Journal on Satisfiability, Boolean Modeling and Computation 2 (2006) 229-237

*Research Note*

# Back to the SAT05 Competition: an *a Posteriori* Analysis of Solver Performance on Industrial Benchmarks

**Emmanuel Zarpas** zarpas@il.ibm.com
*IBM Haifa Research Laboratory*

## Abstract

This paper analyzes the SAT05 solver competition on industrial instances. We carefully investigate the performance of solvers from the competition and demonstration categories. We also present details on solver performance per subsets of the benchmarks per contributor and on SAT versus UNSAT instances. Finally we give recommendations for next SAT competition.

Keywords: *satisfiability, benchmarking, performances, metrics, industrial problems*

*Submitted October 2005; revised December 2005; published March 2006*

## 1. Introduction

This work analyzes the results of the second phase for the industrial category in the SAT'05 competition [2]. The purpose of the competition was to identify new challenging benchmarks, promote new solvers for the propositional satisfiability problem (SAT), and compare them with state-of-the-art solvers. We therefore believe that an analysis of solver performance is essential for learning relevant lessons from the competition. Our investigation carefully examines and compares the performance of solvers from the competition and the demonstration category. A description of the solvers, the benchmark, and the organization is available at the competition Web page [2].

## 2. Analysis of general results for the industrial category

The second stage of the SAT'05 industrial competition ran on a benchmark of 296 CNFs using Althon 1800 processors with 2 GB RAM [2]. Table 1 displays the number of timeouts (set at 12000 sec for the second phase of the competition) for solvers that ran during the second stage[1]. These results encompass solvers from the competition and the demonstration categories, except for vallst, which did not run on the entire benchmark. SateliteGTI clearly outperformed the other solvers according to the timeout metrics. However, this does not mean the other solvers are unable to solve cases that are beyond the capabilities of SateliteGTI.

To achieve a more precise comparison between solvers, we used a new methodology [6] to compare each solver with SateliteGTI. To compare a solver S with SateliteGTI, we discarded

[1] In this paper, all data come from the SAT'05 competition results.

**Table 1.** Solvers ranked by timeout value

| Solver | # timeout | rank | Solver | # timeout | rank |
|---|---|---|---|---|---|
| SateliteGTI | 29 | 1 | csat | 65 | 9 |
| eureka_a | 43 | 2 | zchaff_rand | 70 | 10 |
| siege4 | 46 | 3 | zchaff | 99 | 11 |
| minisat | 46 | 4 | compsat | 107 | 12 |
| jerusat1.31_b | 53 | 5 | sat4j | 116 | 13 |
| HaifaSat | 54 | 6 | hsat.5 | 143 | 14 |
| eureka_c | 56 | 7 | wllsatv1 | 204 | 15 |
| eureka_b | 58 | 8 | dew_satz_1a | 207 | 16 |

the cases where S and SateliteGTI both time out or both run in under five seconds. With the remaining data, we computed the following metrics: the geometrical mean of S/SateliteGTI speedups (geomean), the median of S/SateliteGTI speedups (median), and the "global" speedup (total time for S)/(total time for SateliteGTI) (global). The drawback of this method is that we compare the speedup of different solvers with that of SateliteGTI, even though they were not all computed on exactly the same benchmark. However, the differences are minimal both in number and in relevance, so the comparison remains relevant. This method also allows us to 'penalize' solvers that perform poorly on easy benchmarks, which is something that cannot be done by SAT'05 scoring mechanisms[2].

