Learning preference relations over combinatorial domains

Jérôme Lang and Jérôme Mengin [*]

Institut de Recherche en Informatique de Toulouse
31062 Toulouse Cedex, France

Abstract. We address the problem of learning preference relations over multi-attribute (or combinatorial) domains. We do so by making hypotheses about the dependence structure between attributes that the preference relation enjoys. The first hypothesis we consider is the simplest one, namely, separability (no dependences between attributes: the preference over the values of each attribute is independent of the values of other attributes); then we consider the more general case where the dependence structure takes the form of an acyclic graph. In all cases, what we want to learn is a set of local preference relations (or equivalently, a CP-net) rather than a fully specified preference relation. We consider three forms of consistency between a CP-net and a set of examples, and for two of them we give an exact characterization in the case of separability, as well as complexity results.

1 Introduction

In many applications, especially electronic commerce, it is important to learn the preferences of the user on a set of alternatives. Often, the set of alternatives has a combinatorial (or multiattribute) structure, that is, each alternative is a tuple of values for each of a given number of variables (or attributes). For instance, suppose we want to build a recommender system for movies. The available data are the preferences the user has expressed on the movies she has seen, and we would like to predict her preferences on movies she hasn't seen yet [1, 2]. The most obvious way of doing this is to describe movies using various attributes, such as genre, year, film maker etc. As another example [3, 4], one may want to build a system helping a user finding a flat from a large database. A flat is described by attributes such as price, location, size etc. and from the past interactions with the user, the system has to find out what her preferences are so that it can help us finding her ideal flat while minimizing the number of interactions.

There is an important difference between these two examples. The latter is an instance of what is usually called *preference elicitation*: the system interacts with the user by asking her specific requests, until she has found her target object or left the system [5]. The former is an instance of *passive learning*: the system has in input the whole set of preferences over films (which has been obtained independently) and has to suggest one (or several) new object(s), without any further interaction.

[*] **Acknowledgements:** many thanks to Richard Booth, Kevin Garcia, Peter Haddawy, Mathieu Serrurier and Chattrakul Sombatheera for lots of helpful discussions on this topic. Anonymous referees also gave helpful comments.

In both cases, however, the system has to learn preferences of a user (or, sometimes, a class of users) over a set of alternatives that possesses a combinatorial structure. Preferences over combinatorial domains have been investigated in detail by researchers in multiattribute decision theory (starting with [6]) and in artificial intelligence. Multiattribute decision theory has focused on *modelling* preferences, that is, giving axiomatic characterizations of classes of preference relations or utility functions, while artificial intelligence has focused on designing languages for *representing* preferences that are computationally efficient (they have to express these preferences as succinctly as possible, and to come with algorithms for finding optimal alternatives that are as fast as possible).

Classes of models and languages can be partitioned first according to the mathematical nature of the preferences they consider. Roughly speaking, one distinguishes between *ordinal* preferences (consisting in ranking the alternatives), and *numerical* preferences (consisting of utility functions mapping each alternative to some number). Here we focus on ordinal preferences. They have the advantage of often being easier to obtain from users (as it is well-known that users are ill at ease giving numerical values, except when these values are prices).

A key point, when dealing with ordinal preferences on combinatorial domains, is the dependence structure between attributes. CP-nets [7] are a graphical language for representing preferences that is based on conditional preferential independence [6]. A CP-net is composed of a directed graph representing the preferential dependences between variables, and a set of conditional tables (one for each variable), expressing, for each variable, the local preference on the values of its domain given the possible combination of values of its parents. The transitive closure of these local preferences is a partial order over the set of alternatives, which can be extended into several total orders. This is one of the most popular preference representation languages, and many facets of CP-nets have been studied, such as consistency, dominance checking, and optimization (constrained and unconstrained). One missing brick is the *learning* of CP-nets from a user.

Whereas learning or eliciting numerical preferences over multiattribute domains has been considered in some places (e.g. [8] , [9]), as far as we know learning CP-nets has been considered only by [10]; however, their approach suffers from an important drawback: it searches for a CP-net, whose associated partial order contains all the examples – we say that it *entails* the examples, – while we argue that what we intuitively look for is a CP-net whose associated partial order can be extended into at least one total order that contains the examples – we then say that the CP-net is *consistent* with them. To see this, consider the following example.

Example 1. Suppose we have two binary attributes X_1 and X_2 (with domains $\{x_1, \overline{x_1}\}$ and $\{x_2, \overline{x_2}\}$ respectively), and the set of examples $\mathcal{E} = \{x_1x_2 \succ x_1\overline{x_2},\ x_1\overline{x_2} \succ \overline{x_1}x_2,\ \overline{x_1}x_2 \succ \overline{x_1}\overline{x_2}\}$. The transitive closure of $\mathcal{E}$ is the complete preference relation $x_1x_2 \succ x_1\overline{x_2} \succ \overline{x_1}x_2 \succ \overline{x_1}\overline{x_2}$. This preference relation is *separable*, which means that the agent's preferences over the values of one attribute do not depend on the value of the other attributes: here, the agent unconditionally prefers x_1 to $\overline{x_1}$ and x_2 to $\overline{x_2}$. The fact that $x_1\overline{x_2}$ is preferred to $\overline{x_1}x_2$ simply means that when asked to choose between

X_1 and X_2, the agent prefers to give up X_2 (think of X_1 meaning "getting rich" and X_2 meaning "beautiful weather tomorrow").

What do we expect to learn from the above set of examples $\mathcal{E}$? Intuitively, since $\mathcal{E}$ is a separable preference relation, we expect to output a CP-net $\mathcal{N}$ with an empty graph and the two unconditional preference tables $x_1 \succ \overline{x_1}$ and $x_2 \succ \overline{x_2}$. However, no CP-net implies $\mathcal{E}$, whatever its dependence graph. The CP-net $\mathcal{N}$ induces a *partial* preference relation in which $x_1\overline{x_2}$ and $\overline{x_1}x_2$ are incomparable; and more generally, no CP-net can "explain" that $x_1 \succ \overline{x_1}$ is "directly preferred" to $x_2 \succ \overline{x_2}$ (i.e., with no intermediate alternative). Therefore, if we look for a CP-net *implying* each of the examples, we will simply output 'failure". On the other hand, if we look for a CP-net that is simply *consistent* with the examples we will output the above CP-net.

The explanation is that when an agent expresses a CP-net, the preference relation induced by this CP-net *is not meant to be the whole agent's preference relation, but a subset (or a lower approximation) of it.* In other terms, when an agent expresses the CP-net $\mathcal{N}$, she simply expresses that she prefers x_1 to $\overline{x_1}$ *ceteris paribus* (i.e., for a fixed value of X_2) and similarly for the preference $x_2 \succ \overline{x_1}$; the fact that $x_1\overline{x_2}$ and $\overline{x_1}x_2$ are incomparable in $\mathcal{N}$ surely does not mean that the user really sees them incomparable, but, more technically, that CP-nets are not expressive enough for representing the missing preference $x_1\overline{x_2} \succ \overline{x_1}x_2$[1].

