On Applying Unit Propagation-Based Lower Bounds in Pseudo-Boolean Optimization*

Federico Heras
Universitat Politecnica de Catalunya
Jordi Girona 1,3
Barcelona, Spain
fheras@lsi.upc.edu

Vasco Manquinho
Technical University of Lisbon
IST/INESC-ID
Lisbon, Portugal
vasco.manquinho@inesc-id.pt

Joao Marques-Silva
School of Electronics and Computer Science
University of Southampton
Southampton, United Kingdom
jpms@ecs.soton.ac.uk

Abstract

Unit propagation-based (UP) lower bounds are used in the vast majority of current Max-SAT solvers. However, lower bounds based on UP have seldom been applied in Pseudo-Boolean Optimization (PBO) algorithms derived from the DPLL procedure for Propositional Satisfiability (SAT). This paper enhances a DPLL-style PBO algorithm with an UP lower bound, and establishes conditions that enable constraint learning and non-chronological backtracking in the presence of conflicts involving constraints generated by the UP lower bound. From a theorical point of view, the paper highlights the relationship between the recent UP lower bound and the well-known Maximum Independent Set (MIS) lower bound. Finally, the paper provides preliminary results that show the effectiveness of the proposed approach for representative sets of instances.

Introduction

The algorithmic improvements made to Boolean Satisfiability (SAT) solvers over the last decade (Marques-Silva & Sakallah 1996; Moskewicz *et al.* 2001; Eén & Sörensson 2003) motivated research work in a number of extensions of SAT, including Pseudo-Boolean Optimization (PBO) (Aloul *et al.* 2002; Chai & Kuehlmann 2003; Sheini & Sakallah 2006; Eén & Sörensson 2006) and Max-SAT (Li, Manyà, & Planes 2005; Heras, Larrosa, & Oliveras 2007).

PBO solvers have improved significantly over the last few years with the extensive use of the most effective SAT techniques, commonly used in modern SAT solvers (Marques-Silva & Sakallah 1996; Moskewicz *et al.* 2001; Eén & Sörensson 2003). New practical applications are found for PBO solvers every year and solvers are the subject of a regular evaluation (Manquinho & Roussel 2007). Somewhat independently of the work in PBO, extensive research work has been carried out in Max-SAT, with significant improvements being reported in the last few years. Max-SAT solvers also find a number of strategic applications, and they are also subject to a regular evaluation (Argelich *et al.* 2007). Motivated by the specificity of the problem, algorithms for Max-SAT have evolved somewhat differently from algorithms for SAT and PBO. Max-SAT algorithms are often based on branch and bound search, and employ sophisticated techniques for computing lower bounds. Despite the improvements made in these two areas of research, techniques used in PBO have seldom been used in Max-SAT, and vice-versa.

This paper is a first attempt at integrating techniques from these two research areas. As a result, the paper proposes to integrate Unit Propagation-based (UP) lower bounds, often used in Max-SAT (Li, Manyà, & Planes 2005), with constraint learning and non-chronological backtracking, two techniques widely used in SAT and PBO (Marques-Silva & Sakallah 1996). More concretely, the paper describes how to augment SAT-based PBO algorithms with UP lower bounding capabilities associated with information obtained from the Pseudo-Boolean (PB) constraints and from the cost function. Moreover, the paper establishes conditions for learning new constraints from conflicts associated with the UP lower bound. Finally, the paper shows that these new constraints can be used for performing non-chronological backtracking. From a more theoretical point of view, the paper studies the relationship between the UP lower bound and the well-known Maximum Independent Set (MIS) lower bound (Coudert 1996). Experimental results on several sets of problem instances coming from the Pseudo-Boolean Evaluation 2007 (Manquinho & Roussel 2007) illustrate the effectiveness of the proposed techniques.

The paper is organized as follows. The next section introduces the notation and definitions used throughout the paper. Afterwards, algorithms for PBO are briefly surveyed. Next, the use of UP lower bounds in PBO is detailed, followed by experimental results on representative problem instances. Finally, the paper concludes and suggests directions for further research work.

