Treewidth in Industrial SAT Benchmarks

Robert Mateescu
Microsoft Research
7 J J Thomson Avenue
Cambridge CB3 0FB, UK
romatees@microsoft.com

MSR-TR-2011-22

Abstract

The Boolean Satisfiability (SAT) solvers have been in recent years very successful at solving very large practical problems, with hundreds of thousands of variables and millions of clauses. Although SAT can be viewed as a graphical model, it is intriguing that SAT solvers do not seem to perform anywhere close to the worst case upper bound that is known for graphical models algorithms (e.g., join-tree clustering or variable elimination), which are exponential in the treewidth of the primal graph. In this paper we empirically investigate the correlation between the treewidth of industrial SAT instances from the SAT2009 competition, and their practical hardness as given by the solving time.

Introduction

Boolean Satisfiability (SAT) was the first problem to be recognized as **NP**-complete (Cook 1971), and has since been regarded as the canonical representative of this complexity class. Due to its fundamental importance, the SAT problem has been at the center of a large body of research, both theoretical and practical. More famously, SAT is directly linked to the major open question "**P** vs. **NP**", proposed as one of the Millennium Problems by The Clay Mathematics Institute (Cook 2000).

On the practical level, the challenge in SAT is to develop and implement algorithms that are able to solve increasingly harder instances. In the past decade, SAT solvers have become very powerful, and are currently able to handle practical instances with hundreds of thousands of variables and millions of clauses. SAT solving has numerous applications, including hardware and software verification, cryptanalysis, optimization etc.

The behavior of modern SAT solvers is intriguing. On the one hand, because SAT is an **NP**-complete problem, we would expect the solvers to require time exponential in the size of the input. On the other hand, it is often reported in practice that they can solve very large problems in a very short time. Partial explanations have been proposed through tractable classes (e.g., 2-SAT, Horn formulas), phase transition for random formulas, or fixed parameter tractability.

In this work we investigate the impact of the problem structure, as captured by the *treewidth* parameter, on the practical hardness of SAT instances. The SAT solvers are not designed to exploit the graphical structure of the problem, so this work provides an empirical evaluation of the correlation between structural hardness (high treewidth) and practical hardness (time of solving for a modern SAT solver).

Preliminaries

In this section we define the Satisfiability problem over propositional formulas, introduce graphical models, and describe the treewidth parameter.

Propositional Formulas and Satisfiability

Propositional formulas are defined over a set of n Boolean variables, $\mathbf{X} = \{X_1, \ldots, X_n\}$, where X_i can take values $\{0, 1\}$ (or $\{false, true\}$). The propositional connectives that we consider are $\neg, \wedge, \vee$ (negation, AND - conjunction, OR - disjunction). A literal is either a variable X_i, or its negation (complement) $\neg X_i$. Propositional formulas can be formed with literals, conjunctions, disjunctions and appropriate parenthesis. We will consider formulas in Conjunctive Normal Form (CNF). A formula is said to be CNF if it is a conjunction of clauses, with each clause being a disjunction of literals. A clause is said to be satisfied if at least one of its literals is true. A model of a formula is an assignment to all its variables, such that all the clauses are satisfied. If a formula has no model, it is said to be unsatisfiable (UNSAT).

Example 1 *The formula $\varphi = X_1 \vee (X_2 \wedge X_3) \vee (\neg X_2 \wedge X_4)$ can be arranged in CNF form as a disjunction of three clauses: $\varphi = (X_1 \vee X_2 \vee X_4) \wedge (X_1 \vee \neg X_2 \vee X_3) \wedge (X_1 \vee X_3 \vee X_4)$. Several models for this formula can be obtained just by setting $X_1 = true$. Regardless of the value of the other variables, all the clauses are satisfied in this case.*

DEFINITION **1 (SAT problem)** *Given a CNF formula φ over n Boolean variables, the Boolean Satisfiability (SAT) problem is to find a model for φ, or to report UNSAT if there is no model.*

Graphical Models

Graphical models are languages for knowledge representation that use graphs to capture the interaction between variables. In particular, separation between sets of nodes in the graph can be translated into independency between sets of variables. Examples of graphical models include constraint networks, Bayesian networks, Markov random fields, influence diagrams etc. The typical algorithms for processing graphical models are either search-based (e.g., AND/OR search (Dechter and Mateescu 2004; 2007), recursive conditioning (Darwiche 2001), value elimination (Bacchus, Dalmao, and Pitassi 2003), backtracking with tree decomposition (Terrioux and Jégou 2003) etc.), or inference-based (e.g., variable elimination, tree clustering (Bertelé and Brioschi 1972; Zhang and Poole 1994; Dechter 1999)). These algorithms are designed to exploit the problem structure, and provide a better worst case time guarantee: rather than being exponential in the number of variables, they are exponential only in the treewidth of the underlying graph.

A graphical model is defined as follows.

DEFINITION **2 (graphical model)** *A* graphical model $\mathcal{M}$ *is a 4-tuple,* $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \otimes \rangle$*, where:*

1. $\mathbf{X} = \{X_1, \ldots, X_n\}$ *is a finite set of variables;*
2. $\mathbf{D} = \{D_1, \ldots, D_n\}$ *is the set of their respective finite domains of values;*
3. $\mathbf{F} = \{f_1, \ldots, f_r\}$ *is a set of positive real-valued discrete functions (i.e., their domains can be listed), each defined over a subset of variables* $\mathbf{S}_i \subseteq \mathbf{X}$*, called its scope, and denoted by* $scope(f_i)$*;*
4. $\otimes$ *is a combination operator (e.g., product, sum, join), that can take as input two (or more) real-valued discrete functions, and produce another real-valued discrete function.*

The graphical model represents the combination of all its functions: $\otimes_{i=1}^{r} f_i$.

There are various tasks for graphical models. For example, finding an optimal solution (or the maximum probable explanation in Bayesian networks) or counting solutions (or belief updating in Bayesian networks). These tasks are in general **NP**-hard, or **#P**-hard.

It is possible to view the SAT problem as a graphical model. Given a formula φ, the set $\mathbf{X}$ is the set of variables of φ. Each variable X_i has a Boolean domain $D_i = \{0, 1\}, i \in \{1, \ldots, n\}$. The functions in $\mathbf{F}$ are the set of clauses of φ. Each clause maps an assignment of its variables to value 1 if the clause is satisfied, and maps the unique assignment that falsifies the clause to value 0. The combination operator is the logical conjunction (AND), which can also be viewed as a product. The formula φ is the combination (conjunction) of all its clauses .

DEFINITION **3 (primal graph)** *The* primal graph *of a graphical model is an undirected graph that has variables as its vertices and an edge connects any two variables that appear in the scope of the same function.*

All the advanced algorithms for solving graphical models typically exploit the structure of the primal graph of the problem. Their worst case performance is governed by a parameter of the graph called treewidth.

Treewidth

A graph $G = (V, E)$ is defined by a set of nodes V, and a set of edges (or pairs of nodes) E. The treewidth of a graph is a parameter that intuitively tells how close a graph is to a tree. For example, a tree has treewidth equal to 1, and a complete

Algorithm 1: LowerBound()

```
input  : graph G = (X, E)
output : Lowerbound lb on the treewidth of G
while G is not the empty graph do
    Find v ∈ V of minimum degree d
    lb ← max(lb, d)
    Contract v to a neighbor w (remove if v is
    isolated) by one of two methods below:
        1. Select neighbor w with minimum degree
        2. Select neighbor w that shares the smallest
        number of neighbors with v
return lb
```

graph over n variables has treewidth equal to $n - 1$. We will describe the treewidth parameter by an equivalent definition called induced width.

DEFINITION **4 (induced graph, induced width, treewidth)** *An* ordered graph *is a pair* (G, d)*, where* $G = (V, E)$ *is an undirected graph,* $V = \{X_1, \ldots, X_n\}$ *is the set of nodes, and* $d = (X_1, \ldots, X_n)$ *is an ordering of the nodes. The* width of a node *in an ordered graph is the number of neighbors that precede it in the ordering. The* width of an ordering d*, denoted* $w(d)$*, is the maximum width over all nodes. The* induced width of an ordered graph, $w^*(d)$*, is the width of the induced ordered graph obtained as follows: for each node, from last to first in* d*, its preceding neighbors are connected in a clique. The* induced width of a graph, w^**, is the minimal induced width over all orderings. The induced width is also equal to the* treewidth *of a graph.*

Lower and Upper Bounds on Treewidth

It is known that computing the treewidth of a graph is NP-complete (Arnborg, Corneil, and Proskurowski 1987). Finding the treewidth of a graph can be rephrased as finding an optimal ordering of the variables, or an optimal triangulation. Many heuristics (min-degree, min-cardinality, min-fill etc.) work at the level of variable orderings, which can then be translated to an approximation of the treewidth.

We used heuristics to provide lower and upper bounds on treewidth. We note that when the number of nodes in the graph is in the hundreds of thousands or millions, even simple heuristics such as min-fill (whose worst case complexity is $O(n^3)$) become infeasible.

Lower Bound

Several methods for providing lower bounds on treewidth are based on the degree of vertices. These ideas are described in detail in (Bodlaender 2005). The algorithm we used to provide lower bounds is based on the concepts from minor graph theory (Robertson and Seymour 1983). Algorithm 1 processes each node of the graph G once. The nodes are sorted according to their degree in G, and the ones with small degree are processed first. The basic operation is to contract the selected node v to one of its neighbors w. This means that all the neighbors of v become neighbors of

w, and then v and its incident edges are removed from the graph.

