Algorithms for Solving Satisfiability Problems with Qualitative Preferences

Enrico Giunchiglia and Marco Maratea

DIST, Università di Genova, Viale Causa, 13 – 16145 Genova, Italy
{enrico,marco}@dist.unige.it

Abstract. In this work we present a complete picture of our work on computing optimal solutions in satisfiability problems with qualitative preferences. With this task in mind, we first review our work on computing optimal solutions by imposing an ordering on the way the search space is explored, e.g., on the splitting heuristic in case the DPLL algorithm is used. The main feature of this approach is that it guarantees to compute all and only the optimal solutions, i.e., models which are not optimal are not even computed: For this result, it is essential that the splitting heuristic of the solver follows the partial order on the expressed preferences. However, for each optimal solution, a formula that prunes non-optimal solutions needs to be retained, thus this procedure does not work in polynomial space when computing all optimal solutions.
We then extend our previous work and show how it is possible to compute optimal solutions using a generate-and-test approach: Such a procedure is based on the idea to first compute a model and then check for its optimality. As a consequence, no ordering on the splitting heuristic is needed, but it may compute also non-optimal models. This approach does not need to retain formulas indefinitely, thus it does work in polynomial space.
We start from a simple setting in which a preference is a partial order on a set of literals. We then show how other forms of preferences, i.e., quantitative, qualitative on formulas and mixed qualitative/quantitative can be captured by our framework, and present alternatives for computing "complete" sets of optimal solutions. We finally comment on the implementation of the two procedures on top of state-of-the-art satisfiability solvers, and discuss related work.

1 Introduction

The problem of finding an optimal solution in a satisfiability (SAT) problem with qualitative preferences has attracted a lot of researchers in Artificial Intelligence in general, and in the constraint and logic programming community in particular. As a consequence, several approaches for expressing and reasoning with SAT problems with preferences have been proposed, and viable solutions exist, especially for finding one optimal solution. However, in some cases, it is not desirable to find just one solution. Indeed, it might be desirable to be able to compute more, and possibly all, solutions, e.g., for comparatively evaluate them on the basis of other criteria not captured by the preferences. See, e.g., [4, 34, 8, 6, 25, 10, 18, 49, 3] for approaches for finding one and all optimal solutions.

A simple approach for finding optimal solutions consists in first enumerating all (non necessarily optimal) solutions, and then eliminating a solution μ if there exists another solution μ' which is "preferred" to μ. The first obvious drawback of this approach is that it requires the computation of all solutions, even the non optimal ones. The second drawback is that each solution has to be stored and compared with the others. In [5], in the context of CP-nets [4], the authors noticed, that by imposing an ordering on the splitting heuristic used for searching solutions, it is possible to mitigate the second drawback by comparing a solution only with the previously generated ones, which are already guaranteed to be optimal: In this way, only the so far generated optimal solutions need to be stored. Still, the number of optimal solutions can be exponential and all the solutions (even the non-optimal ones) are computed. Further, it is well known that imposing an ordering on the splitting heuristic may lead to a significant degradation in the performances of the solver used for finding solutions [38, 39].

In this work we present two procedures, based on the Davis-Putnam-Logemann-Loveland (DPLL) algorithm [16, 15], for computing optimal solutions of a SAT problem with qualitative preferences. In our setting, a qualitative preference is a partially ordered set of literals $\langle S, \prec \rangle$: S is the set of literals that we would like to have satisfied, and $\prec$ is a (strict) partial order on S expressing the relative importance of fulfilling each literal in S. The first procedure is guaranteed to compute all and only the optimal solutions, i.e., models which are not optimal are not even computed. For this result, it is essential that the splitting heuristic of the solver follows the partial order on the expressed preferences [34]: As we already said, imposing such ordering can lead to significant degradation in the performances of the solver, though this is not the case for many applications, see, e.g., [35, 45] in the context of satisfiability planning [42] and Answer Set Programming [30, 31]. However, for each optimal solution this approach needs to retain a formula that prunes non-optimal solutions: Thus, such a procedure works in polynomial space when searching for a bounded number of optimal solutions, but not in the general case [20]. The second procedure is based on the idea to first compute a model and then check for its optimality: The check consists in determining whether a better model exists and this task is reduced again to a SAT problem. As a consequence, no ordering on the splitting heuristic is needed. Of course, this second procedure may compute models which are not optimal, but is guaranteed to work in polynomial space given there is no need to retain formulas indefinitely. The solving procedure for finding one optimal solution has been presented in [19].

We then show how qualitative preferences on formulas and quantitative preferences on literals or formulas can be reduced to the basic framework of qualitative preferences on literals: This allows us to use our procedures also in these extended settings and, further, for solving problems with mixed qualitative and quantitative preferences. Our procedures compute "complete" set of optimal solutions, and different complete sets of optimal solutions may exist: We also present alternatives for computing such sets. We finally comment on the implementation of the two procedures on top of state-of-the-art satisfiability solvers, like MINISAT [24], and discuss related work.

The paper is structured as follows. In Section 2 we review the basic definitions and terminology about qualitative preferences on literals. In Section 3 we present the two procedures for computing optimal models, while how to deal with other forms of

preferences and with other concepts of complete sets of optimal models is showed in Section 4. Section 5 discusses implementation and related work, and we conclude the paper in Section 6.

2 Satisfiability and Qualitative Preferences

Consider a finite set P of *variables*. A *literal* is a variable x or its negation $\neg x$. A *formula* is either a variable or a finite combination of formulas using the n-ary connectives $\wedge, \vee$ for conjunction and disjunction ($n \geq 0$), and the unary connective $\neg$ for negation. We use the symbols $\bot$ and $\top$ to denote the empty disjunction and conjunction, respectively. If l is a literal, we write $\overline{l}$ for $\neg l$ and we assume $\overline{\overline{x}} = x$. This notation is extended to sets S of literals, i.e., $\overline{S} = \{\overline{l} : l \in S\}$.

Formulas are used to express hard constraints that have to be satisfied. For example, given the 4 variables *Fish*, *Meat*, *RedWine*, *WhiteWine*, the formula

$$(\overline{\mathit{Fish}} \vee \overline{\mathit{Meat}}) \wedge (\overline{\mathit{RedWine}} \vee \overline{\mathit{WhiteWine}}) \tag{1}$$

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

An *assignment* μ 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 literal l if $l \in \mu$,
- a disjunction $(\varphi_1 \vee \ldots \vee \varphi_n)$ ($n \geq 0$) if and only if μ satisfies at least one disjunct φ_i with $1 \leq i \leq n$,
- a conjunction $(\varphi_1 \wedge \ldots \wedge \varphi_n)$ ($n \geq 0$) if and only if μ satisfies all the φ_i with $1 \leq i \leq n$,
- the negation of a formula $\neg\psi$ if and only if μ does not satisfy ψ.

