Learning Ordinal Preferences on Multiattribute Domains: the Case of CP-nets*

Yann Chevaleyre[1], Frédéric Koriche[2], Jérôme Lang[1], Jérôme Mengin[3], and Bruno Zanuttini[4]

[1] LAMSADE, Université Paris-Dauphine, CNRS, France
`lang@lamsade.dauphine.fr, yann.chevaleyre@lamsade.dauphine.fr`
[2] LIRMM, Université Montpellier II, CNRS, France
`frederic.koriche@lirmm.fr`
[3] IRIT, Université Paul Sabatier, CNRS, France
`jerome.mengin@irit.fr`
[4] GREYC, Université de Caen Basse-Normandie, CNRS, ENSICAEN, France
`bruno.zanuttini@info.unicaen.fr`

Abstract. A recurrent issue in decision making is to extract a preference structure by observing the user's behavior in different situations. In this paper, we investigate the problem of learning ordinal preference orderings over discrete multi-attribute, or combinatorial, domains. Specifically, we focus on the learnability issue of conditional preference networks, or *CP-nets*, that have recently emerged as a popular graphical language for representing ordinal preferences in a concise and intuitive manner. This paper provides results in both passive and active learning. In the passive setting, the learner aims at finding a CP-net compatible with a supplied set of examples, while in the active setting the learner searches for the cheapest interaction policy with the user for acquiring the target CP-net.

1 Introduction

Suppose we observe a user expressing her preferences about airplane tickets. Namely, she prefers an Aeroflot flight landing at Heathrow to a KLM flight landing at Gatwick, while she prefers an Aeroflot flight landing at Heathrow to a KLM flight landing at Heathrow. An intuitively correct hypothesis that explains her behavior is that she prefers Aeroflot to KLM unconditionally, and Heathrow to Gatwick, again unconditionally. Such an hypothesis allows for predicting that she will prefer an Aeroflot flight landing at Heathrow to anything else, and an Aeroflot flight landing at Gatwick to a KLM flight landing at Gatwick. Yet, this hypothesis is not able to predict whether she will prefer an Aeroflot flight landing at Gatwick or a KLM flight landing at Heathrow. Now, if we observe later that she prefers a KLM flight landing at Gatwick to a KLM flight landing at Heathrow, the current hypothesis must be updated. A new possible hypothesis,

* Partially supported by the ANR projects CANAR (ANR-06-BLAN-0383-02) and PHAC (ANR-05-BLAN-0384-01)

among others, could be that she prefers Aeroflot to KLM, and Heathrow to Gatwick when flying on Aeroflot and *vice versa* when flying on KLM.

The learning problem underlying this scenario is to extract a preference structure by observing the user's behavior in situations involving a choice among several alternatives. Each alternative can be specified by many attributes, such as the flight company, the airport location, the arrival and departure time, the number of transits, and so on. As a result, the space of possible situations has a *combinatorial* structure. Furthermore, the preferences induced by the user's behavior are intrinsically related to *conditional preferential independence*, a key notion in multiattribute decision theory [20]. Indeed, the initial hypothesis is unconditional in the sense that the preference over the values of each attribute is independent of the values of other attributes. By contrast, in the final hypothesis, the user's preference between airports is conditioned by the airline company.

Preferences over combinatorial domains have been investigated in detail by researchers in multiattribute Decision Theory (DT) and Artificial Intelligence (AI). In multiattribute DT, researchers have focused on *modeling* preferences, that is, giving axiomatic characterizations of classes of preference relations or utility functions, while researchers in AI have concentrated on the development of languages for *representing* preferences that are computationally efficient; such languages have to express preferences as succinctly as possible, and to come with fast algorithms for finding optimal alternatives.

