Constraint-based Preferential Optimization[*]

S. Prestwich
University College Cork, Ireland
s.prestwich@cs.ucc.ie

F. Rossi and **K. B. Venable**
University of Padova, Italy
{frossi,kvenable}@math.unipd.it

T. Walsh
NICTA and UNSW, Australia
tw@cse.unsw.edu.au

Abstract

We first show that the optimal and undominated outcomes of an unconstrained (and possibly cyclic) CP-net are the solutions of a set of hard constraints. We then propose a new algorithm for finding the optimal outcomes of a constrained CP-net which makes use of hard constraint solving. Unlike previous algorithms, this new algorithm works even with cyclic CP-nets. In addition, the algorithm is not tied to CP-nets, but can work with any preference formalism which produces a preorder over the outcomes. We also propose an approximation method which weakens the preference ordering induced by the CP-net, returning a larger set of outcomes, but provides a significant computational advantage. Finally, we describe a weighted constraint approach that allows to find good solutions even when optimals do not exist.

Introduction

Preferences and constraints occur together in many problems. For instance, in product configuration, there are physical constraints on what can be built (e.g. a convertible car cannot have a roof rack), as well as the user's preferences (if the car is a convertible, then I prefer a boot rack to no boot rack). Preferences have been widely studied in decision-theoretic AI. However, much less is known about reasoning simultaneously with preferences and constraints, as in the product configuration example above. Constrained preference optimization is a challenging problem as often the most preferred outcome is not feasible, and not all feasible outcomes are equally preferred.

An elegant formalism to represent conditional and qualitative preferences is CP-nets (Boutilier *et al.* 1999; Domshlak & Brafman 2002; Boutilier *et al.* 2004a). These model statements of qualitative and conditional preference which are interpreted under the *ceteris paribus* (that is, "all else being equal") assumption. Preference elicitation in such a framework appears to be natural and intuitive.

In this paper, we consider how to reason with CP-nets in the presence of hard constraints. We will show how to replace preferences by hard constraints (called "optimality constraints") that encode the relevant preference information. We will prove that the solutions of such hard constraints are the optimal solutions of the CP-net. This approach works for any CP-net, even cyclic ones.

The presence of cycles in a CP-net occurs often in real-life situations. In fact, a cycle may denote inconsistent or contradicting information, which may be present when the preferences of several agents are collected, or when a single agent gives a description of his preferences through a sequence of interactions with a system. In fact, in this case it is easy to generate contradicting information. Moreover, a cycle can also be intrinsic in the description of the preferences. For example, it may be natural to give preferences over wine depending on the main course, and vice versa. This may be interpreted as the fact that the two features are equally important but dependent on each other.

A constrained CP-net is a CP-net plus a set of hard constraints, and its optimal outcomes (called "feasible Pareto optimals" in (Boutilier *et al.* 2004b)) are all the outcomes which are feasible and not dominated in the CP-net by any other feasible outcome. We propose an algorithm, based on the optimality constraints, which obtains all the optimal outcomes in a constrained CP-net. Such an algorithm achieves the same task as algorithm Search-CP (Boutilier *et al.* 2004b), but works also for cyclic CP-nets and any other preference formalism which produces a preorder over the outcomes. In well defined cases, it can also avoid dominance testing.

In addition, we propose an approximation method to find a superset of the feasible Pareto optimals, which offers computational savings by weakening the induced preference ordering. This approximation method again uses hard constraint solving, and never needs dominance testing.

Finally, we describe a weighted CSP approach which finds feasible Pareto optimals, or optimals according to the approximation method, with no dominance tests. Moreover, if there are no feasible Pareto optimals, while all existing algorithms stop returning no solution, it finds the outcomes which are best w.r.t. both feasibility and preferences.

CP-nets

CP-nets (Boutilier *et al.* 1999; 2004a) (for Conditional Preference nets) are a graphical model for compactly representing conditional and qualitative preference relations. They

[*]Partially supported by Science Foundation Ireland under Grant No. 00/PI.1/C075, ASI project ARISCOM, and MIUR project AIDA.

exploit conditional preferential independence by decomposing an agent's preferences via the **ceteris paribus** (cp) assumption. Informally, CP-nets are sets of **ceteris paribus (cp)** preference statements. For instance, the statement *"I prefer red wine to white wine if meat is served."* asserts that, given two meals that differ *only* in the kind of wine served *and* both containing meat, the meal with a red wine is preferable to the meal with a white wine.