Preliminaries

This section introduces the notations and definitions for SAT, Max-SAT and PBO, used in the remainder of the paper.

A *propositional formula* φ in Conjunctive Normal Form (CNF) denotes a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$. The formula φ consists of a conjunction of propositional clauses, where each clause ω is a disjunction of literals, and

*This work is partially supported by FCT grants POSC/EIA/61852/2004 and PTDC/EIA/76572/2006, by EU grants IST/033709 and ICT/217069, and by EPSRC grant EP/E012973/1.

a literal l is either a variable x_j or its complement $\bar{x}_j$. If a literal assumes value 1, then the clause is satisfied. If all literals of a clause assume value 0, the clause is unsatisfied. Clauses with only one unassigned literal are referred to as unit. Finally, clauses with more than one unassigned literal are said to be unresolved. In a search procedure, a conflict is said to be identified when at least one clause is unsatisfied. In addition, observe that a clause $\omega = (l_1 + \ldots + l_k)$, $k \leq n$, can be interpreted as a linear inequality $l_1 + \ldots + l_k \geq 1$, and the complement of a variable x_j, $\bar{x}_j$, can be represented by $1 - x_j$.

When a clause is unit an *assignment* can be implied. The variable associated with the only non-assigned literal needs to be assigned in such a way that the clause is satisfied. These logical implications correspond to the application of the unit clause rule (Davis, Logemann, & Loveland 1962) and the process of repeatedly applying this rule is called *Unit Propagation*.

Given a propositional formula in CNF, the goal of the *Propositional Satisfiability Problem* (*SAT*) is to find an assignment for all the variables such that all clauses are satisfied or show that there is none. The SAT problem is *NP-Complete* (e.g. (Papadimitriou 1994)). It should be noted that throughout the remainder of this paper some familiarity with backtrack search SAT algorithms is assumed including Unit Propagation, *non-chronological backtracking* and *learning* techniques (Marques-Silva & Sakallah 1996).

An optimization version of the SAT problem is known as *Maximum Satisfiability* (*Max-SAT*). Given a CNF formula, the Max-SAT problem consists of finding an assignment for all the variables such that the number of satisfied clauses is maximized. The Max-SAT problem is *NP-Hard* (e.g. (Papadimitriou 1994)). Current backtracking Max-SAT search algorithms apply intensively Unit Propagation to prune the search space as will be shown later.

An instance P of the *Pseudo-Boolean Optimization* problem (Barth 1995a) can be defined as follows:

$$\begin{array}{ll} \text{minimize} & \sum\limits_{j \in N} c_j \cdot x_j \\ \text{subject to} & \sum\limits_{j \in N} a_{ij} l_j \geq b_i, \\ & x_j \in \{0,1\}, a_{ij}, b_i \in \mathbf{N}_0^+, i \in M \\ & N = \{1, \ldots, n\}, M = \{1, \ldots, m\} \end{array} \quad (1)$$

where c_j is a non-negative integer cost associated with variable $x_j, j \in N$ and a_{ij} denote the coefficients of the literals l_j in the set of m linear constraints. Every pseudo-Boolean formulation can be rewritten such that all coefficients a_{ij} and right-hand side b_i are non-negative. Observe that a linear pseudo-Boolean optimization problem can also be viewed as a special case of integer linear programming. The integer linear programming formulation for the constraints can be obtained if each literal $\bar{x}_j$ is replaced by $1 - x_j$.

Pseudo-Boolean constraints can be classified as *propositional clauses*, *cardinality constraints* or *general pseudo-Boolean constraints*. A constraint where all literal coefficients a_{ij} are 1 and the right-hand side b_i is also 1 is said to be a propositional clause. In a cardinality constraint, one can also have all literal coefficients a_{ij} equal to 1, but $b_i > 1$. All constraints other than propositional clauses or cardinality constraints are classified as general pseudo-Boolean constraints.

