JSAT

Preface to the Special Volume on the SAT 2005 Competitions and Evaluations

Daniel Le Berre leberre@cril.univ-artois.fr
CRIL-CNRS FRE 2499, Université d'Artois
Rue Jean Souvraz SP 18 – F
62307 Lens Cedex, France

Laurent Simon simon@lri.fr
LRI, Université Paris-Sud
Bâtiment 490, U.M.R. CNRS 8623
91405 Orsay Cedex, France

1. From a single event to a special volume: History of the SAT competitions

In 2005 there were numerous "competitive events" in the area of automated reasoning (in the broad sense): The Tenth CASC competition[23], the First Satisfiability Modulo Theory Competition[3], the First CSP competition, the Fourth SAT competition, the Third QBF evaluation and the First Pseudo Boolean evaluation, etc.

Starting such an event is suitable for promoting a common input format and to build a repository of benchmarks in that format: this was the motivation behind the SMT and CSP competitions and the PB evaluation. The First QBF evaluation was also organized two years ago in that spirit and the renewal since then allowed to impose the QDimacs input format and to increase both the number of solvers and benchmarks available each year.

More mature events, such as the CASC competitions, are a bit different: both the pool of benchmarks (TPTP) and the pool of solvers are quite stable and the competitions allow to track the progress of new versions of those solvers on a well studied set of benchmarks.

The SAT competitions are again different: the first SAT competition is older than CASC (1992, [6]), a common input format exists since the Second Dimacs challenge (13 years ago [12]), many SAT benchmarks are available (from a repository such as SATLIB [11] or from academic or private companies web sites) in that format. Because recent SAT benchmarks are really huge (several hundreds of MB each) and others cannot be freely distributed, it is difficult to think about a common repository similar to the TPTP where all the SAT benchmarks could be available. Furthermore, the number of available SAT solvers is growing each year and makes it difficult to maintain up-to-date initiatives such as SATEX [22]. As a consequence, the SAT competition should be seen as a yearly snapshot of a subset of current SAT solvers on a subset of the available SAT benchmarks.

The four consecutive SAT contests have brought the community a large amount of new solvers and benchmarks, and a place where both solver designers and users can meet. The main difference between the initial SAT 2002 competition and the current SAT 2005

Table 1. SAT Competition Overall Progress. The "†" mark means that three launchs were attempted for each randomized solvers, with the same timeout.

	2002	**2003**	**2004**	**2005**
#Solvers	28	34	55	43
#Submit.	19	22	27	22
Timeout (s) 1st	2,400†	900	600	1,200
Timeout (s) 2nd	21,600†	7,200	2,400	6,000/12,000
Total CPU (days)	750	522	516	1,230
Judges	–	J. Franco, H. van Maaren, T. Walsh	F. Bacchus, J. Marques-Silva, H. Kleine Buning	A. van Gelder, A. Biere, O. Kullmann

competition is that SAT solvers are currently being used in various areas, in academia or in the industry, because they can handle some benchmarks with millions of variables and clauses. In the SAT 2005 competition, this is reflected by the variety of benchmarks in the industrial category, and the participation of companies such as IBM and Intel.

1.1 What's new in the SAT 2005 competition

Over the years, new rules were introduced in order to ensure that the community may benefit from the event. We mainly focused on the "anti-black-box" rule in the 2004 contest, but we relaxed a bit that rule this year. It was decided to mimic CASC and to have two divisions in the SAT 2005 competition: the *competition* division and the *demonstration* division. The competition division is the place where solvers, available in source for the research community, can compete and from which winners are selected. The demonstration division is the place where solvers available in binary form, or other solvers whose presence could benefit the research community, are allowed. There was no winner in that second category. At most 3 variants of the same solver (same submitter) were allowed in the competition division. Additional submitted solvers were moved to the demonstration division (the submitters chose which solver ran in which division). This year the three judges were Armin Biere, Allen van Gelder and Oliver Kullmann.

For the first time, we had less solvers submissions than the previous year. We had 43 solvers (see table 1 for details over the years), and we selected 1657 benchmarks over the three categories, 390 random, 675 crafted and 592 industrial (296 original + 296 shuffled). The total amount of CPU time was 1230 CPU-Days on two different clusters of 18 computers each, 680 days of CPU-time for the first stage and 550 days of CPU-time for the second one.

Practically speaking, the hardware was composed of 16 Athlon 1800+ with 1GB RAM, provided by the LRI, Université de Paris-Sud, and with 8 Athlon 1800+ with 2 GB RAM, provided by the LINC Lab, Department of ECECS, University of Cincinnati. Computers were running GNU Linux (RH flavor); Solvers were compiled with GCC 3.3.5.; and the Java solver was launched on Sun Java 1.5.0_02 JVM.