It is important to have a good heuristic for selecting the neighbor w of v. In the algorithm, we give two options, based on ideas that were developed independently in (Gogate and Dechter 2004) and (Bodlaender 2005). The first option is to pick w that has minimum degree. The intuition is that the minimum degree in the residual graph is a lower bound on the treewidth, therefore we should aim to increase this value by the contraction operation. The second option is to pick w that shares the least number of neighbors with v. The intuition in this case is to try to make the graph as dense as possible, by adding the largest number of new neighbors to w. Our experimental results show that the second method is superior, and the results we report are only based on it. This is in agreement with the reports in (Bodlaender 2005) where the two heuristics are compared.

Upper Bound

The usual heuristics for computing upper bounds on the treewidth are based on selecting vertices and inducing a residual graph. After a vertex is selected, all its neighbors are connected in a clique, and the selected vertex is eliminated. This procedure is known to generate a triangulation of the graph, and its maximal clique is an upper bound on the treewidth (it is equal to the induced width along the selected ordering of variables).

In practice, the min-fill heuristic (Kjæaerulff 1990) is reported to give better results. The idea is at each step to pick a variable that would add the least number of edges when connecting its neighbors in a clique. However, in the worst case the complexity of min-fill is $O(n^3)$, and therefore infeasible for the type of large networks that we study.

We chose a weaker heuristic, that is based on minimum degree. We first order the variables according to their degree in the graph. At each step, we pick the node with minimum degree, remove it from the graph, and reorder the remaining nodes (via a heap data structure). This produces an ordering of the variables, where the intuition is that we want the cliques that we create to be as small as possible (min-degree). However, in order to obtain an upper bound, we have to process the graph again and induce it along the ordering that we have, and the maximal clique is a true upper bound.

Experimental Results

We report experimental results on the industrial benchmarks from the SAT2009 competition[1]. We used 287 out of the 292 benchmarks. The remaining 5 were too large to be processed even by the simple heuristics that we used.

The measures we present correspond to two versions of each problem. The first one is the raw input given to the solver in the competition. The second one is the output of the preprocessor SatELite (Eén and Biere 2005). The general wisdom in the SAT competition community is that a SAT solver needs to use a preprocessor similar to SatELite in order to be competitive. Without going into details, we mention that SatELite performs a linear traversal over the variables of the formula, and eliminates by resolution the variables that would not cause a big increase in formula length. As a result, the primal graph of the formula tends to become denser, although it is based on fewer variables.

In all the figures and tables that follow, the instances are divided into three groups: UNSAT, SAT and UNKNOWN, based on how they were reported by the solver `precosat` (Biere 2009), the winner of SAT2009 competition. When we report time, it is the actual solving time (in seconds) of `precosat` in the competition.

Figure 1: Problem Size In Figure 1(a) we show a scatter plot for the number of variables before and after SatELite. As explained above, the number of variables can only decrease with preprocessing. Figure 1(b) shows a similar scatter plot for the number of clauses. SatELite may allow the user to control if, and by how much, the formula can grow in total length. In our parameter setting the number of clauses tends to decrease with preprocessing. Figures 1(a) and 1(b) clearly show the very large size of the industrial instances, often exceeding hundreds of thousands of variables and millions of clauses.

Figures 2-8: Bounds on Treewidth In the remaining of this section we present the bounds on treewidth. Figure 2(a) provides a scatter plot of the lower bound on treewidth, obtained by Algorithm 1 for the original instances, and the ones after preprocessing by SatELite. We note a slight increase in the lower bound after the execution of SatELite. This is expected because the preprocessing by resolution makes the graph denser, and therefore more likely to have a higher treewidth. More importantly, we observe the high values of the lower bounds for many of the problems. Given the observations in (Bodlaender 2005), the lower bound is likely to not be tight, and can be expected to be smaller by an order of magnitude or more on these large problems.

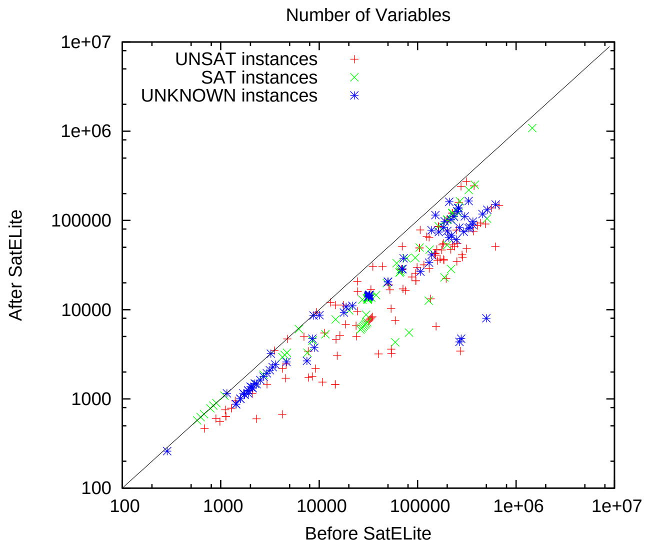
Figure 2(b) shows a scatter plot of the upper bound on treewidth, before and after SatELite. We note that the upper bounds decrease with the application of SatELite mostly for the large size problems. Given the weak heuristic that we used, the upper bound can also be expected to be loose.

The next set of Figures 3-5 show the interval bounds for each instance. Figure 3 shows the bounds for all three classes of problems (UNSAT, SAT, UNKNOWN). The horizontal axis gives the lower bound on treewidth (namely the benchmarks are ordered from left to right based on the lower bound), while the vertical axis shows both the lower bound and the upper bound as a segment. Both axes are logarithmically scaled. Figures 3(a) and 3(b) show the results before and after applying SatELite, respectively.

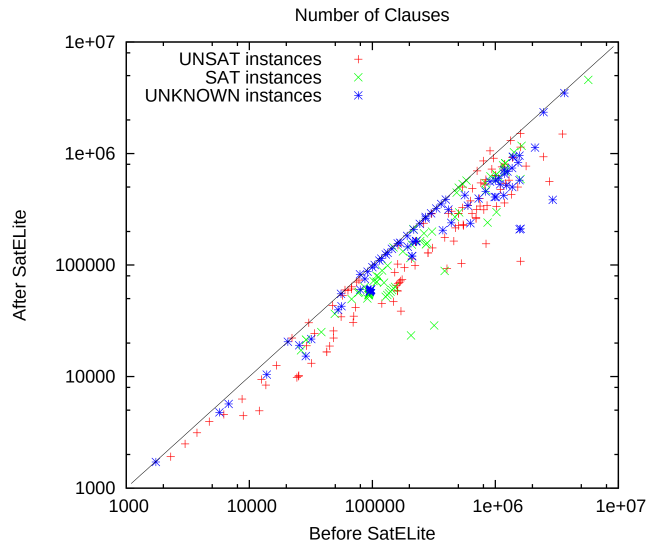
The striking observation is that there are a number of UNKNOWN problems (in blue, on the left side of the plots) for which the upper bound is smaller than the lower bound of solvable problems (either SAT or UNSAT, on the right side of the plot). This gives a strong indication that the ability of the solver to tackle a problem is not directly related to the treewidth of its graphical representation.

To provide a more detailed view on these bounds, Figures 4 and 5 show the separate plots for each of the three classes