If every pseudo-Boolean (PB) constraint represents a propositional clause then P is an instance of the *Binate Covering problem* (BCP). Covering formulations and lower bounds including the *maximum independent set* (MIS) have been the subject of thorough research work (Coudert 1996; Liao & Devadas 1997; Manquinho & Marques-Silva 2004).

Algorithms for Pseudo-Boolean Optimization

The most effective PBO algorithms can be organized into two main classes. The majority of PBO algorithms perform a linear search on the values of the cost function, and at each step solving a set of PB constraints (Barth 1995a; Aloul *et al.* 2002; Chai & Kuehlmann 2003; Sheini & Sakallah 2006; Eén & Sörensson 2006). Another alternative approach is to perform a branch and bound search, while integrating the most effective SAT techniques (Manquinho & Marques-Silva 2004). In this approach, *upper bounds* on the value of the cost function are identified for each solution to the constraints, and *lower bounds* on the value of the cost function are estimated considering the current set of variable assignments. The search can be safely pruned whenever the lower bound estimate is higher than or equal to the most recently computed upper bound. This paper addresses branch and bound algorithms for PBO, focusing on techniques for computing tight lower bounds.

Several lower bound estimation procedures for PBO have been presented in the recent literature such as the ones based on linear-programming relaxations, Lagrangian relaxations, or the Log approximation approach (Manquinho & Marques-Silva 2004). Nevertheless, the approximation of a maximum independent set of clauses (Coudert 1996) is often used, because it represents a good trade-off between accuracy of the lower bound and computational effort. Observe that the tightness of the lower bounding procedure is crucial for the algorithms efficiency, because with higher estimates of the lower bound, the search can be pruned earlier.

With respect to the application of SAT to Boolean Optimization, P. Barth (Barth 1995a) first proposed a SAT-based approach for solving pseudo-Boolean optimization. This approach consists of performing a linear search on the possible values of the cost function, starting from the highest, at each step requiring the next computed solution to have a cost lower than the most recently computed upper bound. Whenever a new solution is found which satisfies all the constraints, the value of the cost function is recorded as the current lowest computed upper bound. If the resulting instance of SAT is not satisfiable, then the optimum value is given by the last recorded solution. More recent PBO solvers integrate the most effective SAT techniques, including clause learning and non-chronological backtracking (Aloul *et al.* 2002; Chai & Kuehlmann 2003; Sheini & Sakallah 2006; Eén & Sörensson 2006).

This paper focuses on the BSOLO branch and bound algorithm presented in (Manquinho & Marques-Silva 2000).

Here, a different algorithmic organization is described, consisting in the integration of several features from SAT algorithms in a branch-and-bound procedure to solve PBO instances. The BSOLO algorithm incorporates the most significant features from both approaches, namely the use of lower bounding from branch and bound algorithms, and the search pruning techniques from SAT algorithms, including a non-chronological backtracking search strategy and conflict-driven clause learning (Marques-Silva & Sakallah 1996). Mainly due to an effective conflict analysis procedure which allows non-chronological backtracking steps to be identified, it performs better than other branch-and-bound algorithms in several classes of instances (Manquinho & Marques-Silva 2000).

The main steps of the BSOLO algorithm can be described as follows:

1. Initialize the upper bound to the highest possible value (i.e. $ub = \sum_{j=1}^{n} \ c_j + 1$).
2. Start by checking whether the current state yields a conflict. This is done by applying unit propagation and, in case a conflict is reached, by invoking the conflict analysis procedure, learning clauses and performing backtracking if necessary.
3. If a solution to the constraints has been identified, update the upper bound according to $ub = \sum_{j=1}^{n} \ c_j \cdot x_j$.
4. Estimate a lower bound given the current variable assignments. If this value is higher than or equal to the current upper bound, a *bound conflict* arises and the conflict analysis procedure is invoked to determine to which part of the search tree the algorithm has to backtrack to. Continue from step 2.

