CP-networks: semantics, complexity, approximations and extensions

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

T. Walsh
Cork Constraint Computation Centre, University College Cork
Cork, Ireland
tw@4c.ucc.ie

July 23, 2002

Abstract

CP-networks are an elegant and compact qualitative framework for expressing preferences. Unfortunately, reasoning with CP-networks can be computationally hard. We propose two remedies to this problem. First, we suggest a new semantics for CP-networks that is more tractable. Second, we show how CP-networks can be approximated with soft constraints. The second remedy also allows us to integrate preference and constraint reasoning in a single formalism.

1 Introduction and motivation

Extracting preferences from users is a notoriously difficult task. It is especially difficult when users are required to provide quantitative preference measures. Users are often happier providing qualitative measures. "All else being equal, I prefer a red dress to a yellow dress". "If the car is a convertible, I prefer a hard top to a soft top". These are conditional ("if the car is convertible") and *ceteris paribus* ("all else being equal") preference statements. In [2], a compact formalism called CP-networks is given for representing and reasoning about preference rankings given such conditional preference statements.

Unfortunately, reasoning about CP-networks is computationally hard in general. We suggest two different ways of tackling this problem. First, we show how to modify the semantics of CP-networks in a modest way, preserving the *ceteris paribus* property but making reasoning more tractable. Second, we show how to abstract a CP-network onto a soft constraint satisfaction problem (and to compute the quantitative preferences in the soft problem). Again, this makes reasoning more tractable. However, this abstraction does loose some information. For example, incomparable items in the CP-network may appear ordered in the soft problem. However, the abstractions are always faithful, in the sense that ordered items in the CP-network remain ordered in the same way in the soft problem. We should note that there has been some recent work on approximating CP-networks [3] using a quantitative representation of preferences However, only one approximation is considered in [3], while we propose several here, within a general framework. Moreover, the user needs to specify the numerical preferences, and the system checks whether they satisfy the ceteris paribus property. Here, we automatically generate the quantitative preferences while maintaining the ceteris paribus condition. Another advantage of the approach taken here is that it allows us to integrate preference and constraint reasoning in a single formalism. This would be useful, for example, in configuration problems where we have both user preferences and physical constraints to satisfy.

2 CP-networks

For the formal and complete definitions about CP nets, we refer to [2]. We assume a set of *features*, $\{A, B, C, \ldots\}$. Each feature can have a finite domain of values. Without loss of generality, we assume features have just two possible values (true or false), written a or $\overline{a}$, b or $\overline{b}$, etc. We assume that the user has a preference ranking, a total preoder $\succeq$ on assignments of values to features. Specifying such a preference ranking explicitly is not practical in general as there are an exponential number of possible assignments to features. Fortunately, the problem can be decomposed into a more manageable form if the user's preferences satisfy the *ceteris paribus* property. This is based on the idea of (conditional) preferential independence. Just as Bayesian networks exploit the independence of probabilities to specify compactly complex conditional probabilities, we can exploit the independence of the user's preferences to specify compactly a complex preference ranking.

The preference ranking is specified by a set of (conditional and unconditional) *ceteris paribus* preference statements. An unconditional preference statement is of the form: $a \succ \overline{a}$. This has the semantics that whatever values are taken by the other features, we will prefer an assignment to A of a over $\overline{a}$. That is, if we take an assignment which assigns $\overline{a}$ to A, we will prefer to swap this by a keeping everything else constant. A conditional preference statement is of the form: $a : b \succ \overline{b}$. This has the semantics that, having assigned a to A, we will prefer an assignment of b to B over $\overline{b}$. That is, if we take an assignment which assigns a to A and $\overline{b}$ to B, we will prefer to swap $\overline{b}$ by b keeping everything else constant. In other words, for an ordering to be ceteris paribus, it must be always possible to partition the remaining elements, w.r.t. the ordered couple, of the ordered set in two sets Z and Y such that $\exists z$ assignment to Z such that if $z\ ya > by$ for any y, assignment to Y, holds. Preference statements define a

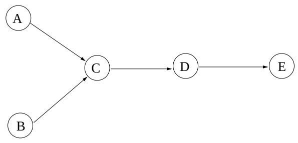

Figure 1: A CP net.

