A new Approach for Solving Satisfiability Problems with Qualitative Preferences

Emanuele Di Rosa and **Enrico Giunchiglia** and **Marco Maratea**[1]

Abstract. The problem of expressing and solving satisfiability problems (SAT) with qualitative preferences is central in many areas of Computer Science and Artificial Intelligence. In previous papers, it has been shown that qualitative preferences on literals allow for capturing qualitative/quantitative preferences on literals/formulas; and that an optimal model for a satisfiability problems with qualitative preferences on literals can be computed via a simple modification of the Davis-Logemann-Loveland procedure (DLL): Given a SAT formula, an optimal solution is computed by simply imposing that DLL branches according to the partial order on the preferences. Unfortunately, it is well known that introducing an ordering on the branching heuristic of DLL may cause an exponential degradation in its performances. The experimental analysis reported in these papers hightlights that such degradation can indeed show up in the presence of a significant number of preferences.

In this paper we propose an alternative solution which does not require any modification of the DLL heuristic: Once a solution is computed, a constraint is added to the input formula imposing that the new solution (if any) has to be better than the last computed. We implemented this idea, and the resulting system can lead to significant improvements wrt the original proposal when dealing with MIN-ONE/MAX-SAT problems corresponding to qualitative preferences on structured instances.

1 Introduction

The problem of expressing and solving satisfiability problems with qualitative preferences is central in many areas of Computer Science and Artificial Intelligence. For instance, in planning, beside the goals that have to be achieved, it is common to have other "soft" goals that it would be desiderable to satisfy: A plan is one solution which achieves all the goals, and an "optimal" plan is one which also achieves as many soft goals as possible. In planning as satisfiability [16] with soft goals [13], the task of finding an optimal plan is reduced to a satisfiability problem with qualitative preferences. Here, for simplicity, we consider qualitative preferences on literals, in which preferences are modeled as a set S of literals, and the relative importance of satisfying each literal in the set S is captured with a partial order on S. In [12, 13], it has been shown that

1. qualitative preferences on formulas and quantitative preferences on literals/formulas can be reduced to qualitative preferences on literals; and
2. that it possible to compute an optimal solution (wrt the expressed preferences) via a simple modification of the Davis-Logemann-Loveland procedure (DLL): In more details, an optimal solution is computed by imposing that branching occurs according to the partial order on the literals in the set of preferences.

This method for computing an optimal solution has the advantage that it only requires a simple modification of existing state-of-the-art SAT solvers all of which are based on DLL. However, it is well known that introducing an ordering on the branching heuristic of DLL may cause an exponential degradation in its performances [15]. OPTSAT is the name given to the related system built on top of MINISAT [10]. The experimental analysis reported in [12, 13] hightlights that such degradation can show up in the presence of a significant number of preferences.

In this paper we propose an alternative solution which does not require any modification of DLL heuristic and thus which does not have the above mentioned disadvantage. In a few words, once a solution is computed, a blocking formula is added to the input formula imposing that the new solution (if any) will be better than the last computed wrt the qualitative preference on literals expressed. Our approach works with *any* qualitative preference on literals, and thus (via the reductions described in [12, 13]) with any qualitative/quantitative preference on literals/formulas. We extended OPTSAT in order to incorporate this new method. In the following, we use OPTSAT-HS to refer to OPTSAT when using the method described in [12], and OPTSAT-BF to refer to OPTSAT when using the method here described.

To comparatively test the effectiveness of the approach, we consider MAX-SAT and MIN-ONE problems, in their non partial/partial[2], qualitative/quantitative versions, as in [12]. Our selection of benchmarks includes problems from the last MAX-SAT evaluation[3], well known satisfiability planning problems, and does not include problems with a (pseudo)-random structure. Indeed, OPTSAT is based on MINISAT, and MINISAT has been designed to solve large but relatively easy industrial SAT problems (and not small but relatively difficult randomly generated problems). In the qualitative case of (partial) MIN-ONE and MAX-SAT problems, the experimental results show that OPTSAT-BF performs better than OPTSAT-HS. The reasons for the good performances of OPTSAT-BF are:

1. The good quality of the first computed solution, and
2. The few iterations required to get to the determined optimal solution.

In the quantitative case, OPTSAT-BF is competitive also with respect to the other state-of-the-art systems for MAX-SAT, including the most performing systems in the recent PB and MAX-SAT evaluations.