CP-nets bear some similarity to Bayesian networks. Both utilize directed graphs where each node stands for a domain variable, and assume a set of features $F = \{X_1, \dots, X_n\}$ with finite domains $\mathcal{D}(X_1), \dots, \mathcal{D}(X_n)$. For each feature X_i, each user specifies a set of **parent** features $Pa(X_i)$ that can affect her preferences over the values of X_i. This defines a **dependency graph** in which each node X_i has $Pa(X_i)$ as its immediate predecessors. Given this structural information, the user explicitly specifies her preference over the values of X_i for *each complete assignment* on $Pa(X_i)$. This preference is assumed to take the form of total or partial order over $\mathcal{D}(X_i)$ (Boutilier *et al.* 1999; 2004a). An **acyclic** CP-net is one in which this dependency graph is acyclic. A CP-net need not be acyclic. For example, my preference for entree may depend on the choice of main course, and my preference for a main course may depend on the choice of entree. In fact, one of our contributions here is to permit the user to have a cyclic dependency graph by proposing new algorithms for reasoning with cyclic CP-nets.

Consider a CP-net whose features are A, B, C, and D, with binary domains containing f and $\overline{f}$ if F is the name of the feature, and with the preference statements as follows: $a \succ \overline{a}$, $b \succ \overline{b}$, $(a \wedge b) \vee (\overline{a} \wedge \overline{b}) : c \succ \overline{c}$, $(a \wedge \overline{b}) \vee (\overline{a} \wedge b) : \overline{c} \succ c$, $c : d \succ \overline{d}$, $\overline{c} : \overline{d} \succ d$. Here, statement $a \succ \overline{a}$ represents the unconditional preference for $A = a$ over $A = \overline{a}$, while statement $c : d \succ \overline{d}$ states that $D = d$ is preferred to $D = \overline{d}$, given that $C = c$.

The semantics of CP-nets depends on the notion of a worsening flip. A worsening flip is a change in the value of a variable to a value which is less preferred by the cp statement for that variable. For example, in the CP-net above, passing from $abcd$ to $ab\overline{c}d$ is a worsening flip since c is better than $\overline{c}$ given a and b. We say that one outcome α is **better** than another outcome β (written $\alpha \succ \beta$) iff there is a chain of worsening flips from α to β. This definition induces a preorder over the outcomes.

In general, finding optimal outcomes and testing for optimality in this ordering is NP-hard. However, in acyclic CP-nets, there is only one optimal outcome and this can be found in linear time (Boutilier *et al.* 1999; 2004a). We simply sweep through the CP-net, following the arrows in the dependency graph and assigning at each step the most preferred value in the preference table. For instance, in the CP-net above, we would choose $A = a$ and $B = b$, then $C = c$ and then $D = d$. The optimal outcome is therefore $abcd$.

Determining if one outcome is better than another according to this ordering (a dominance query) is NP-hard even for acyclic CP-nets. Whilst tractable special cases exist, there are also acyclic CP-nets in which there are exponentially long chains of worsening flips between two outcomes. In the CP-net of the example, $\overline{a}b\overline{c}\overline{d}$ is worse than $abcd$.

Optimality and eligibility in CP-nets

Hard constraints are enough to find optimals of a CP-net and to test whether the CP-net has some optimal outcomes. In fact, it is possible to define a set of hard constraints (that we call the "optimality constraints") from a CP-net, such that their solutions are the optimal outcomes of the CP-net.

Consider a set of cp statements N which define a partial order $\succ$ over the elements in the domain of a variable x under the condition φ of an assignment of values to other variables. Then, for each of such statements, the corresponding optimality constraint is $\varphi \rightarrow \bigvee_j x = a_j$, where the a_j's are the undominated elements of the partial order $\succ$. The optimality constraints $opt(N)$ corresponding to the entire set N are the optimality constraints corresponding to all the cp statements in N.

For example, the cp statements $a \succ \overline{a}$ and $(a \wedge b) : c \succ \overline{c}$ map to the hard constraints a and $(a \wedge b) \rightarrow c$ respectively. These constraints map directly to SAT clauses, so SAT is a convenient technology for solving these problems.

It is not hard to show that an outcome is optimal in the ordering induced by a CP-net N iff it is a satisfying assignment for $opt(N)$.

A CP-net is **eligible** iff it has undominated outcomes (eligibility is called consistency in (Domshlak *et al.* 2003)). Even if the preorder induced by a CP-net has cycles, it may still be useful if it is eligible. Trivially, all acyclic CP-nets are eligible as they have an unique optimal (and hence undominated) outcome.

