Benchmarking SAT Solvers for Bounded Model Checking

Emmanuel Zarpas

IBM Haifa Research Laboratory,
zarpas@il.ibm.com

Abstract. Modern SAT solvers are highly dependent on heuristics. Therefore, benchmarking is of prime importance in evaluating the performances of different solvers. However, relevant benchmarking is not necessarily straightforward. We present our experiments using the IBM CNF Benchmark on several SAT solvers. Using the results, we attempt to define guidelines for a relevant benchmarking methodology, using SAT solvers for real life BMC applications.

1 Introduction

Over the past decade, formal verification via model checking has evolved from a theoretical concept to a production-level technique. It is being actively used in chip design projects across the industry, where formal verification engineers can now tackle the verification of large industrial hardware designs. Bounded Model Checking (BMC) [1] has recently enabled the verification of problems that used to be beyond the reach of formal verification. However, BMC performance relies heavily on the performance of the underlying SAT solver.

We conducted several experiments using the IBM CNF benchmark [12, 11] to evaluate SAT tools. Often, we found discrepancies between our results and the experimental results found in the literature. For example, our results are not necessarily in line with the results of the SAT03 contest [4]. The main goal of this paper is to outline a methodology for benchmarking SAT solvers on Bounded Model Checking problems.

The paper is organized as follows: In Section 2, we present experimental results for some famous SAT solvers. Section 3 presents a new version of the IBM Benchmark and discusses correlation of the CNF characteristics and solvers performance. Section 4 contains a discussion on SAT benchmarking for BMC, where we try to learn from our experimental results in light of other experiments such as the SAT contests. Section 5 concludes the paper.

2 Experiments with IBM 2002 Benchmark: SAT Solver Comparison

Our experiments compared four famous SAT solvers: zChaff I (2001.2.17 version) [7] , zChaff II (zChaff 2004.5.13) [3], BerkMin561 [2] and Siege_v4 [8]. We used the

F. Bacchus and T. Walsh (Eds.): SAT 2005, LNCS 3569, pp. 340–354, 2005.

2002 version of the IBM CNF Benchmark[12], whose CNFs are generated through BMC from the IBM Formal Verification Benchmark Library[11]. This library is a collection of models from real-life industrial hardware verification projects.

For each model, we used the CNF formulas (SAT_dat.k.cnf) for bound with k=1, 10, 15, 20, 25, 30, 35, 40, 45, 50. The time-out was set at 10,000 seconds on a workstation with 867841X Intel(R) Xeon(TM) CPU 2.40GHz, with 512 KB first level cache and 2.5 GB physical memory. We used the default configuration for each engine.

2.1 zChaff I vs. BerkMin561

We ran zChaff I (2001.2.17 version) and BerkMin561 on the 2002 version of the IBM CNF Benchmark[1]. Table 1 is a summary of the results obtained. More

Table 1. The results are displayed in seconds and include the timeout 10,000 seconds

	zChaff	BerkMin561
Total time (10000 sec time-out)	344765	414094
First (# of CNF where the engine is the fastest)	298	131
# of unsolved problems (timeout reached)	25	30
+ (# of CNF where the engine is the fastest by more than a minute and 20%)	67	32
First by model (# of model where the engine is the fastest)	28	18

complete experimental results are presented in Table 8, where for each model in the table, we present the sum of the results of SAT_dat.k.cnf with k=1, 10, 15, 20, 25, 30, 35, 40, 45, 50 (*i.e.*, the BMC translation of each model for the different k). For more details, see the complete results in [13]. Because SAT solvers do not always behave in homogeneous manner (*Cf.* detailed results in [13] or table 2), the complete results analysis should not be disregarded.

The results show that zChaff and BerkMin561 achieved close results. On some CNFs, zChaff runs faster, and on others, BerkMin561 runs faster. In most cases, the differences in their performance is not very significant, the highest speed is not faster by more than one minute or 20%. However, while zChaff seems to perform slightly better overall, BerkMin561 gets better results than zChaff on the UNSAT CNFs. From these results, it is not possible to conclude that BerkMin561 performs better than zChaff. This is not consistent with what can be read in the literature or with the final results of the SAT03 contest (Cf. section 2.4).

This is not a very satisfying conclusion and the previous metrics do not tell us in a simple way how much faster zChaff is than Berkmin561[2]. We need some

[1] BerkMin561 was second in the SAT03 contest industrial benchmarks category in the SAT03 contest; the winner Forklift is not publicly available.

[2] It is very important to be able to give clear and concise values for results dissemination.