Summing up, the main contributions of the paper are:

[1] DIST - Università di Genova, Italy, email: {emanuele,enrico,marco}@dist.unige.it

[2] In the *partial* MIN-ONE (resp. MAX-SAT) problem the optimization has to be performed on a subset of the variables (resp. clauses) of the problem.

[3] `http://www.maxsat07.udl.es/`

- We define a new approach for solving satisfiability problems with qualitative preferences.
- We formally state some properties of our algorithm.
- We extend OPTSAT in order to implement this new approach.
- On (partial) MAX-SAT and MIN-ONE non (pseudo)-random problems, we show that OPTSAT-BF performs better than OPTSAT-HS in the qualitative case, and that is competitive wrt other state-of-the-art systems in the quantitative case.

The paper is structured as follows. In Section 2 we review our formalism for expressing preferences. Section 3 is dedicated to the presentation of the algorithm behind OPTSAT-BF, and its formal properties. Section 4 presents the experimental analysis we conducted. Section 5 ends the paper with some final remarks.

2 Satisfiability and Qualitative Preferences

Consider a finite set P of *variables*. A *literal* is a variable x or its negation $\overline{x}$. We assume $\overline{\overline{x}} = x$. A *clause* is a finite disjunction of literals and a *formula* is a finite conjunction of clauses. As customary in SAT, we also represent clauses as sets of literals and formulas as sets of clauses, and we use $\top$ and $\bot$ to denote the empty set of clauses and the empty clause respectively. For example, given the 4 variables *Fish, Meat, RedWine, WhiteWine*, the formula

$$\{\overline{Fish}, \overline{Meat}\}, \{\overline{RedWine}, \overline{WhiteWine}\} \quad (1)$$

models the fact that we cannot have both fish (*Fish*) and meat (*Meat*), both red (*RedWine*) and white (*WhiteWine*) wine.

An *assignment* is a consistent set of literals. If $l \in \mu$, we say that both l and $\overline{l}$ are *assigned* by μ. An assignment μ is *total* if each literal l is assigned by μ. A total assignment μ *satisfies a formula* φ if for each clause $C \in \varphi$, $C \cap \mu \neq \emptyset$. A *model* μ of a formula φ is an assignment satisfying φ. A formula φ *entails* a formula ψ if the models of φ are a subset of the models of ψ. For instance, (1) has 9 models. In the following, we abbreviate a total assignment with the set of variables assigned to true, and we write $\mu \models \psi$ to indicate that μ is a model of ψ. For instance, we write $\{Fish, WhiteWine\}$ as an abbreviation for the total assignment $\{Fish, \overline{Meat}, WhiteWine, \overline{RedWine}\}$ in which the only variables assigned to true are *Fish* and *WhiteWine*, i.e., the situation in which we have fish and white wine.

A *qualitative preference on literals* is a partially ordered set of literals, i.e., a pair $S, \prec$ where S is a set of literals (also called the *set of preferences*), and $\prec$ is a partial order on S. Intuitively, S represents the set of literals that we would like to have satisfied, and $\prec$ models the relative importance of our preferences. For example,

$$\{Fish, RedWine, WhiteWine\}, \{WhiteWine \prec RedWine\} \quad (2)$$

models the case in which we prefer to have fish and both red and white wine. In the case in which it is not possible to have both red and white wine, we like more to have white than red wine. A qualitative preference $S, \prec$ on literals can be extended to the set of total assignments as follows: Given two total assignments μ and μ', *μ is preferred to μ'* ($\mu \prec \mu'$) if and only if

1. there exists a literal $l \in S$ with $l \in \mu$ and $\overline{l} \in \mu'$; and
2. for each literal $l' \in S \cap (\mu' \setminus \mu)$, there exists a literal $l \in S \cap (\mu \setminus \mu')$ such that $l \prec l'$.

A model μ of a formula φ is *optimal* if it is a minimal element of the partially ordered set of models of φ. For instance, considering the qualitative preference (2), the formula (1) has only one optimal model, i.e., $\{Fish, WhiteWine\}$.

We recall that qualitative preference on formulas can be reduced to qualitative preferences on literals (see [13]); and that by propositional encoding of the objective function to maximize/minimize, it is possible to reduce also quantitative preferences to qualitative ones, see [12].

