Journal on Satisfiability, Boolean Modeling, and Computation ? (2019) ??-??

SAT Competition 2018

Marijn J. H. Heule[*] marijn@heule.nl
Department of Computer Science
The University of Texas at Austin
United States

Matti Järvisalo[†] matti.jarvisalo@helsinki.fi
HIIT, Department of Computer Science
University of Helsinki
Finland

Martin Suda[‡] martin.suda@cvut.cz
Czech Institute of Informatics, Robotics, and Cybernetics
Czech Technical University in Prague
Czech Republic

Abstract

The SAT Competition series, which started in 2002, is arguably one of the central driving forces of SAT solver development and its benchmark suites have been used in evaluations of hundreds of research papers. This article provides an overview of the 2018 edition of the SAT Competitions, including the competition tracks and rules, benchmark submission and selection, and the results of the competition focusing on the best-performing solvers.

Keywords: *SAT, Boolean satisfiability, empirical evaluation, solvers, benchmarks*

Submitted November 2018; revised April 2019; published ?

1. Introduction

The effectiveness of applying Boolean satisfiability (SAT) solvers, i.e., implementations of decision procedures for the propositional satisfiability problem, in solving real-world instances of complex search and optimization problems at large has within the last two decades progressed to a point where SAT can be called one of the success stories of modern computer science. Not restricted to solving "mere" NP-complete decision problems, SAT solvers are today routinely used iteratively as "real-world NP oracles", or core search procedures, in a variety of different complex algorithmic approaches to search and optimization problems with beyond-NP complexity. Seeking further runtime improvements over the current state-of-the-art SAT solver implementations is strongly motivated by the overall impact even minor improvements may have for a range of SAT solver application scenarios.

∗ Supported by the National Science Foundation (NSF) under grant CCF-1813993
† Financially supported by Academy of Finland (grants 276412 and 312662)
‡ Supported by the ERC Consolidator grant AI4REASON 649043, ERC Starting grant SYMCAR 639270, and the Austrian research project FWF S11409-N23.

The SAT Competition series [9], which started in 2002, is arguably one of the central driving forces of SAT solver development. The main goals of the competition series are to support and provide further incentives for the quest for further runtime improvements in SAT solvers, and to provide a yearly snapshot of the performance of the various current SAT solver implementations on heterogenous sets of benchmarks. A further important contribution of the SAT Competitions is the compilation of the benchmark suites made available to the research community. Indeed, the benchmark suites have developed into de facto instance collections for the scientific evaluation of general SAT solving techniques in research papers.

The year 2018 marks the 12th edition of the SAT Competition series, continuing close to two decades of tradition in SAT competitions and related competitive events for SAT solvers. A brief overview of the history of these events is provided in Table 1[1]. The organizers of the competition are yearly selected/invited each year externally by either the SAT conference organizers or the SAT Association. SAT Competition 2018 was organized as part of the 2018 FLoC Olympic Games in conjunction with the 21st International Conference on Theory and Applications of Satisfiability Testing (SAT 2018), which took place in Oxford, UK, as part of the 2018 Federated Logic Conference (FLoC).

This article provides an overview of SAT Competition 2018. In particular, we give details on how the competition was organized, including an overview of its rules for participation and the competition tracks (among other details); and describe the benchmark selection procedures used for establishing the 2018 benchmark suites. Furthermore, we give an overview of the results of the competition, focusing on the best-performing solvers in each competition track.

Beyond what is described in this article, the SAT Competition 2018 website

http://sat2018.forsyte.tuwien.ac.at/

provides further details on the competition, access to the 2018 benchmark suites, as well as full solver logs for the participating solvers for further analysis. Moreover, the proceedings of SAT Competition 2018 [8] include 1–2 page solver descriptions, written by the solver developers, on each of the participating solvers, as well as benchmarks descriptions by authors of new benchmarks for 2018.

2. Overview of SAT Competition 2018

In this section, we describe the 2018 SAT Competition in terms of the competition tracks, requirements for participation, ranking criteria, input/output and proof formats, as well as the computing infrastructure used for executing the competition.

2.1 Competition Tracks

SAT Competition 2018 was composed of four competition tracks, a Main Track and special tracks focusing on Random SAT and parallel solving, as well a "No-Limits" Track with very few requirements for participation.

1. For discussion on the relation of the SAT Competition series with other related solver competitions, we refer the interested reader to [2].

Table 1. History of SAT Competitions and related SAT solver evaluations

year	competition	ref	Organizers
2002	SAT Competition	[14]	Edward A. Hirsch, Daniel Le Berre, Laurent Simon
2003	SAT Competition	[4]	Daniel Le Berre, Laurent Simon
2004	SAT Competition	[5]	Daniel Le Berre, Laurent Simon
2005	SAT Competition		Daniel Le Berre, Laurent Simon
2006	SAT Race		Carsten Sinz
2007	SAT Competition		Daniel Le Berre, Olivier Roussel, Laurent Simon
2008	SAT Race		Carsten Sinz
2009	SAT Competition		Daniel Le Berre, Olivier Roussel, Laurent Simon
2010	SAT Race		Carsten Sinz
2011	SAT Competition		Matti Järvisalo, Daniel Le Berre, Olivier Roussel
2012	SAT Challenge	[2]	Adrian Balint, Anton Belov, Matti Järvisalo, Carsten Sinz
2013	SAT Competition		Adrian Balint, Anton Belov, Marijn Heule, Matti Järvisalo
2014	SAT Competition		Anton Belov, Daniel Diepold, Marijn Heule, Matti Järvisalo
2015	SAT Race		Tomas Balyo, Carsten Sinz, Markus Iser, Armin Biere
2016	SAT Competition	[3]	Marijn Heule, Matti Järvisalo, Tomas Balyo
2017	SAT Competition		Marijn Heule, Matti Järvisalo, Tomas Balyo
2018	SAT Competition		Marijn Heule, Matti Järvisalo, Martin Suda

Main Track was the track for sequential SAT solvers as the main focus of the competition. In terms of benchmarks, the focus was on solving structured, non-random benchmark instances. Solvers participating in the Main Track were required to provide certificates in both satisfiable and unsatisfiable cases. Participation in a sub-track of the Main Track for "Glucose hacks" was encouraged. For the purposes of setting a concrete (although ad-hoc) limit on how much changes would be required to the base solver and still be considered a "hack", a solver was considered a "Glucose hack" if the edit distance between its sources and the sources of the Glucose 3.0 SAT solver was smaller than 1000 non-space characters. In the past, several advances required only a dozen or so new or modified lines. Such advances fit easily in the 1000 character limit.

Random SAT Track was the track for SAT solvers aimed at efficiently providing satisfying truth assignments to randomly generated satisfiable instances. Since 2013 there has been no random UNSAT track due to a lack of interest (as witnessed by lack of participation).

Parallel Track was the track of parallel SAT solvers designed to make use of multiple CPUs or CPU cores.

No-Limits Track was a special track where (in contrast to the other tracks) solver source code and solution certificates were not required, and the solvers were judged solely based on their "yes"/"no" answers. In particular, this track was designed for, e.g., portfolio solvers—combining two or more (core) SAT solvers developed by different groups of authors—as well as closed-source solvers (including industrial participation).

2.2 Requirements for Participation

The following strict requirements were imposed on participation in the competition.

The source code of submitted SAT solvers had to be made available (licensed for research purposes) except for the solvers participating only in the No-Limits Track. The open source policy was strictly enforced; any submission that contained binary code was disqualified. A 1–2 page system description was required for each solver submission, including a list of all authors of the system and their present institutional affiliations, and providing details of any non-standard algorithmic techniques (e.g., heuristics, simplification/learning techniques, etc.) and data structures, as well as references to relevant literature (be they by the authors themselves or by others). The authors of a solver in the solver description had to match the authors listed in the submission system.