These classes of models and languages can be partitioned by examining the mathematical nature of the preferences they consider. Namely, a distinction is made between *ordinal* preferences that consist in ranking the alternatives, and *numerical* preferences consisting of utility functions mapping each alternative to some number. Learning or eliciting numerical preferences has received a great deal of attentions in the literature [7,9,10,16,18]. A related stream of work is preference elicitation in the context of combinatorial auctions [26]; what has to be learnt is the valuation function of every buyer, which associates with every combination of goods the maximum value that she is ready to pay for it. Recently, there has been a growing interest for learning ordinal preferences using numerical models. Many standard machine learning methods, such as neural networks [8] or support vector machines [14], have been adapted to this framework, often called *learning to rank instances* by the machine learning community.

In contrast, learning preferences using *ordinal* models has received much less attention. In fact, the most studied model is the *lexicographic preference model* that provides ordering relations between examples described as pairwise comparisons between tuples of values. Dombi et al. [13] propose a learning algorithm that elicits a lexicographic preference model by guiding the user through a sequence of queries involving test examples. Although it is possible to determine in polynomial time whether there exists a lexicographical model compatible with a set of such examples, Schmitt and Martignon [27] show that the corresponding optimization problem of minimizing preference disagreement is NP-hard, and can even not be approximated in polynomial time to within a constant factor. They also give the Vapnik-Chervonenkis dimension of lexicographical preference relations: it is equal to the number of attributes. Finally, Yaman et al. [32] do not commit to a single lexicographic preference relation but approximate the target using the votes of a *collection* of consistent lexicographic preference relations. In a nutshell, learning lexicographic preference relations proves not to be so hard, but this comes with a price, namely, the induced hypothesis is highly restrictive.

In this paper, we examine a different class of ordinal preference models, where the hypotheses we make bear on the preferential dependence structure. As emphasized in the above scenario, a key point when dealing with ordinal preferences on combinatorial domains is the dependence structure between attributes. To this point, conditional preference networks, also known as *CP-nets*, are a graphical language for representing preferences based on conditional preferential independence [5]. Informally, a CP-net consists in a collection of attributes pointing to a (possibly empty) set of parents, and a set of conditional tables associated to each attribute, expressing the local preference on the values of the attribute given all possible combinations 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. CP-nets and their generalizations are probably the most popular compact representation language for ordinal preferences in multiattribute domains.

While many facets of CP-nets have been studied into detail, such as consistency, dominance checking, and optimization (constrained and unconstrained), the problem of learning CP-nets from examples have only rarely, and very recently, been addressed. One exception is [12], who proposed an algorithm that, given a set of examples, outputs a CP-net which implies them (see Section 5 for more details). Although not directly concerned with CP-nets, a related work is [25] which proposes to learn preference theories in the sense of [15].

The aim of this position paper is to examine the problem of learning CP-nets according to several dimensions that naturally emerges in preference learning. A first dimension is to consider whether the user's preferences are representable, or not, by a CP-net. If the target preference relation can be described by a CP-net, the goal is to identify this network. Alternatively, if the target preference relation is not representable by a CP-net, we can only hope finding an approximation of it. Among the candidate approximations, some of them are particularly relevant from a reasoning viewpoint. From this perspective, we shall concentrate on finding CP-nets for which the target relation is a "completion" of the hypothesized relation. Orthogonally, a second dimension in CP-net learning is to consider whether the learning process is merely *passive*, by simply observing the user's behavior in given situations, or *active*, by allowing the learner to test the user's behavior in some carefully chosen situations. In many contexts, it is relevant to identify, or at least approximate, a CP-net by mixing both active and passive learning. Consider for instance a system helping a user finding a flat from a large database, such as in [29,30]. A flat is described by attributes such as price, location, size etc. The system can start to extract a pool of preferences by observing the user's behavior, and then use queries in order to converge toward the ideal hypothesis while minimizing the number of interactions.

In the different learning models that arise from these dimensions, we will investigate the learnability issues in terms of the worst-case number of resources required to converge toward the desired CP-net, where resources refer both to the running time, the sample complexity in passive learning, and the query complexity in active learning. Section 2 provides the necessary background about CP-nets. In Section 3, we extend the paradigm of concept learning to preference learning and introduce two frameworks, one for the problem of learning partial orderings that are representable by a CP-net, and the other for the problem of learning linear orderings that are not representable by a CP-net. Our learnability results lie in the next three sections. Namely, section 4 focuses on the VC-dimension and the approximate fingerprint property for classes of CP-nets. Section 5 addresses passive learning of CP-nets. Section 6 considers active learning of CP-nets. Finally, section 7 briefly discusses issues for further work.