3 Solving satisfiability problems with preferences

Consider a formula φ and a qualitative preference on literals $S, \prec$. The problem of computing an optimal model of φ wrt $S, \prec$ can be solved by

1. computing a (not necessarily optimal) model μ of φ,
2. adding a formula which restricts the subsequent search for models to those which are preferred to μ,
3. iterating the above two steps up to the point that the last assignment found can no longer be improved.

Crucial for the above procedure is a condition which enables us to say which are the models that are preferred (wrt $S, \prec$) to an assignment μ. The *preference formula for μ wrt $S, \prec$* is

$$(\vee_{l:l \in S, l \notin \mu} l) \wedge (\wedge_{l':l' \in S, l' \in \mu} (\vee_{l:l \in S, l \notin \mu, l \prec l'} l \vee l')). \quad (3)$$

An assignment μ' is preferred to μ wrt $S, \prec$ iff μ' satisfies (3), as stated by the following theorem.

Theorem 1 *Let μ and μ' be two total assignments. Let $S, \prec$ be a qualitative preference. μ' is preferred to μ wrt $S, \prec$ if and only if μ' satisfies the preference formula for μ wrt $S, \prec$.*

As an example of the application of the theorem above consider the following particular cases:

1. $S \subseteq \mu$, (e.g., because there are no preferences, $S = \emptyset$): In this case (3) is equivalent to $\bot$, meaning that there is no assignment which is preferred to μ, i.e., that μ is already optimal;
2. $S, \prec = \{l_1, \ldots, l_n\}, \emptyset$: In this case (3) becomes $(\vee_{l:l \in S, l \notin \mu} l) \wedge (\wedge_{l':l' \in S, l' \in \mu} l')$, meaning that any assignment μ' with $\mu' \prec \mu$ must be such that $\mu \cap S \subset \mu' \cap S$;

Considering the preference (2),

1. if $\mu_1 = \{Meat, RedWine\}$, then (3) is
$$\psi_1 : (Fish \vee WhiteWine) \wedge (WhiteWine \vee RedWine)$$
2. if $\mu_2 = \{Meat, WhiteWine\}$, then (3) is
$$\psi_2 : (Fish \vee RedWine) \wedge WhiteWine$$
3. if $\mu_3 = \{Fish, WhiteWine\}$, then (3) is
$$\psi_3 : RedWine \wedge Fish \wedge WhiteWine.$$

Notice that $\mu_2 \prec \mu_1$ and $\mu_3 \prec \mu_2$: As a consequence ψ_2 entails ψ_1 and ψ_3 entails ψ_2. Further, as the last example makes clear, it is indeed possible that the preference formula for an assignment is inconsistent with the given set of constraints, and this is indeed an obvious consequence of the fact that the definition of (3) does not take into account the input formula: In the case in which the preference formula for an assignment μ is inconsistent with the input set of clauses, μ is optimal.

As we have already said at the beginning of the section, Theorem 1 allows us to use any complete SAT solver as a black box for

$S, \prec$:= a qualitative preference on literals;
φ := the input formula; $\psi := \top$; $\mu_{opt} := \emptyset$

function PREF-DLL($\varphi \cup \psi$,μ)
1 **if** ($\bot \in (\varphi \cup \psi)_\mu$) **return** FALSE;
2 **if** (μ is total) $\mu_{opt} := \mu$; $\psi :=$ *Reason*$(\mu, S, \prec)$; **return** FALSE;
3 **if** ($\{l\} \in (\varphi \cup \psi)_\mu$) **return** PREF-DLL($\varphi \cup \psi, \mu \cup \{l\}$);
4 $l :=$ *ChooseLiteral*$(\varphi \cup \psi, \mu)$;
5 **return** PREF-DLL($\varphi \cup \psi, \mu \cup \{l\}$) **or**
PREF-DLL($\varphi \cup \psi, \mu \cup \{\overline{l}\}$).

Figure 1. The algorithm of PREF-DLL.

computing an optimal assignment. Once a model μ of a formula φ is found, the formula (3) is computed and added to φ and then the SAT solver can be invoked: The returned model is ensured to be preferred to μ. However, given that all the state-of-the-art systems are based on DLL, it is possible, following what has been successfully done in various areas of automated deduction (see, e.g., [2]), to add the formula (3) as soon as μ is determined, i.e., *during* the search. The resulting procedure is represented in Figure 1.