SAT solvers had to conform to DIMACS input/output requirements (see Section 2.4). Solvers were required to output a satisfying truth assignments in case of a satisfiable instance in all tracks expect for "no-limits". Additionally, unsatisfiability certificates (proofs) were required for the Main Track in the DRAT (Delete Resolution Asymmetric Tautologies) format (see Section 2.5).

Each Main Track participant (team) was required to submit 20 new benchmark instances (not seen in previous competitions). At least 10 of those benchmarks were expected to be "interesting": not too easy (solvable by MiniSAT 2.2 in a minute) or too hard (unsolvable by the participant's own solver within one hour on a computer similar to the nodes of the StarExec cluster (see Section 2.6) on which the track was run). We used MiniSAT 2.2 for checking the "not too easy" requirement as this solver is generally perceived as a representative solver of state-of-the-art a decade ago.

Each participant was restricted to be an author of at most four different sequential solvers, two different parallel solvers, and one Glucose Hack Sub-Track solvers. Two solvers were considered different as soon as their sources differed or the compilation options were different. The organizers of SAT Competition 2018 were not allowed to compete in any of the tracks.

Apart from the No-limits Track, participants were not allowed to submit portfolio SAT solvers. For SAT Competition 2018, a combination of two or more (core) SAT solvers developed by different groups of authors would be considered a portfolio solver. The reason for this rule is twofold. First, the organizers want to prevent that participants implement only an algorithm selection tool that calls existing solvers by other authors. Although research in algorithm selection is useful, it is not the focus of this competition. Second, the organizers would like to encourage the SAT community to invest more effort into developing new solver code bases and thus to counter the current trend in which the vast majority of solvers is built on top of MiniSAT.

2.3 Rankings, Awards, and Disqualification Criteria

Solvers were ranked using a PAR-2 score based on a 5000-second timeout.[2.] For the Main Track, in case a proof of unsatisfiability could not be validated within 20 000 seconds (see Section 2.5), the benchmark was considered not solved.

2. A PAR-2 score of a solver is defined as the sum of all runtimes for solved instances plus 2 times timeout for each unsolved instance. The lowest score wins.

Table 2. Overview of the tracks, benchmarks, and solvers

Track	Benchmarks	Solvers	Limits	Cluster
Main	400 main	41	5000 s, 1 core, 24 GB	StarExec
(sequential)	app + crafted		20 000 s DRAT	
Parallel	400 main	21	5000 s / 64 GB	TACC
			24 cores / 48 threads	
Random SAT	(planted) k-SAT	10	5000 s / 24 GB	StarExec
No-limits	400 main	34	5000 s / 24 GB	StarExec

SAT formula

```
p cnf 4 7
 1  2 -3 0
-1 -2  3 0
 2  3 -4 0
-2 -3  4 0
-1 -3 -4 0
 1  3  4 0
-1  2  4 0
```

SAT output

```
v -1 2 4 0
s SATISFIABLE
```

UNSAT formula

```
p cnf 4 8
 1  2 -3 0
-1 -2  3 0
 2  3 -4 0
-2 -3  4 0
-1 -3 -4 0
 1  3  4 0
-1  2  4 0
 1 -2 -4 0
```

DRAT proof

```
  -1 0
d -1 2 4 0
   2 0
   0
```

Figure 1. Small example formulas in the DIMACS CNF format together with an example output for a satisfiable formula and a proof for an unsatisfiable formula

In total 3×8 prizes were distributed (1st prize, 2nd prize, and a 3rd prize): The Glucose Hack Track (SAT+UNSAT), Main Track (SAT, UNSAT, SAT+UNSAT), Parallel Track (SAT, UNSAT, SAT+UNSAT), and Random (SAT) Track. We did not award any solvers in the No-Limits Track as the top solvers in that track performed worse compared to the Main Track with more restrictive rules for participation.

Disqualification A solver was disqualified if the solver produced a wrong answer, i.e., if the solver reported UNSAT on an instance that was proven to be SAT by some other solver, or it reported SAT and provided a wrong certificate. A solver disqualified from the competition was not eligible to win any awards. Disqualified solvers were marked as such on the competition results page. A solver could be withdrawn from SAT Competition 2018 only before the deadline for the submission of the final versions. After this deadline no further changes or withdrawals of the solvers were allowed.

2.4 Input and Output Format

Ever since the first SAT Competition in 2002, the input format has been the DIMACS CNF (Conjunctive Normal Form) format, the de facto standard input format for SAT solvers today. Formulas in DIMACS start with a problem description line `p cnf` followed by the number of variables and the number of clauses. Each subsequent line consists of either a comment (line starts with `c`) or a clause. Clauses are lists of integers with positive integers denoting positive literals and negative integers denoting negative literals. Clause lines terminate with `0`. Figure 1 shows two example formulas.

The output format is as follows. The solver prints either a line `s SATISFIABLE` or a line `s UNSATISFIABLE` depending on the satisfiability of the formula. In case the formula is satisfiable, it should also emit a satisfying assignment on one or more lines starting with `v`, with the literals satisfied by the truth assignment listed on the lines, and the last of these lines ending with 0. For example, both the single line `v -1 2 -4 3 0` and the two consecutive lines `v -1 -4` and `v 2 3 0` represent the same truth assignment over four variables indexed from 1 to 4, assigning the variables indexed with 2 and 3 to true and the variables indexed with 1 and 4 to false. For unsatisfiable formulas, the solvers needs to provide a proof of unsatisfiability, as described next.

2.5 Proof Checking

Proof logging and validation became mandatory for the UNSAT tracks in 2013 and for the Main track(s) of the SAT Competitions since 2014. The 2013 SAT Competition also demonstrated the importance of proof checking: one of the stronger solvers (in terms of the number of instances solved) and the winner of the Hard-combinatorial (SAT+UNSAT) track, `BreakIDGlucose`, turned out to be buggy after the competition. The solver did not produce a wrong answer during the competition, but a parsing bug caused it to immediately claim UNSAT on eight hard instances. These instances could no longer be solved by `BreakIDGlucose` after a bug fix.

All participants in the Main Track are required to emit proofs for unsatisfiable formulas. Proof logging for most solvers is easy and comes down to simply writing to disc the clauses that are learned and deleted. Proofs of unsatisfiability were to be emitted in the DRAT format [16]. The DRAT format is syntactically the same as DIMACS, but extends it by allowing clause deletion (lines with the prefix `d`). Figure 1 (right) shows an example. Clause deletion in proofs of unsatisfiability is mostly to speed up proof validation.

The tool-chain used for proof validation was as follows. A SAT solver produces a proof of unsatisfiability in the DRAT format. Afterwards the tool `DRAT-trim` [16] was used for checking and optimizing this proof, deriving a so-called LRAT proof file. Finally, a formally-verified tool, called `ACL2check` [6], was used for validating the LRAT proof as a correct proof of unsatisfiability of the given formula. An unsatisfiable formula was considered solved if `ACL2check` could validate the corresponding LRAT proof. Otherwise the formula was considered unsolved by the solver in question.

2.6 Computing Environment

All the tracks except for the Parallel Track were run on the StarExec cluster [15], whose nodes are equipped with Intel Xeon 2.4 GHz processors and 128 GB of memory. The time limit enforced on each solver for solving an instance was 5000 s (across all tracks). The solvers were allowed to use up to 24 GB of RAM as in the previous year.