Two types of conflicts can be found in the algorithm described above: *logical conflicts* that occur when at least one of the problem instance constraints becomes unsatisfied, and *bound conflicts* that occur when the lower bound is higher than or equal to the upper bound. When logical conflicts occur, the unsatisfied PB constraint is given to a *conflict analysis procedure* (Marques-Silva & Sakallah 1996; Manquinho & Marques-Silva 2004) which *learns a new clause* and determines to which point of the search procedure should backtrack to. Observe that it is also possible to learn a new PB constraint, as in (Chai & Kuehlmann 2003). In SAT and PBO, this procedure is often extremely effective, allowing *non-chronological backtracking*. Furthermore, learned clauses may avoid visiting useless parts of the search tree later during the search process. Similarly, whenever a bound conflict is identified, a new clause explaining the bound conflict should be provided to the conflict analysis procedure so that it can determine to which level of the search tree the algorithm can safely backtrack to. The approach for learning a new clause in the presence of bound conflicts is outlined in the next section.

Lower Bounds

This section describes two different ways of computing lower bounds for Pseudo-Boolean Optimization (PBO) and shows how they can be integrated in a branch and bound algorithm similar to the one described in the previous section. Accordingly, this section also establishes conditions for learning new propositional clauses from the bound conflicts so that the search procedure can benefit from non-chronological backtracking. Finally, it outlines interesting relations between both lower bounds.

Maximum Independent Set Lower Bound

The *maximum independent set* of constraints (MIS) is a method to estimate a lower bound on the value of the cost function based on an independent set of constraints. Since maximizing the cost of MIS is an NP-hard problem, a greedy computation is commonly used. The greedy procedure consists of finding a set of *disjoint constraints*, i.e. constraints with no literals in common among them.

The MIS lower bound was initially proposed to a special case of PBO (Coudert 1996) where all constraints are propositional clauses. In that approach, one would choose to include in MIS the clauses that maximize the ratio between their weight (defined by the minimum cost to satisfy the clause) and the number of literals. The minimum cost for satisfying the independent set of constraints is a lower bound on the optimal solution and can be defined as:

$$Cost(MIS) = \sum_{\omega \in MIS} Weight(\omega) \tag{2}$$

where $Weight(\omega)$ is the minimum cost to satisfy constraint ω. If all constraints were to be propositional clauses, we could define it as $Weight(\omega) = \min_{x_j \in \omega} c_j$. However, the minimum cost to satisfy a general pseudo-Boolean constraint ω is given by:

$$\begin{array}{ll} \text{minimize} & \sum\limits_{j \in C} c_j \cdot x_j \\ \text{subject to} & \omega \end{array} \tag{3}$$

where C denotes the set of indexes of literals in ω. Observe that (3) is a special case of a PBO problem known as *knapsack 0-1* problem. Nevertheless, it is still an NP-Complete problem (Karp 1972). Therefore, we use an approximation algorithm for the problem of finding the minimum cost of satisfying a pseudo-Boolean constraint by using a greedy algorithm. First, we determine the minimum number of literals that need to be true in order to satisfy ω by reducing it to a cardinality constraint (Barth 1995b).

Suppose that $\omega^{'}$ denotes the cardinality constraint obtained by the cardinality reduction algorithm applied to ω:

$$\omega^{'} = \sum_{j \in C} x_j \geq k \tag{4}$$

a lower bound on the minimum cost to satisfy ω is given by accumulating the cost of the first k literals in a sorted set of literal coefficients in the problem cost function, starting with the lowest c_j.

Unit Propagation Lower Bound

Given a PBO instance, an *inconsistent subset of constraints* (or simply an *inconsistent subset*) is a subset of constraints such that at least one of the constraints is always unsatisfied

by any assignment to the problem variables. Most of the lower bounds in the Max-SAT literature are based on detecting inconsistent subsets of propositional clauses. A general method to detect inconsistent subsets by intensively using Unit Propagation was already proposed for Max-SAT lower bounding (Li, Manyà, & Planes 2005). Later, this technique was generalized for the *Weighted Max-SAT* problem (Heras, Larrosa, & Oliveras 2007) where each clause has an associated cost, and the objective is to maximize the sum of the costs of the satisfied clauses. In what follows, the Unit Propagation lower bound for Max-SAT is extended for PBO.