A *model* of a formula φ is an assignment satisfying φ. A formula φ *entails* a formula ψ ($\varphi \models \psi$) if the models of φ are a subset of the models of ψ. For instance, (1) has 9 models. In the following, we represent a total assignment with the set of variables assigned to true in it. For instance, $\{\mathit{Fish}, \mathit{WhiteWine}\}$ represents the total assignment 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 $\langle S, \prec \rangle$ where

- S is a set of literals, called the *set of preferences*: Intuitively, S represents the set of literals that we would like to have satisfied; and
- $\prec$ is a (strict) partial order on S: Intuitively, $l \prec l'$ models the fact that we prefer l to l'.

For example,

$$\{\mathit{Fish}, \mathit{Meat}, \overline{\mathit{RedWine}}\}, \{\mathit{Fish} \prec \mathit{Meat}\} \tag{2}$$

models the case in which we prefer to have both fish and meat, and avoid red wine; in the case in which it is not possible to have both fish and meat, we prefer to have fish over having meat.

A qualitative preference $\langle S, \prec \rangle$ on literals can be extended to the set of total assignments as follows [49]: Given two total assignments μ and μ', we say that μ *is preferred to* μ' ($\mu \prec \mu'$) if and only if[1]

1. there exists a literal $l \in S$ with $l \in \mu$ and $\bar{l} \in \mu'$; 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 $l \prec l'$.

From the definition, it is clear that for any two total assignments μ and μ':

1. If $S \cap \mu = S \cap \mu'$ then $\mu \not\prec \mu'$: In particular, if the set S of preferences is empty, every model is optimal.
2. If $S \cap \mu' \subset S \cap \mu$ then $\mu \prec \mu'$: Every optimal model has a maximal intersection with S. In the case $\prec$ is empty, every model with a maximal intersection with S is optimal.

$\langle S, \prec \rangle$ induces a partial order on the set of total assignments, as stated by the following theorem.

Theorem 1. *Let $\langle S, \prec \rangle$ be a qualitative preference on literals. The relation $\prec$ extended to the set of total assignments is a partial order.*

This theorem has been presented and proved as Theorem 7 in [49] and Theorem 1 in [20].

A model μ of a formula φ is *optimal* if it is a minimal element of the partially ordered set of models of φ. A model μ *dominates* a model μ' if $\mu \prec \mu'$. For instance, considering the qualitative preference (2), the formula (1) has only two optimal models, i.e., $\{Fish\}$ and $\{Fish, WhiteWine\}$. We call $\{\{Fish\}, \{Fish, WhiteWine\}\}$ a complete, or "P-complete", set of optimal models.

3 Computing all optimal solutions in a SAT problem with preferences

Given a formula φ and a preference, we now show how it is possible to compute all optimal models of φ by extending the famous DPLL procedure [16, 15]. In principle, we could use any complete backtrack-search algorithm that can find all satisfying assignment of φ. For presenting our algorithms, we have chosen DPLL for clarity. However, DPLL does not directly handle arbitrary formulas, but finite sets of clauses, where a *clause* is a finite set of literals to be interpreted disjunctively. This is not a limitation because of well known clause form transformation procedures (see, e.g., [59, 51, 37]). In the following, we will continuously switch between formulas and sets of clauses, intuitively meaning the same thing. We remind that deciding whether a formula belongs to an optimal solution is in Σ_p^2 (see, e.g., Theorem 14 in [49]).

[1] It is easy to see that in the case in which the partial order is empty, our definition corresponds to the standard Pareto's optimality, while, in the case in which the partial order is not empty, it corresponds to the "Inter criterial Pareto Optimality" as defined in [22].

3.1 Computing all optimal solutions by pruning non-optimal models

Consider a formula φ and a preference $\langle S, \prec \rangle$. The problem of computing all optimal models of φ wrt $\langle S, \prec \rangle$ can be solved by

1. determining and printing an optimal model μ of φ by imposing an ordering on the splitting heuristic, as in, e.g., [34];
2. adding to the input formula a new formula which prunes the models which are dominated by μ; and
3. returning FALSE in order to continue the search for other optimal models.

Crucial for the above procedure is a condition which enables us to say which are the total assignments that are dominated by μ (wrt $\langle S, \prec \rangle$). We therefore define a formula whose models are dominated by μ: From our definition such formula is

$$(\vee_{l \in S \cap \mu} \overline{l}) \wedge (\wedge_{l \in S \cap \overline{\mu}} (\overline{l} \vee \vee_{l' \in S \cap \mu, l' \prec l} \overline{l'})). \tag{3}$$

The total assignment μ dominates a total assignment μ' wrt $\langle S, \prec \rangle$ iff μ' satisfies Eq. (3), as stated by the following theorem.

Theorem 2. *Let $\langle S, \prec \rangle$ be a qualitative preference on literals. A total assignment μ dominates a total assignment μ' wrt $\langle S, \prec \rangle$ if and only if μ' satisfies the formula in Eq. (3) wrt $\langle S, \prec \rangle$.*

This theorem has been presented and proved as Theorem 3 in [20].
For example,

1. if $\mu_1 = \{\mathit{Fish}\}$ and $\langle S, \prec \rangle$ is as in (2), then (3) is

   $$(\overline{\mathit{Fish}} \vee \mathit{RedWine}) \wedge (\overline{\mathit{Meat}} \vee \overline{\mathit{Fish}})$$

   which is equivalent to

   $$\overline{\mathit{Fish}} \vee (\mathit{RedWine} \wedge \overline{\mathit{Meat}})$$

   Any total assignment which satisfies $\overline{\mathit{Fish}}$ or $(\mathit{RedWine} \wedge \overline{\mathit{Meat}})$ is dominated by $\{\mathit{Fish}\}$.
2. if $\mu_2 = \{\mathit{Meat}\}$ and $\langle S, \prec \rangle$ is as in (2), then (3) is

   $$(\overline{\mathit{Meat}} \vee \mathit{RedWine}) \wedge \overline{\mathit{Fish}}$$

   Any total assignment which satisfies $\overline{\mathit{Fish}}$ and at least one between *RedWine* and $\overline{\mathit{Meat}}$) is dominated by $\{\mathit{Meat}\}$.

Notice that if μ_1 dominates μ_2 and ψ_1 (resp. ψ_2) is the formula (3) computed for μ_1 (resp. μ_2), then $\psi_2 \models \psi_1$, i.e., the models of ψ_2 are a subset of the models ψ_1: This is a simple consequence of the fact that if $\mu_1 \prec \mu_2$ then μ_1 dominates a superset of the total assignments dominated by μ_2.

As general examples consider the following particular cases:

1. If $S \cap \mu = \emptyset$, then formula (3) is equivalent to the empty disjunction, i.e., FALSE: Indeed, if μ does not satisfy any preference, no assignment is dominated by μ;

$\langle S, \prec \rangle$:= a qualitative preference on literals;
$\psi := \emptyset$;

function nOPT-DLL$_1(\varphi \cup \psi, \mu)$
1 **if** $(\bot \in (\varphi \cup \psi)_\mu)$ **return** FALSE;
2 **if** (μ is total)
3 *Print*$(\mu \cap (P \cup \overline{P}))$;
4 $\psi := \psi \cup$ *Reason*(μ);
5 **return** FALSE;
6 **if** $(\{l\} \in (\varphi \cup \psi)_\mu)$ **return** nOPT-DLL$_1(\varphi \cup \psi, \mu \cup \{l\})$;
7 $l :=$ *ChooseLiteral*$_1(\varphi \cup \psi, \mu)$;
8 **return** nOPT-DLL$_1(\varphi \cup \psi, \mu \cup \{l\})$ **or**
nOPT-DLL$_1(\varphi \cup \psi, \mu \cup \{\bar{l}\})$.