2. The new scoring scheme

The importance of the international SAT competition has grown to being an awaited event in the community. The major impact of being ranked among the best solvers is beneficial both for academic and industrial competitors. As it was stated in our previous reports, our ranking was until now based on a series-basis. One main problem of our scoring scheme (see previous competition report for details [21, 4, 5]) was that all solvers in the second stage were awarded the same score (they solved the same number of series). The final ranking of the second stage was often based on the simple number of solved benchmarks, and the scores were very tight (on some categories, the winner solved only one additional benchmarks than the second one). As an additional argument, because solvers may be used with very large CPU-timeout, it is now important to consider the CPU-time of solvers. A fast solver must be preferred to another one if they both solve the same amount of benchmarks, because the faster one may be efficiently embedded in a real-world application where it will have to scale-up well.

2.1 Design Objectives

One key idea behind the SAT competition is to award a solver that is good on a wide range of SAT instances. In the previous year of the competition, this was implemented using a scoring scheme that ranked the solvers with a tiered system: First, the solvers were ranked by being able to solve *some instance* in a highest number of different series. Ties were then broken using the total number of benchmarks solved. Unfortunately, in this system there is no difference between solving a benchmark solved by all solvers or one solved by only a few solvers. The same applies to series too.

Another key idea of the competition was to focus on solvers that are the *only* ones to solve some benchmarks: in the SAT and CASC competitions, those solvers are called *state-of-the-art contributors* (abbreviated SOTAC). In the previous scoring scheme, the solvers did not benefit directly for being SOTAC in their category, even though SOTAC solvers were usually among the top ranked solvers.

Third, the time needed to solve a given benchmark also needs to be considered. While the CPU time was indirectly used for scoring the solvers in the previous years of the SAT competitions, by using a fixed timeout per benchmark, there was no way to discriminate among the solvers able to solve a given benchmark within that timeout.

Furthermore, the second stage ranking was based only on the number of benchmarks solved during the second stage, among those benchmarks that had not been solved by *any solver* during the first stage. This criterion is based on very strong assumptions:

- The remaining benchmarks are representative of the initial set of benchmarks.
- The solvers will behave in the second stage in a way similar to the first stage.

However, these assumptions did not necessarily hold. Although it is likely that the winners of the previous competitions could have been declared winners using various scoring schemes, nevertheless, the rankings of the remaining top solvers could have changed a lot.

The scoring scheme used for the SAT 2005 competition is designed to address these issues. It incorporates these features:

- It gives more credit for solving hard benchmarks than solving easy ones.
- It gives more credit for solving a benchmark fast.
- It gives extra credit for each series solved.
- It stabilizes the rankings of the solvers at the end of the competition.

While the scoring scheme was designed on a purely theoretical basis, the results of the SAT 2005 Competition indicate that the new scoring scheme meets its expectations in practice.

2.2 The Purse-Based Scoring System

The implemented scoring plan works as follows. A *run* is defined to be the execution of one *solver* on one benchmark instance, or *problem*. Each run is allocated a certain amount of CPU time. If the solver succeeds, *timeUsed* records the time.

For SAT 2005, there are three categories of benchmark, `INDUSTRIAL`, `CRAFTED`, and `RANDOM`. Within each category, there are several *specialties*, such as `SAT`, `SAT+UNSAT`, `UNSAT`, and `CERTIFIED-UNSAT`. The scoring system is applied separately within each combination of category and specialty.

Each problem has a *solution purse*, which is divided equally among all competition solvers that solve the problem. For SAT 2005, all problems have the standard solution purse (*stdP*).

Each problem has a *speed purse*, which is divided *unequally* among all competition solvers that solve the problem. The speed purse is a fixed multiple (*spdM*) of the solution purse for all problems in the entire competition; it gives a weighting between solving and speed.

The formula to divide the speed purse of a problem is the following, where p is problem-id and s and i are solver-ids, times are in seconds, and 10,000 is an arbitrary scale factor.

$$speedFactor(p,i) = \begin{cases} \dfrac{10000}{1 + timeUsed(p,i)} & \text{if } i \text{ solved } p; \\ 0 & \text{if } i \text{ did not solve } p. \end{cases} \quad (1)$$

$$speedAward(p,s) = \frac{speedPurse(p) * speedFactor(p,s)}{\sum_i speedFactor(p,i)} \quad (2)$$

Thus, the *speedAward* is pro rata by *speedFactor*.

The series purses reward breadth of application. Each series (within specialty within category) has a *series purse*, which is divided equally among all competition solvers that solve at least one problem in the series. If no solver solves any problem in a certain series, its series purse is not distributed.

For SAT 2005, all series containing 5 or more benchmark instances have the same series purse, which is a fixed multiple (*serM*) of the standard solution purse. (Recall that scoring is separately applied within each combination of category and specialty, e.g., SAT within RANDOM, or SAT+UNSAT within CRAFTED.) All series containing 4 or fewer benchmarks have the same series purse, which is a fixed multiple (*serM* / 3) of the standard solution purse.