[1] See *http://www.satcompetition.org/2009/*

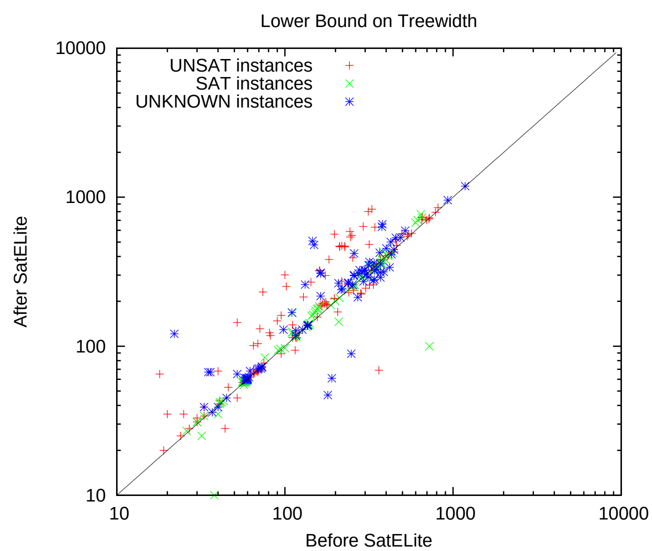


(a) Number of variables

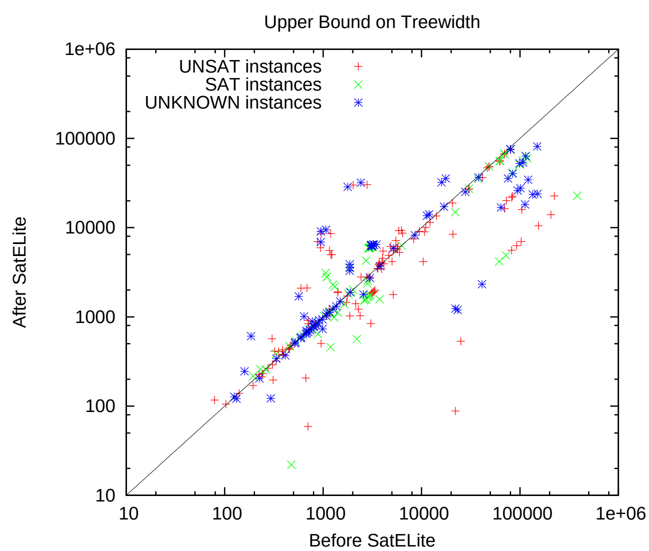


(b) Number of clauses

Figure 1: Size of problem before and after preprocessing by SatELite

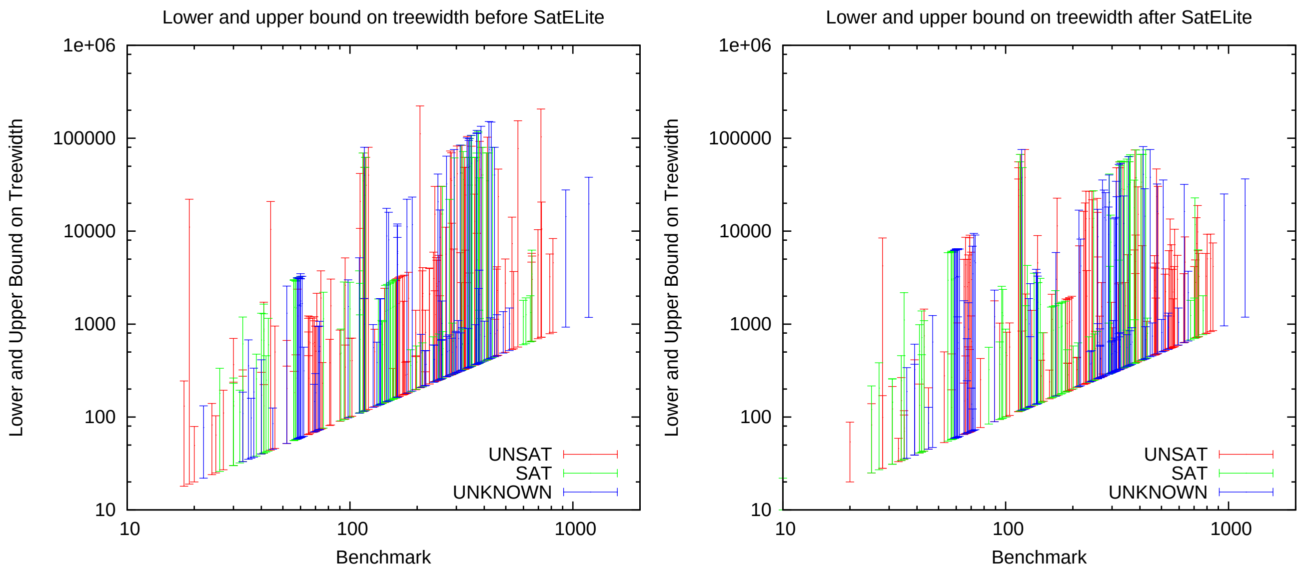


(a) Lower bound



(b) Upper bound

Figure 2: Bounds on treewidth before and after preprocessing by SatELite



(a) Lower and upper bounds before SatELite

(b) Lower and upper bounds after SatELite

Figure 3: Interval bounds on treewidth, before and after preprocessing by SatELite