Given the optimality constraints described above, we can thus test eligibility of any (even cyclic) CP-net by testing the consistency of the optimality constraints $opt(N)$. Thus a CP-net N is eligible iff $opt(N)$ is consistent.

When considering CP-nets with Boolean features and total orders over the domains of the features, the approach described in this section coincides with the one in (Brafman & Dimopoulos 2004).

Finding feasible Pareto optimals

As argued in the introduction, many problems contain both preferences and constraints. A constrained CP-net is a CP-net with some additional constraints on assignments. These constraints can be expressed by generic relations on partial assignments or, in the case of binary features, by a set of Boolean clauses. We consider here just hard constraints but many of our results generalize to soft constraints. (Boutilier *et al.* 2004b) gives a natural semantics for constrained CP-nets in which the optimal solutions are feasible and Pareto optimal. Given a set of hard constraints C and a CP-net N, an outcome is **feasible Pareto optimal** iff it is feasible for C and there is no other feasible outcome which is better in the CP-net N.

For acyclic CP-nets, an algorithm to find the feasible Pareto optimal outcomes has been proposed called Search-CP (Boutilier *et al.* 2004b). The algorithm starts with an empty set of solutions to which it adds new non-dominated solutions which are feasible. At each stage, each new candidate is tested against all the solutions generated up that point, and is added to the set only if no existing member dominates it. To find the first optimal outcome, the algorithm uses branch and bound and thus may require an exponential number of steps. Then, to find other optimal outcomes, it needs to perform dominance tests (as many as the number of optimal outcomes already computed).

We now present a new algorithm, which we call Hard-Pareto, for finding the feasible Pareto optimal outcomes. Unlike Search-CP, this new algorithm works even if the CP-net is cyclic. In fact, it works with any preference formalism which produces a preorder over the outcomes. In addition, we will see that this new algorithm offers computational advantages over Search-CP.

Briefly, algorithm Hard-Pareto finds all feasible Pareto optimals by first finding the outcomes which are both feasible and optimal in the CP-net (that is, the solutions of $C \cup opt(N)$). If there are optimals for the CP-net and they are all feasible, then there are no other feasible Pareto optimals and thus the algorithm may stop. Otherwise, it must perform dominance testing over the feasible outcomes.

```
Pseudo-code of Hard-Pareto
1. Input CP-net $N$ and set of hard constraints $C$;
2. $S_{opt} \leftarrow \emptyset$;
3. if ($C$ inconsistent) return $S_{opt}$;
4. else
5.    $S_{opt} \leftarrow sol(C \cup opt(N))$;
6.    if ($S_{opt} = sol(C)$) return $S_{opt}$;
7.    if ($sol(opt(N)) \neq \emptyset$ and $S_{opt} = sol(opt(N))$) return $S_{opt}$;
8.    $S \leftarrow sol(C) - S_{opt}$;
9.    do {Choose $o \in S$;
10.       if ( $\forall o' \in sol(C) - \{o\}, o' \not\succ o$)
11.          then $S_{opt} \leftarrow S_{opt} \cup \{o\}$;
12.       $S \leftarrow S - \{o\}$;}
13.   while ($S \neq \emptyset$);
14.   return $S_{opt}$;
```

Figure 1: Hard-Pareto: given a CP-net and a set of hard constraints, the algorithm finds all the feasible Pareto optimal solutions.

The pseudo-code for the algorithm Hard-Pareto is given in Figure 1. The algorithm takes as input a set of hard constraints C and a CP-net N (line 1). It then initializes the set of feasible Pareto optimal outcomes S_{opt} to the empty set (line 2). If the set of hard constraints is not consistent, i.e. there are no feasible outcomes, then the algorithm returns the empty set (line 3). Otherwise it puts in S_{opt} the set of solutions of the set of hard constraints $C \cup opt(N)$, denoted in line 5 with $sol(C \cup opt(N))$. If such a set of solutions, S_{opt}, is equal either to the set of solutions of the hard constraints alone, $sol(C)$, or to the set of solutions of the optimality constraints alone, $sol(opt(N))$, then it returns S_{opt} (lines 6 and 7). Otherwise the algorithm considers all the feasible outcomes that are dominated in the CP-net (line 8) and for each of them it tests whether the outcome is dominated in the CP-net by any other feasible outcome. It adds to S_{opt} only those outcomes which are not dominated by any other feasible one (lines 9-13). Once all the feasible dominated outcomes have been considered, it returns S_{opt} (line 14).