As can be seen in Table 2, SateliteGTI [3] remains in first place for the three metrics, while minisat holds second position for geomean and median, and is third for global. In the lower part of the table, we can see that the ranking for zchaff, compsat, sat4j, wllsatv1, and dew_satz does not change significantly with the metrics. This paper gives values for these metrics with two significant digits. However, it is important to remember that a small difference in the least significant digit for any of the metrics is probably not very significant. On the other hand, Jerusat1.31_b, which received a bronze medal at the SAT'05 competition, performs quite poorly according to the geomean, median, and global metrics. The eureka and siege4 [4] solvers, which did not take part in the competition, perform way below SateliteGTI. For the global metric, minisat is only outperformed by eureka_a. Note that siege4, a two-year-old solver, still exhibits good performance. The score, computed using the SAT'05 competition scoring algorithm, is given as additional information. Its mechanisms are such that taking into account the solvers from the demonstration category causes a change in the relative ranking of the other solvers. For example, the relative ranking of the competition solvers changed when the scores were computed with some of the demonstration category solvers. This explains why the score ranking in Table 2 is somewhat different from the SAT'05 competition results. Although the main goal of the competition was to rank the competing solvers, this observation indicates that the score may not be an accurate measure of the solvers' overall performance.

[2] For each CNFs the solvers share a solution and a speed score purses, for detailed explanation see [5].

**Table 2.** Results for industrial benchmark. #cases is the number of "relevant"CNFs on which the comparison was made. Score is computed using the SAT'05 competition scoring algorithm, without discarding any cases.

| Solver | geomean | rank | median | rank | global | rank | #cases | score x1000 | rank |
|---|---|---|---|---|---|---|---|---|---|
| SateliteGTI | 1 | 1 | 1 | 1 | 1 | 1 | | 82 | 1 |
| minisat | 1.3 | 2 | 1.7 | 2 | 1.8 | 3 | 187 | 60 | 2 |
| eureka_b | 1.7 | 3 | 2.8 | 5 | 2.2 | 6 | 193 | 45 | 5 |
| HaifaSat | 1.7 | 3 | 2.9 | 7 | 2.1 | 5 | 180 | 45 | 5 |
| eureka_a | 1.8 | 5 | 2.3 | 4 | 1.6 | 2 | 196 | 46 | 4 |
| siege4 | 2.1 | 6 | 2.2 | 3 | 1.9 | 4 | 194 | 49 | 3 |
| eureka_c | 2.2 | 7 | 2.8 | 5 | 2.2 | 6 | 198 | 45 | 5 |
| zchaff_rand | 2.7 | 8 | 3.7 | 8 | 2.3 | 8 | 206 | 41 | 9 |
| csat | 3.0 | 9 | 4.4 | 9 | 2.5 | 9 | 204 | 34 | 10 |
| Jerusat1.31_b | 3.5 | 10 | 5.5 | 11 | 2.5 | 9 | 201 | 44 | 8 |
| zchaff | 4.8 | 11 | 5.1 | 10 | 3.6 | 11 | 199 | 32 | 11 |
| compsat | 4.8 | 11 | 8.1 | 13 | 4.0 | 12 | 201 | 28 | 12 |
| sat4j | 6.2 | 13 | 6.1 | 12 | 4.6 | 13 | 199 | 24 | 13 |
| hsat.5 | 10.1 | 14 | 12.2 | 13 | 5.5 | 14 | 204 | 19 | 14 |
| wllsatv1 | 40.0 | 15 | 23.2 | 14 | 8.7 | 16 | 211 | 16 | 15 |
| dew_satz_1a | 129.6 | 16 | 28.9 | 15 | 8.5 | 15 | 215 | 12 | 16 |