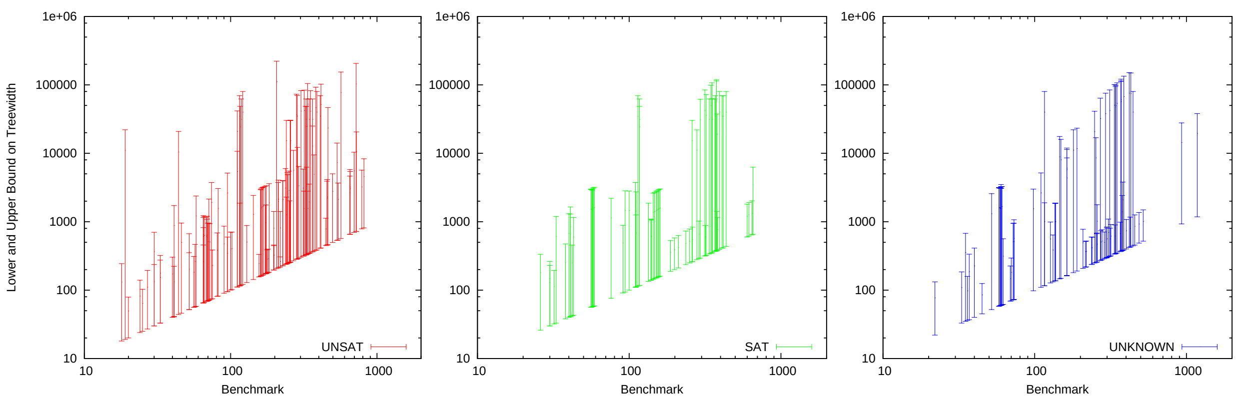


Figure 4: Bounds on treewidth before SatELite

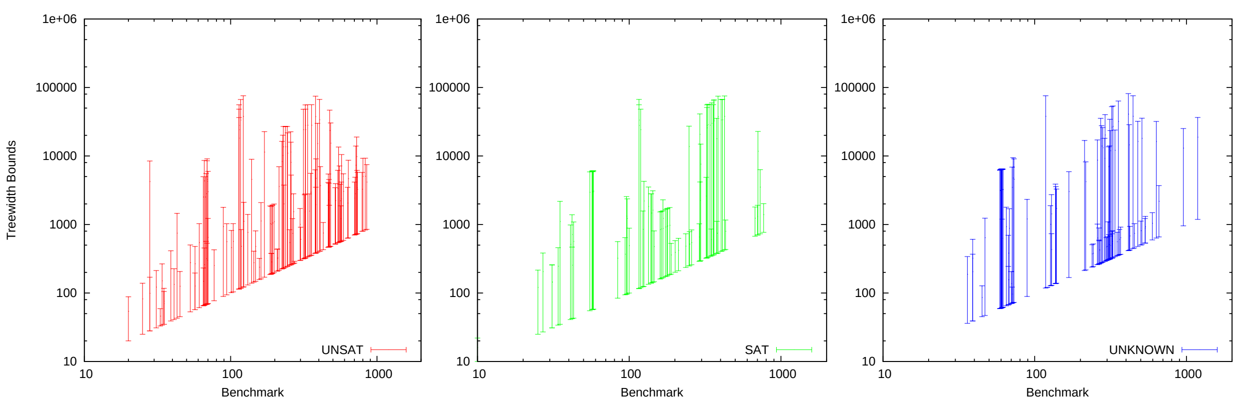


Figure 5: Bounds on treewidth after SatELite

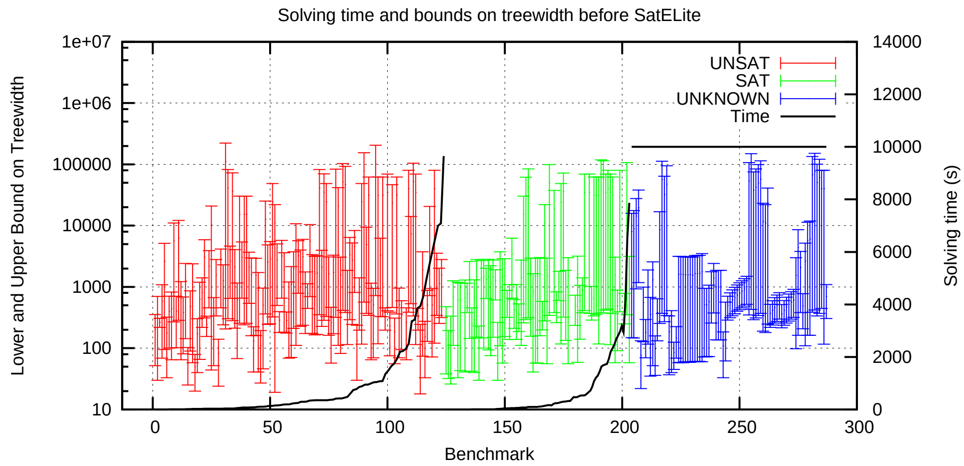


Figure 6: Solving time and bounds on treewidth, before SatELite

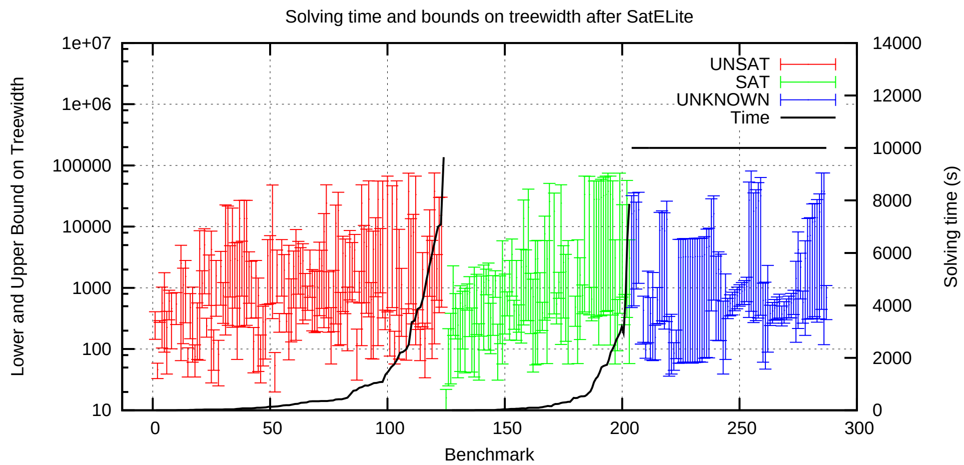


Figure 7: Solving time and bounds on treewidth, after SatELite

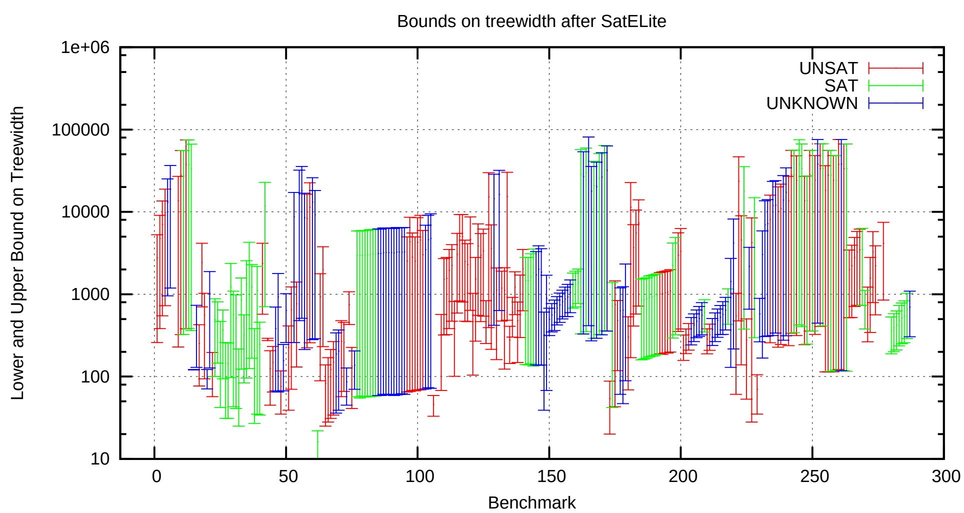


Figure 8: Benchmarks grouped by type

(UNSAT, SAT, UNKNOWN), both before and after applying SatELite. In all these plots we can also identify several classes of problems (the segments that are roughly of the same length and close to each other) that are proposed in the competition, which typically vary in size by controlling certain parameters. In the combined plots of Figure 3 we can also see the transition between UNKNOWN and SAT, or between SAT and UNSAT etc, caused by the variation in the degree of constraint of the formula.

Figures 6 and 7 show the correlation between the type of problem (UNSAT, SAT, UNKNOWN), the bounds and the time reported by `precosat` in the competition. The horizontal axis shows the three types of problems, each one ordered by the time. The time is given on the vertical axis on the right side of the figure. The vertical axis on the left side shows the bounds on treewidth.

An important observation here is that the distribution of bounds for each type of solvable problems (UNSAT and SAT) appears very uniform, perhaps with a very small increase towards the right, where they required more solving time. Again, this shows that the treewidth estimate is not indicative of the required solving time.

Finally, the last plot in Figure 8 has all the benchmarks grouped by benchmark class (the horizontal axis). Some of the classes are divided into even finer subgroups. Here we can see more clearly the transition between solvable and unsolvable, yet the difference in bounds is hardly noticeable.

For completeness, we provide in Tables 1-6 the raw numbers of our results. The results are organized by the type of problem reported by the winning solver of SAT2009, `precosat` (Biere 2009). Tables 1 and 2 show the benchmarks reported UNSAT by `precosat`. Tables 3 and 4 show the benchmarks reported SAT by `precosat`. Finally, Tables 5 and 6 show the benchmarks that could not be solved by `precosat` after 10,000 seconds.

In each table we report: `precosat` time; the number of variables (# Vars); the number of clauses (# Clauses); the lower bound (**LB**); and the upper bound (**UB**). These measures are computed on the original formula (before SatELite), and on the simplified formula (after SatELite).

Conclusion

We provided an experimental evaluation of the treewidth of the industrial instances from the SAT2009 competition. Computing the treewidth of a graph is an NP-hard problem. Indeed computing the treewidth for the SAT2009 benchmarks is infeasible in practice, and we settled on computing bounds for them. We provided a description of simple heuristics for lower and upper bounds.

Our results show that the treewidth parameter is not a good indicator for the hardness of a SAT problem. The treewidth is a good measure for dynamic programming type algorithms (e.g., ones that fully traverse a tree decomposition of the problem). However, the behavior of modern SAT solvers is far different from that. Our results show a few easy instances (namely, solvable in a short time), that exhibit considerably high lower bound on treewidth.

Our analysis may prove useful for problems that involve counting solutions, or perhaps for MAX-SAT problems.

References

Arnborg, S.; Corneil, D. G.; and Proskurowski, A. 1987. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods* 8(2):277–284.

Bacchus, F.; Dalmao, S.; and Pitassi, T. 2003. Value elimination: Bayesian inference via backtracking search. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI'03)*, 20–28.

Bertelé, U., and Brioschi, F. 1972. *Nonserial Dynamic Programming*. Academic Press.

Biere, A. 2009. P{re,i}coSATSC'09. Technical report. SAT 2009 competitive events booklet.

Bodlaender, H. L. 2005. Discovering treewidth. In *SOFSEM 2005: Theory and Practice of Computer Science*, volume 3381 of *Lecture Notes in Computer Science*. 1–16.

Cook, S. A. 1971. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, 151–158.

Cook, S. 2000. The P versus NP problem. In *Clay Mathematical Institute; The Millennium Prize Problem*.

Darwiche, A. 2001. Recursive conditioning. *Artificial Intelligence* 125(1-2):5–41.

Dechter, R., and Mateescu, R. 2004. Mixtures of deterministic-probabilistic networks and their AND/OR search space. In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence (UAI'04)*, 120–129.

Dechter, R., and Mateescu, R. 2007. AND/OR search spaces for graphical models. *Artificial Intelligence* 171(2-3):73–106.

Dechter, R. 1999. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence* 113:41–85.

Eén, N., and Biere, A. 2005. Effective preprocessing in SAT through variable and clause elimination. In *Theory and Applications of Satisfiability Testing*, volume 3569. 61–75.

Eén, N., and Sörensson, N. 2004. An extensible SAT-solver. In *Theory and Applications of Satisfiability Testing*, volume 2919. 333–336.

Gogate, V., and Dechter, R. 2004. A complete anytime algorithm for treewidth. In *UAI 2004: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, 201–208.

Kjæaerulff, U. 1990. Triangulation of graph-based algorithms giving small total state space. Technical report, University of Aalborg, Denmark.

Robertson, N., and Seymour, P. 1983. Graph minors I. excluding a forest. *J. Combin. Theory, Ser. B* 35:39–61.

Terrioux, C., and Jégou, P. 2003. Hybrid backtracking bounded by tree-decomposition of constraint networks. *Artificial Intelligence* 146:43–75.

Zhang, N. L., and Poole, D. 1994. A simple approach to Bayesian network computations. In *Proceedings of the Tenth Canadian Conference on Artificial Intelligence (UAI'94)*, 171–178.