In the figure:

- φ is the input set of clauses, $S, \prec$ is a qualitative preference on literals, μ_{opt} is the (current) optimal assignment, ψ is the set of clauses corresponding to the preference formula for μ_{opt} wrt $S, \prec$; μ is an assignment;
- $(\varphi \cup \psi)_\mu$ is the set of clauses obtained from $\varphi \cup \psi$ by (i) deleting the clauses $C \in \varphi \cup \psi$ with $\mu \cap C \neq \emptyset$, and (ii) substituting the other clauses $C \in \varphi \cup \psi$ with $C \setminus \{\overline{l} : l \in \mu\}$;
- *Reason*$(\mu, S, \prec)$ returns the set of clauses corresponding to the preference formula for μ wrt $S, \prec$;
- *ChooseLiteral*$(\varphi \cup \psi, \mu)$ returns a literal in $\varphi \cup \psi$ which is unassigned by μ.

It is easy to see that PREF-DLL is exactly the same as DLL, except that once a model μ is determined (see line 2),

1. μ is stored in μ_{opt};
2. the preference formula for μ wrt $S, \prec$ is stored in ψ, and
3. FALSE is returned.

Notice that PREF-DLL generalizes DLL in the sense that if there are no preferences (i.e., if $S = \emptyset$), PREF-DLL behaves as DLL: Indeed, if $S = \emptyset$ then any model is optimal, and as soon as one model μ is found, the preference formula for μ wrt $S, \prec$ (i.e., $\bot$) determines the termination of PREF-DLL.

Theorem 2 *Let φ be a formula and $S, \prec$ a qualitative preference on literals.* PREF-DLL$(\varphi, \emptyset)$ *terminates, and then μ_{opt} is empty if φ is unsatisfiable, and an optimal model of φ wrt $S, \prec$ otherwise.*

Beside the above, one interesting property of PREF-DLL is its "anytime" property: The sequence of models $\mu_1, \mu_2, \ldots, \mu_n$ computed by PREF-DLL are ensured to be such that μ_{i+1} is preferred to μ_i, i.e., $\mu_{i+1} \prec \mu_i$ $(0 < i < n)$. Thus, PREF-DLL is as fast as DLL to compute the first model of the input set of clauses, and, time permitting, from that point on, it can only improve the quality of the model found. Also notice that in Figure 1 we called *Reason* the procedure for computing the preference formula (3). Indeed, most of the current SAT solvers (at least those meant for applications) are based on learning: As soon as a clause C becomes empty, C is returned and then used by the learning mechanism of the solver to backjump over irrelevant nodes while backtracking, and, with learning, to prune the subsequent search of the solver. Such clause C is often called "reason" or conflict clause, and it has the property that it is falsified by the assignment μ which caused C to become empty (i.e., for each literal $l \in C$, $\overline{l} \in \mu$). In our case, with solvers based on learning, as soon as the assignment μ is total and no empty clause is detected, we can return the clause C corresponding to the left conjunct of (3) as conflict clause: Indeed, $\vee_{l \in S, l \notin \mu} l$ is falsified by μ. However, we must also add the other clauses corresponding to (3) to the input set of clauses, since these are needed to ensure that the search will continue looking for another model μ' of the input formula with $\mu' \prec \mu$. Fortunately, the clauses added to the input set of clauses, do not need to be indefinitely retained (otherwise PREF-DLL can have an exponential blow up in space): Once a new model μ' with $\mu' \prec \mu$ is found, we can discard the clauses added because of μ since they are entailed by the new clauses added because of μ', as stated by the following theorem.

Theorem 3 *Let $S, \prec$ be a qualitative preference. Let $\mu_1, \mu_2, \ldots, \mu_n$ be the sequence of models computed by* PREF-DLL, *and $\psi_1, \psi_2, \ldots, \psi_n$ be the corresponding preference formulas. For each i, $0 < i < n$, ψ_{i+1} entails ψ_i.*

In PREF-DLL (see Figure 1), the preference formula ψ_i for μ_i is overwritten as soon as a new model μ_{i+1} is determined (line 2). PREF-DLL is thus guaranteed to work in polynomial space in the size of the input formula and qualitative preference.

