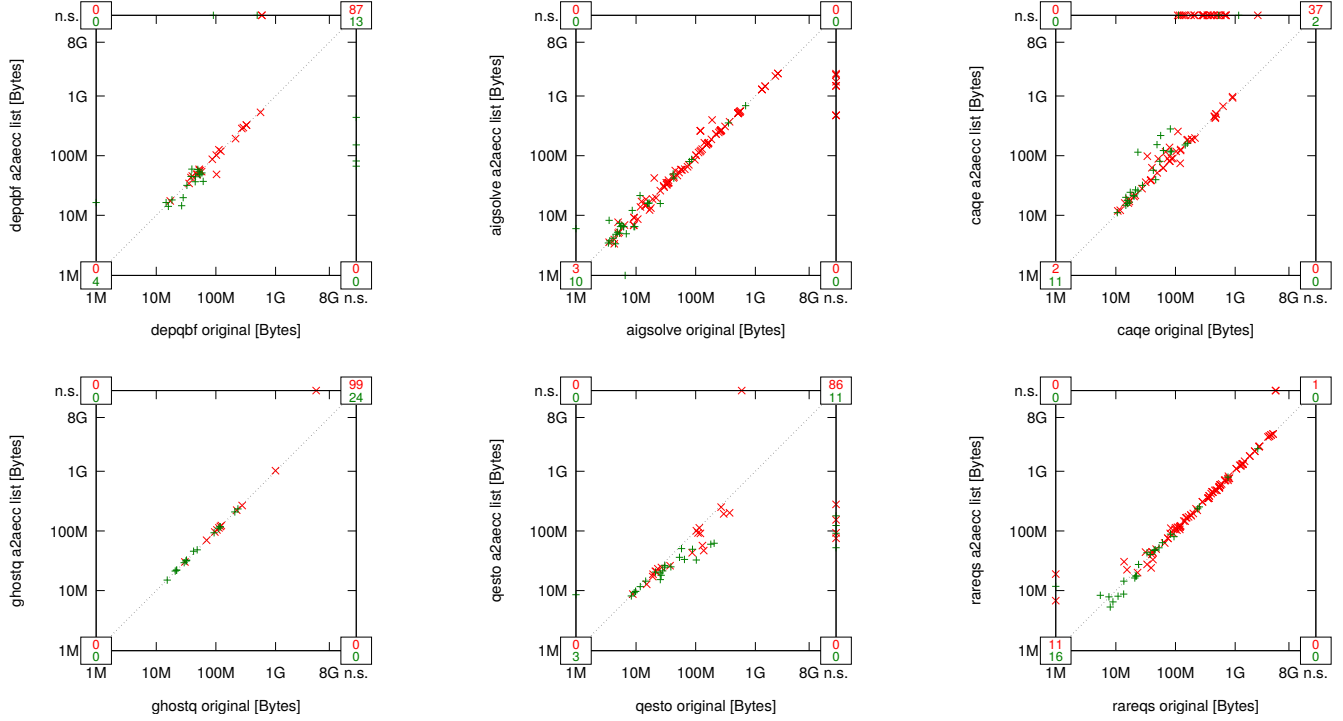


Fig. 1472: Suite Cashmore-Fox-Giunchiglia ($n = 150$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

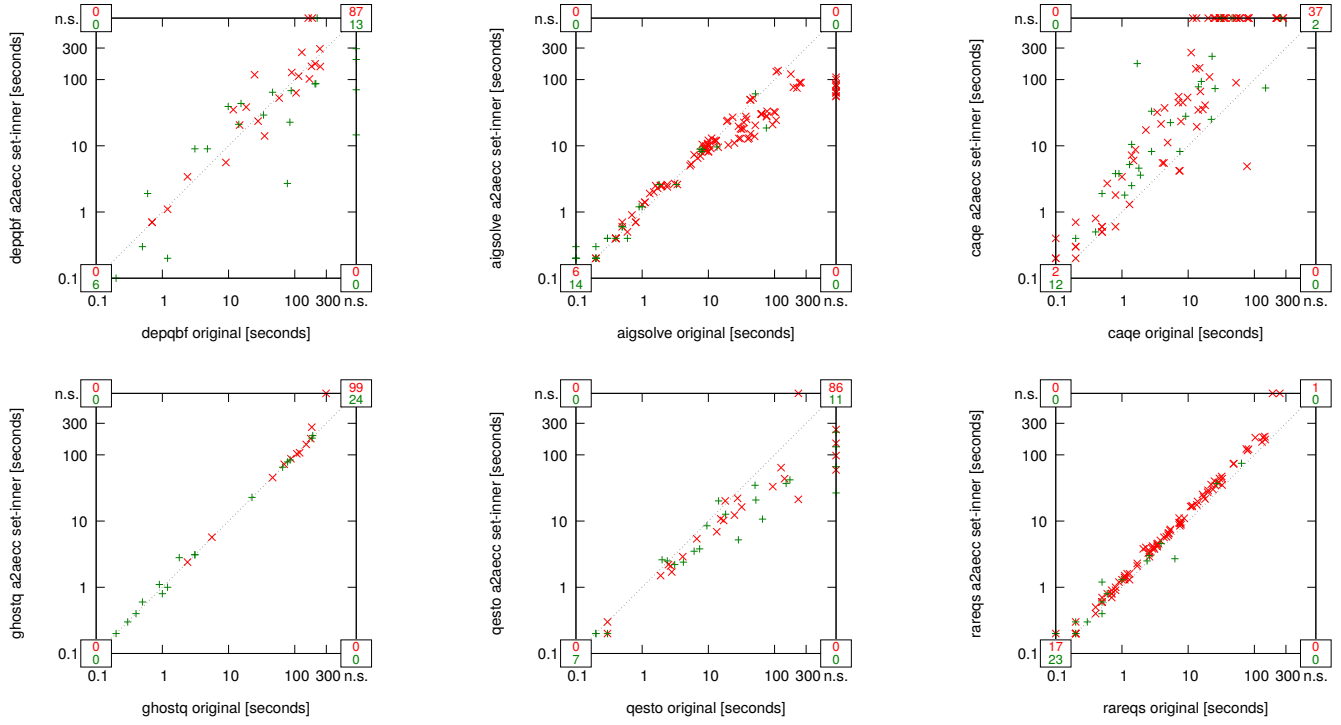


Fig. 1473: Suite Cashmore-Fox-Giunchiglia ($n = 150$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

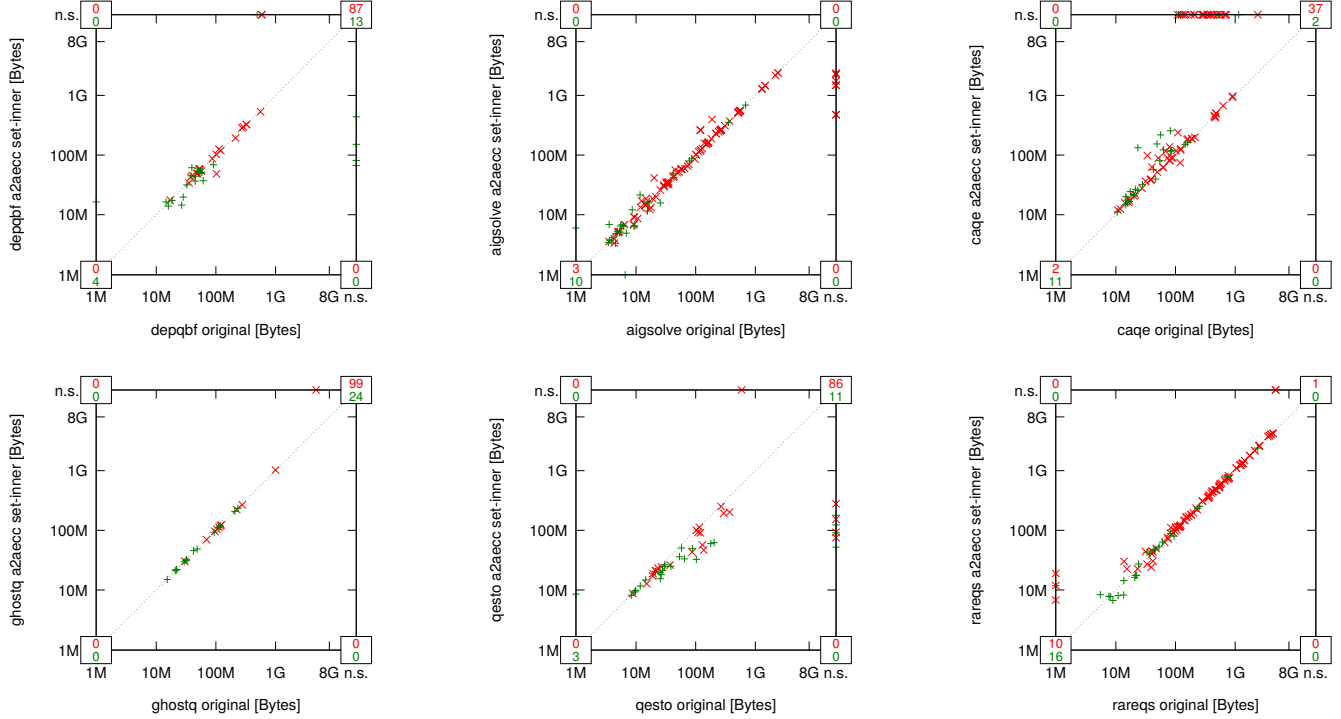


Fig. 1474: Suite Cashmore-Fox-Giunchiglia ($n = 150$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

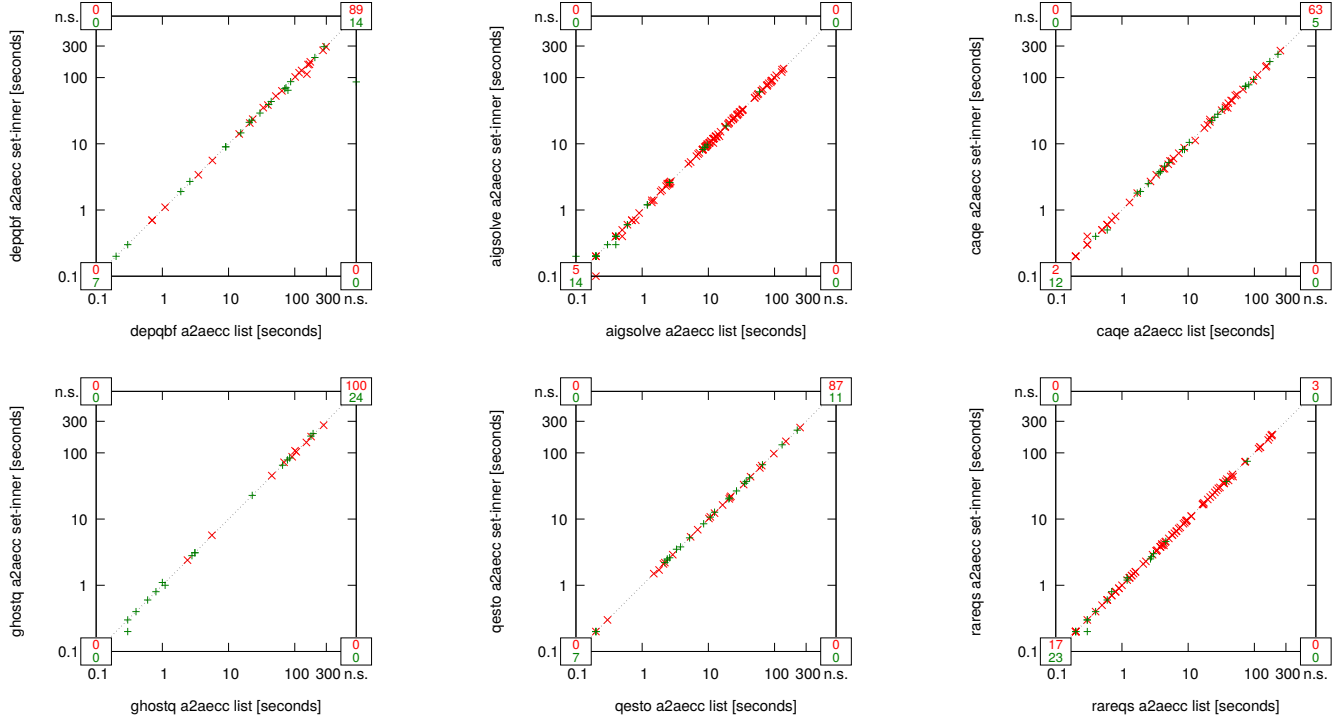


Fig. 1475: Suite Cashmore-Fox-Giunchiglia ($n = 150$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

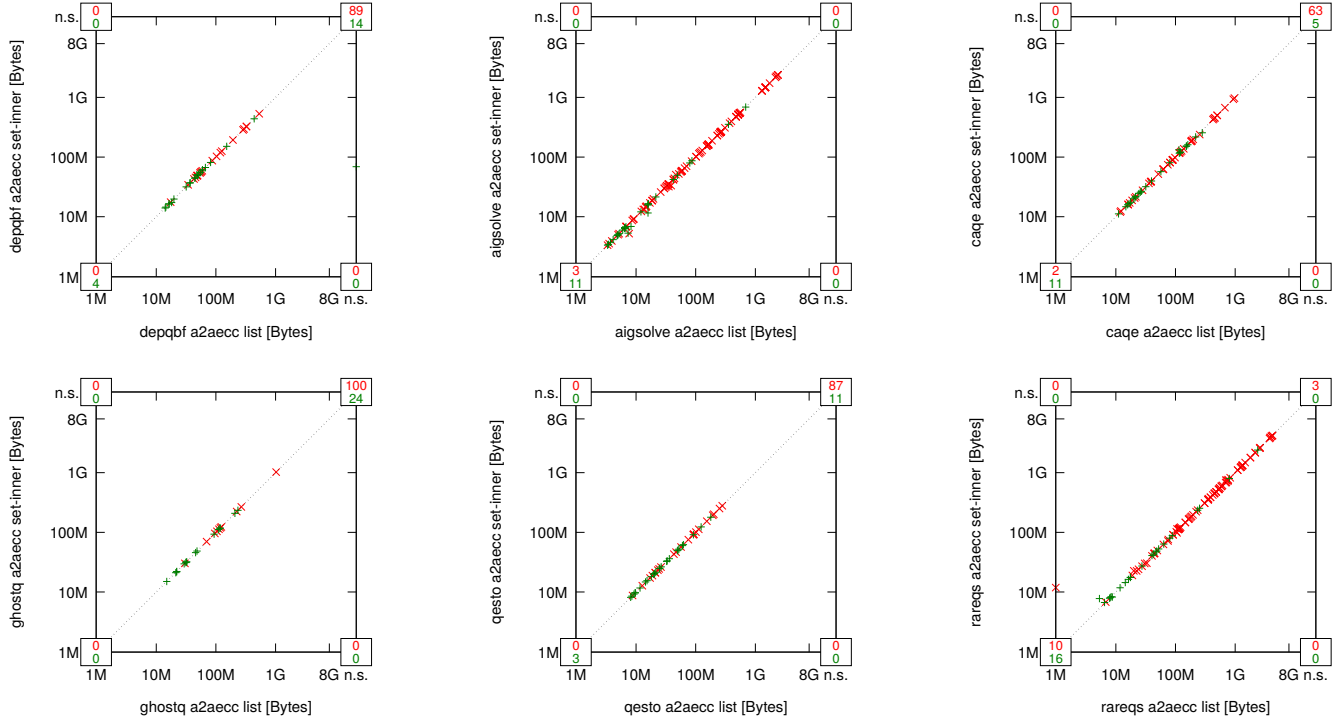


Fig. 1476: Suite Cashmore-Fox-Giunchiglia ($n = 150$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

9) Castellini ($n = 169$):

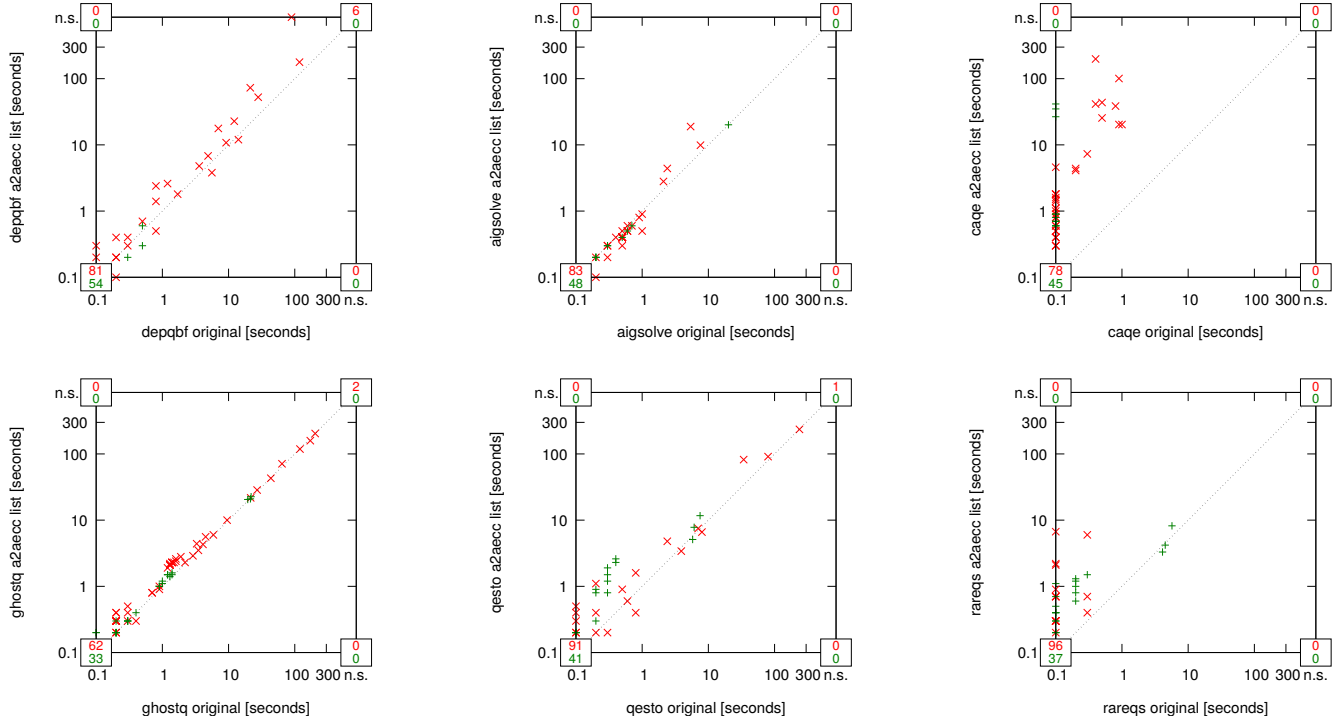


Fig. 1477: Suite Castellini ($n = 169$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

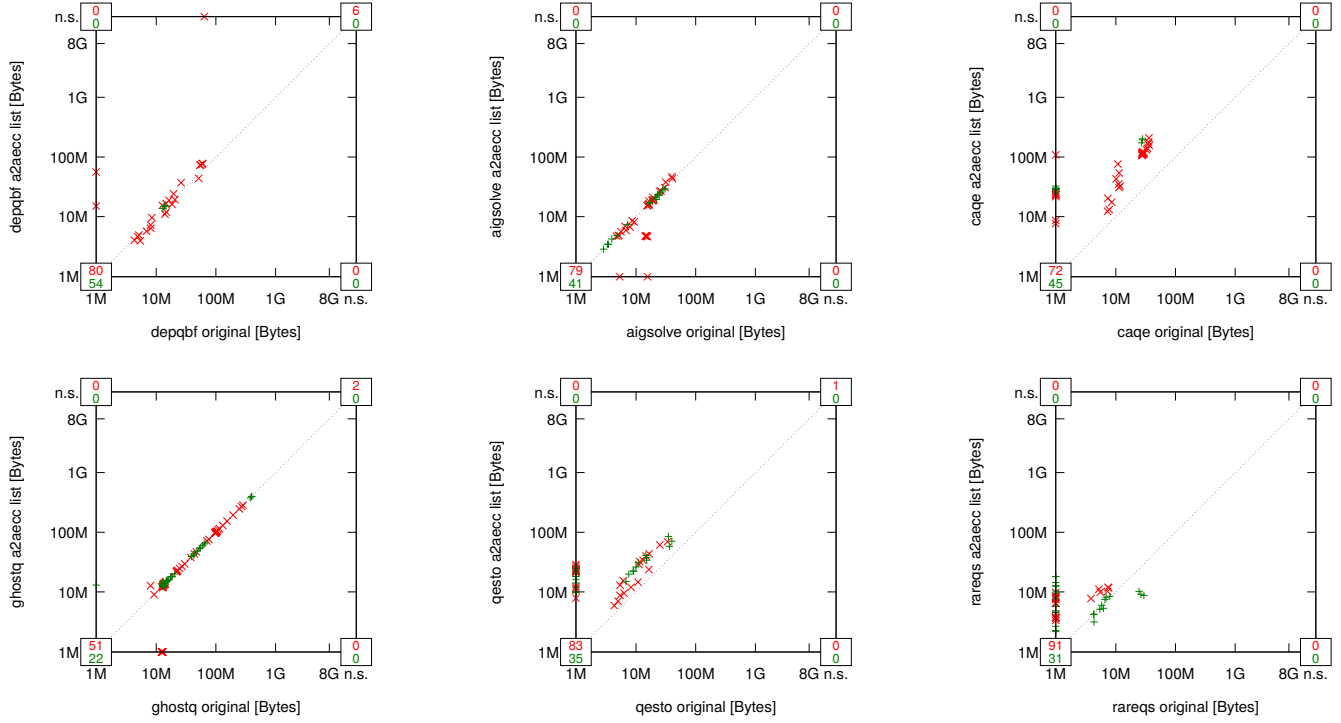


Fig. 1478: Suite Castellini ($n = 169$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

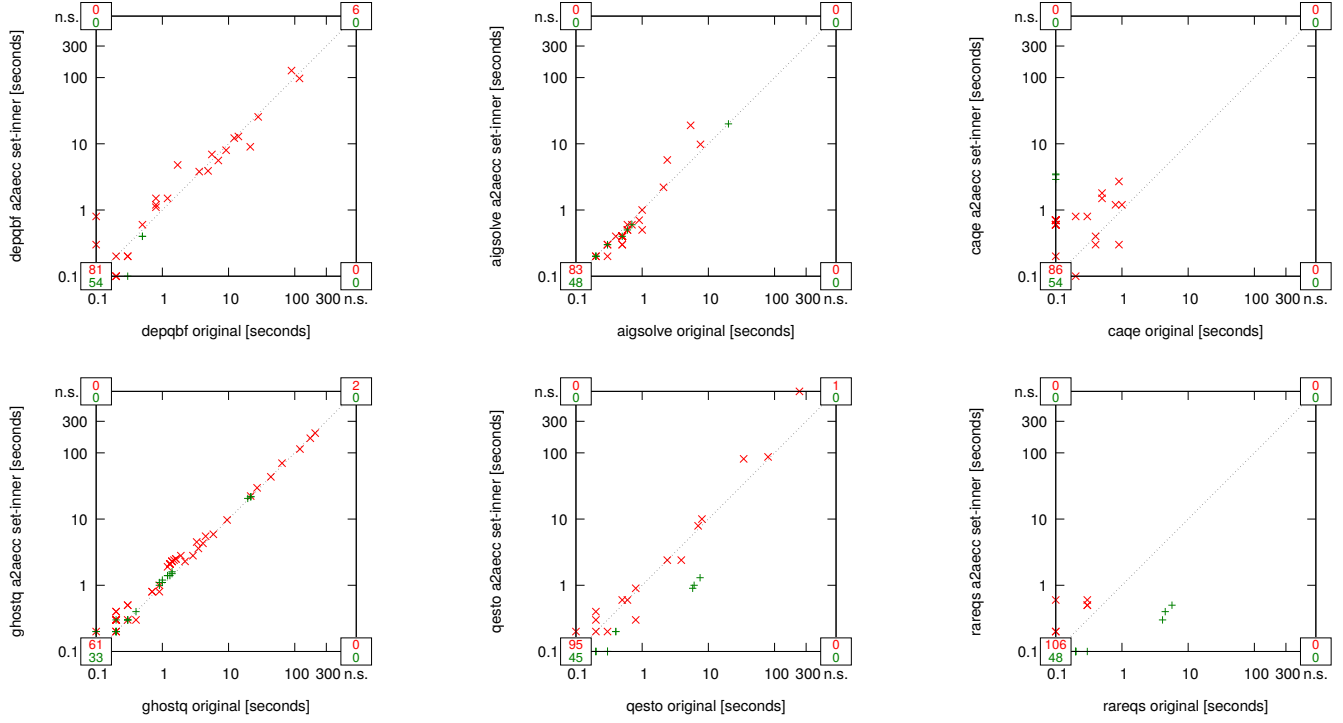


Fig. 1479: Suite Castellini ($n = 169$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

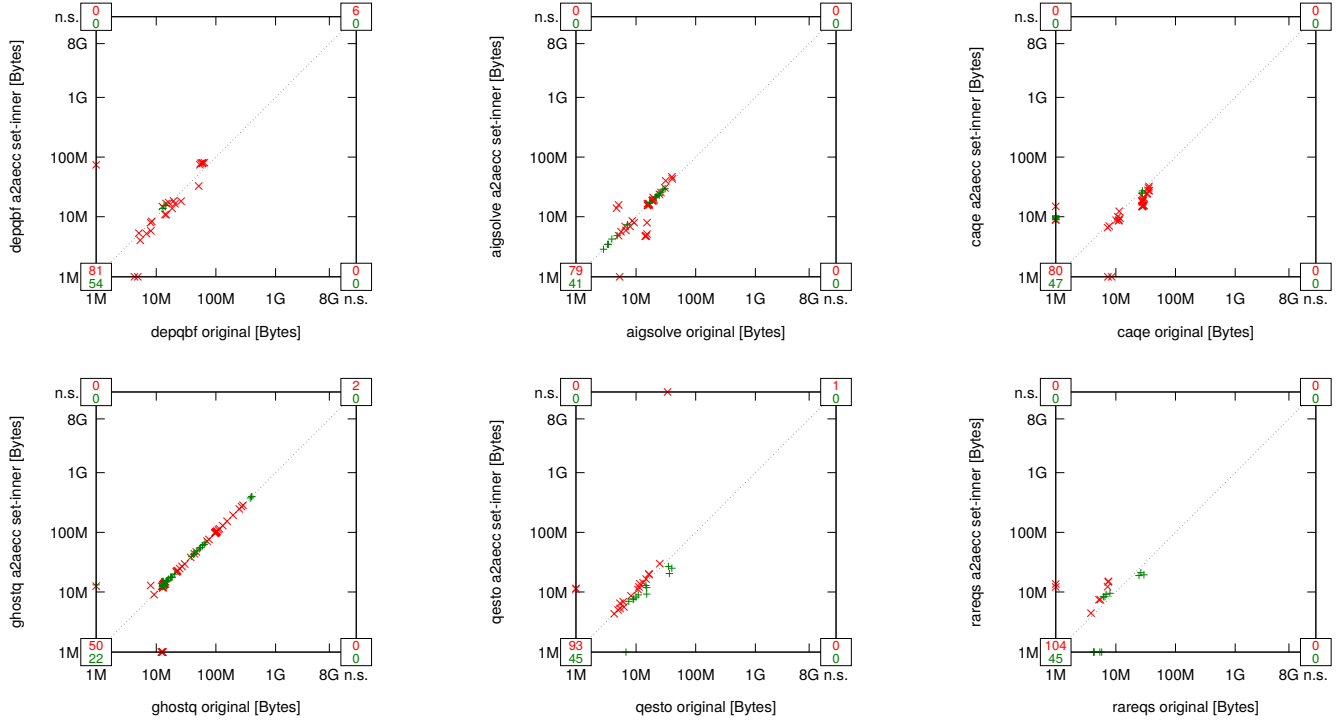


Fig. 1480: Suite Castellini ($n = 169$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

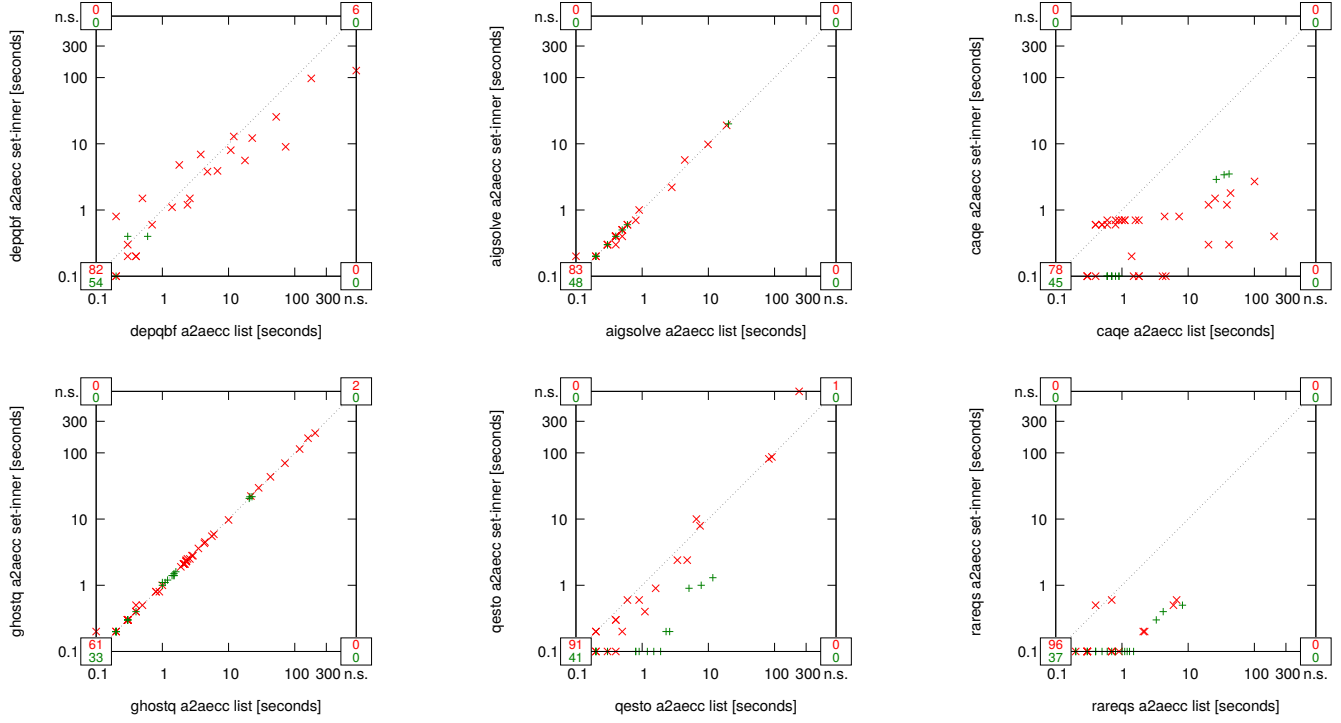


Fig. 1481: Suite Castellini ($n = 169$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

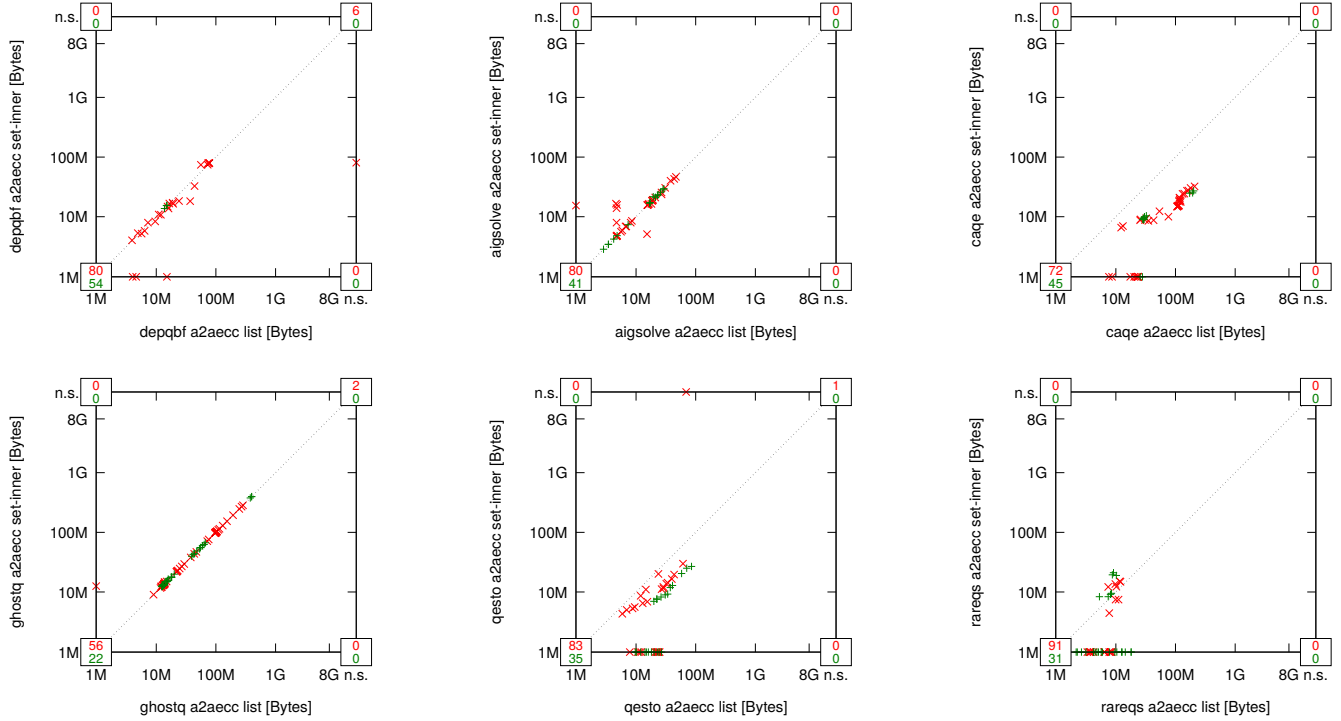


Fig. 1482: Suite Castellini ($n = 169$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

10) Chen-Interian ($n = 194$):

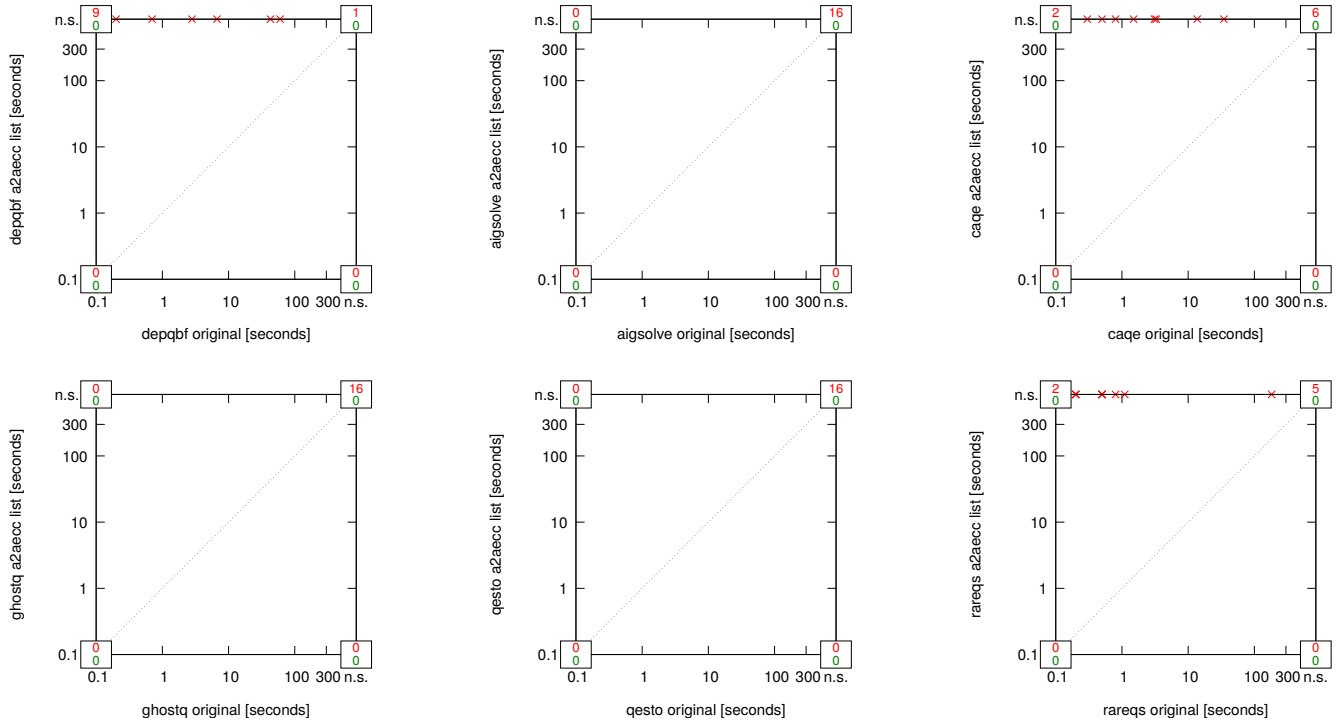


Fig. 1483: Suite Chen-Interian ($n = 194$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

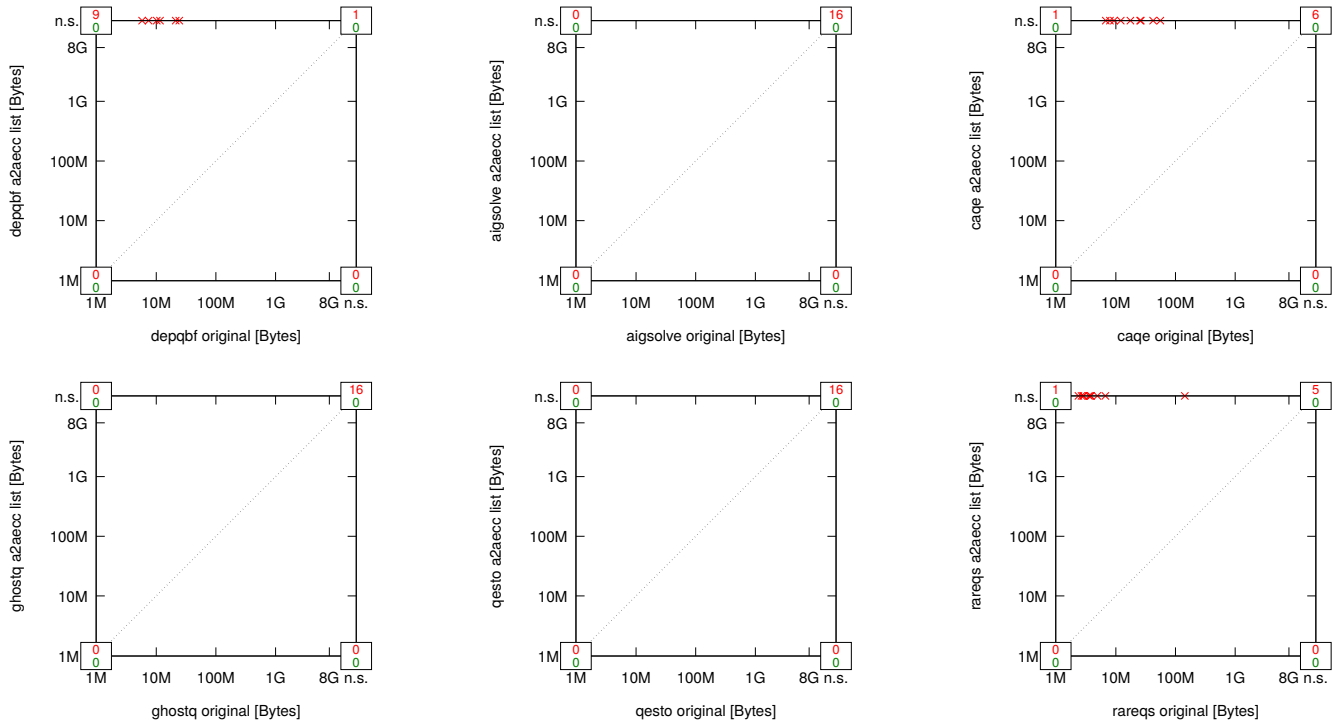


Fig. 1484: Suite Chen-Interian ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

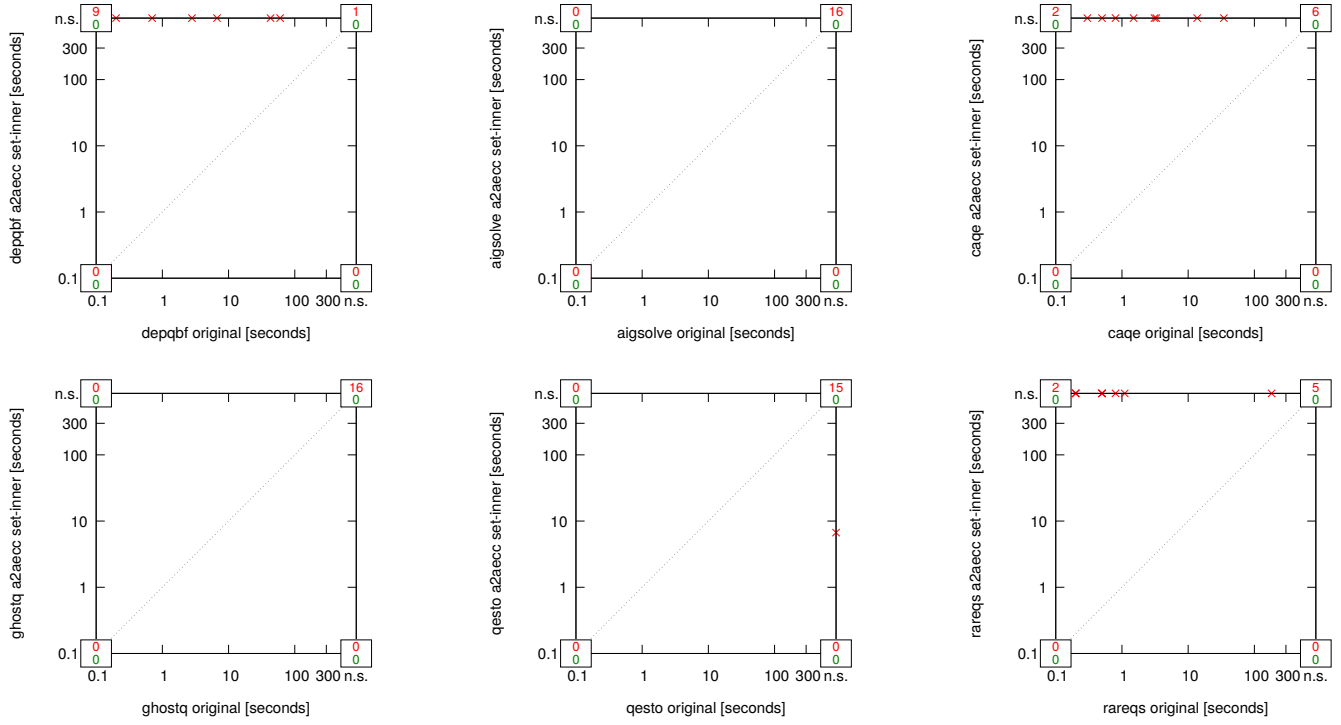


Fig. 1485: Suite Chen-Interian ($n = 194$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

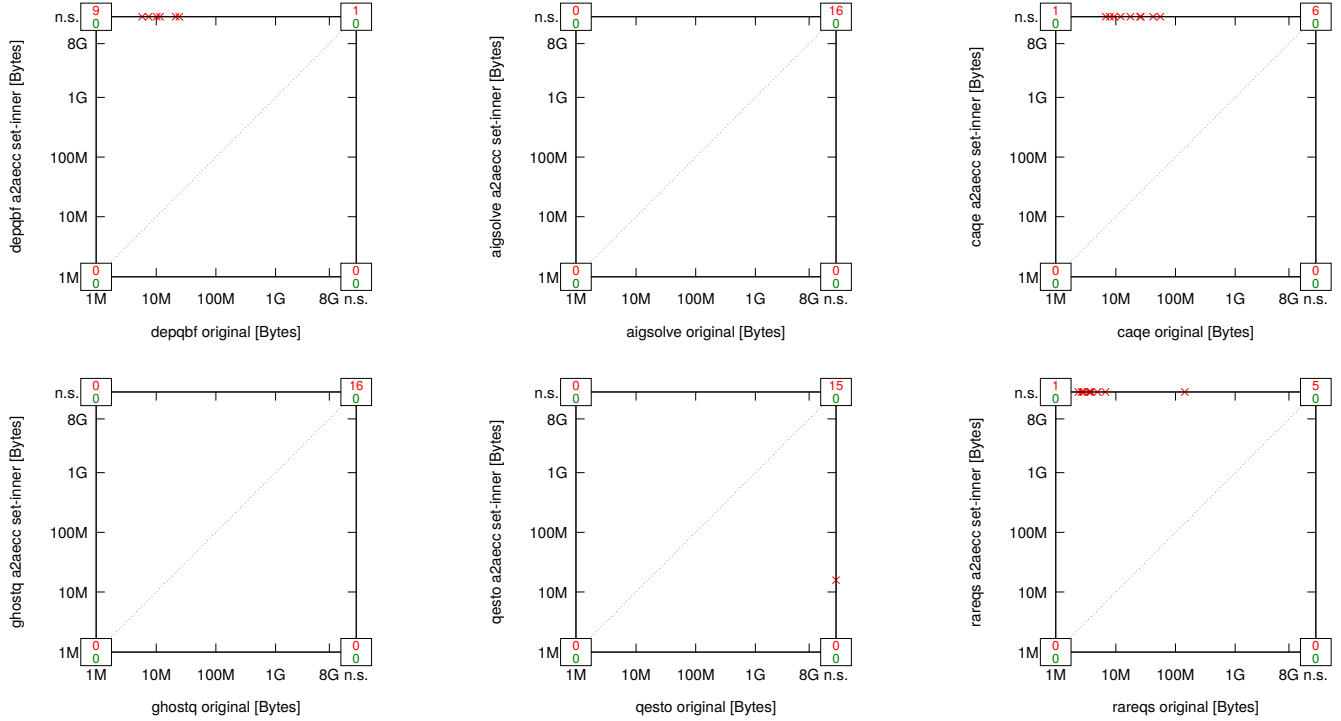


Fig. 1486: Suite Chen-Interian ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

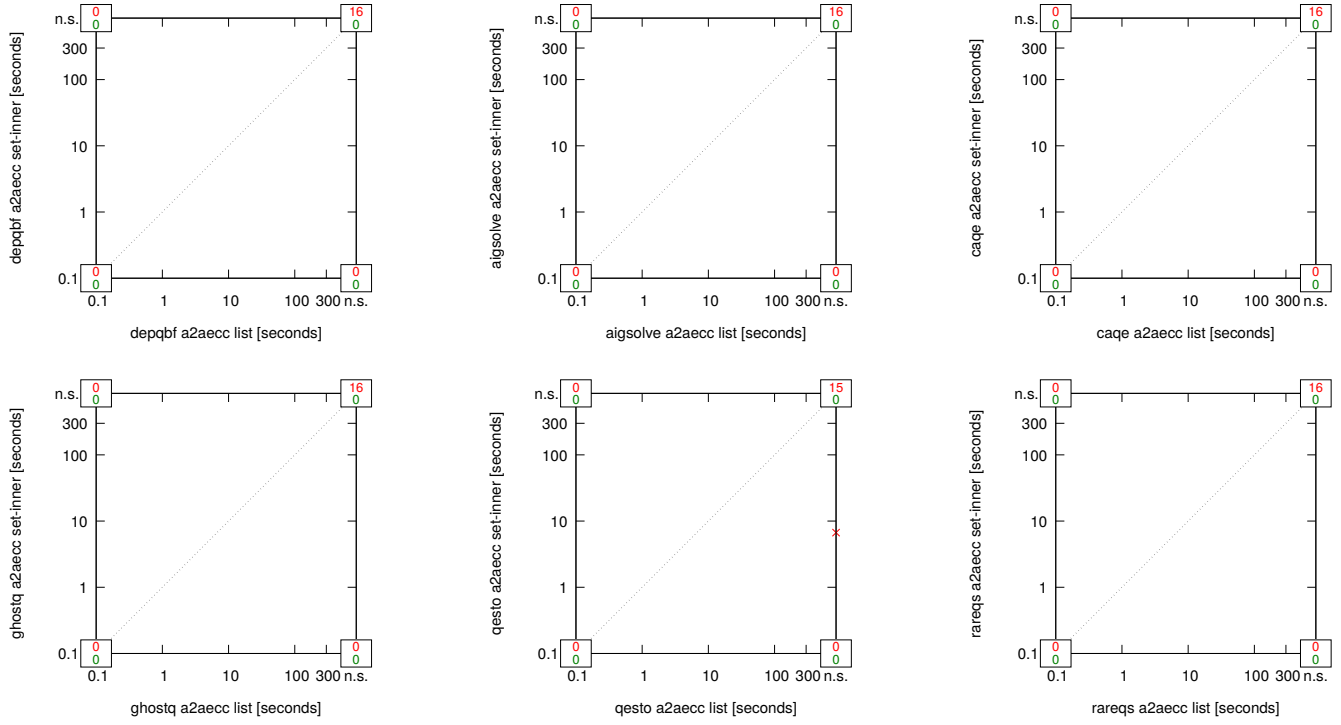


Fig. 1487: Suite Chen-Interian ($n = 194$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

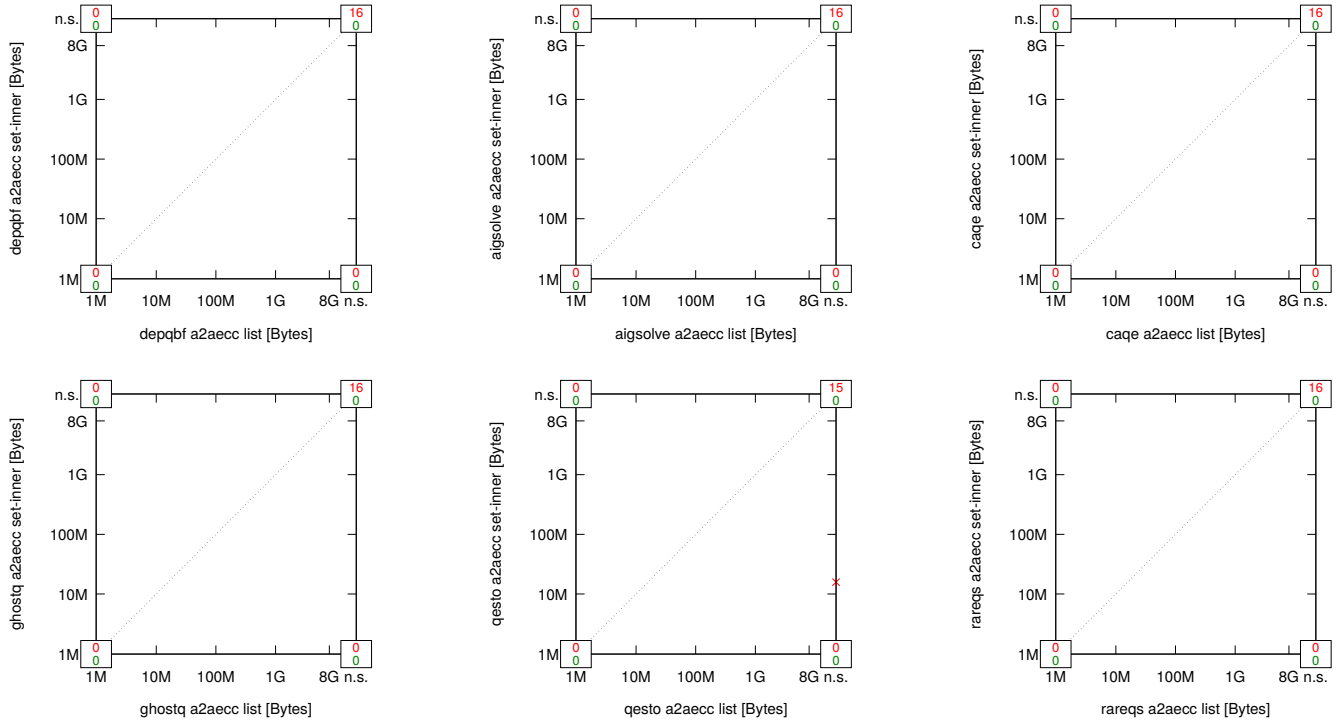


Fig. 1488: Suite Chen-Interian ($n = 194$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

11) Diptarama-Jordan-Shinohara ($n = 14$):

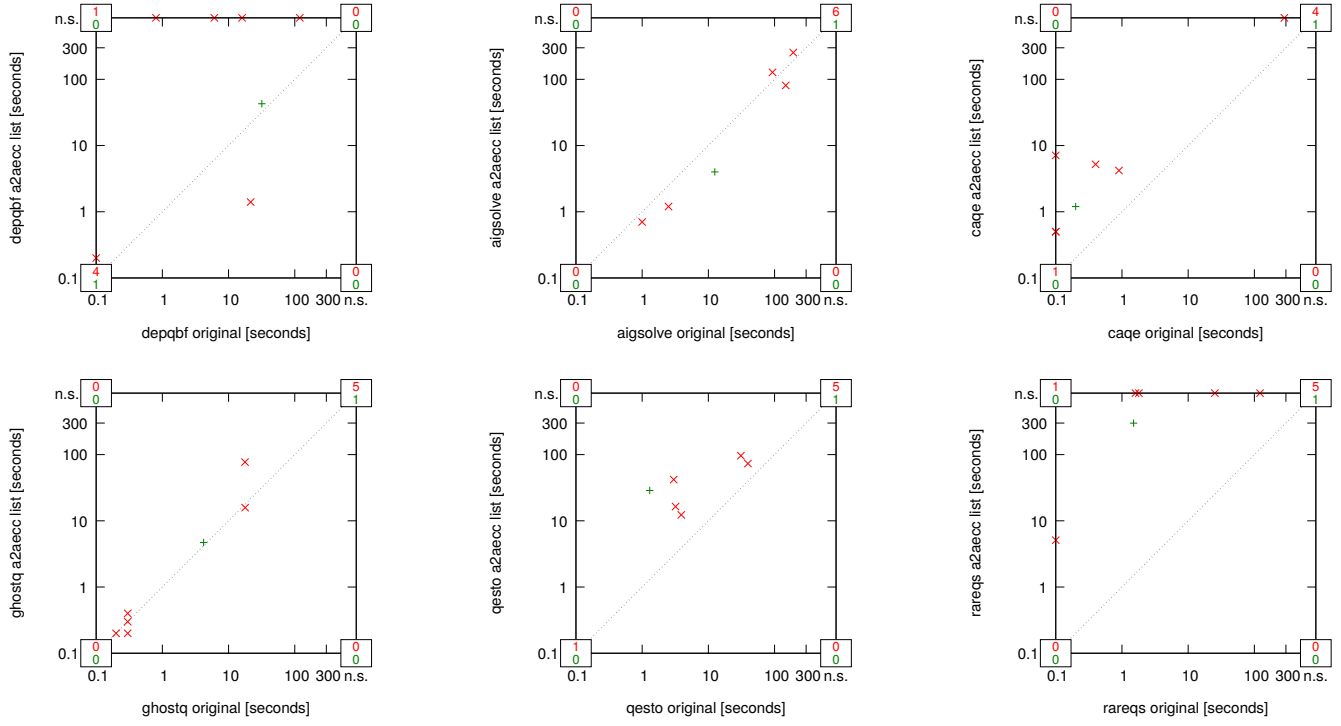


Fig. 1489: Suite Diptarama-Jordan-Shinohara ($n = 14$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

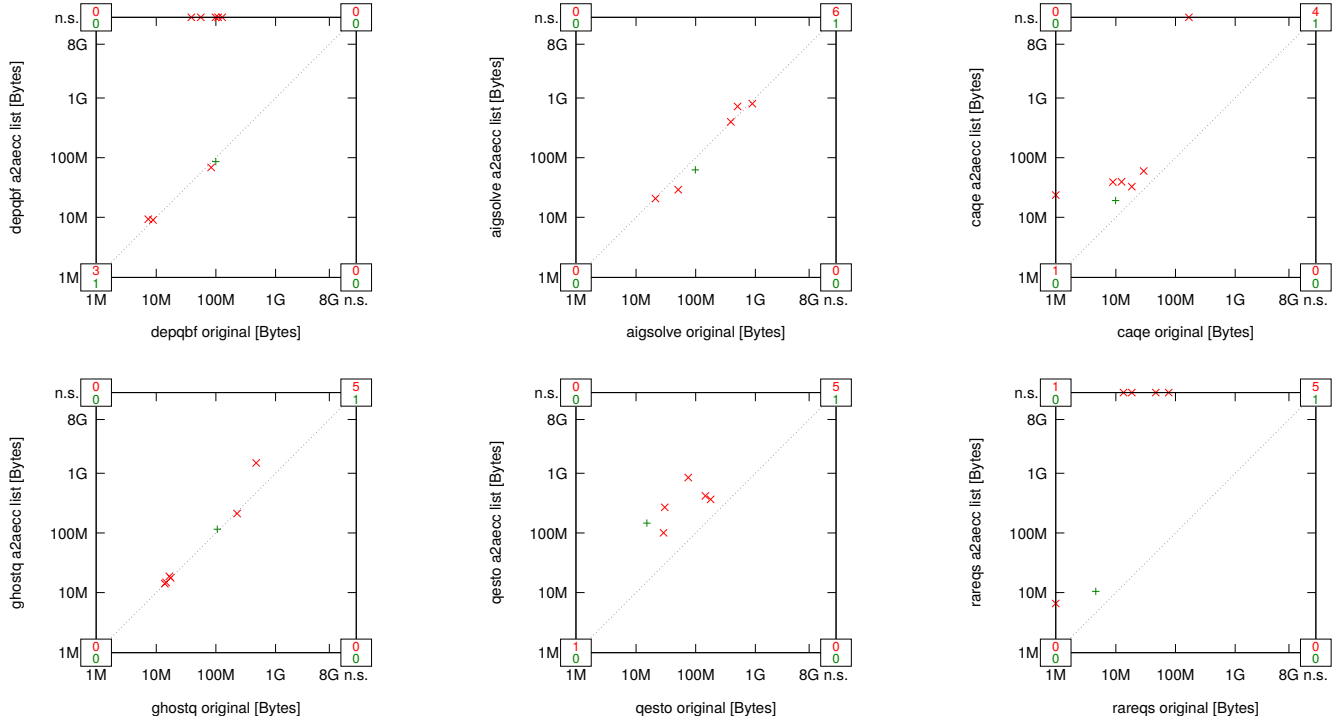


Fig. 1490: Suite Diptarama-Jordan-Shinohara ($n = 14$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

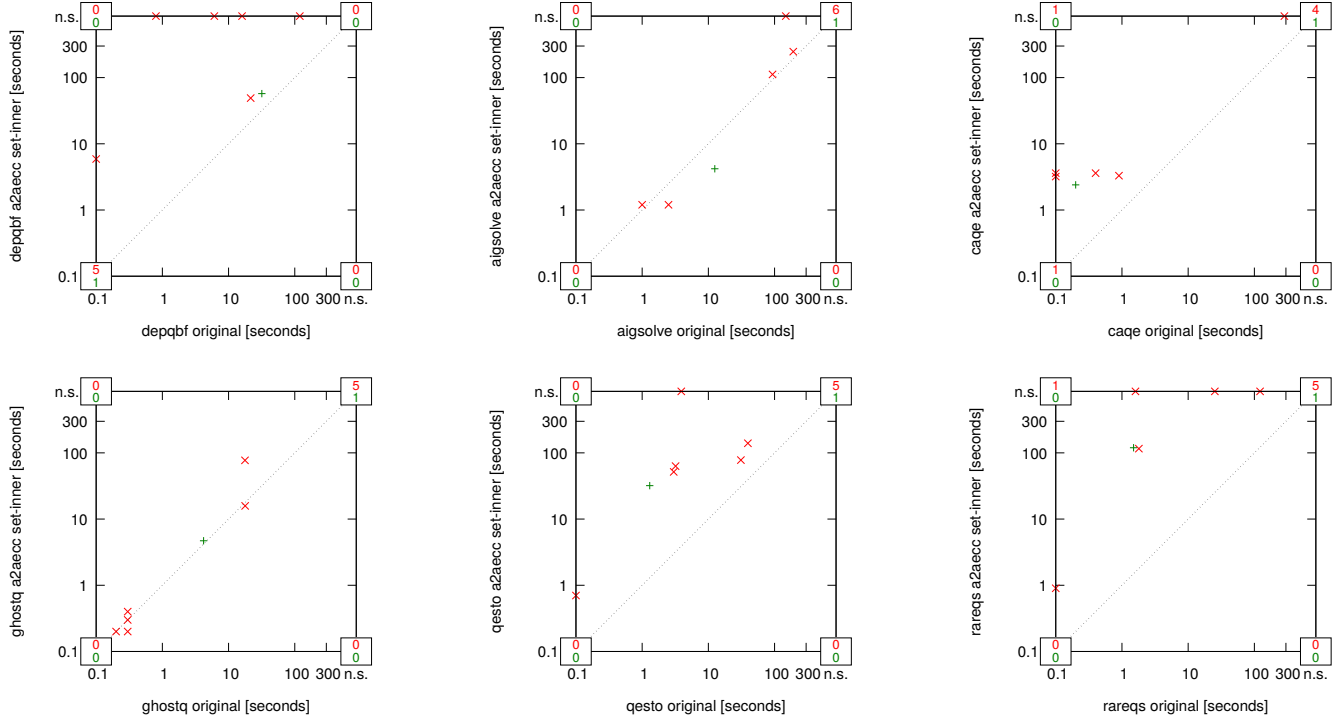


Fig. 1491: Suite Diptarama-Jordan-Shinohara ($n = 14$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

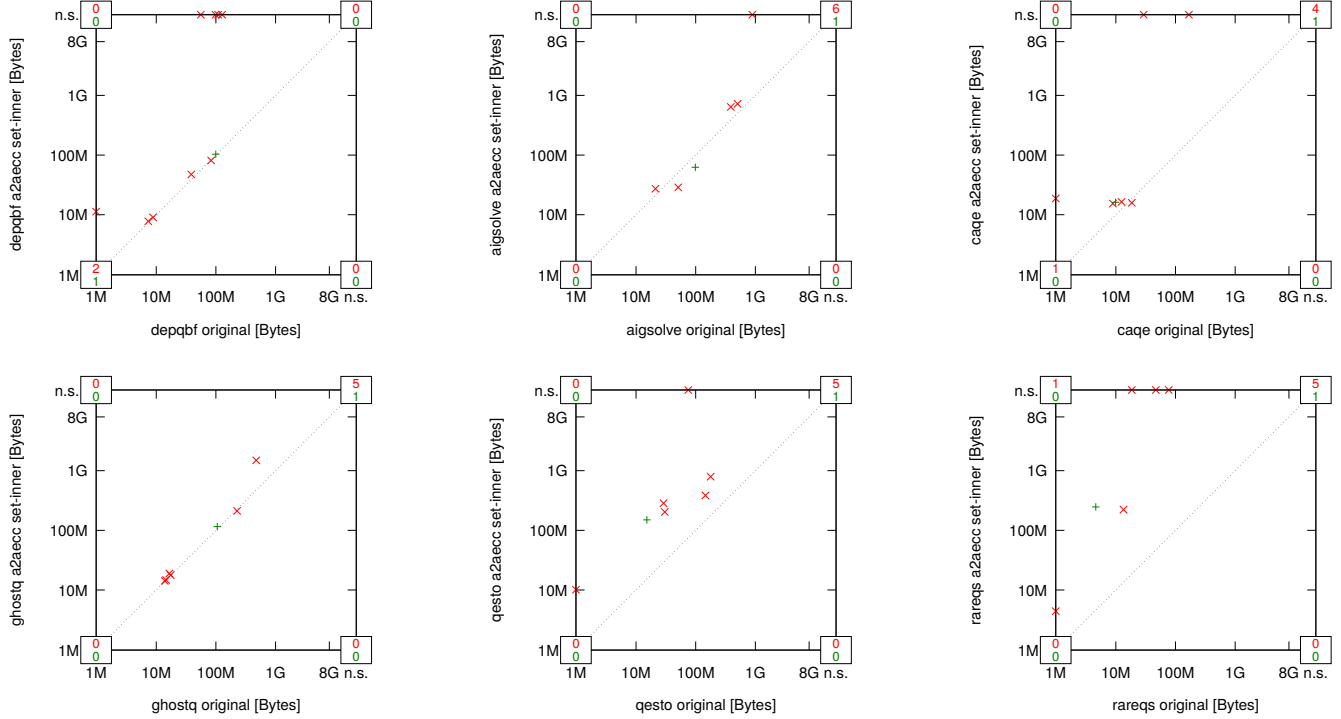


Fig. 1492: Suite Diptarama-Jordan-Shinohara ($n = 14$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

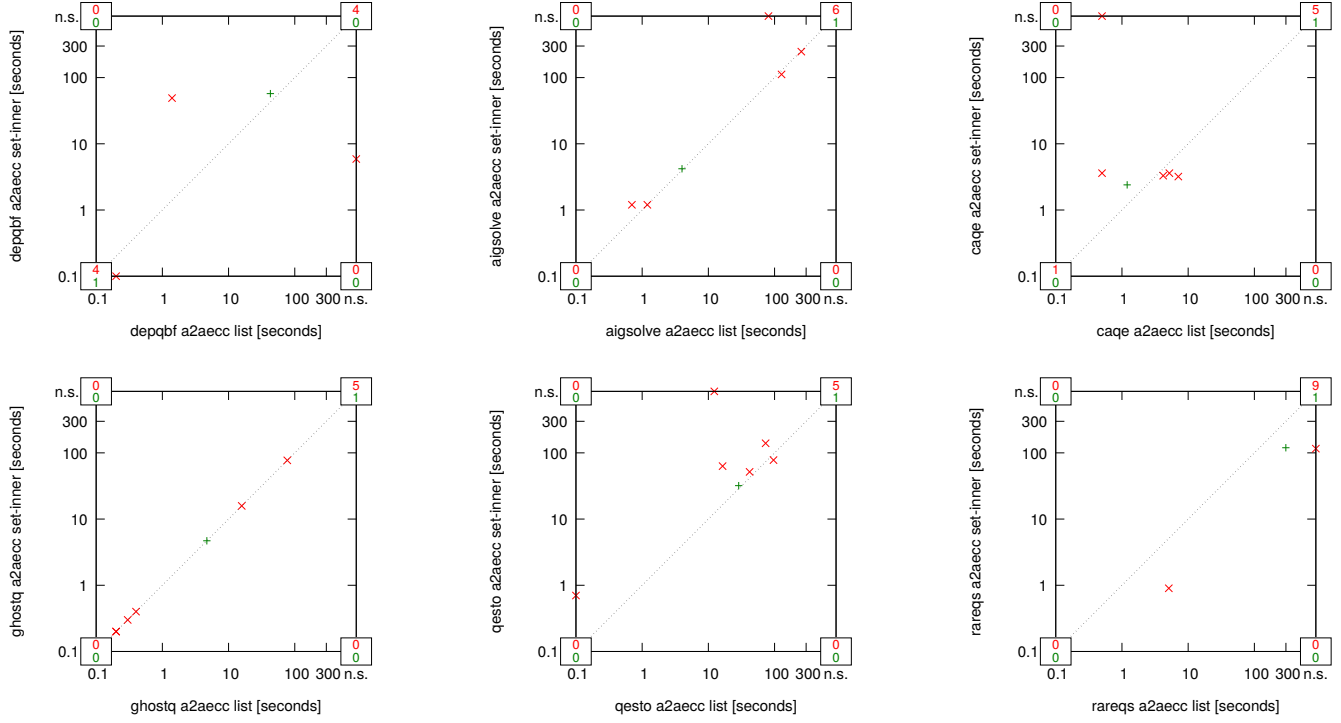


Fig. 1493: Suite Diptarama-Jordan-Shinohara ($n = 14$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

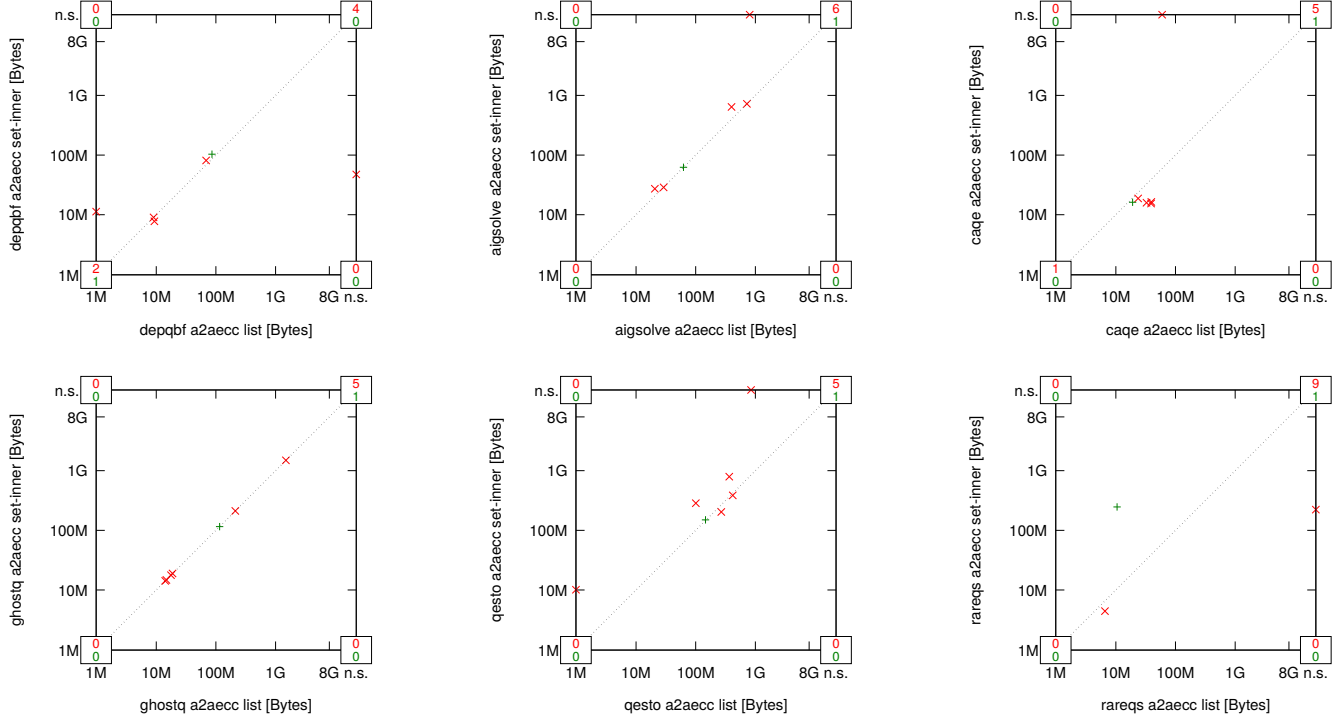


Fig. 1494: Suite Diptarama-Jordan-Shinohara ($n = 14$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

12) Egly-Seidl-Tompits-Woltran-Zolda ($n = 194$):

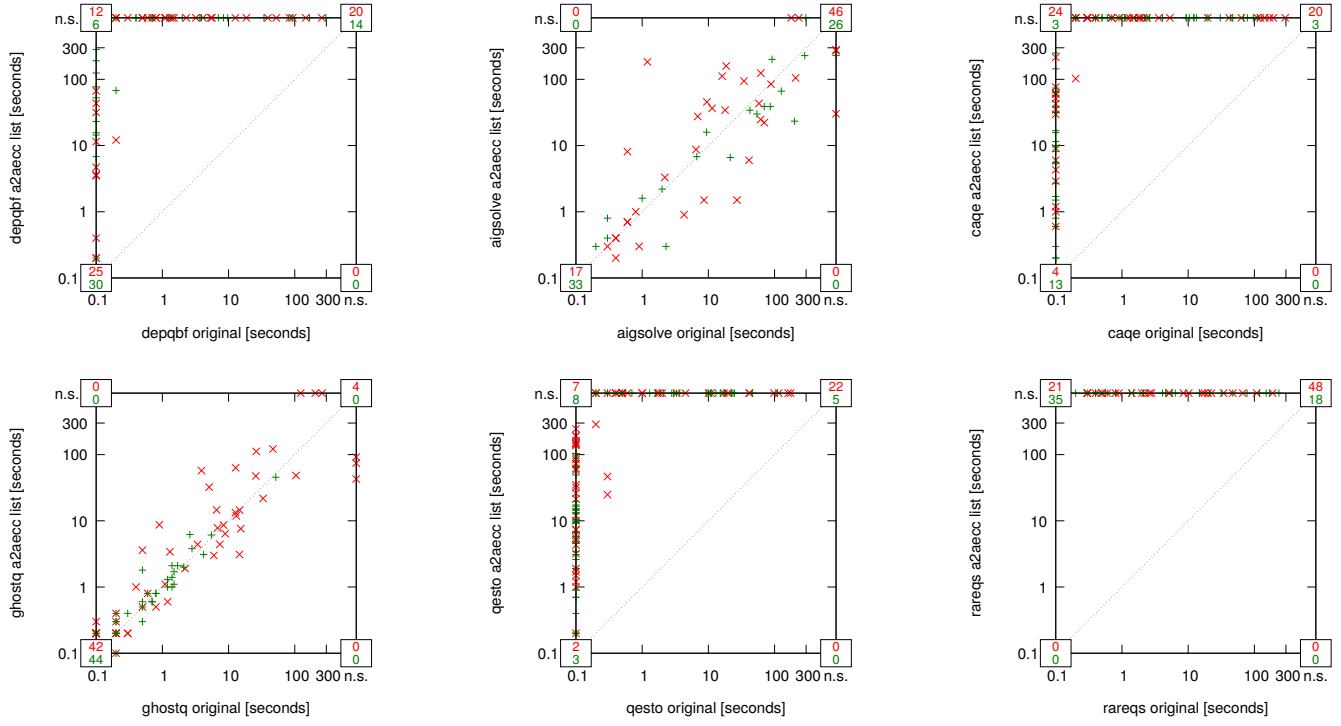


Fig. 1495: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 194$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

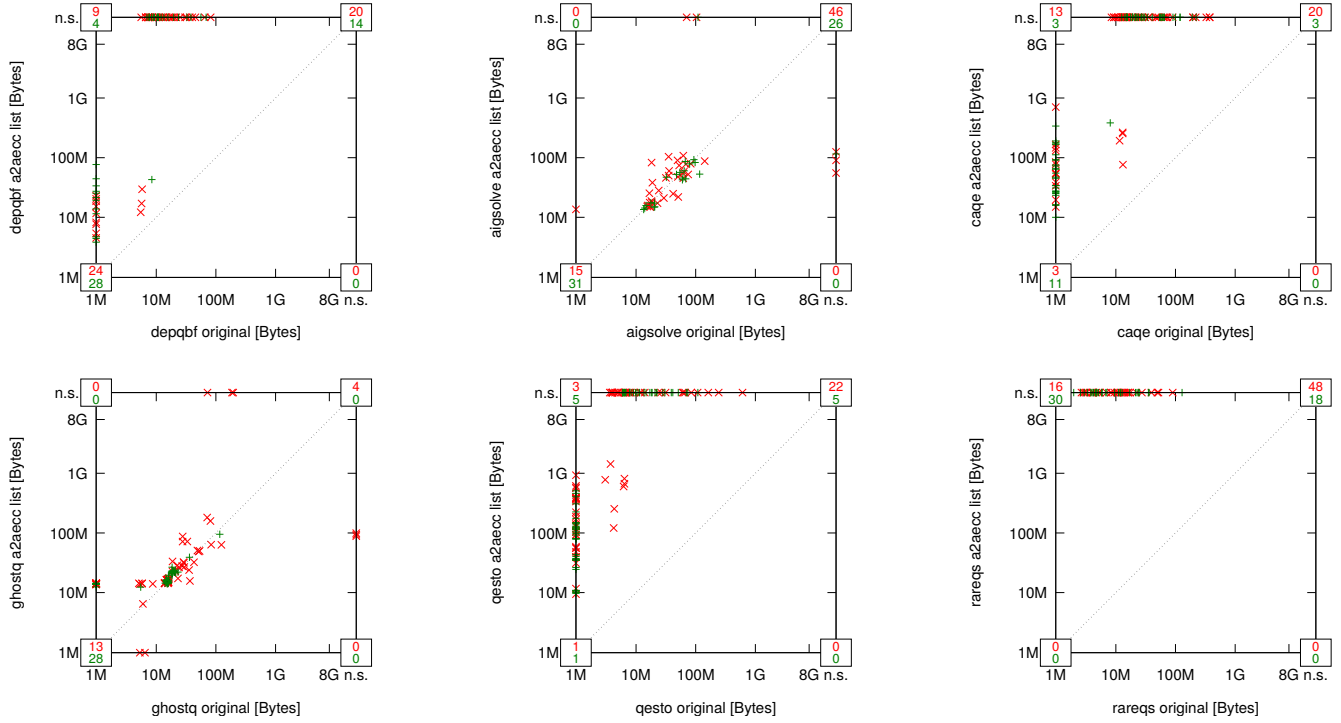


Fig. 1496: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

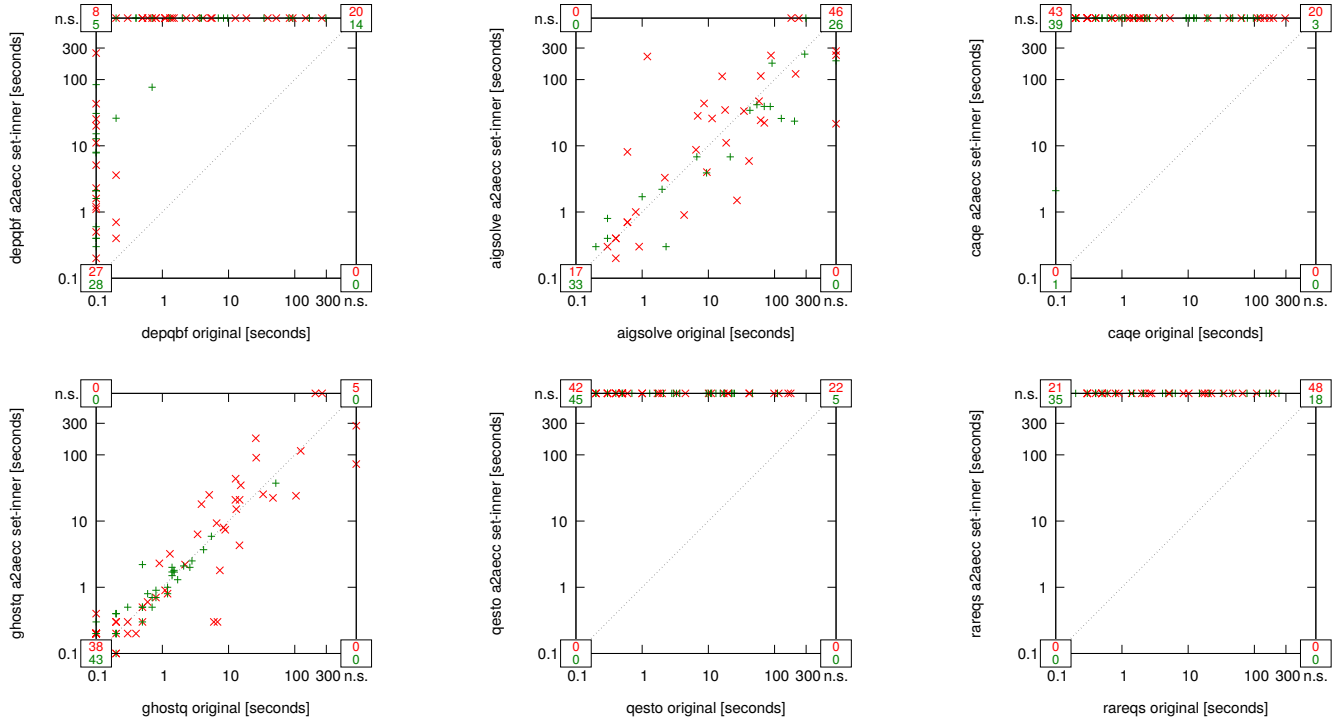


Fig. 1497: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 194$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

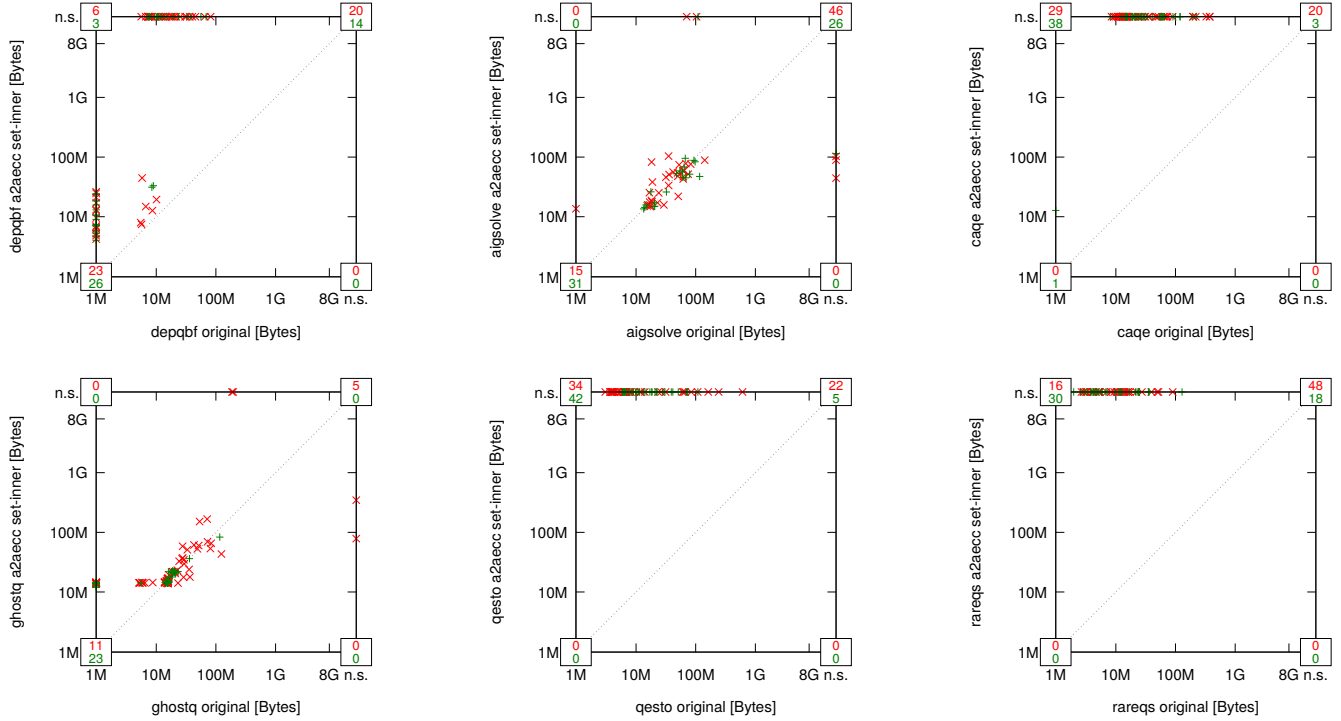


Fig. 1498: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

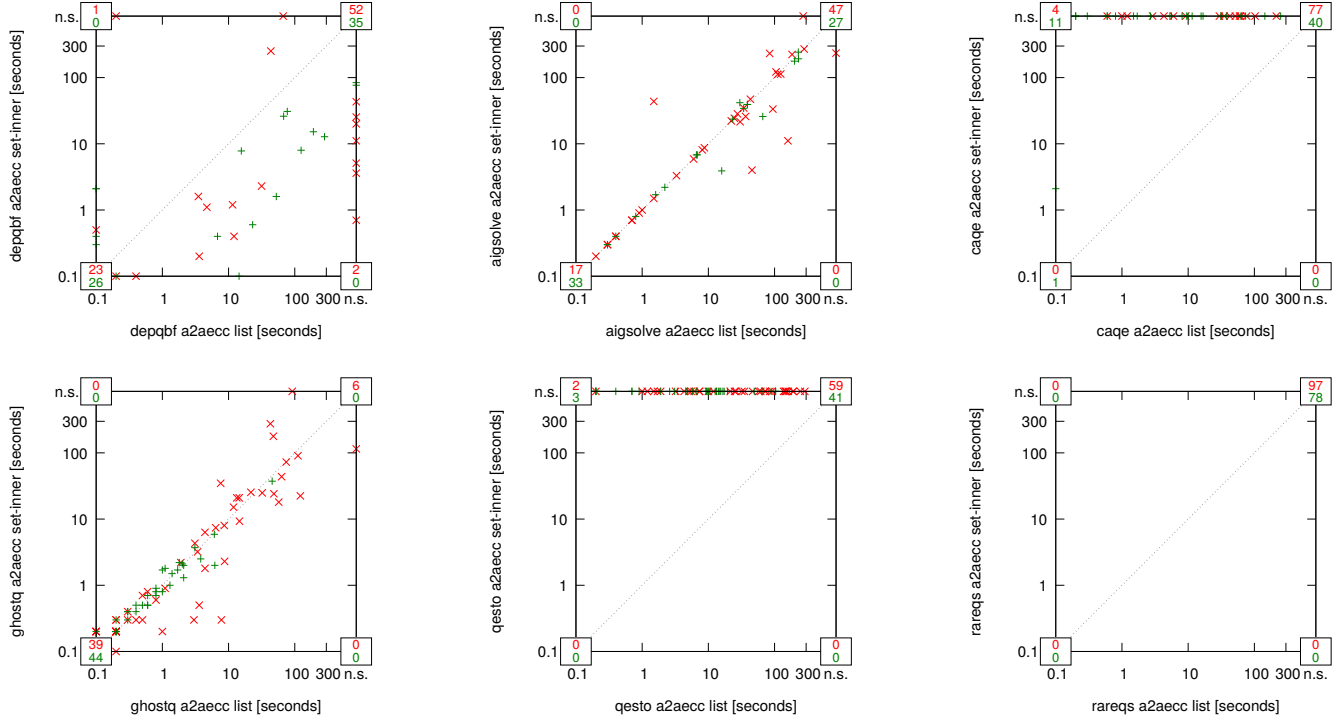


Fig. 1499: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 194$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

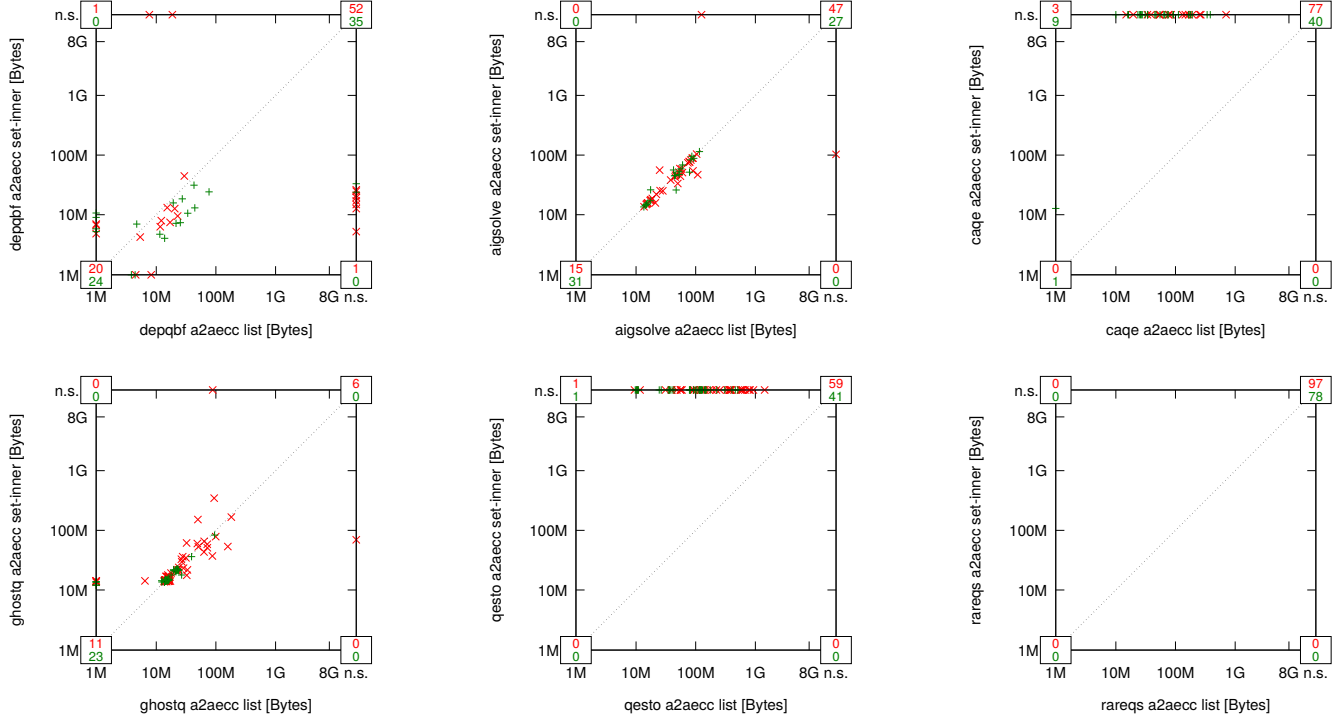


Fig. 1500: Suite Egly-Seidl-Tompits-Woltran-Zolda ($n = 194$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

13) Faber-Leone-Maratea-Ricca ($n = 194$):

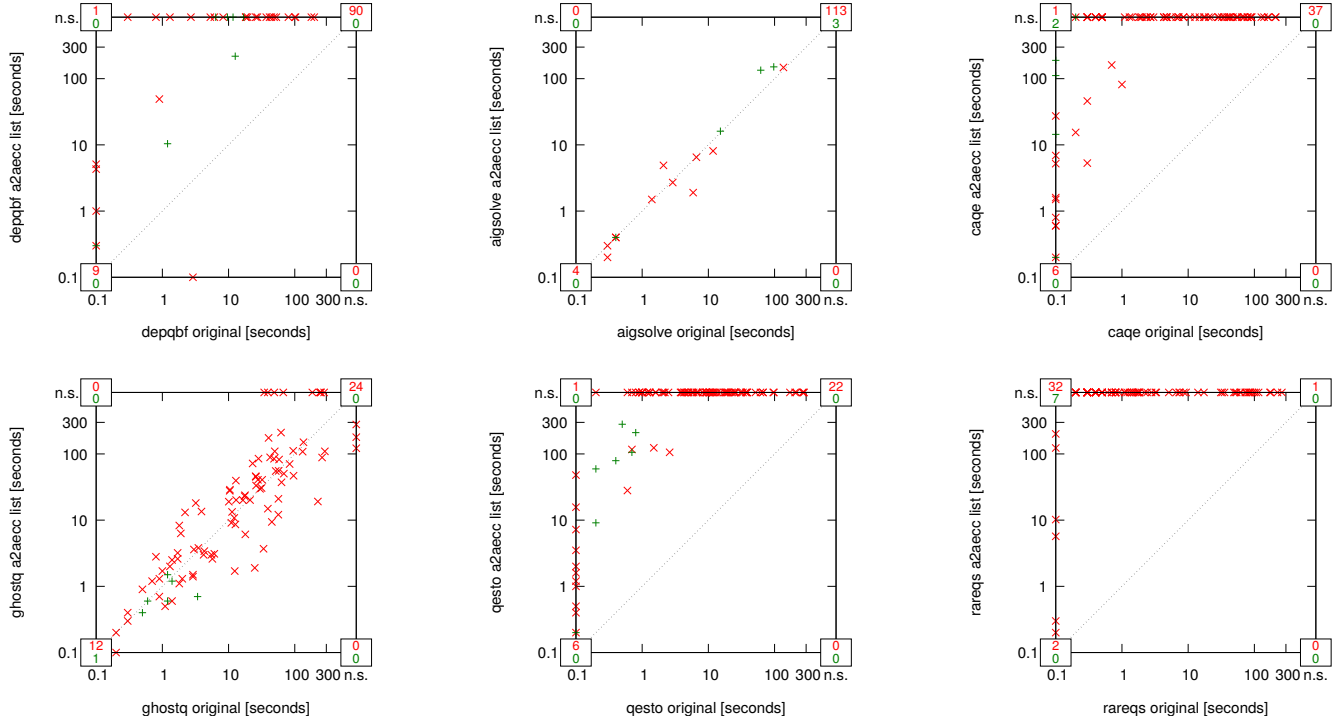


Fig. 1501: Suite Faber-Leone-Maratea-Ricca ($n = 194$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

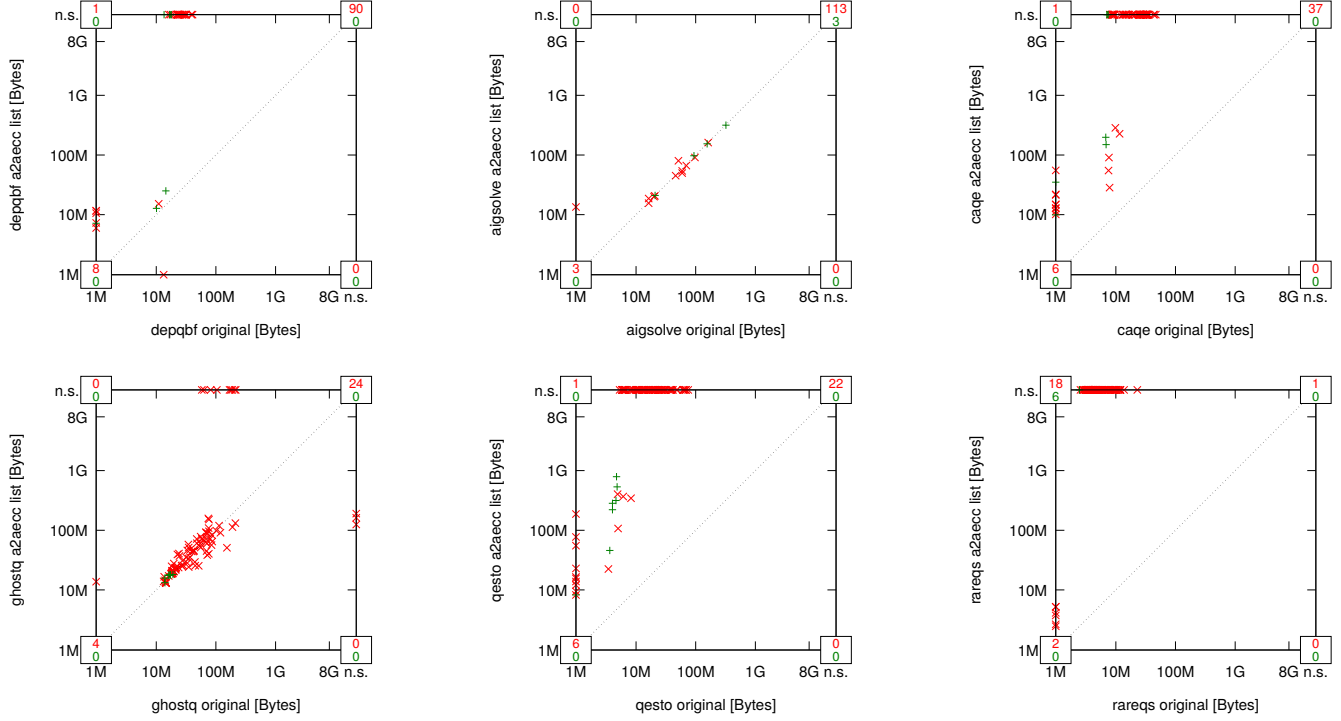


Fig. 1502: Suite Faber-Leone-Maratea-Ricca ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

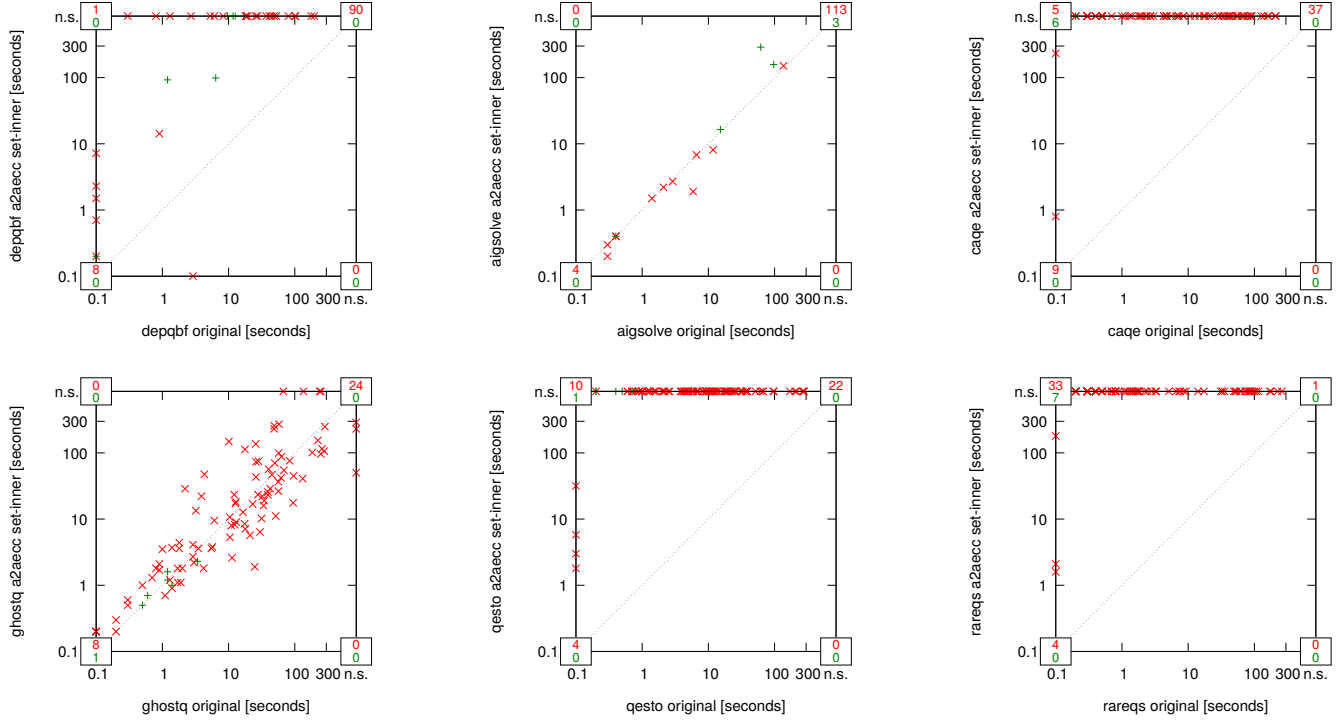


Fig. 1503: Suite Faber-Leone-Maratea-Ricca ($n = 194$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

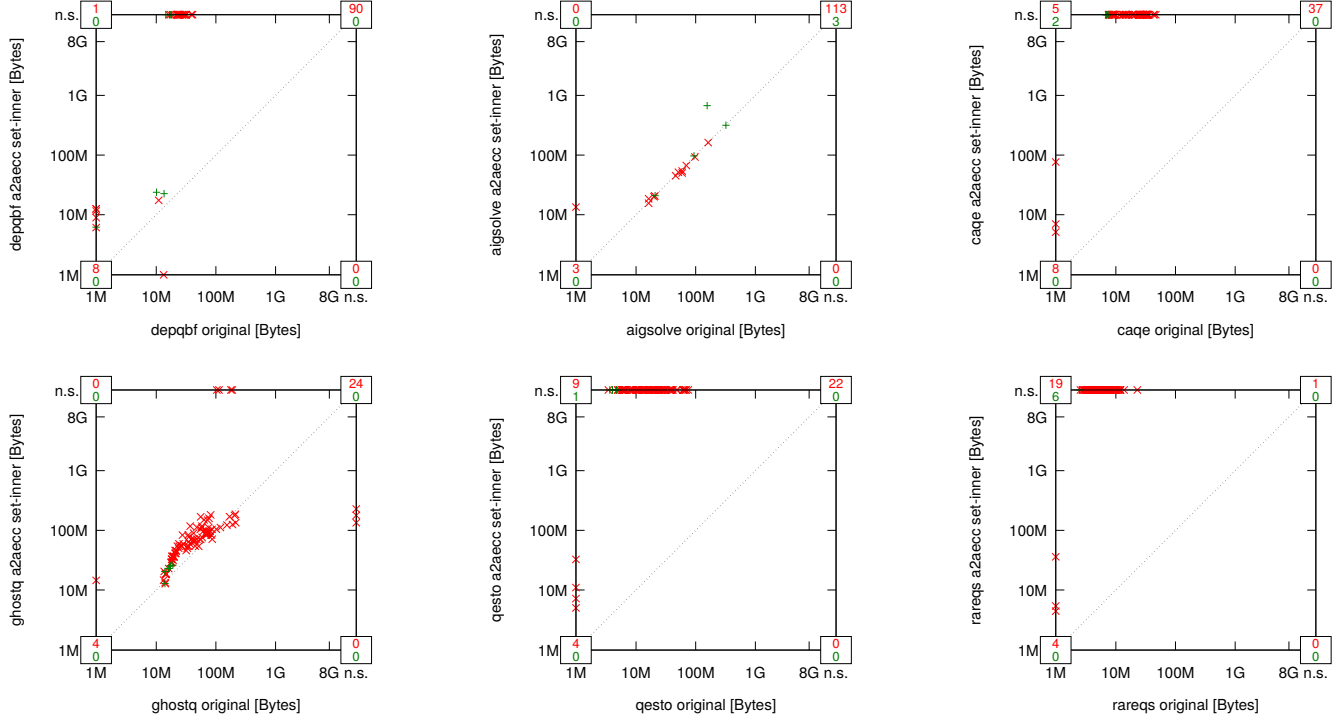


Fig. 1504: Suite Faber-Leone-Maratea-Ricca ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

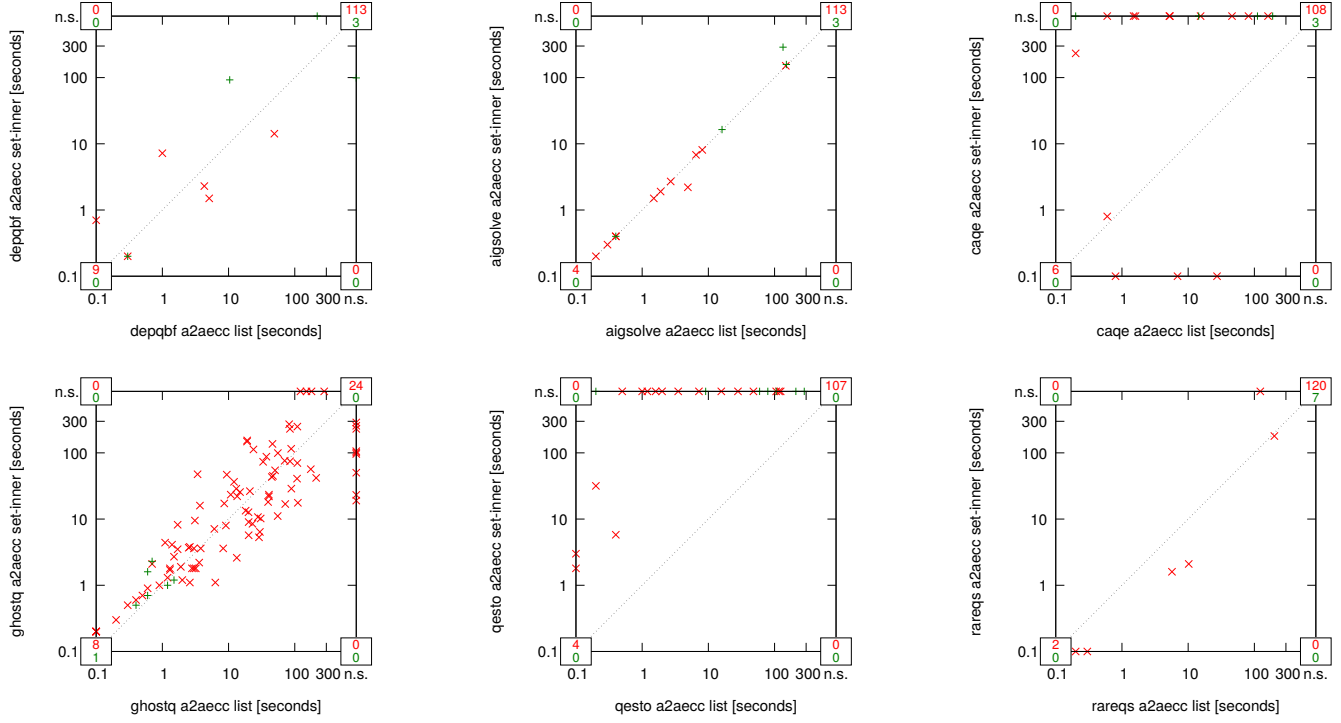


Fig. 1505: Suite Faber-Leone-Maratea-Ricca ($n = 194$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

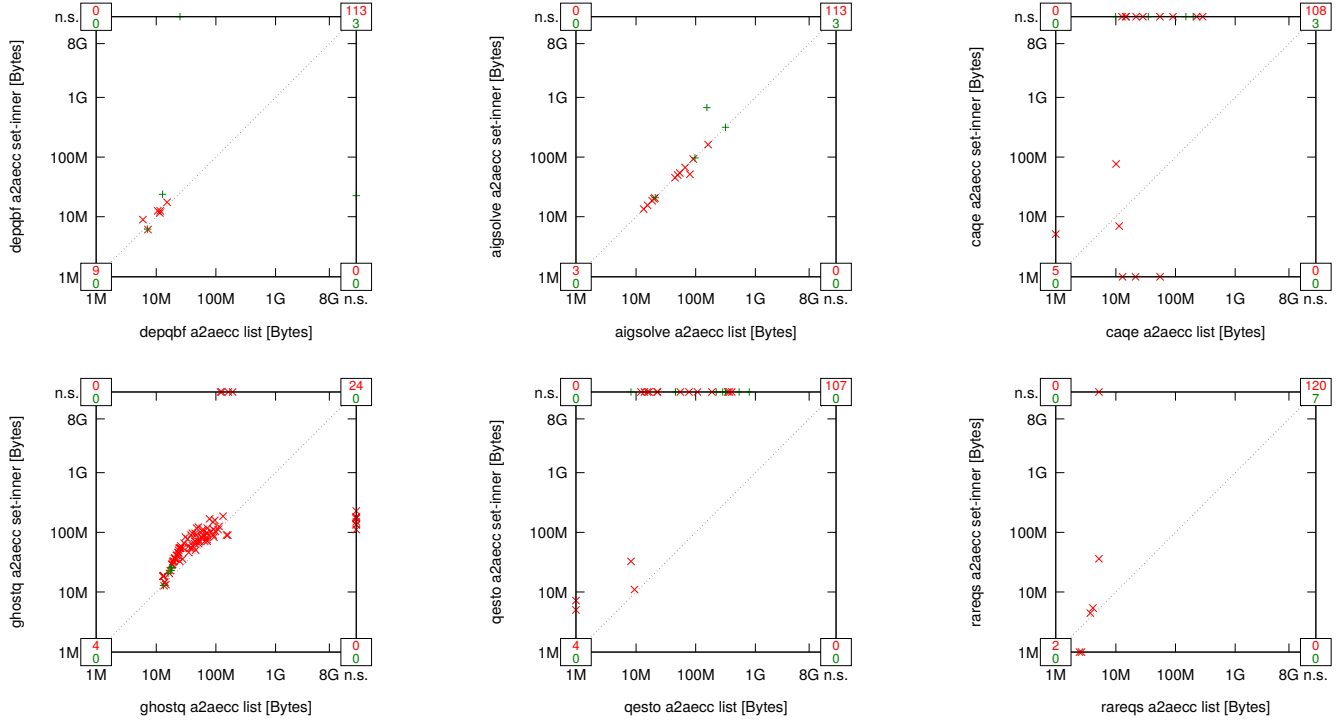


Fig. 1506: Suite Faber-Leone-Maratea-Ricca ($n = 194$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

14) Gent-Rowley ($n = 193$):

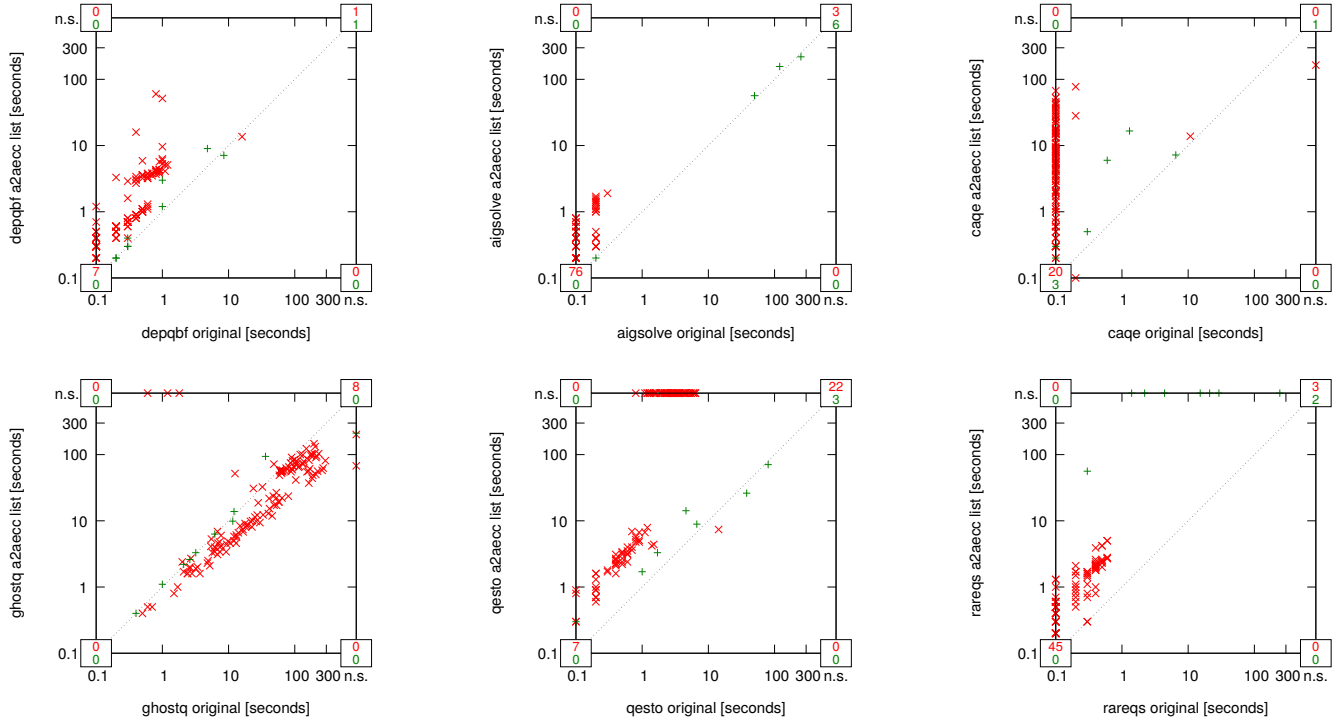


Fig. 1507: Suite Gent-Rowley ($n = 193$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

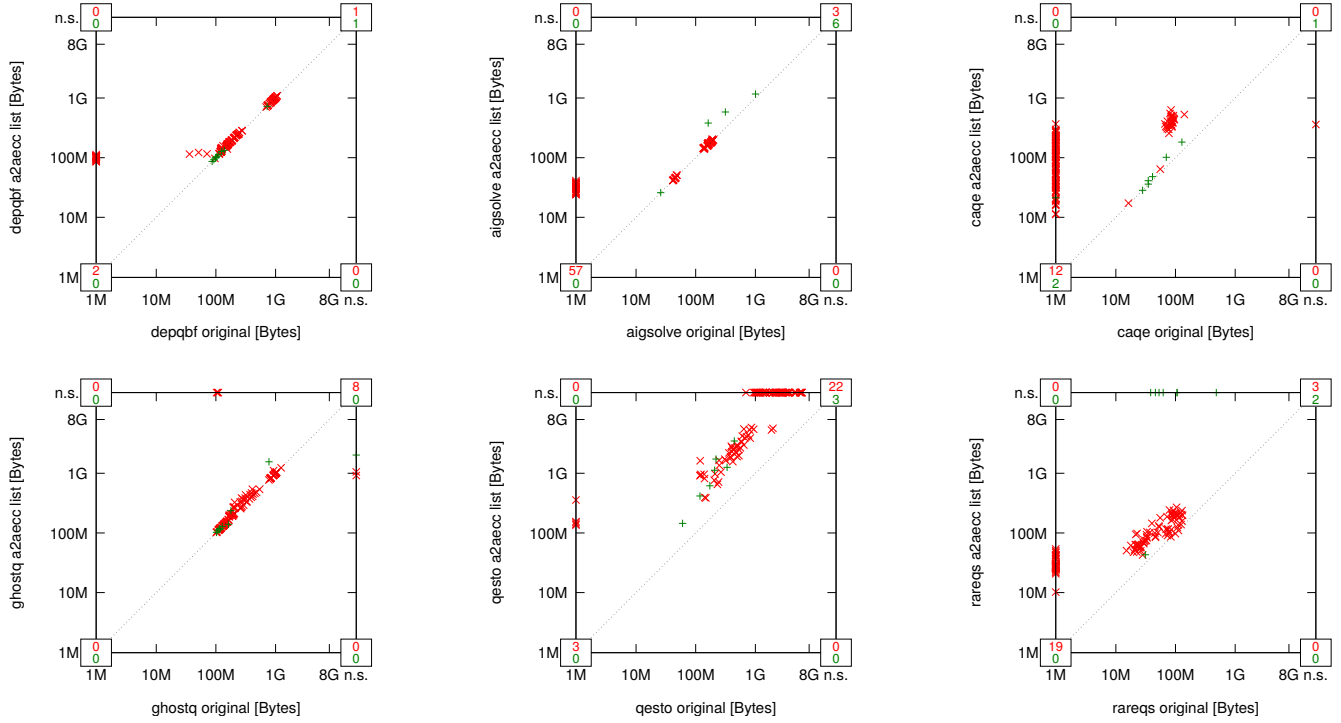


Fig. 1508: Suite Gent-Rowley ($n = 193$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

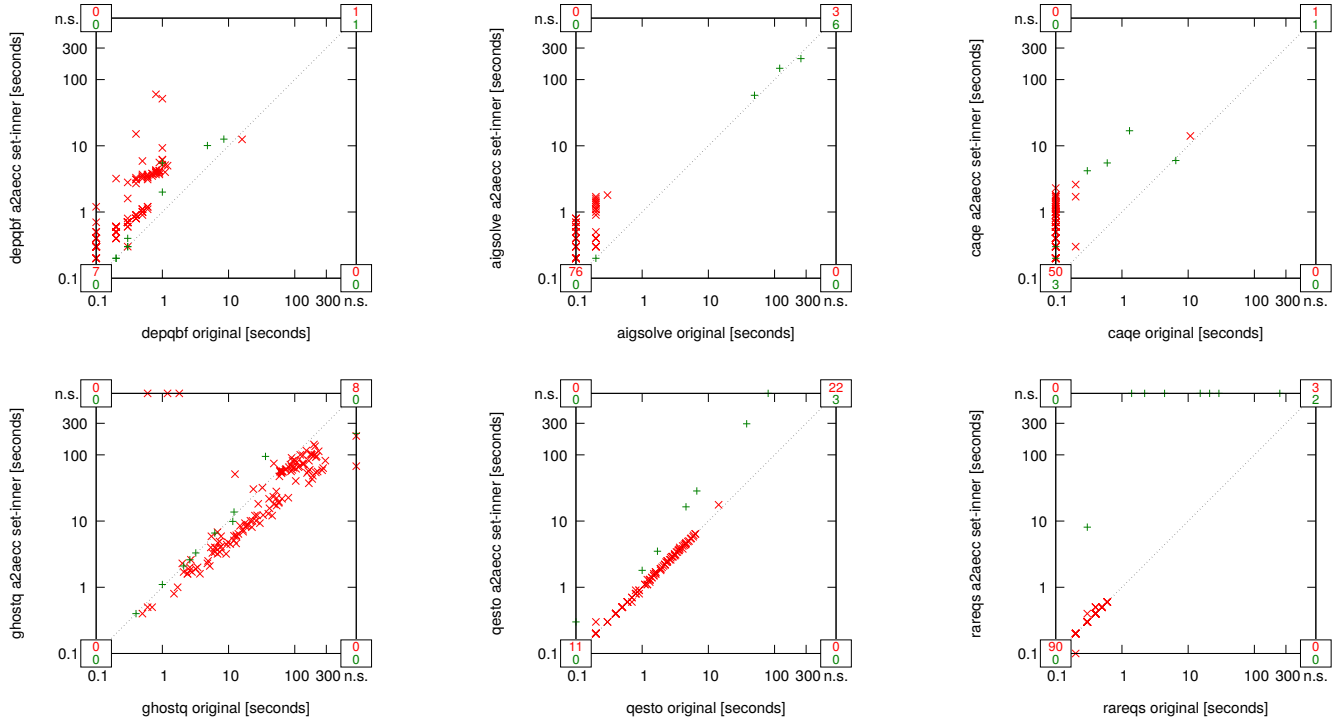


Fig. 1509: Suite Gent-Rowley ($n = 193$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

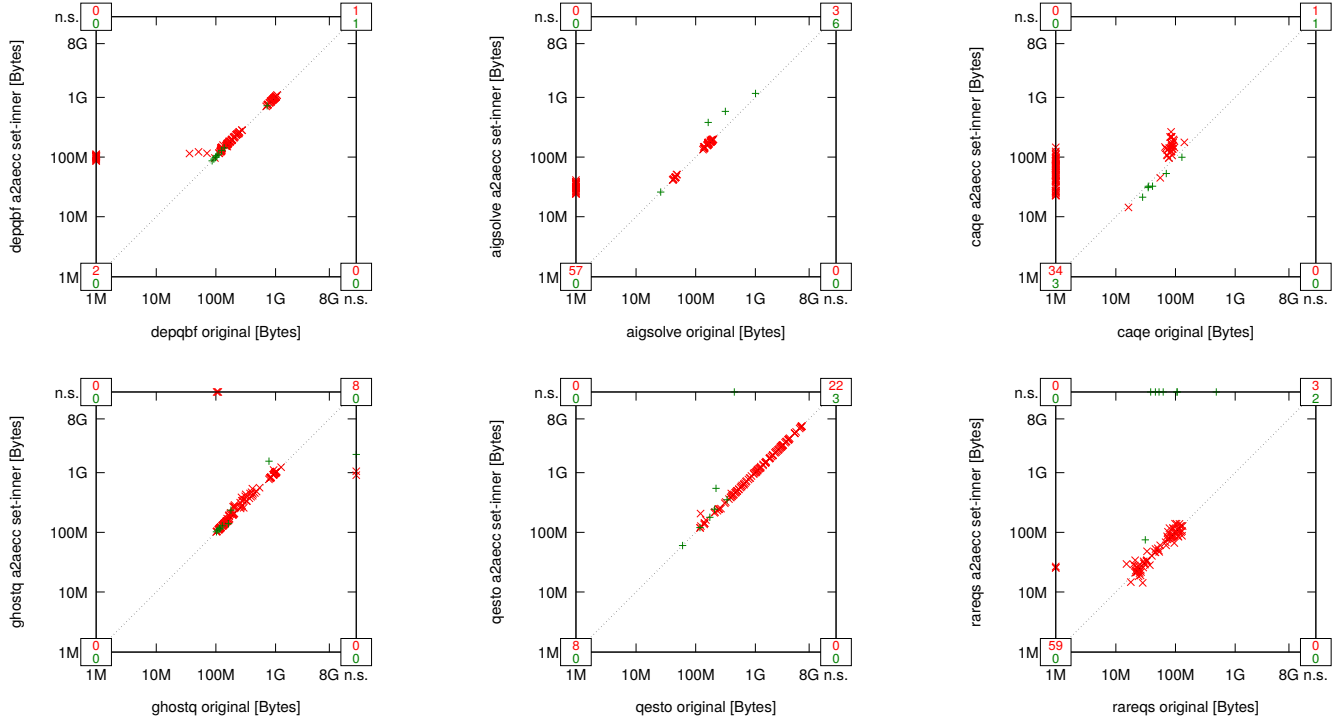


Fig. 1510: Suite Gent-Rowley ($n = 193$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

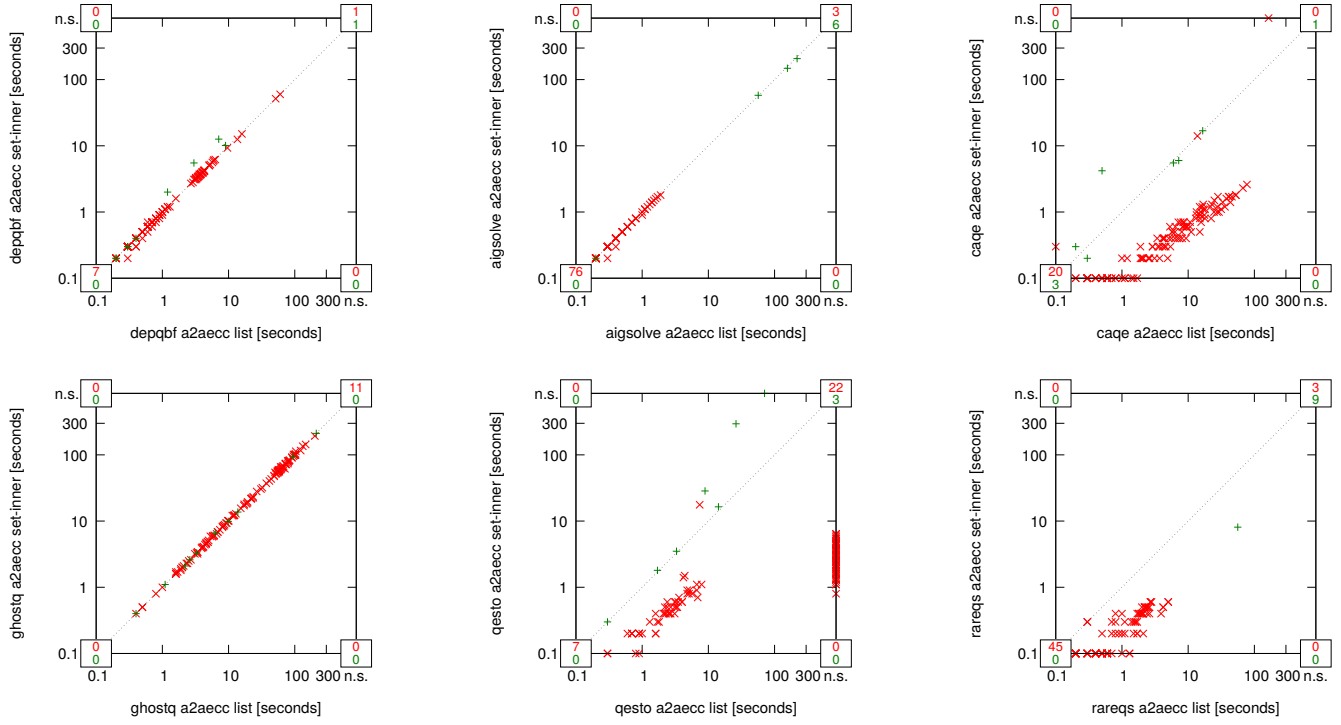


Fig. 1511: Suite Gent-Rowley ($n = 193$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

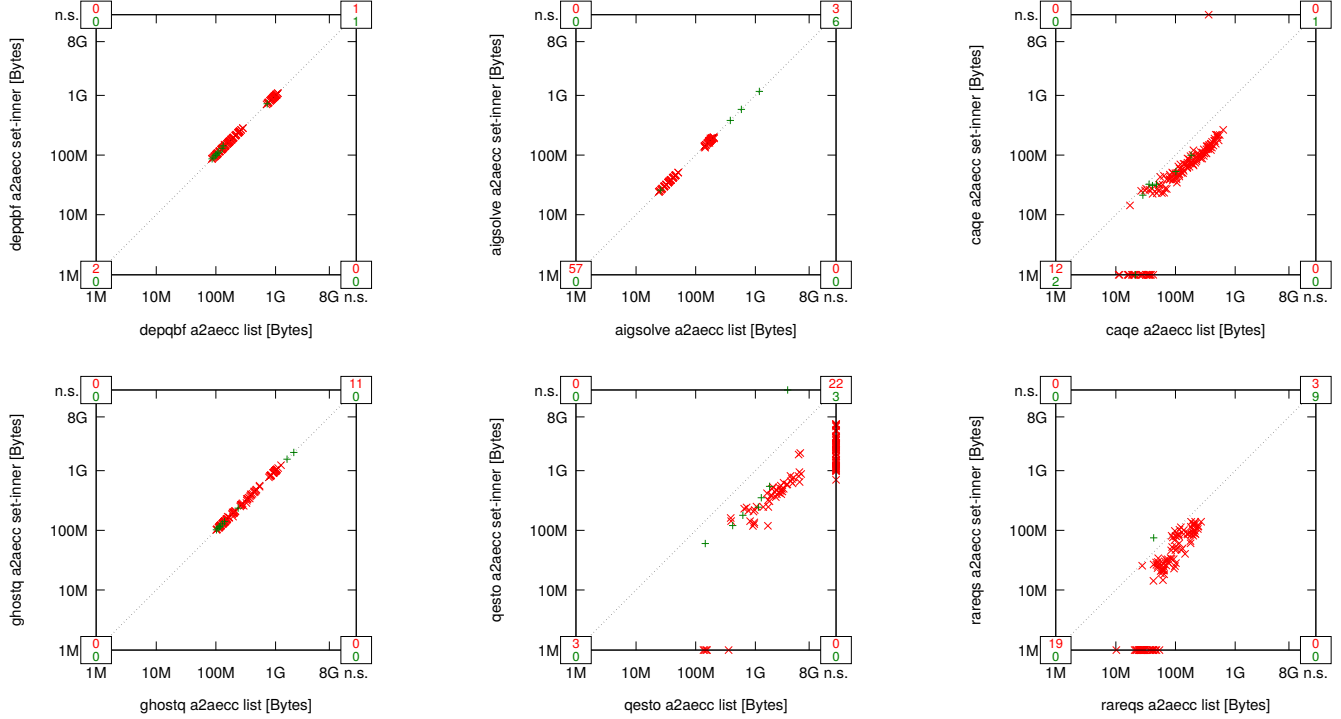


Fig. 1512: Suite Gent-Rowley ($n = 193$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

15) *Herbstritt* ($n = 194$):

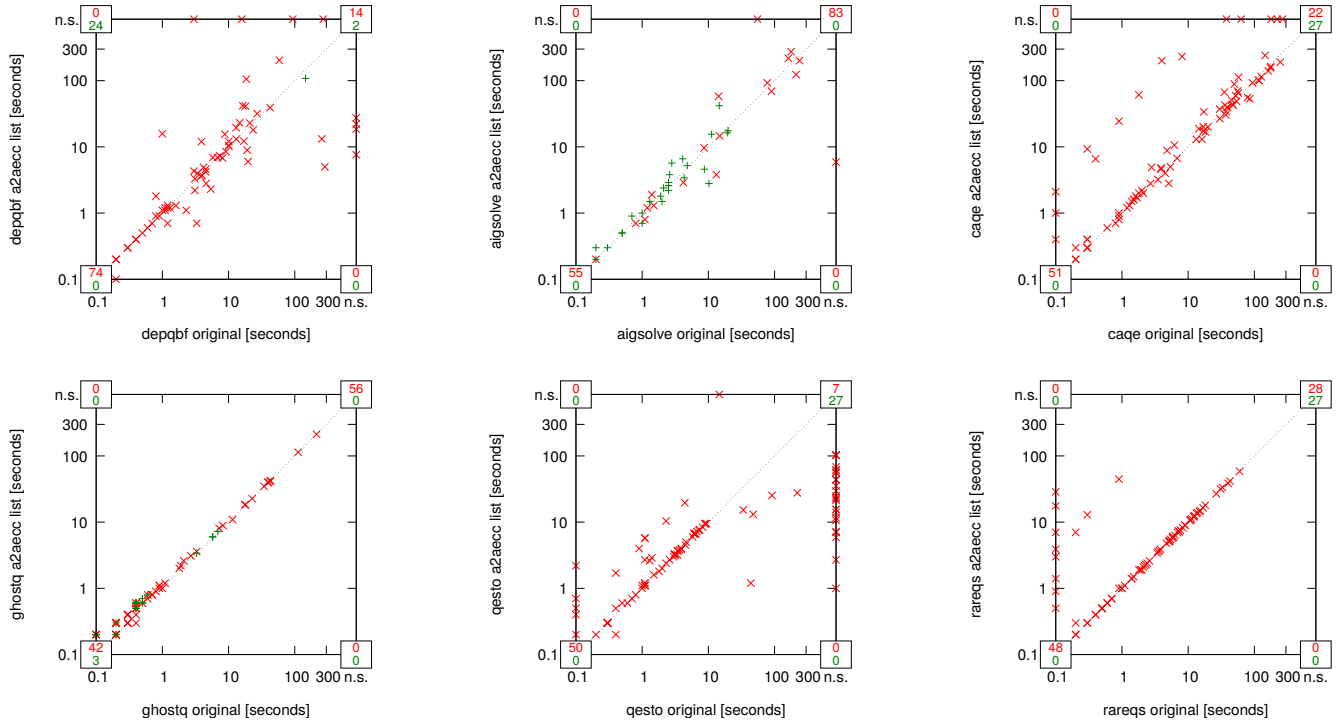


Fig. 1513: Suite *Herbstritt* ($n = 194$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

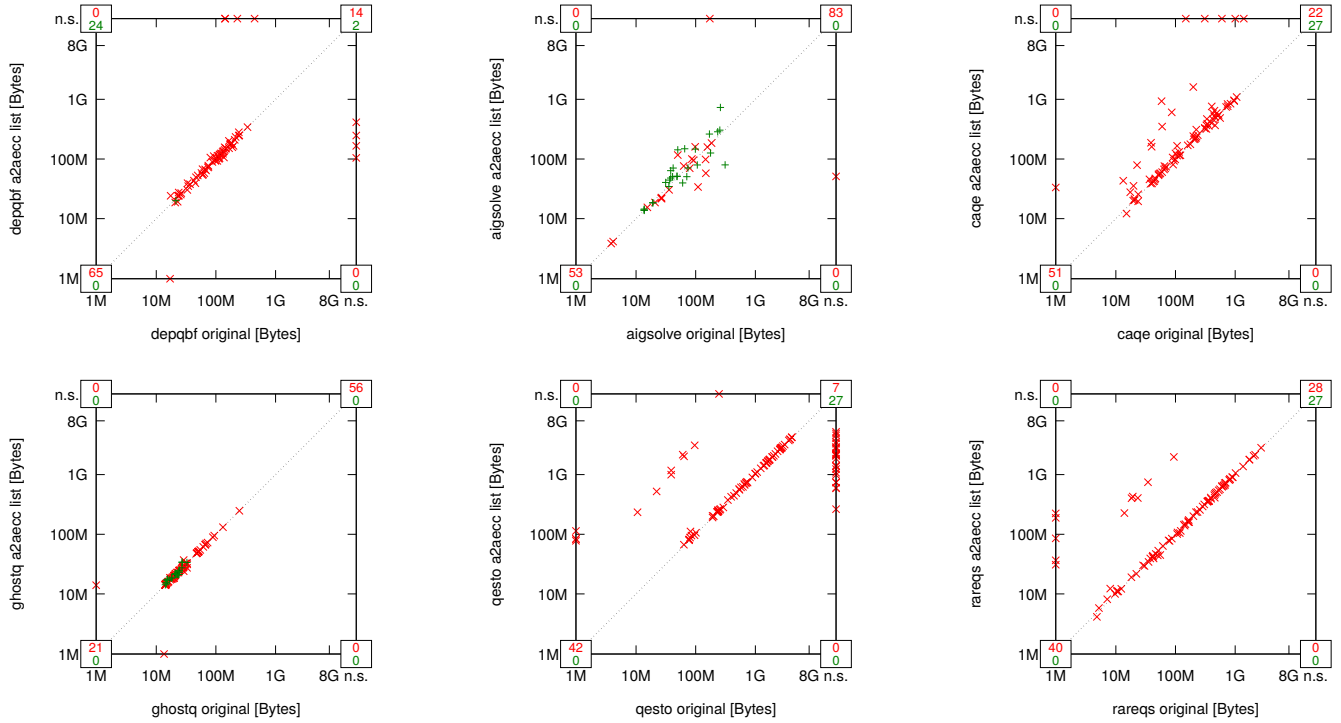


Fig. 1514: Suite *Herbstritt* ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

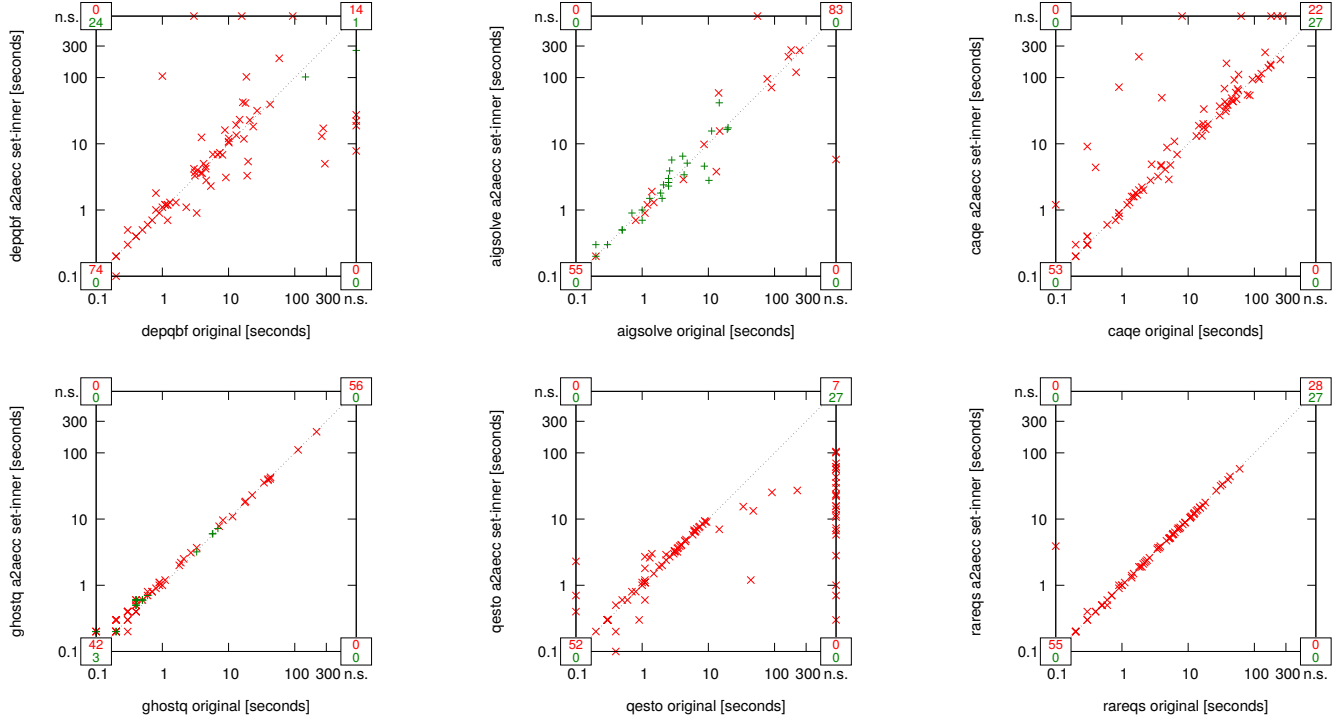


Fig. 1515: Suite Herbstritt ($n = 194$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

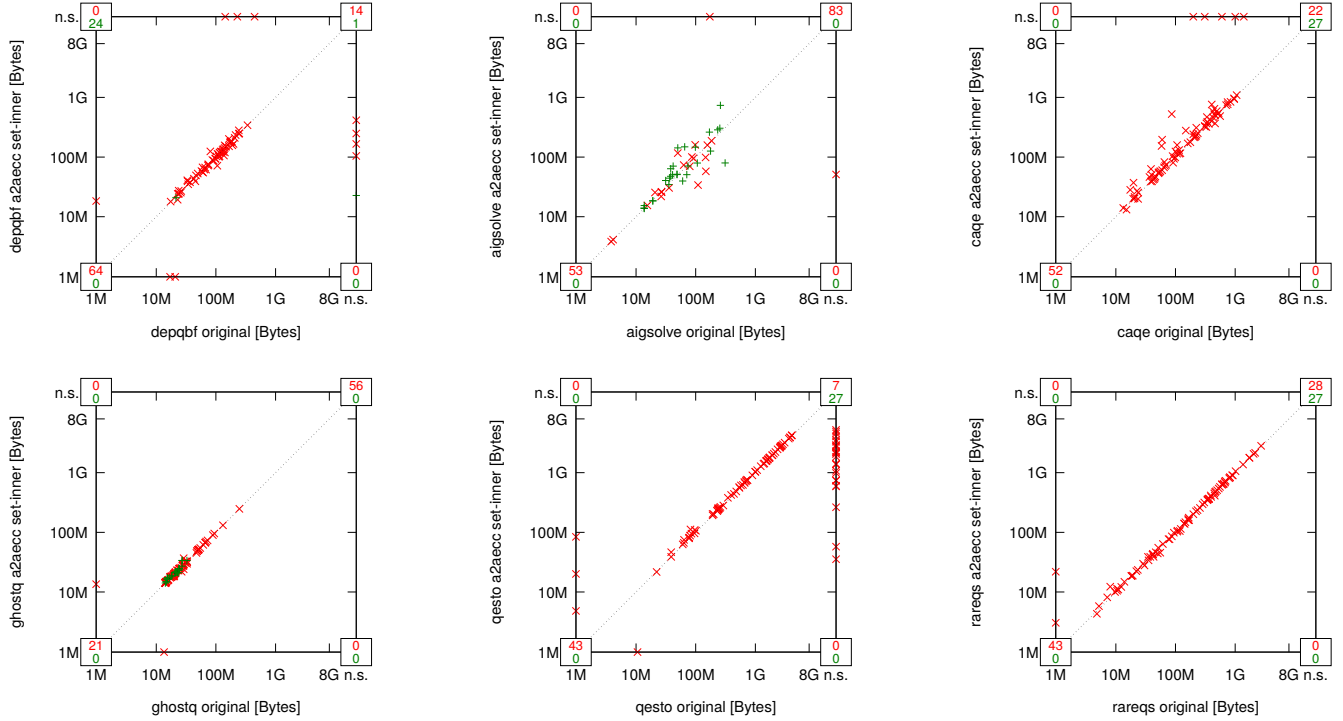


Fig. 1516: Suite Herbstritt ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

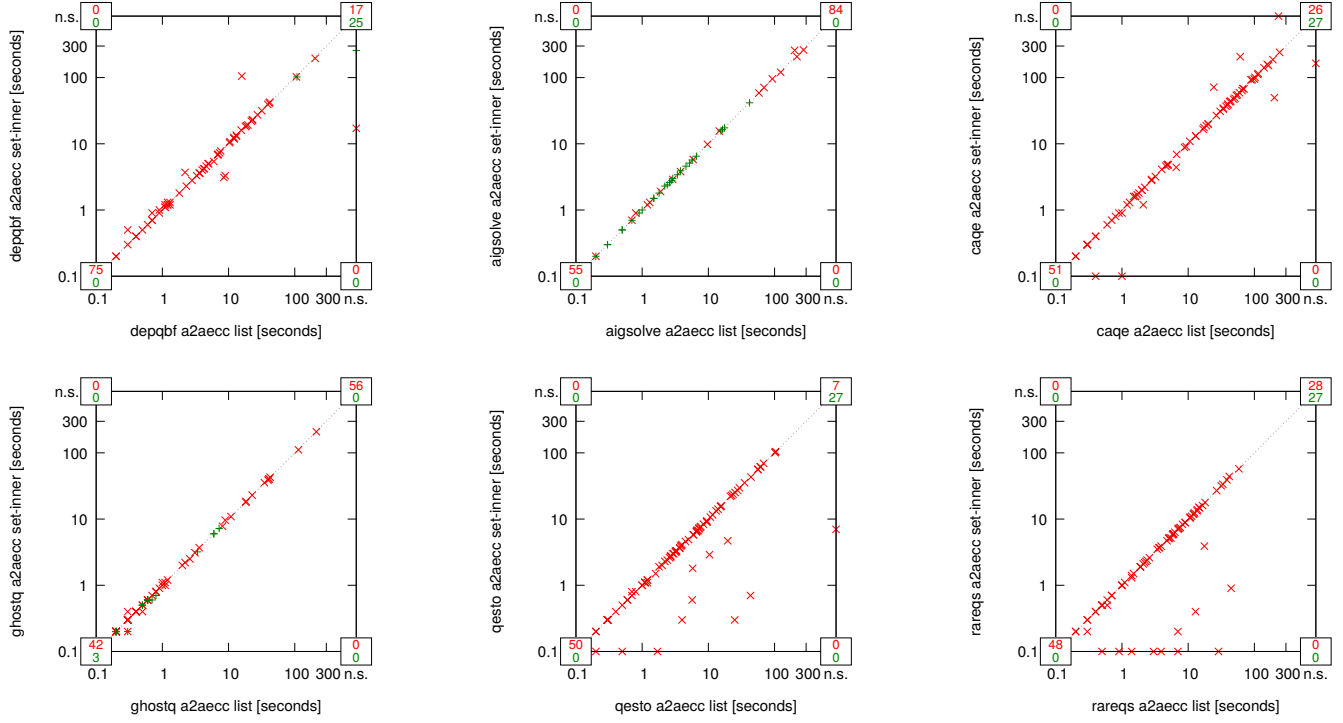


Fig. 1517: Suite Herbstritt ($n = 194$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

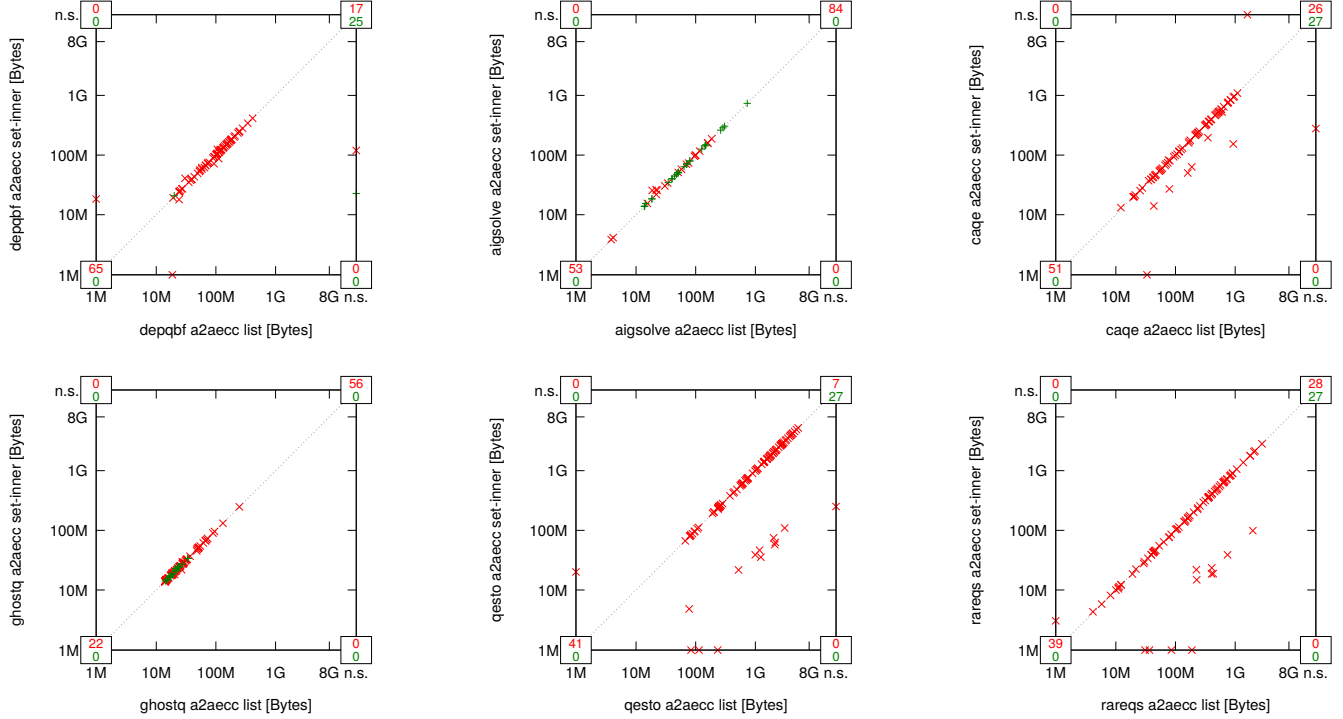


Fig. 1518: Suite Herbstritt ($n = 194$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

16) *Interian* ($n = 193$):

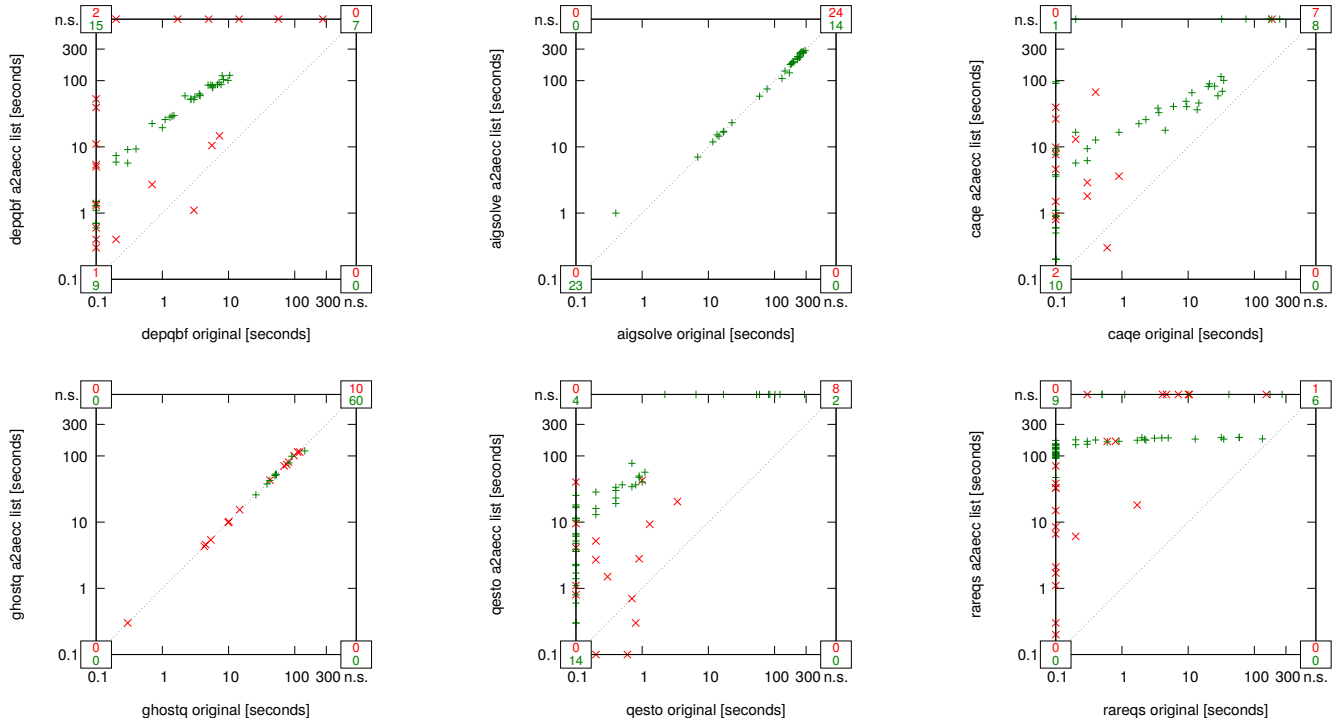


Fig. 1519: Suite *Interian* ($n = 193$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

