# FCP-Nets: extending constrained CP-nets with objective functions

Marco Gavanelli[1] and Maria Silvia Pini[2]

1 : Department of Engineering, University of Ferrara, Italy
E-mail: marco.gavanelli@unife.it
2 : Department of Pure and Applied Mathematics, University of Padova, Italy
E-mail: mpini@math.unipd.it

**Abstract.** CP-Nets are a framework for dealing with qualitative preferences, both conditional and unconditional. They have received a lot of attention recently, and many extensions have been provided. In particular, the framework of constrained CP-Nets aims to choose, amongst the solutions that satisfy a set of constraints, the preferred one.
While the semantics of CP-Nets allows for cycles (and, indeed, cyclic CP-Nets occur in real-life problems), current algorithms are not always able to select the preferred solution in the presence of cycles. This means that, although there might exist feasible solutions, current algorithms are unable (in unlucky instances) to provide a preferred solution, and they simply fail, providing no answer to the user.
In this work, we propose FCP-Net, that is, a framework that adds an objective function to a constrained CP-Net. The objective function (essential in many constraint applications) is able to convey quantitative information into the CP-Net. Besides the higher expressivity of FCP-Nets, we are able to provide a single, best preferred solution in all feasible instances (those in which there is at least a solution), paving the way to a wider use of CP-Nets in practical applications.

## 1 Introduction

Preferences occur naturally in many real-world problems. There are many kinds of preferences, for example, quantitative ("I prefer ice cream at level 0.8") or qualitative ("I prefer meat to fish"). Moreover, preferences can be unconditional ("I prefer fish") or conditional ("If there is white wine, I prefer fish to meat"). Qualitative preferences are often easier to express: expressing a specific quantitative level of preference values to every element of a problem could be tedious and difficult for the user. Moreover, it is natural to allow the user to express, if she wants, her preferences in a way that is conditioned by other preferences.

An elegant formalism to represent conditional and qualitative preferences is *CP-nets* [5, 9, 3]. CP-nets model statements of qualitative and conditional preference which are interpreted under the Ceteris Paribus (that is, "all else being equal") assumption. CP-Nets can contain unconditional preferences, like $meat \succ fish$, that means that, all else being equal, I prefer meat to fish. Such a preference says nothing about two dinners that differ for the dessert: it states that between two identical meals differing only for

the main course, I prefer the one with meat. There are also conditional preferences, as in $meat : red \succ white$, that 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.

Most earlier work has concentrated on the acyclic model. For example, in *acyclic CP-nets* one can determine the most preferred outcome in linear time [5, 3], while finding the optimal in CP-nets containing cycles is, in general, NP-hard [6].

Unluckily, the presence of cycles in a CP-net occurs often in real-life situations. 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 [13]. Consider, for instance, a diner who has to choose either red or white wine, and either fish or meat. Given red wine, he prefers meat, and conversely, given meat he prefers red wine. On the other hand, given white wine, he prefers fish, and conversely, given fish he prefers white wine. This gives a consistent cyclic CP-Net, and there is no acyclic CP-net giving rise to the same preferences on outcomes [12].

Also, cyclic CP-nets emerge naturally when there is a set of interdependent variables, none of which is more important than the other. For example, [10] note that such dependency can emerge naturally among web-page components in their web-personalization tool.

In some cases, a cycle may denote inconsistent or contradicting information, that nevertheless happens in the real world. For example, a cycle may appear from the aggregation of preferences of several agents.

Various works propose methods for dealing with cyclic CP-nets, but a completely satisfactory solution is yet to come. In the original proposal [5] cyclic networks do not always have a preferred assignment. The only answer is that there is an inconsistency in the preferences, and the user is forced to revise her statements and make them consistent. In following works, CP-nets are combined with constraint reasoning [4], and only feasible outcomes are compared through the statements in the CP-Net. Further research found out that even in some cyclic networks there may be a non-dominated outcome and corresponding algorithms have been proposed. But, there are also instances in which a cycle of outcomes exists, and no outcome outside the cycle dominates any outcome in the cycle. In such a case no answer is given, and the algorithm fails.

A new approach [13] tries to give an answer by giving a different semantics for a constrained CP-net. In constrained CP-nets a feasible outcome is *approximately optimal* if no feasible outcome that differs only for one value is better. In this way, if a hard constraint cuts the cycles, one can find an approximately optimal outcome. Moreover, the authors propose how to encode preferences into constraints, which lets them find the approximately optimal outcome by solving a constraint satisfaction problem. Although very smart, this proposal does not always work: there might be cycles that are not cut by constraints, and in this case no (approximately optimal) outcome is found. Also, even when the cycle is cut by constraints, it is not always clear *why* the selected outcome should be considered as the best within the outcomes in the cycle.