Another way of explaining this difference is that there are two ways of seeing a CP-net: either we identify it with its induced partial preference relation, or we identify it with the set of all complete preference relations that extend this partial preference relation. With the second view, the goal of the learning algorithm is to learn a collection of tables such that the examples are consistent with some preference relation in this set.

In the next section, we give some background on preferences on combinatorial domains, and then we introduce three kinds of compatibility between a CP-net and a set of examples, namely, *weak compatibility*, *strong compatibility* and *implicative compatibility*. We then focus on the simplest case, namely separable preference relations (which extend CP-nets with no preferential dependences): we show how weak compatibility wan be reduced to a satisfiability problem and *vice versa*, which allows us to show that deciding weak compatibility is NP-complete ; we also give a way of deciding implicative consistency. We finally go further and show how to extend these results to the situation where the graphical component of the CP-net is fixed but can contain dependencies. In the last section, we conclude with some hints about the general case where we have to learn both the graph *and* the tables.

2 The learning problem

2.1 Multiattribute domains

We assume that we have a finite set $\mathcal{V} = \{X_1, \ldots, X_n\}$ of *attributes* with associated finite *domains* $D_1, \ldots, D_n$. $D = D_1 \times \ldots \times D_n$ is the set of all complete assignments, called *outcomes*.

[1] If we want to do this, we have to resort to a more expressive language such as TCP-nets [11] or conditional preference theories [12].

For any nonempty subset $\mathcal{X}$ of $\mathcal{V}$, we let $D_{\mathcal{X}} = \times_{X_i \in \mathcal{X}} D_i$. Elements of $D_{\mathcal{X}}$ are called $\mathcal{X}$-assignments (i.e. value assignments for all attributes in X); they are denoted using vectorial notation, *e.g.*, $\boldsymbol{x}$. For any disjoint subsets $\mathcal{X}$ and $\mathcal{Y}$ of $\mathcal{V}$, $\boldsymbol{x} \in D_{\mathcal{X}}$ and $\boldsymbol{y} \in D_{\mathcal{Y}}$ then the concatenation of $\boldsymbol{x}$ and $\boldsymbol{y}$, $\boldsymbol{xy}$, is formally defined as $\boldsymbol{x} \cup \boldsymbol{y}$ – i.e., it is the $X \cup Y$-assignment which assigns to attributes in X (resp. Y) the value assigned by $\boldsymbol{x}$ (resp. $\boldsymbol{y}$).

An attribute X_i is *binary* if D_i has two elements, which by convention we note x_i and $\overline{x_i}$.

A *preference relation* on a multiattribute domain D is a weak order on D, that is, a reflexive and transitive binary relation $\succeq$. If furthermore $\succeq$ is connected, that is, if for every $\boldsymbol{x}, \boldsymbol{y} \in D$ we have either $\boldsymbol{x} \succeq \boldsymbol{y}$ or $\boldsymbol{y} \succeq \boldsymbol{x}$ then $\succeq$ is a *complete* preference relation. A *strict preference relation* $\succ$ is an order on D, that is, an irreflexive and transitive (thus asymmetric) binary relation. If moreover $\succ$ is connected then $\succ$ is a *linear preference relation*. From a preference relation $\succeq$ we define a strict preference relation in the usual way: $\boldsymbol{x} \succ \boldsymbol{y}$ iff $\boldsymbol{x} \succeq \boldsymbol{y}$ and not ($\boldsymbol{y} \succeq \boldsymbol{x}$).

Suppose now that we have a set of *examples* $\mathcal{E}$, where each example is a pair of distinct outcomes $(\boldsymbol{x}, \boldsymbol{y})$ (also denoted, equivalently, by $\boldsymbol{x} \succ \boldsymbol{y}$) such that $\boldsymbol{x}$ is preferred to $\boldsymbol{y}$. We sometimes denote such an example by $\boldsymbol{x} \succ \boldsymbol{y}$ rather than $(\boldsymbol{x}, \boldsymbol{y})$. In the following, $\mathcal{E}$ denotes a finite set of examples. In a recommender system for examples, these example may have been recorded in the course of a user interaction with the system.

Ideally, we would like to induce from these examples the complete ordering of all outcomes for the user. That is, we would like to learn how to order every unordered pair of distinct outcomes $\{\boldsymbol{x}, \boldsymbol{y}\}$. Therefore, our target is a linear preference, or else, a complete preference relation (if we allow for indifferences)

However, if we call $\mathcal{O}$ the set of total strict orders on D, and if the n attributes have m possible values each, there are m^n outcomes, thus $m^n!$ total strict orders in $\mathcal{O}$. There are too many elements in $\mathcal{O}$ to represent them efficiently, so we will have to restrict ourselves to a smaller hypothesis space. In the next section, we briefly present CP-nets, which will form our hypothesis space.

2.2 *Ceteris paribus* preferences and CP-nets

Preferences between outcomes that differ in the value of one attribute only, all other attributes being equal (or *ceteris paribus*) are often easy to assert, and to understand. CP-nets [7] are a graphical language for representing such preferences, that is based on conditional preferential independence [6]. A CP-net is composed of a directed graph representing the preferential dependences between attributes, and a set of conditional preference tables (one for each attribute), expressing, for each attribute, the local preference on the values of its domain given the possible combination of values of its parents. This is one of the most popular preference representation languages on multiattribute domains, and many facets of CP-nets have been studied, such as consistency, dominance checking, and optimization (constrained and unconstrained).

Let us call a *swap* any pair of outcomes $\{\boldsymbol{x}, \boldsymbol{y}\}$ that differ in the value of one attribute only, and let us then call *swapped attribute* the attribute that has different values in $\boldsymbol{x}$ and $\boldsymbol{y}$. A CP-net specifies, for every swap $\{\boldsymbol{x}, \boldsymbol{y}\}$, which of $\boldsymbol{x} \succ \boldsymbol{y}$ or $\boldsymbol{y} \succ \boldsymbol{x}$ is true.

This can be achieved in a compact manner when there are many *independences* among attributes.

Example 2. Consider three attributes A, B and C, with respective domains $\{a, \overline{a}\}$, $\{b, \overline{b}\}$ and $\{c, \overline{c}\}$, and suppose that the four swaps on B are ordered as follows: $abc \succ a\overline{b}c$, $\overline{a}\overline{b}c \succ \overline{a}bc$, $ab\overline{c} \succ a\overline{b}\overline{c}$, $\overline{a}\overline{b}\overline{c} \succ \overline{a}b\overline{c}$. We can see that, irrespective of the value of C, if a is the case, then b is preferred to $\overline{b}$, whereas if $\overline{a}$ is the case, then $\overline{b}$ is preferred to b. We can represent this ordering on the B-swaps with two *conditional preferences*: $a : b \succ \overline{b}$ and $\overline{a} : \overline{b} \succ b$, and say that, given A, B is (conditionally) preferentially independant of C.