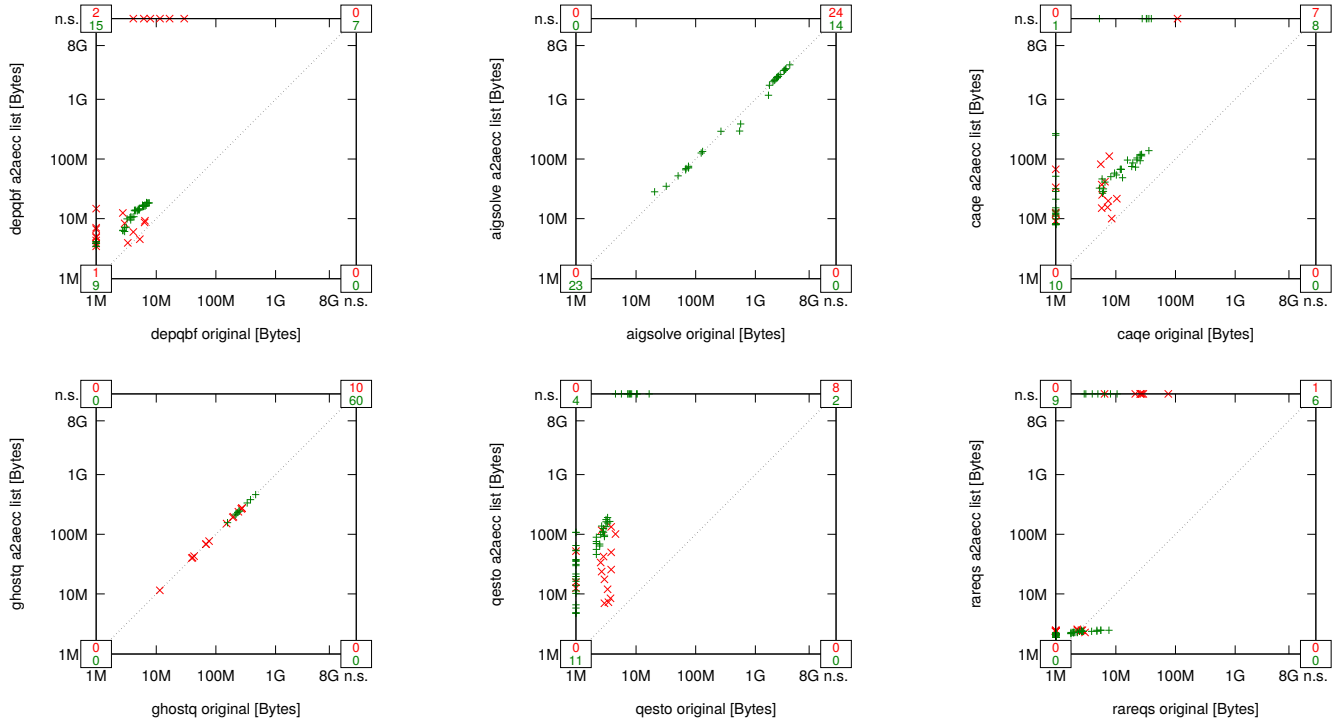


Fig. 1520: Suite *Interian* ($n = 193$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

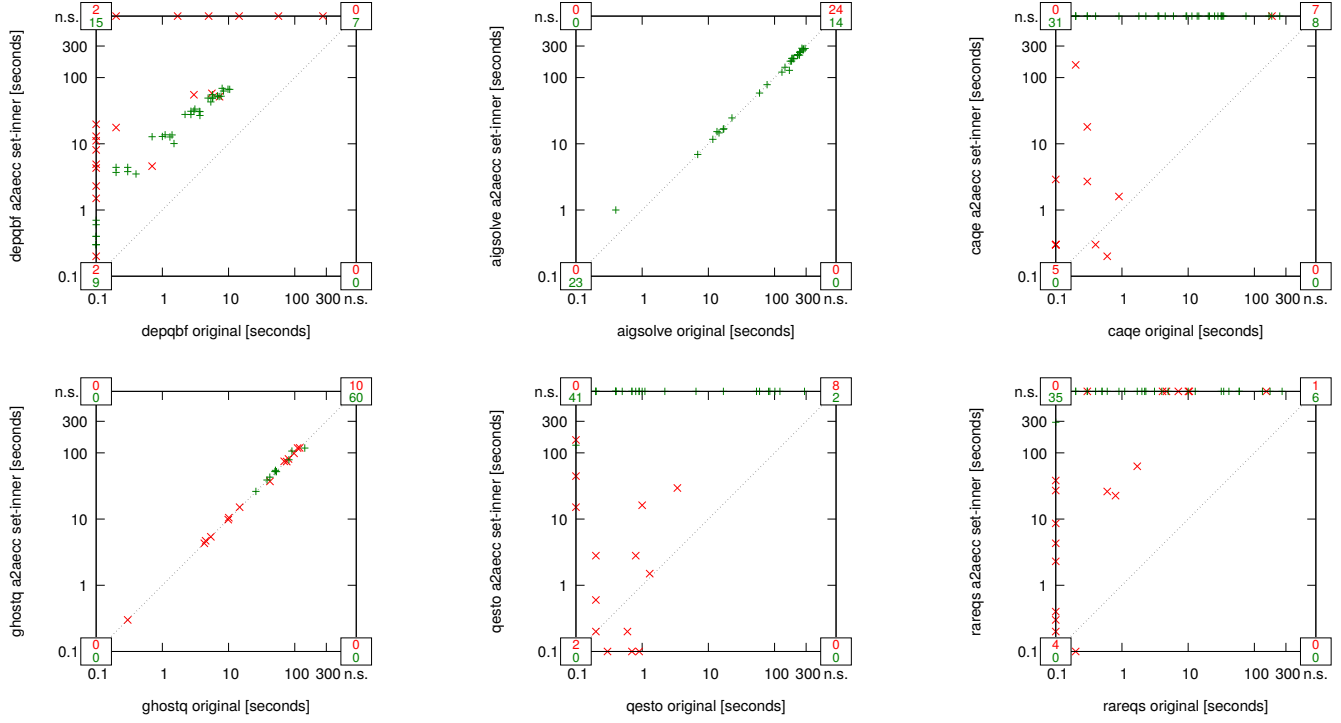


Fig. 1521: Suite Interian ($n = 193$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

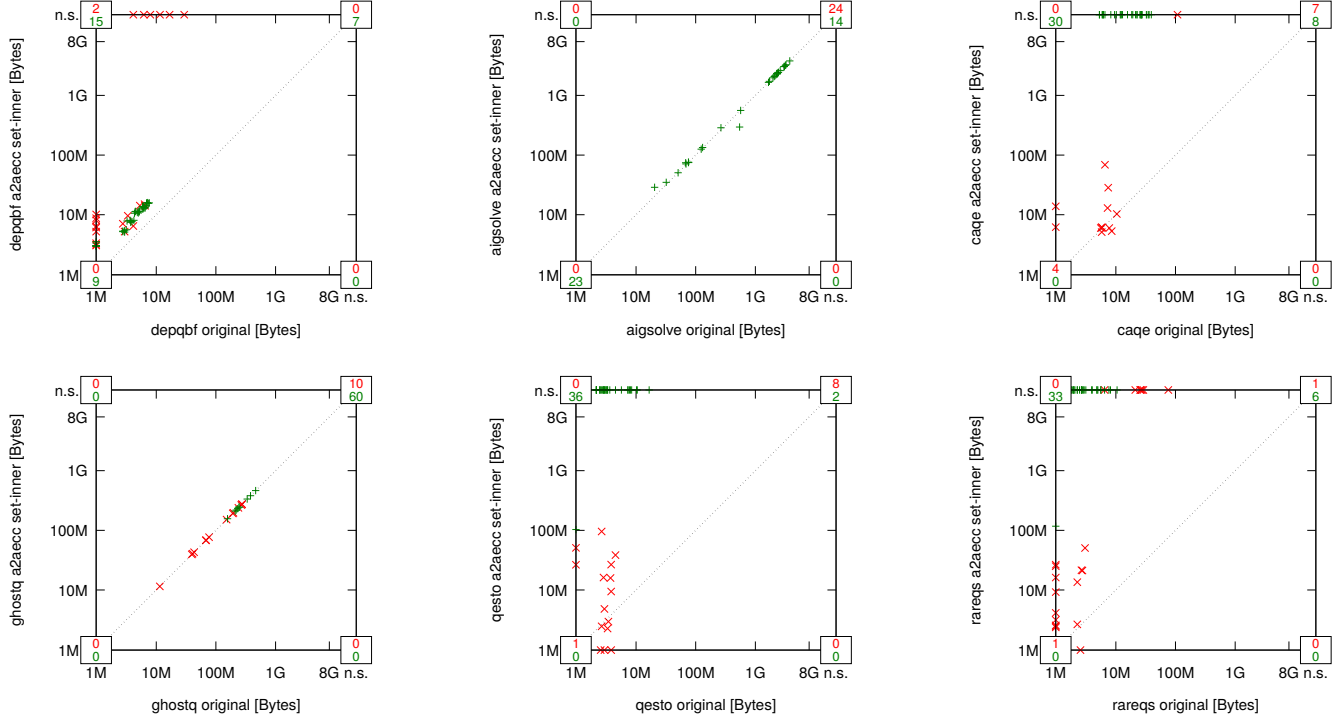


Fig. 1522: Suite Interian ($n = 193$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

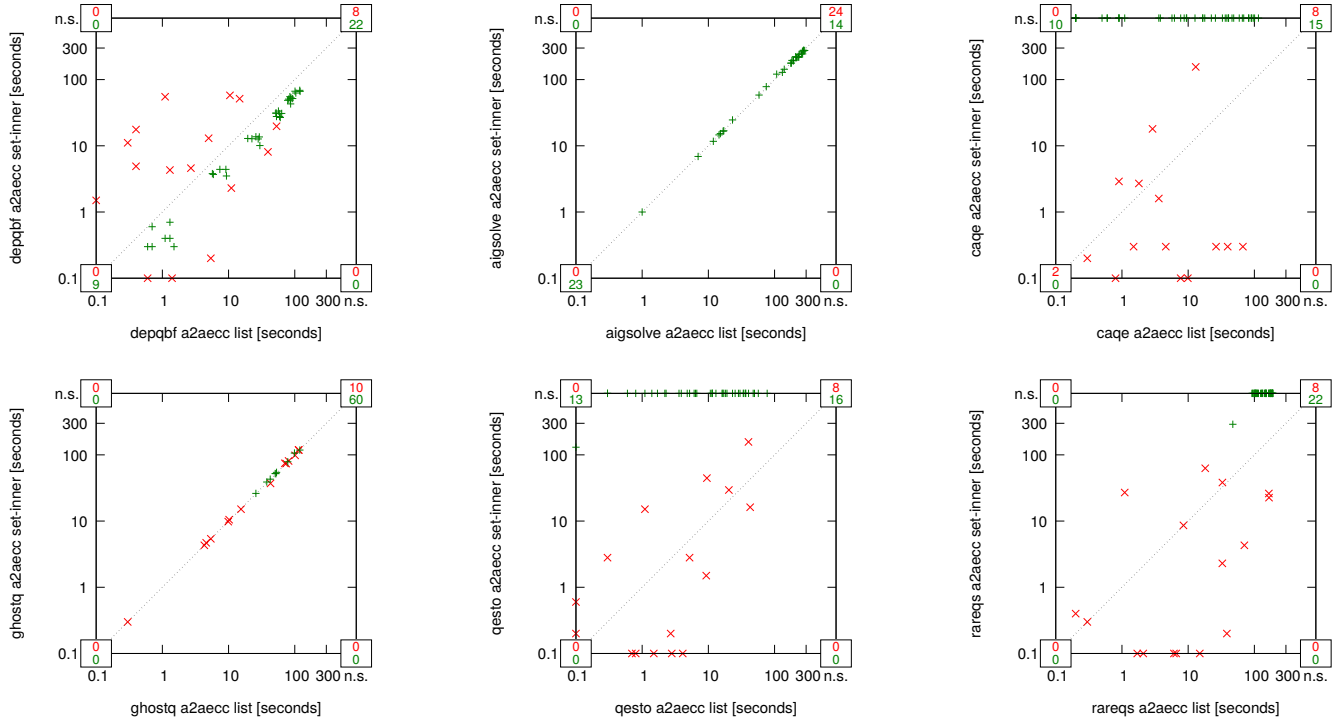


Fig. 1523: Suite Interian ($n = 193$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

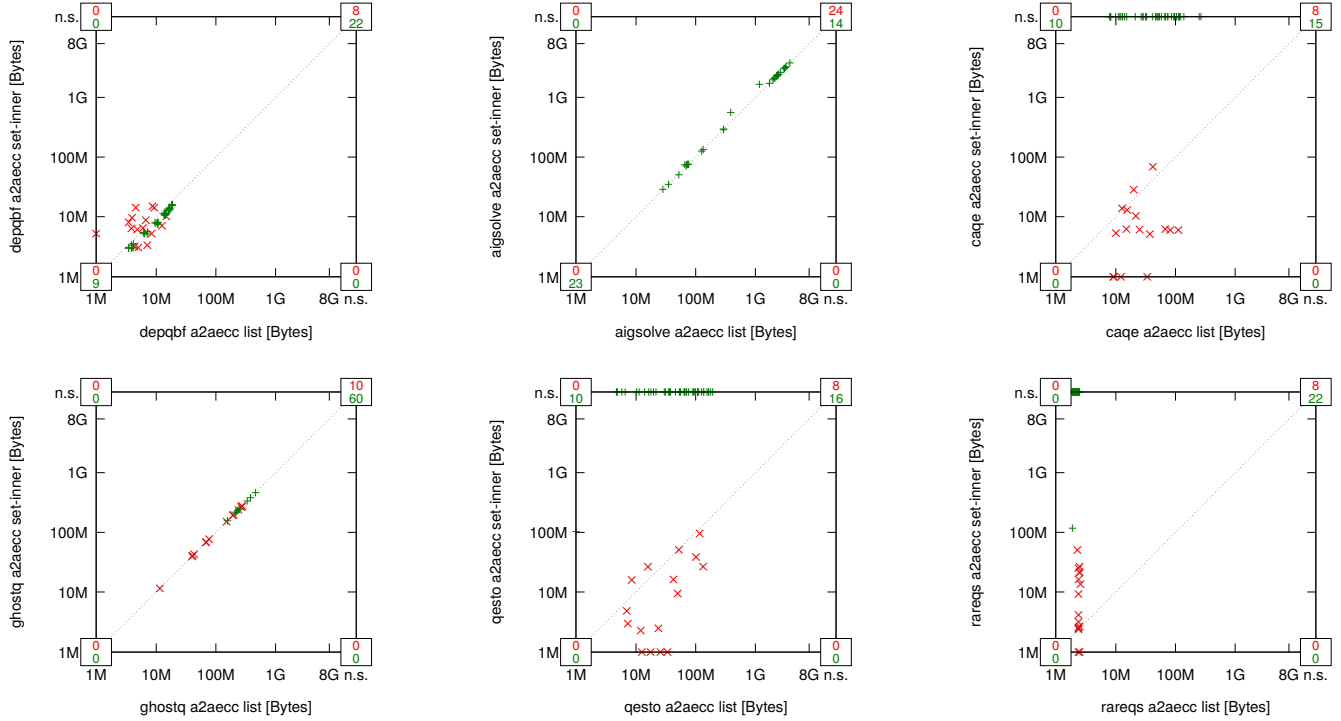


Fig. 1524: Suite Interian ($n = 193$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

17) Jordan-Kaiser ($n = 194$):

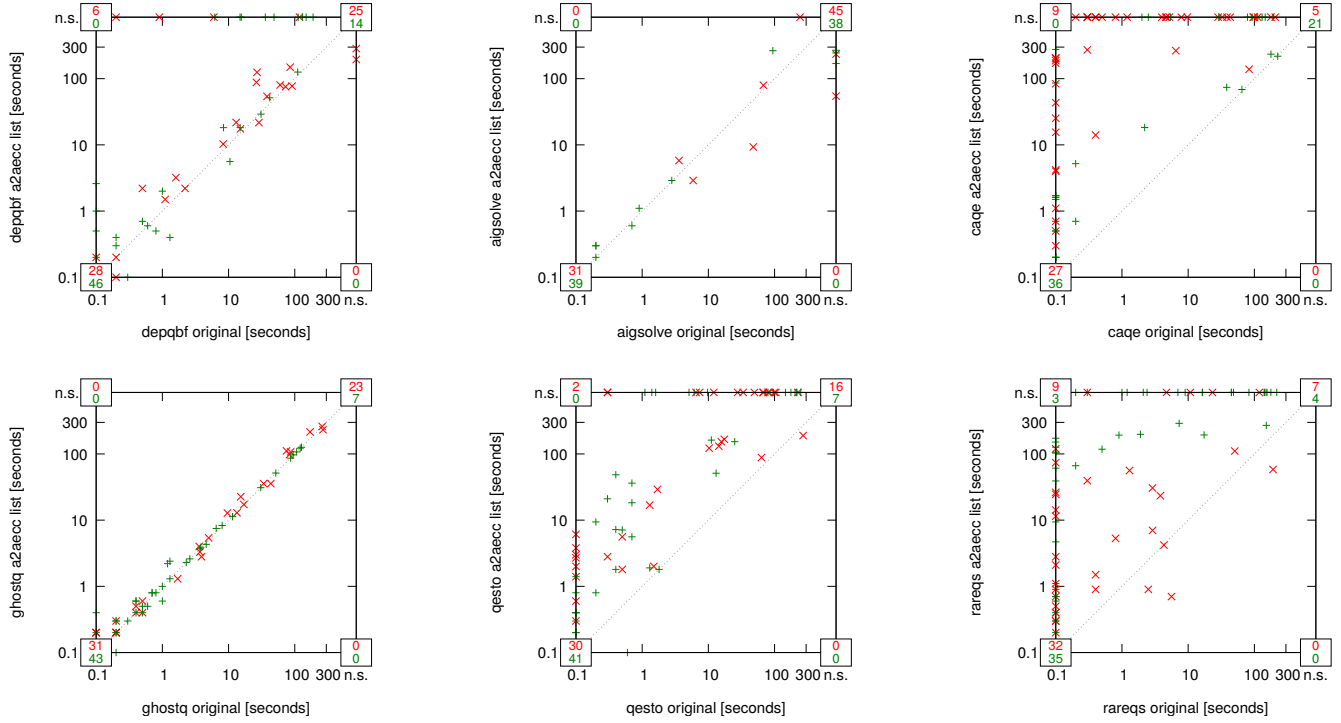


Fig. 1525: Suite Jordan-Kaiser ($n = 194$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

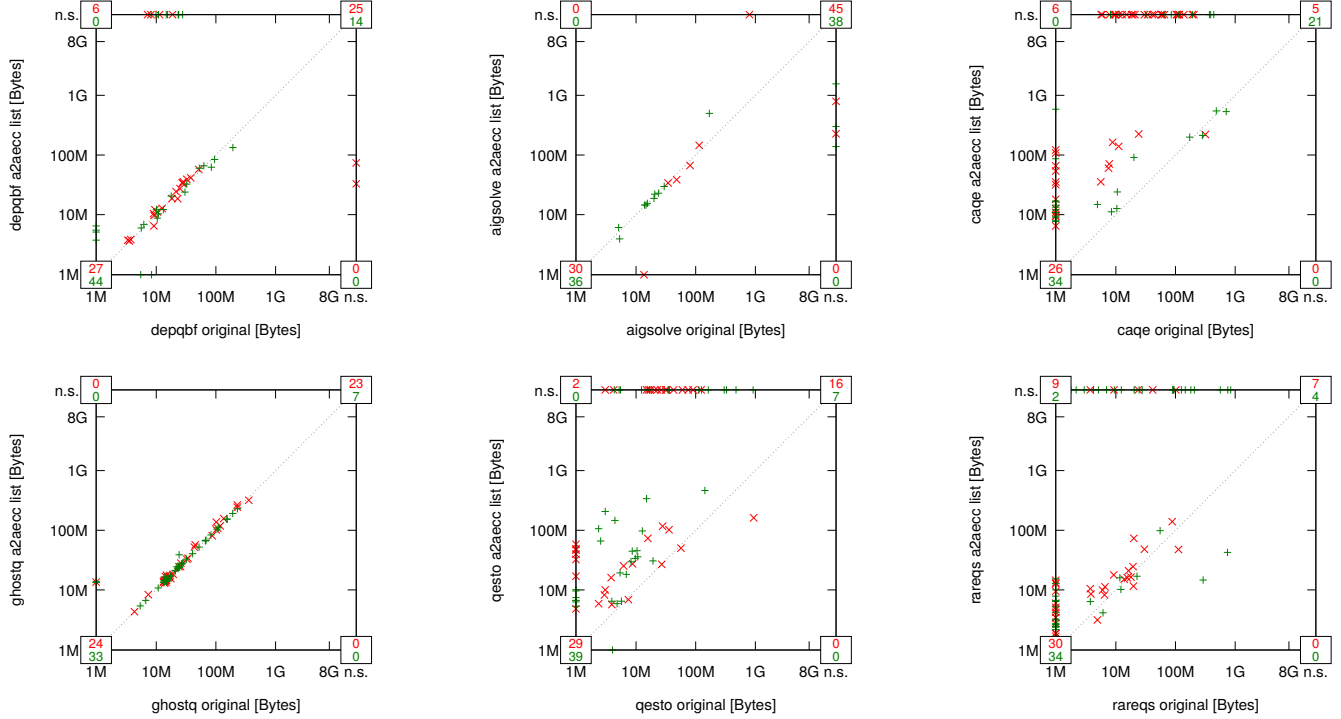


Fig. 1526: Suite Jordan-Kaiser ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

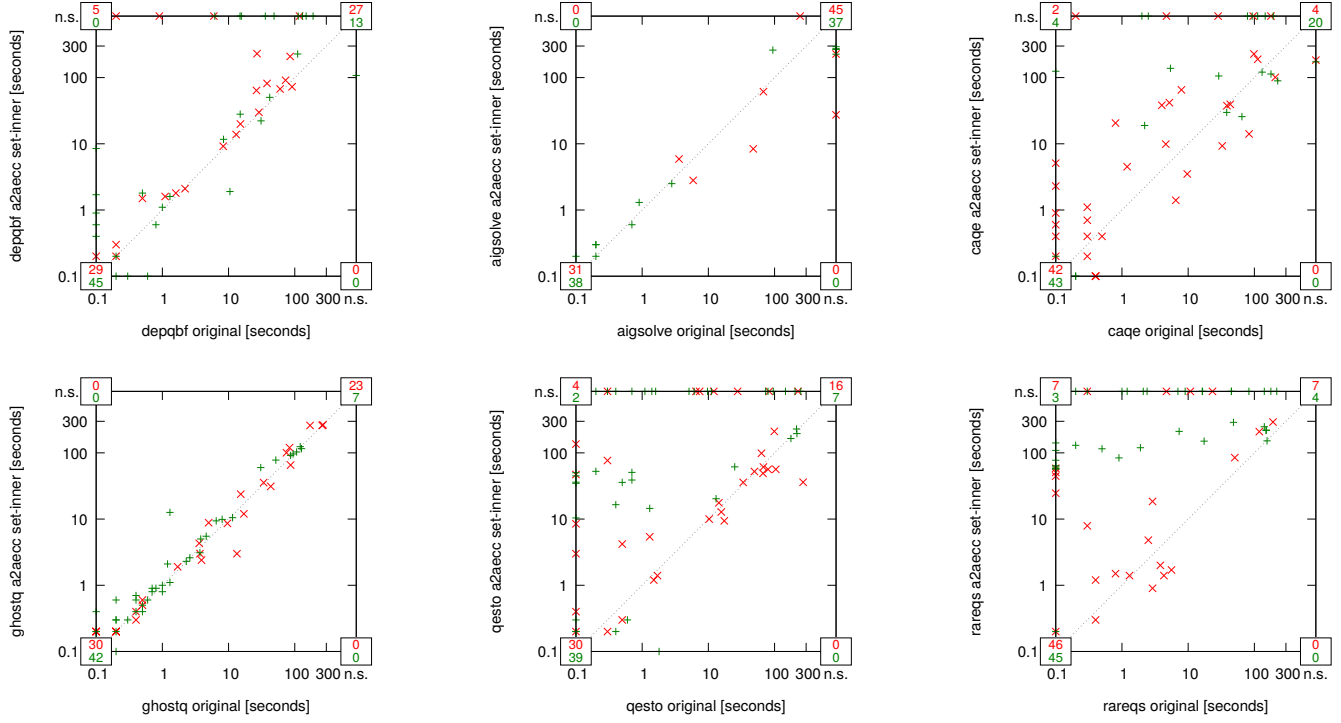


Fig. 1527: Suite Jordan-Kaiser ($n = 194$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

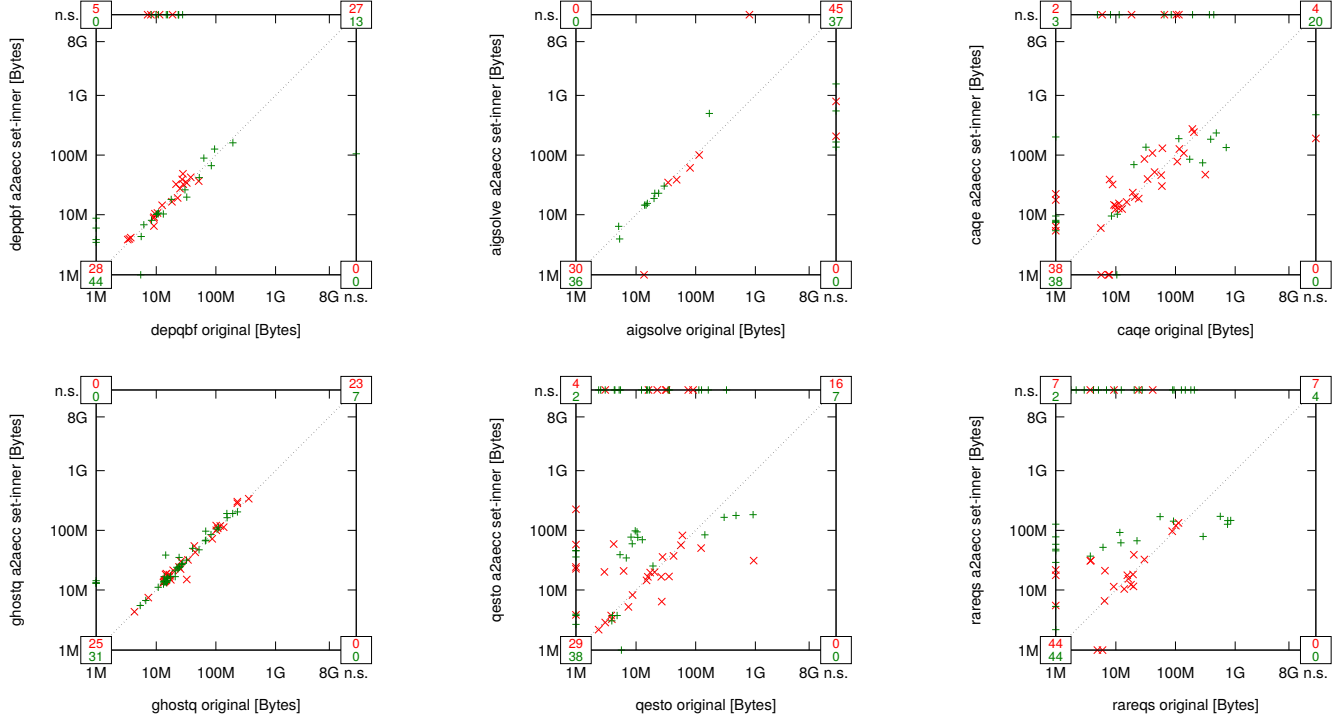


Fig. 1528: Suite Jordan-Kaiser ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

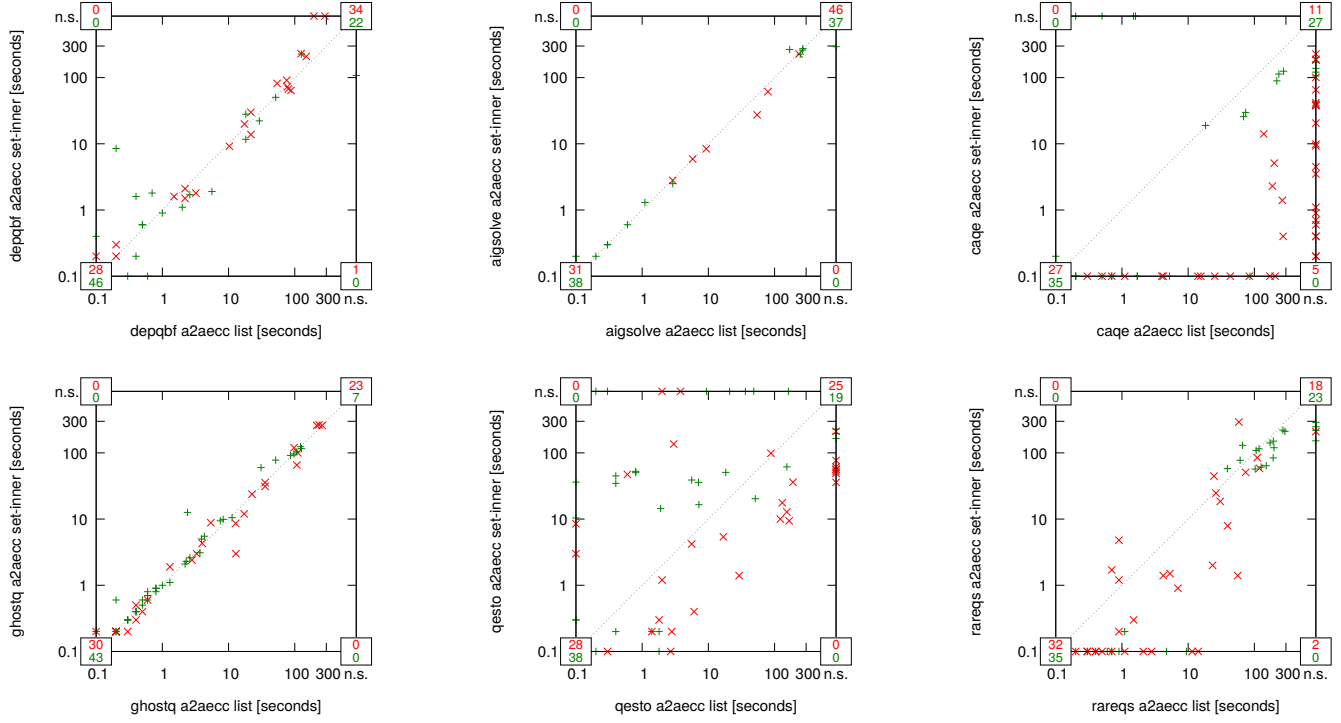


Fig. 1529: Suite Jordan-Kaiser ($n = 194$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

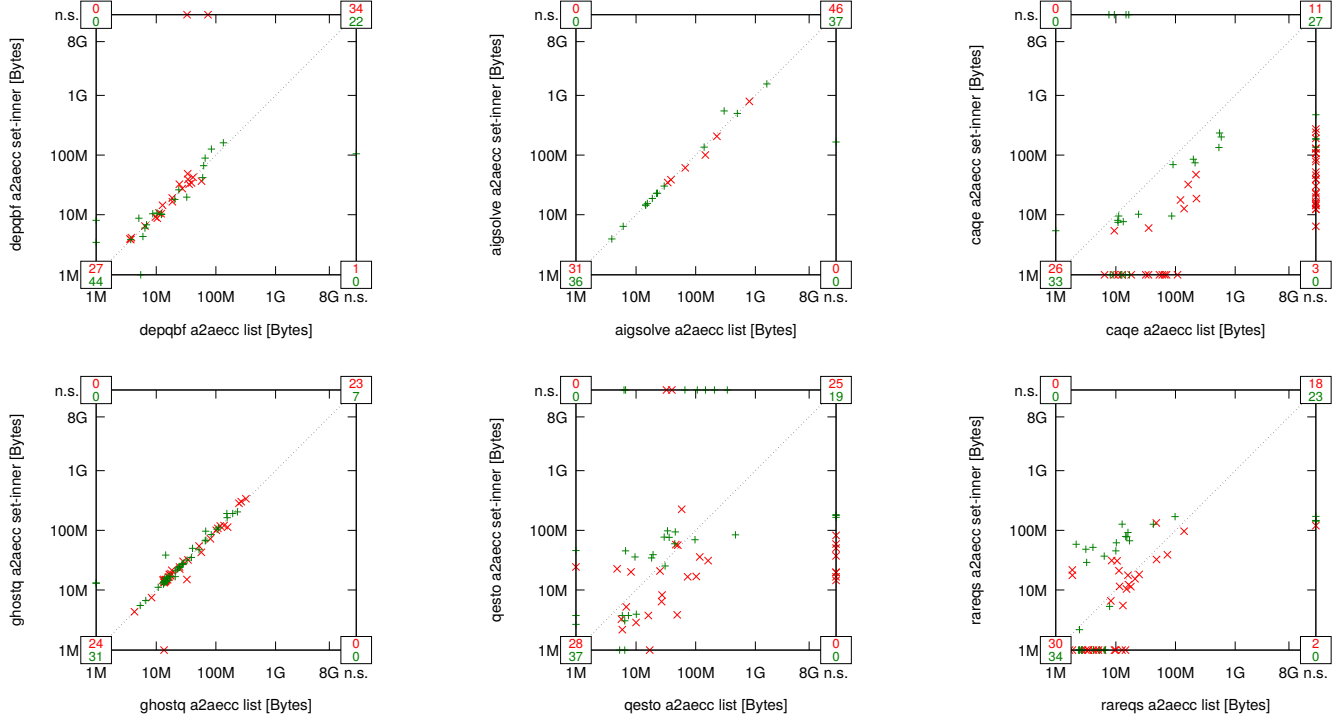


Fig. 1530: Suite Jordan-Kaiser ($n = 194$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

18) Katz ($n = 20$):

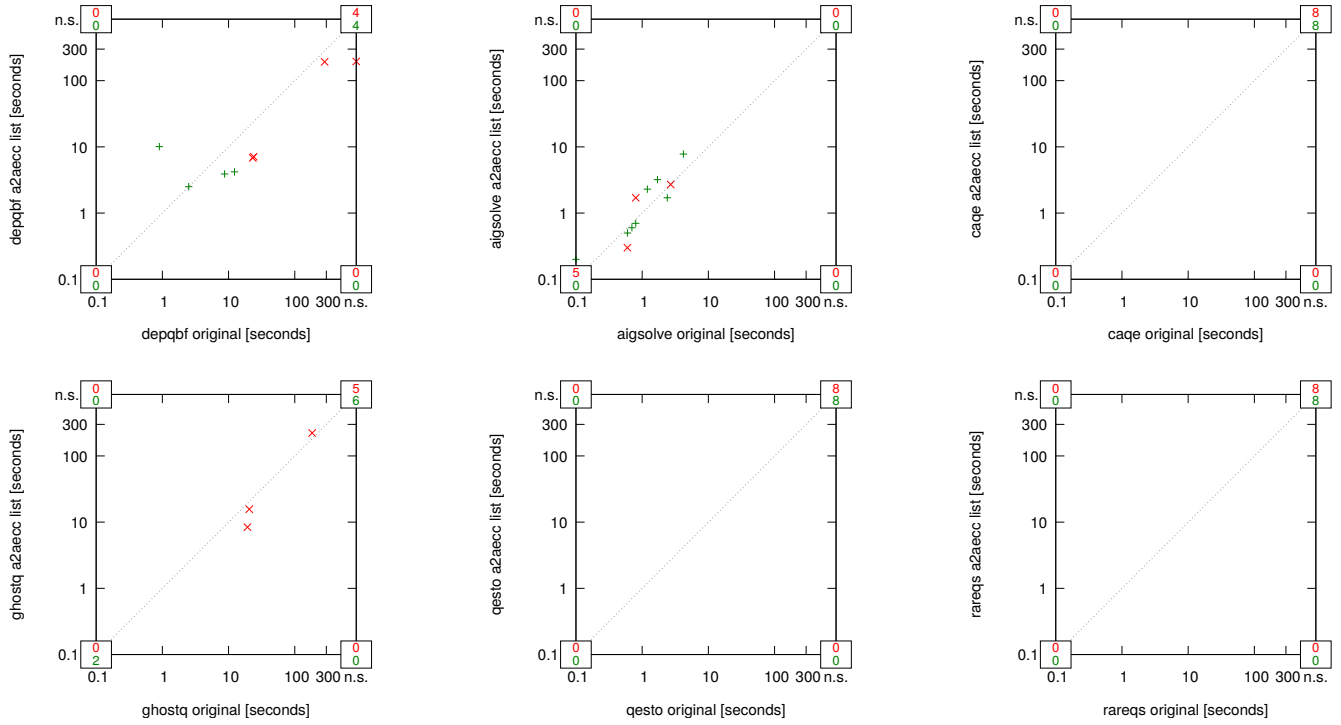


Fig. 1531: Suite Katz ($n = 20$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

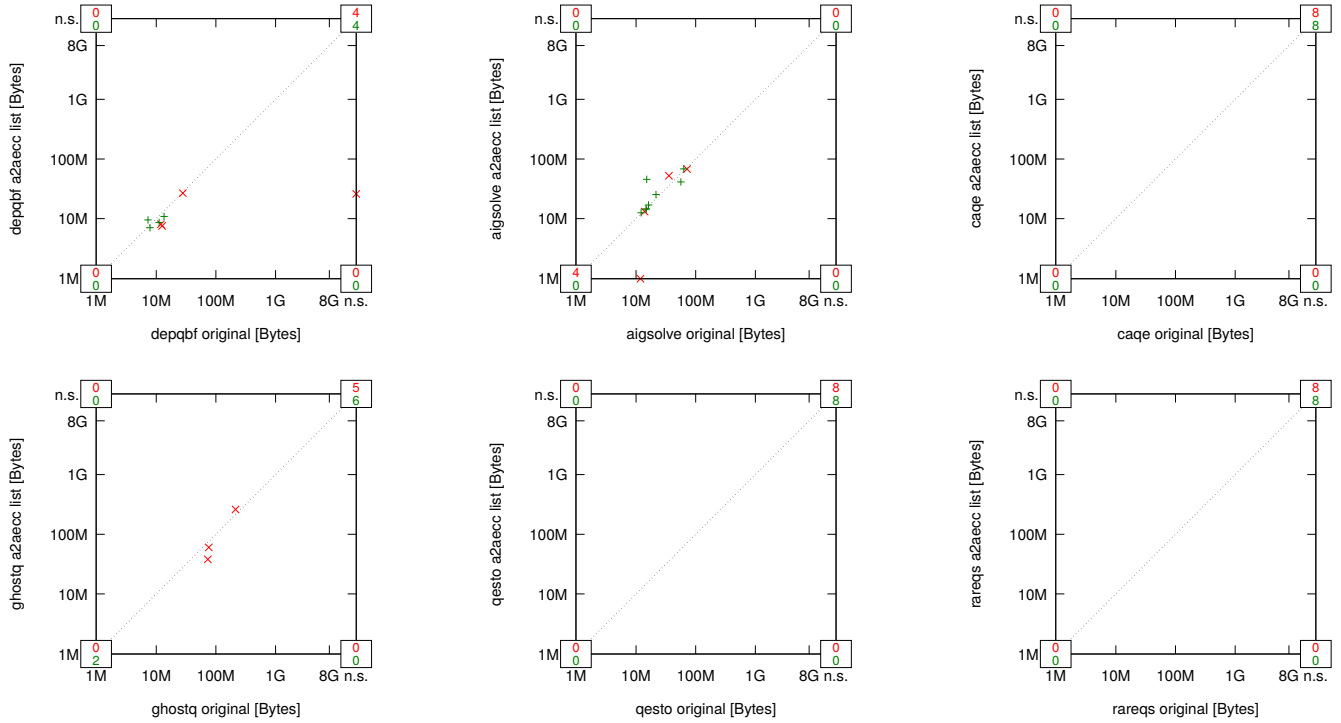


Fig. 1532: Suite Katz ($n = 20$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

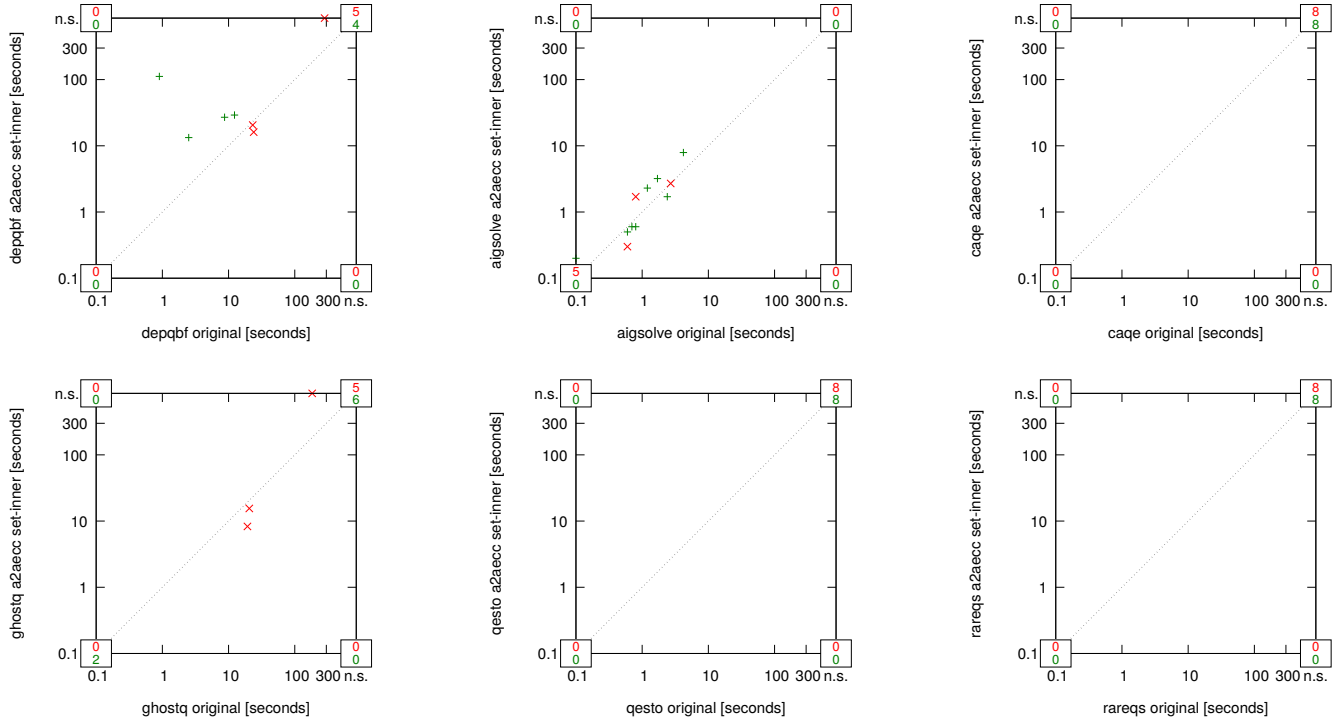


Fig. 1533: Suite Katz ($n = 20$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

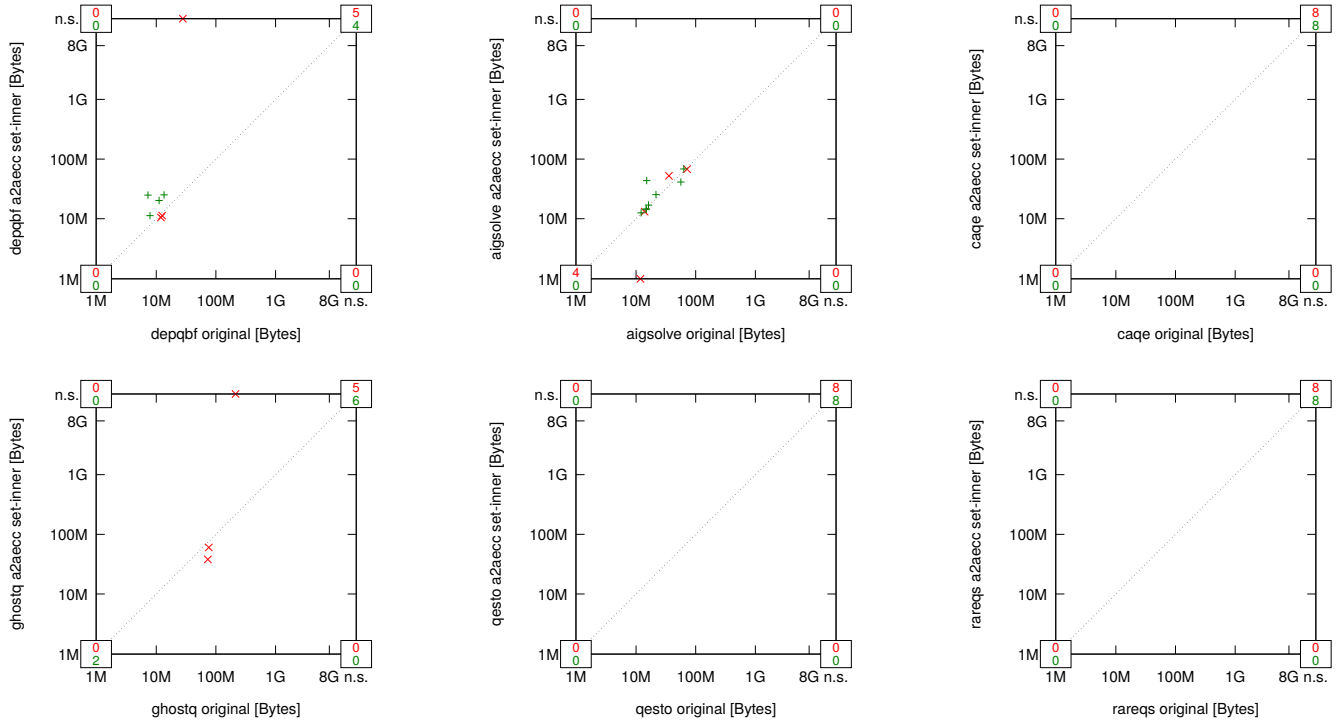


Fig. 1534: Suite Katz ($n = 20$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

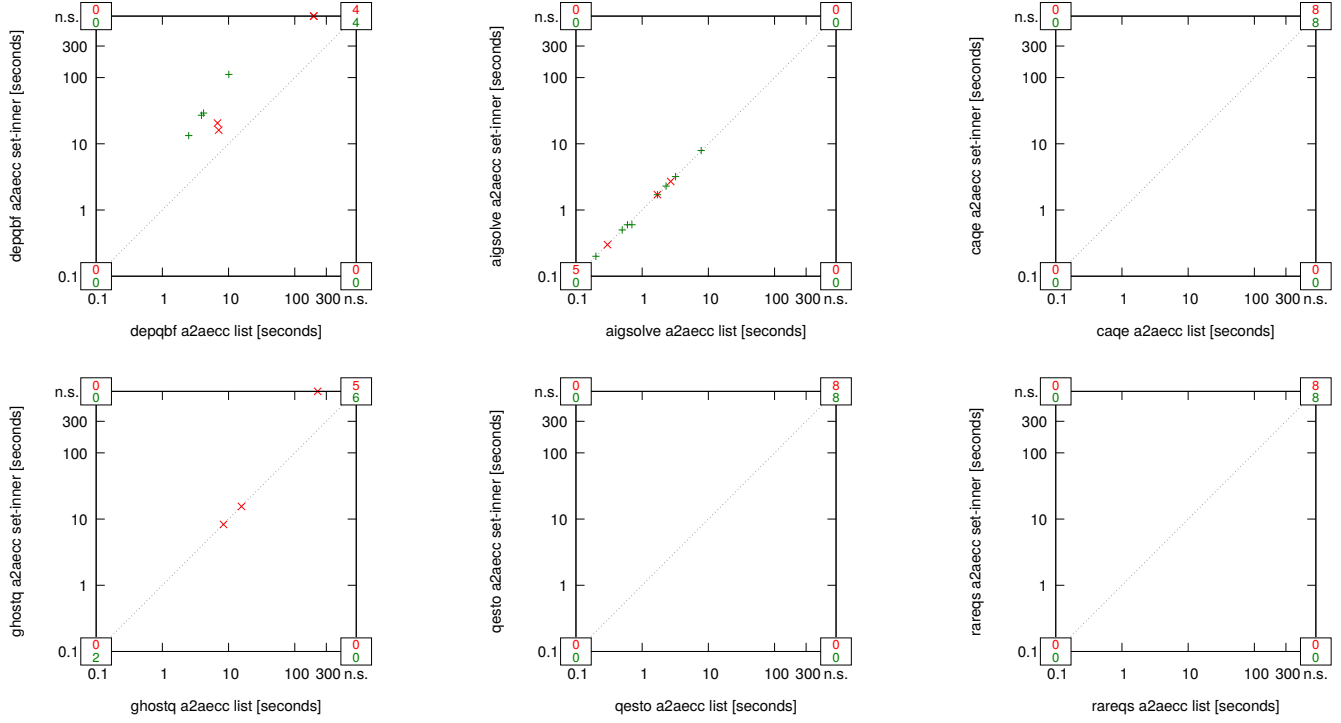


Fig. 1535: Suite Katz ($n = 20$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

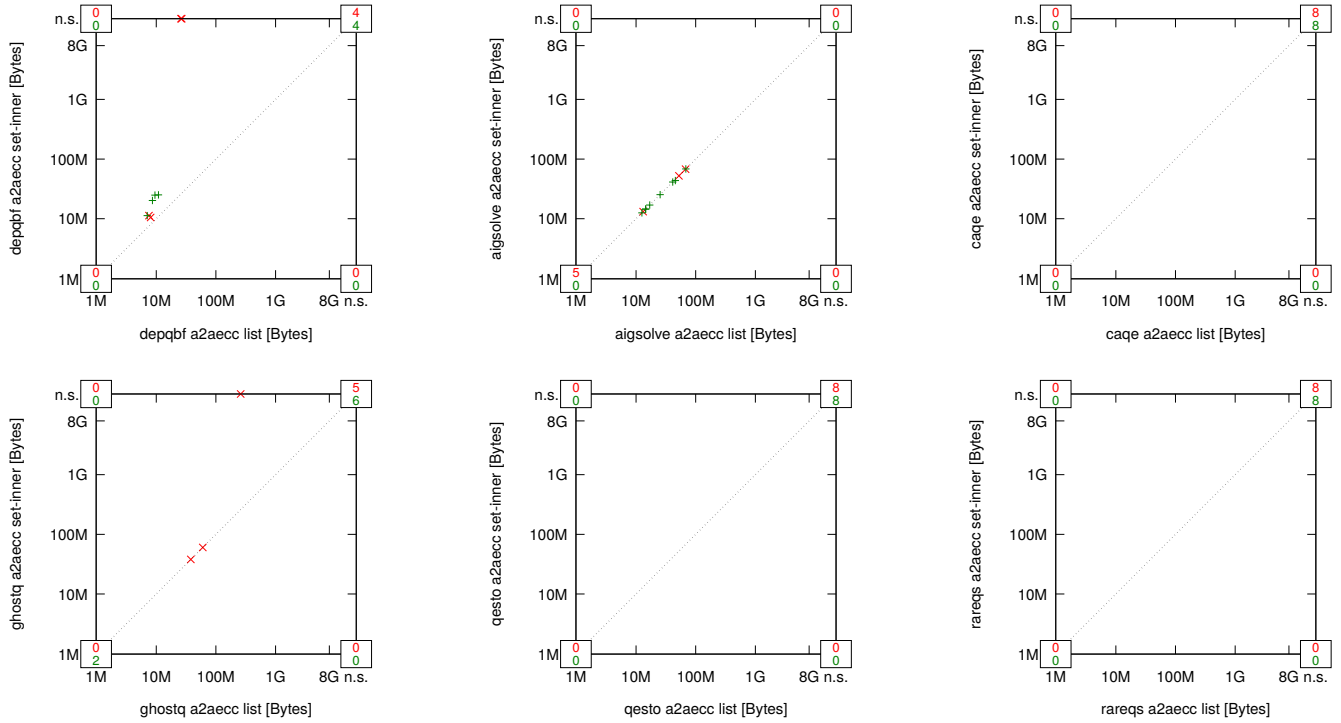


Fig. 1536: Suite Katz ($n = 20$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

19) Klieber ($n = 30$):

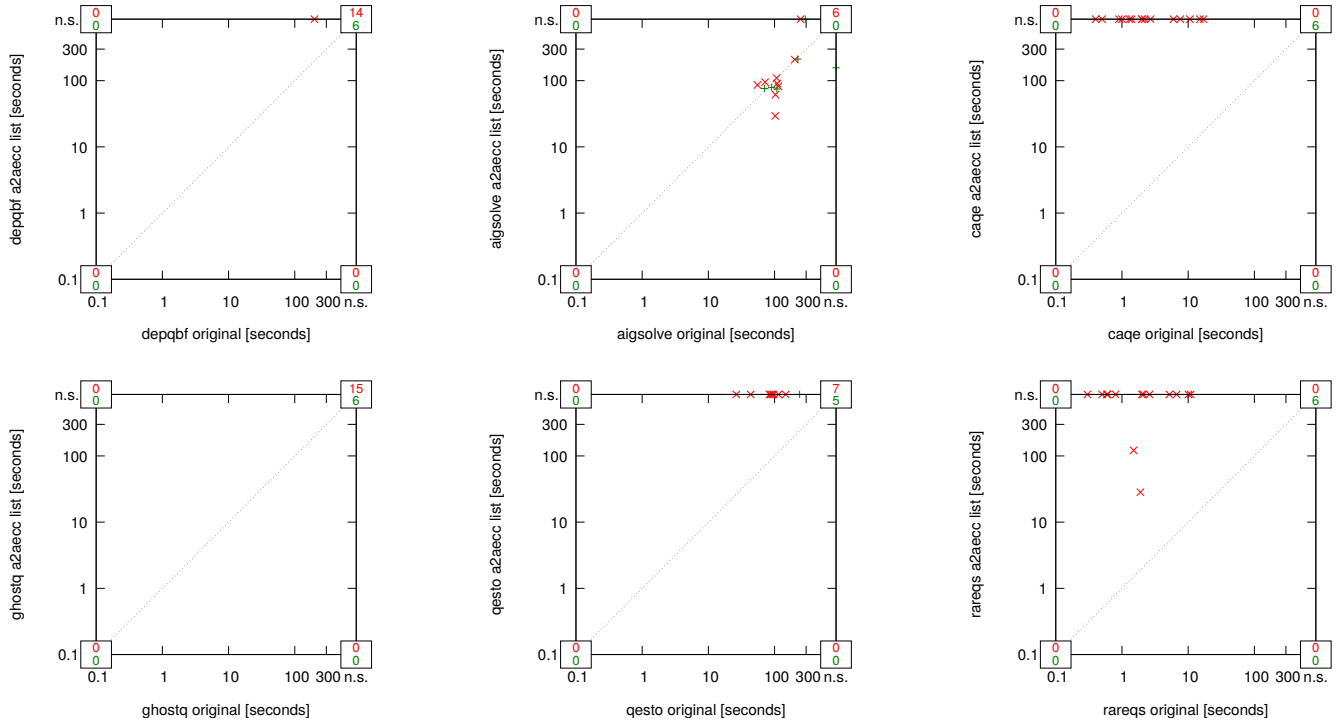


Fig. 1537: Suite Klieber ($n = 30$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

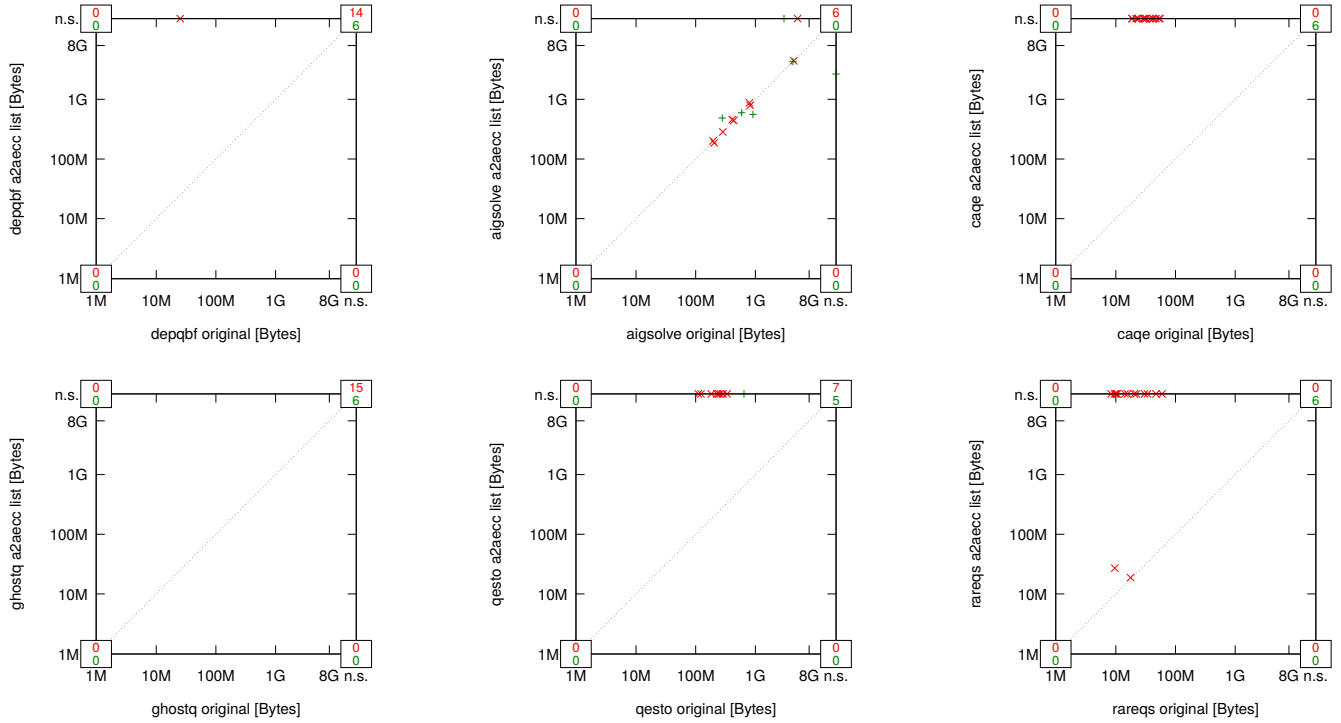


Fig. 1538: Suite Klieber ($n = 30$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

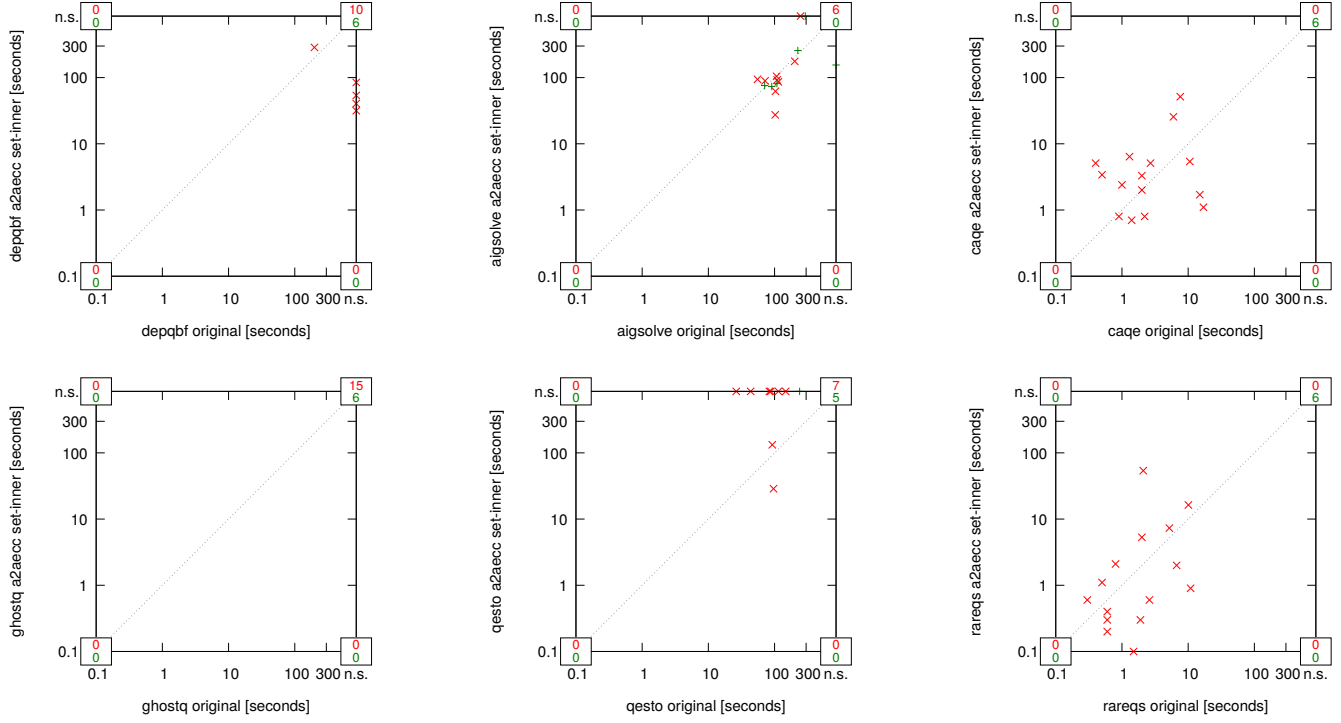


Fig. 1539: Suite Klieber ($n = 30$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

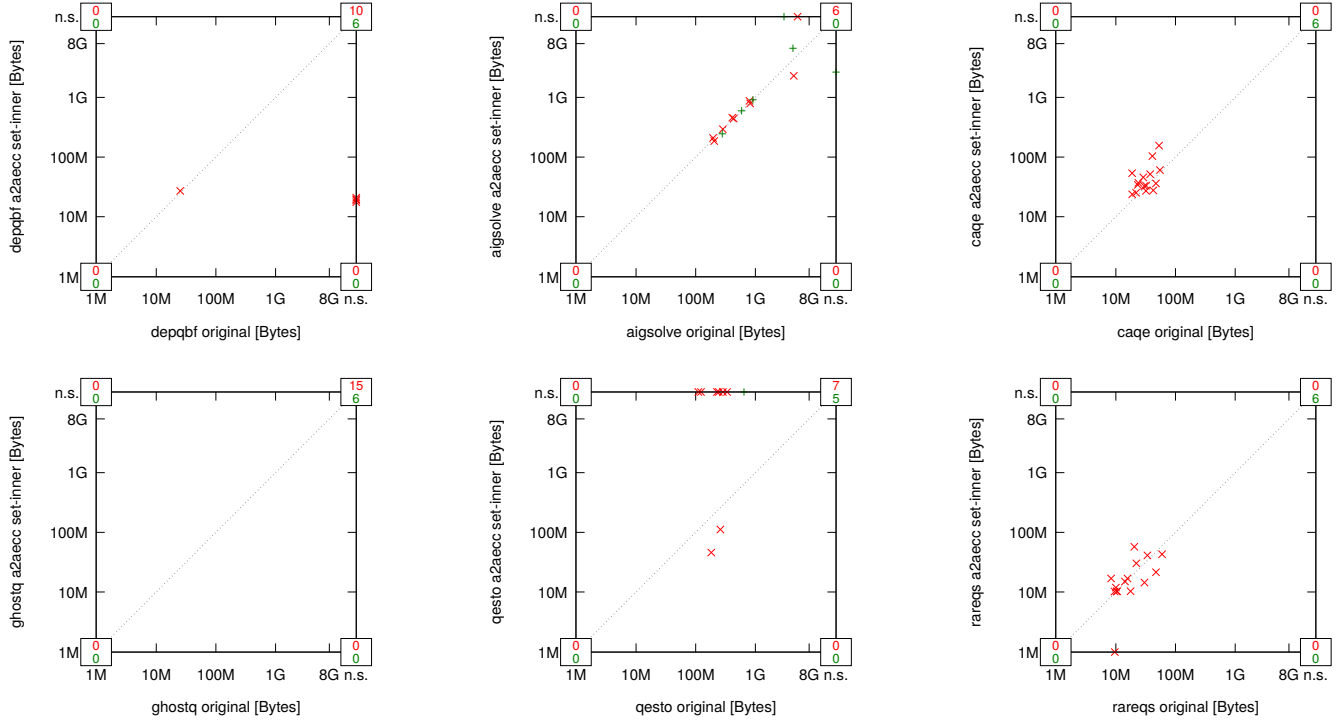


Fig. 1540: Suite Klieber ($n = 30$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

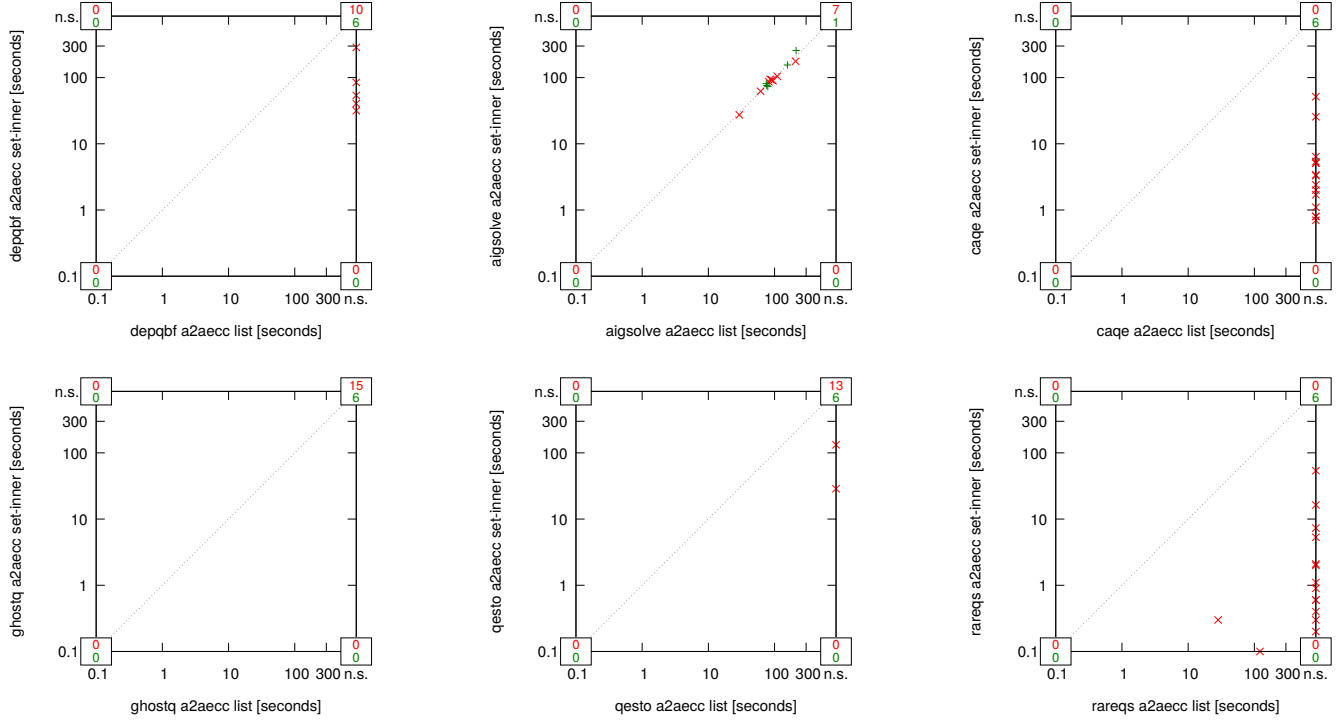


Fig. 1541: Suite Klieber ($n = 30$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

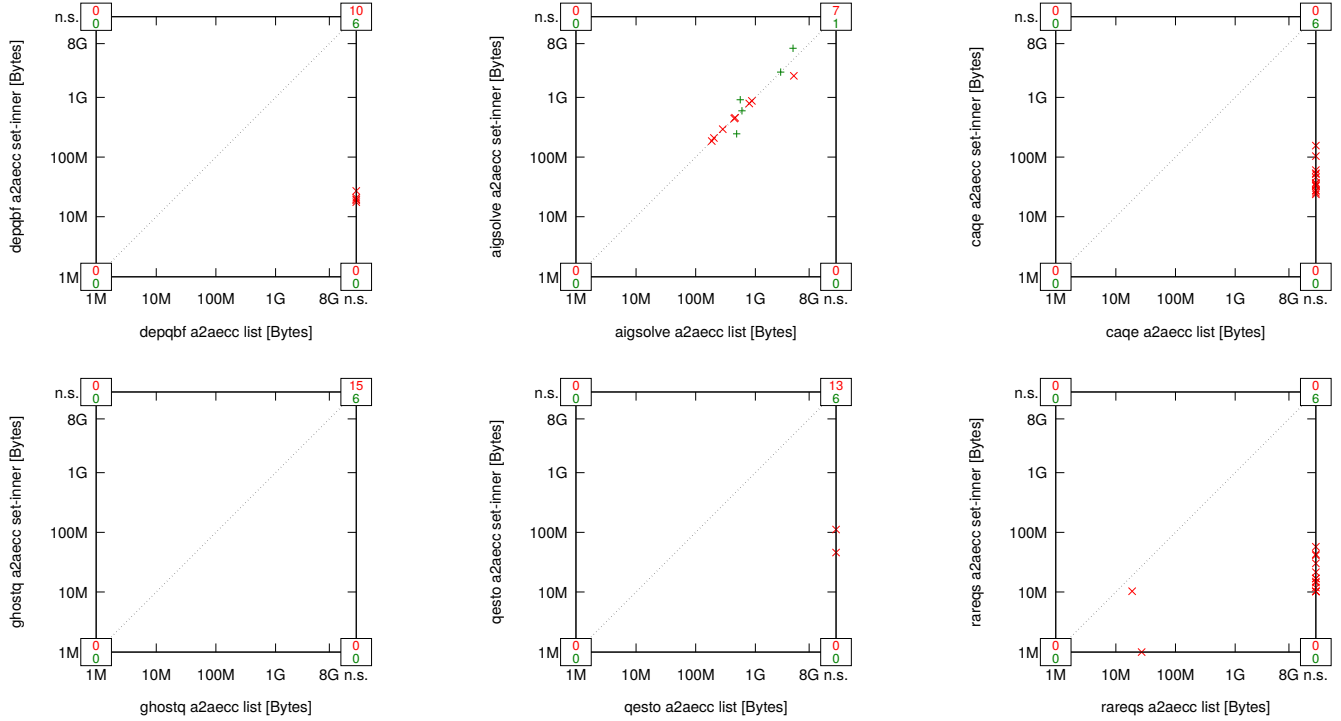


Fig. 1542: Suite Klieber ($n = 30$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

20) Kontchakov ($n = 136$):

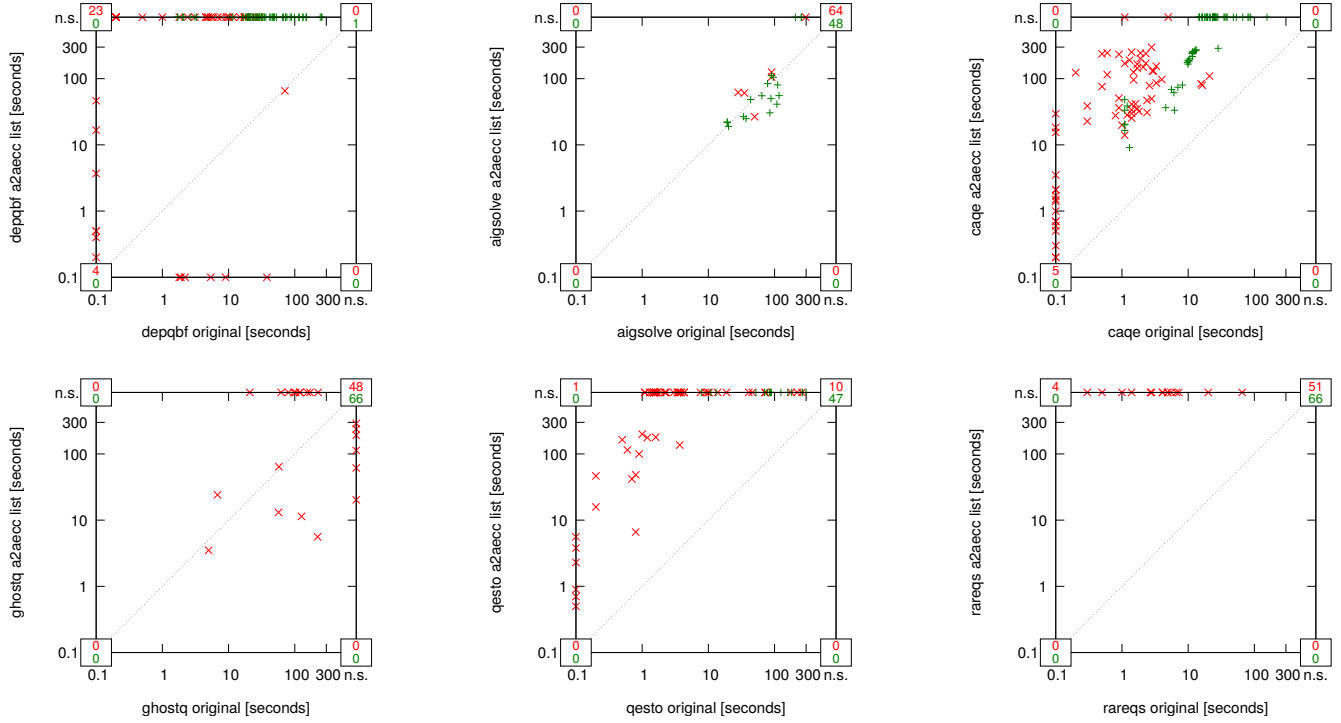


Fig. 1543: Suite Kontchakov ($n = 136$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

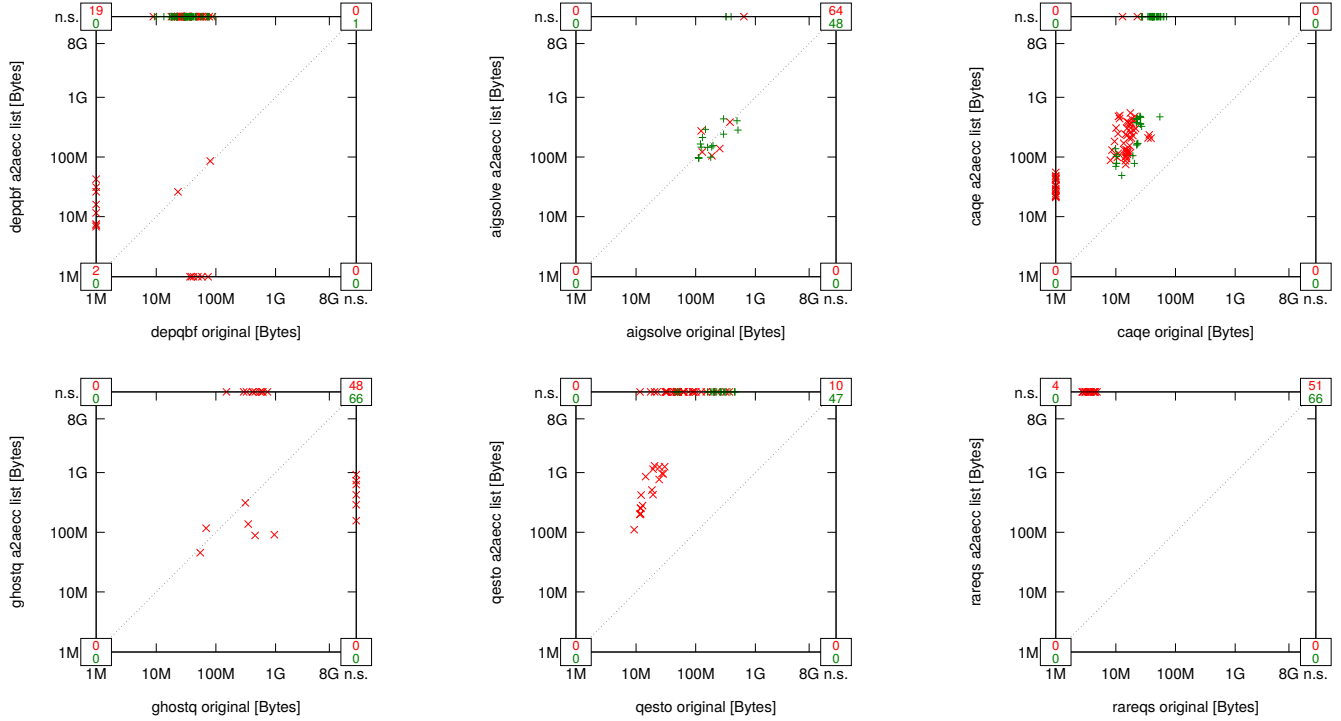


Fig. 1544: Suite Kontchakov ($n = 136$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

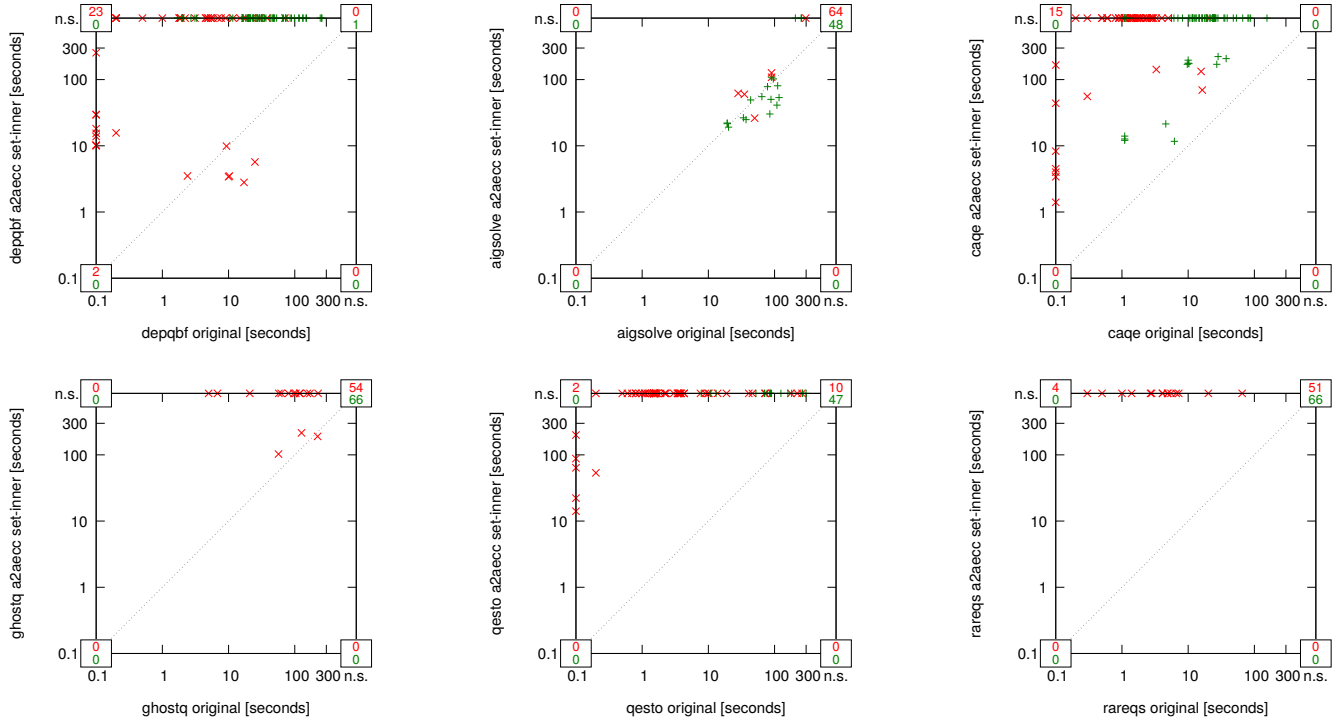


Fig. 1545: Suite Kontchakov ($n = 136$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

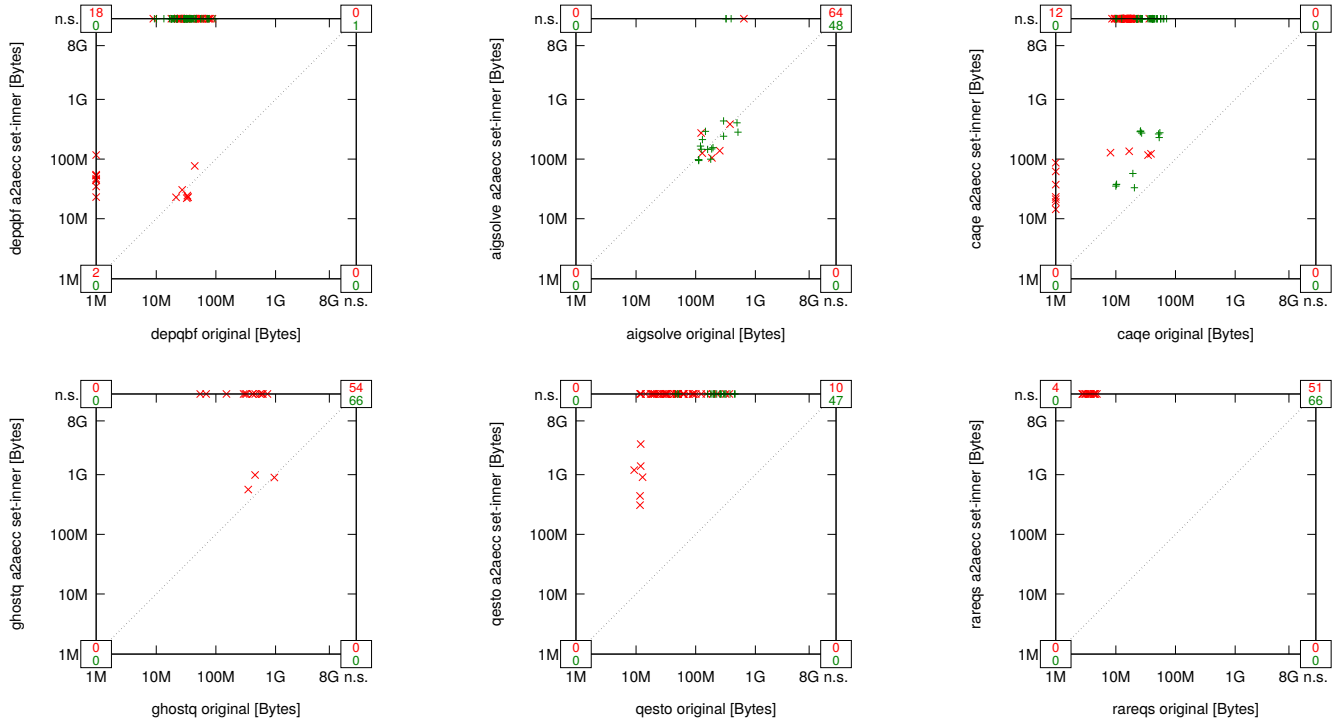


Fig. 1546: Suite Kontchakov ($n = 136$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

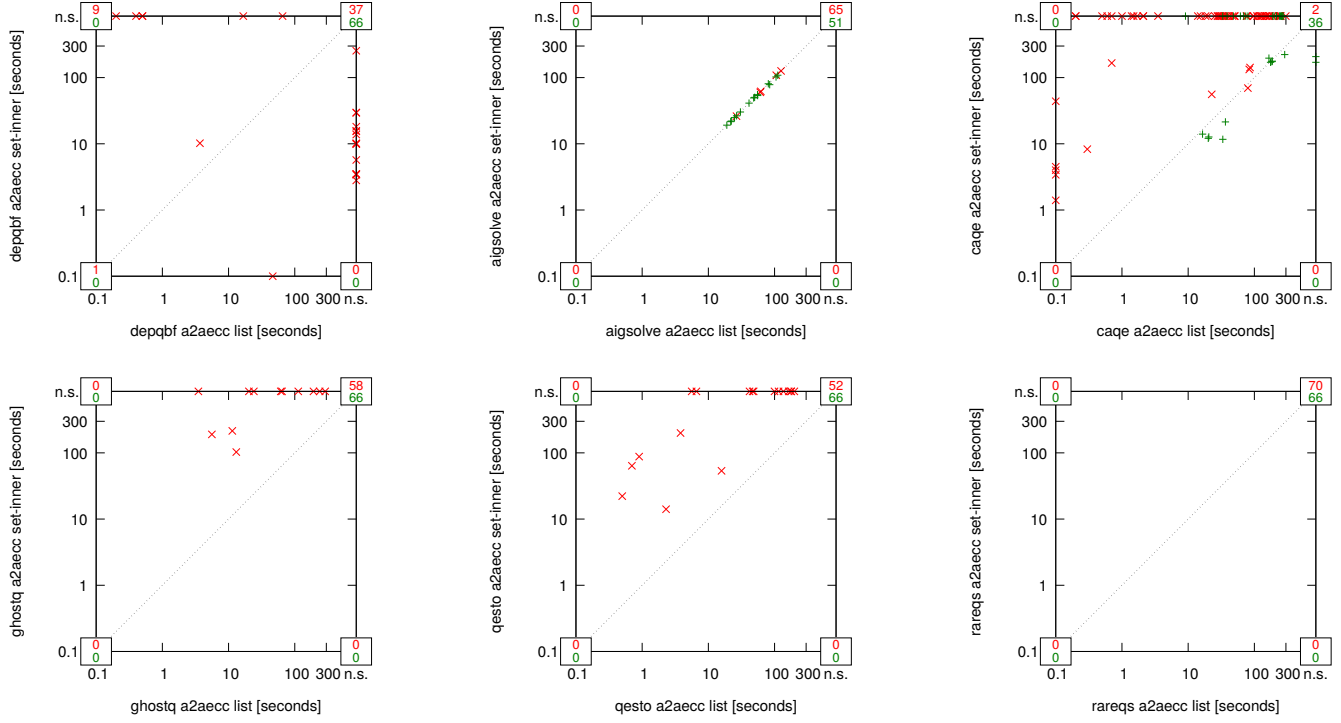


Fig. 1547: Suite Kontchakov ($n = 136$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

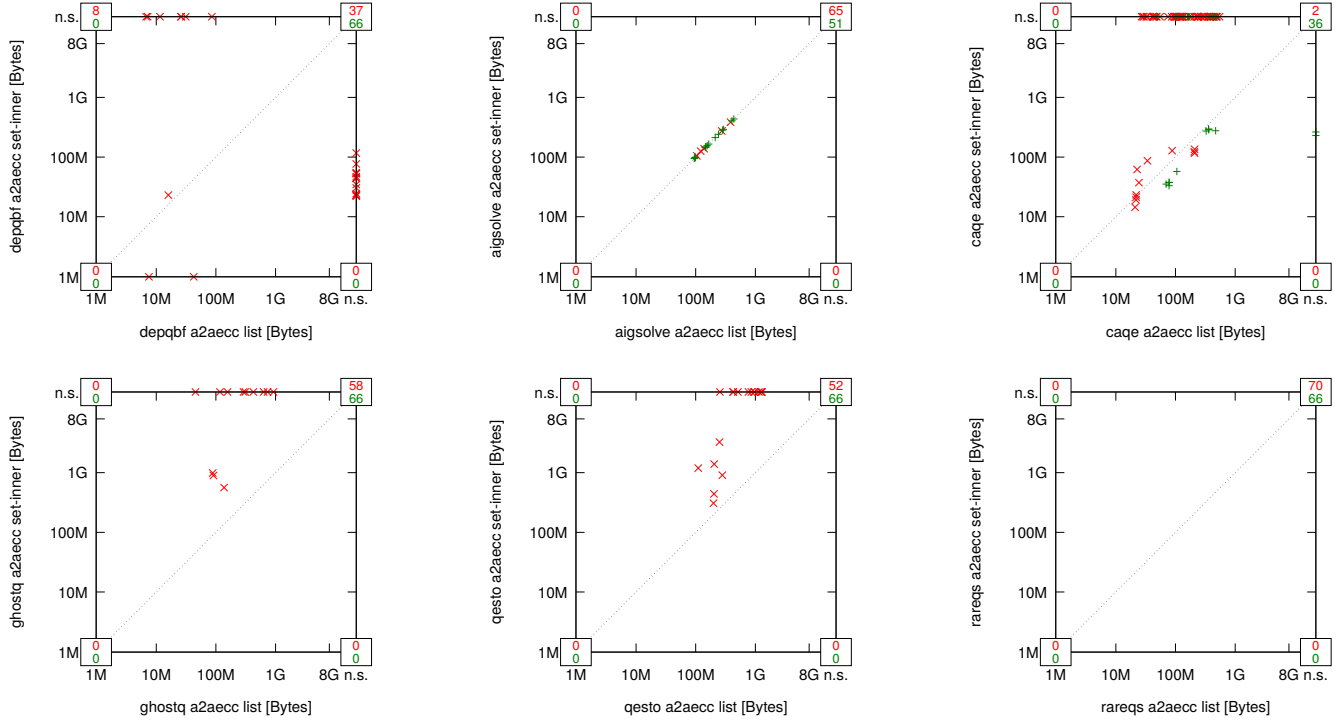


Fig. 1548: Suite Kontchakov ($n = 136$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

21) Kronegger-Pfandler-Pichler ($n = 194$):

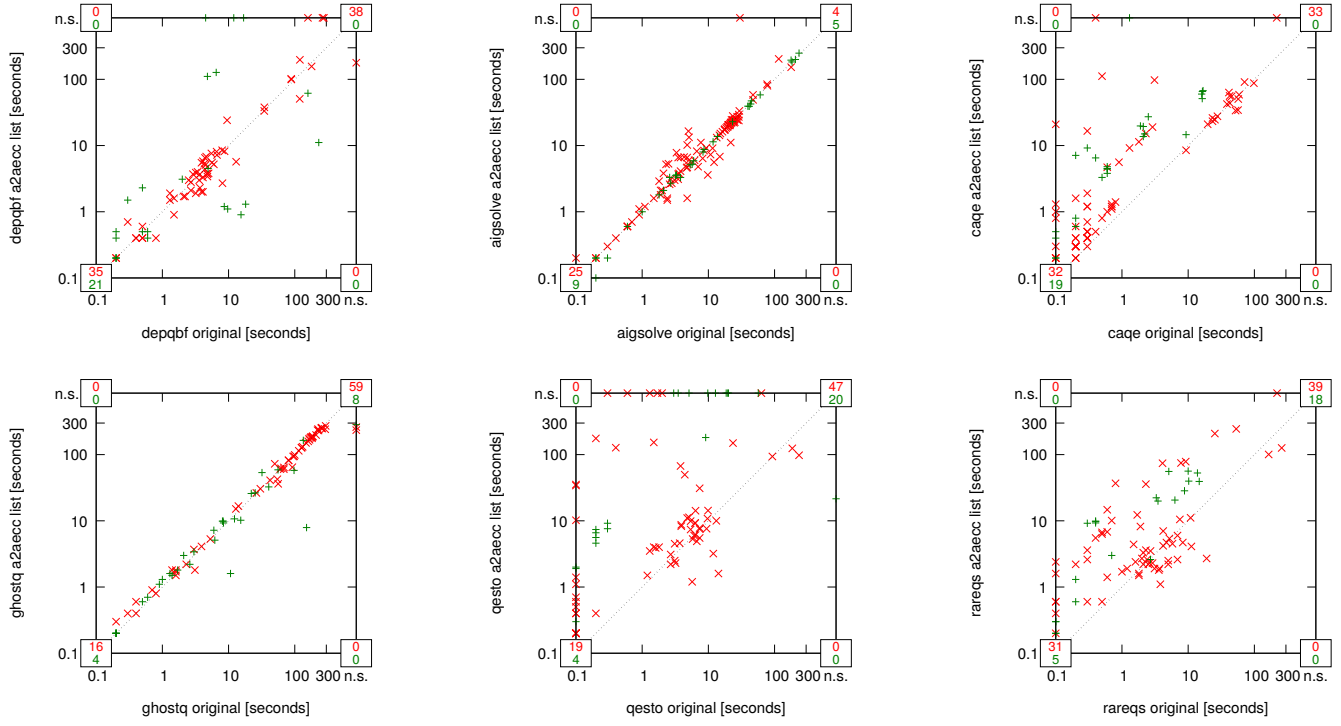


Fig. 1549: Suite Kronegger-Pfandler-Pichler ($n = 194$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

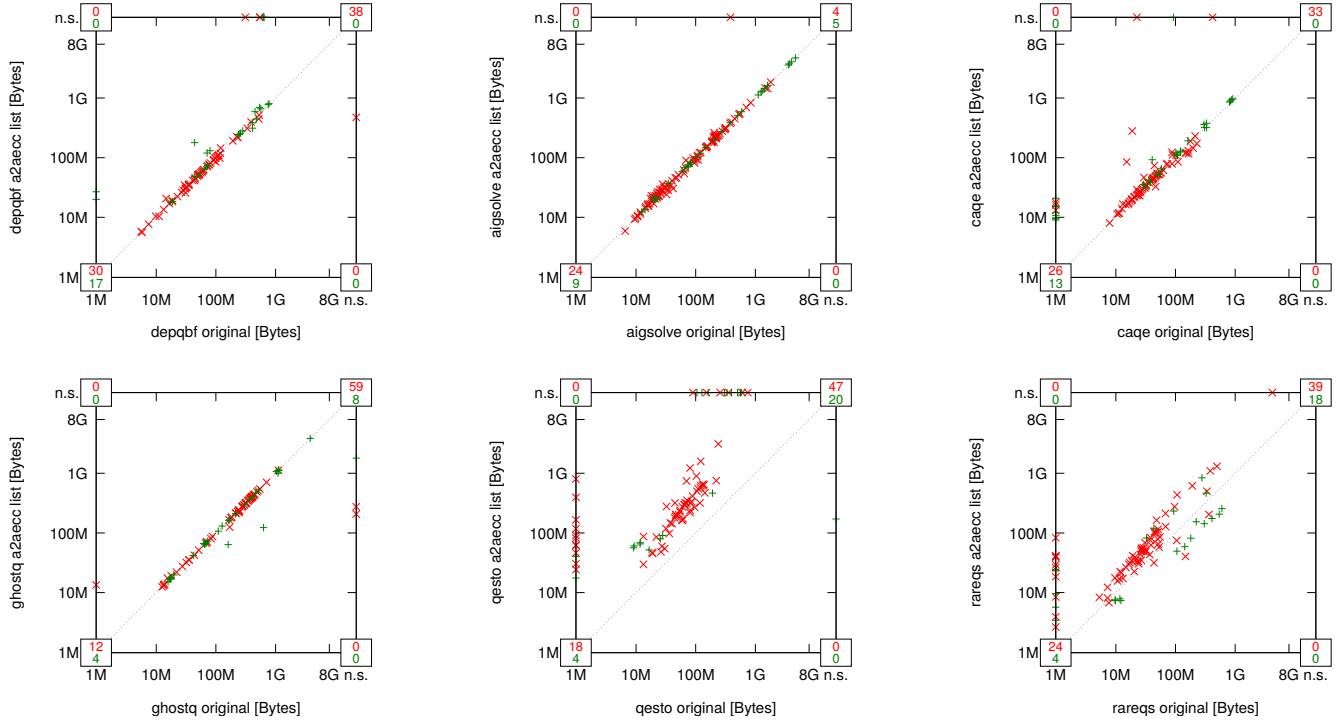


Fig. 1550: Suite Kronegger-Pfandler-Pichler ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

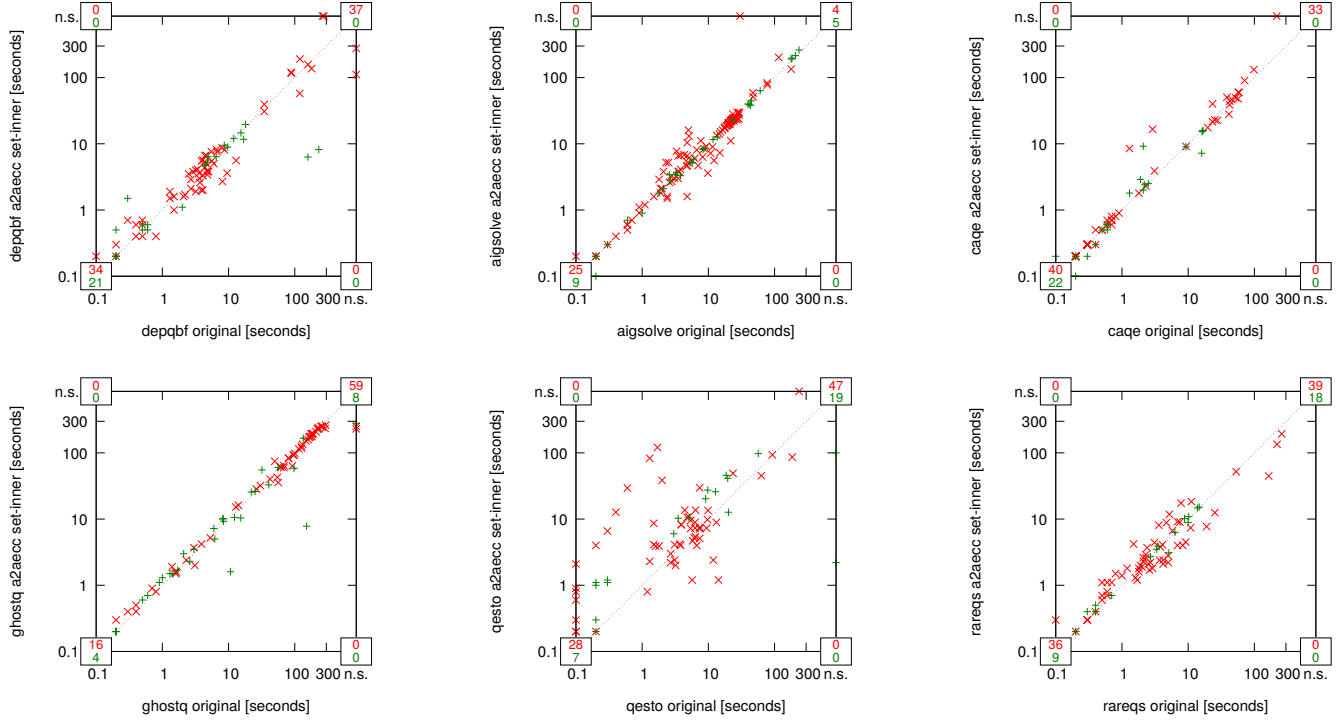


Fig. 1551: Suite Kronegger-Pfandler-Pichler ($n = 194$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

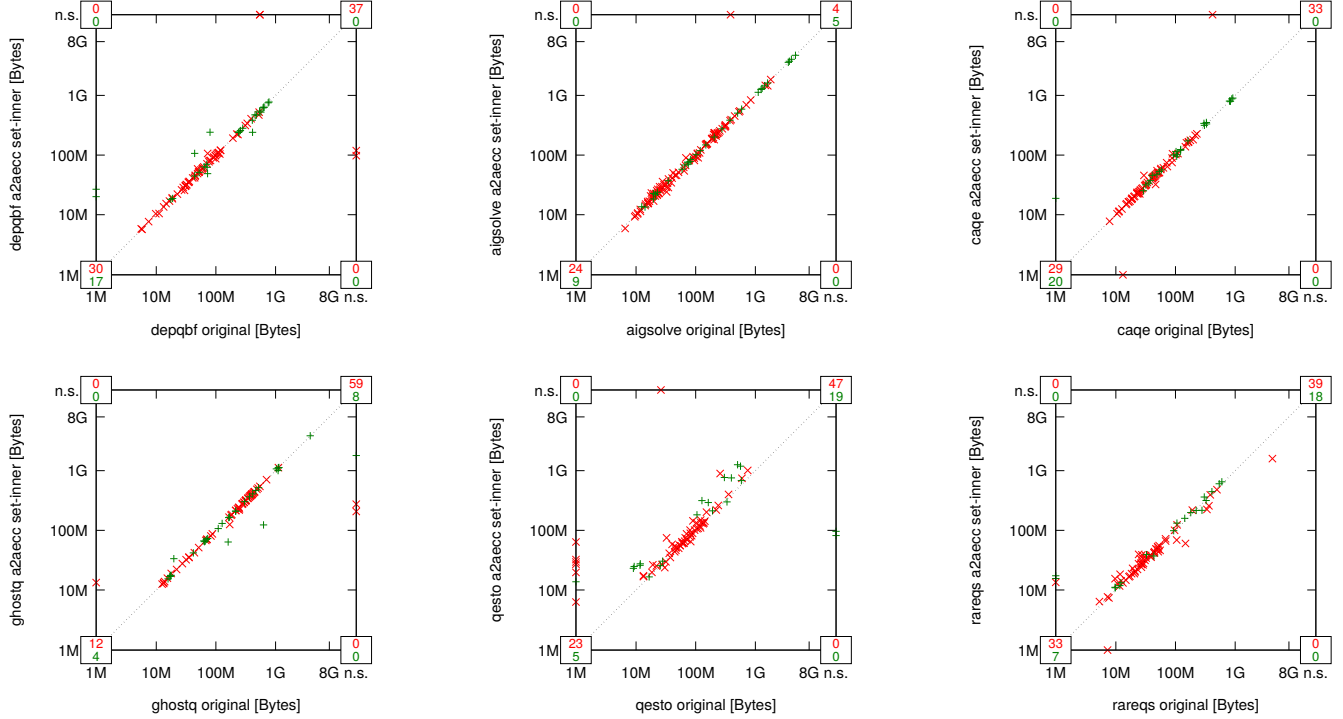


Fig. 1552: Suite Kronegger-Pfandler-Pichler ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

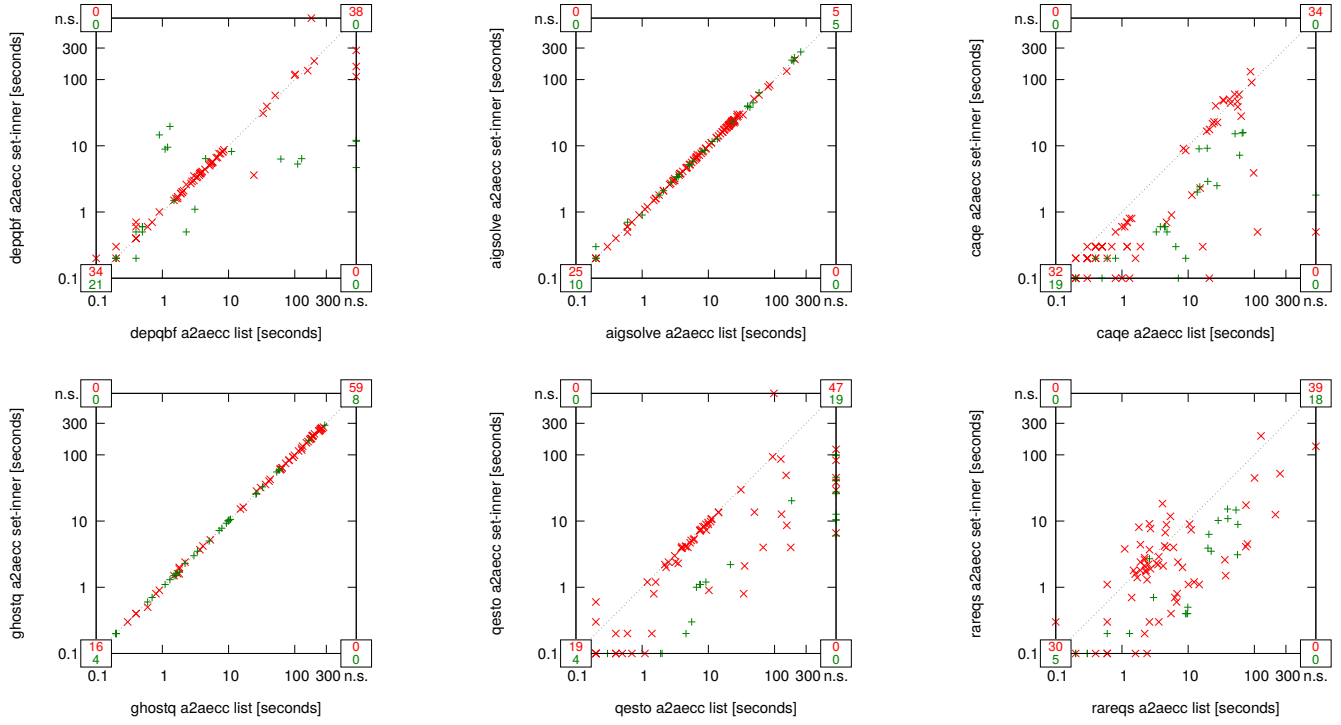


Fig. 1553: Suite Kronegger-Pfandler-Pichler ($n = 194$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

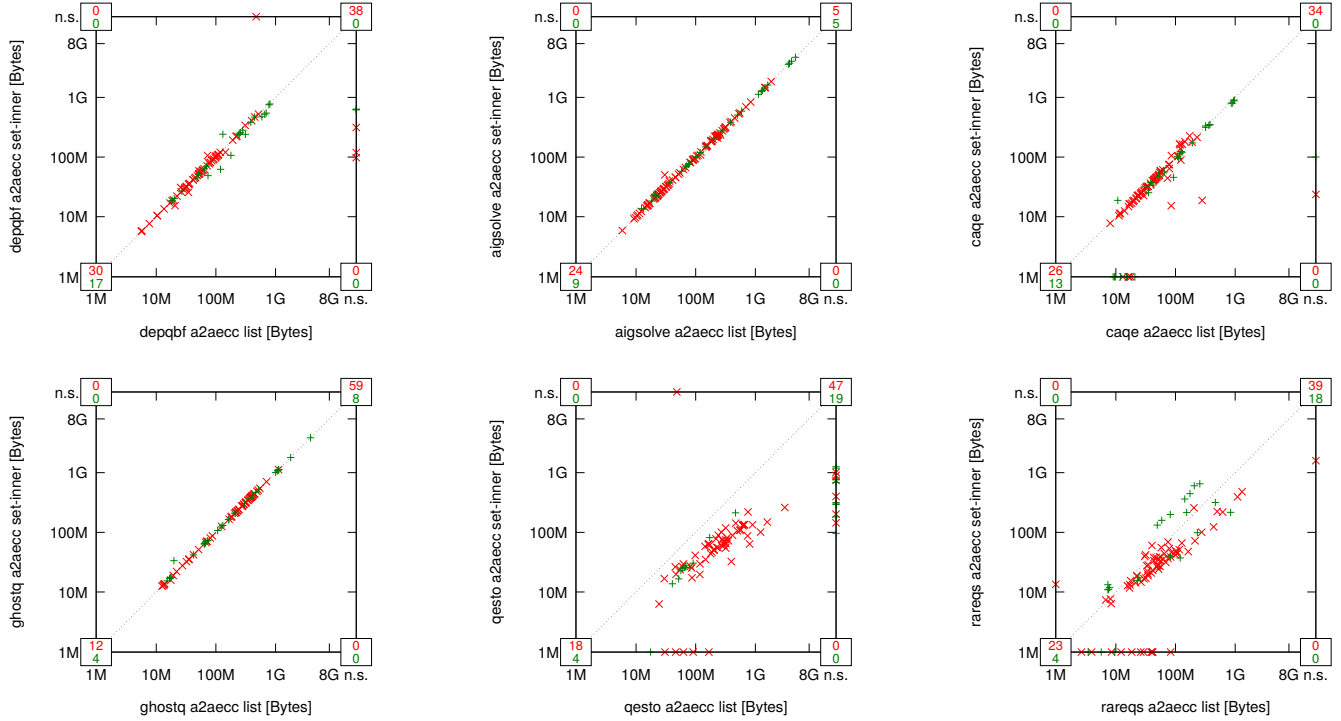


Fig. 1554: Suite Kronegger-Pfandler-Pichler ($n = 194$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

22) Lahiri-Seshia ($n = 3$):

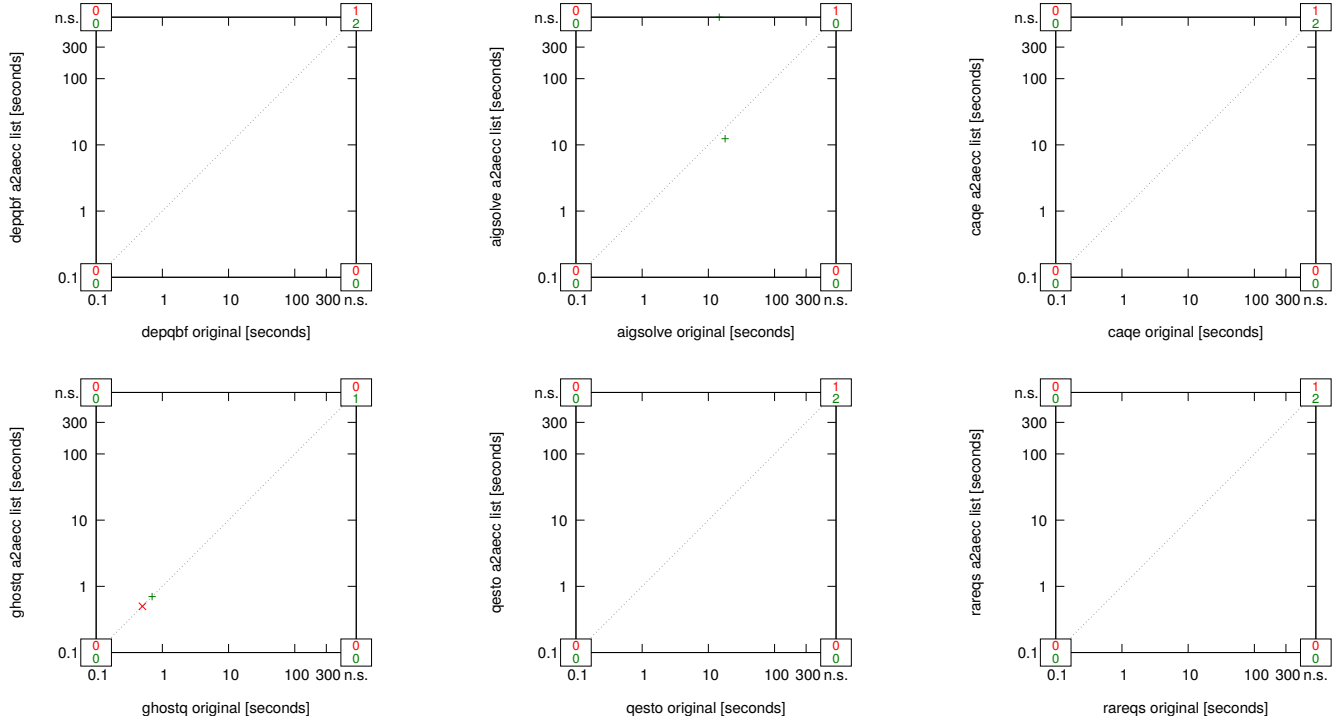


Fig. 1555: Suite Lahiri-Seshia ($n = 3$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

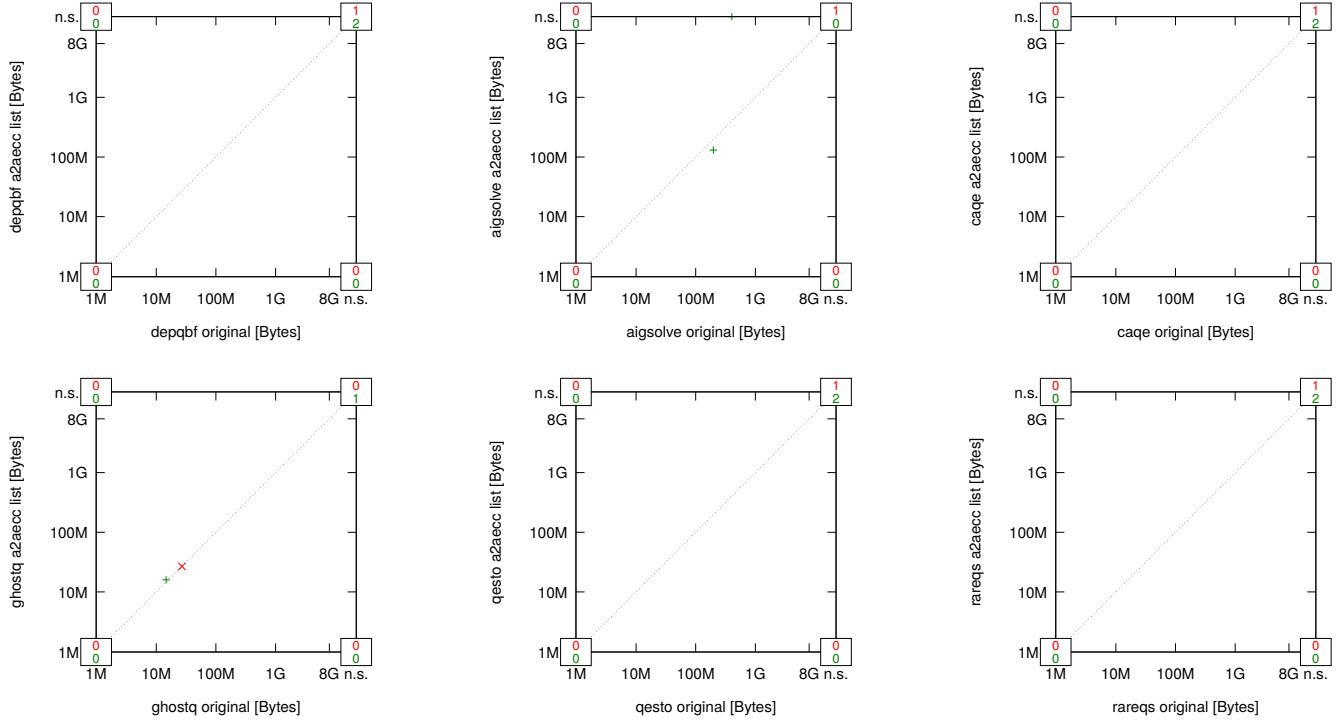


Fig. 1556: Suite Lahiri-Seshia ($n = 3$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

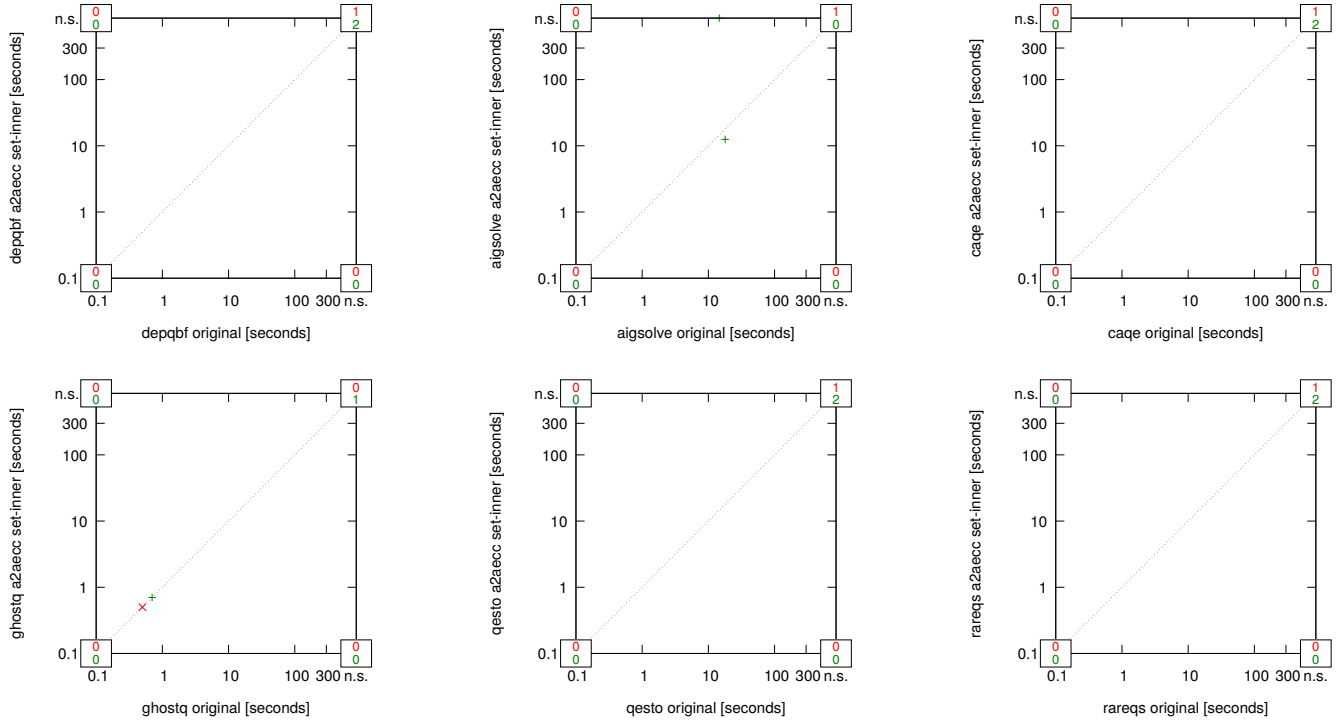


Fig. 1557: Suite Lahiri-Seshia ($n = 3$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

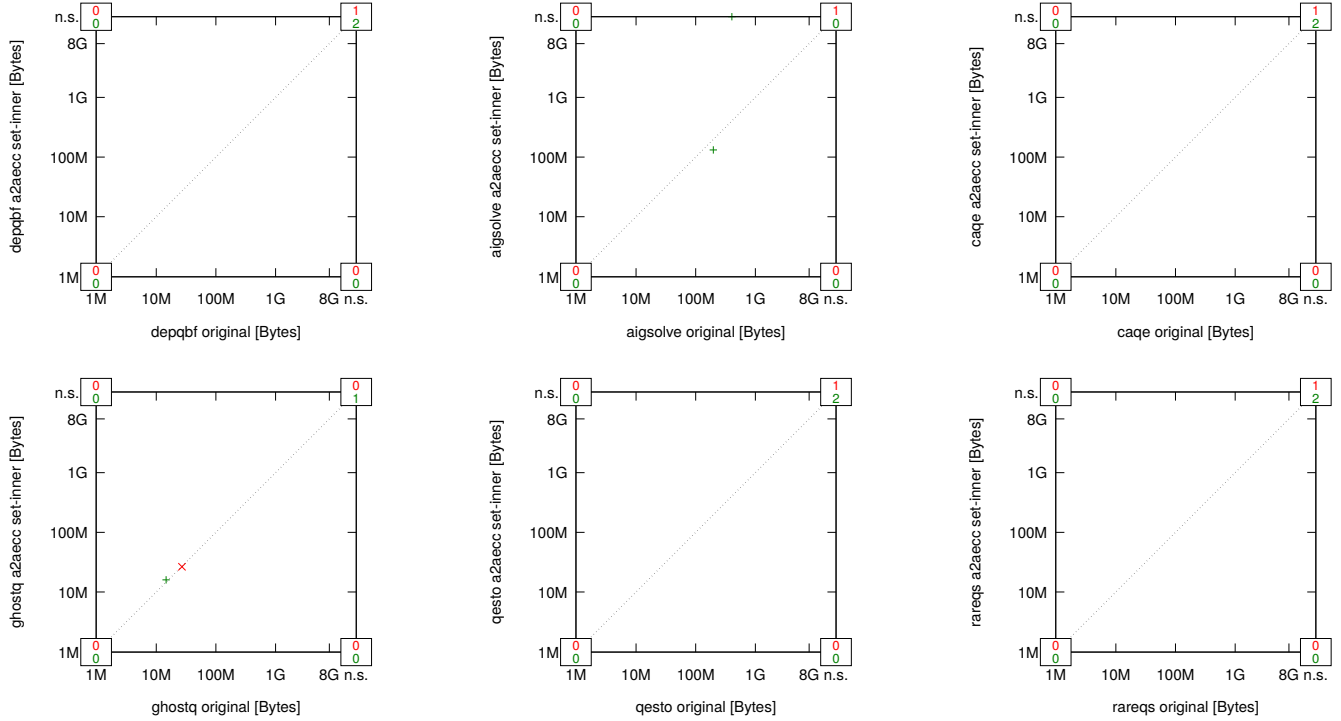


Fig. 1558: Suite Lahiri-Seshia ($n = 3$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

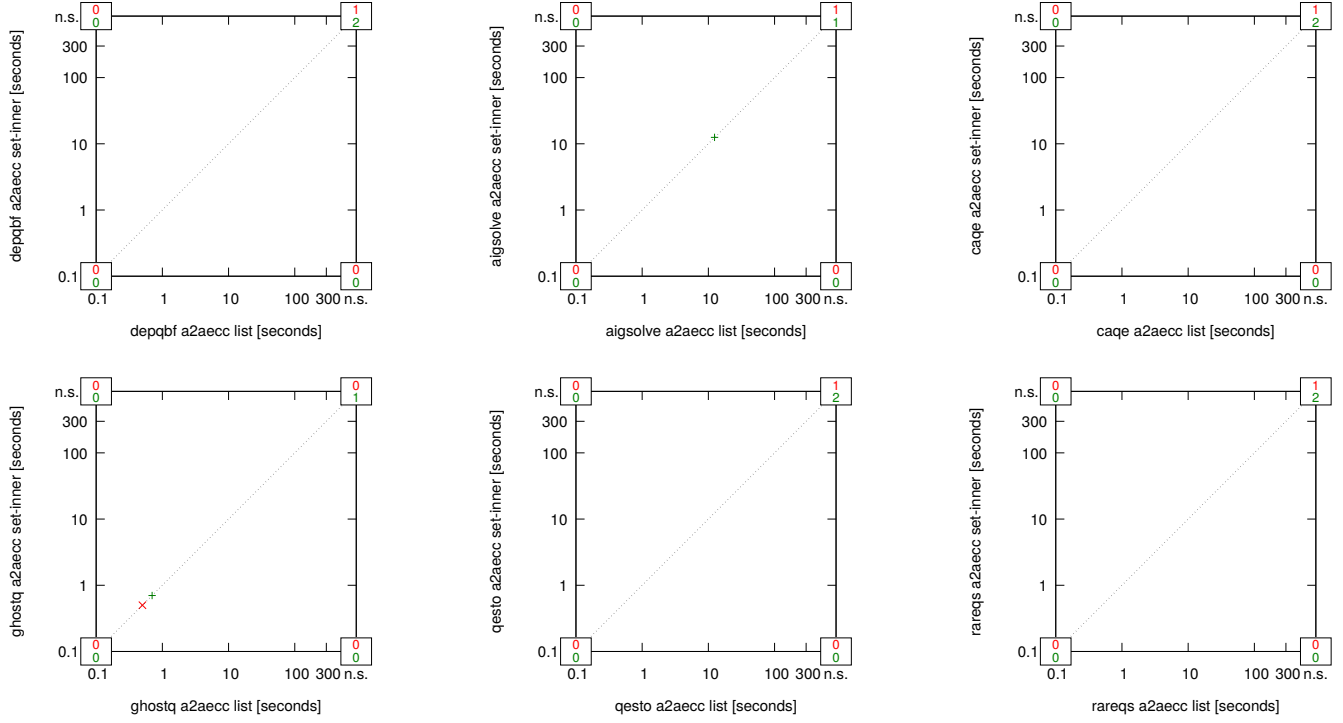


Fig. 1559: Suite Lahiri-Seshia ($n = 3$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

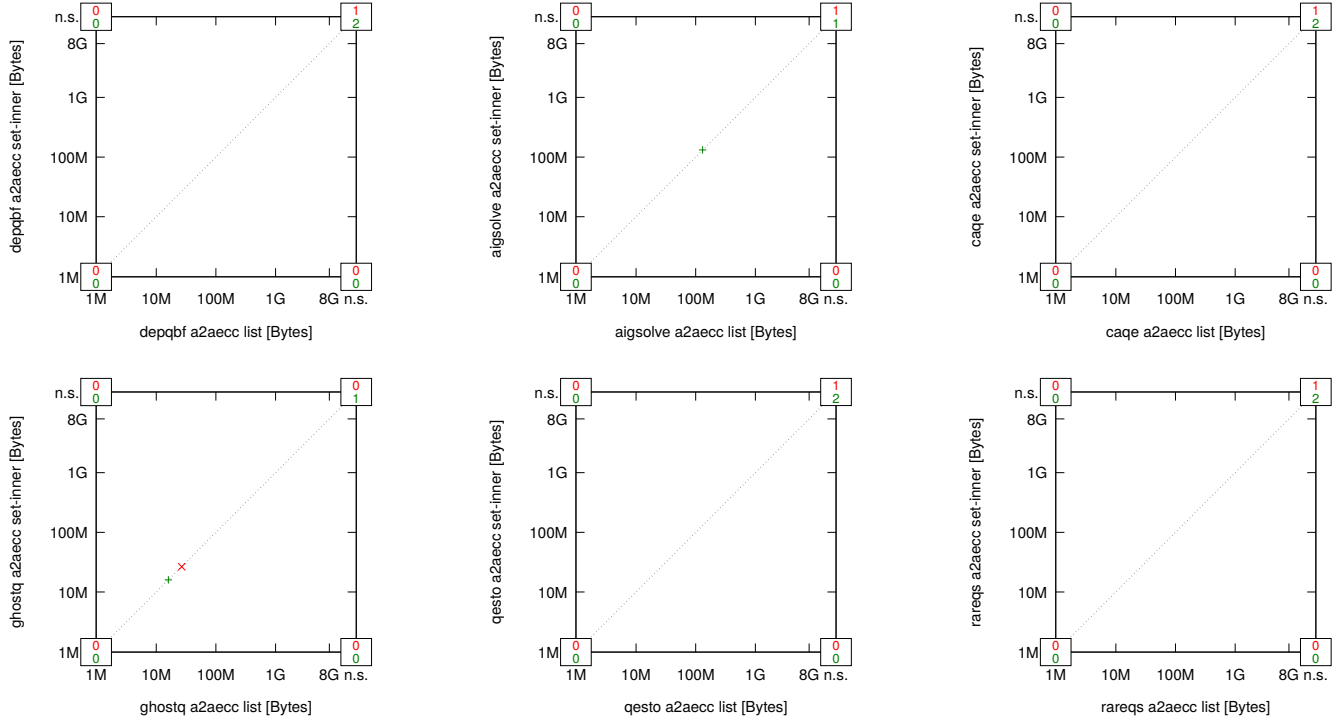


Fig. 1560: Suite Lahiri-Seshia ($n = 3$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

23) Lee-Jiang ($n = 5$):

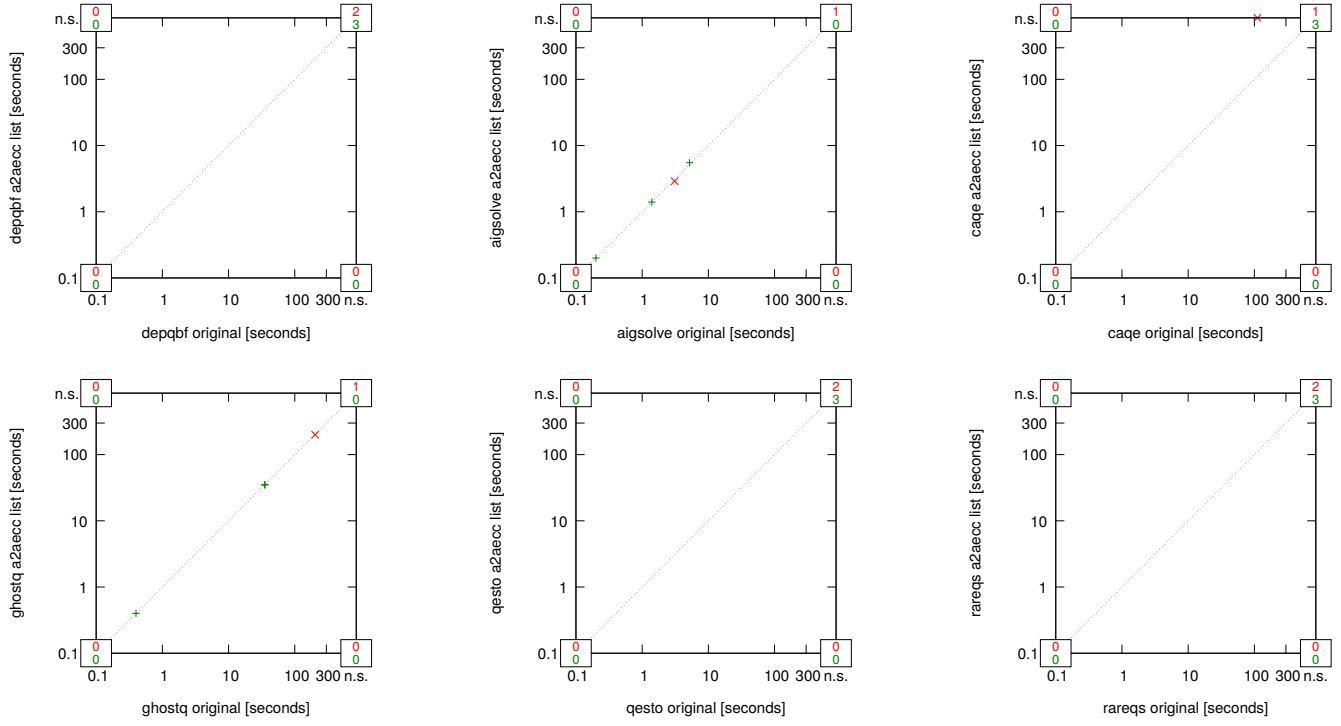


Fig. 1561: Suite Lee-Jiang ($n = 5$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

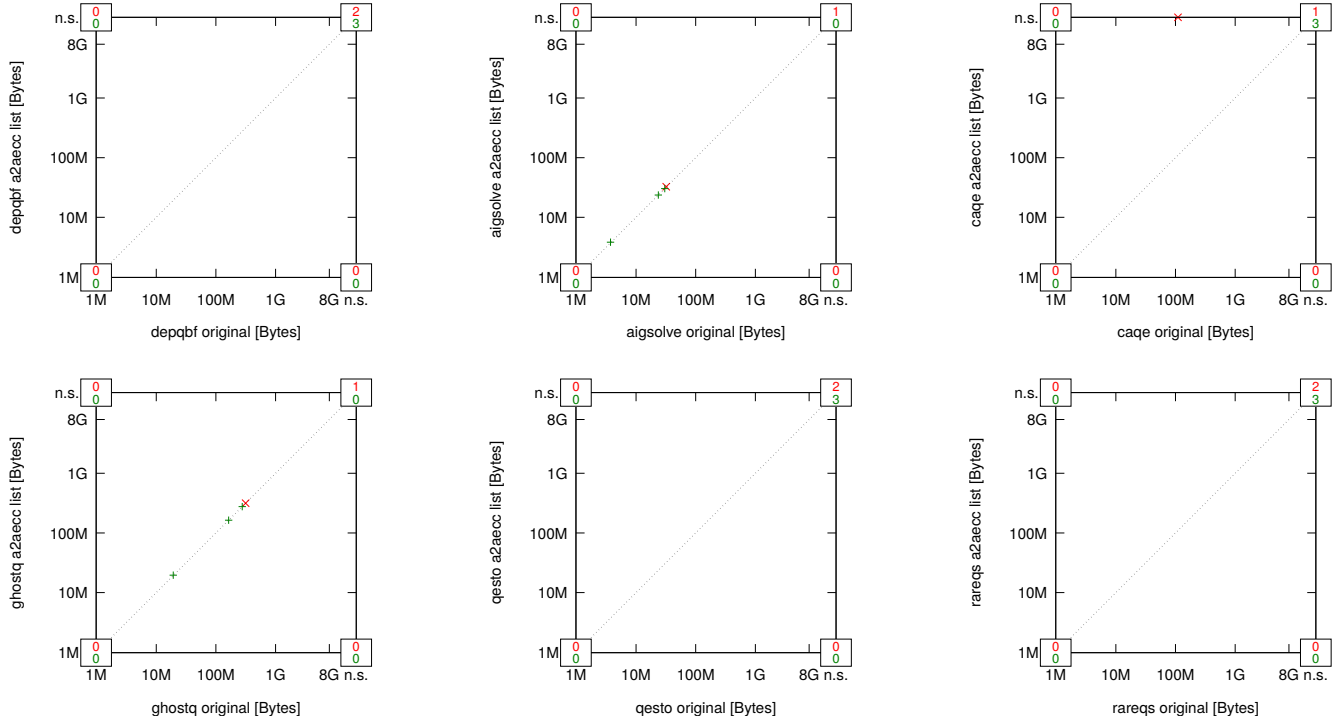


Fig. 1562: Suite Lee-Jiang ($n = 5$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

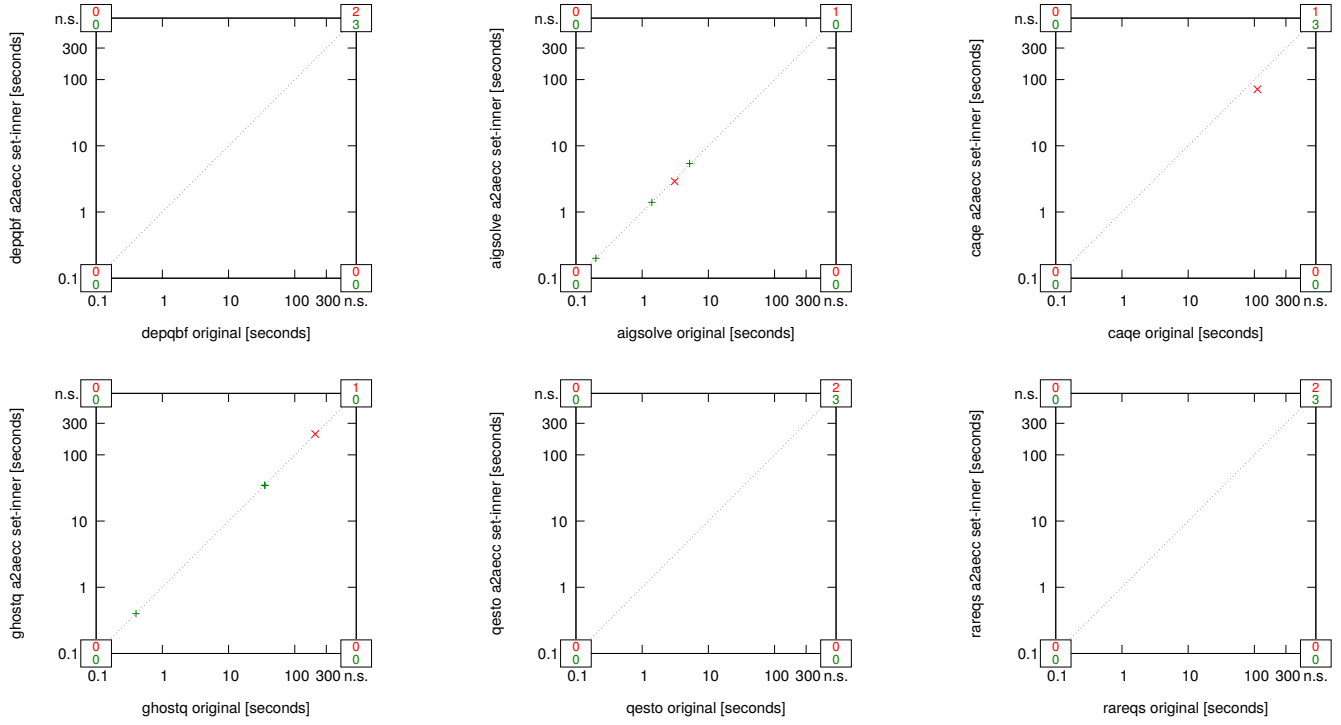


Fig. 1563: Suite Lee-Jiang ($n = 5$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

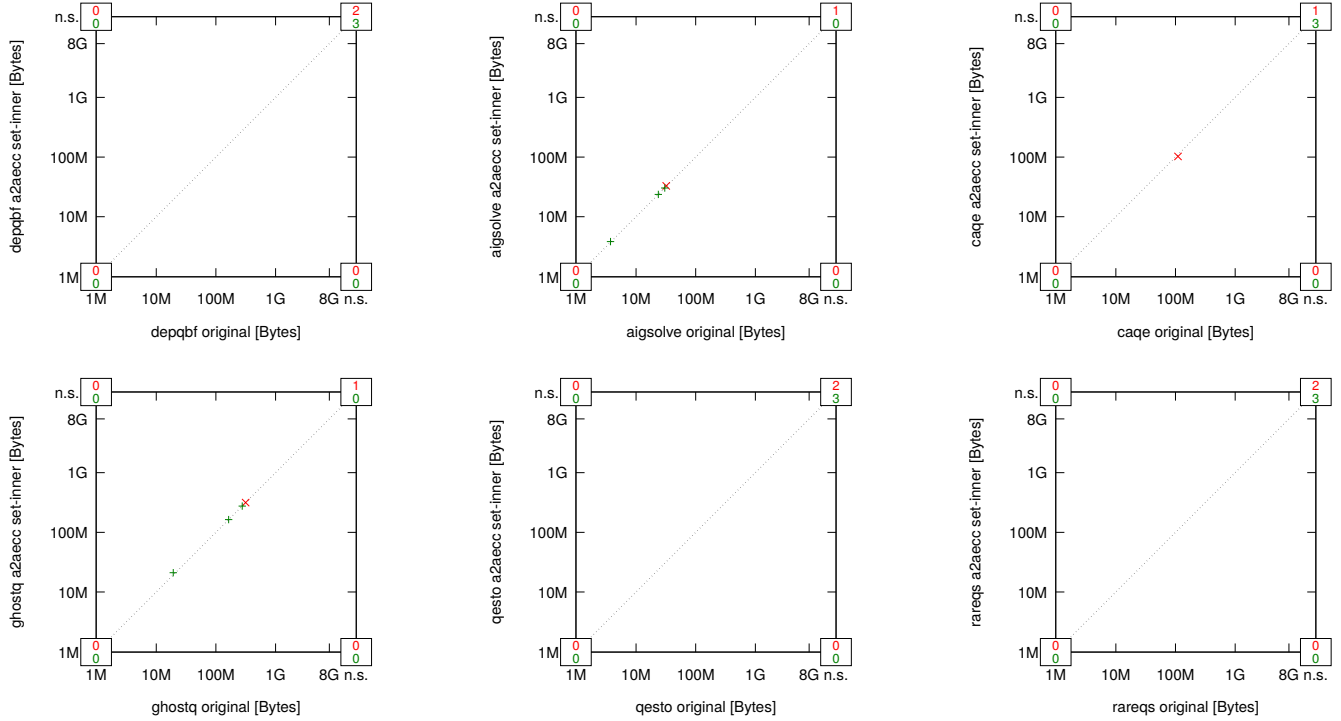


Fig. 1564: Suite Lee-Jiang ($n = 5$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

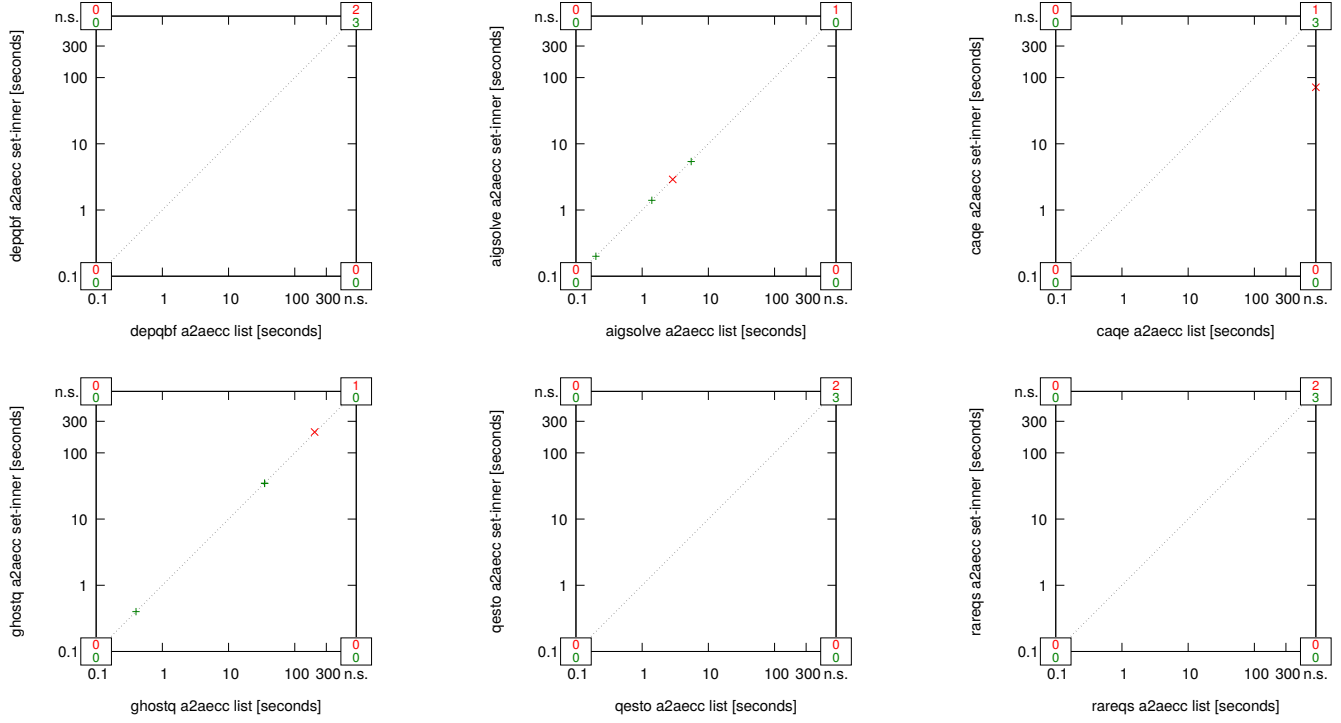


Fig. 1565: Suite Lee-Jiang ($n = 5$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

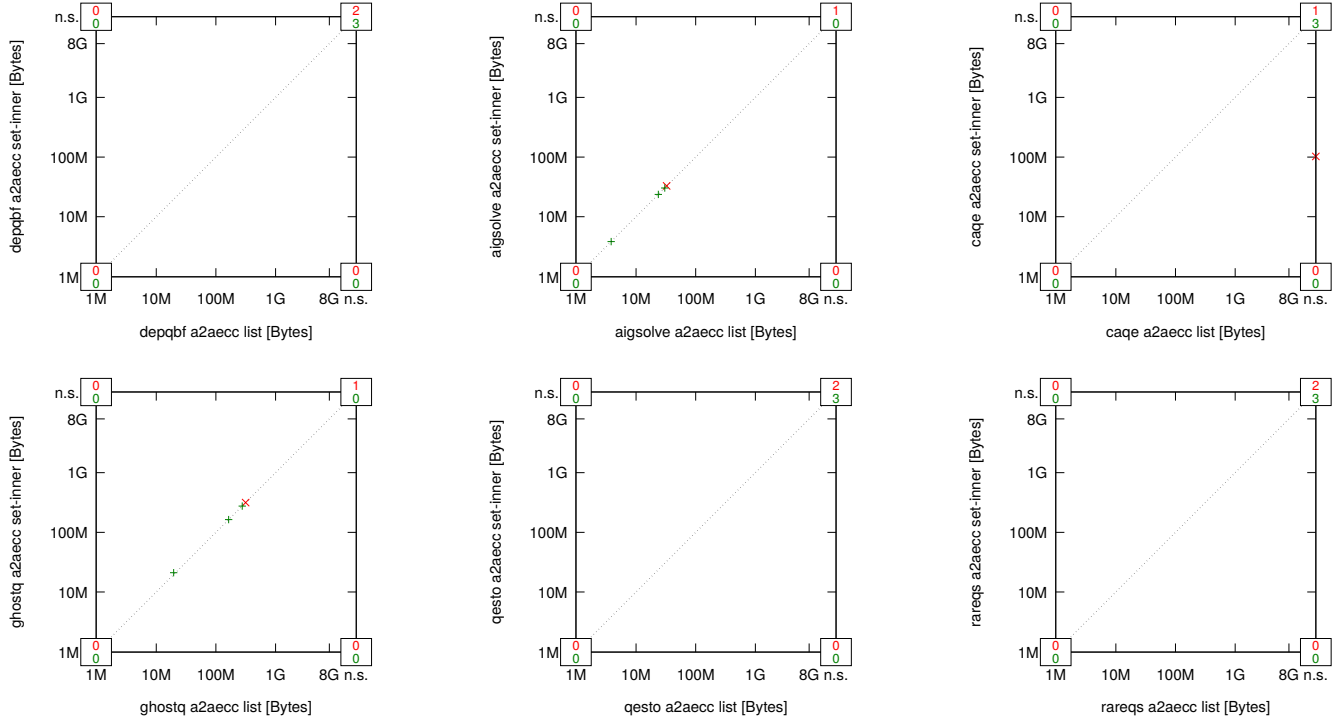


Fig. 1566: Suite Lee-Jiang ($n = 5$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

24) Letombe ($n = 194$):

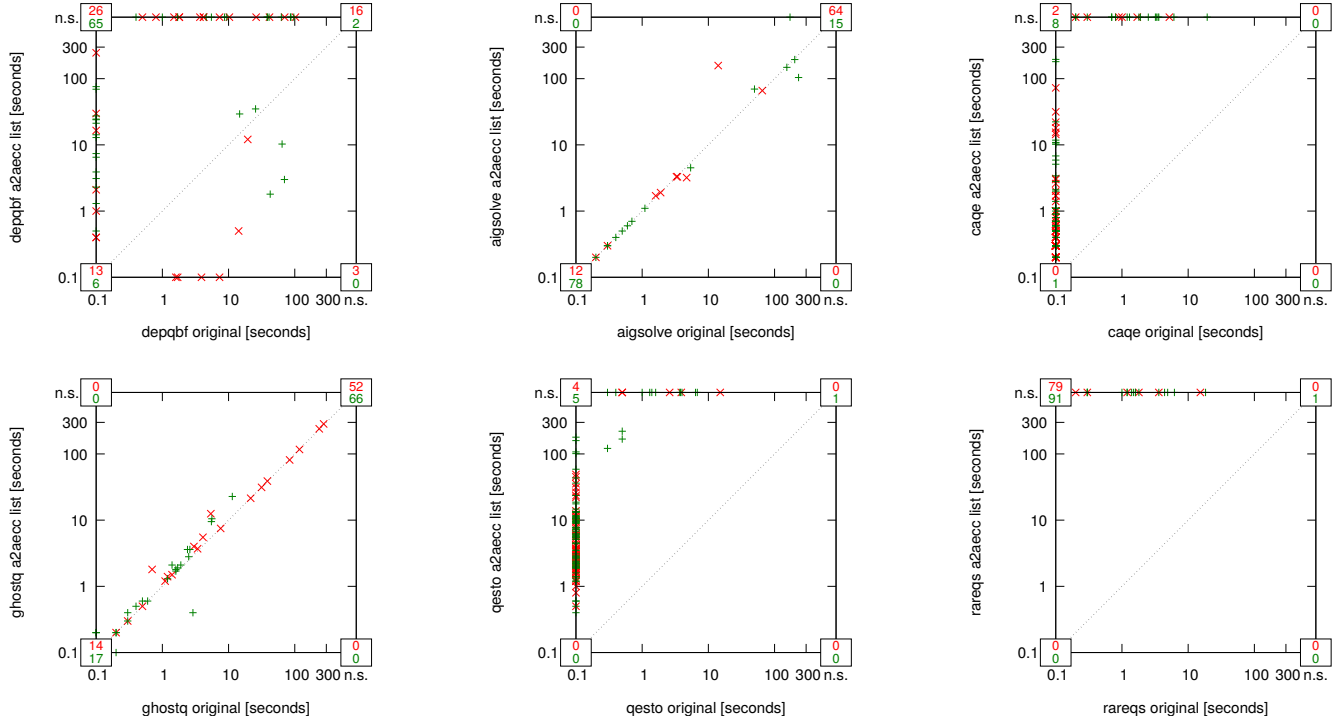


Fig. 1567: Suite Letombe ($n = 194$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