The Parallel Track was run separately on computers equipped with Dual Socket Xeon E5-2690 v3 (Haswell), with 12 cores per socket (24 cores/node), 2.6 GHz processors and 64 GB DDR4-2133 (8 x 8 GB dual rank x8 DIMMS) main memory, and with hyperthreading enabled, thus with 48 threads (logical CPUs) per node. Solvers were provided access to a

Table 3. Overview of the benchmarks submitted and used for the 2018 competition

contributor	application	used	track
Adrian Balint	Random k-SAT q-Planted Solutions	30	Random
Tomas Balyo	Hard Random Satisfiable Benchmarks	168	Random
Armin Biere	Divider and Unique Inverse Benchmarks	20	Main
Nicolas Breton (2017)	Formal Verification Benchmarks from Systerel	20	Main
Jingchao Chen	Relativized Pigeonhole Principle Formulas	20	Main
Shuwei Chen	Hard 3-SAT benchmarks with cls/var-ratio 3.5	20	Main
Md Solimul Chowdhury	GrandTour Puzzle	19	Main
Jo Devriendt	k-Colorability Benchmarks	15	Main
Thorsten Ehlers	Tree Decompositions	20	Main
Jannis Harder	Reversing Elementary Cellular Automata	11	Main
Marijn Heule	Chromatic Number of the Plane Graph Coloring	20	Main
Marijn Heule	Uniform Random 3-SAT, 5-SAT, and 7-SAT	60	Random
Jonathan Heusser	Bitcoin	17	Main
Andrew Johnson	Ringing Bells	20	Main
Rodrigue Konan	Timetabling Benchmarks	12	Main
Jia Hui Liang	Verifying Simple Floating-Point Programs	15	Main
Norbert Manthey	Software Bounded Model Checking	19	Main
Valentin Mayer-Eichberger	Social Golfer Problem	14	Main
Ofer Strichman	Course Scheduling Benchmarks	20	Main
Alexey Porkhunov	Mutilated Chessboards and Cutting Cake	9	Main
Alexander Scheel	Logical Cryptanalysis Benchmarks	20	Main
Mate Soos	Almost Perfect Non-Linear S-box Finder	20	Main
Fan Xiao	Polynomial Multiplication	19	Main
Aolong Zha	Factoring Benchmarks	10	Main
Neng-Fa Zhou	Mix of hard-combinatorial benchmarks	19	Main

32 GB /tmp RAM disk to accelerate IO operations. However, any space taken in /tmp would decrease the total amount of memory available on the node accordingly.

3. Benchmarks

One of the main goals of the competition is to assess which solvers perform best on a diverse suite of benchmark formulas. Such a suite should also be balanced and representative of the applications for which SAT solving is used in practice. Such objectives are not easy to realize, which makes the selection of the benchmarks always somewhat controversial. This year the organizers wanted to counter one of the main flaws from the past: solvers that are optimized for existing benchmarks. We dealt with the issue by selecting only new instances for all benchmark suites. As a consequence, many new instances were required and the organizers decided to make benchmark submission mandatory for participants of the Main Track. The organizers intended to apply SAT-preserving transformations in case we used existing benchmarks. Since no existing benchmarks were used, no transformations were applied.

3.1 Benchmark Domains

Each participant of the Main Track was required to submit benchmark instances. Most of the benchmarks used in the competition were received based on that rule. Table 3 provides an overview of the used benchmarks and their submitters. The first column lists the contributor and, in case of multiple contributors, it lists the actual benchmark submitter. The second column briefly describes the contributed benchmark family. The third and fourth columns list the number of benchmarks used and the track in which the benchmarks were used, respectively. For more details on the individual benchmark families, we refer the interested reader to [8] which includes details as written by the authors of the benchmarks.

3.2 Benchmark Selection

The differences between the top-3 solvers in the Main Track of the competition has been small in recent years. This makes the benchmark selection procedure important. It also makes it somewhat controversial as a different selection can easily change the winners. Over the years several selection procedures have been used, each one with its own advantages and disadvantages. Ideally, one wants the benchmark suite to be representative of the applications for which SAT solvers are used and balanced to avoid creating a bias in favor of one or more solvers. Additionally, the competition should also be interesting and most benchmarks should not be too easy or too hard, otherwise either all solvers succeed on them or none do.

One of the perceived problems in recent years has been that solvers optimize their heuristics to perform well on existing benchmarks, in particular on satisfiable formulas. One can tweak the heuristics in such a way that the solver is "lucky" to quickly solve many existing instances. Various approaches have been proposed to counter this, including shuffling the instances, but most of these approaches have their disadvantages. For example, shuffling breaks the order of clauses and this order can be useful for solving, as it can provide some structure.

For 2018, we decided to construct a single benchmark suite consisting of 400 benchmarks that was used for the Main Track, the Parallel Track, and the NoLimit Track. Furthermore, in contrast to the previous editions of the competition series, we decided to select only instances not seen in previous SAT competitions. Additionally, at most 20 benchmarks from each contributor were used (apart from the benchmarks generated by the organizers). The latter restriction was chosen to limit the influence of benchmark contributors—who in most cases participated in the competition.

Both restrictions, only new benchmarks and at most 20 per contributor, left relatively little room for benchmark selection. We also removed the easier submitted benchmarks, i.e. those that were solvable by MiniSAT 2.2 in 10 minutes.[3] In case we had more than 20 benchmarks by a single contributor, we classified the benchmarks in three categories: medium, hard, and very hard. A benchmark is in the medium category if the average runtime of strong sequential solvers (in terms of the number of instances solved) from prior competitions is less than an hour; a benchmark is in the hard category if the average runtime of strong parallel solvers from prior competitions is less than an hour (wallclock

3. This decision is not related to the "not too easy" requirement of benchmark submissions.

time); and very hard otherwise. We randomly selected 20 instances with a 1/3 probability to be medium, hard, or very hard for each pick.

4. Competition Results

We turn to the results of SAT Competition 2018, focusing especially on the best-performing solvers in each of the competition tracks.

4.1 Main Track

The Main Track is generally seen as the most prestigious track of the competition. This track also attracts most participants: 41 solvers in 2018. Although the number of solvers is high, the differences between most of them is small. The code base of a great majority of the participating solvers originates from MiniSAT [7] which won the Industrial Track in the 2005 SAT Competition. The MiniSAT solver was improved by better prediction of useful clauses, resulting in the solver Glucose [1] which won the Application Track in the 2011 SAT Competition. Glucose in turn was improved by improving the decision heuristics, resulting in the MapleSAT solver [11] which won the Main Track in the 2016. Finally, MapleSAT was extended with inprocessing techniques [10] resulting in the solver Maple_LCM [12], the winner of the Main Track in 2017. Unsurprising, many solvers in this year's competition were based on Maple_LCM.

Overviews of the results of the Main Track, focusing on the best-performing solvers, are shown in Table 4 and Figure 2, with performance on SAT and UNSAT instances, respectively, shown in Figure 3 and Figure 4, respectively. The winner of the Main Track was the solver MapleLCMDistChronoBT by Vadim Ryvchin and Alexander Nadel, who added a form of chronological backtracking [13] to Maple_LCM. Similar to the other improvements in recent years, the use of occasional chronological backtracking appears especially useful on satisfiable formulas. The overview of the Main Track, Table 4, shows that all the solvers based on Maple solve a very similar number of unsatisfiable benchmarks, but the difference is substantial for satisfiable benchmarks. The only solver that clearly solves more unsatisfiable benchmarks is CaDiCaL, which is based on a different code base originating from the submitter.