Table 2. The results are displayed in seconds for the CNFs from 18_rule model. The time-out was set to 10000 seconds. zChaff performs the worst for k=15, 20, 35; BerkMin561 performs the worst for k=30, 40, 50; Siege_v4 has the best performance, except for k=15, 25

18_rule CNF	Result	zChaff	BerkMin	Siege
SAT_dat.k1.cnf	unsat	0	0	0
SAT_dat.k10.cnf	unsat	1	1	0
SAT_dat.k15.cnf	unsat	13	4	5
SAT_dat.k20.cnf	unsat	65	47	41
SAT_dat.k25.cnf	unsat	951	109	251
SAT_dat.k30.cnf	sat	700	755	557
SAT_dat.k35.cnf	sat	time-out	2230	1730
SAT_dat.k40.cnf	sat	5510	8060	247
SAT_dat.k45.cnf	sat	time-out	time-out	959
SAT_dat.k50.cnf	sat	5010	time-out	1370

additional metrics in order to compare these SAT solvers. The arithmetic mean of the speedup seems irrelevant. Let's say we want to compare the performances of solvers A and B. If the speedup of A vs. B is 10 on test_1 and 0.0001 on test_2, the arithmetic mean of the speedup (A vs. B) would be 5 and the mean of the speedup (B vs. A) would be 5000. This is clearly not relevant. The global speedup (Total time A / Total Time B) is good to have but the "long runs" have far more weight than the short ones. The median is much more interesting. however it does not take into account the extreme values (*e.g*, if on test_1 the speedup is equal to $median + 10$ or to $median + 0.1$, it does not change the median value). On the other hand, the median insensivity to extreme values makes it less dependent on the time-outs. The geometrical mean seems to be a good solution. It definitely takes into account the weight of the "negative speedup" (*i.e.*, a speedup of less than 1)[3].

We felt these values should be computed these values only on a relevant subset of the benchmark. For example, we discarded all the cases where the four SAT solvers time-out. In addition, we discarded cases for which the four solvers (zChaffI, Berkmin561, Siege_v4, zChaffII) run in less than five seconds. The rational is that it is difficult to accurately compare very small runtimes and that these very small runtimes are arguably not relevant. The results displayed in Table 3 (b) show that Berkmin561 is nearly two times slower than zChaff I, but that Berkmin behaves somehow better in the "long" cases.

2.2 Siege_v4

Siege was hors-concours for the SAT03 contest, therefore it did not participate in the second stage. Nevertheless, the Siege results were pretty good for the

[3] Note that the logarithm of the geometrical mean is equal to the arithmetical mean of the logarithms.

Table 3. With a 10000 sec. time-out

	Total Time (in seconds)	#time-outs
zChaff I	344,764	25
berkmin561	414,035	30
Siege_v4	197 239	14
zChaff II	389,304	31

(a)

Speedup	$\frac{zChaffI}{Berkmin}$	$\frac{zChaffI}{Siege}$	$\frac{zChaffI}{zChaffII}$
median	0.55	2.90	1.23
geomean	0.41	2.87	1.29
global	0.75	3.58	0.75

(b)

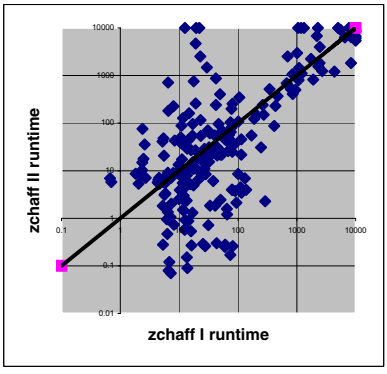
first stage and Siege has a reputation for being one of the best SAT solvers available. We ran Siege_v4[8] on the benchmark using 123456789 as a seed. Table 3 displays the overall conclusions. For more details see Table 8 and [13]. Siege_v4 is the fastest in 298 cases (CNFs) out of 498. In many difficult cases, Siege_v4 is fastest by an order of magnitude, or more. In some cases, Siege_v4 performed significantly worse than zChaff or BerkMin561. For example, for 26_rule, Siege_v4 is slower than zChaff by an order of magnitude and slower than BerkMin by two orders of magnitude. In addition, within the time-out, several CNFs can be solved only by Siege_v4 [4]. In conclusion, we see that Siege_v4 performs significantly better (roughly speaking 2.5 times faster) than zChaff I, Berkmin561 and zChaff II on the IBM CNF 2002 benchmark.