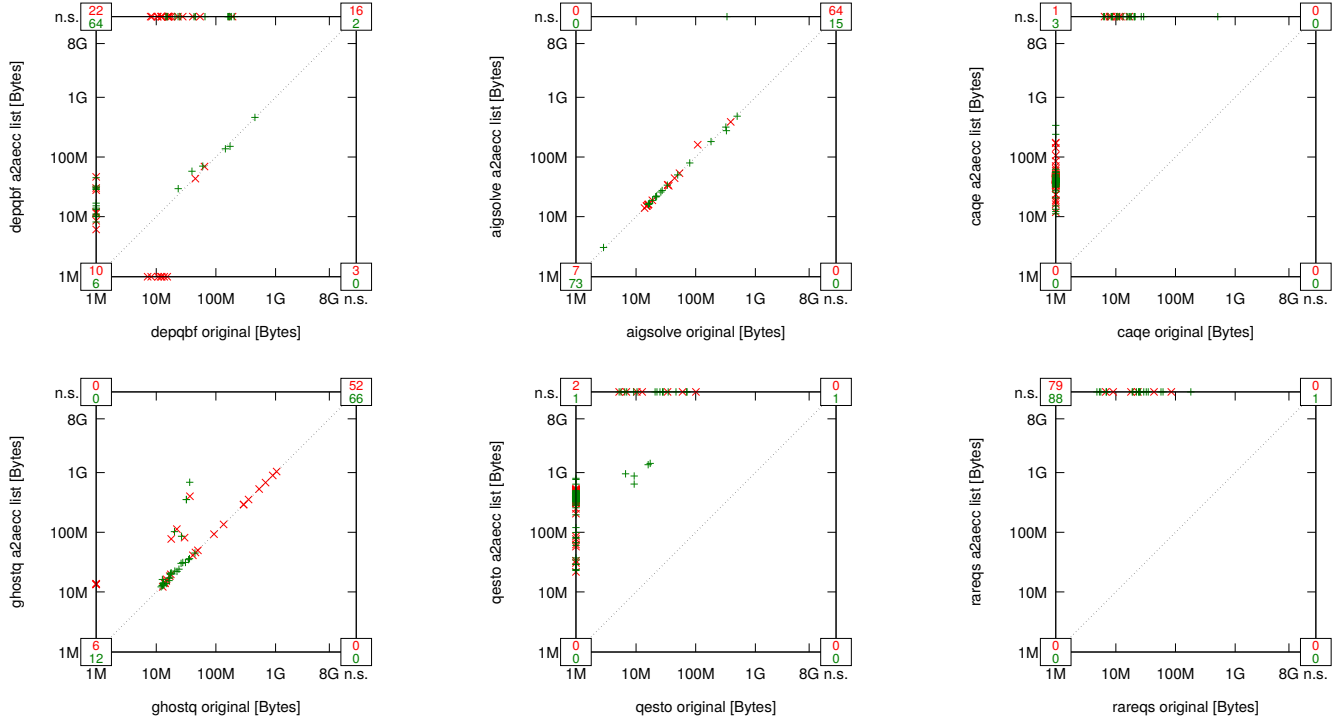


Fig. 1568: Suite Letombe ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

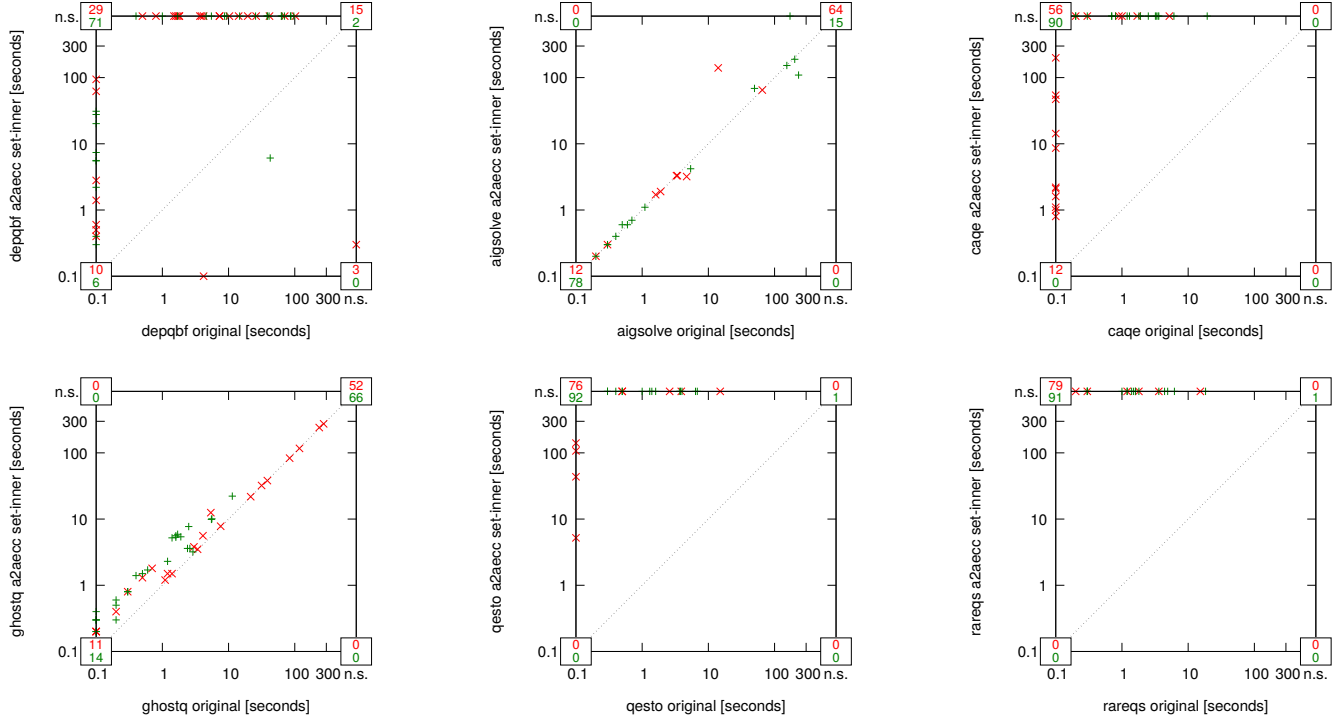


Fig. 1569: Suite Letombe ($n = 194$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

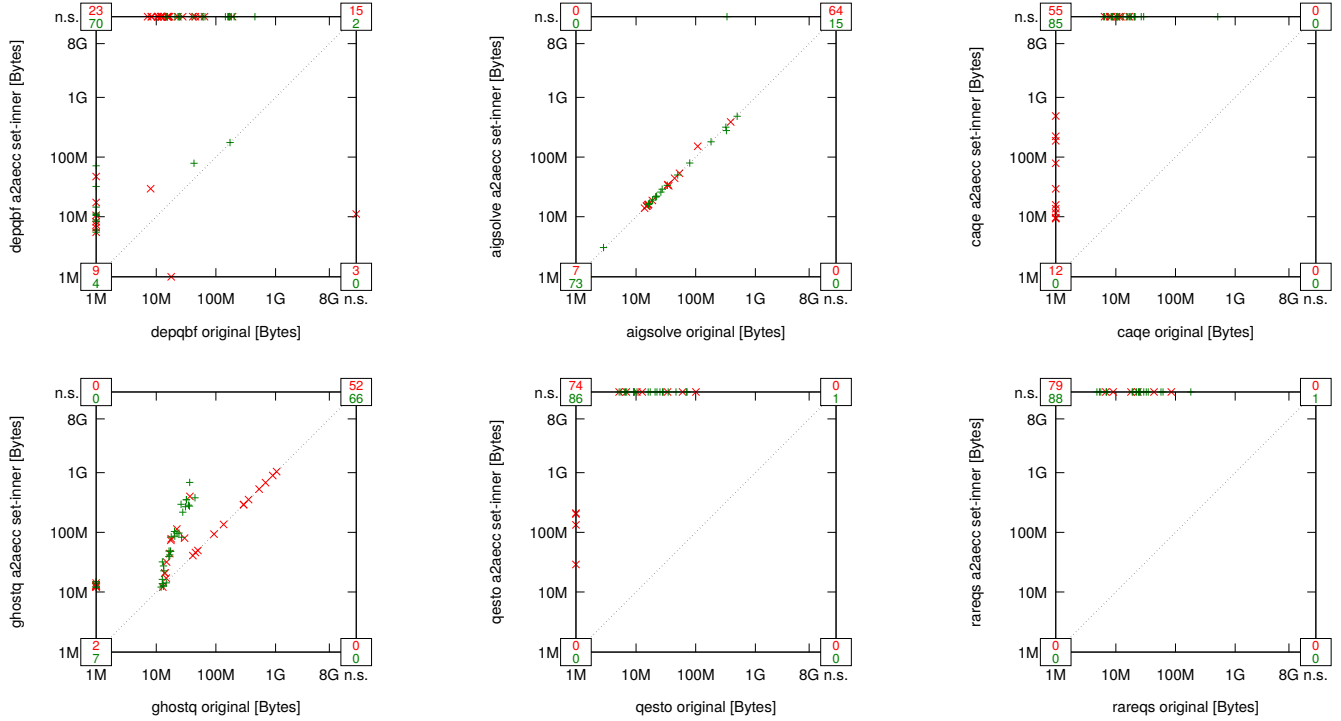


Fig. 1570: Suite Letombe ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

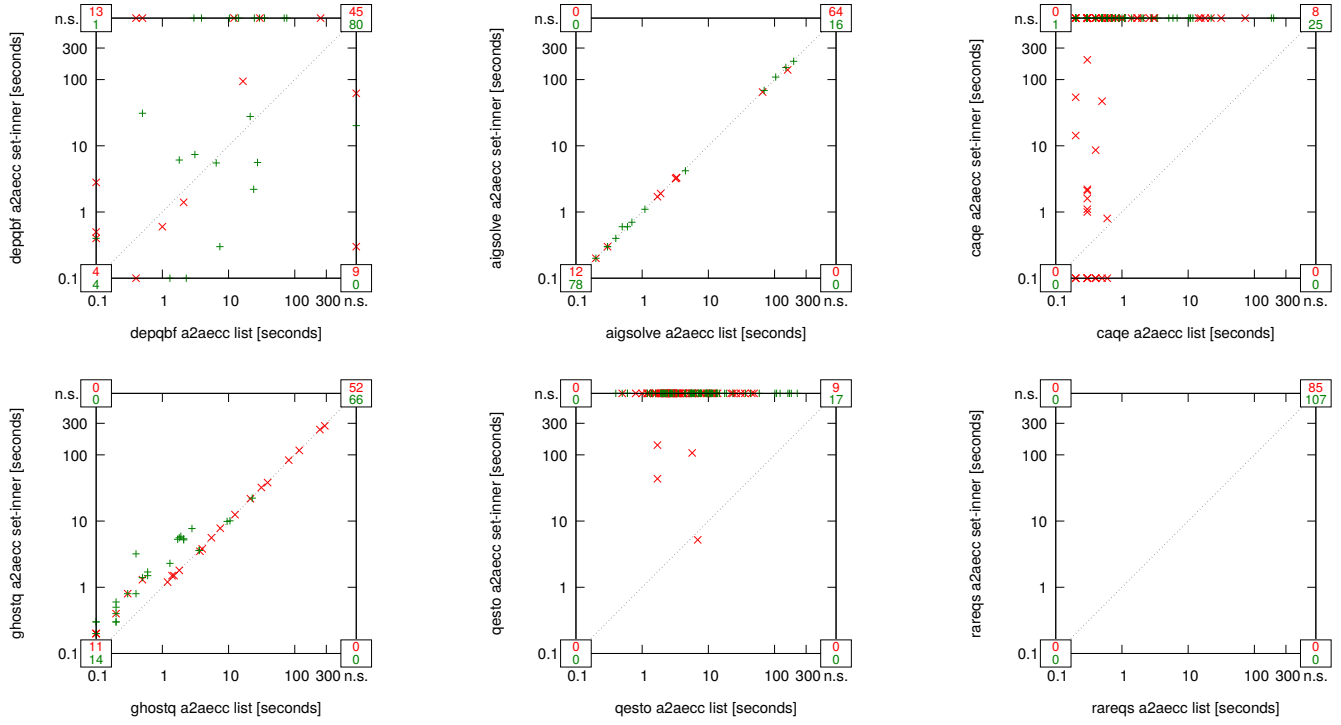


Fig. 1571: Suite Letombe ($n = 194$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

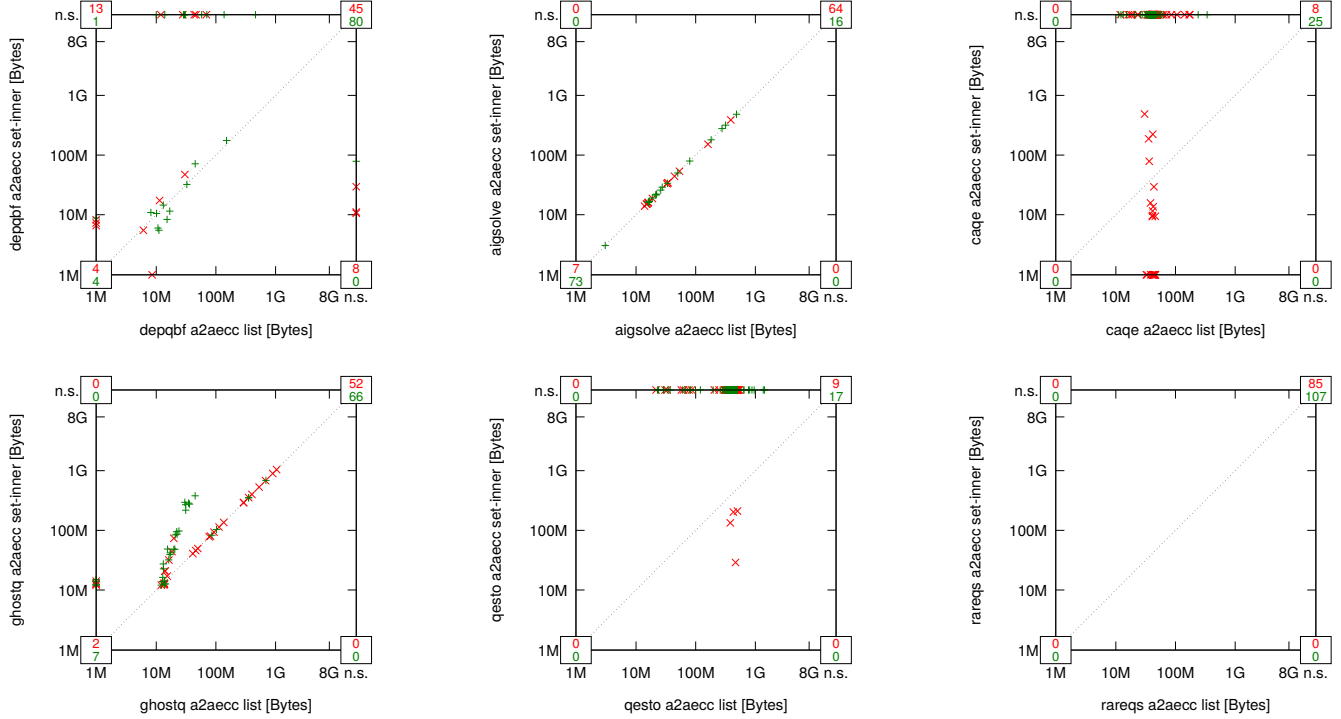


Fig. 1572: Suite Letombe ($n = 194$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

25) Letz ($n = 14$):

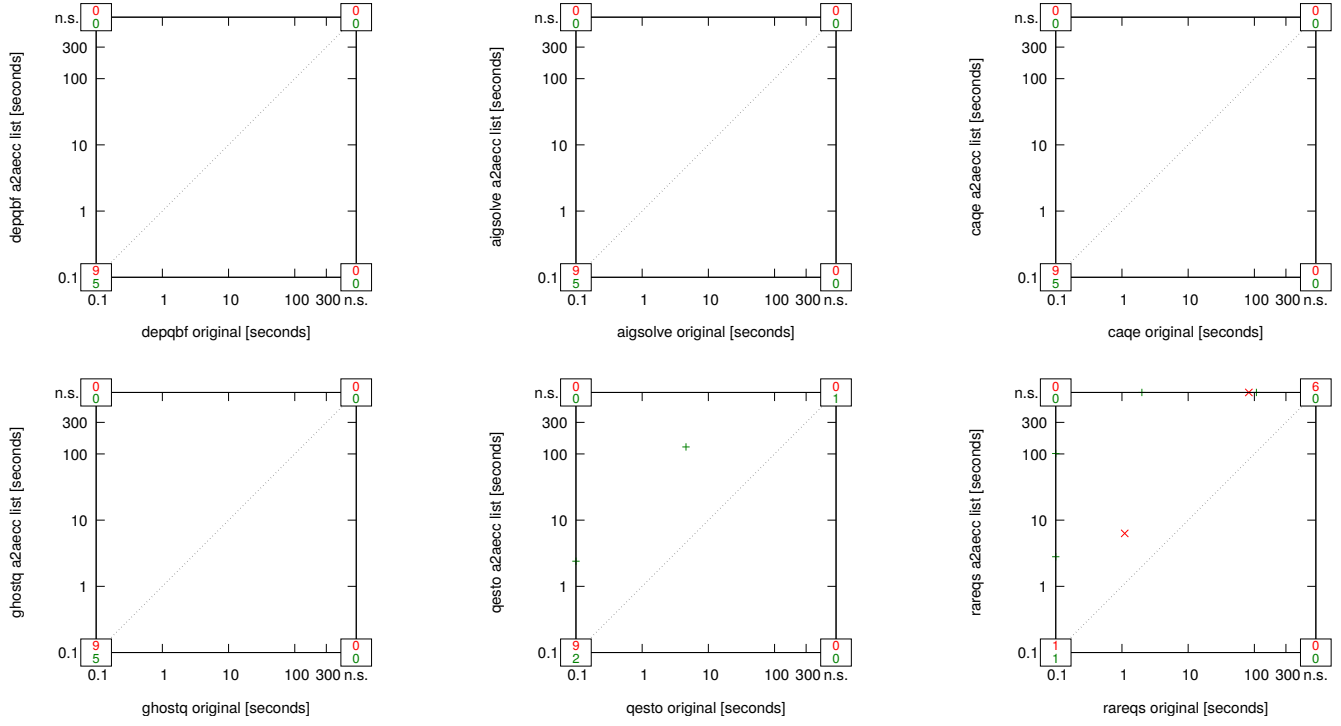


Fig. 1573: Suite Letz ($n = 14$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

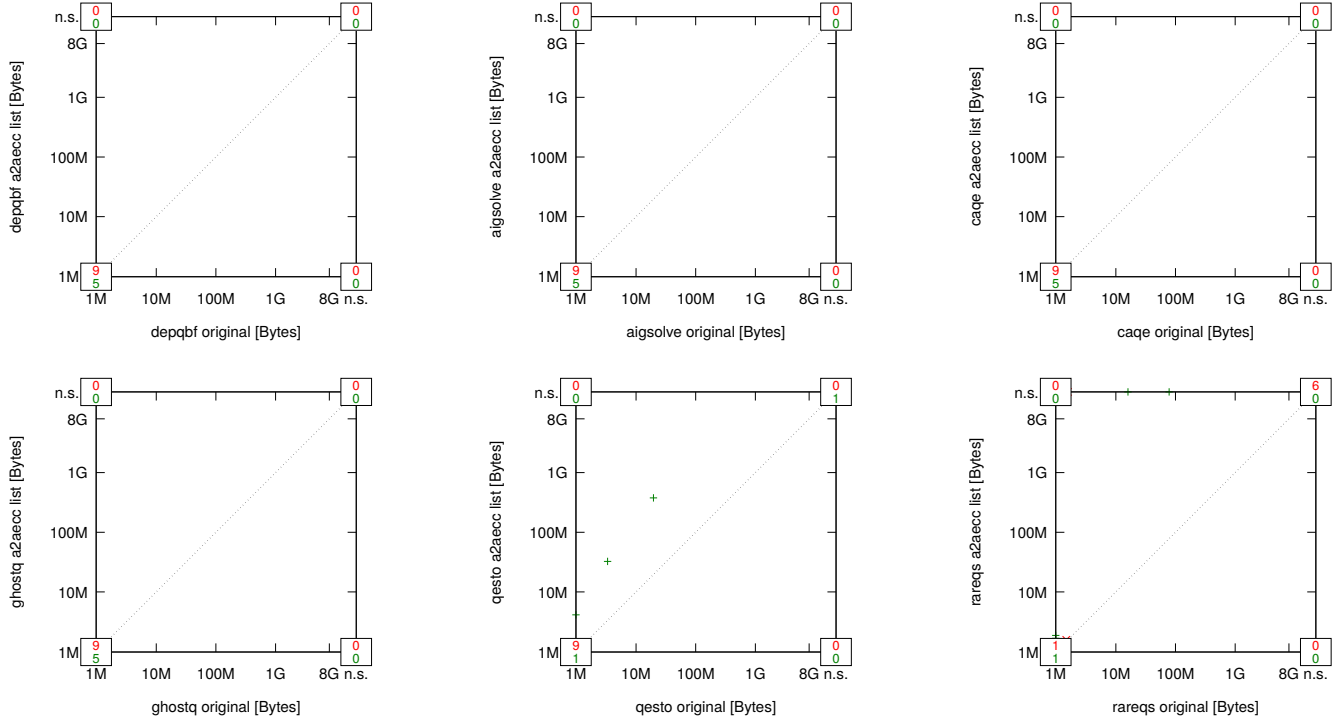


Fig. 1574: Suite Letz ($n = 14$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

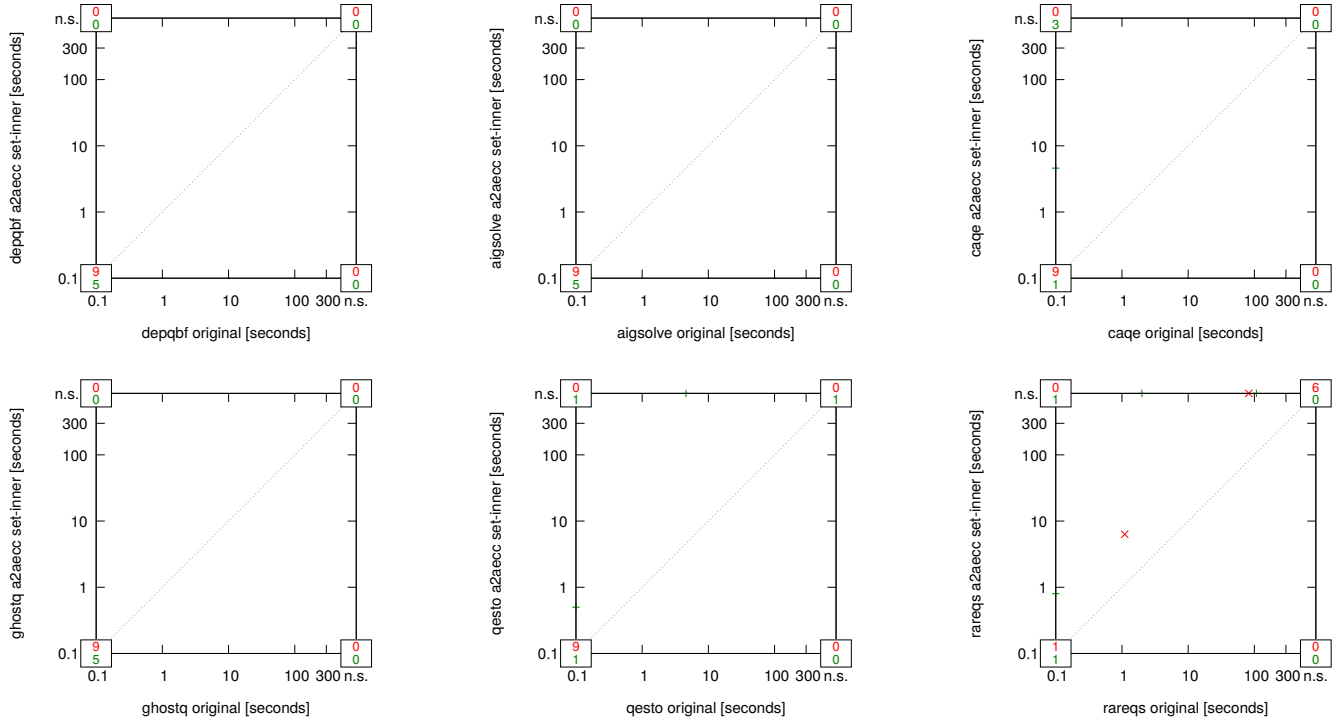


Fig. 1575: Suite Letz ($n = 14$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

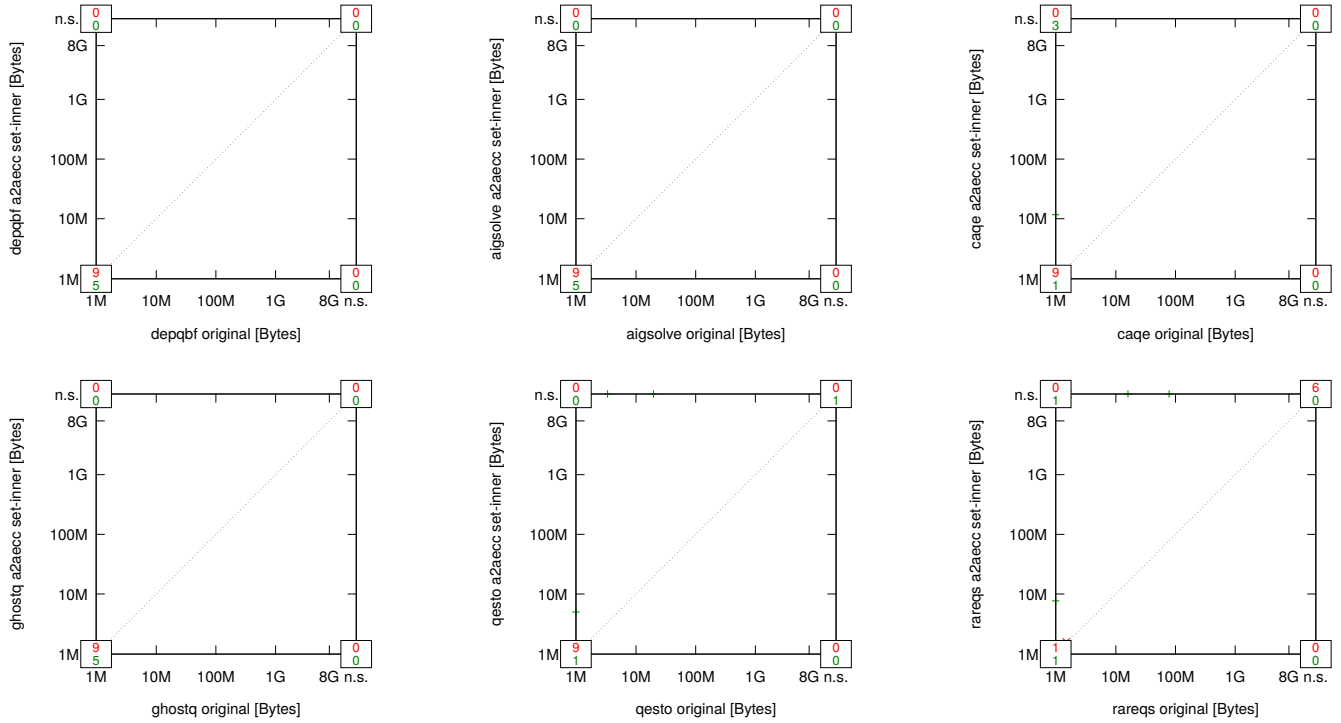


Fig. 1576: Suite Letz ($n = 14$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

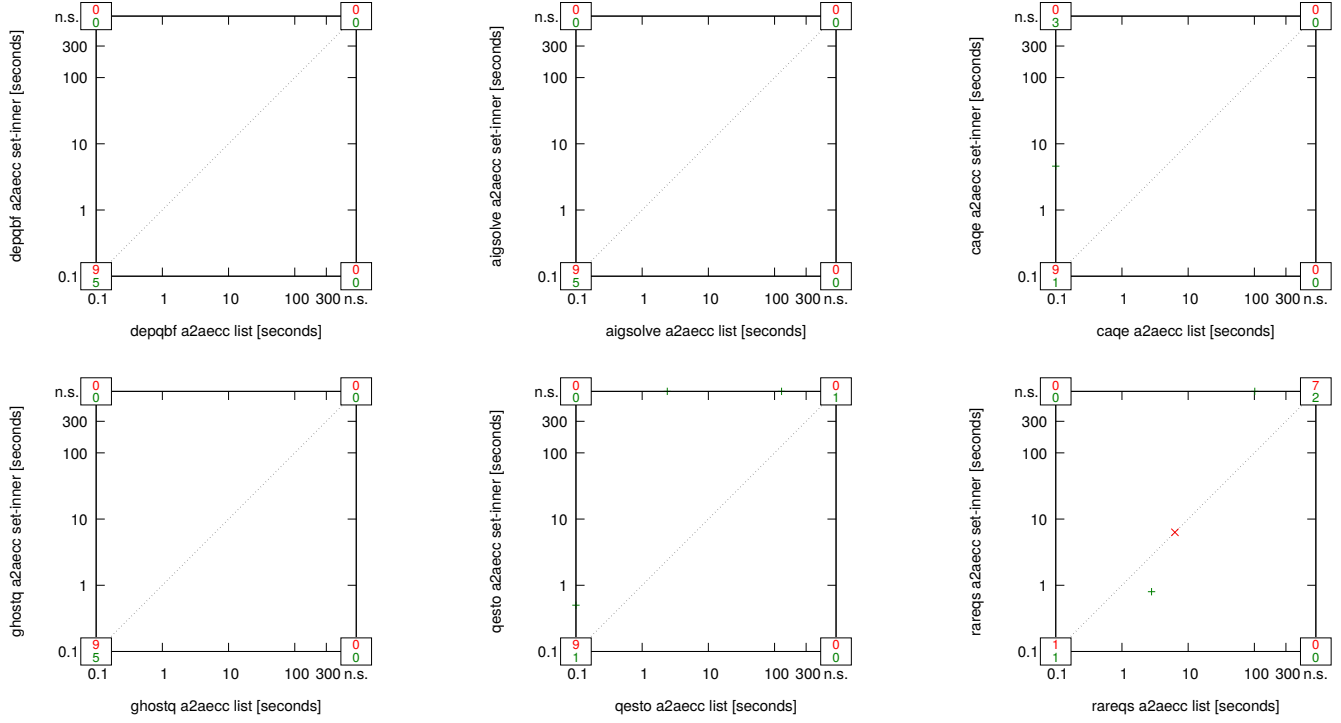


Fig. 1577: Suite Letz ($n = 14$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

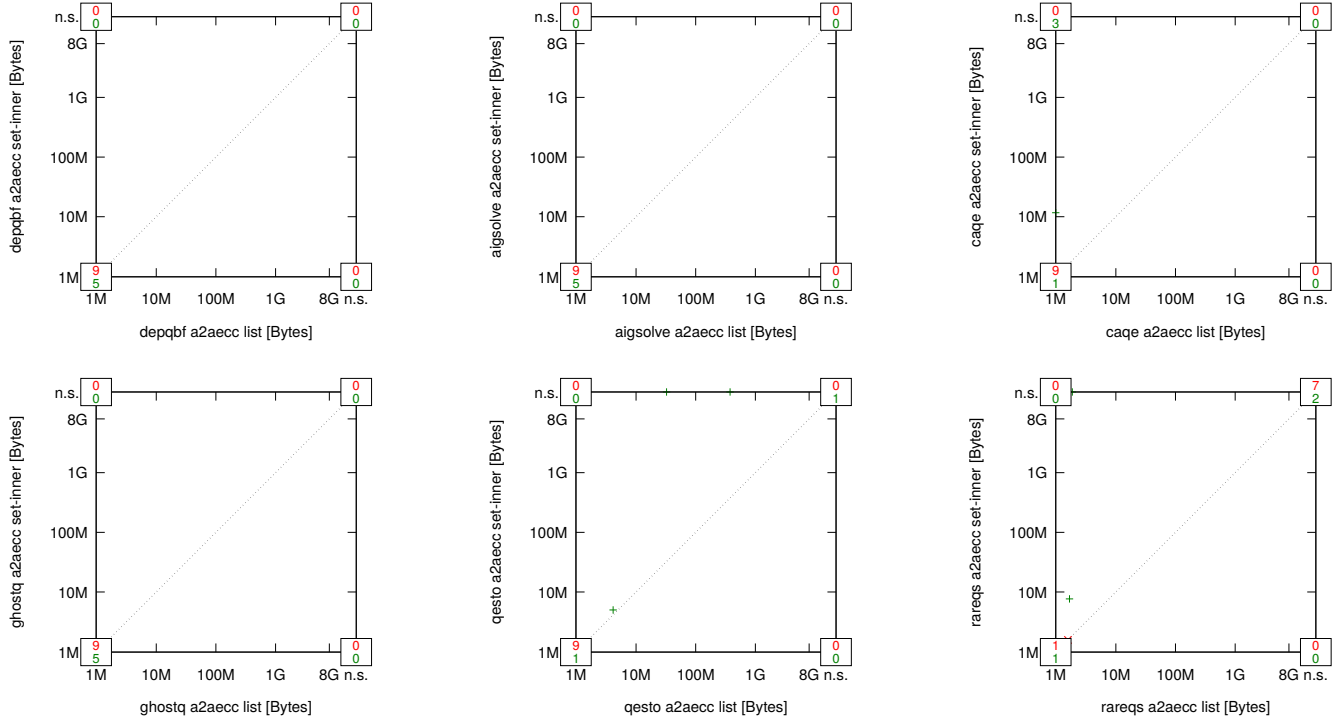


Fig. 1578: Suite Letz ($n = 14$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

26) Ling ($n = 8$):

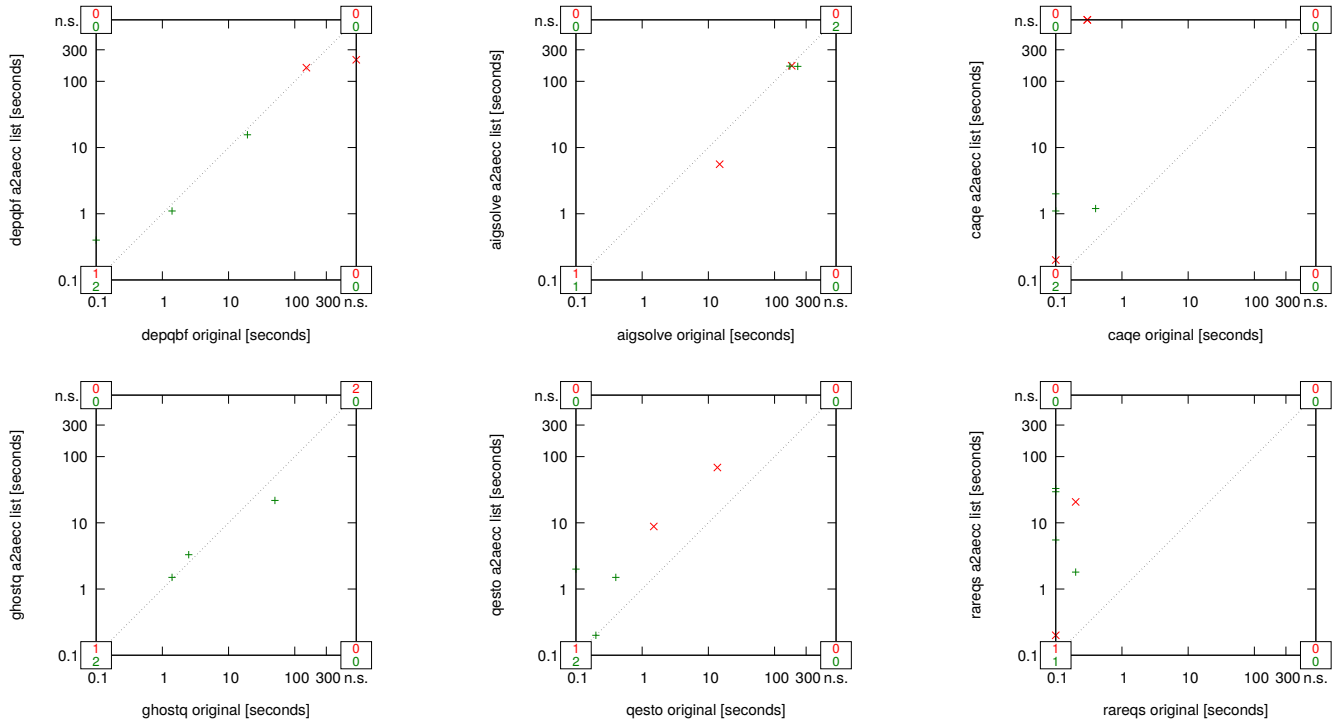


Fig. 1579: Suite Ling ($n = 8$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

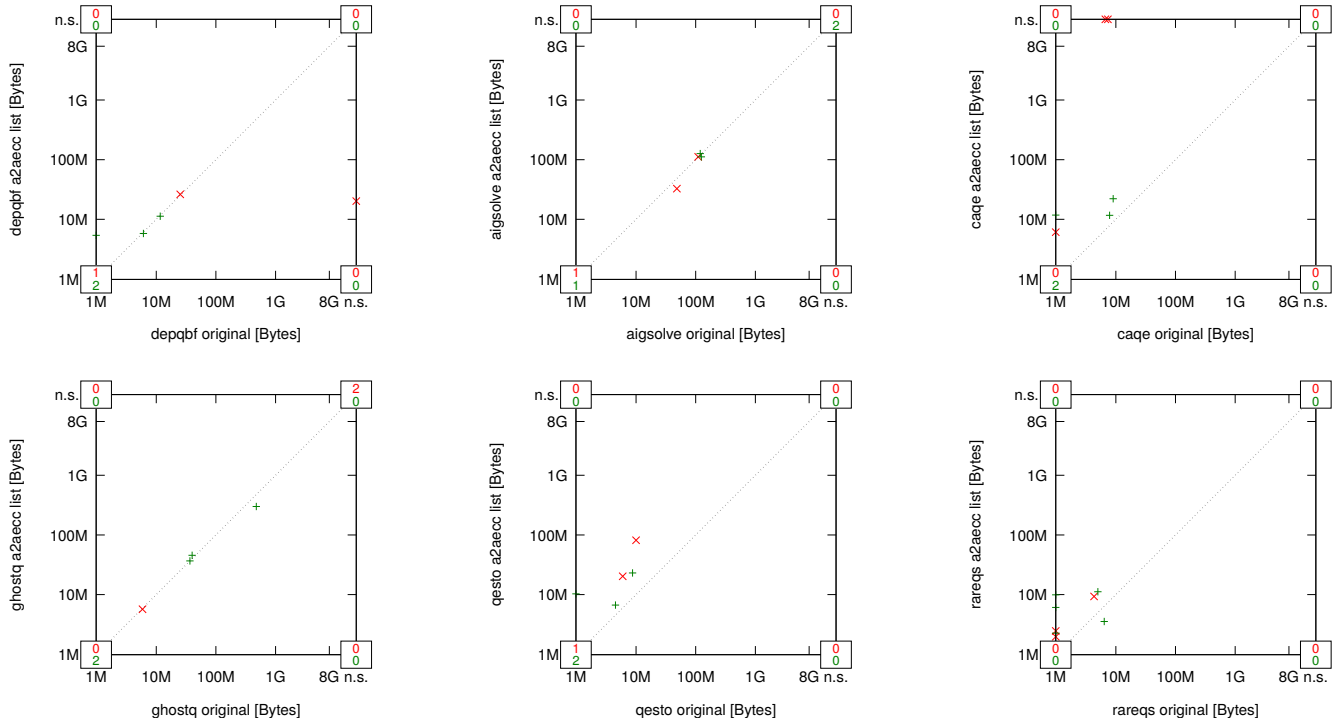


Fig. 1580: Suite Ling ($n = 8$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

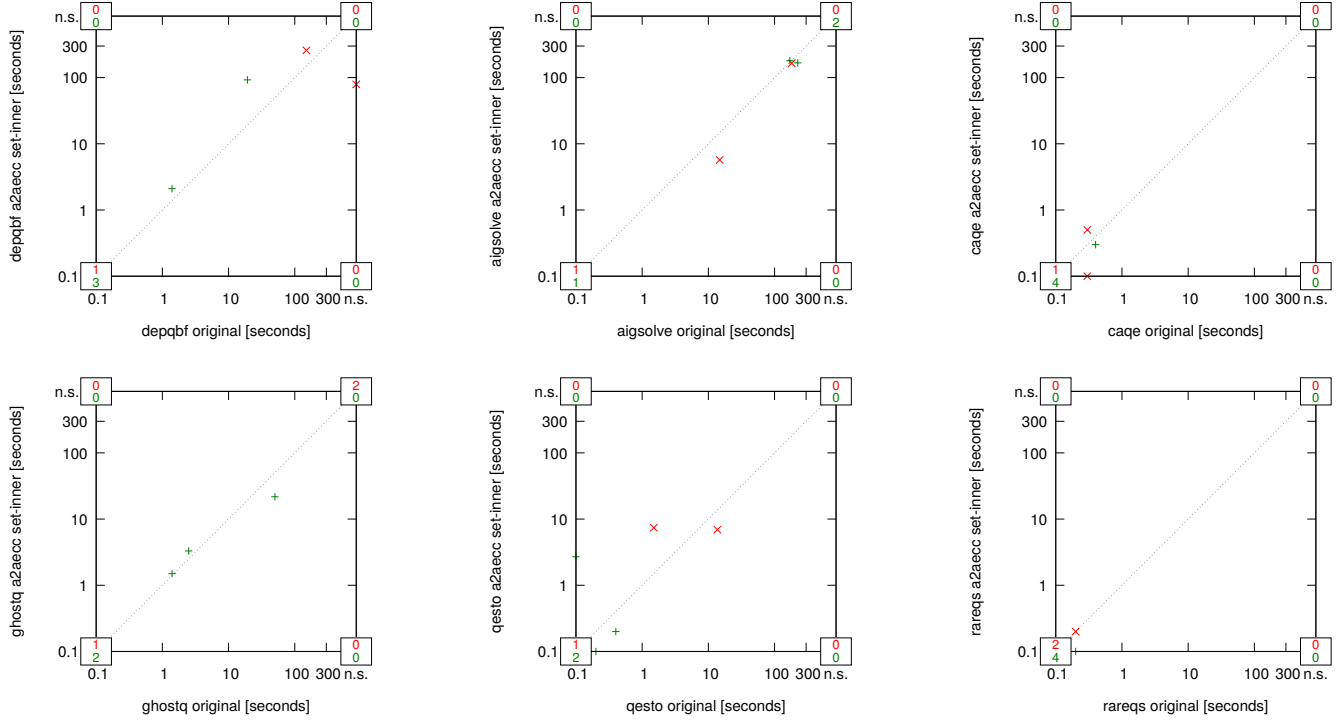


Fig. 1581: Suite Ling ($n = 8$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

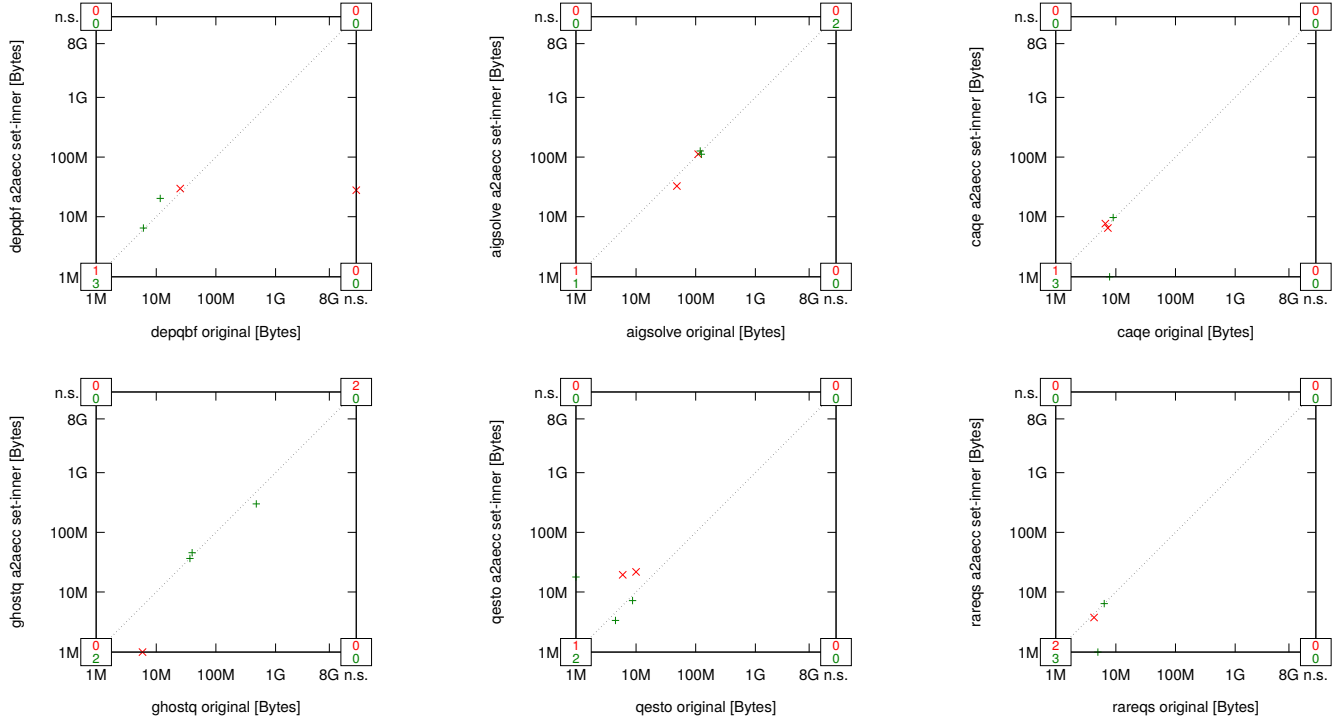


Fig. 1582: Suite Ling ($n = 8$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

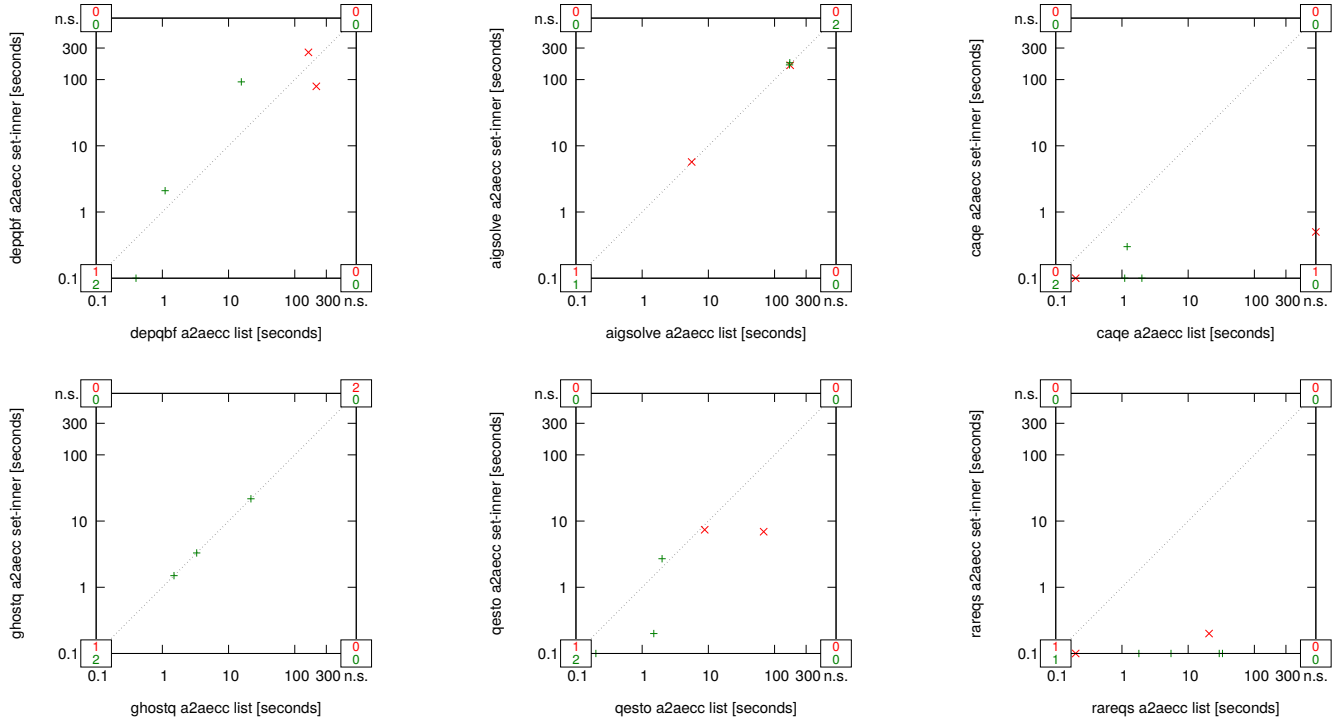


Fig. 1583: Suite Ling ($n = 8$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

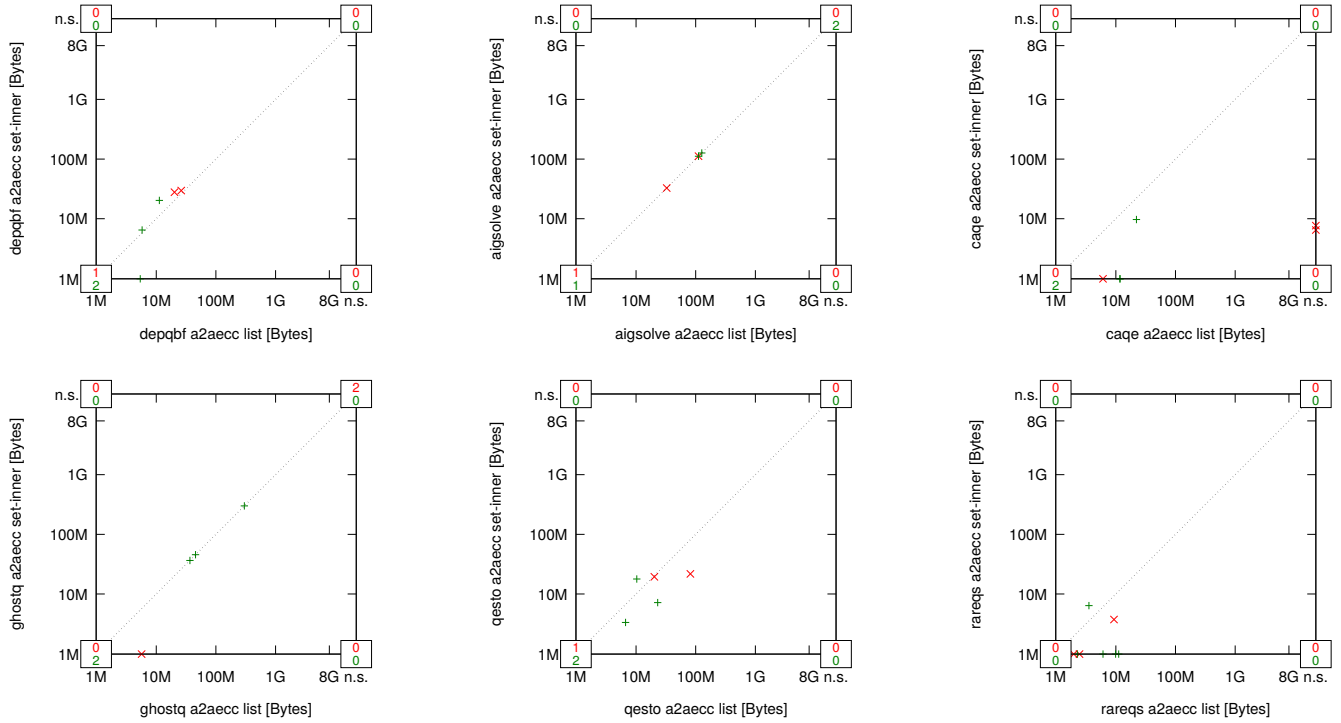


Fig. 1584: Suite Ling ($n = 8$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

27) Mangassarian-Veneris ($n = 170$):

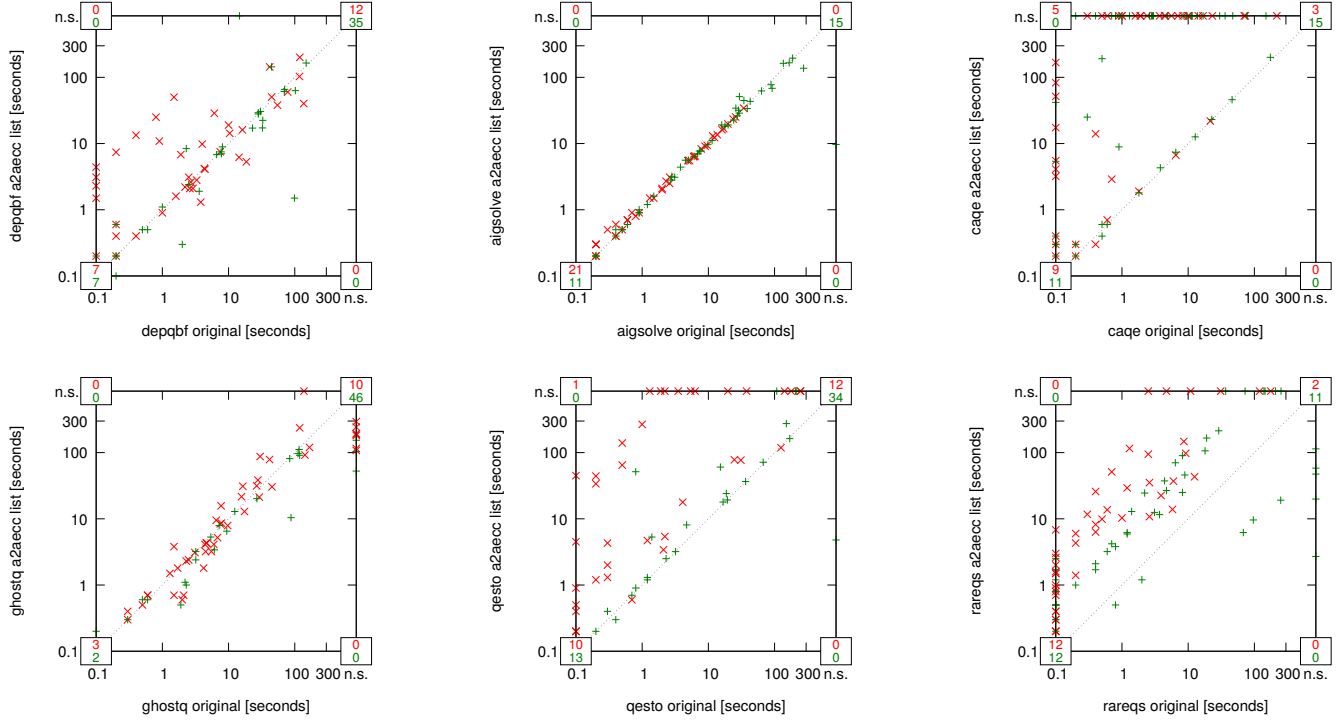


Fig. 1585: Suite Mangassarian-Veneris ($n = 170$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

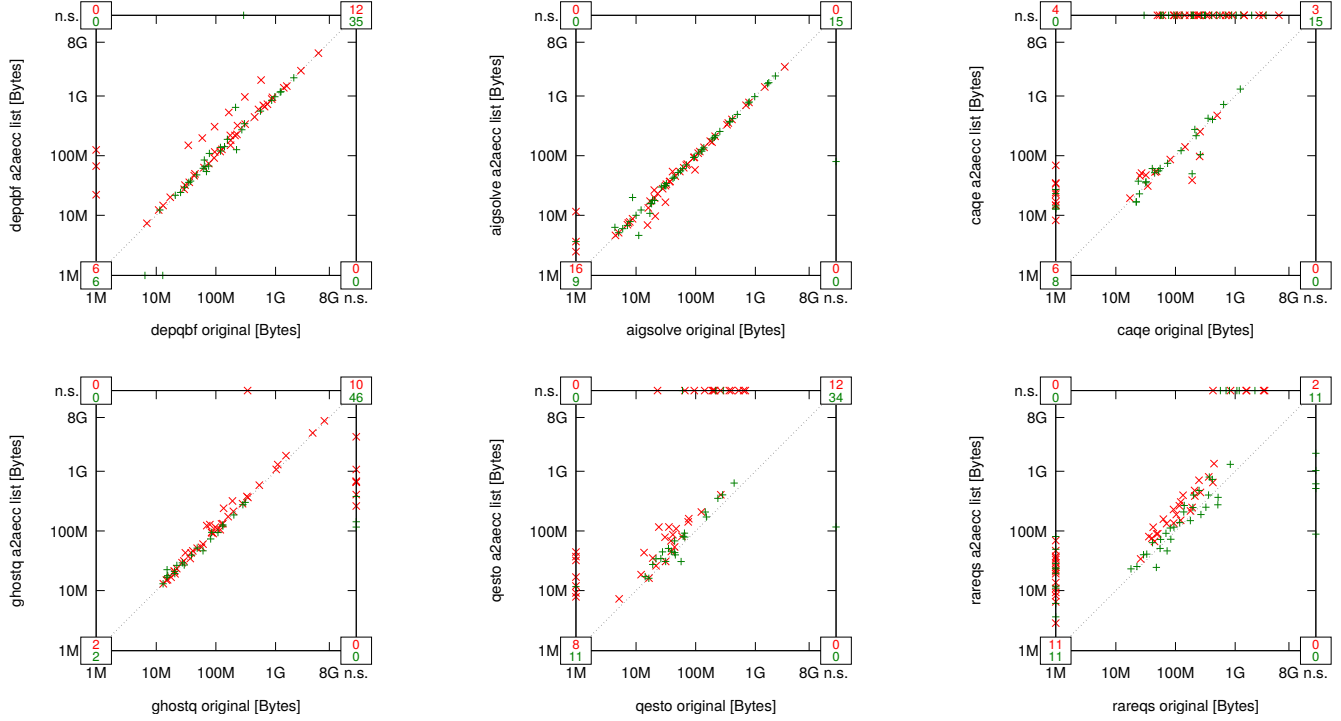


Fig. 1586: Suite Mangassarian-Veneris ($n = 170$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

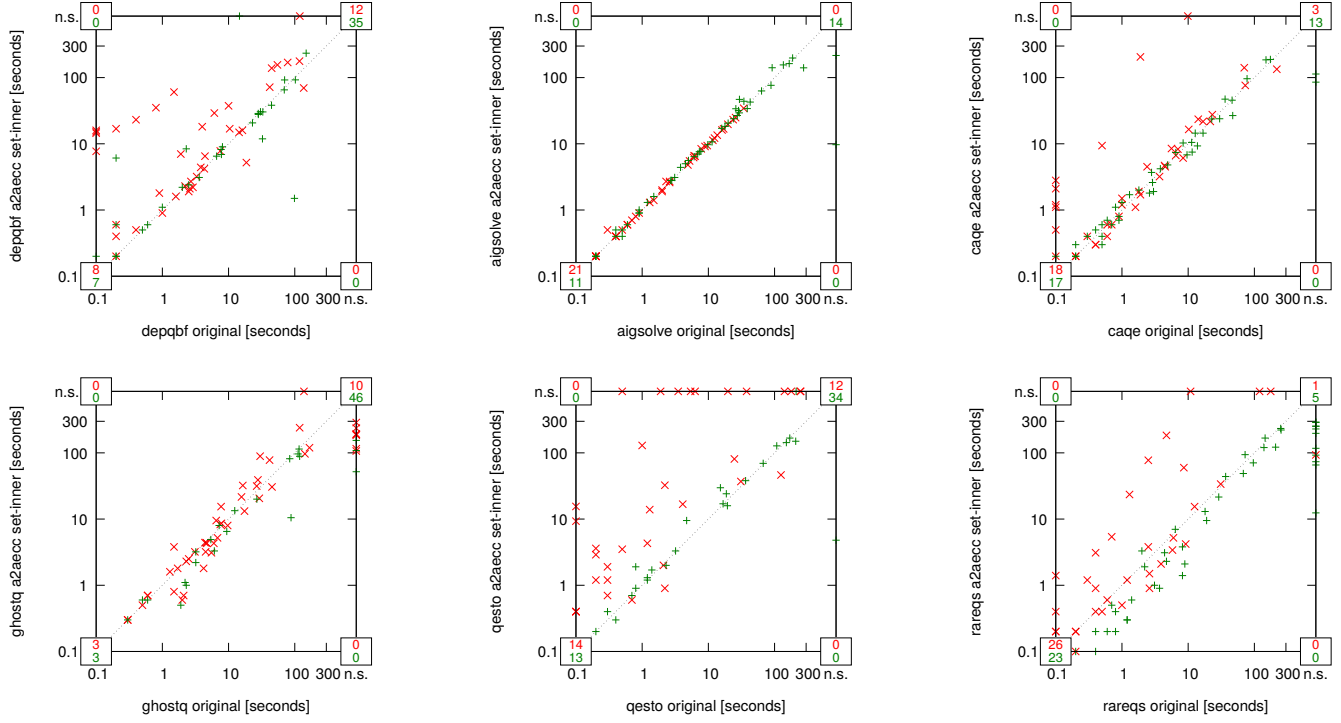


Fig. 1587: Suite Mangassarian-Veneris ($n = 170$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

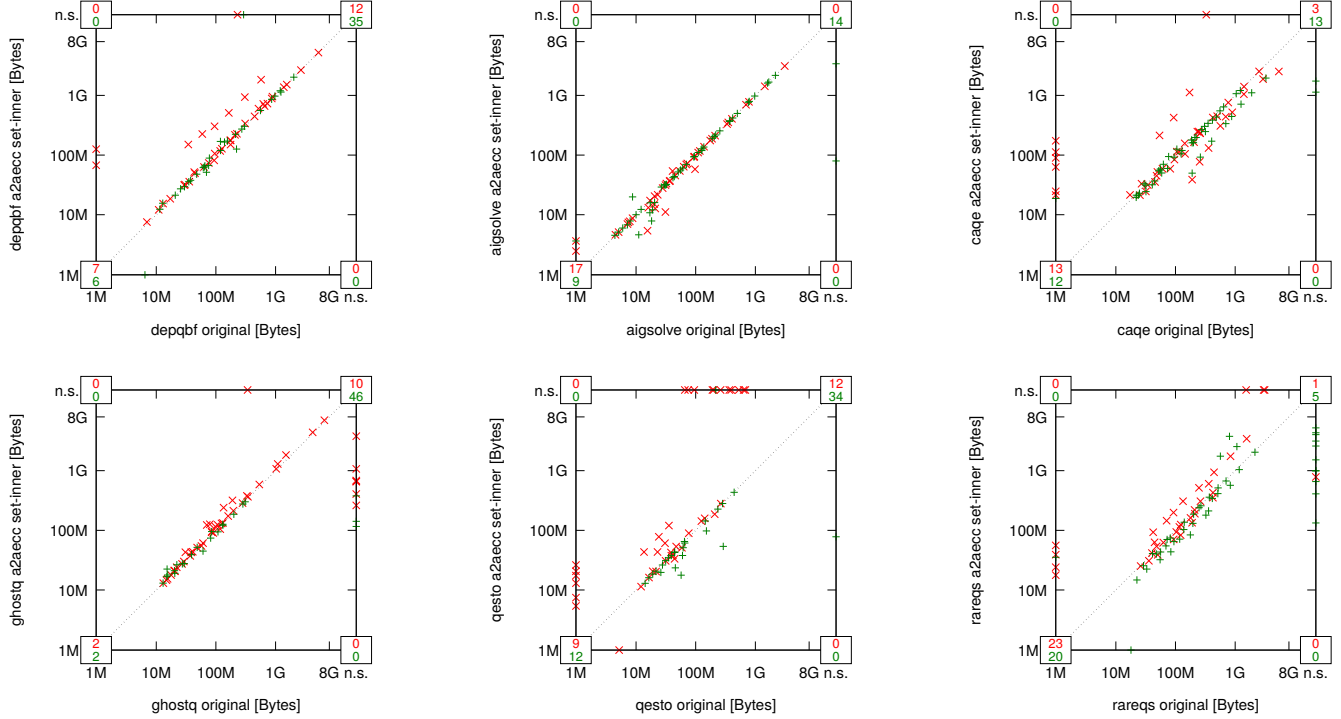


Fig. 1588: Suite Mangassarian-Veneris ($n = 170$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

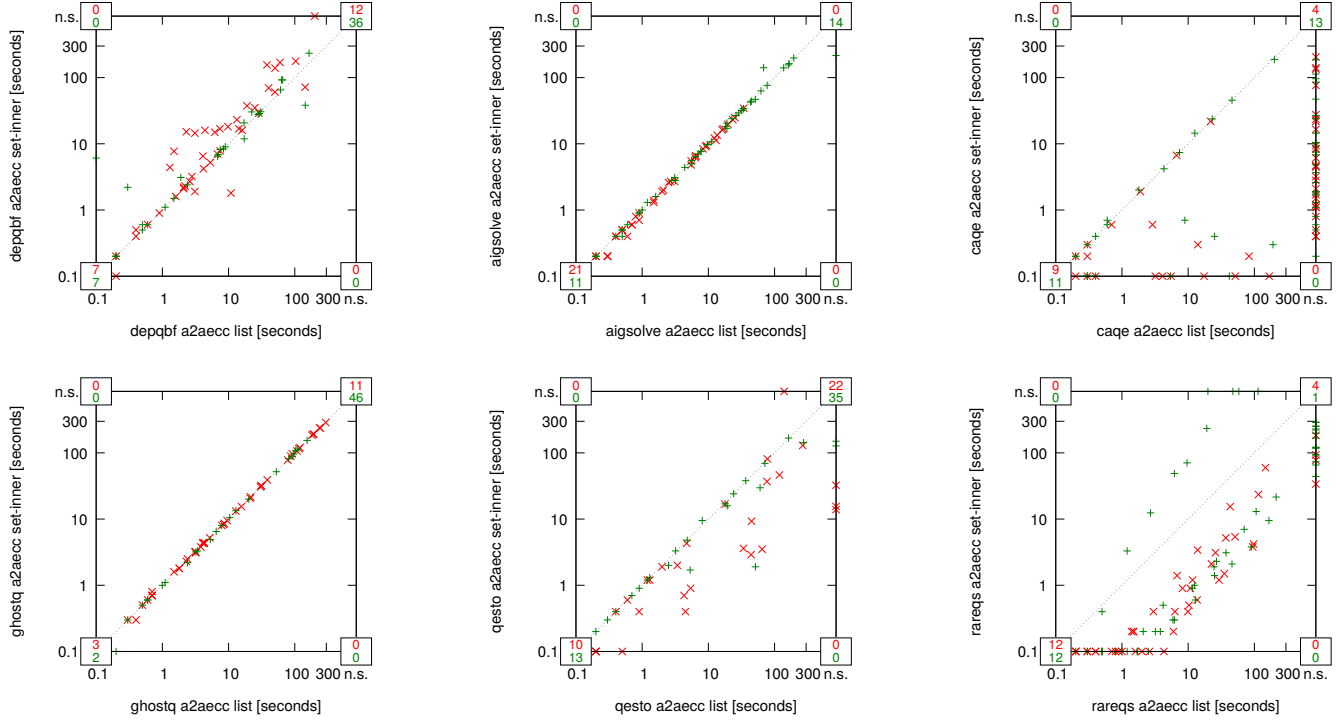


Fig. 1589: Suite Mangassarian-Veneris ($n = 170$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

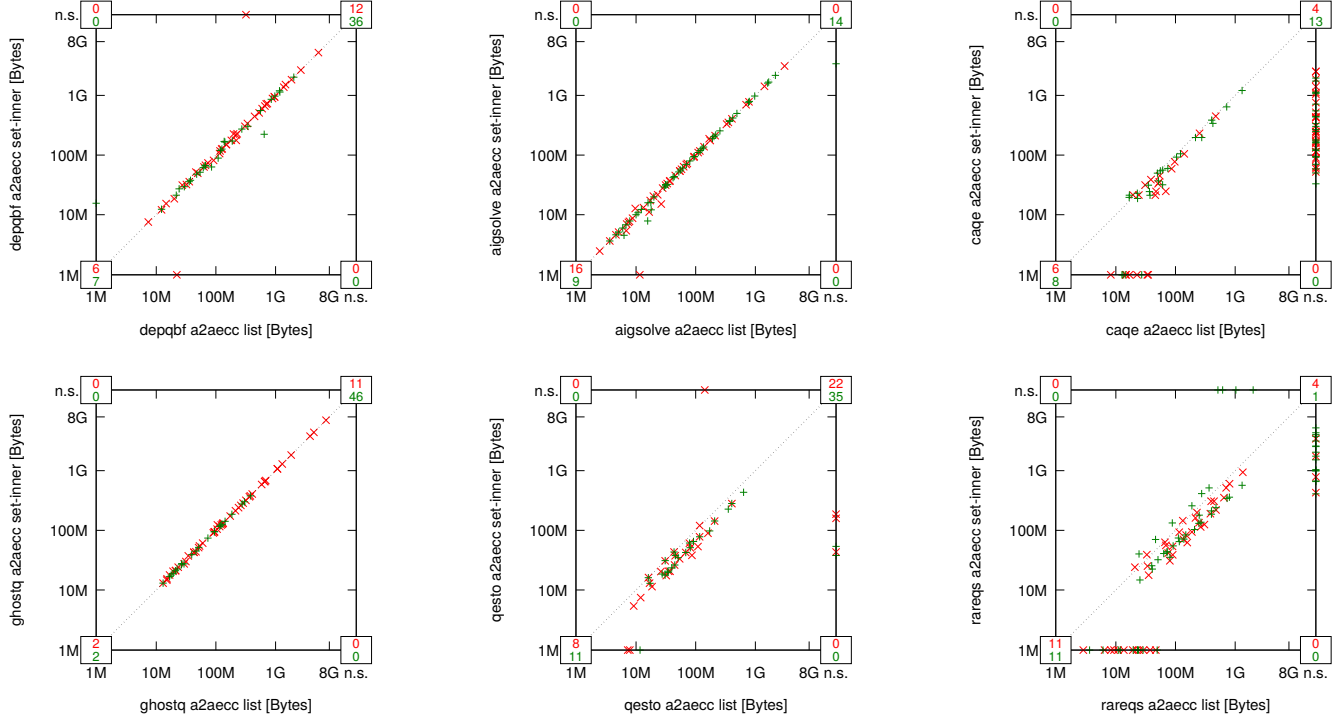


Fig. 1590: Suite Mangassarian-Veneris ($n = 170$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

28) MayerEichberger-Saffidine ($n = 113$):

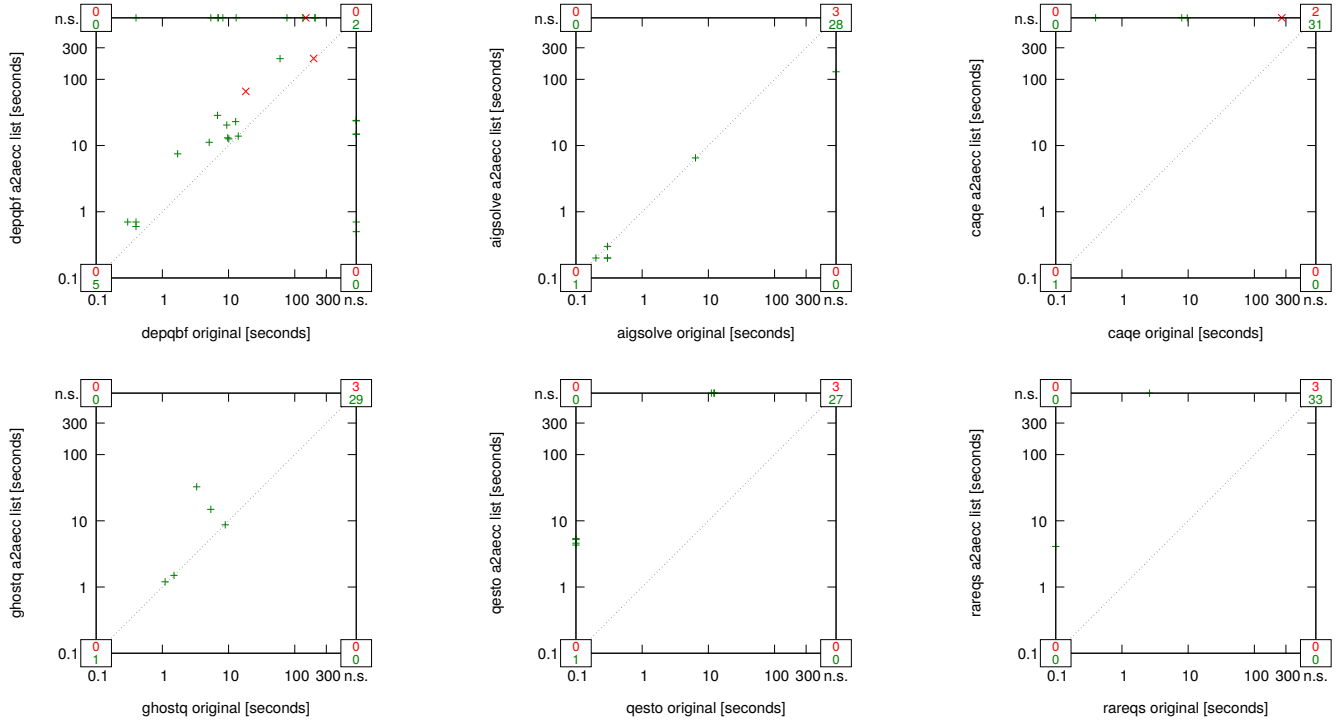


Fig. 1591: Suite MayerEichberger-Saffidine ($n = 113$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

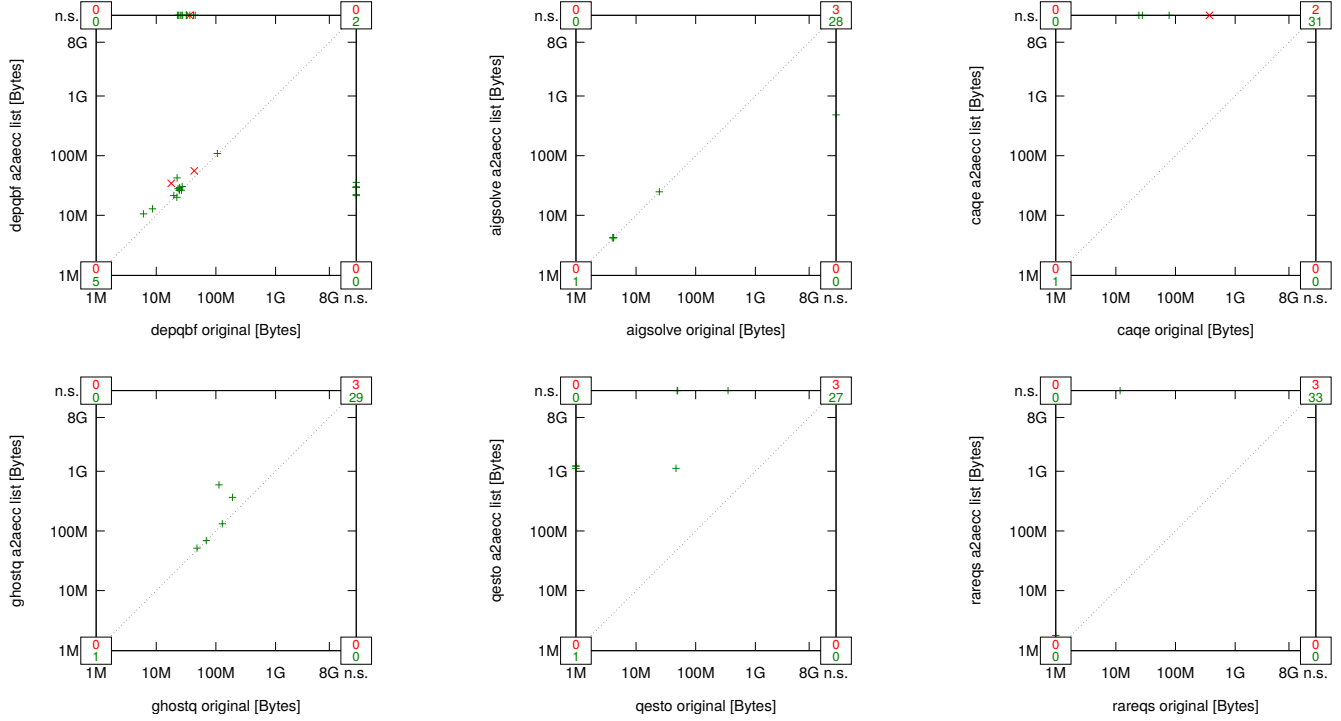


Fig. 1592: Suite MayerEichberger-Saffidine ($n = 113$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

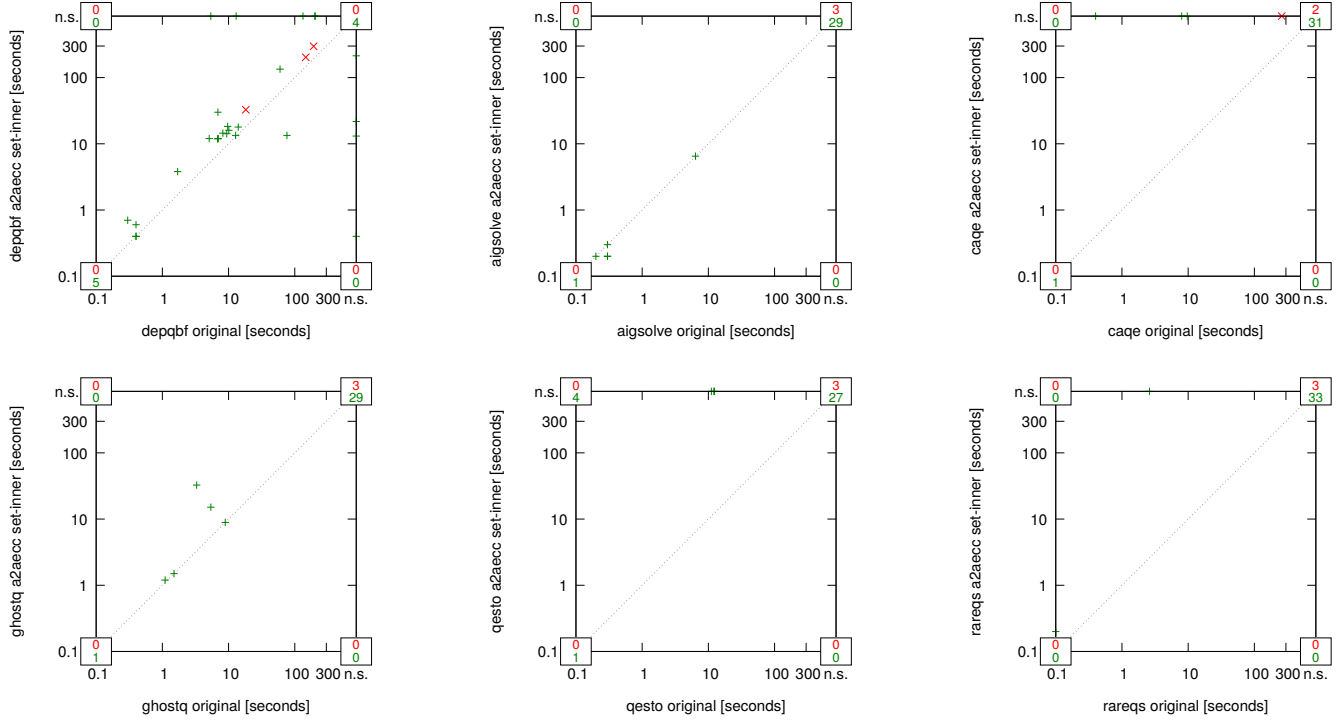


Fig. 1593: Suite MayerEichberger-Saffidine ($n = 113$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

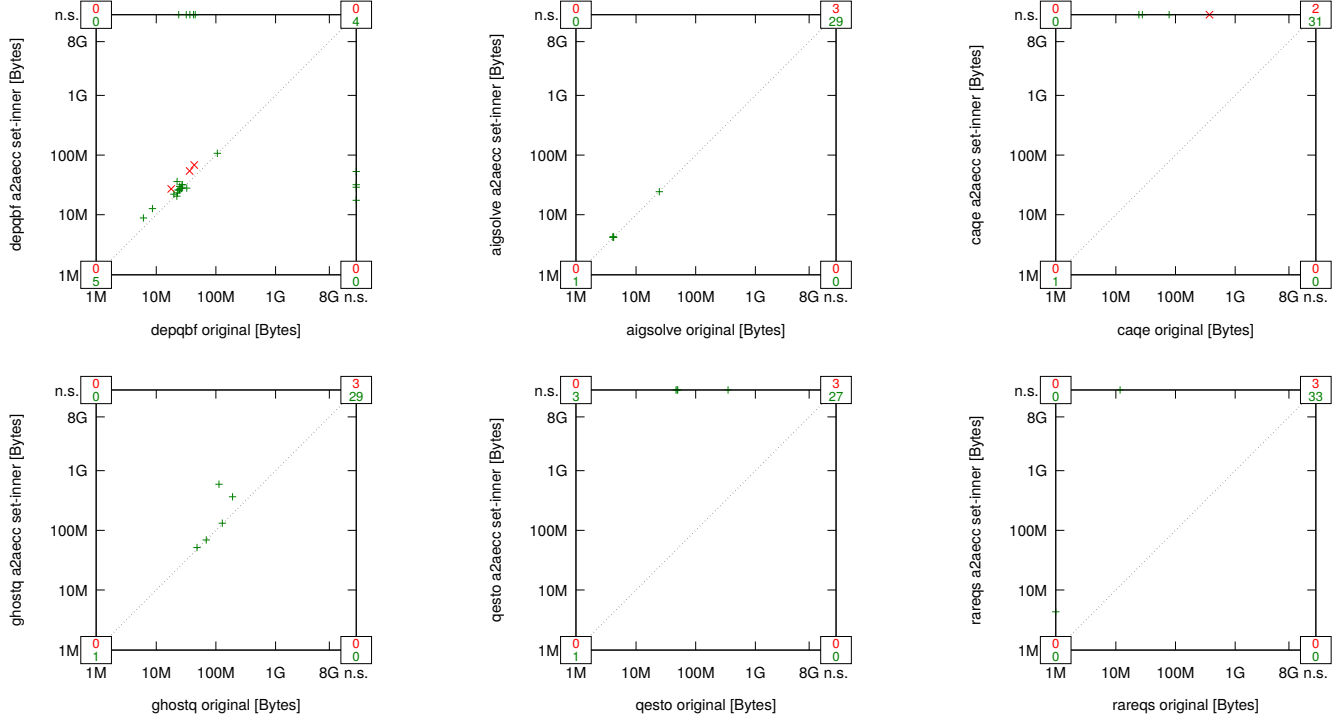


Fig. 1594: Suite MayerEichberger-Saffidine ($n = 113$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

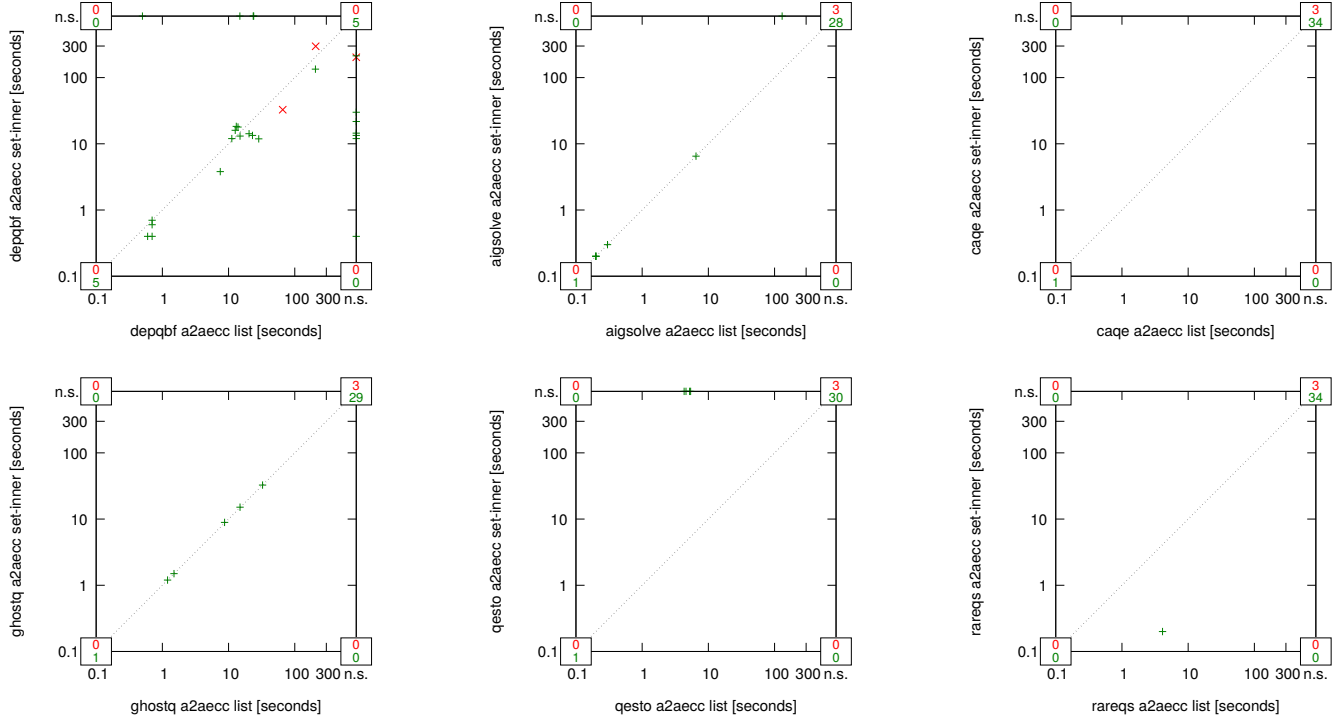


Fig. 1595: Suite MayerEichberger-Saffidine ($n = 113$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

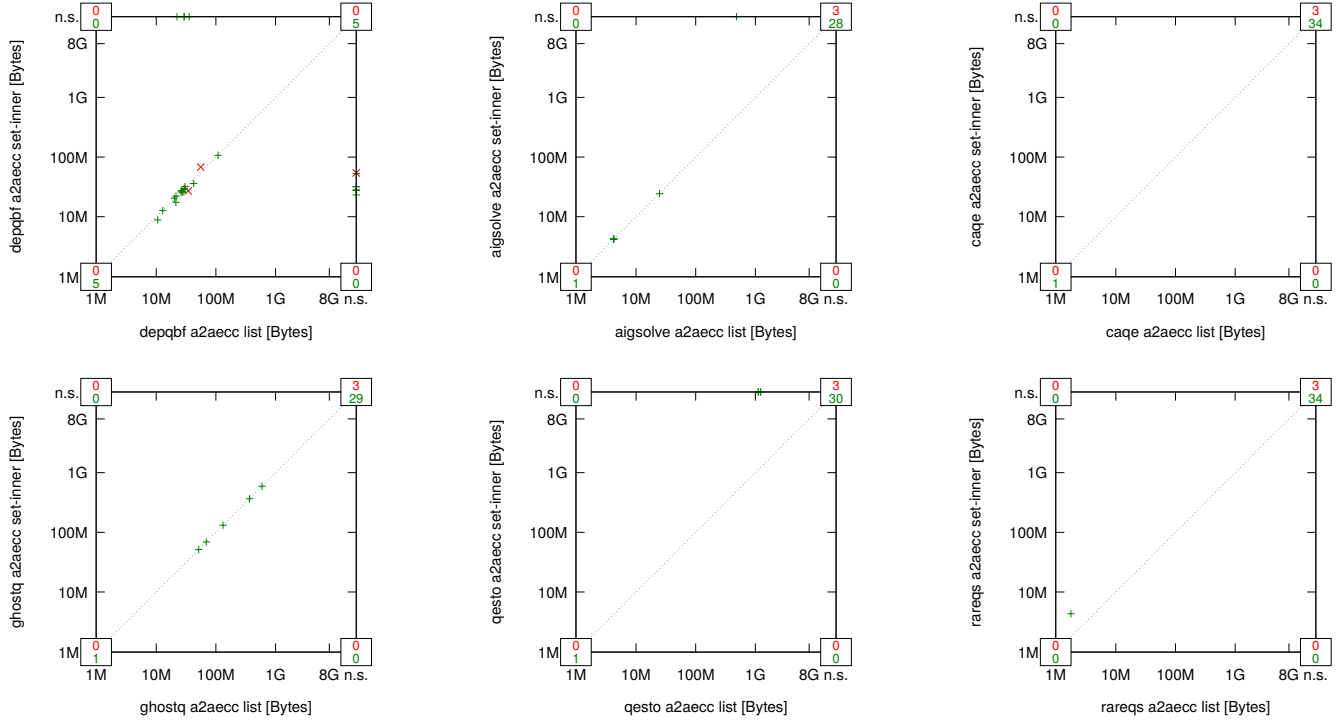


Fig. 1596: Suite MayerEichberger-Saffidine ($n = 113$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

29) *Messinger* ($n = 63$):

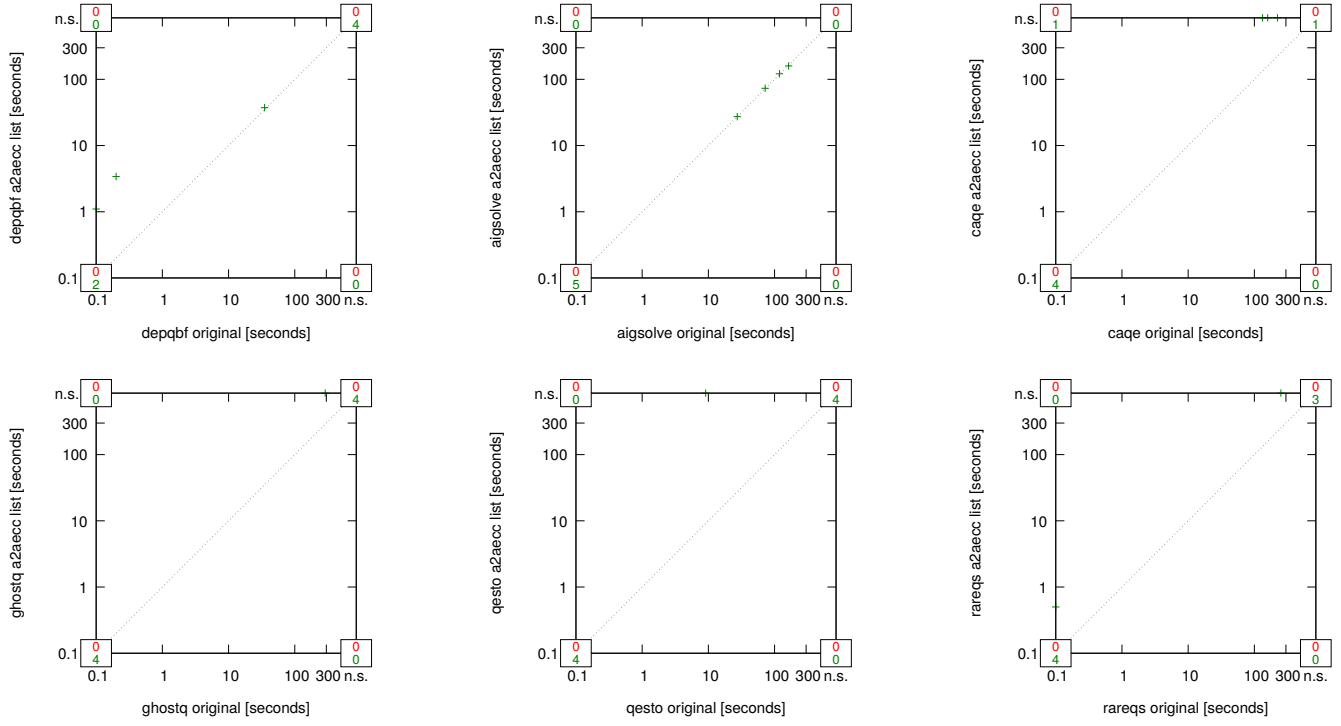


Fig. 1597: Suite Messinger ($n = 63$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

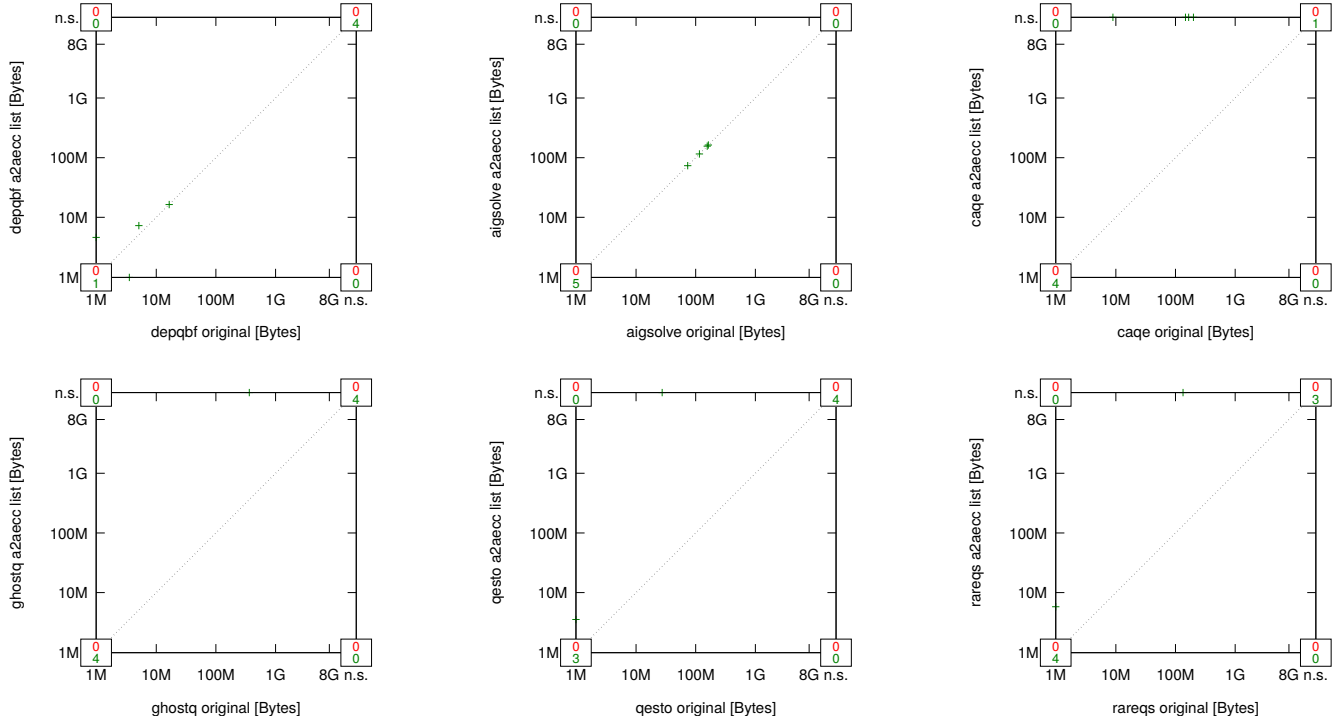


Fig. 1598: Suite Messinger ($n = 63$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

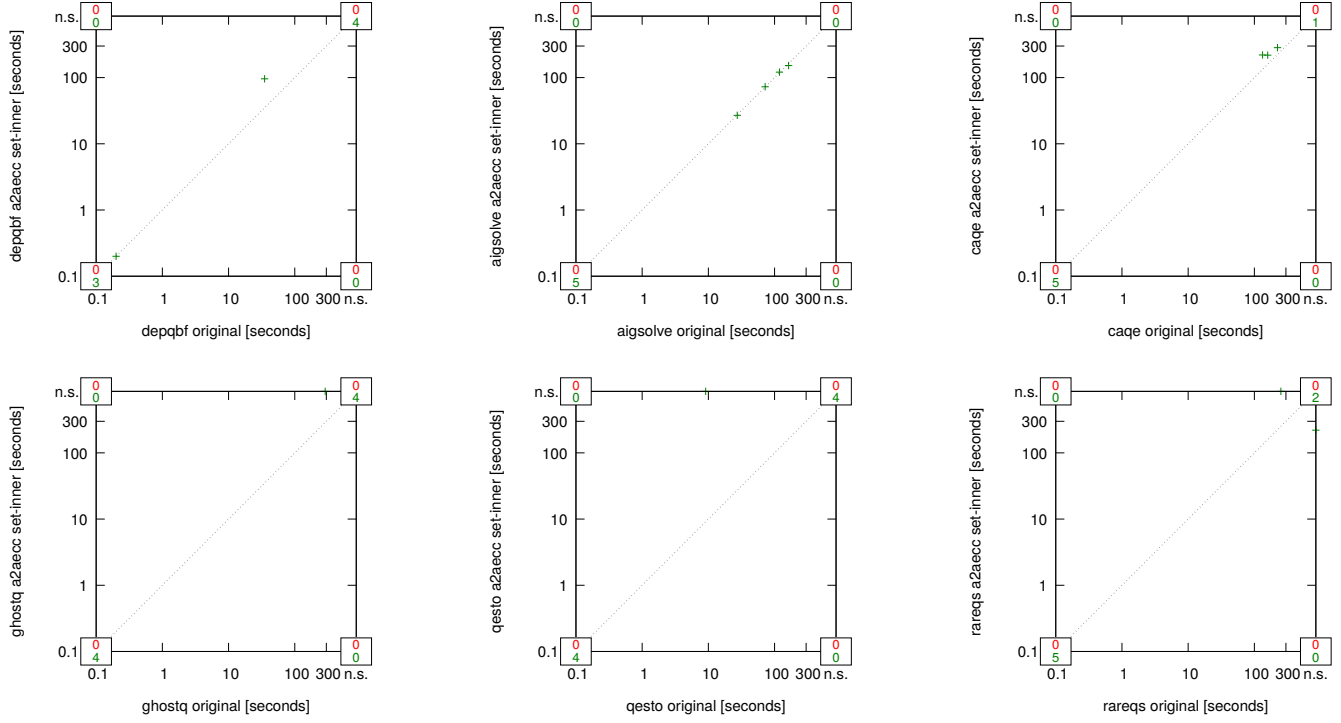


Fig. 1599: Suite Messenger ($n = 63$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

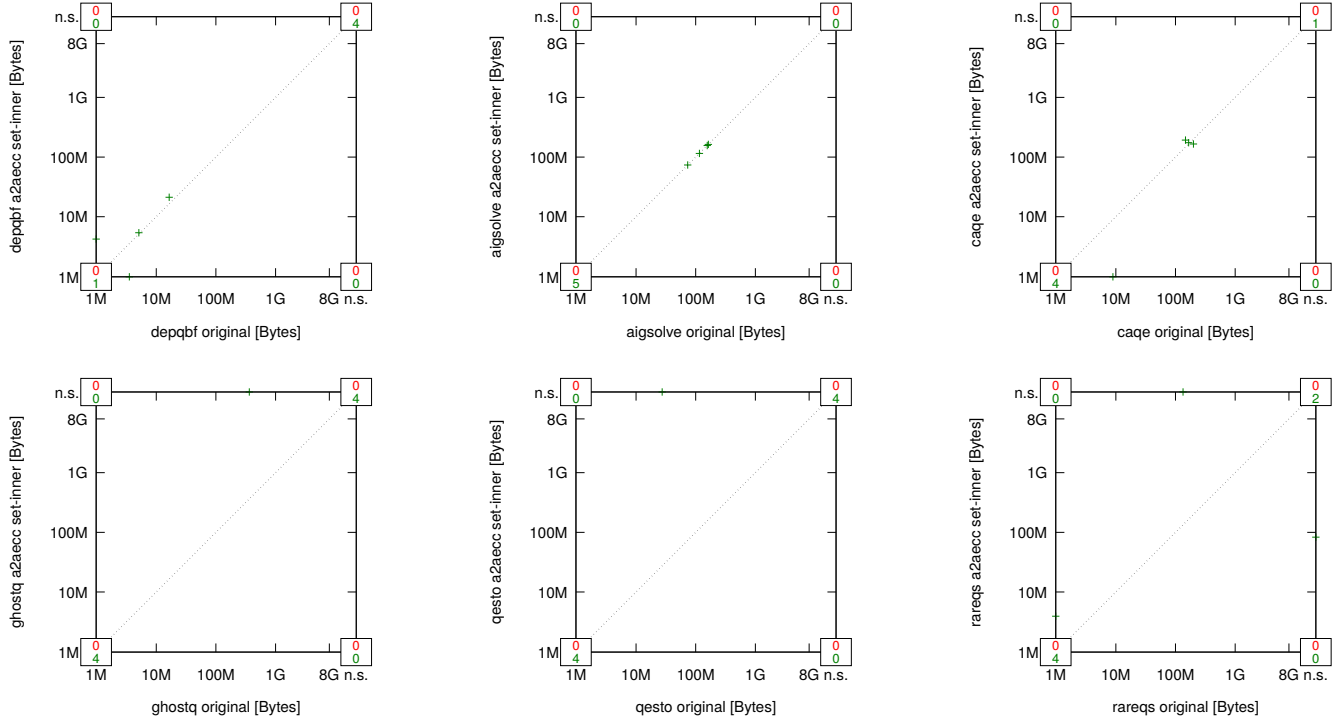


Fig. 1600: Suite Messenger ($n = 63$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

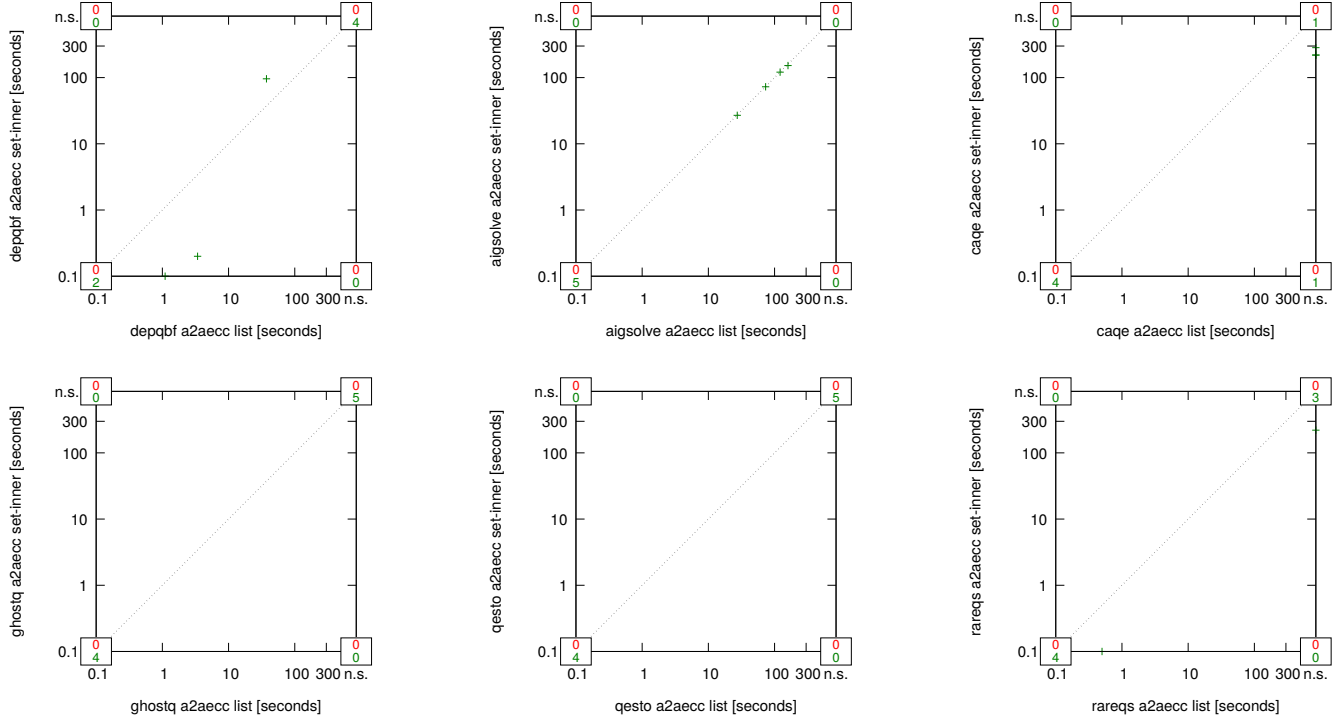


Fig. 1601: Suite Messenger ($n = 63$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

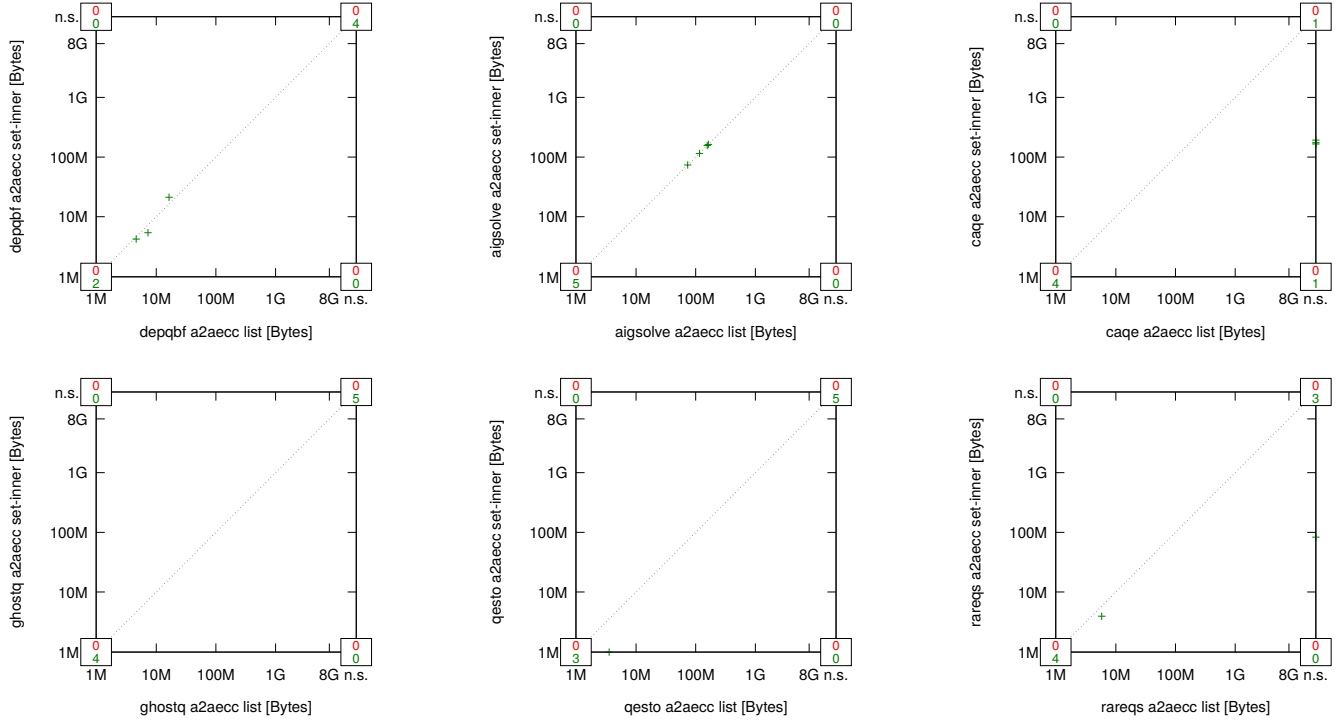


Fig. 1602: Suite Messenger ($n = 63$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

30) Miller-Marin ($n = 194$):

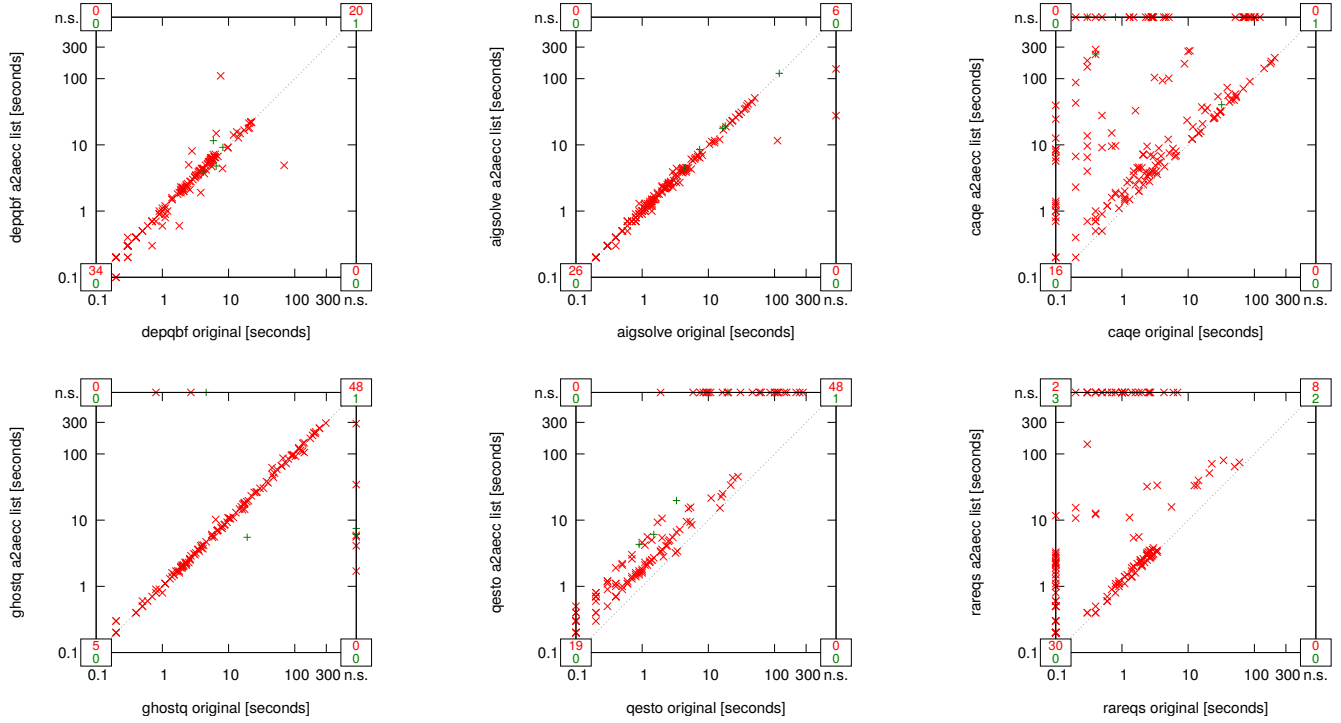


Fig. 1603: Suite Miller-Marin ($n = 194$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

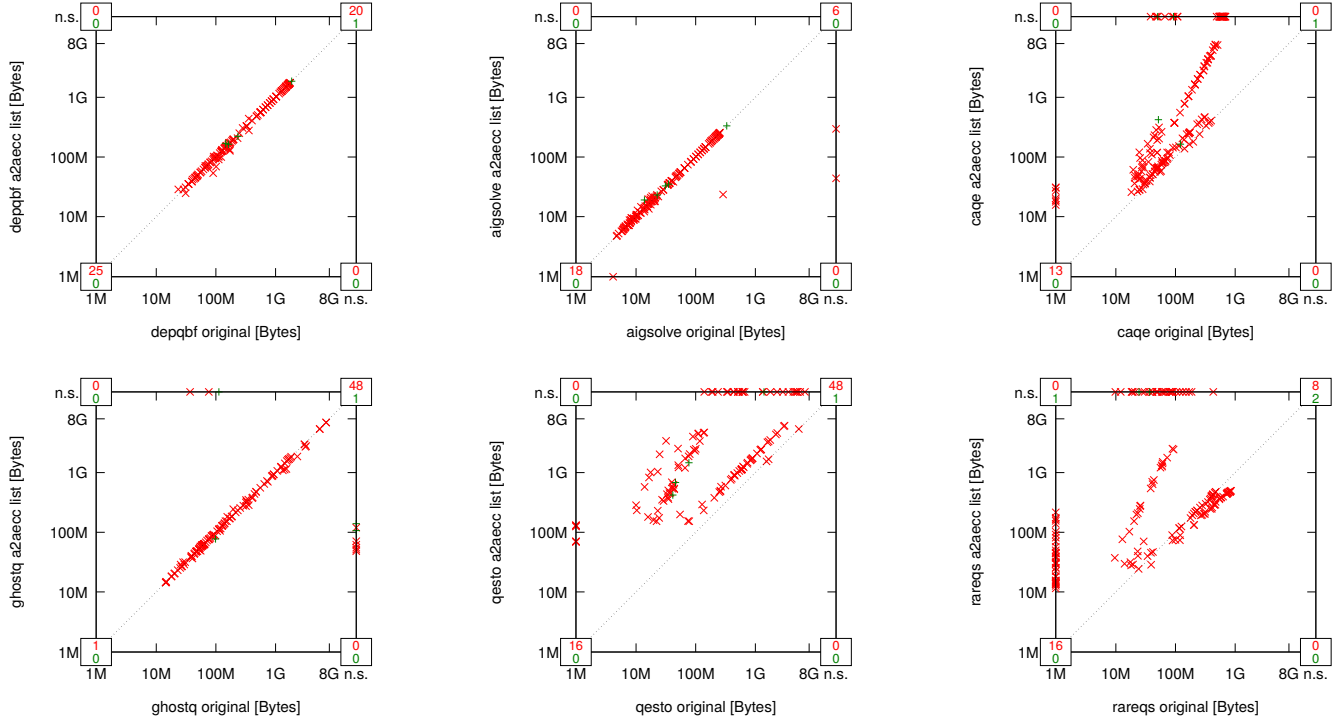


Fig. 1604: Suite Miller-Marin ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

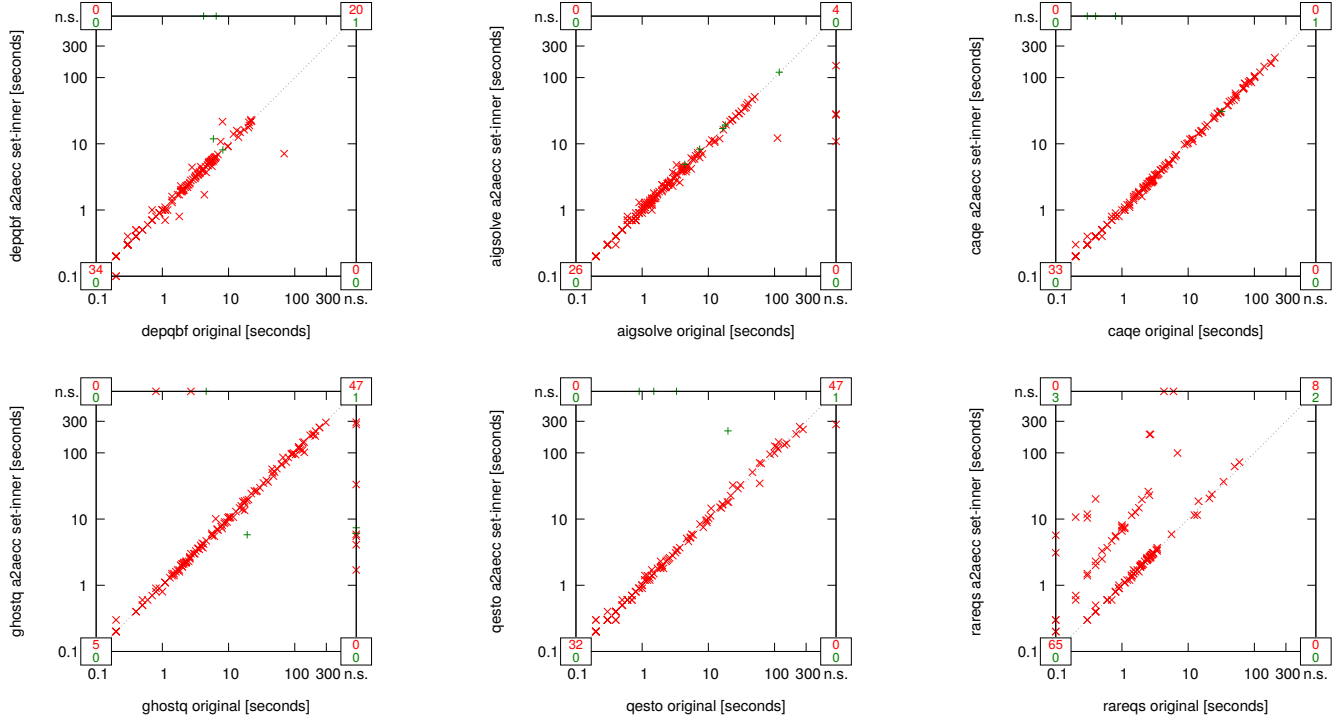


Fig. 1605: Suite Miller-Marin ($n = 194$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

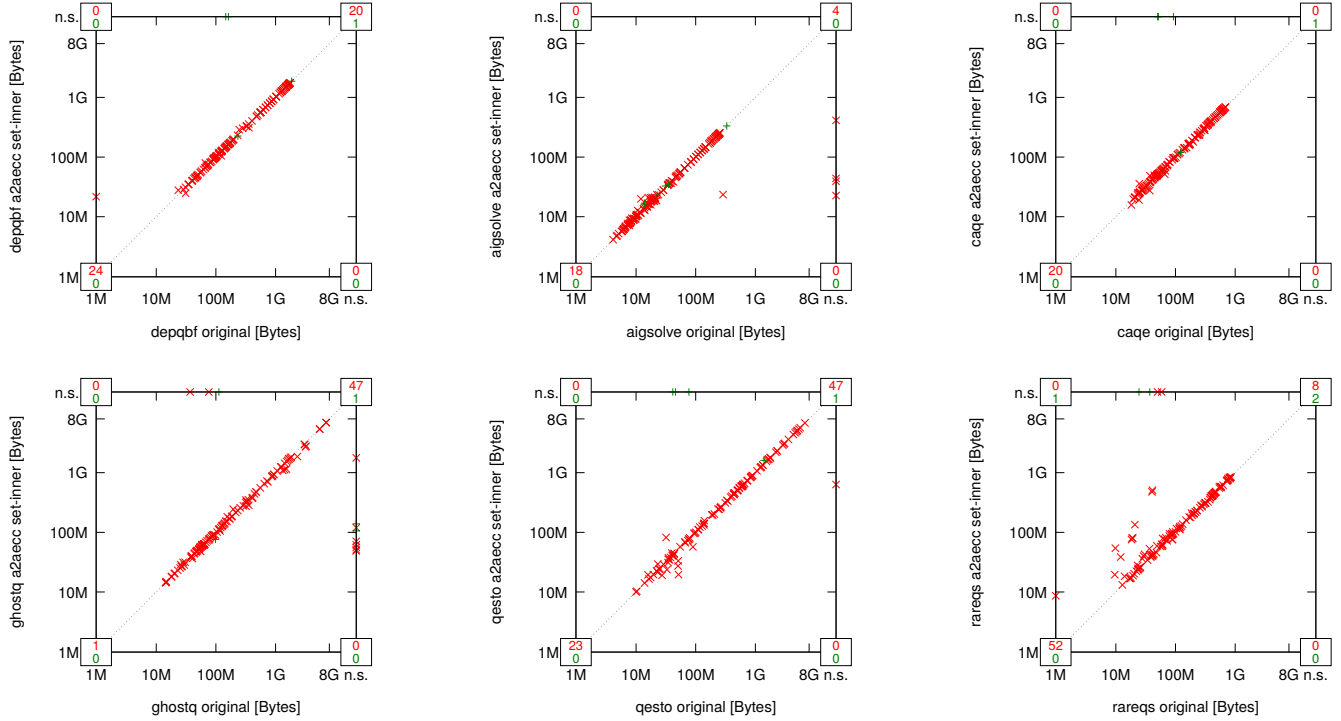


Fig. 1606: Suite Miller-Marin ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

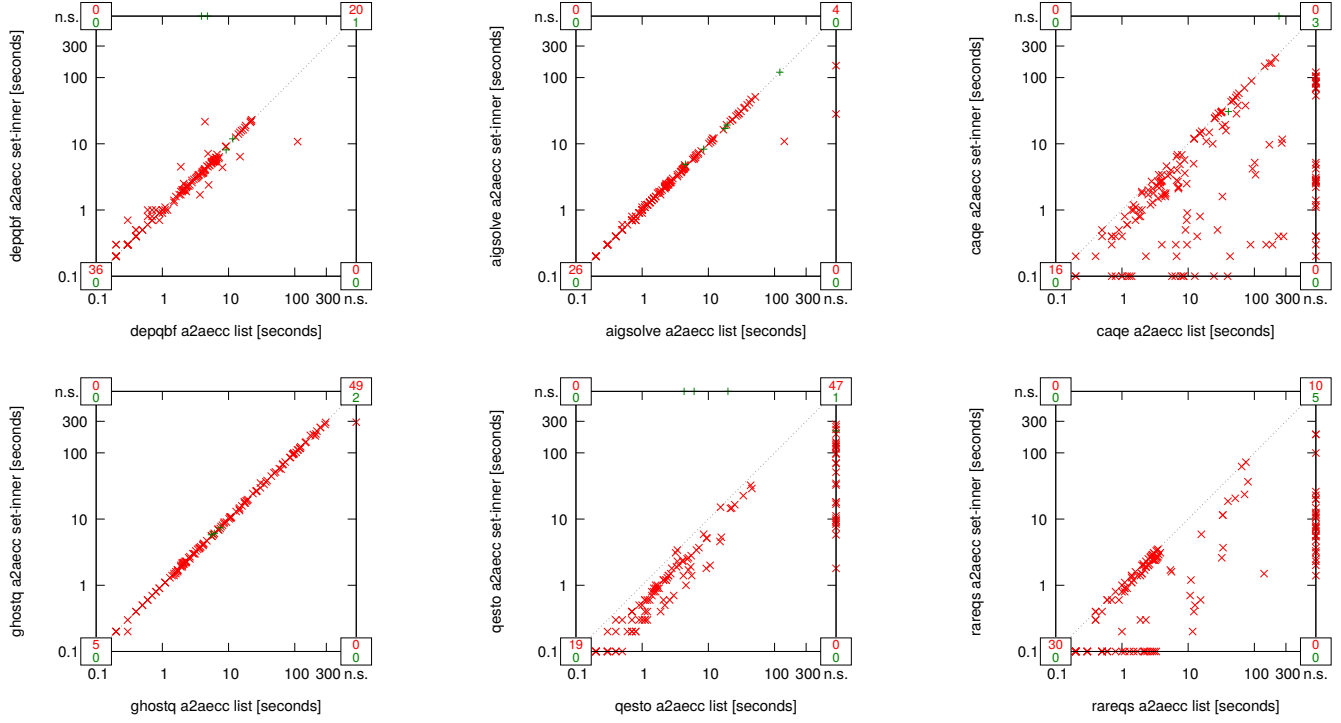


Fig. 1607: Suite Miller-Marin ($n = 194$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

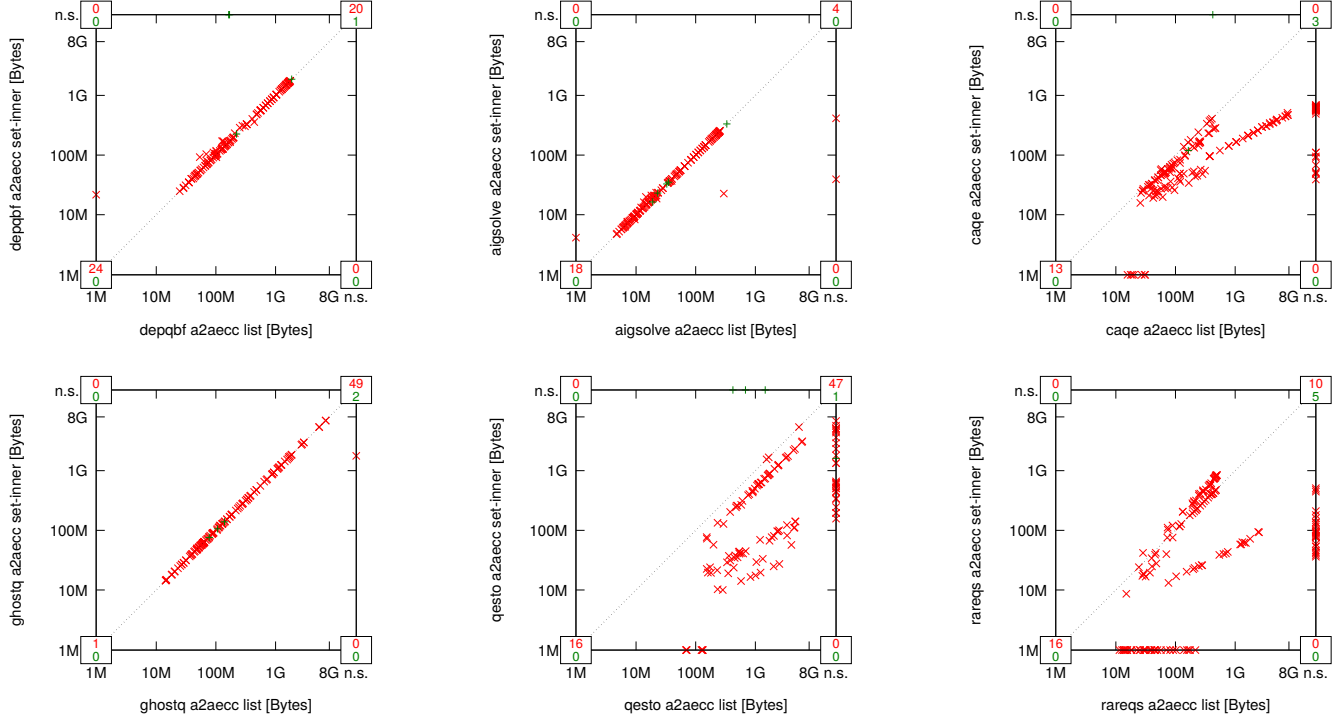


Fig. 1608: Suite Miller-Marin ($n = 194$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

31) Miller-Scholl-Becker ($n = 194$):

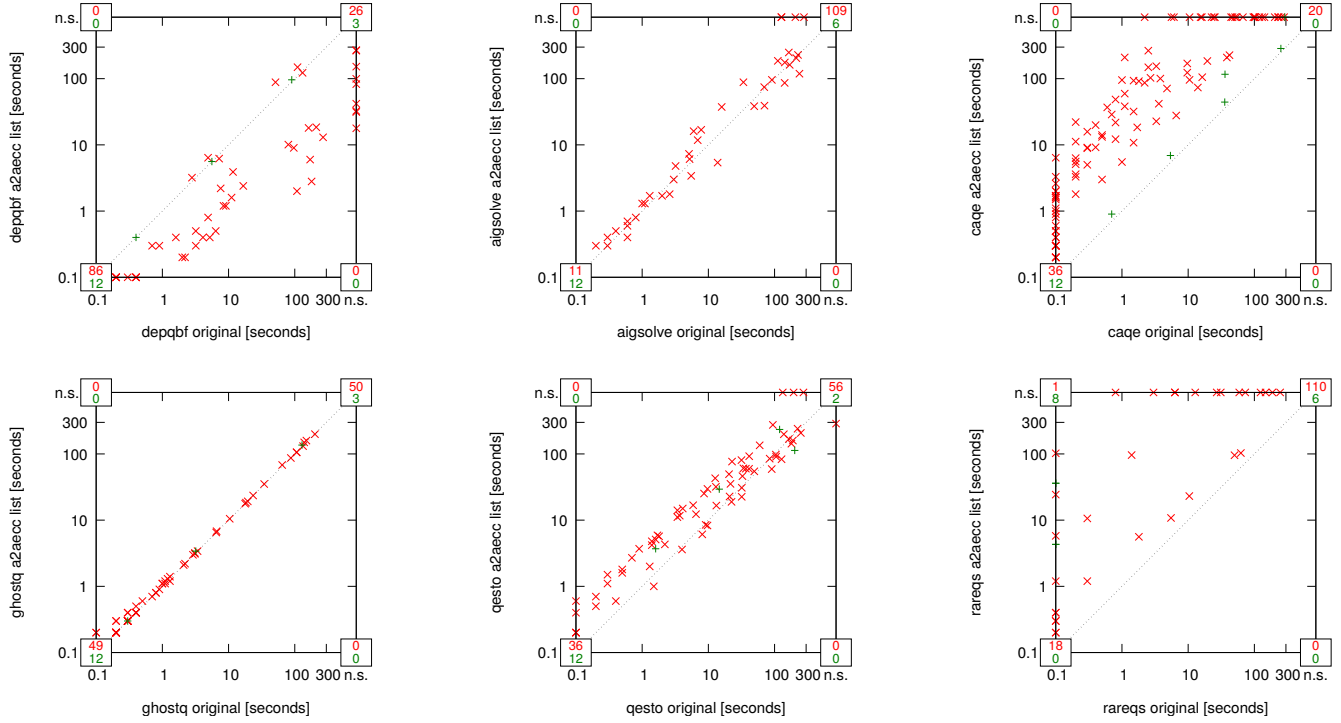


Fig. 1609: Suite Miller-Scholl-Becker ($n = 194$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

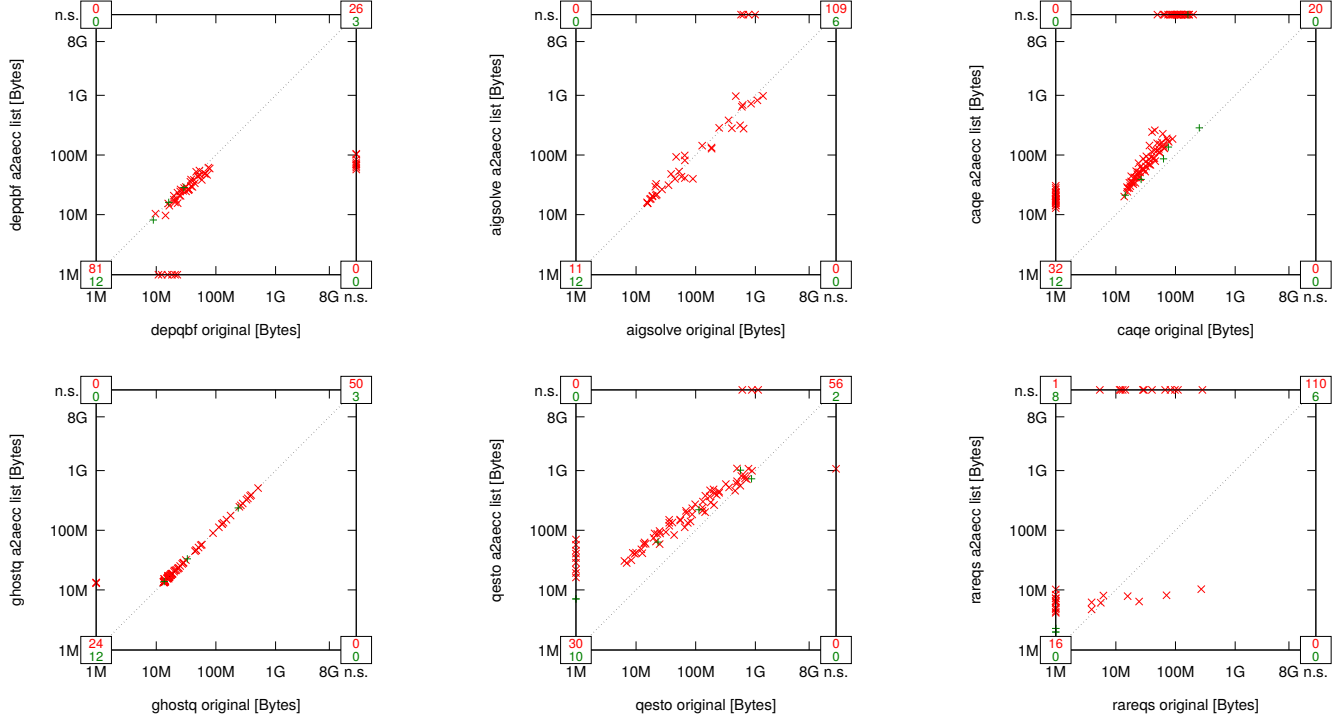


Fig. 1610: Suite Miller-Scholl-Becker ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

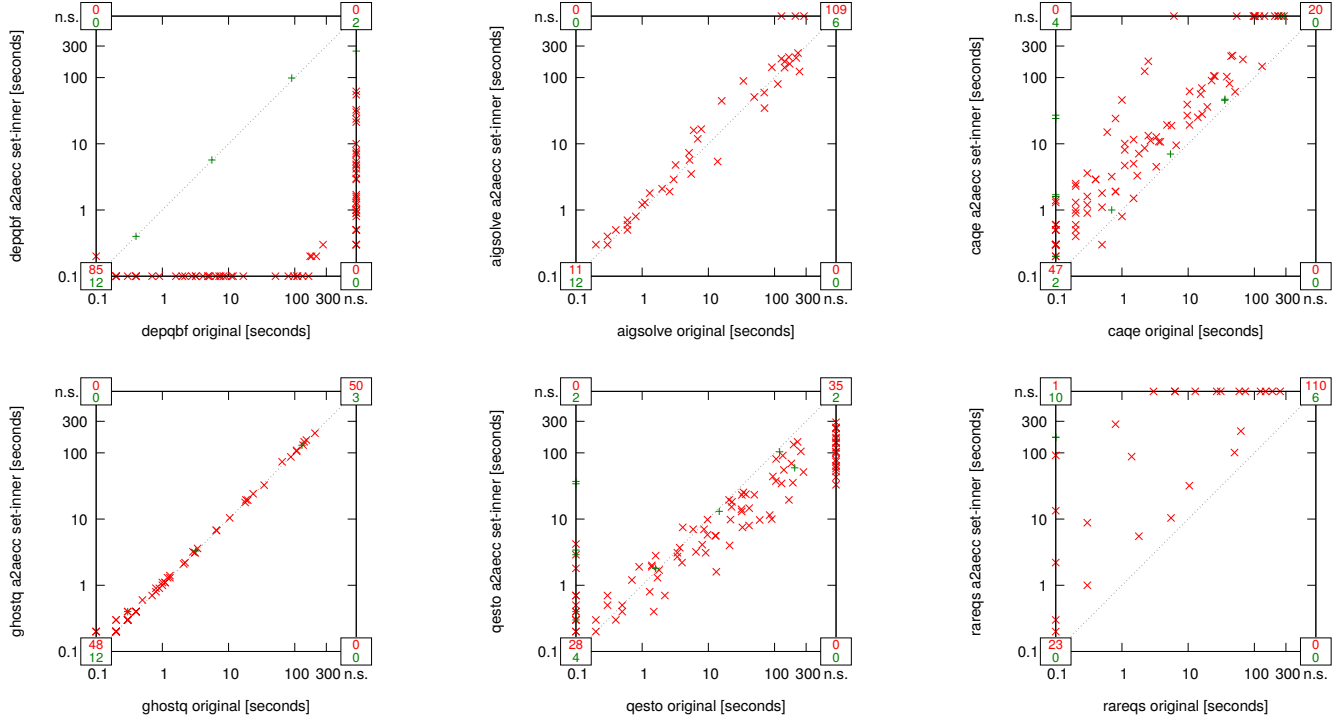


Fig. 1611: Suite Miller-Scholl-Becker ($n = 194$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

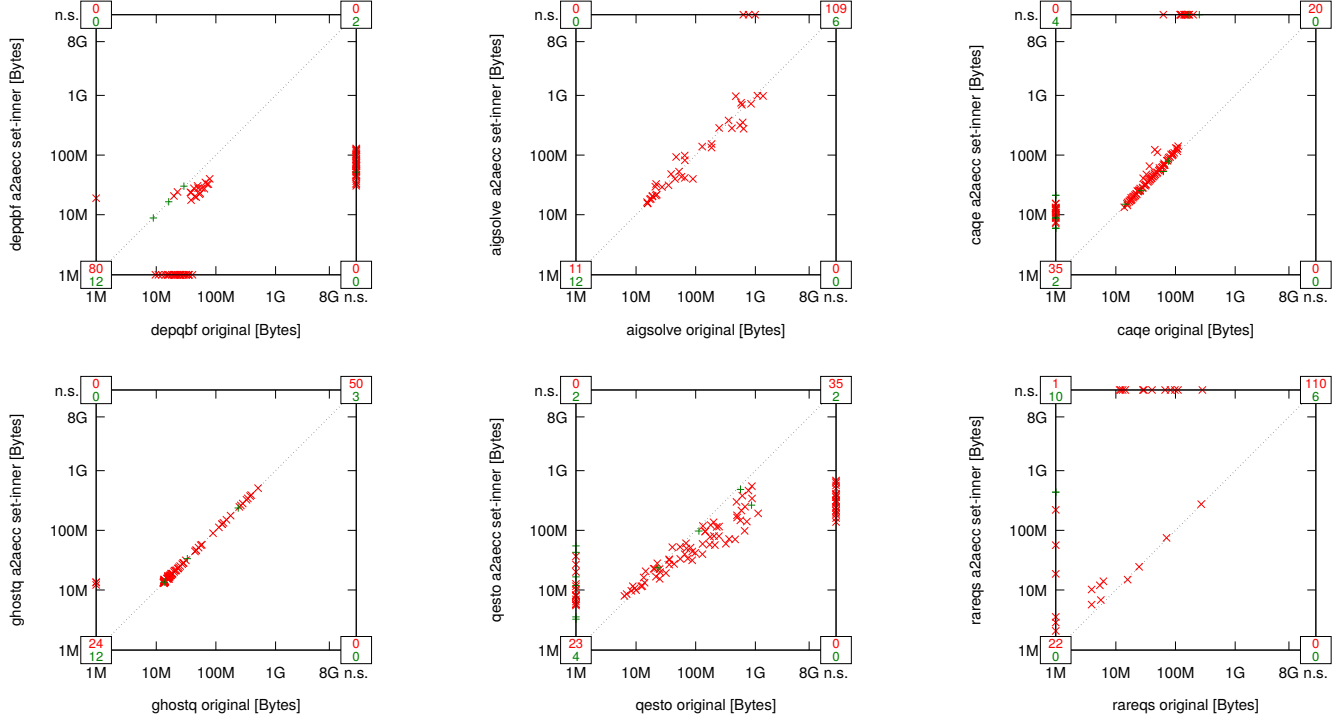


Fig. 1612: Suite Miller-Scholl-Becker ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

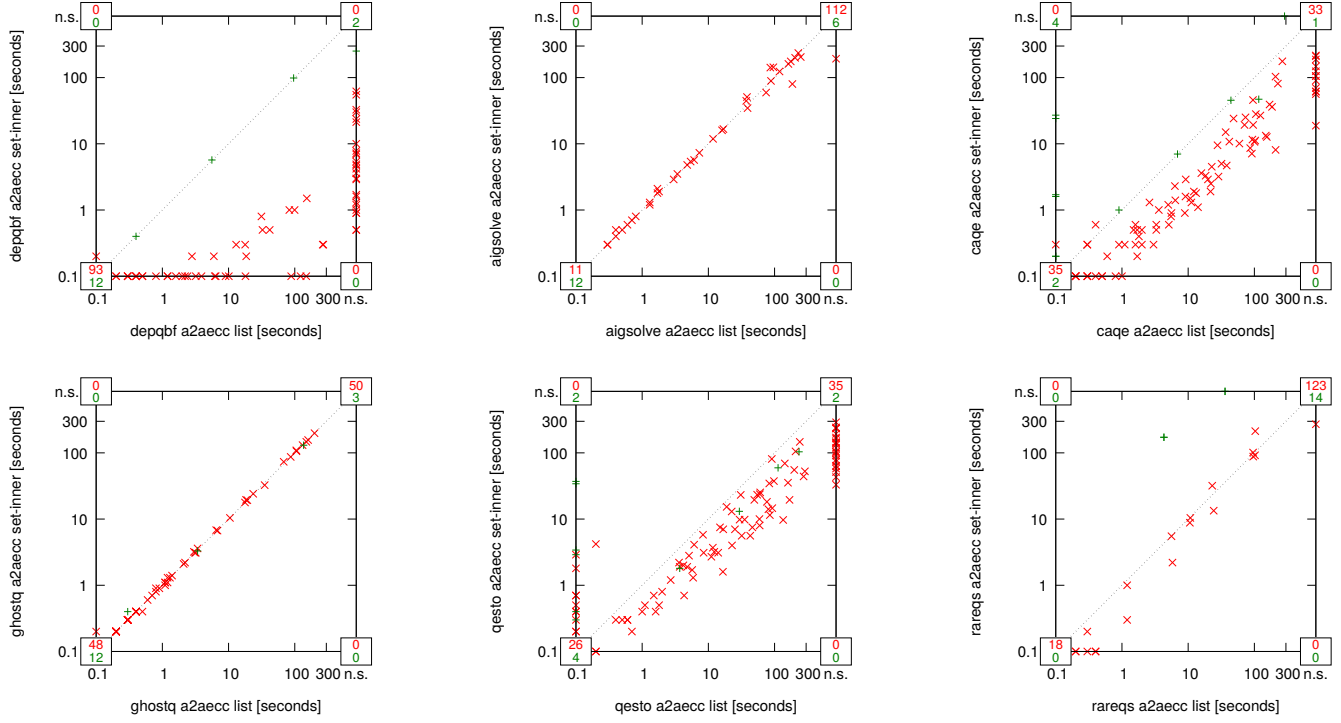


Fig. 1613: Suite Miller-Scholl-Becker ($n = 194$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

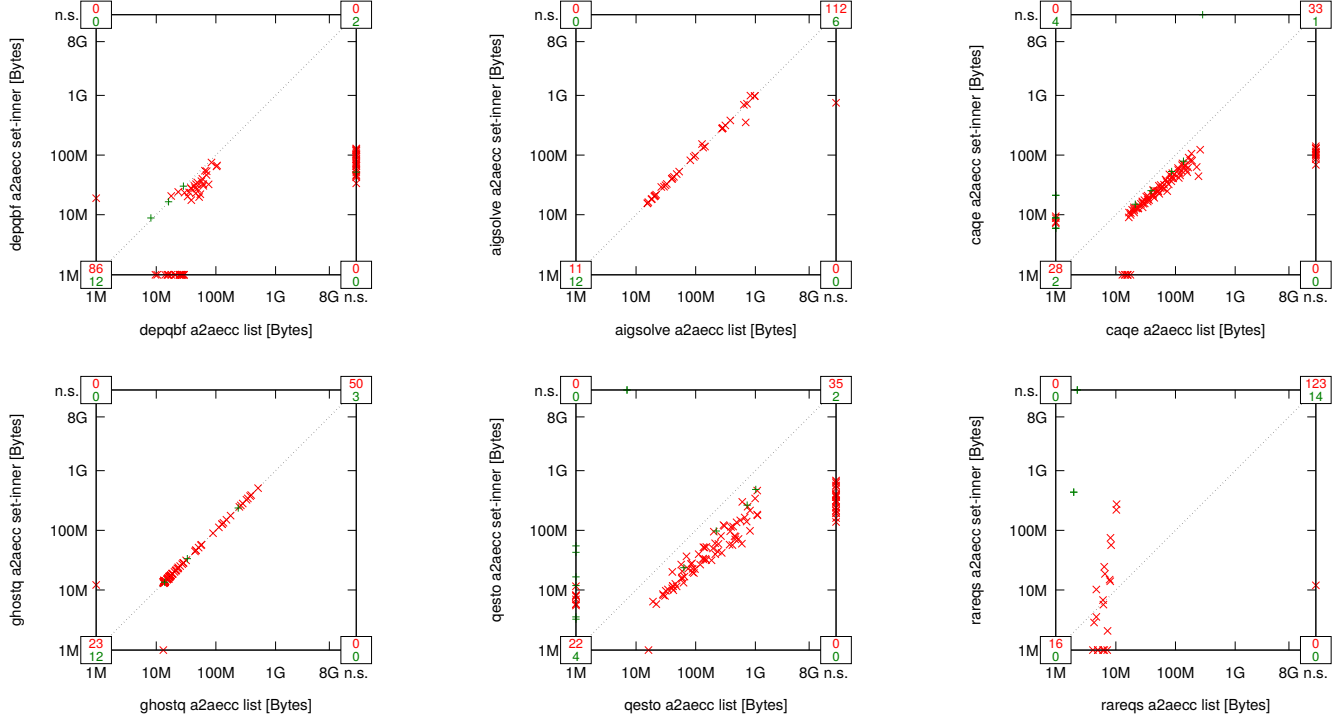


Fig. 1614: Suite Miller-Scholl-Becker ($n = 194$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

32) Mneimneh-Sakallah ($n = 180$):