The coefficients and multiples for SAT 2005 are:

$$stdP = 1000.0; \qquad spdM = 1.0; \qquad serM = 3.0.$$

3. The SAT 2005 settings

3.1 Submitted Solvers

A 2-pages description of solvers is available on the competition web site. Briefly, the submitters to the **competition division** were : Anbulagan [2] (`dew-satz-1a`, `dew-satz-1b`, `dew-satz-1c`) ; Domagoj Babic (`hsat.1`, `hsat.5`, `hsatrr`) ; Armin Biere (`compsat`), pages 201–208 ; Gilles Dequen, Olivier Dubois [7] (`kcnfs-2004`) ; Niklas Eén, Niklas Sörensson [8] (`SatELiteGTI`) ; Zhaohui Fu [15](`zchaff`) ; Roman Gershman [9] (`haifasat`, `haifasat2`) ; Marijn Heule, Mark Dufour, Joris van Zwieten, Hans van Maaren (pages 47-59, `march-dl`) ; Holger Hoos, Dave Tompkins [24](`adaptnovelty`) ; Daniel Le Berre, Mederic Baron, Geoffrey Bourgeois (`sat4j.jar`) ; Chu Min Li, Wen Qi Huang [13] (`g2wsat`) ; Feng Lu, Kai Yang, Kwang-Ting Cheng (`csat`) ; Yogesh Mahajan, Sharad Malik, Lintao Zhang, Zhaohui Fu [15](`zchaff-rand`) ; Alexander Nadel (`jerusat1.31-a`, `jerusat1.31-b`)) ; Richard Ostrowski [18](`lsatv1.1`, `wllsatv1`) ; Duc Nghia Pham, Anbulagan (`ranov`, `rpaws10`, `rrsaps`) ; Steven Prestwich [19] (`vw`) ; Niklas Sörensson, Niklas Eén [8](`minisat-static`) ; Ivor Spence (`tts-3-0`) ; Dave Tompkins, Holger Hoos, Frank Hutter [24](`saps`) ; Daniel Vallstrom, (`vallst.sh`)

In the **demonstration division**, we had : Anbulagan (`dew-satz-1d`, `dew-satz-1e`) ; Gilles Dequen, Olivier Dubois (`kcnfs`) ; Niklas Een, Niklas Sörensson (`satelite-release`) ; HoonSang Jin, Fabio Somenzi (`circusa`, `circusb`, `circusd`) demonstration ; Raihan Kibria, Niklas Eén, Niklas Sörensson (`midisat-static`) ; Alexander Nadel, Ziyad Hanna (`eureka-a`, `eureka-b`, `eureka-c`) ; Duc Nghia Pham, Anbulagan (`rpaws40`, `rpaws5`)

No portfolios (such as `satzilla` in the previous contest) were submitted this year and `kncfs-2004` (same binary as last year winner) was resubmitted by its authors (Gilles Dequen, Olivier Dubois).

3.2 Benchmarks

The random category only contains pure, uniform, non-biased random k-SAT instances, not forced to be SAT or UNSAT. Oliver Kullmann provides full details and analysis of this category pages 61-102.

Crafted benchmarks contained all benchmarks that were neither random nor industrial. Previous tricky benchmarks that were simple but known as needing an exponential number of resolutions (like Urquhart or XOR-chains problems) were mostly taken out last year by `lsat` and not submitted again this year. Benchmarks that looked like random (for instance forced SAT random benchmarks) were also pushed into this category. As a matter of fact, crafted benchmarks represented the largest set of benchmarks this year (675 benchmarks). We had benchmarks of very different kinds and we shuffled all of them for the contest. Armin Biere submitted `LinvRinv` benchmarks; Matti Järvisalo submitted benchmarks based on 3-Regular Graphs (see details pages 27–46); Ashish Sabharwal submitted counting, ordering and pebbling problems; Inês Lynce submitted social golfer problems [14]; Volker Sorge submitted algebraic benchmarks problems [16]; Klas Markström submitted a

family of Eulerian graphs (pages 221-228) and Oliver Roussel submitted PHNF encoding [20] of last year contest, medium hardness, benchmarks.

In the industrial category, new formal verification benchmarks from IBM were submitted by Emmanuel Zarpas. He discusses the choice of the benchs pages 229-236. Miroslav Velev submitted previously-known benchmarks (VLIW-SAT (2.0 and 4.0), VLIW-UNSAT 2.0 and Liveness UNSAT 2.0)[25]. Sanjai Narain submitted VPN models from Alloy [17]; François Grieu submitted VPMC inversion problems and Fréderic Maris submitted Planning problems.

3.3 First stage results

The results of the first stage are detailed in Table 2 for the random category, Table 3 for the crafted category and Table 4 for the industrial category.