Almost all proofs of unsatisfiability were validated within the 20 000-second time limit and the 24-Gb memory limit. Most proofs that were not validated ran out of memory (96 times), while only a few ran out of time (8). As a consequence, the organizers plan to increase the memory limit for future competitions. The number of runs for which the proof validation timed out are shown in fourth column of Table 4. Proof validation for the top-10 solvers timed out most frequently for the winner, meaning that MapleLCMDistChronoBT would have been even stronger if all its proofs could have been validated within the validation time limit. In the future, we want to eliminate the potential impact of proof validation on the results by making the validation tools faster and/or by increasing the validation time limit (subject to available computing resources). Although proof checking has some influence on the scoring, it is a small price to pay to ensure that no solver will win due to a bug (see Section 2.5).

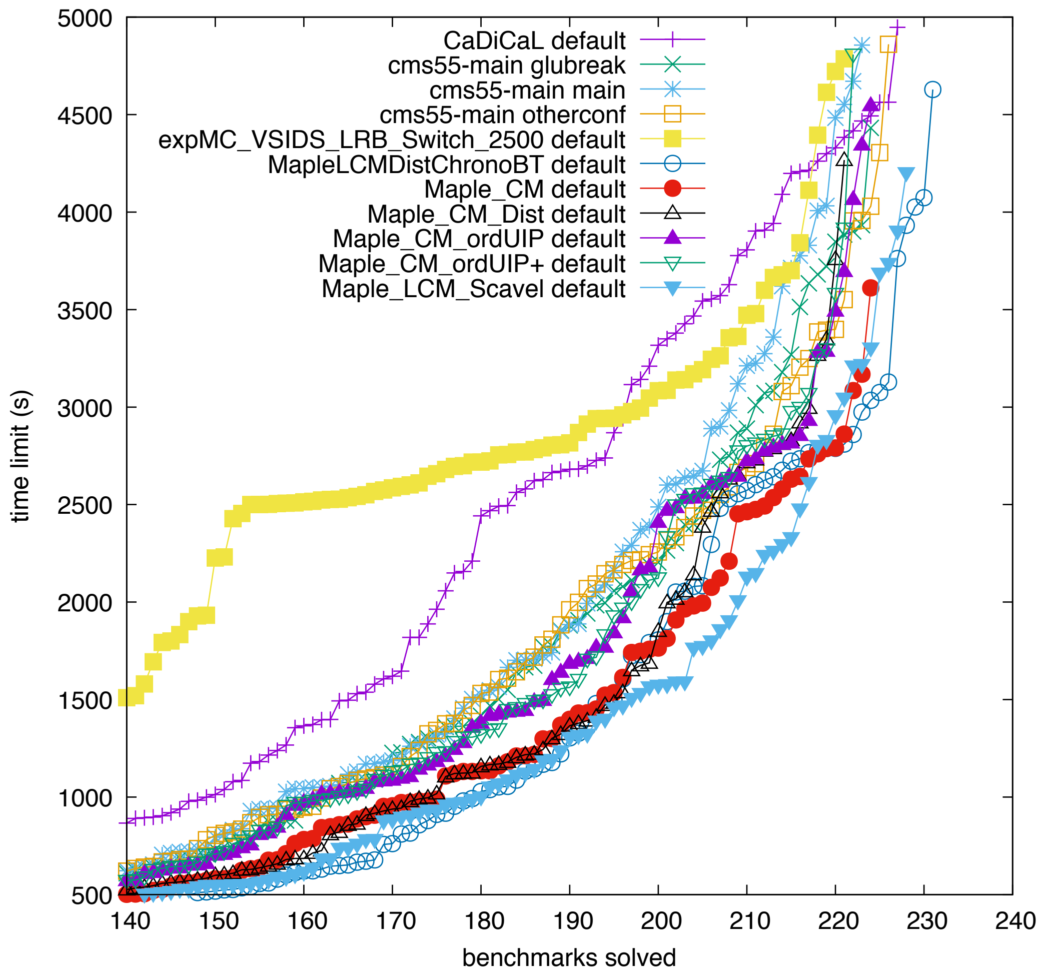

Figure 2. Cactus plot for the Main Track; solvers with 220 or more solved benchmarks.

Table 4. The best 15 solvers from the Main Track ordered by the PAR-2 score. Additionally, the number of solved (satisfiable and unsatisfiable) benchmarks is shown and the number of benchmarks claimed to be unsatisfiable for which this claim could not be checked due to insufficient resources.

PAR-2 score	Solved	SAT/UNS	unchecked	Solver Name and Configuration
1 857 321	231	135 / 96	6	MapleLCMDistChronoBT, default
1 872 489	228	134 / 94	4	Maple_LCM_Scavel, default
1 908 304	224	125 / 99	1	Maple_CM, default
1 931 478	226	133 / 93	4	cms55-main, otherconf
1 934 450	224	125 / 99	1	Maple_CM_ordUIP, default
1 936 290	221	123 / 98	1	Maple_CM_Dist, default
1 946 689	224	130 / 94	4	cms55-main, glubreak
1 947 025	222	123 / 99	2	Maple_CM_ordUIP+, default
1 961 567	217	126 / 91	7	Maple_LCM_Scavel_200, default
1 964 101	223	128 / 95	4	cms55-main, main
1 978 764	218	120 / 98	2	Maple_LCM+BCrestart_M1, default
1 982 469	218	120 / 98	1	Maple_LCM+BCrestart, default
1 986 048	227	120 / 107	2	CaDiCaL, default
1 987 577	216	125 / 91	3	expMC_LRB_VSIDS_Switch_2500, default
1 996 387	215	116 / 99	2	Maple_LCM_M1, default

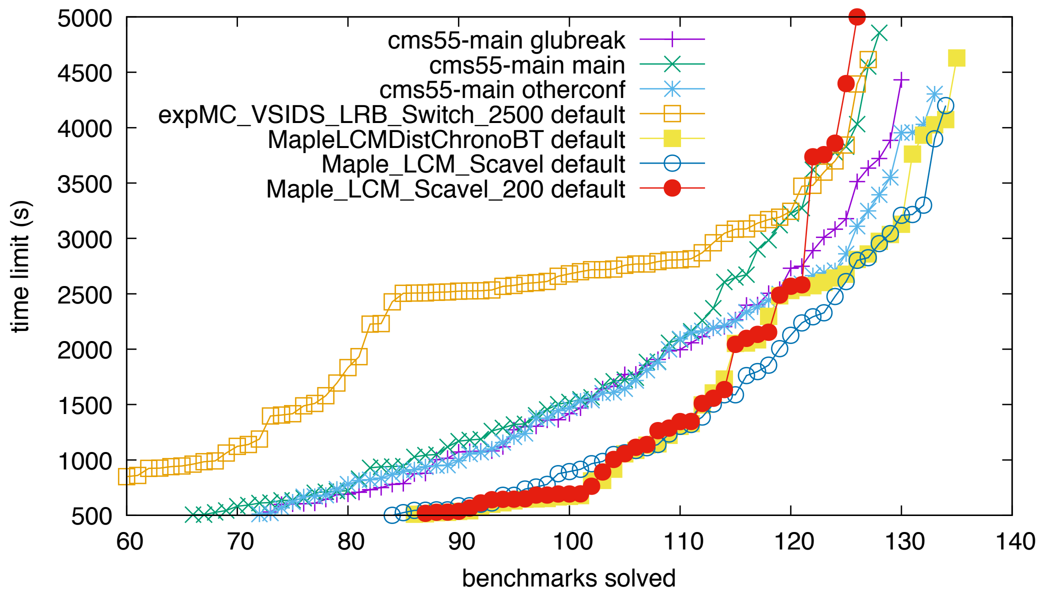

Figure 3. Cactus plot for the Main Track; SAT only; solvers with 126 or more solved benchmarks.