graph of dependencies between features. If this graph does not contain cycles, we can extract an ordering on features. For example, A is ordered before B by the conditional preference statement $a : b \succ \overline{b}$. As a running example, consider the CP-network defined by the following preference statements: $a \succ \overline{a}$, $b \succ \overline{b}$, $(a \wedge b) \vee (\overline{a} \wedge \overline{b}) : c \succ \overline{c}$, $(a \wedge \overline{b}) \vee (\overline{a} \wedge b) : \overline{c} \succ c$, $c : d \succ \overline{d}$ $\overline{c} : \overline{d} \succ d$, $d : \overline{e} \succ e$ $\overline{d} : e \succ \overline{e}$. This is shown in Figure 1.

There are several types of question we can ask of a CP-network. First, is the network consistent? A CP-network is **consistent** iff there exists an assignment that satisfies all the conditional preference statements at the highest level. For example, $a : b \succ \overline{b}$ and $\overline{a} : \overline{b} \succ b$ are satisfied at their highest level by ab and $\overline{a}\overline{b}$. Unfortunately, as we show in the next section, deciding the consistency of a CP-network is intractable in general (assuming P $\neq$ NP).

A second type of question is whether one assignment is better than another. This is usually referred to as a dominance query. In [2], such queries are answered by finding a sequence of worsening flips. One assignment dominates another if we can get from the first to the second by flipping values, each of which moves us down the ordering in a preference statement. For example, in the running example, we can get from $abcde$ to $abc\overline{de}$ in two worsening flips through the intermediate assignment $abc\overline{d}e$. Unfortunately, dominance querying is also intractable as sequences of worsening flips can be exponentially long in the number of features [2, 4].

3 Consistency of CP-networks

The semantics of CP-networks allow for any type of network: e.g. a graph with cycles, an acyclic graph or a tree. The structure of the graph has a strong influence on the consistency. In [2], it is claimed that "...Acyclic graphs always have a unique most-preferred outcome ..." and that "...acyclic graphs are alway consistent ...". Neither of these claims is correct. The acyclic CP-network $a : b \succ \overline{b}$ and $\overline{a} : \overline{b} \succ b$ has two most preferred assignments, ab and $\overline{a}\overline{b}$. And the following CP-network is acyclic but is not consistent: $a \succ \overline{a}$, $b \succ \overline{b}$, $a : c \succ \overline{c}$, $\overline{a} : \overline{c} \succ c$, $b : \overline{c} \succ c$, $\overline{b} : c \succ \overline{c}$.

In general, determining the consistency of CP-networks is NP-complete.

Theorem 1 CP-NETWORK CONSISTENCY *is NP-complete.*

Proof: Clearly it is in NP. An assignment is a polynomial witness that can be checked for optimality in time linear in the number of conditional preference constraints.

To show NP-completeness, we reduce 3-SAT, the satisfiability of 3-cnf formulae to CP-NETWORK CONSISTENCY. For each 3-cnf clause, $x \vee y \vee z$ we construct the conditional preference constraint: $\overline{x} \wedge \overline{y} : z \succ \overline{z}$. Consider any model (satisfying assignment) of the original 3-cnf formulae. At least one of x, y and z will be set true. If x or y are true, then the condition of the constructed conditional preference constraint is not satisfied and we can ignore it. If x and y are both false, then z must be true. However, setting z to true satisfies the conditional preference ordering as this is the most preferred value. Hence, any model of the original 3-cnf formulae is an optimal assignment of the constructed CP-network. The argument reverses and any optimal assignment is also a model. The CP-network is therefore consistent iff the original 3-cnf problem is satisfiable. $\Box$

We can identify a stronger condition than acyclicity that ensures that a CP-network is consistent. That is, if the network is a tree (or a forest of trees) then it must be consistent. Of course, a tree does not contain any cycles.