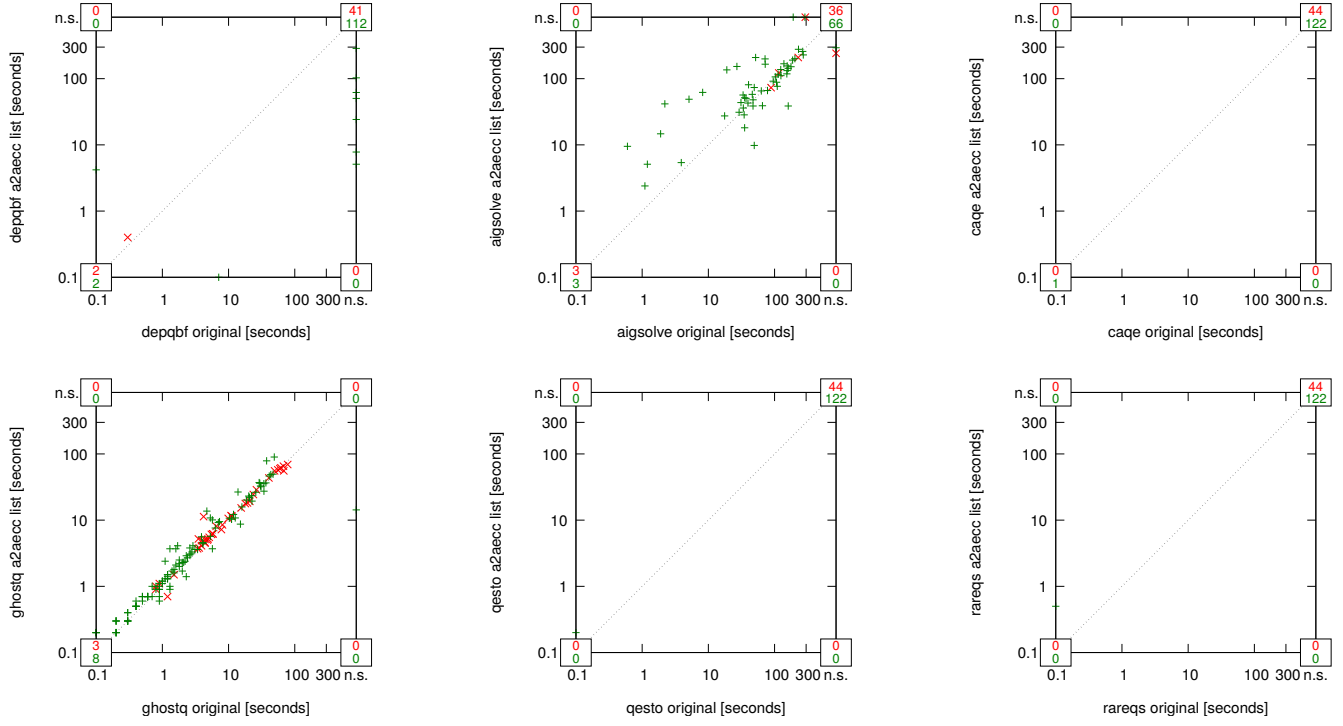


Fig. 1615: Suite Mneimneh-Sakallah ($n = 180$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

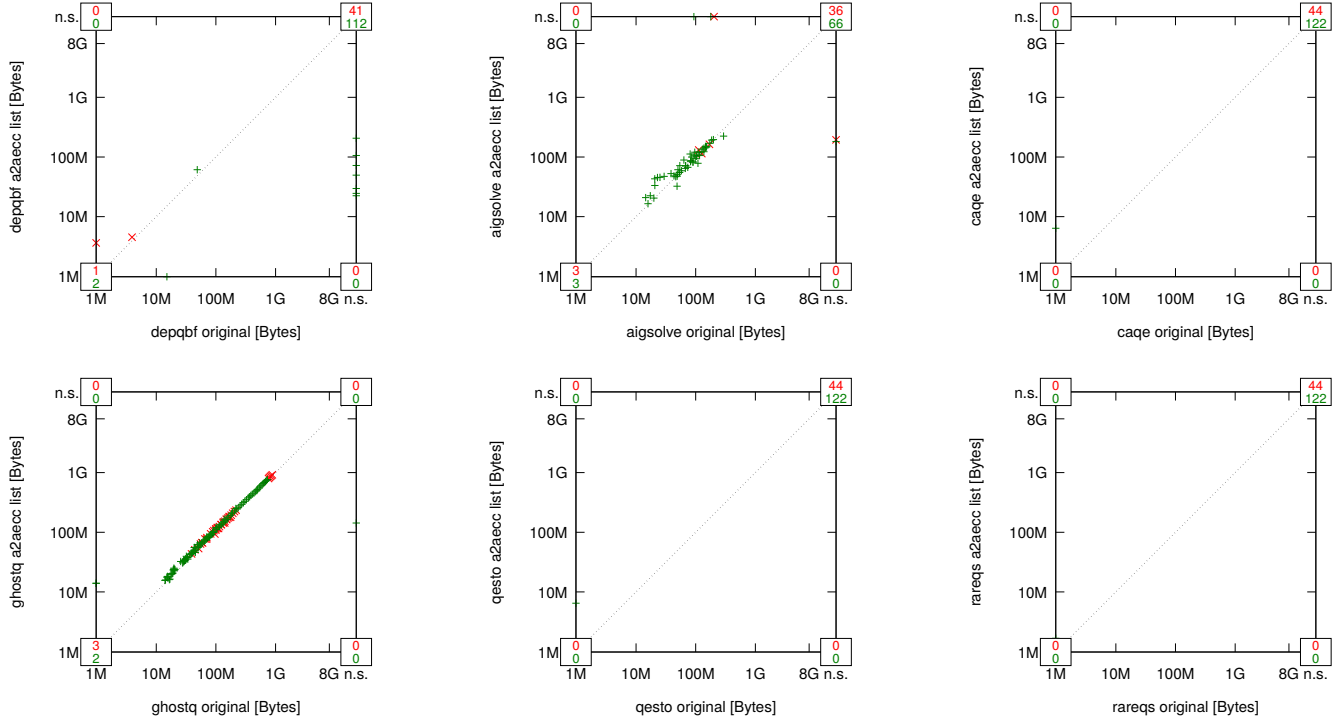


Fig. 1616: Suite Mneimneh-Sakallah ($n = 180$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

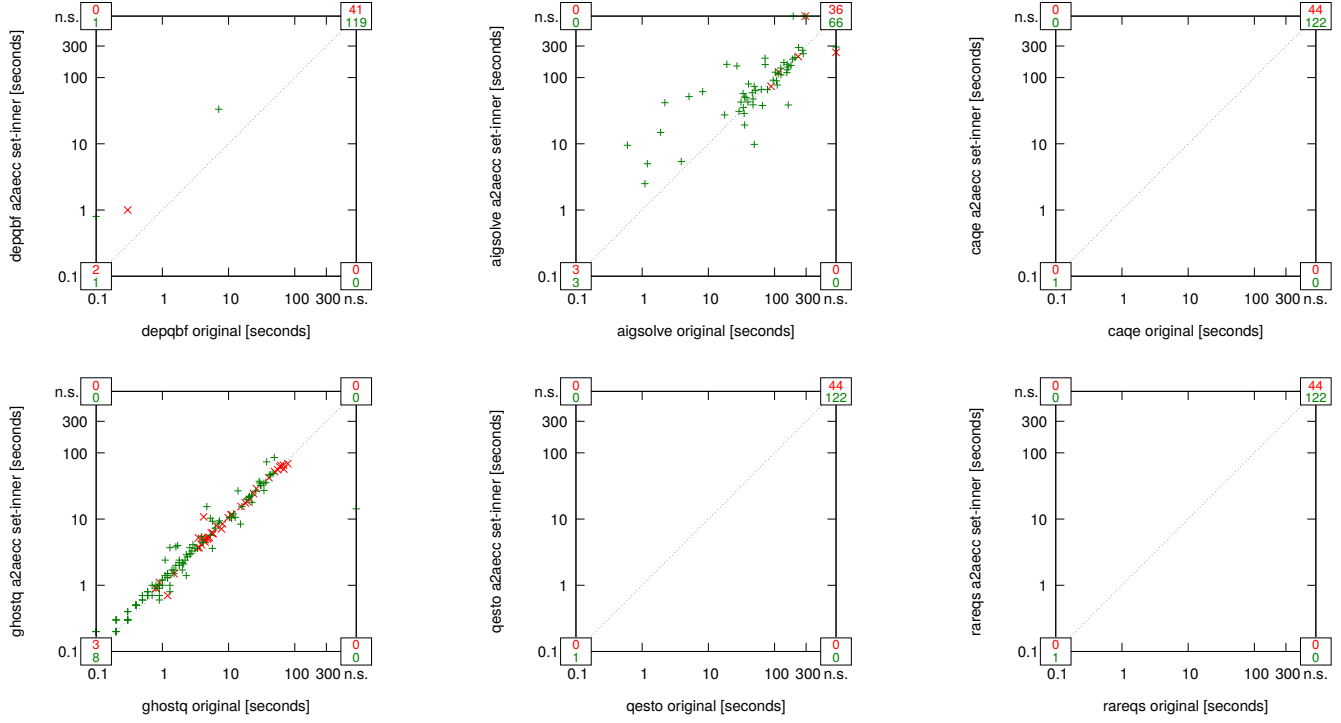


Fig. 1617: Suite Mneimneh-Sakallah ($n = 180$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

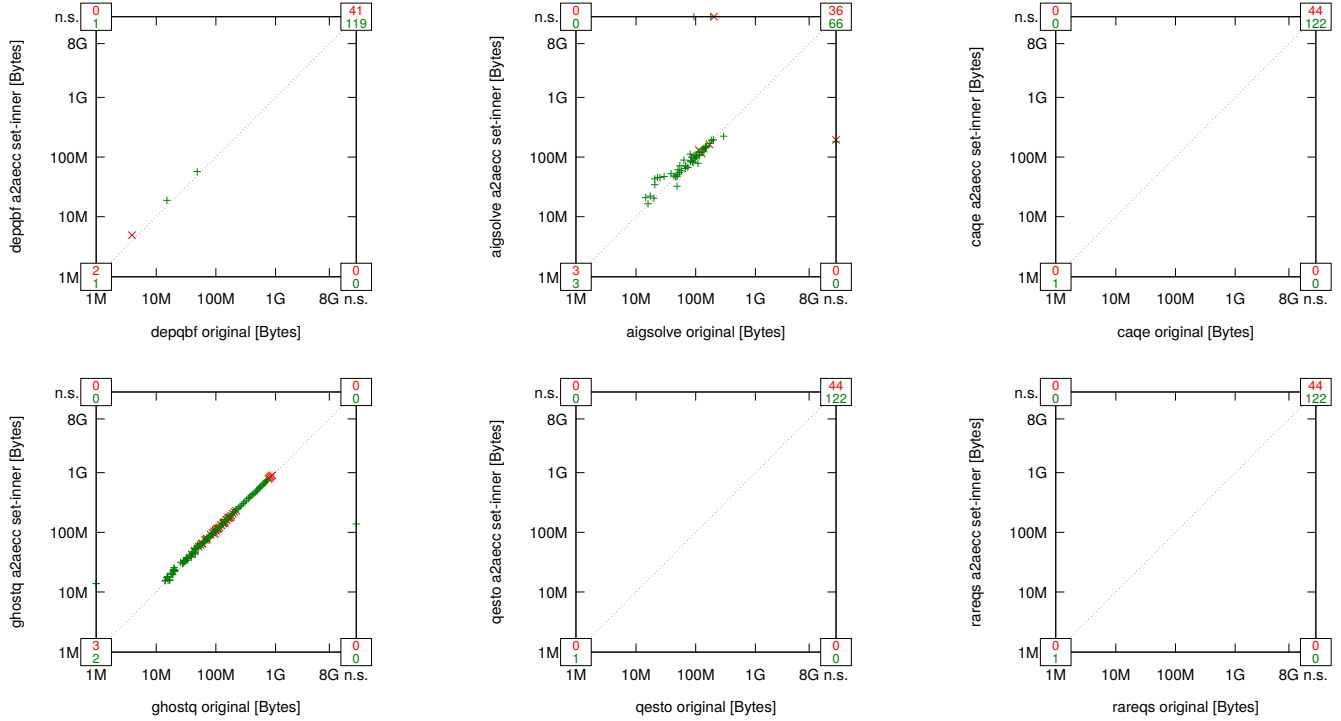


Fig. 1618: Suite Mneimneh-Sakallah ($n = 180$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

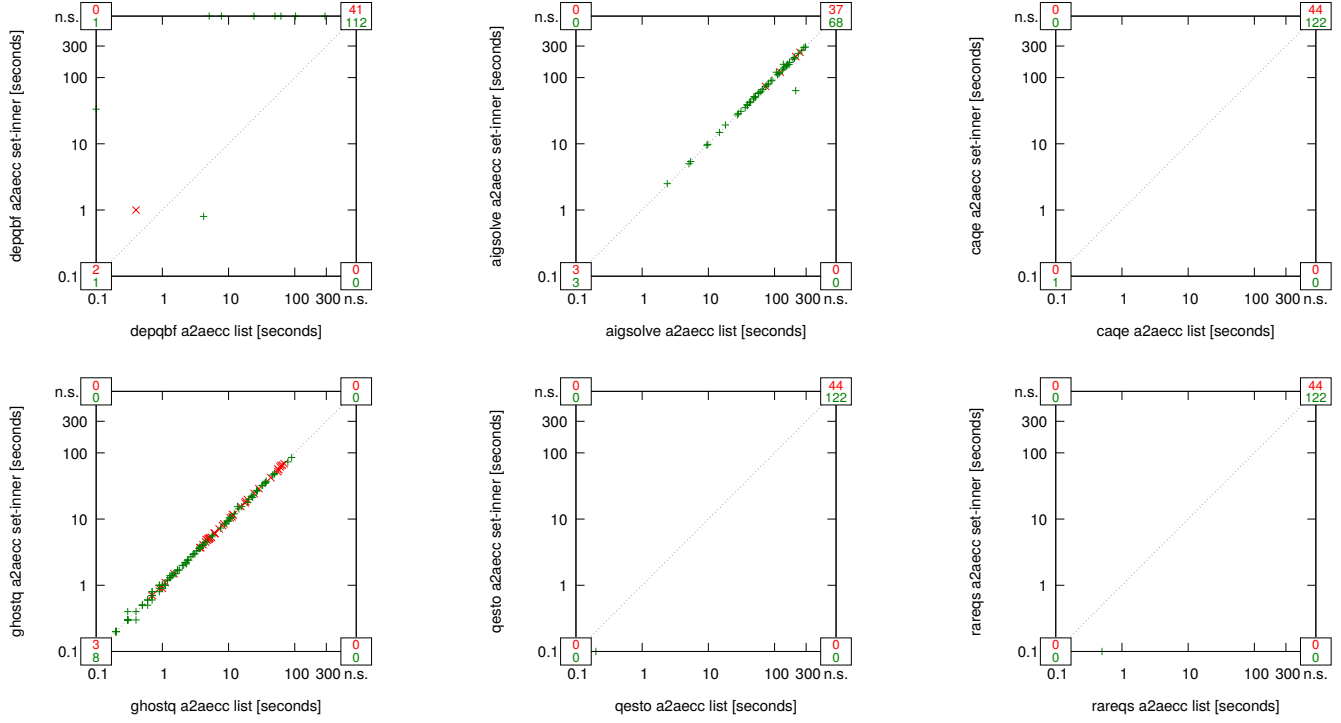


Fig. 1619: Suite Mneimneh-Sakallah ($n = 180$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

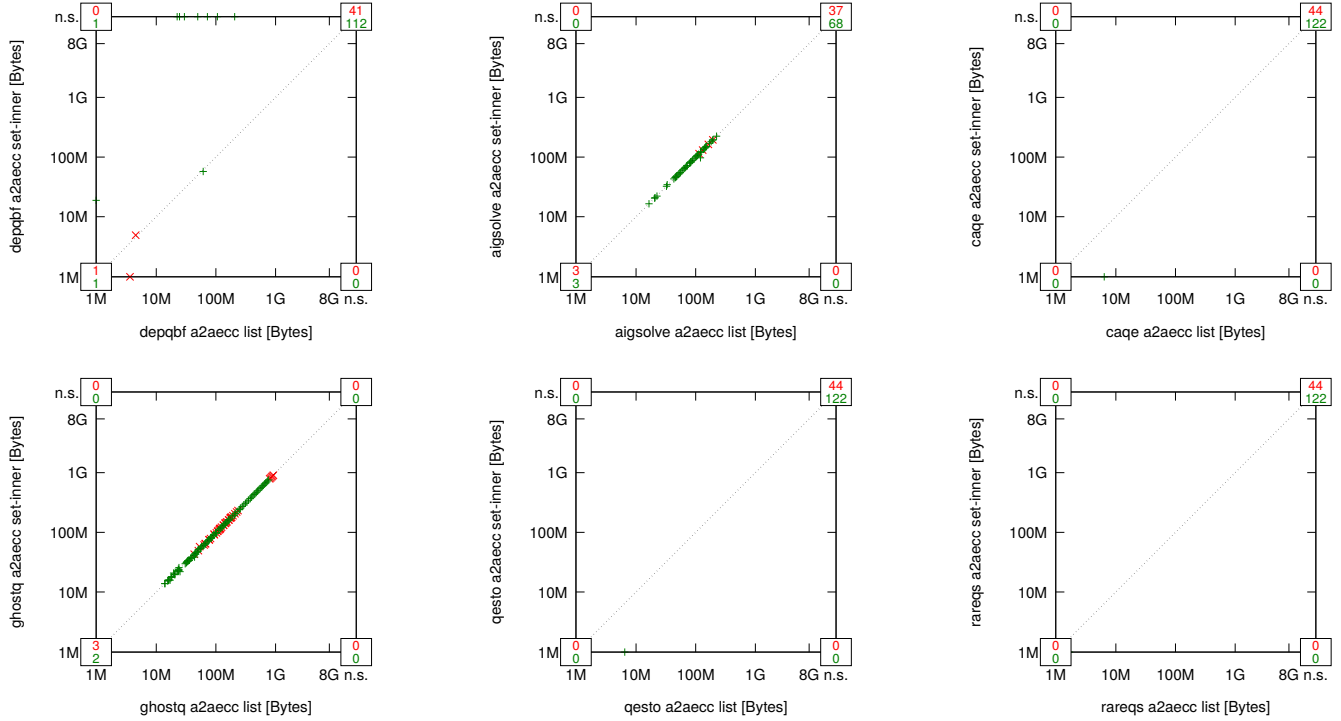


Fig. 1620: Suite Mneimneh-Sakallah ($n = 180$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

33) Narizzano ($n = 193$):

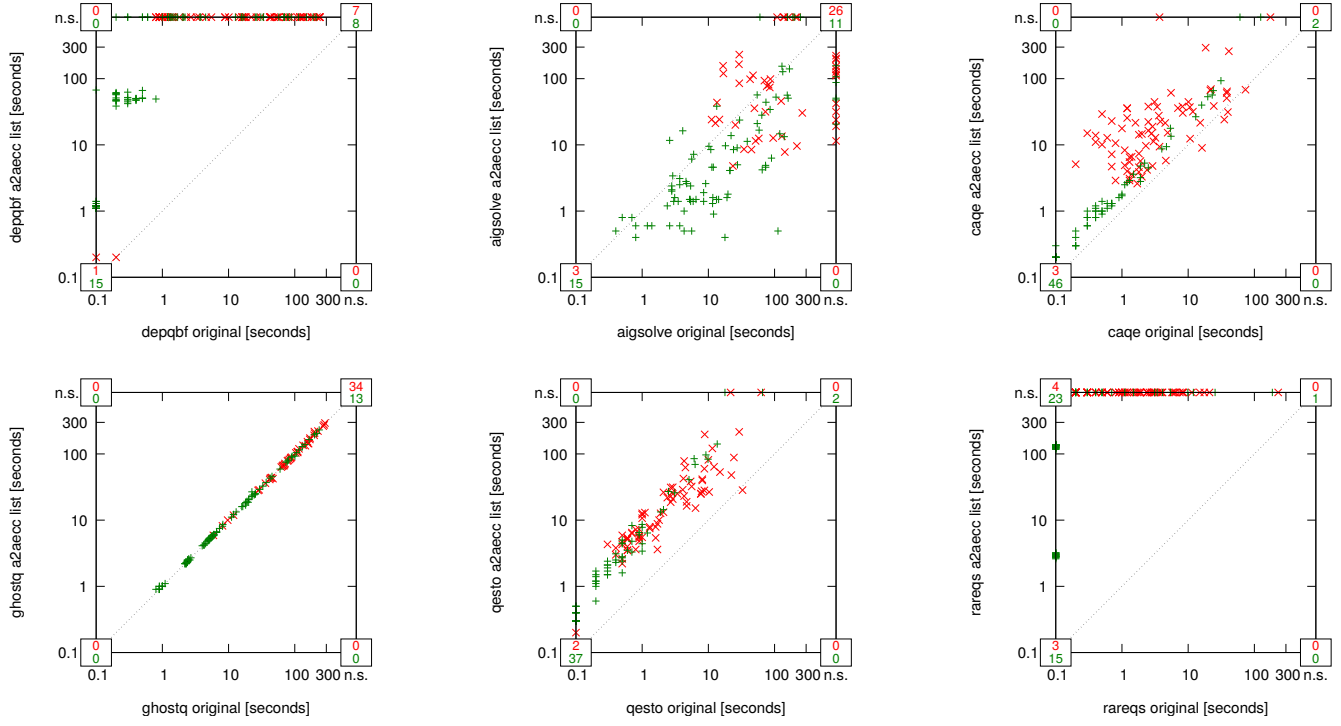


Fig. 1621: Suite Narizzano ($n = 193$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

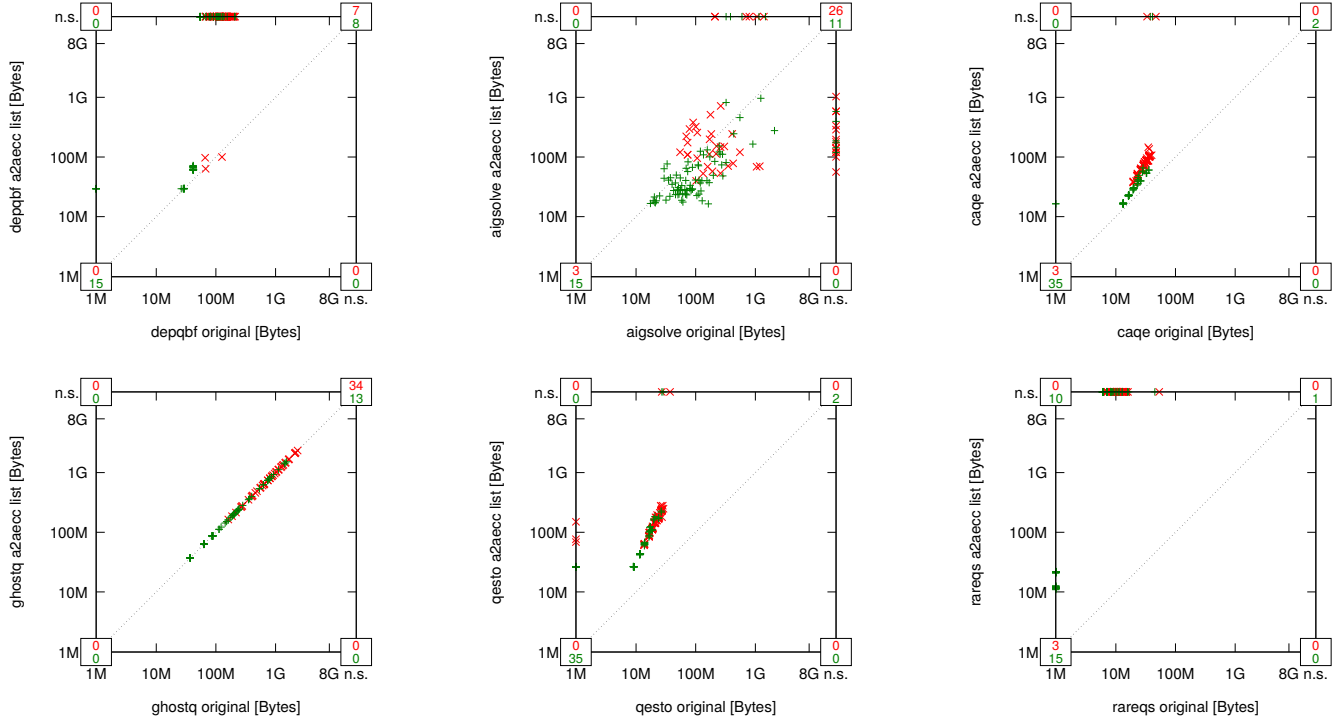


Fig. 1622: Suite Narizzano ($n = 193$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

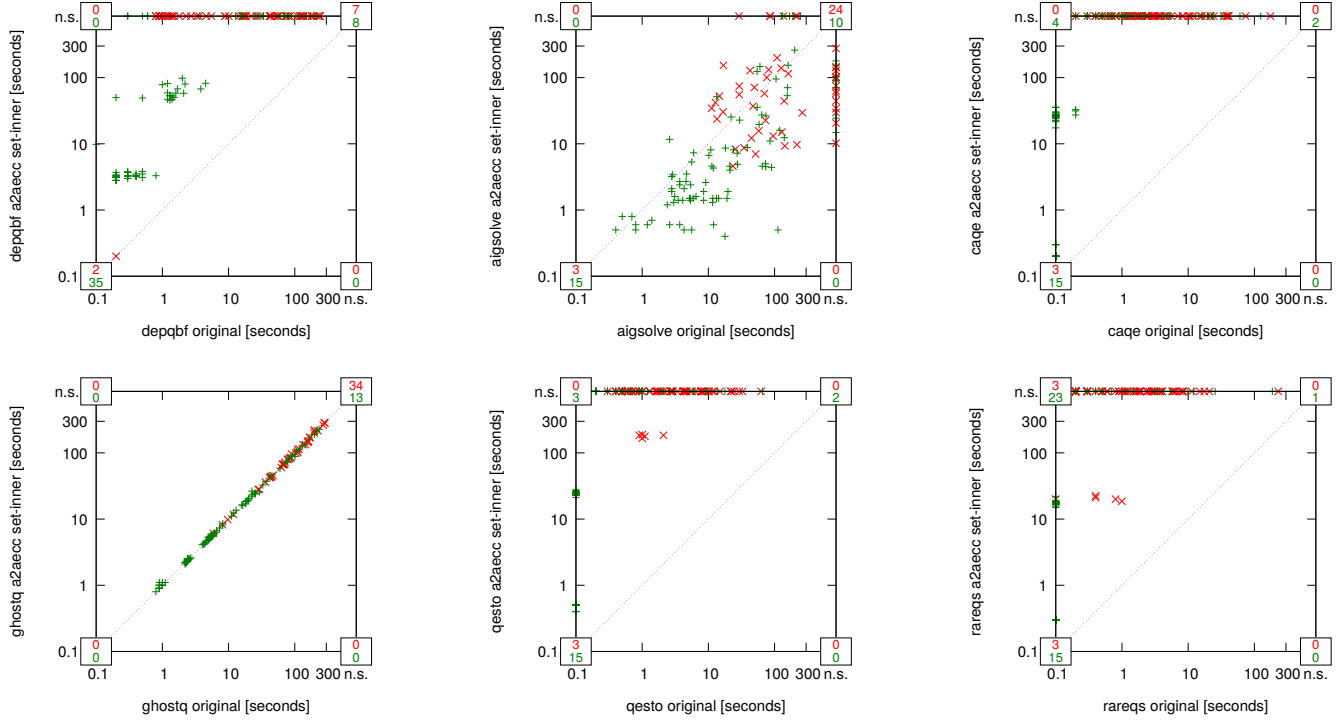


Fig. 1623: Suite Narizzano ($n = 193$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

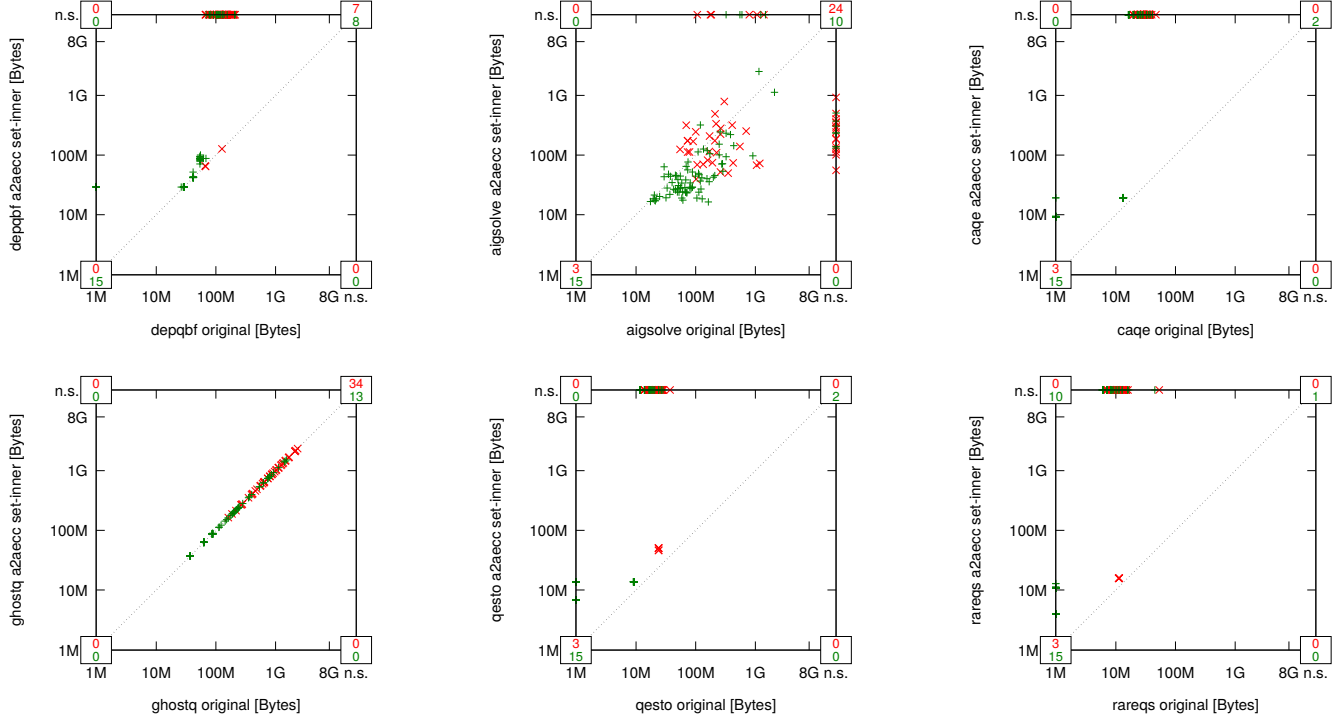


Fig. 1624: Suite Narizzano ($n = 193$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

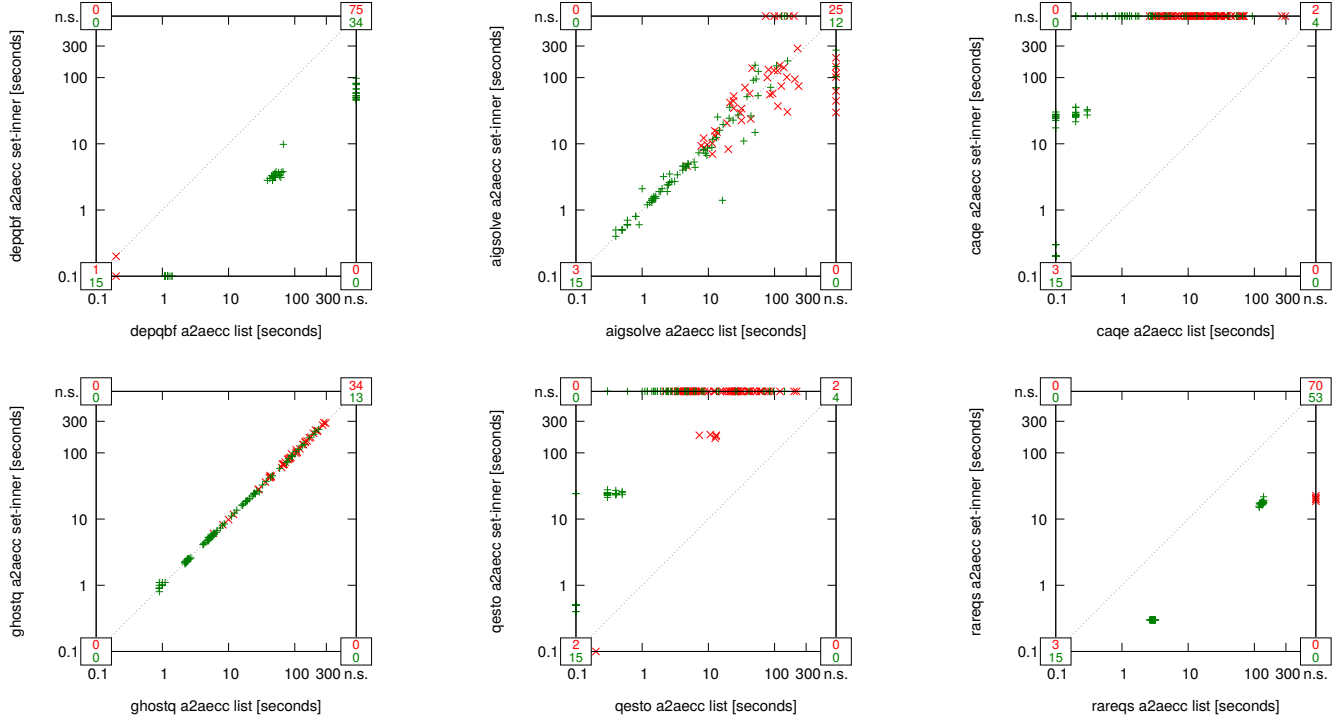


Fig. 1625: Suite Narizzano ($n = 193$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

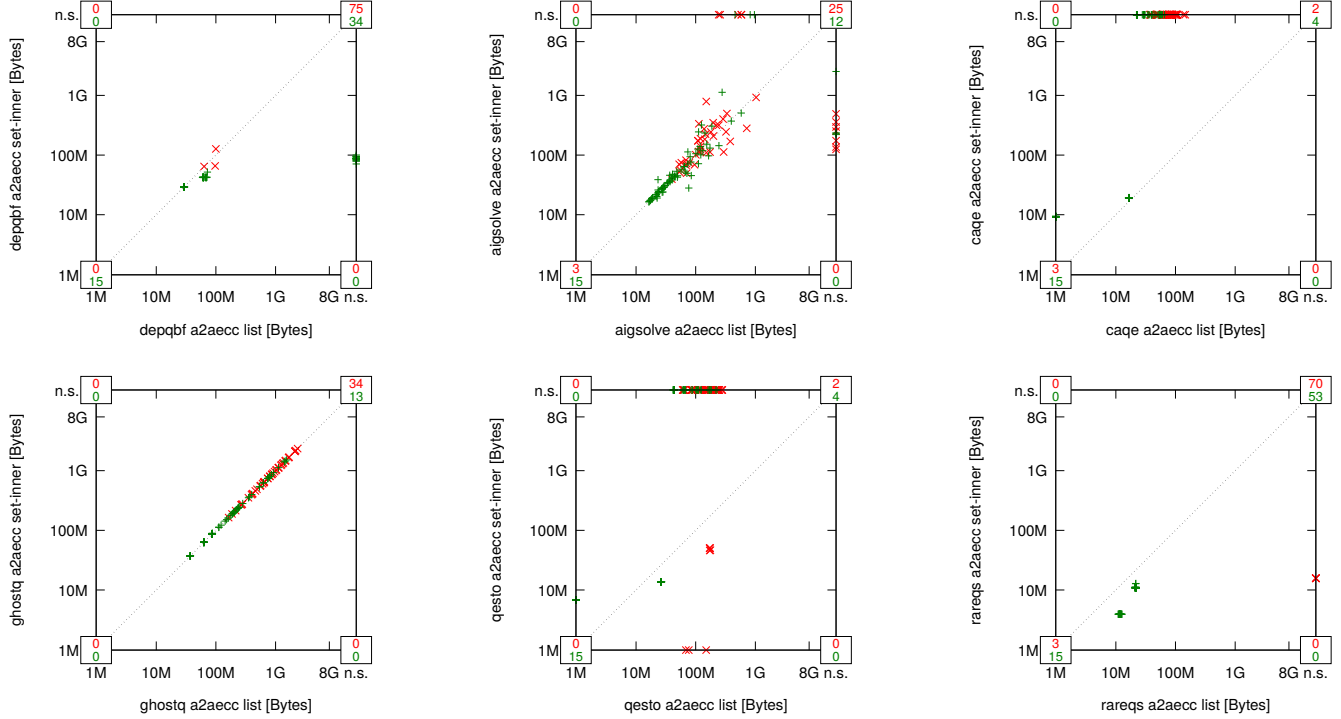


Fig. 1626: Suite Narizzano ($n = 193$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

34) Palacios ($n = 24$):

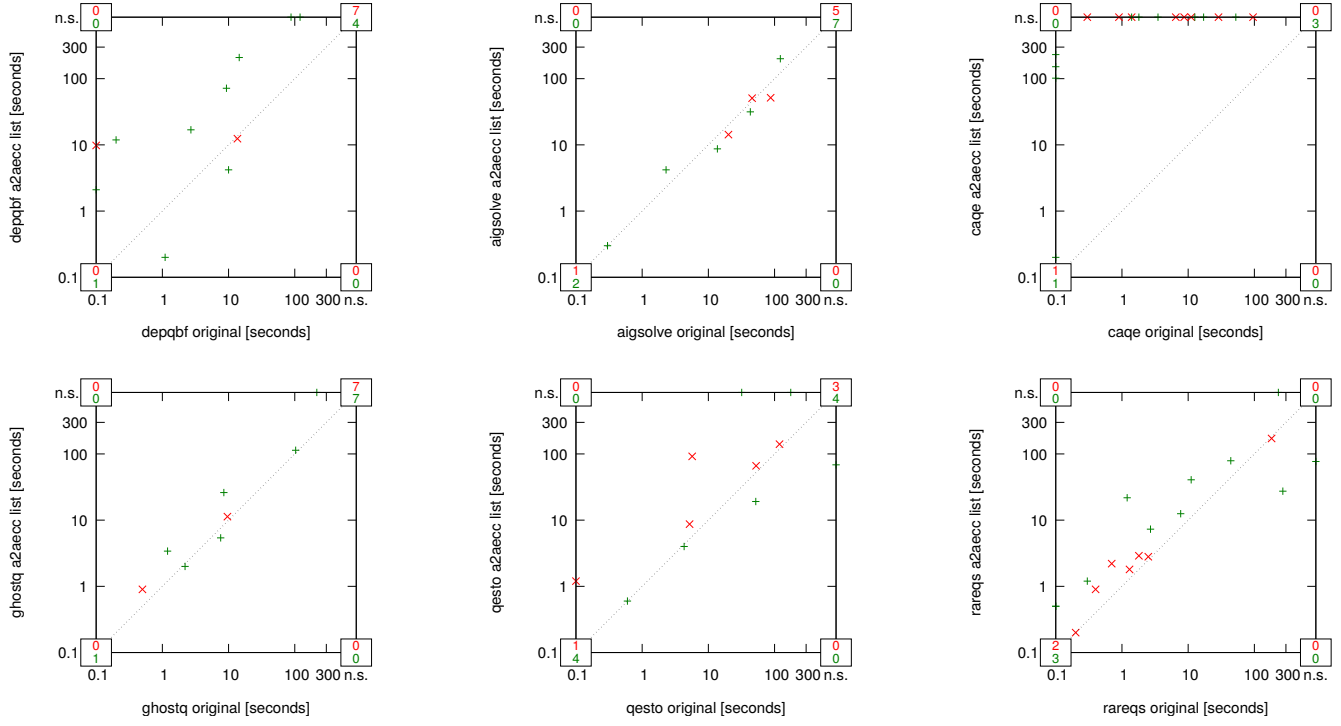


Fig. 1627: Suite Palacios ($n = 24$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

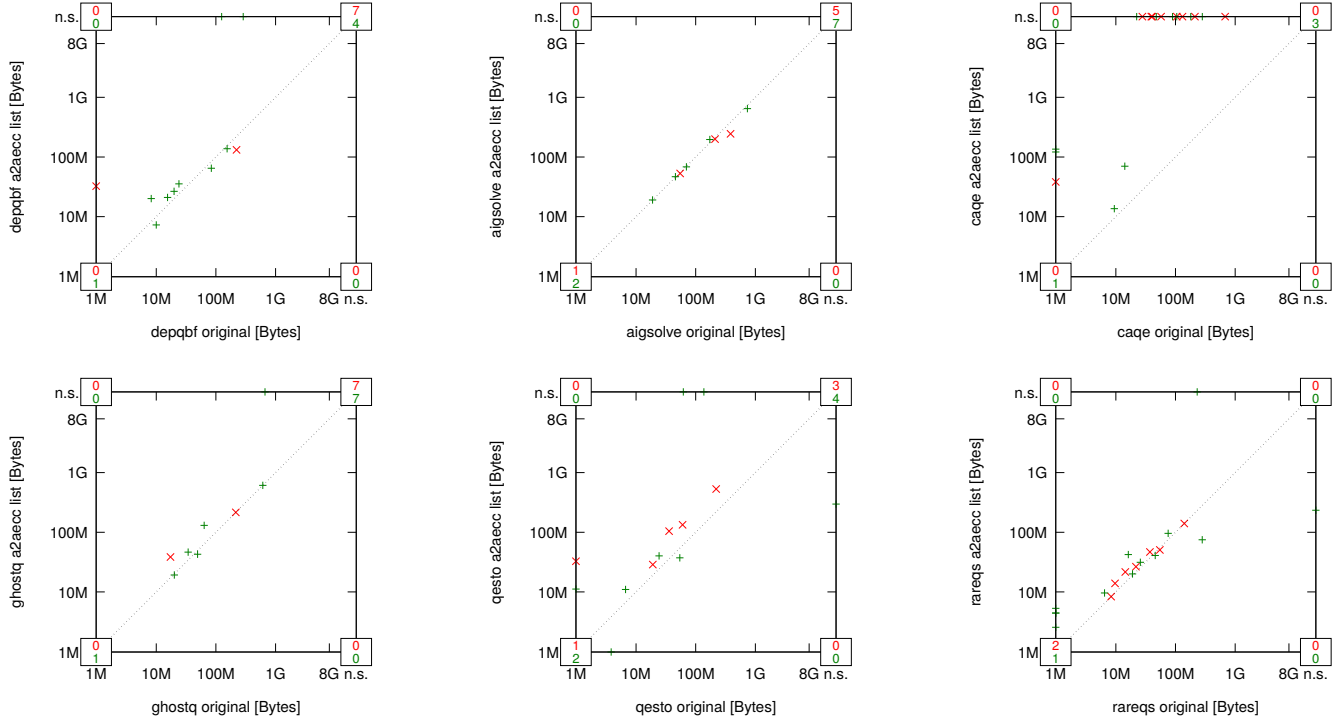


Fig. 1628: Suite Palacios ($n = 24$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

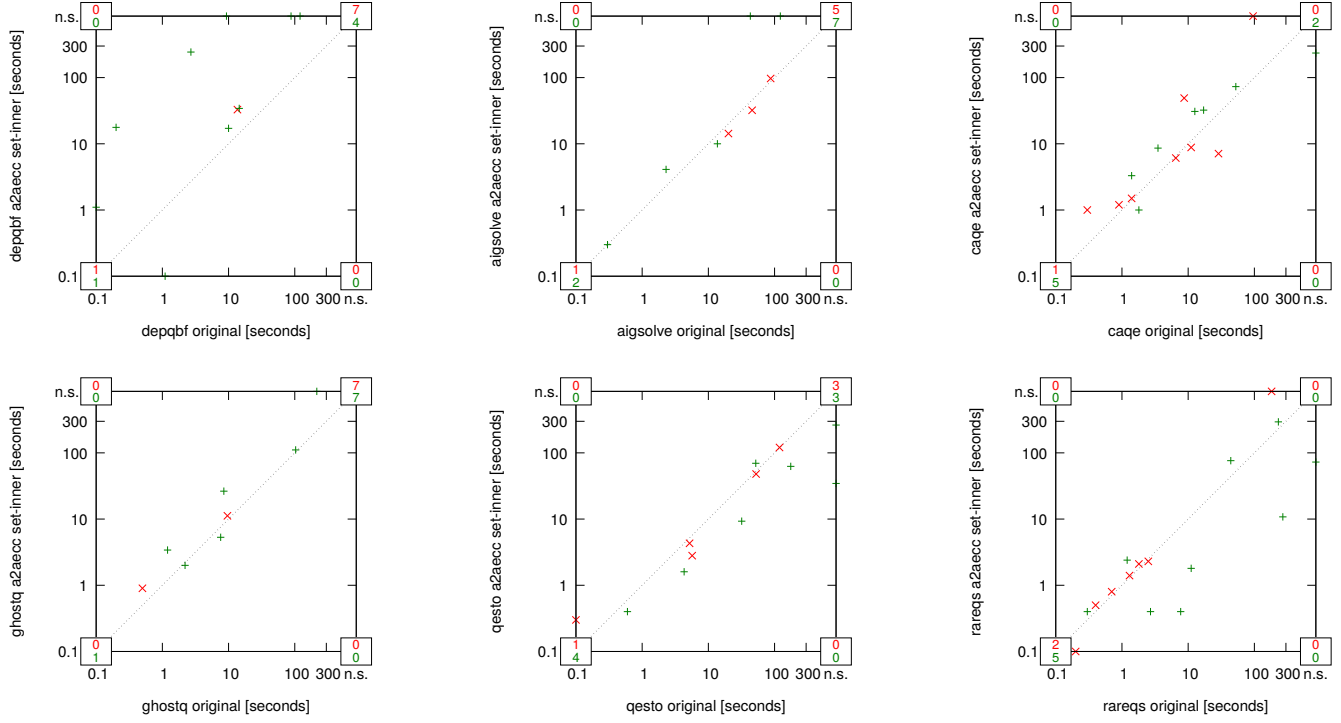


Fig. 1629: Suite Palacios ($n = 24$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

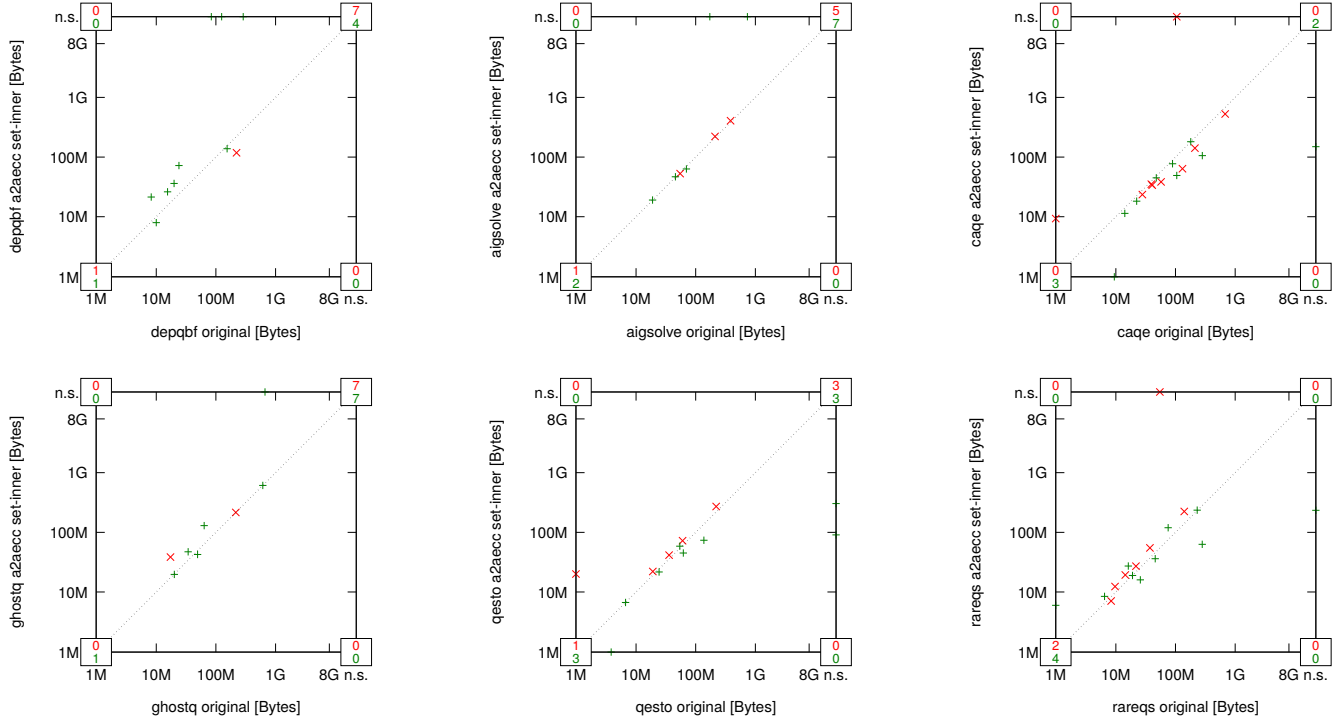


Fig. 1630: Suite Palacios ($n = 24$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

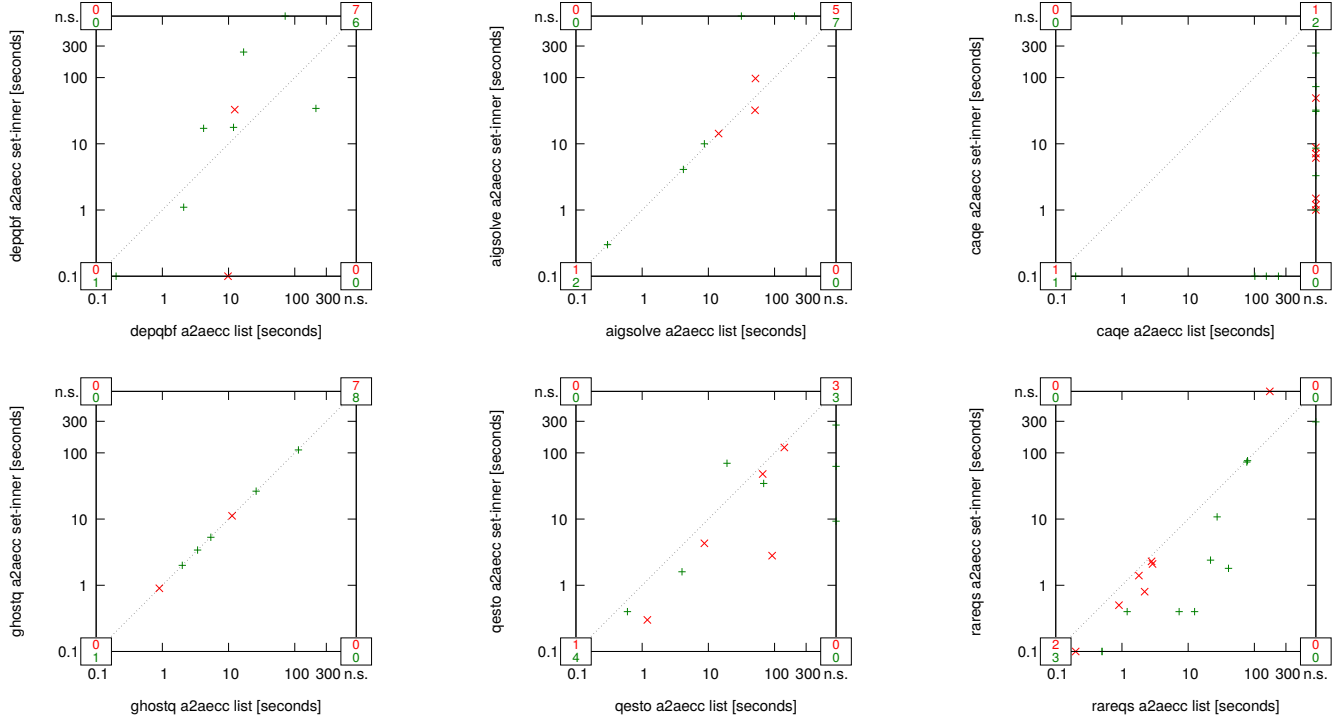


Fig. 1631: Suite Palacios ($n = 24$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

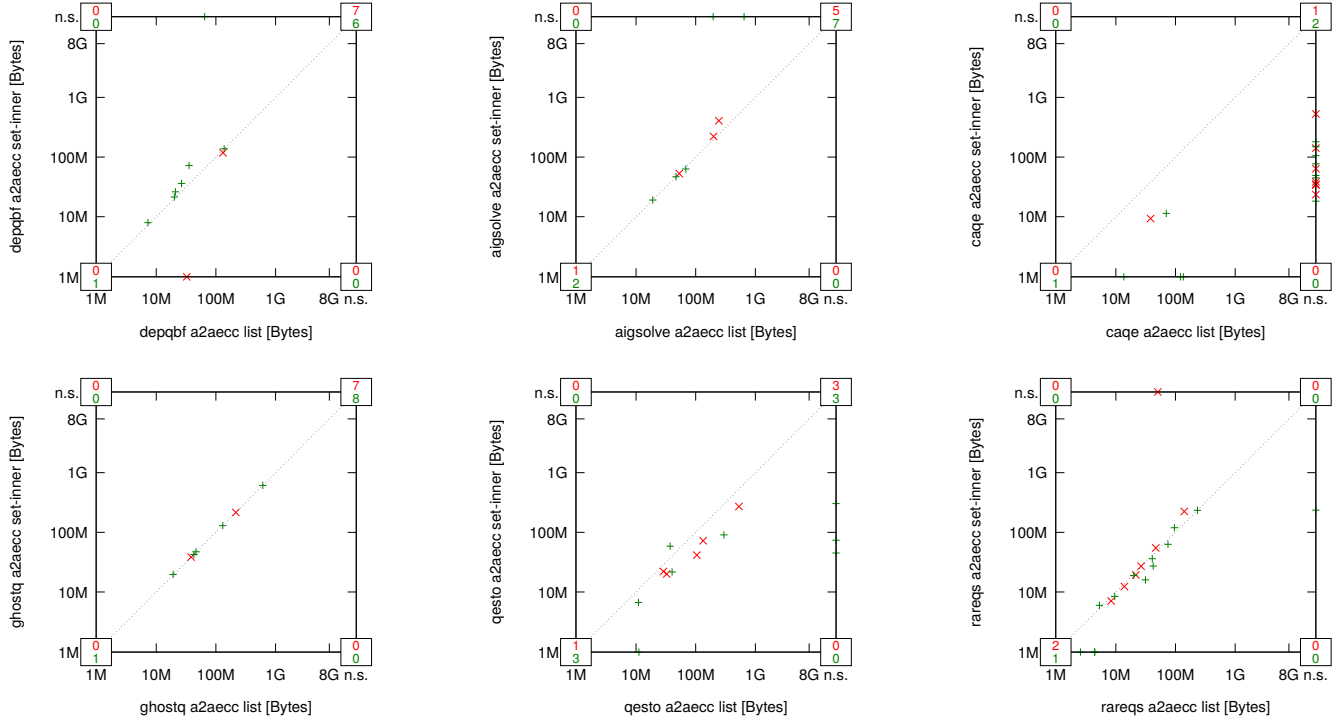


Fig. 1632: Suite Palacios ($n = 24$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

35) *Pan* ($n = 194$):

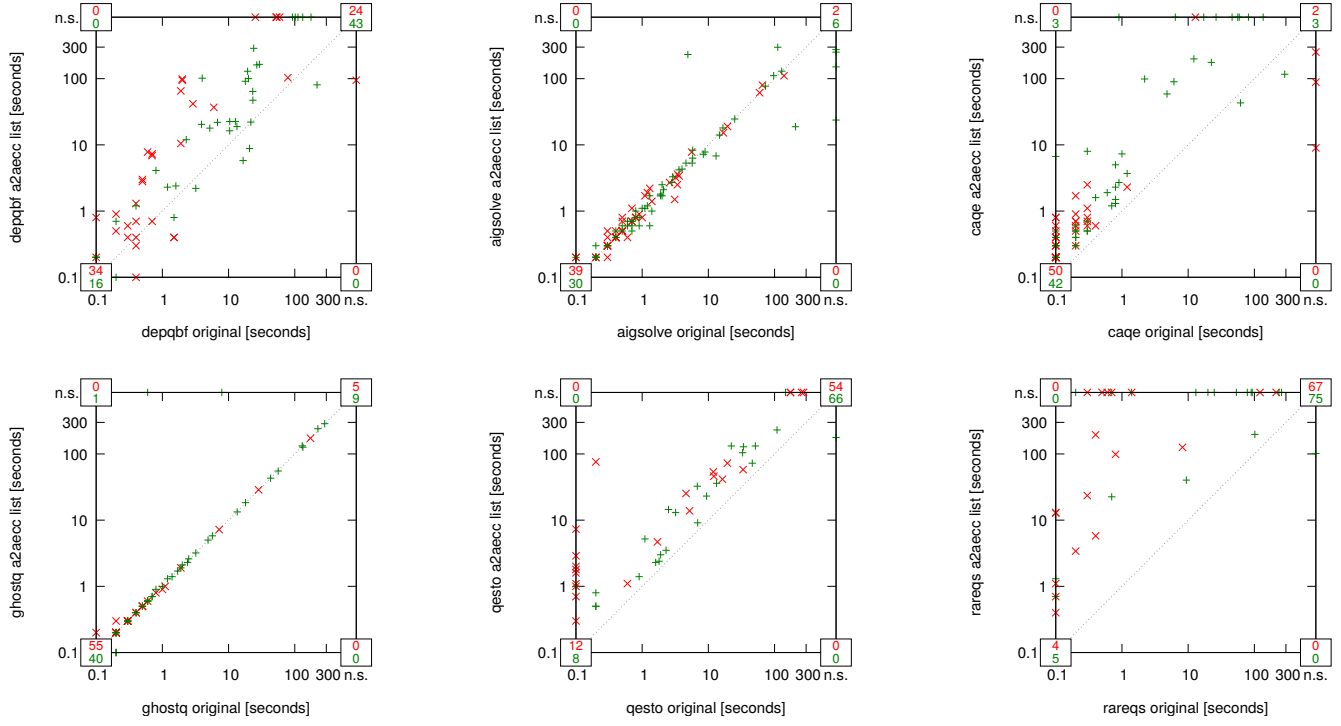


Fig. 1633: Suite Pan ($n = 194$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

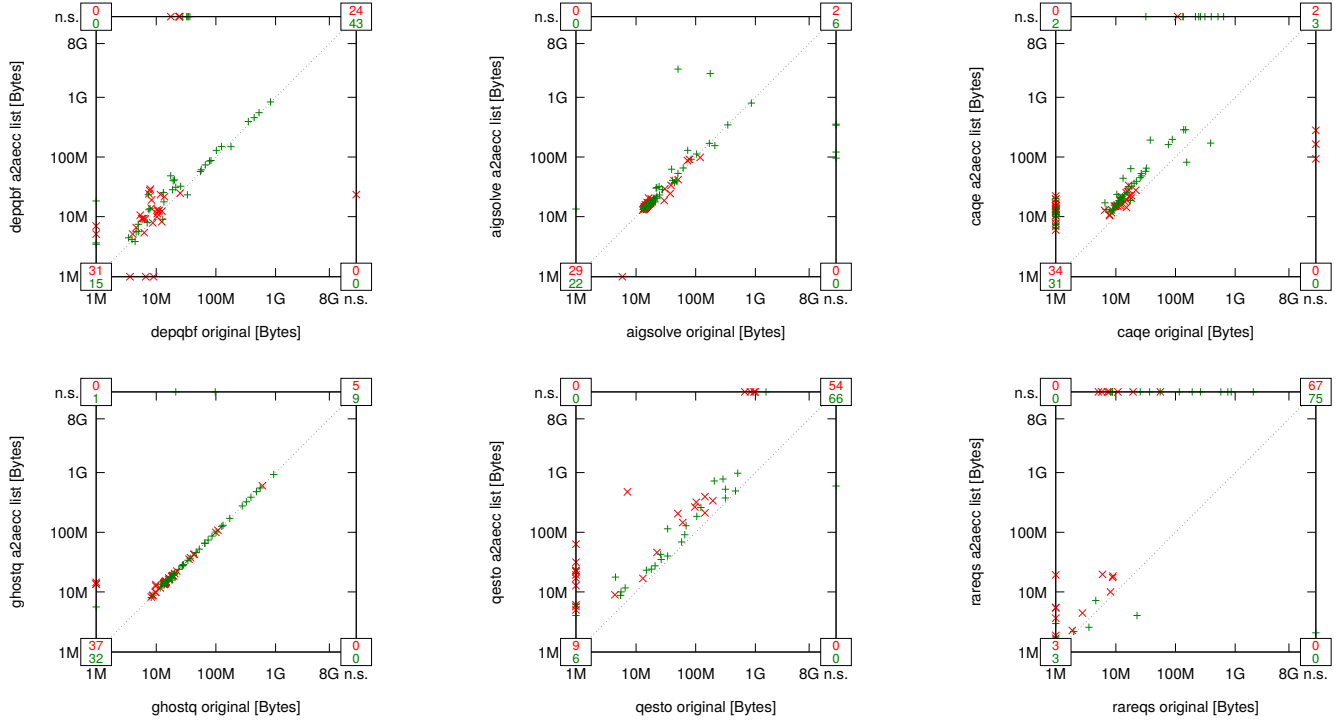


Fig. 1634: Suite Pan ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

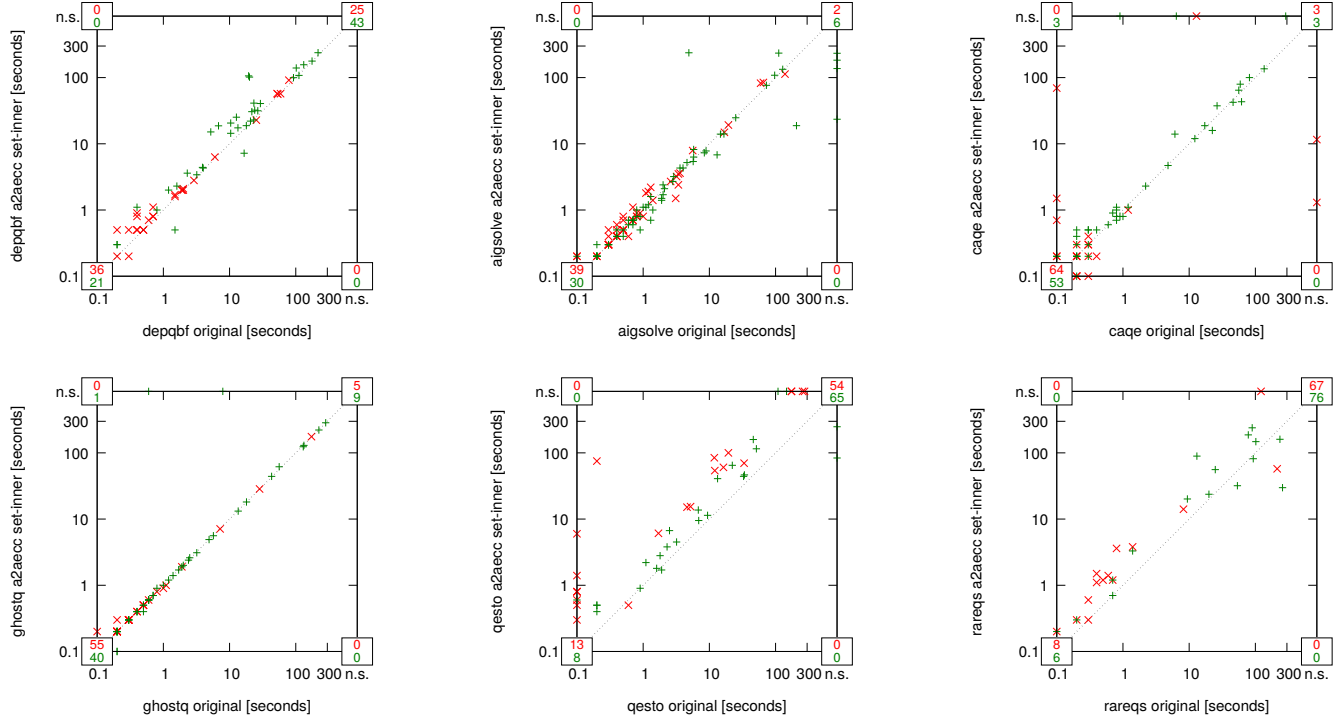


Fig. 1635: Suite Pan ($n = 194$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

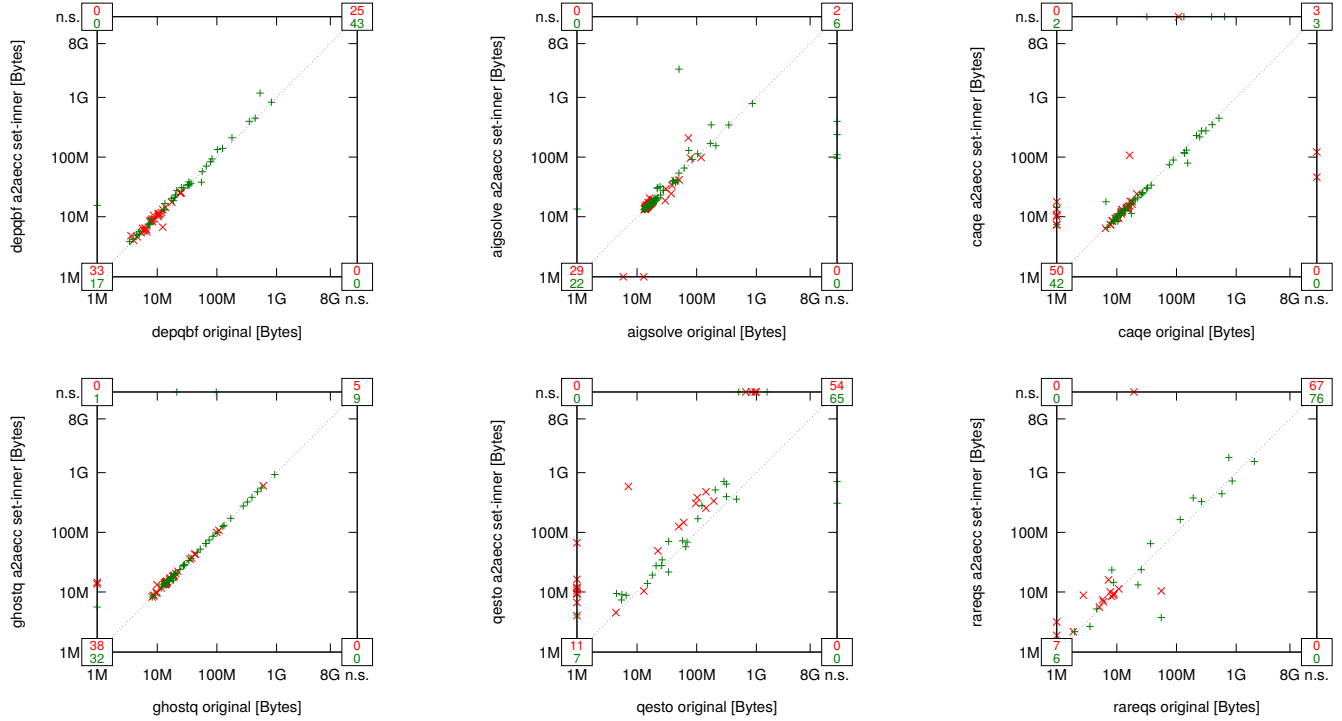


Fig. 1636: Suite Pan ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

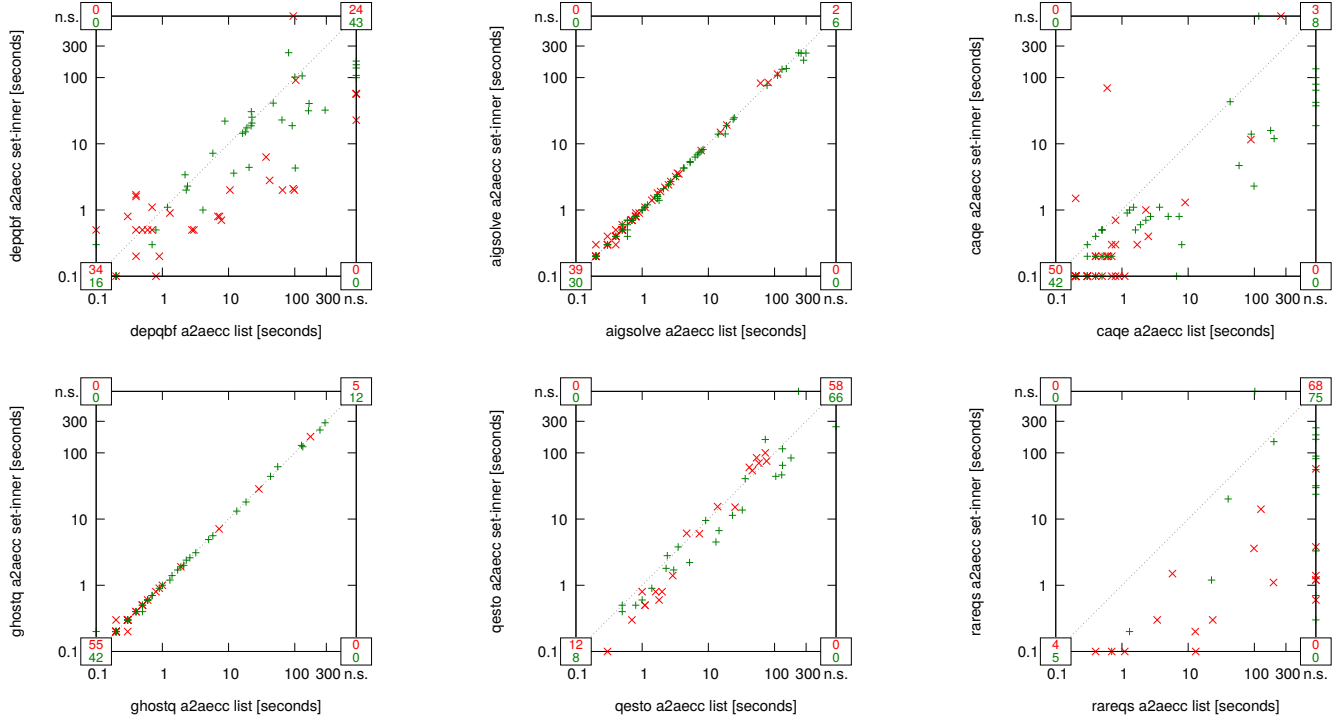


Fig. 1637: Suite Pan ($n = 194$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

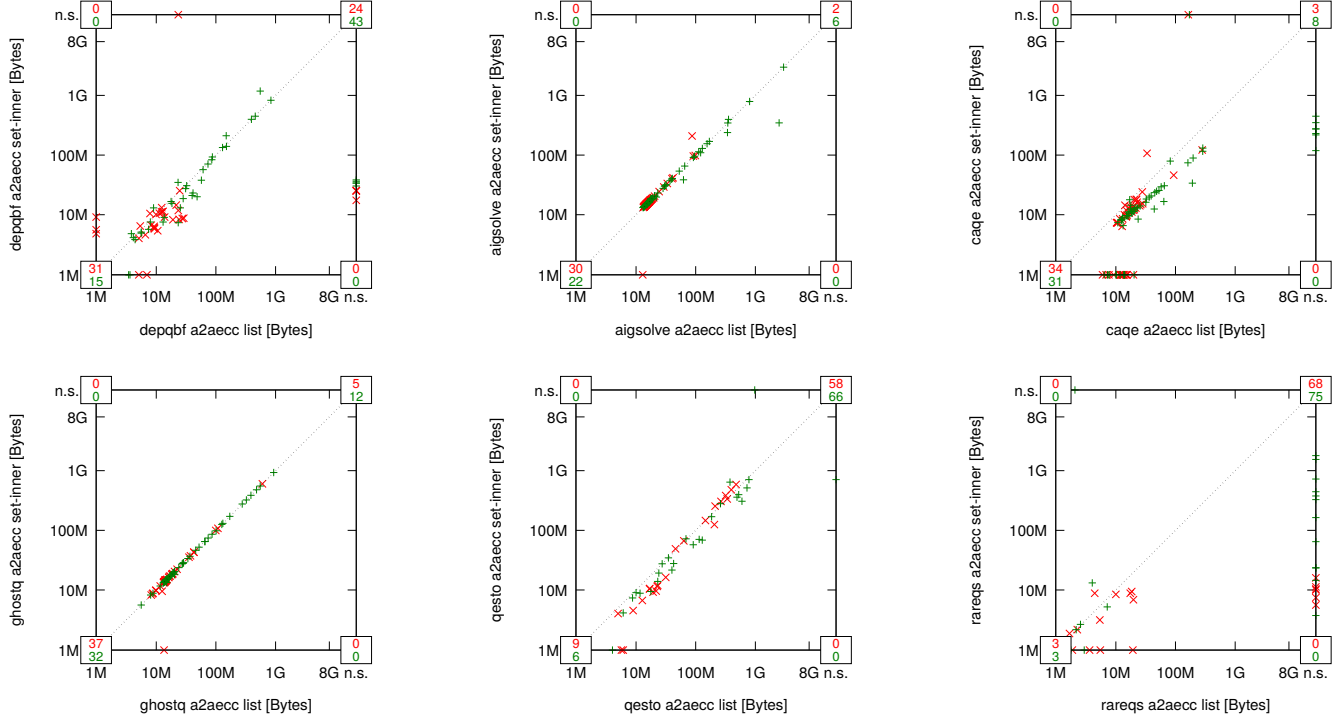


Fig. 1638: Suite Pan ($n = 194$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

36) *Peitl* ($n = 10$):

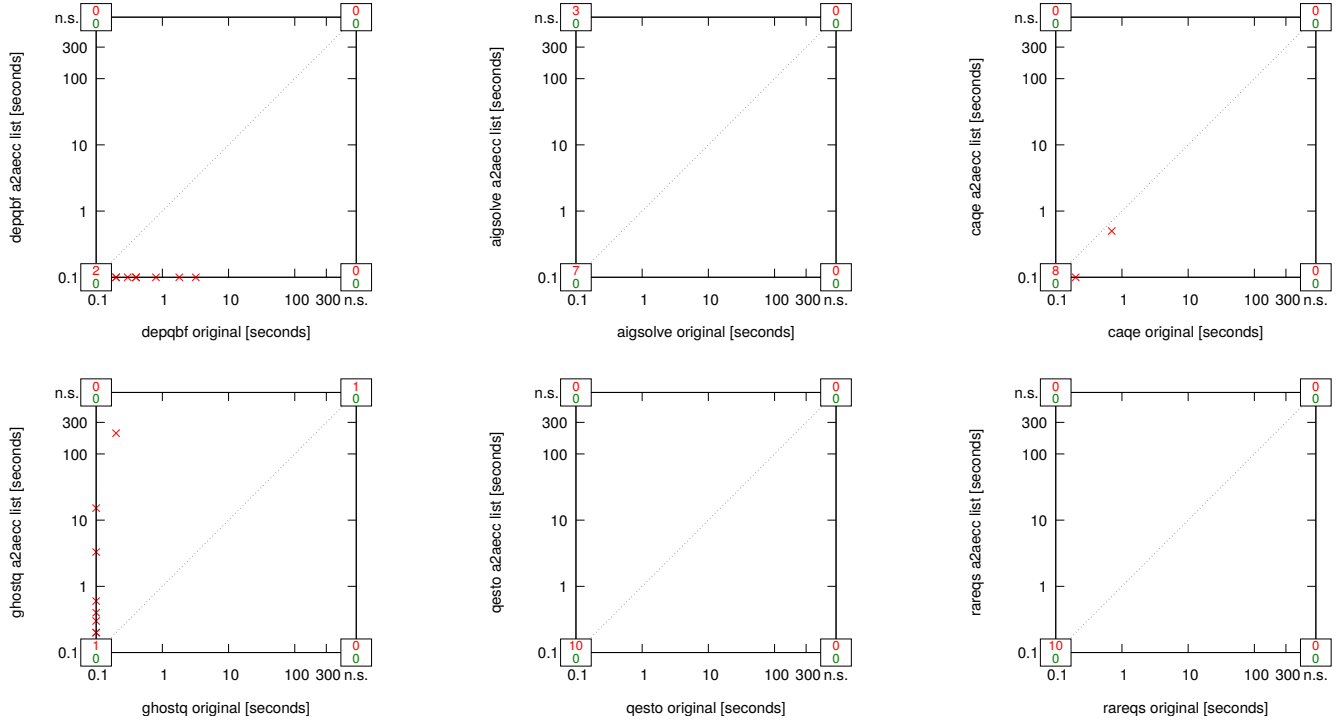


Fig. 1639: Suite *Peitl* ($n = 10$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

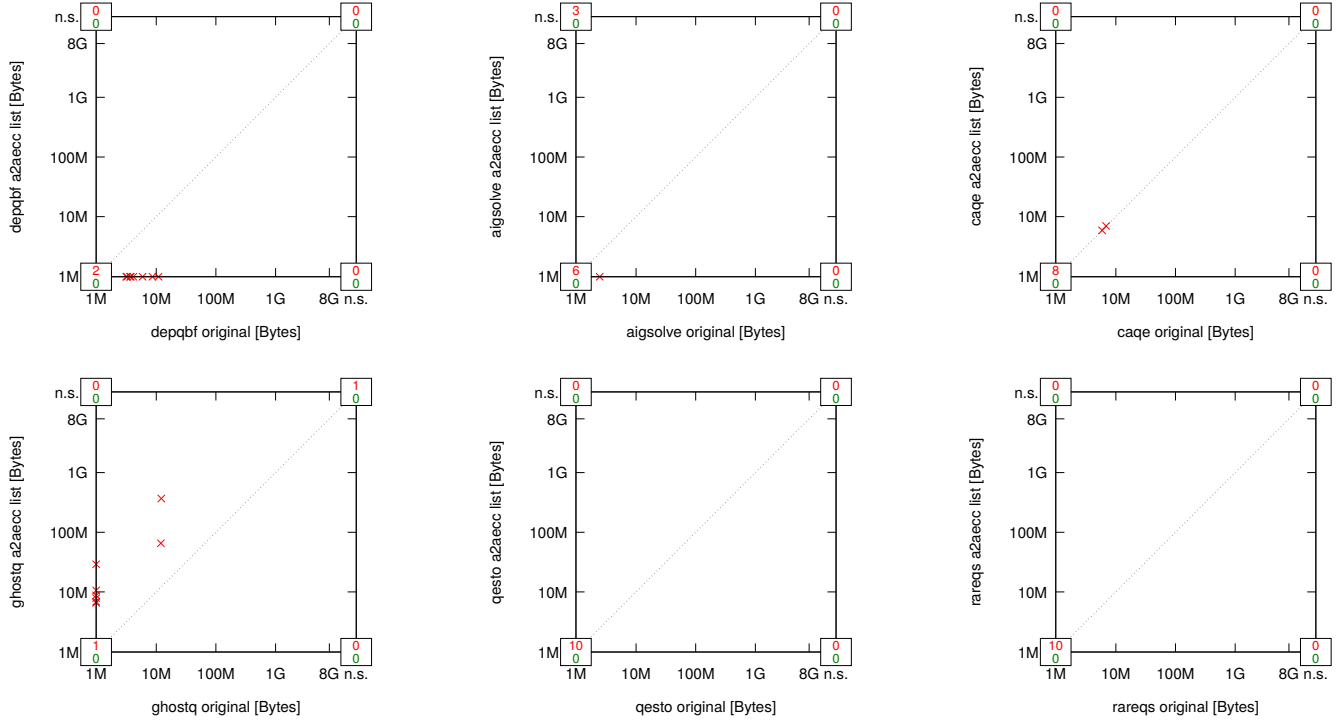


Fig. 1640: Suite *Peitl* ($n = 10$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

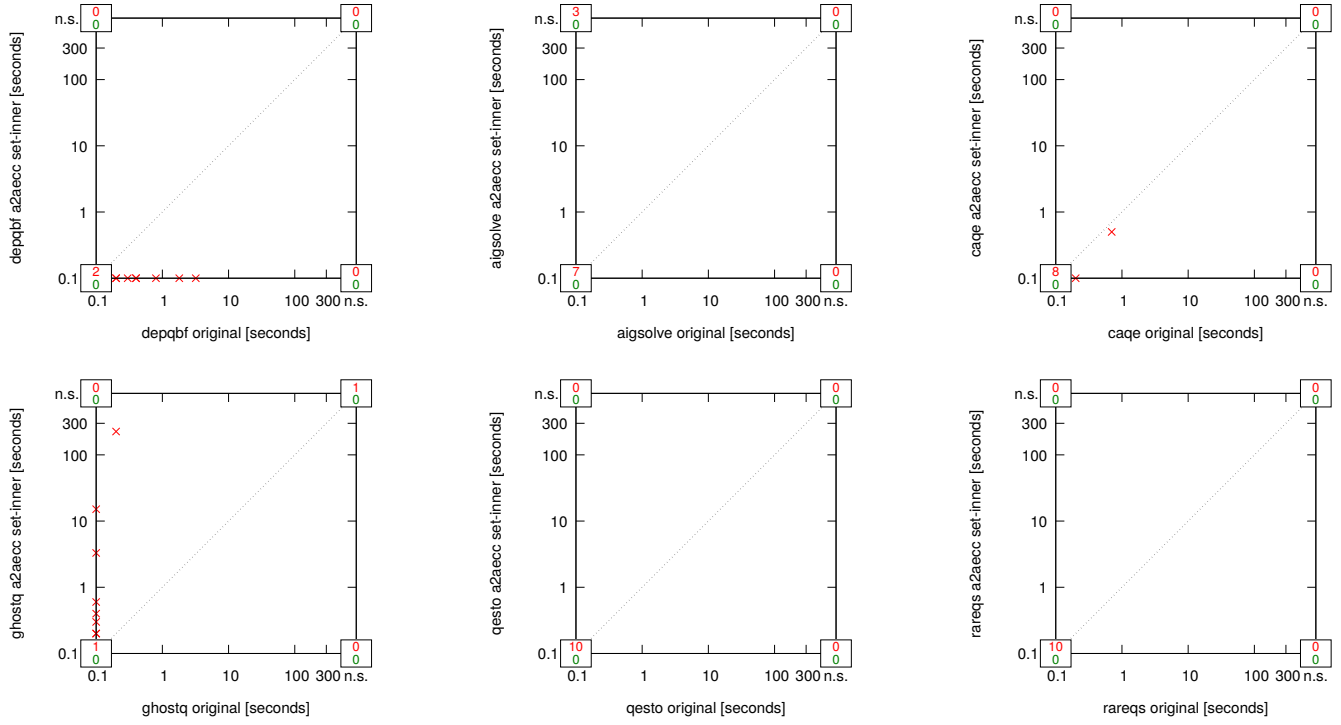


Fig. 1641: Suite Peitl ($n = 10$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

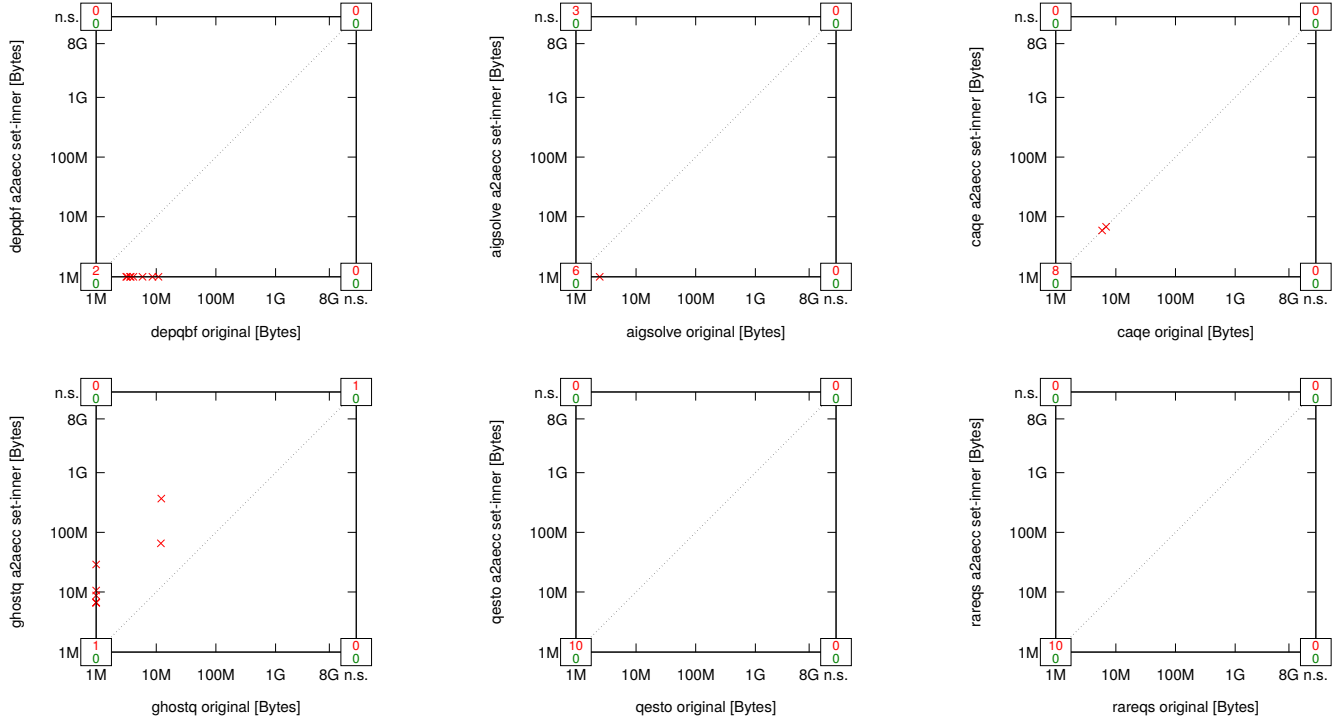


Fig. 1642: Suite Peitl ($n = 10$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

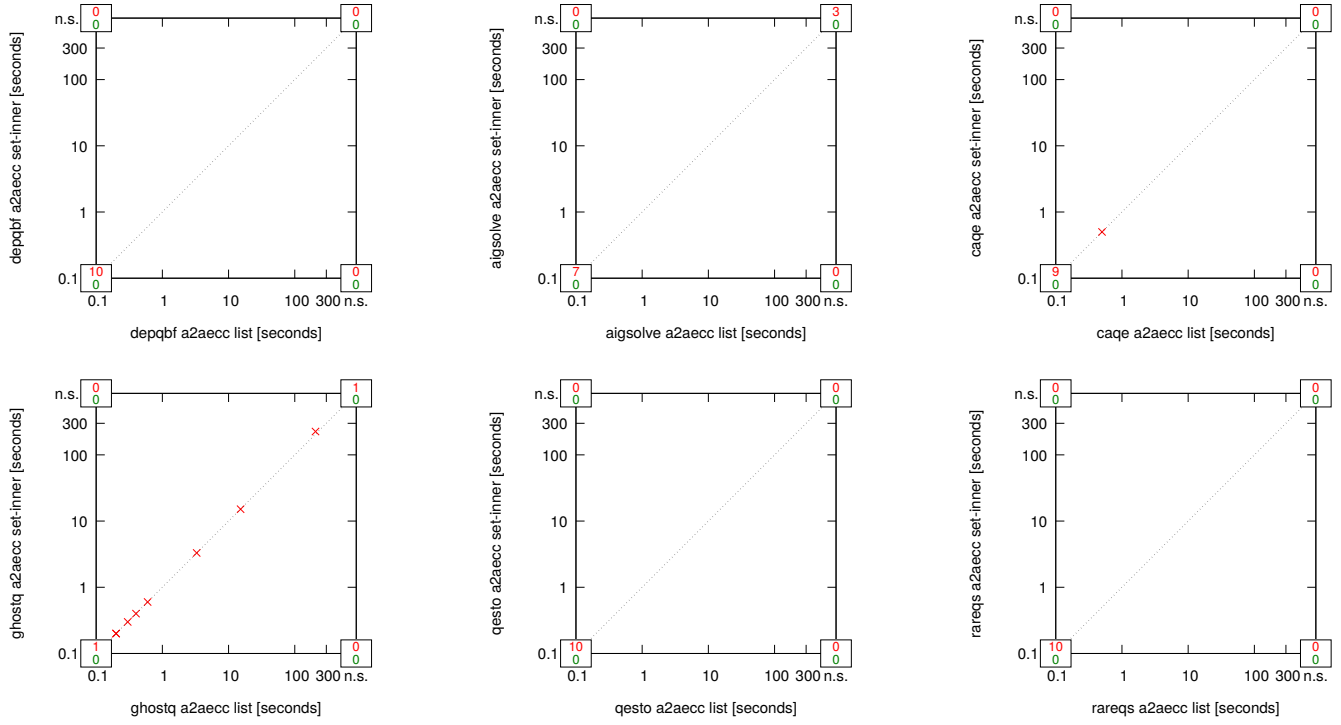


Fig. 1643: Suite Peitl ($n = 10$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

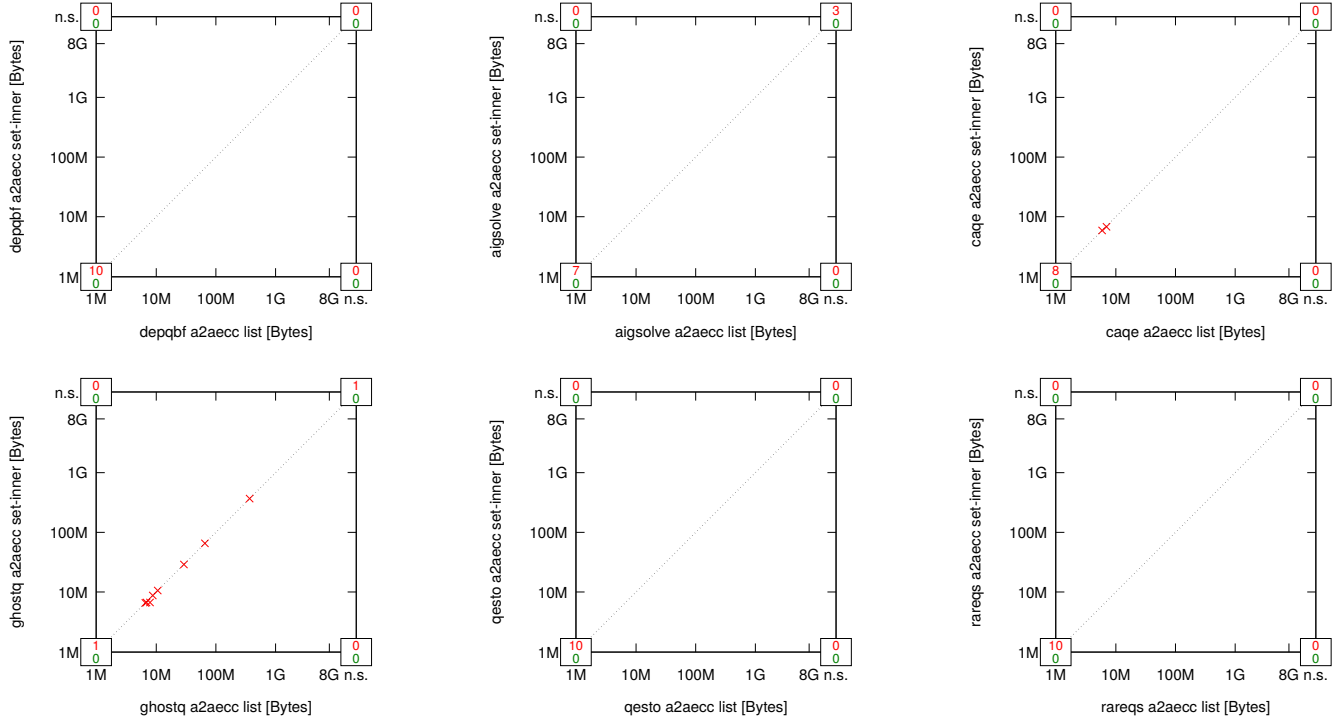


Fig. 1644: Suite Peitl ($n = 10$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

37) Preusser ($n = 12$):

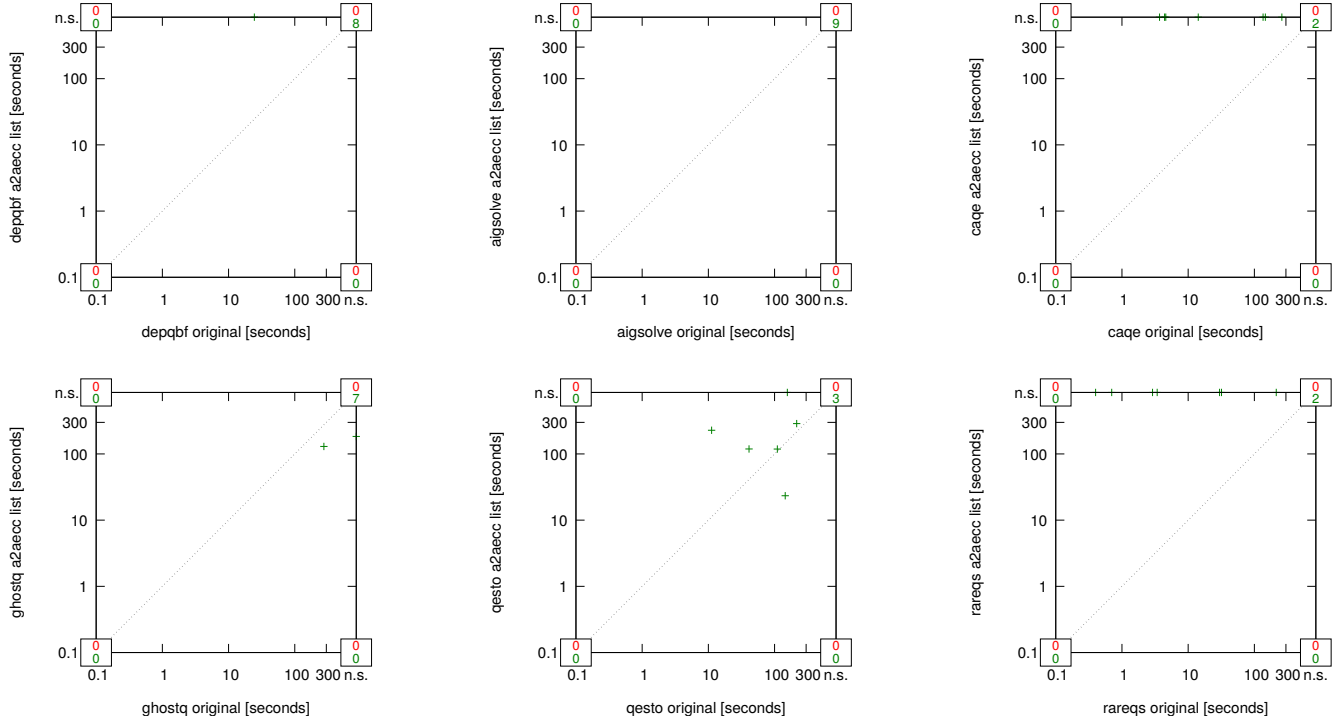


Fig. 1645: Suite Preusser ($n = 12$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

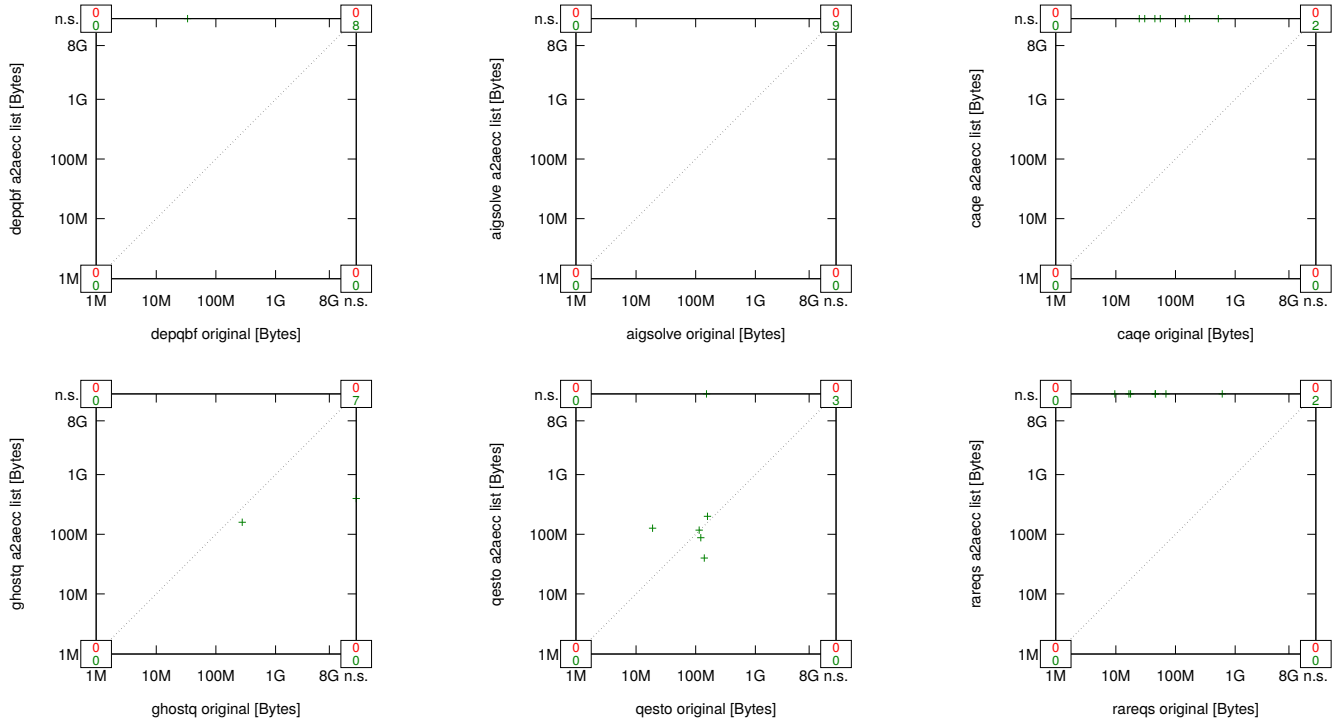


Fig. 1646: Suite Preusser ($n = 12$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

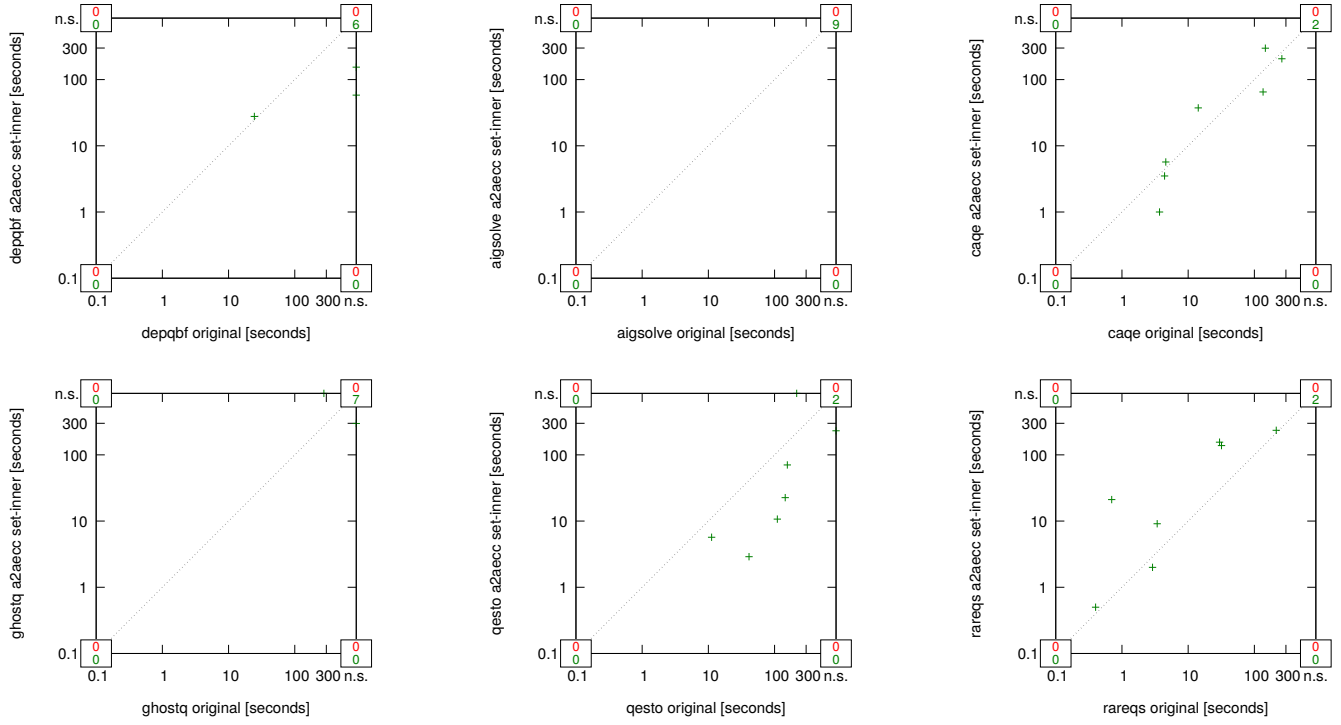


Fig. 1647: Suite Preusser ($n = 12$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

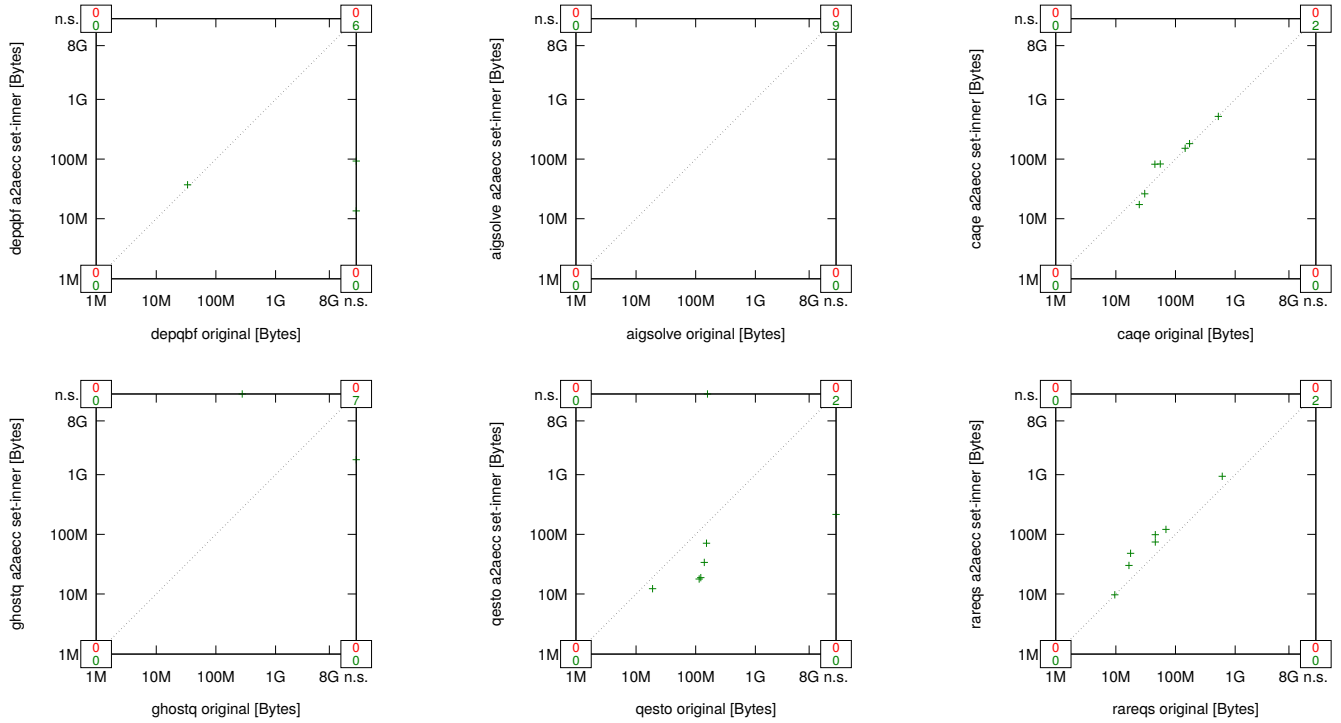


Fig. 1648: Suite Preusser ($n = 12$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

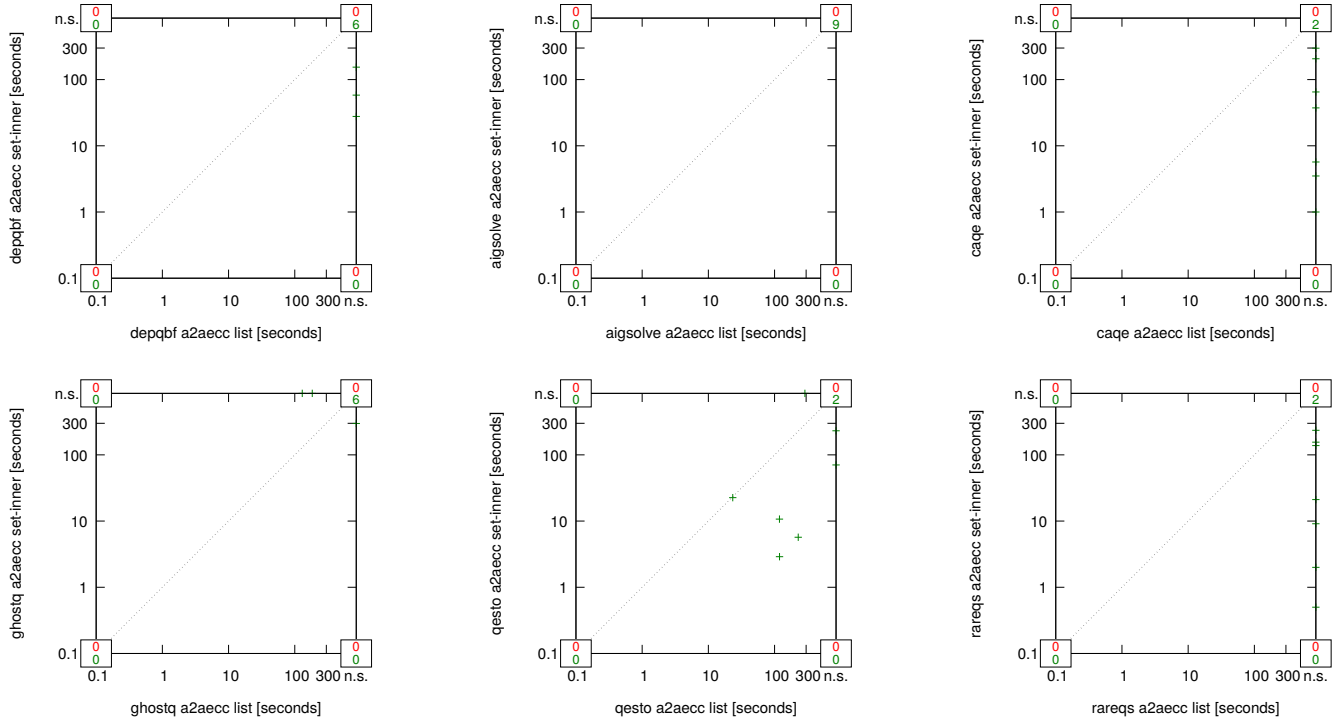


Fig. 1649: Suite Preusser ($n = 12$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

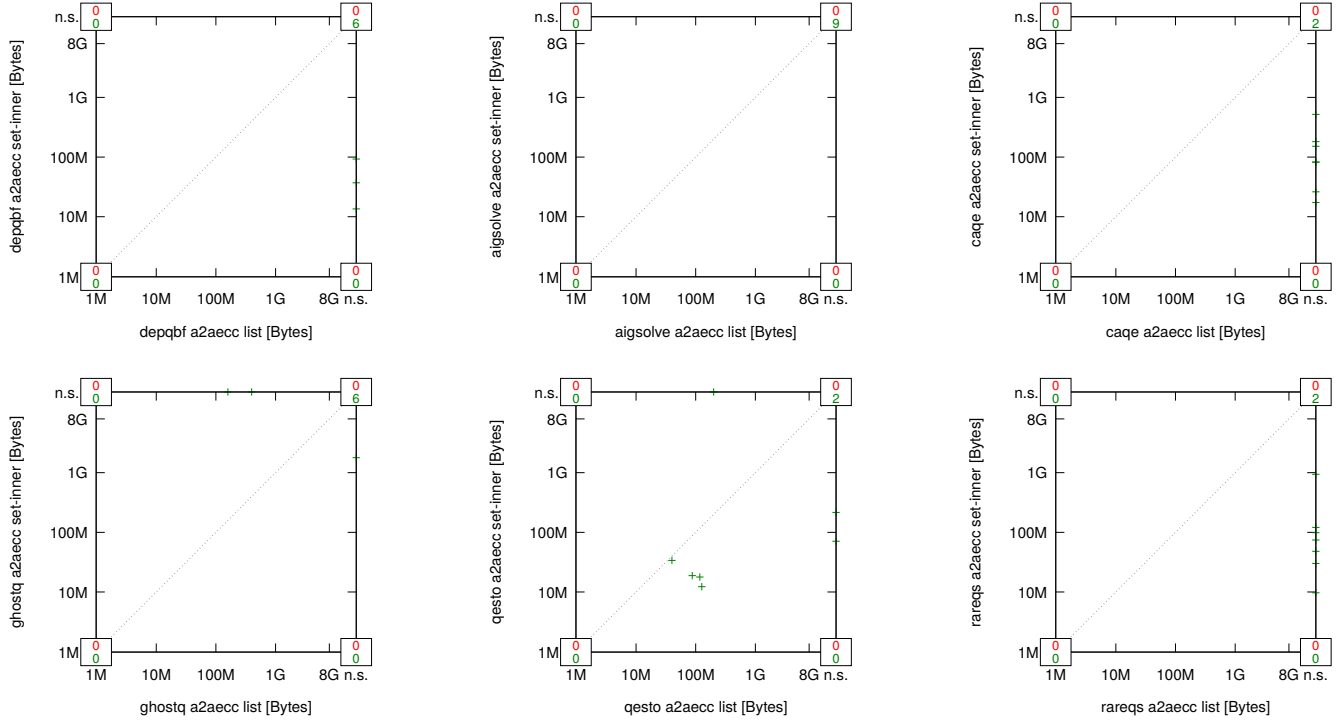


Fig. 1650: Suite Preusser ($n = 12$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

38) *qbfeval12* ($n = 17$):

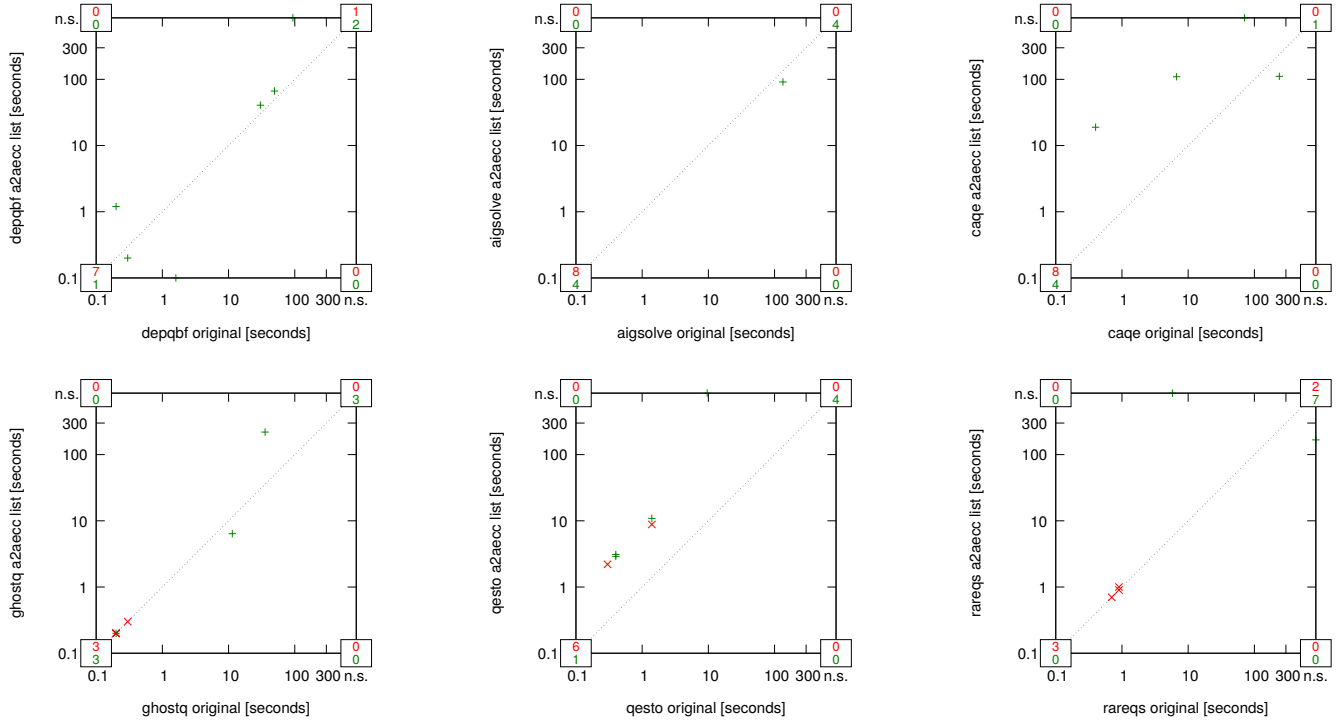


Fig. 1651: Suite *qbfeval12* ($n = 17$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

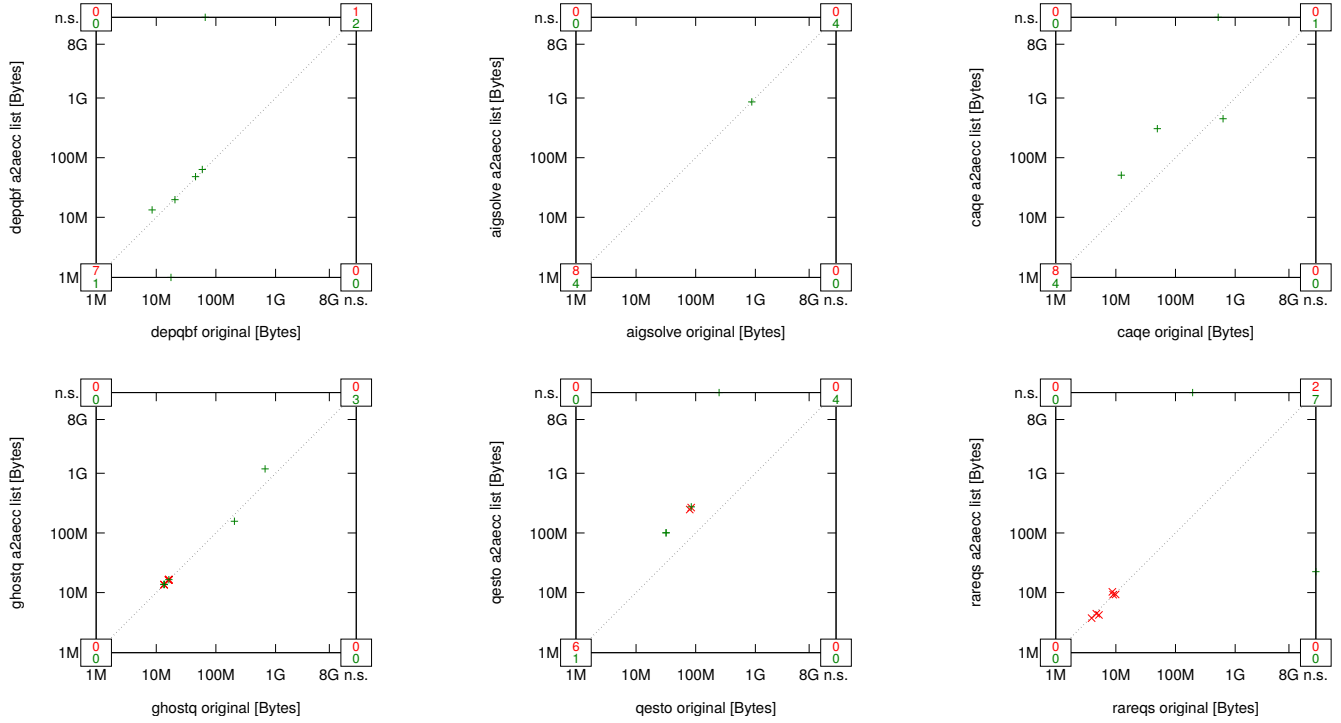


Fig. 1652: Suite *qbfeval12* ($n = 17$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

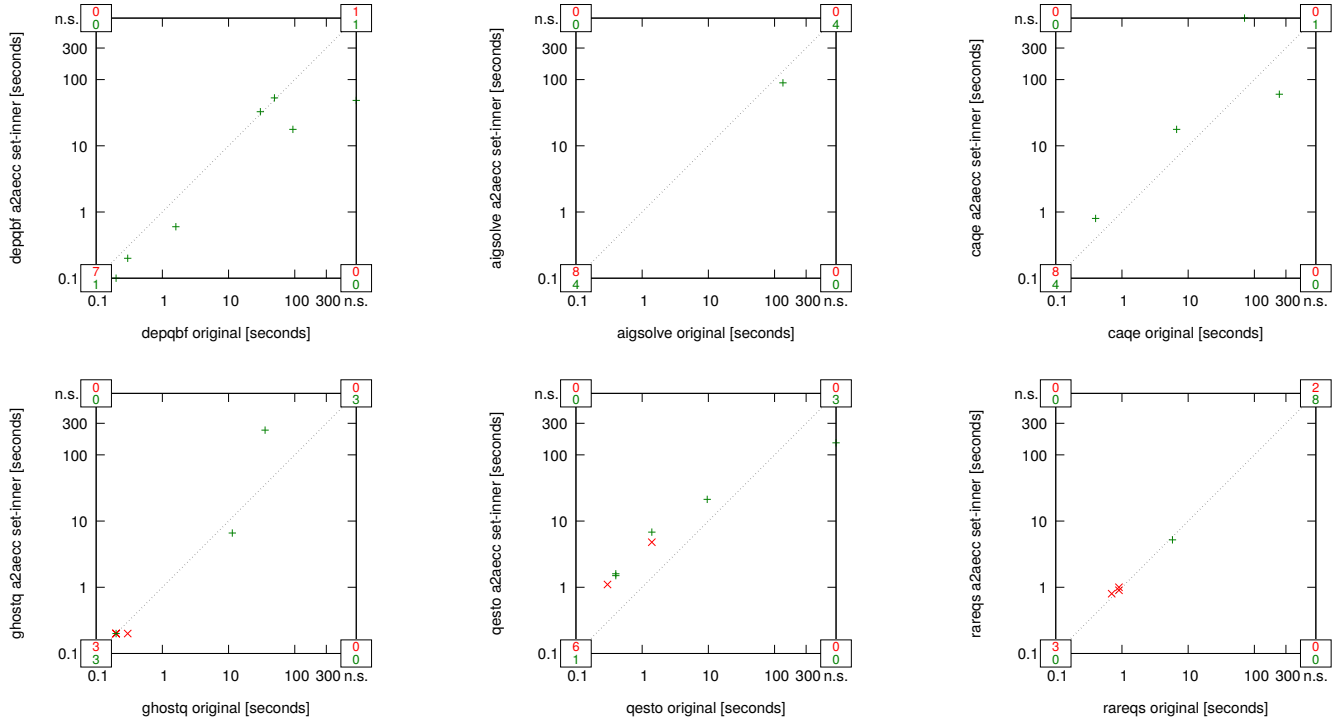


Fig. 1653: Suite qbfeval12 ($n = 17$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

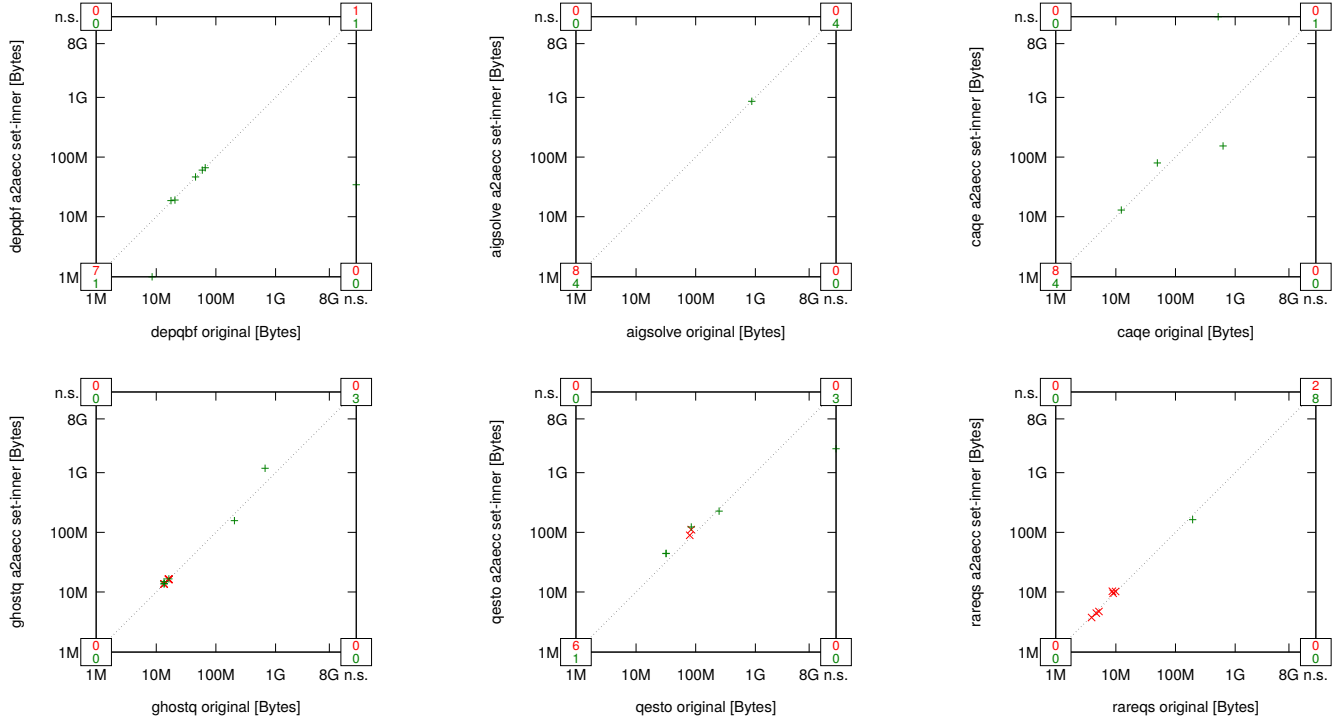


Fig. 1654: Suite qbfeval12 ($n = 17$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

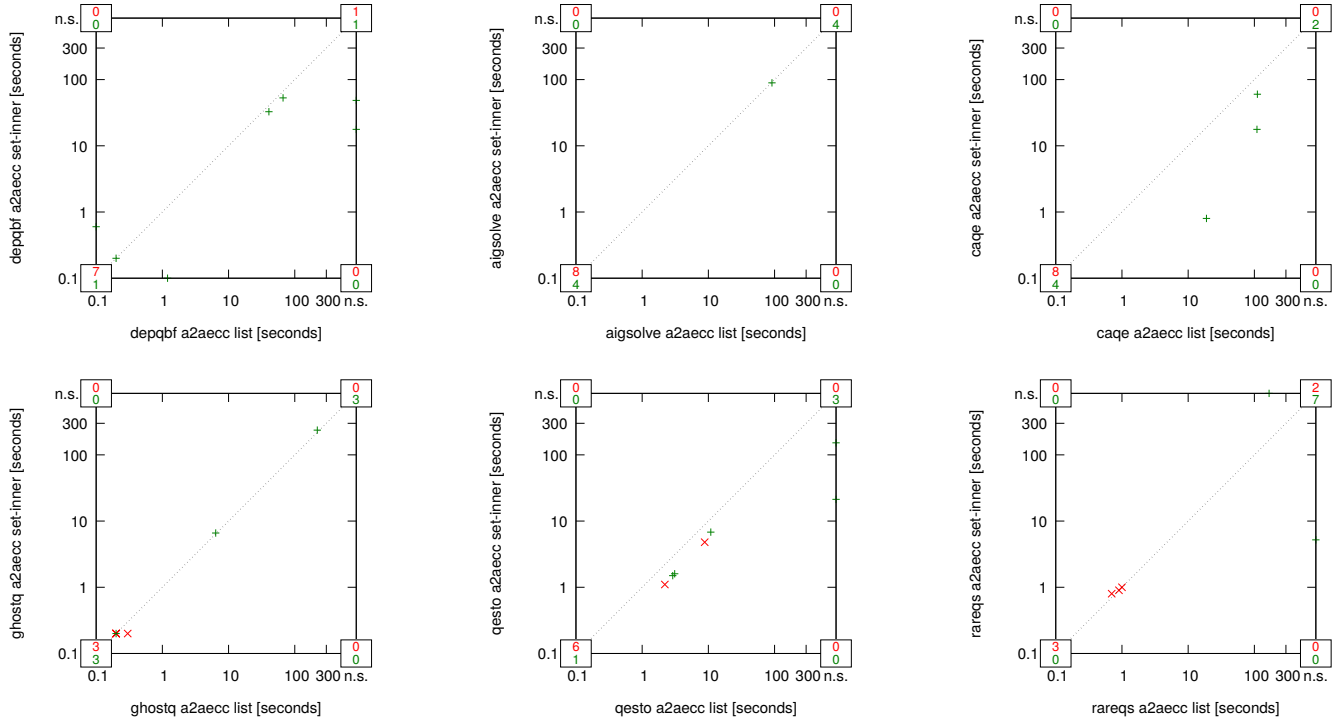


Fig. 1655: Suite qbfeval12 ($n = 17$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

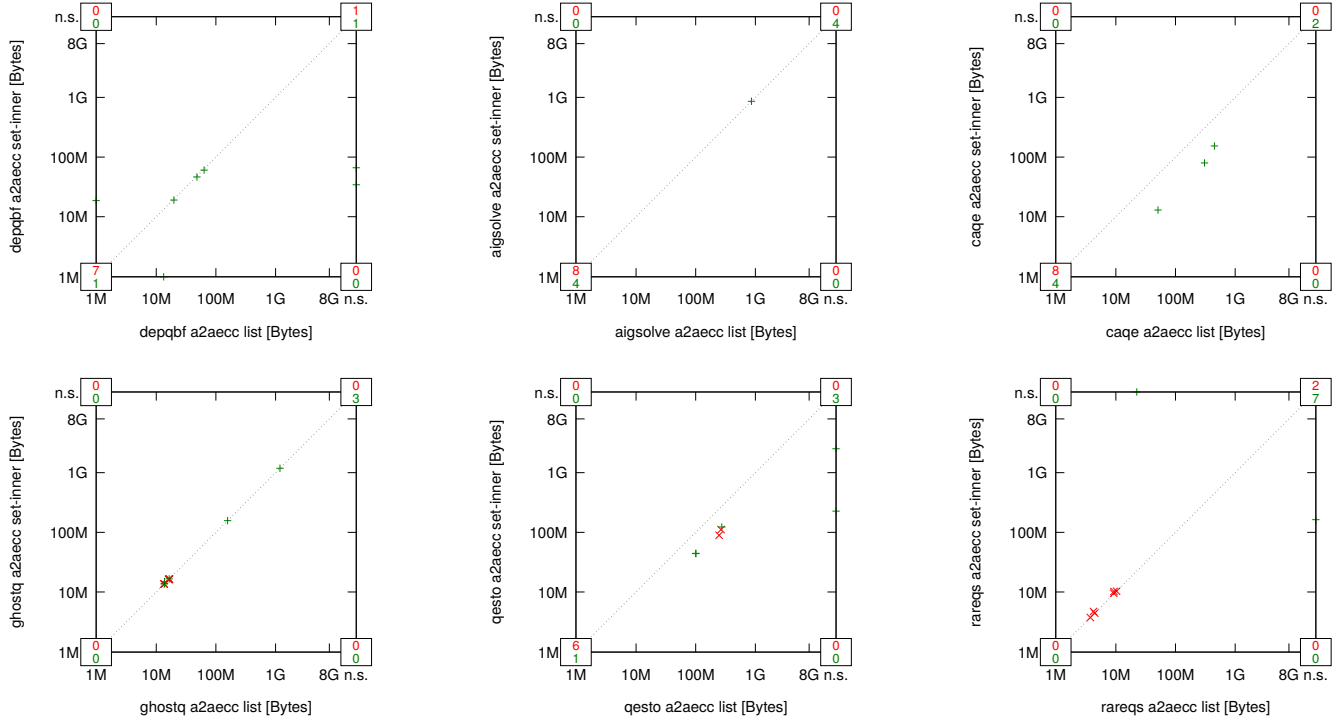


Fig. 1656: Suite qbfeval12 ($n = 17$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

39) Rabe ($n = 14$):

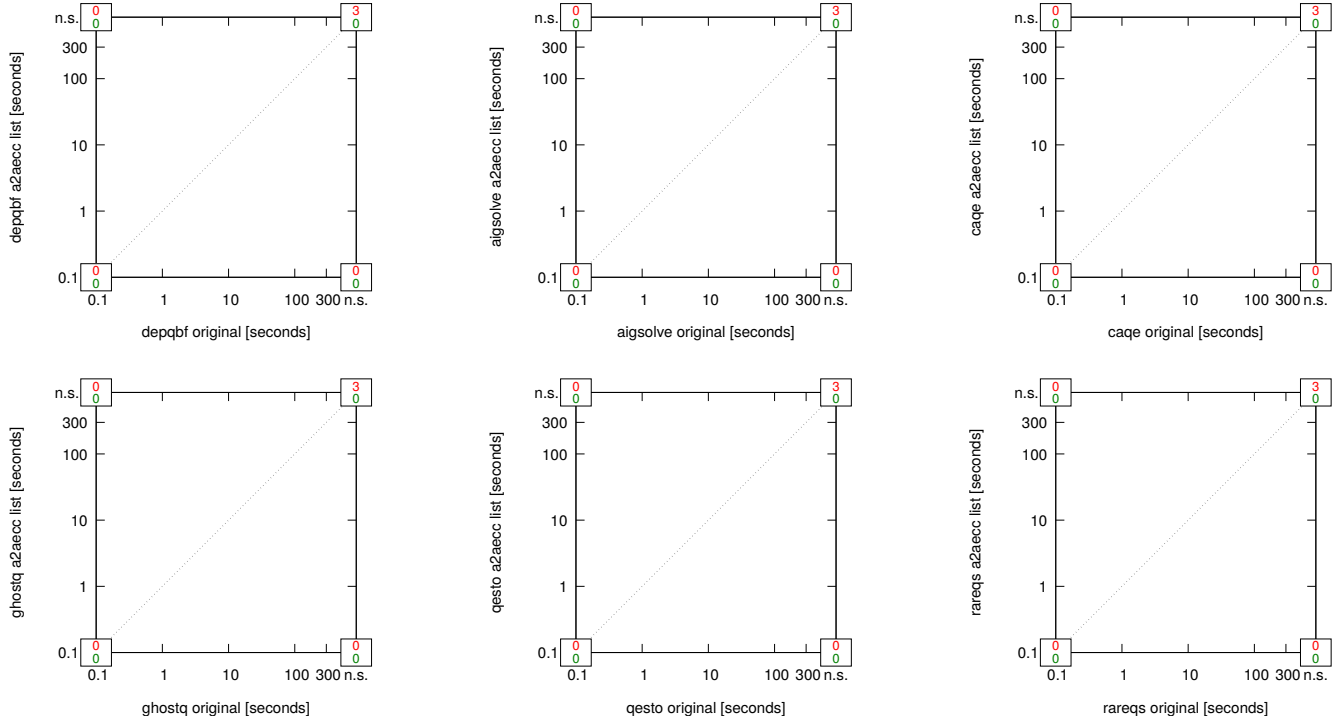


Fig. 1657: Suite Rabe ($n = 14$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

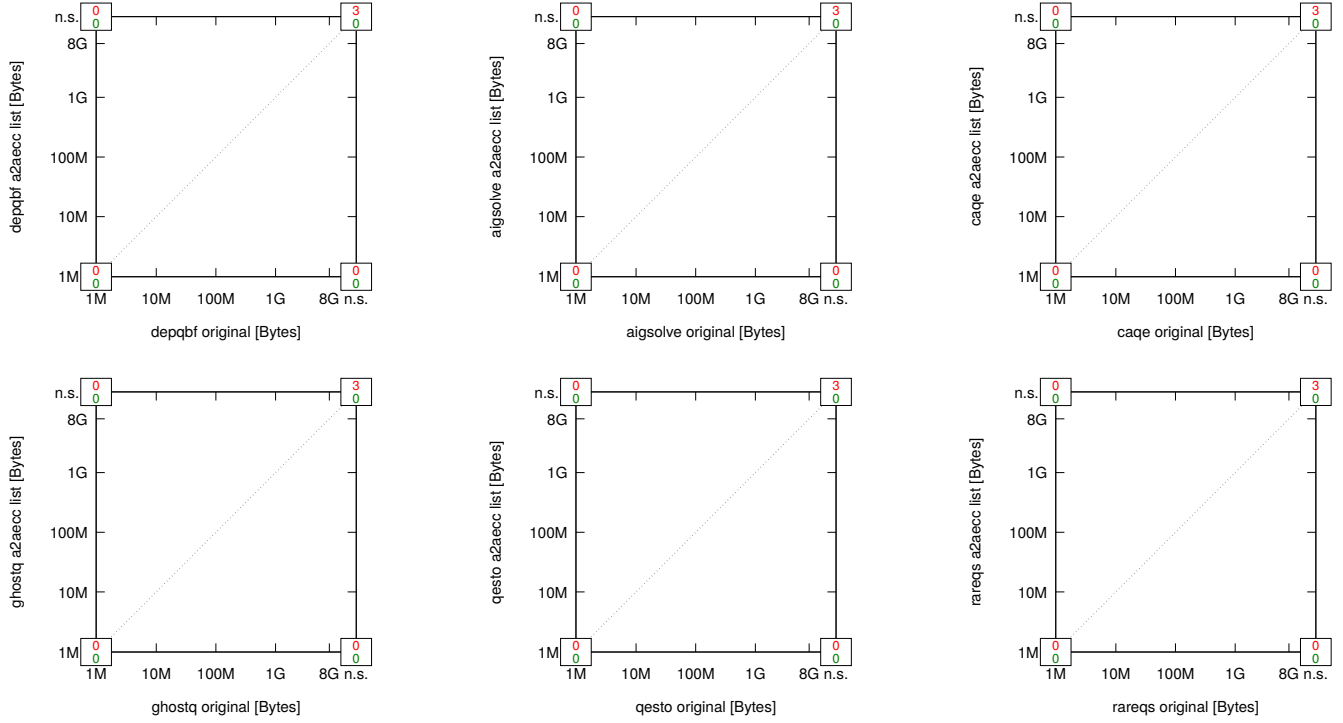


Fig. 1658: Suite Rabe ($n = 14$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

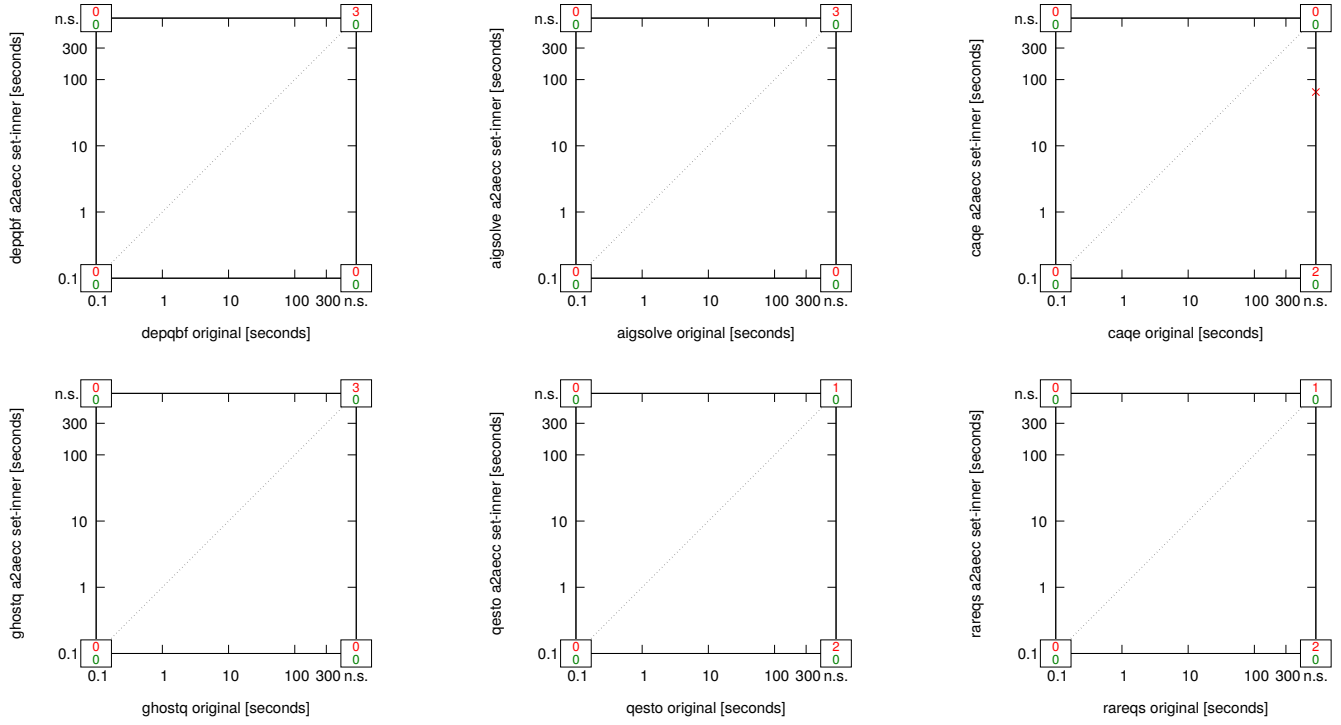


Fig. 1659: Suite Rabe ($n = 14$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

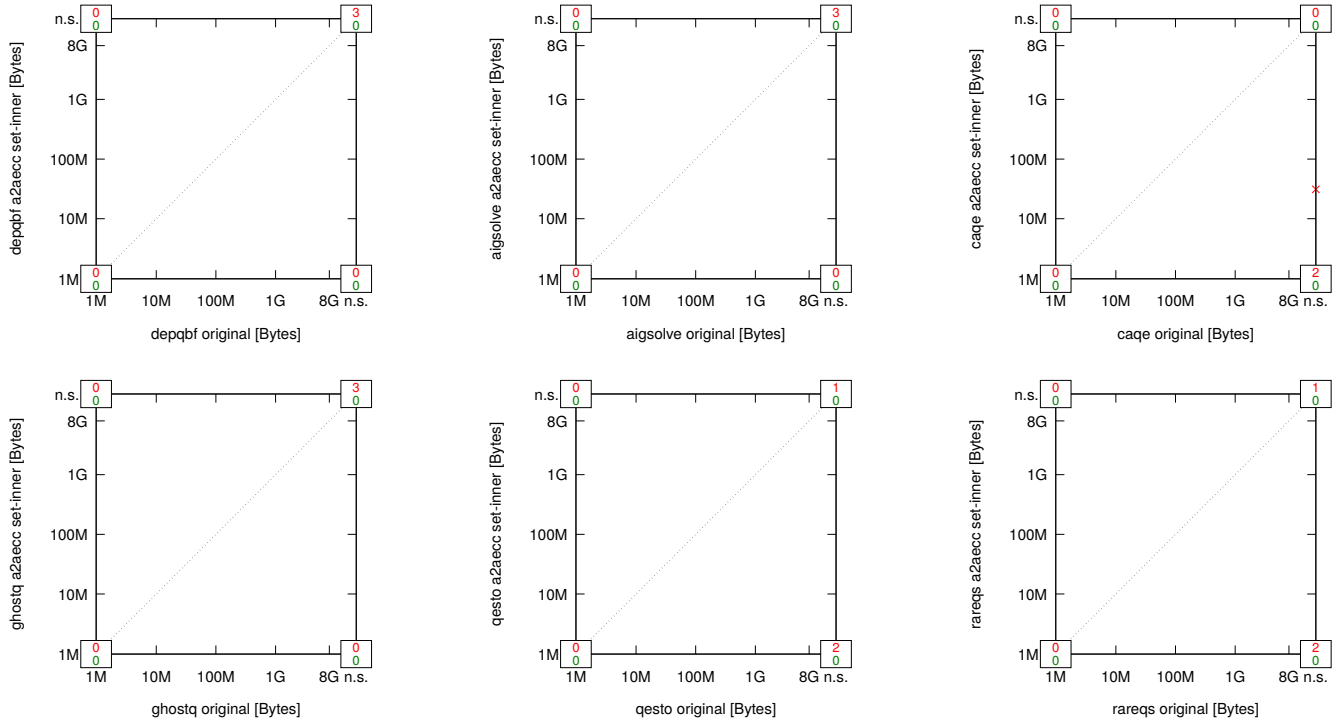


Fig. 1660: Suite Rabe ($n = 14$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

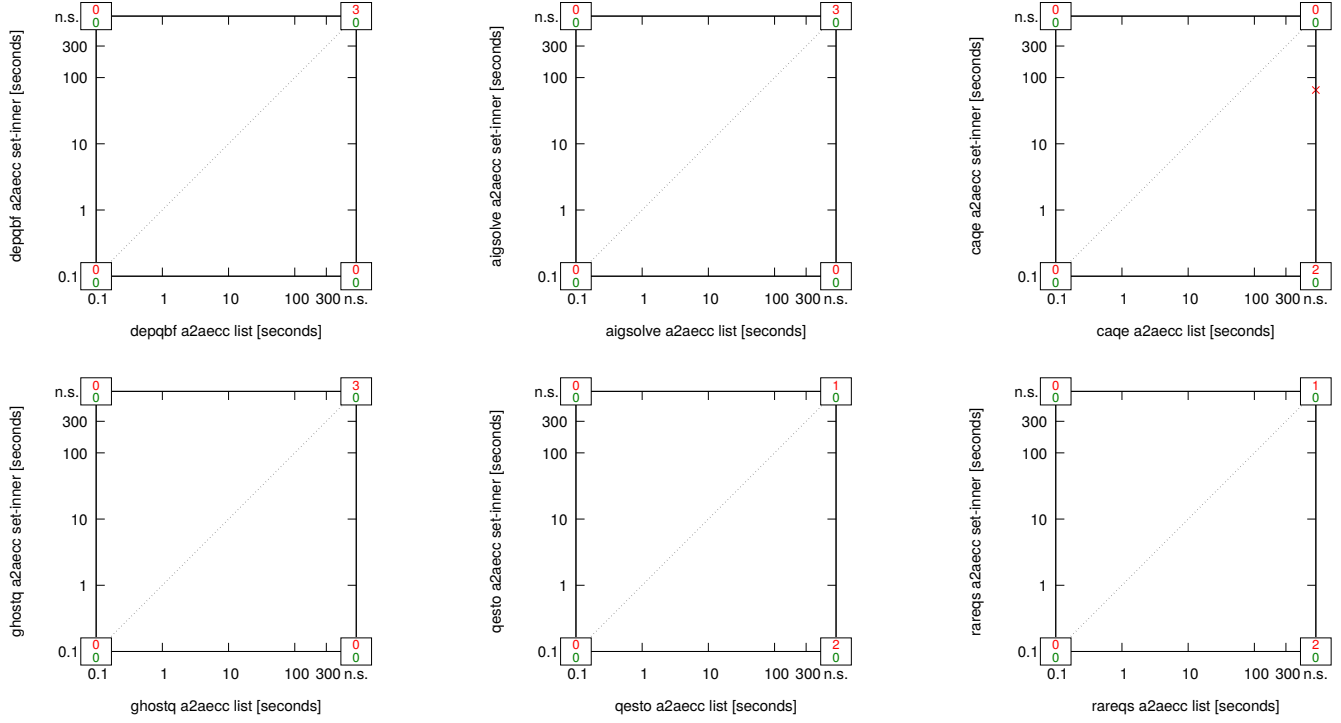


Fig. 1661: Suite Rabe ($n = 14$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

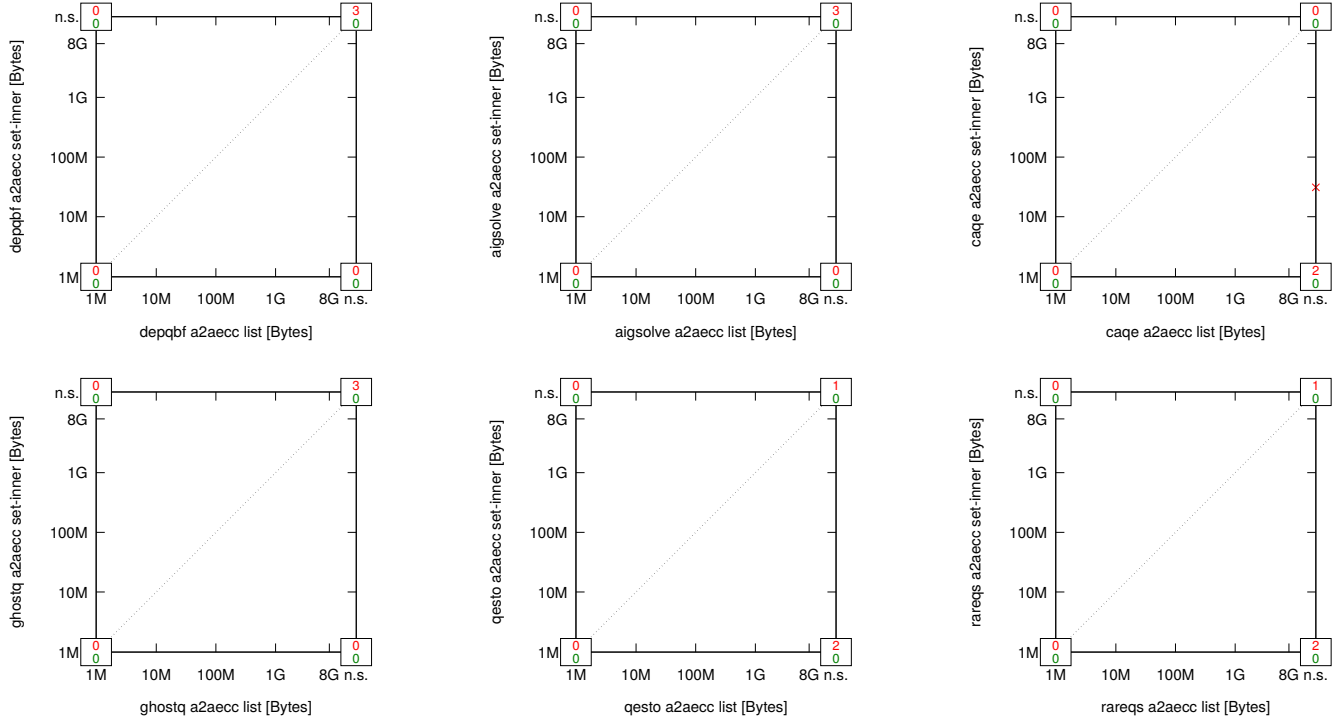


Fig. 1662: Suite Rabe ($n = 14$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

40) Rintanen ($n = 131$):