2.3 zChaff II

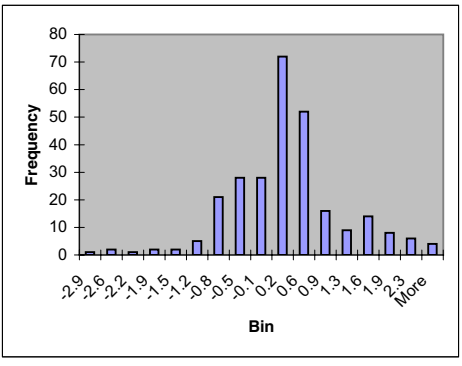
The industrial category of the SAT04 competition [6] was won by zChaff II (zChaff 2004.5.13). However "black-box" solvers such as Forklift (the 2003 edition winner) were "hors-concours" and as such not allowed to enter the second stage of the competition. We focus here on the performance of zChaff II vs zChaff I. zChaff II is faster than: zChaff for 229 CNFs (out of 442), Berkmin561 for 208 CNFs (out of 442), Siege_v4 for 82 CNFs (out of 442). In addition, zChaff II reached time-out more often than zChaff I (for six more cases). Table 3 displays a synthetic view of the results. The zChaff II code was left unchanged, so it used random seeds. Siege_v4 is roughly three time faster than zChaff I. Figure 1 gives us a graphic view of the performance of zChaff II vs. zChaff I. The overall performance difference between zChaff I and zChaff II on our benchmark is small[5], even though the two solvers can behave very differently in specific cases.

[4] Siege is a randomized solver, therefore, it could be argued that the comparison is not fair since the zChaff and BerkMin561, versions we used were not randomized. Nevertheless, even a deterministic solver can be lucky and providing Siege with a seed of Siege makes it deterministic.

[5] It should be note that zChaff II is randomized, so the results could be different for other runs. It would be interesting to have a statistical description of the range of zChaff II performances.



(a)



(b)

Fig. 1. Histogram (b) gives the distribution of the decimal logarithm of zChaff I/zChaff II speedup with 10000 sec time-out

2.4 Comparison with SAT Contest Results

In order to understand whether or not our results are consistent with those of the SAT03 contest[6] (for details results see [5]), we had a look at the results of the first and second stages of the competition.

First stage. Looking at the results[7] for the industrial category [5], we note the following:

- Series *13_rule_*[8] is over-represented (Cf. Table 4). Besides, since all solvers time-out on most of the CNFs from this series, it is not very meaningful.
- Except for series *13_rule_* and *11_rule_*, the series from the IBM CNF benchmark are easy for zChaff, BerkMin561, and Siege_v1.
- Results for *rule_07* are not consistent with our own experiences. This discrepancy is probably caused by the "Lisa syndrome"[4]: CNFs were shuffled for SAT03 and solver performance can differ dramatically between a shuffled CNF and the original.

The SAT03 contest results for BerkMin561, Forklift, Siege_v1, and zChaff on (shuffled) series from the IBM CNF Benchmark are summarized in Table 4. If results from series *07_rule* and *13_rule_* are discarded, zChaff shows better results than BerkMin561.

[6] In the first stage of the SAT04 competition, zChaff I and zChaff II have very close results and the solvers that outperformed them (Forklift, Oepir) are not available for experimentation.

[7] The Siege version used for the SAT03 contest is Siege_v1.

[8] The names of the SAT03 series appears in *italics* in order to differentiate them from the series we used in our experiments. For the exact composition of the SAT03 series, see [5].

Table 4. Partial results from first stage SAT03 contest

	BerkMin561	forklift	Siege_1	zChaff
Total # of Solved Benchmarks	112	112	112	101
Total CPU time needed (sec)	105000	103000	103000	114000
without *13_rule_* (sec)	1510	518	408	4160
without *13_rule* and *07_rule* (sec)	1410	424	268	553

We believe that the differences in results for the first stage of SAT03 contest results for IBM the benchmark and our experiments are due to clause shuffling in SAT03 and to the fact that the SAT03 experiment used a smaller test-bed of CNFs from the IBM benchmark.

Second stage. During the second stage of the competition, solvers were ranked according to the number of CNFs they could solve from a restricted benchmark. Forklift was ranked first, Berkmin561 second, and zChaff I sixth on the industrial benchmark[9].

Clearly, the respective zChaff I and BerkMin561 ranking do not correspond to our experimental results (see Table 3). However, in the second stage, all solvers "timed-out" on the IBM benchmarks selected. In other words, the ranking of the solvers selected for the second stage of the competition did not take into account performance on the IBM benchmarks. This probably explains why our evaluation of zChaff and BerkMin561 on the IBM CNF Benchmark gives results that are not in line with the second stage results (on industrial benchmarks) of the SAT03 contest.