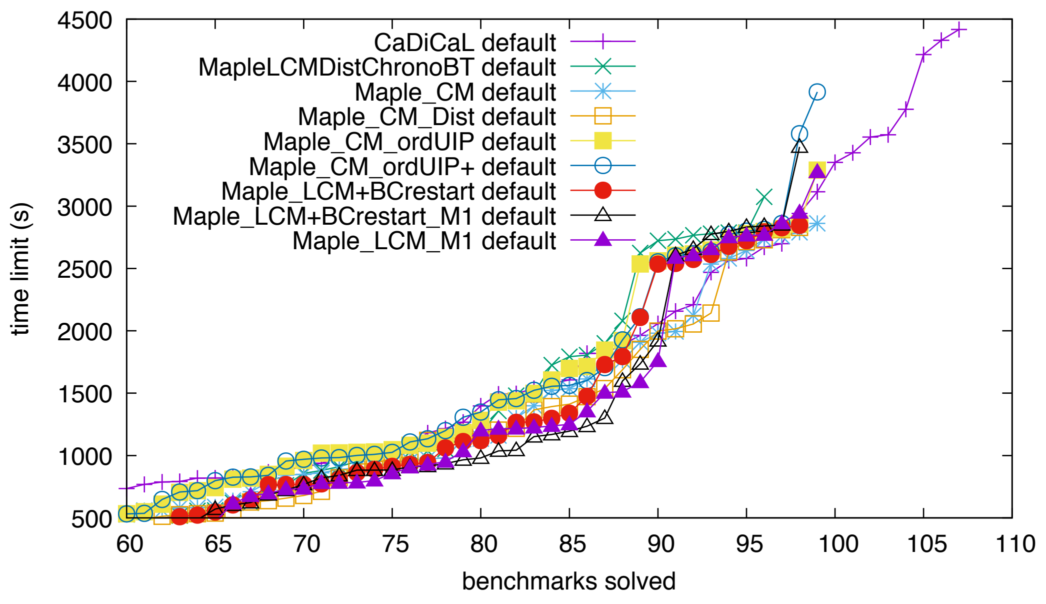

Figure 4. Cactus plot for the Main Track; UNSAT only; solvers with 96 or more solved benchmarks.

Virtual best solver and its contributors In total, from the 400 selected there were 302 benchmarks solved by at least one solver. This is sometimes referred to as the performance of the "virtual best solver" (VBS), referring to the idea of a perfect oracle that could for each benchmark up front select one of the solvers that would succeed on it (givene that at least one of the solvers succeeds). It is interesting to compare to what degree individual solvers contribute to the performance of VBS. One possible way of doing this is to take the set of solved benchmarks (in our case of size 302) and for each of them distribute a "reward" of 1 point evenly among those solvers who solved that benchmark. Thus if a particular solver is unique to solve a certain benchmark it receives the full 1 point for it. However, if e.g. three solvers solve a certain benchmark, each of the three solvers receives 1/3 points.

Table 5 lists the first five solvers of the Main Track according to this VBS Reward. We can see that from this perspective, CaDiCaL dominates other solvers by a large margin. This essentially means that CaDiCaL was the most successful solver to solve many problems that other solvers could not solve and is again likely related to CaDiCaL relying on an independent code base.

Glucose Hack Sub-Track Many improvements in SAT solvers are based on relatively small changes of existing solvers. This resulted in the Glucose Hack Sub-Track, in which participants were allowed to change only up to 1000 non-space characters of Glucose version 3.0. The sub-track started as the MiniSAT Hack Sub-Track in the 2013 SAT Competition. Table 6 shows the overview of the results of this track, which was won by the solver GHack-COMSPS. Notice again that the participants solve almost the same number of unsatisfiable

Table 5. The first 5 solvers by their contribution to the performance of the virtual best solver.

VBS Reward	Solver Name and Configuration
19.52	CaDiCaL, default
11.87	MapleLCMDistChronoBT, default
8.95	Maple_LCM_Scavel, default
8.76	cms55-main, otherconf
8.50	Maple_CM_ordUIP+, default

Table 6. The results of the Glucose Hack Sub-Track (a sub-view of the Main Track) ordered by the PAR-2 score. Unchecked reports on benchmarks claimed to be unsatisfiable but either the provided proof was rejected by the checker (first number) or the checker did not confirm the result within the given resources (second number). glucose3.0 (proofs) participated *hors concours,* being submitted by the organizers as the baseline solver which the others in the category were trying to improve on.

PAR-2 score	Solved	SAT / UNS	(unchecked)	Solver Name and Configuration
2 205 133	196	105 / 91	0 / 3	GHackCOMSPS_drup, ghack_drup
2 246 060	194	110 / 84	0 / 2	inIDGlucose, default
2 262 847	192	106 / 86	0 / 6	glu_mix, default
2 379 070	181	96 / 85	0 / 3	glucose-3.0_PADC_10, default
2 425 018	176	86 / 90	0 / 1	Glucose_Hack_Kiel_fastBVE, default
2 444 829	174	85 / 89	0 / 0	glucose3.0, proofs
2 489 038	169	95 / 74	0 / 8	glucose-3.0_PADC_3, default
2 933 160	116	98 / 18	60 / 0	gluHack, default

benchmarks, while they differ quite a bit in the number of solved satisfiable benchmarks. In particular, GHackCOMSPS solved 20 more satisfiable and 2 more unsatisfiable benchmarks than the base solver, Glucose 3.0.

4.2 No-Limits Track

The No-Limits Track was introduced originally for SAT Competition 2016 to allow participation of solvers for which the source code is not public (e.g. solvers developed in industry) or which use techniques that are not easy to express in DRAT (e.g. pseudo-boolean reasoning). As the source code cannot be checked, solvers in this track were tested on new benchmarks as solver authors could potentially include a lookup table for existing benchmarks. Since only new benchmarks were used in all tracks this year, the same benchmark suite was used for the Main Track and the No-Limits Track. This also facilitates a better comparison of these tracks.

Overviews of the results of the No-Limits Track, focusing on the best-performing solvers, are shown in Table 7 and Figure 5, with performance on SAT and UNSAT instance, shown in Figure 6 and Figure 7, respectively. The most interesting result of the No-Limits Track is that the strongest solver in this track has a higher (and thus worse) PAR-2 score than the winner and runner-up of the Main Track. This was a consequence of the authors of MapleLCMDistChronoBT and Maple_LCM_Scavel not participating in this track. The