Fig. 1. The algorithm of nOPT-DLL$_1$.

2. If $S \subseteq \mu$, then formula (3) is equivalent to

$$\vee_{l \in S} \bar{l}.$$

Each assignment which does not satisfy all the preferences is dominated by μ;
3. If $\prec = \emptyset$, then formula (3) is equivalent to

$$\vee_{l \in S \cap \mu} \bar{l} \wedge \wedge_{l \in S \cap \overline{\mu}} \bar{l}.$$

Each assignment satisfying a strict subset of the set of preferences satisfied by μ, is dominated by μ.

Given a system *SYS* for computing an optimal model of a formula φ wrt a preference $\langle S, \prec \rangle$, Theorem 2 allows us to compute a complete set of optimal models using *SYS* as a black box, according to the following procedure:

1. *SYS* is invoked with input φ and the preference $\langle S, \prec \rangle$;
2. If *SYS* returns that φ is unsatisfiable then all optimal models have been already computed and the procedure stops;
3. If *SYS* returns an optimal model μ, the negation of the formula (3) is computed and added to φ;
4. Go to Step 1.

The resulting procedure, which generalizes the DPLL-based procedure presented in [34] for computing one optimal model, is represented in Figure 1 and returns a complete set of optimal models.

In the figure,[2]

- it is assumed that the input formula φ is a set of clauses; μ is an assignment; ψ is an initially empty set of clauses;

[2] We assume left-associativity for the **or** at line 8 of the procedure.

- $(\varphi \cup \psi)_\mu$ 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 $C \setminus \{\bar{l} : l \in \mu\}$;
- *Reason*(μ) returns a set of clauses equivalent to the negation of (3): Let P be the signature of φ, *Reason*(μ) is a finite set of clauses —possibly in a signature P' extending P— such that
  1. for each total assignment μ satisfying the negation of (3), there exists one assignment μ' in P' extending μ and satisfying *Reason*(μ);
  2. for each total assignment μ' in P' satisfying *Reason*(μ), the restriction of μ' to P satisfies the negation of (3).

  Such a set of clauses can be computed starting from the negation of (3) using the already mentioned clause form transformations [59, 51, 37].
- *ChooseLiteral*$_1(\varphi \cup \psi, \mu)$ returns an unassigned literal l such that
  - if there exists a literal in S which is not assigned by μ, then each literal l' with $l' \prec l$ has to be assigned by μ, and
  - is an arbitrary literal occurring in $\varphi \cup \psi$, otherwise.

nOPT-DLL$_1$ has to be invoked with φ and μ set to the input formula and the empty set, respectively. It is easy to see that if there are no preferences, the computation performed by nOPT-DLL$_1$ for computing the first (optimal) model is the same as the one performed by DPLL. nOPT-DLL$_1$ prints all and only the optimal models of φ wrt $\langle S, \prec \rangle$, as stated by the following theorem.

Theorem 3. *Let $\langle S, \prec \rangle$ be a qualitative preference on literals. Let φ be a set of clauses. nOPT-DLL$_1(\varphi, \emptyset)$ returns all and only the optimal models for φ wrt $\langle S, \prec \rangle$.*

This theorem has been presented and proved as Theorem 4 in [20]. Notice that if the number of optimal models is polynomial, so is the space requirement of nOPT-DLL$_1$, and it is easy to modify nOPT-DLL$_1$ by introducing a bound on the number of optimal models to be generated. However, even in practice we did not experience many problems due to excessive space requirements on most of the benchmarks and applications we considered, in general there can be exponentially many optimal models and thus nOPT-DLL$_1$ is not ensured to run in polynomial space.

3.2 Computing all optimal solutions via generate-and-test

As we have already anticipated, nOPT-DLL$_1$ has two drawbacks: In principle, it may have exponential space requirements and it imposes an ordering on the splitting heuristic. In order to be sure to run in polynomial space, a procedure for computing all optimal models of a formula φ has to give up the idea of storing information about the previously computed optimal models. However, this has the consequence that we are no longer ensured that a generated model is also optimal, even by designing the splitting heuristic as in nOPT-DLL$_1$. Thus, it is necessary to test for optimality of a generated model μ, and this test has to be performed by taking into account only the model μ, the formula φ and the preference $\langle S, \prec \rangle$. In other words, we need a condition enabling

us to determine if a model μ of a formula φ is optimal wrt $\langle S, \prec \rangle$, i.e., if there exists another model μ' of φ with $\mu' \prec \mu$. The *preference formula for μ (wrt $\langle S, \prec \rangle$)* is

$$(\vee_{l \in S \cap \overline{\mu}} l) \wedge (\wedge_{l' \in S \cap \mu} (\vee_{l \in S \cap \overline{\mu}, l \prec l'} l \vee l')). \tag{4}$$

A total assignment μ' is preferred to μ wrt $\langle S, \prec \rangle$ iff μ' satisfies (4), as stated by the following theorem.

Theorem 4. *Let $\langle S, \prec \rangle$ be a qualitative preference on literals. A total assignment μ' is preferred to a total assignment μ wrt $\langle S, \prec \rangle$ if and only if μ' satisfies the preference formula for μ wrt $\langle S, \prec \rangle$.*

The theorem can be proved from the definition of dominance between total assignments, as for Theorem 2.

For example,

1. if $\mu_1 = \{Fish\}$ and $\langle S, \prec \rangle$ is as in (2), then the preference formula for μ_1 is

   $$(Meat \wedge Fish \wedge \overline{RedWine}).$$

   The two total assignments satisfying $(Meat \wedge Fish \wedge \overline{RedWine})$ are preferred to μ_1.
2. if $\mu_2 = \{Meat\}$ and $\langle S, \prec \rangle$ is as in (2), then the preference formula for μ_2 is

   $$Fish \wedge ((Meat \vee Fish) \wedge \overline{RedWine})$$

   equivalent to $Fish \wedge \overline{RedWine}$: The four total assignments satisfying $Fish \wedge \overline{RedWine}$ are preferred to μ_2.

Notice that since $\mu_1 \prec \mu_2$, the preference formula (4) for μ_1 entails the preference formula for μ_2: The set of total assignments which are preferred to μ_1 is a subset of the set of total assignments which are preferred to μ_2. As general examples consider the following particular cases:

1. If $S \subseteq \mu$ (e.g., because $S = \emptyset$), then (4) is equivalent to $\bot$ meaning that there is no assignment which is preferred to μ, i.e., that μ is optimal.
2. If $\prec = \emptyset$, then (4) becomes

   $$(\vee_{l \in S \cap \overline{\mu}} l) \wedge \wedge_{l' \in S \cap \mu} l',$$

   meaning that a total assignment μ' is preferred to μ if and only if $\mu \cap S \subset \mu' \cap S$.