We will now show that Hard-Pareto is a sound and complete algorithm for finding all feasible Pareto optimal solutions. Clearly, if the set of hard constraints C has no solution then there are no feasible outcomes and, thus, there are no feasible Pareto optimals. In such case the algorithm will stop in line 3. Otherwise, if there are feasible outcomes that are solutions of the optimality constraints of the CP-net, then such outcomes are feasible Pareto optimal outcomes as shown by the following theorem.

Theorem 1 *Let $sol(C)$ be the set of solutions of C, sol(opt(N)) be the set of solutions of $opt(N)$, and $S_{opt} = sol(C \cup opt(N))$. If $S_{opt} \neq \emptyset$ then it contains only feasible Pareto optimal solutions.*

Notice that if the set $sol(C \cup opt(N))$, computed in line 5, is not empty then it contains feasible Pareto optimal solutions, which can thus be found without doing dominance testing.

Moreover, if $sol(C \cup opt(N))$ is equal to the set of solutions $Sol(C)$, i.e. $sol(C) \subseteq sol(opt(N))$, then the feasible outcomes are a subset of the undominated outcomes of the CP-net. In such a situation the set of solutions of C is the set of all feasible Pareto optimals. In fact, since $sol(C) \subseteq sol(opt(N))$ then each feasible is undominated in the CP-net and all other outcomes, even if undominated in the CP-net, are not feasible. If this is the case the algorithm will stop in line 6 and no dominance test is required to find all the feasible Pareto optimal solutions.

A similar reasoning applies when the set $sol(C \cup opt(N))$ is equal to the set of undominated outcomes of the CP-net, that is, $sol(opt(N))$ and such set is not empty. This happens iff all the undominated outcomes of the CP-net are feasible, $sol(opt(N)) \subseteq sol(C)$. In this case we can conclude that the $sol(opt(N))$ is the set of all feasible Pareto optimals. In fact each undominated outcome is also feasible and all other feasible outcomes are dominated by at least one of the undominated outcomes of the CP-net. In this situation the algorithm will stop in line 7, allowing to find the set of feasible Pareto optimals with no dominance test.

Notice, however, that if $sol(opt(N))$ is empty, that is the CP-net has no undominated outcome (e.g. when all the outcomes are in a cycle), then, trivially $sol(opt(N)) = sol(C \cup opt(N))$ but it is not possible to conclude that there are no feasible Pareto optimals. This is the case, for example, if there is a cycle of outcomes such that every other outcome (not belonging to the cycle) is dominated only by an outcome in the cycle, and all the outcomes in the cycle are infeasible. In such a case the feasible outcomes outside the cycle are feasible Pareto optimals.

If $sol(opt(N)) = \emptyset$ or if $sol(C \cup opt(N)) \subset sol(C)$ and $sol(C \cup opt(N)) \subset sol(opt(N))$, in order to find all feasible Pareto optimal outcomes, other than those in $sol(C \cup opt(N))$, all those feasible outcomes for C that are undominated in the CP-net by other feasible outcomes must be found. In this case the algorithm stops in line 14, as the following theorem shows.

Theorem 2 *Let $sol(C)$ be the set of solutions of C, and $sol(opt(N))$ be the set of solutions of $opt(N)$. Let $S_{opt} = sol(C \cup opt(N))$. Then if $sol(C) \neq \emptyset$, $S_{opt} \subset sol(C)$, and ($sol(opt(N)) = \emptyset$ or $S_{opt} \subset sol(opt(N))$), then set $S_{opt} \cup \{o \in sol(C) - S_{opt} | \forall o' \in sol(C). o' \not\succ o\}$ is the set of all feasible Pareto optimal solutions.*

Example 1 Consider the following set of statements defining the cyclic CP-net shown in Figure 2, with three Boolean features A, B, and C: $\{c : a \succ \bar{a}, \bar{c} : \bar{a} \succ a, (a \wedge c) \vee (\bar{a} \wedge \bar{c}) : b \succ \bar{b}, (a \wedge \bar{c}) \vee (\bar{a} \wedge c) : \bar{b} \succ b, b : c \succ \bar{c}, \bar{b} : \bar{c} \succ a\}$.