Theorem 2 *A tree shaped CP-network is always consistent, and an optimal assignment can be found in linear time.*

Proof: We give a linear algorithm that is always guaranteed to terminate having constructed an optimal assignment. We start with the variables at the root of the CP-network, assigning them their most preferred values. We then move down the tree. At each step, we consider any conditional preference constraint whose condition is satisfied. We assign the next variable with the most preferred value from this conditional preference constraint. If all the conditions of conditional preference constraints are violated, or there is no preference ordering on a variable, we assign it arbitrarily. For example, consider the conditional preference constraints: $a \succ \overline{a}$, $a : b \succ \overline{b}$, $b : \overline{c} \succ c$, $\overline{b} : c \succ \overline{c}$, and $b \wedge c : d \succ \overline{d}$. Starting at the root, we set a as it is unconditionally preferred over $\neg a$. As $a : b \succ \overline{b}$, we then set b. As $b : \overline{c} \succ c$, we then set $\overline{c}$. Finally, as $b \wedge \overline{c} : \overline{d} \succ d$, we set $\overline{d}$ and terminate. $\Box$

4 An alternative semantics

To recap, dominance queries in CP-networks may require finding an exponentially long chain of worsening flips, whilst consistency testing is NP-complete. Can we reduce the complexity of reasoning with CP-networks by redefining their semantics but preserving the *ceteris paribus* property?

For acyclic networks we could, for example, say that a complete assignment dominates another one if it wins first lexicographically. More precisely, we assume that features are ordered by their dependencies with independent features first. The assignment $a_1 \ldots a_n$ is better than $b_1 \ldots b_n$ iff there exists k such that

$\forall t < k \,.\, a_t = b_t$ and $a_k \succ b_k$. This is the **lexicographic ordering**. It is possible to prove that this ordering satisfies the *ceteris paribus* property. Checking this type of dominance is clearly linear in the number of features.

A more refined notion of dominance would consider all the features that are at the same level w.r.t. the hierarchy induced by the preference statements. Assuming the CP-network is acyclic, there is at least one node with an indegree of zero, corresponding to an independent feature. We can define the level of a feature in the hierarchy as the length of the maximum shortest path between it and any of the nodes representing independent features. We could then say that a complete assignment dominates another assignment if, starting at the highest level, considering features that are at the same level in the hierarchy, its assignments win on the majority. We will refer to this as **majority lexicographic ordering**. This ordering is also *ceteris paribus*. Checking dominance queries is again linear in the number of features.

Consider the CP-network: $a \succ \overline{a}$, $b \succ \overline{b}$, $a : c \succ \overline{c}$, $b : d \succ \overline{d}$, $b : e \succ \overline{e}$, $c : f \succ \overline{f}$, $d : g \succ \overline{g}$. Take the two assignments $s_1 = a\overline{b}\overline{c}\overline{d}efg$ and $s_2 = \overline{a}bcd\overline{e}fg$. According to lexicographic ordering s_1 dominates s_2 since $a \succ \overline{a}$. However, according to majority lexicographic ordering, s_2 dominates s_1. In fact, we have three levels in the hierarchy. In the first level we have $\{A, B\}$ in the second $\{C, D, E\}$ and in the third level $\{F, G\}$. Since s_1 wins on A and s_2 wins on B they tie on the first level so the second level must be considered. Here s_2 wins on 2 out of 3 features. Notice that if dominance is defined in terms of worsening flips, the 2 assignments are incomparable. Such dominance does not discriminate between one violation at a higher level and two or more at lower levels. In a lexicographic ordering, the assignment that better satisfies a higher level is the most preferred, whilst in a majority lexicographic ordering, the assignment that wins on the majority of the most important features is the most preferred.

We can compare these orderings with the original semantics based on worsening flips. First, note that both the lexicographical and the majority lexicographical orderings are total, whilst the original ordering is partial. Given two assignments, s_1 and s_2, if $s_1 \succ s_2$ then s_1 is ordered above s_2 according to both the lexicographical and the majority lexicographical orderings. But the reverse does not necessarily hold: if s_1 is above s_2 according to either the lexicographical or the majority lexicographical orderings, then either $s_1 \succ s_2$ or s_1 and s_2 are incomparable.

5 Semi-ring based approximations

One way to reduce the complexity of reasoning with CP-networks, as we have seen in the previous section, is to change their semantics from a partial order based on chains of worsening flips to a total order based on lexicographical or majority lexicographical orderings. An alternative solution is to approximate CP-networks with soft constraint satisfaction problems (SCSPs) [1]. This will allow for the use of the well established machinery underlying SCSPs to answer dominance queries in linear time. In addition, we can then combine preference reasoning with other forms of (hard and soft) constraint reasoning. This would be useful, for example, in configuration problems where we have both user preferences and hard physical constraints to satisfy.