UNSAT instances		`precosat`	Before SatELite				After SatELite			
No.	Benchmark Name	time (sec)	# Vars	# Clauses	**LB**	**UB**	# Vars	# Clauses	**LB**	**UB**
1	minor032	0.47	4,210	12,053	**52**	**353**	672	4,932	**144**	**408**
2	hoons-vbmc-lucky7	1.79	8,503	25,116	**30**	**702**	1,785	10,054	**33**	**59**
3	cmu-bmc-barrel6	1.82	2,306	8,931	**218**	**307**	598	4,452	**273**	**291**
4	manol-pipe-g6bi	1.89	23,891	69,895	**69**	**1,086**	5,021	30,529	**104**	**1,029**
5	een-tip-uns-nusmv-t5.B	2.49	59,265	170,983	**95**	**5,150**	7,565	38,589	**89**	**1,776**
6	countbitssrl016	2.79	4,567	13,652	**33**	**321**	1,706	8,393	**39**	**413**
7	minor064	3.72	14,578	42,581	**100**	**705**	1,458	16,652	**301**	**914**
8	manol-pipe-c9	3.99	15,214	45,082	**65**	**822**	3,049	18,773	**101**	**818**
9	hsat_vc11803	4.00	276,427	798,348	**269**	**11,035**	241,180	858,792		**10,097**
10	maxand064	4.22	14,578	42,581	**102**	**706**	1,452	16,631	**252**	**837**
11	hsat_vc11813	5.61	312,628	905,106	**286**	**12,184**	273,230	1,057,038		**11,489**
12	gus-md5-04	9.44	68,679	223,994	**65**	**1,225**	28,084	161,315	**65**	**4,977**
13	minandmaxor032	9.90	16,239	48,332	**81**	**686**	5,155	25,655	**123**	**2,105**
14	manol-pipe-g7nidw	10.55	75,496	222,379	**143**	**2,424**	16,348	99,001	**269**	**2,802**
15	smulo016	12.38	2,945	8,738	**25**	**103**	1,459	6,286	**35**	**105**
16	minxorminand032	14.26	10,794	31,997	**90**	**862**	1,541	13,181	**148**	**806**
17	schup-l2s-abp4-1-k31	18.20	14,809	48,483	**58**	**2,378**	4,637	22,181	**61**	**1,028**
18	countbitsrotate016	24.06	2,087	6,212	**20**	**79**	1,144	4,583	**35**	**117**
19	icbrt1_32	24.30	11,309	33,833	**40**	**303**	5,481	24,428	**68**	**568**
20	gus-md5-05	24.84	68,827	224,473	**66**	**1,196**	28,164	161,687	**66**	**8,598**
21	minand128	26.04	53,999	159,692	**218**	**1,411**	3,606	59,022	**472**	**1,909**
22	manol-pipe-c10nidw_s	27.22	404,382	1,204,273	**313**	**5,876**	87,627	540,264	**802**	**9,280**
23	rbcl_xits_06_UNSAT	28.96	980	47,620	**156**	**333**	555	43,330	**157**	**320**
24	uts-l05-ipc5-h26-unsat	30.19	87,475	414,657	**455**	**3,914**	23,197	289,782	**518**	**3,442**
25	simon-s03-fifo8-300	30.26	194,762	530,713	**44**	**20,865**	22,224	103,316	**28**	**8,450**
26	cmu-bmc-longmult15	30.71	7,807	24,351	**52**	**668**	1,731	9,791	**45**	**206**
27	velev-live-uns-2.0-ebuf	32.53	14,628	161,477	**336**	**2,793**	11,264	150,831	**344**	**2,797**
28	eq.atree.braun.8.unsat	33.45	684	2,300	**24**	**140**	465	1,911	**25**	**139**
29	uts-l05-ipc5-h27-unknown	33.69	95,421	439,508	**456**	**4,139**	25,580	303,995	**521**	**3,941**
30	velev-engi-uns-1.0-4nd	34.18	7,000	67,586	**240**	**2,287**	4,968	60,893	**264**	**1,218**
31	q_query_2_L324_coli	36.80	617,854	2,745,225	**206**	**222,920**	50,848	561,758	**170**	**22,668**
32	total-5-13-u	37.96	178,708	806,681	**302**	**82,588**	53,288	312,648	**245**	**21,754**
33	uts-l06-ipc5-h28-unknown	39.92	115,898	649,270	**652**	**4,646**	31,748	485,694	**709**	**4,902**
34	total-5-11-u	40.99	156,980	696,581	**282**	**72,908**	47,375	266,053	**227**	**20,120**
35	manol-pipe-g10id	42.09	159,638	473,596	**212**	**4,059**	35,337	215,434	**465**	**5,464**
36	gus-md5-06	55.19	68,953	224,868	**68**	**1,161**	28,209	161,935	**68**	**5,551**
37	UCG-10-5p0	58.10	104,481	500,875	**256**	**30,300**	49,268	289,617	**237**	**26,981**
38	uts-l06-ipc5-h31-unknown	71.36	151,768	761,906	**656**	**5,410**	40,760	545,239	**735**	**5,782**
39	ACG-10-5p0	72.39	123,925	539,763	**240**	**30,299**	65,873	326,952	**229**	**26,994**
40	UR-10-5p0	76.05	104,479	500,869	**256**	**30,304**	49,247	289,562	**244**	**26,974**
41	velev-pipe-o-uns-1.1-6	78.03	17,710	304,026	**542**	**3,678**	11,253	288,247	**561**	**3,876**
42	manol-pipe-c6bidw_i	79.44	96,089	283,993	**160**	**2,979**	21,086	128,747	**320**	**2,712**
43	goldb-heqc-term1mul	81.12	3,504	22,229	**41**	**223**	3,483	22,155	**41**	**226**
44	manol-pipe-c6nidw_i	96.64	95,863	283,363	**161**	**2,931**	21,067	128,369	**323**	**2,796**
45	post-c32s-gcdm16-23	99.32	135,543	404,326	**41**	**1,723**	13,225	92,866	**43**	**1,452**
46	eq.atree.braun.9.unsat	102.92	892	3,006	**27**	**194**	602	2,492	**28**	**170**
47	simon-s02b-dp11u10	107.20	9,197	25,271	**46**	**954**	2,185	10,218	**53**	**502**
48	post-cbmc-aes-ele-noholes	109.57	270,963	1,601,376	**362**	**25,087**	3,438	108,292	**69**	**533**
49	manol-pipe-g10nid	127.46	217,725	646,228	**245**	**4,869**	47,105	289,120	**540**	**6,175**
50	manol-pipe-g10bidw	134.95	237,485	705,220	**250**	**5,485**	51,262	315,228	**554**	**7,144**
51	UCG-15-5p0	137.42	162,696	821,000	**325**	**48,623**	85,360	531,912	**314**	**48,114**
52	post-c32s-col400-16	147.31	286,329	839,692	**19**	**22,089**	38,751	155,211	**20**	**88**
53	blocks-4-ipc5-h22-unknown	153.72	148,153	937,242	**497**	**5,022**	42,404	725,045	**570**	**4,162**
54	rpoc_xits_07_UNSAT	159.28	1,128	63,345	**180**	**397**	636	59,045	**189**	**380**
55	uts-l06-ipc5-h33-unknown	179.89	175,670	837,466	**653**	**5,770**	46,700	584,998	**730**	**6,170**
56	manol-pipe-c7bidw_i	188.79	131,004	387,841	**183**	**3,602**	28,877	176,334	**382**	**3,502**
57	rbcl_xits_07_UNSAT	194.69	1,128	57,446	**177**	**389**	636	53,156	**190**	**378**
58	gus-md5-07	196.25	69,097	225,325	**69**	**1,200**	28,269	162,247	**67**	**4,974**
59	countbitswegner064	218.53	24,448	73,151	**71**	**2,145**	9,592	41,755	**131**	**1,403**
60	manol-pipe-c7nidw	226.13	169,072	501,421	**211**	**3,744**	36,879	225,452	**469**	**4,004**

Table 1: UNSAT 1: benchmarks from SAT2009 competition reported UNSAT by `precosat`