4 Implementation and experimental analysis

We extended OPTSAT [12] in order to incorporate these ideas. OPTSAT is built on top of MINISAT [10], the 2005 version, winner of the SAT 2005 competition on the industrial benchmarks category (together with the SAT/CNF minimizer SATELITE [9]): Such choice has been motivated by our interest in solving, in particular, large structured problems coming from applications. The two versions of OPTSAT —OPTSAT-HS and OPTSAT-BF— are the ones that we consider in the case of qualitative preferences.

In the case on quantitative preference, OPTSAT encodes the objective function using the methods described in [23, 3]: Here we used the one based on [23]. Table 1 shows the results for OPTSAT-HS and OPTSAT-BF on a variety of problems detailed below. The table shows the results also for various other state-of-the-art solvers included for completeness. In particular we considered both dedicated solvers for

- MAX-SAT problems, like BF [6]; MAXSOLVER [24]; TOOLBAR [21, 17] ver. 3.0; MAXSATZ version submitted to the 2007 Evaluation [18]; MINIMAXSAT ver. 1.0 [14] and abbreviated with MMSAT in the Table; and
- generic Pseudo-Boolean solvers, like OPBDP ver. 1.1.1 [4]; PBS ver. 2.1 and ver. 4 [1]; MINISAT+ ver. 1.13 [11] and abbreviated with MSAT+ in the Table; GLPPB ver. 0.2 by the same authors of PUEBLO [22] as submitted to the 2007 Evaluation[4]; BSOLO ver. 3.0.17 [19].

MAXSATZ and MINIMAXSAT have been the winner of the recent Max-SAT Evaluation 2007 in the "Max-SAT" and "Partial Max-SAT" category, respectively. MINISAT+ was the solver able to prove

[4] http://www.eecs.umich.edu/~hsheini/pueblo/

	class	#I	OPTSAT-HS	OPTSAT-BF	OPBDP	PBS4	MSAT+	BSOLO	MAXSATZ	MMSAT	OPTSAT-HS	OPTSAT-BF
1	Partial MINONE	21	77.99(19)	2.7(21)	–	223.14(15)	43.32(18)	433.21(16)		391.21(12)	74.28(21)	69.89(21)
2	MINONE	26	0.69(26)	0.2(26)	85.37(7)	17.56(19)	7.33(24)	115.73(22)		87.21(24)	93.24(24)	23.99(25)
3	MAXSAT	35	26.68(34)	11.25(35)	20.89(3)	98.55(10)	130.37(31)	192.56(23)	274.38(22)	229.73(21)	218.86(31)	175.12(31)
4	MAXCUT/spinglass	5	0.01(5)	0.01(5)	0.99(1)	66.67(1)	0.86(1)	76.57(1)	33.19(3)	1.09(3)	7.56(1)	7.52(1)
5	MAXCUT/dimacs_mod	62	0.01(62)	0.01(62)	230.33(5)	0.01(2)	247.54(7)	0.01(2)	59.27(52)	194.52(52)	66.86(4)	21.61(3)
6	PSEUDO/garden	7	0.02(7)	0.01(7)	2.2(4)	147.58(4)	0.25(5)	30.18(4)		4.75(5)	22.8(5)	36.66(5)
7	PSEUDO/logic-synthesis	17	0.03(17)	0.01(17)	–	85.88(1)	490.36(5)	–		81.93(2)	90.36(3)	338.26(3)
8	PSEUDO/primes	148	4.81(130)	0.19(131)	16.65(85)	18.08(90)	11.52 (104)	22.23 (94)		62.08 (107)	31.8(103)	60.59(109)
9	PSEUDO/routing	15	11.69(15)	3.12(15)	81.83(5)	102.75(9)	43.74(15)	373.73(8)		109.49(14)	41.49(15)	36.1(15)
10	MAXONE/structured	60	0.96(60)	0.13(60)	296.26(35)	11.48(60)	2.02(58)	40.96(60)		22.5(60)	293(56)	7.87(58)
11	MAXCLIQUE/structured	62	0.01(62)	0.06(62)	70.37(16)	23.79(13)	154.39(22)	248.26(14)		61.97(36)	54.14(19)	178.04(23)

Table 1. Results for solving satisfiability problems with qualitative (columns 4-5) and quantitative (columns 6-13) preferences. Problems are: Partial MIN-ONE (row 1), MIN-ONE (row 2), MAX-SAT (rows 3-5), and partial MAX-SAT (rows 6-11).

