Benchmarks

Chuan Luo[1,2], Holger H. Hoos[2], and Shaowei Cai[3]

[1] Microsoft Research, China
[2] Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands
[3] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China
chuan.luo@microsoft.com, hh@liacs.nl, caisw@ios.ac.cn

1 Benchmarks

The benchmarks we used in our evaluation cover a broad range of well-known applications of SAT solvers. Table 1 provides an overview; for each benchmark set, we list the number of total instances ('#inst'), the number of training instances ('#train') we used for automated configuration, the number of testing instances ('#test'), as well as basic statistics for the number of variables ('#var') and clauses ('#clause'), including the average ('avg'), median ('median'), 0.25- and 0.75-quantiles ('q25' and 'q75', respectively). Further information on each set is given in the following. The total number of instances used in this work is 26 932, and to our best knowledge is much more than those used in previous work on SAT.

`FCC-SAT + FCC-UNKNOWN` Recently, SAT solvers have been prominently used by the US Federal Communication Commission (FCC) for revenue-optimising spectrum repacking in the context of a bandwidth auction that resulted in a 7 billion dollar revenue for the US government [4, 9]. We used the SAT-encoded benchmark instances made available by Newman *et al.* [9],[4] which contains 10 000 SAT instances, 9482 of which are known to be satisfiable and 121 unsatisfiable, while the satisfiability of the remaining 397 instances is unknown. Since SLS solvers such as *PbO-CCSAT* are unable to prove unsatisfiability, we discarded all unsatisfiable instances. Furthermore, we selected 100 of the 9482 satisfiable instances as the training set used for automated solver configuration; we refer to this set as `FCC-SAT [Train]`.[5] The remaining 9382 satisfiable instances make up our test set, `FCC-SAT [Test]`; we also considered an additional test set, `FCC-UNKNOWN`, consisting of the 397 instances whose satisfiability was unknown prior to our experiments.

[4] `https://www.cs.ubc.ca/labs/beta/www-projects/SATFC/cacm_cnfs.tar.gz`

[5] Since we need training sets in which most instances can be solved within a given reasonable cutoff time, we selected 80 instances uniformly at random from those instances which could be solved by the default version of *PbO-CCSAT* within a cutoff time of 5000 CPU seconds and 20 instances uniformly at random from those instances which could not be solved by the default version of *PbO-CCSAT* within a cutoff time of 5000 CPU seconds.

Table 1: Bechmarks for training and testing in our experiments.

Benchmark	#inst	#train/#test	#var		#clause		Reference
			avg q25	median q75	avg q25	median q75	
`FCC-SAT`	9 482	100/9 382	27 633.3 25 712	27 993.0 30 477	653 302.0 553 185	647 061.5 762 108	[9]
`FCC-UNKNOWN`[a]	397	0/397	32 562.9 30 564	32 562.0 34 593	867 845.4 763 157	867 708.0 968 107	[9]
`PTN`	23	11/12	4 071.1 3 696	3 721.0 3 733	14 321.5 13 802	14 160.0 14 455	[8]
`PTN-More`[b]	556	0/556	7 413.7 7 208	7 416.0 7 620	17 800.1 17 226	17 799.0 18 366	[8]
`SMT-QF-BV`	16 434	100/16 334	174 748.3 843	24 906.0 113 975	480 460.9 1 709	71 287.0 305 428	[12]
`Community`	20	5/15	2 200.0 2 200	2 200.0 2 200	9 086.0 9 086	9 086.0 9 086	[6]
`SC17-mp1-9`	20	5/15	1 458.0 1 458	1 458.0 1 458	280 139.1 280 136	280 139.5 280 140	[2]

[a] For *PbO-CCSAT* and its competitors, the configurations trained on `FCC-SAT [Train]` are the ones tested on `FCC-UNKNOWN`.
[b] For *PbO-CCSAT* and its competitors, the configurations trained on `PTN [Train]` are the ones tested on `PTN-More`.

`PTN + PTN-More` SAT techniques have been recently applied to (and play prominent roles in) tackling a long-standing open problem in mathematics known as Boolean Pythagorean Triples (PTN) [8, 7]. We used 23 publically available SAT-encoded PTN instances, 2 of which – `plain7824.cnf`[6] and `bce7824.cnf`[7] – were published by Heule *et al.* [8], while the 21 others were taken from the crafted benchmark of the 2016 SAT Competition.[8] Of these 23 SAT-encoded instances, we selected 11 uniformly random for use as a training set; we refer to this set as `PTN [Train]`. The remaining 12 instances make up our test set, `PTN [Test]`. Moreover, in order further explore the efficiency of *PbO-CCSAT* on PTN instances, we used the PTN encoder[9] by Heule *et al.* [8] to generate 556 additional satisfiable instances, according to the suggestions by the authors, to obtain an additional test set, `PTN-More`.