Consider the following example (Figure 1a). Romeo and Juliet are planning a trip. Romeo prefers a luxury hotel if it is on the seaside ($sea : lux \succ \overline{lux}$), but for him a

non-luxury hotel is better in case there is no sea ($\overline{sea} : \overline{lux} \succ lux$). Juliet wants to go to the sea if the hotel is not luxury ($\overline{lux} : sea \succ \overline{sea}$), but chooses not to go to the sea (e.g., mountain) if the hotel is luxury ($lux : \overline{sea} \succ sea$). The preferences of both spouses make perfectly sense, taken singularly, but there is a dreadful cycle when they are put together: $luxsea \succ \overline{lux}sea \succ \overline{lux}\overline{sea} \succ lux\overline{sea} \succ luxsea$ (Figure 1b). Now, the original semantics [5] says that the two spouses should discuss how to make the network acyclic, and one of the two will probably have to drop one of his/her preferences, favouring the other. Let us add a constraint that breaks the cycle: there is no luxury hotel on the sea. The relaxed semantics [13] says that the only approximately optimal solution is $\overline{lux}sea$, since there is no chain of improving flips from $\overline{lux}sea$ to other solutions, passing only on feasible assignments. Indeed, this answer is better than no answer, but will Romeo accept this explanation, or will he argue to support the option $\overline{lux}\overline{sea}$?

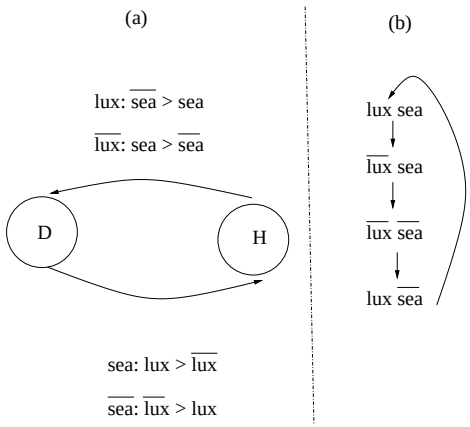


**Fig. 1.** Romeo and Juliet example.

Moreover, there are worse instances: the constraints might not remove elements from the cycle. In this case, neither the original, nor the relaxed semantics is able to give an answer: they both fail.

Picture the following situation. The user defines through constraints the feasible region, and happily finds that there are solutions. Since the solutions are many, and she cannot scan through all of them, she decides to state formally her preferences (possibly, together with those of other users, involved in the decision process). She adopts CP-Nets, and takes one of the available (exact or approximate) semantics [5, 4, 13], together with the corresponding algorithm. To her disappointment, now no solution is feasible any longer: she has either to revise her preferences (which might mean find an agreement with the other users about which preferences should be relaxed), or to accept that all feasible solutions are equally preferred.

In this paper we propose a pragmatic way to deal with such situations. We believe that, since the user spends time and effort in formalising his/her preferences, throwing

away all preferences is not a satisfactory answer. In some cases, signalling that preferences should be revised is not enough. Selecting answers depending on feasibility will not always be acceptable. We believe that cycles should not be avoided at all costs, but they should be considered as sets of equally preferred outcomes (as was already hinted in the original paper [5]). But we need a way to *break ties*.

Also, a practical need in many applications is to be able to reason about numeric values. Indeed, qualitative preferences are easy to express when there is no quantitative information, but when some quantitative information is given, concealing it would be unwise. In all constraint languages there are modules that perform optimisation, typically through the branch-and-bound algorithm; we think that a unique framework able to deal both with qualitative and quantitative information would be very handy.

We propose a framework taking into consideration both qualitative and quantitative information. Qualitative information is given through CP-Nets, widely studied and appreciated framework. Quantitative information is used to break ties in case the CP-Net alone is unable to select one (or more) preferred outcomes. We adopt hard constraints in our language, to let the user express impossibility, besides preferences. We have an objective function to perform optimisation, ubiquitous in CP languages.

In many practical applications there are functions to be optimised. They might be less important than qualitative preferences: indeed, if the user wants to express qualitative preferences, it is probably because he/she is unable to provide numeric values, and deems numeric parts less important. But quantitative preferences very often exist, and they are the perfect candidate for breaking ties. In our Shakespearian example, Romeo and Juliet, unable to make a preferred choice, could raise the cost information, and choose the less expensive option amongst the equally preferable choices. We do not give up qualitative information when there is no preferred outcome, but we select the best outcome (according to an objective function) amongst the options that are equally preferred according to the CP-net.

To handle constrained cyclic CP-nets and to satisfy the goals mentioned above, i.e., to discriminate among various outcomes that are all optimal according to the definition above, and to find a weaker optimal outcome, when no optimal outcome is found, we will introduce a new formalism, called *constrained FCP-net* (where “F” stands for objective function). Such a formalism extends the classical constrained CP-net formalism, by considering, besides hard constraints and the qualitative aspect of the CP-net, also a quantitative aspect, given by an objective function, that may relate some of the variables of the CP-net.

