OPTSAT: A Tool for Solving SAT related Optimization Problems

Enrico Giunchiglia and Marco Maratea

STAR-Lab, DIST, University of Genova
viale Francesco Causa, 13 — 16145 Genova (Italy)
{enrico,marco}@dist.unige.it

1 Introduction

Propositional satisfiability (SAT) is one of the most important and central problems in Artificial Intelligence and Computer Science. Basically, most SAT solvers are based on the well-known Davis-Logemann-Loveland (DLL) procedure. DLL is a decision procedure: given a SAT formula ϕ, it can decide if ϕ is satisfiable (and it can return a satisfying assignment μ), or not. Often, this is not sufficient, in that we would like μ to be also "optimal", i.e., that μ has also to minimize/maximize a given objective function. MAX-SAT, MIN-ONE, DISTANCE-SAT and their weighted versions are popular optimization problems. (In the following, ϕ is the input formula expressed as a set of clauses). Almost all the systems that can deal with these problems follow a classical branch&bound schema: whenever a satisfying assignment μ for ϕ with a cost c_μ is found, the search goes on looking for another satisfying assignment with a lower (or higher, depending on the problem) cost.

In this paper, we present OPTSAT (OPTimal SATisfiability), a tool for solving SAT related optimization problems based on the DLL algorithm. Here, for simplicity, we focus on MAX-SAT and MIN-ONE problems. MAX-SAT is the problem of finding an assignment (i.e., a consistent set of literals) satisfying as many clauses in ϕ as possible; MIN-ONE is the problem of determining an assignment satisfying ϕ and with as few as possible variables assigned to TRUE. MIN-SAT and MAX-ONE are defined analogously. Differently from other systems, OPTSAT does not follow a branch&bound schema, but it solves these optimization problems by imposing a partial ordering on the literals to branch on. MAX-SAT, MIN-ONE but also DISTANCE-SAT and other SAT-related optimization problems can be solved in this way. Moreover, OPTSAT is not limited, like all the other systems, to the computation of assignments which are optimal with respect to a given numeric function. OPTSAT can also solve MAX-SAT$_\subseteq$ and MIN-ONE$_\subseteq$ problems. MAX-SAT$_\subseteq$ is the problem of finding an assignment satisfying a set S of clauses and such that there is no assignment satisfying a set S' of clauses with $S \subset S' \subseteq \phi$. MIN-ONE$_\subseteq$ is defined analogously. Any solution which is "MAX-SAT"-optimal, is also "MAX-SAT$_\subseteq$"-optimal: however, in many application domains it may be sufficient to have a "MAX-SAT$_\subseteq$"-optimal solution, and this can be much easier. In the following, optimality is defined under *cardinality* (resp. *subset inclusion*) if we are considering a MAX-SAT/MIN-ONE (resp. MAX-SAT$_\subseteq$/MIN-ONE$_\subseteq$) or analogous problems.

2 The OPTSAT system

OPTSAT algorithm is described in details in [GM06]. Here we highlight its main points, and we present its input format, the newly implemented encodings, and report about the results obtained with them and with the integration of MINISAT.

The input format. OPTSAT input format is an extension of the well-known DIMACS format for SAT: in the comment lines (the ones starting with "c") there are all the informations that OPTSAT uses to solve the problem, i.e., the type of the problem (if it is a MAX-SAT, MIN-SAT, MAX-ONE, or MIN-ONE problem: this is specified using two flags max/min and SAT/ONE, see Example 1); and the type of optimality considered, if under cardinality or subset inclusion. The line starting with "p" is the usual problem line for DIMACS.