Given that this is a position paper, the proofs of the results are only sketched, or even omitted. They can be found in [11,21,23].

2 Conditional Preference Networks

Throughout this paper, we shall assume a finite list $V = \langle X_1, \dots, X_n \rangle$ of *attributes*, with their associated finite *domains* $D = \langle D_1, \dots, D_n \rangle$. An attribute X_i is *binary* if D_i has two elements, which by convention we note x_i and $\overline{x}_i$. By $\mathcal{V} = \times_{X_i \in V} D_i$, we denote the set of all complete assignments, called *outcomes.*

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

A *preference relation* is a reflexive and transitive binary relation $\succeq$ over $\mathcal{V}$. A *complete preference relation* is a preference relation $\succeq$ that is connected, that is, for every $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ we have either $\boldsymbol{x} \succeq \boldsymbol{y}$ or $\boldsymbol{y} \succeq \boldsymbol{x}$. A *strict preference relation* $\succ$ is an irreflexive and transitive (thus asymmetric) binary relation over $\mathcal{V}$. A *linear preference relation* is a strict preference relation that is connected. 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 $(\boldsymbol{y} \succeq \boldsymbol{x})$.

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 [5] are a graphical language for representing such preferences. Informally, a CP-net is composed of a directed graph representing the preferential dependencies between attributes, and a set of conditional preference tables expressing, for each attribute, the local preference on the values of its domain given all possible combinations of values of its parents.

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 $\boldsymbol{y}$.

Definition 1. *Suppose that V is partitioned into the subsets X, Y and Z. Let $\succ$ be a linear preference relation over $\mathcal{V}$. Then, we say that X is* preferentially independent *of Y given Z (w.r.t. $\succ$) if for all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$, $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathcal{Y}$, $\boldsymbol{z} \in \mathcal{Z}$,*

$$\boldsymbol{x}_1\boldsymbol{y}_1\boldsymbol{z} \succ \boldsymbol{x}_2\boldsymbol{y}_1\boldsymbol{z} \textit{ if and only if } \boldsymbol{x}_1\boldsymbol{y}_2\boldsymbol{z} \succ \boldsymbol{x}_2\boldsymbol{y}_2\boldsymbol{z}$$

Example 1. Consider three binary attributes X_1, X_2 and X_3 and suppose that the four swaps on X_2 are ordered as follows:

$$\begin{aligned} x_1x_2x_3 &\succ x_1\overline{x}_2x_3 \\ \overline{x}_1\overline{x}_2x_3 &\succ \overline{x}_1x_2x_3 \\ x_1x_2\overline{x}_3 &\succ x_1\overline{x}_2x_3 \\ \overline{x}_1\overline{x}_2\overline{x}_3 &\succ \overline{x}_1x_2\overline{x}_3 \end{aligned}$$

We can see that, irrespective of the value of X_3, if x_1 is the case, then x_2 is preferred to $\overline{x}_2$, whereas if $\overline{x}_1$ is the case, then $\overline{x}_2$ is preferred to x_2. This ordering on the X_2-swaps with two *conditional preferences*: $x_1 : x_2 \succ \overline{x}_2$ and $\overline{x}_1 : \overline{x}_2 > x_2$. We remark that X_2, given X_1, is preferentially independent of X_3.