Other formalisms have been defined to introduce a quantitative aspect in the classical CP-net formalism. However, we will show in the related work section that all these formalisms are different from our framework.

The rest of this paper is organized as follows: Section 2 provides necessary background. Section 3 describes constrained FCP-nets and the notion of lex optimality. Section 4 presents the algorithm that we propose to handle constrained FCP-net, that takes as input a constrained FCP-net and it returns the lex optimal outcomes. Section 5 presents related work. We conclude in Section 6 with a discussion of future work.

## 2 Background

We now give the necessary background. In particular, we will give the basic notions of the CP-nets [5] and constrained CP-nets [13].

### 2.1 CP-net

CP nets were introduced in [5] as a tool for compactly and intuitively representing qualitative preference relations. CP-nets are a graphical model for representing conditional and qualitative preferences. They exploit conditional preferential independence by structuring an agent's preferences under the ceteris paribus assumption. Informally, CP nets are sets of conditional ceteris paribus (CP) preference statements. Many philosophers and AI researchers [11] have argued that many of our preferences are of this type. 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 $\{X_1, \ldots, X_n\}$ with finite, discrete domains $D(X_1), \ldots, D(X_n)$. 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 outcome on $Pa(X_i)$. This preference is assumed to take the form of total or partial order [5, 13] over $D(X_i)$. For example, consider a CP-Net whose features are $A$, $B$ and $C$ with binary domains containing $f$ and $\bar{f}$, if $F$ is the name of the feature, and with the CP preference statements as follows: $a \succ \bar{a}$, $a : \bar{b} \succ b$, $\bar{a} : b \succ \bar{b}$, $b : \bar{c} \succ c$, $\bar{b} : c \succ \bar{c}$. For example, the conditional statement $a : \bar{b} \succ b$ states that given $A = a$, then $B = \bar{b}$ is better than $B = b$.

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 $a\bar{b}c$ to $abc$ is a worsening flip since when $A = a$, $\bar{b}$ is better than $b$. We say that one outcome $\alpha$ is *better than* another outcome $\beta$ (written $\alpha \succ \beta$) iff there is a chain of worsening flips from $\alpha$ to $\beta$. Such a definition induces a preorder over the outcomes.

For generic CP-nets has been shown that finding optimal outcomes and testing for optimality in this ordering is PSPACE-complete [12]. However, in acyclic CP-nets, there is only one optimal outcome and this can be found in linear time [5, 3]. 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 then $B = \bar{b}$, and thus $C = c$. The optimal outcome is therefore $a\bar{b}c$.

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, the outcome $\bar{a}\bar{b}\bar{c}$ is worse than $a\bar{b}c$, in fact there is a chain of worsening flips $a\bar{b}c \succ a\bar{b}\bar{c} \succ \bar{a}\bar{b}\bar{c}$.

### 2.2 Constrained CP-net

Our work builds on the nice procedure to handle hard constraints in CP-nets presented in [13]; we give some background on their semantics.

The idea is to translate the CP-statements of the CP-net into a set of hard constraints such that the solutions of these hard constraints (called "optimality constraints") are the optimal solutions of the CP-nets. 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 $\phi$ of an assignment of values to other variables. Then, for each of such statements, the corresponding optimality constraint is $\phi \rightarrow \vee_j (x = a_j)$, 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 $N$. Moreover, an outcome is optimal in the ordering induced by a CP-net $N$ iff it is a satisfying assignment for $opt(N)$ [13].

For example, the CP-statements $a \succ \bar{a}$ and $a \wedge b : c \succ \bar{c}$ map to the hard constraints $a$ and $(a \wedge b) \rightarrow c$ respectively. In the case of boolean variables, these constraints map directly to SAT clauses, so SAT is a convenient technology for solving these problems.

## 3 Constrained CP-net with objective function

We now introduce a formalism that generalizes the classical one of the CP-net, by allowing both the presence of hard constraints and of an objective function.

**Definition 1 (constrained FCP-net).** *A constrained FCP-net is a tuple* $(V_N, V_H, N, H, F)$ *where* $V_N$ *and* $V_H$ *are two sets of variables (non necessarily disjoint),* $N$ *is a CP-Net over variables in* $V_N$*,* $H$ *is a set of hard constraints over variables in* $V_H$*, and* $F$ *:* Assignments$(V_N \cup V_H) \mapsto \mathcal{R}$ *is an objective function, where* Assignments$(V_N \cup V_H)$ *is the set of all the possible assignments to the variables in* $(V_N \cup V_H)$*.*

**Definition 2 (complete outcome).** *A complete outcome in a constrained FCP-net* $(V_N, V_H, N, H, F)$ *is an assignment to all the variables in* $V_N \cup V_H$*.*

**Definition 3 (Solution).** *A complete outcome in a constrained FCP-net* $(V_N, V_H, N, H, F)$ *is feasible iff it satisfies all the constraints in* $H$ *(i.e., iff* $H(s)$ *is true). Such an outcome is also called a* solution *of the FCP-Net.*