3.4 Second stage results: The winners

For the second stage, the judges were asked to select the Top-N solvers in each category (in each category, N may be different). We then used a significantly larger cpu-time (6,000 seconds for the `RANDOM` and `CRAFTED` categories; and 12,000 seconds for the `INDUSTRIAL` category) to relaunch the selected solvers on all the benchmarks on which they failed during the first stage. The table 5 gives the results for the second stage of the competition for random benchmarks. The table 6 gives the results for the second stage of the competition for crafted benchmarks, and the table 7 gives the results for the second stage of the competition for industrial benchmarks. More information, including various figures and the details of all runs can be found on the competition web site: http://www.satcompetition.org/2005/.

4. Introducing this special volume

The special volume contains two types of contributions: articles and research notes. The articles are expected to reach a wider audience than research notes. All the contributions have been reviewed in the same way. Considering the specific topic of the special volume, the SAT competition and the Quantified Boolean Formulae (QBF) and Pseudo Boolean (PB) evaluations organized as satellite events of the SAT conference, the contributions are either reports of events (pages 61–164, 229), description of technologies behind solvers (pages 1, 47,1 65, 191–219), or analysis of new classes of benchmarks (pages 27, 221).

One of the big winner of the SAT 2005 competition and evaluations is certainly `MiniSat`. It is the second solver in numerous categories in the SAT competition, and the first when used with the `SatELite` preprocessor. In conjunction with a pseudo boolean preprocessor, it also shown a great potential in the first pseudo boolean evaluation. Niklas Eén and Niklas Sörensson are presenting in the first paper, *Translating Pseudo-Boolean Constraints into SAT*, pages 1–26, the original approach taken by `MiniSat+`: translating the pseudo boolean constraints into clauses, on which the powerful `minisat` SAT solver can be launched.

Current SAT solvers are now able to handle larger and larger benchmarks, scaling up to millions of variables and clauses. In the second paper, *Hard Satisfiable Clause Sets for Benchmarking Equivalence Reasoning Techniques*, pages 27–46, Haari Haanpää *et al.* are interested in finding a model to generate small, hard, satisfiable benchmarks. They

Table 2. First stage results, RANDOM category

(A) SAT+UNSAT

Solver	Score	#Solved Sat	#Solved Unsat
ranov	142,367	178	0
g2wsat	92,425	158	0
kcnfs-2004	78,776	80	60
rpaws10	74,857	151	0
vw	60,078	148	0
rrsaps	43,125	116	0
march-dl	20,866	44	30
adaptnovelty	19,780	107	0
saps	16,581	101	0
dew-satz-1a	15,983	51	36
dew-satz-1c	15,331	50	35
dew-satz-1b	14,590	50	34
wllsatv1	9,818	38	29
minisat-static	6,859	34	22
satelitegti	6,273	33	21
sat4j	5,540	35	15
csat	2,329	20	7
hsat-5	2,182	19	0
zchaff	2,069	16	0
hsat-1	1,891	17	0
vallst-sh	1,457	17	0
zchaff-rand	1,325	12	0
jerusat1-31-b	1,247	10	1
hsatrr	881	5	0
jerusat1-31-a	673	7	0
haifasat2	547	5	0
lsatv1-1	432	3	0
haifasat	357	3	0
compsat	349	3	0

(B) SAT

Solver	Score	#Solved
ranov	137,700	178
g2wsat	89,425	158
rpaws10	73,190	151
vw	58,078	148
rrsaps	42,459	116
adaptnovelty	19,780	107
saps	16,581	101
kcnfs-2004	13,444	80
dew-satz-1a	7,296	51
dew-satz-1c	7,004	50
dew-satz-1b	6,601	50
march-dl	6,199	44
wllsatv1	3,802	38
minisat-static	3,658	34
sat4j.jar	3,580	35
satelitegti	3,251	33
hsat-5	2,182	19
zchaff	2,069	16
hsat-1	1,891	17
csat	1,621	20
vallst-sh	1,457	17
zchaff-rand	1,325	12
jerusat1-31-b	1,155	10
hsatrr	881	5
jerusat1-31-a	673	7
haifasat2	547	5
lsatv1-1	432	3
haifasat	357	3
compsat	349	3

(C) UNSAT

Solver	Score	#Solved
kcnfs-2004	79,479	60
march-dl	18,089	30
dew-satz-1a	12,335	36
dew-satz-1c	11,974	35
dew-satz-1b	11,636	34
wllsatv1	8,313	29

Solver	Score	#Solved
minisat-static	4,898	22
satelitegti	4,719	21
sat4j.jar	2,907	15
csat	1,281	7
jerusat1-31-b	364	1

introduce a new family of benchmarks, regular XORSAT, that was submitted to the SAT 2005 competition, crafted category.