UNSAT instances		`precosat`	Before SatELite				After SatELite			
No.	Benchmark Name	time (sec)	# Vars	# Clauses	**LB**	**UB**	# Vars	# Clauses	**LB**	**UB**
61	schup-l2s-guid-1-k56	248.19	98,746	307,346	**111**	**10,699**	29,794	142,406	**139**	**8,964**
62	velev-pipe-o-uns-1.0-7	253.57	24,415	711,050	**787**	**5,670**	20,704	698,926	**791**	**5,763**
63	manol-pipe-f9b	287.27	183,371	547,078	**227**	**4,010**	36,594	230,811	**470**	**4,537**
64	9dlx_vliw_at_b_iq1	295.47	24,604	261,473	**237**	**5,960**	15,991	237,180	**259**	**5,274**
65	minxorminand064	299.51	40,042	119,357	**174**	**1,758**	3,190	45,043	**298**	**1,713**
66	goldb-heqc-i10mul	311.84	12,998	77,941	**447**	**1,125**	12,031	73,263	**429**	**1,071**
67	AProVE07-09	342.25	33,582	160,817	**697**	**10,390**	16,861	102,070	**702**	**4,159**
68	q_query_3_l48_lambda	344.61	34,656	174,528	**175**	**3,365**	8,325	74,357	**198**	**1,994**
69	q_query_3_l45_lambda	348.90	32,313	161,529	**165**	**3,158**	7,662	67,933	**186**	**1,864**
70	q_query_3_l46_lambda	349.99	33,090	165,828	**167**	**3,227**	7,884	70,063	**193**	**1,899**
71	q_query_3_L80_coli.sat	350.45	250,567	1,183,233	**351**	**82,224**	34,654	360,690	**352**	**5,581**
72	manol-pipe-f9n	351.87	185,149	552,412	**227**	**3,960**	35,934	225,419	**464**	**4,113**
73	dated-5-15-u	353.60	151,952	697,321	**285**	**69,828**	42,845	260,433	**224**	**16,375**
74	UR-15-5p0	354.27	162,682	820,958	**330**	**48,616**	85,314	531,785	**325**	**48,273**
75	q_query_3_l44_lambda	359.89	31,540	157,264	**162**	**3,089**	7,445	65,849	**189**	**1,832**
76	goldb-heqc-alu4mul	372.42	4,736	30,465	**57**	**467**	4,705	30,376	**57**	**477**
77	q_query_3_l47_lambda	377.99	33,871	170,161	**172**	**3,296**	8,102	72,196	**193**	**1,929**
78	dated-5-17-u	410.35	183,464	855,717	**318**	**83,908**	55,572	342,787	**257**	**22,537**
79	UTI-10-10p0	421.86	130,859	627,223	**111**	**41,687**	64,622	376,304	**114**	**36,407**
80	post-c32s-ss-8	425.05	53,752	148,393	**82**	**3,057**	10,244	46,934	**118**	**842**
81	q_query_3_L100_coli.sat	428.93	315,617	1,522,583	**414**	**102,864**	48,305	500,563	**410**	**6,981**
82	q_query_3_L90_coli.sat	457.27	282,842	1,350,458	**382**	**92,544**	41,200	428,294	**378**	**6,281**
83	AProVE07-16	480.61	52,382	182,485	**115**	**1,868**	16,713	94,756	**94**	**1,027**
84	manol-pipe-f10ni	580.02	368,888	1,100,500	**292**	**6,426**	75,579	476,294	**637**	**8,675**
85	manol-pipe-c10nid_i	701.40	252,516	750,877	**244**	**5,213**	55,124	337,811	**589**	**5,486**
86	velev-pipe-uns-1.0-8	763.44	35,065	1,332,773	**814**	**8,327**	30,321	1,308,900	**851**	**7,470**
87	eq.atree.braun.10.unsat	780.22	1,111	3,756	**30**	**237**	754	3,137	**31**	**213**
88	minxor128	856.85	54,258	160,469	**197**	**1,409**	3,238	58,444	**565**	**1,869**
89	UTI-15-5p0	873.82	162,926	821,689	**116**	**48,617**	85,562	532,282	**114**	**48,238**
90	q_query_3_L150_coli.sat	939.09	486,992	2,456,708	**567**	**154,464**	91,182	935,986	**573**	**10,488**
91	9dlx_vliw_at_b_iq2	944.84	44,095	542,253	**371**	**9,483**	30,589	500,510	**383**	**9,083**
92	UCG-15-10p0	948.38	199,304	1,005,834	**345**	**62,371**	101,275	635,856	**336**	**55,986**
93	countbitssrl032	992.24	18,607	55,724	**65**	**1,153**	6,860	34,360	**70**	**1,226**
94	gus-md5-09	1036.33	69,487	226,581	**70**	**952**	28,473	163,208	**69**	**8,494**
95	q_query_3_L200_coli.sat	1045.30	670,867	3,513,333	**719**	**206,064**	146,532	1,494,031	**717**	**13,988**
96	UTI-15-10p0	1050.29	199,586	1,006,679	**118**	**62,370**	101,546	636,691	**114**	**55,802**
97	manol-pipe-c10nidw	1075.81	433,601	1,291,714	**329**	**6,250**	93,650	576,880	**831**	**9,293**
98	ACG-15-10p0	1091.56	262,253	1,131,732	**327**	**62,374**	158,063	758,663	**324**	**55,632**
99	rpoc_xits_08_UNSAT	1327.90	1,278	74,789	**197**	**456**	785	70,468	**208**	**434**
100	UR-20-5p0	1466.86	224,569	1,190,619	**409**	**69,412**	120,292	802,152	**405**	**67,018**
101	AProVE07-08	1527.95	4,614	16,637	**75**	**386**	2,470	12,585	**77**	**427**
102	schup-l2s-bc56s-1-k391	1706.06	561,371	1,778,987	**462**	**46,602**	137,809	771,029	**475**	**46,750**
103	AProVE07-27	1755.41	7,729	29,194	**56**	**310**	3,423	18,902	**57**	**196**
104	UR-15-10p0	1904.29	199,290	1,005,792	**363**	**62,379**	101,249	635,768	**357**	**55,950**
105	gus-md5-10	2178.61	69,503	226,618	**70**	**953**	28,461	163,195	**69**	**9,066**
106	maxxororand032	2179.05	23,696	70,703	**95**	**595**	6,601	34,783	**161**	**2,091**
107	velev-vliw-uns-4.0-9C1	2267.49	96,177	1,814,189						
108	goldb-heqc-dalumul	2304.26	9,426	59,991	**65**	**457**	9,359	59,764	**66**	**454**
109	ACG-20-10p0	2458.18	374,228	1,603,007	**388**	**79,977**	243,722	1,143,032	**380**	**74,829**
110	9dlx_vliw_at_b_iq3	3234.68	69,789	968,295	**533**	**14,152**	51,142	907,131	**547**	**13,537**
111	total-10-13-u	3392.82	226,208	1,021,181	**336**	**104,588**	53,326	336,047	**258**	**15,918**
112	UTI-20-5p0	3515.55	225,296	1,192,799	**115**	**69,413**	121,016	804,013	**117**	**66,859**
113	rbcl_xits_08_UNSAT	3907.09	1,278	68,055	**197**	**447**	785	63,744	**209**	**440**
114	countbitsarray02_32	3929.95	4,252	12,563	**18**	**244**	2,180	9,405	**65**	**232**
115	emptyroom-4-h21-unsat	4189.95	71,112	208,728	**74**	**3,745**	17,074	113,428	**231**	**3,768**
116	eq.atree.braun.11.unsat	4653.24	1,400	4,732	**33**	**275**	950	3,941	**34**	**266**
117	maxxor064	5164.10	51,064	152,039	**129**	**879**	18,948	85,956	**214**	**6,960**
118	9dlx_vliw_at_b_iq4	5562.54	106,013	1,598,301	**723**	**20,578**	78,290	1,507,407	**726**	**18,899**
119	gus-md5-11	5949.97	69,561	226,787	**72**	**939**	28,474	163,303	**70**	**5,959**
120	UTI-20-10p0	6393.16	259,616	1,374,599	**121**	**79,976**	137,616	917,583	**122**	**75,598**
121	minxorminand128	6733.77	153,834	459,965	**343**	**3,550**	6,499	164,062	**629**	**3,490**
122	maxor128	7133.63	200,308	598,619	**253**	**2,020**	76,300	341,819	**392**	**29,982**
123	minandmaxor128	7259.69	249,327	746,444	**317**	**2,798**	77,849	393,555	**482**	**30,323**
124	velev-vliw-uns-4.0-9-i1	9645.60	154,309	3,230,738						

Table 2: UNSAT 2: benchmarks from SAT2009 competition reported UNSAT by `precosat`

SAT instances		`precosat`	Before SatELite				After SatELite			
No.	Benchmark Name	time (sec)	# Vars	# Clauses	**LB**	**UB**	# Vars	# Clauses	**LB**	**UB**
125	een-tip-sat-texas-tp-5e	0.30	17,985	52,128	**38**	**475**	40	153	**10**	**22**
126	AProVE09-13	0.48	7,606	26,317	**32**	**194**	3,288	17,210	**25**	**216**
127	AProVE09-22	0.63	11,557	38,505	**26**	**334**	5,295	25,070	**27**	**383**
128	AProVE09-21	1.39	29,964	91,044	**111**	**1,258**	8,769	50,118	**167**	**2,291**
129	AProVE09-03	1.54	59,231	206,299	**209**		4,310	23,357	**146**	**807**
130	AProVE09-25	1.86	37,821	124,485	**33**	**1,191**	14,575	69,512	**34**	**458**
131	AProVE09-11	1.88	20,192	78,082	**43**	**1,152**	9,817	57,138	**43**	**1,088**
132	AProVE09-01	1.98	81,870	318,353	**723**		5,538	28,696	**100**	**884**
133	AProVE09-12	2.61	27,495	102,011	**41**	**1,293**	12,954	71,907	**41**	**984**
134	rpoc_xits_17_SAT	2.85	2,718	215,000	**382**	**1,149**	1,863	210,423	**432**	**1,162**
135	q_query_3_l37_lambda	3.01	26,241	128,361	**144**	**2,606**	5,982	51,930	**160**	**1,525**
136	AProVE09-05	4.12	14,685	49,510	**42**	**456**	7,776	36,486	**42**	**468**
137	AProVE09-24	4.70	61,164	209,228	**40**	**1,309**	33,169	152,619	**35**	**2,177**
138	q_query_3_l40_lambda	5.20	28,488	140,544	**152**	**2,813**	6,597	57,746	**173**	**1,663**
139	AProVE09-07	5.80	8,567	28,936	**30**	**230**	4,518	21,426	**31**	**259**
140	q_query_3_l38_lambda	6.76	26,986	132,388	**147**	**2,675**	6,185	53,844	**163**	**1,557**
141	AProVE09-10	7.14	67,186	245,335	**100**	**2,814**	28,201	164,224	**97**	**2,365**
142	ndhf_xits_22_SAT	7.96	4,692	582,514	**644**	**2,021**	3,288	572,603	**769**	**2,018**
143	AProVE09-19	9.61	30,537	112,780	**94**	**2,833**	13,476	78,322	**96**	**2,554**
144	AProVE09-17	9.82	33,894	108,759	**76**	**2,203**	13,647	70,365	**84**	**562**
145	q_query_3_l39_lambda	10.15	27,735	136,449	**151**	**2,744**	6,393	55,806	**167**	**1,584**
146	ndhf_xits_21_SAT	10.23	4,466	542,457	**615**	**1,911**	3,107	532,588	**702**	**1,908**
147	AProVE09-08	23.10	8,564	28,927	**30**	**263**	4,519	21,434	**31**	**256**
148	AProVE09-15	24.85	94,663	305,105	**110**	**3,746**	38,229	199,185	**123**	**1,576**
149	gss-13-s100	32.86	30,867	92,735	**57**	**2,883**	12,993	53,742	**57**	**5,877**
150	q_query_3_l41_lambda	40.04	29,245	144,673	**155**	**2,882**	6,805	59,734	**177**	**1,686**
151	rbcl_xits_14_SAT	42.24	2,220	148,488	**318**	**882**	1,488	143,993	**348**	**859**
152	mizh-md5-47-3	48.47	65,604	273,522	**139**	**1,092**	26,133	153,778	**140**	**2,802**
153	uts-l06-ipc5-h35-unknown	50.71	199,571	912,286	**654**	**6,265**	52,969	625,293	**730**	**6,307**
154	vmpc_26	51.61	676	86,424	**211**	**628**	675	63,480	**211**	**628**
155	mizh-md5-47-4	60.50	65,604	273,506	**139**	**1,092**	26,107	153,650	**139**	**2,802**
156	vmpc_24	61.35	576	67,872	**187**	**530**	575	49,478	**189**	**530**
157	AProVE09-20	71.04	33,054	108,377	**111**	**2,740**	13,909	70,732	**125**	**4,276**
158	UR-10-5p1	73.50	104,856	508,020	**260**	**30,305**	49,633	296,723	**248**	**27,263**
159	q_query_3_L60_coli.sat	73.64	187,517	863,483	**294**	**61,584**	23,017	240,376	**293**	**4,181**
160	partial-5-13-s	95.66	184,675	842,981	**315**	**84,348**	83,779	465,251	**292**	**41,166**
161	q_query_3_l42_lambda	96.57	30,006	148,836	**157**	**2,951**	7,017	61,749	**179**	**1,727**
162	post-c32s-gcdm16-22	97.02	129,652	386,749	**41**	**1,644**	12,623	88,631	**42**	**1,387**
163	gss-14-s100	99.61	31,229	93,855	**57**	**2,951**	12,954	53,595	**55**	**5,865**
164	gss-16-s100	100.00	31,248	93,904	**57**	**2,952**	13,350	55,319	**57**	**5,953**