**Definition 4 (Non-dominated solution).** *A solution* $s$ *of a constrained FCP-net* $(V_N, V_H, N, H, F)$ *is non-dominated iff* $\nexists$ *solution* $s'$ *s.t.* $s' \succ s$*.*

*Example 1.* Consider the CP-Net in Figure 2, let us call it $N$, on the boolean variables $V_N = \{A, B, C\}$. The complete set of outcomes is represented in Figure 3, where the arrows show the dominance relations. Assume also to have a unique constraint over the same variables (thus, in this instance, $V_H = V_N$), that states $\neg(a \wedge b \wedge c)$: this constraint makes infeasible the outcome $abc$.

The values of the objective function $F$ are represented in Figure 3 next to the complete outcome. For example, $f(\bar{a}\bar{b}c) = 5$ and $f(a\bar{b}c) = 7$. □

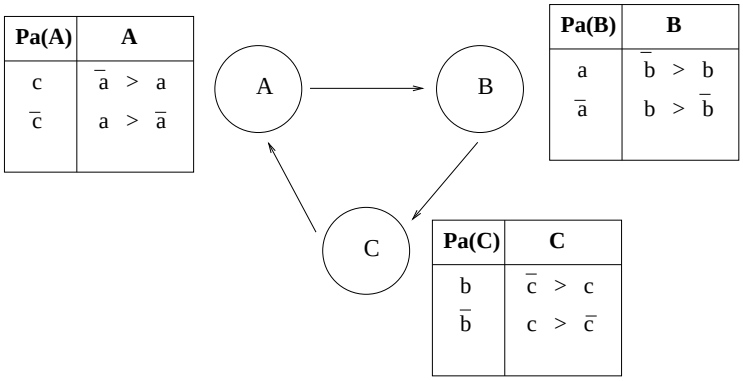


**Fig. 2.** A CP-net.

Amongst the (possibly many) feasible outcomes, we have to select the preferred ones in the CP-Net and with respect to the objective function. The problem is that the CP-Net and the objective function could propose different solutions: we have to select the ones to be presented to the decision maker. In the previous example, the CP-Net is unable, alone, to provide a preferred solution because of the cycle[1]. The objective function alone suggests solution $\bar{a}\bar{b}\bar{c}$, with an optimal value of $28$, but this is hardly the solution intended by the user: in fact, it is the worst possible solution according to the CP-Net (since $abc$ is not even feasible).

Different strategies could be used; in this work we focus on lexicographic combinations, in which the various criteria are sorted for importance.

One could decide that the objective function is the most important criterion. This case is very simple: one can simply solve the constrained optimization problem obtained by dropping the CP-Net, and then choose, amongst the solutions with the same, optimal value of objective function, the preferred one according to the CP-Net. But we could get very bad solutions with respect to the CP-Net (as the previous example showed).

A more challenging problem is the dual one: the CP-Net is more important, and only in cases where it fails to propose a clear winner, we discriminate the potential solutions through the objective function. This is also, probably, the semantics that the decision maker is more inclined to accept: after he defined formally his preference table, he will probably want to exploit the CP-Net at its best. Intuitively, in Figure 3 we have a cycle of solutions that are dominated only by other solutions in the cycle. Thus, the intended solution should be searched for in the cycle. Within the cycle, we select the best solution with respect to the objective function: outcome $\bar{a}\bar{b}\bar{c}$.

**Definition 5 (lex optimal solution).** *Given a constrained FCP-net $P = (V_N, V_H, N, H, F)$, a solution $s$ is lex optimal iff one of the following items holds:*

– *$s$ is non-dominated and $\forall s'$ non-dominated solution of $P$, $F(s) \geq F(s')$;*

[1] And the cycle is not cut by constraints, so the relaxed semantics [13] does not provide a solution either.

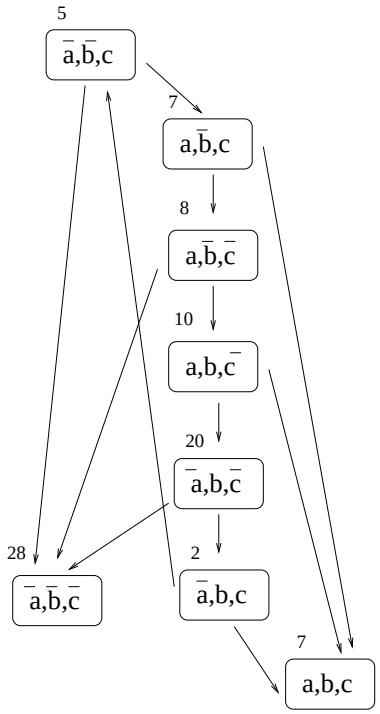


**Fig. 3.** Complete outcomes of the FCP-net in Figure 2.

– *$P$ does not admit non-dominated solutions and, for any solution $s'$, if $s' \succ s$ then $(s \succ s' \wedge F(s) \geq F(s'))$.*