If conflict driven clause learning solvers are suitable for solving industrial benchmarks, lookahead solvers with sophisticate heuristics and deduction processes are another appealing approach for solving both random k-SAT and crafted benchmarks. Marijn Heule and Hans van Maaren introduce in the third paper, *March_dl: Adding Adaptive Heuristics and a New*

Table 3. First stage results, CRAFTED category

(A) SAT+UNSAT

Solver	Score	#Solved Sat	#Solved Unsat
satelitegti	79,069	173	145
vallst-sh	60,869	195	130
tts-3-0	56,951	20	78
march-dl	55,428	182	111
zchaff-rand	52,064	159	119
minisat-static	51,536	163	136
csat	49,476	158	113
jerusat1-31-a	43,555	167	109
hsat-1	41,719	158	105
hsat-5	41,213	157	105
zchaff	37,601	169	97
haifasat	33,840	156	95
jerusat1-31-b	32,660	167	100
hsatrr	31,482	111	66
haifasat2	29,790	153	96
compsat	23,888	143	83
dew-satz-1b	21,250	125	67
sat4j.jar	21,179	136	73
dew-satz-1a	21,063	127	66
dew-satz-1c	19,868	124	63
lsatv1-1	19,055	61	37
vw	17,047	132	0
wllsatv1	11,912	86	50
rrsaps	11,312	146	0
g2wsat	10,393	128	0
ranov	9,561	134	0
kcnfs-2004	8,269	49	34
rpaws10	7,875	101	0
saps	5,221	97	0
adaptnovelty	5,130	87	0

(B) SAT

Solver	Score	#Solved
satelitegti	38,063	173
march-dl	31,656	182
vallst-sh	29,366	195
minisat-static	23,913	163
hsat-1	23,502	158
hsat-5	22,894	157
zchaff	21,040	169
jerusat1-31-a	20,919	167
haifasat	20,078	156
csat	19,257	158
zchaff-rand	18,975	159
jerusat1-31-b	18,887	167
hsatrr	18,487	111
vw	17,869	130
haifasat2	16,959	153
compsat	14,952	143
sat4j.jar	12,395	136
rrsapes	12,184	146
dew-satz-1c	11,392	124
g2wsat	11,060	127
dew-satz-1a	10,848	127
lsatv1-1	10,837	61
dew-satz-1b	10,543	125
ranov	10,360	134
rpaws10	9,420	99
wllsatv1	7,274	86
saps	5,735	96
adaptnovelty	5,566	87
kcnfs-2004	4,205	49
tts-3-0	1,352	20

(C) UNSAT

Solver	Score	#Solved
tts-3-0	55,211	78
satelitegti	46,427	145
csat	35,227	113
zchaff-rand	34,597	117
vallst-sh	33,065	129
minisat-static	31,366	136
march-dl	25,253	111
jerusat1-31-a	24,387	109
hsat-5	20,216	105
hsat-1	20,056	105
zchaff	18,849	97
jerusat1-31-b	15,385	100

Solver	Score	#Solved
hsatrr	14,637	66
haifasat2	14,525	96
haifasat	14,427	95
dew-satz-1b	12,669	67
dew-satz-1a	12,173	66
sat4j.jar	11,204	73
compsat	10,613	83
dew-satz-1c	9,902	63
lsatv1-1	8,551	36
wllsatv1	6,000	50
kcnfs-2004	4,249	35

Table 4. First stage results, INDUSTRIAL category

(A) SAT+UNSAT

Solver	Score	#Solved Sat	#Solved Unsat
satelitegti	85,602	117	78
minisat-static	55,638	114	74
haifasat	39,359	99	79
haifasat2	31,312	94	77
zchaff-rand	30,400	91	78
zchaff	27,201	90	60
compsat	26,955	88	67
csat	25,106	94	78
jerusat1-31-b	22,358	96	65
jerusat1-31-a	18,885	94	65
hsat-5	18,593	91	45
hsat-1	17,645	91	45
sat4j.jar	16,793	88	61
vallst-sh	15,271	80	63
wllsatv1	12,623	82	5
hsatrr	11,820	66	29
march-dl	10,051	67	39
dew-satz-1a	7,116	61	24
rrsaps	6,028	71	0
dew-satz-1c	5,795	60	5
dew-satz-1b	5,530	60	5
vw	5,369	67	0
g2wsat	5,291	65	0
rpaws10	5,248	65	0
ranov	4,563	59	0
lsatv1-1	4,117	56	0
saps	3,442	56	0
kcnfs-2004	3,178	40	0
adaptnovelty	3,154	50	0
tts-3-0	1,544	28	0

(B) SAT

Solver	Score	#Solved
satelitegti	52,497	117
minisat-static	36,651	114
haifasat	19,051	99
compsat	17,756	88
haifasat2	15,821	94
zchaff	15,350	90
jerusat1-31-b	14,598	96
zchaff-rand	14,290	91
csat	13,416	94
wllsatv1	11,605	82
jerusat1-31-a	11,262	94
hsat-5	10,739	91
hsat-1	9,809	91
sat4j.jar	8,445	88
vallst-sh	7,080	80
rrsaps	6,028	71
hsatrr	5,960	66
vw	5,369	67
g2wsat	5,291	65
rpaws10	5,248	65
march-dl	5,175	67
dew-satz-1c	4,647	60
ranov	4,563	59
dew-satz-1a	4,479	61
dew-satz-1b	4,422	60
lsatv1-1	4,117	56
saps	3,442	56
kcnfs-2004	3,178	40
adaptnovelty	3,154	50
tts-3-0	1,544	28