To compute the Unit Propagation lower bound in PBO, we use a structure UP that contains pairs of subsets of inconsistent constraints and their associated costs. The procedure works as follows:

1. Initially $UP = \{\}$.
2. For each unassigned variable x_j with $c_j > 0$, add a new unit clause $\bar{x}_j$ that we call virtual clauses with an associated cost of c_j.
   (a) Apply Boolean propagation until a conflict is found, that is, a constraint becomes unsatisfied. It is well-known that one can retrieve a set of the constraints S_i involved in the conflict by inspection of the implication graph (Marques-Silva & Sakallah 1996). Let S'_i be a subset of S_i that contains only the virtual clauses of S_i, and m_i is the minimum cost of the coefficients associated to the clauses in S'_i. Formally,

$$m_i = \min_{\bar{x}_j \in \omega \ \wedge \ \omega \in S'_i} c_j$$

   (b) $UP = UP \ \cup \{< S_i, m_i >\}$.
   (c) Subtract m_i to all the coefficients associated to the virtual clauses in S'_i. Remove all virtual clauses in S'_i with coefficient 0.
   (d) Repeat steps from (a) to (c) until no more inconsistent subsets are found.

The necessary lower bound value estimated by the Unit Propagation lower bound in order to satisfy the PBO instance is given by:

$$Cost(UP) = \sum_{<S_i, m_i> \in UP} m_i$$

Bound Conflicts with Unit Propagation Lower Bound

During the search process, a bound conflict arises whenever the lower bound value higher than or equal to the upper bound. In this case, it is guaranteed that the current partial assignment cannot be extended such that a better solution can be found. For a given instance P of PBO, a bound conflict occurs when $P.path + P.lower \geq P.upper$, where $P.path$ is the cost of the assignments made from the root node to the current node of the search tree, $P.lower$ is a lower bound estimate on the cost of satisfying the constraints not yet satisfied (as given for example by MIS or Unit Propagation lower bound), and $P.upper$ is the best solution found so far.

Next we focus on bound conflicts due to the value of the Unit Propagation lower bound estimation procedure and show that in these situations non-chronological backtracking in the search tree is possible. To achieve this goal, it is necessary to identify a set of literals associated with the bound conflict such that if one of those literals is assigned value 1, then the bound conflicting condition is changed and may no longer hold.

Clearly, the value of $P.path$ is independent of which lower bound method is being used. Its value solely depends on the assignments of value 1 to variables with positive coefficients in the objective function. Therefore, in order for $P.path$ to decrease, at least one literals in ω_{pp} must have value 1, where ω_{pp} is defined as follows:

$$\omega_{pp} = \{\bar{x}_j : x_j = 1 \wedge c_j > 0\}$$

It is also necessary to define a set of literals ω_{pl} that explains the value of $P.lower$ when using the Unit Propagation lower bounding procedure. In this case, literals assigned value 0 in constraints of each subset of inconsistent constraints S_i are the ones that justify $P.lower$. If these literals were to have a different value, the Unit Propagation lower bound value might decrease. Hence, the set of literals ω_{pl} can be defined as follows:

$$\omega_{pl} = \{l : l = 0 \wedge l \in \omega \wedge \omega \in S_i \wedge < S_i, m_i > \in UP\}$$

Finally, given these sets of literals, it is possible to built a new propositional clause ω_{bc} that justifies the bound conflict where

$$\omega_{bc} = \omega_{pp} \cup \omega_{pl}$$

Observe that ω_{bc} is unsatisfied at the current node of the search tree. This new clause can then be used for performing conflict analysis, learning a new clause, and possibly backtracking non-chronologically.