Definition 1 *Let $\{\mathcal{X}, \mathcal{Y}, \mathcal{Z}\}$ be a partition of the set $\mathcal{V}$ and $\succ$ a linear preference relation over D. $\mathcal{X}$ is* (conditionally) preferentially independent *of $\mathcal{Y}$ given $\mathcal{Z}$ (w.r.t. $\succ$) if and only if for all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in D_{\mathcal{X}}$, $\boldsymbol{y}_1, \boldsymbol{y}_2 \in D_{\mathcal{Y}}$, $\boldsymbol{z} \in D_{\mathcal{Z}}$,*

$$\boldsymbol{x}_1\boldsymbol{y}_1\boldsymbol{z} \succ \boldsymbol{x}_2\boldsymbol{y}_1\boldsymbol{z} \textit{ iff } \boldsymbol{x}_1\boldsymbol{y}_2\boldsymbol{z} \succ \boldsymbol{x}_2\boldsymbol{y}_2\boldsymbol{z}$$

Definition 2 *A* CP-net *over attributes $V = \{X_1, \ldots, X_n\}$ with domains $D_1, \ldots, D_n$ is a pair $\mathcal{N} = \langle G, P \rangle$ where G is a directed graph over $x_1, \ldots, x_n$ and P is a set of conditional preference tables $CPT(X_i)$ for each $X_i \in V$. For attribute X_i, we denote by $\mathsf{Par}(X_i)$ (resp. $\mathsf{NonPar}(X_i)$) the set of parents of X_i in G (resp. $\mathcal{V} - (\{X_i\} \cup \mathsf{Par}(X_i))$). Each conditional preference table is a list of rows of the form $\boldsymbol{u} : x_i^j \succ x_i^k$: it associates a total order on D_i with each instantiation $\boldsymbol{u}$ of $\mathsf{Par}(X_i)$, and indicates, for every possible instantiation $\boldsymbol{z}$ of $\mathsf{NonPar}(X_i)$, that $\boldsymbol{u}x_i^j\boldsymbol{z} \succ \boldsymbol{u}x_i^k\boldsymbol{z}$. When all attributes of $\mathcal{V}$ are binary, a CP-net over $\mathcal{V}$ is said to be* propositional.

Example 3. A CP-net over attributes A, B and C, with respective domains $\{a, \overline{a}\}$, $\{b, \overline{b}\}$ and $\{c, \overline{c}\}$ is:

$$\begin{tikzcd}
\boxed{A} \arrow[r] & \boxed{B} \arrow[r] & \boxed{C} \\
a \succ \overline{a} & \begin{array}{l} a : b \succ \overline{b} \\ \overline{a} : \overline{b} \succ b \end{array} & \begin{array}{l} b : c \succ \overline{c} \\ \overline{b} : \overline{c} \succ c \end{array}
\end{tikzcd}$$

where $\begin{tikzcd} \boxed{X} \arrow[r] & \boxed{Y} \end{tikzcd}$ means "X is a parent of Y". The associated ordering of the swaps is:

$$\begin{tikzcd}
 & & & \overline{a}\overline{b}\overline{c} \arrow[dr] \arrow[ddrrr, bend left] & & & \\
abc \arrow[r] \arrow[rrrrr, bend left] \arrow[drrr, bend right] & ab\overline{c} \arrow[r] \arrow[rrrrrr, bend right] & a\overline{b}\overline{c} \arrow[ur] \arrow[dr] & & \overline{a}\overline{b}c \arrow[r] & \overline{a}bc \arrow[r] & \overline{a}b\overline{c} \\
 & & & a\overline{b}c \arrow[ur] & & &
\end{tikzcd}$$

where $\boldsymbol{x} \longrightarrow \boldsymbol{y}$ means "$\boldsymbol{x}$ is preferred to $\boldsymbol{y}$".

Although a CP-net only specifies an ordering of all swaps, we are naturally interested in the transitive closure of this ordering; for a CP-net $\mathcal{N}$, we note $\succ_{\mathcal{N}}$ this transitive closure. Note that this relation $\succ_{\mathcal{N}}$ may not be total, and it may not be a strict order since it may contain cycles, and thus not be irreflexive. We know from [7] that if G is acyclic, then $\succ_{\mathcal{N}}$ is a strict order (i.e. contains no cycles). In this case we say that $\mathcal{N}$ is consistent; $\succ_{\mathcal{N}}$ may still not be total, it can then be completed in a number of total strict orders of $\mathcal{O}$. If $\succ$ is one of them, that is, if $\succ \in \mathcal{O}$ and if $\succ_{\mathcal{N}} \subseteq \succ$, we say that $\succ$ is a *completion* of $\mathcal{N}$. When $\succ_{\mathcal{N}}$ is not irreflexive, we say that $\mathcal{N}$ is *inconsistent*.

We recall the following property [7], which will be useful later:

Proposition 1 *([7], Th. 7 and 8) Let $\mathcal{N}$ be an acyclic CP-net and $\boldsymbol{x}$, $\boldsymbol{y}$ two outcomes. Then $\boldsymbol{x} \succ \boldsymbol{y}$ is implied by $\mathcal{N}$ if and only if there is a sequence of swaps $\{\boldsymbol{x}^0, \boldsymbol{x}^1\}, \{\boldsymbol{x}^1, \boldsymbol{x}^2\}, \ldots, \{\boldsymbol{x}^{k-1}, \boldsymbol{x}^k\}$ such that $\boldsymbol{x}^0 = \boldsymbol{x}$, $\boldsymbol{x}^k = \boldsymbol{y}$, and for every $0 \leq i < k$, $\boldsymbol{x}^i \succ \boldsymbol{x}^{i+1}$, that is, if X_{j_i} is the attribute swapped between $\boldsymbol{x}^i$ and $\boldsymbol{x}^{i+1}$, and if $\boldsymbol{u}$ is the vector of values commonly assigned by $\boldsymbol{x}$ and $\boldsymbol{y}$ to the parents of X_i, then $\mathcal{N}$ contains $\boldsymbol{u} : x^i_{j_i} \succ x^{i+1}_{j_i}$;*

2.3 Different forms of compatibility between sets of examples and CP-nets

We said earlier that the target of our learning process should be a strict total order[2] over all outcomes, but we have just seen that a CP-net $\mathcal{N}$ does not in general correspond to such an order, since the preference relation $\succ_{\mathcal{N}}$ induced from $\mathcal{N}$ is generally not complete. Actually, $\succ_{\mathcal{N}}$ can be seen as the set of all its completions, that is, a CP-net expresses a *set of linear preference relations*.

If an example $(\boldsymbol{x}, \boldsymbol{y})$ is a swap, then either $\boldsymbol{x} \succ_{\mathcal{N}} \boldsymbol{y}$ or $\boldsymbol{y} \succ_{\mathcal{N}} \boldsymbol{x}$, and clearly we would like $\mathcal{N}$ to be in agreement with the example, that is, for instance, such that $\boldsymbol{x} \succ_{\mathcal{N}} \boldsymbol{y}$. But if $(\boldsymbol{x}, \boldsymbol{y})$ is not a swap, there may be completions $\succ$ and $\succ'$ of $\succ_{\mathcal{N}}$ such that $\boldsymbol{x} \succ \boldsymbol{y}$ and $\boldsymbol{y} \succ' \boldsymbol{x}$.