The first condition says that a solution is lex-optimal if it is a non-dominated solution in the CP-Net and, among the non-dominated solutions, it is the one with the highest value of the objective function. Moreover, there can be cases in which the CP-Net does not provide a non-dominated solution, but we have a *cycle* of solutions, where each solution in the cycle dominates the others (and even itself). In such a case, the usual CP-Net semantics fails to propose a preferred solution. The second condition of Definition 5 says that, if there are no non-dominated solutions, a solution $s$ is a lex-optimal solution if, whenever it is dominated by some solution $s'$, it also dominates $s'$ (i.e., they are in the same cycle) and, moreover, it is better with respect to the objective function $F$.

One interesting property of lex-optimal solutions is that, under reasonable assumptions, in a constrained FCP-Net there exists exactly one lex-optimal solution (provided that the constraints are satisfiable). This property answers a very practical need: even if the CP-Net is not consistent, it is possible to find a unique preferred solution, which avoids two potential problems of the CP-Nets. The first is that the CP-Net could leave too many solutions, amongst which the decision maker has to choose. The second, even

worse, is that there is no solution to choose, and the decision maker has to revise his/her preferences.

**Theorem 1.** *Consider a constrained FCP-net $P = (V_N, V_H, N, H, F)$. If the set of constraints $H$ admits at least a solution, and the function $f$ is injective, then there is exactly one lex-optimal solution to $P$.*

*Proof.* Suppose that the constrained CP-Net has a non-empty set $L$ of non-dominated solutions. Since the set of solutions is finite and $f$ is limited, $f$ admits a maximum value on the set $L$, call it $f(m)$. Since $f$ is injective, the maximum is unique. By Definition 5, $m$ is a lex-optimal solution. Since $f(m)$ is the maximum, any other feasible solution $s \in L$ is $f(s) < f(m)$, and thus it is not lex-optimal.

Suppose now that the set of non-dominated solutions of the constrained CP-Net is empty. By Definition 4, this can happen either if there are no solutions (which is false, by hypothesis) or if all solutions are dominated. Consider solution $s$: it is dominated iff there exists a solution $s' \succ s$. In its turn, $s'$ is dominated iff there is some solution $s'' \succ s'$. Since the set of solutions is finite, and no solution is non-dominated, there must be a cycle. If such cycle is dominated by some solution $s^*$ that is not in the cycle, then we can use the same argument used above to show that $s^*$ belongs to a cycle. Consider now a cycle $C$ such that, for each $s_1, s_2 \in C$, $s_1 \succ s_2$ and $s_2 \succ s_1$, and $\nexists s^* \notin C$ such that $s^* \succ s_1$. Since the set of solutions is finite, $C$ is finite. Since $F$ is injective, there is a unique maximum amongst the elements of $C$; this maximum is by Definition 5 the unique lex-optimal solution of $P$.

We now present an algorithm that finds lex-optimal solutions.

## 4 An algorithm

We now show an algorithm to find lex-optimal solutions of a constrained FCP-net. The algorithm uses procedure $HardPareto$, defined previously in the literature [13], that finds the non-dominated solutions of a constrained CP-Net.

This algorithm (Algorithm 1) takes as input a constrained FCP-net $P = (V_N, V_H, N, H, F)$, and it returns as output a set $L$ coinciding with the set of the lex-optimal solutions of $P$.

The algorithm first checks if there are non-dominated solutions with procedure $HardPareto$. If the set of returned solutions is not empty, then there are non-dominated solutions, thus the lex-optimal ones should be selected amongst those (line 1). Amongs those candidate lex-optimal solutions, we select those with the highest value of the objective function via procedure $SelectOptimal$ (procedure 2), that is basically a search of the maximum element in a set.

$HardPareto$ might return no solutions in two cases [13]: if the set of hard constraints is unsatisfiable, or if no feasible solution is non-dominated, due to the fact that there is a cycle. We distinguish the two cases; in case there are no solutions, that is no feasible outcome for the constraints, we simply fail (line 2).

