SMAX – Implementing a Robust MaxSAT Interface

Norbert Manthey
nmanthey@conp-solutions.com
Dresden, Germany

I. Introduction

The main aim of the MaxSAT solver SMAX is to showcase how a MaxSAT solver can be used as a library. Consequently, it can be understood as a prototype implementation which does not focus on solver performance as the highest priority. The solver has to following properties:

- implement a C++interface as a library
- be easily consumable, by being available as MIT license
- implement in a robust, modular way
- support reproducible partial solving, based on solving steps.

The implementation allows to provide the solver as a shared library, which opens the door for integration into tools that require a Python or Java interface. Consequently, solver failures and exceptions are caught by the library itself, and errors are handled accordingly. All code is version controlled with git submodules, to allow reproduction of solver binaries or libraries. Continuous testing is used to make sure that updated code can be used as solving backend via a shared library, and furthermore checks for errors via valgrind. Finally, a Maxsat fuzzer[1] is used to test whether random input behaves as expected.

II. Solver Internals

The idea behind the solver is open to switch solving backends. The initial implementation follows an approach that makes software licenses simple, by focussing on software that is available as MIT license.

A. Solver Interface

The solver implements a C++interface to a MaxSAT engine. The interface is sketched in Figure 1. The interface aims at allowing failure inside the solver, as well as providing an assignment to start from and a number of steps to perform, so that partial and reproducible solver calls are possible. Discussions on the interface are welcome.

B. MaxSAT Solvers

The solver uses OPEN-WBO [1] as the current MaxSAT backend, which is an open source MaxSAT solver that supports several MaxSAT algorithms and SAT solvers [2], [3], [4]. To make sure all used code is available under MIT license, the GLUCOSE 4.1 code of the OPEN-WBO package had to be removed.

[1]The fuzzer is available at https://github.com/conp-solutions/maxsat-fuzzer.git.

The used OPEN-WBO version is an older version, as initial changes have been added early and rebasing them to a current version has not been considered yet. Consequently, eventual bug fixes might be currently missing in SMAX. Furthermore, additional patches had to be added to OPEN-WBO to allow pragmatic access to internal data structures like the found bound or the last model to be handled internally. Some part of SMAX basically provides an input parser and an output generator based on the available data structures of OPEN-WBO, with the additional abstraction layer in between.

We currently use a single, predefined, configuration of OPEN-WBO to simplify the interface to the MaxSAT library.

C. SAT Solvers

OPEN-WBO is based on the data structures of MINISAT 2.2 [2], [5]. Therefore, solvers based on MINISAT 2.2 can be used as a potential back-end solvers.

MERGESAT [4] is a CDCL solver based on the SAT competition winner of 2018, MAPLELCMDISTCHRONOBT [6]. While being based on MINISAT 2.2, MERGESAT aims at providing recent solving techniques while being compatible with the solver API of MINISAT 2.2. Among others, MERGESAT implements partial backtracking, an efficient version of learned clause minimization, as well as inprocessing based on subsumption and self-subsuming resolution. Compared to MAPLELCMDISTCHRONOBT, the incremental search feature has to been enabled again.

D. Competition Tracks

The solver has been submitted to the complete tracks of the competition only, although the implementation supports partial solving. However, the current implementation does not support to forward a signal from the solver wrapper into the solver to obtain the currently best known solution. Furthermore, support for Top-K has not been integrated yet, also because the used OPEN-WBO version does not support this feature.

Two variants of the solver have been submitted. The only difference is the SAT backend, namely MINISAT 2.2 and MERGESAT. The reason to submit these two solvers is to show the performance difference when using a more recent SAT backend.

III. Availability

The solver SMAX is available under a MIT license in GitHub at https://github.com/conp-solutions/smax. The repository is setup to briefly check new changes with continuous integration.

```
class MaxSATSolver

    /** Return codes for the caller to compute a MaxSAT solution */
    enum ReturnCode

        UNKNOWN = 0,
        SATISFIABLE = 1,
        UNSATISFIABLE = 2,
        OPTIMAL = 3,
        ERROR = 4,
    ;

    /** This integer represents the version of the MaxSAT interface */
    unsigned getVersion () const;

    /** This string contains the name of the used backend */
    const char* getSolverName () const;

    /** Initialize the MaxSAT solver for a given formula */
    MaxSATSolver(int nVars, int nClausesEstimate = 8192);

    /** A call to this method frees all resources of the solver. */
    ~MaxSATSolver();

    /** Return error number code in case the last call to another method failed */
    int getErrno() const;

    /** Add a clause to the solver */
    bool addClause(const std::vector<int> &literals, uint64_t weight = 0);

    /** Add an at-most-k constraint to the solver */
    bool addAtMostK(const std::vector<int> &literals, const unsigned k);

    /** Compute a MaxSAT solution for the added (weighted) formula */
    ReturnCode compute_maxsat(std::vector<int> &model,
                              uint64_t &cost,
                              uint64_t maxCost = UINT64_MAX,
                              const std::vector<int> *startAssignment = 0,
                              int64_t maxMinimizeSteps = -1);
```

Fig. 1. This figure briefly summaries the interface that is offered to the MaxSAT solver backend implementation. A well documented version of this file can be found at https://github.com/conp-solutions/smax/blob/master/include/MaxSATSolver.h.

Acknowledgments

The current solver uses Open-WBO as a backend engine, and is heavily based on that solver. Hence, the contributors of that solver own a large part of this tool: Ruben Martins, as well as Miguel Terra-Neves, Vasco Manquinho and Inês Lynce. Additionally, we would like to thank all the collaborators on previous versions of Open-WBO, namely Saurabh Joshi and Mikoláš Janota.

References

[1] R. Martins, V. Manquinho, and I. Lynce, “Open-WBO: a Modular MaxSAT Solver,” in *SAT*, ser. LNCS, vol. 8561. Springer, 2014, pp. 438–445.

[2] N. Eén and N. Sörensson, “An Extensible SAT-solver,” in *SAT*. Springer, 2003, pp. 502–518.

[3] G. Audemard and L. Simon, “Predicting Learnt Clauses Quality in Modern SAT Solvers,” in *IJCAI*, 2009, pp. 399–404.

[4] N. Manthey, “Mergesat,” in *Proceedings of SAT Competition 2019: Solver and Benchmark Descriptions*, 2019.

[5] N. Sörensson, N. Een, and N. Manthey. (2018, May) GitHub repository for MiniSat. https://github.com/conp-solutions/minisat.

[6] A. Nadel and V. Ryvchin, “Chronological backtracking,” in *SAT*. Springer, 2018, pp. 111–121.