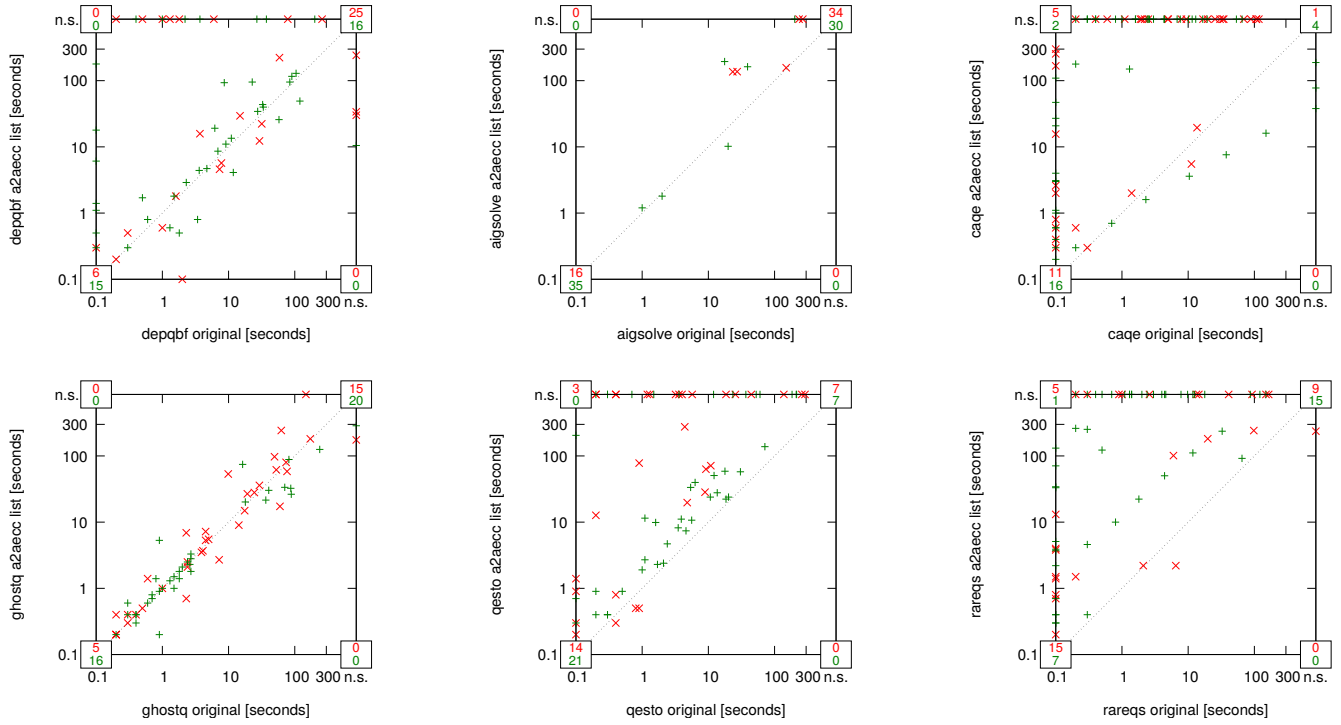


Fig. 1663: Suite Rintanen ($n = 131$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

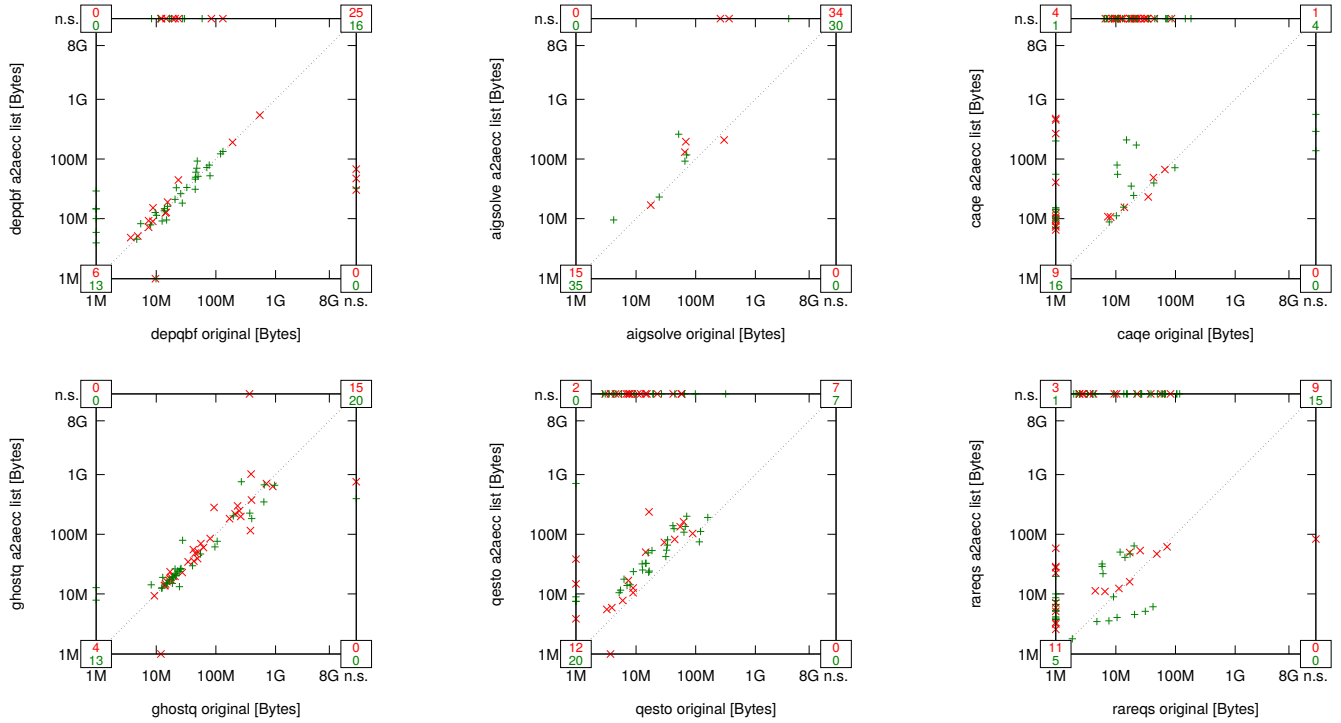


Fig. 1664: Suite Rintanen ($n = 131$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

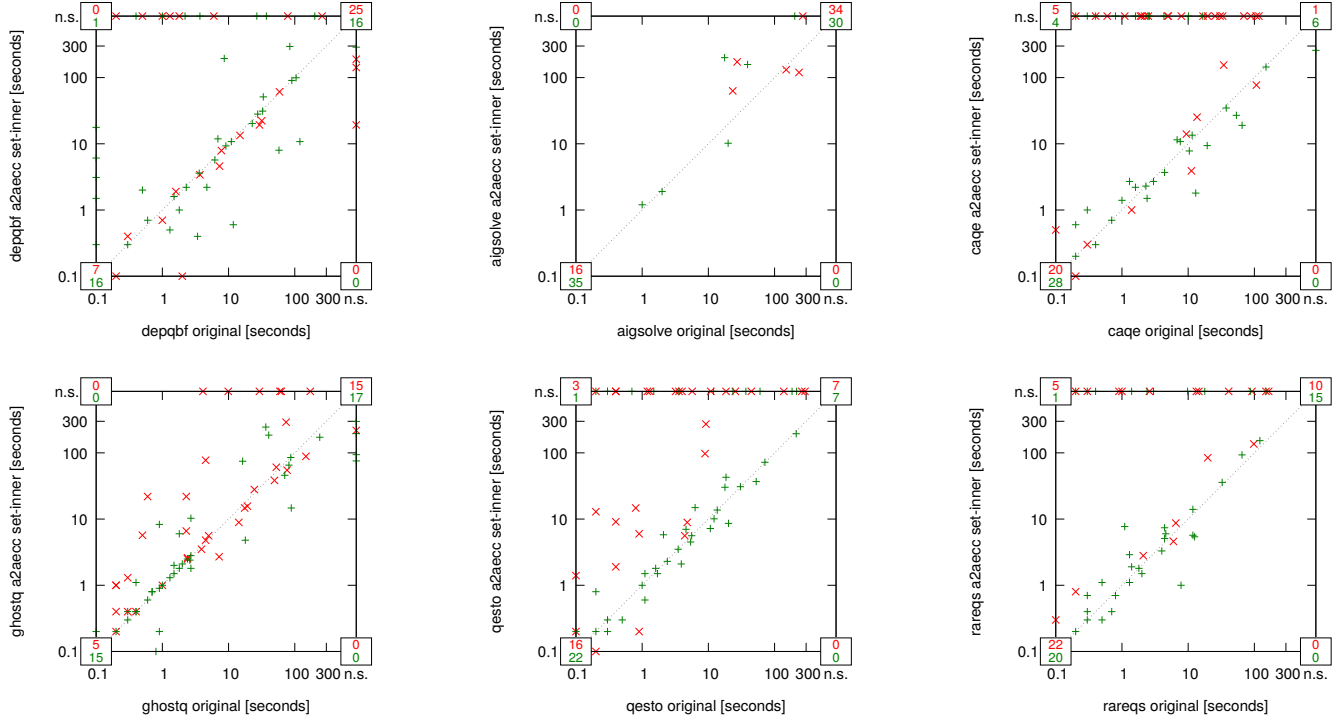


Fig. 1665: Suite Rintanen ($n = 131$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

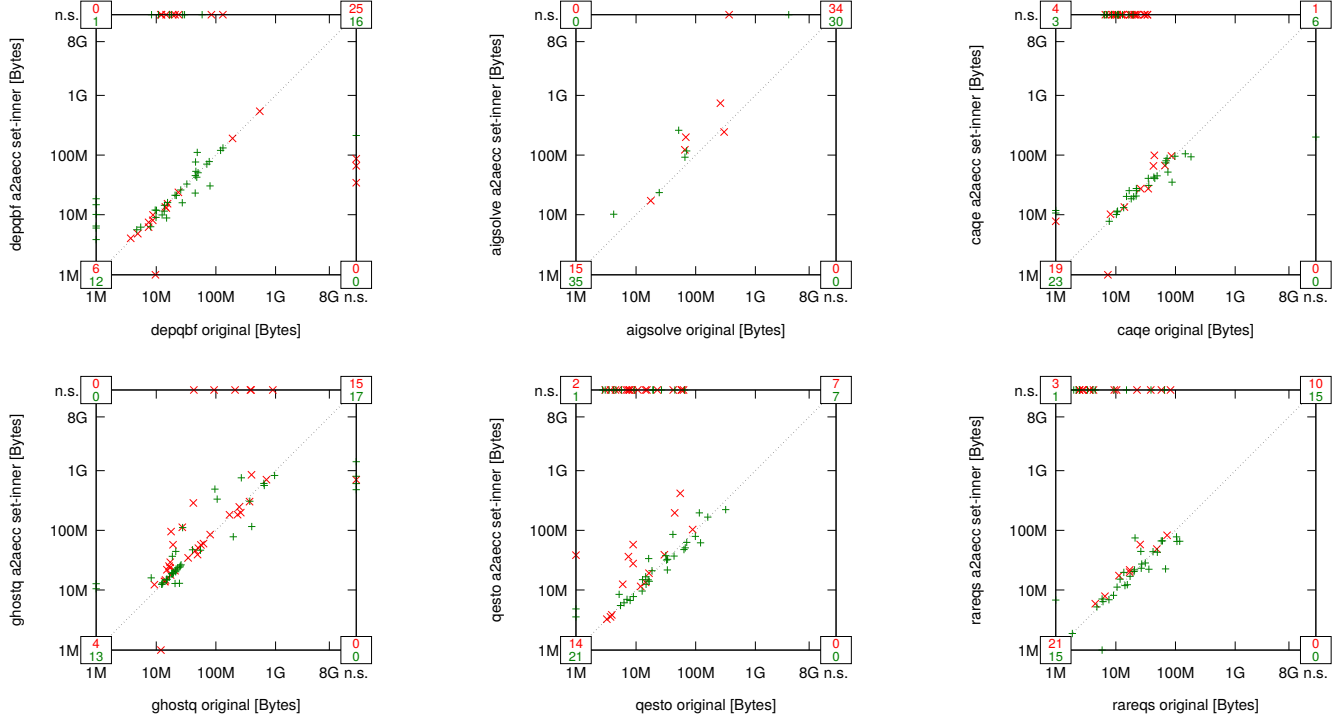


Fig. 1666: Suite Rintanen ($n = 131$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

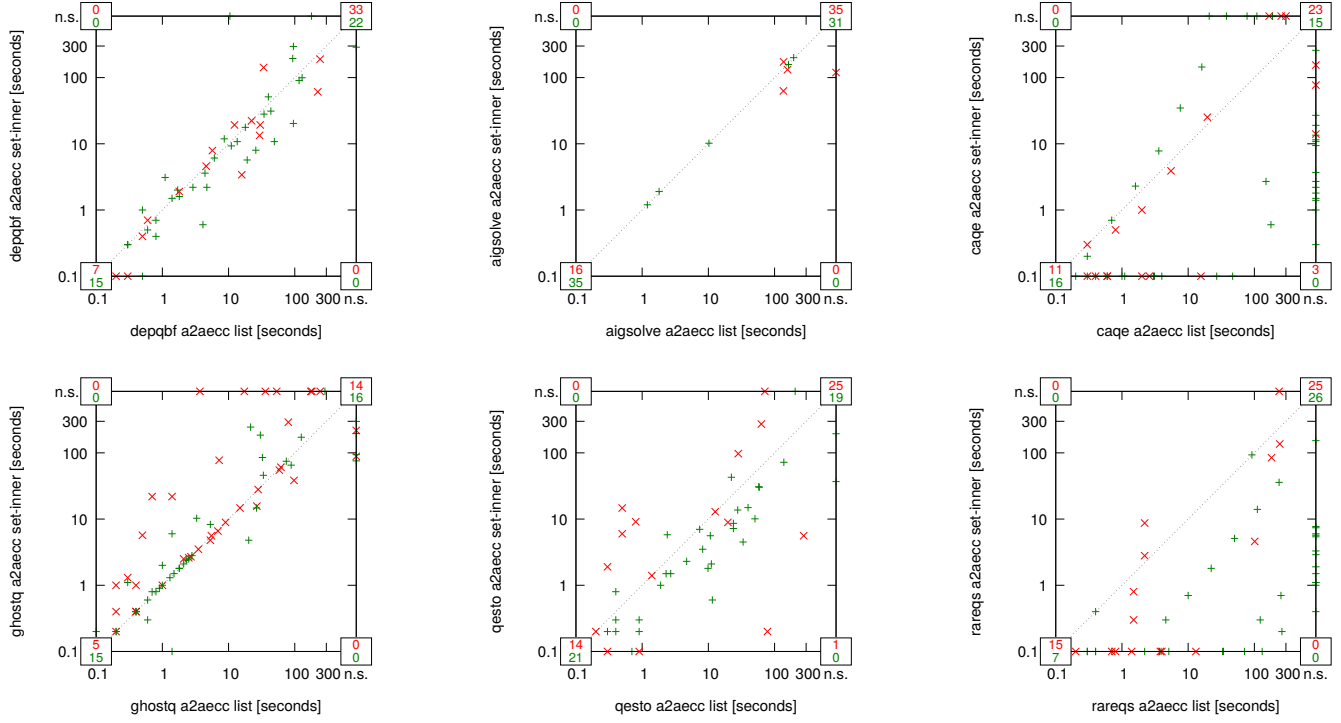


Fig. 1667: Suite Rintanen ($n = 131$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

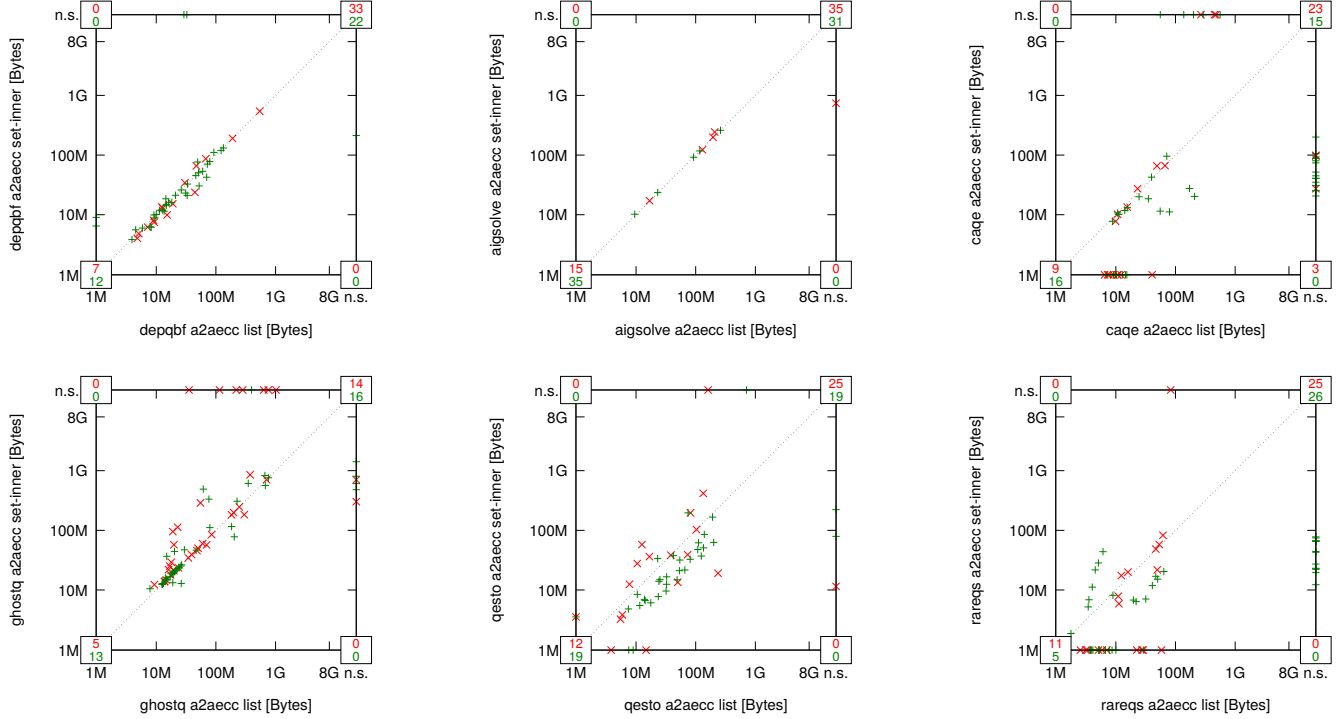


Fig. 1668: Suite Rintanen ($n = 131$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

41) Sauer-Reimer ($n = 193$):

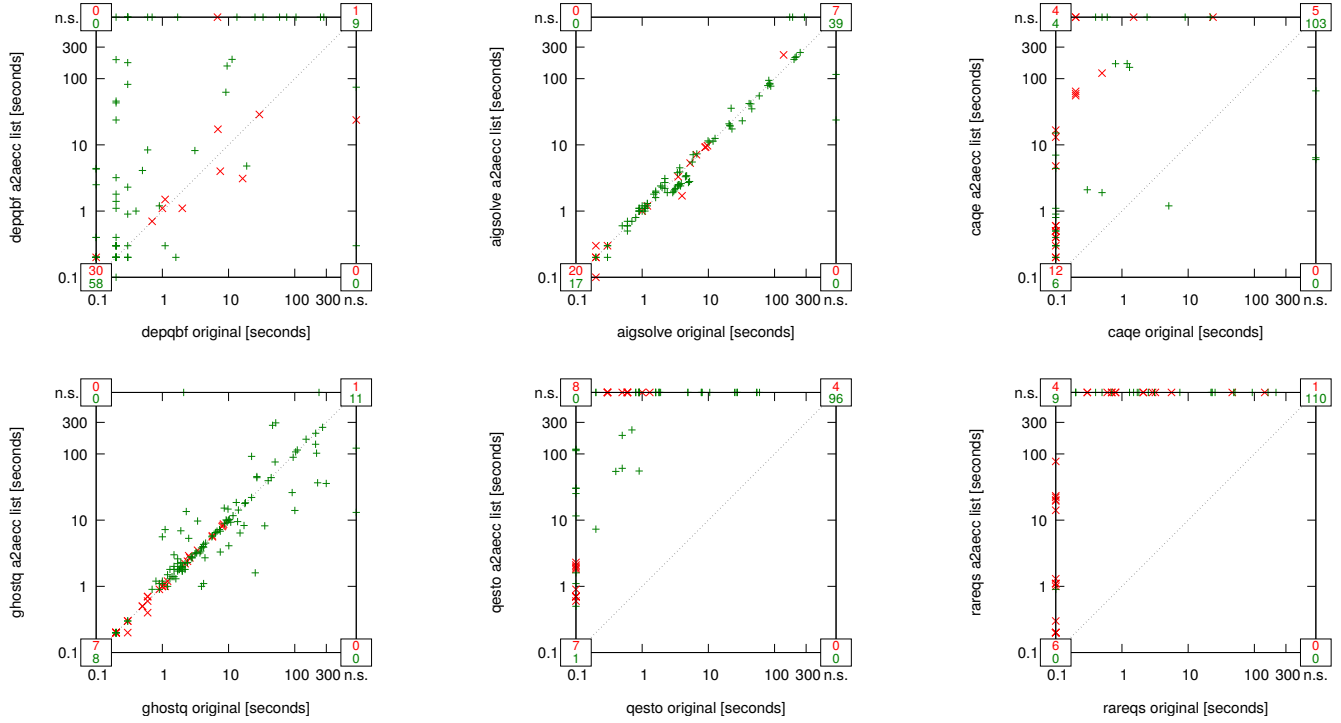


Fig. 1669: Suite Sauer-Reimer ($n = 193$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

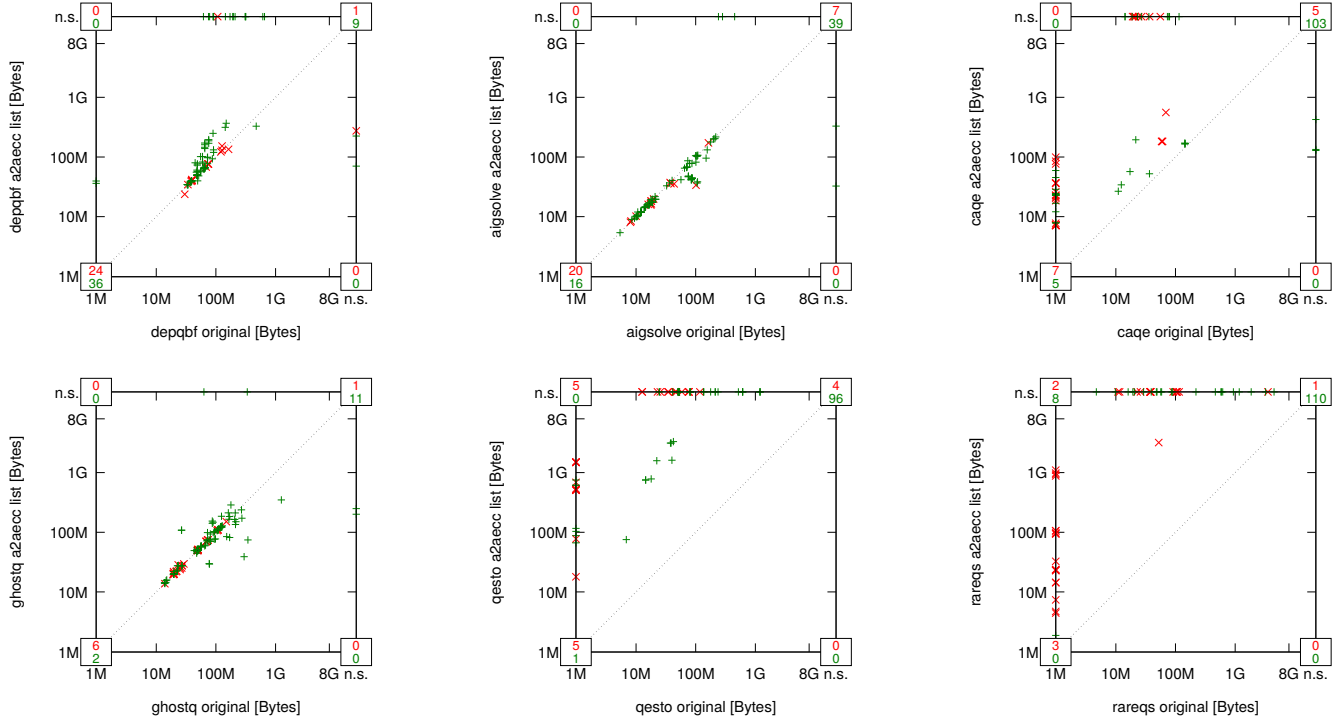


Fig. 1670: Suite Sauer-Reimer ($n = 193$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

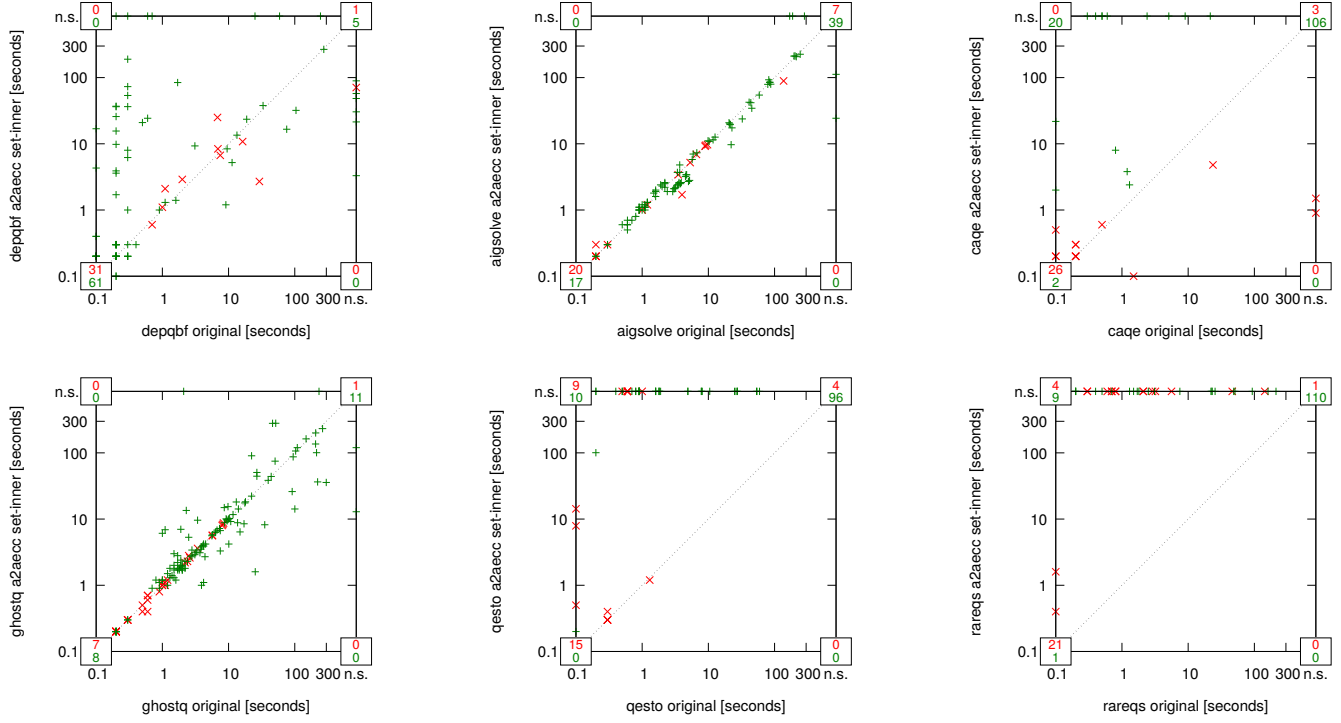


Fig. 1671: Suite Sauer-Reimer ($n = 193$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

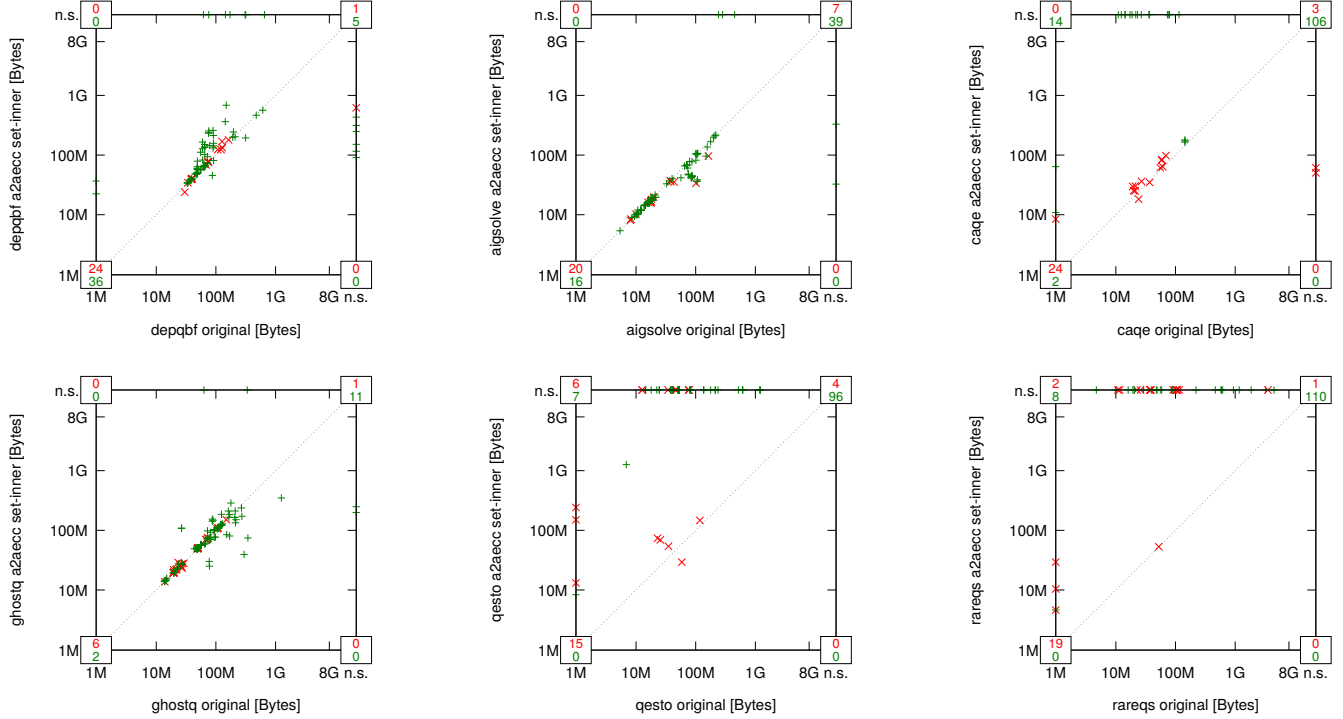


Fig. 1672: Suite Sauer-Reimer ($n = 193$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

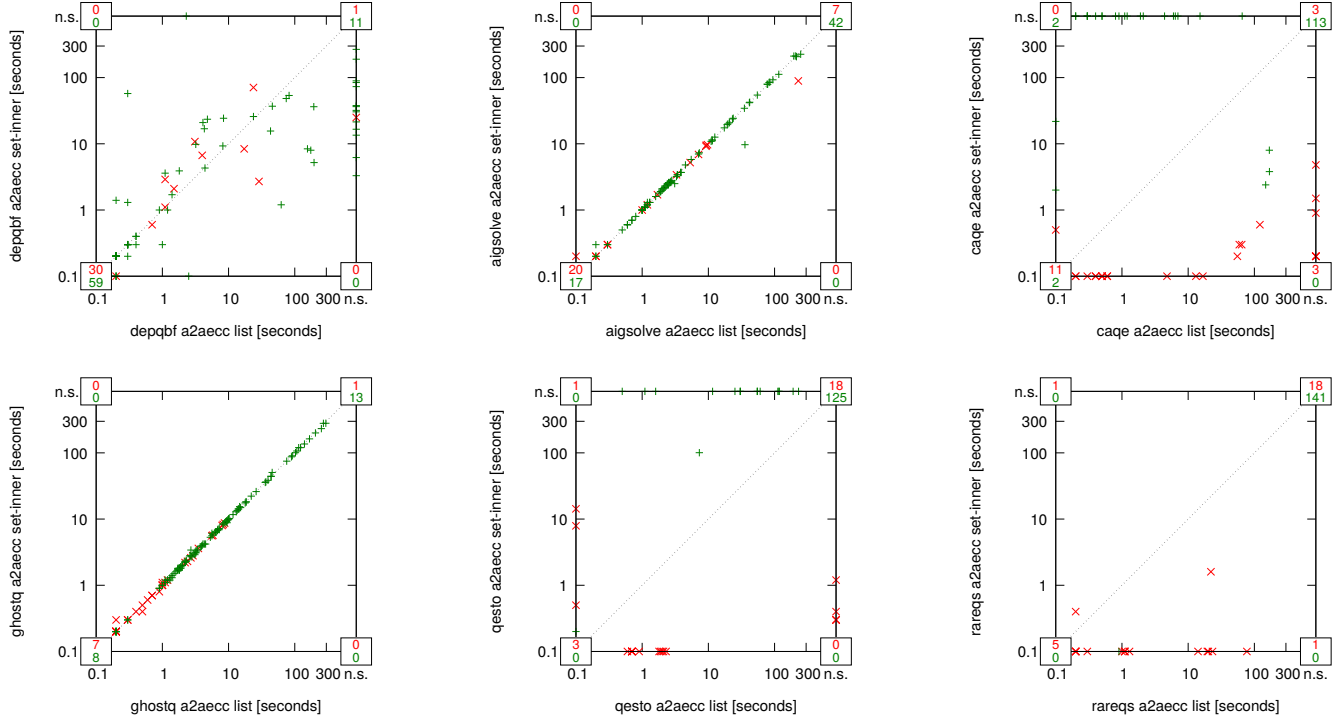


Fig. 1673: Suite Sauer-Reimer ($n = 193$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

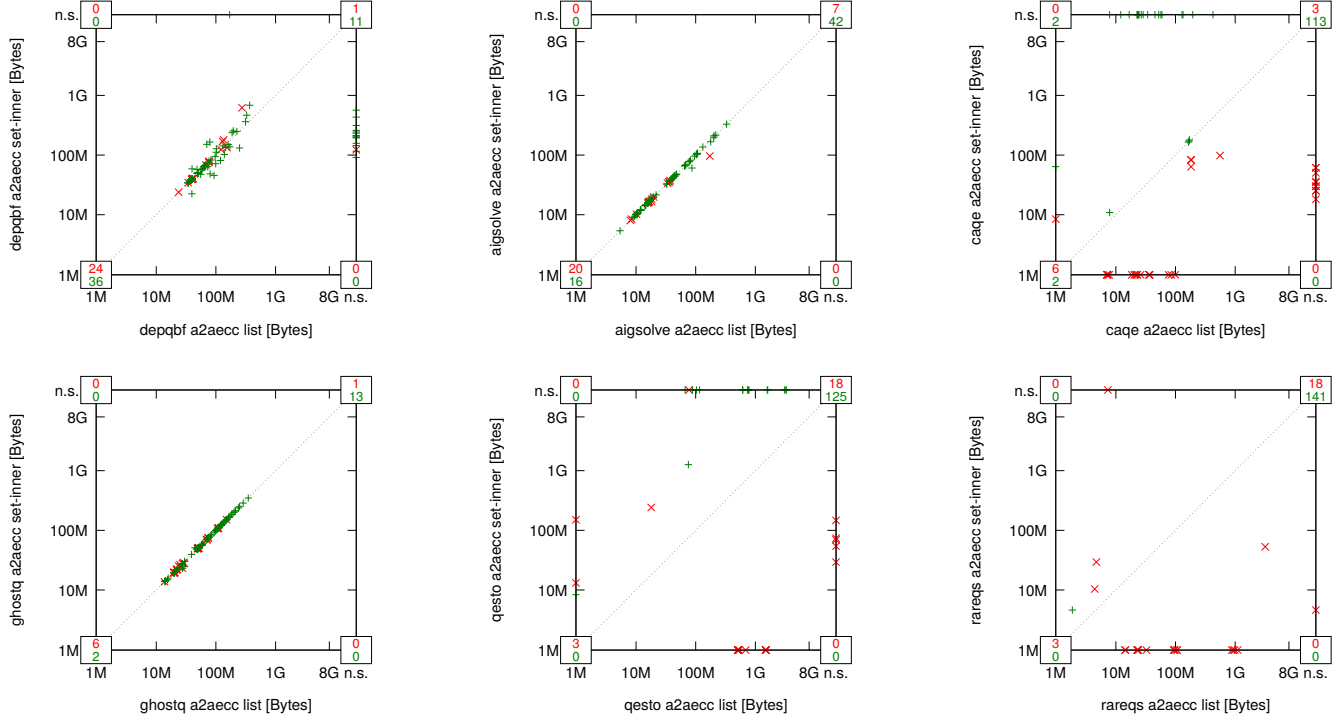


Fig. 1674: Suite Sauer-Reimer ($n = 193$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

42) Scholl-Becker ($n = 64$):

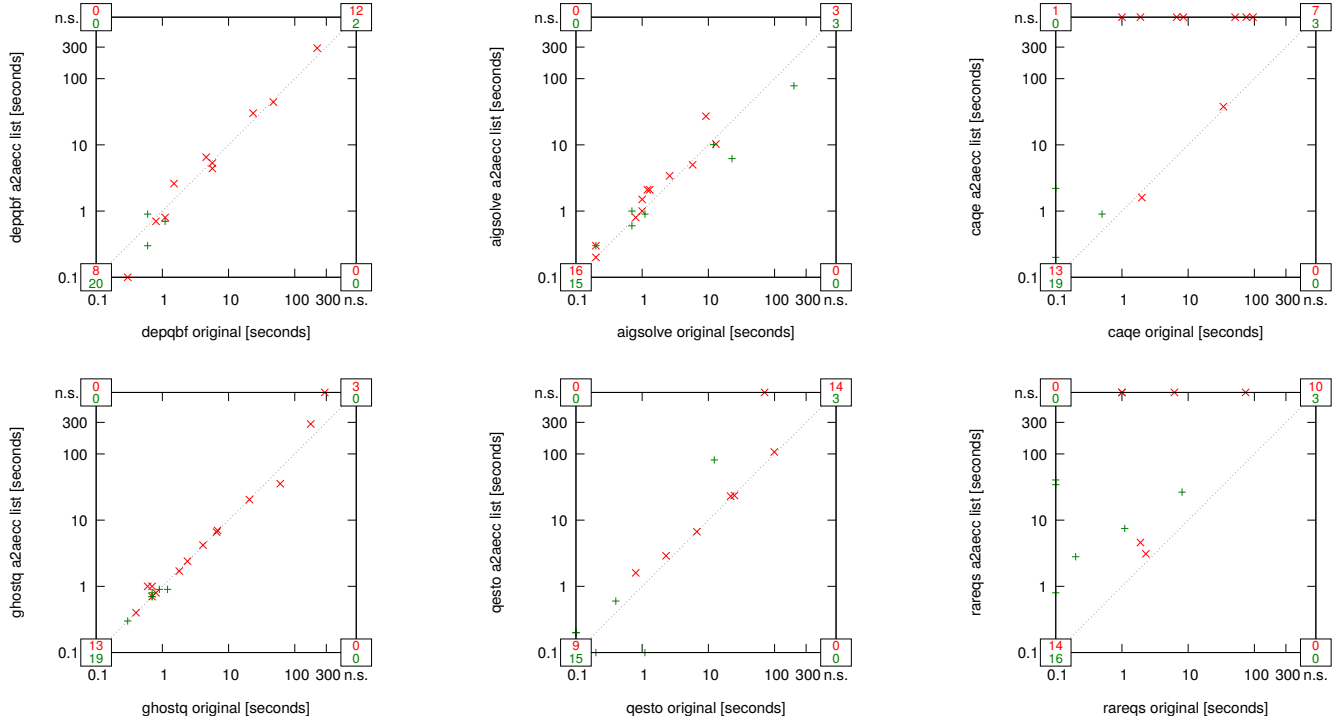


Fig. 1675: Suite Scholl-Becker ($n = 64$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

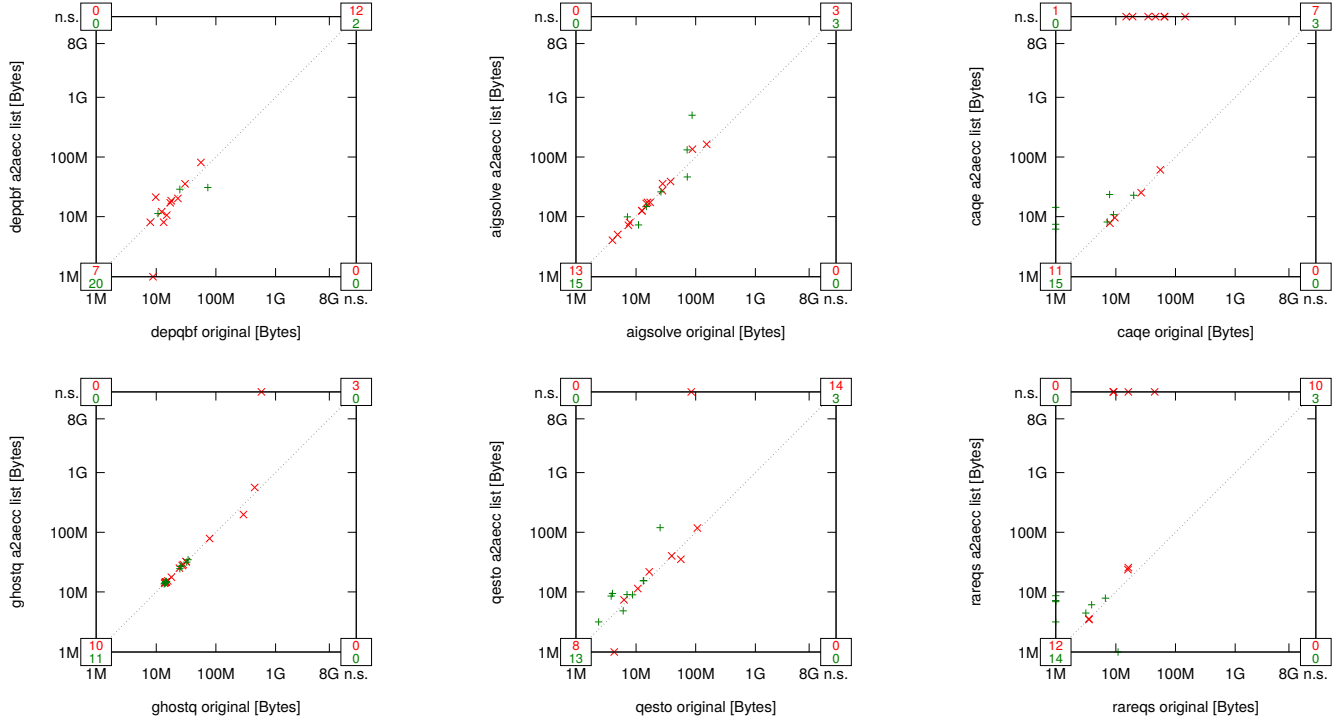


Fig. 1676: Suite Scholl-Becker ($n = 64$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

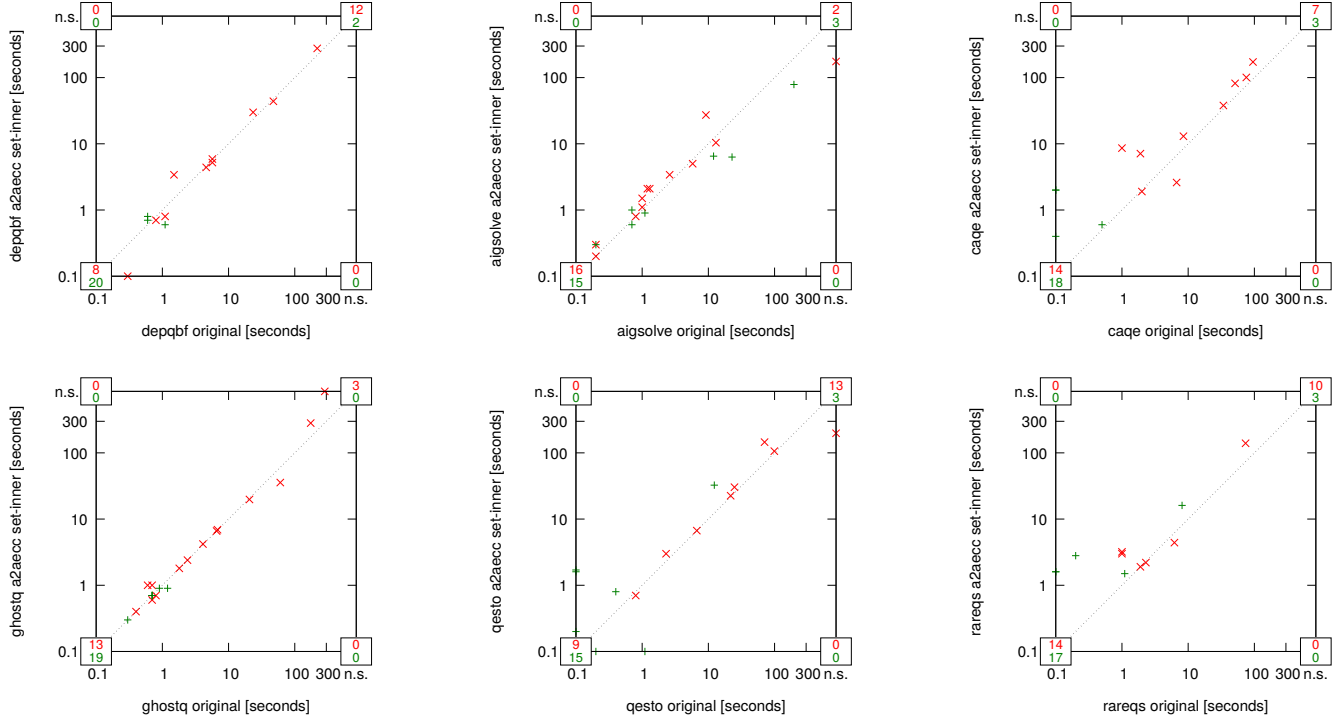


Fig. 1677: Suite Scholl-Becker ($n = 64$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

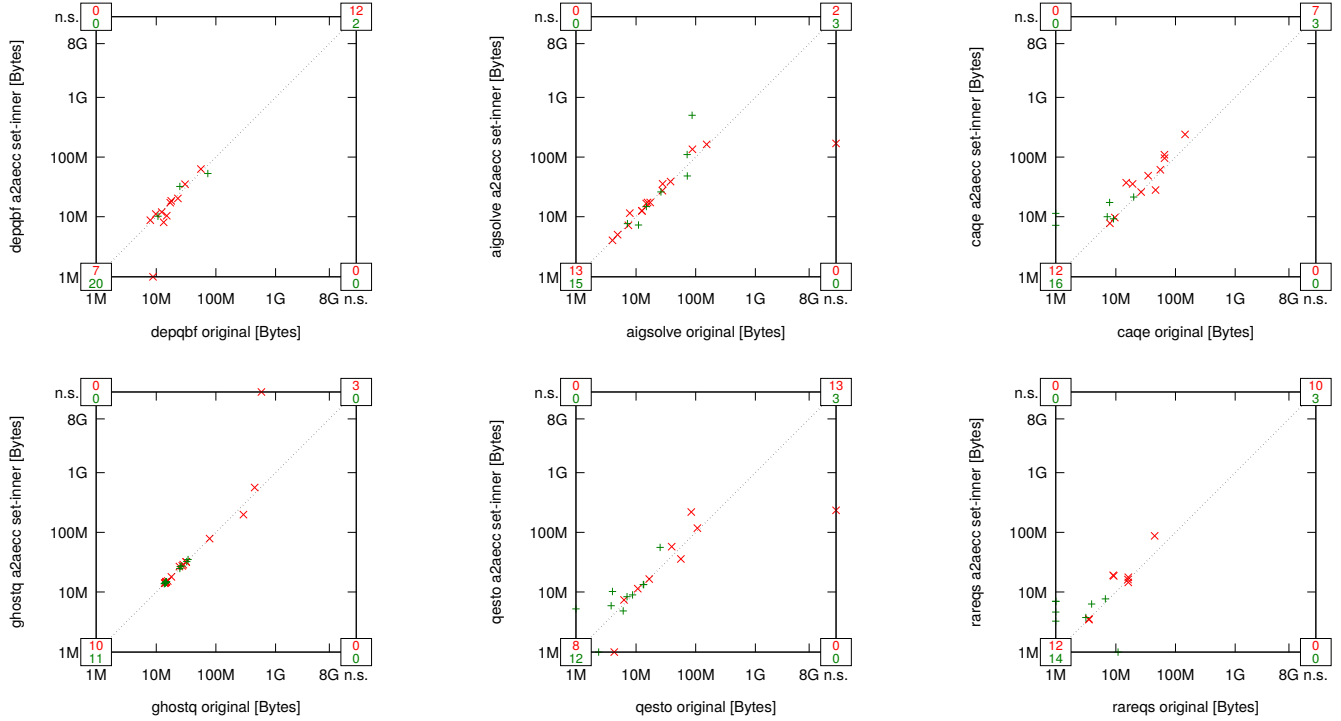


Fig. 1678: Suite Scholl-Becker ($n = 64$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

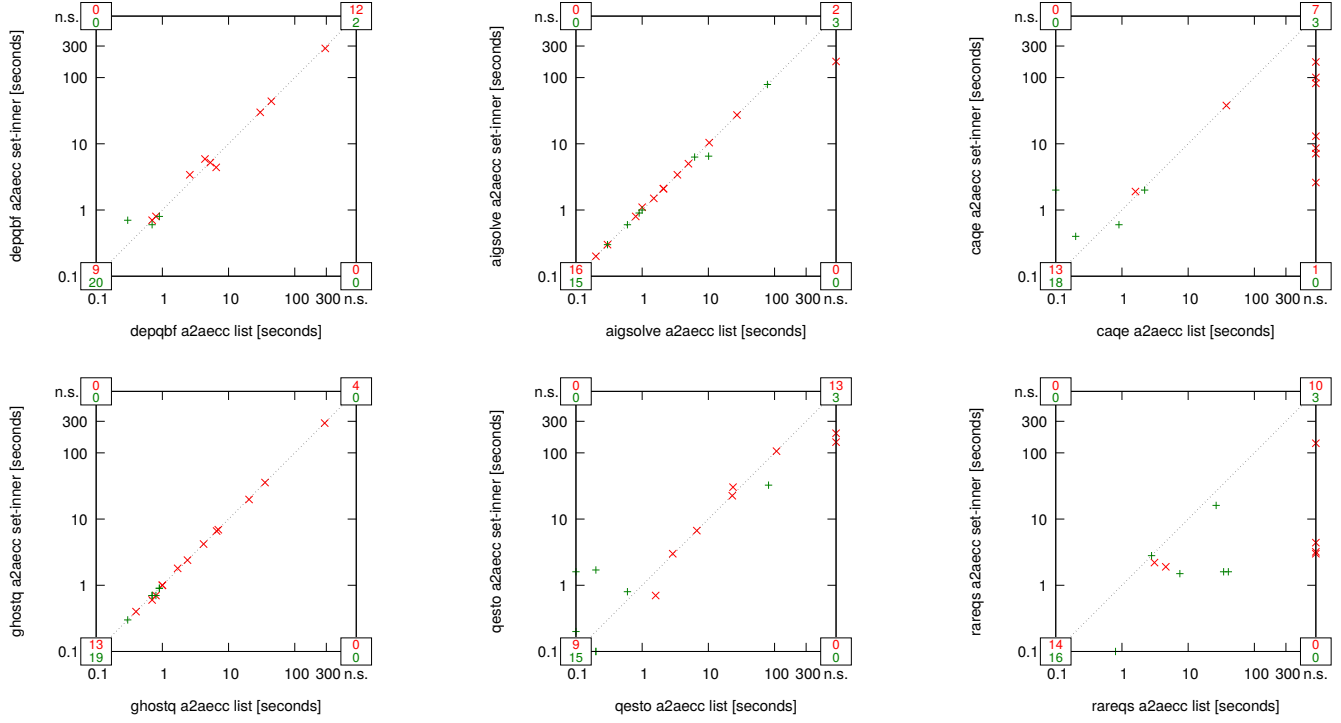


Fig. 1679: Suite Scholl-Becker ($n = 64$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

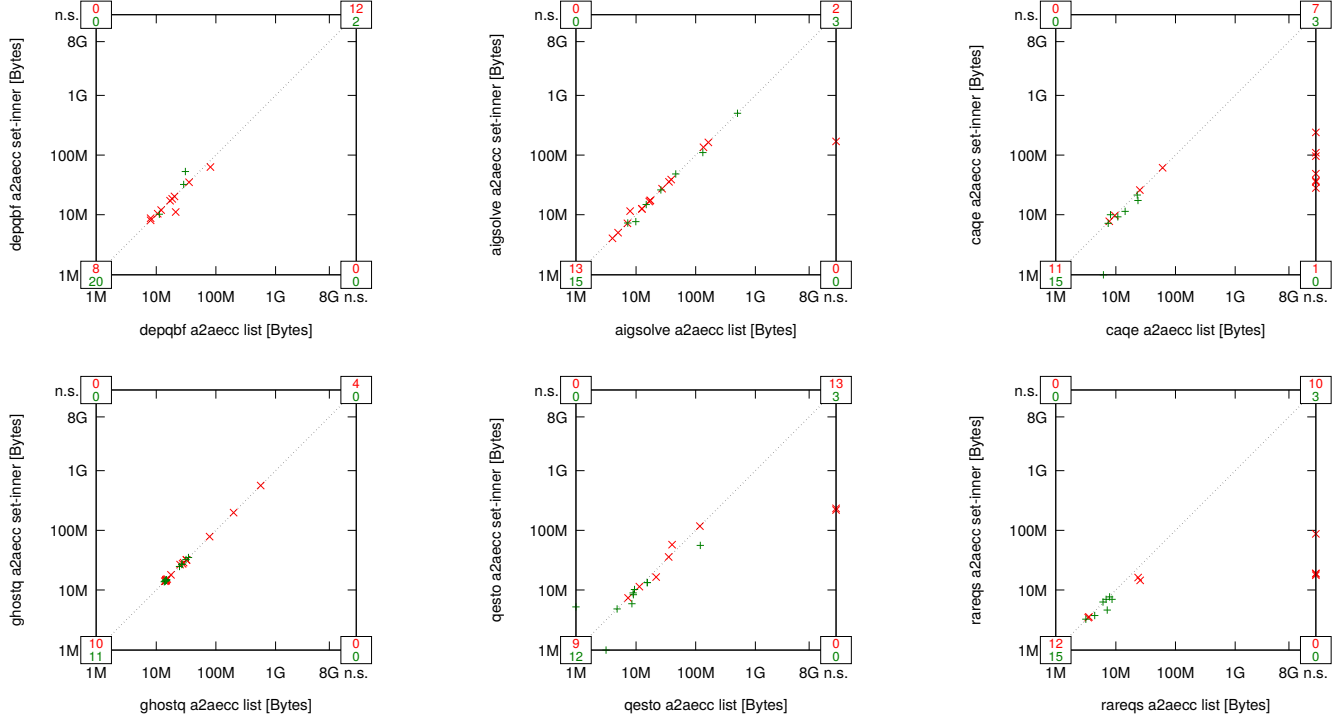


Fig. 1680: Suite Scholl-Becker ($n = 64$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

43) Seidl ($n = 194$):

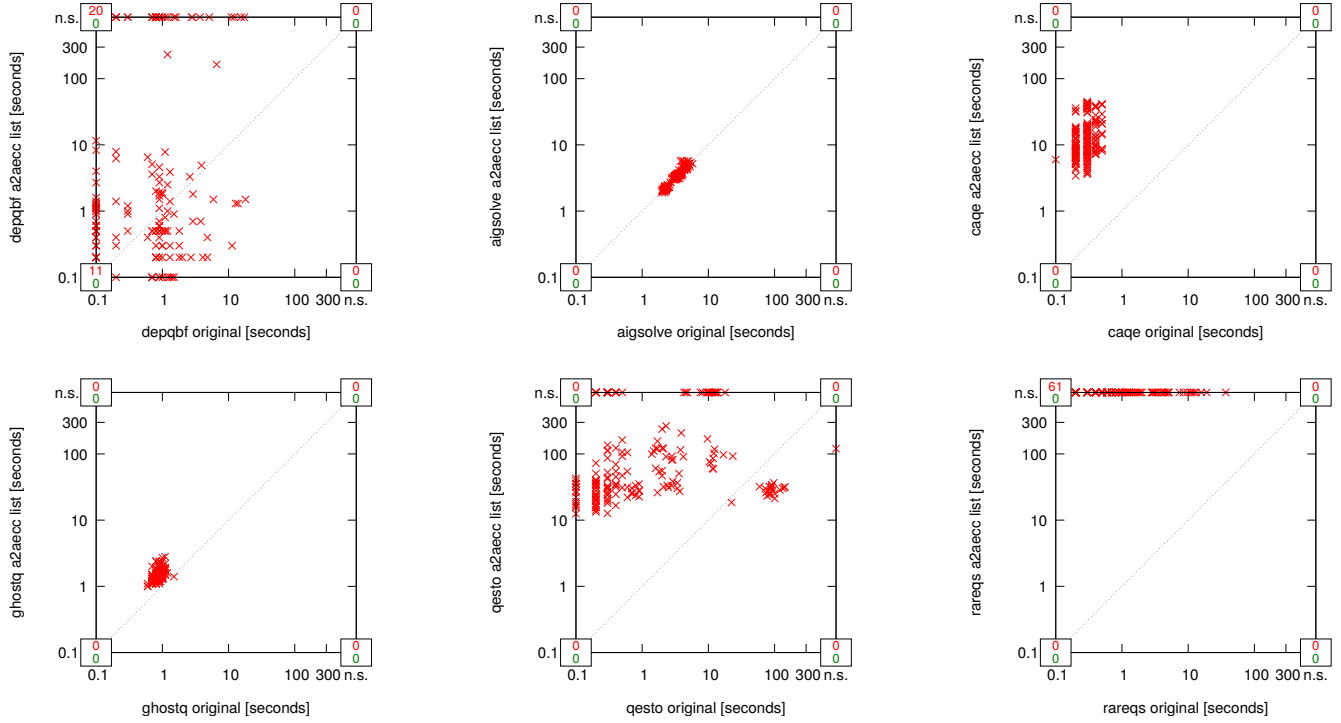


Fig. 1681: Suite Seidl ($n = 194$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

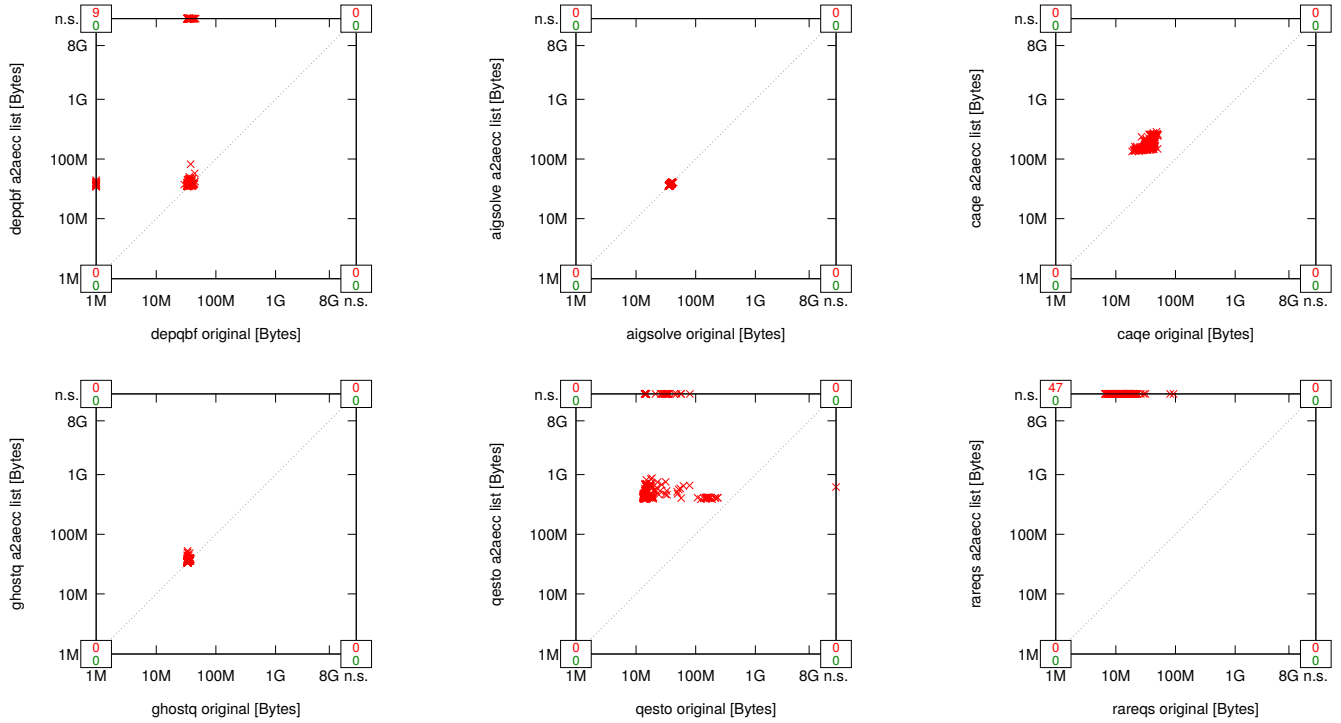


Fig. 1682: Suite Seidl ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

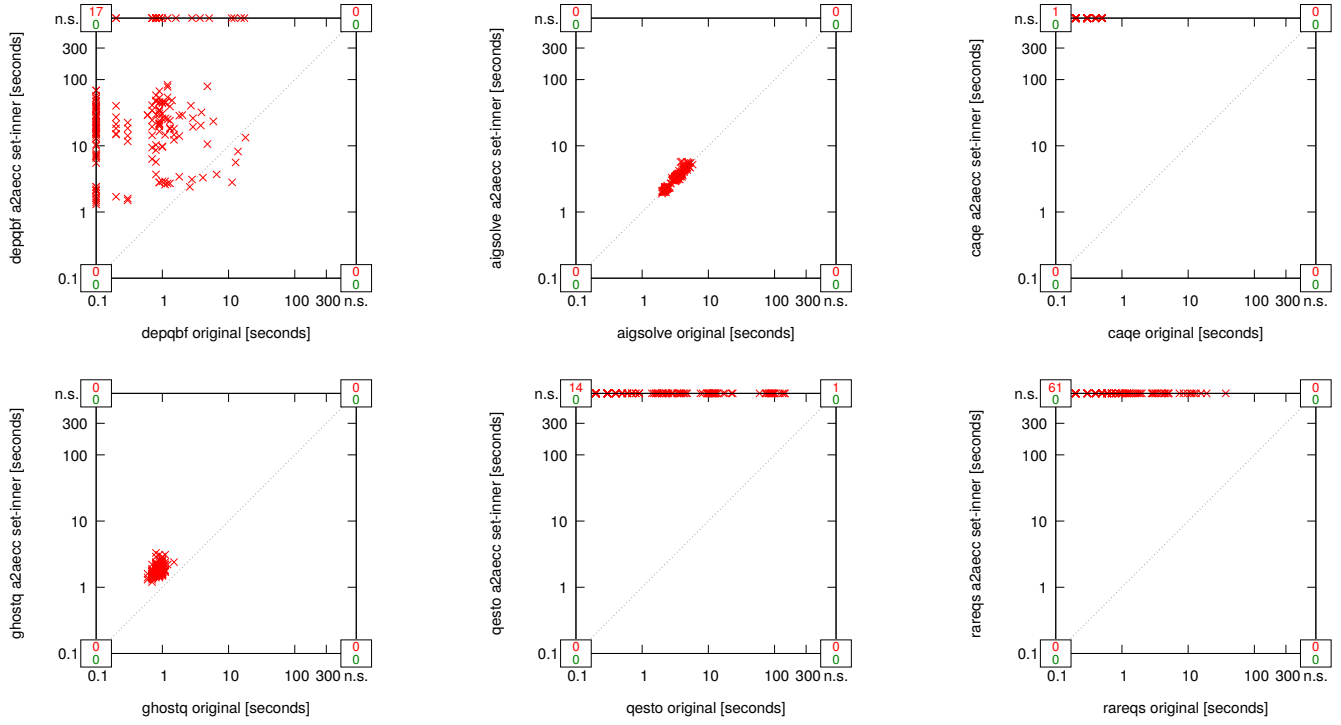


Fig. 1683: Suite Seidl ($n = 194$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

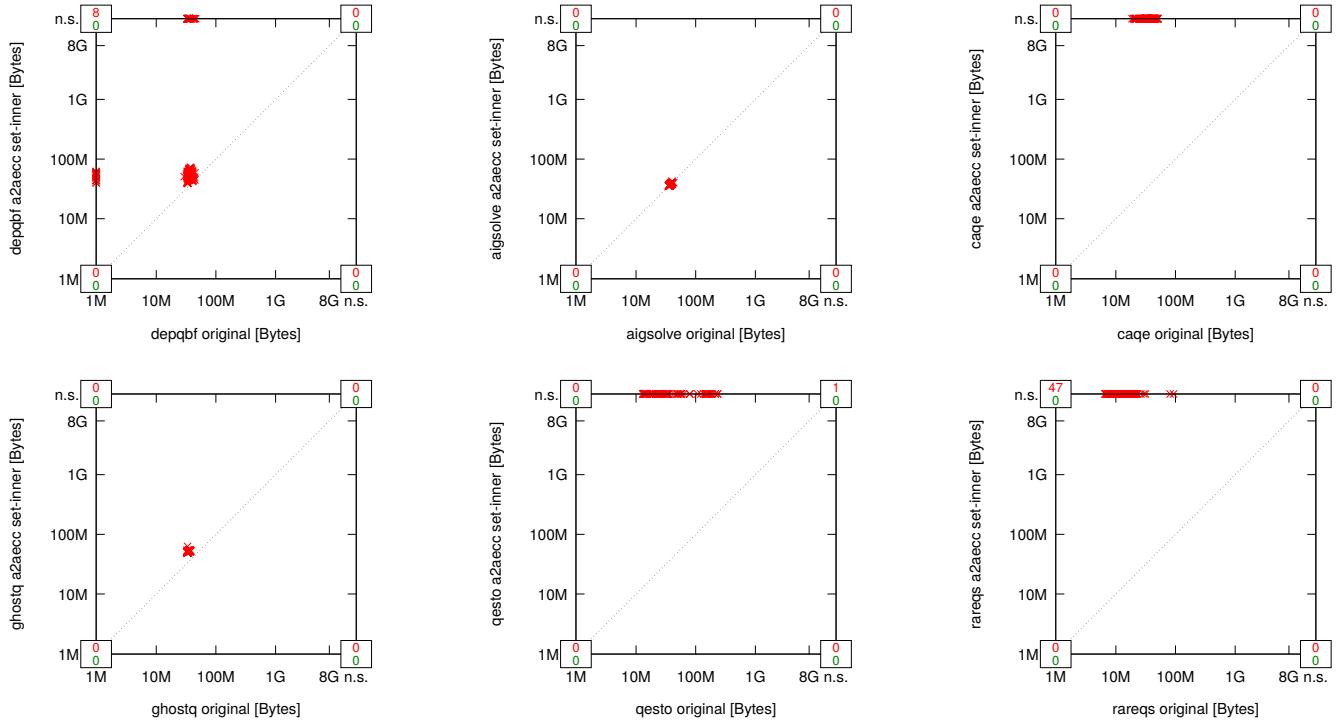


Fig. 1684: Suite Seidl ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

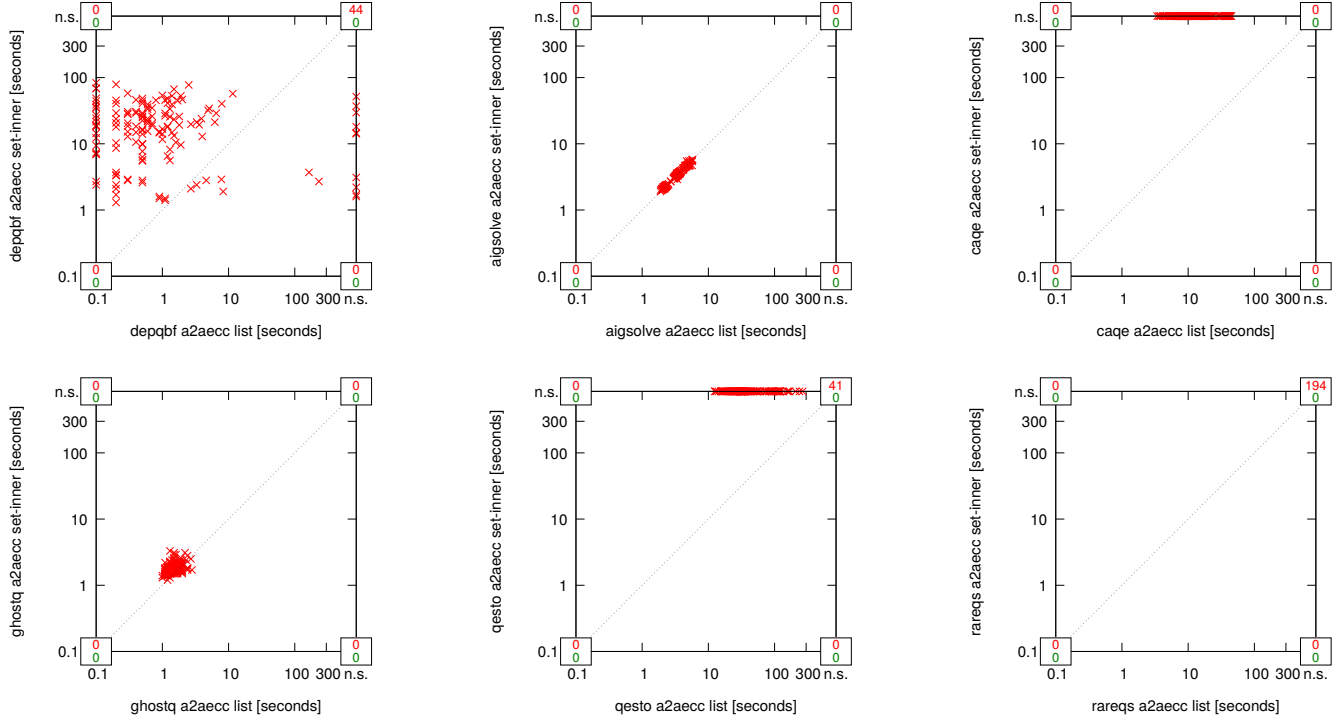


Fig. 1685: Suite Seidl ($n = 194$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

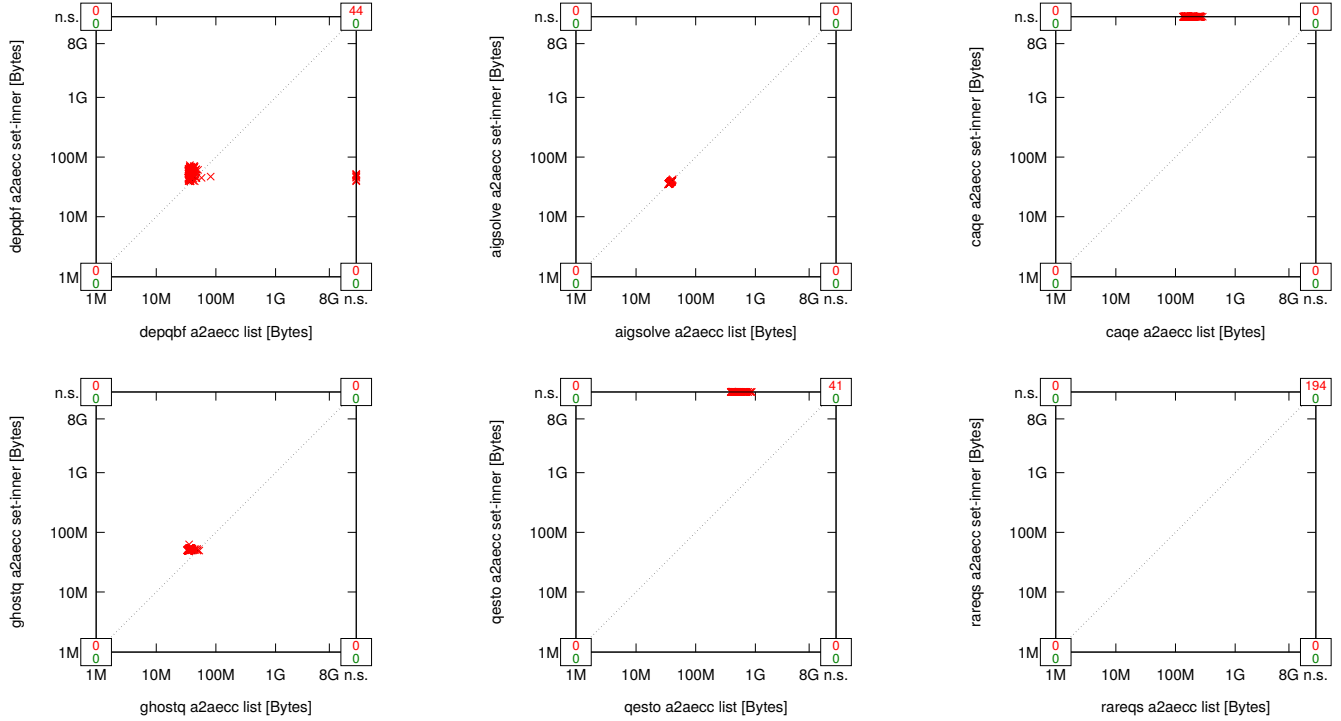


Fig. 1686: Suite Seidl ($n = 194$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

44) Tacchella ($n = 193$):

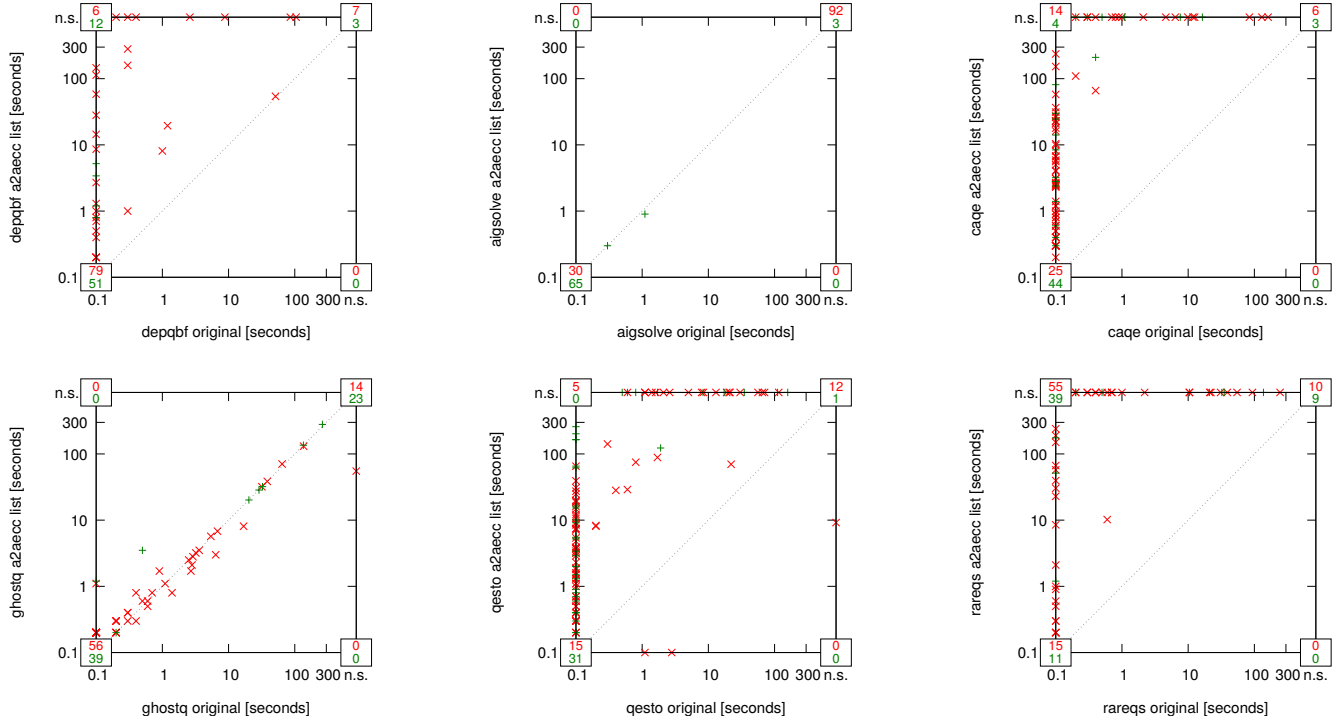


Fig. 1687: Suite Tacchella ($n = 193$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

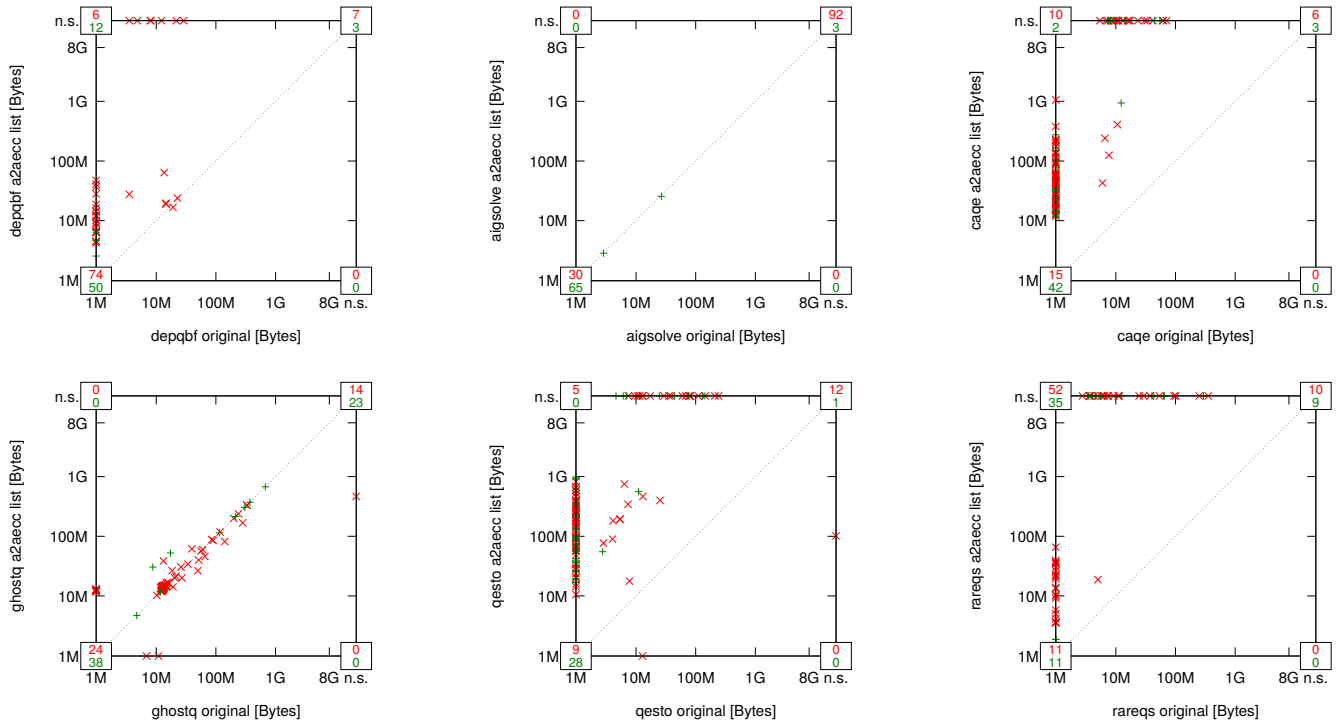


Fig. 1688: Suite Tacchella ($n = 193$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

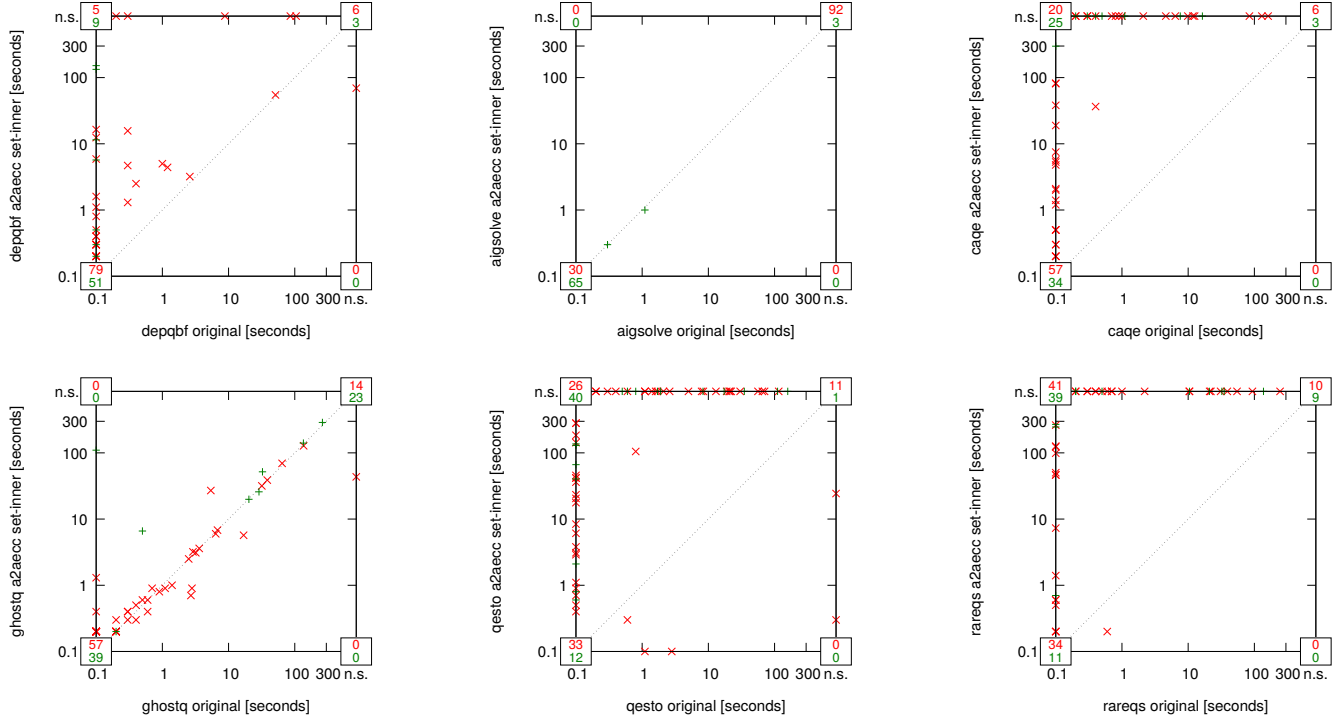


Fig. 1689: Suite Tacchella ($n = 193$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

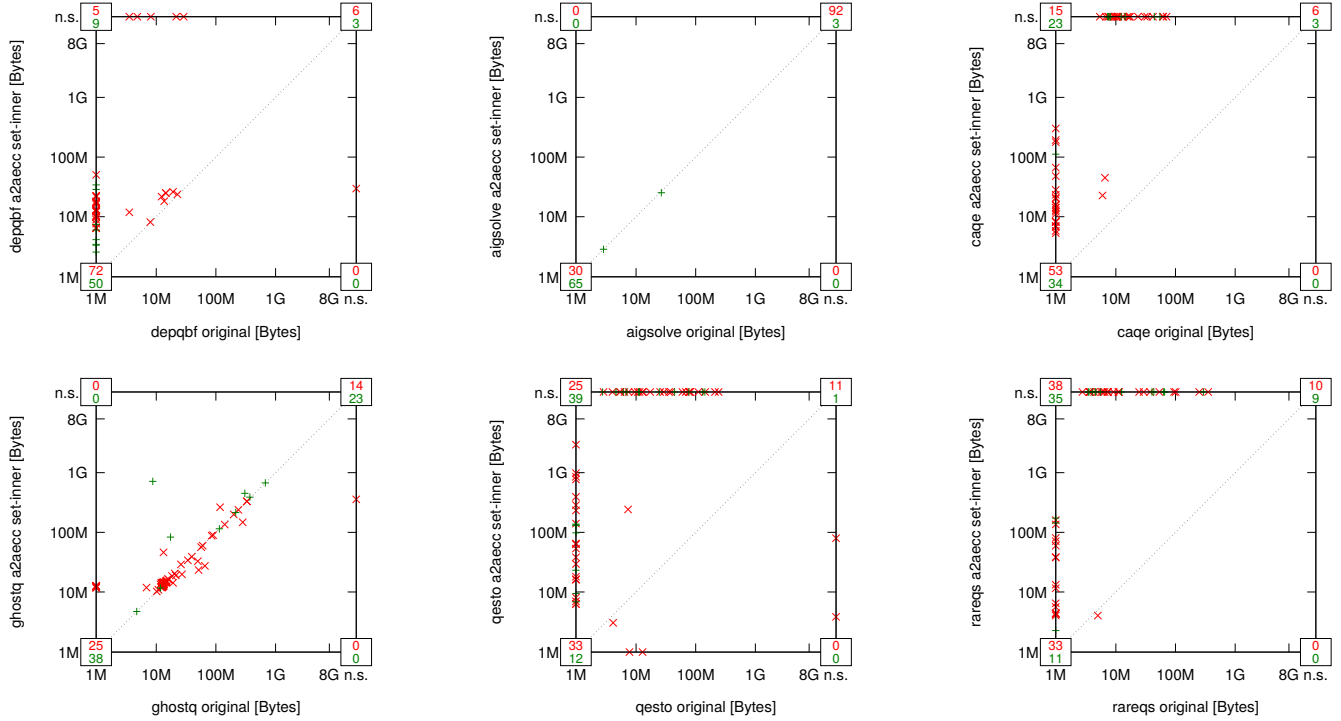


Fig. 1690: Suite Tacchella ($n = 193$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

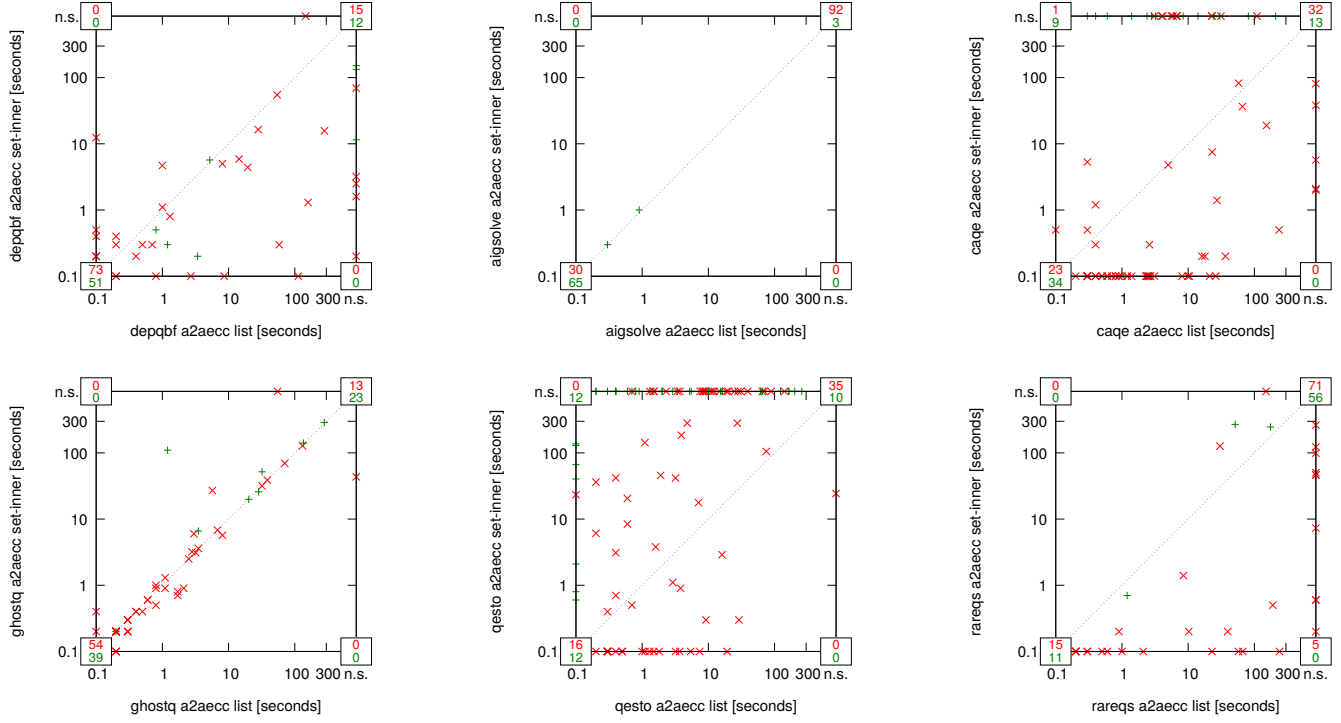


Fig. 1691: Suite Tacchella ($n = 193$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

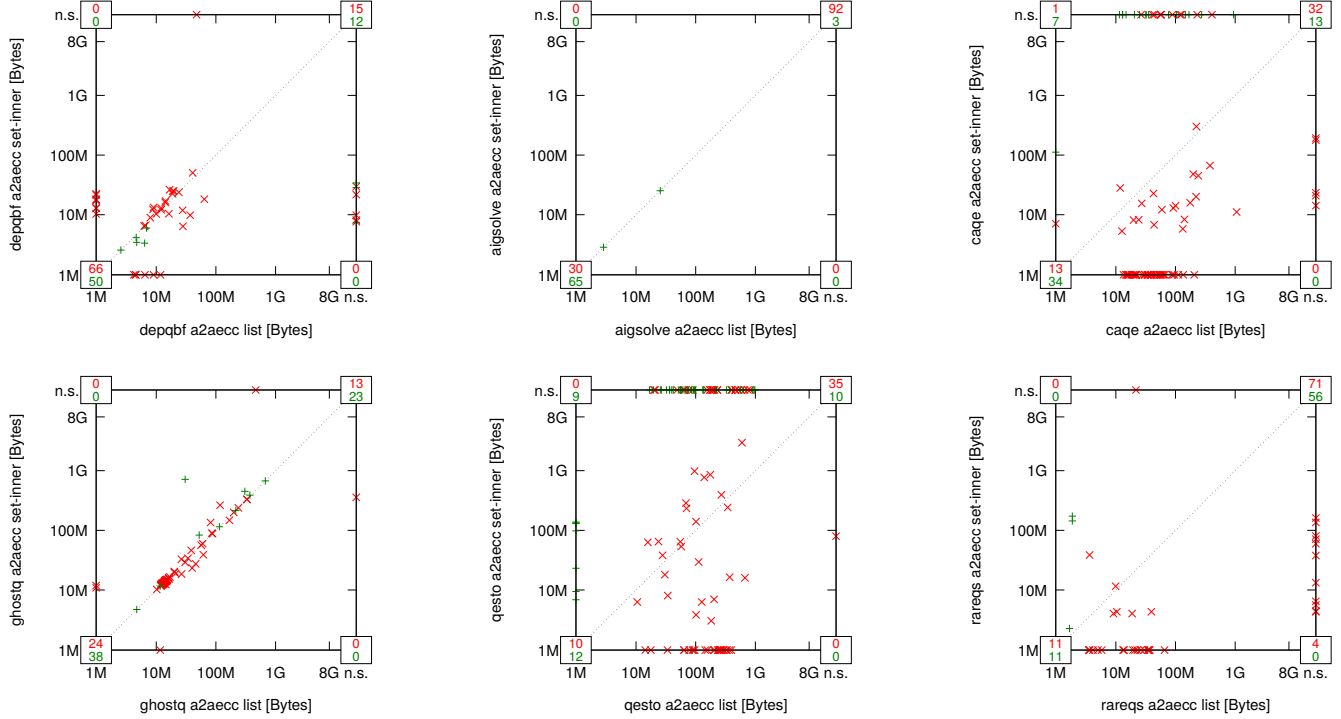


Fig. 1692: Suite Tacchella ($n = 193$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

45) Tentrup ($n = 74$):

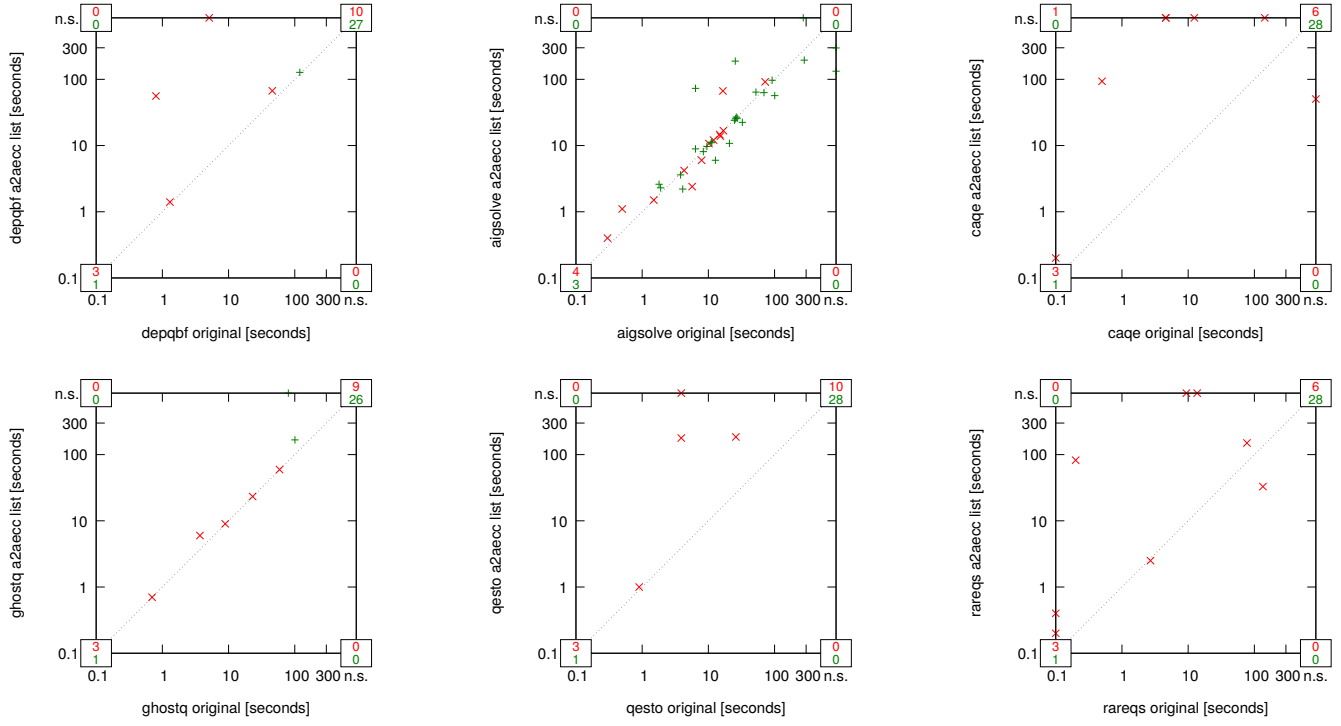


Fig. 1693: Suite Tentrup ($n = 74$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

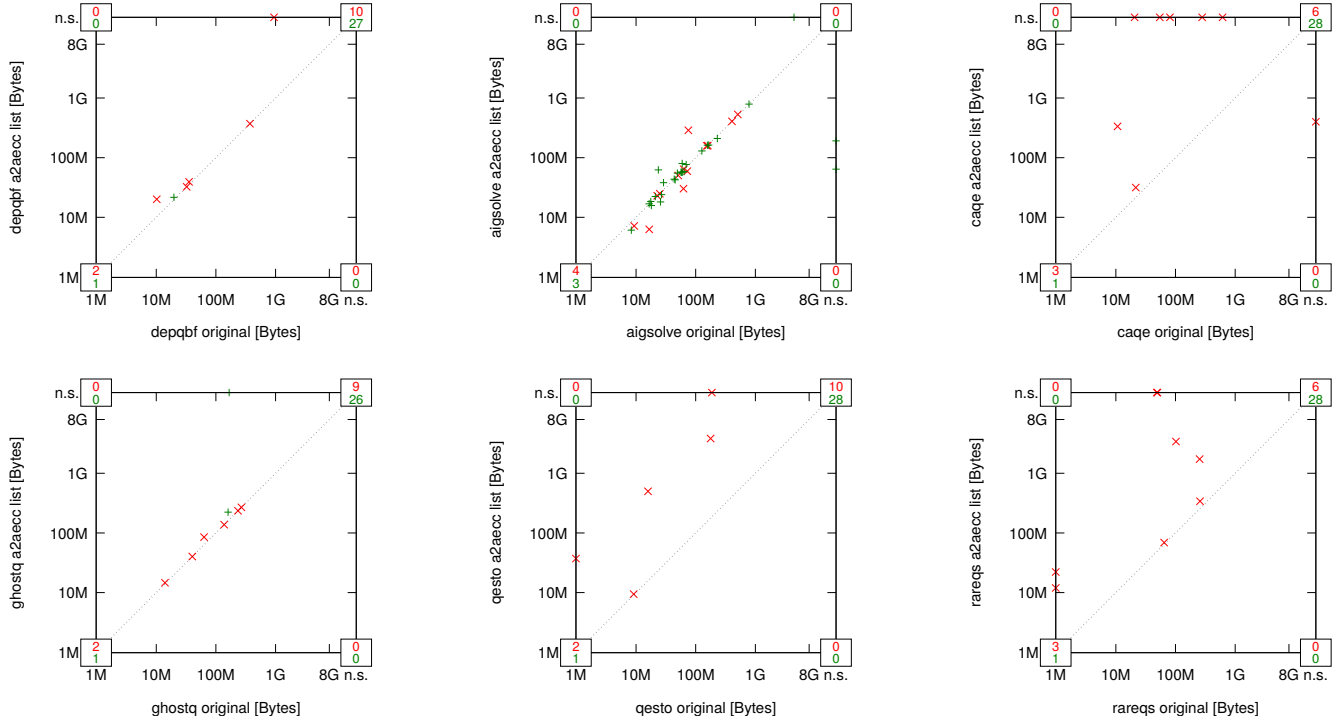


Fig. 1694: Suite Tentrup ($n = 74$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

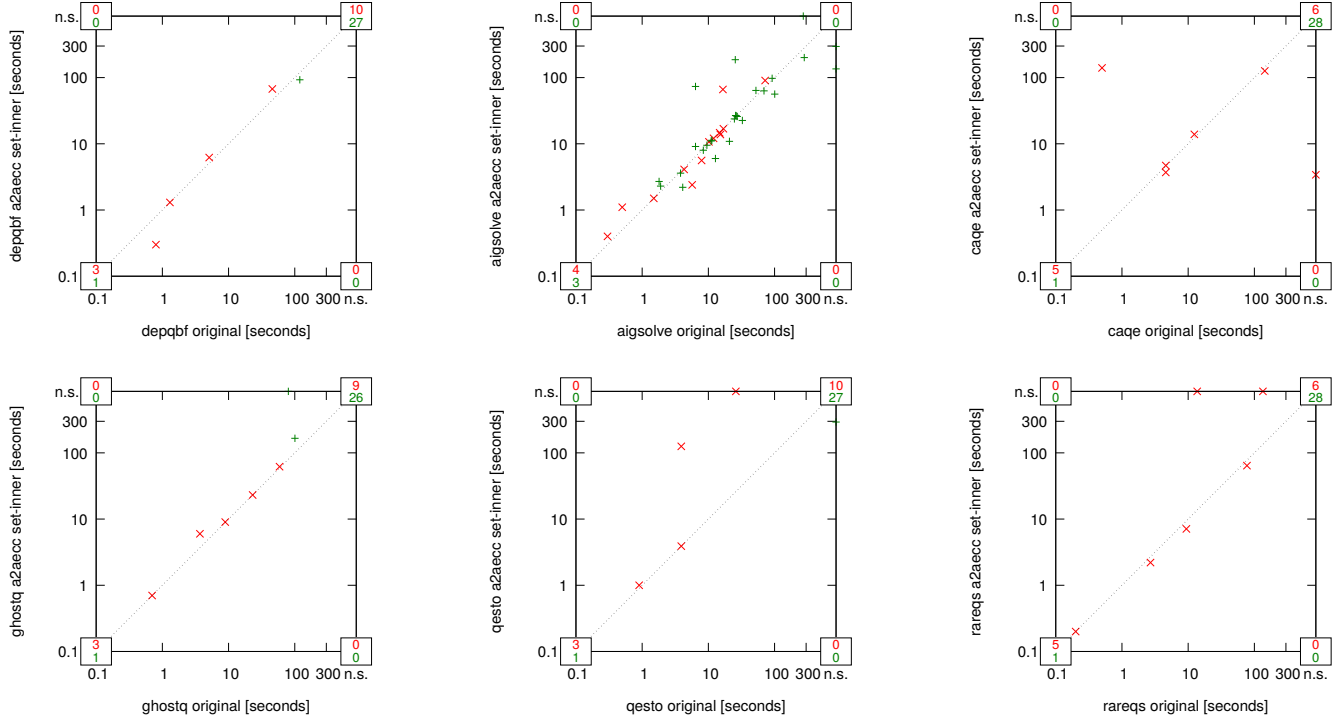


Fig. 1695: Suite Tentrup ($n = 74$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

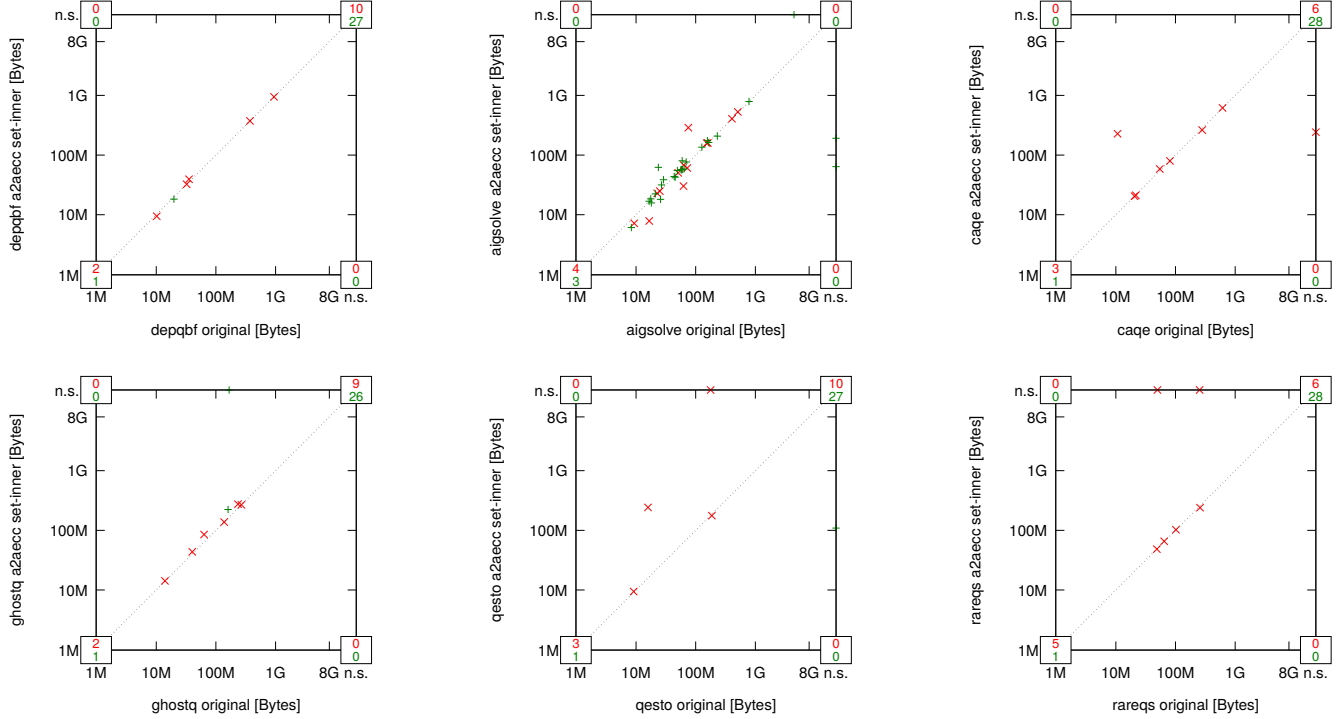


Fig. 1696: Suite Tentrup ($n = 74$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

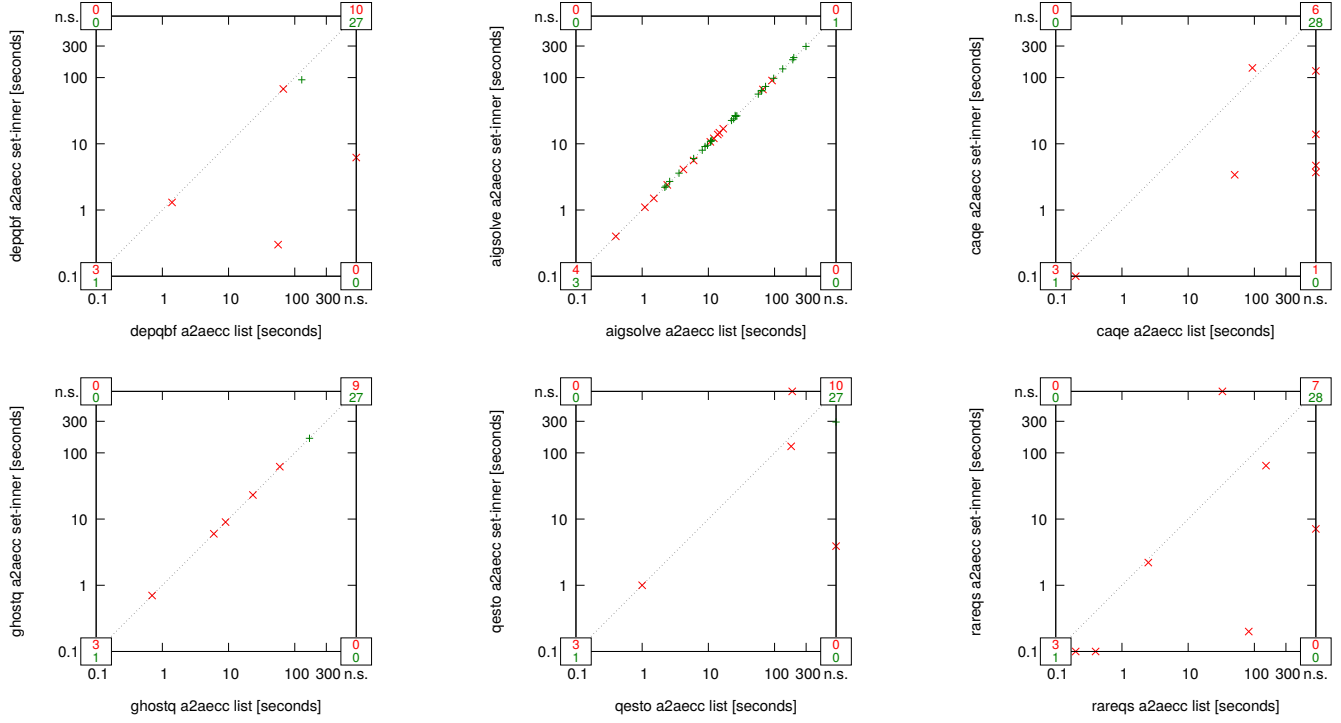


Fig. 1697: Suite Tentrup ($n = 74$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

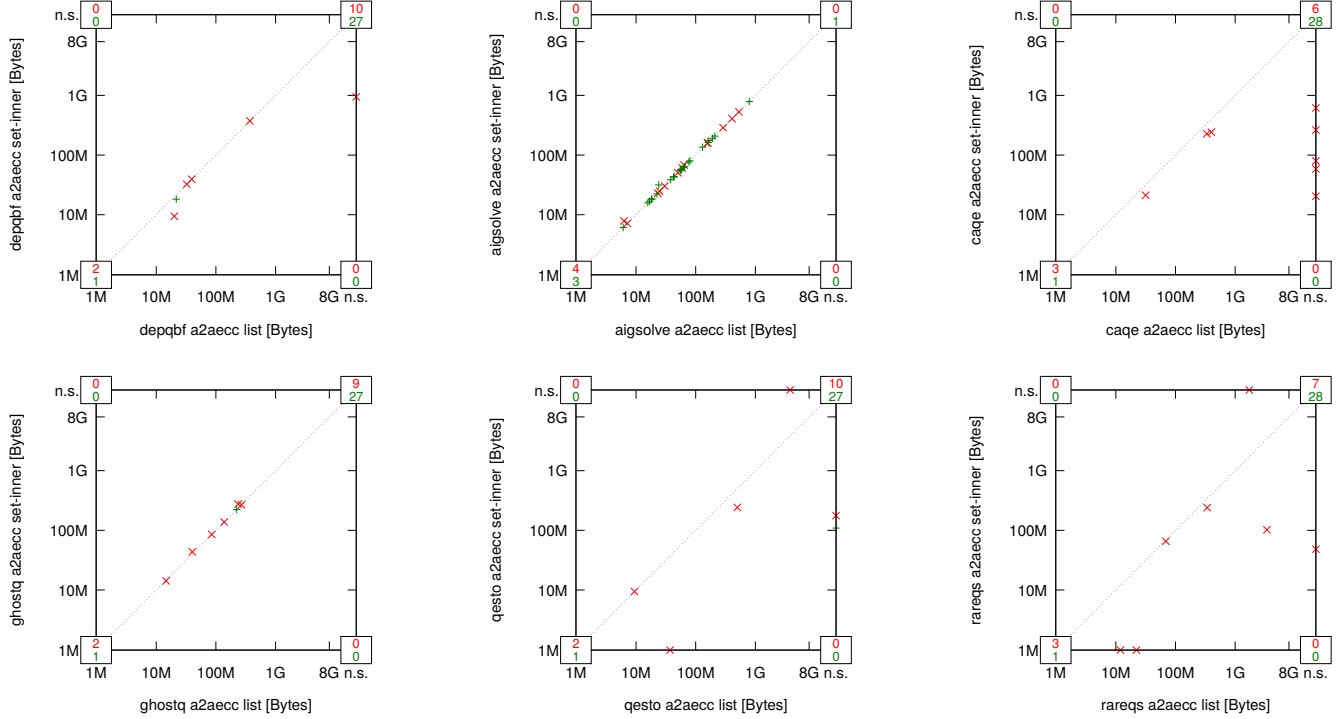


Fig. 1698: Suite Tentrup ($n = 74$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

46) Wintersteiger ($n = 194$):

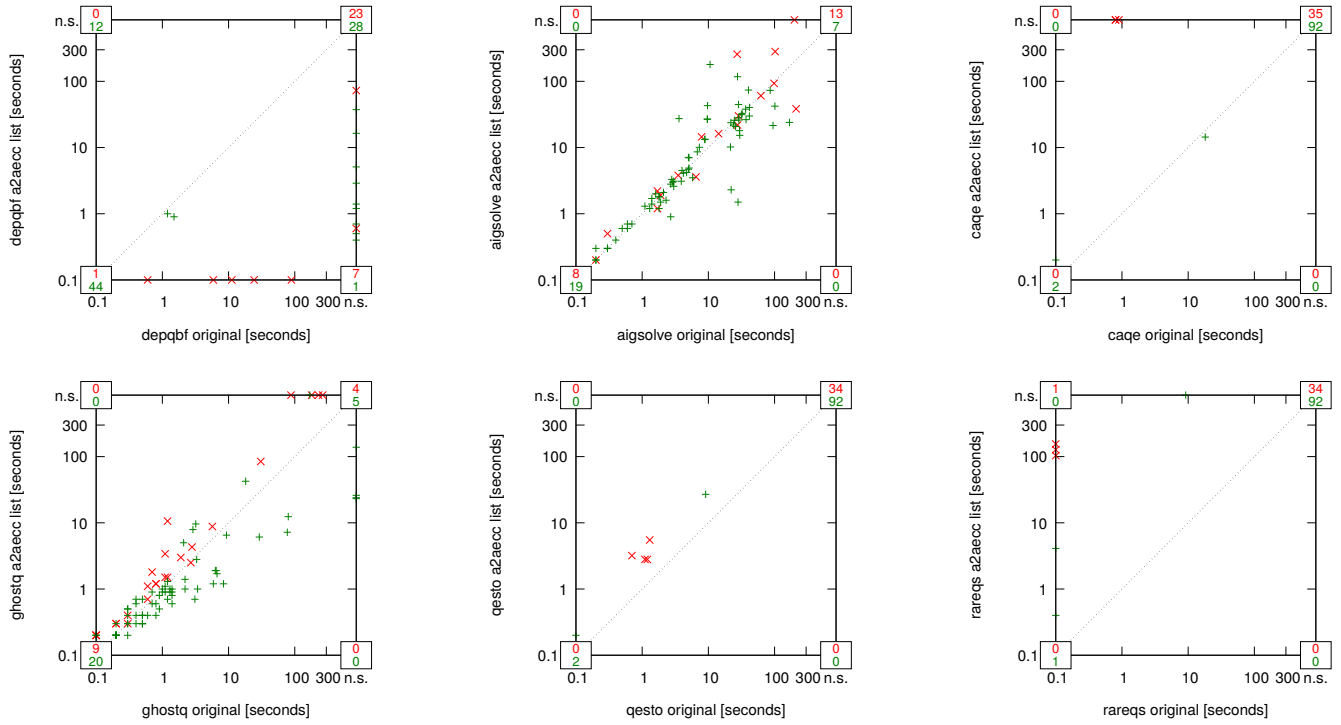


Fig. 1699: Suite Wintersteiger ($n = 194$): Comparing run times for solving the transformed versus the original instances with list semantics in different QBF solvers (run time in seconds).

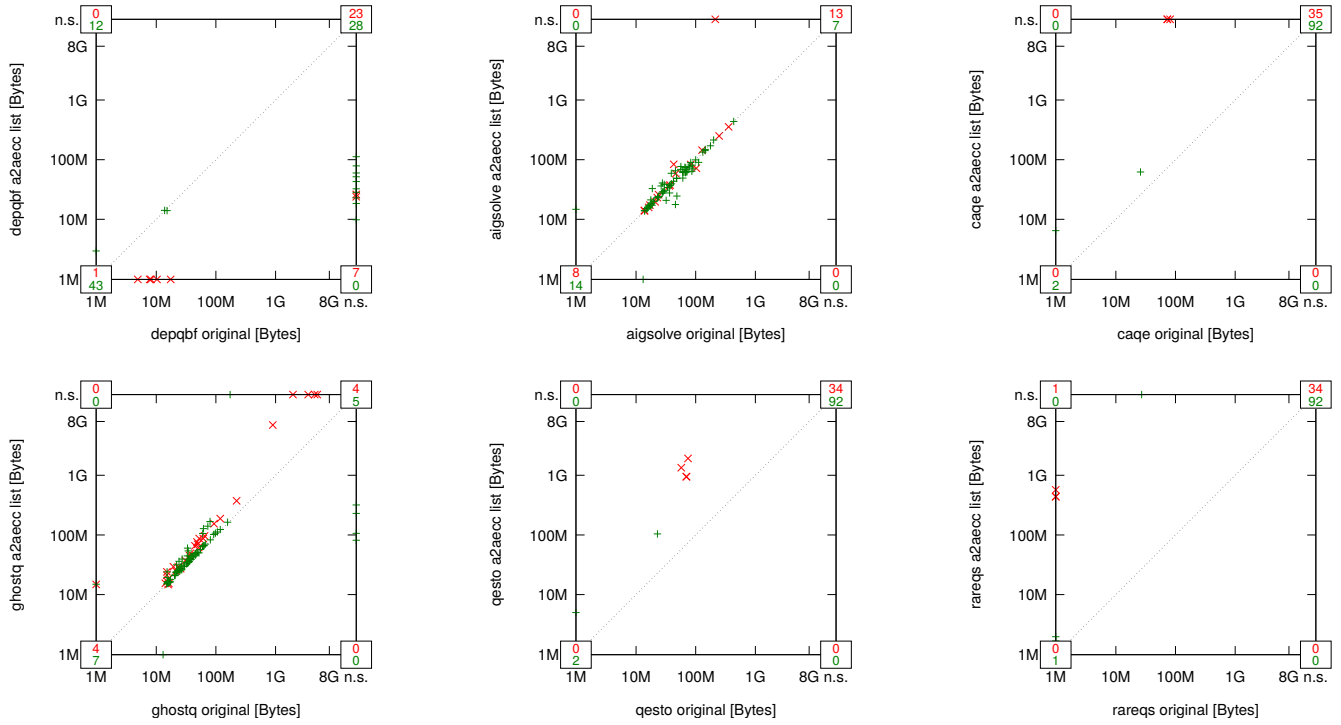


Fig. 1700: Suite Wintersteiger ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with list semantics in different QBF solvers (memory usage in Bytes).

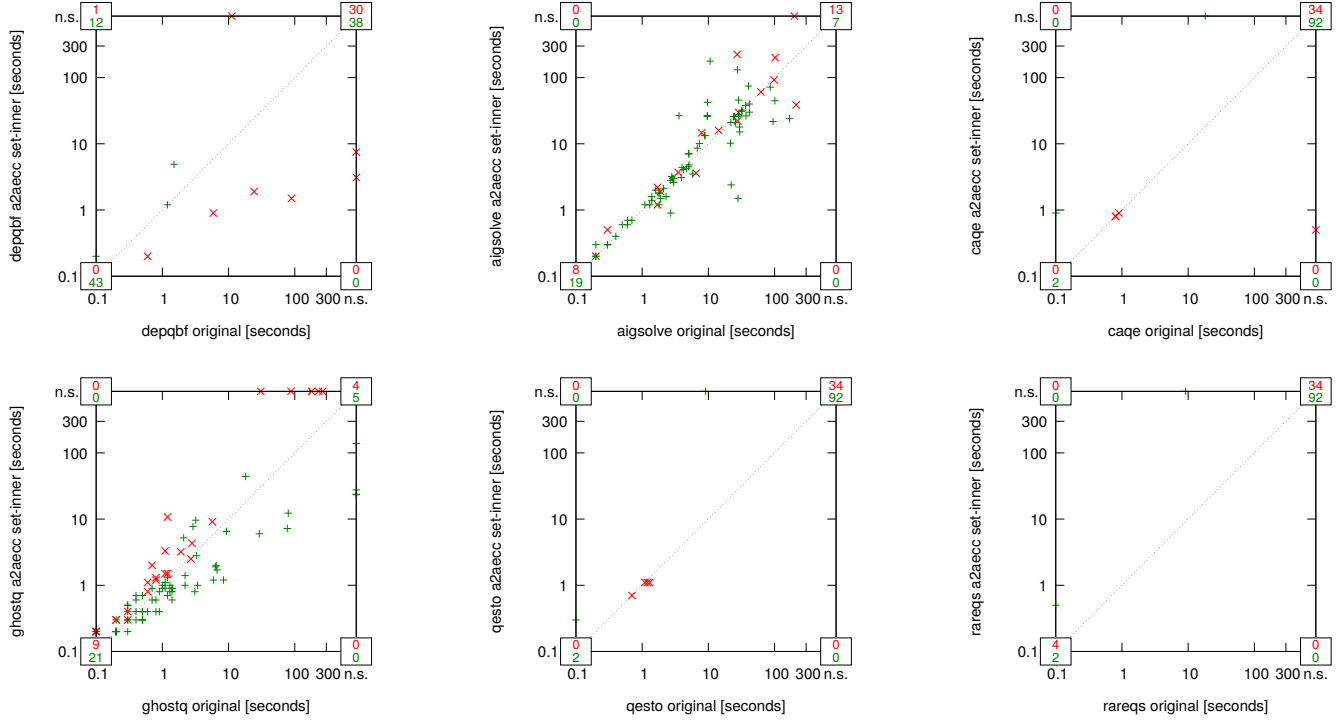


Fig. 1701: Suite Wintersteiger ($n = 194$): Comparing run times for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (run time in seconds).

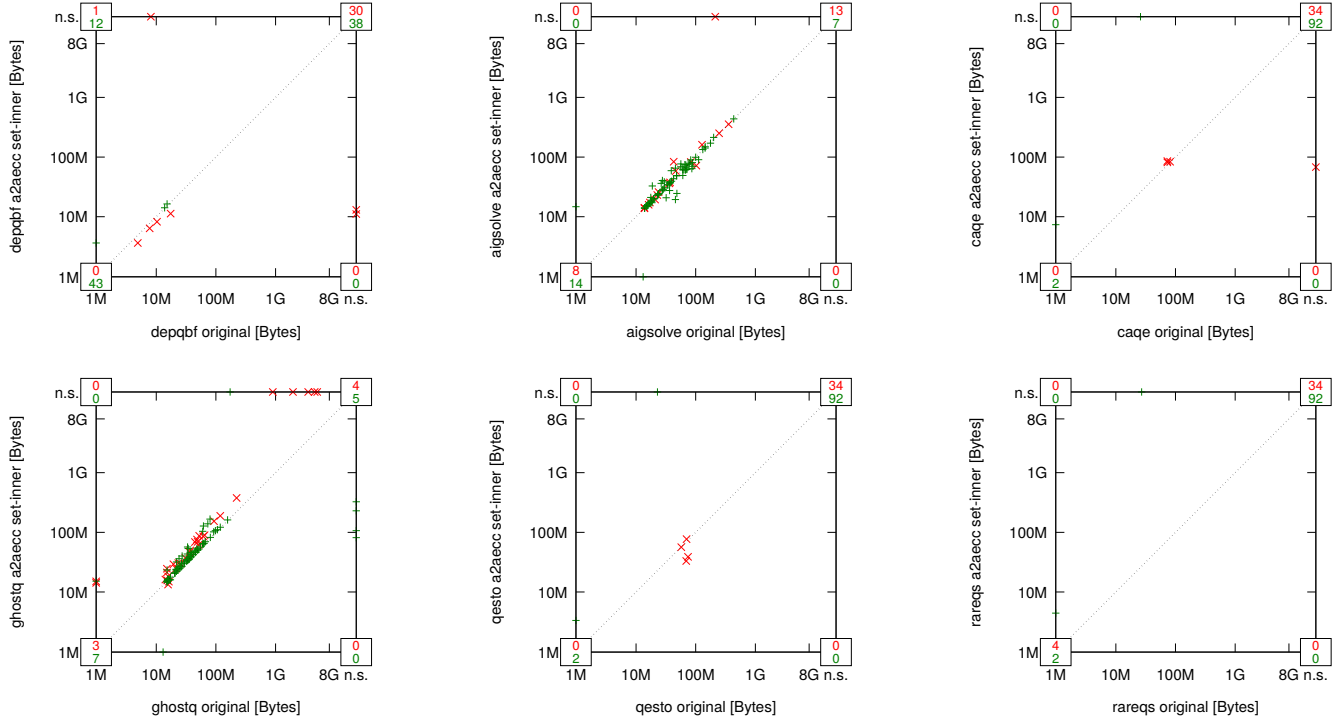


Fig. 1702: Suite Wintersteiger ($n = 194$): Comparing memory usage for solving the transformed versus the original instances with set-inner semantics in different QBF solvers (memory usage in Bytes).

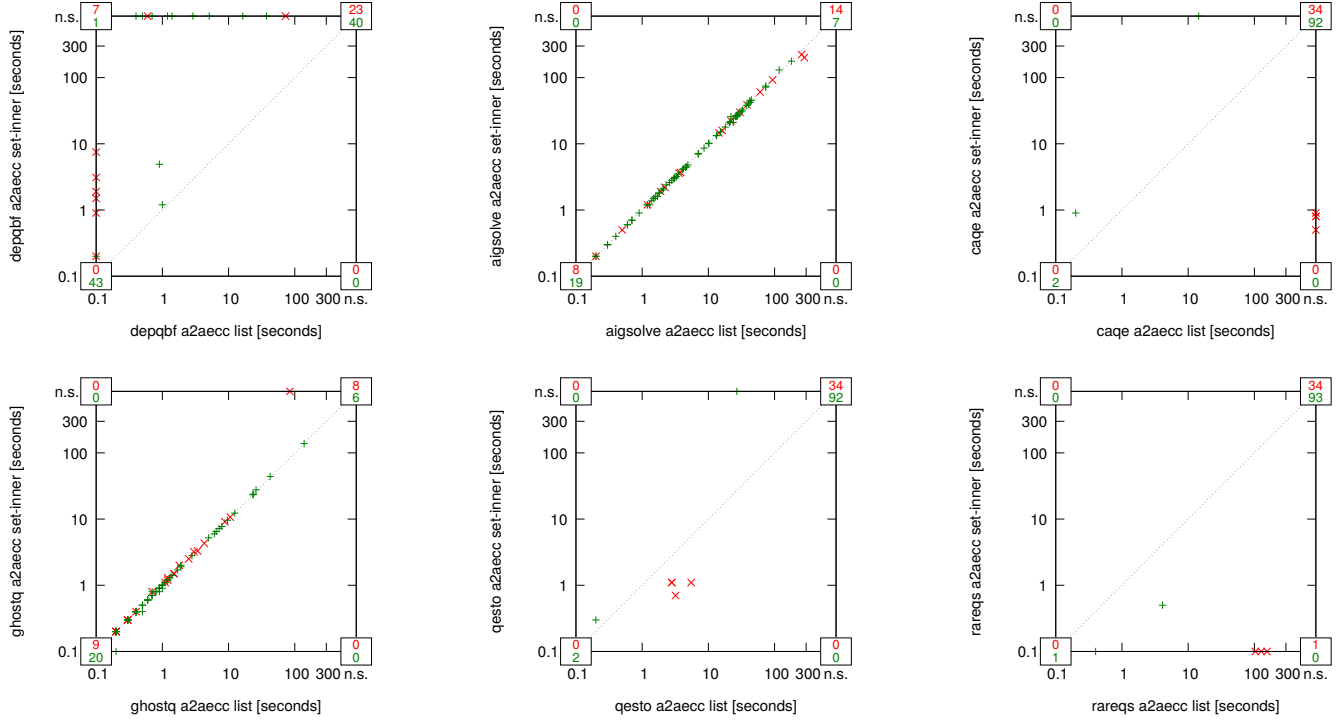


Fig. 1703: Suite Wintersteiger ($n = 194$): Comparing run times for solving the transformed instances in set-inner versus list semantics in different QBF solvers (run time in seconds).

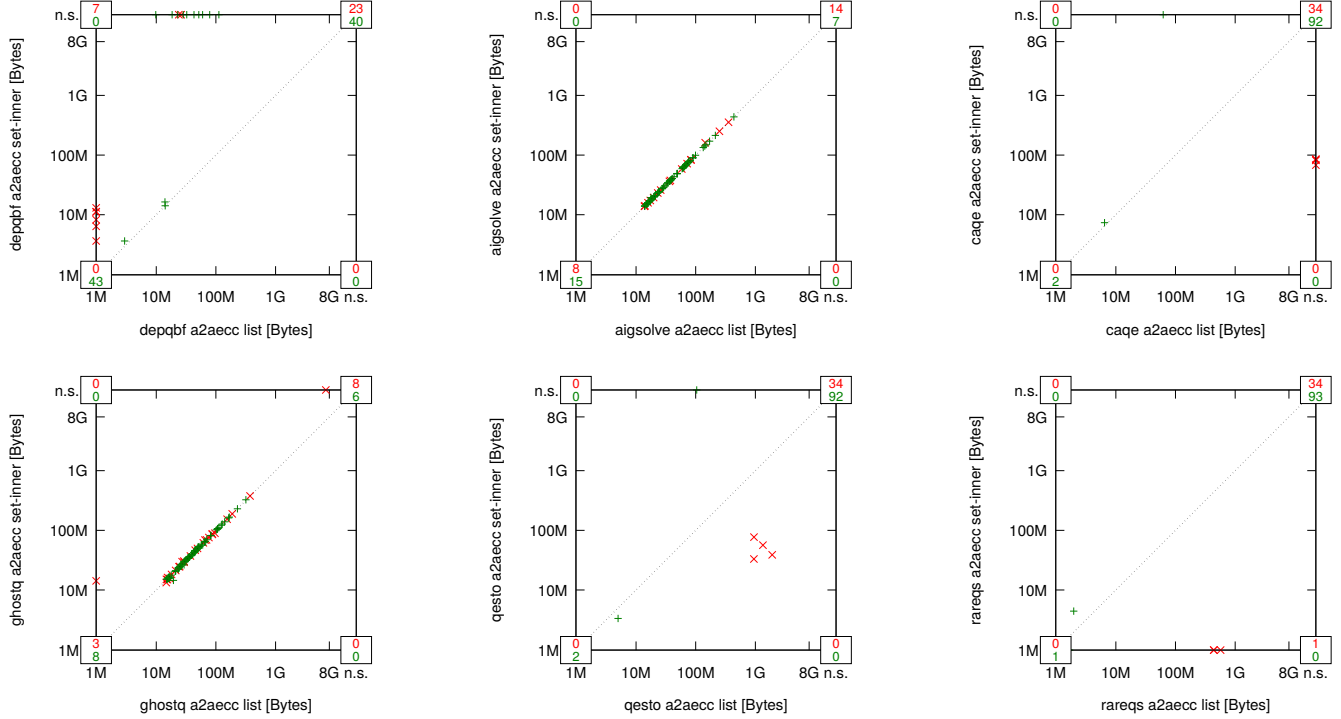


Fig. 1704: Suite Wintersteiger ($n = 194$): Comparing memory usage for solving the transformed instances in set-inner versus list semantics in different QBF solvers (memory usage in Bytes).

B. Partitioned by Number of \forall Quantified Variables

In this subsection there are 6 figures for each solver with subfigures for partitions of the benchmarks according to the number of \forall quantified variables:

- 1) Comparing run times for solving the transformed versus the original instances with list semantics.
- 2) Memory usage for 1.
- 3) As 1. but with set-inner semantics.
- 4) Memory usage for 3.
- 5) A direct comparison of solving the transformed instances with set-inner versus list semantics.
- 6) Memory usage for 5.

a) DepQBF:

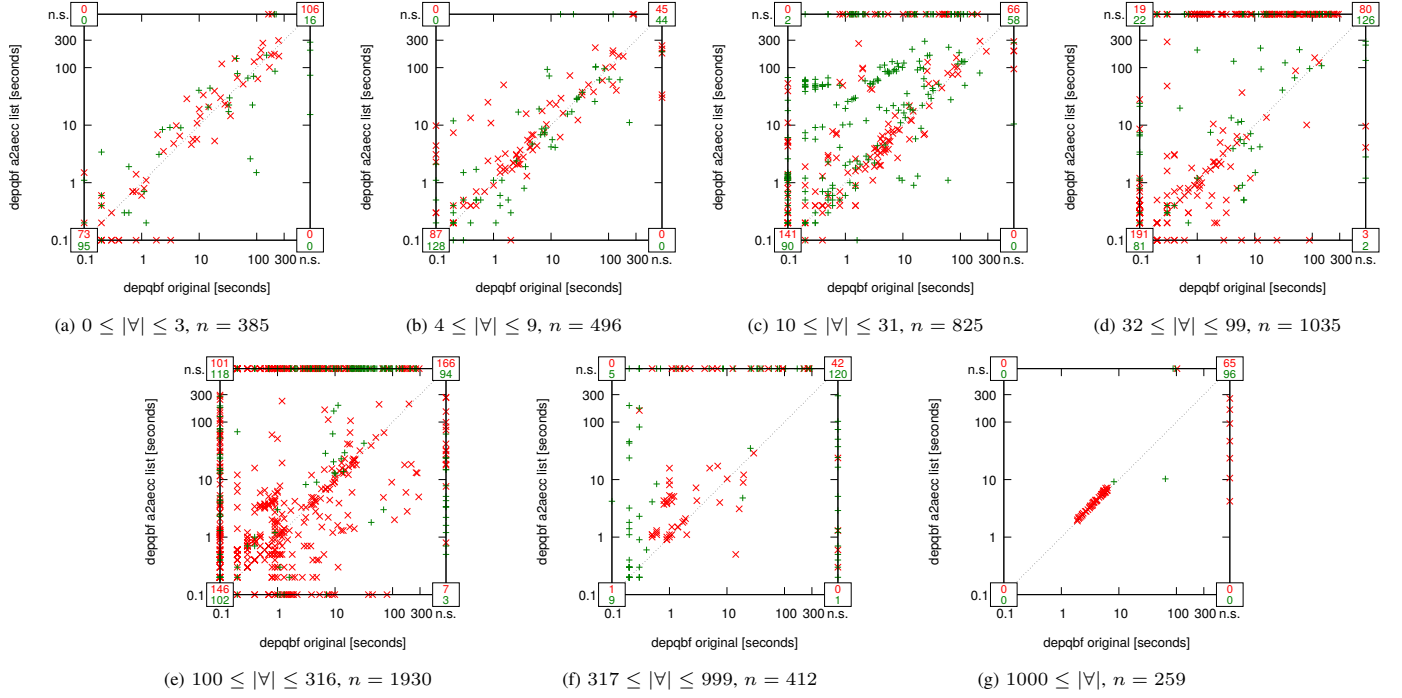


Fig. 1705: Solver DepQBF: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by number of \forall quantified variables (run time in seconds).

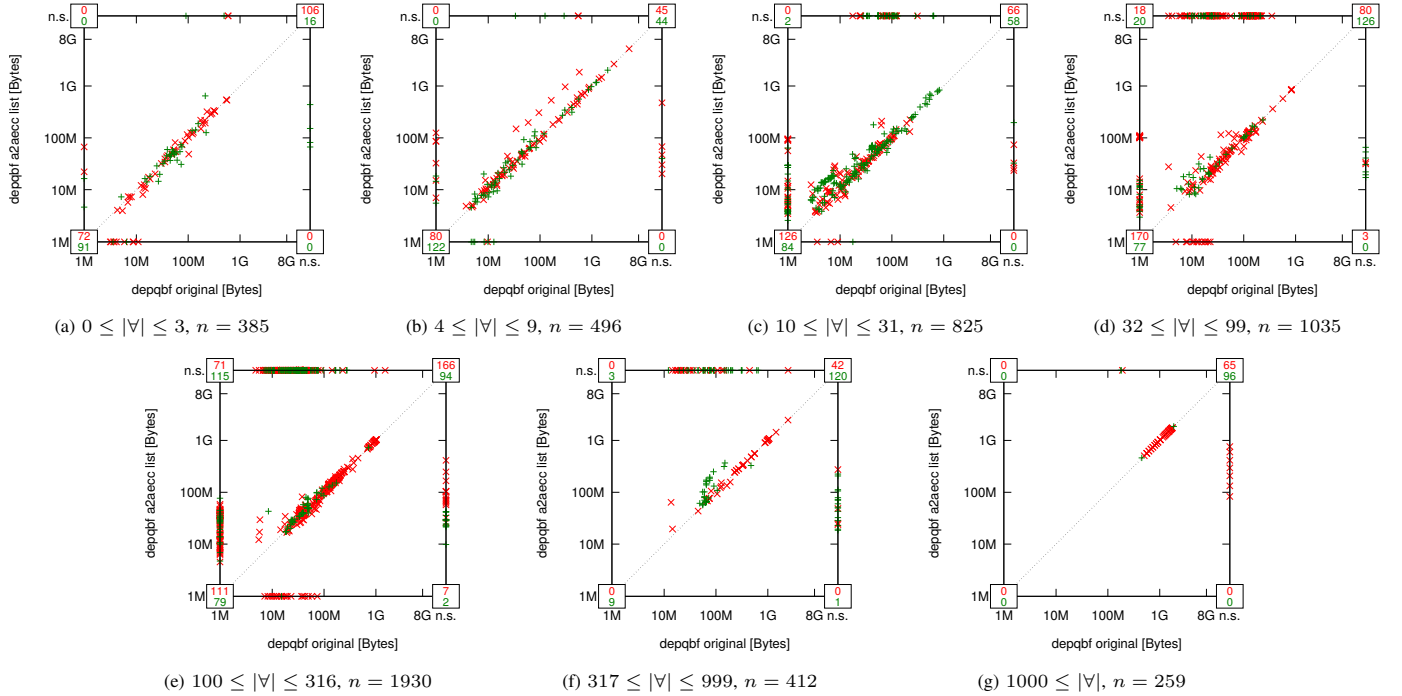


Fig. 1706: Solver DepQBF: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

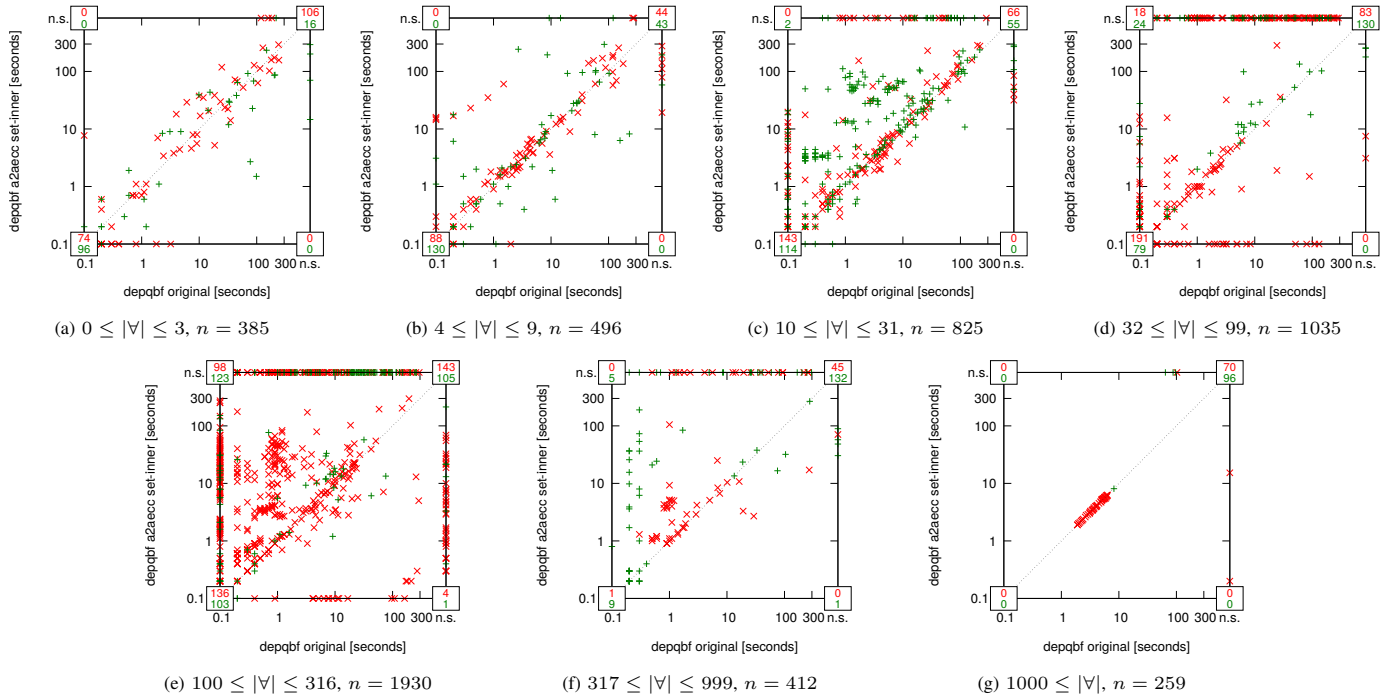


Fig. 1707: Solver DepQBF: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by number of \forall quantified variables (run time in seconds).

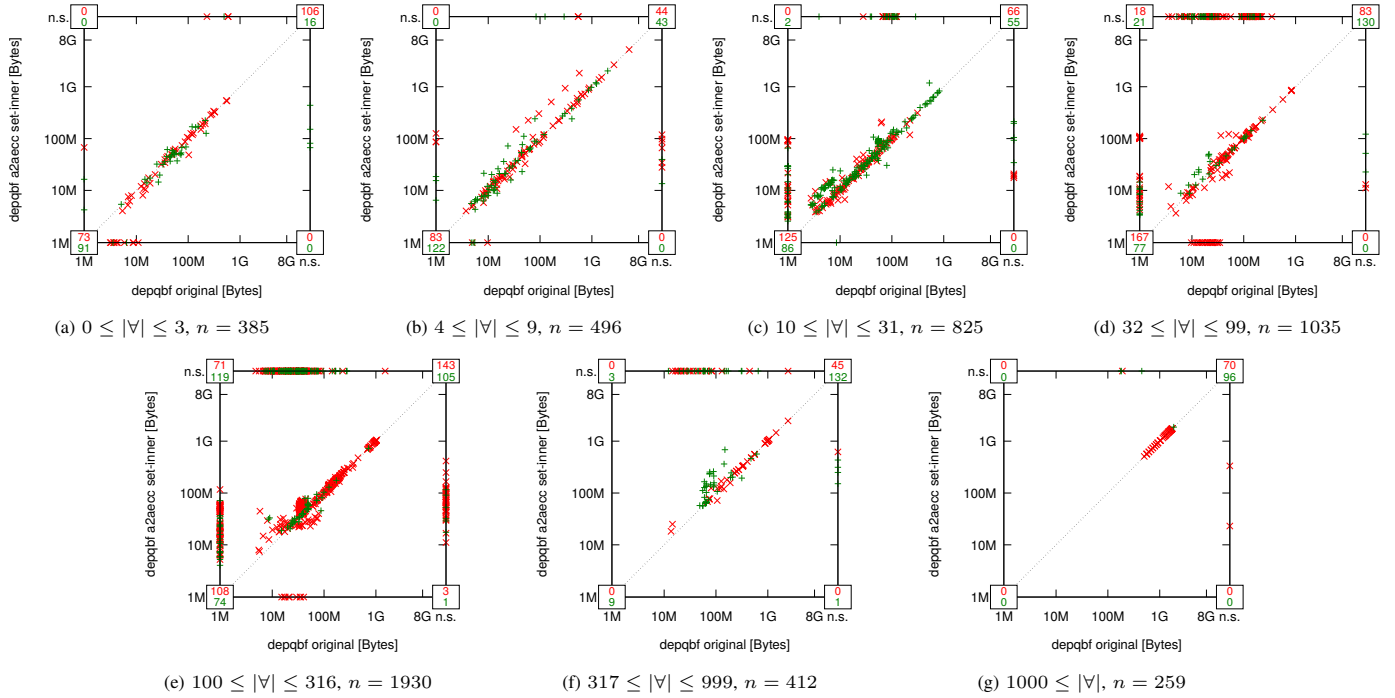


Fig. 1708: Solver DepQBF: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

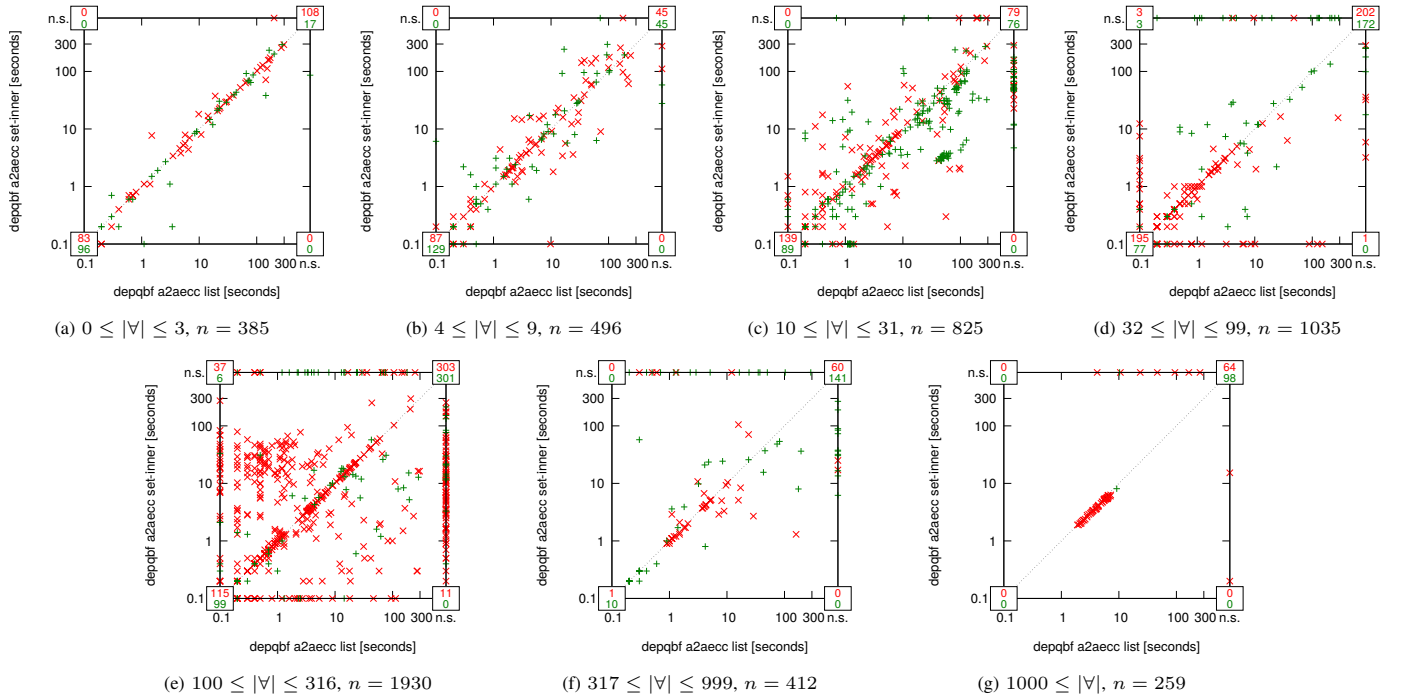


Fig. 1709: Solver DepQBF: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by number of \forall quantified variables (run time in seconds).