Thanks to Theorem 4 we can check if a model μ of a formula φ is optimal by checking the satisfiability of φ and the preference formula ψ for μ. We can thus easily generate a complete set of optimal solutions by modifying DPLL in order to

1. compute a (not necessarily optimal) model μ of φ;
2. test if μ is optimal, in which case μ is printed; and
3. return FALSE in order to continue the search for other (possibly optimal) models.

In Figure 2 we maintain the same assumptions and notations used in Figure 1, extended with:

$\langle S, \prec \rangle$:= a qualitative preference on literals;
$\psi := \emptyset$;

function nOPT-DLL$_2(\varphi \cup \psi, \mu)$
1 **if** $(\bot \in (\varphi \cup \psi)_\mu)$ **return** FALSE;
2 **if** (μ is total)
3 $\quad \psi :=$ *NewReason*(μ, ψ);
4 $\quad$ **if** ($UNSAT(\varphi \cup$ *Prefwff*$(\mu))$) *Print*(μ);
5 $\quad$ **return** FALSE;
6 **if** $(\{l\} \in (\varphi \cup \psi)_\mu)$ **return** nOPT-DLL$_2(\varphi \cup \psi, \mu \cup \{l\})$;
7 $l :=$ *ChooseLiteral*$_2(\varphi \cup \psi, \mu)$;
8 **return** nOPT-DLL$_2(\varphi \cup \psi, \mu \cup \{l\})$ **or**
$\quad\quad\quad\quad$ nOPT-DLL$_2(\varphi \cup \psi, \mu \cup \{\bar{l}\})$.

Fig. 2. The algorithm of nOPT-DLL$_2$.

- *NewReason*(μ, ψ) returns a subset of the clauses in *Reason*$(\mu) \cup \psi$;
- *Prefwff*(μ) returns the set of clauses equivalent to the preference formula for μ;
- $UNSAT(\varphi \cup$ *Prefwff*$(\mu))$ is an invocation to a SAT solver returning TRUE if the input set of clauses is unsatisfiable, and FALSE otherwise;
- *ChooseLiteral*$_2(\varphi \cup \psi, \mu)$ returns an arbitrary unassigned literal l.

The nOPT-DLL$_2$ algorithm in Figure 2 has to be invoked with φ and μ set to the input formula and the empty set, respectively. It is easy to see that if there are no preferences, the computation performed by nOPT-DLL$_2$ for computing the first (optimal) model is the same as the one performed by DPLL. nOPT-DLL$_2$ prints all optimal models, as stated by the following theorem.

Theorem 5. *Let $\langle S, \prec \rangle$ be a qualitative preference on literals. Let φ be a set of clauses. nOPT-DLL$_2(\varphi, \emptyset)$ prints all optimal models for φ.*

Proof. The theorem follows from:

1. The correctness and completeness of DPLL as models enumerator, and
2. the correctness of the $UNSAT(\varphi \cup$ *Prefwff*$(\mu))$ function.

The first point is proved in, e.g., [33], while the second point holds by definition of *Prefwff*(μ).

As an optimization of the above procedure, when a model μ is computed, we can add to the input formula a subset of the clauses in *Reason*$(\mu) \cup \psi$ at line 3 of nOPT-DLL$_2$, e.g., the ones corresponding to the negation of (3). The goal of these clauses is to prune the models of μ which are guaranteed not to be optimal because dominated by μ: They are not needed for the correctness of the procedure and can be removed at any time.

No formula is needed to be retained indefinitely in this algorithm, thus if *NewReason* computes a polynomial number of clauses, then nOPT-DLL$_2$ is guaranteed to work in polynomial space.

4 Extensions

In this section we describe two extensions of our initial setting of (i) computing P-complete sets of optimal solutions (ii) of SAT problems with qualitative preferences on literals. Subsection 4.1 shows how to deal with an alternative concept of "complete" set of optimal models, while Subsection 4.2 shows how problems with quantitative preferences, or with qualitative preferences defined on formulas, or with mixed qualitative/quantitative preferences, can be captured by our basic setting (ii). This last part has been already presented in [20].

4.1 Alternative complete sets of optimal models

The two procedures we have presented compute what we have called a P-complete set of optimal models: Considering the formula (1) and the preference (2), a P-complete set of optimal models for this problem is $\{\{\mathit{Fish}\}, \{\mathit{Fish}, \mathit{WhiteWine}\}\}$. Given the task to compute all optimal models, it can be the case that it is not interesting to distinguish between the two optimal models $\{\mathit{Fish}\}$ and $\{\mathit{Fish}, \mathit{WhiteWine}\}$ given that they differ only for the truth value assigned to *WhiteWine*, and the set of preferences says nothing about the desired truth value assigned to *WhiteWine*.

More formally, consider a formula φ, a qualitative preference $\langle S, \prec \rangle$ and a set Γ of optimal models of φ, Γ is *P-complete* if it contains all the optimal models of φ: Indeed, there can be only one P-complete set of optimal models. An alternative form, we call *S-complete*, considers that for each optimal model μ of φ there exists exactly one model μ' in Γ with $\mu \cap S = \mu' \cap S$. Intuitively, any two models in Γ have to differently evaluate some of the literals in S. There can be more than one S-complete sets of optimal models, for example, the sets of models $\{\{\mathit{Fish}\}\}$ and $\{\{\mathit{Fish}, \mathit{WhiteWine}\}\}$ are both S-complete for (1), assuming $\langle S, \prec \rangle$ is (2).

Some updates are needed in order to find an S-complete set of optimal models. First, the definition of dominance: A model μ dominates a model μ' if either $\mu \prec \mu'$ or $\mu \cap S = \mu' \cap S$.

Then, the formula (3) that defines which are the total assignments dominated by a given total assignment μ becomes

$$\wedge_{l \in S \cap \overline{\mu}} \, (\, \overline{l} \vee \vee_{l' \in S \cap \mu, l' \prec l} \, \overline{l'} \,). \tag{5}$$

As general examples consider the following particular cases:

1. If $S \subseteq \mu$ (e.g., because $S = \emptyset$), then (5) is equivalent to $\top$ meaning that all assignments are dominated by μ, i.e. $\{\mu\}$ is S-complete.
2. If $\prec = \emptyset$, then (5) becomes
$$(\wedge_{l \in S \cap \overline{\mu}} \, \overline{l})$$
meaning that a total assignment μ is preferred to μ' if and only if $\mu' \cap S \subseteq \mu \cap S$.

In our example,

1. if $\mu_1 = \{\mathit{Fish}\}$ and $\langle S, \prec \rangle$ is as in (2), then (5) is
$$(\overline{\mathit{Meat}} \vee \overline{\mathit{Fish}}).$$
Any total assignment which satisfies $\overline{\mathit{Fish}}$ or $\overline{\mathit{Meat}}$ is dominated by $\{\mathit{Fish}\}$.

2. if $\mu_2 = \{\mathit{Meat}\}$ and $\langle S, \prec \rangle$ is as in (2), then (5) is

$$\overline{\mathit{Fish}}.$$

Any total assignment which satisfies $\overline{\mathit{Fish}}$ is dominated by $\{\mathit{Meat}\}$.