So we should start by discussing the possible ways of measuring to which extent a given CP-net generalises from a given set of examples.

Definition 3 *Let $\mathcal{N}$ be a CP-net over $\mathcal{V}$. An example $(\boldsymbol{x}, \boldsymbol{y})$ is*

- *implied by $\mathcal{N}$ if $\boldsymbol{x} \succ \boldsymbol{y}$ for every completion $\succ$ of $\succ_{\mathcal{N}}$;*
- *consistent with $\mathcal{N}$ if there is a completion $\succ$ of $\succ_{\mathcal{N}}$ such that $\boldsymbol{x} \succ \boldsymbol{y}$.*

Furthermore, we will say that a set of examples $\mathcal{E}$ is:

- *implied by $\mathcal{N}$ if for every completion $\succ$ of $\succ_{\mathcal{N}}$, for every $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{E}$, $\boldsymbol{x} \succ \boldsymbol{y}$ (that is, if every example is implied by $\mathcal{N}$);*
- *globally (or strongly) consistent with $\mathcal{N}$ if there is a completion $\succ$ of $\succ_{\mathcal{N}}$ such that, for every $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{E}$, $\boldsymbol{x} \succ \boldsymbol{y}$.*
- *weakly consistent with $\mathcal{N}$ if for every $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{E}$, there is a completion $\succ$ of $\succ_{\mathcal{N}}$ such that, $\boldsymbol{x} \succ \boldsymbol{y}$ (that is, if every example $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{E}$ is individually consistent with $\mathcal{N}$).*

[2] Our methodology and results would easily carry on to the problem of learning nonstrict preference relations (where indifference is allowed). We stick here to strict preference relation because the presentation is simpler.

Clearly, strong consistency implies weak consistency, and, if $\mathcal{N}$ is consistent and if $\mathcal{E}$ is implied by $\mathcal{N}$, then it is strongly consistent with $\mathcal{N}$.

Notice that an example $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{E}$ is implied by $\mathcal{N}$ if and only if $\boldsymbol{x} \succ_{\mathcal{N}} \boldsymbol{y}$. Also, it can be shown easily that $\mathcal{E}$ is strongly consistent with $\mathcal{N}$ if and only if the transitive closure of $\succ_{\mathcal{N}} \cup \mathcal{E}$ contains no cycle.

[10] try to learn a CP-net that implies a given set of examples. However, there are cases where learning such a CP-net may not be appropriate.

We have (cf. Ex. 1) that the transitive closure of a set of examples may be a total order over the set of alternatives, that the preferences specified by this set of examples may be separable, although they cannot be implied by any CP-net. Consider now a second example.

Example 4. $\mathcal{E} = \{x_1x_2 \succ x_1\overline{x_2},\ x_1\overline{x_2} \succ \overline{x_1}x_2\}$. This set of examples is obtained from that of Example 1 by removing the third example. Intuitively, there is no reason to think that the agent's preference relation is not separable. However, there exists no CP-net with $G = \langle \mathcal{V}, \emptyset \rangle$ that implies $\mathcal{E}$. When allowing for the more complicated structure $G = \langle \mathcal{V}, \{X_1 \to X_2\} \rangle$ we obtain the conditional preference tables $x_1 \succ \overline{x_1}$, $x_1 : x_2 \succ \overline{x_2}$, $\overline{x_1} : \overline{x_2} \succ x_2$, whose induced preference relation is $x_1x_2 \succ x_1\overline{x_2} \succ \overline{x_1}\overline{x_2} \succ \overline{x_1}x_2$. Therefore, asking for a CP-net that *implies* $\mathcal{E}$ leads to a more complicated structure than necessary (a preferential dependence between X_1 and X_2 whereas nothing tells us there should be one).

In both Examples 1 and 4, $\mathcal{E}$ is strongly (and *a fortiori* weakly) consistent with the separable CP-net $\{x_1 \succ \overline{x_1},\ x_2 \succ \overline{x_2}\}$. Therefore weak and strong compatibility look like more reasonable notions when it comes to learn CP-nets.

The difference between weak and strong compatibility (that we could also call local and global compatibility) is more subtle. While strong compatibility requires that the examples are all compatible with a *single* preference relation extending $\mathcal{N}$, weak compatibility only requires each example to be compatible with *some* preference relation extending $\mathcal{N}$. As a consequence, if $\mathcal{E}$ is inconsistent (for instance because it contains two opposite examples $\boldsymbol{x} \succ \boldsymbol{y}$ and $\boldsymbol{y} \succ \boldsymbol{x}$, or more generally a series of examples $\boldsymbol{x}_1 \succ \boldsymbol{x}_2, \boldsymbol{x}_2 \succ \boldsymbol{x}_3, \ldots, \boldsymbol{x}_p \succ \boldsymbol{x}_1$), then there cannot be an $\mathcal{N}$ such that $\mathcal{E}$ is strongly consistent with $\mathcal{N}$, whereas it might still be the case that there is an $\mathcal{N}$ such that $\mathcal{E}$ is weakly consistent with $\mathcal{N}$, as it can be seen on the following example.

Example 5. $\mathcal{E} = \{\overline{x_1}x_2 \succ x_1\overline{x_2},\ x_1\overline{x_2} \succ \overline{x_1}x_2\}$. $\mathcal{E}$ is clearly inconsistent, and yet $\mathcal{E}$ is weakly consistent with the separable CP-net $\mathcal{N}$ whose tables are $\{x_1 \succ \overline{x_1}, x_2 \succ \overline{x_2}\}$. Because $\mathcal{E}$ is inconsistent, it is not strongly consistent with $\mathcal{N}$ (nor with any other CP-net).

Note that if $\mathcal{N}$ is itself inconsistent (i.e., possesses cycles), then no set of examples can be strongly consistent with $\mathcal{N}$ whereas there are sets of examples which are implied by $\mathcal{N}$ (and, a fortiori, are weakly consistent with $\mathcal{N}$).

If the examples all come from a single user and are reliable, then weak consistency is much too weak. However, if they come from multiple users (given that we want to learn the generic preferences of a group of users), or a single user in different contexts, then it becomes reasonable: for instance, we may learn that all users in the group unconditionally prefer x_1 to $\overline{x_1}$ and x_2 to $\overline{x_2}$, whereas their preferences between $x_1\overline{x_2}$ and $\overline{x_1}x_2$ may differ (think as x_1 and x_2 as, respectively, "being invited to a fine dinner" and "receiving a \$50 award"). Moreover, weak consistency is relevant even for a single user if we allow for errors or for changes of mind.

In the sequel, we will focus on weak consistency, because it is easier to characterize. The computation of strong consistency is left for future research.