## 3. Detailed analysis

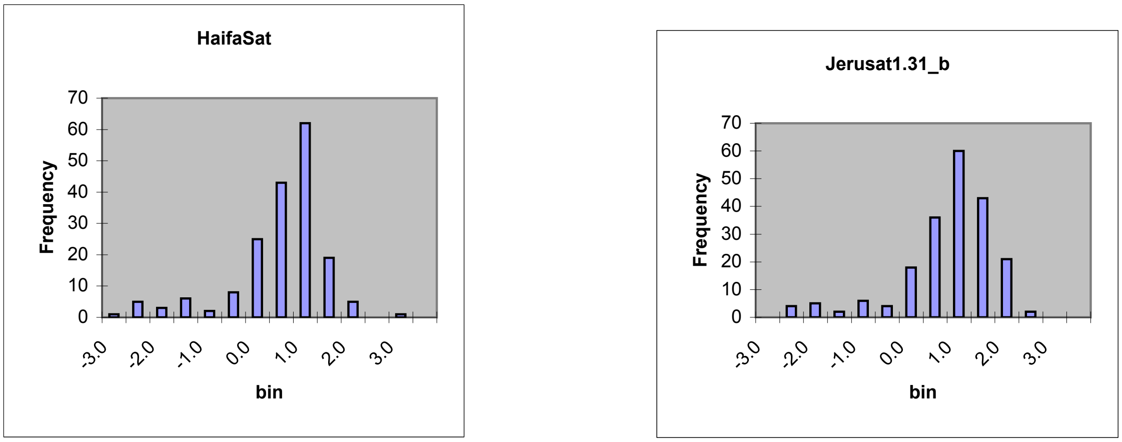
In this section, we present further details using three orthogonal views: a speedup histogram analysis on the overall sample, an analysis of the solvers performance on the different authors benchmarks, and an analysis on the SAT/UNSAT subsets.

### 3.1 Histogram analysis

In order to carry out a more precise analysis, the solvers should be compared two by two, as in [6], and not just using SateliteGTI as a reference. This is, however, not realistic for more than a few solvers. To increase accuracy, for each solver $S$, we looked at the histograms of the logarithms of the speedups $S$/SateliteGTI, displayed in Figures 1 and 2. Table 3 displays kurtosis and skewness (respectively measures of the "peakedness" and the asymmetry of the histogram) of the log(speedup) for eight of the solvers (on samples skimmed from the "irrelevant" cases as in Section 3). This confirms the feeling we get looking at the figures. The distributions are not normal, therefore the speedup's sample distributions are not log-normal. On one hand, the histograms have "high peaks" and on the other hand, many of them seem to be "heavy tailed". Somewhat surprisingly, the heavier tail is not the right but the left one; there are many cases where SateliteGTI performs very poorly. If not for these cases, the metrics defined previously (geomean, median, global, score) would favor SateliteGTI even more than the others solvers, aside from siege4.

**Table 3.** Skewness and kurtosis on the "relevant" sample of the industrial benchmark

| Solvers | Siege4 | zchaff | zchaff_rand | HaifaSat | minisat | Jerusat | eureka_a | eureka_b |
|---|---|---|---|---|---|---|---|---|
| kurtosis | 2.0 | 2.9 | 2.9 | 2.8 | 4.3 | 2.1 | 1.5 | 1.3 |
| skewness | -0.4 | -0.7 | -1.1 | -1.5 | -1.4 | -1.4 | -0.7 | -1.2 |



(a) (b)

**Figure 1.** Histogram of the log(speedup solver/SateliteGTI)

### 3.2 Analysis per author

For the second phase of the industrial category, benchmarks were provided by five contributors: grieu, maris, nagrain, velev, and zarpas [2]. The benchmark distribution is as follows: zarpas benchmarks are 58% of the original cases and 82% of the cases relevant for the minisat/SateliteGTI comparison; maris benchmarks are 23% of the original cases and 0% of the cases relevant for the comparison (maris benchmarks are too easy to be relevant); grieu benchmarks are 4% and 5%, respectively; nagrain benchmarks are 2% and 2%, respectively; and velev benchmarks are 13% and 11%, respectively. The zarpas benchmarks are clearly overwhelming, therefore, the global analysis of the results does not give a balanced view of the performance for different types of industrial CNFs. A separate analysis for each author is the more straightforward way to overcome this problem. However, maris cases are too easy to solve, and they are therefore unable to play a significant role in the solver comparison. Additionally, there are not enough nagrain cases for statistical analysis.

