Competitors

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 Competitors

In our experiments, we assessed the performance of *PbO-CCSAT* against that of 13 state-of-the-art SAT solvers, including 6 SLS, 6 CDCL and 1 hybrid solver; these solvers were chosen based on their performance on well-known SAT benchmarks and prominence within the SAT community.

SLS-based solvers:

- ***GNovelty+PCL*** [13] is a high-performance SLS solver known to perform well on SAT-encoded spectrum repacking instances [11]. In our experiments, we used the version that participated in the 2013 Configurable SAT Solver Challenge and automatically configured 5 parameters.[4]
- ***DDFW*** [5] is an efficient dynamic local search solver for SAT. We include this solver in our comparison since it is known to perform well on PTN instances [4]. In our experiments, we used the *UBCSAT* implementation of *DDFW*, which is efficient and readily available [14], and automatically configured 2 parameters.[5]
- ***SATenstein*** [6] is a unified SLS solver framework that integrates components from a broad range of prominent SLS-based SAT algorithms. According to the literature, *SATenstein* performs well on at least two structured SAT benchmarks, `CBMC` and `FAC`. In our experiments, we used the latest version of *SATenstein* made available by its authors [6] and automatically configured 76 parameters.[6]
- ***YalSAT*** [2] is the winner of Random Track of the 2017 SAT Competition. In our experiments, we used the competition version as provided by the author and automatically configured 32 parameters.[7]
- ***Sparrow*** [1] is the winner of Random Track of the 2011 SAT Competition and is known to be quite effective in solving structured SAT instances. In our experiments, we used the *UBCSAT* implementation of *Sparrow*, which

[4] `http://www.cs.ubc.ca/labs/beta/Projects/CSSC2013/cssc_final.tgz`
[5] `http://ubcsat.dtompkins.com/downloads/ubcsat-beta-12-b18.tar.gz`
[6] `http://www.cs.ubc.ca/labs/beta/Projects/SATenstein/SATensteinAIJ.rar`
[7] `https://baldur.iti.kit.edu/sat-competition-2017/solvers/random/yalsat-03s.zip`

is efficient and readily available [14], and we automatically configured 4 parameters.[5]
- ***Sattime*** [7] is an efficient local search solver for solving structured SAT instances. In our experiments, we used the version that participated in the 2013 SAT Competition.[8] *Sattime* does not expose any configurable parameters and therefore ran in our experiments exactly as provided by its authors.

CDCL-based solvers:

- ***Lingeling*** [2] is an efficient CDCL solver that won a number of awards in SAT Competitions. In our experiments, we used the competition version as provided by the author and configured 333 parameters.[9]
- ***MapleCOMSPS*** [8] is the winner of Main Track of the 2016 SAT Competition. In our experiments, we used competition version as provided by its authors and automatically configured 20 parameters.[10]
- ***Maple_LCM_Dist*** [9] is the winner of Main Track of the 2017 SAT Competition. In our experiments, we used the competition version as provided by its authors and automatically configured 20 parameters.[11].
- ***COMiniSatPS_Pulsar*** [12] is the winner of NoLimit Track of the 2017 SAT Competition. In our experiments, we used the competition version as provided by the author and automatically configured 18 parameters.[12]
- ***MapleLCMDistChronoBT*** [10] is the winner of Main Track of the 2018 SAT Competition. In our experiments, we used the competition version as provided by the author and automatically configured 2 parameters. [13]
- ***MapleLCMDistChrBt-DL-v3*** is the winner of the 2019 SAT Race. In our experiments, we used the competition version as provided by the author and automatically configured 2 parameters.[14]

Hybrid solver:

We also included ***Dimetheus*** [3] in our comparison – a complex solver that effectively hybridises preprocessing, CDCL, SLS and message passing techniques. In our experiments, we used the version of *Dimetheus* that won the Random Track of the 2016 SAT Competition, as provided by the author, and automatically configured 145 parameters.[15]

[8] `http://satcompetition.org/edacc/SATCompetition2013/solver-download/950`
[9] `https://baldur.iti.kit.edu/sat-competition-2017/solvers/main/lingeling-bbe.zip`
[10] `https://baldur.iti.kit.edu/sat-competition-2016/solvers/main/MapleCOMSPS_DRUP.zip`
[11] `https://baldur.iti.kit.edu/sat-competition-2017/solvers/main/Maple_LCM_Dist.zip`
[12] `https://baldur.iti.kit.edu/sat-competition-2017/solvers/nolimits/COMiniSatPS_Pulsar_no_drup.zip`
[13] `http://sat2018.forsyte.tuwien.ac.at/solvers/main_and_glucose_hack/MapleLCMDistChronoBT.zip`
[14] `http://sat-race-2019.ciirc.cvut.cz/solvers/MapleLCMDiscChronoBT-DL-v3.zip`
[15] `https://baldur.iti.kit.edu/sat-competition-2016/solvers/random/dimetheus.zip`

References

1. Balint, A., Fröhlich, A.: Improving stochastic local search for SAT with a new probability distribution. In: Proceedings of SAT 2010. pp. 10–15 (2010)
2. Biere, A.: CaDiCaL, Lingeling, Plingeling, Treengeling and YalSAT entering the SAT competition 2017. In: Proceedings of SAT Competition 2017: Solver and Benchmark Descriptions. pp. 14–15 (2017)
3. Gableske, O.: On the interpolation between product-based message passing heuristics for SAT. In: Proceedings of SAT 2013. pp. 293–308 (2013)
4. 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)
5. Ishtaiwi, A., Thornton, J., Sattar, A., Pham, D.N.: Neighbourhood clause weight redistribution in local search for SAT. In: Proceedings of CP 2005. pp. 772–776 (2005)
6. KhudaBukhsh, A.R., Xu, L., Hoos, H.H., Leyton-Brown, K.: SATenstein: Automatically building local search SAT solvers from components. Artificial Intelligence **232**, 20–42 (2016)
7. Li, C.M., Li, Y.: Satisfying versus falsifying in local search for satisfiability. In: Proceedings of SAT 2012. pp. 477–478 (2012)
8. Liang, J.H., Ganesh, V., Poupart, P., Czarnecki, K.: Learning rate based branching heuristic for SAT solvers. In: Proceedings of SAT 2016. pp. 123–140 (2016)
9. Luo, M., Li, C., Xiao, F., Manyà, F., Lü, Z.: An effective learnt clause minimization approach for CDCL SAT solvers. In: Proceedings of IJCAI 2017. pp. 703–711 (2017)
10. Nadel, A., Ryvchin, V.: Chronological backtracking. In: Proceedings of SAT 2018. pp. 111–121 (2018)
11. Newman, N., Fréchette, A., Leyton-Brown, K.: Deep optimization for spectrum repacking. Communications of the ACM **61**(1), 97–104 (2018)
12. Oh, C.: COMiniSatPS Pulsar and GHackCOMSPS. In: Proceedings of SAT Competition 2017: Solver and Benchmark Descriptions. pp. 12–13 (2017)
13. Pham, D.N., Duong, T., Sattar, A.: Trap avoidance in local search using pseudo-conflict learning. In: Proceedings of AAAI 2012. pp. 542–548 (2012)
14. Tompkins, D.A.D., Hoos, H.H.: UBCSAT: An implementation and experimentation environment for SLS algorithms for SAT and MAX-SAT. In: Proceedings of SAT 2004. pp. 306–320 (2004)