Relating MIS and UP Lower Bounds

This section briefly presents some properties relating the MIS and Unit Propagation (UP) lower bounds. The following propositions state that MIS lower bound can always be *simulated* using the UP lower bound, but the converse is not true.

Proposition 1 *Let $M = \{\omega_1, \ldots, \omega_n\}$ be a maximum independent set of constraints of a formula φ found by any greedy execution of the MIS lower bound. It always exists some execution of the UP lower bound able to obtain the MIS estimation.*

Proof-Sketch 1 *Each constraint ω_i selected by MIS lower bound is in M since it exists a given number of literals in ω_i with coefficient $c_j > 0$ that must be assigned value 1 in order for ω_i to be satisfied. Hence, it is enough to propagate the virtual clauses with literals l_j associated with coefficients with $c_j > 0$ in the same way they appear sequentially in constraints $\{\omega_1, \ldots, \omega_n\}$.*

Proposition 2 *Let $I = \{I_1, \ldots, I_n\}$ be a set of inconsistent subsets of constraints of a formula φ found by any greedy execution of the UP lower bound. There may not exist an execution of the MIS lower bound able to obtain the same estimation.*

Proof-Sketch 2 *Suppose $c_1 = c_2 = 1$ and $c_3 = 0$ are the objective coefficients of variables x_1, x_2 and x_3. Consider that the problem contains the constraints $x_1 + x_3 \geq 1$ and $\bar{x}_3 + x_2 \geq 1$. In this case, the MIS lower bound estimation procedure returns 0 since the minimum cost to satisfy each constraint is 0. However, the UP lower bound procedure would return 1 since it would be able to find an inconsistent set with virtual clauses $\bar{x}_1$ and $\bar{x}_2$.*

The next propositions state that the estimation obtained with UP cannot be improved further by applying the MIS lower bound after the UP lower bound. However, in some cases, the MIS lower bound can be improved by later applying the UP lower bound procedure.

Proposition 3 *Let $I = \{I_1, \ldots, I_n\}$ be a set of inconsistent subsets of constraints of a formula φ found by any greedy execution of the UP lower bound. Afterwards, no greedy execution of MIS lower bound can improve the previous estimation.*

Proof-Sketch 3 *After applying the UP lower bound, it does not exist any constraint such that its minimum cost to satisfy is greater than 0 and is disjoint from all constraints in I. Otherwise, the UP lower bound procedure would be able to find another inconsistent set. Hence, MIS lower bound applied to the remaining constraints does not improve the UP lower bound estimation.*

Proposition 4 *Let $M = \{\omega_1, \ldots, \omega_n\}$ be a maximum independent set of clauses of a formula φ found by any greedy execution of the MIS lower bound. Afterwards, a greedy execution of UP lower bound may improve the previous estimation.*

Proof-Sketch 4 *Suppose that $c_1 = c_2 = 1$ and $c_3 = 0$ are the objective coefficients of variables x_1, x_2 and x_3. Consider that the problem contains the constraints $x_1 + x_3 \geq 1$ and $\bar{x}_3 + x_2 \geq 1$ and that x_1 and x_2 do not appear in any constraint in M. Then, in this case, the application of the UP lower bound would increase the lower bound estimation provided by MIS.*

Since MIS and UP lower bounds are greedy algorithms we cannot compare them directly. Nevertheless, from the last proposition, we can conclude that a good strategy would be to execute first the MIS lower bound, and later improve its estimation with the UP lower bound.

Experimental Results

This section evaluates the techniques described in the paper and compares them with other state of the art Pseudo-Boolean Optimization solvers. The experimental results were conducted on representative instances of the optimization of digital filters (Aksoy *et al.* 2007; Manquinho & Roussel 2007) and are shown in Table 1. The first column identifies each problem instance. The second column gives the optimum value (if unknown, then it is replaced with –). The rest of columns present the CPU time for each solver in seconds. The CPU times were obtained on a 3 GHz Xeon 5160 server with 4 GB of RAM running RHE Linux. If the time limit of 1800 seconds was reached, the table provides an indication of which was the best upper bound (ub) value found when the search was stopped (for example, ub10 means that the best solution found was 10).