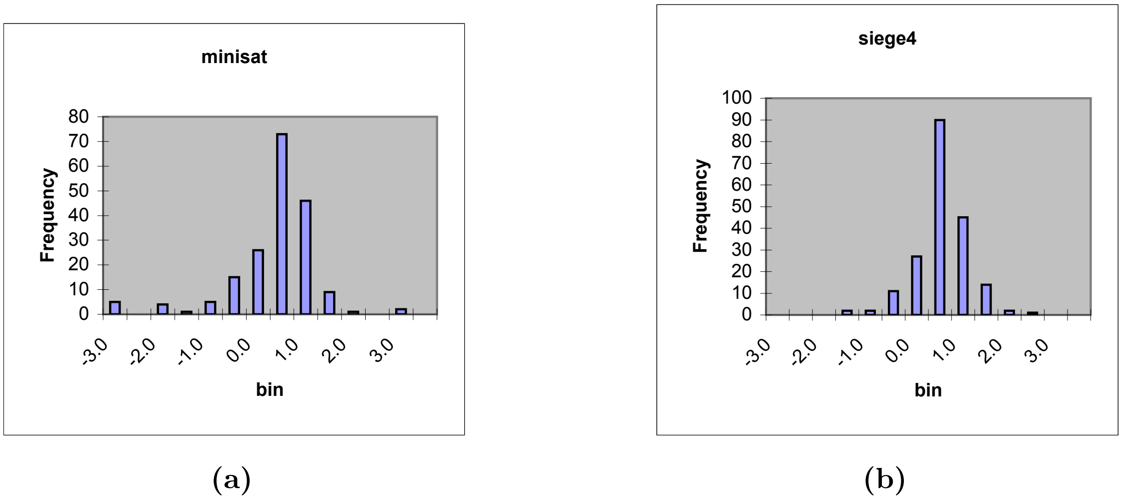


**Figure 2.** Histogram of the log(speedup solver/SateliteGTI)

**Table 4.** Results for grieu benchmark

| Solver | geomean | rank | median | rank | global | rank | #cases | Score<br>x100 | rank |
|---|---|---|---|---|---|---|---|---|---|
| Jerusat1.31_b | 0.2 | 1 | 0.4 | 1 | 0.6 | 2 | 10 | 35 | 3 |
| compsat | 0.2 | 1 | 0.6 | 5 | 0.6 | 2 | 8 | 24 | 5 |
| eureka_b | 0.4 | 3 | 0.5 | 3 | 1.0 | 7 | 10 | 26 | 4 |
| zchaff | 0.7 | 4 | 0.5 | 3 | 0.7 | 4 | 11 | 36 | 2 |
| HaifaSat | 0.8 | 5 | 0.8 | 7 | 1.3 | 13 | 9 | 16 | 6 |
| zchaff_rand | 0.8 | 5 | 0.9 | 9 | 1.0 | 7 | 13 | 60 | 1 |
| eureka_a | 0.8 | 5 | 0.8 | 7 | 0.7 | 4 | 10 | 16 | 6 |
| hsat.5 | 0.9 | 8 | 0.4 | 1 | 0.8 | 6 | 9 | 12 | 11 |
| SateliteGTI | 1 | 9 | 1 | 10 | 1 | 7 |  | 13 | 9 |
| minisat | 1.0 | 9 | 1.0 | 10 | 1.0 | 7 | 8 | 13 | 9 |
| siege4 | 1.1 | 11 | 1.6 | 14 | 1.7 | 14 | 8 | 12 | 11 |
| sat4j | 1.3 | 12 | 0.6 | 5 | 0.5 | 1 | 9 | 15 | 8 |
| csat | 1.5 | 13 | 1.1 | 12 | 1.2 | 11 | 9 | 9 | 14 |
| eureka_c | 1.9 | 14 | 1.3 | 13 | 1.2 | 11 | 9 | 10 | 13 |
| wllsatv1 | 5.1 | 15 | 2.4 | 15 | 2.0 | 15 | 9 | 7 | 15 |
| dew_satz_1a | 5.2 | 16 | 3.3 | 16 | 2.0 | 15 | 9 | 7 | 15 |