unsatisfiability and optimality to a larger number of instances than all the other solvers that entered into the Pseudo-Boolean Evaluation 2005 [20], and the best performing solver (together with BSOLO) also in the Pseudo-Boolean Evaluation 2006, category OPT-SMALLINT-LIN. BSOLO and GLPPB have been the best performing PB solvers in the OPT-SMALLINT-LIN category of the recent Pseudo-Boolean evaluation 2007. Considering the dedicated solvers for MAX-SAT, we discarded BF, MAXSOLVER and TOOLBAR after an initial analysis because they seem to be tailored for randomly generated problems, and are thus not suited to deal with the problems we consider here. About the Pseudo-Boolean solvers, we do not show the results for PBS ver. 2.1 and GLPPB because they are almost always slower than PBS ver. 4.0 and BSOLO, respectively, and, ultimately, they manage to solve only a few of the instances we consider.

About the benchmarks, we considered a wide set of instances, mainly coming for real-world applications. In particular, we used SATPLAN 2004, release of 10 Feb. 2006 to generate the partial MIN-ONE problems of row 1: In more details, we considered several domains from previous International Planning Competitions (IPCs); generated the first satisfiable instances with SATPLAN; and, for such instance, we considered the partial MIN-ONE problem of minimizing the set of action variables set to true. For MIN-ONE and MAX-SAT problems, we selected well known satisfiable and unsatisfiable SAT instances from several domains, i.e., Formal Verification instances from the Bejing'96 competition, planning problems from SATPLAN contributed by Kautz and Selman, Data Encryption Standard (DES) instances, quasi group instances, and bounded model checking (BMC) problems used in the original BMC paper [5], miter-based circuit equivalence benchmarks by Joao Marques-Silva: Each of these satisfiable instances corresponds to a MIN-ONE problem and the results are presented in row 2, while the unsatisfiable instances correspond to the MAX-SAT problems whose results are in row 3. Finally, we included in our analysis also (partial) MAX-SAT problems from the recent MAX-SAT evaluation, rows 4-11: As it emerges from the results of this evaluation[5], these benchmarks are hard; the performances of the best solvers differ only for a factor, no solver clearly wins; and it is difficult to solve even a single instance more than the other solvers.

Each solver has been run using its default settings. All the experiments have been run on a Linux box equipped with a Pentium IV 3.2GHz processor and 1GB of RAM. CPU time is measured in seconds; timeout has been set to 1800 seconds. In Table 1,

- column 2 is the class of the problems;
- column 3 is the number of instances in the class;
- columns 4-5 are dedicated to qualitative preferences;
- columns 6-14 are for the quantitative case.

Results for solvers are cumulatively presented as in the report of the MAX-SAT Evaluations: Given a set of instances, we show the mean CPU time of the solved instances, and the number of solved ones (in parenthesis). MAXSATZ can only deal with MAX-SAT problems, and this is why the corresponding results for MIN-ONE and partial MIN-ONE/MAX-SAT are missing.

In the qualitative case we can see that OPTSAT-BF (column 5) is consistently better than OPTSAT-HS (column 4), both in terms of mean CPU time and solved instances: OPTSAT-BF solves the same number of instances of OPTSAT-HS, or higher, and in less time, sometimes dramatically (see, e.g, rows 1 and 8), but for row 11 which is nonetheless solved very easily by both solvers.

In the quantitative case, OPTSAT-BF performs also well on these benchmarks. We have to remind that these benchmarks do not include many problems from the last evaluations because of their (pseudo)-random structure which is not suited for our solver. For fairness, this also implies that it is not clear whether the problem we selected are suited for the other solver in our analysis. Indeed, we conducted a preliminary analysis on the (pseudo)-random problems we excluded, and we got a different picture, in which other solvers (and in particular MMSAT) emerge.

	class	T_1	Q_1	#Sols	T_f	Q_f
1	Partial MINONE	2.68	45.5	2.5	2.7	44.1
2	MINONE	0.19	751.6	2	0.2	751.6
3	MAXSAT	0.05	8605.2	21.2	11.25	8847.6
4	MAXCUT/spinglass	0.01	770.4	2	0.01	770.4
5	MAXCUT/dimacs_mod	0.01	695.9	2.2	0.01	701.9
6	PSEUDO/garden	0.01	496	2	0.01	496
7	PSEUDO/logic-synthesis	0.01	152.2	2	0.01	152.2
8	PSEUDO/primes	0.18	368.4	2	0.19	368.4
9	PSEUDO/routing	3.12	58.7	2	3.12	58.7
10	MAXONE/structured	0.12	240.5	8.4	0.13	249.8
11	MAXCLIQUE/structured	0.06	430.4	2	0.06	430.4