Example 1. We specify a MIN-ONE$_\subseteq$ problem with the header

```
c min ONE
c subset
p cnf n m
```

The parsing phase. Consider ϕ having n variables and m clauses. In this phase, the input file is parsed, and some operations are performed in order to define the SAT formula ϕ' that will be fed to the back-end SAT solver. First, if the problem is MAX-SAT or MIN-SAT, following a well-known method, ϕ is modified as follow: each clause $C_i \in \phi$ is replaced by C'_i defined as $\{\neg s_i \cup C_i\}$, where s_i is called *clause selector* and is a newly introduced variable. Second, if optimality is by cardinality, we compute a formula encoding the objective function. We call this function adder(S), where S is defined depending on the problem we are facing, i.e. (*i*) in the case of a MIN-ONE or MAX-ONE problem, S is the set of variables in ϕ or, (*ii*) in the case of a MIN-SAT or MAX-SAT problem, S is the set $\{s_1, \ldots, s_m\}$ of clause selectors. The goal of adder(S) is to define a sequence $b_v, \ldots, b_0$ of new variables such that for any assignment μ of the extended signature, the value of the objective function when considering the assignment μ corresponds to an assignment of $b_v, \ldots, b_0$. adder(S) can be realized, in polynomial time, in many ways. In OPTSAT, adder(S) is implemented in the following ways:

1. Bailleux/Boufkhad (BB) [BB03]. In this encoding a unary representation of integers is used: an integer x s.t. $0 \leq x \leq n$ is represented using n propositional variables $\{x_1, \ldots, x_n\}$ with (the first) "x" variables assigned to 1 (TRUE), and the others to 0 (FALSE). This representation has the property that when a variable x_j has value TRUE, all the variables x_k with $1 \leq k < j$, are TRUE as well; and similarly if x_k has value FALSE. The encoding is efficient wrt unit-propagation but it adds a quadratic number of new clauses.
2. BB$_{mod}$ (BB modified): we modified the BB encoding, in order to enforce that when b_i is assigned to FALSE (resp. TRUE), also b_{i+1} (resp. b_{i-1}) is assigned to FALSE (resp. TRUE) by unit propagation.
3. Warners [War98]. Here is used a binary representation of integers, e.g., the value x of the objective function is represented as $x = \Sigma_{i=0}^{M} 2^i x_i$, where x_i are again propositional variables, and $M = \lfloor log_2(x) \rfloor$ for $x > 0$, and $M = 0$ otherwise. This is a linear time and space encoding, that relies on sums via adder circuits and, as presented in the paper, works directly with objective functions with weights. In OPTSAT, the encoding is optimized for the non weighted case, and the size of the encoding is approximately halved.

The first two encodings are new for OPTSAT.

Solving algorithm. As we already said, OPTSAT is a modification of the DLL algorithm: it takes as input the SAT formula $\phi' = \phi \cup \mathsf{adder}(S)$ (with ϕ considered here after the introduction of clause selectors in case of MIN-SAT or MAX-SAT problem, and $\mathsf{adder}(\mathrm{S}) = \emptyset$ if the optimality is under subset inclusion), an assignment μ (initially set to $\emptyset$), and a partial order on the set of literals. The main change that has to be made to the DLL algorithm is in the heuristic.

Consider first the case of an optimization under cardinality problem in which the sequence $b_v, \ldots, b_0$ encode the value of the objective function, b_v being the "most significant" variable. Then, the heuristic returns (i) the first not yet assigned atom b_i (i.e., the variable with the highest index i), if any, or an arbitrary variable in ϕ' (according with the heuristic of the solver); (ii) if a variable in $\{b_v, \ldots, b_0\}$ is chosen, the variable is assigned to TRUE in the case of MAX-SAT or MAX-ONE problems, and to FALSE otherwise; if the variable is chosen by the heuristic of the solver, it is left to the solver the decision about how to assign it.

In the case of an optimization under subset inclusion problem, it returns an un-assigned atom in S, if any, and assign it to FALSE in the case of MIN-SAT$_\subseteq$ or MIN-ONE$_\subseteq$ problems, and to TRUE in the other cases; or an arbitrary atom.

OPTSAT returns an optimal solution if one exists, or that no solution exists otherwise. In order to see why this is the case, observe that, in the case of minimality under cardinality, variables are preferentially and in order chosen from b_v to b_0, while in the case of minimality for subset inclusions, atom in S are chosen. Only when all these variables are assigned, the choice is delegated to the heuristic of the underlying solver. Thus, considering, e.g., a MIN-ONE or MIN-SAT problem, the algorithm first explores (assuming no literal in $\{b_v, \ldots, b_0\}$ are assigned by unit) the branches with $b_v, \ldots, b_0$ assigned by FALSE; if all such branches fail, then it explores the branches with $\{b_v, \ldots, b_1\}$ assigned to FALSE and b_0 to TRUE; if also these branches fail b_0 and b_1 are flipped and the search goes on until a satisfying assignment μ is found, or the entire search space has been explored.