3 Experiments with IBM 2004 Benchmark: Search for Correlation

3.1 Description of the IBM 2004 Benchmark

Bounded Model Checking translation technology is continuously evolving. Therefore, we re-generated the IBM CNF Benchmark from the IBM Model Benchmark using an alternative BMC tool. We ran the new translation for the following bounds $k = 0..10, k = 11..15, k = 16..20, k = 21..25, \ldots, k = 95..100$. The 2004 CNFs are available on-line. In this paper, we refer to the 2002 and 2004 versions as the old and the new benchmarks, respectively .

Table 5 and 6 present statistical descriptions of the old and the new benchmarks. In these two tables, average is the arithmetical mean, STDEV is the standard deviation, $\frac{clauses}{var}$ is the ratio between the number of clauses and the number of variables, %unit is the percent of unit (*i.e.*, of length one) clauses in a CNF, %bin is the percent of clauses of length two, %ter is the percent of clauses

[9] Siege was hors-concours, therefore not selected for the second stage.

Table 5. Old (2002) benchmark

	var	clauses	$\frac{clauses}{var}$	%unit	%bin	%ter	%l=4	$\%l > 4$
average	80,167	343,826	4.12	0.3	75.5	14.2	3.5	6.5
median	54,857	220,180	4.08	0.2	77.0	12.2	3.5	6.6
STDEV	81,924	388,156	0.52	0.3	6.4	7.7	1.0	1.3
max	636,089	3,172,107	5.42	1.6	84.5	55.3	5.2	9.1
min	3,645	14,681	2.48	0	40.8	4.3	0.1	0.1

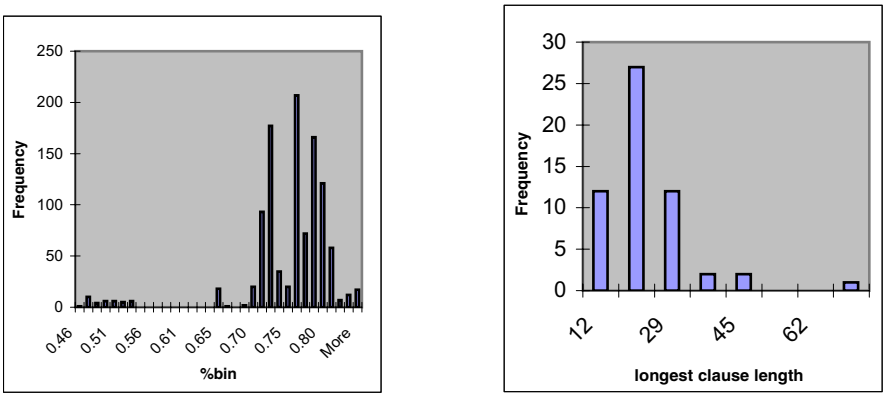
Table 6. New (2004) Benchmark

	var	clauses	$\frac{clauses}{var}$	%unit	%bin	%ter	%l=4	$\%l > 4$
average	73,414	305,301	3.99	0.6	74.5	15.0	3.6	6.3
median	50,897	195,612	3.98	0.4	76.1	13.1	3.6	6.6
STDEV	75,553	349,248	0.45	0.6	6.1	7.3	0.9	1.2
max	565,889	2,760,502	5.48	4.6	84.5	51.6	4.9	9.1
min	3,606	14,104	2.55	0	46.5	4.4	0.1	0.1

of length three, %l=4 is the percent of clauses of length 4 and $\%l > 4$ is the percent of clauses longer than 4.

These two tables tell us that the new benchmark CNFs are roughly 10% smaller than the old benchmark CNFs. However, they encompass roughly the same proportion of length two, three, and four clauses. The higher proportion of unit clauses in the new benchmark is due to specific optimizations. We also see that the ratio $\frac{clauses}{var}$ is slightly lower in the new benchmark (see the next subsection for a discussion on the relevance of this ratio). Figure 2 displays the histogram of %bin, where two models (07 and 13_1) have a "non standard" percent of binary clauses and are in the 45%-55% and not in the 70%-85%.

We looked at some statistics for the structure of the CNFs of the new benchmark. We found that for any model of the benchmark, there are four real numbers a, b, c, d such that for all the CNFs of the model, $\#(variables) \approx ak + b$ and $\#(clauses) \approx ck + d$, with k being the bound used to generate the CNFs. In a more general way, for any model and for any integer n strictly greater than 0, there are two real numbers e and f such that for all CNFs of the model #(clauses of length n) $\approx ek + f$ (of course e and f can be null, for example in most models, the number of unit clauses is a constant). The correlations (for a discussion about statistic techniques for NP-complete problems see [10]) between the series from the benchmark and the series predicted with the previous equations are about 0.99. This is not surprising since the CNFs are generated from the model in a way essentially linear to the bound. In addition, we found out that, in our benchmark, the length of the longest clause is the same for all CNFs generated from the same model. Figure 2 (b) displays the histogram of the longest clauses in the new benchmark. Note that two models, 07 and 13_1, which have the greater longest clauses, are also the ones with a "non standard" percent of binary clauses.