Definition 4 *Let $G = \langle \mathcal{V}, E \rangle$ be a graph over $\mathcal{V}$ and $\mathcal{E}$ a set of examples. $\mathcal{E}$ is*

- weakly G-compatible *if there there exists a CP-net $\mathcal{N}$ with graph G such that $\mathcal{E}$ is weakly consistent with $\mathcal{N}$.*
- strongly G-compatible *if there there exists a CP-net $\mathcal{N}$ with graph G such that $\mathcal{E}$ is strongly consistent with $\mathcal{N}$.*
- implicatively G-compatible *if there there exists a CP-net $\mathcal{N}$ with graph G such that $\mathcal{E}$ is implied by $\mathcal{N}$.*

Observation 1 *If G is acyclic then implicative G-compatibility implies strong G-compatibility, and strong G-compatibility implies weak G-compatibility.*

This is just because if G is acyclic then any CP-net $\mathcal{N}$ whose associated graph is G is consistent. We also have the obvious fact:

Observation 2 *If $G \subseteq G'$ and $\mathcal{E}$ is weakly (resp. strongly, implicatively) G'-compatible, then $\mathcal{E}$ is weakly (resp. strongly, implicatively) G-compatible.*

As usual in machine learning, we have a preference for learning *simple* structures: here, simplicity is measured by the size of the tables, which is directly related to the number of edges in the graph. In particular, the simplest CP-nets are the separable ones (i.e., those without edges); next Section is dedicated to this specific class of CP-nets.

3 Learning separable preference relations

3.1 Computing a CP-net weakly consistent with a set of examples

We start by the simplest case of separable preference relations over binary domains, that is:

- $G = \emptyset$;
- $D = \{x_1, \overline{x_1}\} \times \ldots\ldots \{x_n, \overline{x_n}\}$.

We first define the following translation from sets of examples $\mathcal{E}$ to sets of clauses. Let $\boldsymbol{x} \succ \boldsymbol{y}$ be an example. Define $Diff(\boldsymbol{x}, \boldsymbol{y}) = \{x_i \mid (\boldsymbol{x})_i = x_i \text{ and } (\boldsymbol{y})_i = \overline{x_i}\} \cup \{\overline{x_i} \mid (\boldsymbol{x})_i = \overline{x_i} \text{ and } (\boldsymbol{y})_i = x_i\}$.

Now, with each example $\boldsymbol{x} \succ \boldsymbol{y}$ we associate the following clause $C_{\boldsymbol{x}\succ\boldsymbol{y}}$ that contains x_i iff $x_i \in Diff(\boldsymbol{x}, \boldsymbol{y})$ and $\neg x_i$ iff $\overline{x_i} \in Diff(\boldsymbol{x}, \boldsymbol{y})$.

For instance, if $\boldsymbol{x} = x_1\overline{x_2}x_3x_4$ and $\boldsymbol{y} = \overline{x_1}x_2x_3\overline{x_4}$ then $Diff(\boldsymbol{x}, \boldsymbol{y}) = \{x_1, \overline{x_2}, x_4\}$ and $C_{\boldsymbol{x}\succ\boldsymbol{y}} = x_1 \vee \neg x_2 \vee x_4$.

If $\mathcal{E}$ is a set of examples then $\Phi_{\mathcal{E}}$ is the set of clauses defined by $\Phi_{\mathcal{E}} = \{C_e \mid e \in \mathcal{E}\}$.

Lastly, we define the following one-to-one correspondence between truth assignments over $\{x_1, \dots, x_n\}$ and separable CP-nets over $\mathcal{V}$. If M is such a truth assignment, then the $\mathcal{N}_M$ contains the preference table $x_i \succ \overline{x_i}$ for every i such that $M \models x_i$ and the preference table $\overline{x_i} \succ x_i$ for every i such that $M \models \neg x_i$. For instance, if $M(x_1) = \top$, $M(x_2) = \bot$, $M(x_3) = \bot$ and $M(x_4) = \top$ then $\mathcal{N}_M$ contains the preference tables $\{x_1 \succ \overline{x_1},\ \overline{x_2} \succ x_2,\ \overline{x_3} \succ x_3,\ x_4 \succ \overline{x_4}\}$.

Proposition 2
$M \models \Phi_{\mathcal{E}}$ if and only if $\mathcal{E}$ is weakly consistent with $\mathcal{N}_M$.

Before proving Proposition 2 we establish the following simple Lemma, which is a consequence of Proposition 1 (since it is very simple, we give its proof anyway).

Lemma 1. *Let $\mathcal{N}$ be a CP-net with G containing no edges, and $\boldsymbol{y} \neq \boldsymbol{x}$. Then $\mathcal{N} \models \boldsymbol{x} \succ \boldsymbol{y}$ if and only if $\mathcal{N}$ contains $x_i \succ \overline{x_i}$ for every $x_i \in Diff(\boldsymbol{x}, \boldsymbol{y})$ and $\overline{x_i} \succ x_i$ for every $\overline{x_i} \in Diff(\boldsymbol{x}, \boldsymbol{y})$.*

Proof: Without loss of generality, let $\boldsymbol{x} = x_1 \dots x_n$ and $\boldsymbol{y} = \overline{x_1} \dots \overline{x_i} x_{i+1} \dots x_n$. If $\mathcal{N}$ contains $x_1 \succ \overline{x_1}, \dots, x_i \succ \overline{x_i}$ then $\mathcal{N} \models \boldsymbol{x} \succ \boldsymbol{y}$. Conversely, without loss of generality assume $\mathcal{N}$ does not contain $x_1 \succ \overline{x_1}$, which implies that it contains $\overline{x_1} \succ x_1$. Consider a lexicographic preference relation $\succ$ on D in which X_1 is the most important attribute. We have $\boldsymbol{y} \succ \boldsymbol{x}$, and yet $\succ$ extends $\succ_{\mathcal{N}}$, therefore we cannot have $\mathcal{N} \models \boldsymbol{x} \succ \boldsymbol{y}$. ■

Now we establish Proposition 2:

Proof:

- *($\Leftarrow$)* Let M be an interpretation and $\mathcal{N}_M$ the CP-net associated with $\mathcal{N}$. Assume $M \not\models \Phi_{\mathcal{E}}$, *i.e.*, there exists an example $\boldsymbol{x} \succ \boldsymbol{y}$ in $\mathcal{E}$ such that $M_{\mathcal{N}} \models \neg C_{\boldsymbol{x} \succ \boldsymbol{y}}$. Without loss of generality, let $\boldsymbol{x} = x_1 \dots x_n$ and $\boldsymbol{y} = \overline{x_1} \dots \overline{x_i} x_{i+1} \dots x_n$. Then we have $M \models \neg x_1 \wedge \dots \wedge \neg x_i$, therefore $\mathcal{N}_M$ contains $x_1 \succ \overline{x_1}, \dots, x_i \succ \overline{x_i}$, which by Lemma 1 implies that $\mathcal{N} \models \boldsymbol{y} \succ \boldsymbol{x}$, therefore $\mathcal{N}_M$ is not consistent with $\boldsymbol{x} \succ \boldsymbol{y}$, and *a fortiori*, $\mathcal{N}_M$ is not weakly consistent with $\mathcal{E}$.
- *($\Rightarrow$)* Let M be an interpretation over $x_1, \dots, x_n$ and $\mathcal{N}_M$ the CP-net associated with M. Assume that $\mathcal{N}_M$ is not weakly consistent with $\mathcal{E}$, which means that there exists an example $\boldsymbol{x} \succ \boldsymbol{y}$ in $\mathcal{E}$ such that $\mathcal{N}_M \models \boldsymbol{y} \succ \boldsymbol{x}$. Without loss of generality, let $\boldsymbol{x} = x_1 \dots x_n$ and $\boldsymbol{y} = \overline{x_1} \dots \overline{x_i} x_{i+1} \dots x_n$. By Lemma 1 this implies that $\mathcal{N}_M$ contains $\overline{x_1} \succ x_1, \dots, \overline{x_i} \succ x_i\}$. This implies that $M \models \neg x_1 \wedge \dots \wedge \neg x_i$, therefore $M \not\models C_{\boldsymbol{x} \succ \boldsymbol{y}}$, and *a fortiori*, $M \not\models \neg \Phi_{\mathcal{E}}$.