5.1 Soft Constraints

Several formalizations of the concept of *soft constraints* are currently available. We will use the c-semi-ring formalism [1] which generalizes and can express many of the others. We assume the reader is familiar with most of the concepts so give just a brief introduction here to explain notation. A longer survey can be found at [1].

A semi-ring is a tuple $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ such that: A is a set and $\mathbf{0}, \mathbf{1} \in A$; $+$ is commutative, associative and $\mathbf{0}$ is its unit element; $\times$ is associative, distributes over $+$, $\mathbf{1}$ is its unit element and $\mathbf{0}$ is its absorbing element. A *c-semi-ring* is a semi-ring $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ in which $+$ is idempotent, $\mathbf{1}$ is its absorbing element and $\times$ is commutative. Let us consider the relation $\leq$ over A such that $a \leq b$ iff $a + b = b$. Then $\leq$ is a partial order, $+$ and $\times$ are monotone on $\leq$, $\mathbf{0}$ is its minimum and $\mathbf{1}$ its maximum, $\langle A, \leq\rangle$ is a complete lattice and, for all $a, b \in A$, $a + b = lub(a, b)$. Moreover, if $\times$ is idempotent: $+$ distributes over $\times$; $\langle A, \leq\rangle$ is a complete distributive lattice and $\times$ its glb. Informally, the relation $\leq$ compares semi-ring values and constraints. In fact, when we have $a \leq b$, we will say that *b is better than a*. Given a semi-ring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$, a finite set D (the domain of the variables) and an ordered set of variables V, a *constraint* is a pair $\langle def, con\rangle$ where $con \subseteq V$ and $def : D^{|con|} \to A$. A constraint specifies a set of variables, and assigns to each tuple of values of these variables an element of the semi-ring.

A *soft constraint problem* is a pair $\langle C, con\rangle$ where $con \subseteq V$ and C is a set of constraints: con is the set of variables of interest for the constraint set C, which may also concern variables not in con. Note that a classical CSP is a SCSP with the c-semi-ring: $S_{CSP} = \langle\{false, true\}, \vee, \wedge, false, true\rangle$. Fuzzy CSPs [6, 5] can be modeled in the SCSP framework by choosing the c-semi-ring $S_{FCSP} = \langle[0, 1], max, min, 0, 1\rangle$. Many other "soft" CSPs (probabilistic, weighted, ...) can be modeled by using suitable semi-rings ($S_{prob} = \langle[0, 1], max, \times, 0, 1\rangle$, $S_{weight} = \langle\mathcal{R}, min, +, 0, +\infty\rangle$, ...).

Finding an optimal solution in a soft constraint problem is an NP-complete problem, which is usually solved by a combination of branch and bound and constraint propagation (when possible), or via local search. On the other hand, given two solutions, checking whether one is better than another one is a very simple problem, which can be easily solved by computing the semi-ring values of the two solutions (via a multiplication of the semi-ring values of the constraints) and then by comparing such two values.

5.2 Mapping CP-networks to soft constraints

Given a set of preference statements, we will now show how to build a corresponding SCSP. We first need to identify the independent features. For each, we introduce a variable in our SCSP that will have the domain of the feature, and a unary constraint that will have as its scope just that variable and as its relation the set of possible assignments to that feature, i.e. the domain of the variable. When we have finished with the independent features, we start considering conditional statements of the form: $z_1 \wedge z_2 \wedge \cdots \wedge z_h : x1 \succ x2 \succ \cdots \succ x_k$. We introduce two new variables: V_z corresponding to set of features in Z and with domain $\{z_1, \cdots, z_h\}$ and V_x, corresponding to the features in X and with domain the Cartesian product of the domains of the features in X. We add a constraint that has as scope V_x and V_z, and, as relation the set $z_1x_1, \cdots, z_1x_k, \cdots, \cdots, z_hx_1, \cdots, z_hx_k$. The preference function will assign penalties or rewards, depending on the underlying semi-ring, according to the conditional preference statements. We also add a hard constraint between V_z and all the variables representing features that appear in V_z. The constraint simply forces assignments to the same feature, induced by assignments to different variables to be consistent (similar to the constraints between dual and ordinary variables in the hidden representation of non-binary constraints as binary ones).