Definition 2. *A* CP-net *N over $V = \{X_1, \cdots, X_n\}$ consists in a directed graph over V, and a set of preference tables* $\mathrm{CPT}(X_i)$ *associated to each $X_i \in V$. For attribute X_i, we denote by* $\mathrm{Pa}(X_i)$ *the set of parents of X_i in the graph of N, and by* $\mathrm{NonPa}(X_i)$ *the set $V \setminus (\{X_i\} \cup \mathrm{Pa}(X_i))$.*

Each conditional preference table $\mathrm{CPT}(X_i)$ *is a list of rows, also called* entries *or* rules, *of the form $\boldsymbol{u} : x_{i_1} \succ \cdots \succ x_{i_m}$, where $\boldsymbol{u}$ is an instantiation of* $\mathrm{Pa}(X_i)$ *and $x_{i_1} \succ \cdots \succ x_{i_m}$ is a linear ordering of the domain D_i (with $m = |D_i|$). It indicates that $\boldsymbol{uz}x_{i_j} \succ \boldsymbol{uz}x_{i_{j+1}}$ for every possible instantiation $\boldsymbol{z}$ of* $\mathrm{NonPa}(X_i)$.

Example 2. A CP-net over the binary attributes X_1, X_2 and X_3 is:

$$\begin{tikzcd} X_1 \arrow[r] & X_2 \arrow[r] & X_3 \\ x_1 \succ \overline{x}_1 & \begin{array}{c} x_1 : x_2 \succ \overline{x}_2 \\ \overline{x}_1 : \overline{x}_2 \succ x_2 \end{array} & \begin{array}{c} x_2 : x_3 \succ \overline{x}_3 \\ \overline{x}_2 : \overline{x}_3 \succ x_3 \end{array} \end{tikzcd}$$

where $X \longrightarrow Y$ means "X is a parent of Y". The associated ordering of the swaps is:

$$\begin{tikzcd} & & & \overline{x}_1\overline{x}_2\overline{x}_3 \arrow[dr] \arrow[drrr] & & & \\ x_1x_2x_3 \arrow[r] \arrow[rrrrr, bend left] \arrow[drrr, bend right] & x_1x_2\overline{x}_3 \arrow[r] \arrow[rrrrr, bend right] & x_1\overline{x}_2\overline{x}_3 \arrow[ur] \arrow[dr] & & \overline{x}_1\overline{x}_2x_3 \arrow[r] & \overline{x}_1x_2x_3 \arrow[r] & \overline{x}_1x_2\overline{x}_3 \\ & & & x_1\overline{x}_2x_3 \arrow[ur] & & & \end{tikzcd}$$

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

The *size* of a CP-net N, denoted by $|N|$, is the number of entries in the preference tables of N:

$$|N| = \sum_{X_i \in V} \prod_{X_j \in \mathrm{Pa}(X_i)} |\mathcal{X}_j|$$

which in the case of binary CP-nets boils down to $|N| = \sum_{X_i \in V} 2^{|\mathrm{Pa}(X_i)|}$.

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 N, we write $\succ_N$ for this transitive closure. Note that this relation $\succ_N$ may not be total, and it may not be a irreflexive since it may contain cycles. Yet, we know from [5] that if the graph of N is acyclic, then $\succ_N$ is a strict preference relation (i.e. contains no cycles), and we say that N is *consistent*. Otherwise, we say that N is inconsistent. Note that even if N is consistent, $\succ_N$ may still not be connected; it can then be completed in a number of linear preference relations. If $\succ$ is one of them, we say that $\succ$ is a *completion* of N, or that it is *compatible* with N.

It is important to keep in mind that several CP-nets can induce the same preference relation: for example, if a CP-net N over the binary attributes X_1 and X_2 contains the table $x_2 : x_1 \succ \overline{x}_1$ and $\overline{x}_2 : x_1 \succ \overline{x}_1$, then the CP-net N' in which X_1 has no parent and the table $x_1 \succ \overline{x}_1$ is equivalent to N. In general, for every consistent CP-net N there is a unique CP-net N' equivalent to N that is minimal in the number of parents/table entries for each variable.