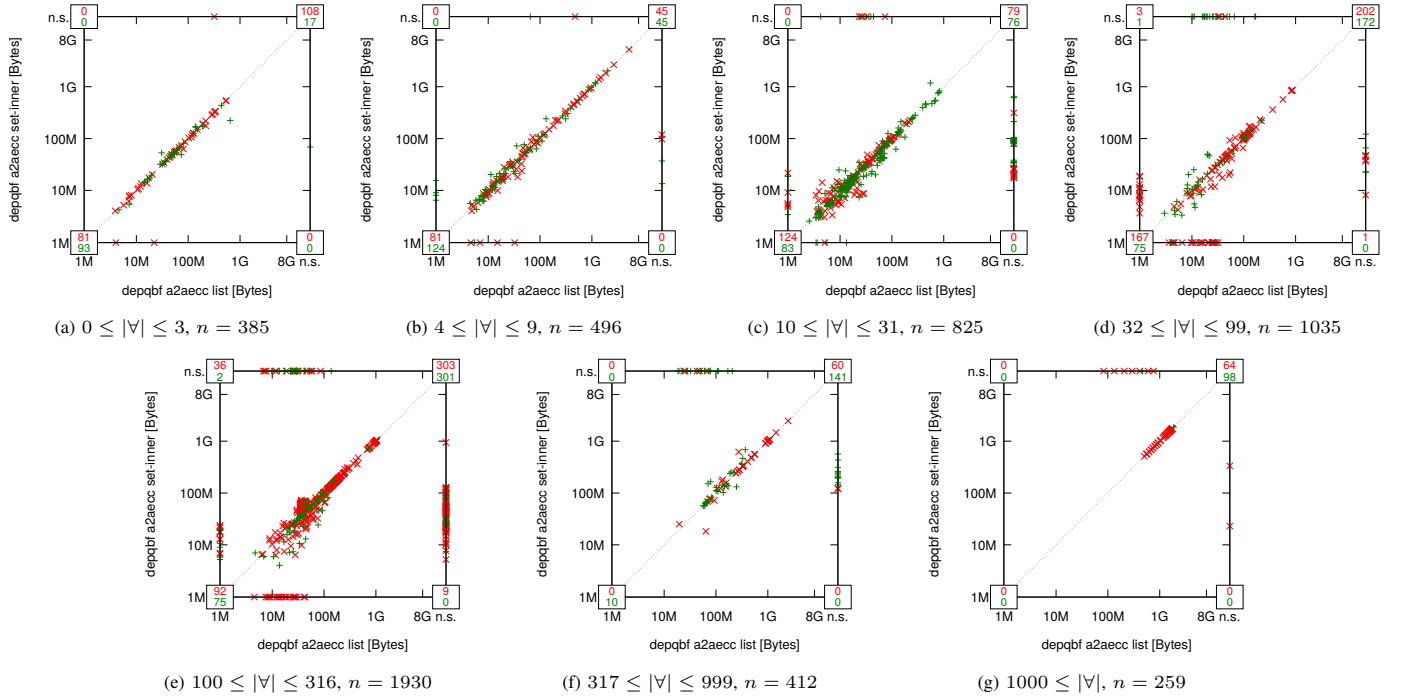


Fig. 1710: Solver DepQBF: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

b) AIGSolve:

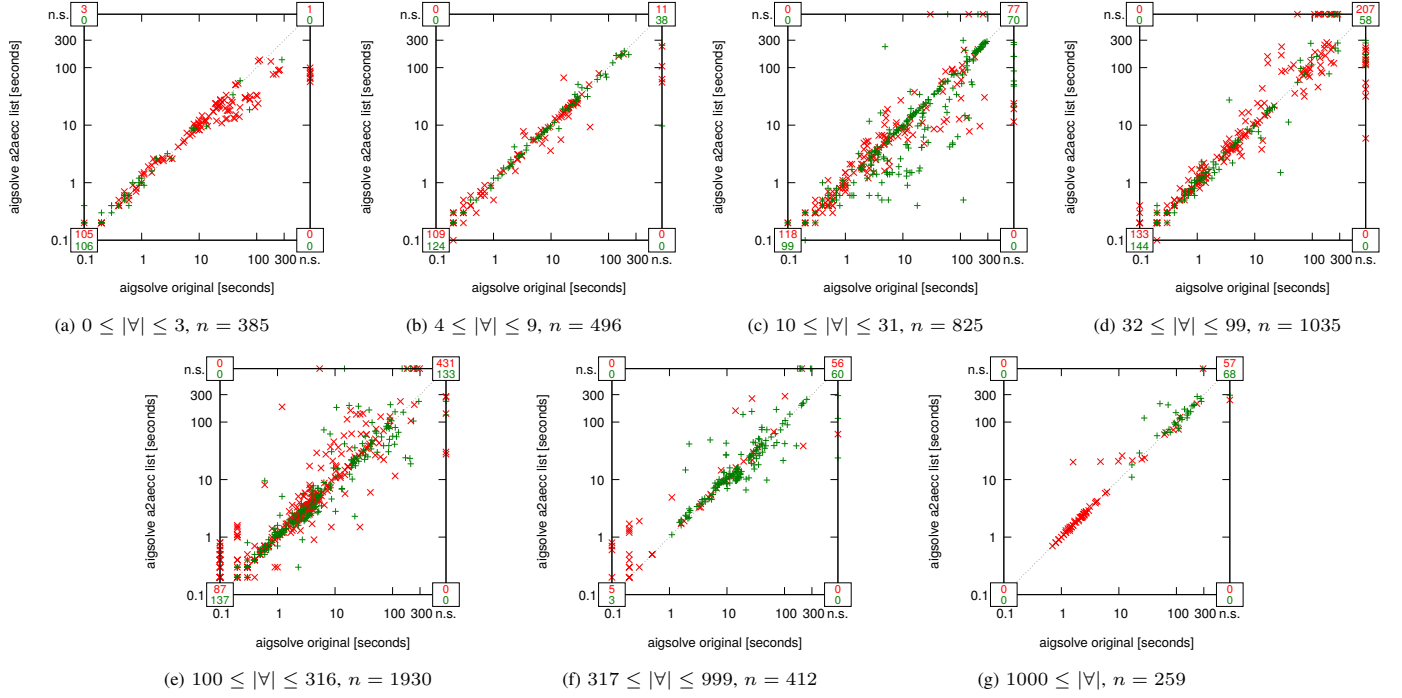


Fig. 1711: Solver AIGSolve: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by number of \forall quantified variables (run time in seconds).

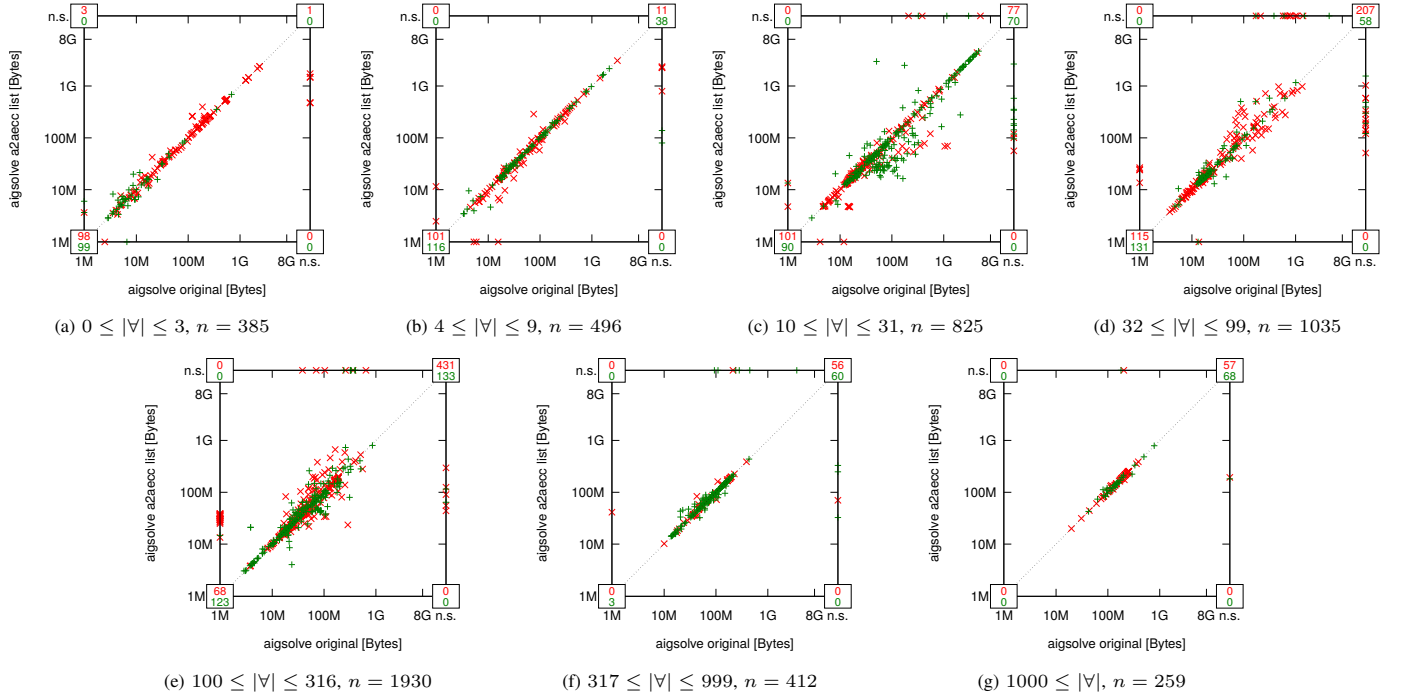


Fig. 1712: Solver AIGSolve: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

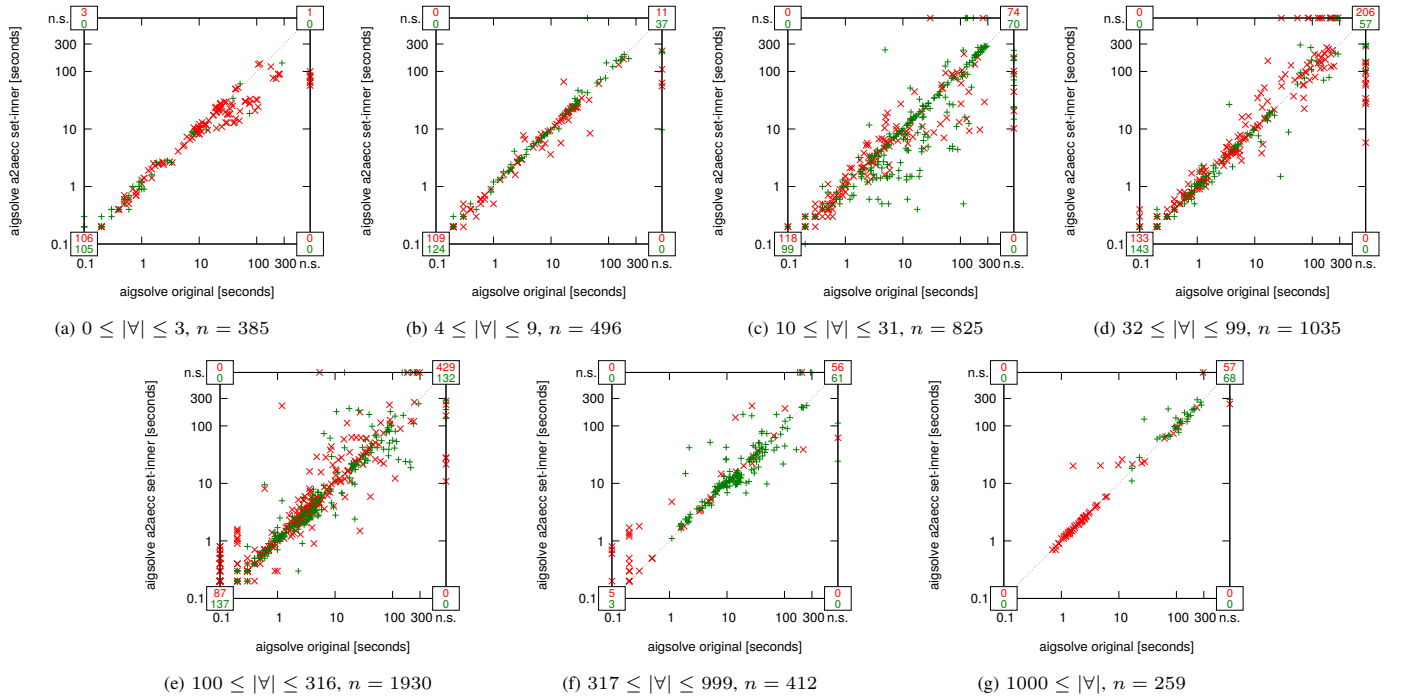


Fig. 1713: Solver AIGSolve: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by number of \forall quantified variables (run time in seconds).

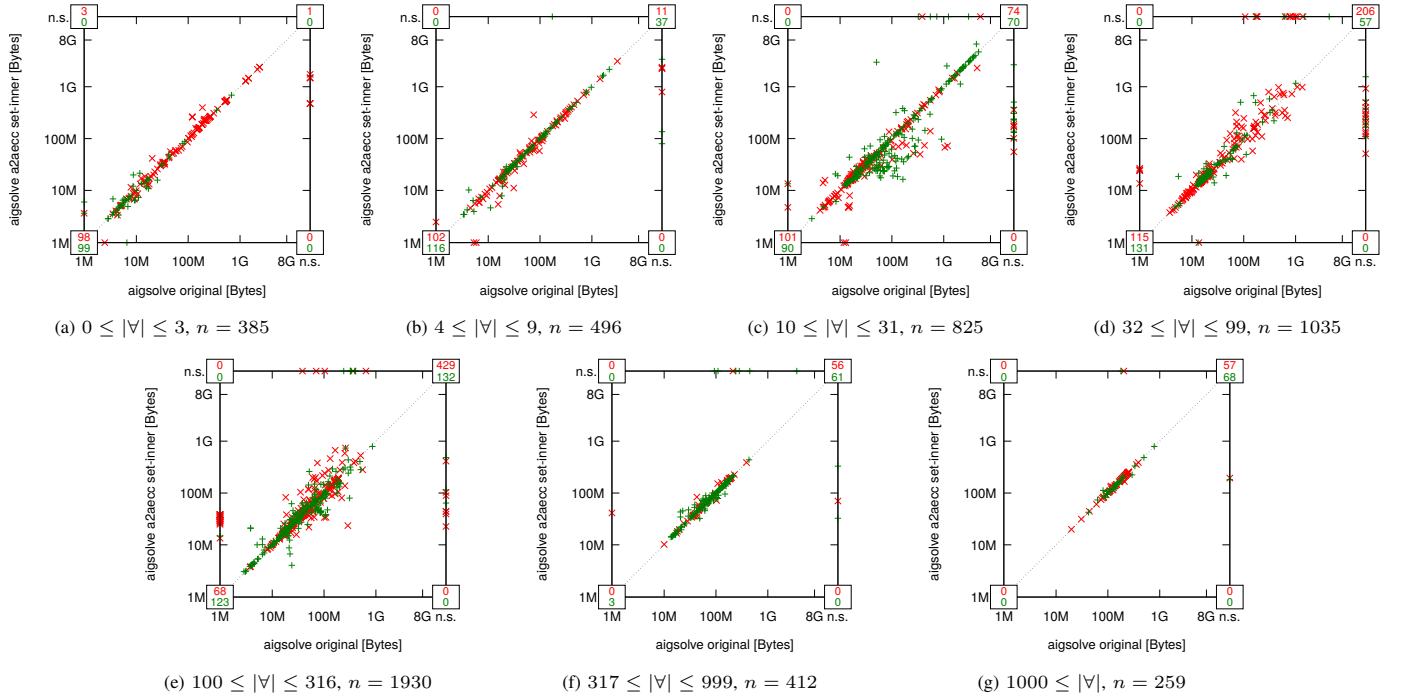


Fig. 1714: Solver AIGSolve: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

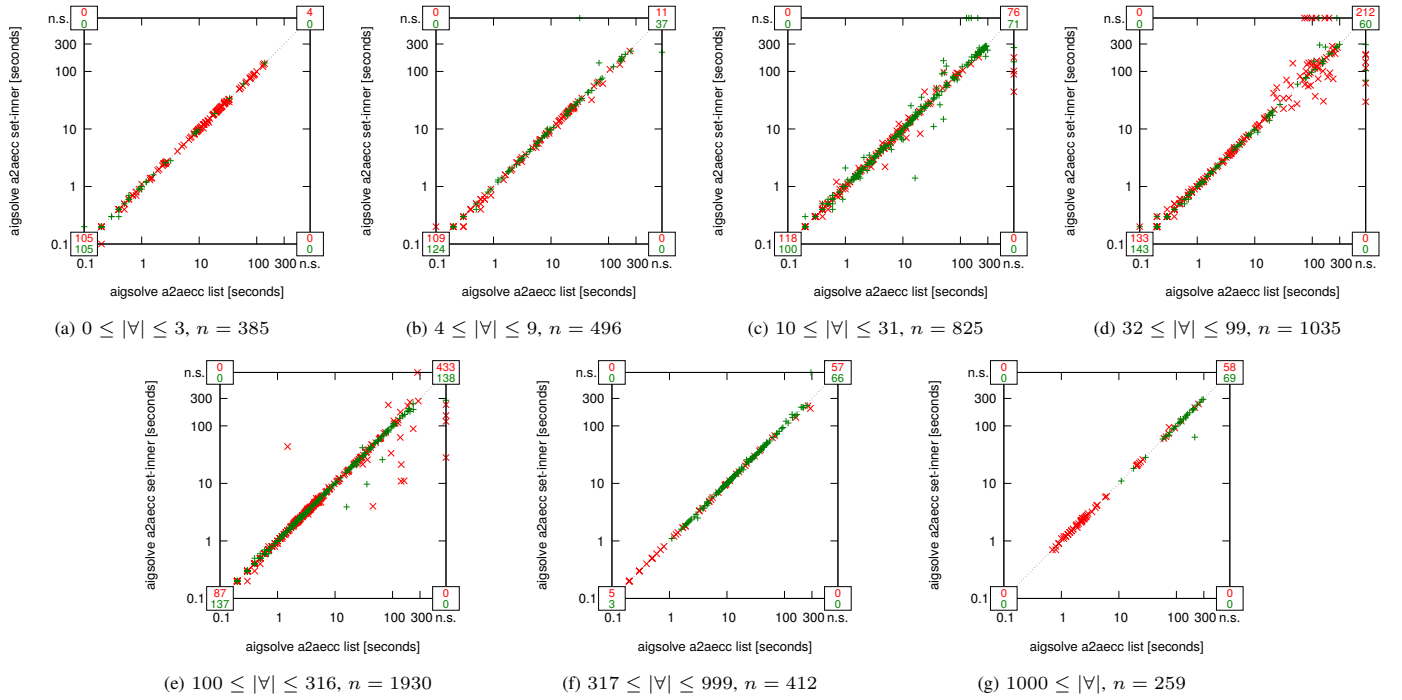


Fig. 1715: Solver AIGsolve: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by number of \forall quantified variables (run time in seconds).

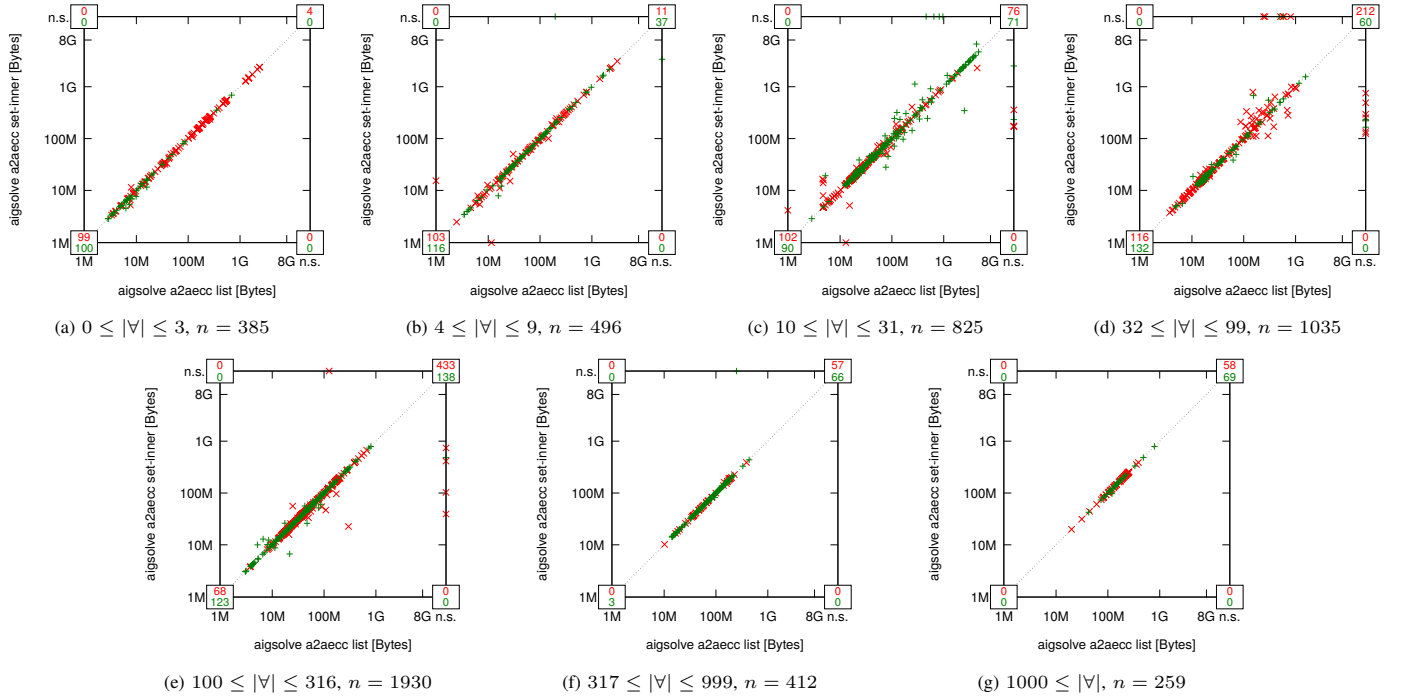


Fig. 1716: Solver AIGsolve: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

c) CAQE:

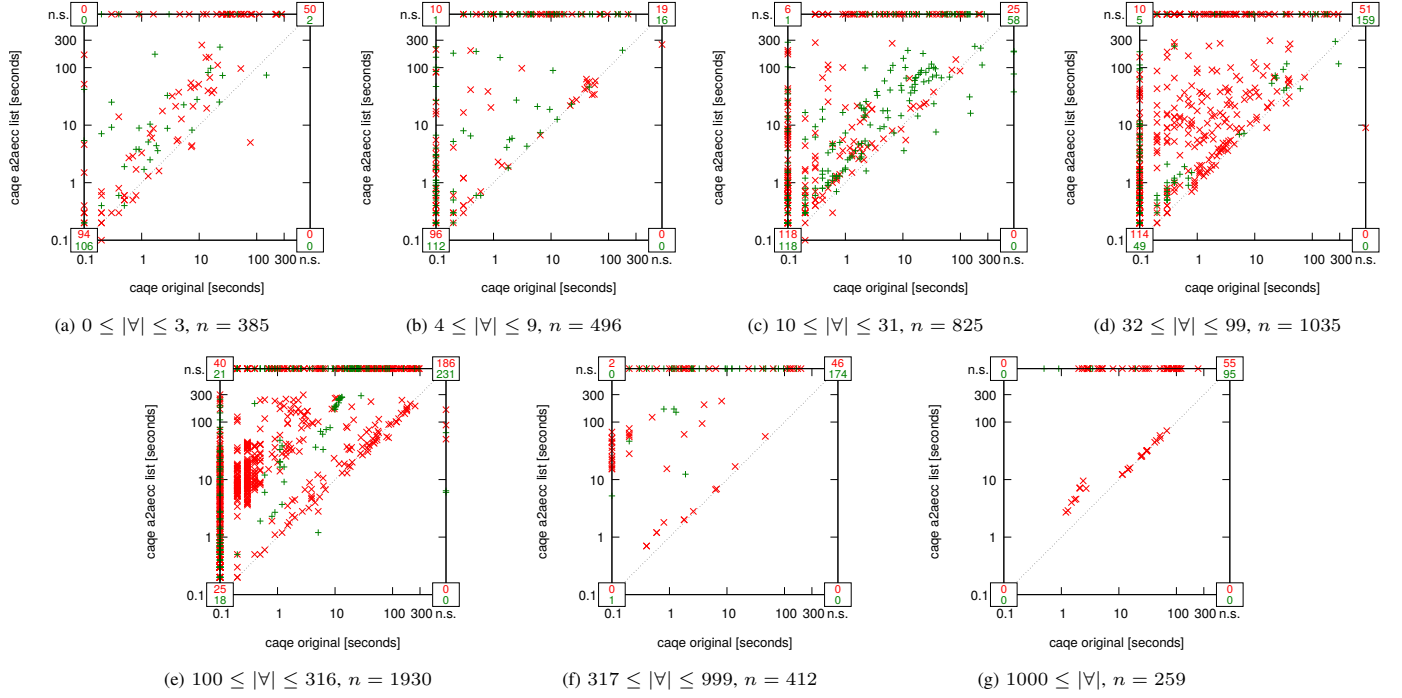


Fig. 1717: Solver CAQE: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by number of \forall quantified variables (run time in seconds).

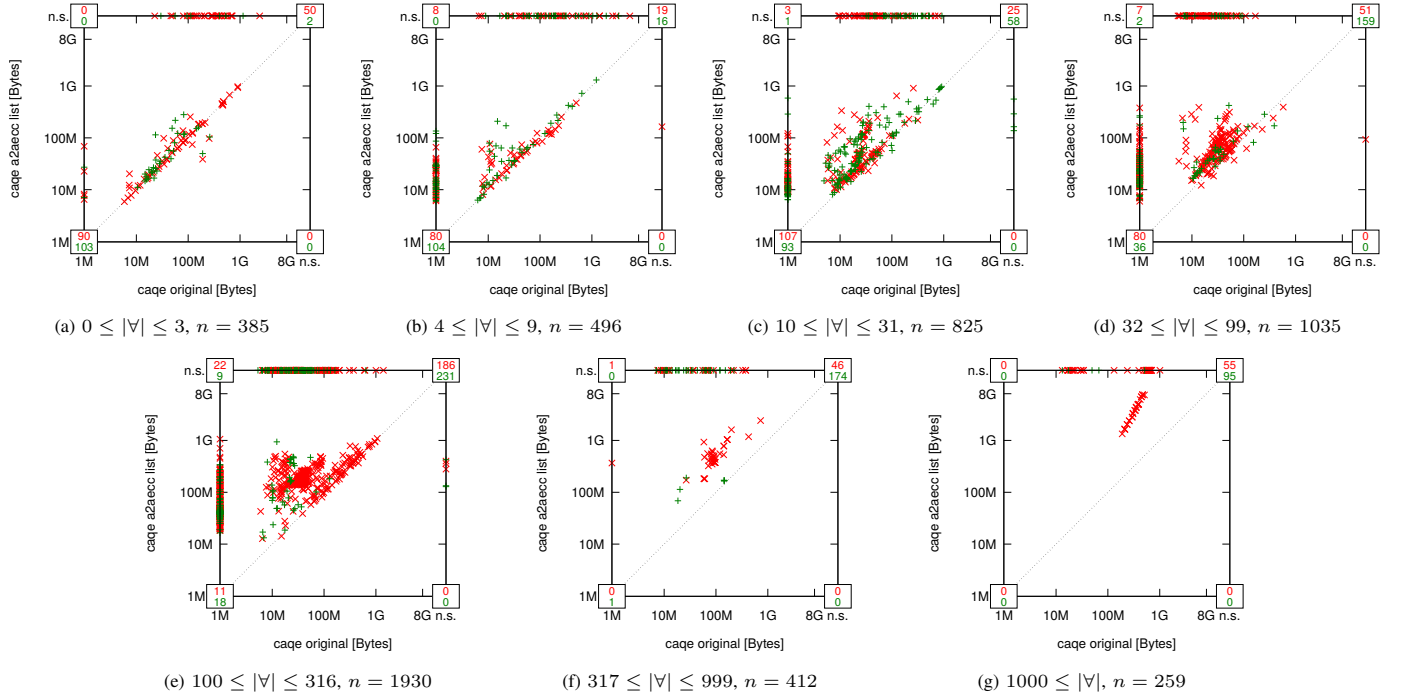


Fig. 1718: Solver CAQE: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

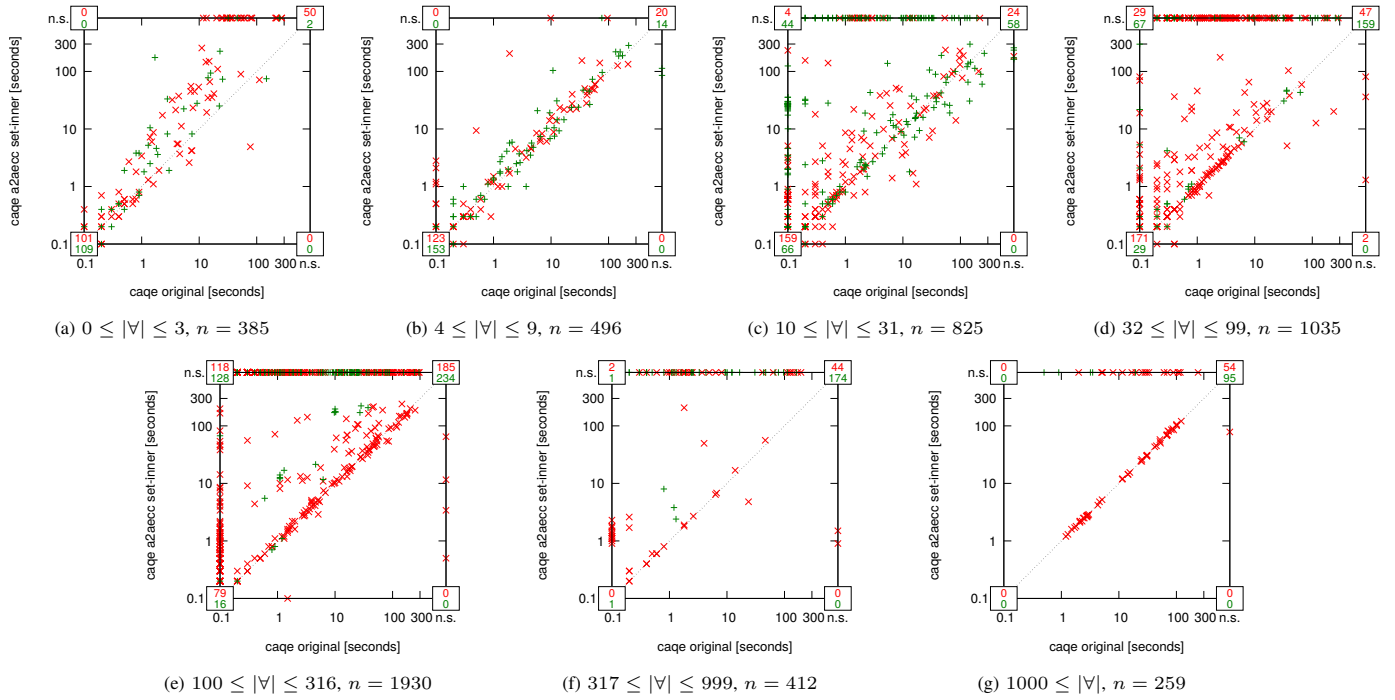


Fig. 1719: Solver CAQE: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by number of \forall quantified variables (run time in seconds).

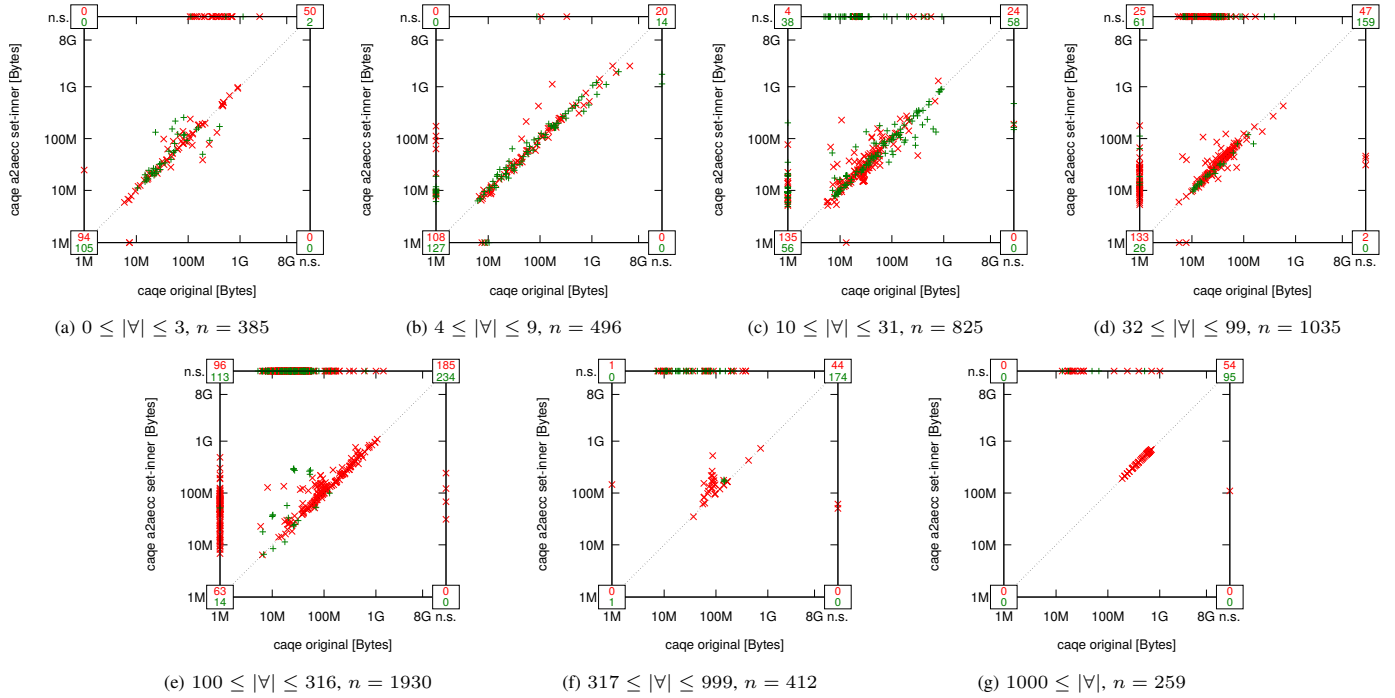


Fig. 1720: Solver CAQE: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

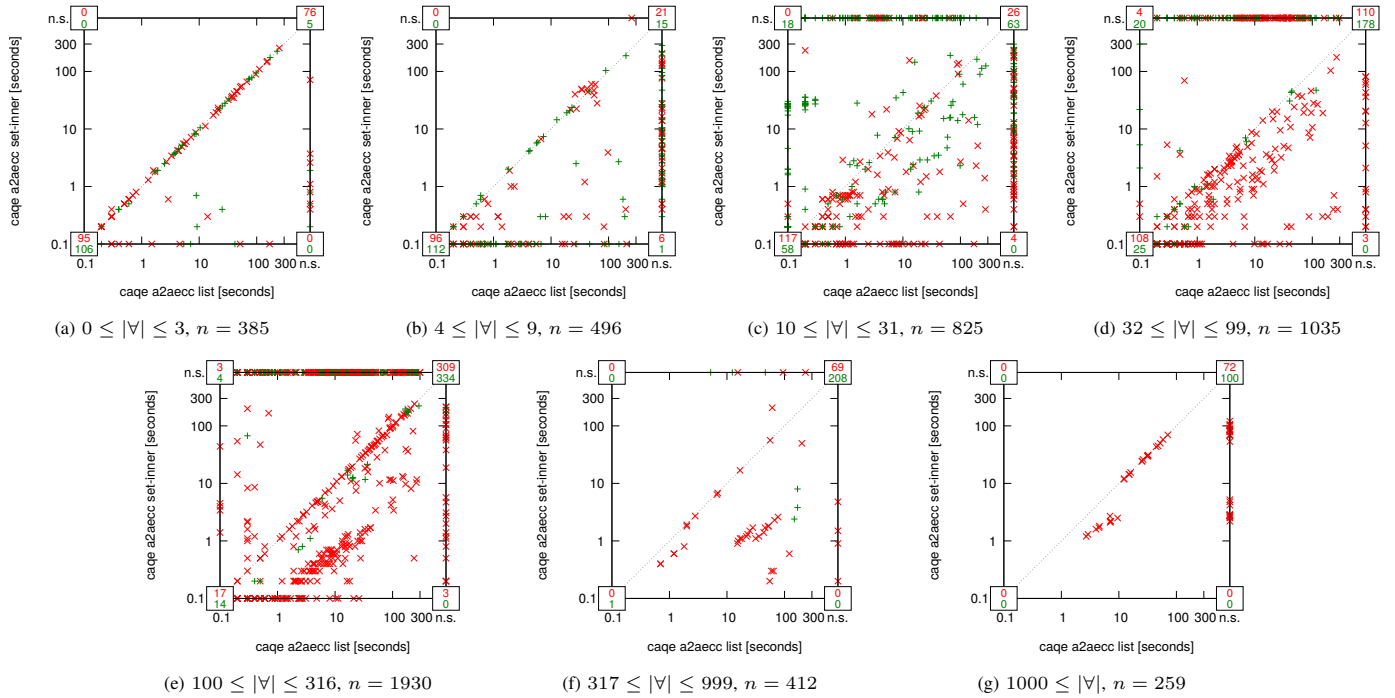


Fig. 1721: Solver CAQE: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by number of \forall quantified variables (run time in seconds).

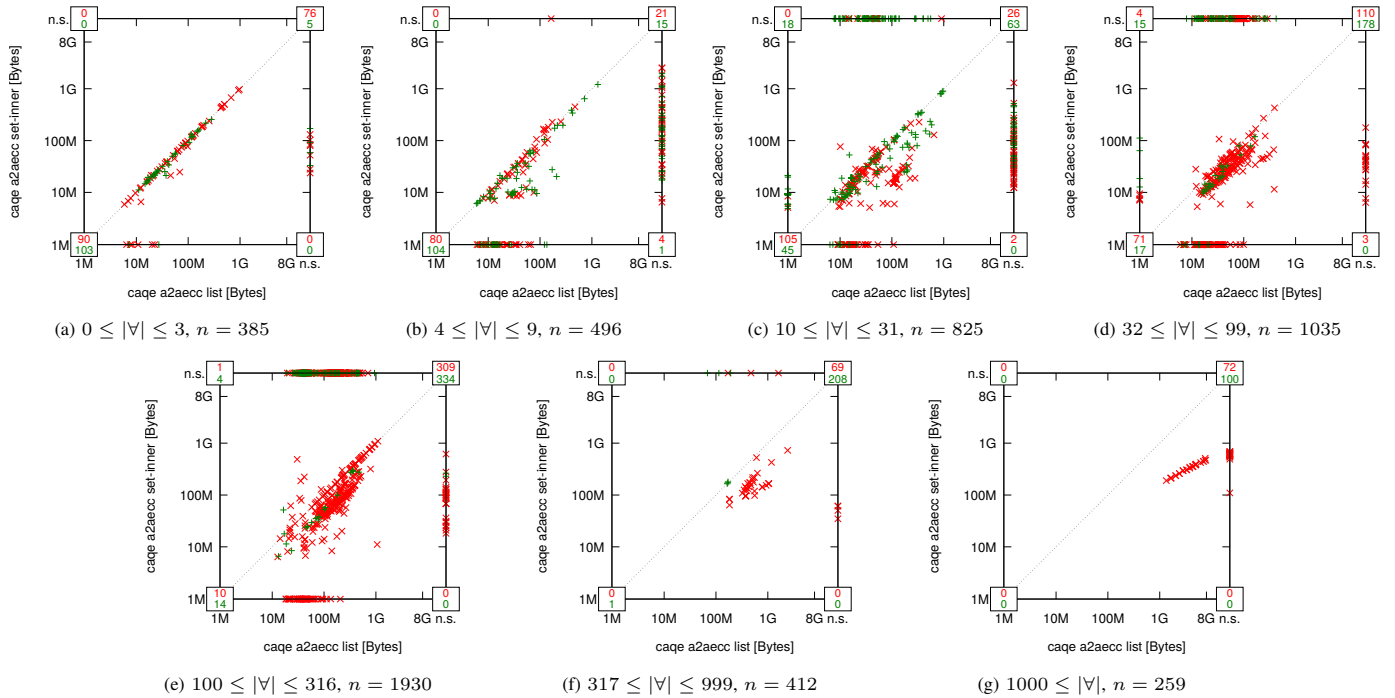


Fig. 1722: Solver CAQE: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

d) GhostQ:

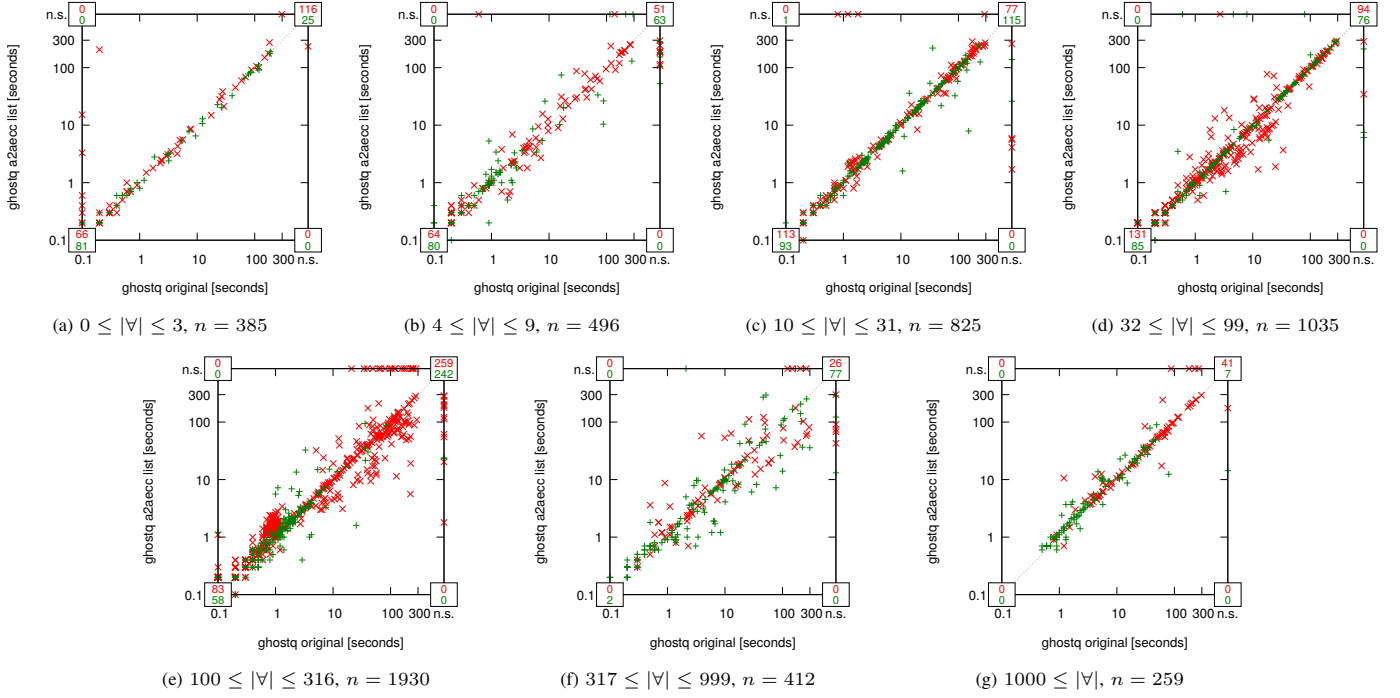


Fig. 1723: Solver GhostQ: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by number of \forall quantified variables (run time in seconds).

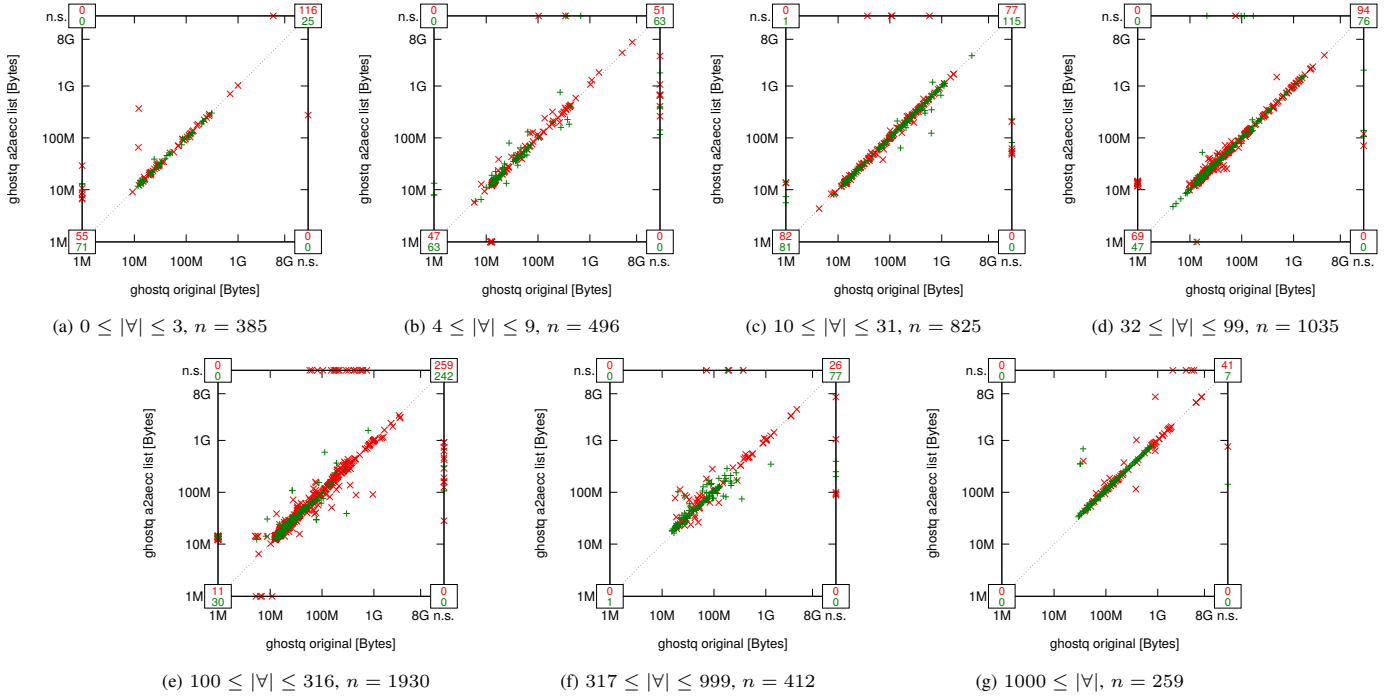


Fig. 1724: Solver GhostQ: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

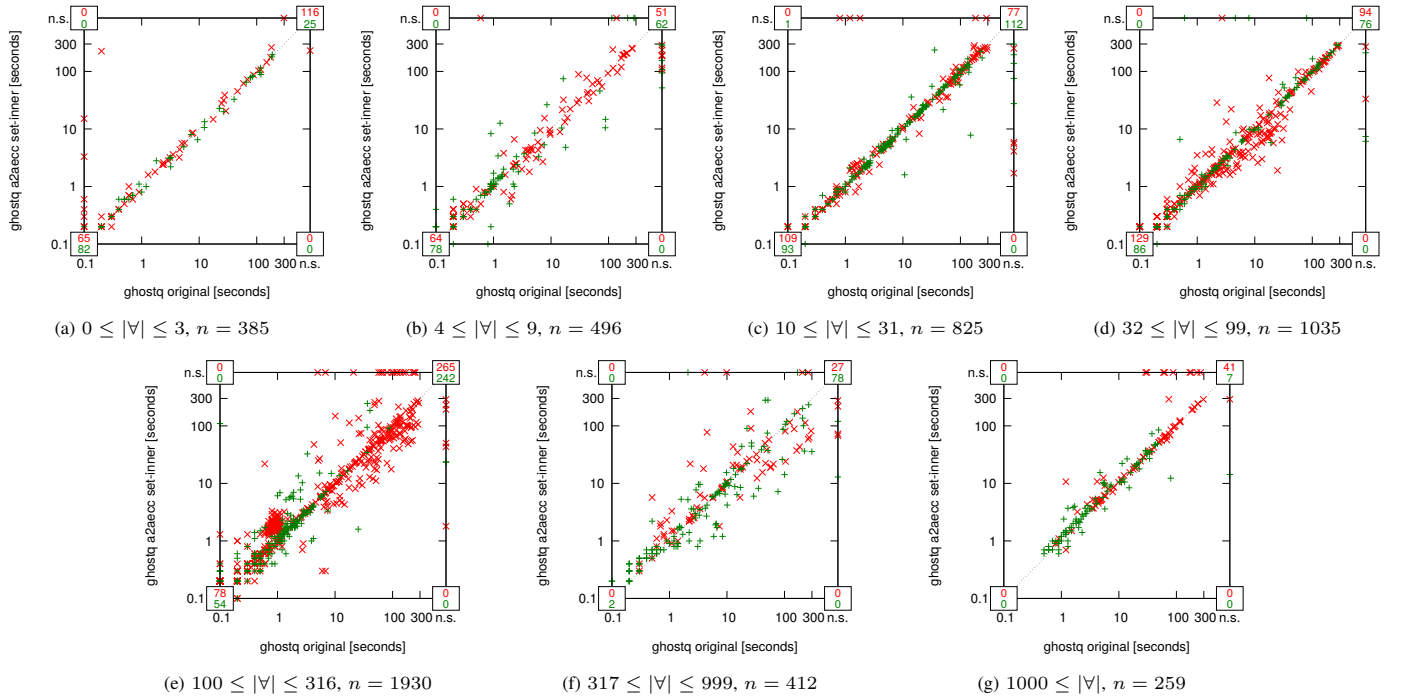


Fig. 1725: Solver `GhostQ`: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by number of \forall quantified variables (run time in seconds).

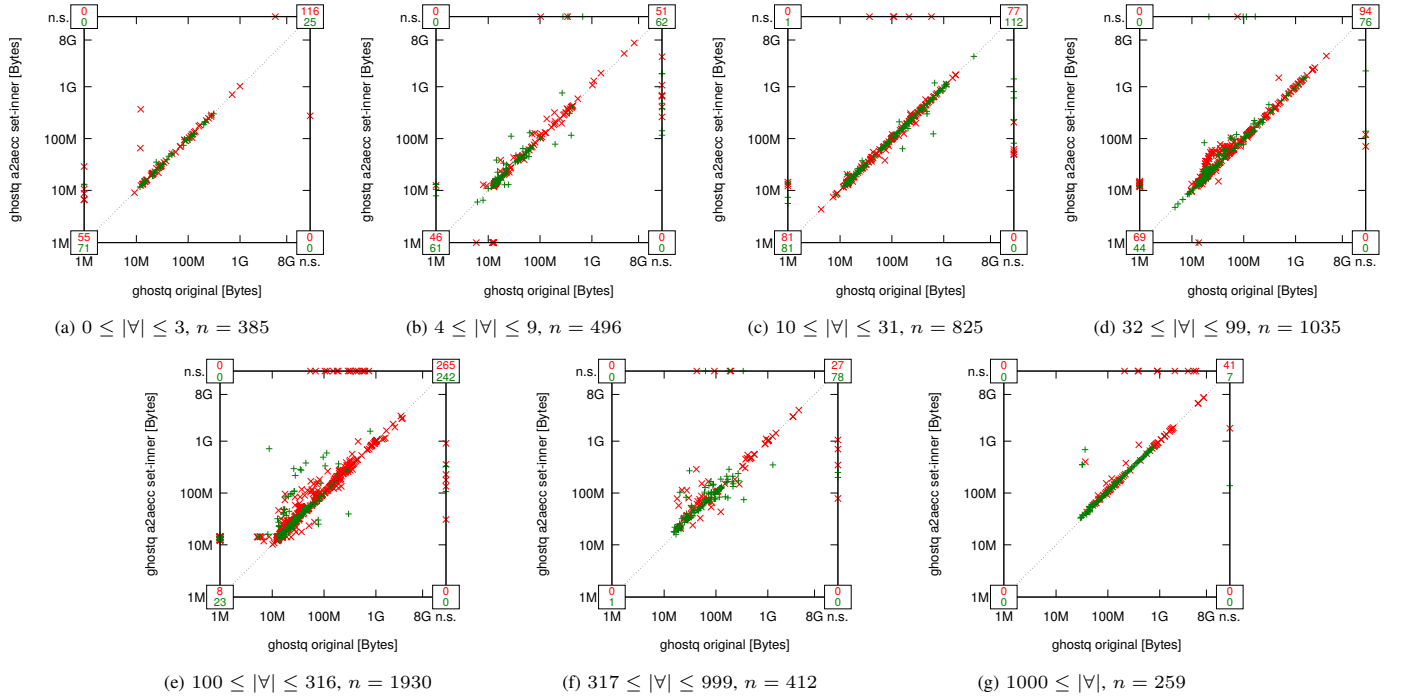


Fig. 1726: Solver `GhostQ`: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

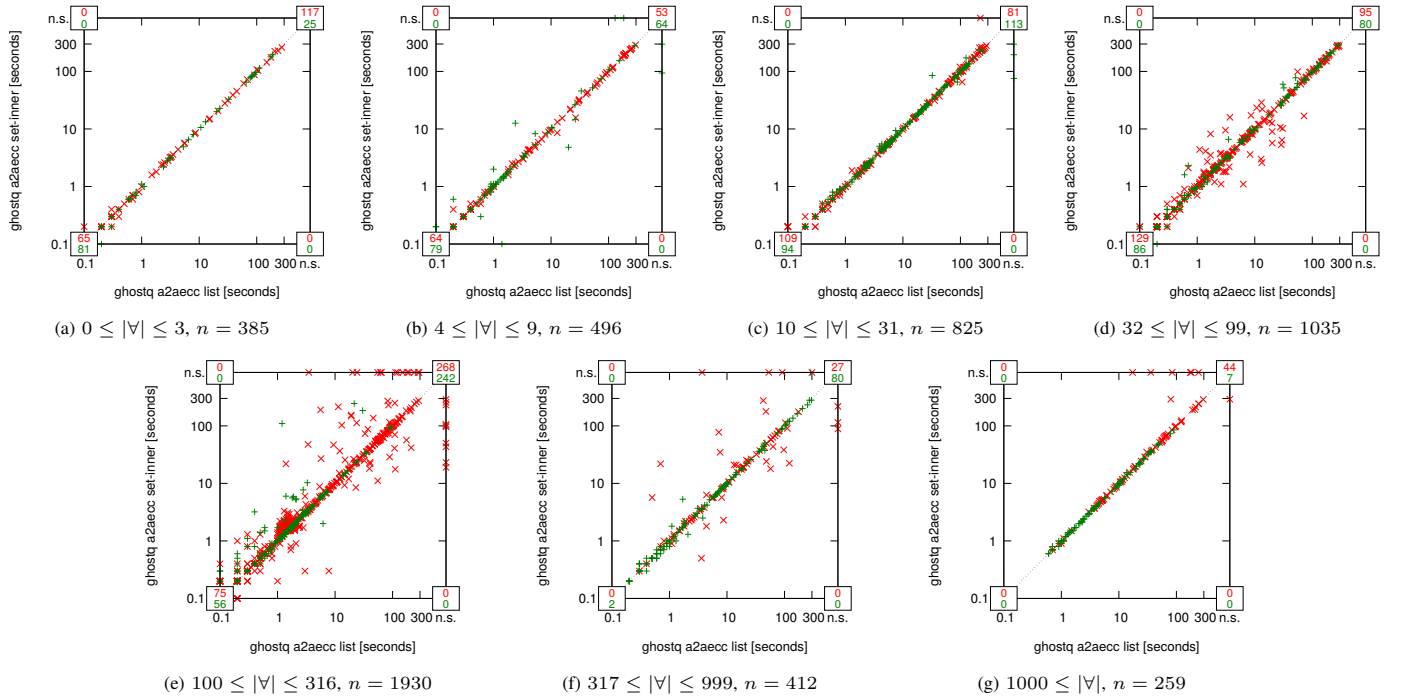


Fig. 1727: Solver GhostQ: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by number of \forall quantified variables (run time in seconds).

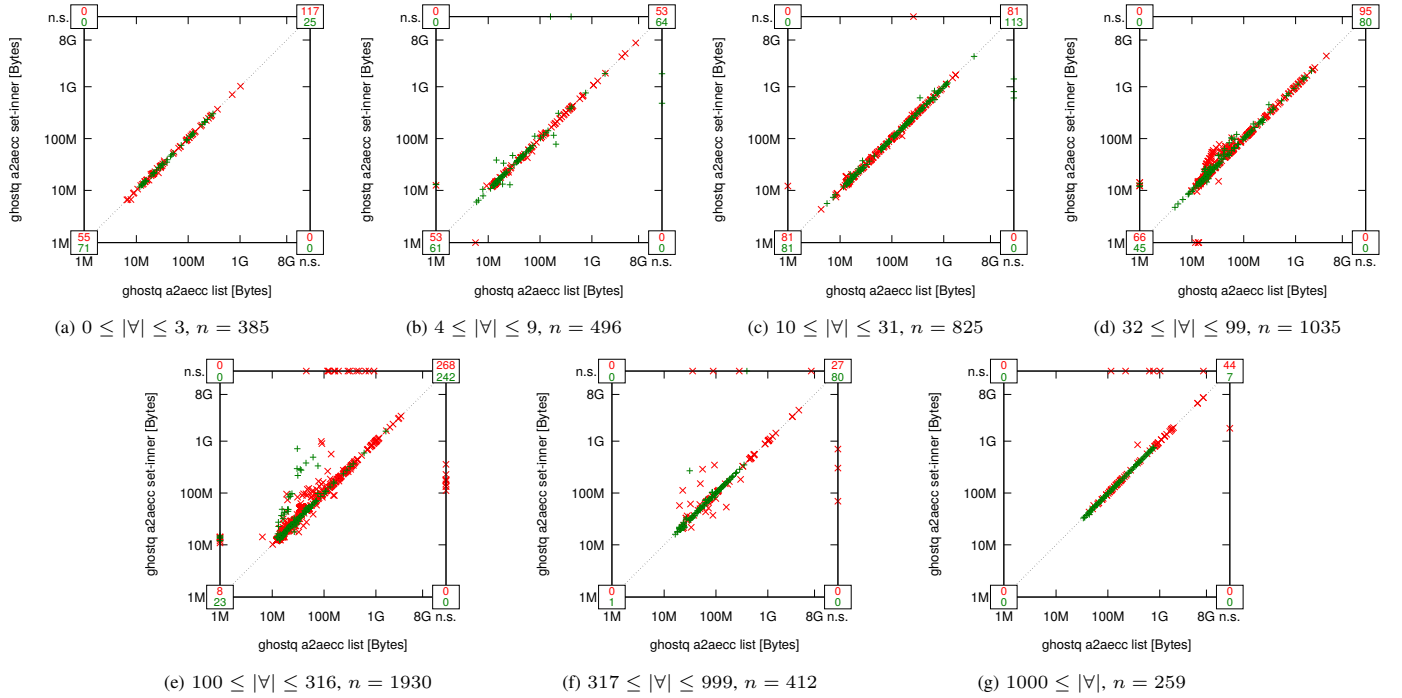


Fig. 1728: Solver GhostQ: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

e) QESTO:

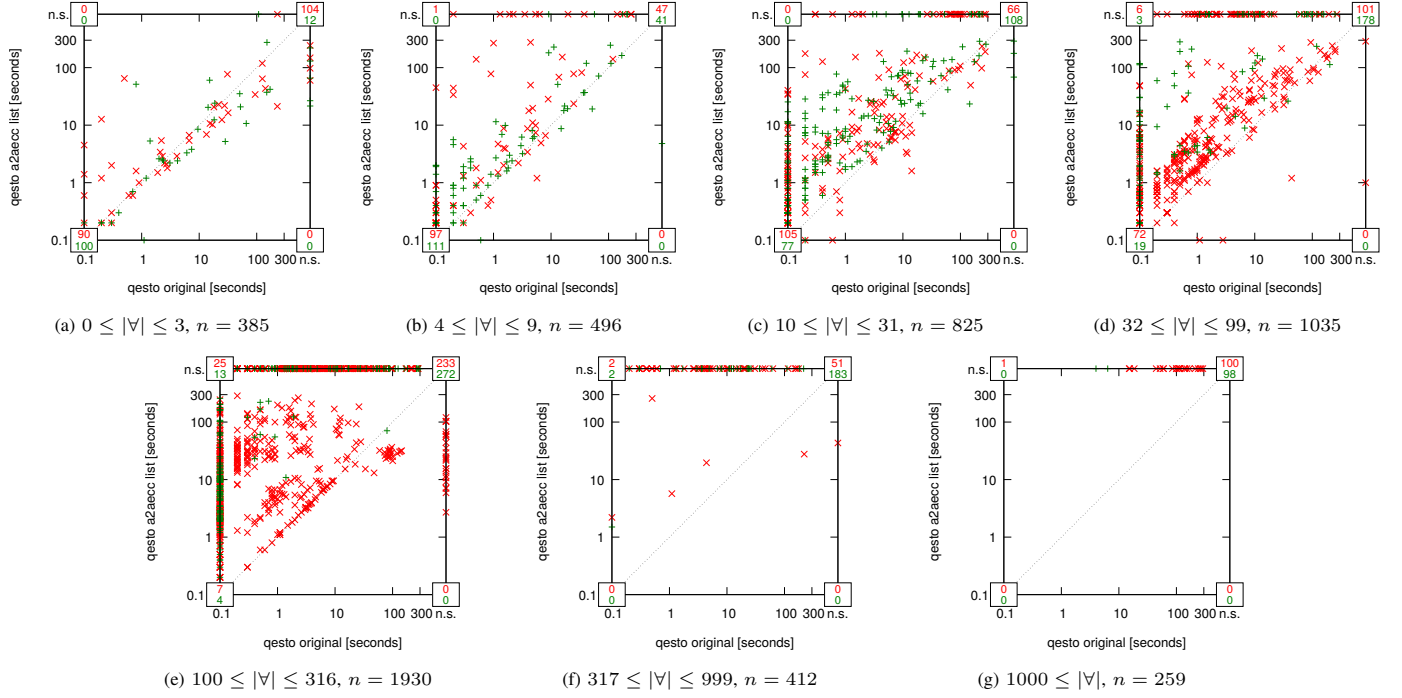


Fig. 1729: Solver QESTO: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by number of \forall quantified variables (run time in seconds).

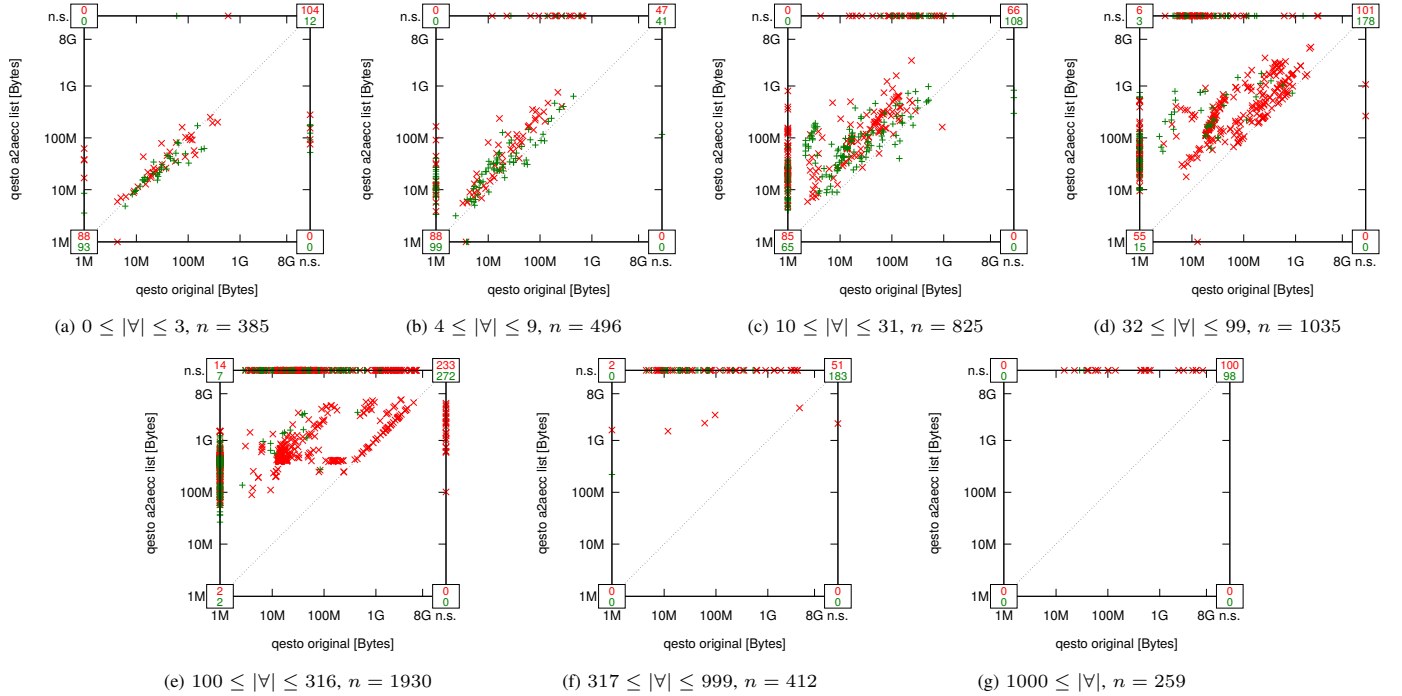


Fig. 1730: Solver QESTO: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

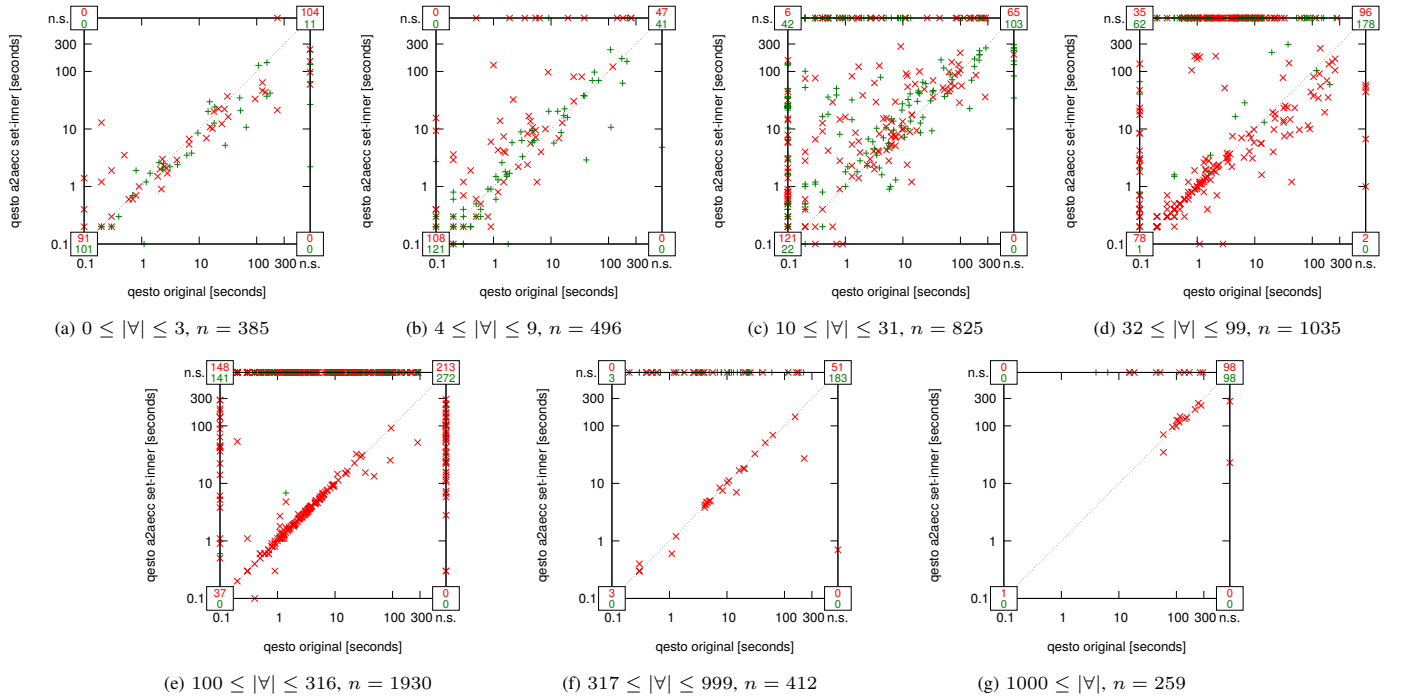


Fig. 1731: Solver QESTO: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by number of \forall quantified variables (run time in seconds).

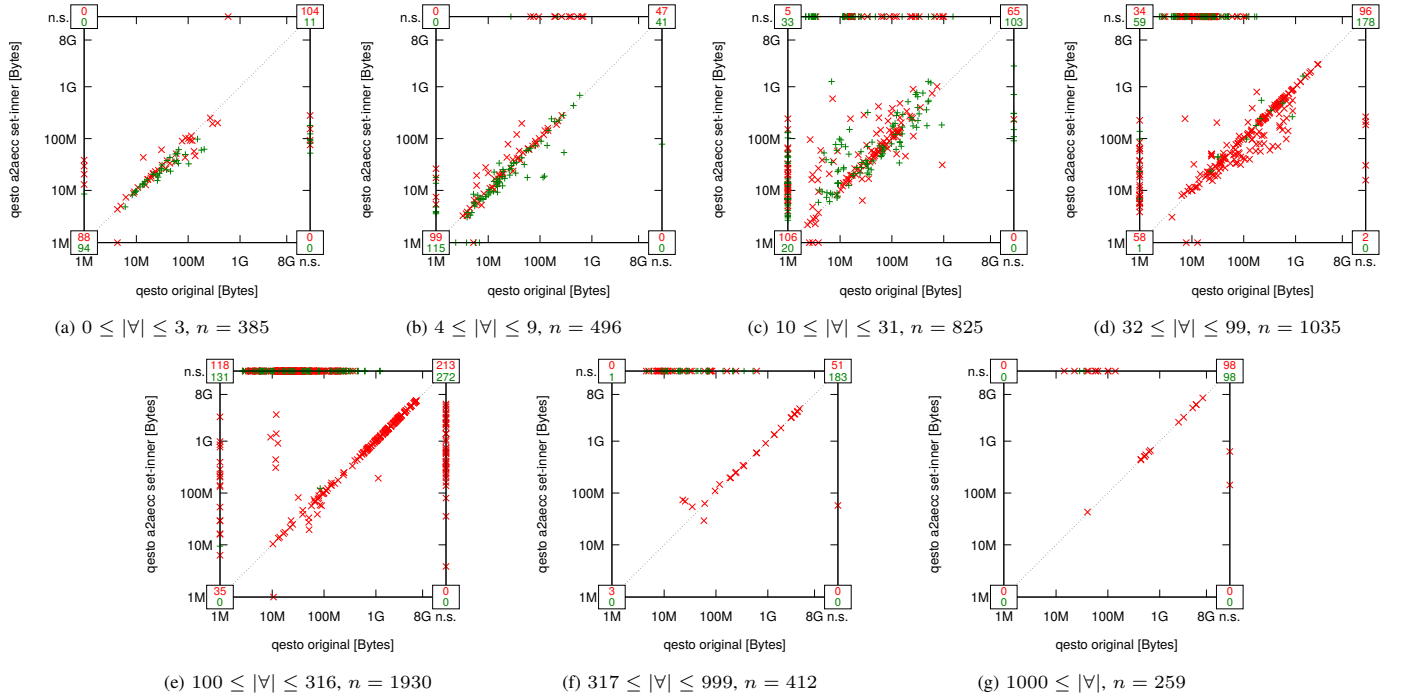


Fig. 1732: Solver QESTO: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

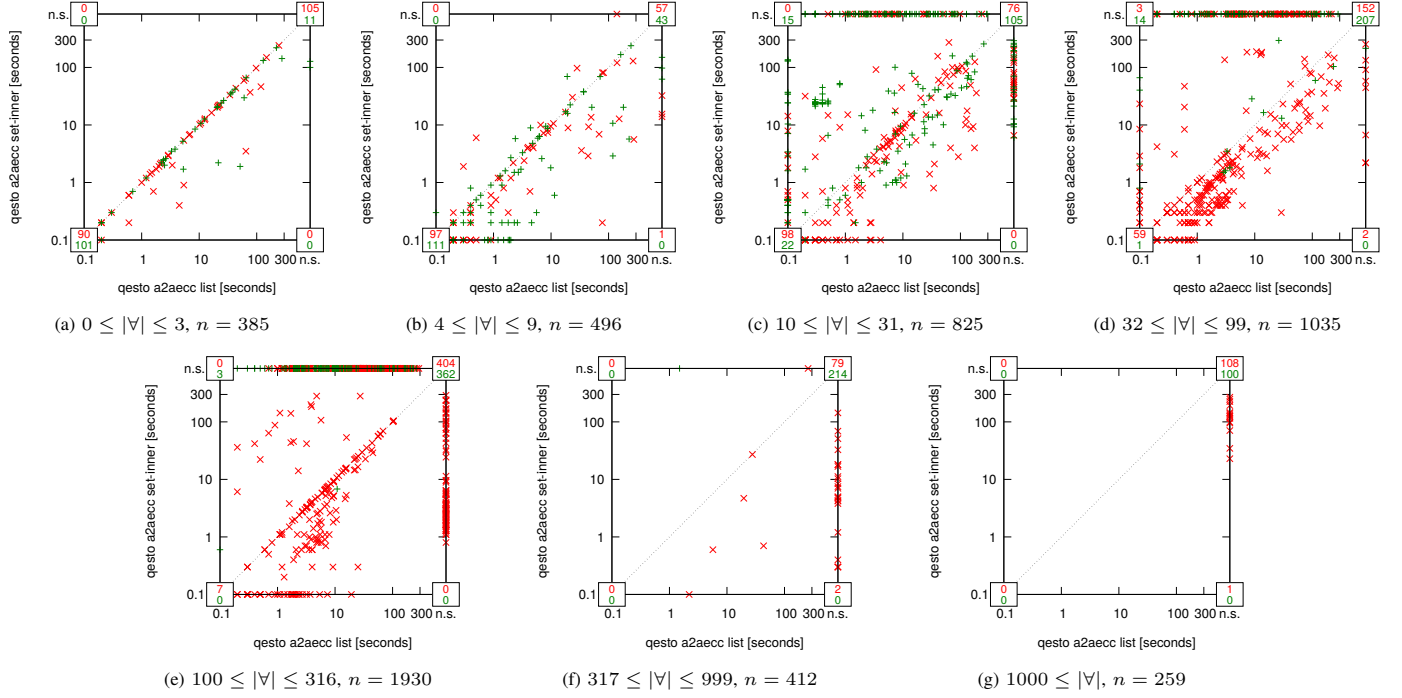


Fig. 1733: Solver QESTO: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by number of \forall quantified variables (run time in seconds).

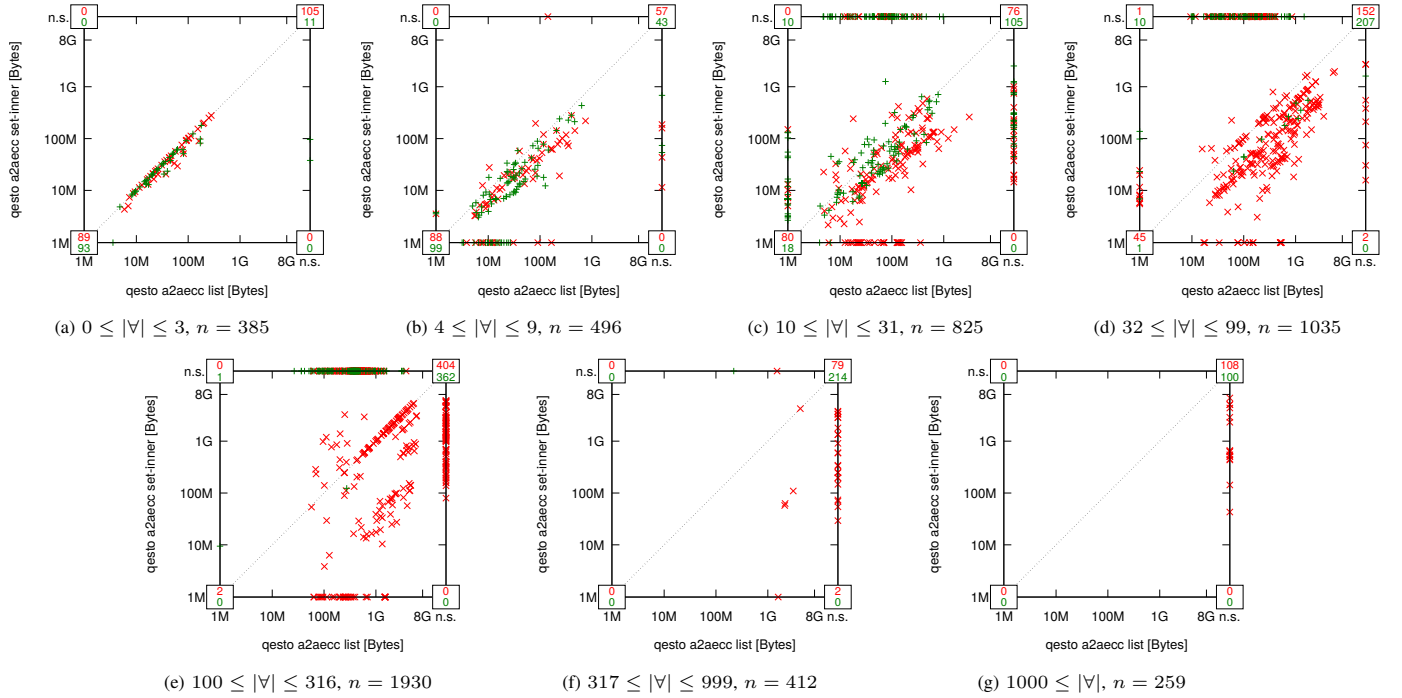


Fig. 1734: Solver QESTO: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

f) RReQS:

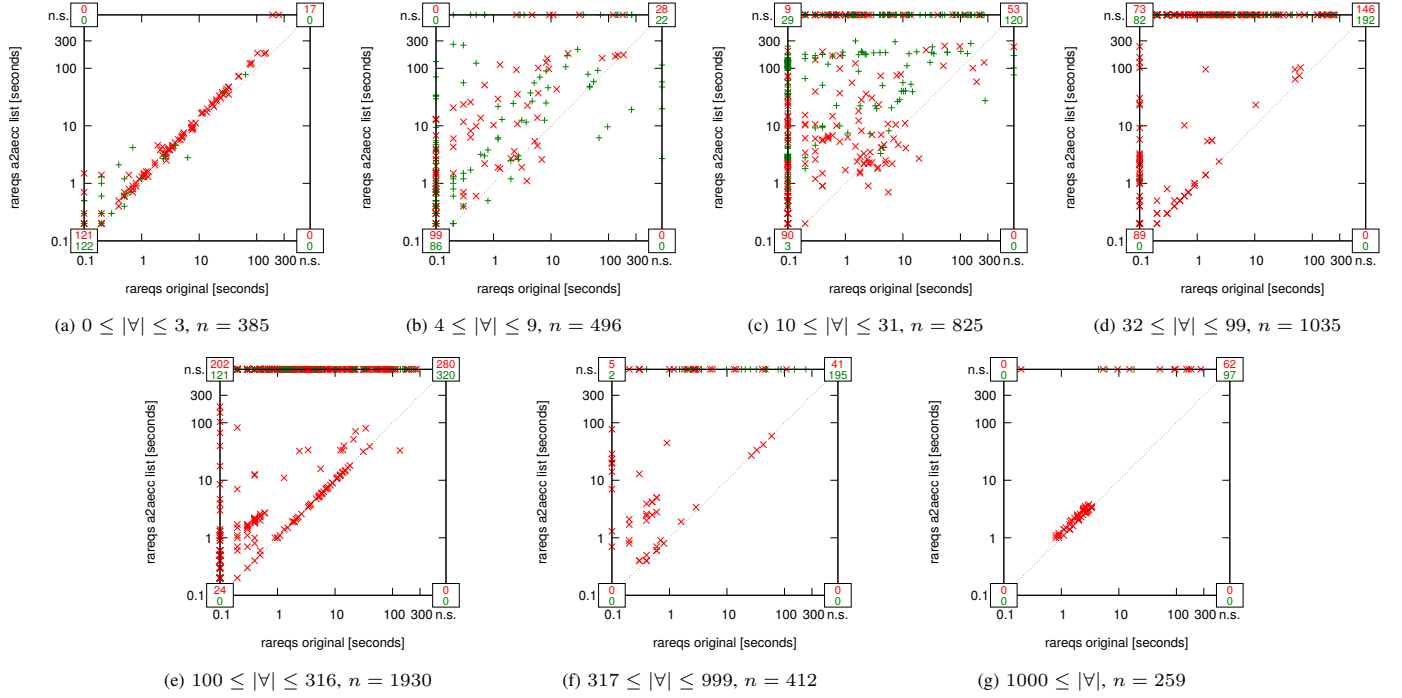


Fig. 1735: Solver RReQS: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by number of \forall quantified variables (run time in seconds).

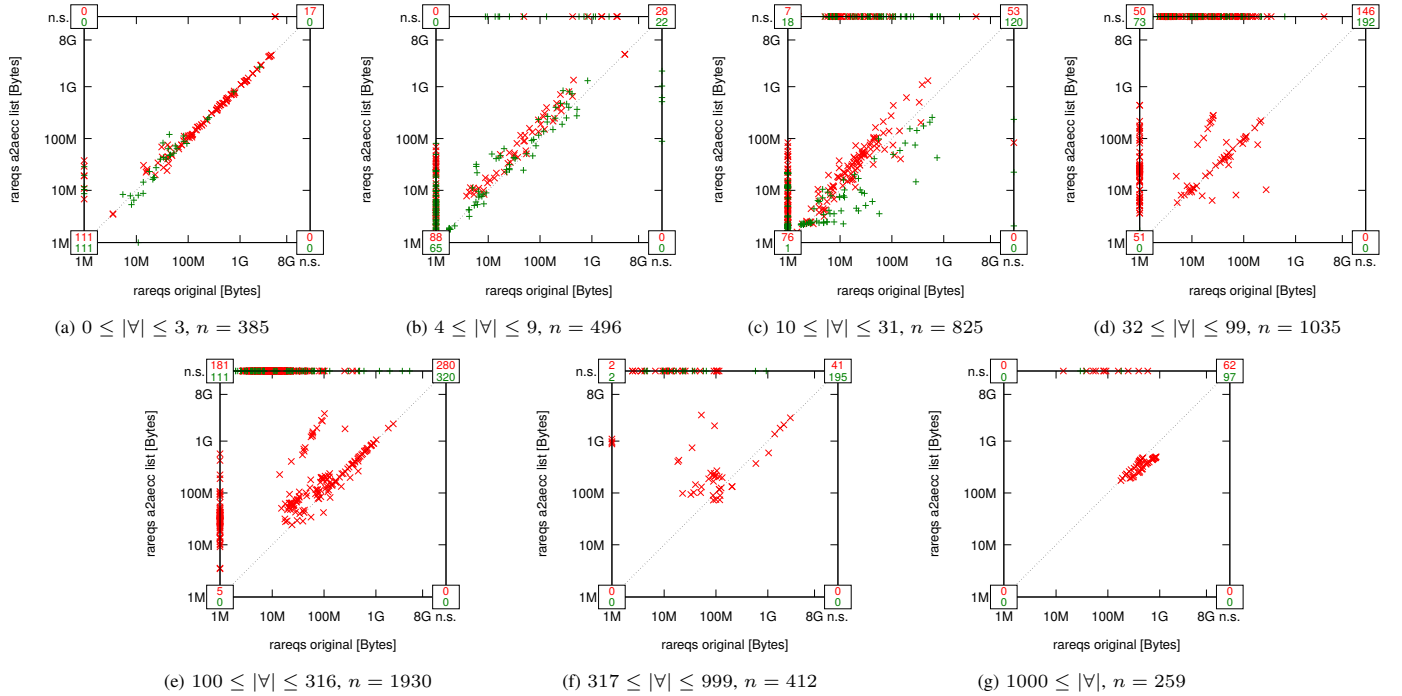


Fig. 1736: Solver RReQS: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

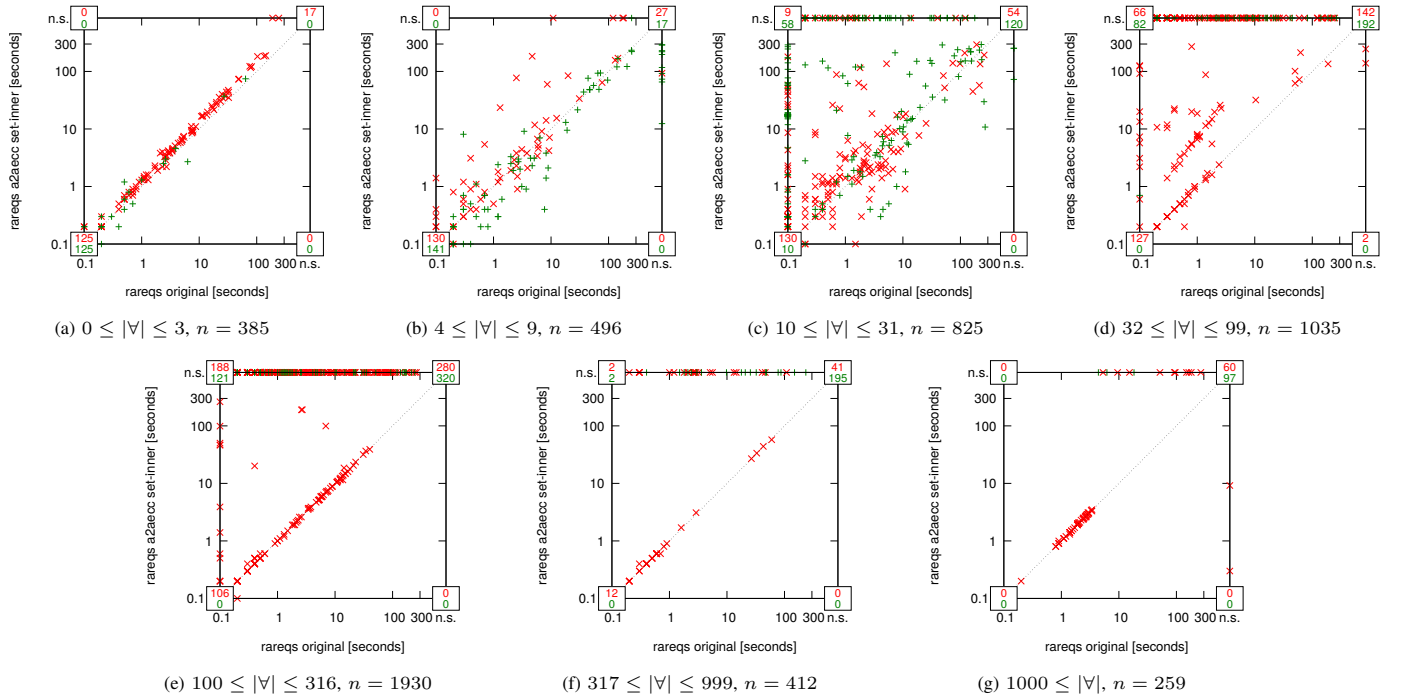


Fig. 1737: Solver RAREQS: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by number of \forall quantified variables (run time in seconds).

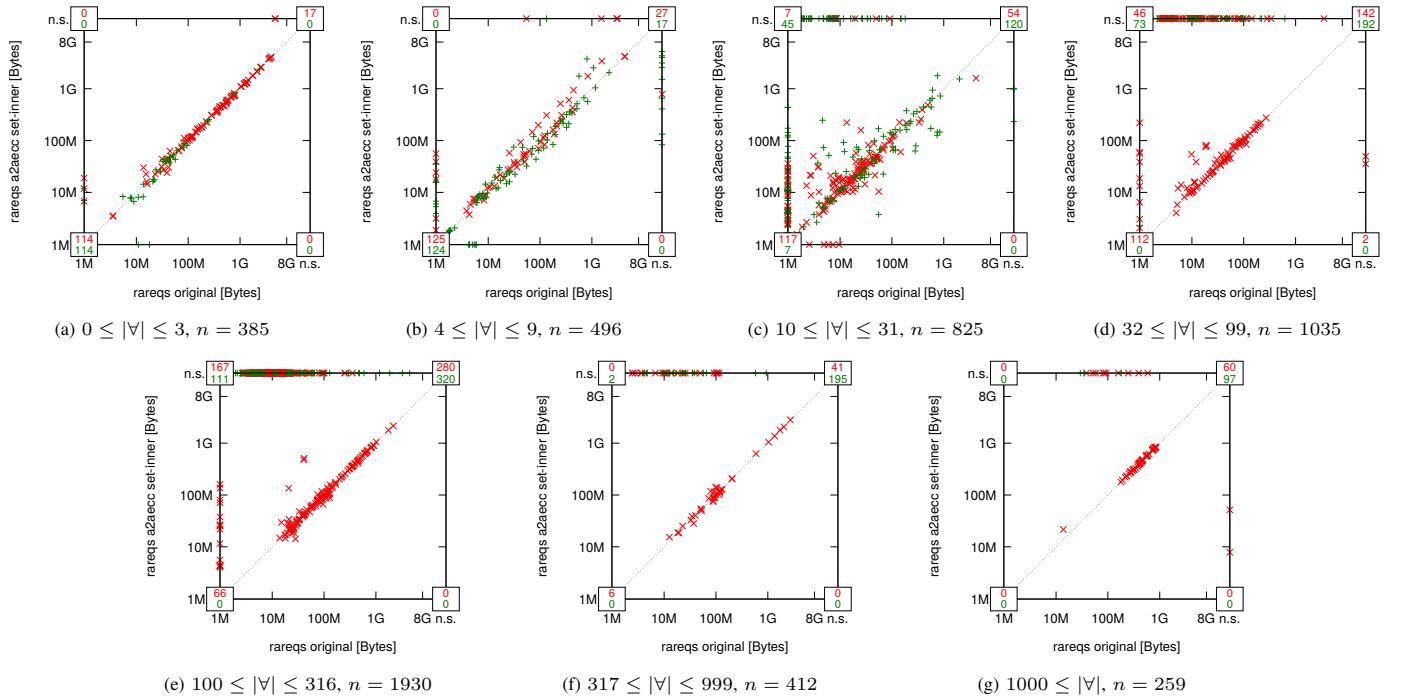


Fig. 1738: Solver RAREQS: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

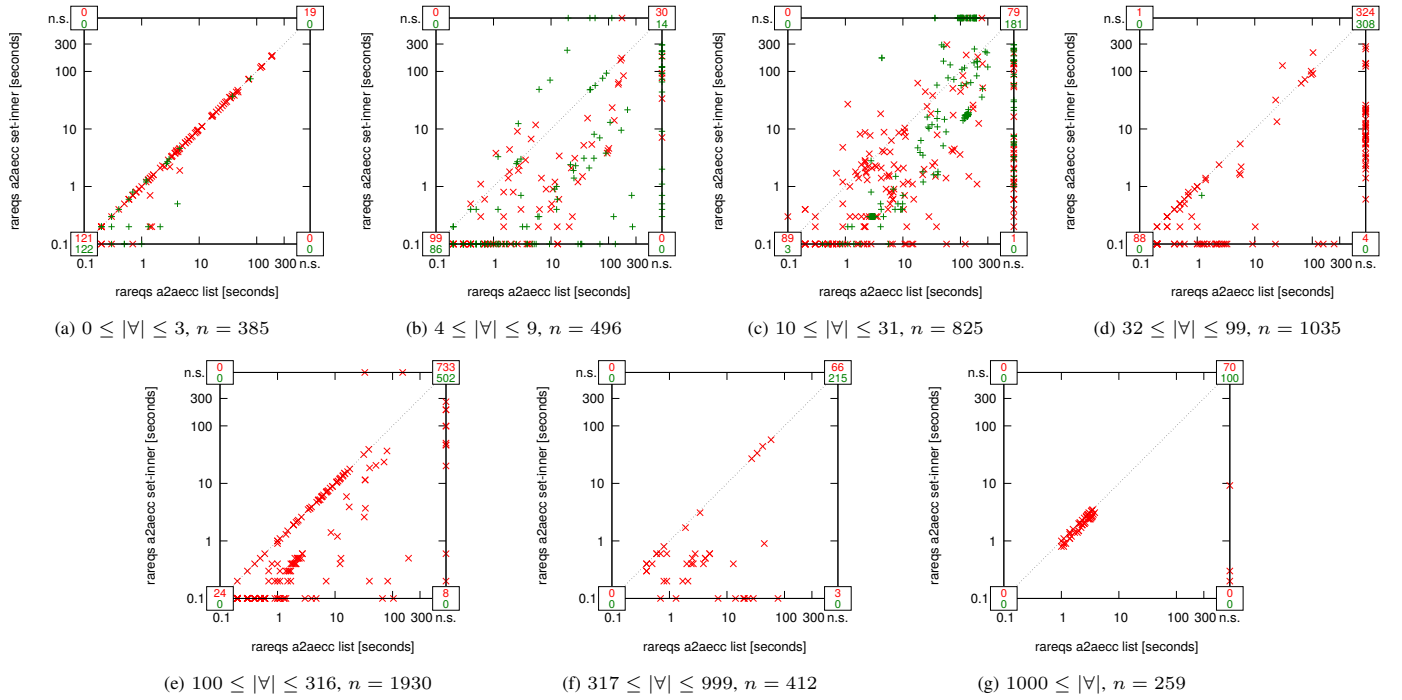


Fig. 1739: Solver RAReQS: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by number of \forall quantified variables (run time in seconds).

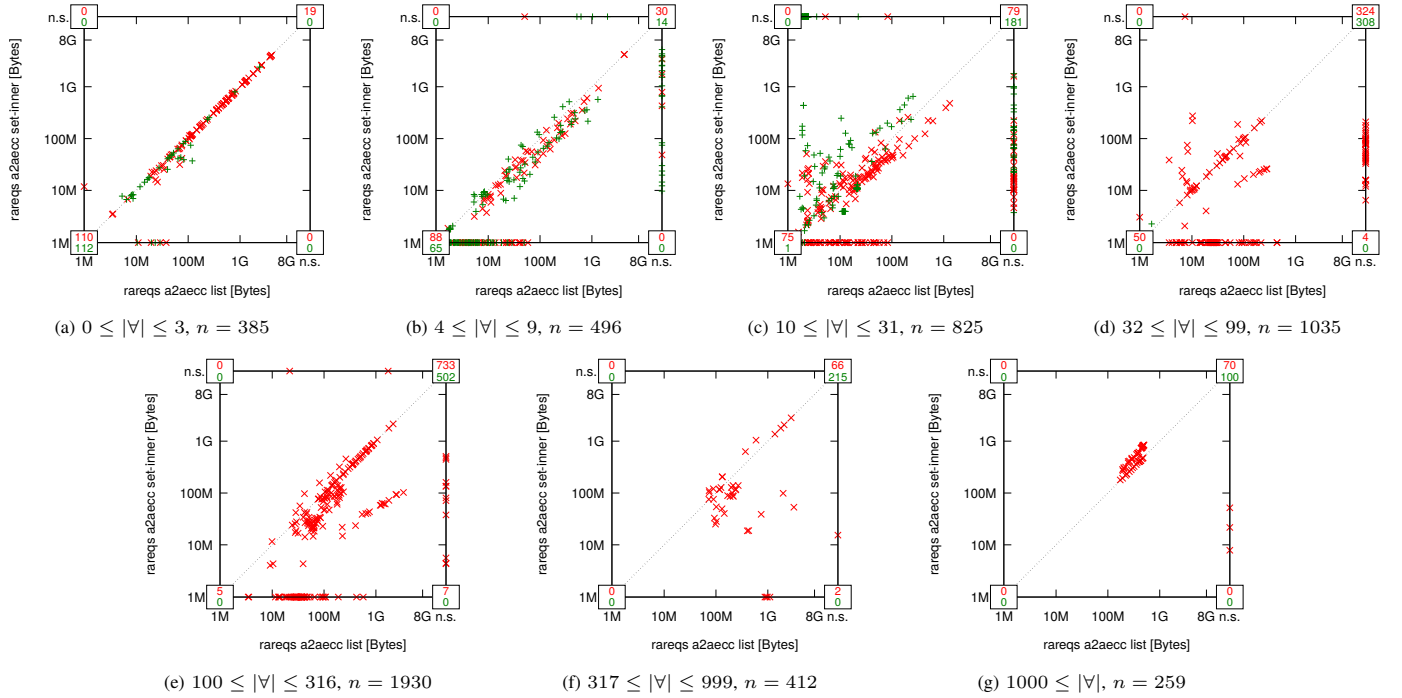


Fig. 1740: Solver RAReQS: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by number of \forall quantified variables (memory usage in Bytes).

C. Partitioned by Alternation Depth

In this subsection there are 6 figures for each solver with subfigures for partitions of the benchmarks according to their alternation depths:

- 1) Comparing run times for solving the transformed versus the original instances with list semantics.
- 2) Memory usage for 1.
- 3) As 1. but with set-inner semantics.
- 4) Memory usage for 3.
- 5) A direct comparison of solving the transformed instances with set-inner versus list semantics.
- 6) Memory usage for 5.

a) DepQBF:

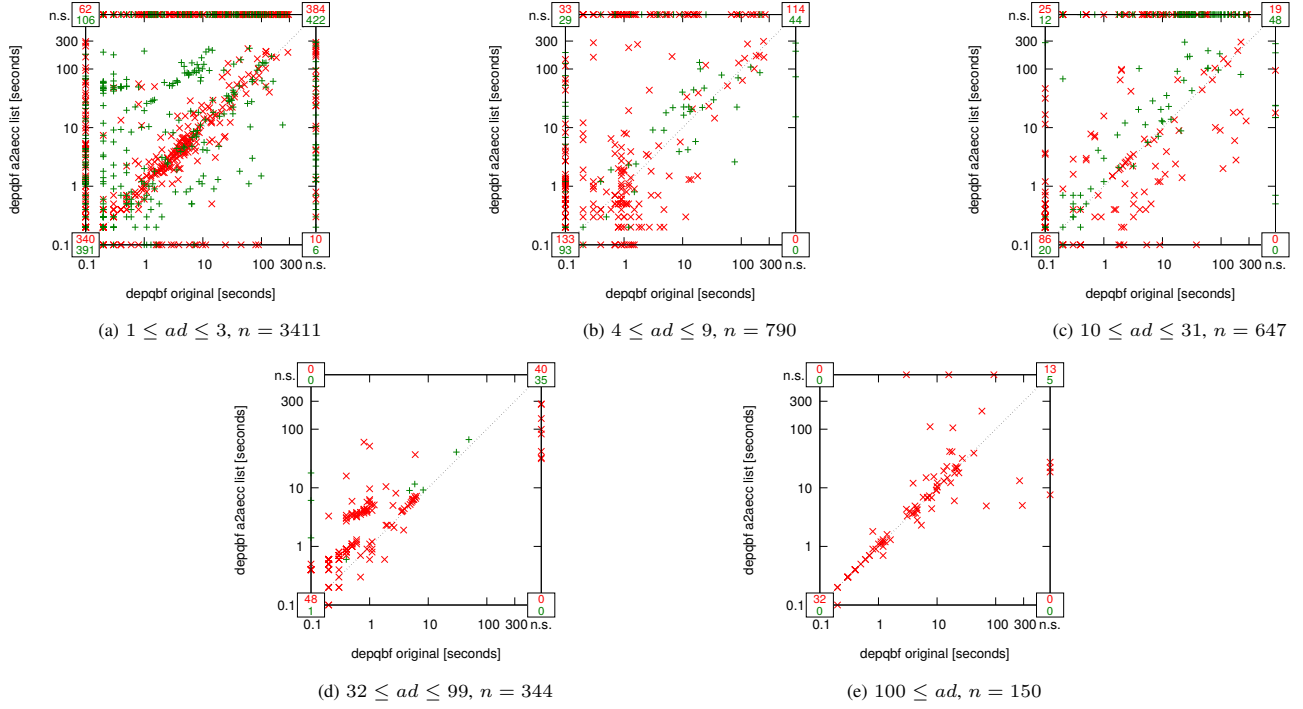


Fig. 1741: Solver DepQBF: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by alternation depth (run time in seconds).

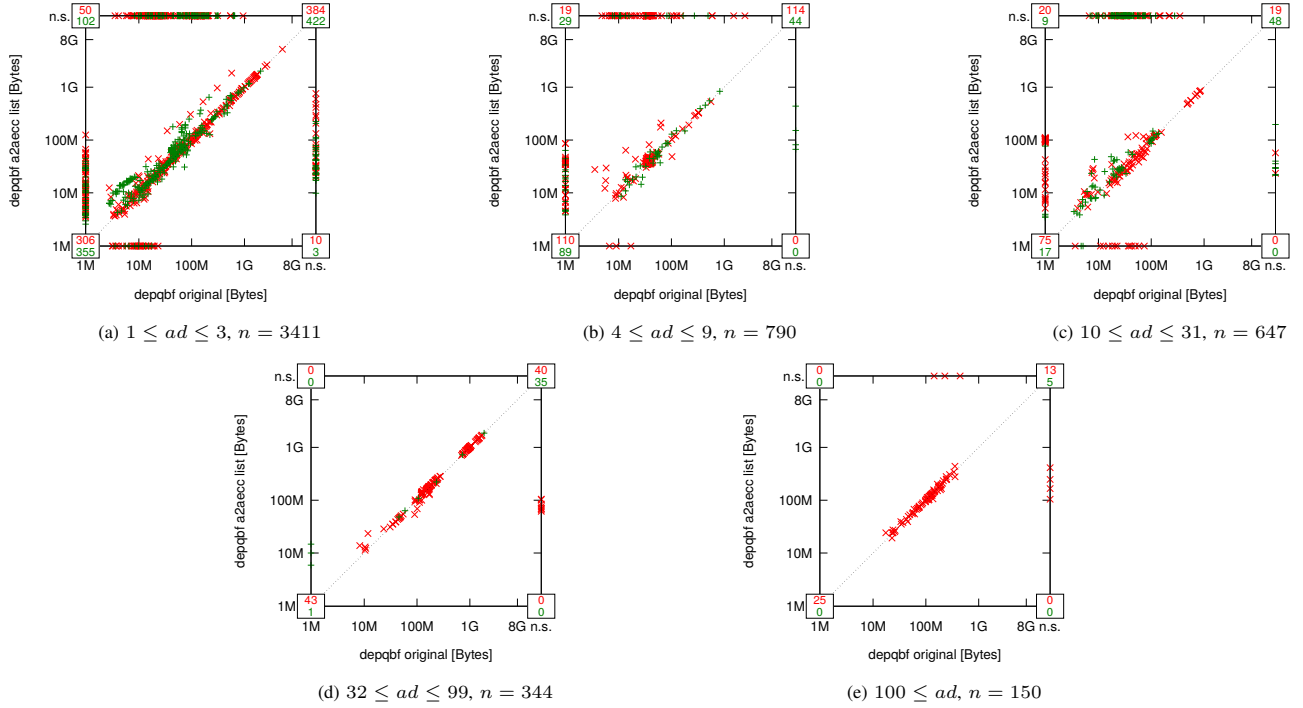


Fig. 1742: Solver DepQBF: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by alternation depth (memory usage in Bytes).

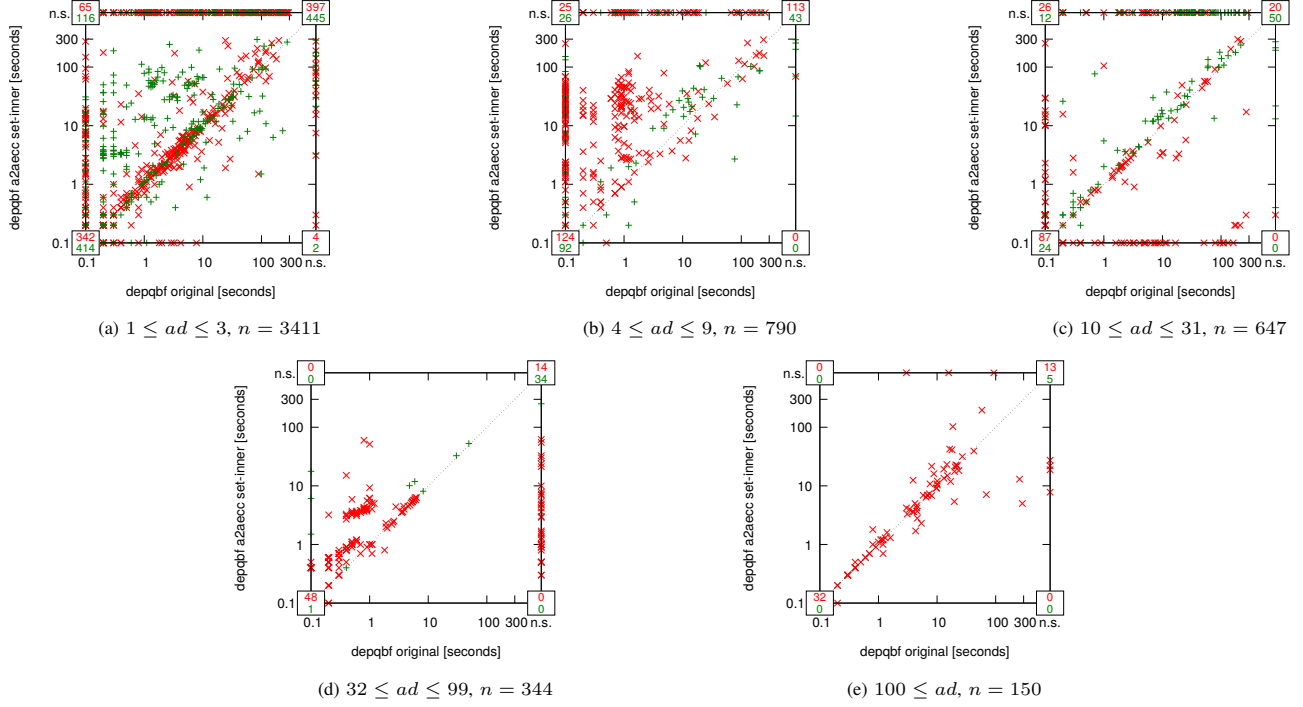


Fig. 1743: Solver DepQBF: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by alternation depth (run time in seconds).

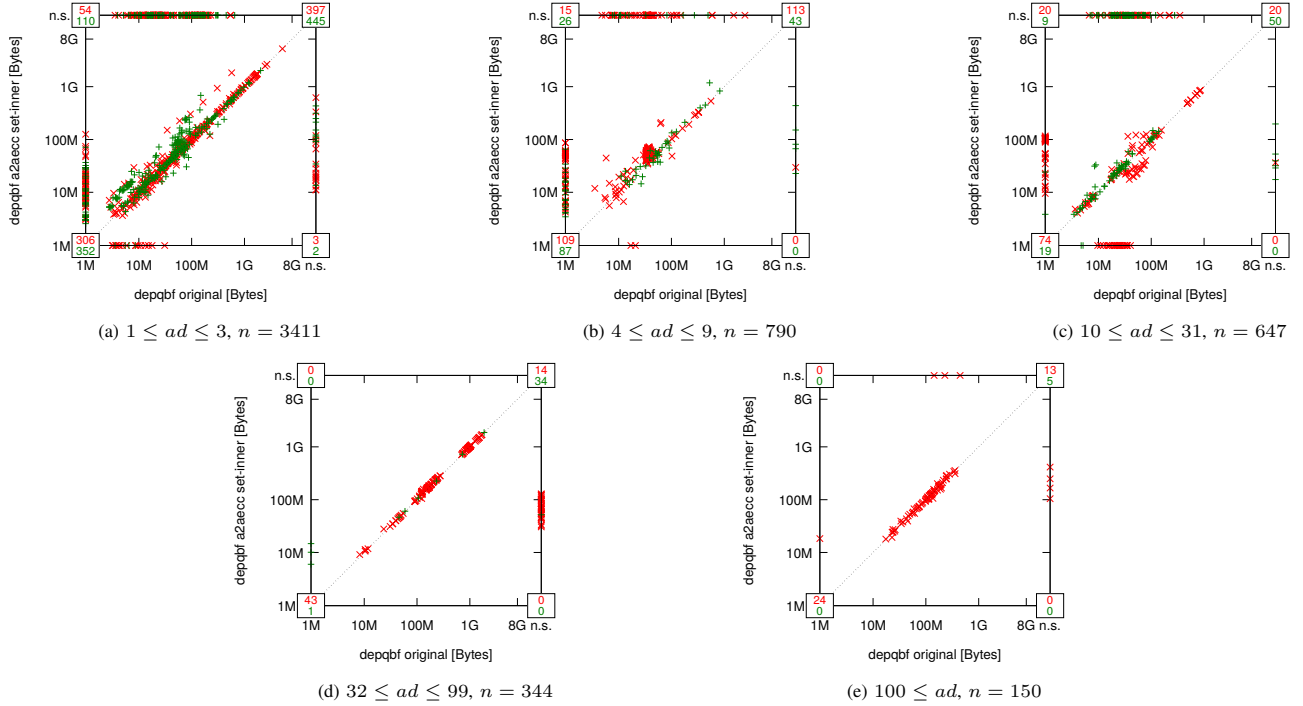


Fig. 1744: Solver DepQBF: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by alternation depth (memory usage in Bytes).

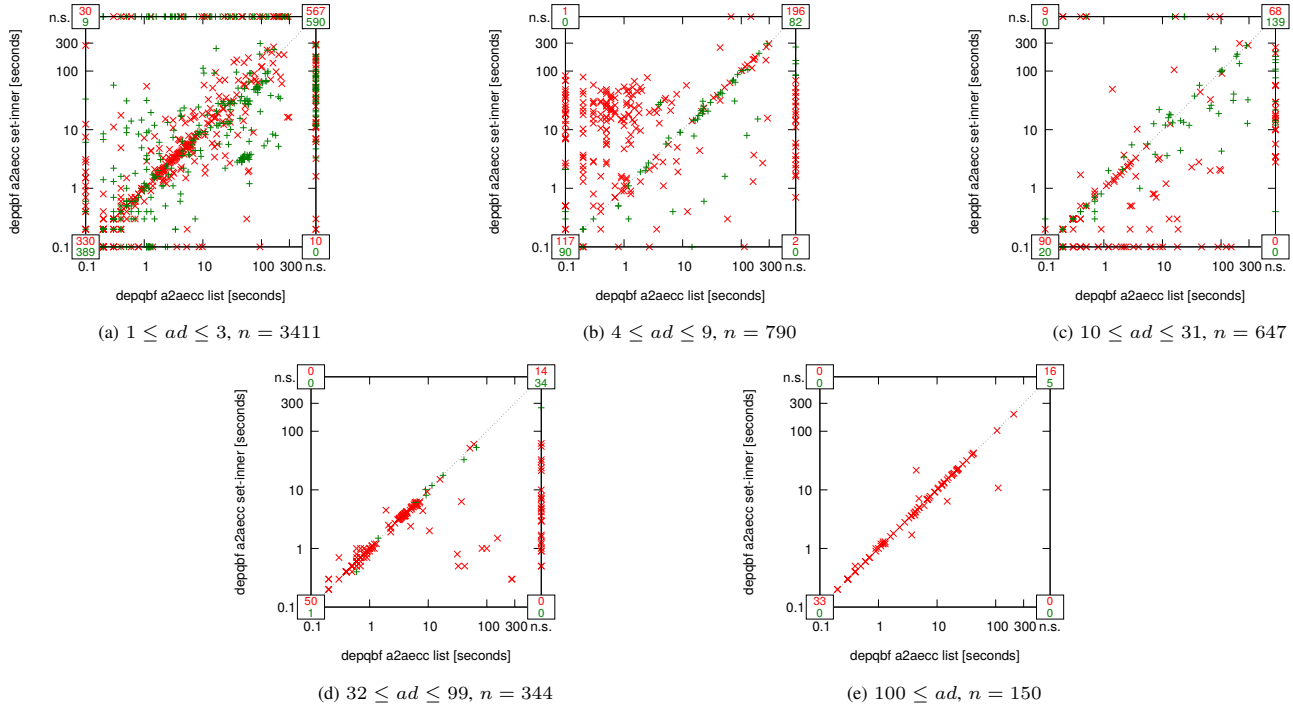


Fig. 1745: Solver DepQBF : Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by alternation depth (run time in seconds).

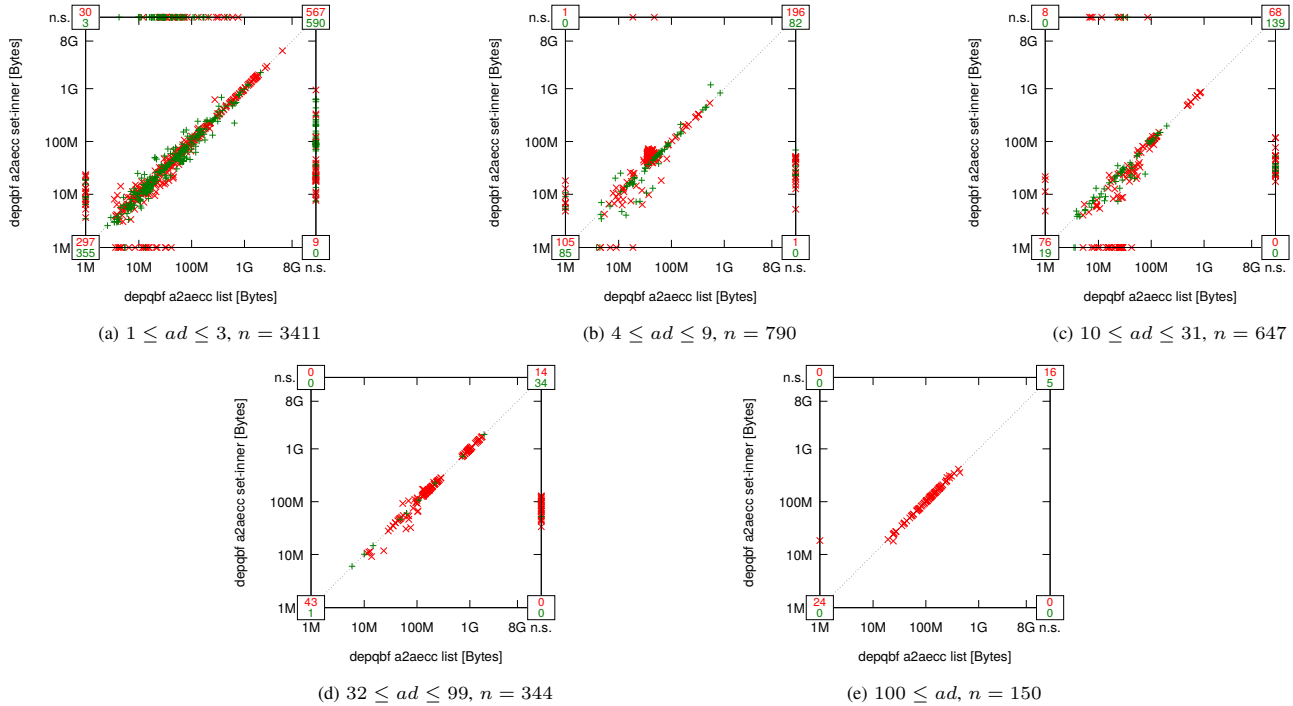


Fig. 1746: Solver DepQBF : Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by alternation depth (memory usage in Bytes).

b) AIGSolve:

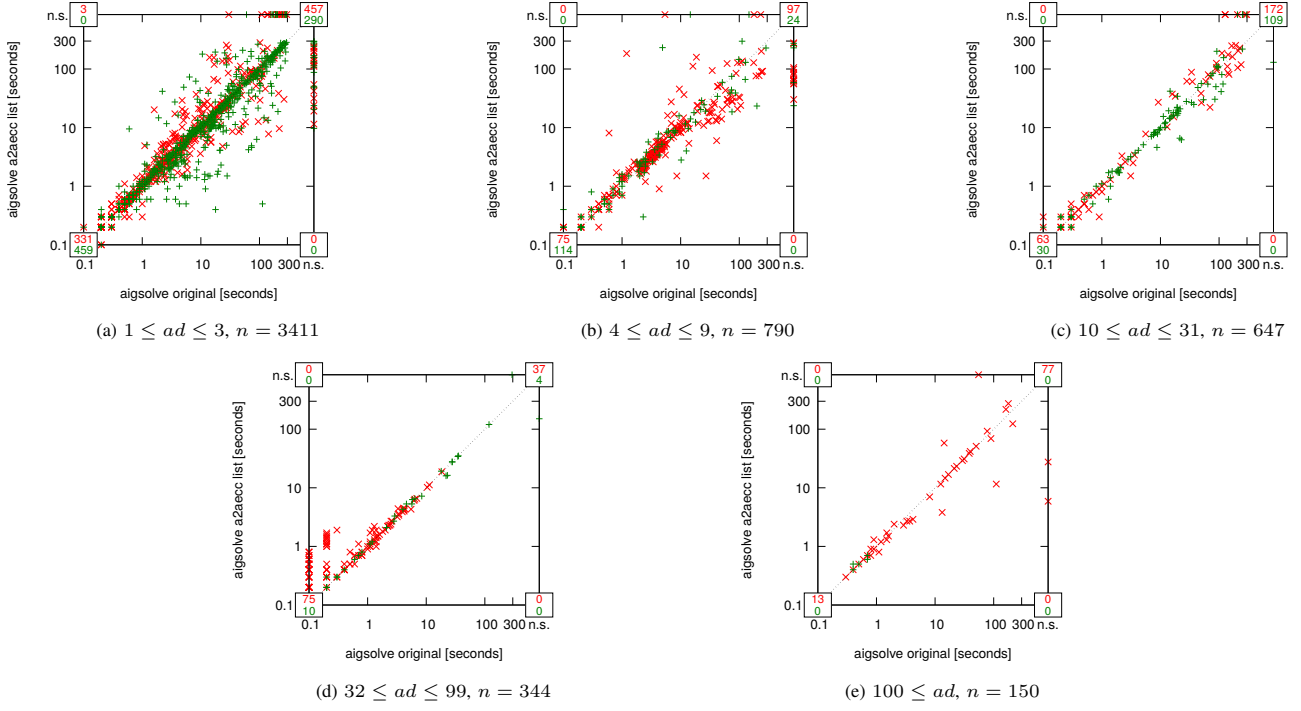


Fig. 1747: Solver AIGSolve: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by alternation depth (run time in seconds).

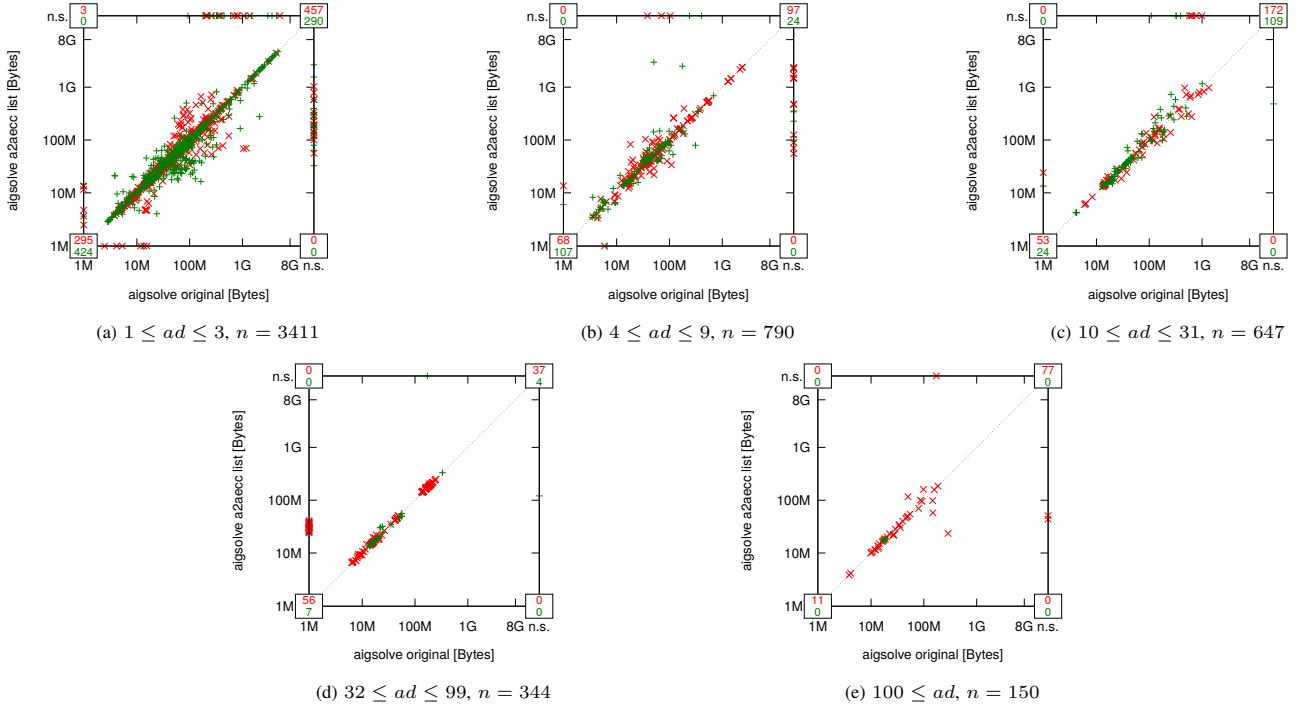


Fig. 1748: Solver AIGSolve: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by alternation depth (memory usage in Bytes).

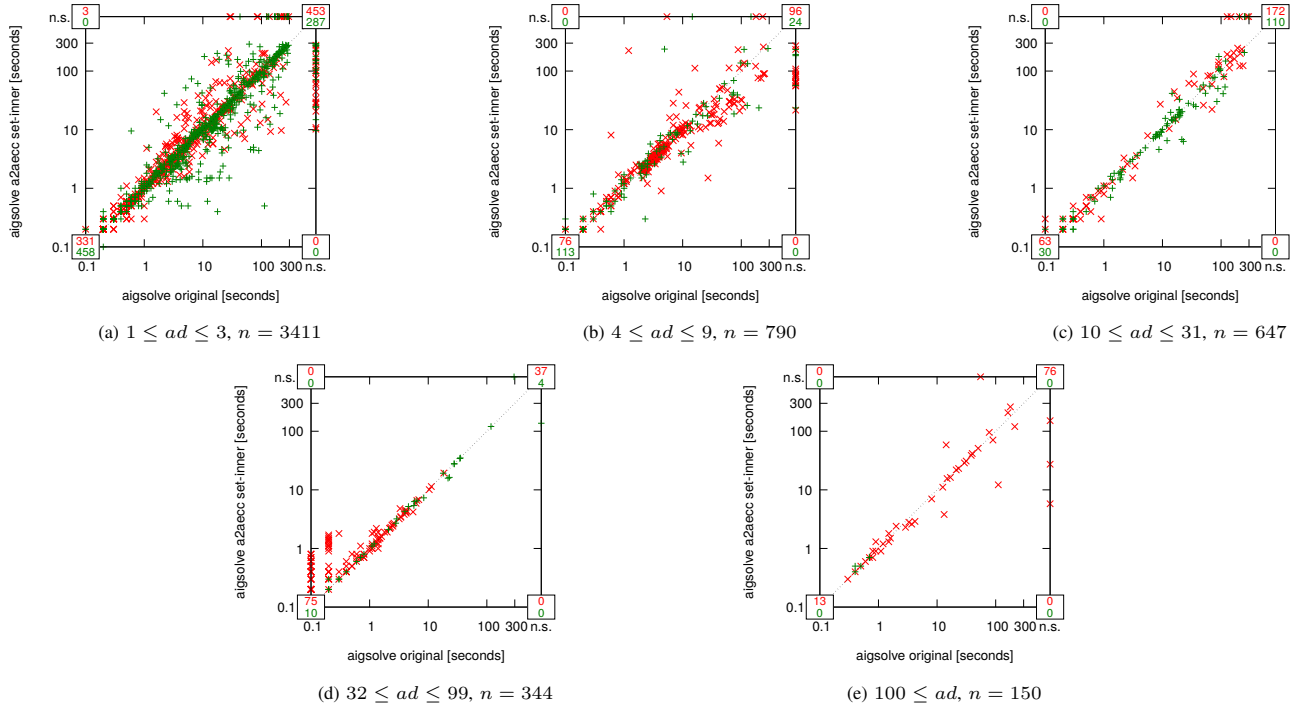


Fig. 1749: Solver AIGSolve: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by alternation depth (run time in seconds).

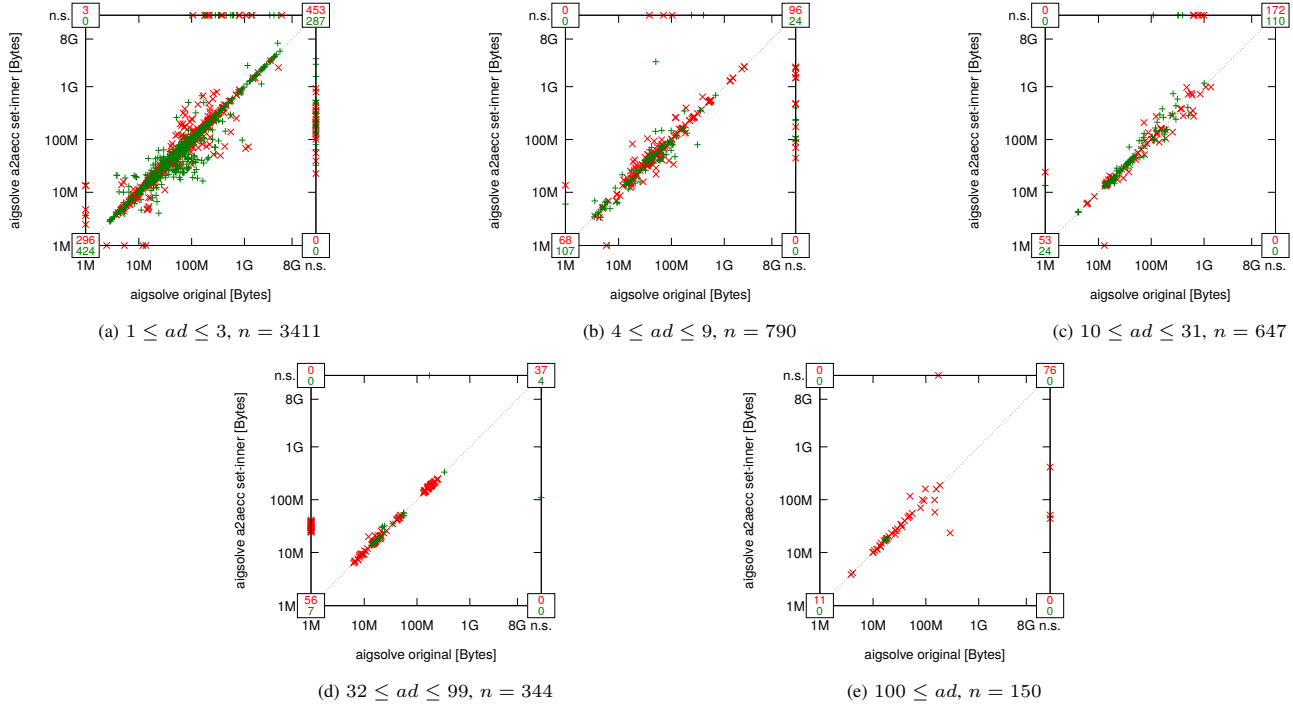


Fig. 1750: Solver AIGSolve: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by alternation depth (memory usage in Bytes).

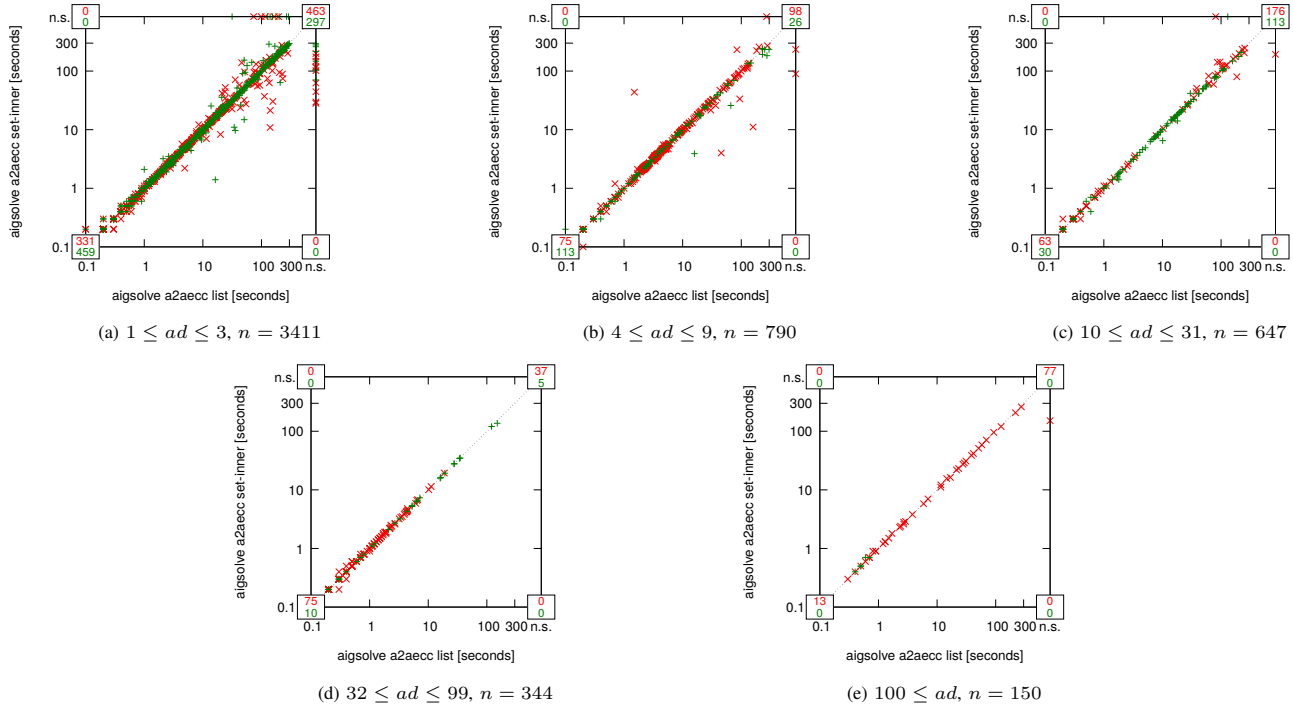


Fig. 1751: Solver AIGSolve: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by alternation depth (run time in seconds).

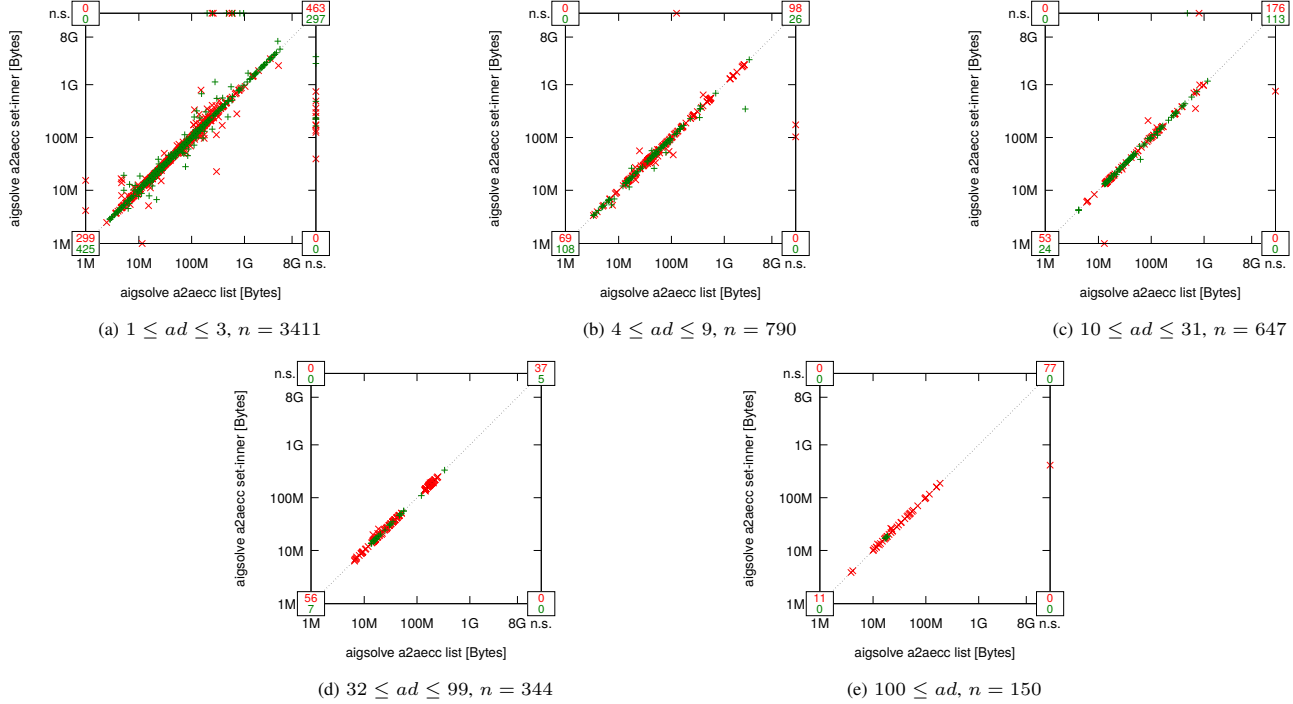


Fig. 1752: Solver AIGSolve: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by alternation depth (memory usage in Bytes).

c) CAQE:

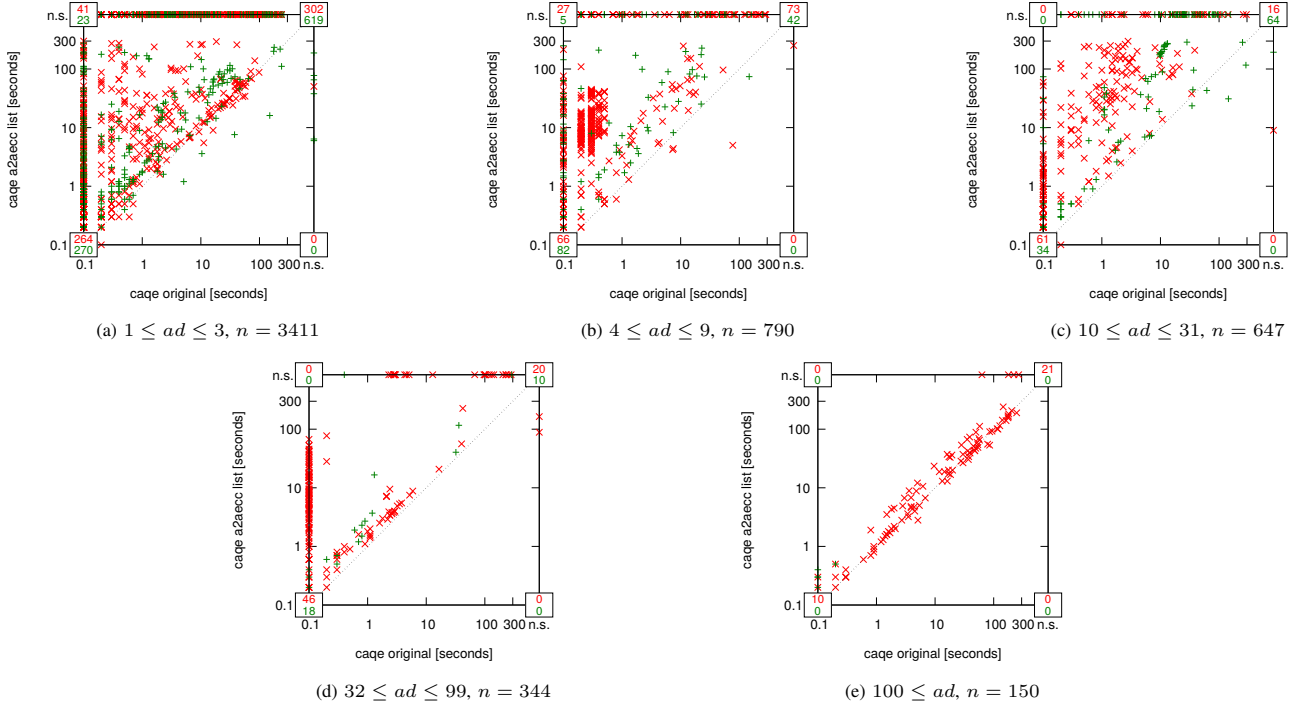


Fig. 1753: Solver CAQE: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by alternation depth (run time in seconds).

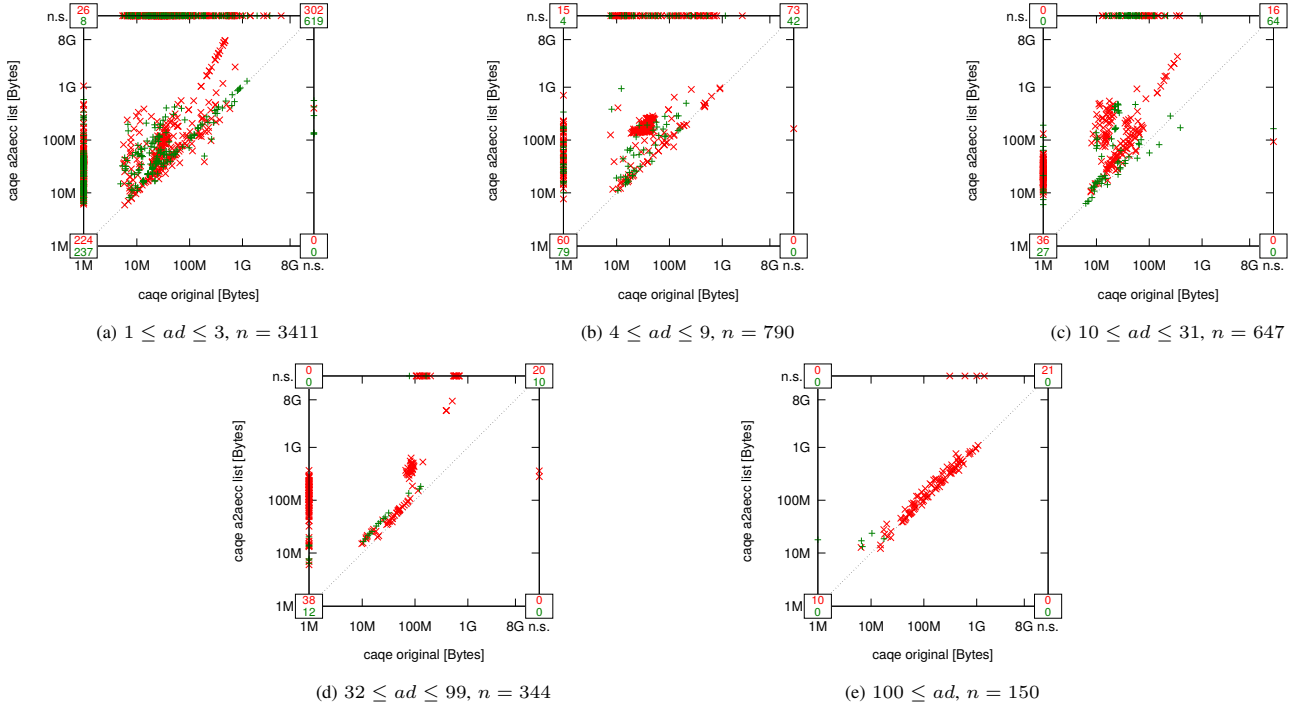


Fig. 1754: Solver CAQE: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by alternation depth (memory usage in Bytes).

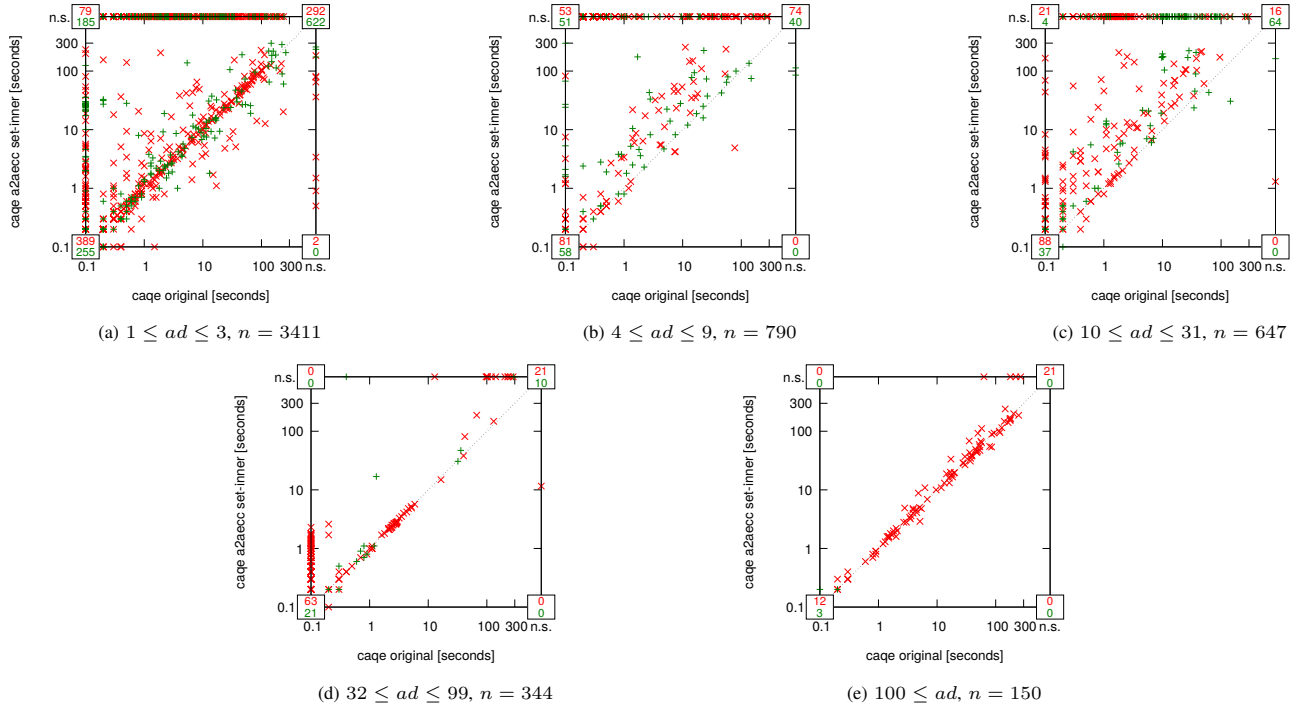


Fig. 1755: Solver CAQE: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by alternation depth (run time in seconds).

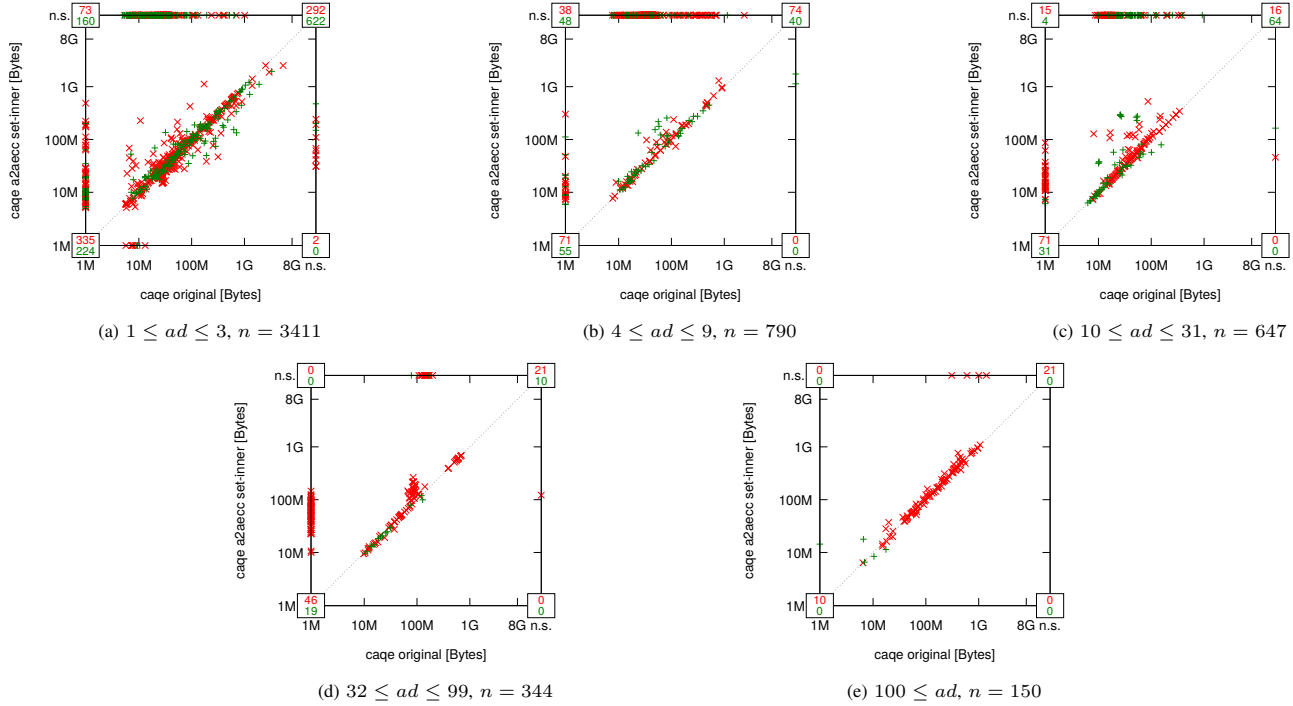


Fig. 1756: Solver CAQE: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by alternation depth (memory usage in Bytes).

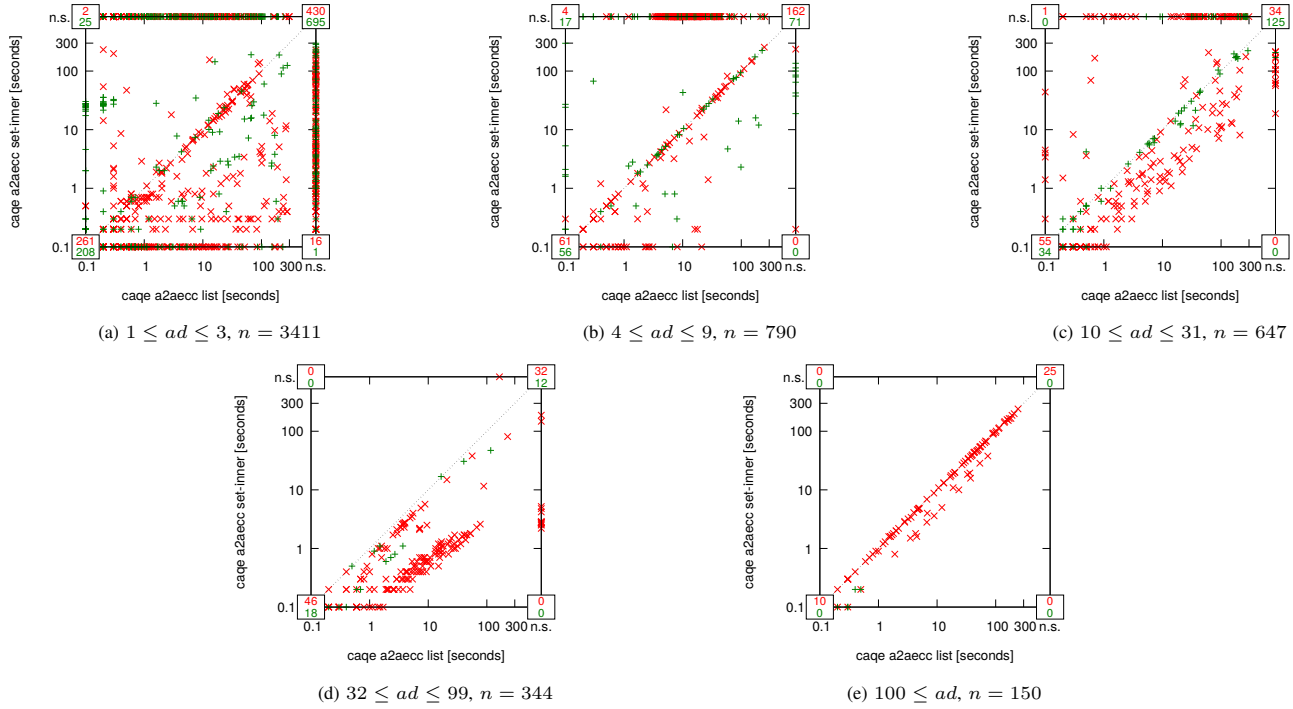


Fig. 1757: Solver CAQE: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by alternation depth (run time in seconds).