Table 2. CPU time for finding first (column T_1) and optimal (column T_f) solution. 1 + number of models computed by OPTSAT-BF (column $\#Sols$). Quality of the first (column Q_1) and optimal (column Q_f) solution.

[5] See the slides about the results, available at http://www.maxsat07.udl.es/ms07-pre.pdf.

In order to understand the good behavior of our algorithm, Table 2 shows, for each class, the average of the CPU times for finding the first (even if not optimal) (column T_1) and optimal (column T_f) solution; the average quality[6] of the first (column Q_1) and optimal (column Q_f) solution; and the average of 1 + the number of models computed by OPTSAT-BF (column $\#Sols$). Looking at the table, we see that the good performances of OPTSAT-BF can be explained by the following factors:

1. the relative quality of the first solution (i.e., Q_f/Q_1 for rows 1-2 and Q_1/Q_f for rows 3-11) is usually very high, greater than 0.96; and
2. the low number of intermediate solutions generated before the optimal one: For 9 classes out of 11, the number in column $\#Sols$ is lower or equal than 2.5. Considering that 2 indicates that the first computed model is already optimal, this means that the algorithm converges to an optimal model very quickly.

Finally note how, for the two classes in which the first solution is of a low quality, i.e., rows 3 and 10 in Table 2, the convergence is very different: For the MAXSAT class in row 3, T_1 is negligible, and all CPU time is spent in "filling the gap" with the optimal result; while for the MAXONE/structured class, most of the time is spent looking for the first solution. As a consequence, in MAX-SAT (resp. MAXONE/structured) the optimal solution is reached by a serie of relatively difficult (resp. easy) intermediate steps.

5 Conclusions

We have defined and implemented a new approach based on DLL for solving satisfiability problems with preferences which does not need any modification to DLL heuristic. The basic idea is that whenever a solution is found, a formula is added to the input set of clauses ensuring that the new model (if any) will be better than the last computed one. The experimental analysis performed on a wide set of, mainly structured, (partial) MAX-SAT and MIN-ONE benchmarks has shown that it leads in most cases to significant improvements when dealing with qualitative preferences, and that it is also competitive with other state-of-the-art systems in the quantitative case.

There is a huge literature on expressing and reasoning with preferences, see, e.g. [8], and the various events on preferences taking place every year. If we do not take into account [12, 13], the closest work to ours seems to be the one on CP-nets [7]: In the paper, the authors show that exploring the search space according to the partial order on the values of the variables, the first solution determined is guaranteed to be optimal. CP-nets allows for non-Boolean variables, but on the other hand they only allow to express preferences between values of a same variable: Thus, modeling "I prefer a to b" where a and b are distinct propositional variables cannot be directly captured.

REFERENCES

[1] Fadi A. Aloul, Arathi Ramani, Igor L. Markov, and Karem A. Sakallah, 'PBS: A backtrack search pseudo-Boolean solver', in *Proc. SAT*, (2002).

[2] Alessandro Armando, Claudio Castellini, Enrico Giunchiglia, Fausto Giunchiglia, and Armando Tacchella, 'SAT-based decision procedures for automated reasoning: a unifying perspective', in *Mechanizing Mathematical Reasoning: Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday*, volume 2605 of *LNCS*, Springer Verlag, (2005).

[3] Olivier Bailleux and Yacine Boufkhad, 'Efficient CNF encoding of Boolean cardinality constraints.', in *Proc. CP*, pp. 108–122, (2003).

[4] P. Barth, 'A Davis-Putnam enumeration algorithm for linear pseudo-boolean optimization', Technical report, Max Plank Instutute for Computer Science, (1995). technical Report MPI-I-95-2-2003.

[5] A. Biere, A. Cimatti, E. M. Clarke, M. Fujita, and Y. Zhu, 'Symbolic model checking using SAT procedures instead of BDDs', in *Proceedings of the 36th Design Automation Conference (DAC' 99)*, pp. 317–320. Association for Computing Machinery, (1999).