Table 4 shows that the solvers behave quite differently on grieu cases than they do for the entire industrial benchmark. Note that SateliteGTI pre-processing does not appear to improve minisat performance for grieu benchmarks. Arguably, there are not enough relevant grieu benchmarks to draw conclusions. Fortunately, there are more relevant velev benchmarks. As we can see in Table 5, the results are still different. For instance, on velev, SateliteGTI pre-processing does not improve, and even reduces, minisat performance. Results on the zarpas benchmark are not that different from the results on the whole industrial benchmark. However, there are a few discrepancies, as can be seen in Table 6. The above observations show that the results on the whole benchmark are not really relevant. The zarpas benchmark weight is far too heavy, yet there are still discrepancies between the whole benchmark and the zarpas benchmark.

**Table 5.** Results on velev benchmark. wllsatv1 and dew_satz_1a time out on all velev cases.

| Solver | geomean | rank | median | rank | global | rank | #cases | Score x100 | rank |
|---|---|---|---|---|---|---|---|---|---|
| minisat | 0.4 | 1 | 0.2 | 1 | 1.3 | 4 | 21 | 100 | 1 |
| eureka_a | 0.9 | 2 | 1.3 | 4 | 0.4 | 1 | 30 | 96 | 2 |
| siege4 | 0.9 | 2 | 0.8 | 2 | 0.8 | 2 | 27 | 64 | 4 |
| SateliteGTI | 1 | 4 | 1 | 3 | 1 | 3 |  | 58 | 5 |
| compsat | 1.5 | 5 | 1.4 | 5 | 1.4 | 5 | 22 | 44 | 7 |
| zchaff | 2.0 | 6 | 1.6 | 7 | 1.5 | 9 | 25 | 44 | 7 |
| zchaff_rand | 2.2 | 7 | 2.6 | 8 | 1.4 | 5 | 30 | 70 | 3 |
| csat | 2.5 | 8 | 3.6 | 10 | 1.3 | 4 | 41 | 46 | 6 |
| sat4j | 2.7 | 9 | 1.5 | 6 | 8.4 | 13 | 20 | 33 | 12 |
| HaifaSat | 3.0 | 10 | 3.4 | 9 | 1.4 | 5 | 25 | 42 | 9 |
| eureka_b | 4.4 | 11 | 22 | 13 | 2.5 | 11 | 26 | 39 | 10 |
| eureka_c | 4.7 | 12 | 15 | 12 | 2.4 | 10 | 26 | 39 | 10 |
| Jerusat1.31_b | 8.9 | 13 | 11 | 11 | 3.9 | 12 | 22 | 32 | 13 |
| hsat.5 | 14 | 14 | 23 | 13 | 9.3 | 14 | 21 | 28 | 14 |

### 3.3 Analysis on the SAT/UNSAT subsets

Up to now we studied the results on SAT and UNSAT CNFs. However, the solvers behave very differently on the SAT and UNSAT instances used for the competition. It is impossible to know whether this is specific to the benchmark. It might be caused by other criteria that should not be correlated with SAT/UNSAT in general, but would be correlated in the benchmark used for the competition. In this subsection, we discuss the performance on the SAT and UNSAT subsets of benchmark of SateliteGTI, minisat, siege4, eureka_a, Jerusat1.31_b, HaifaSat and zchaff_rand. The benchmarks encompass about 50% more SAT than UNSAT relevant CNFs (for the minisat/SateliteGTI, as in Section 3.2). Tables 7 and 8 show that performances on the SAT and UNSAT subsets of the benchmark are quite different. If we rank the solvers on UNSAT subset according to the geometrical mean of the speedup or the "global" speedup, SateliteGTI is no longer the fastest of the eight solvers, ranking third according to "geomean" and fifth according to "global". However,