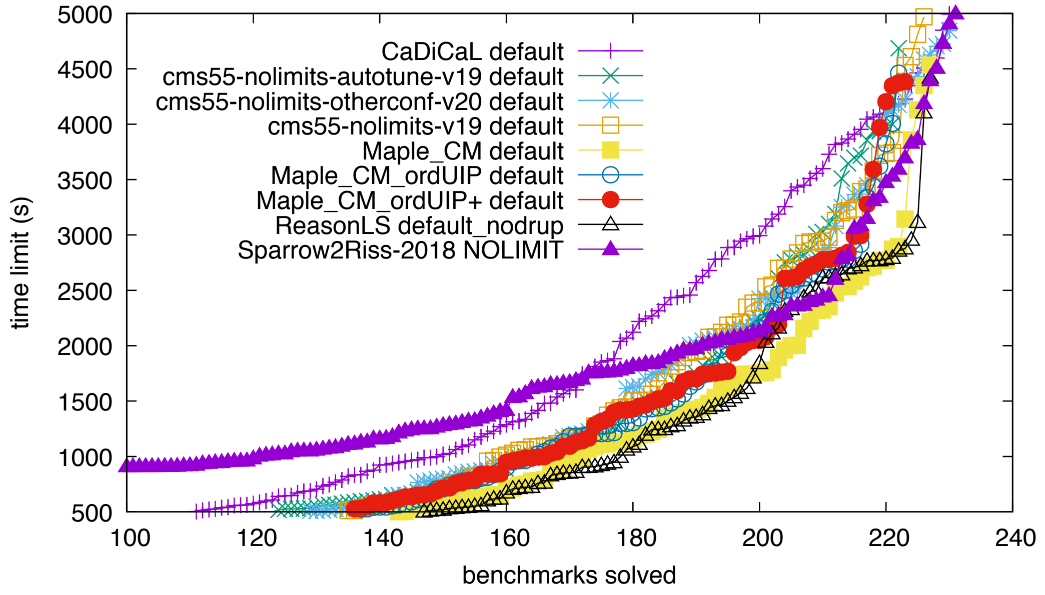

Figure 5. Cactus plot for the No-Limits Track; solvers with 222 or more solved benchmarks.

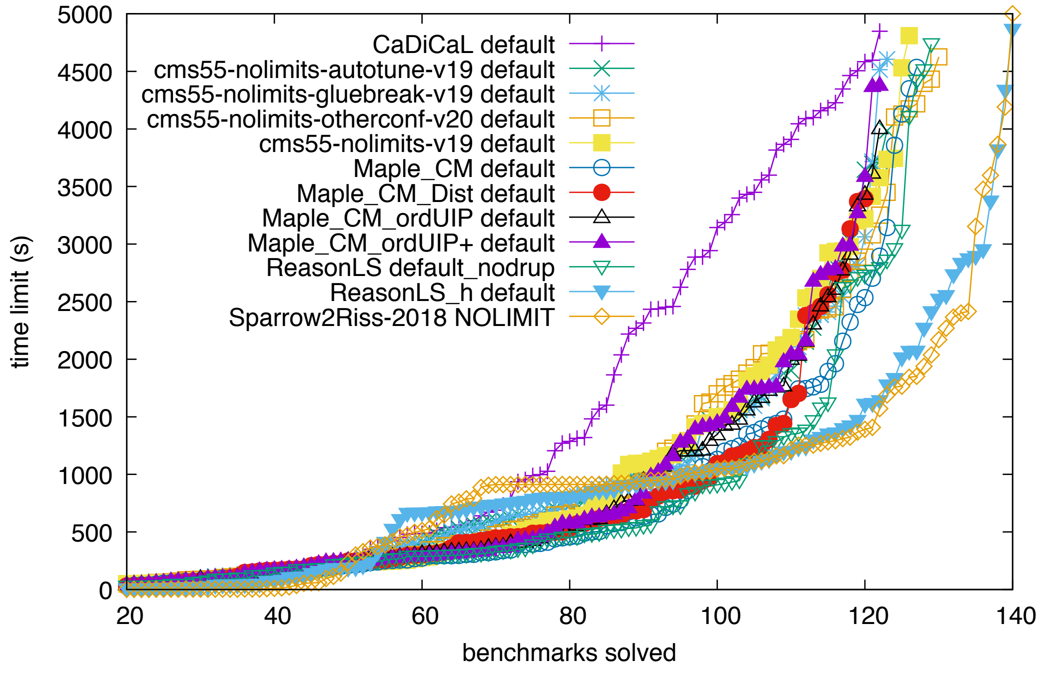

Figure 6. Cactus plot for the No-Limits Track; SAT only; solvers with 120 or more solved benchmarks.

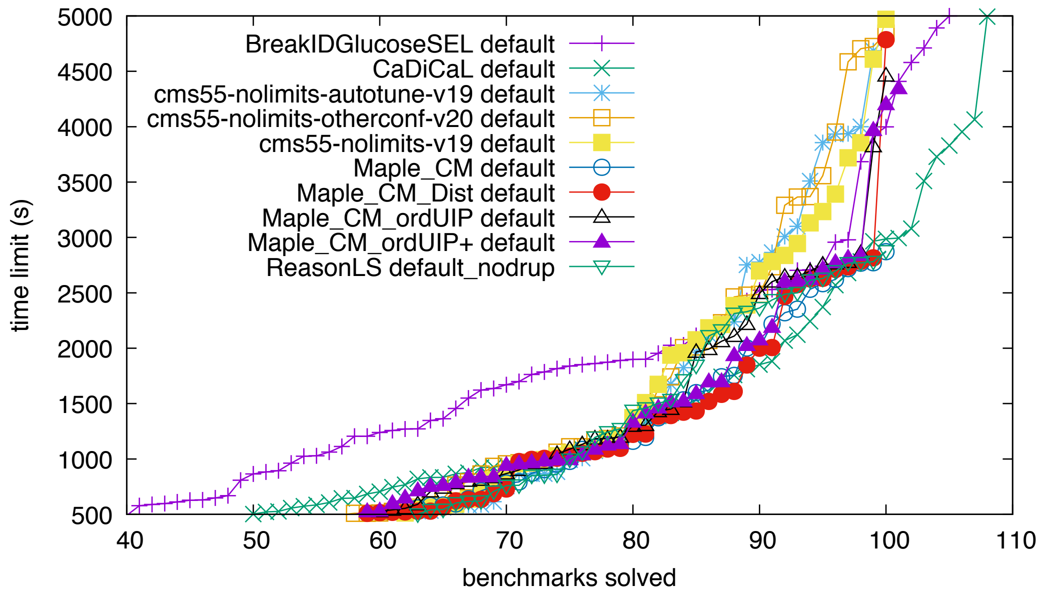

Figure 7. Cactus plot for the No-Limits Track; UNSAT only; solvers with 99 or more solved benchmarks.

Table 7. The best 15 solvers from the No-Limits Track ordered by the PAR-2 score.

PAR-2 score	Solved	SAT / UNS	Solver Name and Configuration
1 875 448	229	129 / 100	ReasonLS, default_ND
1 890 452	227	127 / 100	Maple_CM, default
1 915 985	230	130 / 100	cms55-nolimits-otherconf-v20, default
1 935 934	226	126 / 100	cms55-nolimits-v19, default
1 943 354	223	122 / 101	Maple_CM_ordUIP+, default
1 943 988	220	120 / 100	Maple_CM_Dist, default
1 946 595	222	122 / 100	Maple_CM_ordUIP, default
1 958 979	230	122 / 108	CaDiCaL, default
1 959 258	231	140 / 91	Sparrow2Riss-2018, NOLIMIT
1 960 117	222	123 / 99	cms55-nolimits-autotune-v19, default
1 967 312	221	123 / 98	cms55-nolimits-gluebreak-v19, default
1 986 412	216	123 / 93	MapleCOMSPS_LRB_VSIDS_2_ND, LRB_VSIDS_2_ND
2 016 132	214	118 / 96	COMiniSatPS_Pulsar_ND, ND
2 079 101	210	117 / 93	MapleCOMSPS_LRB_VSIDS_ND, LRB_VSIDS_ND
2 090 919	208	116 / 92	smallsat, default

Table 8. The results of the Random Track ordered by the PAR-2 score. (Recall that all benchmarks in the track are satisfiable.)