Similar changes hold for the preference formula (4). Moreover, in the second algorithm the generation of an S-complete set of models is more difficult than computing a P-complete set. Indeed, we cannot have two optimal models which satisfy the same set of preferences. If μ is an already determined optimal model, in order to avoid the generation of models satisfying the preferences of μ (i.e., the literals in $S \cap \mu$) we could add a clause containing the negation of the literals in $S \cap \mu$, as soon as μ is determined to be optimal. However, the resulting procedure is no longer guaranteed to run in polynomial space. A simple solution that guarantees the polynomial space requirement is to force DPLL in order to first split on the literals l such that either $l \in S$ or $\bar{l} \in S$: When a model μ is found, regardless of whether it is optimal or not, a set of clauses corresponding to the negation of (5) is added to the input formula. The goal of these clauses is to force the procedure to backtrack up to one of the literals in S, in this way avoiding the generation of models which satisfy the same preferences as μ. Once the procedure backtracks to one such literal, these clauses can be removed, thus guaranteeing the polynomial space property of the procedure. Thus, the definition of *ChooseLiteral*$_2$ needs to be updated in this case: *ChooseLiteral*$_2$ returns (i) a literal l with $l \in S$ or $\bar{l} \in S$ if not all the literals in S are assigned by μ, and (ii) an arbitrary unassigned literal otherwise.

All theorems and results can be restated in terms of S-complete set of models.

4.2 Quantitative and Qualitative Preferences on Formulas and their mixing

Quantitative Preferences on literals. Given a set of preferences S and a formula ψ, if it is not possible to satisfy both S and ψ, an alternative approach to model the relative importance of the preferences in S is to define a function $c : S \mapsto \mathbb{N}^+$: Intuitively, $c(l)$ is the reward for satisfying $l \in S$. A pair $\langle S, c \rangle$ is a *quantitative preference* and a model μ of ψ is *optimal* if it maximizes the *objective function* defined as[3]

$$\sum_{l \in S \cap \mu} c(l). \tag{6}$$

Consider a quantitative preference $\langle S', c \rangle$ and a satisfiable set of clauses φ'.

The problem of finding a complete set of optimal models of φ' wrt $\langle S', c \rangle$ can be solved again using nOPT-DLL$_1$ or nOPT-DLL$_2$ as core engine. The basic idea is to encode the value of the objective function (6) as a sequence of bits $b_{n-1}, \ldots, b_0$ and then consider the qualitative preference $\langle \{b_{n-1}, \ldots, b_0\}, \{b_i \prec b_j : 0 \leq j < i < n\} \rangle$. In more details, let *adder*(S', c) be a set of clauses such that:

[3] Assuming we want $c(l) < 0$ for some $l \in S$, we can replace l with $\bar{l}$ in S and define $c(\bar{l}) = -c(l)$: The set of optimal models does not change. Given $\langle S, c \rangle$ and assuming we are interested in minimizing the objective function (6), we can consider the quantitative preference $\langle \overline{S}, c' \rangle$ with $c'(l) = c(\bar{l})$, and then look for a model maximizing $\sum_{l \in \overline{S} \cap \mu} c'(l)$.

1. If $n = \lceil log_2(\sum_{l \in S'} c(l)+1) \rceil$, *adder*$(S', c)$ contains n new variables $b_{n-1}, \ldots, b_0$; and
2. A total assignment μ satisfies φ' iff there exists a unique total assignment μ' to the variables in φ' and in *adder*(S', c) such that
   (a) μ' extends μ and satisfies both φ' and *adder*(S', c), and
   (b) $\sum_{l \in S' \cap \mu} c(l) = \sum_{i=0}^{n-1} \mu'(b_i) \times 2^i$, where $\mu'(b_i)$ is 1 if $b_i \in \mu'$, and is 0 otherwise.

If the above conditions are satisfied, we say that *adder*(S', c) is a *Boolean encoding of* $\langle S', c \rangle$ *with output* $b_{n-1}, \ldots, b_0$. *adder*(S', c) can be realized in polynomial time in many ways, see, e.g., [61]. In the above hypotheses, if

1. φ is the set of clauses in φ' or in *adder*(S', c), and
2. $\langle S, \prec \rangle$ is the qualitative preference $\langle \{b_{n-1}, \ldots, b_0\}, \{b_i \prec b_j : 0 \leq j < i < n\} \rangle$

then nOPT-DLL$_1$ and nOPT-DLL$_2$ return a complete set of optimal solutions of φ' wrt $\langle S', c \rangle$. The following theorem formally states this result.

Theorem 6. *Let φ' be a set of clauses and let $\langle S', c \rangle$ be a quantitative preference on literals. Let adder(S', c) be a Boolean encoding of $\langle S', c \rangle$ with output $b_{n-1}, \ldots, b_0$. If*

1. *φ is the set of clauses in φ' or in adder(S', c),*
2. *$\langle S, \prec \rangle$ is the qualitative preference $\langle \{b_{n-1}, \ldots, b_0\}, \{b_i \prec b_j : 0 \leq j < i < n\} \rangle$, and*
3. *M is the set of models of φ printed by nOPT-DLL$_1$ in Figure 1, or by nOPT-DLL$_2$ in Figure 2,*

then the models in M, restricted to the signature of φ', are all the optimal models of φ' wrt $\langle S', c \rangle$.

This theorem has been presented and proved as Theorem 5 in [20].

Qualitative and Quantitative Preferences on Formulas. So far, a preference is a literal, and we have seen how it is possible to use DPLL to find optimal models wrt both qualitative and quantitative preferences on literals. We now show that the hypothesis that preferences are literals can be waved, i.e., that it is possible to generalize the previous concepts and results from literals to arbitrary formulas. The basic idea is to introduce definitions [59] or "names" [51] for the formulas at hand.

First, we define a *qualitative preference on formulas* to be a pair $\langle S, \prec \rangle$ where S is a finite set of formulas and $\prec$ is a (strict) partial order on S. The set S of preferences does not need to be consistent. Then, as in Section 2, the partial order on S induces a partial order on the sets of total assignments according to which, if μ and μ' are two total assignments, $\mu \prec \mu'$ if and only if

1. there exists a formula $\psi \in S$ satisfied by μ and not by μ'; and
2. for each formula $\psi' \in S$ satisfied by μ' and not by μ, there exists a formula $\psi \in S$ satisfied by μ and not by μ' such that $\psi \prec \psi'$.

It is easy to see that if the formulas in S are literals, then the above definition coincides with the one given in Section 2. It is also straightforward to generalize the result of Theorem 1 saying that the if $\langle S, \prec \rangle$ is a qualitative preference on formulas, the relation $\prec$ extended to the set of total assignments is a partial order.

A model μ of a formula ψ is *optimal wrt a qualitative preference on formulas* $\langle S, \prec \rangle$ if μ is a minimal element of the partial order on the models of ψ.