■

Corollary 1. *$\mathcal{E}$ is weakly $\langle \mathcal{V}, \emptyset \rangle$-compatible if and only if $\Phi_{\mathcal{E}}$ is satisfiable.*

This correspondence between unconditional CP-nets and interpretations over $\mathcal{V}$ enables us to draw the following result:

Proposition 3 *Deciding whether a set of examples is weakly $\langle \mathcal{V}, \emptyset \rangle$- compatible is* NP-*complete .*

Proof: Membership is easy: given a set of examples $\mathcal{E}$, guess an unconditional CP- net $\mathcal{N}$ and check that $\mathcal{E}$ is weakly consistent with $\mathcal{N}$, which can be done in time $\mathcal{O}(|\mathcal{E}|.n)$ using Lemma 1. For hardness we use the following reduction from 3SAT. Let $\Phi = \{C_1, \ldots, C_p\}$ be a set of 3-clauses. For every $C = l_1 \vee l_2 \vee l_3$ in Φ create an example $e_C = (\boldsymbol{x} \succ \boldsymbol{y})$ with

– $\boldsymbol{x} = \varepsilon_1.x_1\varepsilon_2.x_2 \ldots \varepsilon_n.x_n$,
– $\boldsymbol{y} = \varepsilon'_1.x_1\varepsilon'_2.x_2 \ldots \varepsilon'_n.x_n$,
– for every i, $\varepsilon_i.x_i = \begin{cases} x_i & \text{if } l_j = x_i \text{ for some } j \\ \neg x_i & \text{if } l_j = \neg x_i \text{ for some } j \\ x_i & \text{otherwise} \end{cases}$
– for every i, $\varepsilon'_i.x_i = \begin{cases} \neg x_i & \text{if } l_j = x_i \text{ for some } j \\ x_i & \text{if } l_j = \neg x_i \text{ for some } j \\ x_i & \text{otherwise} \end{cases}$

Now, let $\mathcal{E}_\Phi = \{e_C \mid C \in \Phi\}$. For example, if $\Phi = \{x_1 \vee \neg x_2 \vee x_3, \neg x_1 \vee x_2 \vee x_4, x_2 \vee x_3 \vee x_4\}$ then $\mathcal{E}_\Phi = \{x_1\overline{x_2}x_3x_4 \succ \overline{x_1}x_2\overline{x_3}x_4, \overline{x_1}x_2x_3x_4 \succ x_1\overline{x_2}x_3\overline{x_4}, x_1x_2x_3x_4 \succ x_1\overline{x_2x_3x_4}\}$. We easily check that $\Phi_{\mathcal{E}_\Phi} = \mathcal{E}$, therefore, using Corollary 1 we get that Φ is satisfiable if and only if $\mathcal{E}$ is xeakly $\langle \mathcal{V}, \emptyset \rangle$-compatible. ■

The generalization to non-binary domains is not difficult. Instead of having one propositional symbol per attribute, we have one propositional symbol for each pair of values of a attribute. For instance, if we have a attribute X whose domain is $\{d_1, d_2, d_3\}$ then we have the three propositional symbols $d_1 \succ d_2$, $d_1 \succ d_3$ and $d_2 \succ d_3$. The main difference with the binary case is the transitivity requirement. Let $Trans = \bigwedge_{X_i \in \mathcal{V}} Trans_{X_i}$ be the propositional formula expressing transitivity – for instance, for $D_1 = \{d_1, d_2, d_3\}$ we have $Trans_{X_1} = (d_1 \succ d_2 \wedge d_2 \succ d_3 \rightarrow d_1 \succ d_3) \wedge (d_1 \succ d_3 \wedge \neg(d_2 \succ d_3) \rightarrow \neg(d_1 \succ d_2) \wedge$ .... Note that $Trans$ is polynomially long.

The one-to-one correspondence between interpretations and CP-nets now works only for interpretations satisfying $Trans$, and Proposition 2 is generalized into:

Proposition 4 $M \models \Phi_\mathcal{E} \wedge Trans$ *if and only if* $\mathcal{N}_M$ *is weakly consistent with* $\mathcal{N}_M$.

And Corollary 1 becomes:

Corollary 2. $\mathcal{E}$ *is weakly* $\langle \mathcal{V}, \emptyset \rangle$*-compatible if and only if* $\Phi_\mathcal{E} \wedge Trans$ *is satisfiable.*

3.2 Computing a CP-net implied by a set of examples

It is easy to characterize whether there exists a CP-net that implies $\mathcal{E}$. With each example $\boldsymbol{x} \succ \boldsymbol{y}$ we associate the following cube (conjunction of literals) $\Gamma_{\boldsymbol{x} \succ \boldsymbol{y}}$: it contains x_i iff $x_i \in Diff(\boldsymbol{x}, \boldsymbol{y})$ and $\neg x_i$ iff $\overline{x_i} \in Diff(\boldsymbol{x}, \boldsymbol{y})$. Let $\Gamma_\mathcal{E} = \bigwedge\{\Gamma_e \mid e \in \mathcal{E}\}$. Using Lemma 1, we get the following result:

Proposition 5 $M \models \Gamma_\mathcal{E}$ *if and only if* $\mathcal{N}_M$ *implies* $\mathcal{E}$.

Corollary 3. *There exists a CP-net implying* $\mathcal{E}$ *if and only if* $\Gamma_\mathcal{E}$ *is satisfiable.*

Corollary 4. *Deciding whether there exists a CP-net implying* $\mathcal{E}$ *is in* P.

4 Learning non-separable preference relations over a fixed acyclic structure

We no longer assume that the preference relation is separable, but we assume that it can be represented by a CP-net over a fixed acyclic graph $(\mathcal{V}, E)$: we are given a set of examples of pairwise comparisons $\mathcal{E}$, and we want to generate a set $\mathcal{P}$ of preference tables for the graph $(\mathcal{V}, E)$ such that the CP-net $\mathcal{N} = ((\mathcal{V}, E), \mathcal{P})$, is weakly compatible with $\mathcal{E}$. Ideally, we would be able to generalize in a simple way the technique given from Section 3: that is, we translate examples into propositional clauses, the models of which would correspond to CP-nets that are weakly consistent with the examples. Unfortunately, this is not simple.