Otherwise, there must be a cycle that contains the lex-optimal solutions. In this case, we break the cycles by removing some of the preferences. Function $AcyclicReduced$

**Algorithm 1**: *LexOptimalSolutions*($CPNet, CSP, F$)

```
   Input: P = (N, CSP, F): a constrained FCP-net;
   Output: L: a set of LexOptimal solutions
   L ← HardPareto(CSP, N);
   if |L| > 0 then
1      return SelectOptimal(L, F)
   else
      // either there are no solutions, or there is a top cycle
2     if Solve(CSP)=fail then
3        return fail
      else
         // there is a top cycle
4        A ← AcyclicReducedCPnets(N)
         L ← ∅
         foreach A_i ∈ A do
            L ← L ∪ HardPareto(CSP, A_i)
5        L ← RemoveDominated(L, N)
         L ← SelectOptimal(L, F)
6        return L
```

$CPnets$ (line 4) finds all the maximal ways to make the CP-Net acyclic. More precisely, it finds all the ways that make the CP-net acyclic by eliminating a set of edges that is minimal (with respect to set inclusion). In other words, it computes the set of spanning trees of the dependence graph of the CP-Net. For each of the acyclic reduced CP-Nets, we compute the set of the non-dominated solutions with algorithm $HardPareto$. Then, among these solutions, we select only those that are non-dominated in the given cyclic CP-net. We achieve this with the $RemoveDominated$ procedure (line 5), that performs dominance testings among these solutions and returns only the non-dominated ones. The chosen solutions are the candidate lex-optimal solutions, that will be again selected with the $SelectOptimal$ procedure.

**Theorem 2.** *Consider a constrained FCP-net $P = (V_N, V_H, N, H, F)$. The set returned by Algorithm 1 applied to $P$ coincides with set of the lex optimal solutions of $P$.*

*Proof.* The proof consists of two parts. In the first (soundness) we will show that every solution returned by Algorithm 1 applied to the FCP-net $P$ is lex optimal for $P$, while in the second part (completeness) we will show that every lex optimal solution of $P$ is returned by Algorithm 1 applied to $P$.

*Soundness.* Given a constrained FCP-net $P = (V_N, V_H, N, H, F)$, every solution returned by Algorithm 1 applied to $P$ is lex optimal. In fact,

– If the hard constraints have no solutions, the algorithm returns failure in line 3.

**Procedure** `SelectOptimal`$(L, F)$

**Input**: $L$= a set of complete outcomes; $F$: the objective function
**Output**: the optimal solutions amongst $L$
$Opt \leftarrow \emptyset$
$F^* \leftarrow -\infty$
**foreach** $X \in L$ **do**
  **if** $F(X) = F^*$ **then**
    $Opt \leftarrow Opt \cup \{X\}$
  **if** $F(X) > F^*$ **then**
    $Opt = \{X\}$
    $F^* \leftarrow F(X)$
**return** $Opt$

**Procedure** `RemoveDominated`$(L, N)$

**Input**: $L$= a set of complete outcomes; $N$: a CP-Net
**Output**: the non-dominated solutions
**foreach** $X \in L$ **do**
  **foreach** $Y \in L$ **do**
1   **if** $(Y \succ X) \wedge (X \not\succ Y)$ **then**
      $L \leftarrow L \setminus \{X\}$
**return** $L$

– If the algorithm Hard-Pareto succeeds (line 1), then the constrained CP-Net has non-dominated solutions, and, among the non-dominated solutions, we choose those with the highest value of the objective function, thus these solutions are obviously lex-optimal.
– Otherwise, suppose the algorithm returns a solution $s$ in line 6. Since algorithm Hard-Pareto failed on the original CP-Net $N$, this means that $s$ was dominated by some $s'$ in $N$, while by making $N$ acyclic this is no longer the case. On the other hand, if $s$ is dominated by $s'$ and $s$ does not dominate $s'$, then $s$ is removed from the set of candidate solutions in line 5.
  Finally, considering all solutions that are dominated only by solutions involved in the same cycle, only the best solution with respect to $f$ is returned (see Procedure 2), thus if $s$ is returned then it is lex-optimal.

*Completeness.* Suppose that there exists a lex-optimal solution $s$; we prove that $s$ is returned by Algorithm 1.

– If $s$ is non-dominated in $N$, then Algorithm Hard-Pareto will find it, and will be returned in line 1.
– If $s$ is dominated by another feasible solution $s'$, and $s$ is lex-optimal, then they must be in the same cycle, thus also $s' \succ_N s$. Thus, there must be a sequence of assignments $s' \succ_N a_1 \succ_N a_2 \succ_N \cdots \succ_N a_k \succ_N s$, where $s$ is obtained with one worsening flip from $a_k$. Such a worsening flip must be due to an arrow $e$ of

the CP-net. If we remove arrow $e$ from $N$, then $s$ becomes non-dominated, thus algorithm Hard-Pareto is able to find it. The arrow $e$ is clearly in a cycle, thus there exists a way to make $N$ acyclic that removes $e$.
Finally, solution $s$ will not be removed by Procedure 2 because it has the optimal value of $f$ within the feasible ones in the cycle. □

*Example 2.* Let us consider the constrained FCP-Net shown in Example 1.