(a) **(b)**

Fig. 2. Histograms (a) and (b) give respectively the distribution of %bin and length of the longest clause

3.2 Experimentations and Correlations

How do zChaff I and Siege behave on the new benchmark? We ran zChaff I and Siege_v4 on both benchmarks (for $k = 10, 15, 20, \ldots, 50$) on the same workstation described in Section 1. Figure 3 displays the overall results.

Strangely enough, the "long" cases of the new benchmark are more difficult for zChaff I but easier for Siege_v4. For both solvers, the correlation between performance on old and new benchmark is somewhat higher than 0.7. The correlation between the zChaff I and Siege runtimes on the new benchmark is also 0.72.

Can the "hardness" of a CNF be predicted from the value of the ratio (number of clauses)/(number of variables) ? We could not find any relevant correlation between this ratio and the zChaff I and Siege performances. In fact, as we saw previously, for each model, #clauses and #var are strongly correlated with the bound k; therefore, for each model, there are a, b, c, d four real numbers, such that, the ratio is also strongly correlated with $(ak + b)/(ck + d)$. This explains the appearance of Figure 4.

Can we predict the "hardness" of a CNF from the number of variables? This could also be the number of clauses, since both numbers are strongly correlated. In our experiments we found a relatively low (about 0.3) correlation between the number of variables and the runtimes for zChaff I and Siege. However, when we "standardized"[10] the runtimes, we got a higher correlation (about 0.7). See Figure 5 for illustration.

[10] For a given SAT solver, we standardized the runtimes in the following way: For each model, we divided each of the nine runtimes (corresponding to the nine bounds $k = 0..10, k = 11..15, k = 16..20, \ldots, k = 46..50$) by the greatest of these nine runtimes. The idea is to be able to compare results between different models.

Speedup	zChaff I	Siege
median	1.69	1.07
geomean	2.13	0.97
global	1.02	1.19

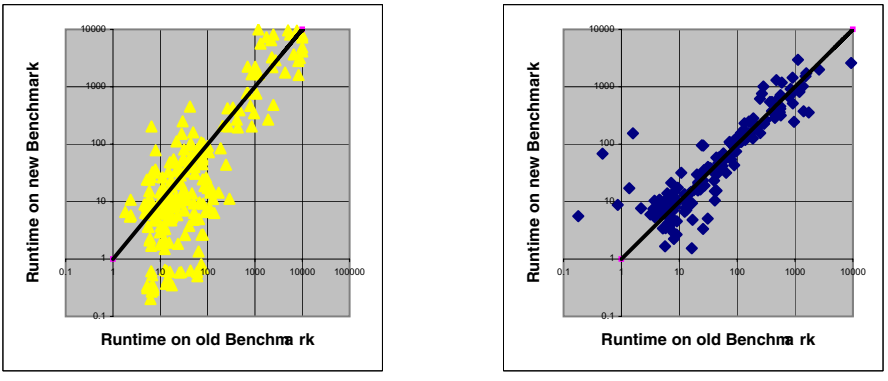


Fig. 3. Performance comparison between the old and the new benchmark. Time-out: 10000 sec. (a) and (b) display zChaff I and Siege runtimes on the old benchmark vs. on the new benchmark. In both comparaisons, we discarded the cases which ran in less than 5 seconds in both cases or which time-out in both cases

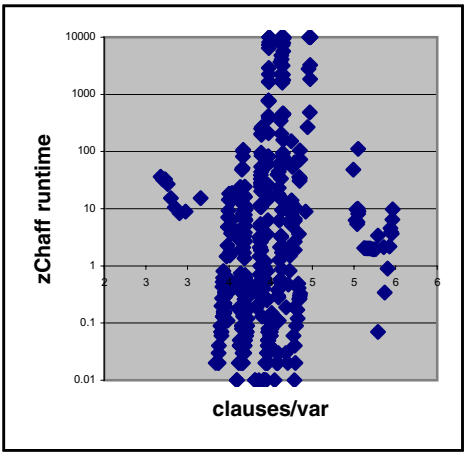
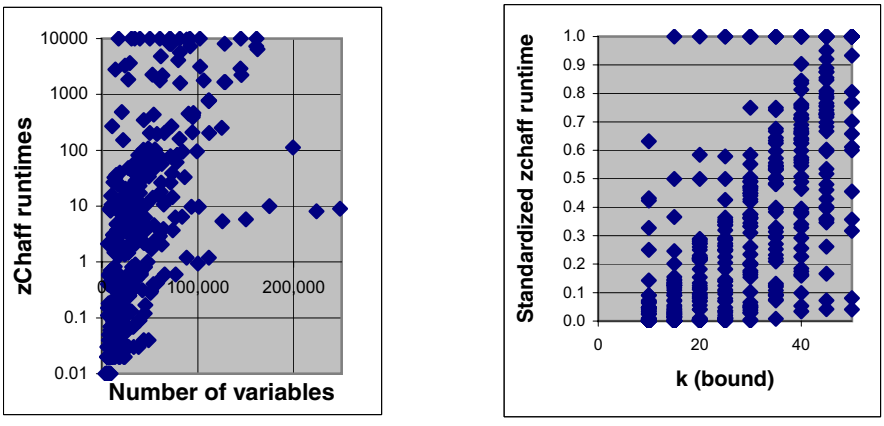