(C) UNSAT

Solver	Score	#Solved
satelitegti	34,481	78
haifasat	20,802	79
minisat-static	20,029	74
zchaff-rand	15,985	78
haifasat2	15,324	77
csat	12,684	78
zchaff	11,631	60
compsat	8,470	67
sat4j.jar	7,919	61
jerusat1-31-b	8,420	65
vallst-sh	8,266	63
jerusat1-31-a	7,480	65
hsat-5	7,411	45
hsat-1	7,393	45
hsatrr	5,836	29
march-dl	4,576	39
dew-satz-1a	2,436	24
wllsatv1	994	5
dew-satz-1c	947	5
dew-satz-1b	907	5

Table 5. Second stage results, `Random` category

(A) `SAT+UNSAT`

Solver	Score	#Solved Sat	#Solved Unsat
`kcnfs-2004`	95,075	92	75
`march-dl`	27,141	56	43
`dsatz-1a`	22,940	68	50
`wllsatv1`	16,145	59	45
`satelitegti`	10,074	46	33
`minisat`	10,058	45	33

(B) `SAT`

Solver	Score	#Solved
`ranov`	163,903	209
`g2wsat`	101,286	178
`vw`	76,002	170
`adaptnovelty`	21,748	119
`saps`	15,603	104
`kcnfs-2004`	14,604	92
`dsatz-1a`	8,943	68
`march-dl`	7,444	56
`wllsatv1`	7,202	59
`satelitegti`	5,198	46
`minisat`	5,147	45

(C) `UNSAT`

Solver	Score	#Solved
`kcnfs-2004`	97,930	75
`march-dl`	25,228	43
`dsatz-1a`	19,456	50
`wllsatv1`	12,902	45
`minisat`	7,369	33
`sateLitegti`	7,335	33

Branching Strategy, pages 47–59, the new version of `march_dl`, one of the strong solvers of the SAT 2005 competition (3 silver and 2 bronze medals in the random and crafted categories).

One of the reasons of the success of the random k-SAT model in the SAT community is it's strong mathematical basis. Oliver Kullmann, one of the judges of the SAT 2005 competition, was in charge of the design of the random category. He reports in *The SAT 2005 Solver Competition on Random instances*, pages 61-102, the way he built the set of benchmarks to be used for the competition and analyzes the solvers performances on those benchmarks.

The most successful special track of the SAT competition was the Pseudo Boolean evaluation. Vasco Manquinho and Olivier Roussel report in details the event in *The First Evaluation of Pseudo-Boolean Solvers (PB'05)*, pages 103–143. The report is especially interesting because many solvers were found incorrect during the evaluation for mainly two reasons: there was no common input format before the evaluation so I/O errors were frequent and the arithmetic on the coefficients needed special attention to avoid overflow. As a consequence, the organizers decided to rerun corrected version of those solvers after the SAT conference. The report is presenting results updated in September 2005.

For the third time, a QBF evaluation was organized. Massimo Narizzano *et al.* detail in *The third QBF solvers comparative evaluation*, pages 145–164, the way the set of benchmarks used was designed and classified. The behavior of the solvers on those benchmarks is then discussed. The evaluation once again emphasized the difficulty to check the correctness of a QBF solver: 5 solvers out of 13 answered incorrectly this year.

Table 6. Second stage results, `Crafted` category

(A) `SAT+UNSAT`

Solver	Score	#Solved Sat	#Solved Unsat
`vallst.sh`	56,445	138	100
`satelitegti`	53,128	122	126
`march-dl`	52,432	138	99
`minisat`	43,691	122	121
`hsat-1`	39,497	130	90
`csat`	38,324	113	112
`zchaff`	27,455	112	89
`zchaff-rand`	24,171	107	78
`tts-3-0`	21,298	5	54
`jerusat-a`	19,632	104	77

(B) `SAT`

Solver	Score	#Solved
`vallst.sh`	31,258	138
`march-dl`	27,656	138
`hsat-1`	20,156	130
`satelitegti`	17,418	122
`minisat`	17,210	122
`csat`	13,791	113
`zchaff`	13,692	112
`zchaff-rand`	11,431	107
`jerusat-a`	10,702	104
`tts`	475	5

(C) `UNSAT`

Solver	Score	#Solved
`satelitegti`	35,639	126
`minisat`	26,159	121
`vallst.sh`	25,532	100
`march-dl`	25,371	99
`csat`	23,878	112
`tts-3-0`	20,765	54
`hsat-1`	19,936	90
`zchaff`	14,359	89
`zchaff-rand`	12,419	78
`jerusat-a`	9,275	77

Hossein Sheini and Karem Sakallah present their new pseudo boolean solver in *Pueblo: A Hybrid Pseudo-Boolean SAT Solver*, pages 165–189. The main feature of that extension of the `MiniSat` solver for pseudo boolean constraints is to use both clauses and pseudo boolean constraints during learning and backjumping. The details needed to implement a similar solver on top of Minisat are provided.