The following result, by [7], provides a condition that ensures weak compatibility of a CP-net with a given example $\boldsymbol{o} \succ \boldsymbol{o}' \in \mathcal{E}$:

Proposition 6 *([7], Corollary 4.1) If $\mathcal{N}$ is an acyclic CP-net, $\boldsymbol{o}$, $\boldsymbol{o}'$ are two outcomes, and if there exists an attribute $X \in \mathcal{V}$ such that $\boldsymbol{o}$ and $\boldsymbol{o}'$ assign the same values to all ancestors of X in $\mathcal{N}$, and such that, given the values assigned by $\boldsymbol{o}$ and $\boldsymbol{o}'$ to the parents of X, $\boldsymbol{o}$ assigns a more preferred value to X than that assigned by $\boldsymbol{o}'$ according to the preference table of $\mathcal{N}$ for X, then $\boldsymbol{o}' \not\succ_{\mathcal{N}} \boldsymbol{o}$.*

Note the condition is not necessary. However, given a set $\mathcal{E}$ of examples and a given structure $(\mathcal{V}, E)$, we can generate propositional clauses that correspond to these conditions: if we find a set of tables that satisfies these clauses, then we are certain that the CP-net is weakly consistent with the set of examples.

Let us first define the propositional literals that will be used in the clauses. Given an acyclic graph $(\mathcal{V}, E)$ and an attribute $X \in \mathcal{V}$, let U be the set of its parents in the graph: the table for X in a CP-net over $(\mathcal{V}, E)$ contains, for every assignment u for the attributes in U, and any pair of distinct values x, x' in the domain of X, either $u : x \succ x'$ or $u : x' \succ x$. So we can define a propositional variable whose truth value says which is the case of the two possibilities. So our propositional language contains a propositional variable for every pair of possible values for every attribute $X \in \mathcal{V}$ and every assignment for the parents of X. Without explicitly giving a name to this variable, we will simply represent the literals for variable X and assignment u by $u : x \succ x'$ and $u : x' \succ x$.

Now, given a pairwise comparison of two outcomes $\boldsymbol{o} \succ \boldsymbol{o}'$, and a graph $\mathcal{G} = (\mathcal{V}, E)$, we define a clause $\Psi_{\mathcal{G}, \boldsymbol{o} \succ \boldsymbol{o}'}$ as the set of the literals $u : x \succ x'$ for every variable X such that $\boldsymbol{o}$ and $\boldsymbol{o}'$ assign the same values to all ancestors of X in $\mathcal{G}$ and the value assigned to X by $\boldsymbol{o}$ is x, that assigned by $\boldsymbol{o}'$ is x', and $x \neq x'$, and where u is the value assigned by $\boldsymbol{o}$ and $\boldsymbol{o}'$ to the parents of X. Note that if $\mathcal{G} = (\mathcal{V}, \emptyset)$, then $\Psi_{\mathcal{G}, \boldsymbol{o} \succ \boldsymbol{o}'} = \Phi_{\boldsymbol{o} \succ \boldsymbol{o}'}$. We will denote by $\Psi_{\mathcal{G}, \mathcal{E}}$ the conjunction of the clauses corresponding to all examples of $\mathcal{E}$.

For example, suppose we have four binary variables A, B, C and D, and the following graph $\mathcal{G}$:

$$\begin{tikzcd} A \arrow[r] & B \arrow[r] & C & D \arrow[l] \end{tikzcd}$$

Then $\Psi_{\mathcal{G},abcd\succ a\bar{b}\bar{c}d} = a : b \succ \bar{b}$, whereas $\Psi_{\mathcal{G},abcd\succ \bar{a}\bar{b}\bar{c}\bar{d}} = a \succ \bar{a} \vee d \succ \bar{d}$.[3]

Corollary 5. *If $\mathcal{G}$ is an acyclic graph and $\Psi_{\mathcal{G},\mathcal{E}}$ is satisfiable, then $\mathcal{E}$ is weakly $\mathcal{G}$-compatible.*

In order to obtain strong compatibility, we can use a stronger formula: we let, for every $(\boldsymbol{o},\boldsymbol{o}') \in \mathcal{E}$, $\Lambda_{\mathcal{G},\boldsymbol{o}\succ\boldsymbol{o}'} = \Psi_{\mathcal{G},\boldsymbol{o}\succ\boldsymbol{o}'} \wedge \neg\Psi_{\mathcal{G},\boldsymbol{o}'\succ\boldsymbol{o}}$. Then $\Lambda_{\mathcal{G},\mathcal{E}}$ is the conjunction of the formulas corresponding to all examples of $\mathcal{E}$.

Proposition 7 *If $\mathcal{G}$ is an acyclic graph and $\Lambda_{\mathcal{G},\mathcal{E}}$ is satisfiable, then $\mathcal{E}$ is strongly $\mathcal{G}$-compatible.*

Proof: We first define two relations between outcomes: we let $\boldsymbol{o} \geqq_{\mathcal{N}} \boldsymbol{o}'$ when the conditions of proposition 6 are met: there exists a variable $X \in \mathcal{V}$ such that $\boldsymbol{o}$ and $\boldsymbol{o}'$ assign the same values to all ancestors of X in $\mathcal{N}$, and such that, given the values assigned by $\boldsymbol{o}$ and $\boldsymbol{o}'$ to the parents of X, $\boldsymbol{o}$ assigns a more preferred value to X than that assigned by $\boldsymbol{o}'$ according to the preference table of $\mathcal{N}$ for X. Let then $\boldsymbol{o} \gg_{\mathcal{N}} \boldsymbol{o}'$ if $\boldsymbol{o} \geqq_{\mathcal{N}} \boldsymbol{o}'$ but $\boldsymbol{o}' \not\geqq_{\mathcal{N}} \boldsymbol{o}$. [7] prove that the transitive closure of $\gg_{\mathcal{N}}$ is irreflexive and contains $\succ_{\mathcal{N}}$. Now suppose that a CP-net $\mathcal{N}$ satisfies $\Lambda_{\mathcal{G},\mathcal{E}}$, then $\gg_{\mathcal{N}}$ is such that $\boldsymbol{o} \gg_{\mathcal{N}} \boldsymbol{o}'$ for every $(\boldsymbol{o},\boldsymbol{o}') \in \mathcal{E}$. Consider now a completion $\succ$ of $\gg_{\mathcal{N}}$: it satisfies $\mathcal{E}$, and it is a completion of $\succ_{\mathcal{N}}$. ■

Note that the formula $\Lambda_{\mathcal{G},\mathcal{E}}$ is very strong: in the case where the graph has no vertex, it ensures that the resulting CP-net implies the examples.