After repeating this procedure for all the preference statements, we will have built a soft constraint satisfaction problem representing the preference ranking. As usual this can be represented graphically throught a constraint graph, that we call the SC-network, in which each node corresponds to a variable and will be labeled with the set of features the variable represents, and each edge corresponds to a constraint. Moreover we associate to each soft constraint c, a weight w_c and a cardinality s_c defined as the cardinality of the domain of V_x. If we assume to have n features and m cp statements we can measure the complexity in terms of new constraints that must be added to the SC-network for each statement. We'll have at most n independent statements, i.e. on the domain of a variable, and for each of the $m-n$ statements remaining at most (m) new constraints will have to be defined (one representing the actual statements and (m-1) hard constraints).

This will be useful to model the network so that the *ceteris paribus* property is respected.

As an example, the SC-network corresponding to the example in section 2, when ignoring the last statement, is shown in figure 2, where edges without weights are hard constraints. In the example note that $X = C$, $Z = A, B$.

In the following sections we will consider two different semi-rings for the soft constraints. In each case, we will see how preferences p_i and weights w_i can be set in order to match the *ceteris paribus* semantics.

First let's show how this representation can answer dominance queries in a straightforward fashion. Suppose we have two complete assignments, e.g. $\vec{v}$ and $\vec{u}$, and let us compute their preference using the semi-ring operators. That is, if $\vec{v} = v_1v_2 \cdots v_n$, where v_i is the value assigned in $\vec{v}$ to the i-th feature in the order, then the preference associated with v will be: $pref(\vec{v}) = \prod_{\nu} w_{c_\nu} f(\vec{v}\Downarrow_{c_\nu})$, where

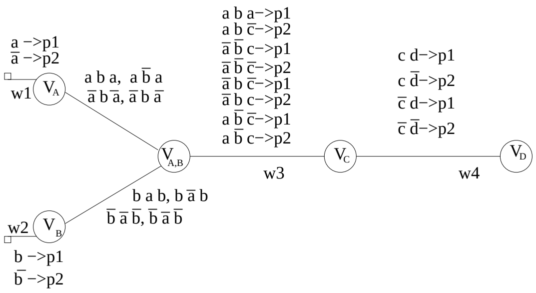

Figure 2: An SC net.

c_ν is a constraint and $\nu = 1, \cdots, T$, where T is the total number of constraints, and w_{c_ν} is the weight of that constraint. The preference of a total assignment $\overrightarrow{v}$ is obtained by combining the preferences assigned to each partial assignment, and projecting $\overrightarrow{v}$ on each constraint. The subtuple's preference, in turn, is obtained by multiplying the internal preference it is assigned in the constraint by the weight of the constraint. The combination of subtuple preferences is obtained by applying the multiplicative operator of the semi-ring.

Once we have $pref(\overrightarrow{v})$ and $pref(\overrightarrow{u})$ we compare the two values using the additive operator of the semi-ring. The original query $\overrightarrow{v} \succ \overrightarrow{u}$ becomes $pref(\overrightarrow{v}) + pref(\overrightarrow{u})) = pref(\overrightarrow{v})$. We will now show how to instantiate this general framework by choosing specific semi-rings.

5.3 Weighted soft constraints

In this section we will consider weighted CSPs, which, as noted before, are based on the min+ semi-ring $S_{WCSP} = \langle R_+, min, +, +\infty, 0 \rangle$. We assign preferences using real positive numbers (or penalties) and prefer assignments whose total penalty (which is the sum of all the penalties) is smaller. Larger penalties will be assigned to values that appear later in the ordering to the right of a preference statement. If we consider the previous example we can say that $p_1, p_2 \in R_+$ and $p_1 < p_2$.