The Unit Propagation lower bound was implemented in the BSOLO solver and compared with other Pseudo-Boolean solvers, namely MINISAT+ (Eén & Sörensson 2006) and PUEBLO (Sheini & Sakallah 2006). Moreover, we also present results for MINIMAXSAT (Heras, Larrosa, & Oliveras 2007) a Max-SAT solver that uses a Max-SAT Unit Propagation lower bound procedure and can also handle Pseudo-Boolean optimization instances. For BSOLO we provide results when using both MIS and the UP lower bounding procedure.

Results in Table 1 show that Pseudo-Boolean solvers are more effective than MINIMAXSAT. Moreover, results also show the performance improvements of BSOLO by using the proposed lower bound procedure. We can observe that the use of MIS is not effective in these instances, but UP lower bound is able to provide tighter lower bound values, allowing the solver to prune the search earlier. As a result, not only there are more instances that can be solved, but better upper bounds can be found when the solver was stopped at the time limit. For these instances, BSOLO with UP lower bound is able to perform much better than the other state of the art PBO solvers.

Conclusions

The paper describes the integration of the Max-SAT UP lower bound technique in SAT-based algorithms for Pseudo-Boolean Optimization, and outlines conditions for performing constraint learning and non-chronological backtracking based on computed lower bounds. These conditions provide novel mechanisms for extending the most effective SAT techniques to the use of the UP lower bound in PBO. Preliminary experimental results indicate that the utilization of the UP lower bound can be effective for instances of optimization of digital filters. Moreover, for the most well-known PBO solvers that do not integrate lower bounding techniques, the experimental results indicate that accurate lower bounding can be essential for representative instances of pseudo-Boolean optimization.

References

Aksoy, L.; da Costa, E. A. C.; Flores, P. F.; and Monteiro, J. C. 2007. Optimization of area in digital FIR filters using gate-level metrics. In *Design Automation Conf.*, 420–423.

Aloul, F.; Ramani, A.; Markov, I.; and Sakallah, K. 2002. Generic ILP versus specialized 0-1 ILP: An update. In *International Conf. on Computer Aided Design*, 450–457.

Argelich, J.; Li, C. M.; Manyà, F.; and Planes, J. 2007. MaxSAT evaluation. http://www.maxsat07.udl.es/.

Barth, P. 1995a. A Davis-Putnam enumeration algorithm for linear pseudo-Boolean optimization. Technical Report MPI-I-95-2-003, Max Plank Institute for CS.

Barth, P. 1995b. *Logic-Based 0-1 Constraint Programming*. Kluwer Academic Publishers.

Chai, D., and Kuehlmann, A. 2003. A fast pseudo-Boolean constraint solver. In *Design Automation Conf.*, 830–835.