Consider a formula ψ and a qualitative preference on formulas $\langle S, \prec \rangle$. Instead of ψ and $\langle S, \prec \rangle$ we can consider

1. the qualitative preference on literals $\langle L_s, \prec_S \rangle$, where
   - L_S has a newly introduced variable x_α for each formula $\alpha \in S$, and
   - $x_\alpha \prec_S x_\beta$ if and only if $\alpha \prec \beta$; and
2. the formula
$$\psi \wedge \wedge_{\alpha \in S}(x_\alpha \equiv \alpha). \tag{7}$$

Then, if
$$\mu_S = \mu \cup \{x_\alpha : \alpha \in S, \mu \models \alpha\} \cup \{\neg x_\alpha : \alpha \in S, \mu \not\models \alpha\}$$
it is straightforward to see that a model μ of ψ is optimal wrt the qualitative preference on formulas $\langle S, \prec \rangle$ iff μ_S is an optimal model of (7) wrt the qualitative preference on literals $\langle L_S, \prec_S \rangle$. It is also easy to see that (7) can be simplified to
$$\psi \wedge \wedge_{\alpha \in S}(\neg x_\alpha \vee \alpha) \tag{8}$$
and we obtain again the desired correspondence between the models of ψ and (8).

Introducing definitions [59] or "names" [51] for the formulas in the preferences allows us also to reduce quantitative preferences on formulas (defined in the obvious way) to qualitative preferences on literals. Further, it allows us to use nOPT-DLL$_1$ and nOPT-DLL$_2$ as core engines for computing optimal models of ψ given a qualitative/quantitative preference on formulas.

An advantage of reducing quantitative preferences to qualitative ones is that it makes also possible to mix the two, e.g., we can ask (we assume $b_{n-1}, \ldots, b_0$ to be the output bits of *adder*(S', c)):

1. Which among the optimal models according to a qualitative preference $\langle S, \prec \rangle$ are optimal according to a quantitative preference $\langle S', c \rangle$: Such assignments correspond to the optimal models of $\psi \wedge$ *adder*(S', c) wrt the qualitative preference
$$\langle S \cup \{b_{n-1}, \ldots, b_0\}, \prec \cup \{b_i \prec b_j : 0 \leq j < i < n\} \cup \{\alpha \prec b_i : \alpha \in S, 0 \leq i < n\} \rangle.$$
   This preference, e.g., forces nOPT-DLL$_1$ to consider first $\langle S, \prec \rangle$ and then $\langle S', c \rangle$,
2. or which among the optimal models according to a quantitative preference $\langle S', c \rangle$, are optimal according to a qualitative preference $\langle S, \prec \rangle$: Such assignments correspond to the optimal models of $\psi \wedge$ *adder*(S', c) wrt the qualitative preference
$$\langle S \cup \{b_{n-1}, \ldots, b_0\}, \prec \cup \{b_i \prec b_j : 0 \leq j < i < n\} \cup \{b_i \prec \alpha : \alpha \in S, 0 \leq i < n\} \rangle.$$

5 Discussion and Related work

Our procedures have been implemented on top of MINISAT [24], the 2005 version, winner of the SAT 2005 competition on the industrial benchmarks category (together with the SAT/CNF minimizer SATELITE [23]). We have used MINISAT as models generator in both algorithms, given it is a CDCL [48, 57, 29] solver and thus satisfies the terms of Section 3. We also rely on MINISAT for the SAT test at line 4 of nOPT-DLL$_2$. Experimental analysis of our procedures for finding both one optimal solution and a complete set of optimal solutions, on both randomly generated and real-world SAT problems, with both qualitative and quantitative preferences, can be found in [19–21].

In the context of SAT and Constraint Satisfaction (CSP) problems with qualitative preferences, the idea of computing "optimal" (according to some given definition) models by modifying the heuristic in order to follow the expressed preferences on literals has been already proposed in [13] for SAT and in [5] for acyclic CP-nets [4]. [13] introduced the idea to compute all optimal models by adding constraints pruning the models dominated by the already computed optimal models: Other works which exploit further techniques to eliminate previously computed solutions in SAT include [46, 54, 41, 40] in the context of symbolic model checking [47]. CP-nets [4] (where CP stands for *Conditional Preference*) are a well-known and powerful method for expressing and graphically representing qualitative preferences. In [5] the authors have presented an algorithm, SEARCHCP, for finding more than one optimal solution. The algorithm, similarly to nOPT-DLL$_1$, follows the given partial order on qualitative preferences, but it is backtracking-free. From the computation point of view, if compared with nOPT-DLL$_1$, SEARCHCP computes also non-optimal models that has to be tested with all previously computed optimal solutions and, if compared with nOPT-DLL$_2$, it does not run in polynomial space when looking for all solutions. Moreover, as far as we know, no related implementation is available. Other differences wrt out work and the underlying formalisms used in [13, 5] for expressing preferences are (i) in the language: Both [13] and [5] allow for expressing preferences on literals, but in these approaches it is not possible to rank the preferences according to a partial order; and (ii) in the semantics: Even considering the case in which preferences are expressed as a consistent set S of literals, the order on models induced by S in [13, 5] is different from our (see [20] for details). On the other hand, [5] can deal with non-Boolean domains. As far as the generate-and-test approach is concerned, the idea of adding a constraint that forces a new solution to be better than the current one has been previously employed in, e.g., [26, 52] in the context of constraint optimization problem and constraint logic programming, respectively.

In the context of ASP, several works have dealt with qualitative preferences: In [49], a similar way, in comparison with our approach, of extending preferences on literals to total assignments is used. In [18], several preference handling approaches, not restricted to ASP, are reviewed and compared. Logic Programs with Ordered Disjunction [6] is an extension of normal logic programs with a connective which allows representing alternative, ranked options for problem solutions in the heads of ASP rules: An implementation based on the SMODELS ASP system [58] is presented in [6, 9]. Answer Set Optimization (ASO) programs [10] are another extension of normal logic program for representing qualitative preferences on rule heads, also allowing for formulas in the

heads. Extensions of ASO are presented in, e.g., [7, 55], allowing for "complex preferences" and aggregates in ASO programs. Another approach for computing preferred answer sets has been followed in [25], where meta-interpreters are used to implement different combinations of ASP and preferential information, i.e., the approaches in [8, 17, 60] on top of the DLV ASP system [43]. Recently, in [28], another framework based on meta-interpreters to various forms of qualitative preferences among answer sets, e.g., inclusion-based minimization or Pareto efficiency, is presented.

In the literature of quantitative SAT and CSP, the kind of problem solved in (6) is also known as Binate Covering Problem [14], recently generalized in [44] to Weighted Boolean Optimization problems. In the context of ASP, quantitative preferences are taken into account in, e.g., [12], for computing weighted solutions, [50] for solving Max-ASP problems, [11] with weak constraints, solving pseudo-Boolean problems with CLASP [29] and ASP under multi-criteria optimization [27].

Similar modeling approaches for reducing preferences on formulas to preferences on literals, by introducing definitions [59] or "names" [51], have been presented in, e.g., [36, 53, 2, 1].

6 Conclusions

In this paper we have presented a complete picture of our work on computing optimal solutions in satisfiability problems with preferences, by reviewing some results and presenting new ones. In particular, we have presented two solving procedures, different forms of preferences, ranging from qualitative on literals to mixed qualitative/quantitative on formulas, for finding two types of complete sets of optimal models.

The system implementing the presented procedures is available at `http://www.star.dist.unige.it/~emanuele/sat&pref/`.