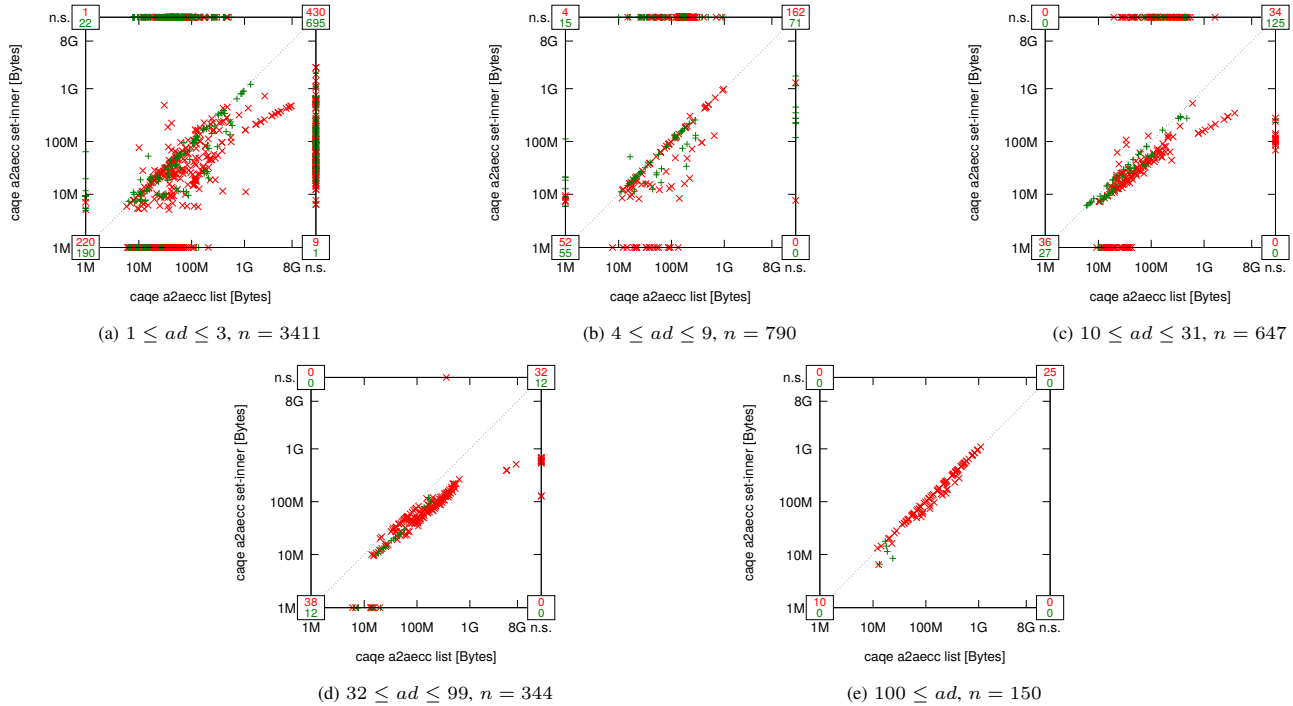


Fig. 1758: Solver CAQE: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by alternation depth (memory usage in Bytes).

d) GhostQ:

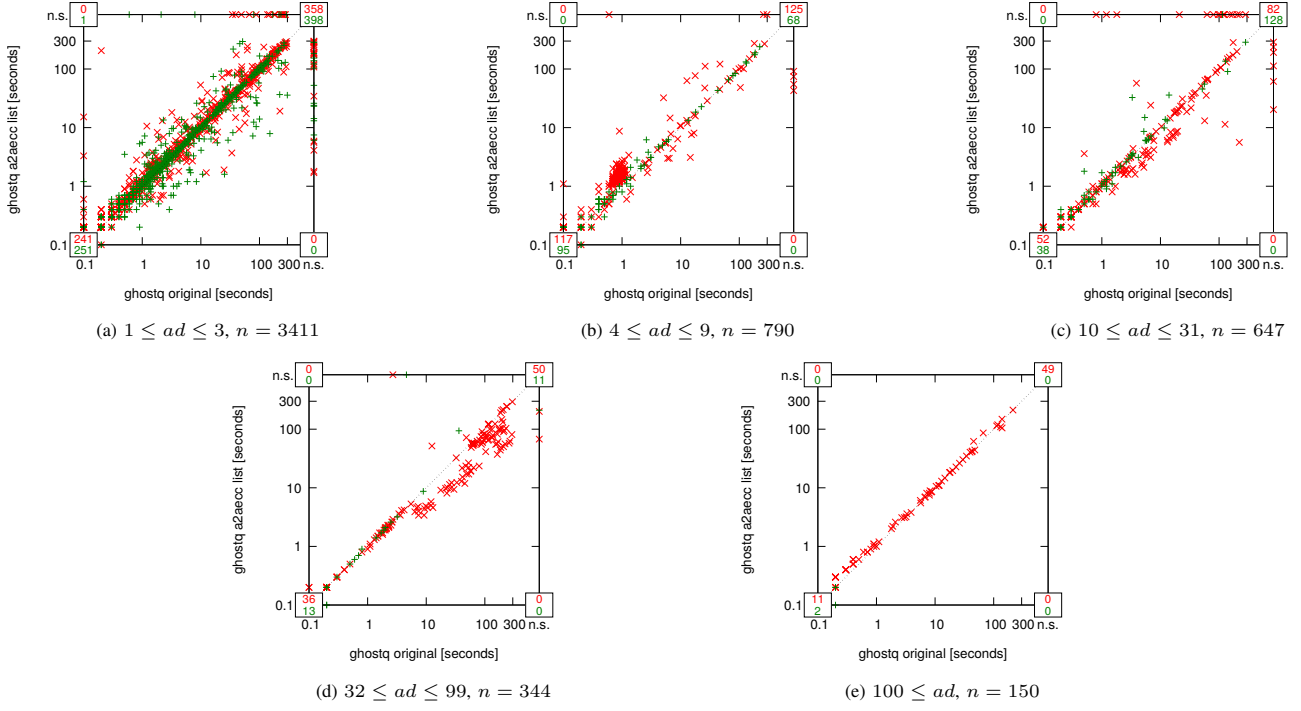


Fig. 1759: Solver GhostQ: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by alternation depth (run time in seconds).

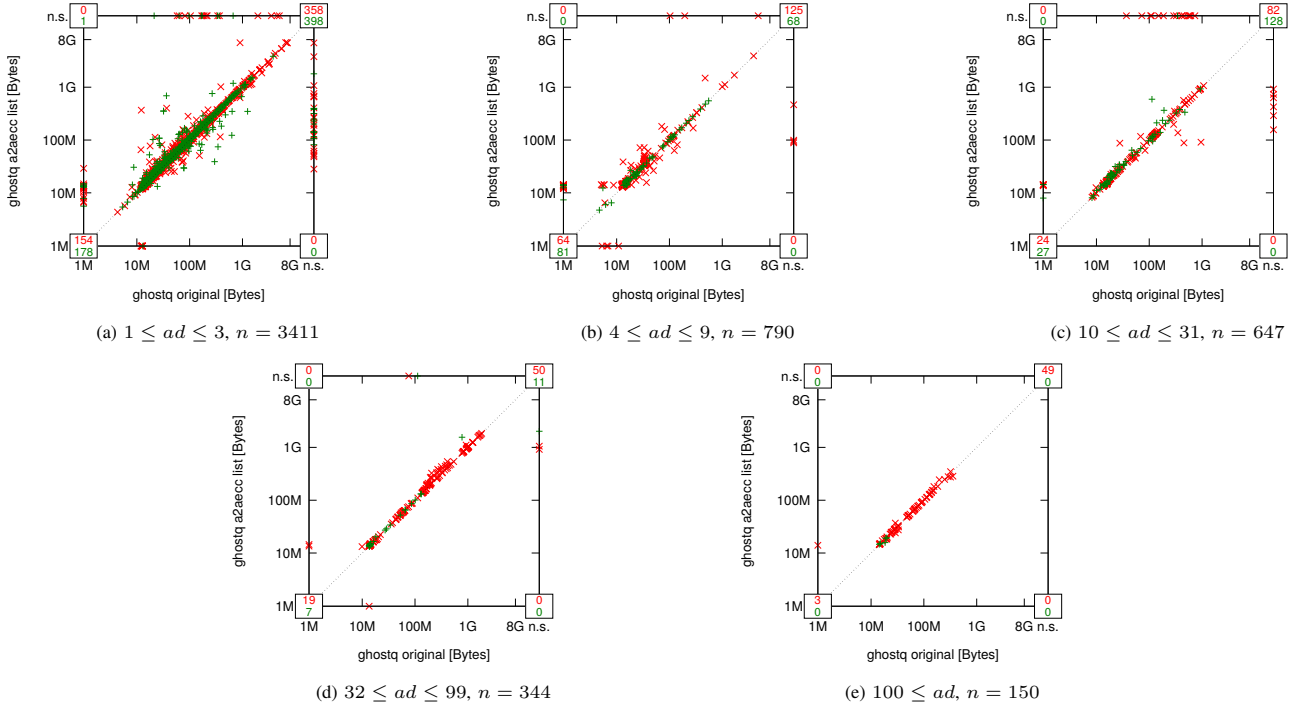


Fig. 1760: Solver GhostQ: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by alternation depth (memory usage in Bytes).

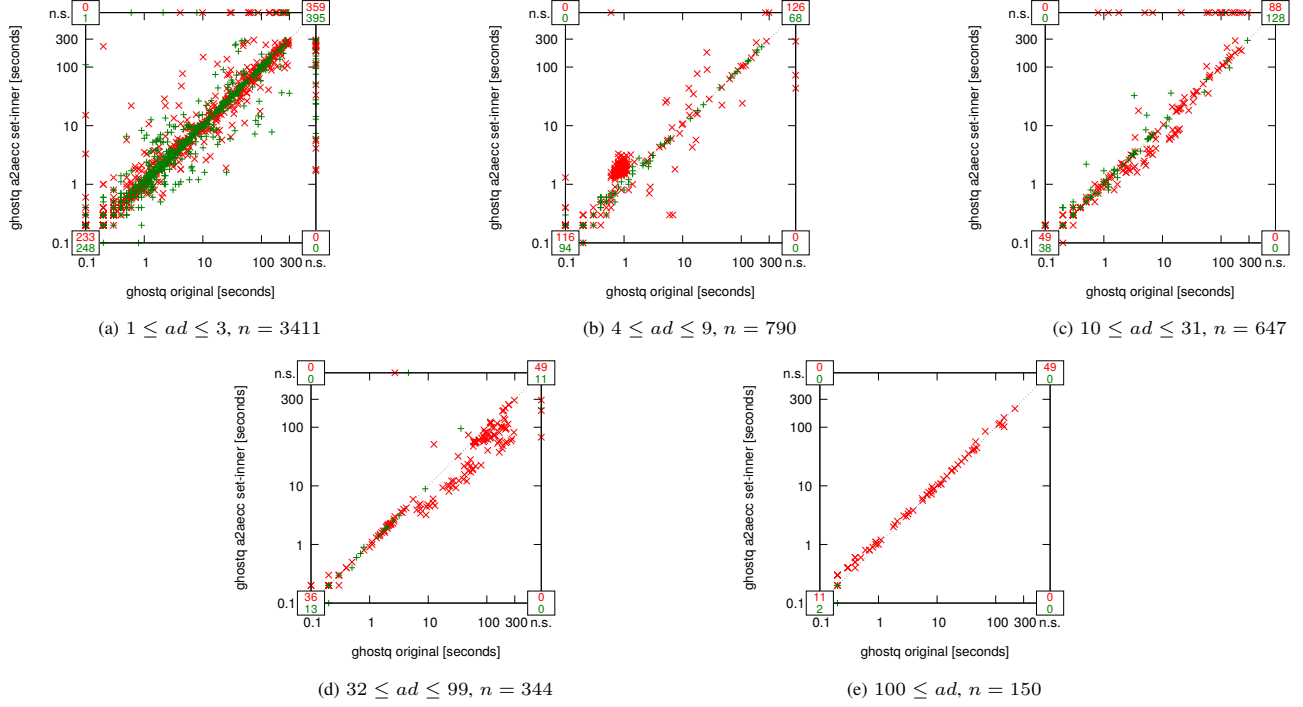


Fig. 1761: Solver `GhostQ`: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by alternation depth (run time in seconds).

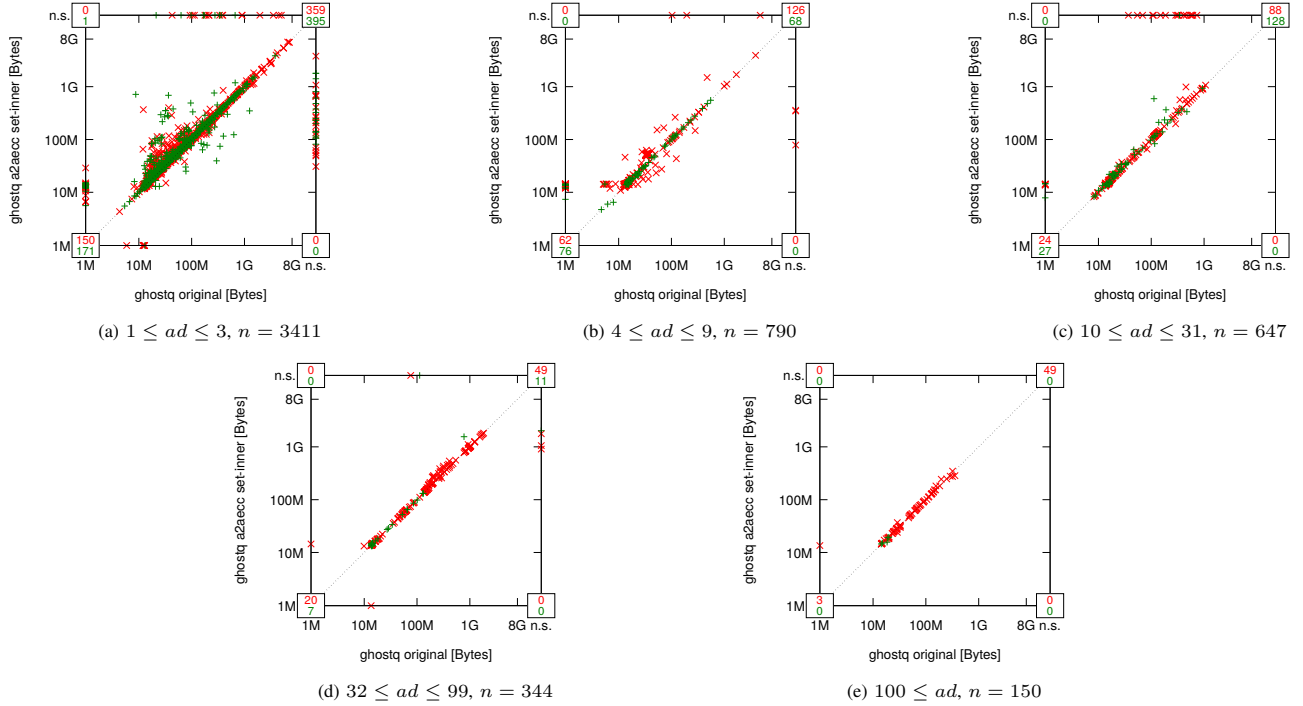


Fig. 1762: Solver `GhostQ`: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by alternation depth (memory usage in Bytes).

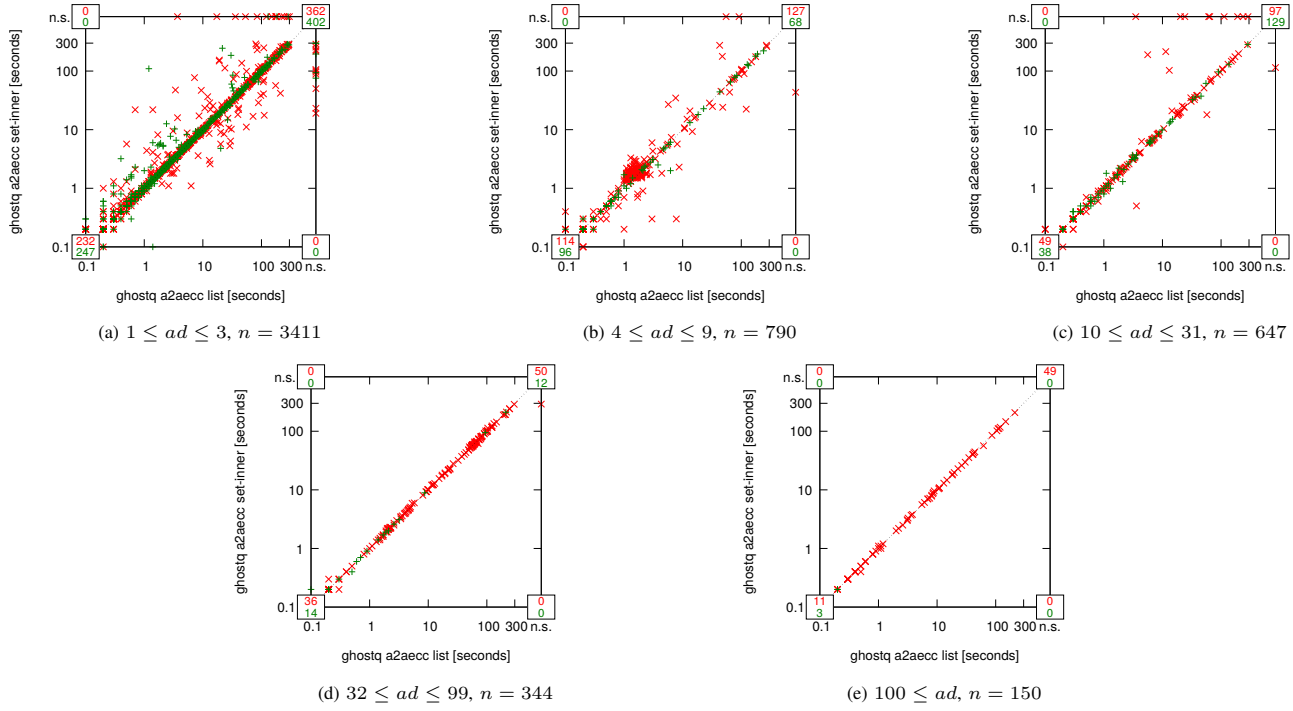


Fig. 1763: Solver *GhostQ*: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by alternation depth (run time in seconds).

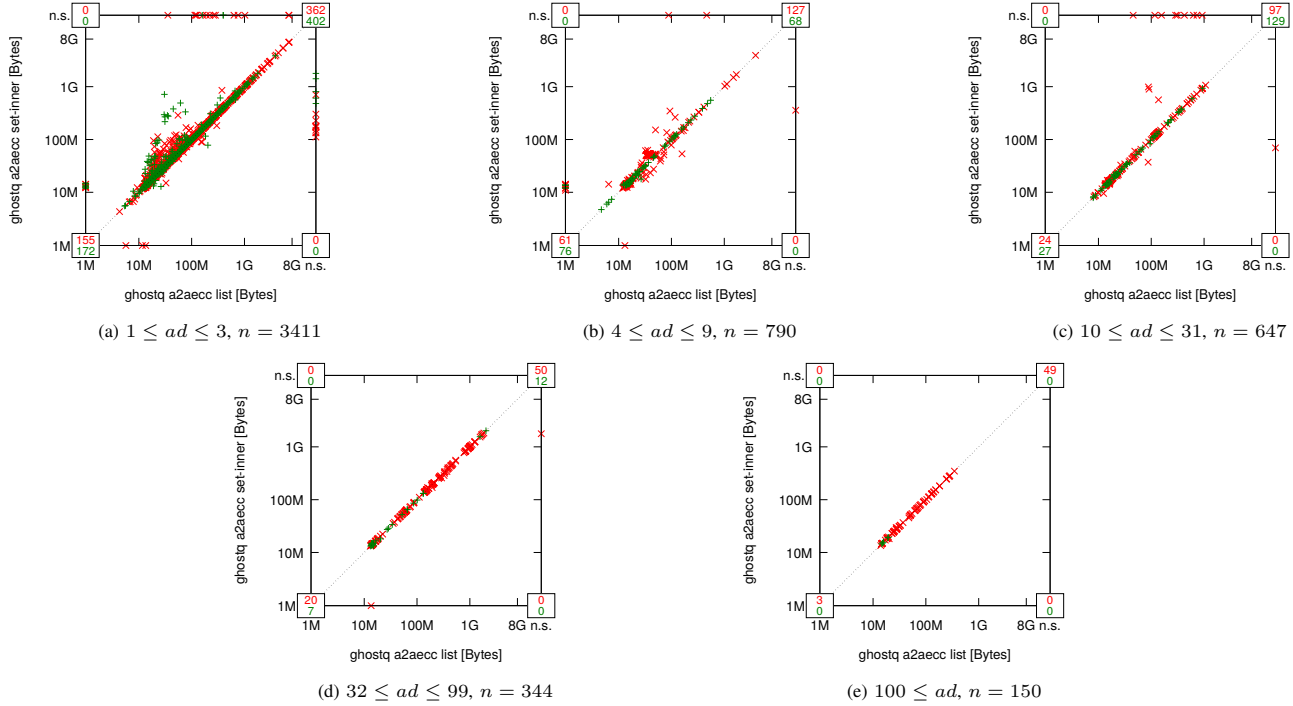


Fig. 1764: Solver *GhostQ*: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by alternation depth (memory usage in Bytes).

e) QESTO:

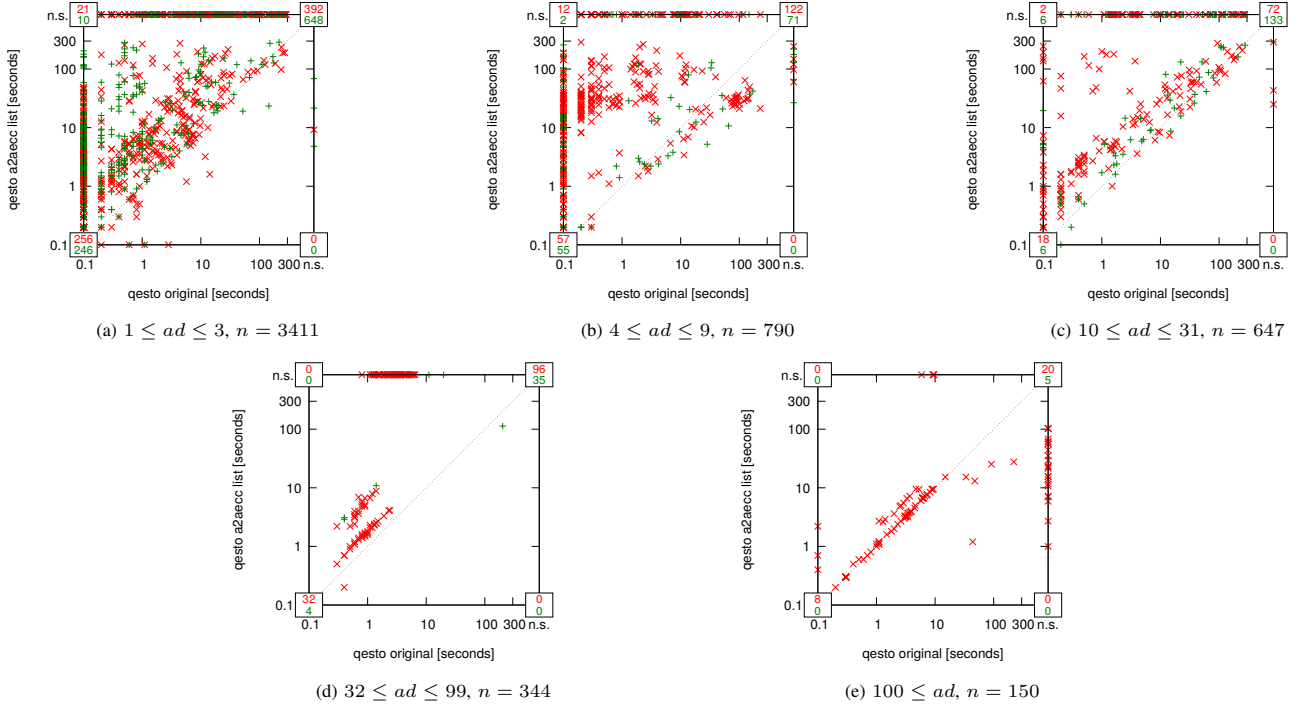


Fig. 1765: Solver QESTO: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by alternation depth (run time in seconds).

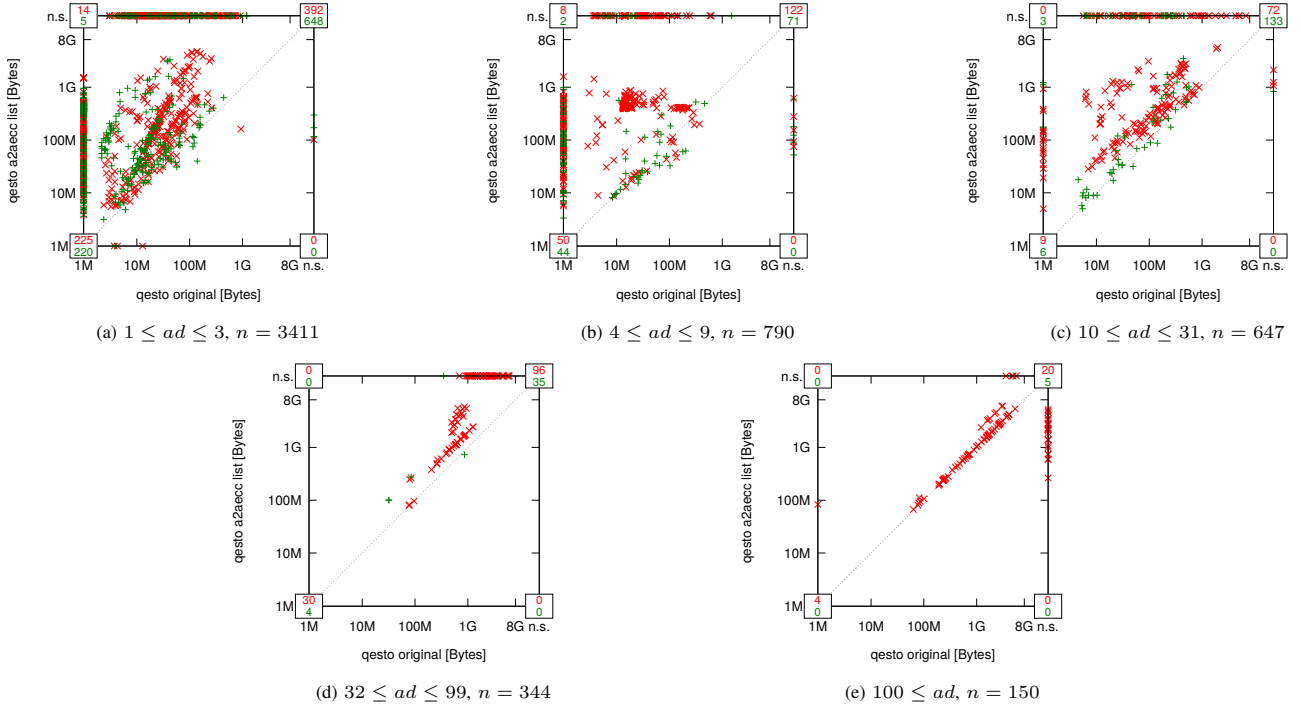


Fig. 1766: Solver QESTO: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by alternation depth (memory usage in Bytes).

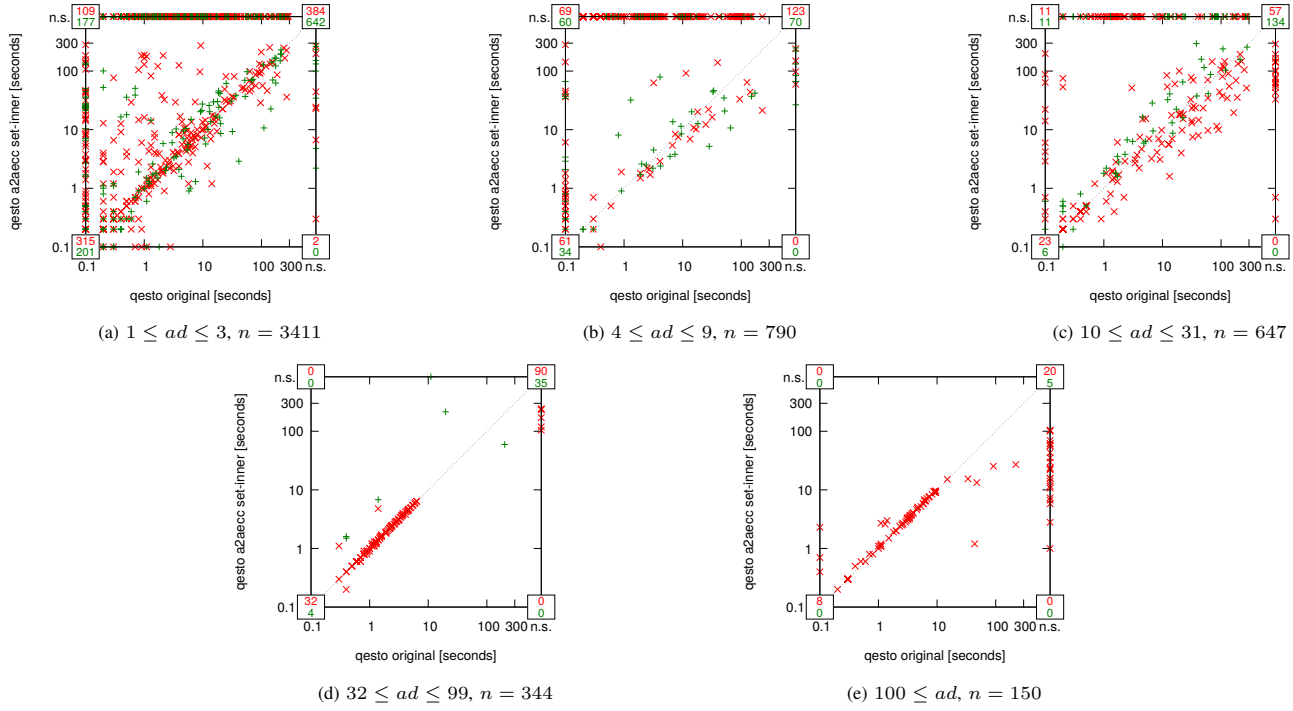


Fig. 1767: Solver QESTO: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by alternation depth (run time in seconds).

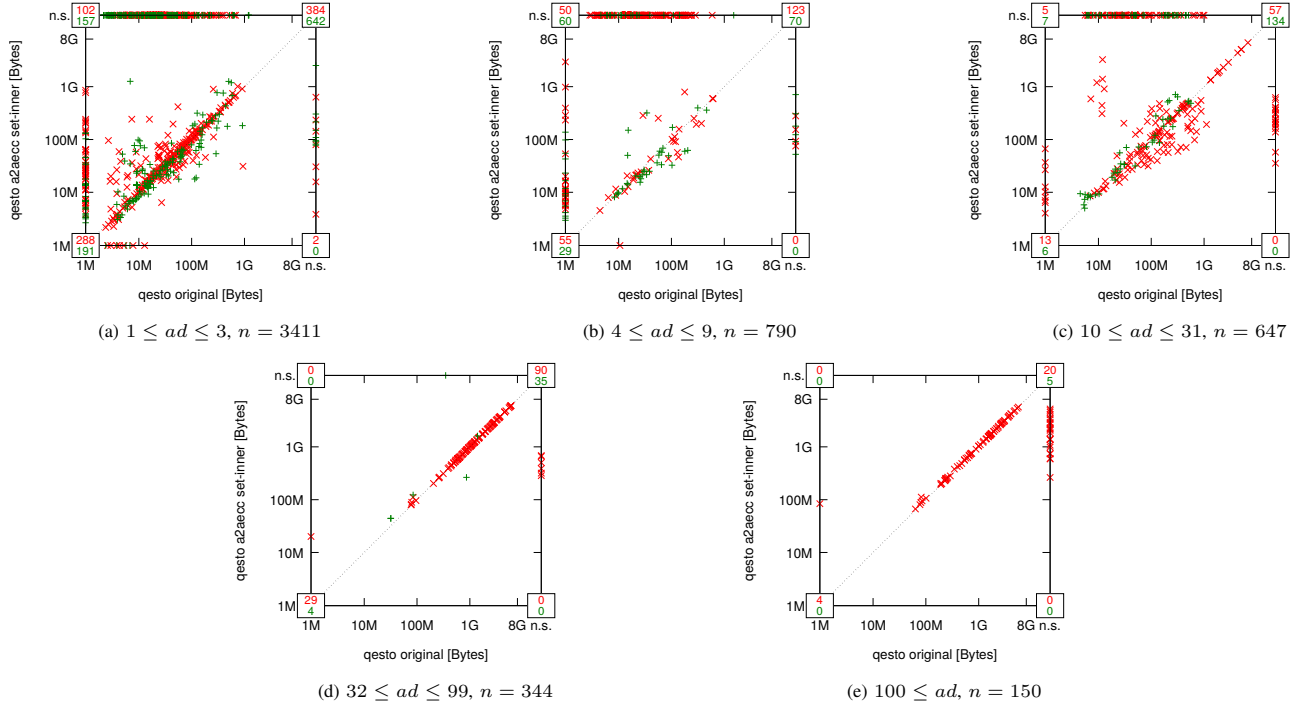


Fig. 1768: Solver QESTO: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by alternation depth (memory usage in Bytes).

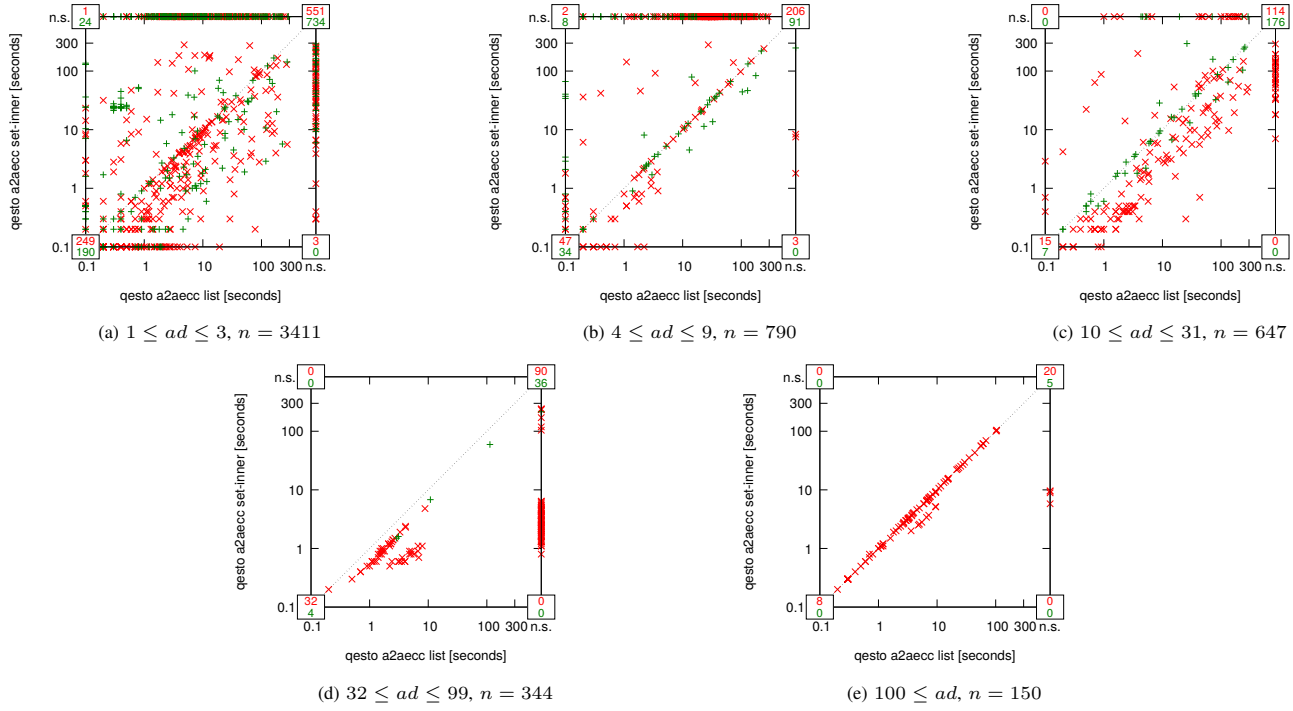


Fig. 1769: Solver QESTO: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by alternation depth (run time in seconds).

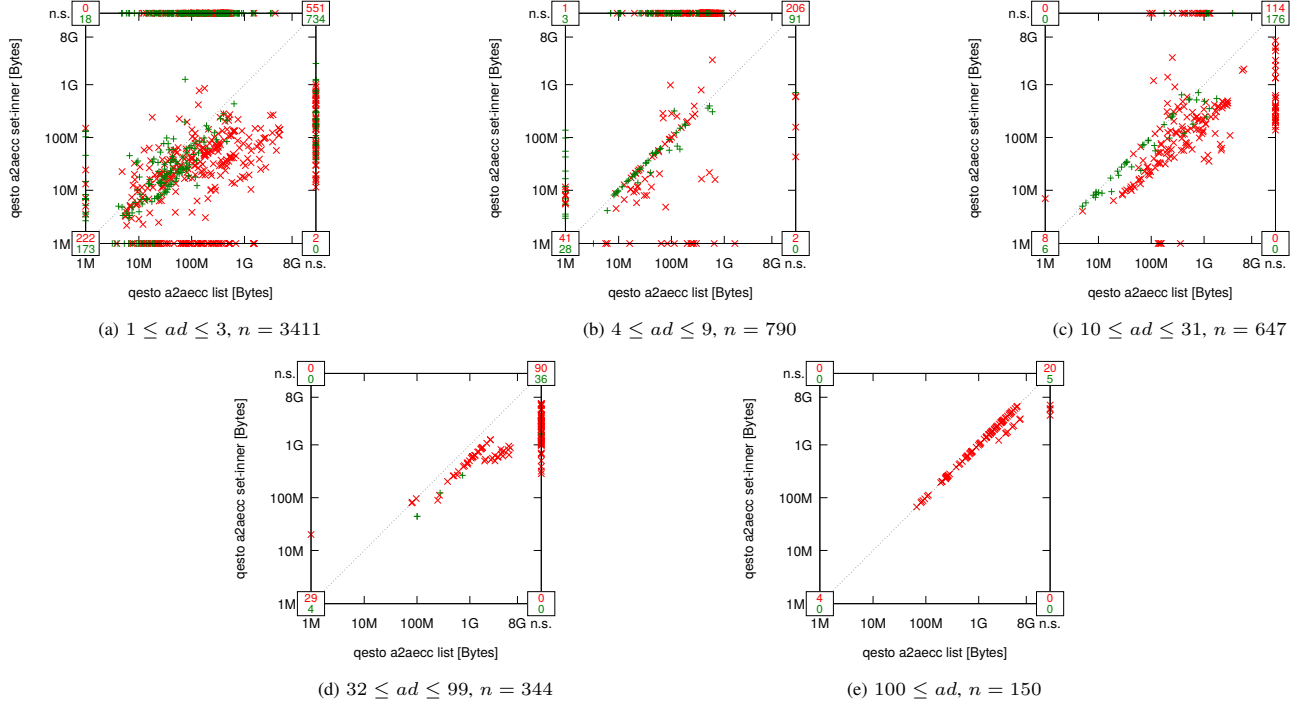


Fig. 1770: Solver QESTO: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by alternation depth (memory usage in Bytes).

f) RAReQS:

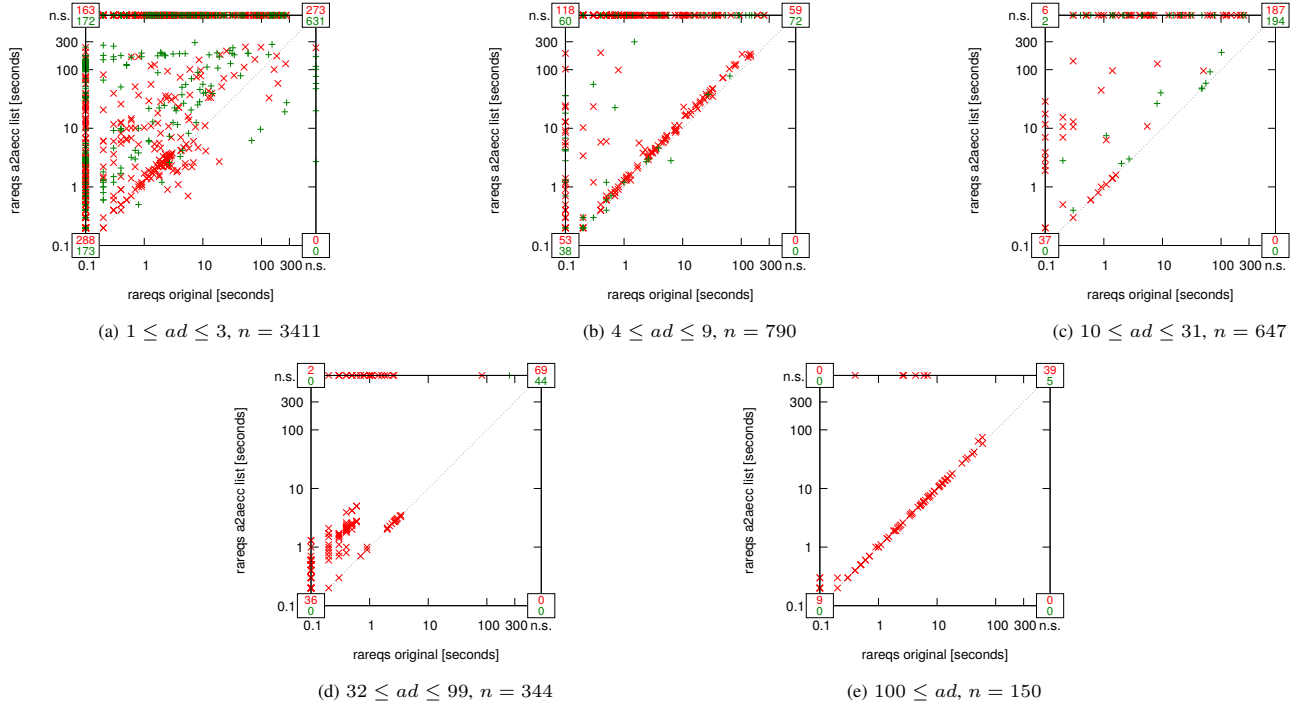


Fig. 1771: Solver RAReQS: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by alternation depth (run time in seconds).

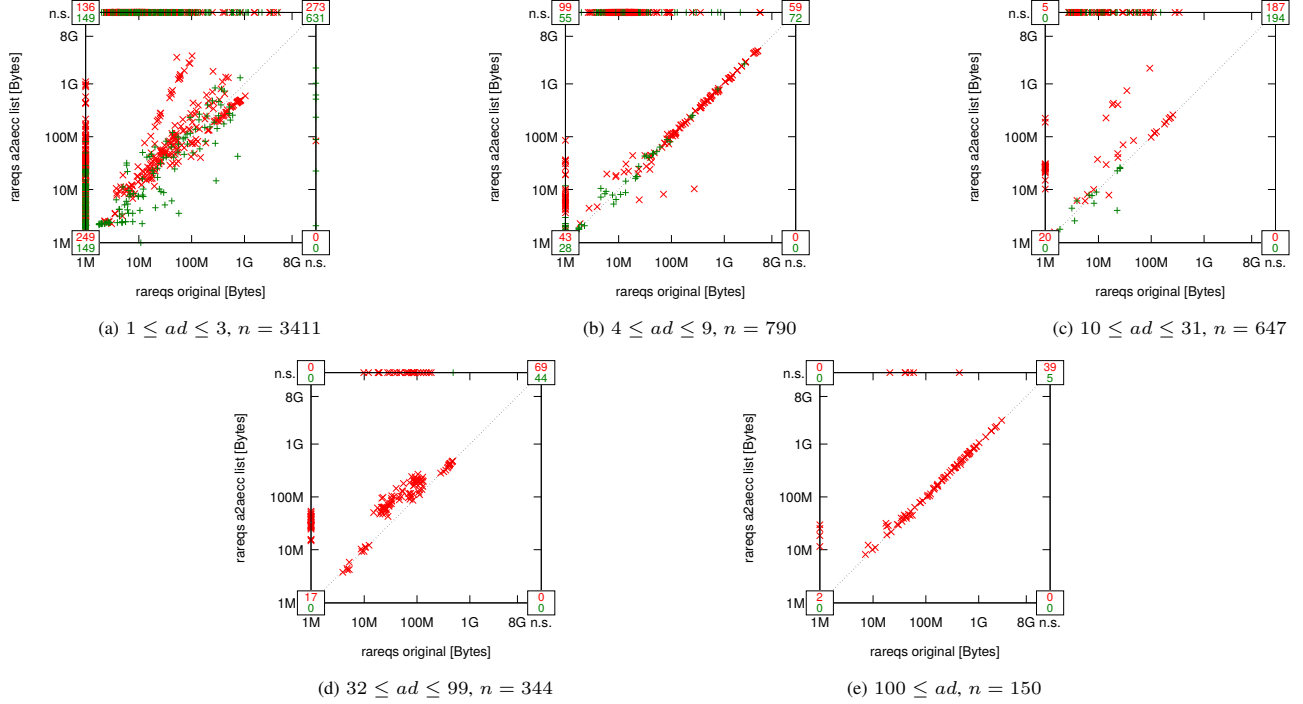


Fig. 1772: Solver RAReQS: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by alternation depth (memory usage in Bytes).

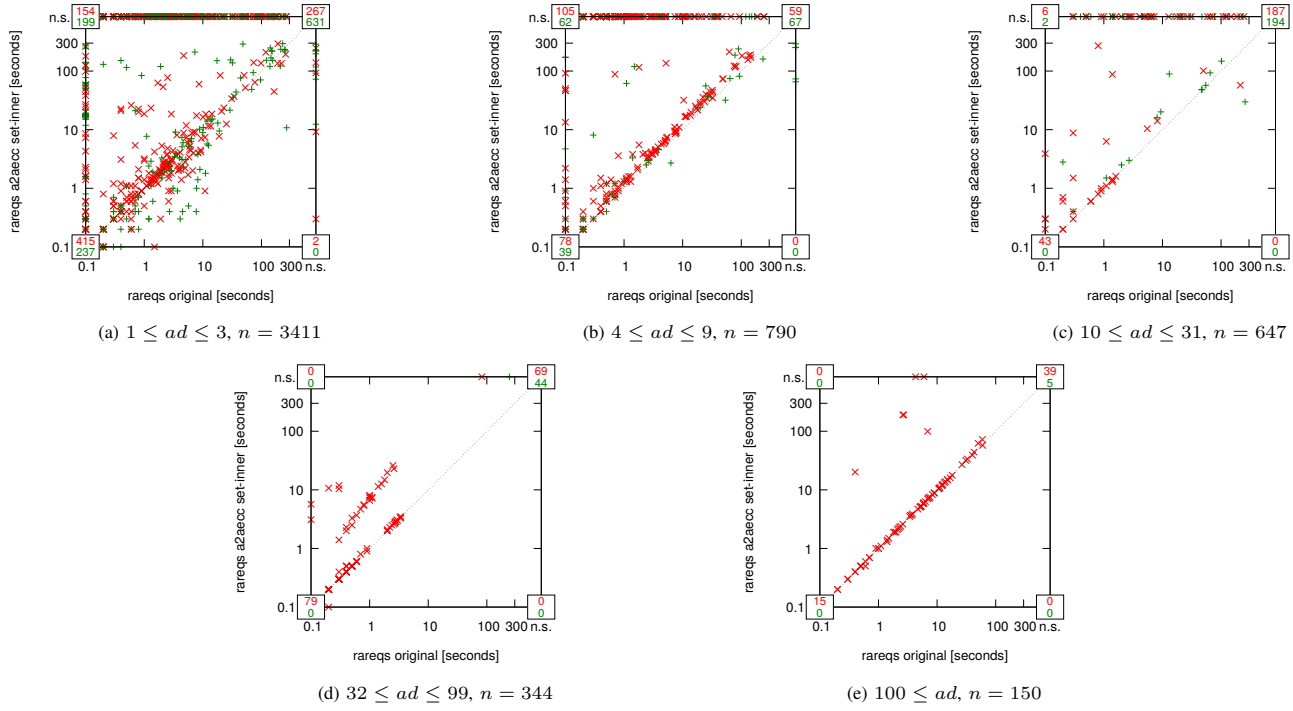


Fig. 1773: Solver RAREQS: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by alternation depth (run time in seconds).

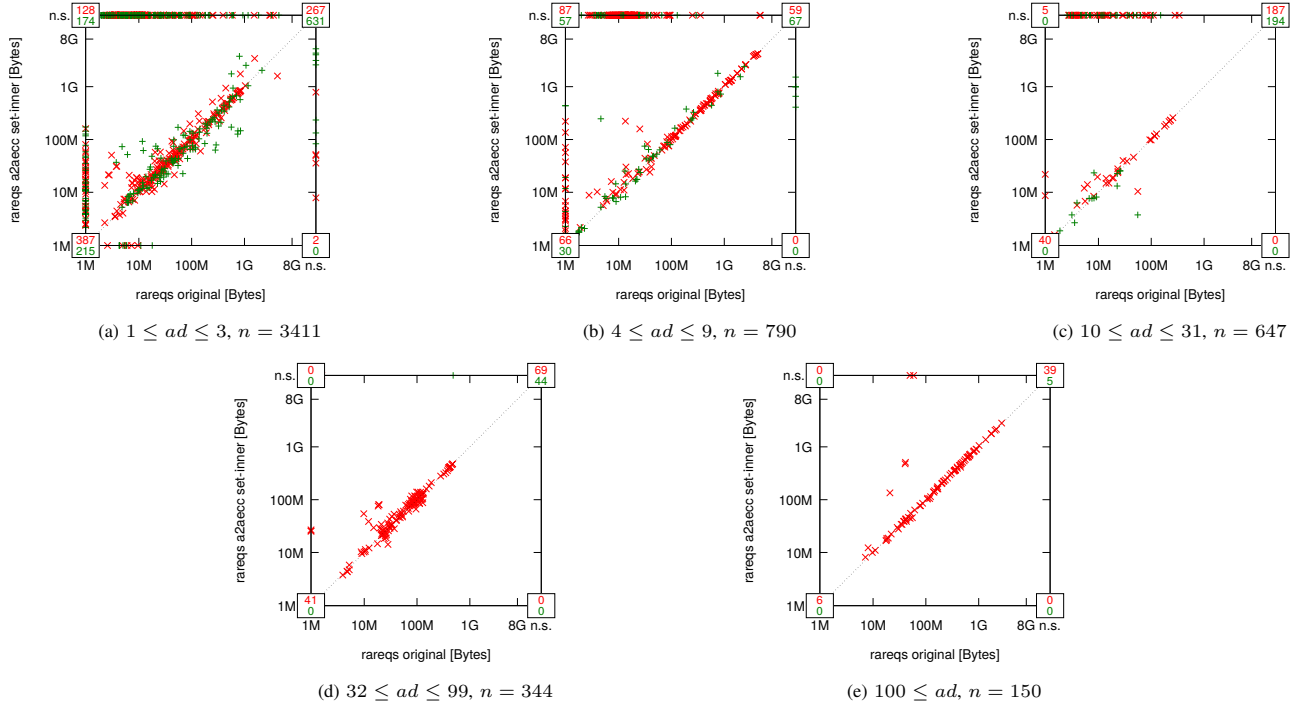


Fig. 1774: Solver RAREQS: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by alternation depth (memory usage in Bytes).

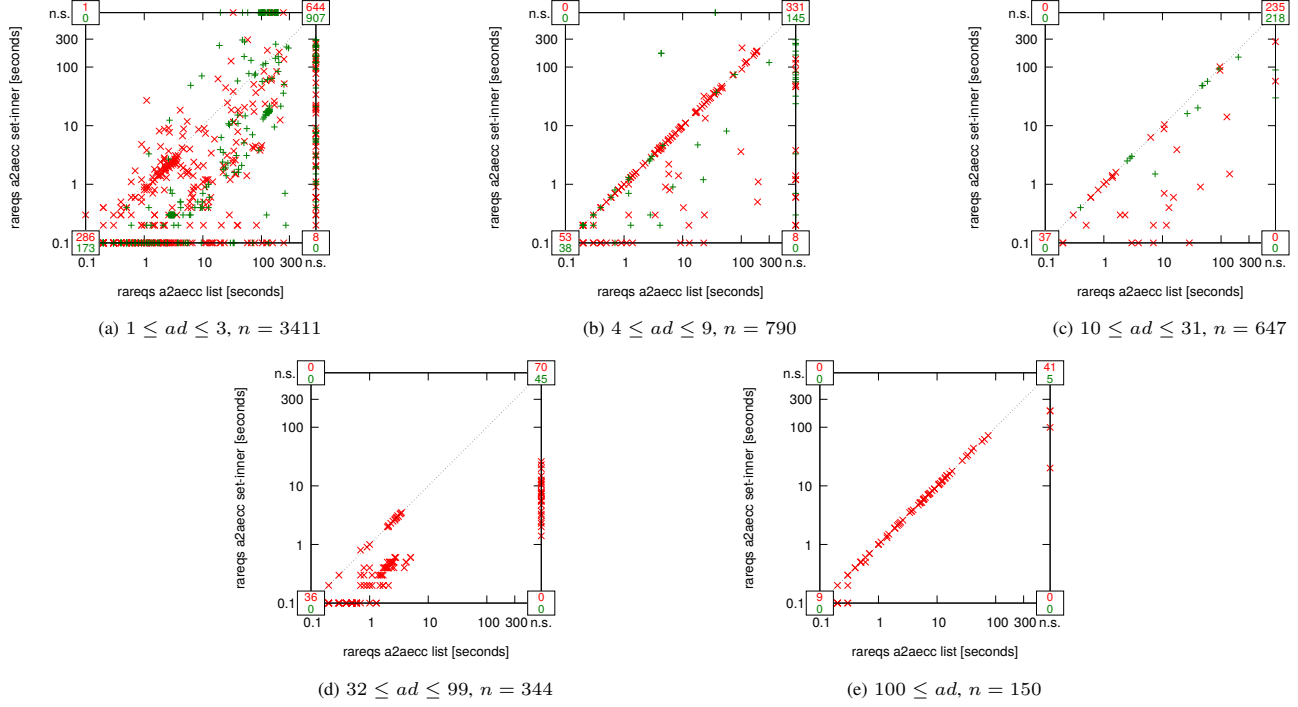


Fig. 1775: Solver RAReQS: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by alternation depth (run time in seconds).

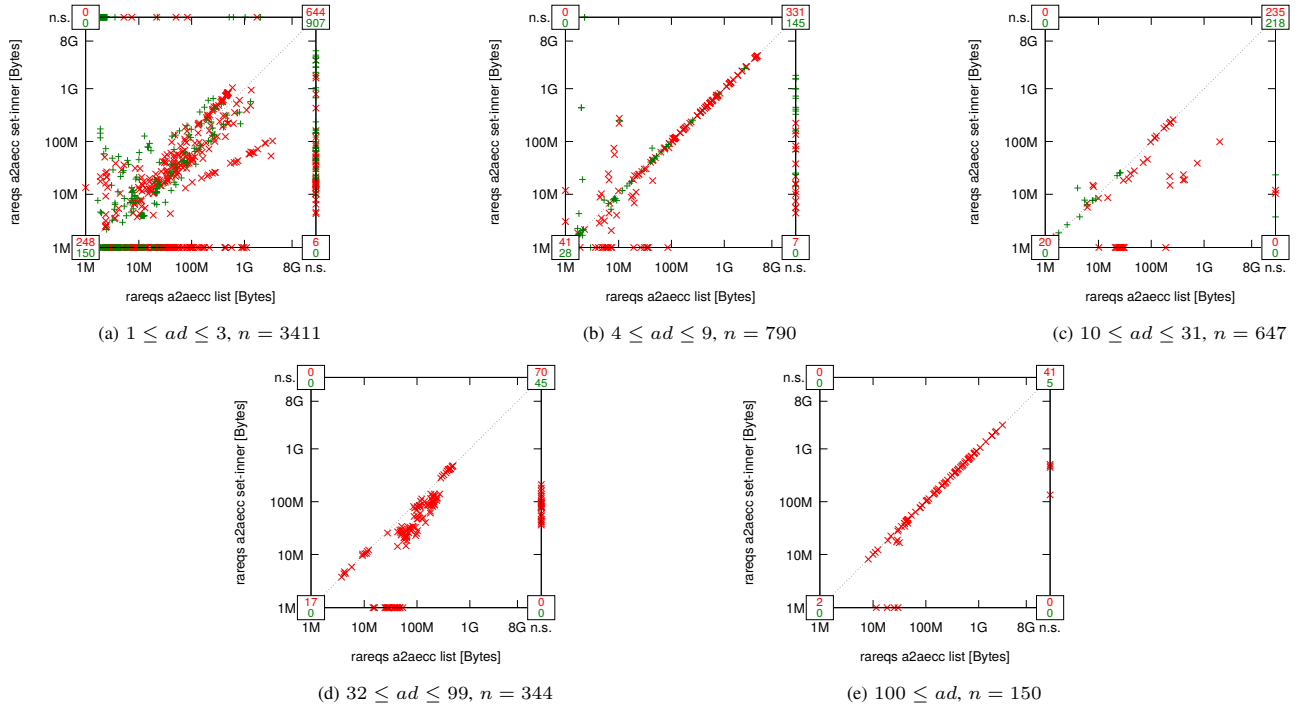


Fig. 1776: Solver RAReQS: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by alternation depth (memory usage in Bytes).

D. Partitioned by Number of Clauses

In this subsection there are 6 figures for each solver with subfigures for partitions of the benchmarks according to their number of clauses.

- 1) Comparing run times for solving the transformed versus the original instances with list semantics.
- 2) Memory usage for 1.
- 3) As 1. but with set-inner semantics.
- 4) Memory usage for 3.
- 5) A direct comparison of solving the transformed instances with set-inner versus list semantics.
- 6) Memory usage for 5.

a) DepQBF:

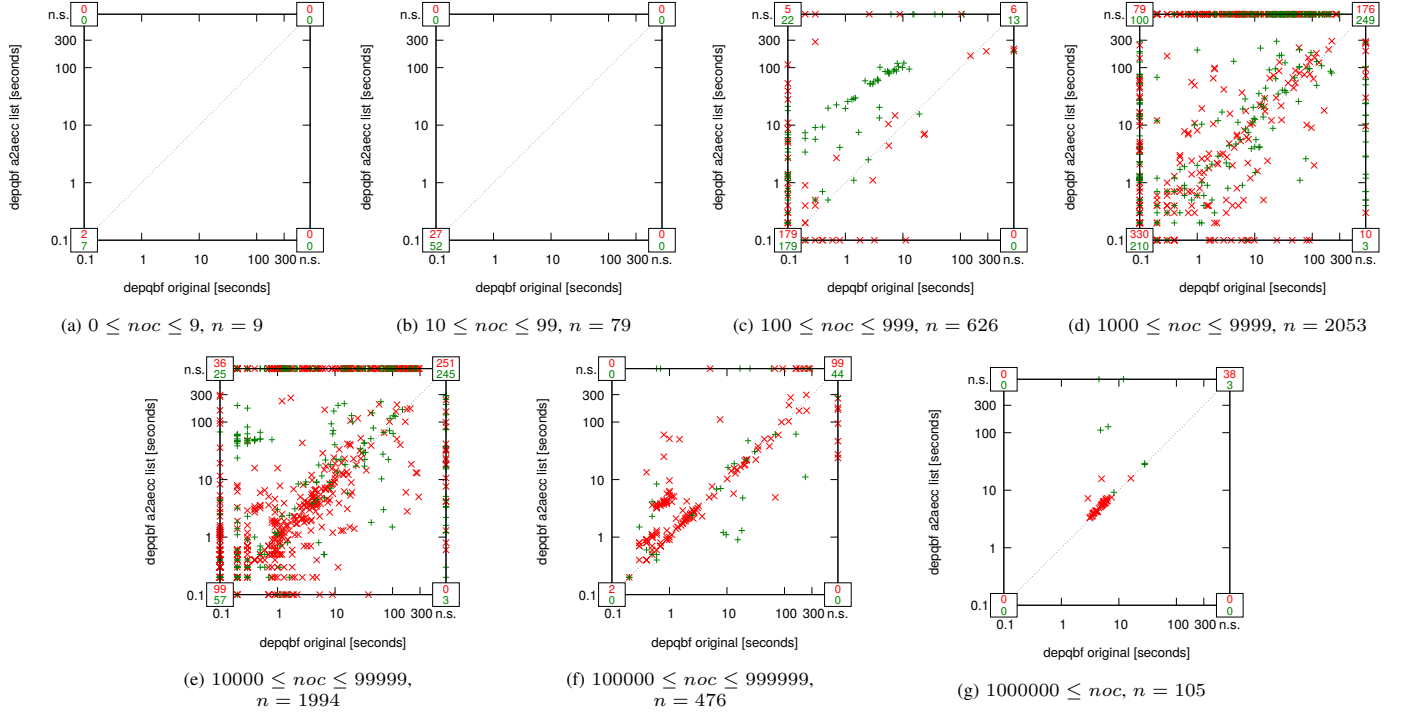


Fig. 1777: Solver DepQBF: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by number of clauses (run time in seconds).

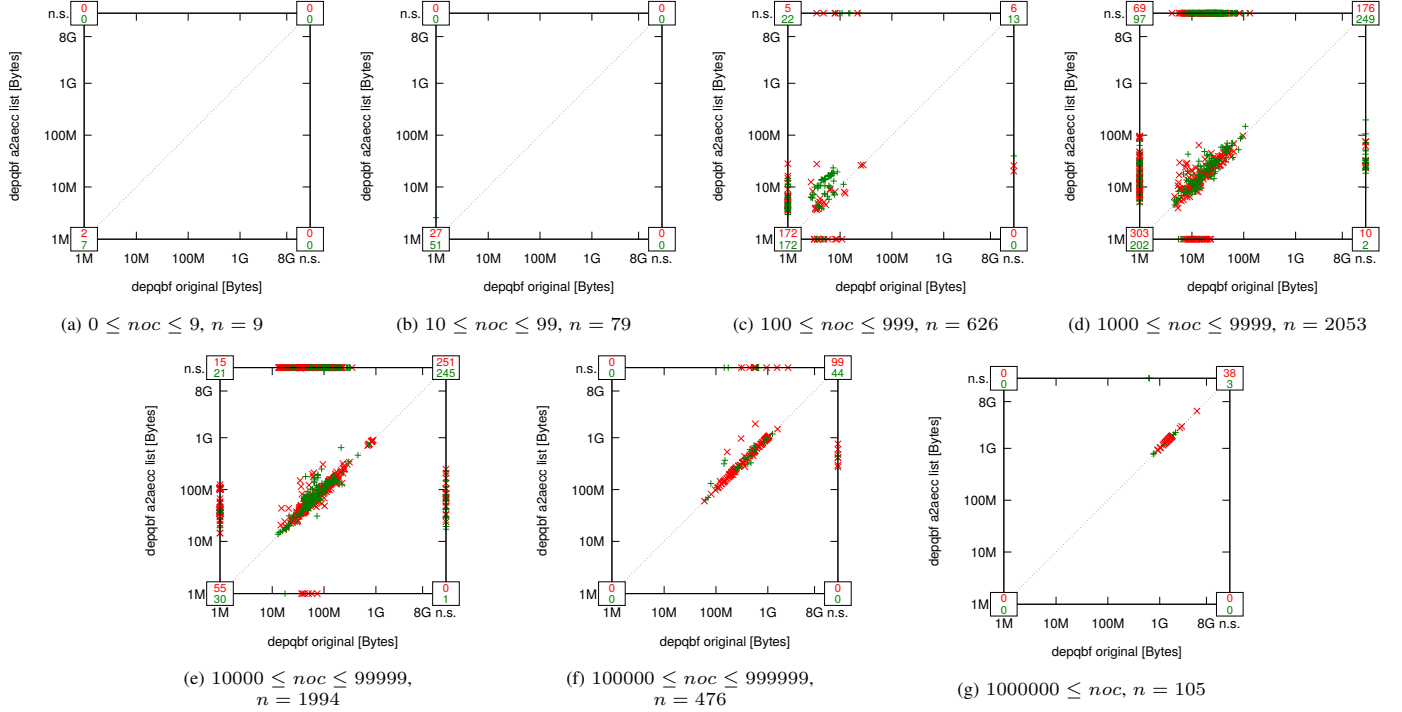


Fig. 1778: Solver DepQBF: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by number of clauses (memory usage in Bytes).

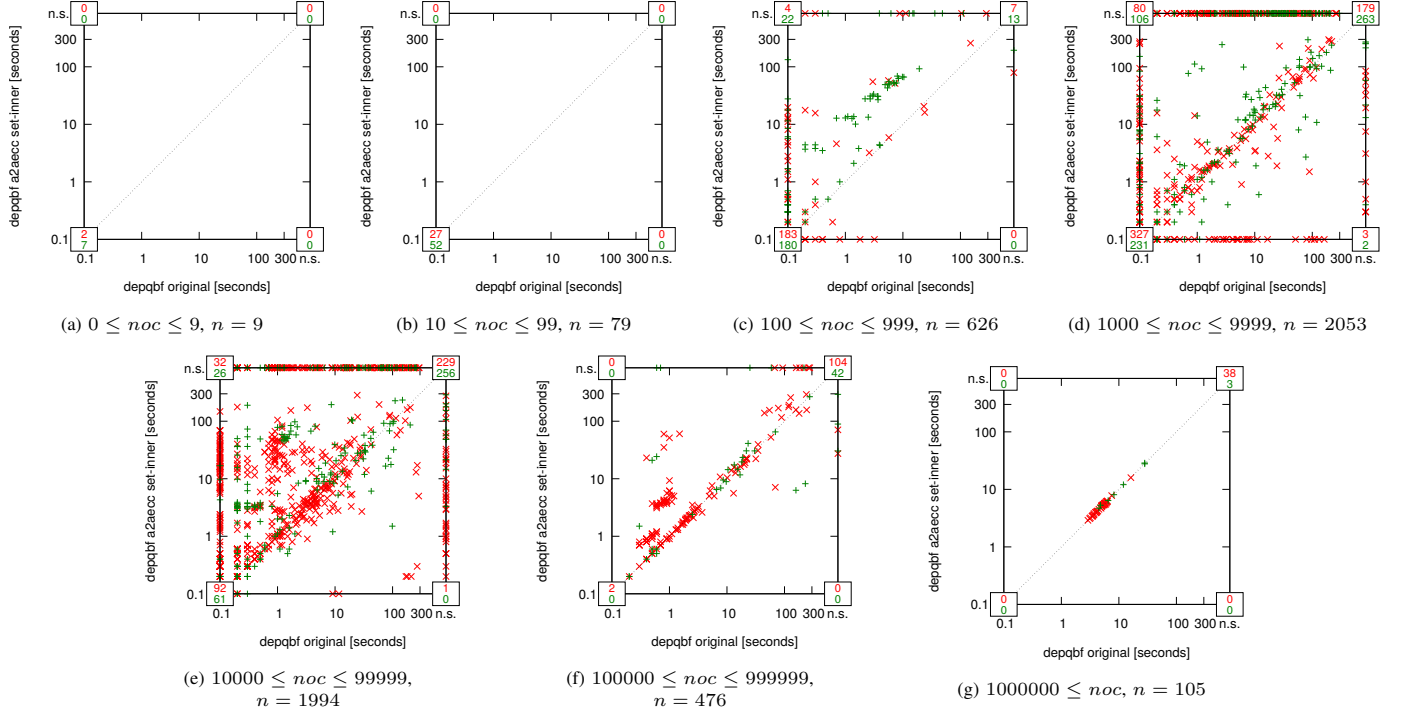


Fig. 1779: Solver DepQBF: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by number of clauses (run time in seconds).

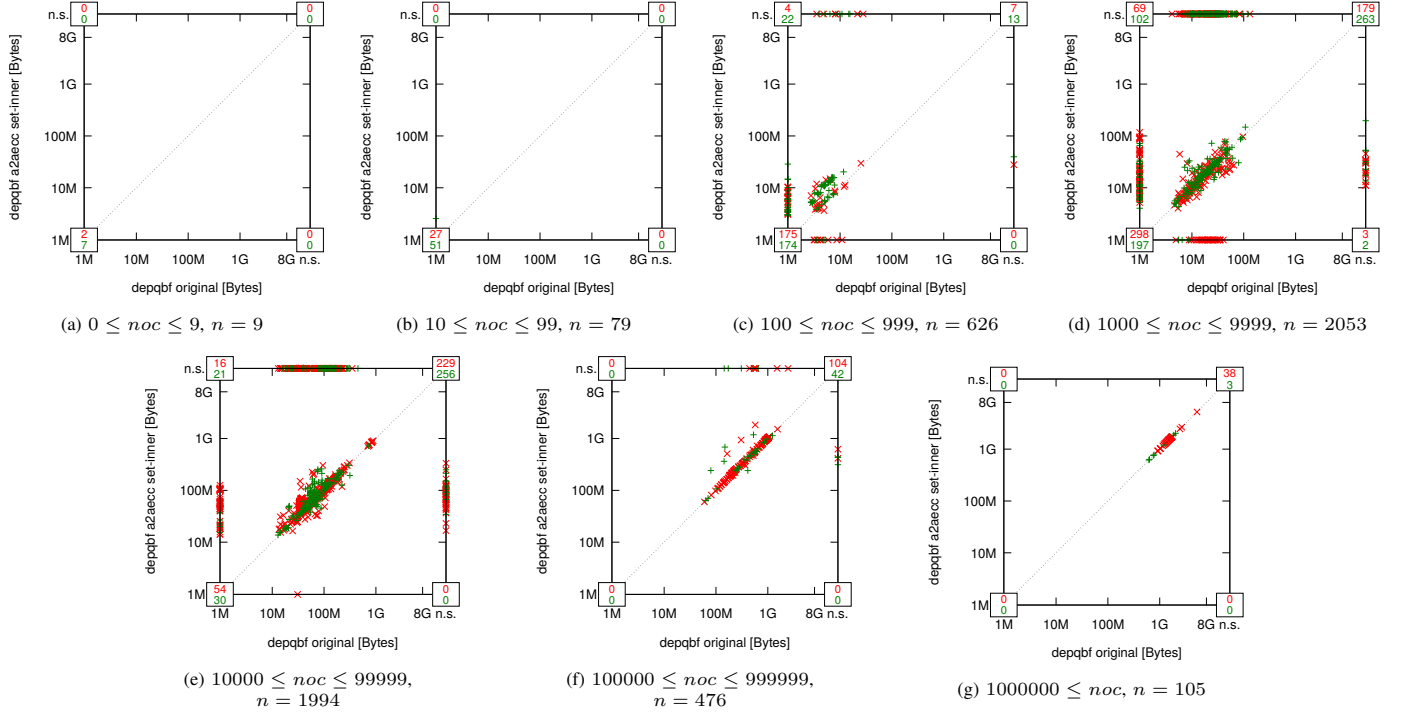


Fig. 1780: Solver DepQBF: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by number of clauses (memory usage in Bytes).

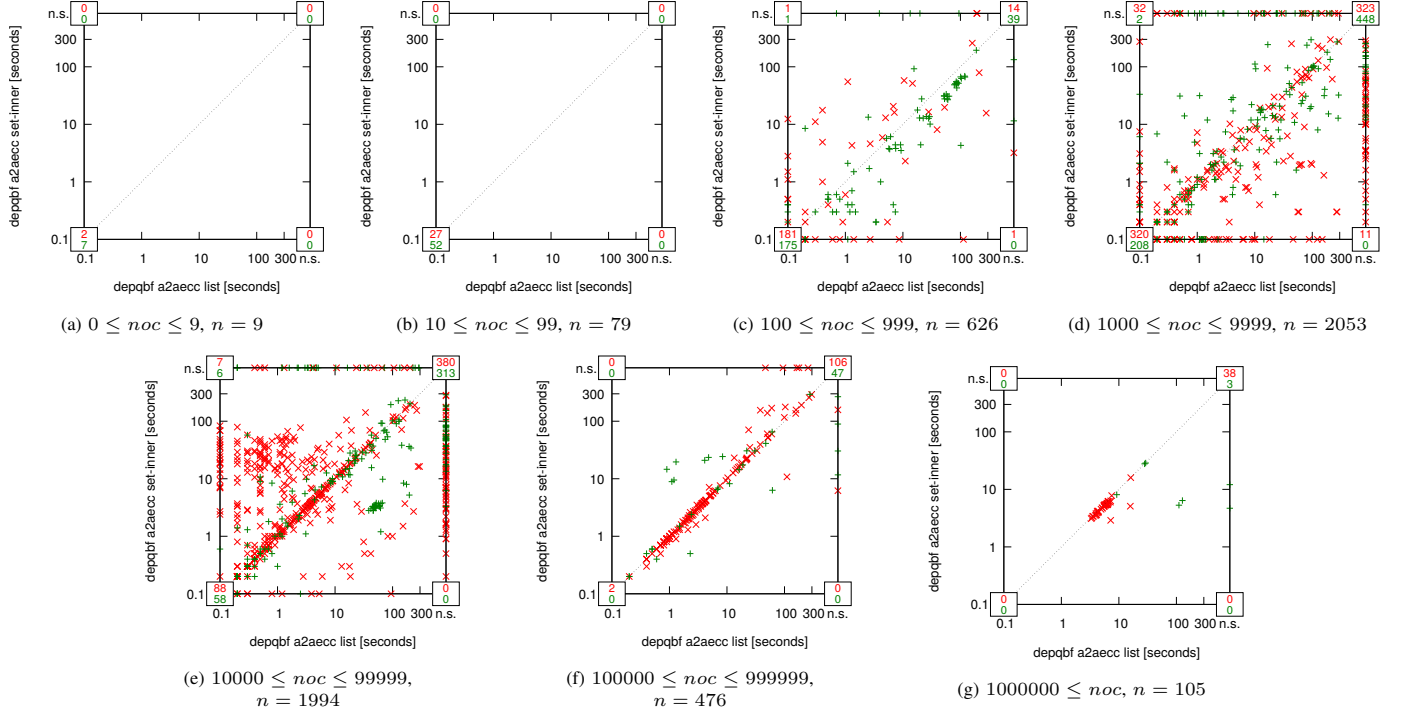


Fig. 1781: Solver DepQBF: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by number of clauses (run time in seconds).

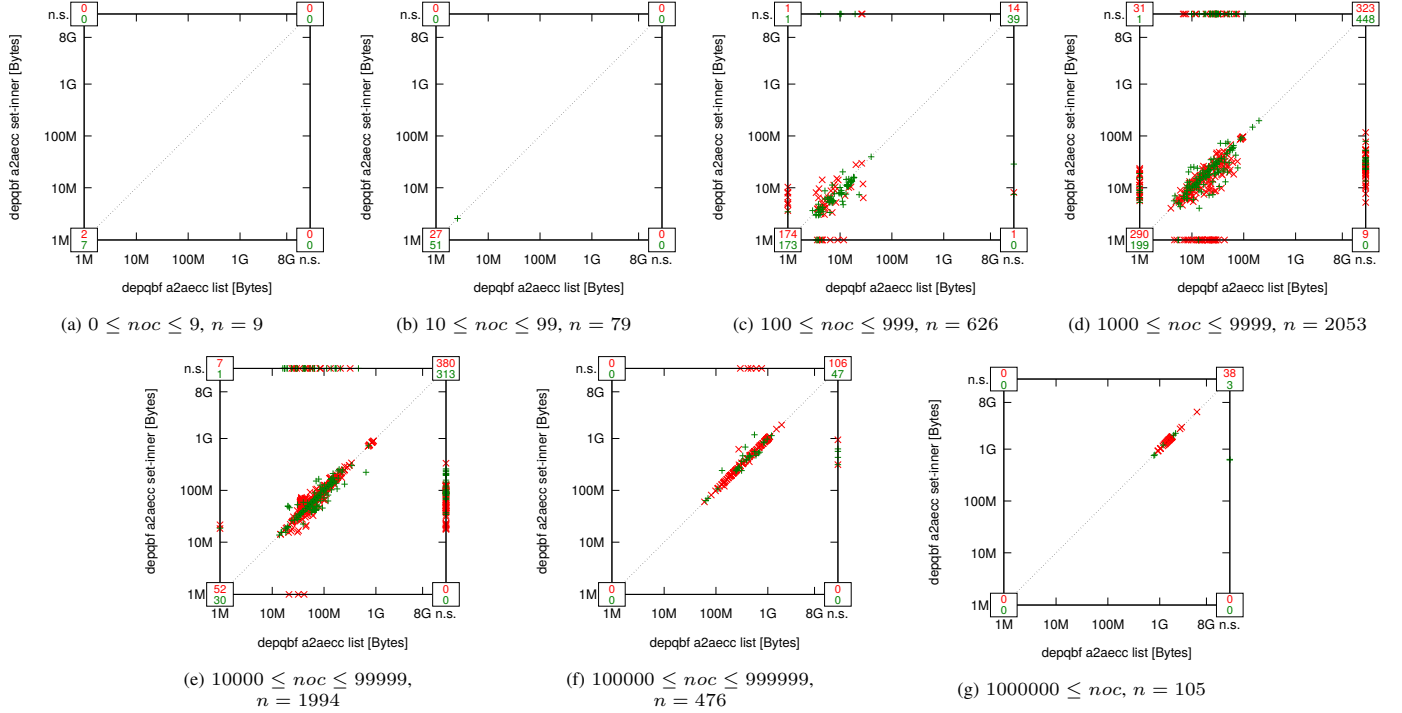


Fig. 1782: Solver DepQBF: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by number of clauses (memory usage in Bytes).

b) AIGSolve:

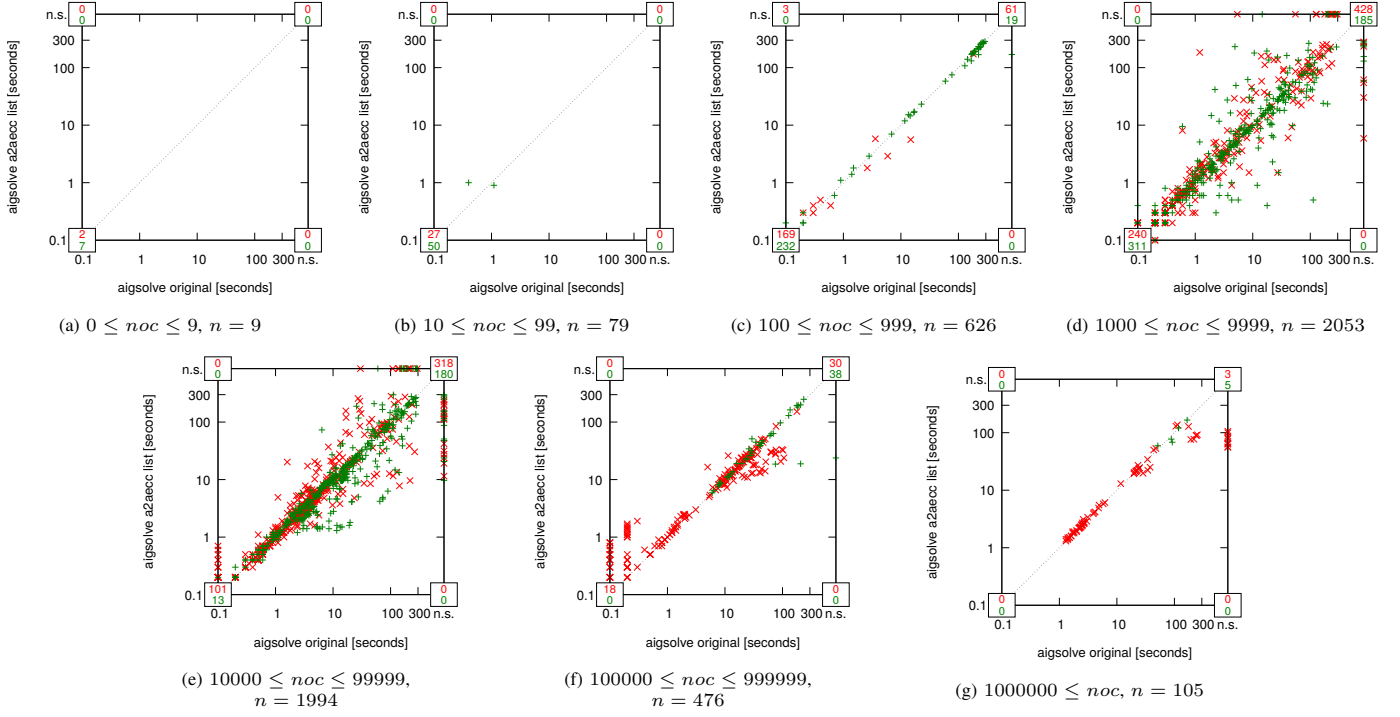


Fig. 1783: Solver AIGSolve: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by number of clauses (run time in seconds).

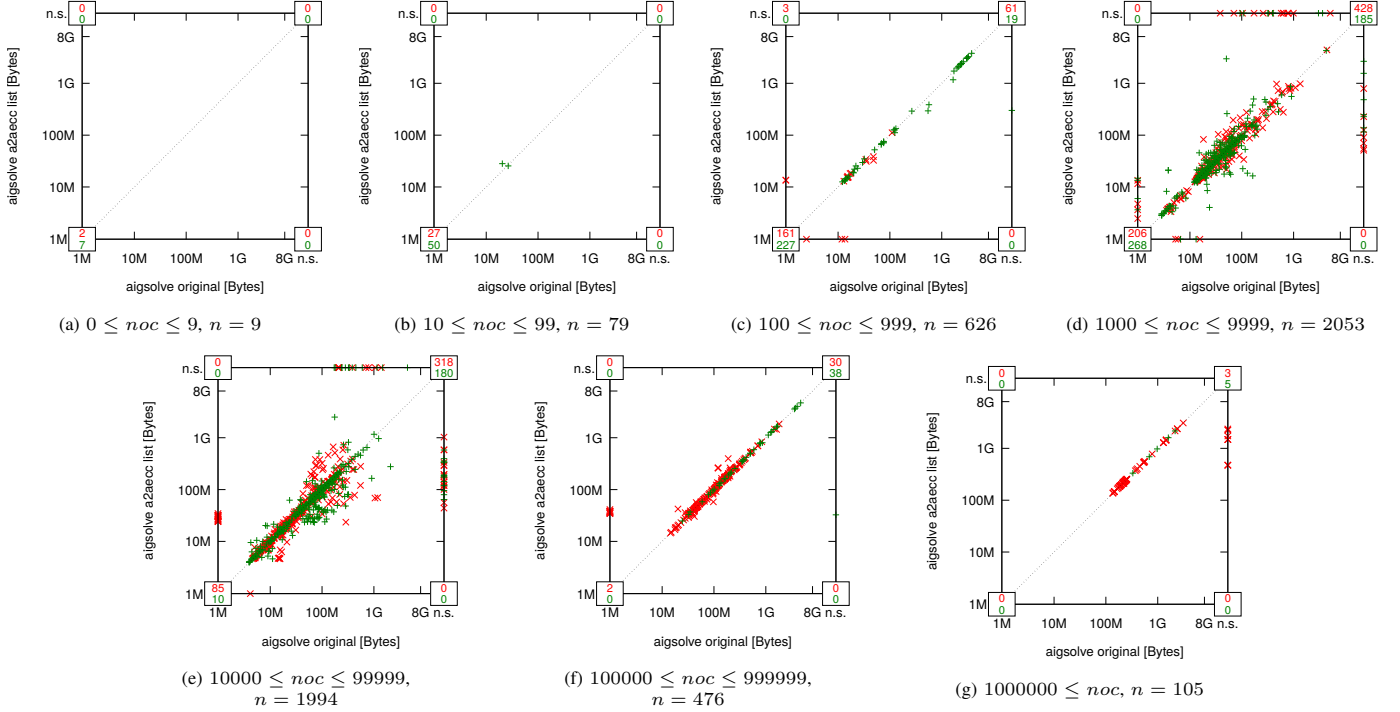


Fig. 1784: Solver AIGSolve: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by number of clauses (memory usage in Bytes).

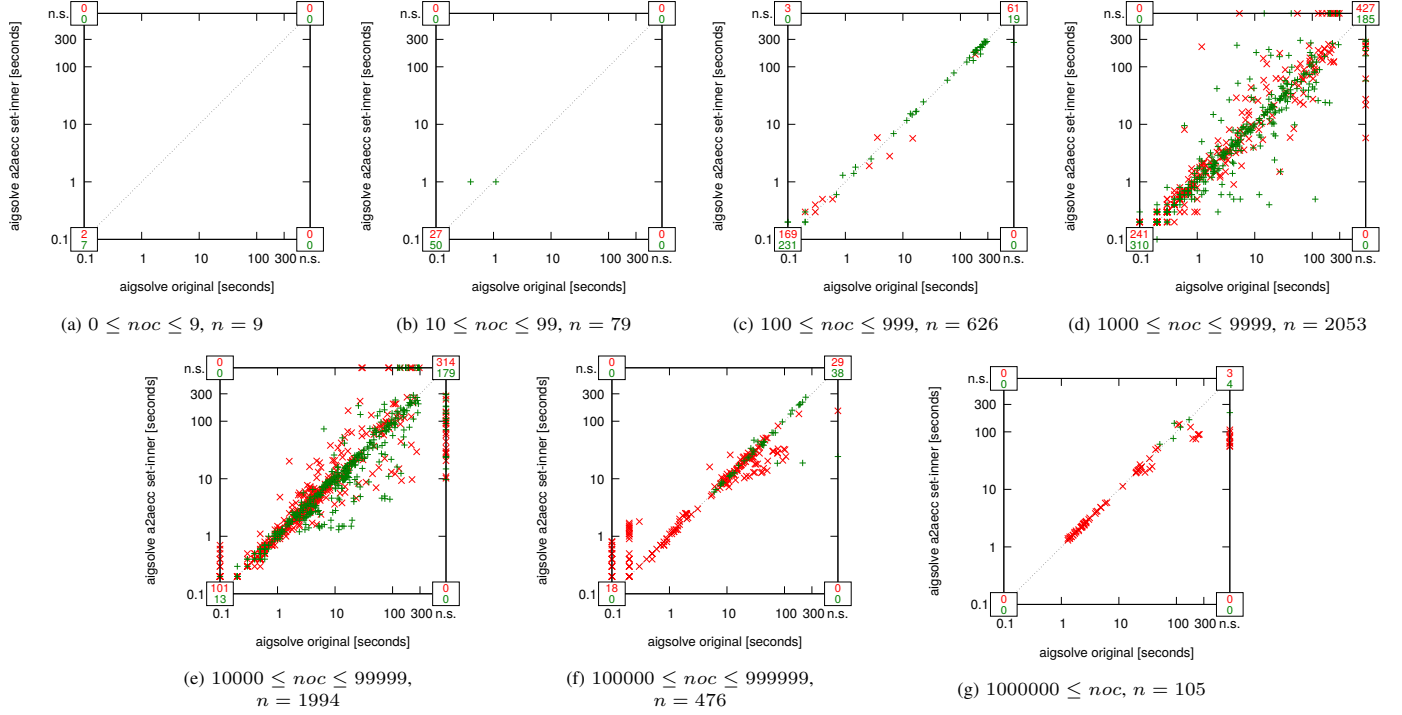


Fig. 1785: Solver AIGSolve: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by number of clauses (run time in seconds).

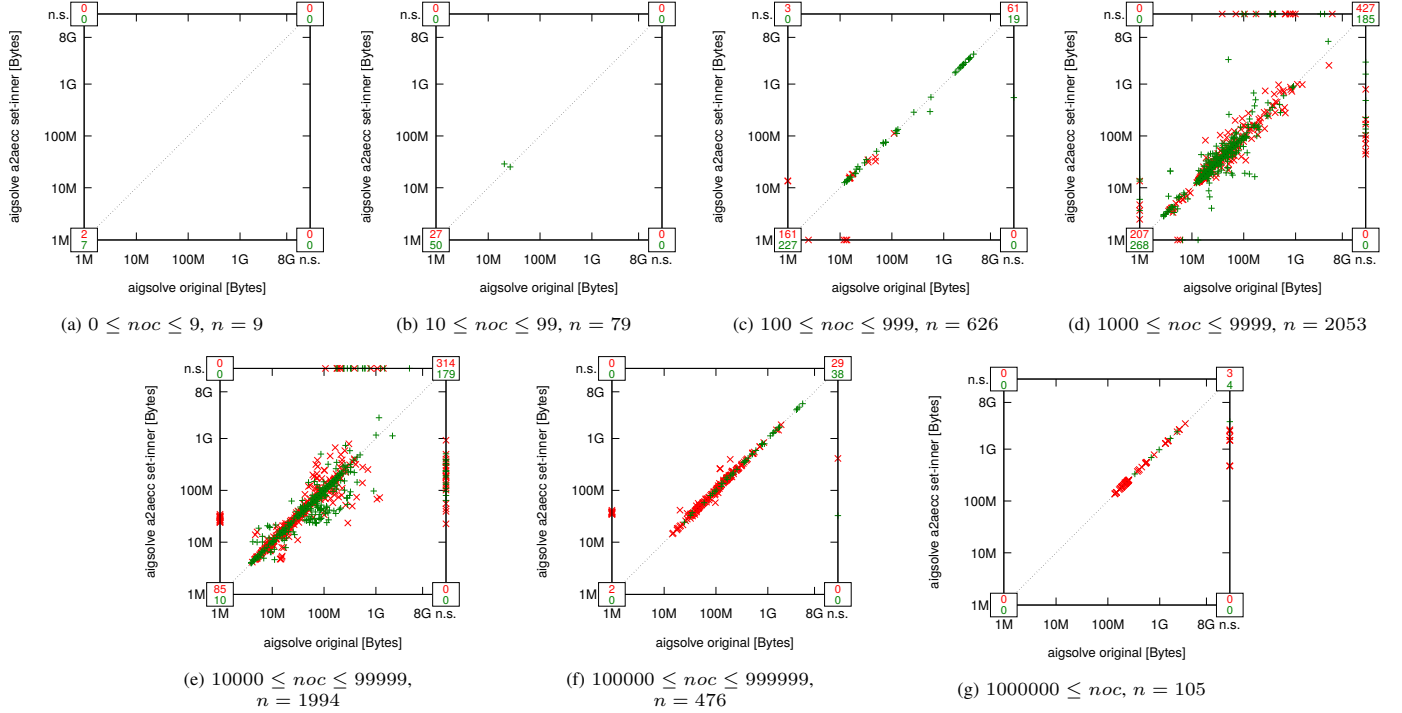


Fig. 1786: Solver AIGSolve: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by number of clauses (memory usage in Bytes).

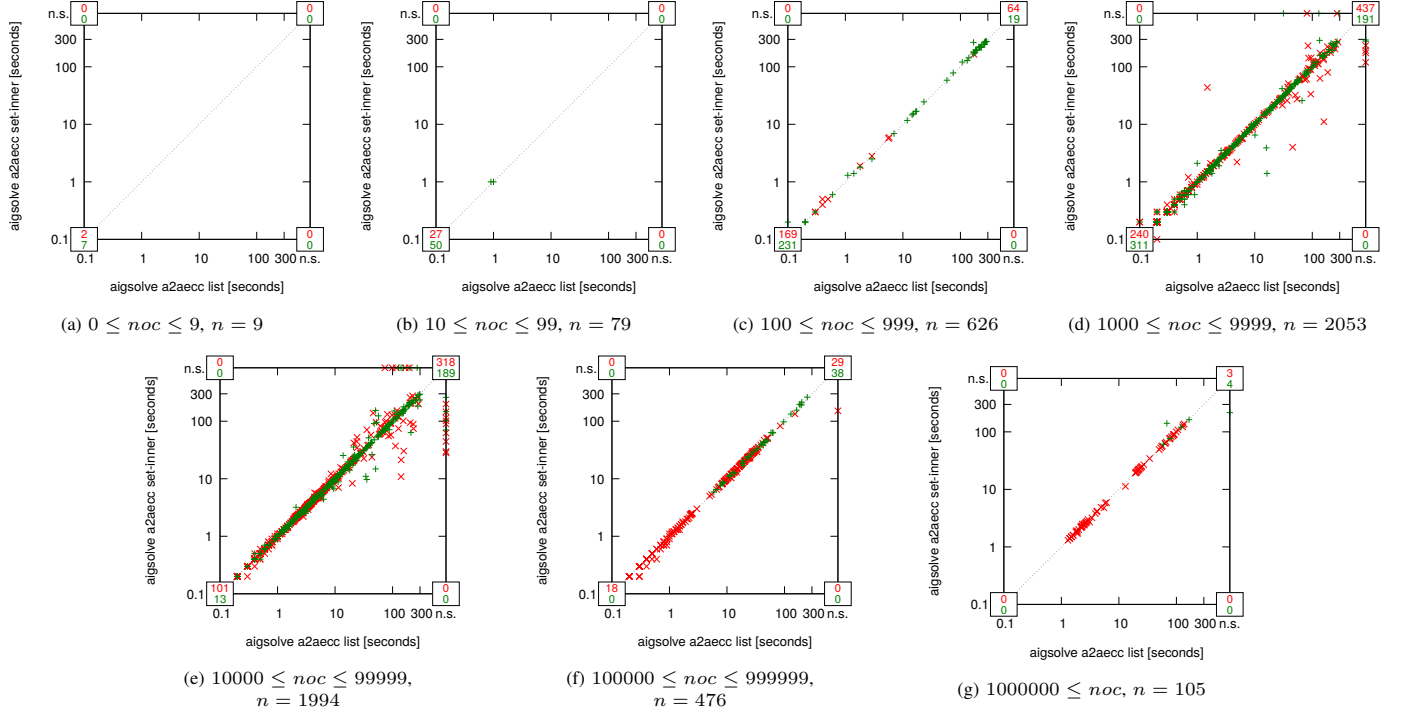


Fig. 1787: Solver AIGSolve: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by number of clauses (run time in seconds).

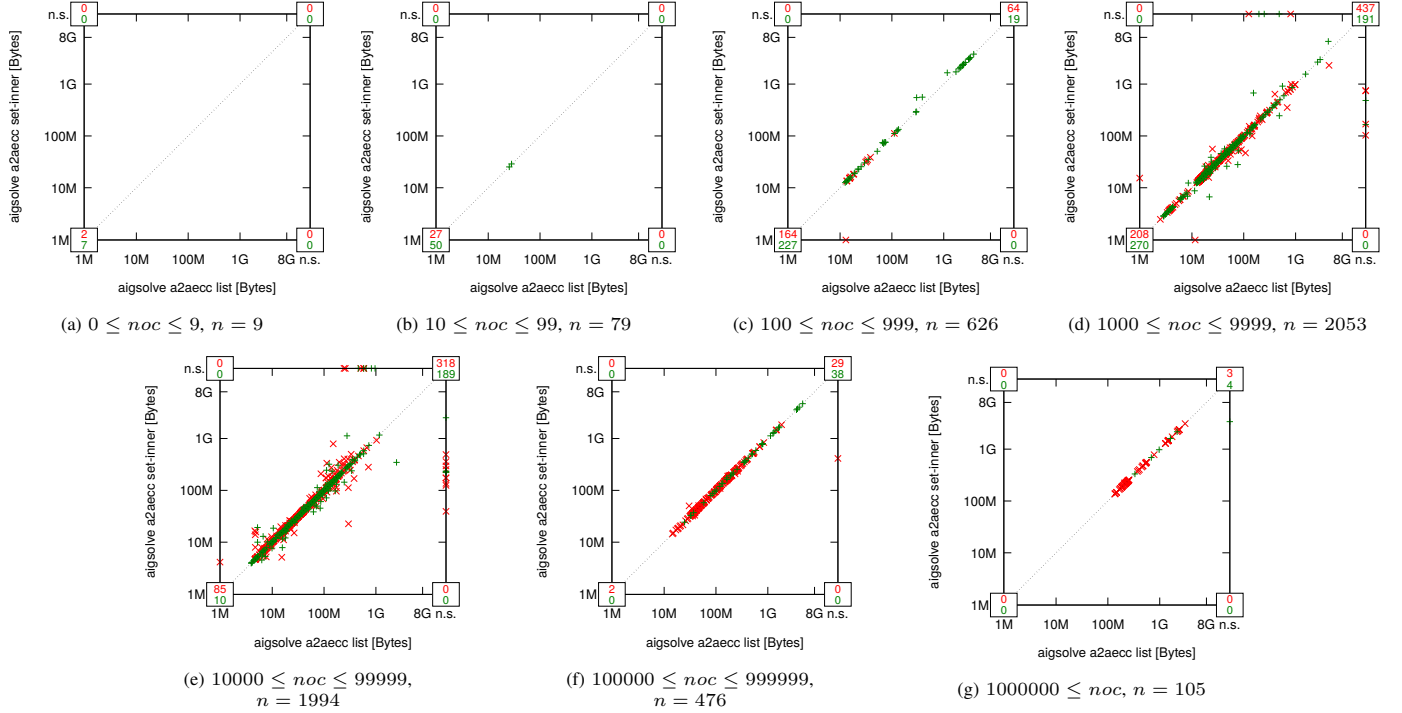


Fig. 1788: Solver AIGSolve: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by number of clauses (memory usage in Bytes).

c) CAQE:

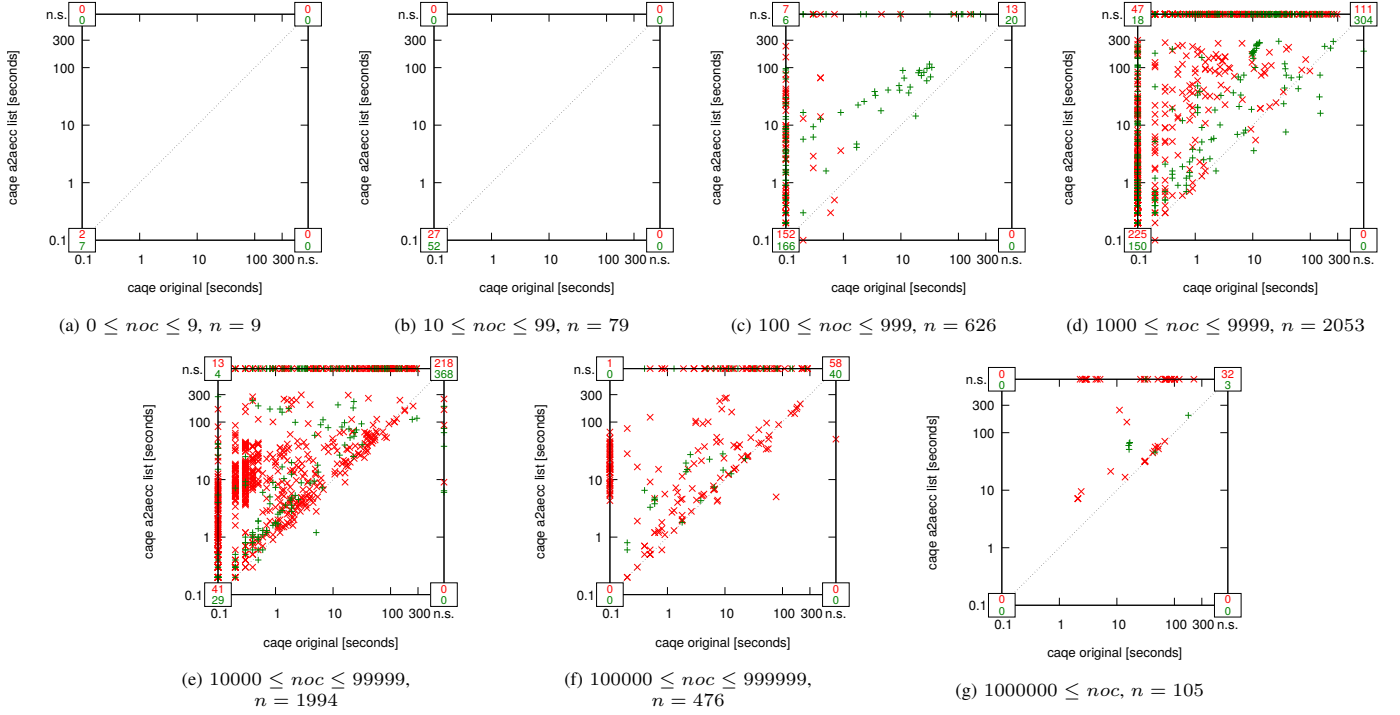


Fig. 1789: Solver CAQE: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by number of clauses (run time in seconds).

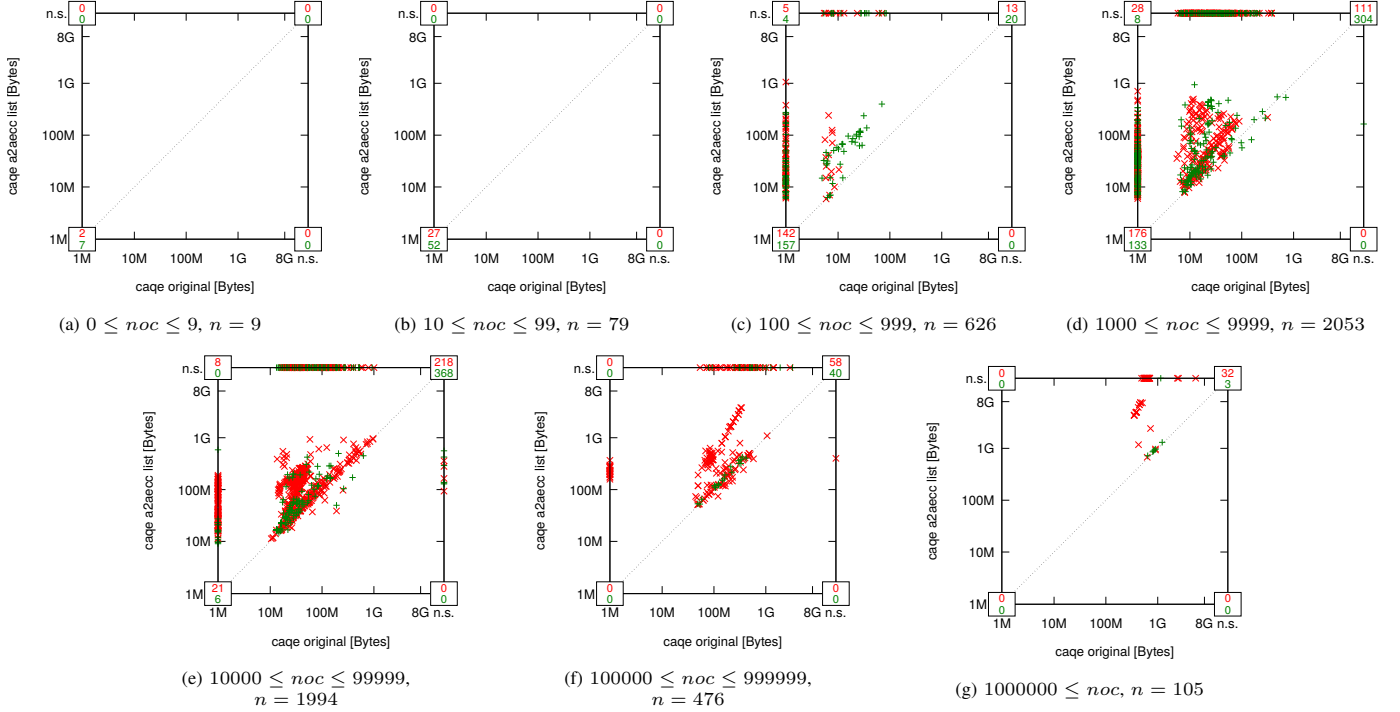


Fig. 1790: Solver CAQE: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by number of clauses (memory usage in Bytes).

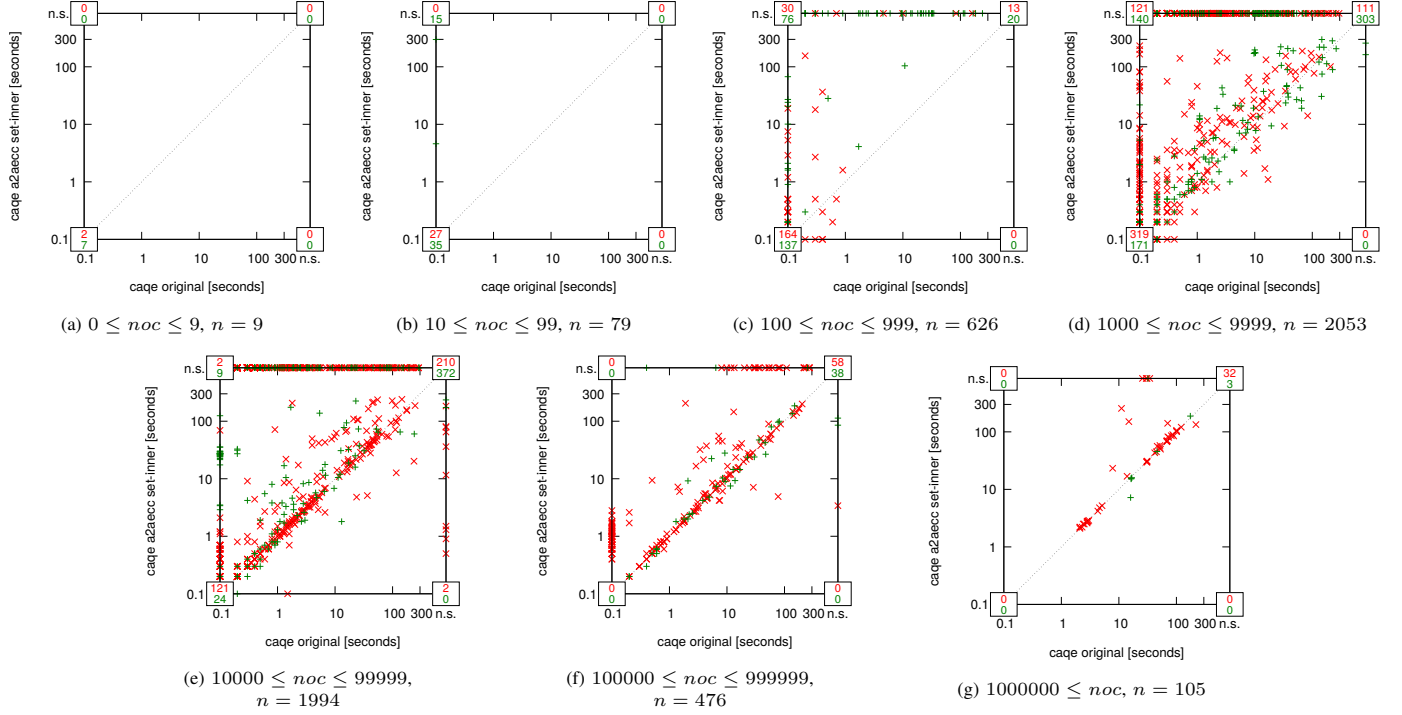


Fig. 1791: Solver CAQE: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by number of clauses (run time in seconds).

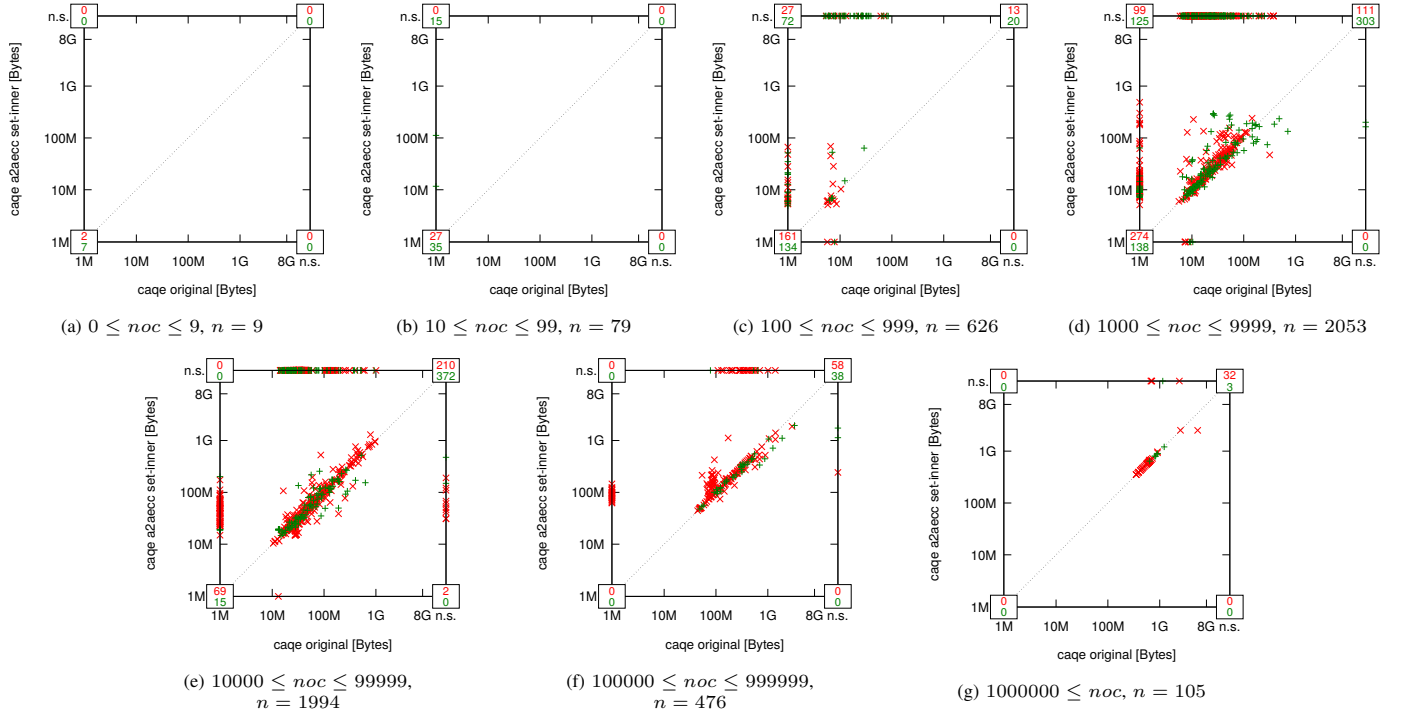


Fig. 1792: Solver CAQE: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by number of clauses (memory usage in Bytes).

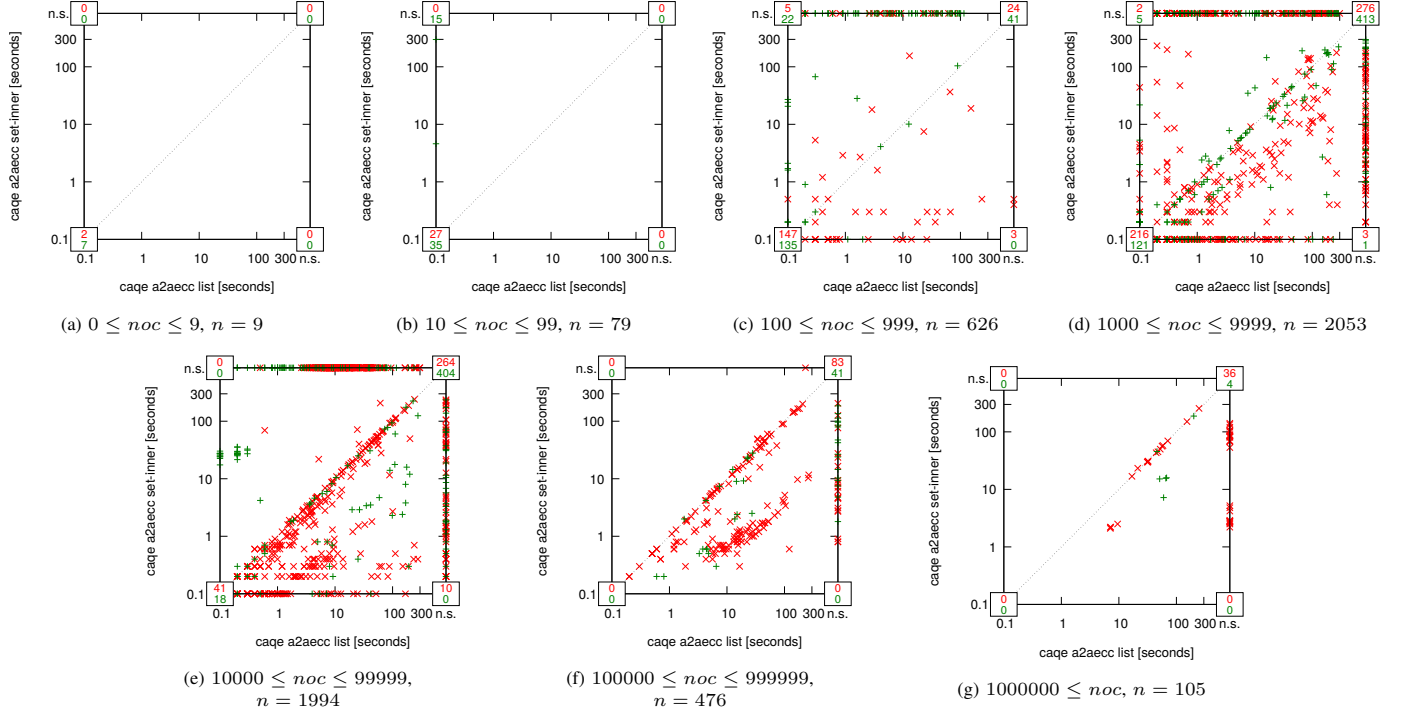


Fig. 1793: Solver CAQE: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by number of clauses (run time in seconds).

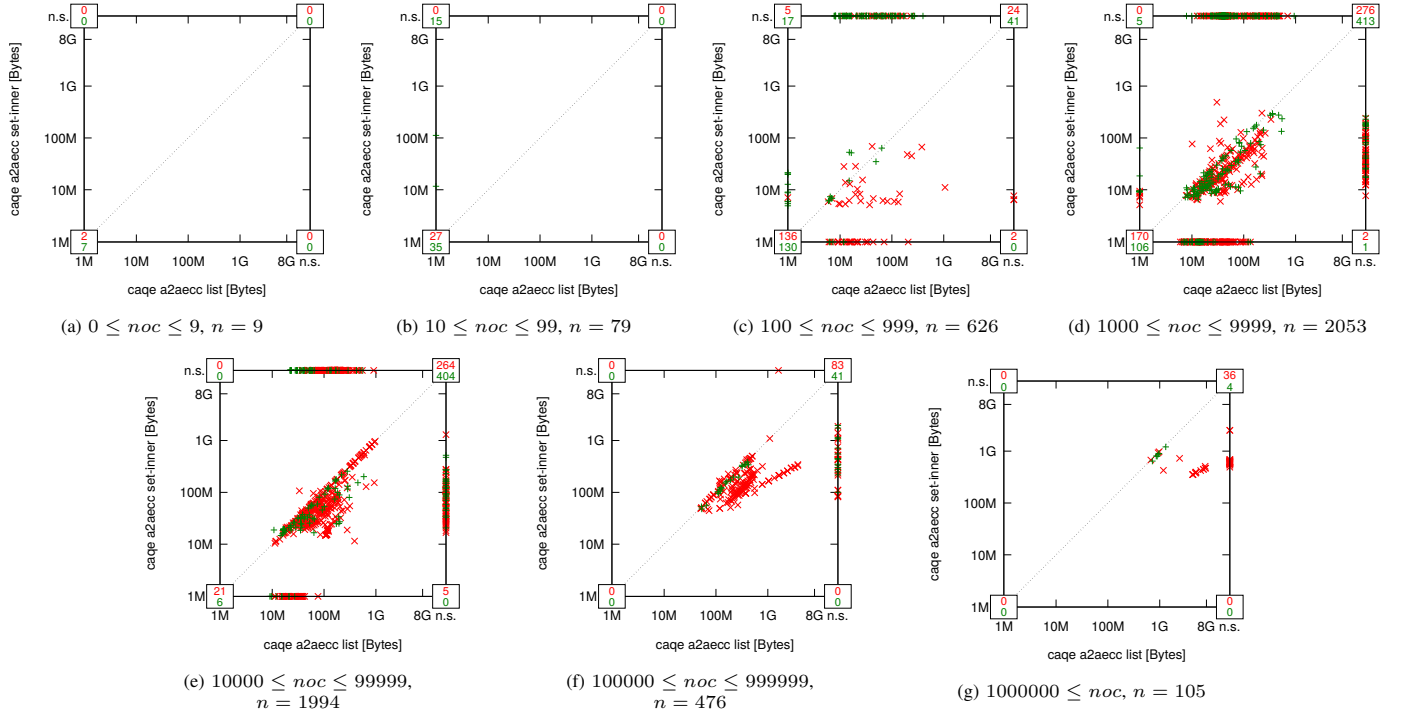


Fig. 1794: Solver CAQE: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by number of clauses (memory usage in Bytes).

d) GhostQ:

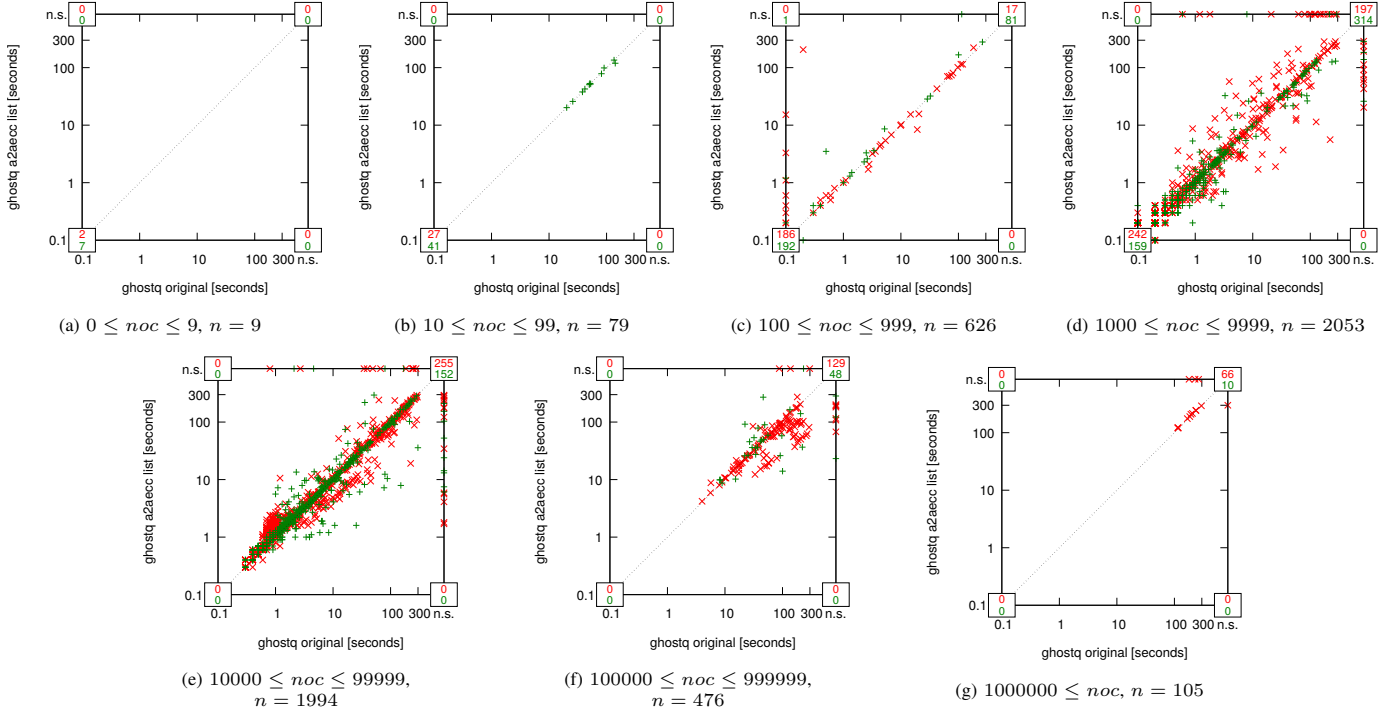


Fig. 1795: Solver GhostQ: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by number of clauses (run time in seconds).

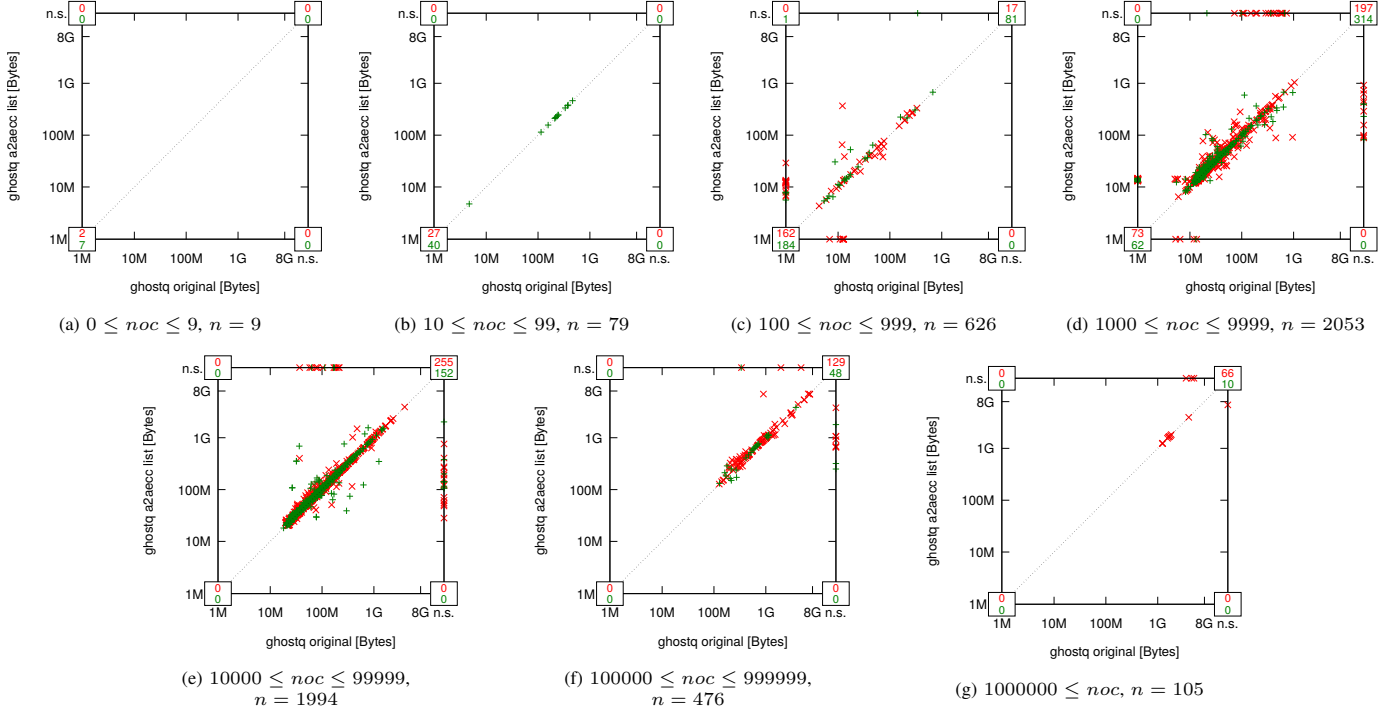


Fig. 1796: Solver GhostQ: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by number of clauses (memory usage in Bytes).

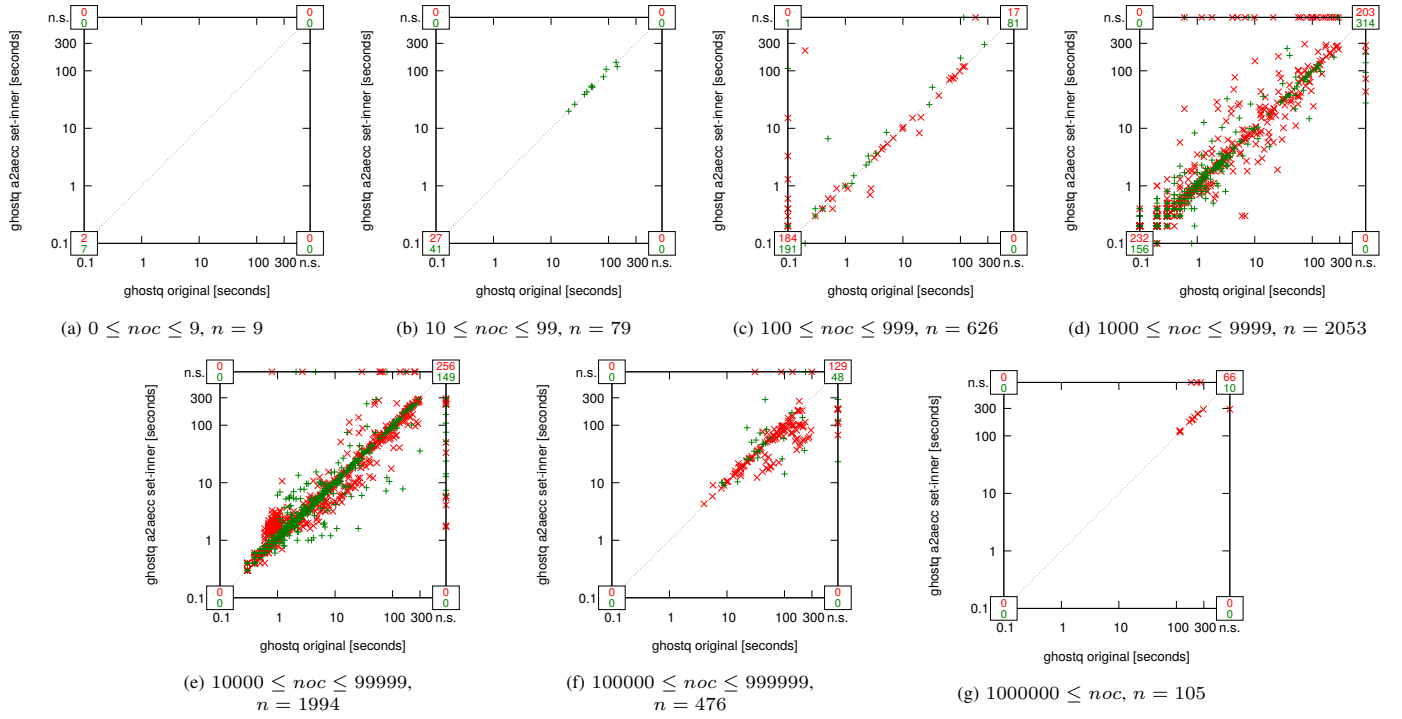


Fig. 1797: Solver GhostQ: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by number of clauses (run time in seconds).

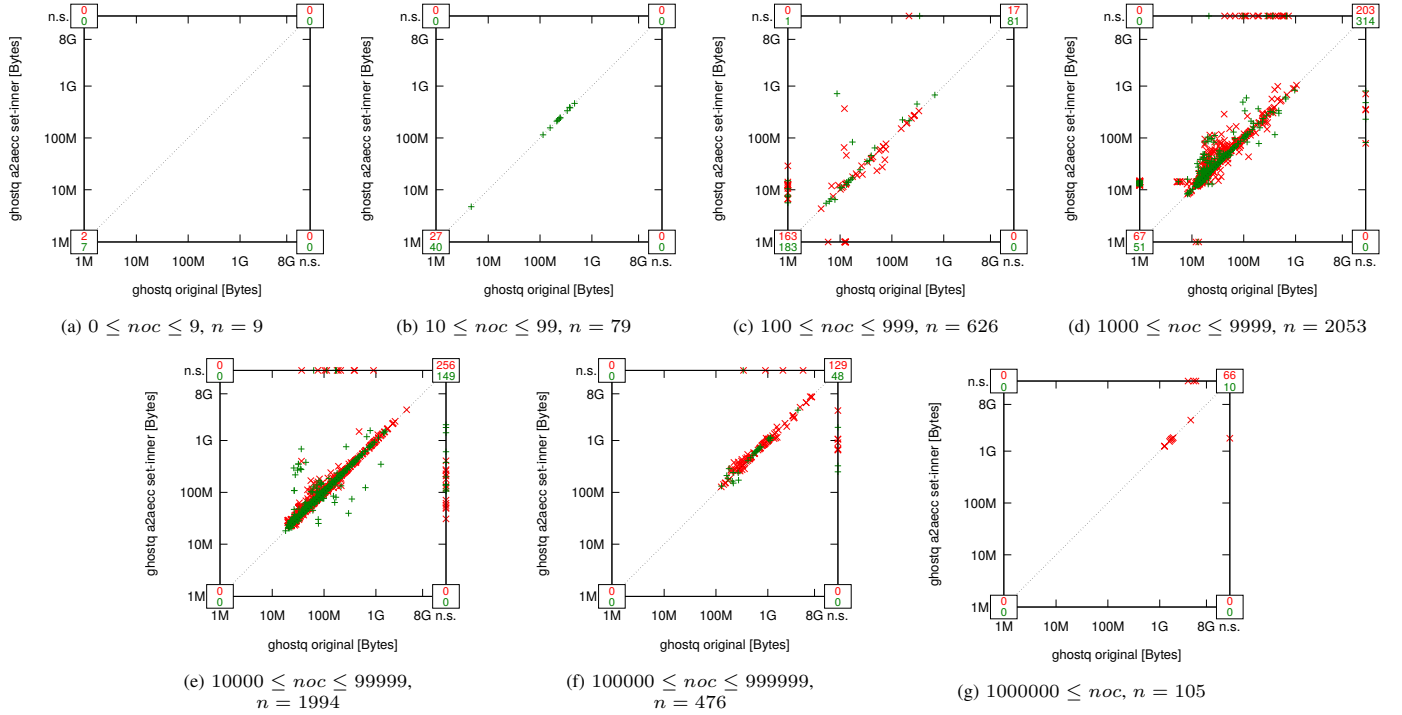


Fig. 1798: Solver GhostQ: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by number of clauses (memory usage in Bytes).

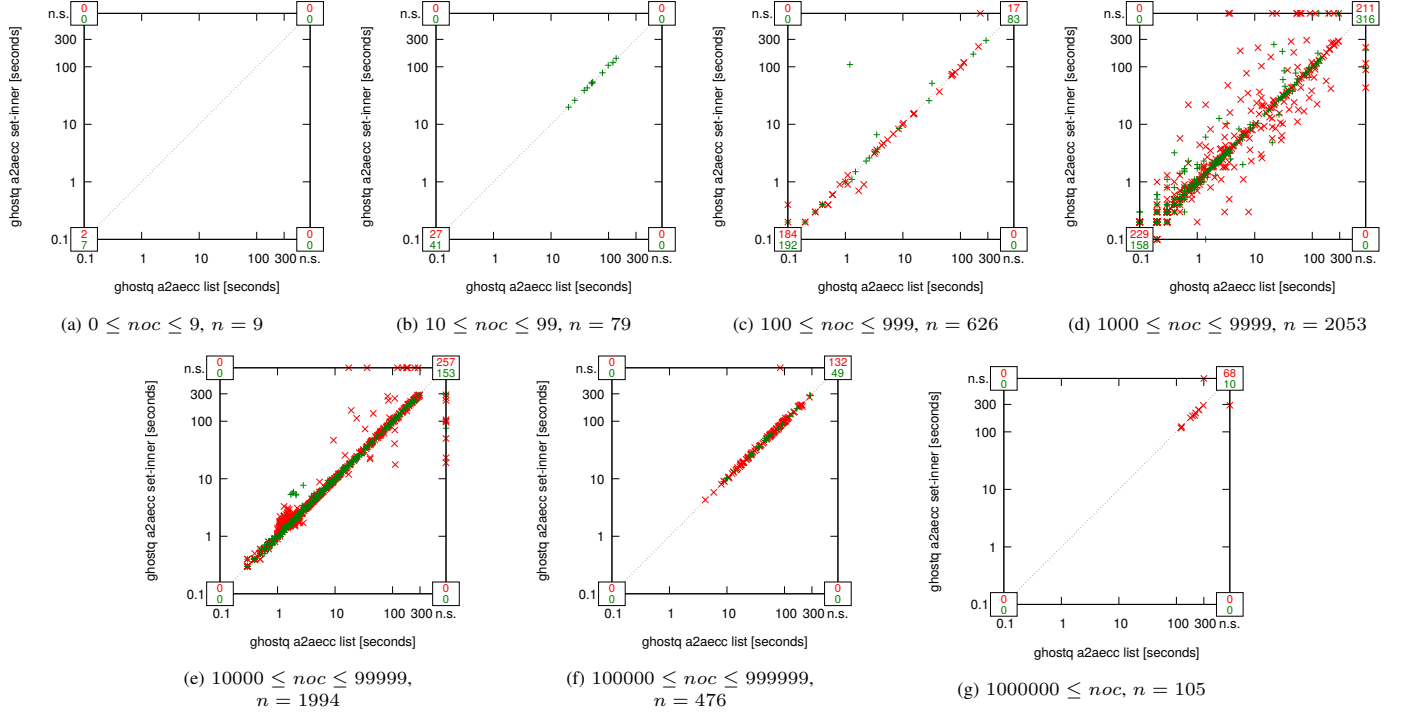


Fig. 1799: Solver GhostQ: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by number of clauses (run time in seconds).

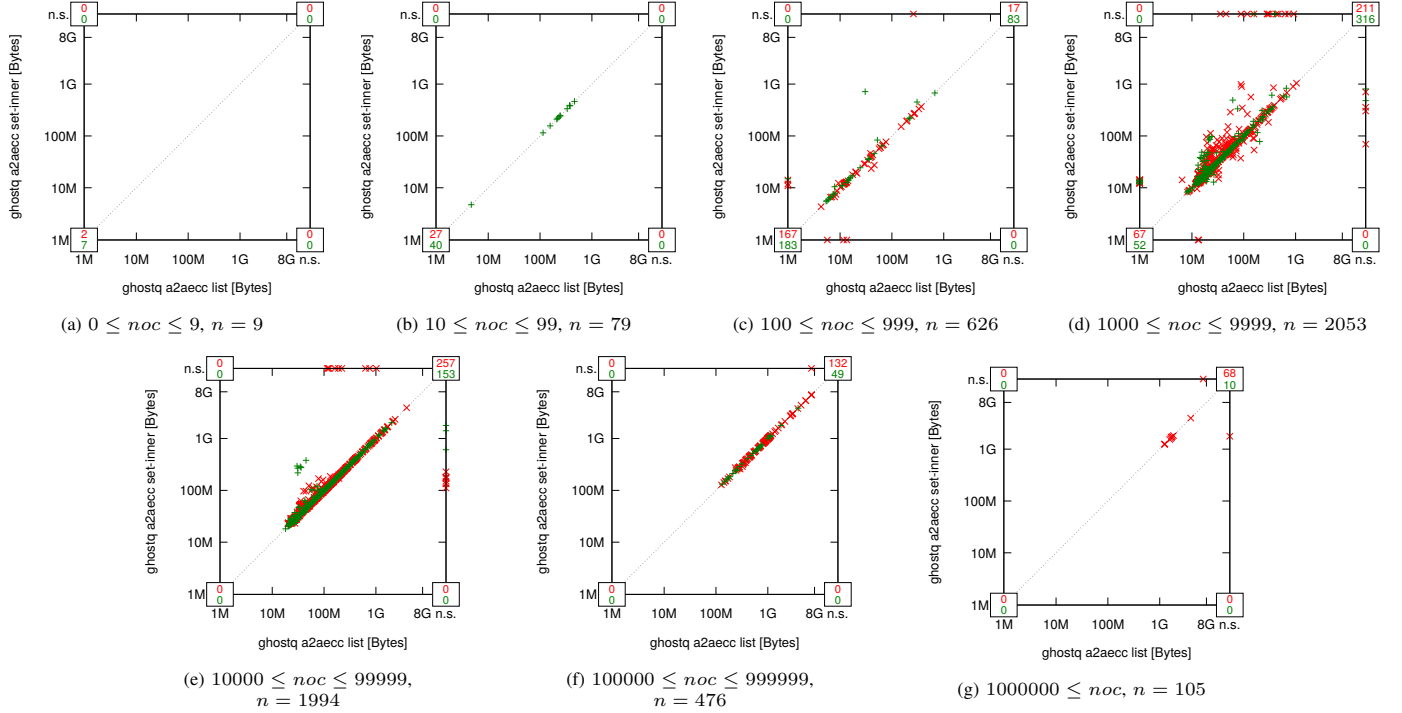


Fig. 1800: Solver GhostQ: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by number of clauses (memory usage in Bytes).

e) QESTO:

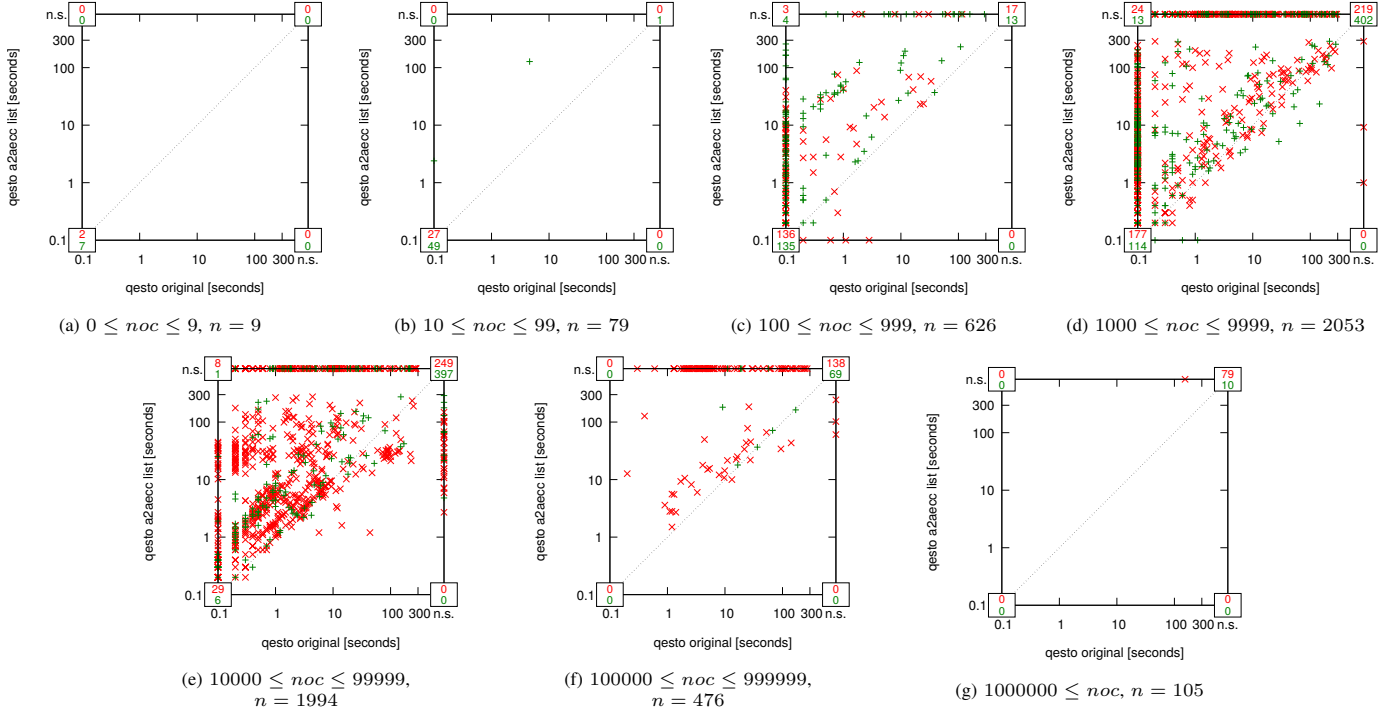


Fig. 1801: Solver QESTO: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by number of clauses (run time in seconds).

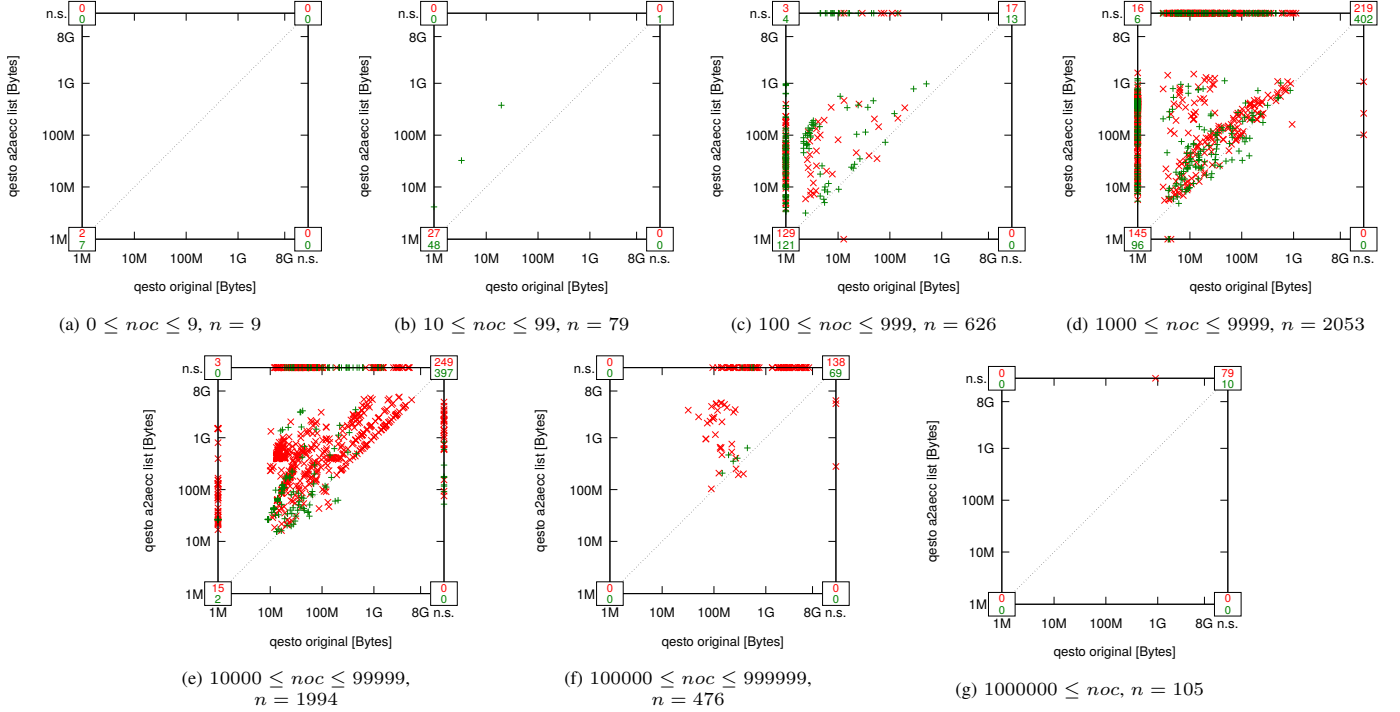


Fig. 1802: Solver QESTO: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by number of clauses (memory usage in Bytes).

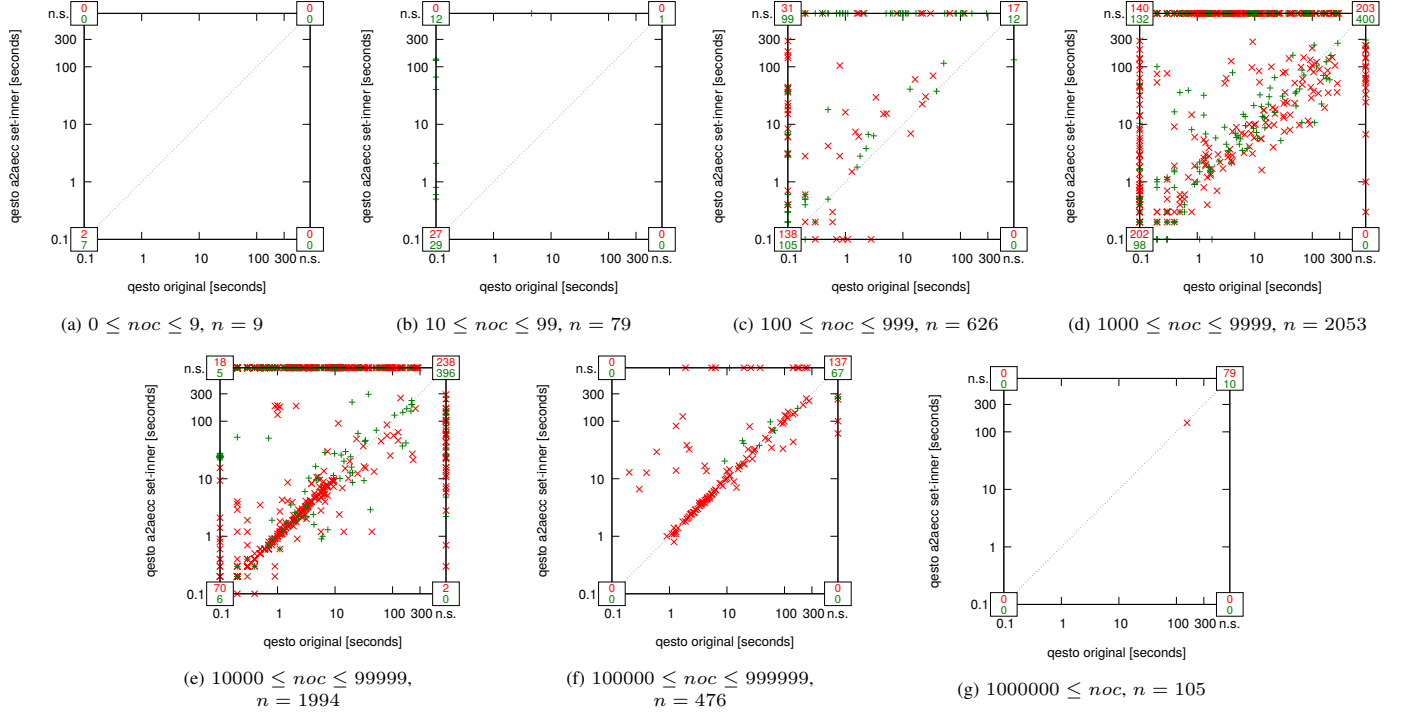


Fig. 1803: Solver QESTO: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by number of clauses (run time in seconds).