Algorithm 1 invokes Hard-Pareto, that returns no solutions, since there does not exist any non-dominated solution with respect to the given CP-Net. The algorithm checks if the set of constraints is satisfiable (line 2). Since the corresponding CSP is satisfiable, the CP-Net of Figure 2 is made acyclic in all possible ways: namely by removing either arc $A \rightarrow B$, or $B \rightarrow C$ or $C \rightarrow A$. We run Hard-Pareto on the three acyclic CP-Nets, and we obtain the candidate solutions in Table 1.

| sub-CP-Net | candidate solutions | |
|---|---|---|
| $\{A \rightarrow B, B \rightarrow C\}$ | $abc$ | $\bar{a}b\bar{c}$ |
| $\{B \rightarrow C, C \rightarrow A\}$ | $ab\bar{c}$ | $\bar{a}\bar{b}c$ |
| $\{C \rightarrow A, A \rightarrow B\}$ | $\bar{a}bc$ | $a\bar{b}\bar{c}$ |

**Table 1.** Candidate solutions for the problem in Example 1.

The set $L$ of candidate solutions is then $\{\bar{a}\bar{b}c, a\bar{b}c, a\bar{b}\bar{c}, ab\bar{c}, \bar{a}b\bar{c}, \bar{a}b\bar{c}, \bar{a}bc\}$. Function $RemoveDominated$ does not remove any of these solutions; in fact none of these elements is dominated by some solution that is not dominated by them. Finally, $SelectOptimal$ selects only the solution with maximum value of objective function: namely $\bar{a}b\bar{c}$. □

### 4.1 Discussion

The algorithms reasoning on CP-Nets are blessed by the efficient test of optimality of CP-Nets, but are cursed by the complexity of the dominance test, that is in PSPACE. The same holds for algorithms dealing with constrained CP-Nets: for example, one of the improvements of Hard-Pareto [13] with respect to Search-CP [4] is that it avoids dominance testing in some significant cases: when there are no (feasible) solutions and when the set of optimal solutions coincides with the solutions of a CSP built by adding optimisation constraints to the set $H$ of hard constraints.

Algorithm 1 does not discard a priori solutions that belong to cycles, but considers them as equivalent with respect to the CP-Net, and differentiates them through the secondary criterion given by the objective function. This means that we can answer user's questions even in cases where other semantics fail. In our semantics, showing that a solution is dominated is not enough to discard it, as in constrained CP-Nets, because we have more ambitious objectives. In other words, we ask more to our semantics, so, not surprisingly, the algorithm achieving it can be more complex.

Nevertheless, Algorithm 1 has some notable features. First, it uses Hard-Pareto to preprocess the FCP-Net; this means that when Hard-Pareto succeeds in finding solutions, we have the same complexity (and we do not need to perform further dominance tests).

Second, we do not perform dominance tests on all the possible solutions, but only on the selected solutions that are feasible Pareto optimal for an acyclic CP-Net.

**Algorithm 4**: *LexOptimalSolutions*($CPNet, H, F$)

```
   Input: P = (N, H, F): a constrained FCP-net;
   Output: L: a set of LexOptimal solutions
   L ← HardPareto(H, N);
   if |L| > 0 then
1      return SelectOptimal(L, F)
   else
       // either there are no solutions, or there is a top cycle
2      if Solve(H)=fail then
3          return fail
       else
           // there is a top cycle
4          A ← AcyclicReducedCPnets(N)
           L ← ∅; L_nd ← ∅
5          foreach A_i ∈ A do
6              X ← Optimal(A_i)
7              if H(X) then
                   L_nd ← L_nd ∪ {X}
                   A ← A \ {A_i}
           foreach A_i ∈ A do
               L ← L ∪ HardPareto(H, A_i)
8          L ← RemoveDominated(L, L_nd, N)
           L ← SelectOptimal(L, F)
9          return L
```

Algorithm 4 is an improvement of Algorithm 1, that exploits some features of acyclic CP-Nets. In an acyclic CP-Net, finding the optimal has linear cost; if the optimal is also feasible, then it is obviously non-dominated. Algorithm 4 reduces the number of invocations of $HardPareto$; moreover it reduces the number of domination tests. If the optimal solution of an acyclic subnet is feasible (line 7), then it is added to a distinguished set $L_{nd}$; moreover there is no need to invoke $HardPareto$ on this subnet, because we already know the only optimal solution. Note that for each element $X \in L_{nd}$, we have that $\forall Y, X \succ Y$, so there is no need to invoke $RemoveDominated$ for the elements of $L_{nd}$ (condition 1 of Procedure 3 is always false); in this way, we can potentially save many dominance checks.