One of the main property of the algorithm is that, when the first satisfying assignment is found, we are guaranteed that it is also "optimal".

DISTANCE-SAT and problems with weights. DISTANCE-SAT(μ) [BM06] is the problem of determining an assignment which satisfies the input formula and differing in as few as possible literals from μ. DISTANCE-SAT$_\subseteq$ is defined in the obvious way. In the weighted version of all the problems we presented, the objective function to minimize is linear function of the variables. All these problems –but also others– can be solved by OPTSAT, as shown in [GM06].

3 Experimental results

In OPTSAT, we can now choose between ZCHAFF ver. of 5.13.2004[1] and MINISAT ver. 1.14[2] to be used as back-end: each of them has been modified accordingly to the consideration made in Section 2. These are the winners of the last two SAT competitions [LS05,LS06] in the industrial categories (MINISAT together with the SAT/CNF minimizer SATELITE).

---

[1] `http://www.princeton.edu/~chaff/`

[2] `http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat/`

In [GM06], we showed that OPTSAT is highly competitive on MAX-SAT/MAX-SAT$_\subseteq$ and MIN-ONE/MIN-ONE$_\subseteq$ problems if compared with a variety of solvers, both tailored for a specific optimization problem, and with the solvers that showed the best performances in the PB evaluation [MR06]. To be also noticed that OPTSAT does not use any "problem-dependent" optimization, like the computation of an upper-bound of the optimal solution, using incomplete SAT solvers, performed by most of the MAX-SAT solvers. Here we extend the results in [GM06], by report about the results obtained with the new encodings and with the integration of MINISAT in our system. For lack of space we do not put any table here: the results can be found as an appendix at the system home page reported below.

In general, and as expected, because no clauses need to be added, finding a solution which is optimal under subset inclusion is easier than finding an optimal solution under cardinality. Considering the CPU times, between the BB based encodings, BB$_{mod}$ almost always is faster or competitive with BB, and in general very competitive for objective functions having a relatively low number of variables. But, when this number is high, it incurs in memory out. The use of MINISAT generally helps in reducing the time to solve a problem. The reduction is dramatic, up to 3 orders of magnitude, when considering MAX-SAT$_\subseteq$ problems. This highlights one of the main features of our approach, i.e., the possibility of levering on the enhancement that are continuously made in SAT. For this reason, we expect our system to further improve its performances thanks to the upcoming SAT race and future competitions.

Availability of the system. The binary of the system, along with benchmarks in the OPTSAT input format and a parser to the format of the PB evaluation, are available at: `http://www.star.dist.unige.it/~marco/optsat/`.

References

[BB03] Olivier Bailleux and Yacine Boufkhad. Efficient cnf encoding of boolean cardinality constraints. In *9th International Conference on Principles and Practice of Constraint Programming (CP-03)*, LNCS. Springer, 2003.

[BM06] O. Bailleux and P. Marquis. Some computational aspects of DISTANCE-SAT. *Journal of Automated Reasoning (JAR), To appear*, 2006.

[GM06] E. Giunchiglia and M. Maratea. Solving optimization problems with DLL. Accepted to ECAI 2006. Available at `http://www.star.dist.unige.it/~marco/Data/06ecai.pdf.gz`, 2006.

[LS05] D. LeBerre and L. Simon. Fifty-five solvers in vancouver: The SAT 2004 competition. In *8th International Conference on Theory an Applications of Satisfiability Testing. Selected Revised Papers.*, LNCS 3542. Springer, 2005.

[LS06] D. LeBerre and L. Simon. Preface to the special volume on the sat 2005 competitions and evaluations. *Journal of Satisfiability, Boolean Modeling and Computation (JSAT)*, 2006.

[MR06] V. M. Manquinho and O. Roussel. The first evaluation of pseudo-boolean solvers (PB05). *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)*, 2:103–143, 2006.

[War98] J. P. Warners. A linear-time transformation of linear inequalities into CNF. *Information Processing Letters*, 68(2):63–69, 1998.