MaxHS in the 2020 MaxSat Evaluation

Fahiem Bacchus
Department of Computer Science
University of Toronto
Ontario, Canada
Email: fbacchus@cs.toronto.edu

1. MaxHS

MaxHS is a MaxSat solver originally developed in [6]. It was the first MaxSat solver to utilize the Implicit Hitting Set (IHS) approach, and its core components are described in [6], [4], [5], [7]. Additional useful insights into IHS are provided in [8], [9]. IHS solvers utilize both an integer programming (IP) solver and a SAT solver in a hybrid approach to MaxSat solving. This separation also allows additional techniques from IP solving to be exploited in the solver [1].

MaxHS utilizes MiniSat v2.2 as its SAT solver and IBM's CPLEX v12.8 as its IP solver. Interestingly experiments with more sophisticated SAT solvers like Glucose http://www.labri.fr/perso/lsimon/glucose/ and Lingeling http://fmv.jku.at/lingeling/ yielded inferior performance. This indicates that the SAT problems being solved are quite simple, too simple for the more sophisticated techniques used in these SAT solvers to pay off. In fact, simpler SAT problems are one of the original motivations behind MaxHS [4].

Nevertheless, some changes have been made to the underlying MiniSat solver. In particular, this year the MiniSat code base was updated to maintain LBD clause scores and to perform clause deletion using LBD scores in a manner identical to the Glucose solver. However, this is the only part of the Glucose that was added.

The 2020 version of MaxHS saw a large change to implement abstract cores [2]. The notion of abstract cores is described in the cited paper; their benefit is that the solver can generate a single abstract core that represents an exponential number of non-abstract cores. With abstract cores the IHS approach avoids having to generate an exponential number of cores in some cases [6].

Also a version for the Top-K track was submitted.

A summary of the main features of MaxHS that extend the basic IHS approach are listed below. Most of these features are unchanged from 2019.

1.0.1. Abstract Cores. In MaxHS cores are sets of soft clauses that jointly cannot be satisfied in any feasible solution (i.e., a solution satisfying the hard clauses). A cores is represented in the solver by a clause containing the soft clause's blocking variables. The IHS approach is to use a SAT solver to collect cores and an IP solver to compute a minimum cost set of soft clauses that can satisfy all of the core constraints (i.e., each core specifies the constraint that at least one soft clause in each core must be falsified). MaxHS keep track of the computed cores and uses the Louvain clustering algorithm [3] to find clusters of soft clauses that frequently appear together in cores. These clusters of soft clauses are then grouped together as inputs to a totalizer. Now instead of generating cores by assuming individual soft clauses to be true, totalizer outputs are assumed that limit the number of soft clauses in a cluster that can be falsified. Thus the specific identity of the soft clauses to be falsified is ignored—all that matters is how many are falsified. MaxHS only clusters the soft clauses when it is failing to make progress. Furthermore, it continues to generate ordinary non-abstract cores as well as abstract cores.

1.0.2. Top-K. In the Top-K track the solver is to return a sequence of non-decreasing cost optimal solutions each of which falsifies a different set of soft clauses. To achieve this the existing code base was modified—later these changes can be added as an option. The following changes were required

- A preprocessing time pure literal detection was removed. Literals that appear in only one polarity in both the hard *and the soft* clauses of the input formula can be set to true without affecting the optimal solution. However, subsequent top-k solutions might require negating a pure literal. So pure literal reduction was turned off in the top-k version.
- Boolean Lexicographic Optimization (BLO) was turned off. This also is a preprocessing step. Let S be the set of soft clauses and H be the set of hard clauses of the instance. In BLO if a set $B \subseteq S$ of soft clauses can be found such that (a) $B \cup H$ is satisfiable and (b) the sum of weights of the remain soft clauses (i.e., those in $S - B$) is less than the minimum weight soft clause in B, then all clauses in B can be made hard. That is, BLO detects the case where all of the clauses in B can be satisfied and it is better to falsify all clauses not in B rather than falsify a single clause in B. MaxHS performs BLO hardening of soft clauses as a preprocessing step, but for top-k this was turned off.

- LP reduced cost hardening [1] used in MaxHS was turned off. This technique relies on an upper bound on the solution cost to harden soft clauses, however with top-k when looking for the next solution the upper bound can increase making such hardening invalid.

Other than these changes the implementation of top-k is simple: simply add a blocking clause computed from the previous found solution and then reoptimize with this added constraint. This blocking clause specifies that at least one of the previously satisfied soft clauses must now be falsified, forcing the solver to find an new optimal solution falsifying a different set of soft clauses from before as required. (The new solution is an optimal solution of the now more constrained formula, it is not usually an optimal solution of the original formula.)

