Computing all optimal solutions in Satisfiability problems with Preferences

Emanuele Di Rosa, Enrico Giunchiglia, and Marco Maratea

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

Abstract. The problem of finding an optimal solution in a constraint satisfaction problem with preferences has attracted a lot of researchers in Artificial Intelligence in general, and in the constraint programming community in particular. As a consequence, several approaches for expressing and reasoning about satisfiability problems with preferences have been proposed, and viable solutions exist for finding one optimal solution. However, in many cases, it is not desirable to find just one solution. Indeed, it might be desirable to be able to compute more, and possibly all, optimal solutions, e.g., for comparatively evaluate them on the basis of other criteria not captured by the preferences.
In this paper we present a procedure for computing all optimal solutions of a satisfiability problem with preferences. The procedure is guaranteed to compute all and only the optimal solutions, i.e., models which are not optimal are not even computed.

1 Introduction

The problem of finding an optimal solution in a constraint satisfaction problem with qualitative preferences has attracted a lot of researchers in Artificial Intelligence in general, and in the constraint programming community in particular.

As a consequence, several approaches for expressing and reasoning about satisfiability (SAT) problems with preferences have been proposed, and viable solutions exist for finding one optimal solution, see, e.g., [1, 2]. However, in many cases, it is not desirable to find just one solution. Indeed, it might be desirable to be able to compute more, and possibly all, optimal solutions, e.g., for comparatively evaluate them on the basis of other criteria not captured by the preferences. As an example of the practical importance of the issue, from the ILOG web page[1]: ”ILOG CPLEX 11 introduces the solution pool feature, which allows users to consider multiple solutions to a MIP model. In practice, a single, even optimal, solution is not always sufficient, because every aspect of a problem cannot always be perfectly captured in a MIP model. The solution pool feature offers a mechanism for exploring the effects of subjective preferences on the solution space without enforcing them as constraints in the model”.

A simple approach for finding all optimal solutions consists in first enumerating all (non necessarily optimal) solutions, and then eliminating a solution μ if there exists

[1] http://www.ilog.com/products/cplex/news/whatsnew.cfm.

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 [3], in the context of CP-nets, 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.

In this paper we present a procedure for computing all optimal solutions of a SAT problem with qualitative preferences which is guaranteed to compute all and only the optimal solutions, i.e., models which are not optimal are not even computed. In our setting, a qualitative preference is a partially ordered set of literals $S, \prec$: S is the set of literals that we would like to have satisfied, and $\prec$ is partial order on S expressing the relative importance of fulfilling the literals in S. For this result, it is essential that the splitting heuristic follows the partial order on the expressed preferences: Imposing such ordering can lead to significant degradations in the performances of the solver [4], though this has been shown to happen only when the number of preferences is very high (in the order of the number of variables in the problem [2]), and this is not the case for many applications, see, e.g., [5].

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* 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 and only if $l \in \mu$,
- $(\varphi_1 \vee \ldots \vee \varphi_n)$ $(n \geq 0)$ if and only if μ satisfies at least one φ_i with $1 \leq i \leq n$,
- $(\varphi_1 \wedge \ldots \wedge \varphi_n)$ $(n \geq 0)$ if and only if μ satisfies all 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 a total 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 represent a total assignment as the set of variables assigned to true. 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 $S, \prec$ where (i) S is a set of literals, called the *set of preferences*, which represents the set of literals that we would like to have satisfied; and (ii) $\prec$ is a partial order on S: $l \prec l'$ 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 the fish more than the meat.

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', we say that μ *is preferred to* μ' *($\mu \prec \mu'$)* if and only if (i) there exists a literal $l \in S$ with $l \in \mu$ and $\overline{l} \in \mu'$; and (ii) 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 two optimal models, i.e., $\{\mathit{Fish}\}$ and $\{\mathit{Fish}, \mathit{WhiteWine}\}$.

Consider a formula φ, a qualitative preference $S, \prec$ and a set Γ of optimal models of φ. Γ is said to be *complete (wrt $S, \prec$)* if contains all the optimal models of φ. With this notion, there can be only one complete set of optimal models, which, in the case of (1) and the preference (2), is the set $\{\{\mathit{Fish}, \mathit{WhiteWine}\}, \{\mathit{Fish}\}\}$. Other notions of completeness are possible.

3 Computing all optimal solutions

Given a formula φ and a preference, we now show how it is possible to compute a complete set of models of φ by extending the Davis-Logemann-Loveland procedure (DLL) [6] and the procedure in [2] for computing one optimal solution. DLL is the most used decision procedure for checking satisfiability of formulas. However, DLL 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., [7, 8]).

In the following, we will continuously switch between formulas and sets of clauses, intuitively meaning the same thing.

Consider a formula φ and a preference $S, \prec$. An assignment μ dominates *an assignment μ' (wrt $S, \prec$)* if $\mu \prec \mu'$. The problem of computing a complete set of optimal models of φ wrt $S, \prec$ can be solved by considering the following crucial condition which enables us to say which are the assignments that are dominated by μ (wrt $S, \prec$). We therefore define a formula whose models are dominated by μ. Consider a total assignment μ.