**Procedure** `RemoveDominated`$(L, L_{nd}, N)$

**Input**: $L$: a set of complete outcomes; $L_{nd}$: a set of candidate Lex-optimal solutions; $N$: a CP-Net
**Output**: the non-dominated solutions
**repeat**
  choose $X \in L$
  $L \leftarrow L \setminus \{X\}$
  **if** $\nexists Y \in L \cup L_{nd}$ *such that* $(Y \succ X) \wedge (X \not\succ Y)$ **then**
    $L_{nd} \leftarrow L_{nd} \cup \{X\}$
**until** $L \neq \emptyset$ ;
**return** $L_{nd}$

## 5 Related work

Other formalisms have been defined to introduce a quantitative aspect in the classical CP-net formalism. For example, utility CP-net (UCP-net) formalism [2] can be viewed as an extension of the CP-net model, that allows one to represent quantitative utility information rather than simple preference orderings. However, in such a formalism only acyclic CP-nets have been considered, while we consider also cyclic ones. Moreover, their objective function is a GAI (Generalized Additive Independent) [1], that depends on the CP-net, while our objective function is a generic function, and thus it is independent from the CP-net.

Another formalism that adds to the CP-net formalism a quantitative aspect is the Tradeoff CP-net (TCP-net) [7, 8]. In such a formalism relative importance statements have been added to the qualitative and conditional preference representation, since it is very natural to express the fact that one variable's value is more important than another's. Adding an explicit importance relation, CP-nets induce an importance relation between nodes and their descendants only. This approach is different from our approach, since, to handle the presence of cycles in the CP-nets, we do not introduce as in [7, 8] an explicit importance relation among variables, but we take into account the presence of an objective function.

## 6 Conclusions and future work

We have proposed a new formalism, the constrained FCP-net formalism, that extends the classical constrained CP-net framework, by considering, besides hard constraints and the qualitative aspect of the CP-net, also a quantitative aspect, given by an objective function, that may relate some of the variables of the CP-net. Such a quantitative aspect is used to break ties in case the CP-Net alone is unable to select one (or more) preferred outcomes. We have defined a new notion of optimal solution, and we have given an algorithm to find such a kind of optimal solutions also in the case of cyclic CP-nets. Such an algorithm returns always at least one of these solutions, if there is at least a feasible outcome in the CP-net. We plan to implement a tool to handle FCP-nets and to test it empirically over classes of these problems.

## Acknowledgements

This work has been partially supported by Italian MIUR PRIN project "Constraints and Preferences" (n. 2005015491).

## References

1. F. Bacchus and A. J. Grove. Graphical models for preference and utility. In *Proceedings of UAI 1995*, pages 3–10. Morgan Kaufmann, 1995.
2. C. Boutilier, F. Bacchus, and R. I. Brafman. UCP-networks: A directed graphical representation of conditional utilities. In J. S. Breese and D. Koller, editors, *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5*, pages 56–64. Morgan Kaufmann, 2001.
3. C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artif. Intell. Res. (JAIR)*, 21:135–191, 2004.
4. C. Boutilier, R. I. Brafman, Carmel Domshlak, H. H. Hoos, and D. Poole. Preference-based constraint optimization with CP-nets. *Computational Intelligence*, 20(2):137–157, 2004.
5. C. Boutilier, R. I. Brafman, H. H. Hoos, and D. Poole. Reasoning with conditional ceteris paribus preference statements. In K. B. Laskey and H. Prade, editors, *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30-August 1*, pages 71–80. Morgan Kaufmann, 1999.
6. R. I. Brafman and I. Dimopoulos. A new look at the semantics and optimization methods of CP-networks. In *Proceedings of IJCAI 2003*, pages 1033–1038. Morgan Kaufmann, 2003.
7. R. I. Brafman and C. Domshlak. Introducing variable importance Tradeoffs into CP-nets. In A. Darwiche and N. Friedman, editors, *UAI '02, Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence, University of Alberta, Edmonton, Alberta, Canada, August 1-4*, pages 69–76. Morgan Kaufmann, 2002.
8. R. I. Brafman, C. Domshlak, and S. E. Shimony. On graphical modeling of preference and importance. *JAIR*, 25:389–424, 2006.
9. C. Domshlak and R. I. Brafman. CP-nets: Reasoning and consistency testing. In D. Fensel, F. Giunchiglia, D. L. McGuinness, and M. Williams, editors, *Proceedings of the Eight International Conference on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France, April 22-25*, pages 121–132. Morgan Kaufmann, 2002.
10. C. Domshlak, R. I. Brafman, and S. E. Shimony. Preference-based configuration of web page content. In *Proceedings of IJCAI 2001*, pages 1451–1456. Morgan Kaufmann, 2001.
11. J. Doyle and M. Wellman. Representing preferences as ceteris paribus comparatives. In S. Hanks, S. Russell, and M. Wellman, editors, *Decision-Theoretic Planning: Papers from the 1994 Spring AAAI Symposium*, pages 69–75. AAAI Press, Menlo Park, California, 1994.
12. J. Goldsmith, J. Lang, and M. Truszczynski N. Wilson. The computational complexity of dominance and consistency in cp-nets. In *Proceedings of IJCAI 2005*, pages 144–149. Professional Book Center, 2005.
13. S. D. Prestwich, F. Rossi, K. B. Venable, and T. Walsh. Constraint-based preferential optimization. In *AAAI-05*, pages 461–466, 2005.