sls-mcs and sls-lsu: Description

Andreia P. Guerreiro[1], Miguel Terra-Neves[1,2], Inês Lynce[1], José Rui Figueira[3], and Vasco Manquinho[1]

[1]INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal
{andreia, ines, vmm}@sat.inesc-id.pt
[2]OutSystems, Portugal
miguel.neves@outsystems.com
[3]CEG-IST, Instituto Superior Técnico, Universidade de Lisboa, Portugal
figueira@tecnico.ulisboa.pt

1 Introduction

We developed improved versions of `sls-mcs` and `sls-lsu`, two solvers that integrate SAT-based techniques in a Stochastic Local Search (SLS) solver for MaxSAT. In these solvers, the control of the solving process changes from SAT-based procedures to stochastic procedures and vice-versa. At each step, each procedure tries to build upon the information received from the other, instead of being independent procedures. The idea is to use the SLS solver as the main procedure and, occasionally, use an unsatisfiability-based algorithm to correct the SLS current (unsatisfiable) assignment into a satisfiable one, and use a procedure based on Minimal Correction Subset (MCS) enumeration or on the Linear Sat-Unsat (LSU) algorithm to improve the current solution. We submitted new versions of `sls-mcs` and `sls-lsu` for the unweighted incomplete MaxSAT track, and two new versions of `sls-mcs` for the weighted incomplete MaxSAT track.

2 Using SAT Techniques in Local Search

One of the shortcomings of SLS algorithms is that these solvers have difficulties in dealing with highly constrained formulas. Therefore, it might be the case that the SLS algorithm is unable to satisfy the set of hard clauses or gets stuck in some local minima. In these cases, using SAT-based techniques to find a satisfiable assignment would be beneficial.

2.1 Assignment Correction

Consider the case when the SLS algorithm is unable to change from an unsatisfiable assignment ν into a better assignment. Our solver performs a correction to ν in order to guide the SLS algorithm to the feasible region of the search space. First, we start by building a set of assumption literals corresponding to the assignment ν. Next, a SAT call on the set of hard clauses, ϕ_h, is made. Clearly, if ν is not feasible, then this call returns `UNSAT` and returns an unsatisfiable core. The assumption literals that occur in such an unsatisfiable core are removed from the set of assumptions, and a new SAT call is made. The same procedure is repeated until a satisfiable assignment is found.

A conflict limit is defined for the correction procedure. If the conflict budget is not enough to find a satisfiable assignment, then our algorithm applies a similar procedure with a more aggressive strategy where at each iteration 50% of the literals in the set of assumptions are removed. Since the correction procedure only depends on the hard clauses, there is no guarantee regarding its quality. As a result, we also apply a SAT-based improvement procedure.

2.2 Assignment Improvement

Given a MaxSAT instance ϕ, a set of assumptions $\mathcal{A}$, a satisfiable assignment ν, and conflict budget, the goal of this assignment improvement algorithm is to find a better quality solution for ϕ through an MCS enumeration procedure.

The algorithm starts by building a working formula from the set of hard clauses ϕ_h and the set of assumptions $\mathcal{A}$. Next, the algorithm iterates over all MCSes of ϕ, constrained to the set of assumptions $\mathcal{A}$ and returns the best assignment found. Each time a new MCS is found, a blocking clause is added to prevent the enumeration of the same MCS later on. The algorithm returns the best solution found before the conflict budget runs out. Note that the set of literals $\mathcal{A}$ restricts the MCS enumeration procedure. This results in a localized MCS enumeration.

Many different improvement procedures can be devised, including the usage of complete methods. For example, an alternative is to replace the MCS enumeration algorithm by a call to a Linear Sat-

Unsat algorithm (LSU). The call to the LSU algorithm is also limited to a number of conflicts, and all literals in $\mathcal{A}$ are forced to be satisfied. Hence, the LSU call is also restricted to a localized region of the search space.

3 Incomplete Track

We developed the solvers `sls-mcs` and `sls-lsu`, that integrate an SLS algorithm with the assignment correction and assignment improvement procedures [1], which we submitted to MaxSAT Evaluation 2019. As SATLike [3], an SLS algorithm, was one of the best performing solvers in the incomplete solver track in the MaxSAT Evaluation 2018, we used SATLike in our implementation [1]. We developed and submitted improved versions of `sls-mcs` and `sls-lsu`. The main difference to the versions proposed in [1] and submitted to MaxSAT Evaluation 2019 is in the first call to the assignment correction algorithm. If no satisfiable assignment was yet found by the SLS algorithm, then the set of assumptions $\mathcal{A}$ is empty in the first SAT call.

3.1 Unweighted Instances

Two solvers were submitted for the unweighted incomplete track: `sls-mcs` and `sls-lsu`. In `sls-mcs`, the `SATLike` solver[1] is extended with the assignment correction algorithm and the assignment improvement algorithm based on MCS enumeration. The difference from `sls-mcs` to `sls-lsu` is on the assignment improvement algorithm. In `sls-lsu`, the linear sat-unsat assignment improvement algorithm is used.

Both `sls-mcs` and `sls-lsu` use the Glucose SAT solver (version 4.1) on the assignment correction procedure. Moreover, the CLD [4] algorithm is used as the MCS algorithm in `sls-mcs`. The linear sat-unsat algorithm used in `sls-lsu` is an adapted version of the one available at the `open-wbo` open source MaxSAT solver. The conflict limits of the correction and the improvement algorithms were set to 10^5. In both `sls-mcs` and `sls-lsu`, the assignment correction/improvement algorithm is called when SATlike has reached half of the maximum number of iterations without improvement. In such a case, the correction algorithm is called if the current assignment ν does not satisfy all hard clauses, otherwise the improvement algorithm is directly called with approximately half of the literals in the current assignment ν as assumptions. These assumption literals are randomly chosen from ν.

[1]The source code of `SATLike` is publicly available at the 2018 MaxSAT evaluation `https://maxsat-evaluations.github.io/2018/descriptions.html`

3.2 Weighted Instances

Two versions of the solver were submitted for the weighted incomplete track: `sls-mcs` and `sls-mcs2`. In both versions, the stratified CLD algorithm [5] is used as the MCS algorithm. Unlike `sls-mcs`, `sls-mcs2` does not consider the assumptions $\mathcal{A}$ as hard clauses in the MCS enumeration procedure.

Acknowledgments

We would like to thank Zhendong Lei and Shaowei Cai for making the code of SATLike available online. This work was supported by national funds through FCT under projects UIDB/50021/2020 and PTDC/CCI-COM/31198/2017.

References

[1] Guerreiro, A.P., Terra-Neves, M., Lynce, I., Figueira, J.R., Manquinho, V.: Constraint-based techniques in stochastic local search maxsat solving. In: Schiex, T., de Givry, S. (eds.) Principles and Practice of Constraint Programming. pp. 232–250. Springer International Publishing (2019)

[2] Lang, J. (ed.): Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden. ijcai.org (2018)

[3] Lei, Z., Cai, S.: Solving (weighted) partial maxsat by dynamic local search for SAT. In: Lang [2], pp. 1346–1352

[4] Marques-Silva, J., Heras, F., Janota, M., Previti, A., Belov, A.: On Computing Minimal Correction Subsets. In: International Joint Conference on Artificial Intelligence. pp. 615–622 (2013)

[5] Terra-Neves, M., Lynce, I., Manquinho, V.M.: Stratification for constraint-based multi-objective combinatorial optimization. In: Lang [2], pp. 1376–1382