Table 3: SAT 1: benchmarks from SAT2009 competition reported SAT by `precosat`

SAT instances		`precosat`	Before SatELite				After SatELite			
No.	Benchmark Name	time (sec)	# Vars	# Clauses	**LB**	**UB**	# Vars	# Clauses	**LB**	**UB**
165	q_query_3_l43_lambda	108.38	30,771	153,033	**160**	**3,020**	7,230	63,791	**185**	**1,768**
166	vmpc_25	120.55	625	76,775	**199**	**581**	624	56,191	**201**	**581**
167	simon-s03-w08-15	140.28	132,555	469,519	**279**	**22,025**	47,093	269,328	**296**	**14,913**
168	gss-15-s100	160.24	31,238	93,878	**56**	**2,952**	13,135	54,384	**58**	**5,911**
169	partial-5-15-s	163.29	218,494	1,013,700	**345**	**99,308**	103,146	583,284	**325**	**51,055**
170	velev-pipe-sat-1.0-b10	163.31	118,040	8,804,672						
171	schup-l2s-motst-2-k315	230.69	507,145	1,601,920	**380**	**37,572**	104,775	590,065	**377**	**35,400**
172	vmpc_30	238.75	900	133,080	**261**	**832**	899	98,929	**258**	**832**
173	ndhf_xits_20_SAT	250.19	4,242	503,783	**599**	**1,803**	2,938	493,954	**675**	**1,801**
174	UTI-15-5p1	281.42	163,388	832,285	**117**	**48,624**	86,036	542,912	**119**	**48,304**
175	q_query_3_L70_coli.sat	299.18	218,792	1,020,908	**322**	**71,904**	28,599	298,042	**322**	**4,888**
176	mizh-sha0-35-3	302.60	48,689	204,067	**134**	**1,866**	19,864	115,548	**134**	**3,529**
177	gss-17-s100	324.50	31,318	94,116	**56**	**3,000**	13,388	55,473	**57**	**5,945**
178	velev-pipe-sat-1.0-b7	332.36	118,040	8,804,672						
179	9vliw_m_9stages_iq3_C1_bug8	452.36	521,179	13,378,617						
180	vmpc_28	468.90	784	108,080	**236**	**731**	784	80,113	**236**	**731**
181	gss-21-s100	477.03	31,613	95,104	**59**	**3,188**	13,864	57,372	**58**	**6,118**
182	mizh-md5-48-5	520.15	66,892	279,256	**140**	**1,059**	26,711	157,466	**143**	**3,104**
183	vange-col-abb313GPIA-9-c	536.01	6,228	484,871	**379**	**1,411**	6,052	446,229	**379**	**1,115**
184	UCG-20-5p1	537.51	224,986	1,204,430	**410**	**69,409**	120,702	815,915	**404**	**66,772**
185	9vliw_m_9stages_iq3_C1_bug5	600.20	520,770	13,380,350						
186	ACG-20-5p1	679.55	331,196	1,416,850	**370**	**69,412**	219,455	1,025,433	**362**	**66,531**
187	vmpc_33	804.74	1,089	177,375	**288**	**1,024**	1,087	132,531	**290**	**1,024**
188	ACG-15-10p1	1070.38	268,258	1,155,731	**337**	**62,373**	164,044	782,950	**325**	**55,640**
189	UCG-15-10p1	1231.23	200,003	1,019,221	**353**	**62,373**	101,967	649,241	**341**	**56,053**
190	UTI-15-10p1	1345.76	200,152	1,019,667	**117**	**62,369**	102,119	649,399	**116**	**55,817**
191	partial-10-15-s	1608.48	261,020	1,211,106	**376**	**118,668**	120,580	686,014	**350**	**59,625**
192	partial-5-17-s	1684.88	252,328	1,189,896	**376**	**114,268**	126,748	723,645	**361**	**64,081**
193	UR-20-5p1	1719.65	224,962	1,204,358	**417**	**69,413**	120,690	815,881	**412**	**67,011**
194	UCG-20-10p1	1803.15	259,258	1,388,006	**436**	**79,977**	137,278	931,375	**425**	**75,133**
195	UTI-20-5p1	2175.73	225,926	1,207,249	**114**	**69,412**	121,319	818,082	**116**	**66,899**
196	UR-15-10p1	2345.26	199,996	1,019,200	**367**	**62,378**	101,975	649,225	**359**	**55,975**
197	gss-19-s100	2565.53	31,435	94,548	**58**	**3,073**	13,701	56,714	**57**	**6,098**
198	AProVE09-06	2780.75	77,262	263,137	**91**	**885**	37,724	192,750	**94**	**644**
199	ACG-20-10p1	2944.73	381,708	1,632,906	**394**	**79,974**	251,132	1,172,854	**385**	**74,836**
200	clauses-8	3167.09	1,461,772	5,687,554		**382,386**	1,085,745	4,572,347	**706**	**22,815**
201	vmpc_29	3202.60	841	120,147	**250**	**784**	841	89,294	**250**	**784**
202	partial-10-13-s	5981.52	234,673	1,071,339	**351**	**107,228**	110,779	609,126	**330**	**56,973**
203	gss-20-s100	7869.33	31,503	94,748	**58**	**3,147**	13,733	56,849	**58**	**6,091**

Table 4: SAT 2: benchmarks from SAT2009 competition reported SAT by `precosat`

UNKOWN for `precosat`		`precosat`	Before SatELite				After SatELite			
No.	Benchmark Name	time (sec)	# Vars	# Clauses	**LB**	**UB**	# Vars	# Clauses	**LB**	**UB**
204	cube-11-h13-unsat	? (MO)	455,627	1,367,522	**149**	**15,955**	118,075	741,668	**478**	**32,142**
205	cube-11-h14-sat	? (MO)	509,512	1,529,045	**146**	**17,620**	131,809	828,250	**509**	**35,631**
206	9dlx_vliw_at_b_iq5	> 10,000	151,669	2,465,731	**929**	**27,823**	114,805	2,347,281	**955**	**25,166**
207	9dlx_vliw_at_b_iq6	> 10,000	209,724	3,634,677	**1179**	**38,080**	161,659	3,486,112	**1187**	**36,522**
208	aloul-chnl11-13	> 10,000	286	1,742	**22**	**132**	260	1,716	**121**	**121**
209	AProVE07-01	> 10,000	7,502	28,770	**127**	**990**	2,667	15,251	**129**	**732**
210	AProVE07-25	> 10,000	8,920	31,884	**70**	**294**	3,746	21,718	**71**	**122**
211	AProVE07-26	> 10,000	21,734	79,766	**116**	**1,891**	11,056	60,210	**127**	**1,881**
212	countbitsarray08_32	> 10,000	17,818	52,685	**35**	**676**	9,278	39,696	**67**	**697**
213	countbitsarray32_32	> 10,000	72,202	213,533	**52**	**2,568**	37,710	161,284	**65**	**1,787**
214	countbitsrotate032	> 10,000	8,527	25,484	**36**	**159**	4,734	19,090	**67**	**245**
215	countbitsrotate128	> 10,000	138,559	415,292	**132**	**639**	77,742	314,951	**259**	**1,013**
216	countbitssrl128	> 10,000	301,759	904,892	**253**	**16,897**	110,923	560,498	**259**	**17,177**
217	dated-10-19-u	> 10,000	246,502	1,169,243	**370**	**112,508**	60,817	420,689	**289**	**18,169**
218	dated-5-13-u	> 10,000	138,808	626,501	**271**	**64,108**	41,076	236,980	**213**	**16,828**
219	dated-5-19-u	> 10,000	208,002	986,543	**341**	**94,908**	64,079	408,121	**281**	**26,039**
220	eq.atree.braun.12.unsat	> 10,000	1,694	5,726	**37**	**335**	1,155	4,782	**36**	**340**
221	eq.atree.braun.13.unsat	> 10,000	2,010	6,802	**40**	**412**	1,372	5,687	**39**	**369**
222	goldb-heqc-frg1mul	> 10,000	3,230	20,575	**45**	**125**	3,215	20,534	**45**	**127**
223	goldb-heqc-x1mul	> 10,000	8,760	55,571	**69**	**225**	8,603	55,158	**70**	**205**
224	gss-22-s100	> 10,000	31,616	95,110	**58**	**3,146**	13,884	57,461	**59**	**6,196**
225	gss-23-s100	> 10,000	31,711	95,400	**59**	**3,148**	13,973	57,794	**59**	**6,192**
226	gss-24-s100	> 10,000	31,821	95,735	**58**	**3,132**	14,187	58,766	**61**	**6,366**
227	gss-25-s100	> 10,000	31,931	96,111	**60**	**3,085**	14,297	59,235	**60**	**6,371**
228	gss-26-s100	> 10,000	31,942	96,138	**59**	**3,084**	14,301	59,230	**61**	**6,323**
229	gss-27-s100	> 10,000	31,951	96,161	**59**	**3,105**	14,301	59,238	**60**	**6,391**
230	gss-28-s100	> 10,000	32,151	96,786	**60**	**3,062**	14,375	59,521	**59**	**6,271**
231	gss-31-s100	> 10,000	32,310	97,279	**60**	**3,248**	14,550	60,185	**60**	**6,458**
232	gss-32-s100	> 10,000	32,312	97,285	**60**	**3,248**	14,549	60,218	**61**	**6,440**
233	gss-33-s100	> 10,000	32,402	97,556	**61**	**3,264**	14,599	60,392	**62**	**6,431**
234	gss-34-s100	> 10,000	32,465	97,748	**60**	**3,489**	14,655	60,615	**61**	**6,467**
235	gus-md5-14	> 10,000	69,641	227,010	**72**	**948**	28,481	163,405	**71**	**6,890**
236	gus-md5-15	> 10,000	69,695	227,171	**73**	**949**	28,490	163,502	**73**	**9,082**
237	gus-md5-16	> 10,000	70,139	228,634	**73**	**1,070**	28,581	164,193	**72**	**9,471**
238	maxxor128	> 10,000	200,440	599,015	**258**	**1,775**	76,171	341,157	**420**	**28,605**
239	maxxororand128	> 10,000	365,168	1,093,967	**377**	**2,419**	97,137	529,534	**633**	**31,908**
240	mizh-sha0-36-2	> 10,000	50,073	210,239	**137**	**1,867**	20,535	120,186	**139**	**3,290**
241	mizh-sha0-36-3	> 10,000	50,073	210,235	**136**	**1,867**	20,544	120,212	**137**	**3,885**
242	mizh-sha0-36-4	> 10,000	50,073	210,235	**137**	**1,867**	20,555	120,279	**138**	**3,566**
243	mulhs016	> 10,000	4,656	13,871	**33**	**185**	2,599	10,416	**39**	**607**