PAR-2 score	Solved	Solver Name and Configuration
687 420	188	Sparrow2Riss-2018, NOLIMIT
901 550	165	gluHack, default
902 011	165	glucose-3.0_PADC_10_ND, default
902 244	165	glucose-3.0_PADC_3_ND, default
904 236	165	expGlucoseSilent, default
946 811	163	CPSparrow, default
1 035 064	155	dimetheus, randomsat
1 198 881	138	probSAT, RANDOM
1 280 179	129	YalSAT, default
1 747 096	81	lawa, default

organizers therefore decided not to award any of the solvers in the No-Limits Track as winning this track only makes sense if a solver can outperform all solvers in the Main Track, which need to produce certificates for both satisfiability and unsatisfiability (and need to make the source code available). A comparison between solvers that participated in both the No-Limits Track and the Main Track shows that their PAR-2 scores are very similar. This indicates that the overhead of proof logging is very small.

4.3 Random SAT Track

The Random SAT Track was somewhat different this year compared to prior SAT Competitions, although this was unintended. The script for the small uniform random k-SAT formulas at the phase transition failed to generate the instances, which was discovered only after the competition results were finalized. As a consequence no such formulas were included in the benchmark set of this track. Thereby the benchmark set consists of large uniform random k-SAT formulas below the phase transition and, furthermore, non-uniform random k-SAT formulas contributed by Adrian Balint and Tomas Balyo.

An overview of the results of the Random SAT Track are shown in Table 8 and Figure 8. The results are surprising as CDCL solvers outperformed local search solvers in this track. It is well known that CDCL solvers cannot compete at all with local search solvers on small uniform random k-SAT formulas at the phase transition. Yet, apparently, CDCL solvers perform relatively stronger when non-uniformly yet still random k-SAT instances are included in the benchmark set.

Notice that solvers perform very differently on random SAT problems (see Figure 8) compared to structured SAT problems (see Figure 3). On random SAT problems hardly any formulas are solved after 1000 seconds (out of 5000 seconds), while on structured SAT problems many instances are solved between 1000 and 5000 seconds. This suggests that there could be quite some room for improvement on random SAT instances, for example by completely changing the search strategy every 1000 seconds.

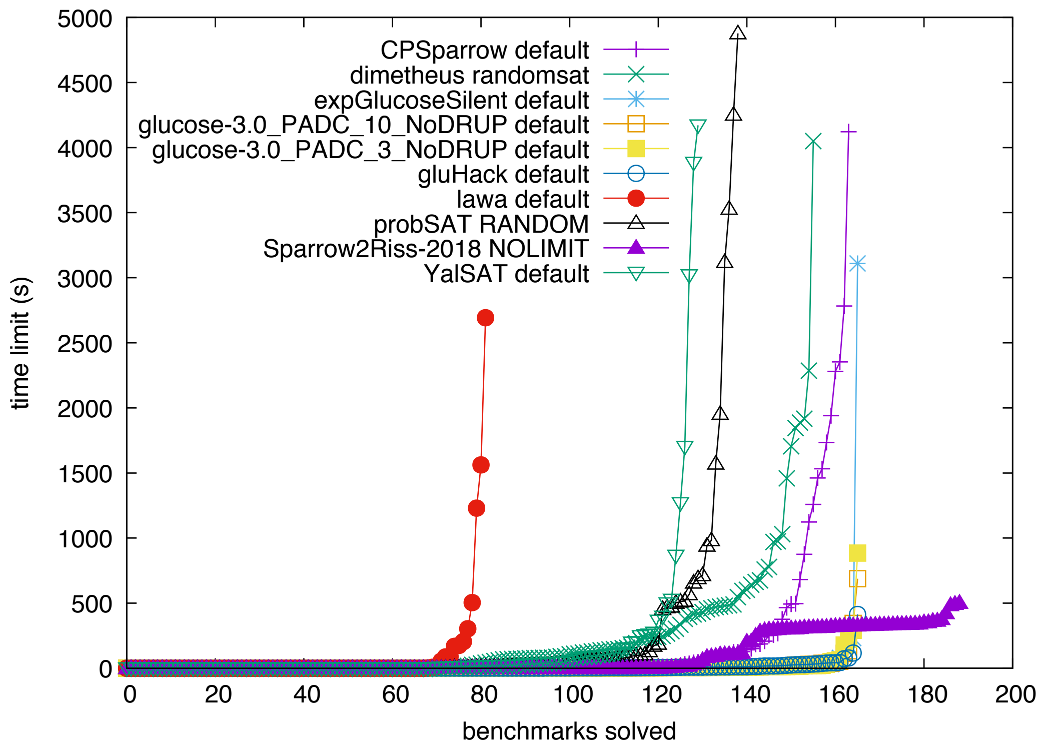

Figure 8. Cactus plot for the Random Track (only SAT benchmarks).

4.4 Parallel Track

The results of the Parallel Track are somewhat underwhelming; see Table 9 and Figure 9 for an overview. In particular, the hardware used in the Parallel Track is faster than for the other tracks (even when using only a single core) and, furthermore, parallel solvers were not required to produce proofs of unsatisfiability. So one would expect that the parallel solvers would exhibit substantially better performance than the sequential solvers in wall-clock time (on the same benchmark suite), especially if they can effectively make use of the 24 cores. However, only 4 solvers were clearly stronger than the sequential solvers. Furthermore, two of those parallel solvers turned out to be buggy (both TopoSAT versions reporting UNSAT on some satisfiable instances). It appears that only painless, the winner of the Parallel Track, and plingeling, the runner up, were able to significantly benefit (in terms of performance) from using the available multiple cores. The difference between these solvers and the others is also clear from the cactus plot (see Figure 9).

It seems that–at least based on the performance of the submitted parallel solvers on the benchmark suite of the 2018 competition—parallel SAT solving is not yet in a mature state. A central issue that needs to be resolved is memory management: several solvers ran out of memory on instances that they would have solved when running also on a single core. Various participants submitted two versions of their solver to the Parallel Track, for example, a 12 and a 24 core version of cms55-parallel and 24 and 48 threads version of syrup. The versions with fewer threads always outperformed the one with more threads. These results only represent the performance of parallel solvers on a single suite on a single cluster, but these issues can likely be observed in many other situations.

Some solvers in the Parallel Track were buggy—in contrast to the sequential solvers in the Main Track. This may be the result of requiring proof logging for sequential solvers, which forces participants to involve proof production and proof validation during solver development. The organizers have no plans to require proof logging of parallel solvers as this is more complicated and may slow the necessary improvements. However, supporting proofs of unsatisfiability could help making parallel solvers less buggy.

5. Conclusions

The 12th edition of the SAT Competition was organized as part of the FLoC 2018 Olympic Games in conjunction with the SAT 2018 conference. As typical, the 2018 SAT Competition attracted many participants, with over a hundred solvers competing in the four tracks of the competition. In this article, we outlined the main organization details of SAT Competition 2018, provided an overview of the new benchmarks contributed by the research community for use in the competition, gave overview statistics on the participants, and provided details on the competition results, focusing on the best performing solvers of 2018.