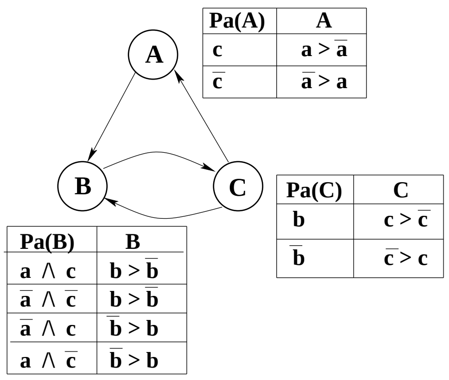
A
B
C
Pa(A) | A
c | $a > \bar{a}$
$\bar{c}$ | $\bar{a} > a$
Pa(C) | C
b | $c > \bar{c}$
$\bar{b}$ | $\bar{c} > c$
Pa(B) | B
$a \wedge c$ | $b > \bar{b}$
$\bar{a} \wedge \bar{c}$ | $b > \bar{b}$
$\bar{a} \wedge c$ | $\bar{b} > b$
$a \wedge \bar{c}$ | $\bar{b} > b$

Figure 2: The dependence graph and preference statements for the CP-net in Example 1.

Figure 3 shows the induced preference ordering of the CP-net in Figure 2. The induced ordering contains cycles, but is eligible since outcome abc is undominated.

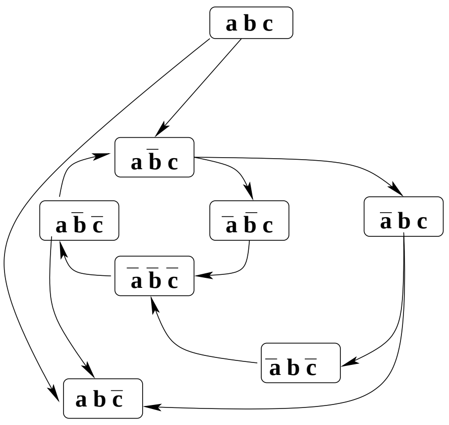

Figure 3: The induced preference ordering of the CP-net in Example 1. As it can be seen, it contains cycles.

The optimality constraints corresponding to the CP-net are: $c \rightarrow a$, $\bar{c} \rightarrow \bar{a}$, $(a \wedge c) \vee (\bar{a} \wedge \bar{c}) \rightarrow b$, $(a \wedge \bar{c}) \vee (\bar{a} \wedge c) \rightarrow \bar{b}$, $b \rightarrow c$, and $\bar{b} \rightarrow \bar{c}$.

Consider the following set of hard constraints $C = \{b \vee c\}$. In this case, $sol(C) \cap sol(opt(N)) = \{abc\} = sol(opt(N))$. Thus, we can conclude that $\{abc\}$ is the only feasible Pareto optimal outcome without doing the dominance test against $\{\bar{a}bc\}$. On this instance Hard-Pareto stops in Line 7.

Consider now a different set of constraints $C' = \{\bar{b}\}$. In this case, all the feasible outcomes, $a\bar{b}c$, $\bar{a}\bar{b}c$, $\bar{a}\bar{b}\bar{c}$, and $a\bar{b}\bar{c}$ are in a cycle. Thus, there is no feasible Pareto optimal. Hard-Pareto will return the empty set at Line 13.

Notice that algorithm Search-CP cannot be applied to these examples since the CP-net is cyclic.

Unlike Search-CP, Hard-Pareto works with cyclic CP-nets. Moreover, if Hard-Pareto stops in line 6 or 7, because all feasible are undominated or all undominated are feasible then the complexity to find all feasible Pareto optimal solutions is the same as finding just one and no dominance testing is required. Search-CP by comparison will always require dominance testing if more than one solution is needed. On the other hand, if no undominated outcomes are feasible, then Hard-Pareto may require dominance testing just to find the first solution, while Search-CP merely performs branch and bound. Wilson shows that it is possible to modify the algorithm Search-CP by replacing each dominance test by an easier test in a different partial order (Wilson 2004). This procedure is guaranteed to compute a non empty subset of the feasible Pareto optimal solutions. However, this approach again requires that the dependence graph be acyclic, whilst Hard-Pareto works with cyclic CP-nets.

We can easily modify algorithm Hard-Pareto to find just one feasible Pareto optimal of a possibly cyclic CP-net. However, it has to be noted that in some cases dominance testing may be necessary for Hard-Pareto even to find one optimal. On the other hand, algorithm Search-CP cannot help us if we look for one optimal of a cyclic CP-net. In the restrictive case of an acyclic CP-net, then Search-CP is more convenient if we look for one optimal only. However, even in this case, if we need more than one optimal, then Hard-Pareto may find many of them, even all, without dominance testing, while Search-CP needs dominance testing for each of them.