Table 5: UNKNOWN 1: benchmarks from SAT2009 competition reported UNKNOWN by `precosat`

UNKOWN for `precosat`		`precosat`	Before SatELite				After SatELite			
No.	Benchmark Name	time (sec)	# Vars	# Clauses	**LB**	**UB**	# Vars	# Clauses	**LB**	**UB**
244	mulhs032	> 10,000	18,792	56,183	**62**	**565**	10,761	42,789	**68**	**1,696**
245	ndhf_xits_09_UNSAT	> 10,000	1,910	167,931	**287**	**707**	1,151	158,551	**314**	**677**
246	ndhf_xits_10_UNSAT	> 10,000	2,112	191,788	**314**	**803**	1,317	182,360	**349**	**770**
247	ndhf_xits_11_UNSAT	> 10,000	2,316	216,962	**339**	**897**	1,463	207,500	**362**	**849**
248	ndhf_xits_12_UNSAT	> 10,000	2,522	243,459	**366**	**985**	1,625	233,949	**424**	**938**
249	ndhf_xits_13_UNSAT	> 10,000	2,730	271,285	**401**	**1,075**	1,774	261,748	**454**	**1,030**
250	ndhf_xits_14_UNSAT	> 10,000	2,940	300,446	**426**	**1,167**	1,937	290,863	**501**	**1,122**
251	ndhf_xits_15_UNKNOWN	> 10,000	3,152	330,948	**456**	**1,261**	2,093	321,327	**534**	**1,223**
252	ndhf_xits_16_UNKNOWN	> 10,000	3,366	362,797	**487**	**1,363**	2,270	353,123	**536**	**1,315**
253	ndhf_xits_17_UNKNOWN	> 10,000	3,582	395,999	**519**	**1,491**	2,427	386,290	**596**	**1,490**
254	partial-10-13-u	> 10,000	234,064	1,059,777	**349**	**107,228**	110,173	597,630	**328**	**53,741**
255	partial-10-19-u	> 10,000	330,372	1,570,953	**430**	**149,468**	164,936	956,524	**414**	**81,388**
256	partial-5-11-u	> 10,000	164,158	730,719	**293**	**75,548**	74,250	394,516	**272**	**35,542**
257	partial-5-13-u	> 10,000	184,196	833,889	**312**	**84,348**	83,311	456,210	**291**	**40,198**
258	partial-5-15-u	> 10,000	217,930	1,002,997	**342**	**99,308**	102,569	572,569	**322**	**52,465**
259	partial-5-17-u	> 10,000	252,004	1,178,557	**371**	**114,268**	126,109	712,023	**356**	**63,443**
260	post-cbmc-aes-d-r2-noholes	> 10,000	276,895	1,607,567	**190**	**23,280**	4,736	211,270	**61**	**1,197**
261	post-cbmc-aes-ee-r2-noholes	> 10,000	266,199	1,575,975	**180**	**22,072**	4,345	209,253	**47**	**1,236**
262	post-cbmc-aes-ee-r3-noholes	> 10,000	498,723	2,928,183	**248**	**41,176**	8,006	384,066	**89**	**2,322**
263	rbcl_xits_09_UNKNOWN	> 10,000	1,430	79,453	**219**	**520**	871	82,700	**243**	**520**
264	rbcl_xits_10_UNKNOWN	> 10,000	1,584	91,646	**239**	**593**	1,003	87,270	**266**	**577**
265	rbcl_xits_11_UNKNOWN	> 10,000	1,740	104,640	**257**	**669**	1,114	100,241	**297**	**644**
266	rbcl_xits_12_UNKNOWN	> 10,000	1,898	118,441	**273**	**738**	1,240	114,014	**308**	**709**
267	rbcl_xits_13_UNKNOWN	> 10,000	2,058	133,055	**298**	**809**	1,350	128,608	**322**	**791**
268	rpoc_xits_09_UNSAT	> 10,000	1,430	87,044	**216**	**516**	871	75,119	**239**	**506**
269	rpoc_xits_10_UNKNOWN	> 10,000	1,584	100,116	**237**	**598**	1,003	95,730	**266**	**594**
270	rpoc_xits_11_UNKNOWN	> 10,000	1,740	114,011	**258**	**682**	1,114	109,602	**300**	**662**
271	rpoc_xits_12_UNKNOWN	> 10,000	1,898	128,735	**278**	**764**	1,240	124,298	**324**	**731**
272	rpoc_xits_13_UNKNOWN	> 10,000	2,058	144,294	**299**	**837**	1,350	139,837	**325**	**813**
273	rpoc_xits_14_UNKNOWN	> 10,000	2,220	160,694	**318**	**912**	1,488	156,189	**368**	**915**
274	safe-30-h29-unsat	> 10,000	130,928	437,794	**98**	**2,995**	33,684	238,732	**129**	**2,729**
275	safe-50-h49-unsat	> 10,000	620,037	2,098,249	**163**	**8,574**	150,826	1,130,095	**217**	**8,212**
276	simon-s02b-k2f-gr-rcs-w8	> 10,000	10,056	271,393	**380**	**3,800**	8,681	269,987	**659**	**3,702**
277	smulo128	> 10,000	188,473	564,650	**208**	**775**	97,667	422,957	**265**	**890**
278	sortnet-7-ipc5-h15-unsat	> 10,000	106,701	374,254	**110**	**5,176**	26,639	205,137	**168**	**5,895**
279	sortnet-8-ipc5-h18-unsat	> 10,000	340,179	1,187,589	**163**	**11,348**	83,984	655,309	**307**	**13,613**
280	sortnet-8-ipc5-h19-sat	> 10,000	361,125	1,254,773	**163**	**11,939**	89,426	689,968	**311**	**14,061**
281	total-10-17-u	> 10,000	293,864	1,370,917	**387**	**134,508**	74,747	500,234	**315**	**23,500**
282	total-10-19-u	> 10,000	331,202	1,571,183	**420**	**151,228**	81,759	576,937	**338**	**23,916**
283	total-5-15-u	> 10,000	219,756	1,008,653	**335**	**101,068**	67,330	407,288	**276**	**27,671**
284	total-5-17-u	> 10,000	265,064	1,236,517	**371**	**121,308**	82,969	516,795	**313**	**34,374**
285	UR-20-10p1	> 10,000	259,234	1,387,934	**445**	**79,975**	137,235	931,250	**445**	**75,593**
286	UTI-20-10p1	> 10,000	260,342	1,391,257	**116**	**79,974**	138,003	933,914	**118**	**75,709**
287	vmpc_34	> 10,000	1,156	194,072	**306**	**1,091**	1,155	145,571	**304**	**1,091**

Table 6: UNKNOWN 2: benchmarks from SAT2009 competition reported UNKNOWN by `precosat`