**Table 6.** Results on zarpas benchmark

| Solver | geomean | rank | median | rank | global | rank | #cases | Score x1000 | rank |
|---|---|---|---|---|---|---|---|---|---|
| SateliteGTI | | 1 | 1 | 1 | 1 | 1 | | 64 | 1 |
| eureka_b | 1.5 | 2 | 2.8 | 4 | 2.3 | 4 | 151 | 28 | 6 |
| minisat | 1.5 | 2 | 2.1 | 2 | 2.0 | 2 | 154 | 37 | 2 |
| HaifaSat | 1.6 | 4 | 2.8 | 4 | 2.3 | 4 | 142 | 29 | 4 |
| eureka_c | 1.9 | 5 | 2.8 | 4 | 2.2 | 3 | 156 | 29 | 4 |
| eureka_a | 2.2 | 6 | 2.9 | 7 | 2.4 | 7 | 152 | 24 | 8 |
| siege4 | 2.4 | 7 | 2.3 | 3 | 2.3 | 4 | 155 | 31 | 3 |
| csat | 3.1 | 8 | 4.5 | 9 | 3.0 | 9 | 163 | 18 | 9 |
| zchaff_rand | 3.1 | 8 | 3.8 | 8 | 3.2 | 10 | 159 | 18 | 9 |
| Jerusat1.31_b | 3.5 | 10 | 5.3 | 10 | 2.7 | 8 | 162 | 27 | 7 |
| compsat | 5.9 | 11 | 10 | 14 | 4.6 | 11 | 164 | 12 | 13 |
| zchaff | 6.3 | 12 | 5.7 | 11 | 4.9 | 13 | 157 | 13 | 11 |
| sat4j | 7.0 | 13 | 8.3 | 12 | 4.8 | 12 | 161 | 11 | 14 |
| hsat.5 | 8.6 | 14 | 9.4 | 13 | 5.4 | 14 | 160 | 13 | 11 |
| wllsatv1 | 34 | 15 | 19 | 15 | 8.2 | 16 | 164 | 7 | 15 |
| dew_satz_1a | 110 | 16 | 23 | 16 | 7.8 | 15 | 166 | 3 | 16 |

**Table 7.** Results on SAT benchmark

| Solvers | geomean | median | global | kurtosis | skewness |
|---|---|---|---|---|---|
| minisat | 1.3 | 1.6 | 2.1 | 0.3 | -0.6 |
| siege4 | 2.4 | 2.2 | 2.5 | 0.4 | 0.3 |
| zchaff_rand | 5.0 | 5.5 | 3.5 | 5.2 | -0.6 |
| Jerusat | 4.0 | 3.5 | 2.5 | 0.6 | 0.3 |
| HaifaSat | 3.2 | 3.5 | 3.1 | 2.1 | -0.4 |
| eureka_a | 3.0 | 3.2 | 2.6 | 3.8 | -0.3 |

according to the median of the speedups, SateliteGTI still ranks first as in the official results of the competition. It would be tempting to conclude that the "median" is a more robust metric than "geomean" and "global", however kurtosis and skewness of the speedups' log are surprisingly different for the SAT and UNSAT benchmarks. In order to better understand these differences between SAT and UNSAT, we reviewed the runtimes. The median of the SateliteGTI and Solver (Solver being any one of minisat, siege4, zchaff_rand, Jerusat, HaifaSat, eureka_a) runtimes on the SAT subset relevant for SateletiteGTI/Solver is at least one order of magnitude greater than the median of the SateliteGTI and Solver runtimes