Fig. 4. $\frac{\#clause}{\#var}$ vs zChaff I runtime

Can we predict the "hardness" of a CNF from the proportion of clauses of length one, two, three and four ? We did not find any relevant correlation between the proportion of clauses of lenght two, three, four and more and the zChaff I and



(a) **(b)**

Fig. 5. (a): zChaff I runtime vs bound k, (b): standardized zChaff I runtime vs bound k

Table 7. Performance on SAT/UNSAT CNFs

		zChaff I	Siege	speedup
All CNFs	median	30	22	2.28
	geomean	56.52	31.80	1.75
	Total	192,323	44,632	
	global			4.31
UNSAT (109 CNFs)	median	18	12	2.46
	geoman	39	20	1.94
	total	77,120	14,642	
	global			5.27
SAT (67 CNFs)	median	55	59	1.84
	geoman	98	66	1.48
	total	115,203	29,989	
	global			3.84

Siege runtimes. We did find a low negative correlation (about -0.4) between the standardized proportion of unit clauses and the runtimes. We think this can be explained by the fact that for most models, the unit clause number is constant, so $\%unit \approx \frac{c}{ak+b}$, which explains the correlation.

Are "SAT" and "UNSAT" CNFs as hard? This is difficult to say since it is obviously not possible to compare performance on the same CNFs. It is not relevant to compare the geometrical mean of the runtime on the SAT and on the UNSAT CNFs, as displayed in Table 7. However, it is easier to compare the behavior of two SAT solvers on SAT and UNSAT CNFs. When we look at Table 7, we see that Siege behaves slower than to zChaff I on SAT CNFs as opposed to UNSAT CNFs.

4 Lessons for Benchmarking

Our experimental results, especially the comparison between zChaff and BerkMin561, are not in line with other results, such as those from SAT03[4]. In this section, we attempt to learn lessons from the previous experiments.

Use relevant benchmarks. To compare two SAT solvers for BMC, it is extremely important to use relevant benchmarks from BMC applications. For instance, results from theoretical problems such as 3-SAT problems, hand-made problems, or problems from applications others than BMC (such as planning), are not relevant and will not provide a good comparison between SAT solver performance for BMC applications. It is not sufficient to use only the CNFs generated by BMC; the models from which the CNFs are generated should be relevant. Therefore, it is of prime importance to use models from real-life industrial verification projects.

Use relevant time-out. The time-out used for the SAT03 contest was 600 seconds or 900 seconds for the first stage, and up to 2400 seconds for the second stage of the competition. For our experimentation, we used a 10000 second time-out. From the detailed experimental results [13], it is clear that the lower the time-out, the less relevant the results. Additionally, the lower the time-out, the more time-out results would be received for all the SAT solvers. Consequently, these time-outs can actually cover up different situations. For example, in our first experiments for CNF k45 from model 11_rule_1, with a time-out of 900 seconds, zChaff and BerkMin561 both timed out. However, with a 10000 second time-out, we realized that one solver performs seven times faster than the other on this CNF.

The ideal solution would be not to use any time-out or at least use very long time-outs (e. g., one week). In this way, the results for the SAT solvers can always (or almost always) be compared for a given CNF (at least a "reasonable" CNF). The obvious drawback is that this makes the experiments very, very long, hence limiting the scope of such experiments. In the SAT contests, the choice of a short time-out allows a great number of SAT solvers to be run against a very broad spectrum of CNFs. In our experiments, we ran a very limited number of SAT solvers against a smaller (but more relevant for BMC) number of CNFs with a longer time-out. We believe that this is one of the main explanations for the difference in results between SAT03 and our experiments. More precisely, if a longer time-out would have been used for the second stage of the SAT03 contest, some solvers would probably have solved the benchmark from the IBM CNF Benchmark; therefore, the final results would more likely have been consistent with our experiments. In summary, using shorter time-outs can be very useful for larger scopes experiments with many SAT solvers and very broad benchmarks. However, to get more relevant results for BMC[11], these kinds of experiments must be refined by limiting the scope (e. g., the number of SAT solvers).