[6] Brian Borchers and Judith Furman, 'A two-phase exact algorithm for max-SAT and weighted max-SAT problems.', *J. Comb. Optim.*, **2**(4), 299–306, (1998).

[7] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole, 'CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements', *J. Artif. Intell. Res. (JAIR)*, **21**, 135–191, (2004).

[8] Jon Doyle, 'Prospects for preferences', *Computational Intelligence*, **20**(2), 111–136, (2004).

[9] Niklas Eén and Armin Biere, 'Effective preprocessing in sat through variable and clause elimination', in *Theory and Applications of Satisfiability Testing, 8th International Conference, SAT 2005*, volume 3569 of *Lecture Notes in Computer Science*, pp. 61–75. Springer, (2005).

[10] Niklas Eén and Niklas Sörensson, 'An extensible SAT-solver', in *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003, Selected Revised Papers*, pp. 502–518, (2003).

[11] Niklas Eén and Niklas Sörensson, 'Translating pseudo-Boolean constraints into SAT', *Journal on Satisfiability, Boolean Modeling and Computation*, **2**, 1–26, (2006).

[12] E. Giunchiglia and M. Maratea, 'Solving optimization problems with DLL', in *Proc. of 17th European Conference on Artificial Intelligence (ECAI)*, pp. 377–381, (2006).

[13] Enrico Giunchiglia and Marco Maratea, 'Planning as satisfiability with preferences', in *In Proc. of 22nd AAAI Conference on Artificial Intelligence*, pp. 987–992. AAAI Press, (2007).

[14] Federico Heras, Javier Larrosa, and Albert Oliveras, 'MiniMaxSat: A new weighted max-sat solver', in *Proc. of Theory and Applications of Satisfiability Testing - SAT 2007, 10th International Conference*, volume 4501 of *LNCS*, pp. 41–55. Springer, (2007).

[15] Matti Järvisalo, Tommi Junttila, and Ilkka Niemelä, 'Unrestricted vs restricted cut in a tableau method for Boolean circuits', *Annals of Mathematics and Artificial Intelligence*, **44**(4), 373–399, (August 2005).

[16] Henry Kautz and Bart Selman, 'Planning as satisfiability', in *Proc. ECAI*, pp. 359–363, (1992).

[17] Javier Larrosa, Federico Heras, and Simon de Givry, 'A logical approach to efficient Max-SAT solving', *Artificial Intelligence*, **172**, 204–233, (2008).

[18] Chu Min Li, Felip Manya, and Jordi Planes, 'New inference rules for max-sat', *Journal of Artificial Intelligence Research (JAIR). To appear.*, (2007).

[19] V. M. Manquinho and J. P. Marques-Silva, 'On using cutting planes in pseudo-boolean optimization', *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)*, **2**, 209–219, (2006).

[20] Vasco Miguel Manquinho and Olivier Roussel, 'The first evaluation of pseudo-Boolean solvers (PB'05)', *Journal on Satisfiability, Boolean Modeling and Computation*, **2**, 103–143, (2006).

[21] P. Meseguer S. De Givry, J. Larrosa and T. Schieux, 'Solving Max-SAT as weighted CSP', in *Proc. of 9th International Conference on Principles and Practice of Constraint Programming (CP 2003)*, volume 2833 of *Lecture Notes in Computer Science*, pp. 363–376, (2003).

[22] Hossein M. Sheini and Karem A. Sakallah, 'Pueblo: A modern pseudo-boolean sat solver', in *2005 Design, Automation and Test in Europe Conference and Exposition (DATE 2005), 7-11 March 2005, Munich, Germany*, pp. 684–685. IEEE Computer Society, (2005).

[23] Joost P. Warners, 'A linear-time transformation of linear inequalities into conjunctive normal form.', *Information Processing Letters*, **68**(2), 63–69, (1998).

[24] Z. Xing and W. Zhang, 'MaxSolver: An efficient exact algorithm for (weighted) maximum satisfiability', *Artificial Intelligence*, **164**(1-2), 47–80, (2005).

[6] Quality is measured in terms of number of variables assigned to true for MIN-ONE and partial MIN-ONE problems, and in terms of number of satisfied clauses for MAX-SAT and partial MAX-SAT problems.