Acknowledgement.. The authors would like to thank Emanuele Di Rosa for the implementation of the system, and Torsten Schaub for useful comments on the topic of the paper.

References

1. Fadi A. Aloul, Arathi Ramani, Igor L. Markov, and Karem A. Sakallah. Generic ILP versus specialized 0-1 ILP: an update. In Lawrence T. Pileggi and Andreas Kuehlmann, editors, *Proc. of the 2002 IEEE/ACM International Conference on Computer-aided Design (ICCAD 2002)*, pages 450–457. ACM, 2002.
2. Leila Amgoud, Claudette Cayrol, and Daniel LeBerre. Comparing arguments using preference ordering for argument-based reasoning. In *Proc. of the 8th International Conference on Tools with Artificial Intelligence (ICTAI 1996)*, pages 400–403. IEEE Computer Society, 1996.
3. Meghyn Bienvenu, Jérôme Lang, and Nic Wilson. From preference logics to preference languages, and back. In Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski, editors, *Proc. of the 12th International Conference on Principles of Knowledge Representation and Reasoning (KR 2010)*. AAAI Press, 2010.
4. 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. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.
5. Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. Preference-based constrained optimization with CP-nets. *Computational Intelligence*, 20(2):137–157, 2004.
6. Gerhard Brewka. Logic programming with ordered disjunction. In Rina Dechter and Richard S. Sutton, editors, *Proc. of the 18th National Conference on Artificial Intelligence (AAAI 2002)*, pages 100–105. AAAI Press / The MIT Press, 2002.
7. Gerhard Brewka. Complex preferences for answer set optimization. In Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, editors, *Proc. of the 9th International Conference on Principles of Knowledge Representation and Reasoning (KR2004)*, pages 213–223. AAAI Press, 2004.
8. Gerhard Brewka and Thomas Eiter. Preferred answer sets for extended logic programs. *Artificial Intelligence*, 109(1-2):297–356, 1999.
9. Gerhard Brewka, Ilkka Niemelä, and Tommi Syrjänen. Logic programs with ordered disjunction. *Computational Intelligence*, 20(2):335–357, 2004.
10. Gerhard Brewka, Ilkka Niemelä, and Miroslaw Truszczynski. Answer set optimization. In Georg Gottlob and Toby Walsh, editors, *Proc. of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 867–872. Morgan Kaufmann, 2003.
11. Francesco Buccafurri, Nicola Leone, and Pasquale Rullo. Strong and weak constraints in disjunctive datalog. In Jürgen Dix, Ulrich Furbach, and Anil Nerode, editors, *Proc. of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 1997)*, volume 1265 of *Lecture Notes in Computer Science*, pages 2–17. Springer, 1997.
12. Duygu Cakmak, Esra Erdem, and Halit Erdogan. Computing weighted solutions in answer set programming. In Esra Erdem, Fangzhen Lin, and Torsten Schaub, editors, *Proc. of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2009)*, volume 5753 of *Lecture Notes in Computer Science*, pages 416–422. Springer, 2009.
13. Thierry Castell, Claudette Cayrol, Michel Cayrol, and Daniel Le Berre. Using the Davis and Putnam procedure for an efficient computation of preferred models. In Wolfgang Wahlster, editor, *Proc. of the 12th European Conference on Artificial Intelligence (ECAI 1996)*, pages 350–354. John Wiley and Sons, Chichester, 1996.
14. Olivier Coudert. On solving covering problems. In Thomas Pennino and Ellen J. Yoffa, editors, *Proc. of the 33rd Conference on Design Automation (DAC 1996)*, pages 197–202. ACM Press, 1996.

15. Martin Davis, George Logemann, and Donald W. Loveland. A machine program for theorem proving. *Communication of ACM*, 5(7):394–397, 1962.
16. Martin Davis and Henry Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–215, 1960.
17. James P. Delgrande and Torsten Schaub. Expressing preferences in default logic. *Artificial Intelligence*, 123(1-2):41–87, 2000.
18. James P. Delgrande, Torsten Schaub, Hans Tompits, and Kewen Wang. A classification and survey of preference handling approaches in nonmonotonic reasoning. *Computational Intelligence*, 20(2):308–334, 2004.
19. Emanuele DiRosa, Enrico Giunchiglia, and Marco Maratea. A new approach for solving satisfiability problems with qualitative preferences. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, and Nikolaos M. Avouris, editors, *Proc. of the 18th European Conference on Artificial Intelligence (ECAI 2008)*, volume 178 of *Frontiers in AI and Applications*, pages 510–514. IOS Press, 2008.
20. Emanuele DiRosa, Enrico Giunchiglia, and Marco Maratea. Solving satisfiability problems with preferences. *Constraints*, 15(4):485–515, 2010.
21. Emanuele DiRosa, Enrico Giunchiglia, and Barry O'Sullivan. Optimal stopping methods for finding high quality solutions to satisfiability problems with preferences. In William C. Chu, W. Eric Wong, Mathew J. Palakal, and Chih-Cheng Hung, editors, *Proc. of the 2011 ACM Symposium on Applied Computing (SAC 2011)*, pages 901–906. ACM, 2011.
22. Jon Doyle and Ulrich Junker. Preferences. Tutotial at the 19th National Conference on Artificial Intelligence (AAAI 2004), 2004.
23. Niklas Eén and Armin Biere. Effective preprocessing in SAT through variable and clause elimination. In Fahiem Bacchus and Toby Walsh, editors, *Proc. of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT 2005)*, volume 3569 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 2005.
24. Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Proc. of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT 2003). Selected Revised Papers*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2004.
25. Thomas Eiter, Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. Computing preferred answer sets by meta-interpretation in answer set programming. *Theory and Practice of Logic Programming*, 3(4-5):463–498, 2003.
26. Marco Gavanelli. Partially ordered constraint optimization problems. In Toby Walsh, editor, *Proc. of the 7th International Conference on Principles and Practice of Constraint Programming (CP 2001)*, volume 2239 of *Lecture Notes in Computer Science*, page 763. Springer, 2001.
27. Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Multi-criteria optimization in answer set programming. In John P. Gallagher and Michael Gelfond, editors, *Technical Communications of the 27th International Conference on Logic Programming (ICLP 2011)*, volume 11 of *LIPIcs*, pages 1–10. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.
28. Martin Gebser, Roland Kaminski, and Torsten Schaub. Complex optimization in answer set programming. *Theory and Practice of Logic Programming*, 11(4-5):821–839, 2011.
29. Martin Gebser, Benjamin Kaufmann, André Neumann, and Torsten Schaub. Conflict-driven answer set solving. In Manuela M. Veloso, editor, *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 386–391, 2007.
30. Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert Kowalski and Kenneth Bowen, editors, *Proc. of the 5th International Conference and Symposium on Logic Programming (ICLP/SLP 1988)*, pages 1070–1080, 1988.

31. Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
32. Ian Gent, Hans Van Maaren, and Toby Walsh, editors. *SAT2000. Satisfiability Research in the Year 2000*. IOS Press, 2000.
33. Enrico Giunchiglia, Fausto Giunchiglia, and Armando Tacchella. SAT-based decision procedures for classical modal logics. *Journal of Automated Reasoning*, 28:143–171, 2002. Reprinted in [32].
34. Enrico Giunchiglia and Marco Maratea. Solving optimization problems with DLL. In Gerhard Brewka, Silvia Coradeschi, Anna Perini, and Paolo Traverso, editors, *Proc. of the 17th European Conference on Artificial Intelligence (ECAI 2006)*, volume 141 of *Frontiers in Artificial Intelligence and Applications*, pages 377–381. IOS Press, 2006.
35. Enrico Giunchiglia, Alessandro Massarotto, and Roberto Sebastiani. Act, and the rest will follow: Exploiting determinism in planning as satisfiability. In Jack Mostow and Chuck Rich, editors, *Proc. of the 15th National Conference on Artificial Intelligence (AAAI 1998)*, pages 948–953. AAAI Press / The MIT Press, 1998.
36. Simon De Givry, Javier Larrosa, Pedro Meseguer, and Thomas Schieux. Solving Max-SAT as weighted CSP. In Francesca Rossi, editor, *Proc. of the 9th International Conference on Principles and Practice of Constraint Programming (CP 2003)*, volume 2833 of *Lecture Notes in Computer Science*, pages 363–376. Springer, 2003.
37. Paul Jackson and Daniel Sheridan. Clause form conversions for Boolean circuits. In Holger H. Hoos and David G. Mitchell, editors, *Proc. of the 7th International Conference on Theory and Applications of Satisfiability Testing (SAT 2004). Revised Selected Papers*, volume 3542 of *Lecture Notes in Computer Science*, pages 183–198. Springer, 2004.
38. 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.
39. Matti Järvisalo and Tommi A. Junttila. Limitations of restricted branching in clause learning. *Constraints*, 14(3):325–356, 2009.
40. HoonSang Jin, HyoJung Han, and Fabio Somenzi. Efficient conflict analysis for finding all satisfying assignments of a Boolean circuit. In Nicolas Halbwachs and Lenore D. Zuck, editors, *Proc. of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2005)*, volume 3440 of *Lecture Notes in Computer Science*, pages 287–300. Springer, 2005.
41. HoonSang Jin and Fabio Somenzi. Prime clauses for fast enumeration of satisfying assignments to Boolean circuits. In William H. Joyner Jr., Grant Martin, and Andrew B. Kahng, editors, *Proc. of the 42nd Design Automation Conference (DAC 2005)*, pages 750–753. ACM, 2005.
42. Henry Kautz and Bart Selman. Planning as satisfiability. In Bernd Neumann, editor, *Proc. of the 10th European Conference on Artificial Intelligence (ECAI 1992)*, pages 359–363. John Wiley and Sons, 1992.
43. Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 7(3):499–562, 2006.
44. Vasco M. Manquinho, João P. Marques Silva, and Jordi Planes. Algorithms for weighted Boolean optimization. In Oliver Kullmann, editor, *Proc. of the 12th International Conference on Theory and Applications of Satisfiability Testing (SAT 2009)*, volume 5584 of *Lecture Notes in Computer Science*, pages 495–508. Springer, 2009.
45. Marco Maratea, Francesco Ricca, and Pierfrancesco Veltri. DLV^{MC}: Enhanced model checking in DLV. In Tomi Janhunen and Ilkka Niemelä, editors, *Proc. of the 12th European Conference on Logics in Artificial Intelligence (JELIA 2010)*, volume 6341 of *Lecture Notes in Computer Science*, pages 365–368. Springer, 2010.

46. Kenneth L. McMillan. Applying SAT methods in unbounded symbolic model checking. In Ed Brinksma and Kim Guldstrand Larsen, editors, *Proc. of the 14th International Conference on Computer Aided Verification (CAV 2002)*, volume 2404 of *Lecture Notes in Computer Science*, pages 250–264. Springer, 2002.
47. K.L. McMillan. *Symbolic Model Checking: an Approach to the State Explosion Problem*. Kluwer Academic Publishers, 1993.
48. David G. Mitchell. A SAT solver Primer. *Bulletin of the EATCS*, 85:112–132, 2005.
49. Davy Van Nieuwenborgh and Dirk Vermeir. Preferred answer sets for ordered logic programs. *Theory and Practice of Logic Programming*, 6(1-2):107–167, 2006.
50. Emilia Oikarinen and Matti Järvisalo. Max-ASP: Maximum satisfiability of answer set programs. In Esra Erdem, Fangzhen Lin, and Torsten Schaub, editors, *Proc. of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2009)*, volume 5753 of *Lecture Notes in Computer Science*, pages 236–249. Springer, 2009.
51. David A. Plaisted and Steven Greenbaum. A structure-preserving clause form translation. *Journal of Symbolic Computation*, 2:293–304, 1986.
52. Steve Prestwich. Three implementation of branch-and-cut in CLP. In *Proc. of the 4th Compulog-Net Workshop on Parallelism and Implementation Technologies*, 1996.
53. Miquel Ramirez and Hector Geffner. Structural relaxations by variable renaming and their compilation for solving MinCostSAT. In Christian Bessiere, editor, *Proc. of the 13th International Conference on Principles and Practice of Constraint Programming (CP 2007)*, volume 4741 of *Lecture Notes in Computer Science*, pages 605–619. Springer, 2007.
54. Kavita Ravi and Fabio Somenzi. Minimal assignments for bounded model checking. In Kurt Jensen and Andreas Podelski, editors, *Proc. of the 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2004)*, volume 2988 of *Lecture Notes in Computer Science*, pages 31–45. Springer, 2004.
55. Emad Saad and Gerhard Brewka. Aggregates in answer set optimization. In James P. Delgrande and Wolfgang Faber, editors, *Proc. of the 11th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2011)*, volume 6645 of *Lecture Notes in Computer Science*, pages 211–216. Springer, 2011.
56. Jörg Siekmann and Graham Wrightson, editors. *Automation of Reasoning: Classical Papers in Computational Logic 1967–1970*, volume 1-2. Springer-Verlag, 1983.
57. João P. Marques Silva, Inês Lynce, and Sharad Malik. Conflict-driven clause learning SAT solvers. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 131–153. IOS Press, 2009.
58. Patrik Simons, Ilkka Niemelä, and Soininen Timo. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1–2):181–234, 2002.
59. G. Tseitin. On the complexity of proofs in propositional logics. *Seminars in Mathematics*, 8, 1970. Reprinted in [56].
60. Kewen Wang, Lizhu Zhou, and Fangzhen Lin. Alternating fixpoint theory for logic programs with priority. In John W. Lloyd, Verónica Dahl, Ulrich Furbach, Manfred Kerber, Kung-Kiu Lau, Catuscia Palamidessi, Luis Moniz Pereira, Yehoshua Sagiv, and Peter J. Stuckey, editors, *Proc. of the 1st International Conference on Computational Logic (CL 2000)*, volume 1861 of *Lecture Notes in Computer Science*, pages 164–178. Springer, 2000.
61. Joost P. Warners. A linear-time transformation of linear inequalities into conjunctive normal form. *Information Processing Letters*, 68(2):63–69, 1998.