On the other hand we want to respect the *ceteris paribus* property. To see this, in the example, consider the preference statement $c : d \succ \overline{d}$. Among any two assignments containing c, that differ only by the value given to D, the one assigning d is to be preferred. In particular, this should hold for *abcde* and $abc\overline{d}e$. In the SC-network, changing d to $\overline{d}$ increases the penalty from p_1 to p_2 in the constraint between C and D but lowers the penalty from p_2 to p_1 in the constraint from D and E. We must therefore set the weights in a precise way. In order to do this we consider the SC-network as a DAG (in general in a constraint network the edges are not directed). An edge will go from the variable of the features that appear on the left of the preference statement, into the variable of the features on the right. The setting of the penalties and the weights is done dynamically during the construction of the SC-network. Hence we must consider unary and binary soft constraints. Since we are assuming that all the preference statements on domains of independent feature will appear first in the list, when we set the corresponding unary constraint we can set the weight to 1 and the penalties increasing as the the order of the statement decreases adding 1 at each step. Notice that this implies that for each soft constraint the maximum penalty assigned is $s_c - 1$. When we add a new binary constraint, we set its weight to 1 and its penalties as before. However all the weights of the edges, in the graph to which we are adding the new edge, must be updated. The update is $w_{new} = \sum_e w_e \cdot (s_e - 1)$, where e varies among the edges that can be reached along any path from the node to which the edge points, each edge counted only once.

When a dominance query, $v \succ u$ is posed, the total penalties of the two assignments are computed: $pref(\vec{v}) = \sum_\nu w_{c_\nu} f(\vec{v}\Downarrow_{c_\nu})$, where we have used the multiplicative operator of S_{WCSP}, that is $+$. Then the two values are compared through the additive operator, so the original query becomes $min(pref(\vec{v}), pref(\vec{u})) = pref(\vec{v})$, which explicitly means $\sum_\nu w_{c\nu} f(\vec{u}\Downarrow_{c_\nu}) \geq \sum_\nu w_{c\nu} f(\vec{v}\Downarrow_{c_\nu})$ and that we will write $\vec{v} \geq_{min+} \vec{u}$.

It is possible to prove that the min+ ordering satisfies the ceteris paribus property. It is also possible to derive the correspondence between a worsening flip in the original CP-network model and a variation of the preference assigned to one or more subtuples in the SC-network. In fact, given two assignments $s_1 = a_1 \ldots a_n$ and $s_2 = b_1 \ldots b_n$, suppose there is a worsening flip from s_1 to s_2 on the i-th component. There is therefore a preference statement in which a_i is preferred to b_i. In the SC-network, there is a constraint on the variable corresponding to the i-th feature that assigns a higher penalty to the subtuple containing b_i w.r. to the one containing a_i. However, a worsening flip doesn't affect only the constraint corresponding to the preference statement on the base of which the flip is legal, but its consequences are propagated to all the constraints between the variable which has its value flipped and all other possible depending variables. It is the way weights and local penalties are set that allows us to say that to each worsening flip corresponds an increase on the total penalty.

Now we are ready to compare the two orderings $\succeq$ and $\geq_{min+}$ defined over the set of assignments. We write $\vec{v} \asymp \vec{u}$ iff $\vec{v} \succeq \vec{u}$ and $\vec{u} \succeq \vec{v}$, and $\vec{v}\ ?\ \vec{u}$ iff neither $\vec{v} \succeq \vec{u}$ nor $\vec{u} \succeq \vec{v}$ (i.e. they are incomparable).

It is possible to prove the following:

- If assignment $\vec{v}$ is preferred to $\vec{u}$ in our model, that is, $\vec{v} >_{min+} \vec{u}$, then it's either preferred to or incomparable with $\vec{u}$ in the original model.
- If two assignments are equally preferred in the min+ model than they are either equal or incomparable in the original model.
- If two assignments are ordered in the original model, they will maintain

their ordering in the min+ model.

- If two assignments are incomparable in the original model, they can be equal or ordered in any way in the min+ model.
- If two assignments are equally preferred in the original model they are equally preferred also in the min+ model.

Since the min+ ordering is total, it is a linearization of the original partial ordering. Assignments that are incomparable in the original ordering are ordered in min+. In compensation, however, dominance querying is now linear time. It is also important to notice that min+ respects the original ordering in the sense that it assignments that are preferred in the original remain preferred in the min+ ordering.