Finding approximately optimal outcomes

In some situations, both Search-CP or Hard-Pareto may be computationally too expensive. For example, in online configuration, we may need very fast algorithms to perform constrained preferential optimization. To deal with such situations, we propose an "approximation" of the usual semantics of constrained CP-nets that is less expensive to compute.

We begin by weakening the preference ordering. The main difference with respect to the usual ordering is that we now restrict ourselves to chains of feasible outcomes.

Given a constrained CP-net $\langle N, C \rangle$, outcome O_1 is **approximately better** than outcome O_2 (written $O_1 \succ^* O_2$) iff there is a chain of flips from O_1 to O_2, where each flip is worsening for N and each outcome in the chain satisfies C.

We say that an outcome is **approximately optimal** iff no other outcome is approximately better. For example, assume we have a CP-net with two Boolean features, A and B, and the following CP statements: $a \succ \bar{a}$, $a : b \succ \bar{b}$, $\bar{a} : \bar{b} \succ b$, and the constraint $\bar{a} \vee b$ which rules out $a\bar{b}$. Then, the CP-net orders outcomes as follows: $ab \succ a\bar{b} \succ \bar{a}\bar{b} \succ \bar{a}b$. Both ab and $\bar{a}\bar{b}$ are approximately optimal, but only ab is feasible Pareto optimal. This is true also in general: the set of approximately optimal outcomes is a superset of the set of feasible Pareto optimal outcomes.

This change has some beneficial effects. First, we observe that the $\succ^*$ relation is still a preorder. Second, checking if an outcome is approximately optimal is linear: we merely need to check it is feasible and any flip to a feasible outcome is worsening. By comparison, checking if an outcome is optimal is NP-hard. Third, adding constraints to an acyclic CP-net does not eliminate all the approximately optimal outcomes, unless it eliminates all outcomes. By comparison, adding constraints to a CP-net may make all optimal outcomes infeasible. For example, if we have $O_1 \succ O_2 \succ O_3$ in a CP-net, and the constraints make O_1 infeasible, then O_2 is approximately optimal, but no feasible outcome is optimal.

Theorem 3 *A constrained and acyclic CP-net either has no feasible outcomes or has at least one feasible and approximately optimal outcome.*

To find approximately optimal outcomes, we use "optimality constraints" which generalize the optimality constraints used to find (unconstrained) optimal outcomes in Section . For simplicity, we will first describe the construction for Boolean features where the constraints are given in conjunctive normal form. Given a constrained CP-net $\langle N, C\rangle$, the **approximate optimality constraints** $opt^*(N, C)$ are $\{opt_C(p) \mid p \in N\}$.

Function opt_C maps the conditional preference statement $\varphi : a \succ \overline{a}$ onto the hard constraint:

$$(\varphi \wedge \bigwedge_{\psi \in C, \overline{a} \in \psi} \psi|_{a=true}) \rightarrow a$$

where $\psi|_{a=true}$ is the clause ψ where we have deleted $\overline{a}$. The purpose of $\psi|_{a=true}$ is to identify what has to be true so that we can safely assign a to true, its more preferred value. Suppose we have the constrained CP-net $\langle N, C\rangle$ where $N = \{a : b \succ \overline{b}, \overline{a} : \overline{b} \succ b, \overline{b} : a \succ \overline{a}. \ b : \overline{a} \succ a\}$ and $C = \{a \vee \overline{b}\}$. The optimality constraints corresponding to this constrained CP-net are the following clauses: $(a \wedge a) \rightarrow b$, $(b \wedge \overline{b}) \rightarrow \overline{a}$, $\overline{a} \rightarrow \overline{b}$, $\overline{b} \rightarrow a$. These simplify to: $a \rightarrow b$, $True$, $\overline{a} \rightarrow \overline{b}$, $\overline{b} \rightarrow a$. The only satisfying assignment for these constraints, plus the constraint in C, is ab. This is also the only approximately optimal outcome of the constrained CP-net. In general, the satisfying assignments of $C \cup opt^*(N, C)$ are exactly the feasible and approximately optimal outcomes of the constrained CP-net.