[11] The SAT contest goals are not identical to ours. We are mainly interested in performance for real-life BMC problems, while the SAT contests try to establish "global" performance.

Shuffling clauses in benchmark CNFs is tricky. Shuffling clauses in benchmark CNFs can have a dramatic effect [4] and potentially change the ranking for solvers performance. However, in real life, SAT users don't shuffle their CNFs. One of the major differences between our experiments and SAT03 is that we did not shuffle CNFs.

Use a broad benchmark: easy for a SAT solver, does not mean easy for all. When assessing the performance of a new SAT solver, a common trend is to run the new solver against CNFs that are difficult to solve with a well-established solver (e. g., zChaff). Although it seems reasonable, this can be very misleading. Modern SAT solvers rely heavily on heuristics; therefore, a CNF can be solved very easily with one solver and still be very difficult with others. For example, the CNFs from model 01_rule are very easy for zChaff to solve but difficult for BerkMin561. On the 01_rule series, zChaff typically runs faster than BerkMin561 by two orders of magnitude. On the 26_rule series Siege_v4 typically runs slower than zChaff by one order of magnitude and slower than BerkMin561 by two orders of magnitude. The reason for this kind of behavior is not that some models are "special", but more likely the limitation of the heuristics used by different solvers. The fact that solver performance ranking often varies for different CNFs generated from the same model[12] comforts us with the idea that the relationship between solvers (and heuristics) performance and model specifics is not as strong as one would think.

Discard cases with unrelevant runtimes. It does not make a lot of sense to use a CNF C to, say, discriminate between two solvers A and B, if both A and B solve C in a very short period of time. Indeed, it is likely that the time sampling would not be accurate and relevant, especially because most solvers run on Unix/Linux machines. In the second part of our experiments, we took 5 seconds as a threshold.

For results analysis, there is no real reason to discriminate between results for satisfiable and unsatisfiable CNFs. We believe that it does not make much sense to differentiate between the performance of solvers for satisfiable and unsatisfiable CNFs – at least for BMC applications. Firstly, BMC is usually run in an incremental manner (e.g., $k = 0 \ldots 10, k = 11 \ldots 15, \ldots$). Therefore, before you can get a satisfiable result you often have to get several unsatisfiable results. Secondly, some models cannot be falsified and CNFs generated by BMC will always be unsatisfiable. Therefore, only the global performance is really relevant for real-life BMC SAT use.

Use several metrics for comparisons. Since SAT solvers rely heavily on heuristics, it is unlikely (or at least rare) that a SAT solver would be better on all possible CNFs (from real-life models). For example, even though Siege_v4 performs better than zChaff and BerkMin561 on most CNFs of the IBM benchmark, it does not perform better for all (e.g., model 26 in Table 8). The following metrics can be used to compare two SAT solvers, solver 1 and solver 2:

[12] In addition, Table 2 shows that the evolution between the bound k and the runtime is not the same for the three solvers.

Table 8. The results are displayed in seconds. The time-out was set to 10000 seconds. For each model, the number of time-outs, if any, appears in brackets

Model	zChaff	BerkMin561	Siege_v4	Model	zChaff	BerkMin561	Siege_v4
01_	166	22800 (1)	262	14_2	3490	2480	378
02_1_1	347	34	38	15_	6	21100 (1)	1
02_1_2	336	69	104	16_2_1	0	10	0
02_1_3	26	16	10	16_2_2	0	9	1
02_1_4	34	12	13	16_2_3	0	11	0
02_1_5	32	16	19	16_2_4	3	16	0
02_2_	118	83	11	16_2_5	1	17	2
02_3_1	96	157	22	16_2_6	1	18	1
02_3_2	702	34	13	17_1_1	0	509	0
02_3_3	154	133	18	17_1_2	219	427	70
02_3_4	333	39	13	17_2_1	1	28	0
02_3_5	126	172	25	17_2_2	1	98	0
02_3_6	396	34	16	18_	32200(2)	31200 (2)	5160
02_3_7	262	86	39	19_	127	2910	482
03_	190	733	126	20_	21800 (1)	9590	4020
04_	170	597	161	21_	150	2245	383
05_	33	267	124	22_	5290	5980	582
06_	296	1140	130	23_	38700 (2)	28500 (1)	2820
07_	61	73	69	26_	155	26	2850
09_	7	1	2	27_	53	364	134
11_1	5080	21900 (1)	1760	28_	385	843	96
11_2	6640	14500 (1)	982	29_	35700 (2)	58200 (5)	16700
11_3	24100 (2)	24000 (2)	3135	30_	76600 (7)	72000 (7)	66000(5)
12_	0	0	0	14_1	191	719	126
13_1	90000 (9)	90000 (9)	90000 (9)				