5.4 SLO soft constraints

We will now choose a different semi-ring to approximate CP-networks via soft constraints. The SLO semi-ring is defined as follows: $S_{SLO} = \langle A, max_s, min_s, \mathbf{MAX}, \mathbf{0} \rangle$, where A is the set of sequences of n integers from 0 to MAX. The additive operator, max_s and the multiplicative operator, min_s are defined as follows:

- given $s = s_1 \cdots s_n$ and $t = t_1 \cdots t_n$, $s_i = t_i, i = 1 \leq k$ and $s_{k+1} \neq t_{k+1}$, then $max_s(s,t) = q_1 \cdots q_n$ where $q_i = s_i, i \leq k$, and for $j > k$. $q_j = s_j$ if $s_{k+1} \succ t_{k+1}$ else $q_j = t_j$;
- $min_s(s,t) = q = q_1 \cdots q_n$ where $q_i = s_i, i = 1 \leq k$, and for $j > k$. $q_j = s_j$ if $s_{k+1} \prec t_{k+1}$ else $q_j = t_j$.

MAX is the sequence of n elements all equal to MAX, and **0** is the sequence of n elements equal to 0. It is easy to show that this is a semi-ring deriving the properties of max_s and min_s from the ones of max and min. It is also easy to prove that the ordering induced from max_s on A is lexicographic ordering.

We can now model a CP-network as a soft constraint problem based on S_{SLO}. First of all, we set $MAX + 1$=cardinality of the largest domain and n= number of preference statements. All the weights of the edges are set to 1. Considering the binary soft constraint as usual between a variable representing the features on the left, $F_1, \cdots, F_k$ and a variable representing the depending variable on the right, F, its preference function will map the tuple of assignments $(v_1, \cdots, v_k, v)$ to the sequence of n integers that has $MAX - i + 1$, where i is the position of v in the ordering, in the position corresponding to the constraint and MAX in every other position.

Suppose we are given two complete sequences of assignments to features, **s** and **t**, and we want to compare them. First of all we must compute $pref(s)$ and $pref(t)$. Since we are working with semi-rings this is defined as: $pref(s) = \min_s(f(s_{\downarrow c_j}))$ and $pref(t) = \min_s(f(t_{\downarrow c_j}))$, where min_s is defined over the set of constraints and $s_{\downarrow c_j}$ is the projection of s on the j-th constraint, $j = 1 \cdots n$. By the definition of min_s, $pref(s)$ and $pref(t)$ are sequences of n integers each corresponding to a constraint and containing the level at which that constraint is satisfied. To compare $\mathbf{s}$ and $\mathbf{t}$, we have to check which between $pref(s)$ and $pref(t)$ is better, which by definition of max_s means to check what is the lexicographic ordering relation between $pref(s)$ and $pref(t)$. This can be done directly since we are now dealing with integer numbers.

Does the SLO representation preserve the ceteris paribus condition? Let's consider the example: $a \succ \overline{a}$, $b \succ \overline{b}$, $a : c \succ \overline{c}$, $b : d \succ \overline{d}$, $b : e \succ \overline{e}$, $c : f \succ \overline{f}$, $d : g \succ \overline{g}$. In particular $d : g \succ \overline{g}$, means that given any two assignments such that the first one contains dg and the second $d\overline{g}$ and they differ only on the value given to G, then the first one should dominate the second one. In the arrays corresponding to the two assignments the integers would be exactly the same up to the position corresponding to the G feature. They will differ on that position and possibly on the position corresponding to features that depend on G. When checking lexicographic ordering all the values until the G position will be the same and the first assignment will win on G. The semi-ring therefore respects the ceteris paribus condition.

The relationship between a worsening flip and a change on the SC-network is as follows. When a worsening flip is performed, the integer in the position corresponding to the constraint that expresses the statement used to legalize that flip is lowered. However, some integers in subsequent positions can also change, either growing or getting smaller. As the lexicographic ordering stops as soon as it encounters an inequality, such changes are harmless.

A we did for the min+ semiring, we now compare the ordering over solutions induced by the SLO semiring to the one originally used in CP nets:

- If assignment $\vec{v}$ is preferred to $\vec{u}$ in SLO, then it's either preferred or incomparable with $\vec{u}$ in the original model.
- If two assignments are equally preferred in the SLO model, then they are equally preferred also in the original model.
- Two assignments ordered in the original model will maintain their ordering in the SLO model.
- If two assignments are incomparable in the original model, they can be ordered in any way in SLO.

The SLO model is very useful to answer dominance queries as it inherits the linear complexity from its semi-ring structure. In addition, sequences of integers, despite the way one chooses to order them, show directly the "goodness" of an assignment, i.e. where it actually is good and where it fails. We think that this representation is very expressive and can be further exploited to reason about conditional preferences.