The above construction can be extended to non-Boolean features. Notice that in this case the constraints are no longer clauses but hard constraints over a set of variables with a certain domain. Given a constrained CP-net $\langle N, C\rangle$, consider any conditional preference statement p for feature x in N of the form $\varphi : a_1 \succ a_2 \succ a_3$. For simplicity, we consider just 3 values. However, all the constructions and arguments extend easily to more values. The approximate optimality constraints corresponding to this preference statement are:

$$\varphi \wedge (C_x \wedge x = a_1) \downarrow_{var(C_x)-\{x\}} \rightarrow x = a_1$$

$$\varphi \wedge (C_x \wedge x = a_2) \downarrow_{var(C_x)-\{x\}} \rightarrow x = a_1 \vee x = a_2$$

where $\downarrow X$ projects onto the variables in X, and C_x is the subset of constraints in C which involve variable x. We now show that this construction gives all the approximately optimal outcomes.

Theorem 4 *Given a (possibly cyclic) constrained CP-net $\langle N, C\rangle$ an outcome is feasible and approximately optimal for $\langle N, C\rangle$ iff it satisfies $C \cup opt^*(N, C)$.*

It immediately follows that we can test if an outcome is feasible and approximately optimal in polynomial time: we just need to test the satisfiability of the approximate optimality constraints. On the other hand, determining if a constrained CP-net has any feasible and approximately optimal outcomes is NP-complete. To find such outcomes, we can use any existing constraint or satisfiability solver on the approximate optimality constraints, including local search algorithms.

An obvious drawback of our approximation is that it generates a possibly large superset of the set of feasible Pareto optimal. Thus some of the returned outcomes could be not desired. For example, consider the following constrained CP-net: $a \succ \overline{a}$, $a : b \succ \overline{b}$, $\overline{a} : \overline{b} \succ b$, $\overline{a} \vee b$. Both ab and $\overline{a}\overline{b}$ are approximately optimal but only ab is feasible Pareto optimal.

However, it is possible to improve the approximation by excluding at least some of the undesired outcomes. More precisely, we can have a hierarchy of approximations that come arbitrarily close to the feasible Pareto optimal semantics. In brief, while our approximation just checks one flip away, others can check k flips away for $k > 1$. The larger the k is, the closer the approximation is to defining the feasible Pareto optimal semantics. Given any k, it is also possible to define constraints similar to the optimality constraints in $opt^*(N, C)$, and depending on k, such that their solutions are the feasible Pareto optimals plus those outcomes which are undominated in the CP-net if looking just k improving flips away. If k is larger than the length of the longest improving flip sequence, then the approximation returns a set of optimals which coincides with the set of feasible Pareto optimals.

A Weighted CSP approach

Not every constrained CP-net has a feasible Pareto optimal solution. For example, if we consider the CP-net in Figure 2 and a constraint which eliminates the outcome abc, then we can see in Figure 3 that there is no feasible Pareto optimal.

If there are some feasible Pareto optimals, then we can use algorithm Hard-Pareto to find them, or our less costly approximation technique to find a superset of them. If they do not exist, then we can still use the approximation technique to find the solutions of $C \cup opt^*(N, C)$, which are feasible outcomes which are undominated by one-flip-away outcomes. However, to know whether feasible Pareto outcomes exist or not may be costly. Another possibility is that no approximately optimal solutions exist even though C is satisfiable. In this case we may wish to find an outcome that is feasible and satisfies as many preferences as possible. Again, it may be very costly to prove that no approximately optimal solutions exist. Thus we have a hierarchy of desirability of outcomes, but we do not know in advance the properties of the problem.

We will now propose a technique which follows the reasoning above and returns the best outcomes w.r.t. both constraints and preferences even if there are no feasible Pareto optimals. The technique is based on a reformulation of the problem of finding optimal outcomes of a (cyclic) constrained CP-net as a Weighted Constraint Satisfaction Problem (WCSP). A WCSP is a set of variables plus a set of constraints where each constraint has a weight representing the cost of violating it. An optimal solution of a WCSP is an assignment of values to all the variables which minimizes the sum of the costs of the violated constraints.

The WCSP corresponding to a constrained CP-net is constructed in such a way that more desirable outcomes have lower weight, and a WCSP solver can be used to find as desirable an outcome as possible.

Given a constrained CP-net $\langle N, C\rangle$, we construct a WCSP $P = C \cup opt(N) \cup opt^*(N, C)$, using weight 1 for $opt(N)$-constraints, weight A for $opt^*(N, C)$-constraints and weight B for C-constraints, where A, B are constants satisfying $A > |opt(N)|$ and $B > A \times |opt^*(N, C)| + |opt(N)|$. To solve P, we must minimize the sum of the weights of the unsatisfied constraints.