In terms of future developments in the SAT Competitions, we note that, apart from the Random Track, a great majority of the participating solvers are based on not only the same algorithmic paradigm, CDCL, but also built on top of the same source code base, originating from the MiniSAT solver. While one should not de-emphasize the importance of incremental improvements obtained by demonstrating that small new variations to existing solvers may be beneficial, seeking ways of providing further incentives for more radical new algorithmic ideas could be considered. This is especially true, for example, in the case

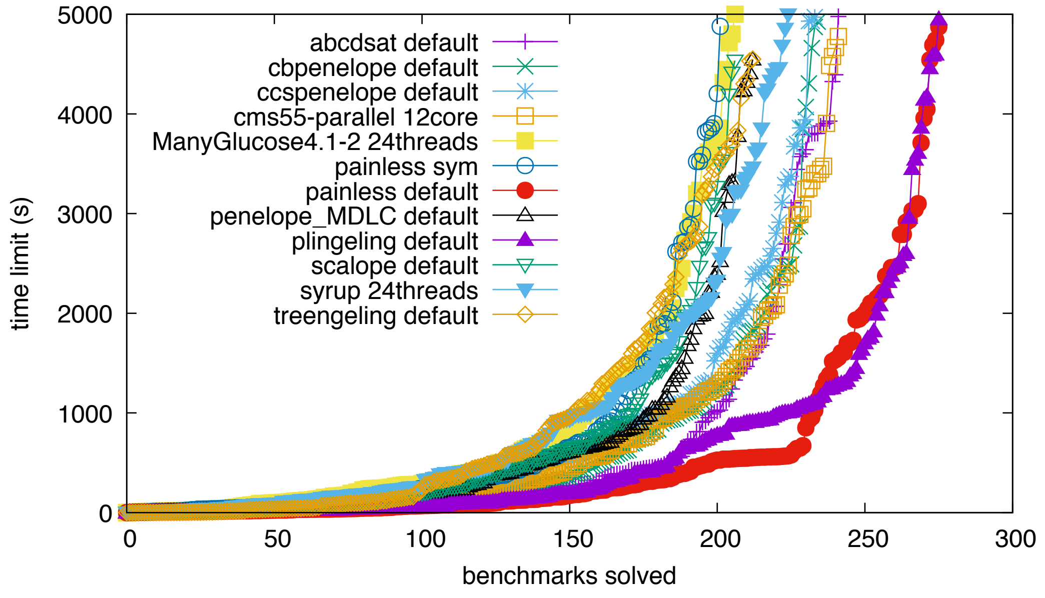

Figure 9. Cactus plot for the Parallel Track; solvers with 200 or more solved benchmarks.

Table 9. The best 15 solvers from the Parallel Track ordered by the PAR-2 score. TopoSAT (both plain and lazy) was disqualified for producing incorrect results.

PAR-2 score	Solved	SAT / UNS	Threads	Solver Name and Configuration
1 252 833	289	164 / 125	24	TopoSAT*, plain
1 304 789	284	159 / 125	24	TopoSAT*, lazy
1 397 524	275	153 / 122	24	painless, default
1 410 158	275	157 / 118	24	plingeling, default
1 736 011	241	148 / 93	24	abcdsat, default
1 755 049	241	153 / 88	12	cms55-parallel, 12core
1 797 550	238	140 / 98	24	cbpenelope, default
1 835 313	236	138 / 98	24	ccspenelope, default
1 962 633	224	132 / 92	24	syrup, 24threads
2 006 302	212	135 / 77	24	penelope_MDLC, default
2 058 429	212	127 / 85	24	treengeling, default
2 062 025	206	129 / 77	24	scalope, default
2 100 147	206	121 / 85	24	ManyGlucose4.1-2, 24threads
2 115 484	202	122 / 80	24	painless, sym
2 177 525	198	121 / 77	48	syrup, 48threads

of the Parallel Track, where surprisingly little progress has been made in terms of solver performance, especially when considering the fact that parallel computing infrastructures are less limiting for developing novel algorithmic ideas. On another note, in 2018 a very healthy number of new benchmark instances were contributed by the research community, thanks to which the 2018 main benchmark suit (used in the Main, Parallel, and No-Limits tracks) was decidedly constructed solely from the new benchmarks. One must note that, due to this, the 2018 benchmark set does not contain instances from several domains typically included in the competition benchmark sets. While using solely new benchmarks is a strong way to combat solver overfitting towards known benchmarks, the choice of using only new benchmarks, domains of which were decided by the benchmark contributors, can also be criticized due to not covering several typical SAT application domains.

Acknowledgments

We would like to thank everyone who contributed to SAT Competition 2018 by submitting solvers or benchmarks. We also thank StarExec at the University of Iowa and Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing computing infrastructure which allowed for smoothly running the competition.

References

[1] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern SAT solvers. In Craig Boutilier, editor, *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 399–404. Morgan Kaufmann Publishers Inc., 2009.

[2] Adrian Balint, Anton Belov, Matti Järvisalo, and Carsten Sinz. Overview and analysis of the SAT Challenge 2012 solver competition. *Artificial Intelligence*, **223**:120–155, 2015.

[3] Tomás Balyo, Marijn J. H. Heule, and Matti Järvisalo. SAT Competition 2016: Recent developments. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 5061–5063. AAAI Press, 2017.

[4] Daniel Le Berre and Laurent Simon. The essentials of the SAT 2003 Competition. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers*, **2919** of *Lecture Notes in Computer Science*, pages 452–467. Springer, 2004.

[5] Daniel Le Berre and Laurent Simon. Fifty-five solvers in Vancouver: The SAT 2004 Competition. In Holger H. Hoos and David G. Mitchell, editors, *Theory and Applications of Satisfiability Testing, 7th International Conference, SAT 2004, Vancouver, BC, Canada, May 10-13, 2004, Revised Selected Papers*, **3542** of *Lecture Notes in Computer Science*, pages 321–344. Springer, 2005.

[6] Luís Cruz-Filipe, Marijn J. H. Heule, Warren A. Hunt Jr., Matt Kaufmann, and Peter Schneider-Kamp. Efficient certified RAT verification. In Leonardo de Moura, editor, *Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings*, **10395** of *Lecture Notes in Computer Science*, pages 220–236. Springer, 2017.

[7] Niklas Eén and Niklas Sörensson. An extensible sat-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers*, **2919** of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2004.

[8] Marijn J.H. Heule, Matti Järvisalo, and Martin Suda, editors. *Proceedings of SAT Competition 2018: Solver and Benchmark Descriptions*, **B-2018-1** of *Department of Computer Science Series of Publications B*. University of Helsinki, 2018.

[9] Matti Järvisalo, Daniel Le Berre, Olivier Roussel, and Laurent Simon. The international SAT solver competitions. *AI Magazine*, **33**(1), 2012.

[10] Matti Järvisalo, Marijn Heule, and Armin Biere. Inprocessing rules. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, **7364** of *Lecture Notes in Computer Science*, pages 355–370. Springer, 2012.

[11] Jia Hui Liang, Vijay Ganesh, Pascal Poupart, and Krzysztof Czarnecki. Learning rate based branching heuristic for SAT solvers. In Nadia Creignou and Daniel Le Berre, editors, *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*, **9710** of *Lecture Notes in Computer Science*, pages 123–140. Springer, 2016.

[12] Mao Luo, Chu-Min Li, Fan Xiao, Felip Manyà, and Zhipeng Lü. An effective learnt clause minimization approach for CDCL SAT solvers. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 703–711, 2017.

[13] Alexander Nadel and Vadim Ryvchin. Chronological backtracking. In Olaf Beyersdorff and Christoph M. Wintersteiger, editors, *Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, **10929** of *Lecture Notes in Computer Science*, pages 111–121. Springer, 2018.

[14] Laurent Simon, Daniel Le Berre, and Edward A. Hirsch. The SAT2002 competition. *Annals of Mathematics and Artificial Intelligence*, **43**(1):307–342, 2005.

[15] Aaron Stump, Geoff Sutcliffe, and Cesare Tinelli. StarExec, a cross community logic solving service. https://www.starexec.org, 2012.

[16] Nathan Wetzler, Marijn Heule, and Warren A. Hunt Jr. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In Carsten Sinz and Uwe Egly, editors,

Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings, **8561** of *Lecture Notes in Computer Science*, pages 422–429. Springer, 2014.