It is easy to verify that every linear preference relation is compatible with exactly one (minimal) CP-net. For instance, $x_1x_2 \succ \overline{x}_1\overline{x}_2 \succ x_1\overline{x}_2 \succ \overline{x}_1x_2$ is compatible with the CP-net

$$\begin{tikzcd} X_1 \arrow[r, leftrightarrow] & X_2 \end{tikzcd}$$

$$\begin{array}{ll} x_2 : x_1 \succ \overline{x}_1 & x_1 : x_2 \succ \overline{x}_2 \\ \overline{x}_2 : \overline{x}_1 \succ x_1 & \overline{x}_1 : \overline{x}_2 \succ x_2 \end{array}$$

The following property will be frequently used in the remaining sections.

Proposition 1. *([5]) Let N be an acyclic CP-net and $\boldsymbol{x}$, $\boldsymbol{y}$ two outcomes. Then $\boldsymbol{x} \succ_N \boldsymbol{y}$ iff 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 $0 \leq i < k$, $\boldsymbol{x}^i \succ_N \boldsymbol{x}^{i+1}$, 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_{j_i}, then N contains $\boldsymbol{u} : x^i_{j_i} \succ x^{i+1}_{j_i}$.*

Though a CP-net is usually defined as above, most of the results presented here extend to *possibly incomplete CP-nets.* Such a CP-net is one in which each conditional preference table $CPT(X_i)$ contains *at most* one conditional preference rule per instantiation of $\mathrm{Pa}(X_i)$ (instead of *exactly one*). A particular case is when some of the tables are empty. The semantics of an incomplete CP-net is still given by the transitive closure of the dominance relation on swaps induced by the rules. An important difference is that in an incomplete CP-net, not all swaps are comparable.

The rationale for considering incomplete CP-nets can be understood with the following example. A user may well know that she prefers traveling by bus rather than in the subway in Paris, *vice-versa* in London, but be unable to state her preference for a city in which she has never been, say Madrid. This does not mean that she would not have a preference, rather that she does not know it (so far). In this case, a variable encoding the transportation means (with values *subway* and *bus*) would have the variable encoding the city as its parent, with values *Paris*, *London*, and *Madrid*, but would contain only two rules.

3 Learning CP-nets: learning what?

The problem of concept learning is to extrapolate from a collection of examples, each labeled as either positive or negative by some unknown target concept, a representation of this concept that accurately labels future, unlabeled examples. Most concept learning algorithms operate over some *concept class*, which captures the set of concepts that the learner can potentially generate over all possible sets of training examples.

In the setting suggested by our framework, a *concept* is a strict preference relation $\succ$ over $\mathcal{V}$. We say that a concept $\succ$ is *representable* by a CP-net N if the induced ordering $\succ_N$ coincides with $\succ$, that is, $\succ = \succ_N$. For example, the preference relation $\succ$ defined by $\{ab \succ a\overline{b}, ab \succ \overline{a}b, \overline{a}\overline{b} \succ a\overline{b}, \overline{a}\overline{b} \succ \overline{a}b\}$ is representable by the CP-net N specified by $\{a : b \succ \overline{b},\ \overline{a} : \overline{b} \succ b,\ b : a \succ \overline{a},\ \overline{b} : \overline{a} \succ a\}$. A *representation class* is a collection $\mathcal{N}$ of consistent CP-nets, and the *concept class* $\mathcal{C}_{\mathcal{N}}$ defined over $\mathcal{N}$ is the set of all preference relations $\succ$ that are representable by a CP-net N in $\mathcal{N}$.

Since we consider consistent CP-nets only, any target concept $\succ$ in a class $\mathcal{C}_{\mathcal{N}}$ can be represented by a unique minimal CP-net N. So, with a slight abuse of language, we shall simply say that $\mathcal{C}_{\mathcal{N}}$ is the class of all CP-nets in $\mathcal{N}$. For instance, the concept class $\mathcal{C}_{\text{ACY}}$ is the class of all acyclic CP-nets; and $\mathcal{C}_{\text{TREE}}$ is the class of tree-structured CP-nets.

With these notions in hand, we assume that the user has in mind a target preference ordering $\succ$, and the learner has at its disposal a predetermined and known class of CP-nets $\mathcal{C}_{\mathcal{N}}$. In this study, we shall consider two different types of target concepts.