Given an optimal solution S with weight w_S, the following properties hold:

- If $w_S = 0$, then S is a feasible Pareto optimal. This happens when $Sol(opt(N)) \cap Sol(C) \neq \emptyset$. The set of all optimal solutions is then $Sol(opt(N)) \cap Sol(C)$, which is a subset of the feasible Pareto optimals.
- If $0 < w_S < A$, then S is an approximately optimal solution. The reason is that it violates only some constraints in $opt(N)$. In this case, S could be feasible Pareto optimal but we cannot be sure.
- If $A \leq w_S < B$, then S is a feasible solution which satisfies the maximum number of constraints in $opt(N)$. The reason is that it violates at least one constraint in $opt^*(N, C)$, possibly some constraints in $opt(N)$ (the minimum number since it is optimal), but no constraint in C. Notice that this case occurs when all solutions are in a cycle. In this case $opt^*(N, C)$ has no solution, so there is no feasible Pareto optimal. However, the optimal solutions of the weighted constraints are the best among the feasible outcomes w.r.t. the preferences.
- If $B \leq w_S$, then S satisfies the maximum number of constraints in C, and then the maximum number of constraints in $opt(N)$. In this case C has no solution, so again there is no feasible Pareto optimal. However, the optimal solutions of the weighted constraints are the best outcomes w.r.t. first feasibility and then preferences.

If we find all the optimal solutions of the WCSP, we obtain either a subset (if the optimal weight is 0) or a superset (if the optimal weight is between 1 and $A-1$) of the feasible Pareto optimals, if they exist. However, if they do not exist, then we obtain all the best solutions w.r.t. first feasibility and then preferences. To do this, we never use dominance testing, and the complexity is that of solving a WCSP.

If instead we find just one optimal solution of the WCSP, then by looking at the weight of such a solution we can derive some knowledge about the given constrained CP-net. In particular, if the weight is 0, we know that there are feasible Pareto optimals, and the solution found is one of them. On the other hand, if the weight is at least A, then we know that there are no feasible Pareto optimals, but the solution found is the best outcome with respect to constraints and/or preferences. Last, if the weight is positive but lower than A, then we don't know if there are feasible Pareto optimals, but at least we know that the solution found is approximately optimal.

Future work

We plan to generalize the solver Hard-Pareto by replacing the hard constraints with soft constraints. We have already generalized the approximate method to constrained CP-nets where the constraints are soft rather than hard. In this case, the optimality constraints are still hard constraints but depend on the highest level of consistency of the soft constraints. Finding optimals is now as hard as solving a set of hard and soft constraints, while testing optimality is still easy. Finally, we plan to study the impact of trade-offs, as introduced in (Brafman & Domshlak 2002).

References

Boutilier, C.; Brafman, R. I.; Hoos, H. H.; and Poole, D. 1999. Reasoning with conditional ceteris paribus preference statements. In *UAI '99*, 71–80. Morgan Kaufmann.

Boutilier, C.; Brafman, R. I.; Domshlak, C.; Hoos, H. H.; and Poole, D. 2004a. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artif. Intell. Res. (JAIR)* 21:135–191.

Boutilier, C.; Brafman, R. I.; Domshlak, C.; Hoos, H. H.; and Poole, D. 2004b. Preference-based constraint optimization with CP-nets. *Computational Intelligence* 20(2):137–157.

Brafman, R., and Dimopoulos, Y. 2004. Extended semantics and optimization algorithms for cp-networks. *Computational Intelligence* 20, 2:218–245.

Brafman, R. I., and Domshlak, C. 2002. Introducing variable importance Tradeoffs into CP-nets. In *UAI '02*, 69–76. Morgan Kaufmann.

Domshlak, C., and Brafman, R. I. 2002. CP-nets: Reasoning and consistency testing. In *KR-02*, 121–132. Morgan Kaufmann.

Domshlak, C.; Rossi, F.; Venable, K. B.; and Walsh, T. 2003. Reasoning about soft constraints and conditional preferences: complexity results and approximation techniques. In *IJCAI-03*, 215–220. Morgan Kaufmann.

Wilson, N. 2004. Extending CP-nets with stronger conditional preference statements. In *AAAI-04*, 735–741. AAAI Press / The MIT Press.