1.0.3. Termination based on Bounding. MaxHS maintains an upper bound (and best model found so far) and a lower bound on the cost of an optimal solution (the IP solver computes valid lower bounds). MaxHS terminates when the gap between the lower bound and upper bound is low enough (with integer weights when this gap is less than 1, the upper bound model is optimal). This means that MaxHS no longer needs to wait until the IP solver returns an hitting set whose removal from the set of soft clauses yields SAT; it can return when the IP solver's best lower bound is close enough to show that the best model is optimal.

1.0.4. Early Termination of Cplex. In early versions of MaxHS, the IP solver was run to completion forcing it to find an optimal solution every time it is called. However, with bounding, optimal solutions are not always needed. In particular, if the IP solver finds a feasible solution whose cost is better than the current best model it can return that: either the IP solution is feasible for the MaxSat problem, in which case we can lower the upper bound, or it is infeasible in which case we can obtain additional cores to augment the IP model (and thus increase the lower bound). Terminating the IP solver before optimization is complete can yield significant time savings.

1.0.5. Reduced Cost fixing via the LP-Relaxation. Using an LP relaxation and the reduced costs associated with the optimal LP solution, some soft clauses can be hardened or immediately falsified. See [1] for more details.

1.0.6. Mutually Exclusive Soft Clauses. Sets of soft clauses of which at most one can be falsified or at most one can be satisfied are detected. Sets where at most one soft clause can be satisfied are re-encoded to a new MaxSat problem with one soft clause to represent condition that one of these previous soft clauses is satisfied. With abstract cores it was found that sets where at most one soft clause can be falsified were no longer helpful, and these are not used in this version.

1.0.7. Other clauses to the IP Solver. Problems with a small number of variables are given entirely to the IP solver, so that it directly solves the MaxSat problem. In this case the SAT solver is used to first compute some additional clauses and cores, and to find a better initial model for the IP solver. This additional information from the SAT solver often makes the IP solver much faster than just running the IP solver and represents an alternate way of hybridizing SAT and IP solvers.

1.0.8. Other techniques for finding Cores. MaxHS iteratively calls the IP solver to obtain a hitting set of the cores computed so far. If that hitting set does not yield an optimal MaxSat solution then more cores must be added to the IP solver. In some of these iterations very few cores can be found causing only a slight improvement to the IP solver's model. This results in a large number of time consuming calls to the IP solver. Two method were developed to aid this situation (a) we ask the IP solver for more solutions and generate cores from these as hitting sets as well and (b) if we have a new upper bound model we try to improve this model by converting it to a minimal correction set (MCS). In converting the upper bound model to an MCS we either find a better model (lowering the upper bound) or we compute additional conflicts that can be added to the IP solver.

References

[1] Bacchus, F., Hyttinen, A., Järvisalo, M., Saikko, P.: Reduced cost fixing in maxsat. In: Beck, J.C. (ed.) Principles and Practice of Constraint Programming - 23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10416, pp. 641–651. Springer (2017), https://doi.org/10.1007/978-3-319-66158-2_41

[2] Berg, J., Bacchus, F., Poole, A.: Abstract Cores in Implicit Hitting Set MaxSat solving. In: Theory and Applications of Satisfiability Testing - SAT 2020 - 23rd International Conference, SAT 2019 Proceedings. Lecture Notes in Computer Science, Springer (2020), in press

[3] Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment 2008(10), P10008 (Oct 2008), https://doi.org/10.1088%2F1742-5468%2F2008%2F10%2Fp10008

[4] Davies, J., Bacchus, F.: Solving MAXSAT by solving a sequence of simpler SAT instances. In: Proc. CP. Lecture Notes in Computer Science, vol. 6876, pp. 225–239. Springer (2011)

[5] Davies, J., Bacchus, F.: Exploiting the power of MIP solvers in MaxSAT. In: Proc. SAT. Lecture Notes in Computer Science, vol. 7962, pp. 166–181. Springer (2013)

[6] Davies, J.: Solving MAXSAT by Decoupling Optimization and Satisfaction. Ph.D. thesis, University of Toronto (2013), http://www.cs.toronto.edu/~jdavies/Davies_Jessica_E_201311_PhD_thesis.pdf

[7] Davies, J., Bacchus, F.: Postponing optimization to speed up MAXSAT solving. In: Proc. CP. Lecture Notes in Computer Science, vol. 8124, pp. 247–262. Springer (2013)

[8] Saikko, P., Berg, J., Järvisalo, M.: LMHS: A SAT-IP hybrid MaxSAT solver. In: Proc. SAT. Lecture Notes in Computer Science, vol. 9710, pp. 539–546. Springer (2016)

[9] Saikko, P.: Re-implementing and Extending a Hybrid SAT-IP Approach to Maximum Satisfiability. Master's thesis, University of Helsinki (2015), http://hdl.handle.net/10138/159186