(A) The target concept is a preference relation $\succ$ that belongs to the learner's concept class $\mathcal{C}_{\mathcal{N}}$. In other words, there is a CP-net N in $\mathcal{N}$, such that $\succ_N$ coincides with $\succ$. In this context, the goal of the learner is to find N.

(B) The target concept is a preference relation $\succ$ that does *not necessarily* belong to the learner's concept class $\mathcal{C}_{\mathcal{N}}$. For example, we can easily observe that the linear ordering $\succ$ defined by $\{ab \succ \overline{a}\overline{b} \succ a\overline{b} \succ \overline{a}b\}$ cannot be represented by any CP-net. Still, we shall make the assumption that $\succ$ is a completion of some representation in $\mathcal{N}$. Specifically, we say that $\succ$ is a *completion* of a CP-net N if $\boldsymbol{x} \succ_N \boldsymbol{y}$ implies $\boldsymbol{x} \succ \boldsymbol{y}$ for any pair of outcomes $(\boldsymbol{x}, \boldsymbol{y})$. For example, the above ordering $\succ$ is a completion of $N = \{a : b \succ \overline{b},\ \overline{a} : \overline{b} \succ$

b, $b : a \succ \overline{a}$, $\overline{b} : \overline{a} \succ a\}$. In this "agnostic" setting, the goal of the learner is to find a CP-net N of which $\succ$ is a completion.

Note that the distinction between (A) and (B) is not on the set of objects we want to learn, but on their interpretation, which has crucial consequences on how examples are interpreted.

3.1 Learning a preference relation induced by a CP-net

Let us start with context (A) where the target concept is representable by a CP-net N of some given representation class $\mathcal{N}$. Here, an instance, or *example*, is a pair $(\boldsymbol{x}, \boldsymbol{y})$ of outcomes, and an *instance class* is a set $\mathcal{E}$ of examples. Given a target concept $\succ_N$, and an example $(\boldsymbol{x}, \boldsymbol{y})$, we say that $(\boldsymbol{x}, \boldsymbol{y})$ is *positive* for $\succ_N$ if $\boldsymbol{x} \succ_N \boldsymbol{y}$, that is, $\boldsymbol{x}$ dominates $\boldsymbol{y}$ according to $\succ_N$. Dually, $(\boldsymbol{x}, \boldsymbol{y})$ is *negative* for $\succ_N$ if $\boldsymbol{x} \not\succ_N \boldsymbol{y}$. It is important to keep in mind that, in general, if the pair $(\boldsymbol{x}, \boldsymbol{y})$ is a negative example of $\succ_N$ then the reverse pair $(\boldsymbol{y}, \boldsymbol{x})$ is *not* necessarily a positive example of $\succ_N$.

In this context, our learning problem can be seen as a standard concept learning problem: given a set $\mathcal{T}$ of positive and negative training examples, we want to find a CP-net that "implies" all positive examples and no negative example.

Definition 3. *Let N be a CP-net over $\mathcal{V}$. An example $(\boldsymbol{x}, \boldsymbol{y})$ is* entailed by N *if $\boldsymbol{x} \succ_N \boldsymbol{y}$. A set of examples $\mathcal{T}$ is* implicatively consistent with N*, or* implied *by N, if*

- *all positive examples in $\mathcal{T}$ are entailed by N;*
- *no negative example in $\mathcal{T}$ is entailed by N.*

Finally, we shall say that a training set $\mathcal{T}$ is *implicatively compatible* if it is implied by at least one CP-net.