Five research notes complete this special volume. First, Olivier Bailleux *et al* submitted to the PB 05 evaluation a solver also based on a translation of the original problem into SAT. *A Translation of Pseudo Boolean Constraints to SAT*, page 191–200, completes the previous work on `MiniSat+` by a new encoding.

Armin Biere and Carsten Sinz propose in *Connected components in Compsat*, pages 201–208 a way to deals with connected components in conflict driven clause learning solvers, as implemented in the SAT 2005 competitor `Compsat`.

Vasco Manquinho and João Marques-Silva describe in *On Using Cutting Planes in Pseudo-Boolean Optimization*, pages 209–219 the solver `bsolo`, a branch and bound PB solver in which cutting planes are used both for conflict analysis and to improve lower bounds.

Klas Markström proposes in *Hard SAT-instances and locality*, pages 221–227, a new class of problems based on Eulerian graphs whose instances are expected to be hard for resolution-based SAT solvers.

Table 7. Second stage results, `Industrial` category

(A) `SAT+UNSAT`

Solver	Score	#Solved Sat	#Solved Unsat
satelitegti	99,662	180	87
minisat	69,485	166	84
haifasat	50,931	151	91
zchaff-rand	50,515	132	94
jerusat-b	47,487	163	80
csat	36,526	140	91
zchaff	31,702	121	76
compsat	25,399	114	75
sat4j	21,097	110	70
hsat-5	20,995	99	54
vallst.sh	16,874	85	69
wllsatv1	12,467	86	6

(B) `SAT`

Solver	Score	#Solved
satelitegti	73,506	180
minisat	50,985	166
jerusat-b	38,625	163
haifasat	28,428	151
zchaff-rand	24,885	132
csat	21,997	140
zchaff	19,236	121
compsat	16,715	114
sat4j	12,898	110
wllsatv1	11,390	86
hsat-5	11,046	99
vallst.sh	7,757	85

(C) `UNSAT`

Solver	Score	#Solved
satelitegti	27,518	87
zchaff-rand	26,792	94
haifaSat	23,666	91
minisat	19,863	84
csat	15,892	91
zchaff	13,829	76
jerusat-b	10,225	80
hsat-5	10,029	54
vallst	9,192	69
compsat	9,097	75
sat4j	8,654	70
wllsatv1	1,053	6

Emmanuel Zarpas concludes the special volume by *Back to SAT05 competition: an a posteriori analysis of solvers performances on industrial benchmarks*, pages 229–237, in which the solvers are compared mainly on the various SAT-based Bounded Model Checking benchmarks coming from the IBM Formal Verification Benchmarks Library.

Special Thanks

We would like to thank the numerous reviewers involved in that special volume: Fadi Aloul, Anbulagan, Gilles Audemard, Olivier Bailleux, Peter Barth, Roberto Bayardo, Armin Biere, Alexander Bockmayr, Yacine Boufkhad, Franc Brglez, Gilles Dequen, Heidi Dixon, Nilkas Eén, Uwe Egly, Marijn Heule, Eugene Goldberg, Edward A. Hirsch, Holger Hoos, Andreas Kuehlmann, Oliver Kullmann, Chu-Min Li, Inês Lynce, João Marques-Silva, Massimo Narizano, Alexander Nadel, Guoqiang Pan, Steven Prestwich, Jussi Rintanen, Olivier Roussel, Karem Sakallah, Carsten Sinz, Niklas Sörensson, Armando Tacchella, Hans van Maaren Miroslav Velev, Ke Xu, Yinlei Yu and Riccardo Zecchina.

We also would like to thank the three judges for their active involvement in all the choices made at each step of the contest. Armin Biere was our consultant on industrial benchmarks and solvers. Olivier Kullmann generated all the benchmarks for the random

category. Allen van Gelder, who proposed the new scoring scheme, was particularly active. We also want to thank the LRI and Michal Kouril (EECS) for providing us with a very large amount of CPU time.

Finally, we would like to thank Marijn Heule for his help with all the editing issues and Hans van Maaren, the editor in chief of JSAT, for all its help for making this volume a reality.

References

[1] *Eighth International Conference on Theory and Applications of Satisfiability Testing SAT'05*, St. Andrews, Scotland, June 19th-23rd 2005.