It is possible to define another formula, the unsatisfiability of which ensures that $\mathcal{E}$ is not weakly $\mathcal{G}$-compatible, but at a high cost: Proposition 1 indicates that an example $(\boldsymbol{x},\boldsymbol{y})$ is implied by a CP-net if and only if there is a swapping sequence $\boldsymbol{x} = \boldsymbol{x}_0,\ \boldsymbol{x}_1,\ldots,\ \boldsymbol{x}_n = \boldsymbol{y}$ from $\boldsymbol{x}$ to $\boldsymbol{y}$ such that $x_i \succ x_{i+1}$ for every i; so the example is weakly consistent if an only if for every sequence of swaps $\boldsymbol{y} = \boldsymbol{x}_0,\ \boldsymbol{x}_1,\ldots,\ \boldsymbol{x}_n = \boldsymbol{x}$ from from $\boldsymbol{y}$ to $\boldsymbol{x}$, $x_i \succ x_{i+1}$ holds for at least one i. However, the translation of a set of examples into a set of clauses using this characterisation does not seem to be of practical use, since there can be too many decreasing sequences of flips from one outcome to another. Even if we restrict our attention to what we may call *direct* sequences, ones in which no variable is flipped more that once, there will be be $|Diff(\boldsymbol{x},\boldsymbol{y})|!$ such sequences; and by doing so we would lose the practical usefulness of the approach.

We end up this section by briefly addressing complexity issues. We know from [13] that dominance checking in acyclic CP-nets is NP-complete. Therefore, checking whether an acyclic CP-net implies (resp. is weakly consistent with) a set of examples is

[3] Note that the clause $\Psi_{\boldsymbol{o}\succ\boldsymbol{o}'}$ is never empty: assume it is, and take any order on $\mathcal{V}$ (w.l.o.g., $X_1 \succ \ldots \succ X_n$) being compatible with $\mathcal{G}$ (which is possible because $\mathcal{G}$ is acyclic); we prove by induction on k that $\boldsymbol{o}$ and $\boldsymbol{o}'$ assign the same values to X_k. This is true for $k = 1$, because $\Psi_{\boldsymbol{o}\succ\boldsymbol{o}'}$ does not contain any literal referring to X_1, and X_1 has no parents in G. Assume it is true for all $j \leq k$. $\Psi_{\boldsymbol{o}\succ\boldsymbol{o}'}$ does not contain any literal referring to X_{k+1}, and the values of the parents of X_{k+1} (which are contained in $\{\mathcal{X}_1,\ldots,X_k\}$) are the same in $\boldsymbol{o}$ and $\boldsymbol{o}'$, therefore o and o' must assign the same values to X_{k+1}. Therefore we have $\boldsymbol{o} = \boldsymbol{o}'$, which is impossible because examples involve distinct outcomes.

NP-complete (resp. coNP-complete). However, checking whether a given set of examples is weakly or implicatively $\mathcal{G}$-compatible requires first finding the suitable CP-net; such a CP-net is exponentially large in the maximum number of parents in $\mathcal{G}$, therefore weak and implicative $\mathcal{G}$-compatibility may well be above NP and coNP, except in the specific case where the number of parents in bounded by a constant (in the latter case, implicative $\mathcal{G}$-compatibility is NP-complete and weak $\mathcal{G}$-compatibility is in Σ_2^p).

5 Conclusion

Learning a good representation of an ordering of multiattribute outcomes is a difficult task because of the exponential number of these orderings. CP-nets provide a compact, but incomplete, representation for such an ordering. Because not all orderings can be exactly captured by a CP-net, it seems reasonable to learn a CP-net consistent with all examples, rather than to look for a CP-net that would imply all of them. In the specific case of separable preference relations, checking if a set of examples can be implied by an acyclic CP-net can be achieved in polynomial time, but on the other hand, we may often fail to find such a CP-net implying the examples. Moreover, although checking if there exists a CP-net that is consistent with each example taken individually is NP-complete (still in the case of separable preference relations), our translation of the problem into clauses means that a good SAT solver may be able to find such a CP-net in reasonable time. As for ensuring global consistency, this seems much more difficult: we do not even have a proof of membership to NP.

We have assumed in the paper that a structure $\mathcal{G}$ is given, and that we want to learn a CP-net over that graph. In general, we may not know the structure. In this case, the goal of the learning problem is to learn the structure of the CP-net as well as conditional preference tables for this graph. It seems natural to try to learn a CP-net over a structure as simple as possible, as suggested in [10]. In order to achieve that, we can start with a graph with no edge, and try to learn tables for this graph. If this is not successful, we try to learn tables for CP-nets with one edge. Again, if this is not successful, we can try to find tables for CP-nets with two edges and so on... Of course, enumerating all possible structures would not be feasible, and would not be desirable either: it may be the case that only a CP-net with a very complex structure is weakly consistent with all examples; this may indicate that we are over-fitting the data, because data is noisy or because the total ordering that we should learn cannot be represented by a CP-net. In order to keep the CP-net simple, rather than aiming for a CP-net that is 100% weakly compatible with the examples, we can compute, for each CP-net, its rate of weak compatibility with the examples, and learn a CP-net that represents a good tradeoff between this rate of compatibility and simplicity.

References

1. Perny, P., Zucker, J.D.: Preference-based search and machine learning for collaborative filtering: the film-conseil movie recommender system. I3 **1**(1) (2001) 1–40
2. Miller, B.N., Albert, I., Lam, S.K., Konstan, J.A., Riedl, J.: Movielens unplugged: Experiences with an occasionally connected recommender system. In: Proceedings of ACM

2003 Conference on Intelligent User Interfaces (IUI'03), Chapel Hill, North Carolina, ACM (2003)
3. Viappiani, P., Faltings, B., Pu, P.: Evaluating preference-based search tools: a tale fo two approaches. In: Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI-06). (2006) 205–210
4. Viappiani, P., Faltings, B., Pu, P.: Preference-based search using example-critiquing with suggestions. Journal of Artificial Intelligence Research **27** (2006) 465–503
5. Chen, L., Pu, P.: Survey of preference elicitation methods. Technical Report 200467, Ecole Polytechnique Fédérale de Lausane (2004)
6. Keeney, R.L., Raiffa, H.: Decision with Multiple Objectives: Preferences and Value Tradeoffs. Wiley (1976)
7. Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H., Poole, D.: CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements. Journal of Artificial Intelligence Research **21** (2004) 135–191
8. Ha, V.A., Haddawy, P.: Problem-focused incremental elicitation of multi-attribute utility models. In: UAI. (1997) 215–222
9. Guo, Y., Müller, J., Weinhardt, C.: Learning user preferences for multi-attribute negotiation: An evolutionary approach. In: CEEMAS. (2003) 303–313
10. Athienitou, F., Dimopoulos, Y.: Learning CP-networks: a preliminary investigation. In: Proceedings of the 3rd Multidisciplinary Workshop on Advances in Preference Handling (PREF'07). (2007)
11. Brafman, R.I., Domshlak, C.: Introducing variable importance tradeoffs into CP-nets. In: Uncertainty in Artificial Intelligence: Proceedings of the Eighteenth Annual Conference. (2002) 69–76
12. Wilson, N.: Consistency and constrained optimisation for conditional preferences. In: Proceedings of the 16th European Conference on Artificial Intelligence (ECAI'04). (2004) 888–892
13. Domshlak, C., Brafman, R.: CP-nets—reasoning and consistency testing. In: Proceedings of KR0-2. (2002) 121–132