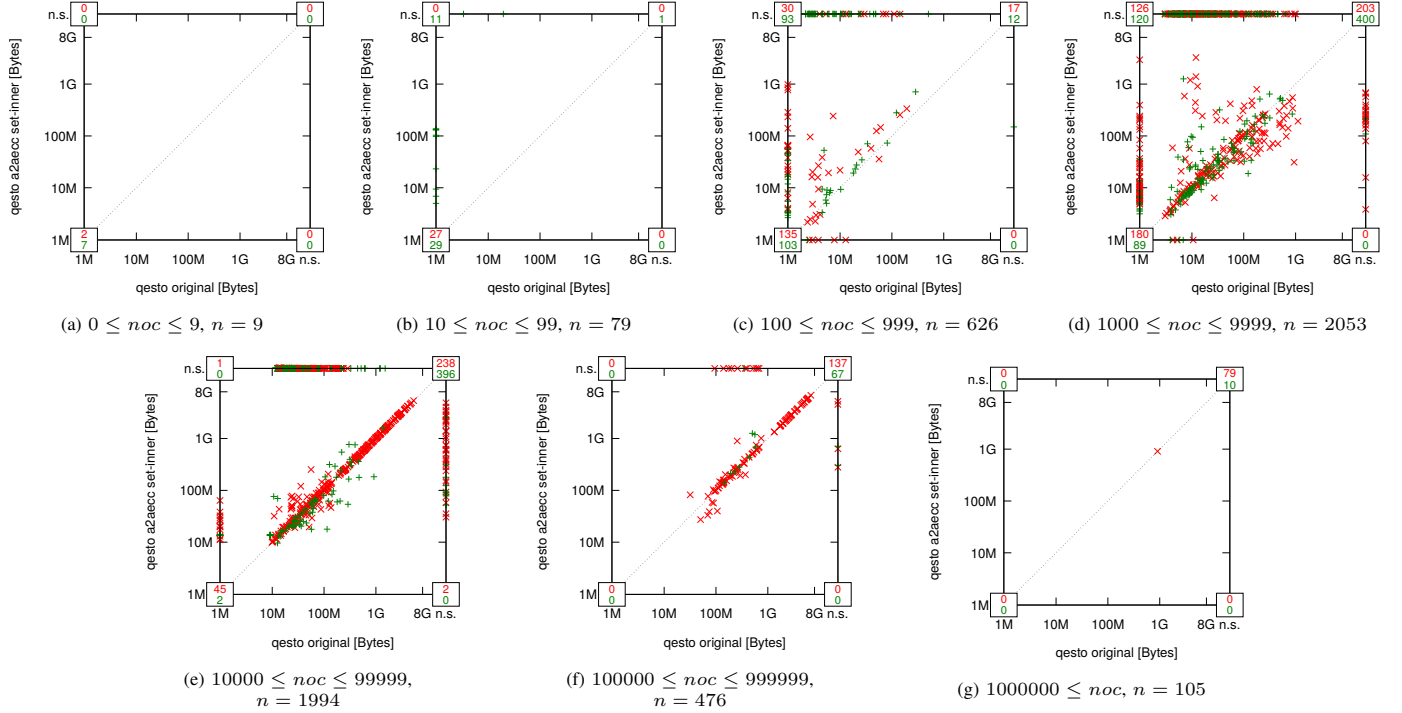


Fig. 1804: Solver QESTO: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by number of clauses (memory usage in Bytes).

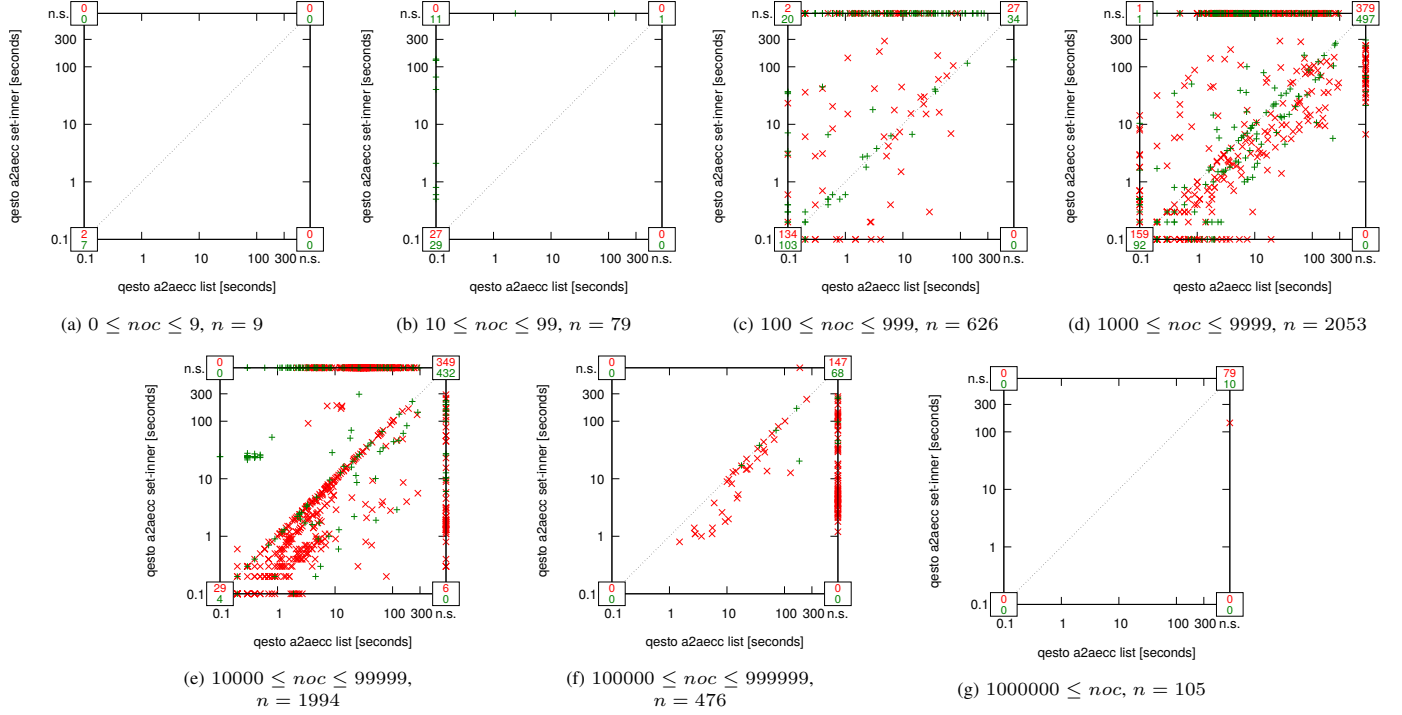


Fig. 1805: Solver QESTO: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by number of clauses (run time in seconds).

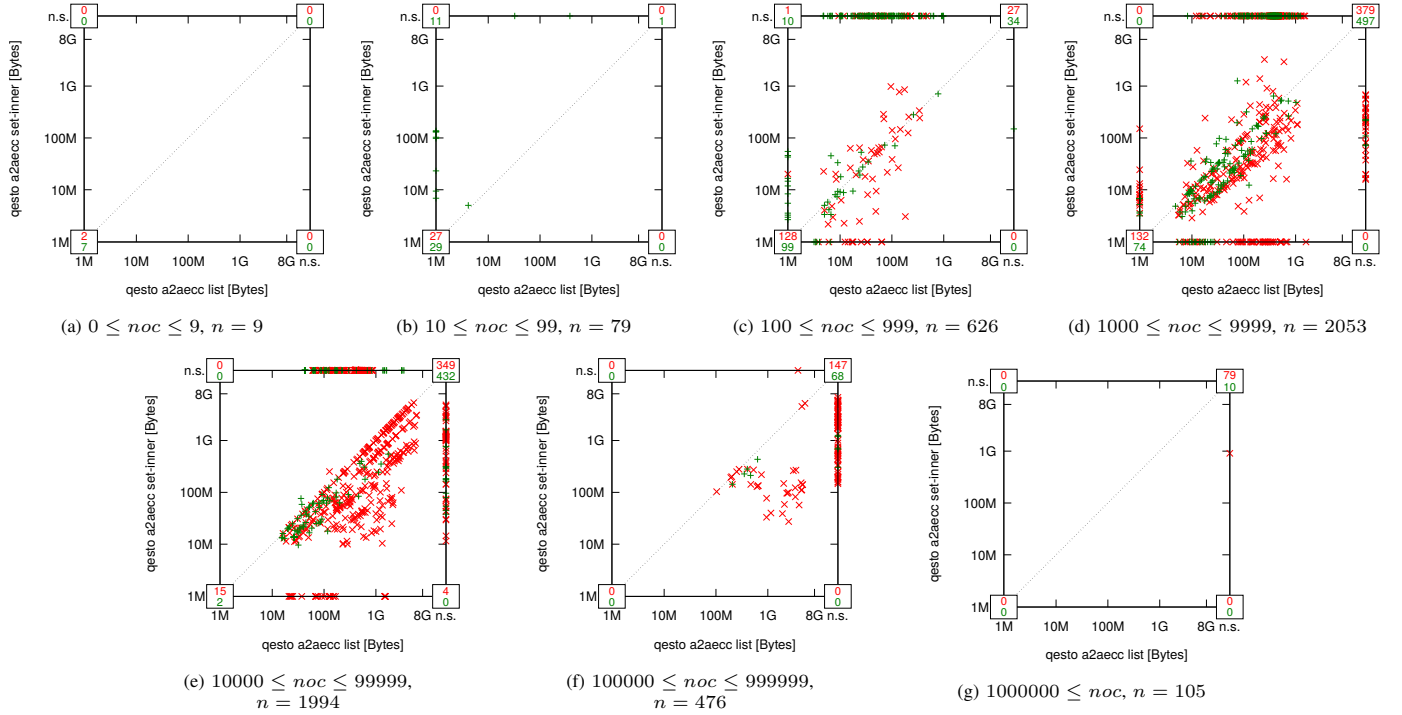


Fig. 1806: Solver QESTO: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by number of clauses (memory usage in Bytes).

f) RAReQS:

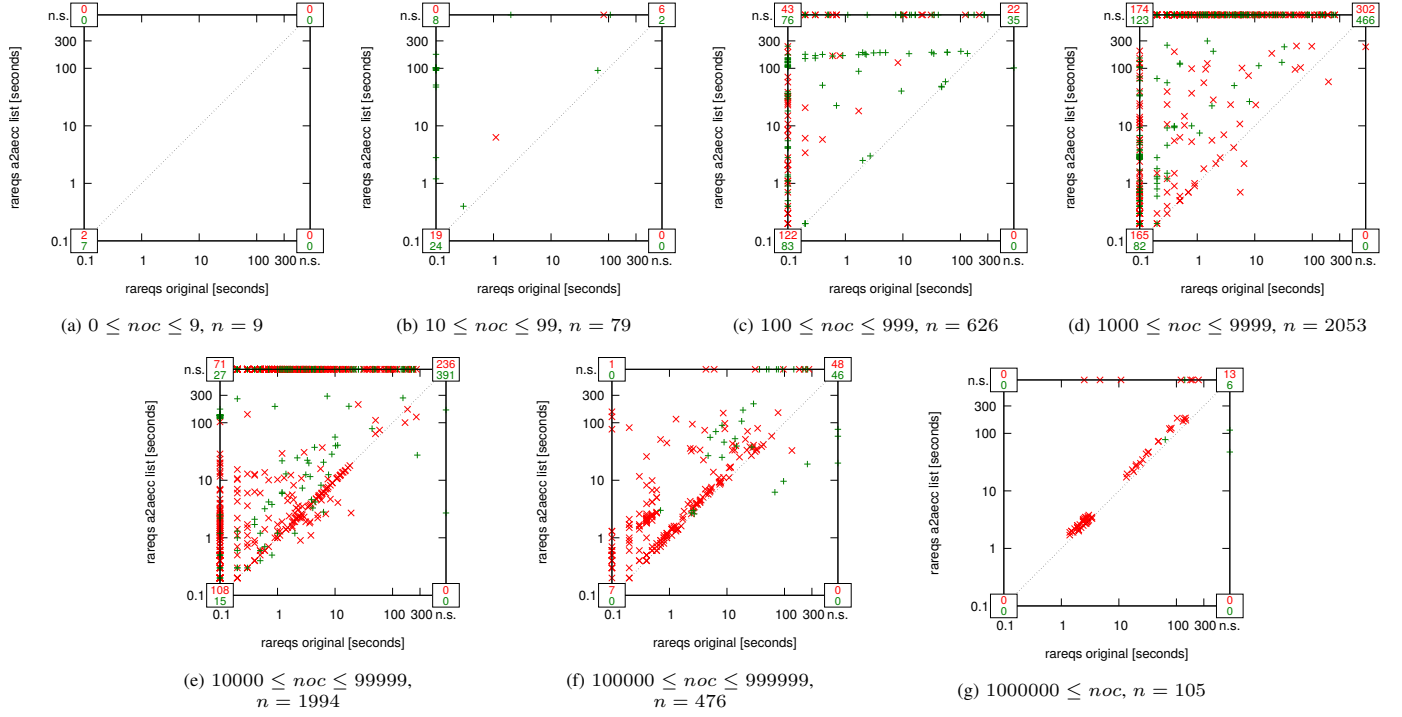


Fig. 1807: Solver RAReQS: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by number of clauses (run time in seconds).

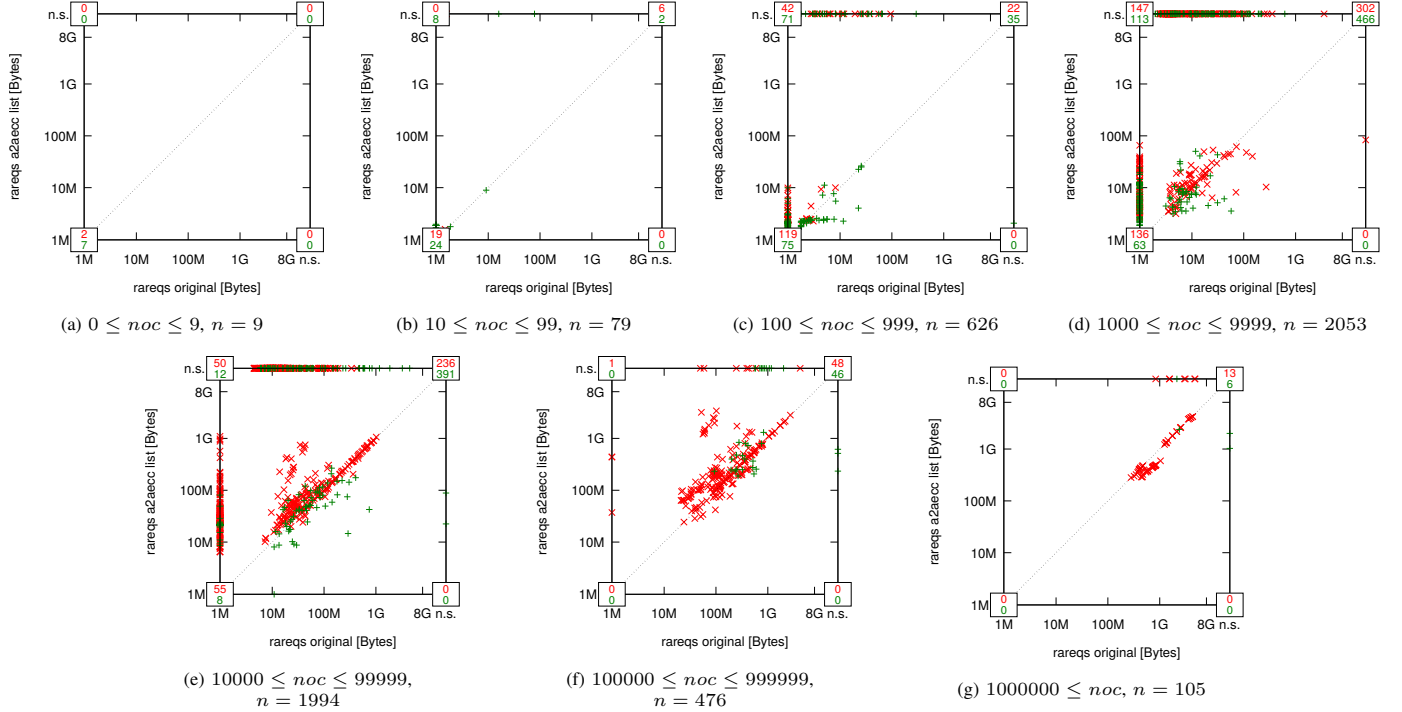


Fig. 1808: Solver RAReQS: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by number of clauses (memory usage in Bytes).

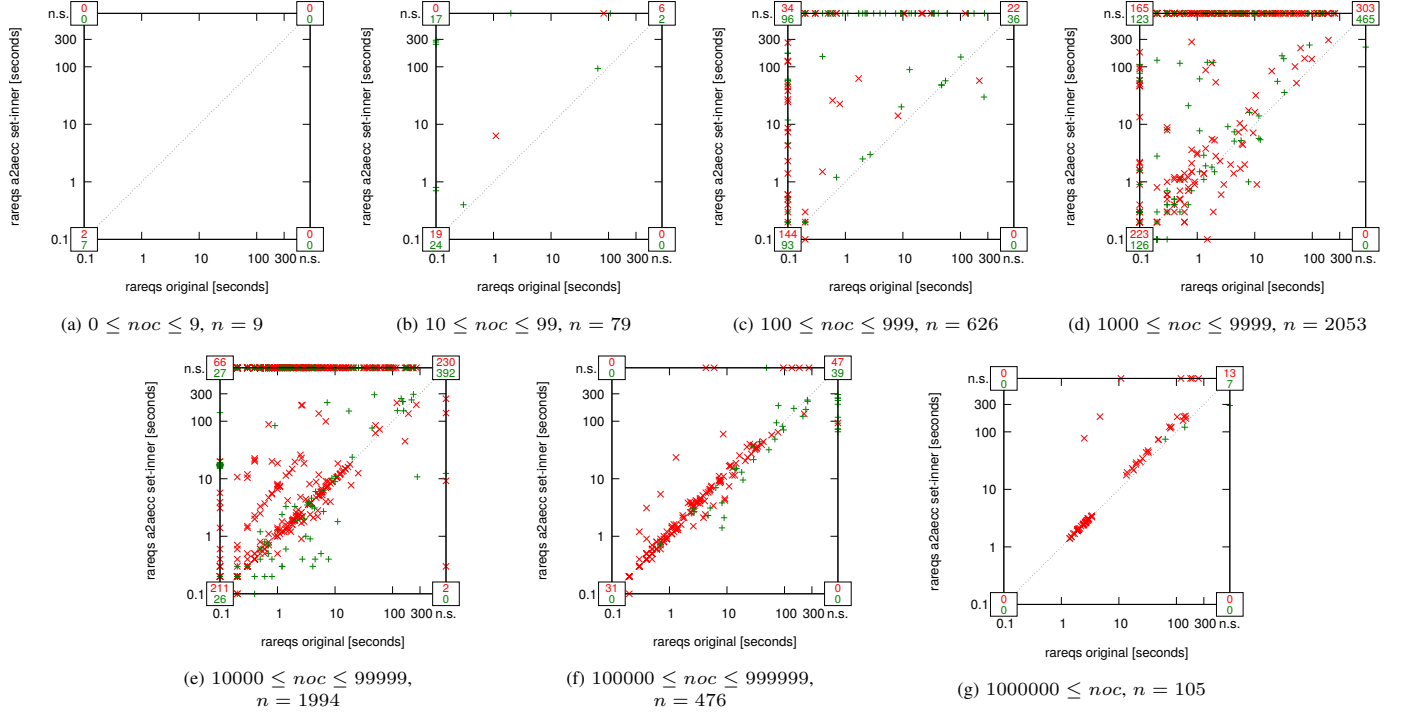


Fig. 1809: Solver RAREQS: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by number of clauses (run time in seconds).

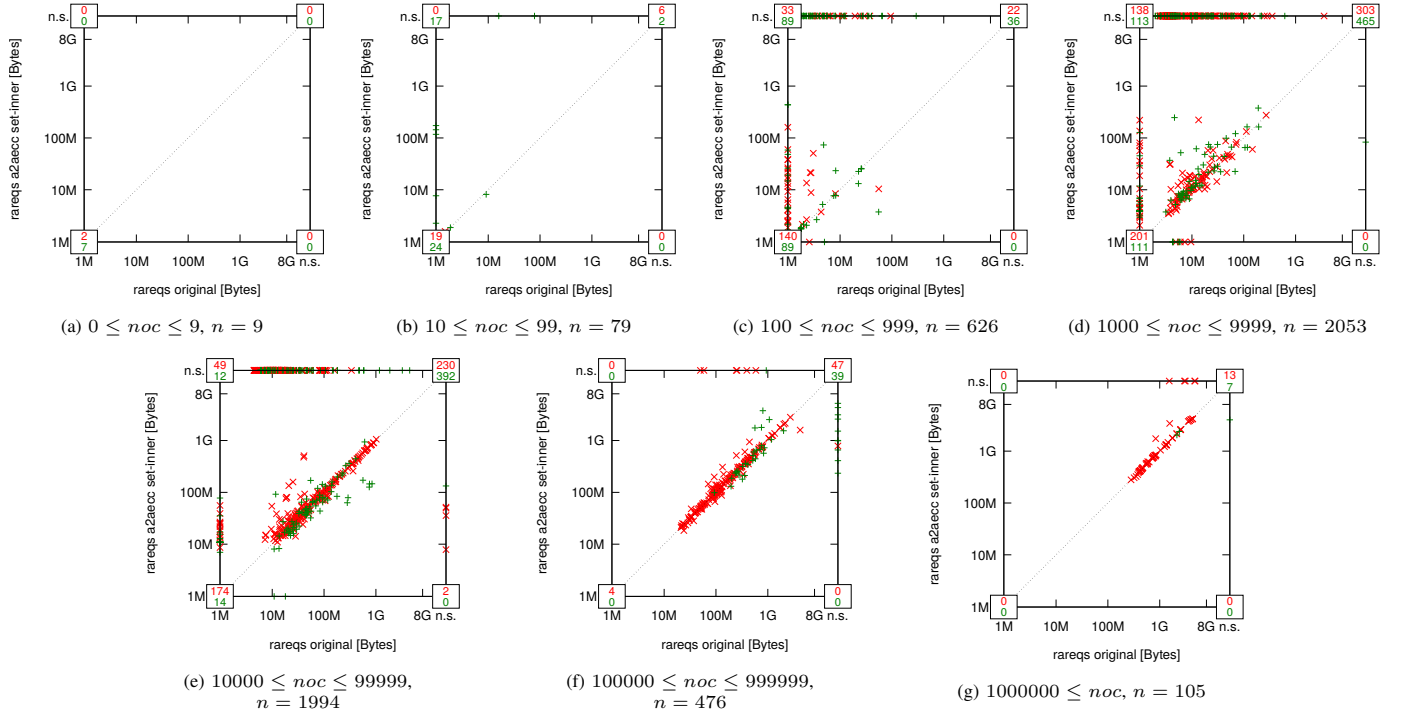


Fig. 1810: Solver RAREQS: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by number of clauses (memory usage in Bytes).

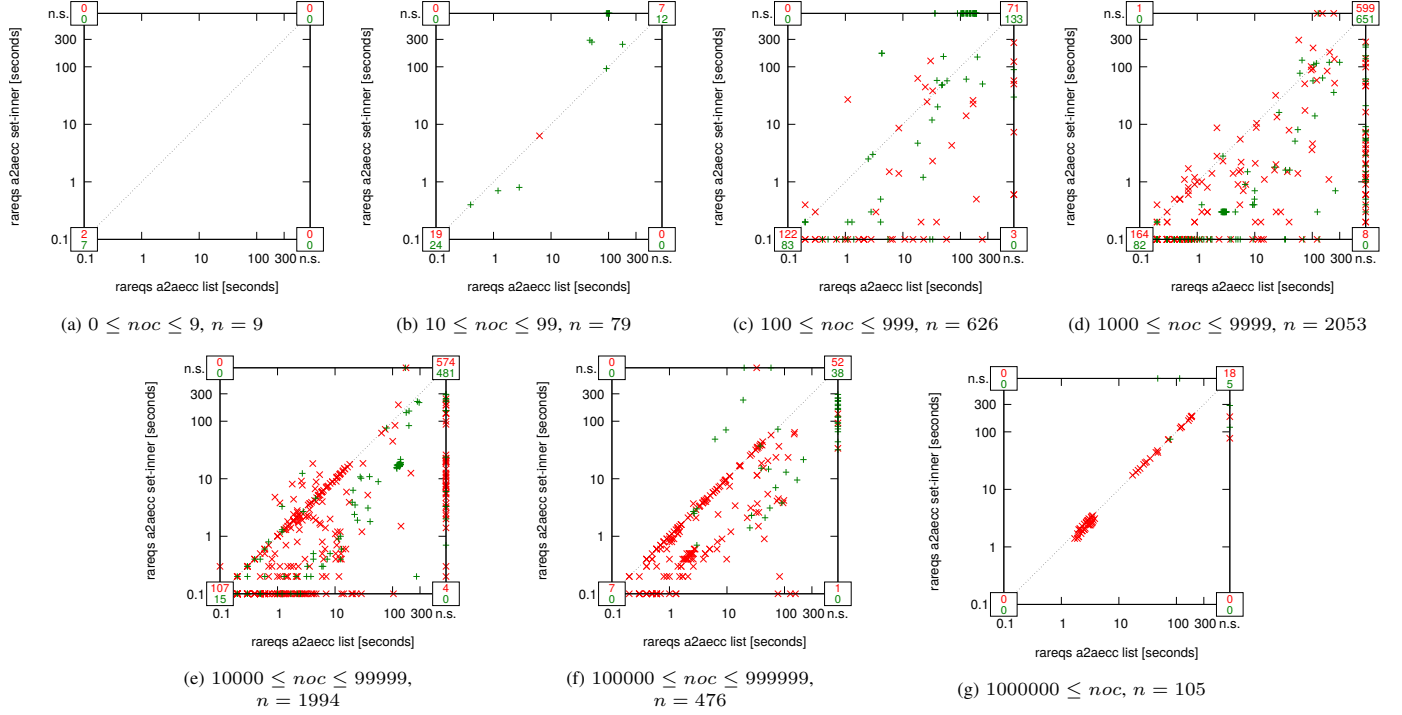


Fig. 1811: Solver RAReQS: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by number of clauses (run time in seconds).

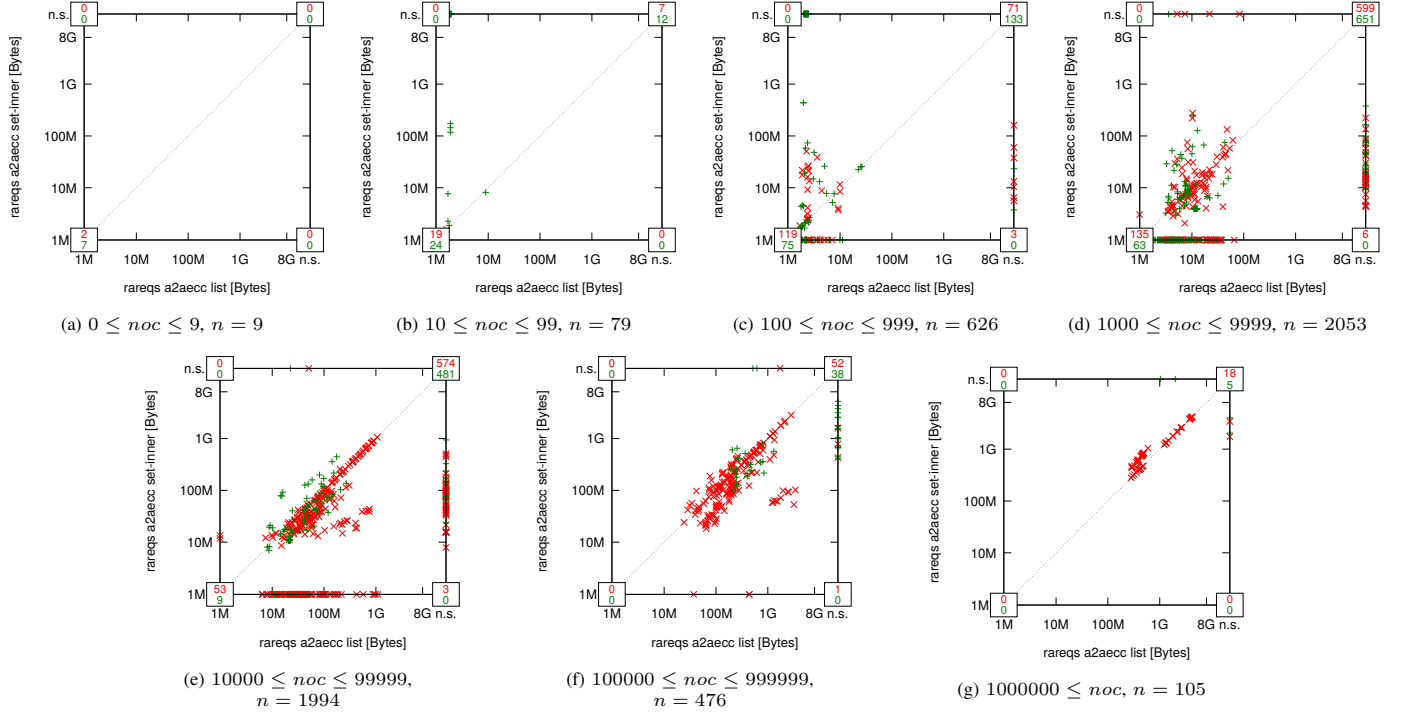


Fig. 1812: Solver RAReQS: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by number of clauses (memory usage in Bytes).

E. Partitioned by Maximum Variable Index

In this subsection there are 6 figures for each solver with subfigures for partitions of the benchmarks according to their maximum variable index.

- 1) Comparing run times for solving the transformed versus the original instances with list semantics.
- 2) Memory usage for 1.
- 3) As 1. but with set-inner semantics.
- 4) Memory usage for 3.
- 5) A direct comparison of solving the transformed instances with set-inner versus list semantics.
- 6) Memory usage for 5.

a) DepQBF:

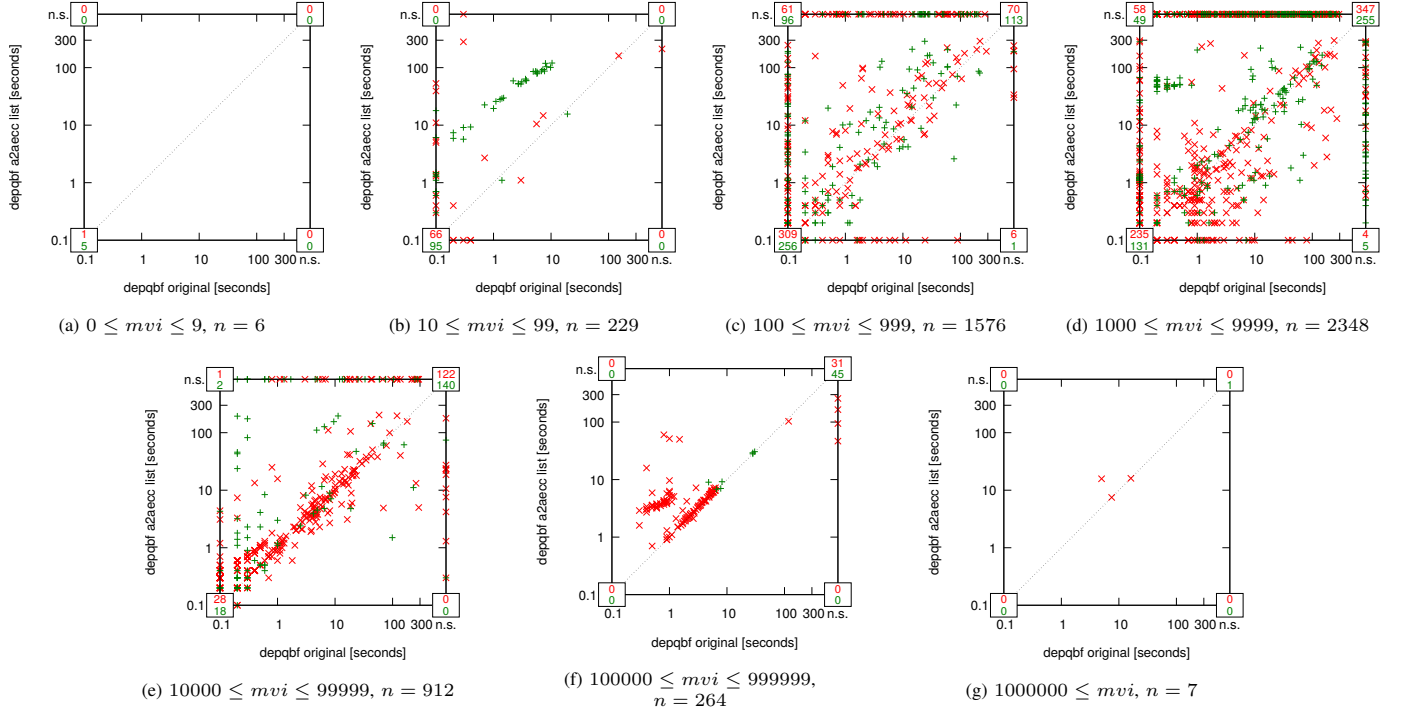


Fig. 1813: Solver DepQBF: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by maximum variable index (run time in seconds).

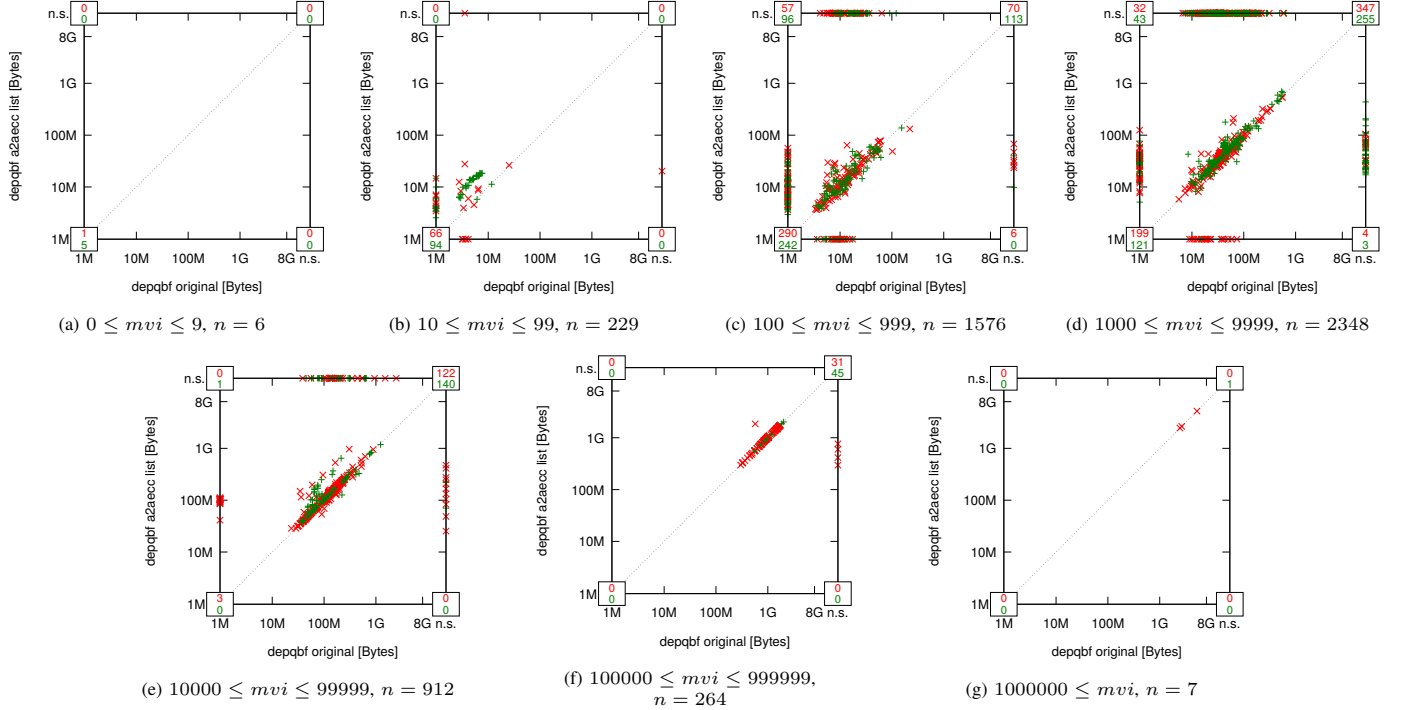


Fig. 1814: Solver DepQBF: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by maximum variable index (memory usage in Bytes).

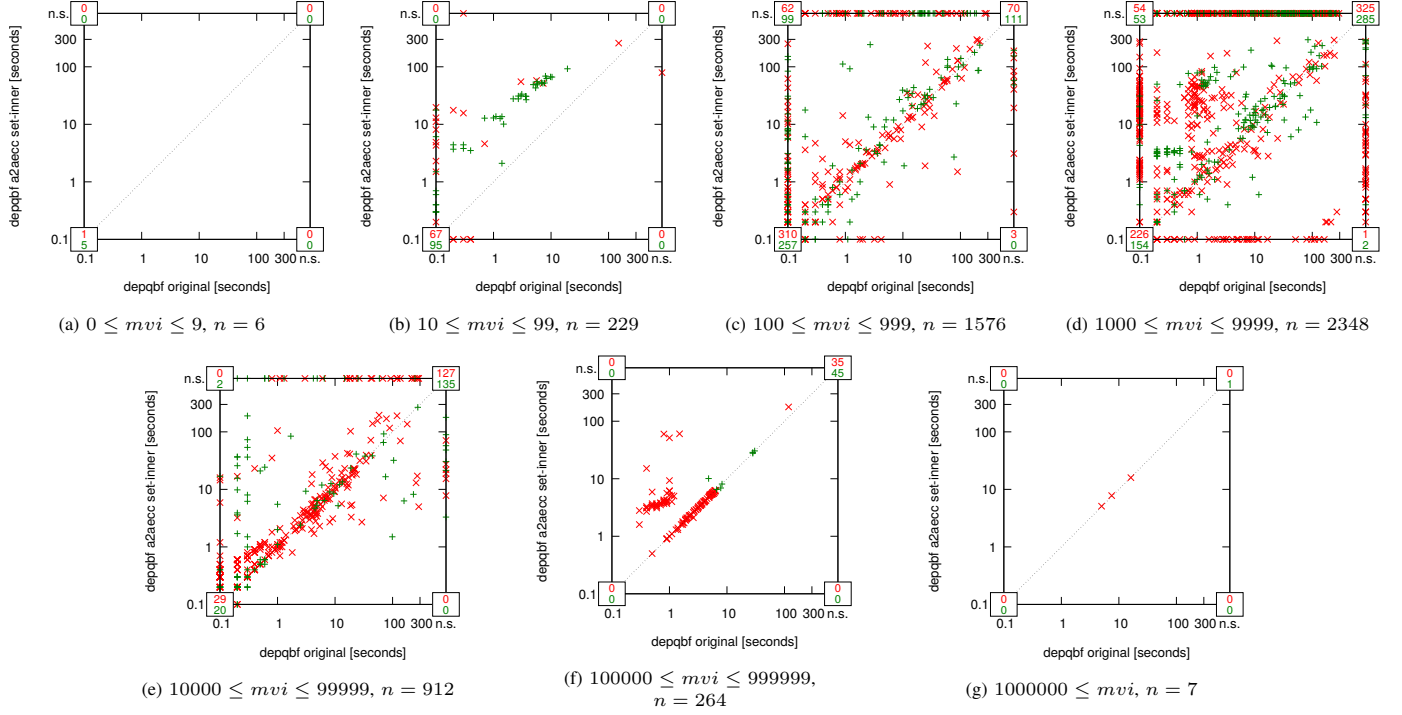


Fig. 1815: Solver DepQBF: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by maximum variable index (run time in seconds).

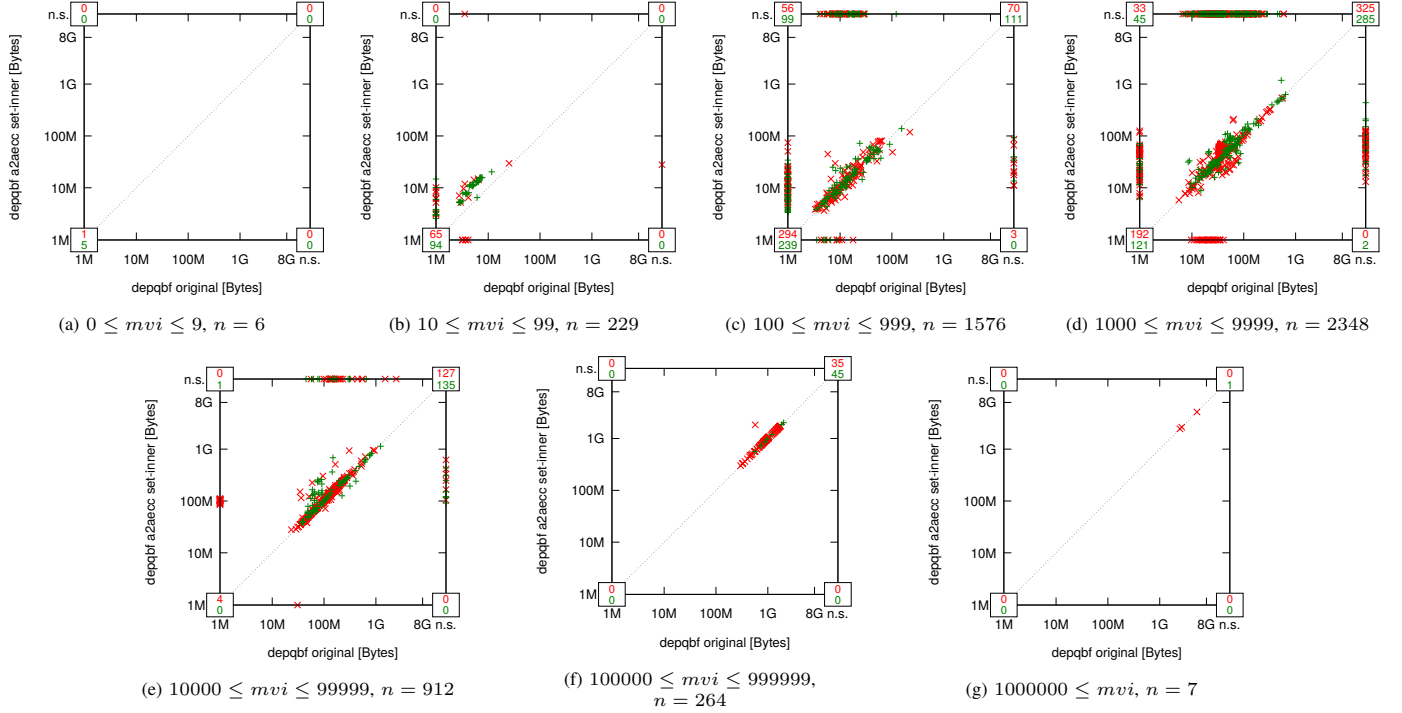


Fig. 1816: Solver DepQBF: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by maximum variable index (memory usage in Bytes).

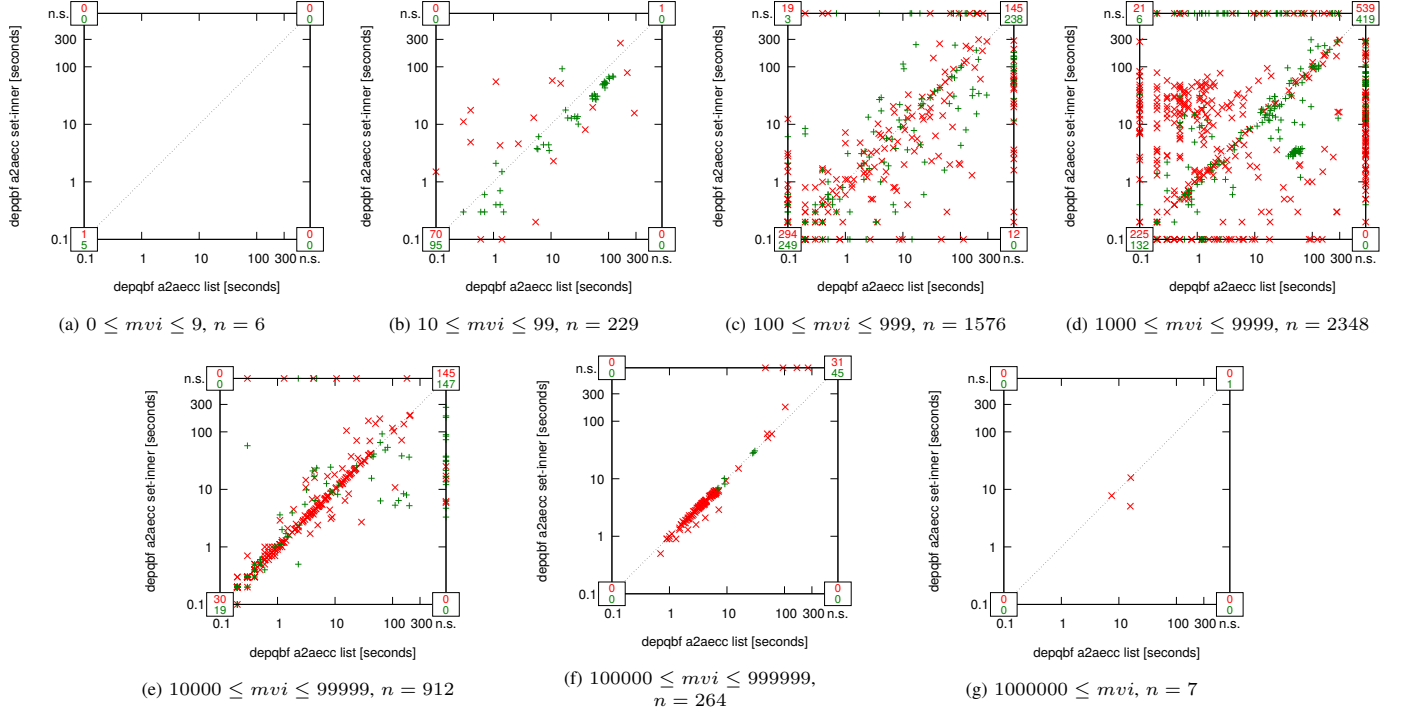


Fig. 1817: Solver DepQBF : Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by maximum variable index (run time in seconds).

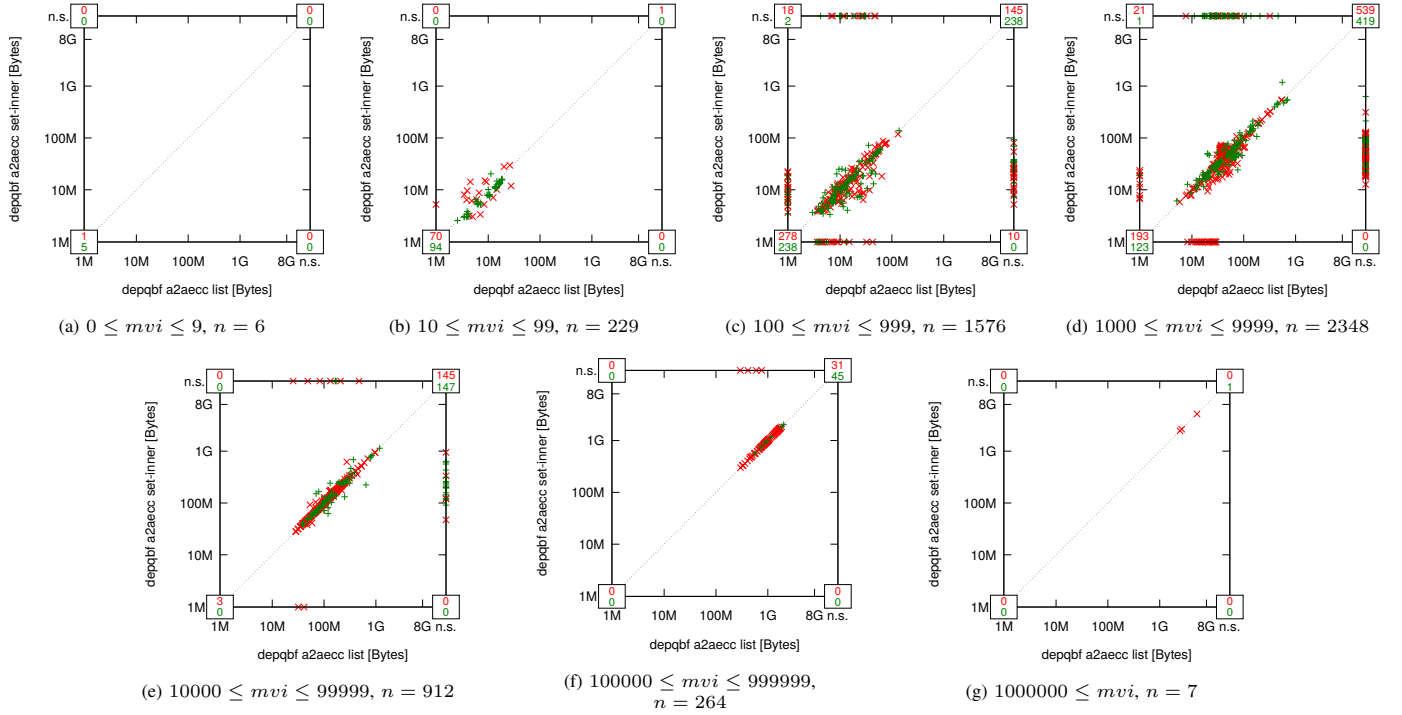


Fig. 1818: Solver DepQBF : Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by maximum variable index (memory usage in Bytes).

b) AIGSolve:

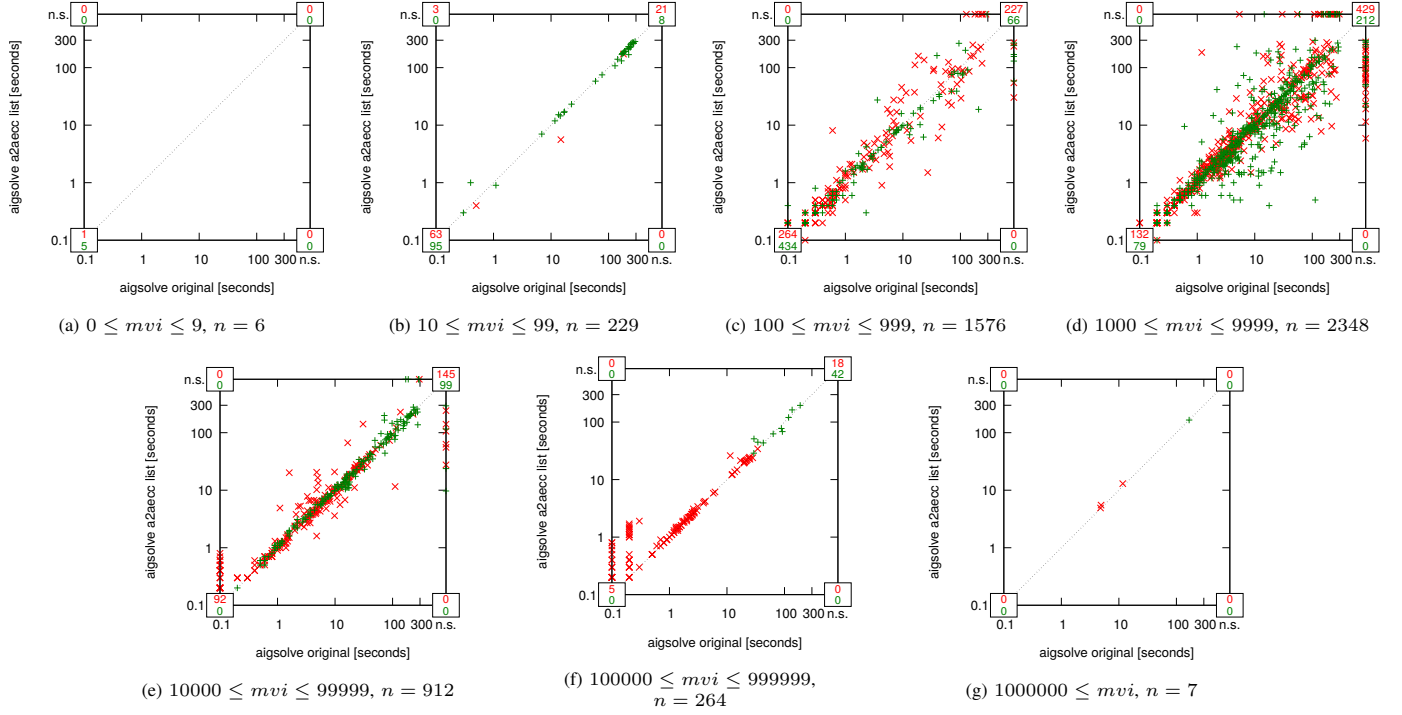


Fig. 1819: Solver AIGSolve: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by maximum variable index (run time in seconds).

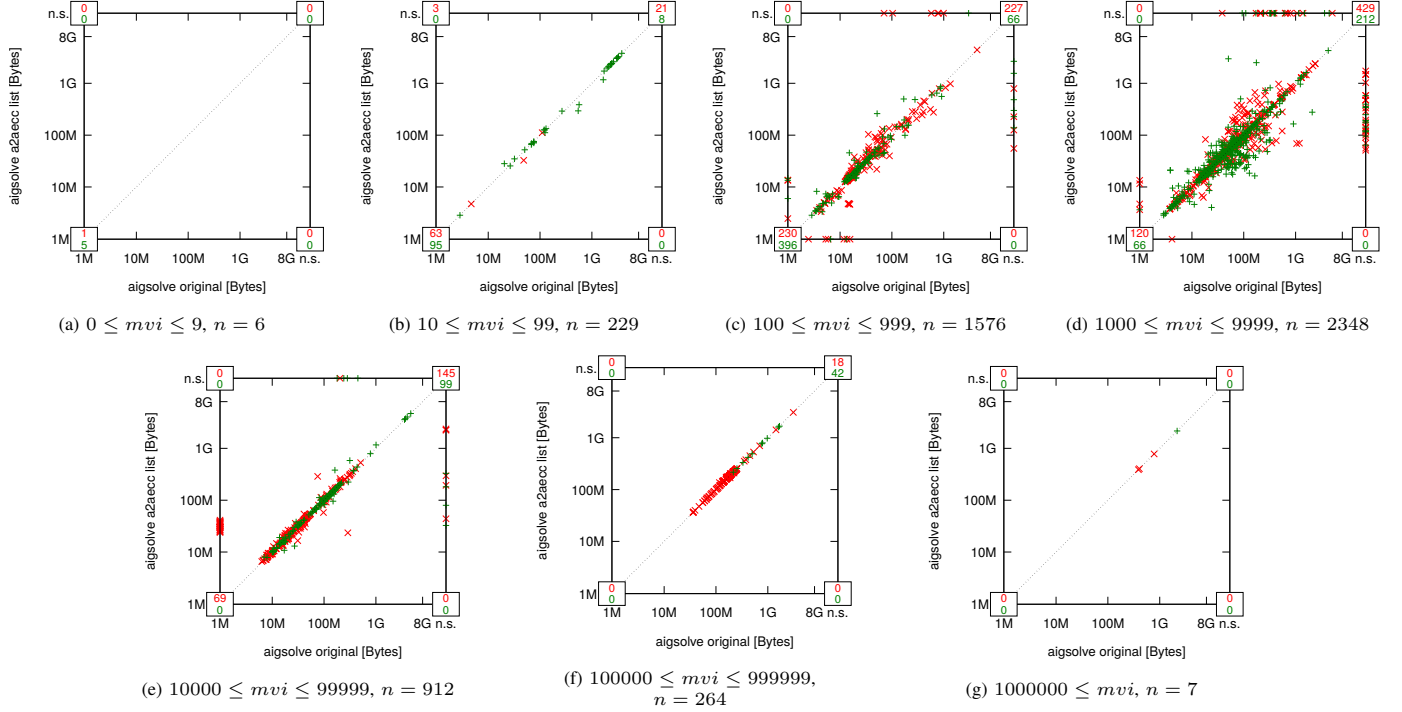


Fig. 1820: Solver AIGSolve: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by maximum variable index (memory usage in Bytes).

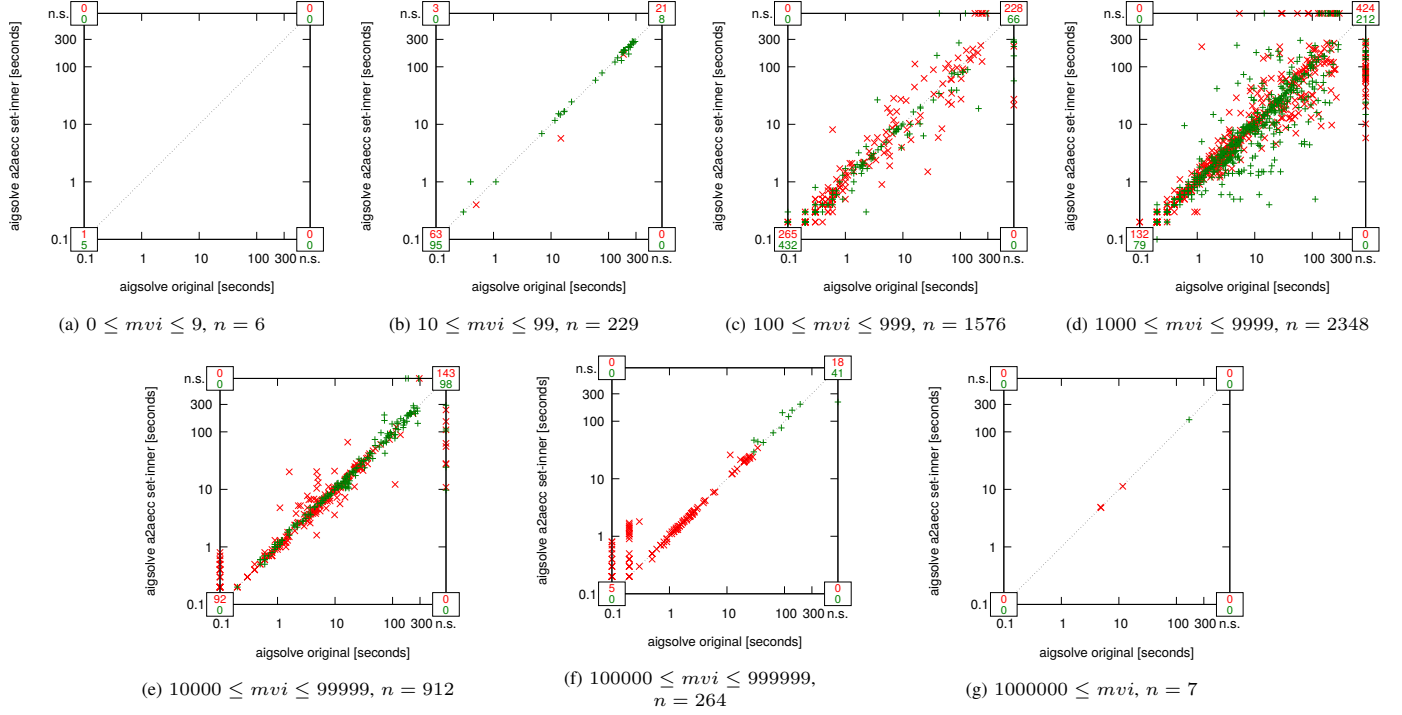


Fig. 1821: Solver AIGSolve: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by maximum variable index (run time in seconds).

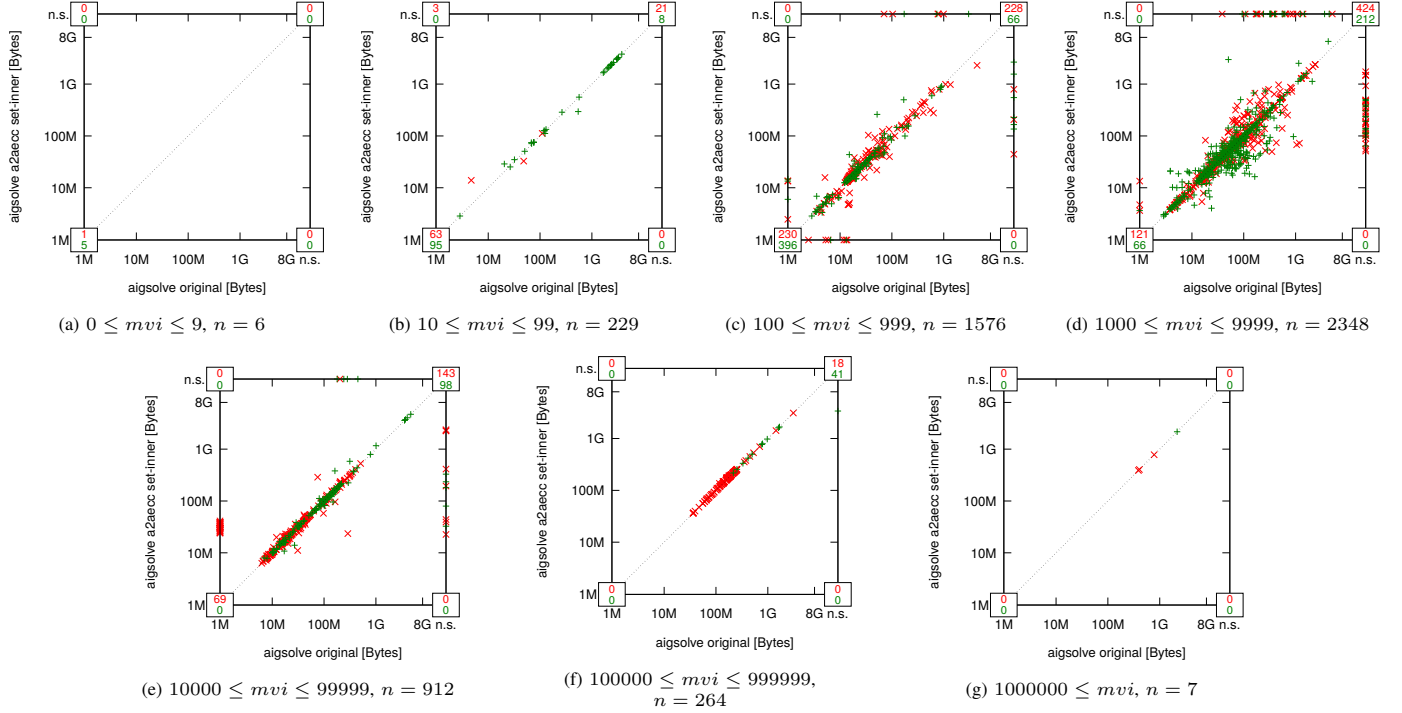


Fig. 1822: Solver AIGSolve: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by maximum variable index (memory usage in Bytes).

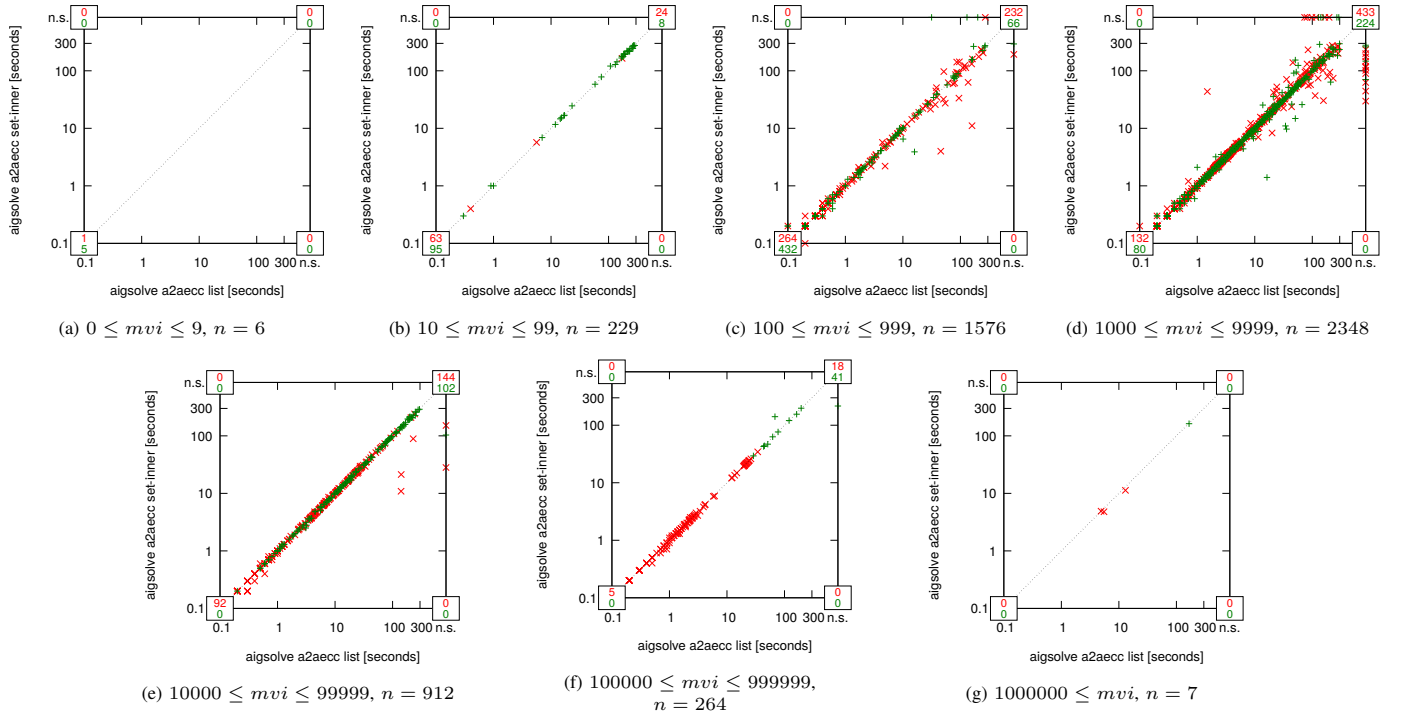


Fig. 1823: Solver AIGSolve: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by maximum variable index (run time in seconds).

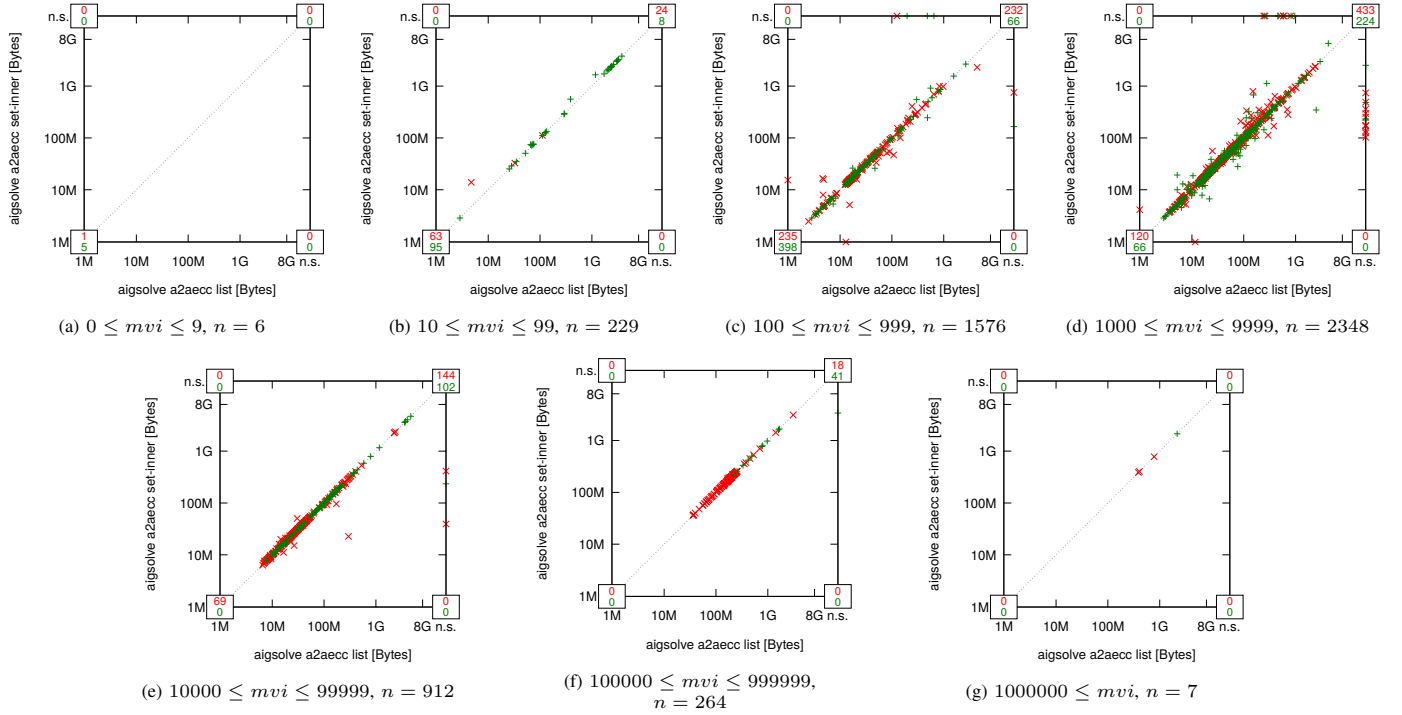


Fig. 1824: Solver AIGSolve: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by maximum variable index (memory usage in Bytes).

c) CAQE:

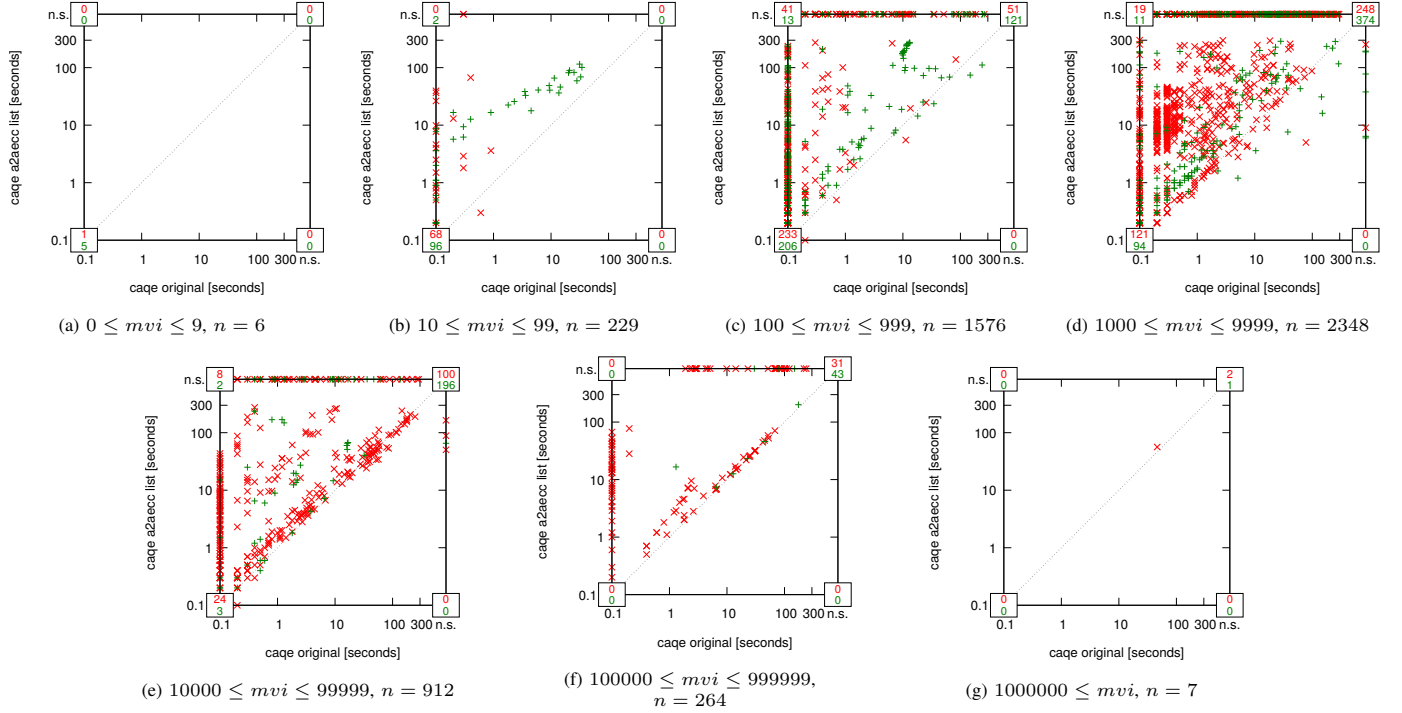


Fig. 1825: Solver CAQE: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by maximum variable index (run time in seconds).

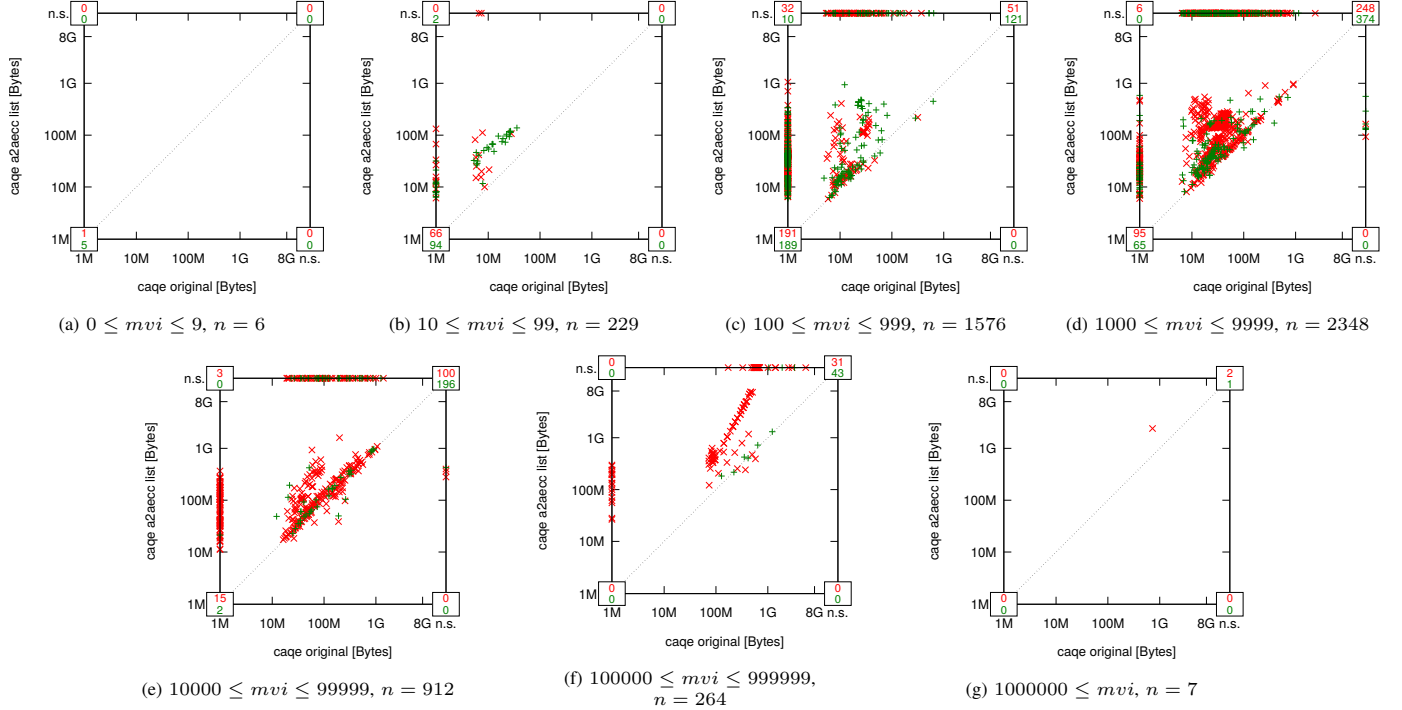


Fig. 1826: Solver CAQE: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by maximum variable index (memory usage in Bytes).

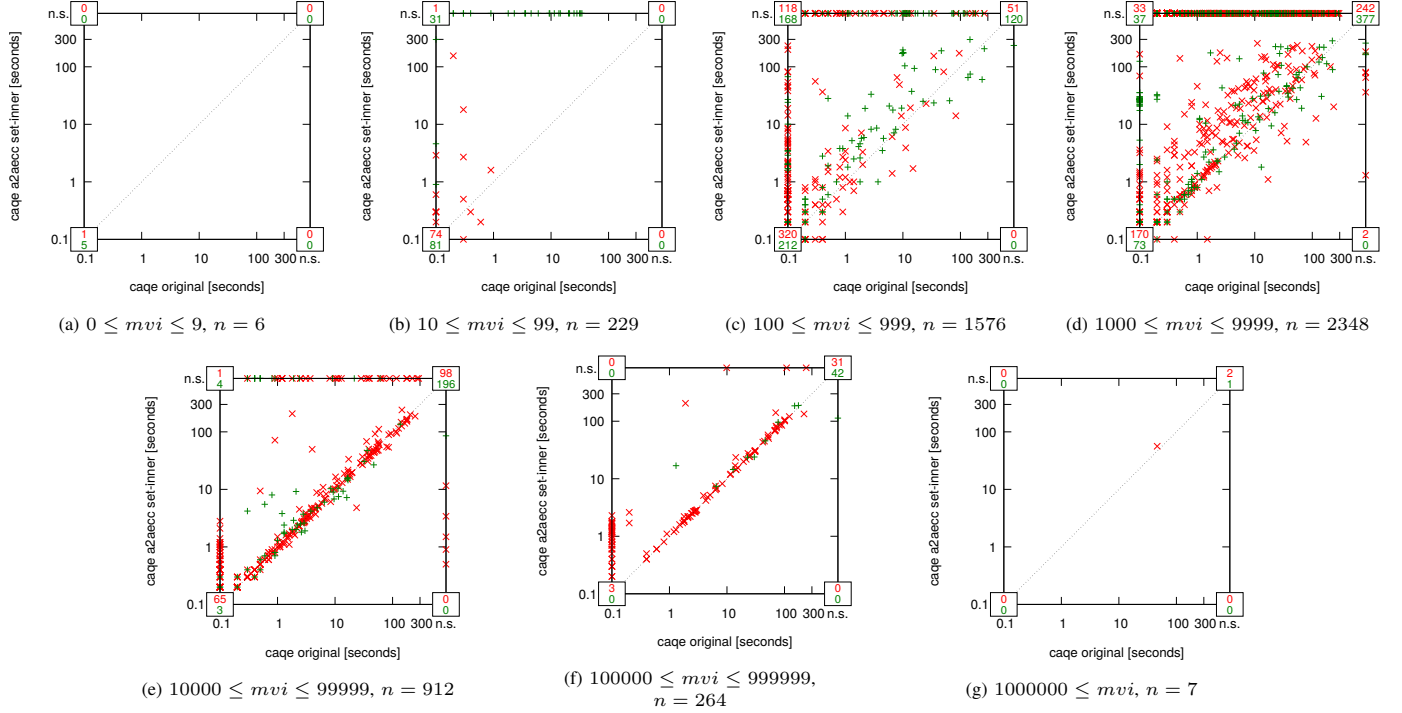


Fig. 1827: Solver CAQE: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by maximum variable index (run time in seconds).

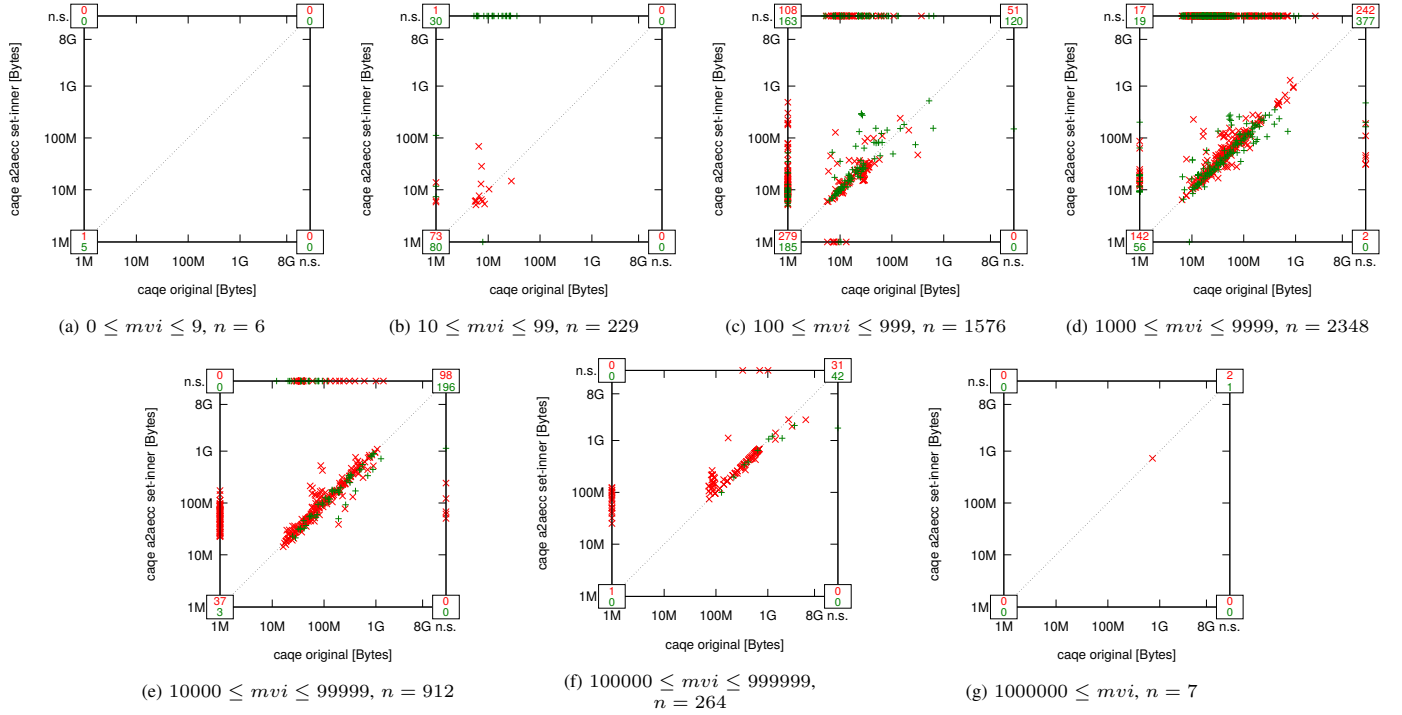


Fig. 1828: Solver CAQE: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by maximum variable index (memory usage in Bytes).

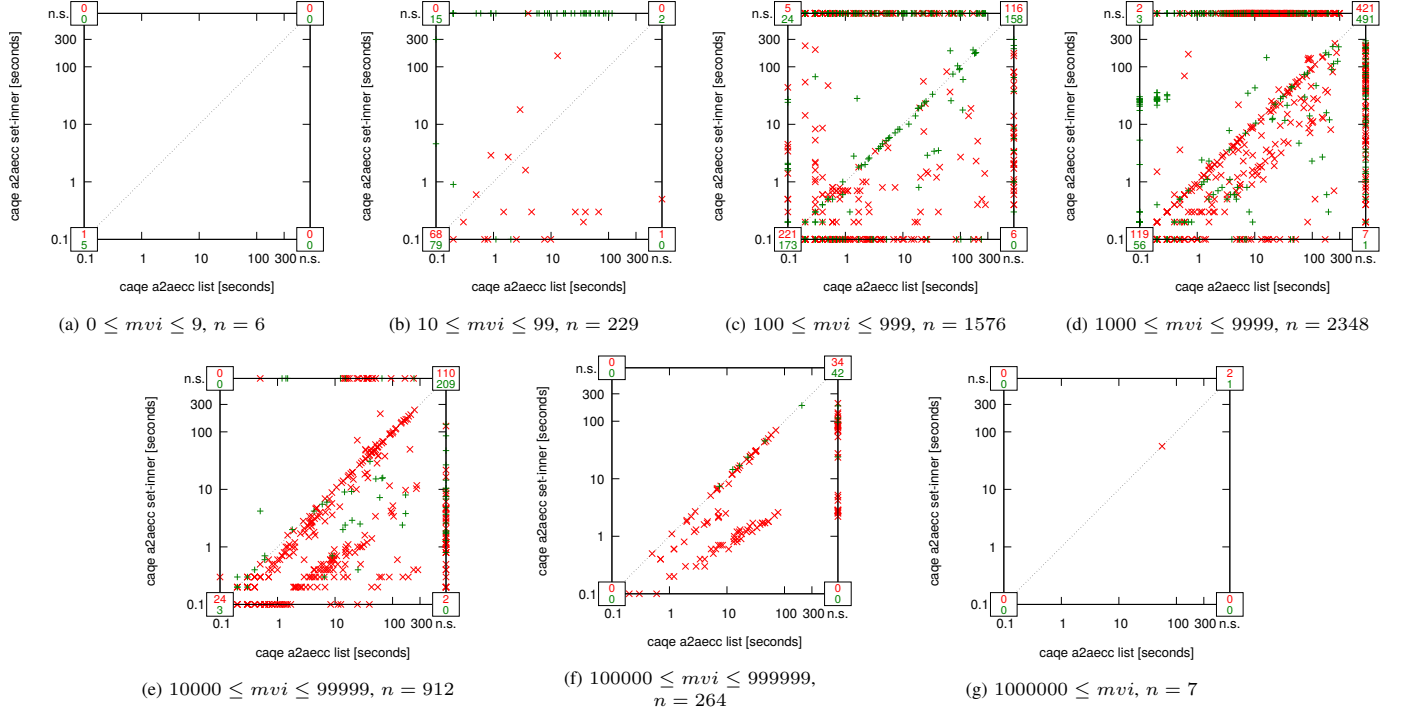


Fig. 1829: Solver CAQE: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by maximum variable index (run time in seconds).

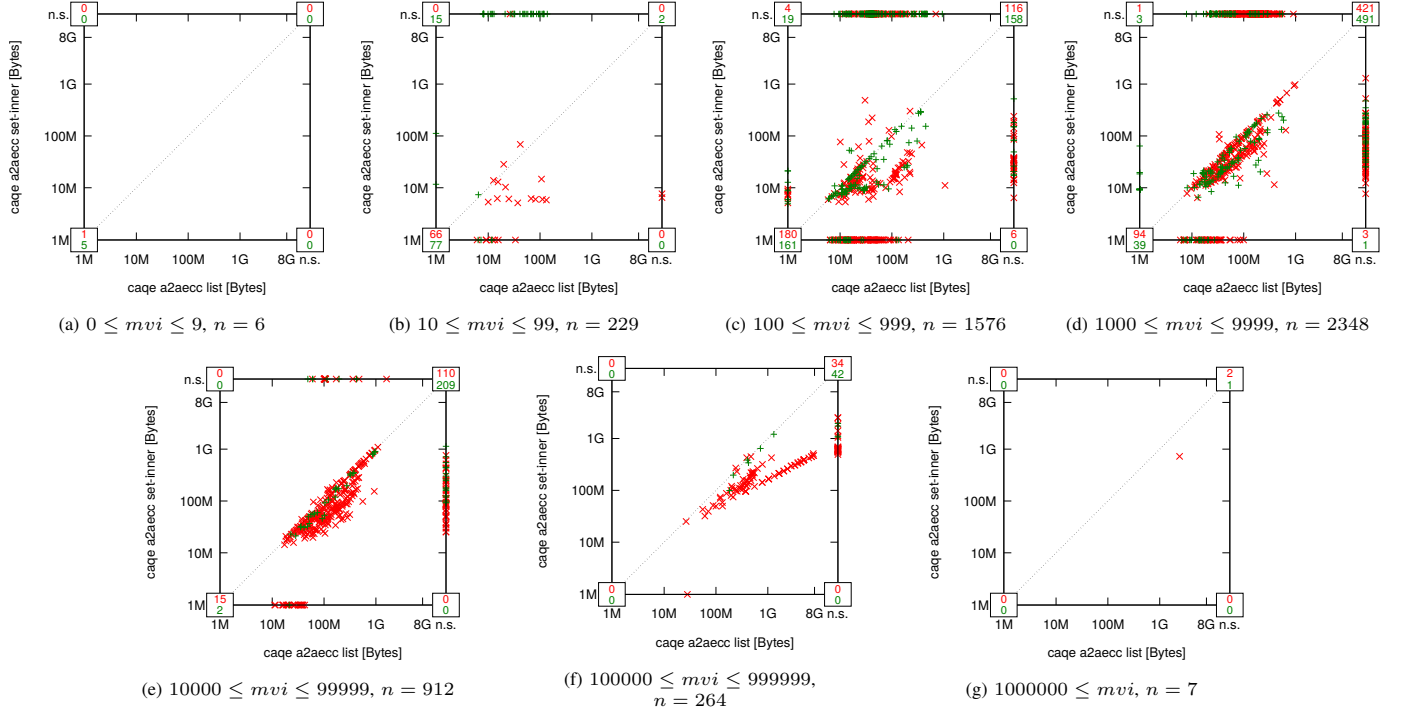


Fig. 1830: Solver CAQE: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by maximum variable index (memory usage in Bytes).

d) GhostQ:

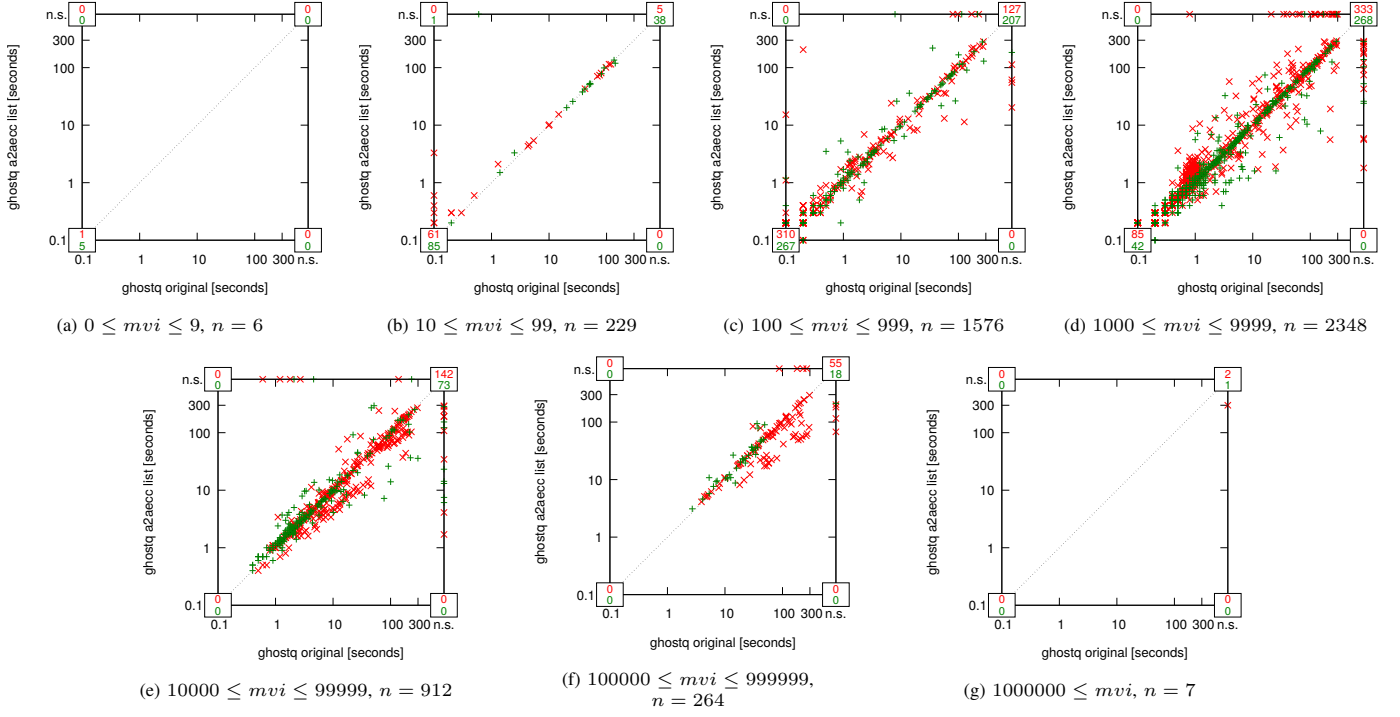


Fig. 1831: Solver GhostQ: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by maximum variable index (run time in seconds).

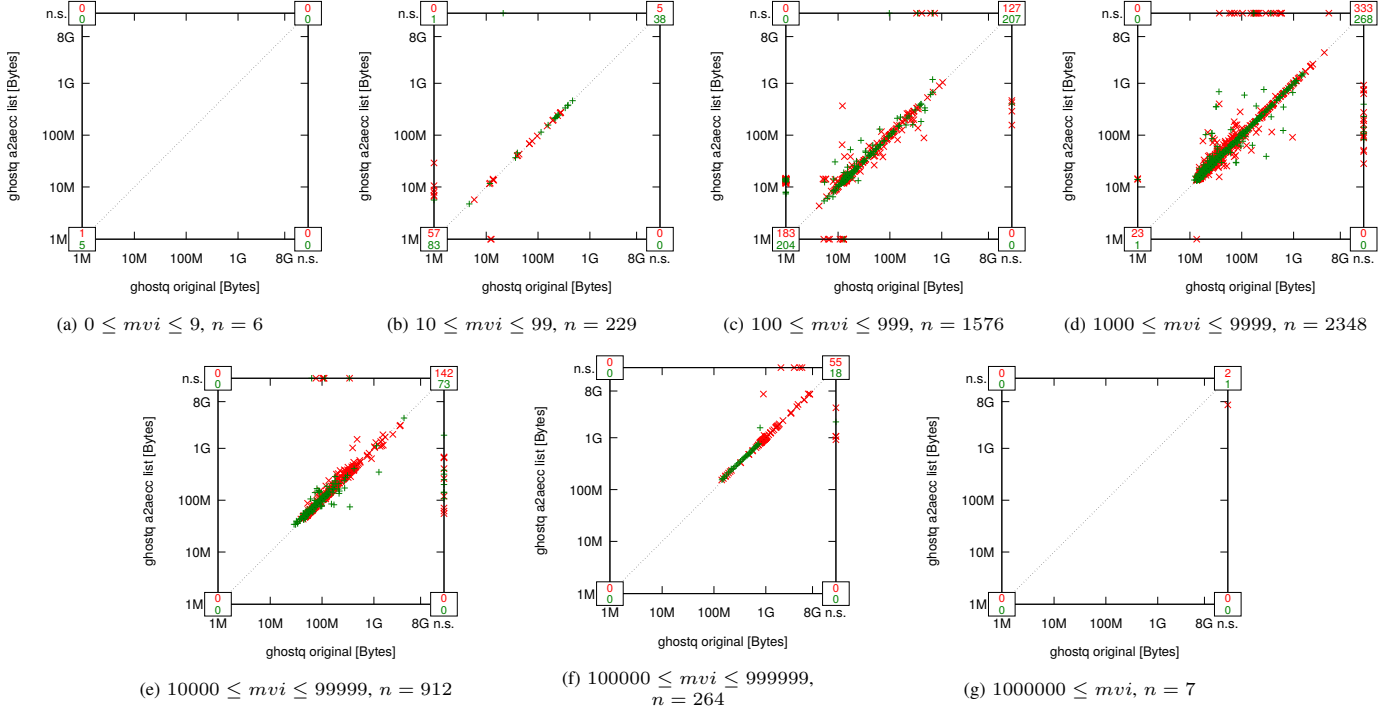


Fig. 1832: Solver GhostQ: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by maximum variable index (memory usage in Bytes).

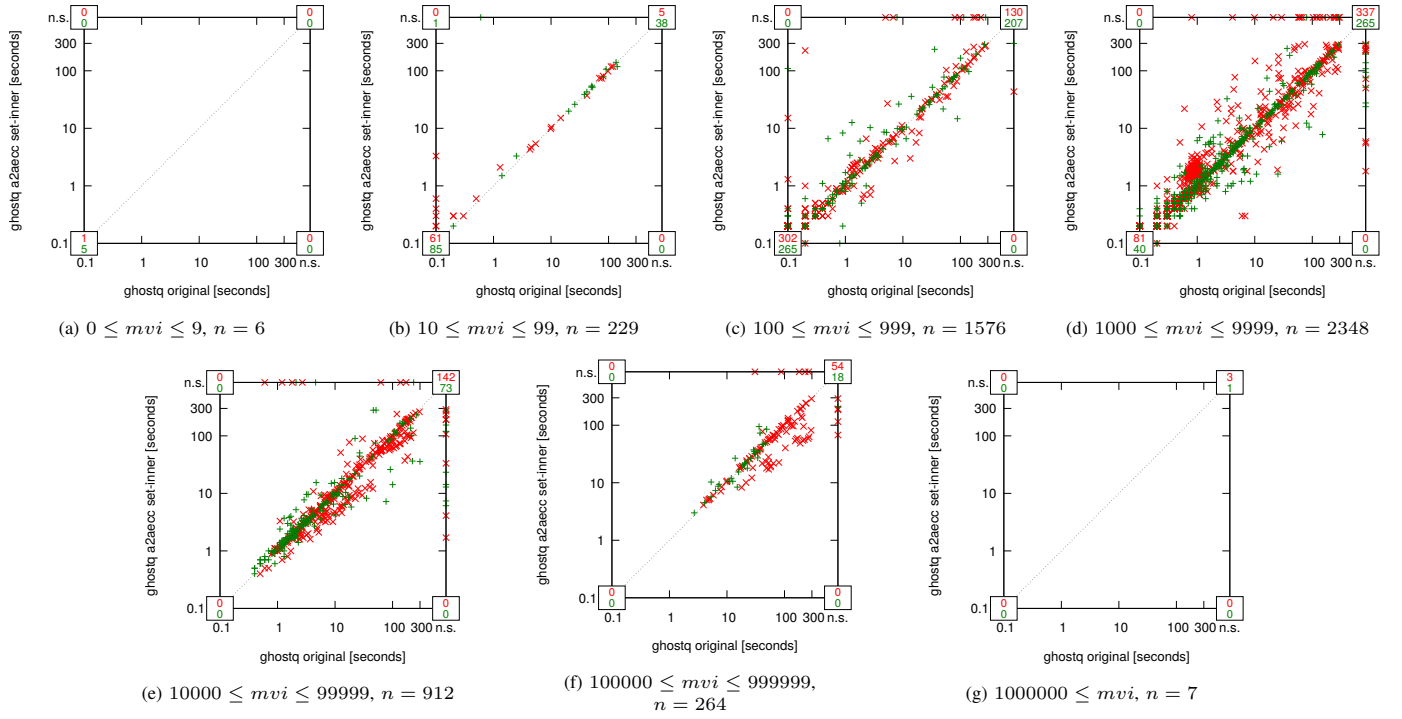


Fig. 1833: Solver GhostQ: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by maximum variable index (run time in seconds).

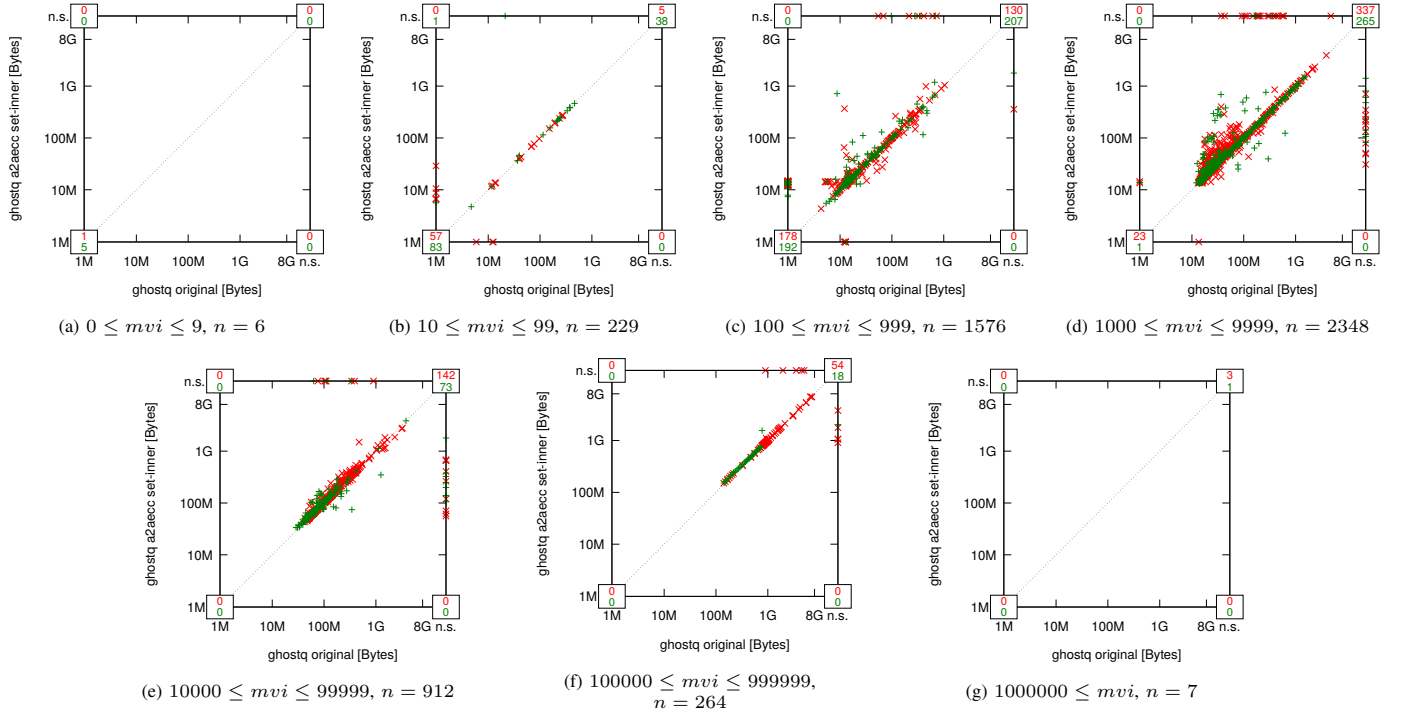


Fig. 1834: Solver GhostQ: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by maximum variable index (memory usage in Bytes).

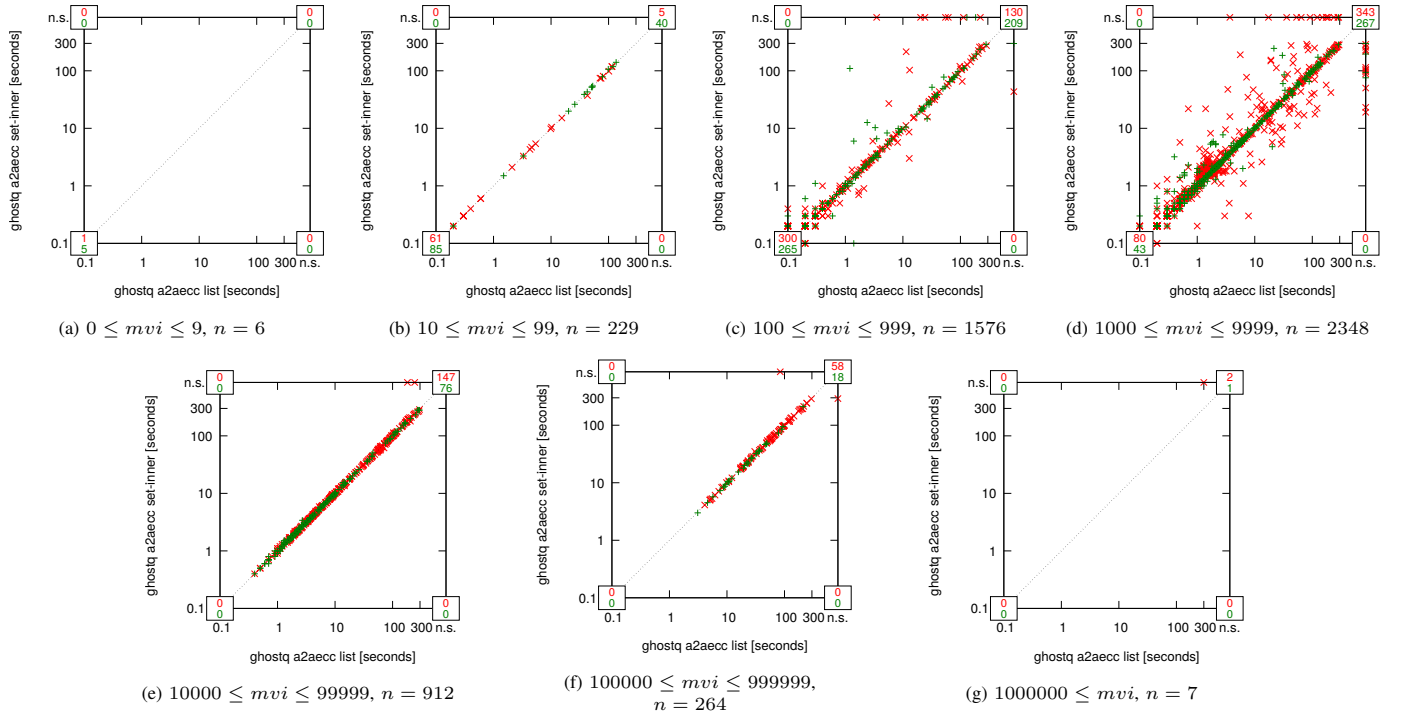


Fig. 1835: Solver GhostQ: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by maximum variable index (run time in seconds).

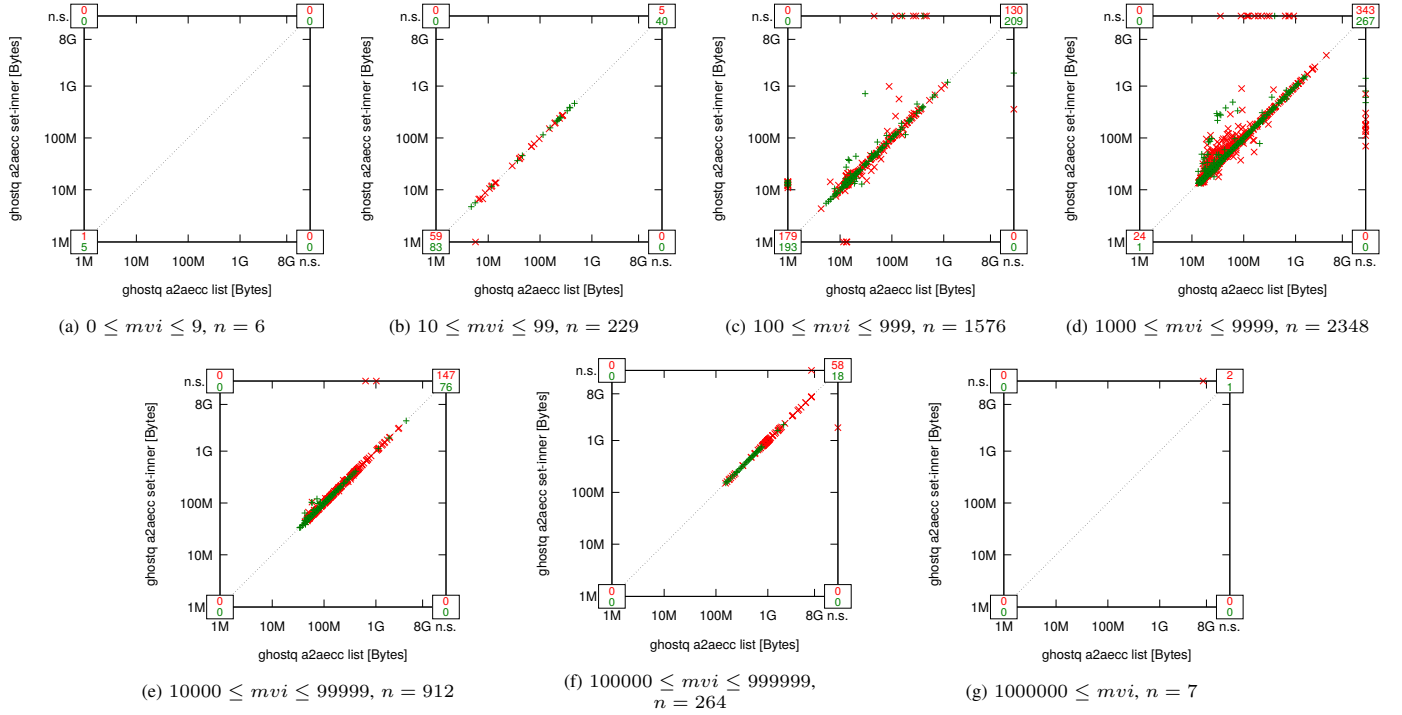


Fig. 1836: Solver GhostQ: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by maximum variable index (memory usage in Bytes).

e) QESTO:

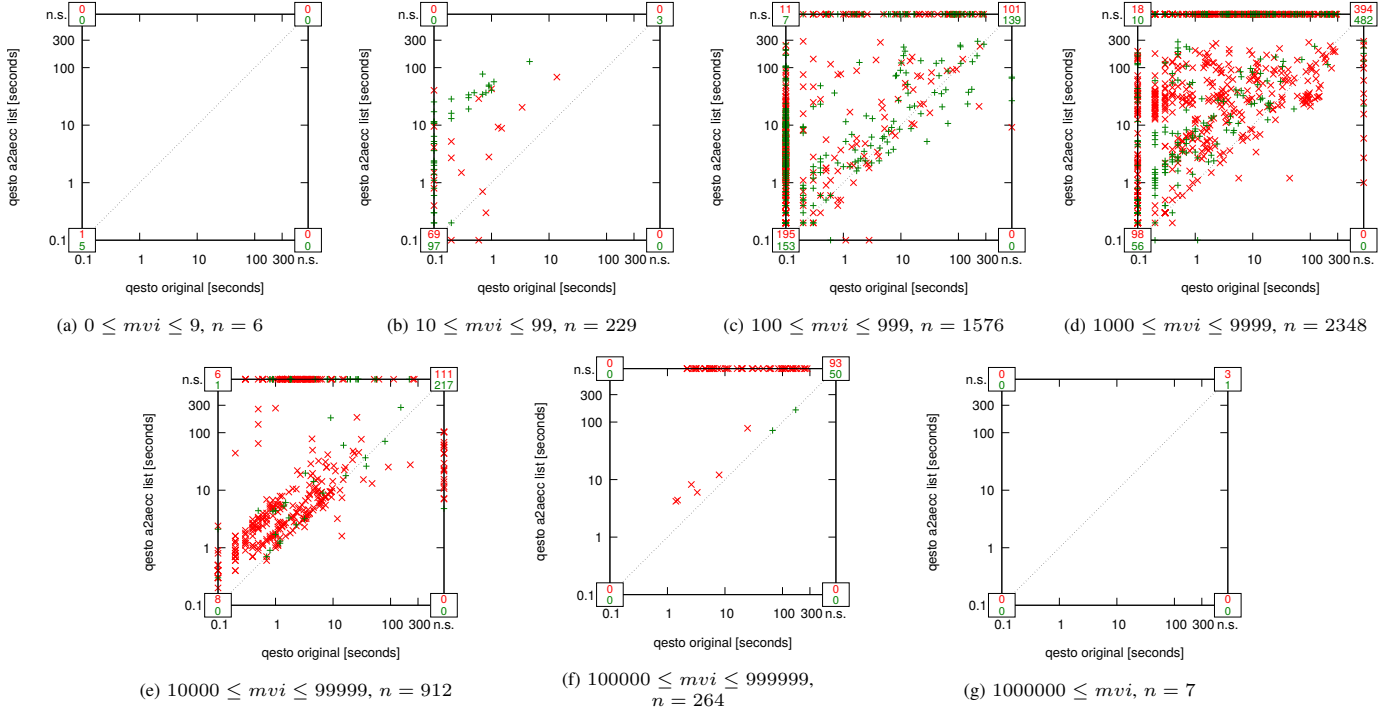


Fig. 1837: Solver QESTO: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by maximum variable index (run time in seconds).

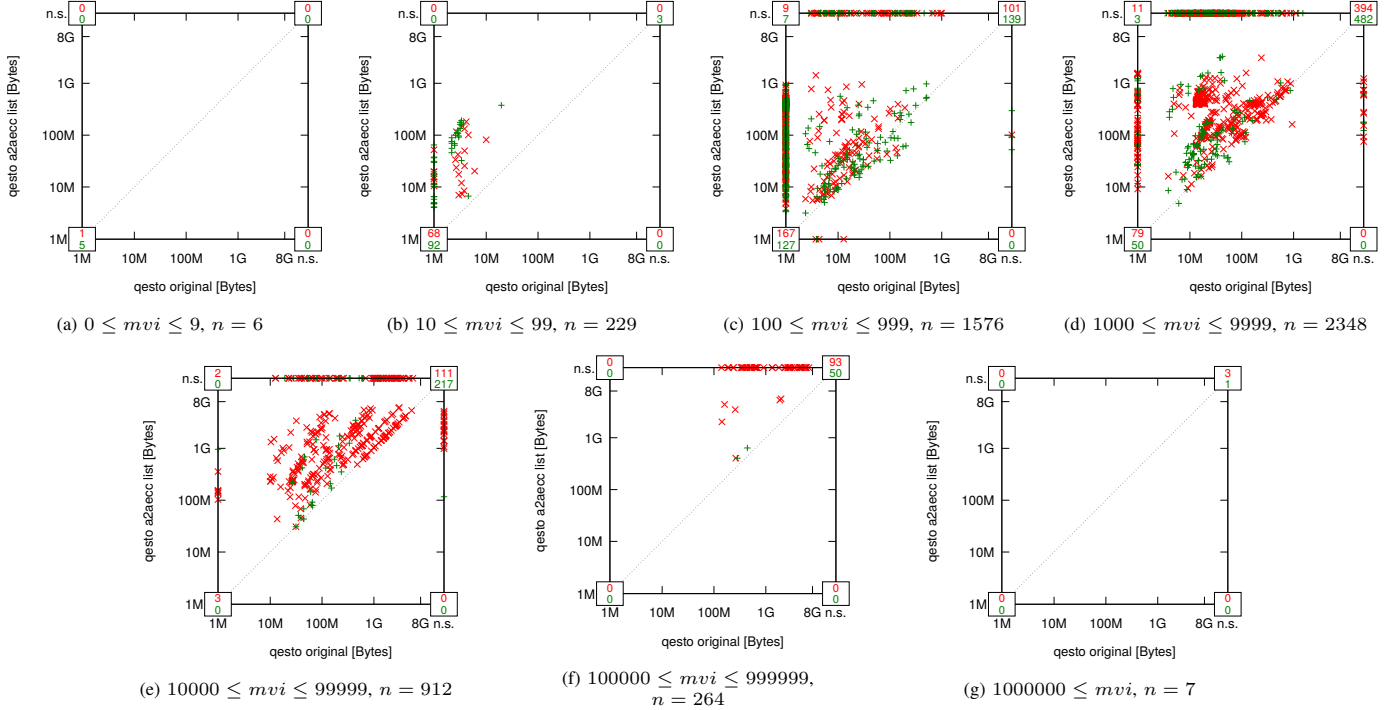


Fig. 1838: Solver QESTO: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by maximum variable index (memory usage in Bytes).

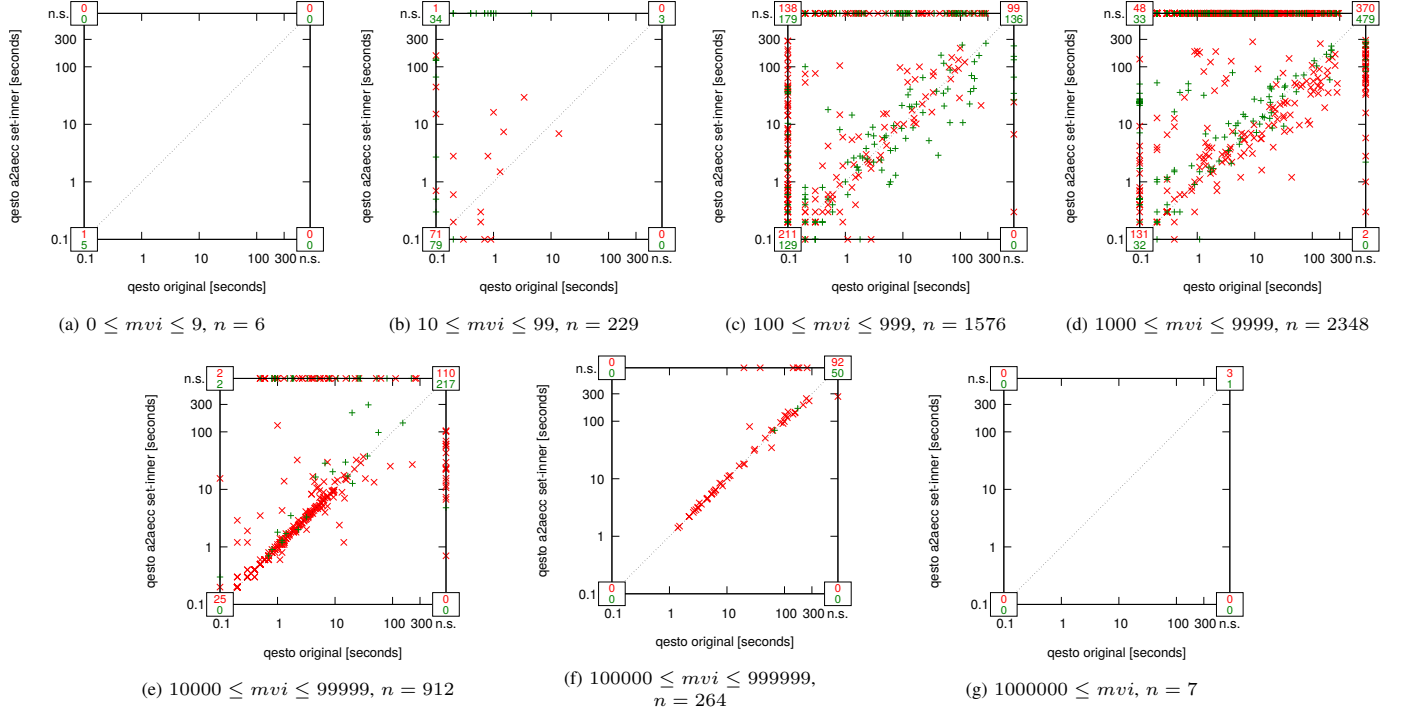


Fig. 1839: Solver QESTO: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by maximum variable index (run time in seconds).

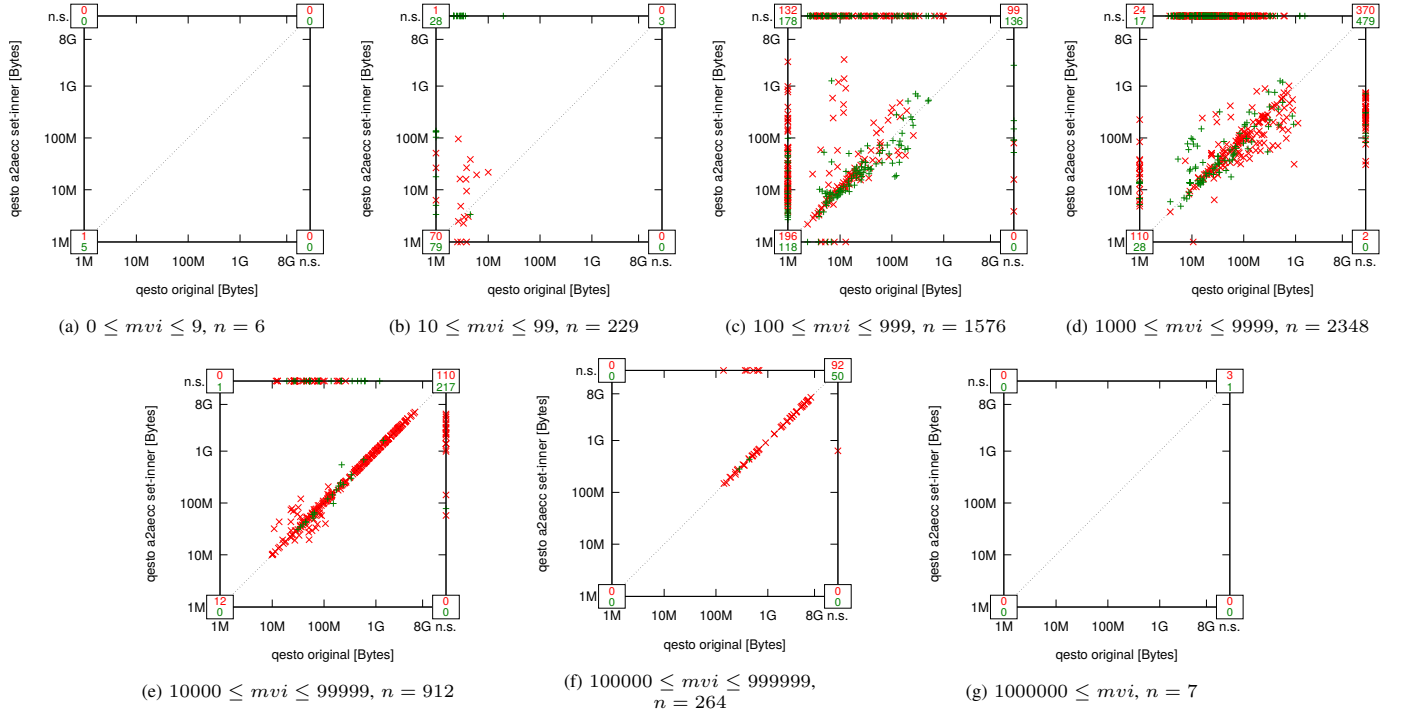


Fig. 1840: Solver QESTO: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by maximum variable index (memory usage in Bytes).

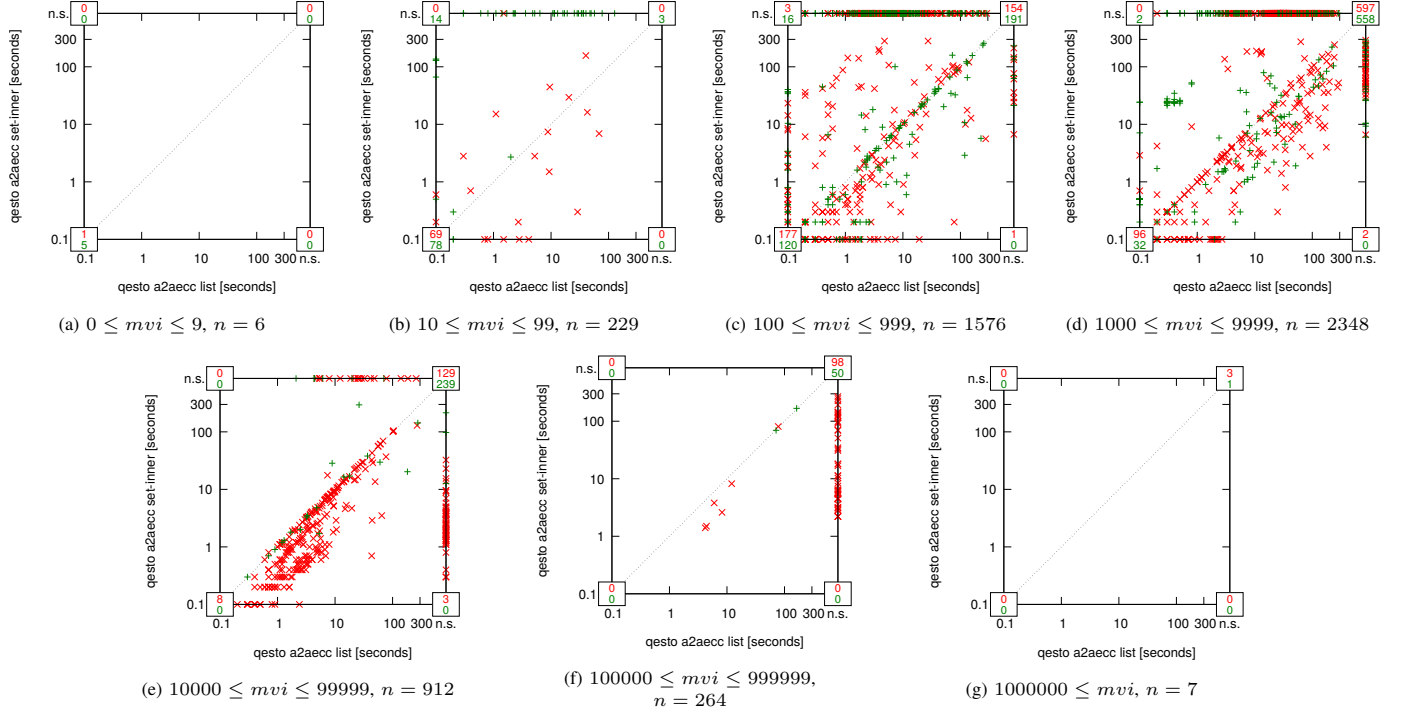


Fig. 1841: Solver QESTO: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by maximum variable index (run time in seconds).

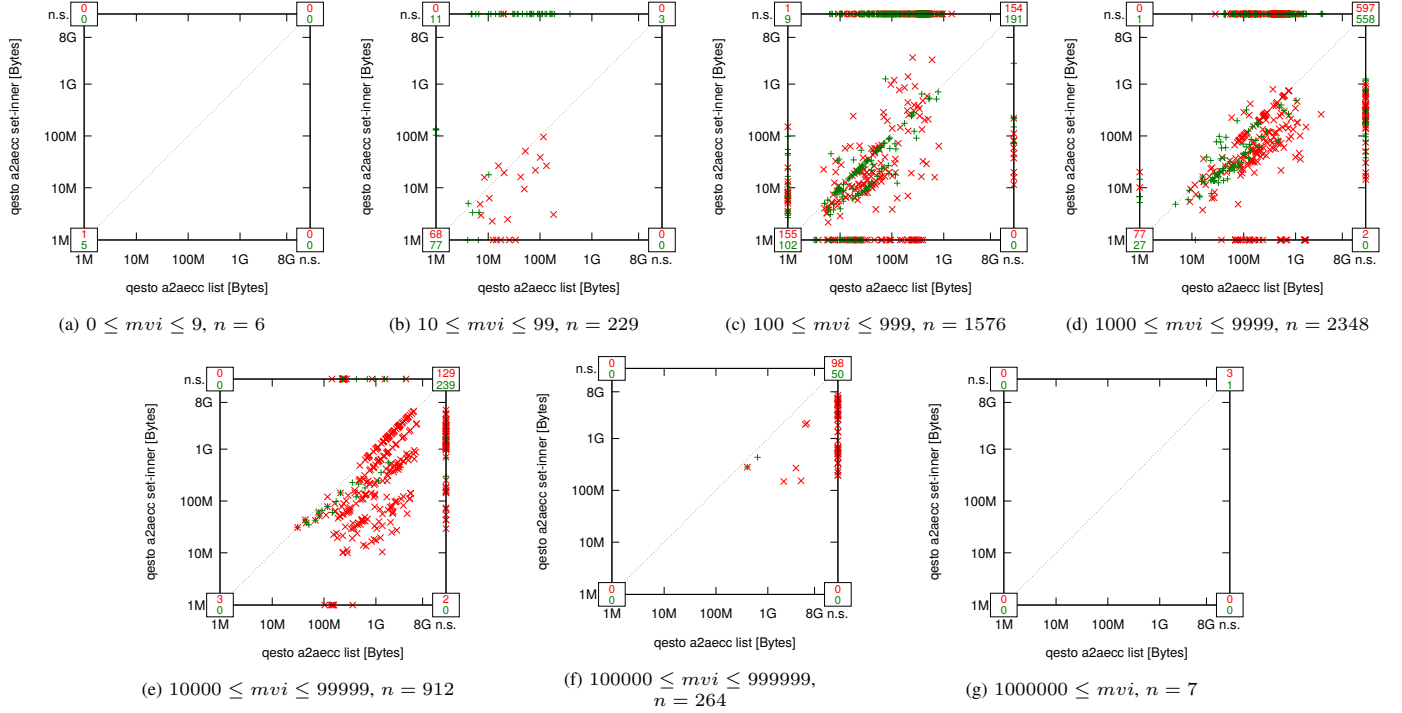


Fig. 1842: Solver QESTO: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by maximum variable index (memory usage in Bytes).

f) RAReQS:

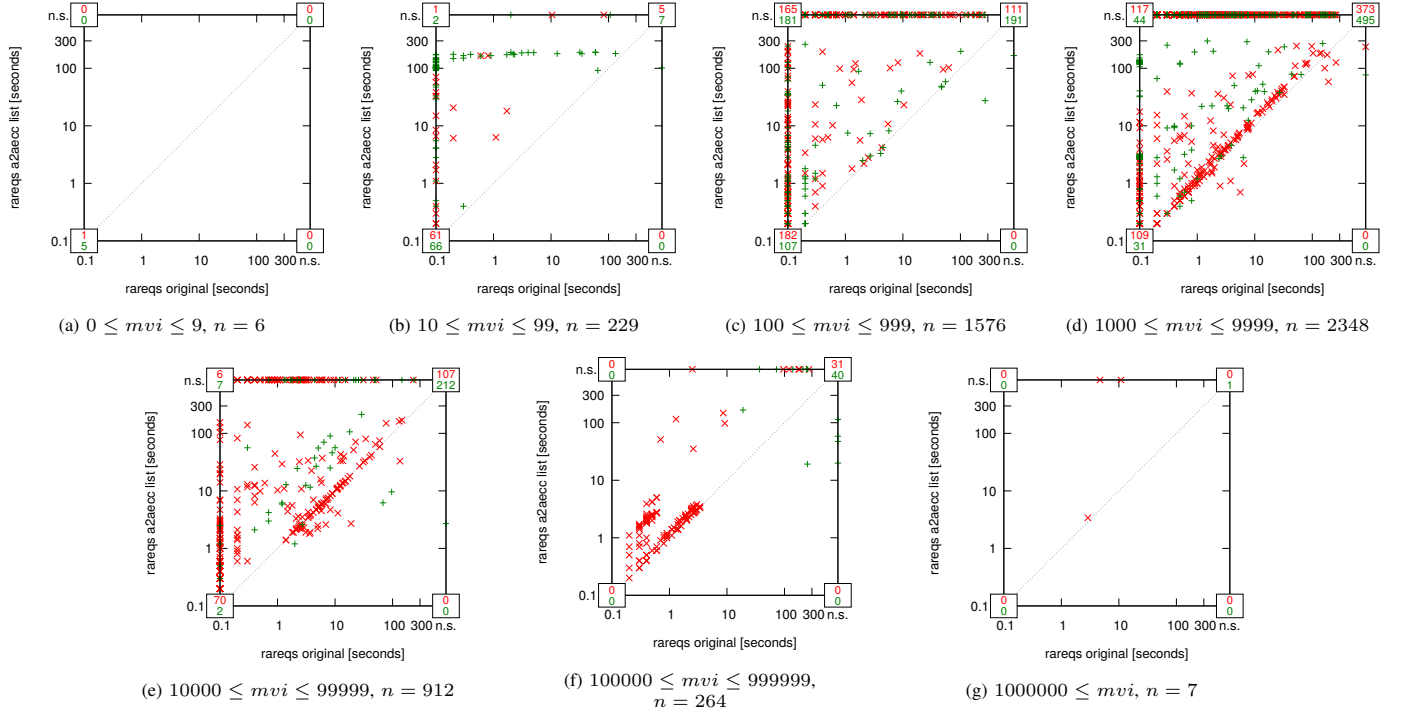


Fig. 1843: Solver RAReQS: Comparing run times for solving the transformed versus the original instances with list semantics partitioned by maximum variable index (run time in seconds).

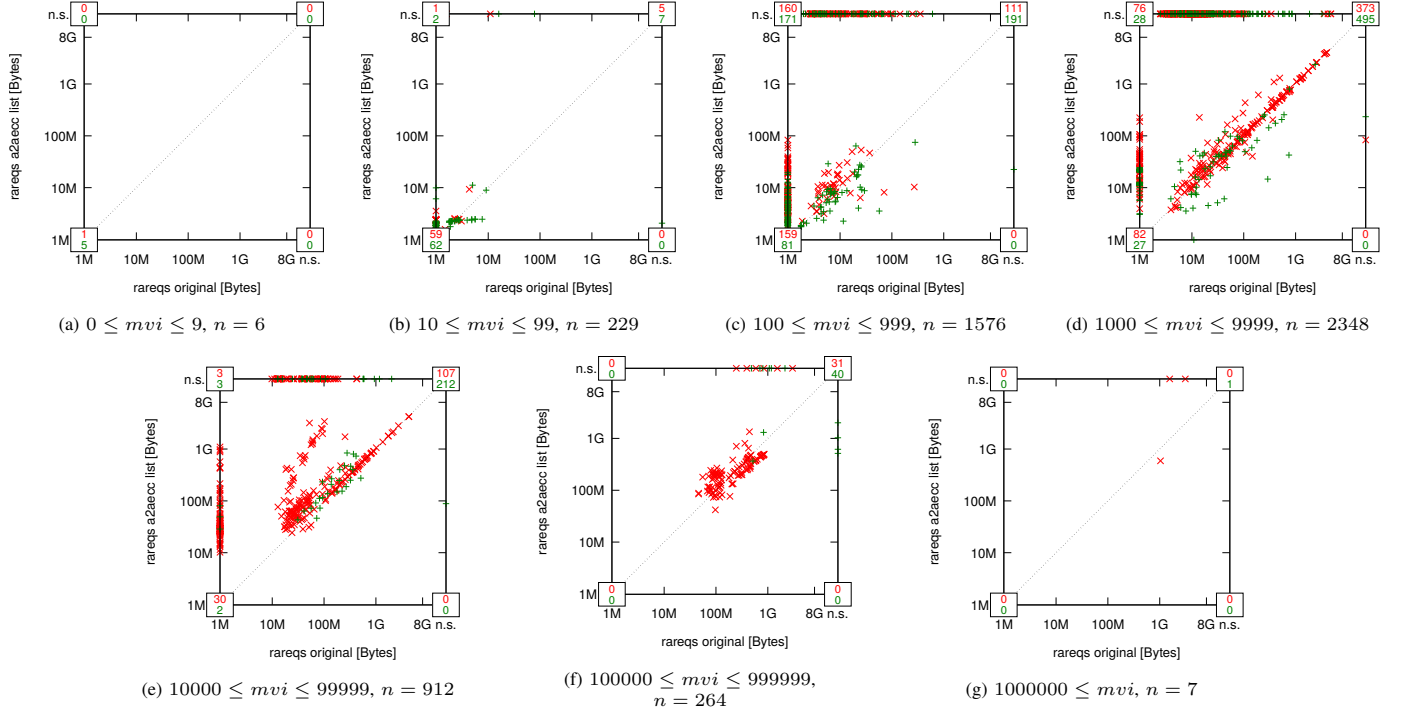


Fig. 1844: Solver RAReQS: Comparing memory usage for solving the transformed versus the original instances with list semantics partitioned by maximum variable index (memory usage in Bytes).

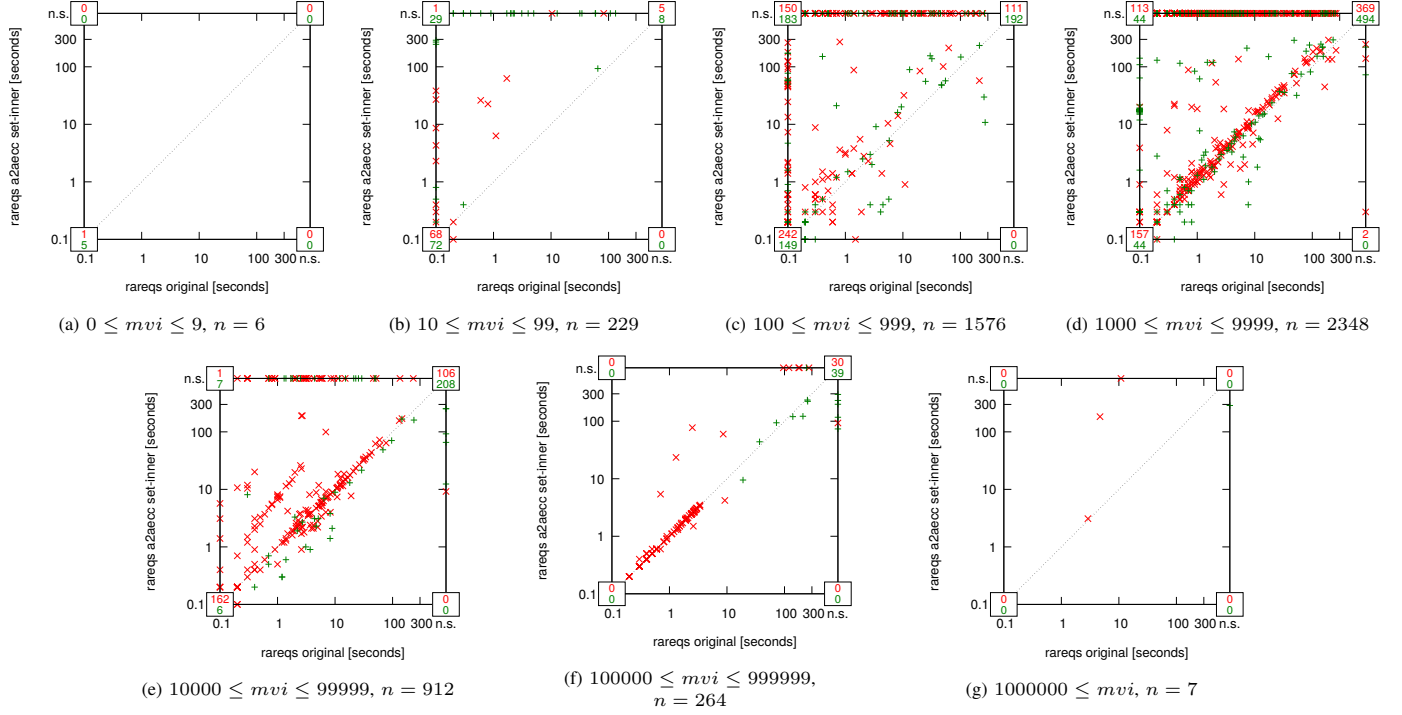


Fig. 1845: Solver RAREQS: Comparing run times for solving the transformed versus the original instances with set-inner semantics partitioned by maximum variable index (run time in seconds).

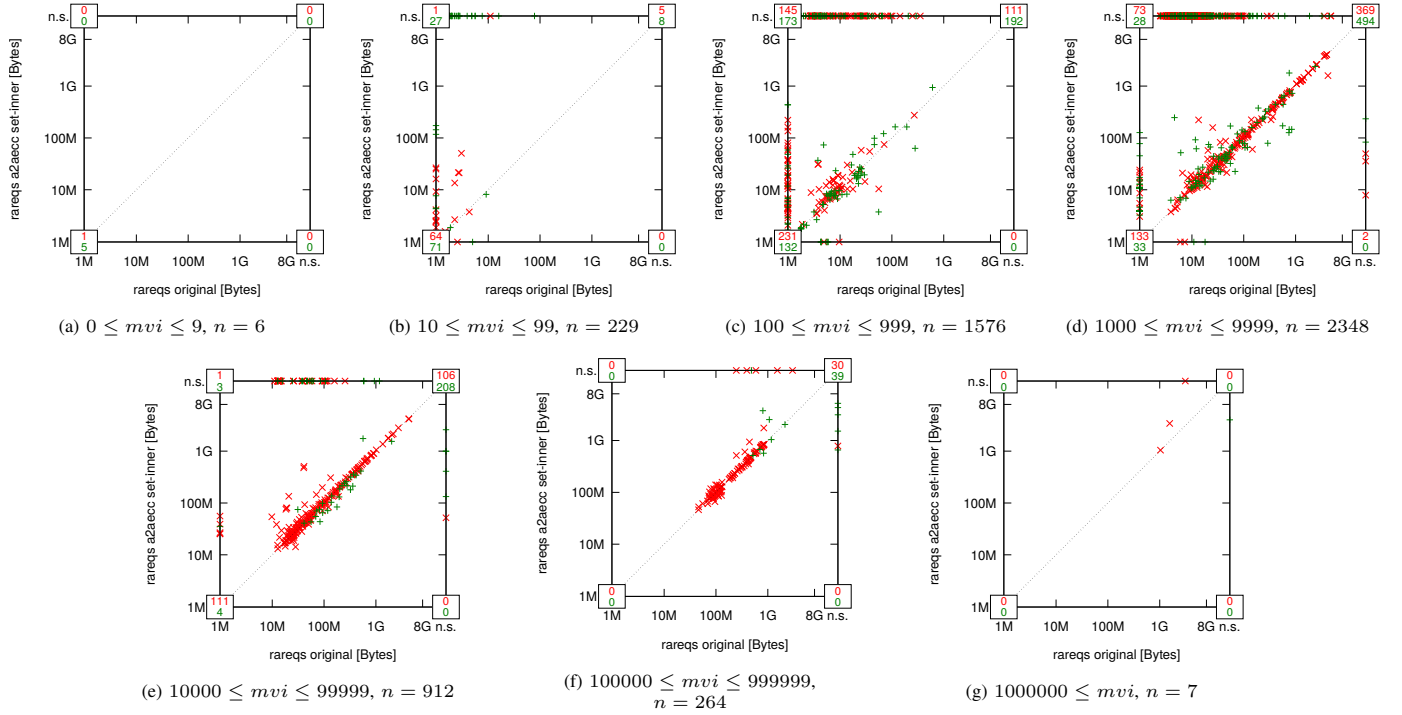


Fig. 1846: Solver RAREQS: Comparing memory usage for solving the transformed versus the original instances with set-inner semantics partitioned by maximum variable index (memory usage in Bytes).

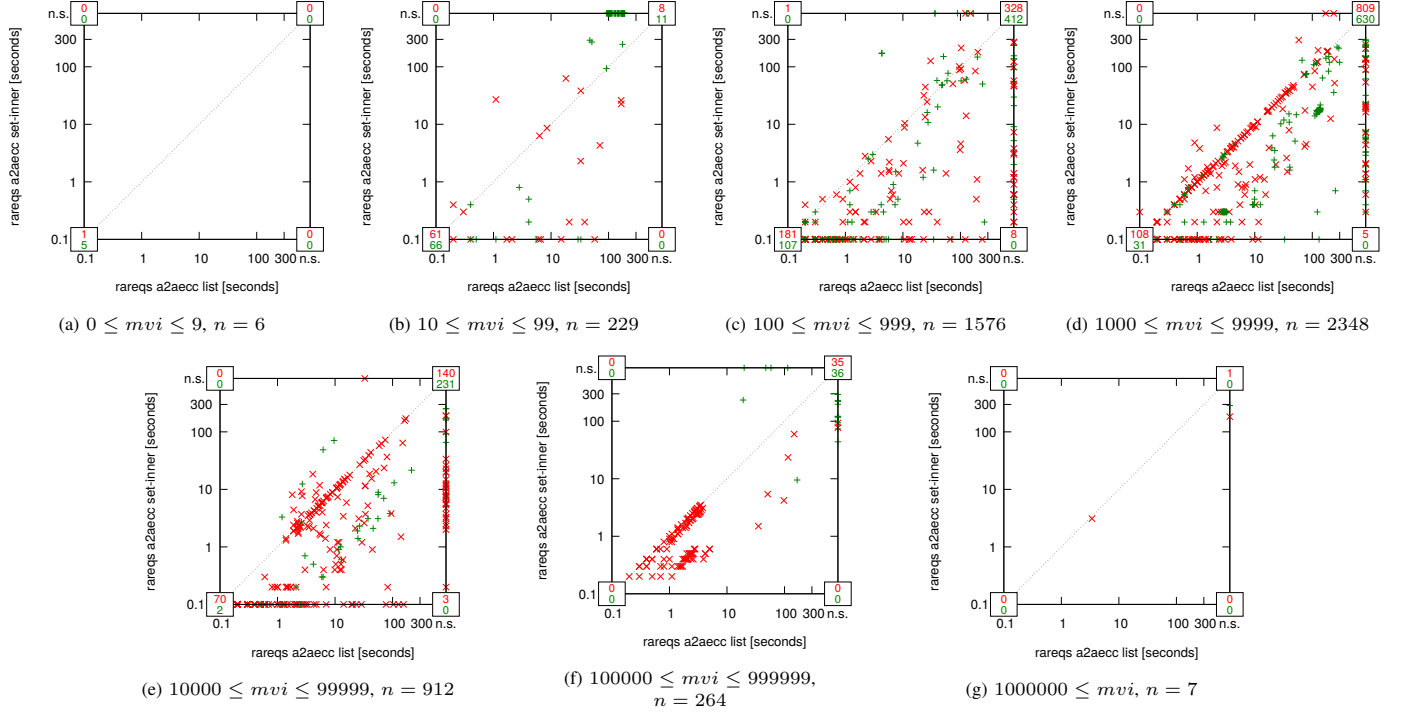


Fig. 1847: Solver RAReQS: Comparing run times for solving the transformed instances in set-inner versus list semantics partitioned by maximum variable index (run time in seconds).

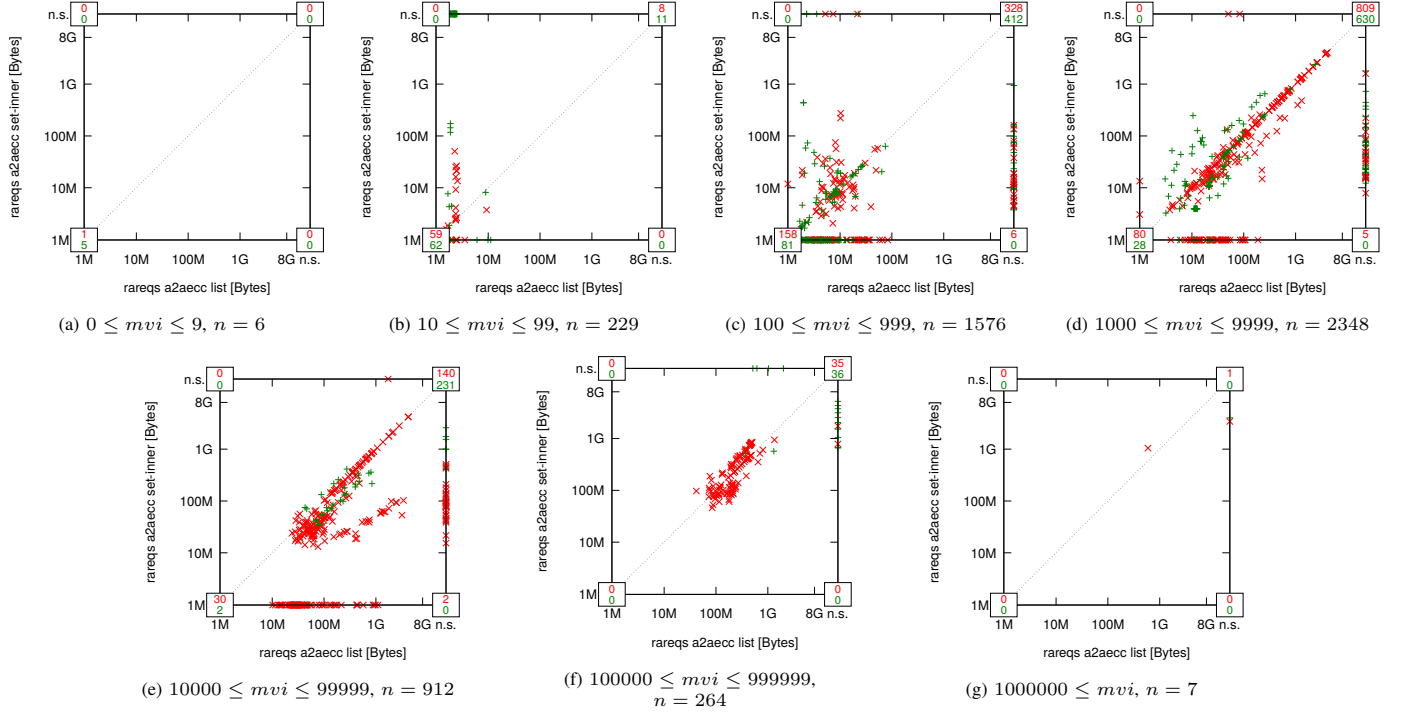


Fig. 1848: Solver RAReQS: Comparing memory usage for solving the transformed instances in set-inner versus list semantics partitioned by maximum variable index (memory usage in Bytes).