[2] Anbulagan and John Slaney. Lookahead saturation with restriction for sat. In *Eleventh International Conference on Principles and Practice of Constraint Programming*, pages 727–731, 2005.

[3] C. Barrett, L. de Moura, and A. Stump. SMT-COMP: Satisfiability Modulo Theories Competition. In K. Etessami and S. Rajamani, editors, *17th International Conference on Computer Aided Verification*, pages 20–23. Springer, 2005.

[4] D. Le Berre and L. Simon. The essentials of the sat'03 competition. In E. Giunchiglia and A. Tacchella, editors, *Proceedings of the Sixth International Conference on Theory and Applications of Satisfiability Testing (SAT2003)*, LNCS, pages 452–467. 2003.

[5] D. Le Berre and L. Simon. Fifty-five solvers in vancouver: The sat 2004 competition. In Hoos and Mitchell [10], pages 468–485. Revised Selected Papers.

[6] M. Buro and H. K. Büning. Report on a sat competition. *Bulletin of the European Association for Theoretical Computer Science*, **49**:143–151, 1993.

[7] Olivier Dubois and Gilles Dequen. A backbone-search heuristic for efficient solving of hard 3-sat formulae. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, Seattle, Washington, USA, August 4th-10th 2001.

[8] Niklas Een and Armin Biere. Effective preprocessing in SAT through variable and clause elimination. [1], pages 61–75.

[9] Roman Gershman and Ofer Strichma. Cost-Effective Hyper-Resolution for Preprocessing CNF Formulas. [1], pages 423–429.

[10] Holger H. Hoos and David G. Mitchell, editors. *Procedings of the Seventh International Conference on Theory ans Applications of Satisfiability Testing (SAT2004)*, volume **3542** of *LNCS*, Vancouver, BC, Canada, May 2004. Springer. Revised Selected Papers.

[11] Holger H. Hoos and Thomas Stützle. Satlib: An online resource for research on sat. In I.P.Gent, H.v.Maaren, and T.Walsh, editors, *SAT 2000, Highlights of Satisfiability Research in the Year 2000*, pages 283–292. IOS Press, 2000. http://www.satlib.org/.

[12] D. S. Johnson and M. A. Trick, editors. *Second DIMACS implementation challenge : cliques, coloring and satisfiability*, volume **26** of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1996.

[13] Chu-Min Li and Wen Qi Huang. Diversification and determinism in local search for satisfiability. [1], pages 158–172.

[14] Inês Lynce and Ian P. Gent. A SAT Encoding for the Social Golfer Problem. In *IJCAI'05 workshop on Modelling and Solving Problems with Constraints*, July 2005.

[15] Yogesh S. Mahajan, Zhaohui Fu, and Sharad Malik. Zchaff2004: An efficient sat solver. In Hoos and Mitchell [10], pages 360–375. Revised Selected Papers.

[16] Andreas Meier and Volker Sorge. A New Set of Algebraic Benchmark Problems for SAT Solvers. [1], pages 459–466.

[17] Sanjai Narain. Network configuration management via model finding. In *Proceedings of USENIX Large Installation System Administration (LISA) Conference*, San Diego, December 4–5 2005.
http://alloy.mit.edu/papers/NetConfigAlloy.pdf.

[18] R. Ostrowski, E. Grégoire, B. Mazure, and L. Sais. Recovering and exploiting structural knowledge from cnf formulas. In *Proc. of the Eighth International Conference on Principles and Practice of Constraint Programming (CP'2002)*, LNCS, pages 185–199, Ithaca (N.Y.), September 2002. Springer.

[19] Steven Prestwich. Random Walk With Continuously Smoothed Variable Weights. [1], pages 203–215.

[20] O. Roussel. Another SAT to CSP Conversion. In *16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'04)*, pages 558–565, 2004.

[21] L. Simon, D. Le Berre, and E. Hirsch. The sat2002 competition. *Annals of Mathematics and Artificial Intelligence (AMAI)*, **43**:343–378, 2005.

[22] Laurent Simon and Philippe Chatalic. SATEx: a web-based framework for SAT experimentation. In Henry Kautz and Bart Selman, editors, *Electronic Notes in Discrete Mathematics*, volume **9**. Elsevier Science Publishers, June 2001.
http://www.lri.fr/~simon/satex/satex.php3.

[23] G. Sutcliffe. The IJCAR-2004 Automated Theorem Proving Competition. *AI Communications*, **18**(1):33–40, 2005.

[24] Dave A.D. Tompkins and Holger H. Hoos. UBCSAT: An Implementation and Experimentation Environment for SLS Algorithms for SAT and MAX-SAT. In Hoos and Mitchell [10], pages 306–320. Revised Selected Papers.

[25] M.N. Velev. Exploiting signal unobservability for efficient translation to cnf in formal verification of microprocessors. In *Formal Verification of Microprocessors, Design, Automation and Test in Europe (DATE '04)*, pages 266–271, 2004.