1. $n(\mu)$ is the set of preferences not satisfied by μ, i.e., $n(\mu) = S \cap \overline{\mu}$

2. for each $l \in S$, $d(l, \mu)$ is the set of literals in μ which are preferred to l according to $\prec$, i.e., $d(l, \mu) = \{l' : l' \in \mu, l' \prec l \textbf{ or } \overline{l'} \prec l\}$.

Then the *μ-dominates formula* (wrt $S, \prec$) is

$$\neg((\vee_{l \in n(\mu)} (\wedge_{l' \in d(l,\mu)} l' \wedge l)) \vee (\wedge_{l \in \mu, l \in (S \cup \overline{S})} l \wedge (\vee_{l' \in \mu, l' \notin (S \cup \overline{S})} \overline{l'}))) \qquad (3)$$

The total assignment μ dominates a total assignment μ' wrt $S, \prec$ iff μ' satisfies the corresponding μ-dominates formula, as stated by the following theorem.

Theorem 1. *Let $S, \prec$ be a qualitative preference on literals. A total assignment dominates a total assignment μ' wrt $S, \prec$ if and only if μ' satisfies the μ-dominates formula wrt $S, \prec$.*

For example,

1. if $\mu_1 = \{\textit{Fish}\}$ and $S, \prec$ is as in (2), then
   (a) $n(\mu_1)$ is $\{\textit{Meat}\}$, and $d(\textit{Meat}) = \{\textit{Fish}\}$,
   (b) the μ_1-dominates formula is $\neg((\textit{Fish} \wedge \textit{Meat}) \vee (\textit{Fish} \wedge \overline{\textit{Meat}} \wedge \overline{\textit{RedWine}} \wedge \textit{WhiteWine}))$: Any total assignment which does not satisfy $(\textit{Fish} \wedge \textit{Meat})$ or $(\textit{Fish} \wedge \overline{\textit{RedWine}} \wedge \textit{WhiteWine})$ is dominated by $\{\textit{Fish}\}$. Notice that the total assignment $\{\textit{Fish}, \textit{WhiteWine}\}$ is not dominated by $\{\textit{Fish}\}$, as expected.
2. if $\mu_2 = \{\textit{Meat}\}$ and $S, \prec$ is as in (2), then
   (a) $n(\mu_2)$ is $\{\textit{Fish}\}$, and $d(\textit{Fish}) = \emptyset$,
   (b) the μ_2-dominates formula is $\neg(\textit{Fish} \vee (\overline{\textit{Fish}} \wedge \textit{Meat} \wedge \overline{\textit{RedWine}} \wedge \textit{WhiteWine}))$: μ_2 does not dominate the total assignments satisfying *Fish* or $(\overline{\textit{Fish}} \wedge \textit{Meat} \wedge \overline{\textit{RedWine}} \wedge \textit{WhiteWine})$.

Notice that since $\mu_1 \prec \mu_2$, the μ_1-dominates formula is entailed by the μ_2-dominates formula: μ_1 dominates a superset of the total assignments dominated by μ_2.

It is thus possible to generalize the DLL-based procedure presented in [2] for computing an optimal model, in order to return complete sets of optimal models. The resulting procedure is represented in Figure 1. In the figure,

- 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;
- $(\varphi \cup \psi)_\mu$ 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 each clause $C \in \varphi \cup \psi$ with $C \setminus \overline{\mu}$;
- *Reason*(μ) returns a set of clauses equivalent to the negation of the μ-dominates formula.
- *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
  – l is an arbitrary literal occurring in $\varphi \cup \psi$, otherwise.

nOPT-DLL has to be invoked with φ and μ set to the input formula and the empty set respectively. nOPT-DLL prints a complete set of optimal models, as stated by the following theorem.

Theorem 2. *Let $S, \prec$ be a qualitative preference on literals. Let φ be a set of clauses. nOPT-DLL$(\varphi, \emptyset)$ prints a complete set of optimal models for φ.*

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

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

Fig. 1. The algorithm of nOPT-DLL.

4 Conclusions

In this paper we have presented an algorithm for computing all solutions in SAT problems with preferences. The algorithm computes only optimal models, by following the given partial order on preferences, but is not ensured to work in polynomial space. Future work comprises the design of algorithms which are guaranteed to work in polynomial space.

References

1. 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.
2. E. Giunchiglia and M. Maratea. Solving optimization problems with DLL. In *Proc. of 17th European Conference on Artificial Intelligence (ECAI)*, pages 377–381, 2006.
3. 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.
4. 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.
5. Enrico Giunchiglia and Marco Maratea. Planning as satisfiability with preferences. In *In Proc. of 22nd AAAI Conference on Artificial Intelligence*, pages 987–992. AAAI Press, 2007.
6. Martin Davis, George Logemann, and Donald W. Loveland. A machine program for theorem proving. *Communication of ACM*, 5(7):394–397, 1962.
7. G. Tseitin. On the complexity of proofs in propositional logics. *Seminars in Mathematics*, 8, 1970. Reprinted in [9].
8. D.A. Plaisted and S. Greenbaum. A Structure-preserving Clause Form Translation. *Journal of Symbolic Computation*, 2:293–304, 1986.
9. Jörg Siekmann and Graham Wrightson, editors. *Automation of Reasoning: Classical Papers in Computational Logic 1967–1970*, volume 1-2. Springer-Verlag, 1983.