**Table 8.** Results on UNSAT benchmark

| Solvers | geomean | median | global | kurtosis | skewness |
|---|---:|---:|---:|---:|---:|
| minisat | 1.2 | 1.7 | 1.2 | 2.5 | -1.4 |
| siege4 | 1.7 | 2.2 | 0.9 | 3.9 | -1.4 |
| zchaff_rand | 1.2 | 2.8 | 0.6 | 1.0 | -1.3 |
| Jerusat | 3.7 | 7.2 | 2.7 | 0.7 | -1.3 |
| HaifaSat | 0.6 | 1.4 | 0.5 | 0.1 | -1.1 |
| eureka_a | 0.9 | 1.6 | 0.4 | -0.3 | -0.7 |

on the UNSAT subset relevant for SateletiteGTI/Solver. Therefore, it appears that the relevant SAT subset of the benchmark is significantly more difficult than the UNSAT one. Note that the CNFs, which always time out, are discarded. The difference in ranking between solvers on the SAT and UNSAT subsets might be explained by the fact that the UNSAT CNFs are easier than the UNSAT for the considered solvers (SateliteGTI, minisat, siege4, zchaff_rand, Jerusat, HaifaSat, eureka_a), and not the satisfiability of the CNFs.

## 4. Conclusions

This paper offers an in-depth analysis of the results of the second phase of the SAT'05 competition industrial category, and includes results of solvers from the demonstration category. This analysis offers a more complete understanding of the SAT'05 benchmarks and offers feedback on performances of the solvers. We hope this feedback will be useful in improving future performances. The best solvers from the competition are, roughly speaking, two times faster than siege4 (see Table 2). This means that even if the best competition solvers are significantly faster than the 2003 state-of-the-art, their performance is still within the same order of magnitude. The benchmarking done in [7] on a subset of the IBM CNF benchmark [8] suggests that E-solver (the OneSpin Solution SAT-solver) is significantly faster than siege4. This implies that the best competition solvers are not necessarily faster than the 2004 industrial state-of-the-art.

Our analysis shows that results on the whole benchmark do not necessarily reflect the performance on the benchmarks by different authors, nor do they reflect on the performance for different kinds of industrial applications. For example, solvers behave quite differently for the velev and the zarpas benchmark. Although both originate from formal hardware methods, the zarpas benchmark is generated by BMC and the velev is not. This examination shows that the industrial category is too broad. More precise results could be achieved by creating separate categories for the different types of applications. In future SAT competitions, benchmarks from different authors or applications could be balanced better, or new industrial sub-categories could be introduced (e.g., BMC, equivalence checking, etc.).

The lack of understanding of SAT solver runtime distribution, or speedup vs. reference solver distribution as considered in this paper, makes it very difficult to estimate which benchmarks are 'wide enough' or 'reasonably significant'. We believe that the issue of benchmark relevance and representativeness will be one of the main issues for the next

SAT competition. We suggest that the relevance of the benchmark be checked before the competition or during its first phase; then sub-categories could be created for each type of application.

## References

[1] A. Biere, E. Clarke, and Y. Zhu. Symbolic Model Checking Without BDDs. *Proc. of the Workshop on Tools and Algorithms for Construction and Analysis of Systems*, *LNCS* **1579**, 1999.

[2] D. Le Berre, L. Simon. SAT05 competition web page.
http://www.satcompetition.org/2005/.

[3] N. Eén, and A. Biere. Effective Preprocessing in SAT Through Variable and Clause Elimination. *Proc. of the SAT05 conference*, *LNCS* **3569**, 2005.

[4] L. Ryan. The siege satisfiability solver.
http://www.cs.sfu.ca/~loryan/personal/.

[5] Oliver Kullmann. The SAT 2005 Solver Competition on Random Instances. *Journal on Satisfiability, Boolean Modeling and Computation*, volume **2** (2006), pages 61–102.

[6] E. Zarpas. Benchmarking SAT Solvers for Bounded Model Checking. *Proc. of the SAT05 conference*, *LNCS* **3569**, 2005.

[7] E. Zarpas *et al.* Improved Decision Heuristic for High Performance SAT Based Static Property Checking. PROSYD Research Report, FP6-IST-507219, June 2004.

[8] CNF Benchmarks from IBM Formal Verification Benchmarks Library.
www.haifa.il.ibm.com/projects/verification/RB_Homepage/bmcbenchmarks.html.