One might wonder which of these two semi-rings (min+ and SLO) is the best linearization of the original partial order. Let's consider the sets $S_{\alpha<}$, $S_{\alpha>}$, $S_{\alpha=}$ and $S_{o?}$ containing all the pairs that are judged to be respectively in the $<$, $>$, and $=$ order in the model α, and incomparable in the original model. We will put $\alpha = o$ when referring to the original model, $\alpha = min+$ for the min+ model, and $\alpha = SLO$ when referring to the SLO model.

From the results of the previous section, we can infer that $S_{o<} \subset S_{min+<}$ and $S_{o>} \subset S_{min+>}$. Moreover, we know that $S_{o=} = S_{min+=}$ holds, and that $S_{o?} \subseteq S_{min+>} \cup S_{min+<} \cup S_{min+=}$. In other words, whatever pair is equal or ordered in some way in the original model,it maintains its ordering in the min+ model, while the pairs of assignments that are incomparable in the original model are distributed among the three possibilities in the min+ model.

Let us now consider the SLO model. From the results of this section, we can infer that $S_{o<} \subset S_{SLO<}$ and $S_{o>} \subset S_{SLO>}$. Moreover, $S_{o=} \subset S_{SLO=}$, which implies that $S_{o?} \subseteq S_{SLO>} \cup S_{SLO<}$. In other words, as in min+, also in the SLO model pairs of assignments maintain their ordering, but none of the incomparable pairs become equally preferred in the SLO model.

From the previous inclusions we can derive that $S_{min+<} \subseteq S_{SLO<}$ and $S_{min+>} \subseteq S_{SLO>}$. Incomparability of the original model becomes equality in the min+ model. However, incomparabile assignments in the original model are forced into one of the two strict orderings in the SLO model. Therefore, the min+ total order is a less brutal linearization of the original ordering of CP nets w.r.t. the SLO model. Mapping incomparability onto equality would seem to be more reasonable than mapping it onto an arbitrary ordering, since the choice is still left to the final user. We might conclude therefore that the min+ model is to be preferred to the SLO model, as far as the approximation of the original model is concerned. However maximizing the minimum reward, as in any fuzzy framework [6, 5], has proved its usefulness in problem representation. In the end, we suggest choosing the semi-ring considering the tradeoff between linearization of the order and suitability of the representation provided.

6 Conclusions and future work

We have proposed two remedies to the fact that consistency checking and dominance query answering in CP nets are NP-complete problems. First, we have given a new semantics for CP nets that is more tractable to reason with. Second, we have shown how CP nets can be approximated with soft constraints. This second remedy also allows us to integrate preference and constraint reasoning in a single formalism.

We plan to use our approach in preference elicitation systems, to guarantee the consistency of the user preferences, and to guide the user to a consistent scenario. Moreover, we also plan to study the issue of abstracting one order with another one, which has been considered here in several instances. This will allow to derive a formal theory of abstracting orders, in which we can determine whether the order is "adequately" preserved.

References

[1] S. Bistarelli, U. Montanari and F. Rossi. Semi-Ring-based Constraint Solving and Optimization. Journal of the ACM, Vol.44, n.2, March 1997.

[2] Craig Boutilier, Ronen I. Brafman, Holger H. Hoos, and David Poole. Reasoning with ceteris paribus preference statements. Proc. 15th Conf. on Uncertainty in AI, pp. 71-80, 1999.

[3] Craig Boutilier, Fahiem Bacchus, and Ronen I. Brafman. UCP-Networks: A Directed Graphical Representation of Conditional Utilities. Proc. UAI 2001.

[4] Carmel Domshlak and Ronen I. Brafman. CP-nets – Reasoning and Consistency Testing. Proc. KR'02, 2002.

[5] D. Dubois, H. Fargier and H. Prade. The calculus of fuzzy restrictions as a basis for flexible constraint satisfaction. Proc. IEEE International Conference on Fuzzy Systems. IEEE, 1993.

[6] T. Schiex. Possibilistic constraint satisfaction problems, or "How to handle soft constraints?", Proc. 8th Conf. of Uncertainty in AI, pp. 269–275, 1992.