Table 1: Results on area optimization in digital filter synthesis

Benchmark	optimum	MINIMAXSAT	MINISAT+	PUEBLO	BSOLO MIS	BSOLO UP
fir01_area_delay	5	0.02	0.01	0.01	0.01	0.01
fir02_area_delay	8	35.64	0.19	0.30	66.73	0.23
fir03_area_delay	10	ub195	166.77	ub10	ub11	2.45
fir04_area_delay	12	1182.62	0.78	10.49	ub15	0.95
fir05_area_delay	14	ub538	1247.31	ub14	ub16	578.66
fir06_area_delay	15	ub584	ub16	ub16	ub16	28.97
fir07_area_delay	–	ub4702	ub20	ub23	ub21	ub19
fir08_area_delay	–	–	–	ub40	ub31	ub31
fir09_area_delay	–	ub5023	ub22	ub25	ub25	ub23
fir10_area_delay	–	ub18884	–	ub44	ub37	ub37
fir01_area_opers	10	0.08	0.01	0.01	0.01	0.01
fir02_area_opers	18	242.00	0.34	6.84	ub18	0.03
fir03_area_opers	17	ub654	475.53	ub20	ub18	7.36
fir04_area_opers	29	ub29	0.51	22.88	ub29	0.03
fir05_area_opers	35	ub639	ub35	ub39	ub35	0.15
fir06_area_opers	23	ub2807	ub25	ub31	ub28	157.64
fir07_area_opers	–	ub7704	ub33	ub42	ub35	ub35
fir08_area_opers	–	–	–	ub70	ub57	ub57
fir09_area_opers	–	ub8180	ub43	ub46	ub42	ub42
fir10_area_opers	–	–	–	ub77	ub56	ub55
fir02_area_partials	19	0.01	0.01	0.01	0.01	0.01
fir03_area_partials	18	0.48	0.09	0.05	0.44	0.07
fir04_area_partials	30	0.01	0.02	0.01	0.01	0.01
fir05_area_partials	36	0.55	0.02	0.02	0.07	0.02
fir06_area_partials	24	0.57	0.25	0.03	0.32	0.13
fir07_area_partials	33	0.02	0.01	0.01	0.05	0.01
fir08_area_partials	49	ub203	20.27	ub53	ub53	ub52
fir09_area_partials	35	212.22	2.01	4.08	303.67	21.90
fir10_area_partials	52	ub52	3.42	6.57	ub54	510.76
fir01_trarea_ac	13949	5.09	24.78	1.83	2.26	0.01
fir02_trarea_ac	24890	177.70	ub26876	ub25328	ub26591	9.00

Coudert, O. 1996. On solving covering problems. In *Design Automation Conference*, 197–202.

Davis, M.; Logemann, G.; and Loveland, D. 1962. A machine program for theorem-proving. *Communications of the Association for Computing Machinery* 5:394–397.

Eén, N., and Sörensson, N. 2003. An extensible SAT solver. In *Sixth International Conference on Theory and Applications of Satisfiability Testing*, 502–518.

Eén, N., and Sörensson, N. 2006. Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation* 2:1–25.

Heras, F.; Larrosa, J.; and Oliveras, A. 2007. MiniMaxSat: An efficient weighted Max-SAT solver. *Journal of Artificial Intelligence Reserach. To appear.*

Karp, R. M. 1972. Reducibility among combinatorial problems. In Miller, R. E., and (eds.), J. W. T., eds., *Complexity of Computer Computations*. Plenum Press. 85–103.

Li, C.-M.; Manyà, F.; and Planes, J. 2005. Exploiting unit propagation to compute lower bounds in branch and bound max-sat solvers. In *International Conference on Principles and Practice of Constraint Programming*, 403–414.

Liao, S. Y., and Devadas, S. 1997. Solving covering problems using LPR-based lower bounds. In *Design Automation Conf.*, 117–120.

Manquinho, V., and Marques-Silva, J. P. 2000. Search pruning conditions for boolean optimization. In *European Conference on Artificial Intelligence*, 130–107.

Manquinho, V., and Marques-Silva, J. P. 2004. Satisfiability-based algorithms for boolean optimization. *Annals of Mathematics and Artificial Intelligence* 40(3-4).

Manquinho, V., and Roussel, O. 2007. Pseudo-Boolean evaluation 2007. http://www.cril.univ-artois.fr/PB07.

Marques-Silva, J. P., and Sakallah, K. A. 1996. GRASP: A new search algorithm for satisfiability. In *International Conference on Computer-Aided Design*, 220–227.

Moskewicz, M.; Madigan, C.; Zhao, Y.; Zhang, L.; and Malik, S. 2001. Engineering an efficient SAT solver. In *Design Automation Conf.*, 530–535.

Papadimitriou, C. 1994. *Computational Complexity*. USA: Addison-Wesley.

Sheini, H., and Sakallah, K. 2006. Pueblo: A hybrid pseudo-Boolean SAT solver. *Journal on Satisfiability, Boolean Modeling and Computation* 2:157–181.