- Global times or the global speedup. This presents two drawbacks: 1) There is no perfect solution to take time-outs into account, and 2) All the weight is on the CNFs that take the longest to be solved.
- Ratio between the number of CNFs solved more quickly by solver 1 and the number of CNFs solved more quickly by solver 2) Optionally, only the cases where performance differences are significant can be taken into account. The median of the speedups is a better and more concise metric.
- The geometrical mean of the speedup is also a very interresting metric. However, it is also quite sensitive to the time-out definition.
- Time-out numbers. There are quite relevant if one considers that the time-out value is roughly equivalent to the real-life time-out (*i. e.*, the maximum reasonable waiting time of the real-life formal tool users). On other hand, this metric can easily give a biased view of reality. Imagine, for example, that the geometrical mean of the speedups for solver A vs. solver B. is 3, but solver A time-out slightly more often than solver B, should we really conclude that solver A is the slowest.

Each of these metrics have their pros and cons. We believe that the best solution is to use more than one of them to compare two SAT solvers. We believe that it is difficult to decide which solver is the best when, for example the geometrical mean, the median, and the global speedups give contradictory results.

5 Conclusion

In this paper, we showed that benchmarking is not a trivial task and that it can be misleading. For example, our experiments on zChaff and BerkMin561 present results that are contradictory with what is commonly accepted by the SAT community (*i.e.* BerkMin561 outperforms zChaff I). We also showed that it can be difficult to compare two SAT solvers (e. g., CNFs that are difficult for zChaff I are not the same as CNFs that are difficult for BerkMin561). Even when a SAT solver such as Siege_v4 seems to clearly outperform BerkMin561 and zChaff, it is not necessarily so for all benchmark CNFs. In order to help benchmarking and compare between solvers results, we proposed guidelines that sketch out a rough methodology. We believe that the systematic use of such a benchmarking methodology can improve the general quality of experimental performance evaluations for SAT and will help to produce practical and fundamental advances in the area. We plan to keep investigating possible correlations between CNFs static characteristics and the performances of SAT solvers.

Acknowledgment. The author wishes to thank Daniel Le Berre for his valuable comments and remarks on an earlier version of this paper.

References

1. Biere A.*et al*, "Symbolic Model Checking Without BDDs". *Proc. of the workshop on Tools and Algorithms for Construction and Analysis of Systems*, LNCS 1579, 1999.
2. Goldberg E., Novikov Y., "BerkMin: a Fast and Robust SAT-solver" *Proc. of the Design, Automation and Test in Europe*, IEEE Computer Society, 2002.
3. Mahajan Y., Fu Z., Malik S., "Zchaff2004: An Efficient SAT Solver". To appear in *SAT 2004* Special Volume.
4. Le Berre D., Simon L., "The essentials of the SAT 2003 competition", *Proc. of the 6th Int'l Conf. on Theory and Applications of satisfiability Testing*, LNCS 2919, 2003.
5. Le Berre D., Simon L., SAT2003 contest results. http://www.lri.fr/simon/contest03/ results/
6. Le Berre D., Simon L., "55 Solvers in Vancouver: The SAT 2004 competition". To appear in *SAT 2004* Special Volume.
7. Moskewicz M. *et al*, "Chaff: Engineering an Efficient SAT Solver". *38th Design Automation Conference*. ACM/IEEE, 2001.
8. Ryan L., The siege satisfiability solver. http://www.cs.sfu.ca/ loryan/personal/

9. Shacham O., Zarpas E., "Tuning the VSIDS Decision Heuristic for Bounded Model Checking". *Proc. of the 4th International Workshop on Microprocessor, Test and Verification*, IEEE Computer Society, 2003.
10. Van Gelder A.,Tsuji Y. K., "Satisfiability Testing with More Reasoning and Less Guessing". *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge.* American Mathematical Society, 1996.
11. Zarpas, E, "Simple yet efficient improvements of SAT based Bounded Model Checking". *Proc. of Formal Methods in CAD: 5th Int'l Conf.*, LNCS, 3312, 2004
12. CNF Benchmarks from IBM Formal Verification Benchmarks Library. www.haifa.il.ibm.com/projects/verification/RB_Homepage/bmcbenchmarks.html
13. IBM CNF Benchmark Illustration. www.haifa.il.ibm.com/projects/verification/RB_Homepage/papers/comparaison2.zip