`SMT-QF-BV` SLS solvers for SAT have recently achieved promising results on solving the Satisfiability Modulo Theories (SMT) problem in the theory of quantifier-free bit-vectors (QF-BV) [5]. We first downloaded 16 436 SMT-encoded QF-BV instances made available by Niemetz *et al.* [11, 12];[10] then, we utilised *Boolector*[11] [10] and *aigtocnf*[12] [3] to translate those SMT-encoded instances into SAT, according to suggestions by Niemetz *et al.* [11, 12]. Since the translation process of 2 instances (`bench_3774.smt2` and `bench_3843.smt2`) could not be completed within 24 CPU hours on our reference machines, we discarded those 2 instances and used the resulting 16 434 SAT-encoded instances. Of these, we selected 100

[6] `http://www.cs.utexas.edu/~marijn/ptn/plain7824.cnf`
[7] `http://www.cs.utexas.edu/~marijn/ptn/bce7824.cnf`
[8] `https://baldur.iti.kit.edu/sat-competition-2016/downloads/crafted16.zip`
[9] `http://www.cs.utexas.edu/~marijn/ptn/ptn-encode.c`
[10] `http://fmv.jku.at/fmsd16/fmsd16-benchmarks-sat.7z`
[11] `http://fmv.jku.at/boolector/boolector-2.4.1-with-lingeling-bbc.tar.bz2`
[12] `http://fmv.jku.at/aiger/aiger-1.9.9.tar.gz`

instances as training set; we refer to this set as SMT-QF-BV [Train].[13] The remaining 16 334 instances make up our test set, SMT-QF-BV [Test].

Community This benchmark consists of instances with community structure which are generated by an industrial model called *Community Attachment* [6]. Also, this benchmark is included in the application track of the 2016 SAT Competition [1]. We used all 20 Community instances from the application track of the 2016 SAT Competition.[14] We selected 5 instances uniformly at random to obtain a training set Community [Train]; the remaining 15 instances form the testing set Community [Test].

SC17-mp1-9 Finally, we considered a set of satisfiable instances from the main track of the 2017 SAT Competition [2].[15] Starting with the complete set of application instances from the competition, we filtered out all instances known to be unsatisfiable, resulting in a reduced set, SC17. We note that the application benchmarks used in SAT competitions are heterogeneous in that they include various types of structured instances, with most instance families containing relatively few instances. Since automated algorithm configuration is primarily intended for performance optimisation on relatively homogeneous families of problem instances, we selected from SC17 those instance families containing at least 20 instances. This resulted in three instance families, mp1-9, mp1-ps and g2-ak128, from SC17. In preliminary experiments, we found that all solvers we considered (including SLS, CDCL and hybrid solvers) were unable to solve any of the instances in mp1-ps and may well be unsatisfiable, as are all instances from this family that were solved in the competition. Furthermore, in preliminary experiments, 5 of the instances in g2-ak128 could be solved by SLS solvers within the cutoff time of 5000 CPU seconds, but these 5 instances were all trivial for SLS solvers such as *PbO-CCSAT* if all variables were set to *false* in the initial assignment. In light of this, since our interest is in solving non-trivial, structured SAT instances, we also eliminated instance family g2-ak128. This left us with one benchmark: SC17-mp1-9, consisting of all 20 instances in mp1-9. We selected 5 instances uniformly at random from each of the set to obtain the training set, SC17-mp1-9 [Train]; the remaining 15 instances form the testing set, SC17-mp1-9 [Test].

References

1. Balyo, T., Heule, M.J.H., Järvisalo, M. (eds.): Proceedings of SAT Competition 2016: Solver and Benchmark Descriptions. University of Helsinki (2016)
2. Balyo, T., Heule, M.J.H., Järvisalo, M. (eds.): Proceedings of SAT Competition 2017: Solver and Benchmark Descriptions. University of Helsinki (2017)

[13] We selected 80 instances uniformly at random from those instances that were solved by the default configuration of *PbO-CCSAT* within a cutoff time of 5000 CPU seconds, and 20 instances uniformly at random from those instances that remained unsolved within this cutoff time.

[14] https://baldur.iti.kit.edu/sat-competition-2016/downloads/app16.zip

[15] https://baldur.iti.kit.edu/sat-competition-2017/benchmarks/Main.zip

3. Biere, A., Heljanko, K., Wieringa, S.: AIGER 1.9 and beyond. Tech. Rep. Technical Report 11/2, Institute for Formal Models and Verification, Johannes Kepler University (July 2011)
4. Fréchette, A., Newman, N., Leyton-Brown, K.: Solving the station repacking problem. In: Proceedings of AAAI 2016. pp. 702–709 (2016)
5. Fröhlich, A., Biere, A., Wintersteiger, C.M., Hamadi, Y.: Stochastic local search for satisfiability modulo theories. In: Proceedings of AAAI 2015. pp. 1136–1143 (2015)
6. Giráldez-Cru, J., Levy, J.: Generating SAT instances with community structure. Artificial Intelligence **238**, 119–134 (2016)
7. Heule, M.J.H., Kullmann, O.: The science of brute force. Communications of the ACM **60**(8), 70–79 (2017)
8. Heule, M.J.H., Kullmann, O., Marek, V.W.: Solving and verifying the Boolean pythagorean triples problem via cube-and-conquer. In: Proceedings of SAT 2016. pp. 228–245 (2016)
9. Newman, N., Fréchette, A., Leyton-Brown, K.: Deep optimization for spectrum repacking. Communications of the ACM **61**(1), 97–104 (2018)
10. Niemetz, A., Preiner, M., Biere, A.: Boolector 2.0 system description. Journal on Satisfiability, Boolean Modeling and Computation **9**, 53–58 (2015)
11. Niemetz, A., Preiner, M., Biere, A.: Precise and complete propagation based local search for satisfiability modulo theories. In: Proceedings of CAV 2016. pp. 199–217 (2016)
12. Niemetz, A., Preiner, M., Biere, A.: Propagation based local search for bit-precise reasoning. Formal Methods in System Design **51**(3), 608–636 (2017)