3.2 Learning a CP-net which approximates the user's preferences

Now, we turn to context (B) and make the assumption that the user's preference relation $\succ$ is a linear order, in general not exactly representable by any CP-net. In this framework, we start by examining an appropriate notion of consistency between a CP-net and a training set. Consider the following example:

Example 3. We have two binary attributes X_1 and X_2 (with domains $\{x_1, \overline{x}_1\}$ and $\{x_2, \overline{x}_2\}$), and the set of positive examples

$$\mathcal{T} = \{(x_1x_2\, ,\, x_1\overline{x}_2),\ (x_1\overline{x}_2\, ,\, \overline{x}_1x_2),\ (\overline{x}_1x_2\, ,\, \overline{x}_1\overline{x}_2)\}$$

What do we expect to learn from the above set of examples $\mathcal{T}$? The transitive closure of $\mathcal{T}$ is the complete preference relation $x_1x_2 \succ x_1\overline{x}_2 \succ \overline{x}_1x_2 \succ \overline{x}_1\overline{x}_2$. This preference relation is *separable* (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 $\overline{x}_1x_2$ 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"). Intuitively, since $\mathcal{T}$ is separable, we expect to output a structure N that contains $x_1 \succ \overline{x}_1$ and $x_2 \succ \overline{x}_2$. However, no CP-net implies $\mathcal{T}$, whatever the dependencies. The structure N induces a *partial* preference relation in which $x_1\overline{x}_2$ and $\overline{x}_1x_2$ are incomparable. More generally, no *ceteris paribus* structure can "explain" that $x_1 \succ \overline{x}_1$ is "more important" than $x_2 \succ \overline{x}_2$ (i.e., with no intermediate alternative). Therefore, if we look for a structure *implying* all the examples, we will simply output "failure". On the other hand, if we look for a separable CP structure that is simply *contingent* with the examples, i.e., that does not imply the contrary of the examples, we will output N.

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 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 $x_2 \succ \overline{x}_2$; the fact that $x_1\overline{x}_2$ and $\overline{x}_1x_2$ are incomparable in 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 $x_1\overline{x}_2 \succ \overline{x}_1x_2$[5].

Therefore, in such cases we do not look for a CP-net which implies the examples. Rather, we look for one whose preference relation is consistent with the examples. A first way of understanding consistency is to require that the learnt CP-net N be such that the examples are consistent with at least one preference relation extending N. Yet, there are cases where it may even be too strong to require that one of the completions of $\succ_N$ contains all the examples, in particular 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:

Example 4. Suppose that we learn that all users in a 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 $\overline{x}_1x_2$ may differ (think as x_1 and x_2 as, respectively, "being invited to a fine dinner" and "receiving a \$50 award"): then $\mathcal{T} \supseteq \{(\overline{x}_1x_2\,,\, x_1\overline{x}_2),\ (x_1\overline{x}_2\,,\, \overline{x}_1x_2)\}$. $\mathcal{T}$ is clearly inconsistent, so there cannot be any preference structure whose ordering can be completed into a linear preference relation that contains $\mathcal{T}$. However, if $\mathcal{N} = \{x_1 \succ \overline{x}_1, x_2 \succ \overline{x}_2\}$, then each example in $\mathcal{T}$ is (individually) contained in at least one completion of $\succ_N$.

Such considerations lead us to define two new notions of compatibility of a CP-net with a set of examples. Note that because the target concept is a linear order, $(\boldsymbol{x}, \boldsymbol{y})$ is a negative example if and only if $(\boldsymbol{y}, \boldsymbol{x})$ is a positive one. For this reason, we can make the assumption that all examples in the learner's training set $\mathcal{T}$ are *positive*, with the implicit knowledge that the reverse $(\boldsymbol{y}, \boldsymbol{x})$ of any pair $(\boldsymbol{x}, \boldsymbol{y})$ in $\mathcal{T}$ is negative.

[5] If we want to do this, we have to resort to a more expressive language such as TCP-nets [6] or conditional preference theories [31].

Definition 4. *Let N be a CP-net over $\mathcal{V}$. An example $(\boldsymbol{x}, \boldsymbol{y})$ is* consistent by completion with N *if there is a completion $\succ$ of $\succ_N$ such that $\boldsymbol{x} \succ \boldsymbol{y}$. Furthermore, we will say that a set of examples $\mathcal{T}$ is:*

– strongly consistent by completion with N *if there is a completion $\succ$ of $\succ_N$ such that for all $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{T}$, $\boldsymbol{x} \succ \boldsymbol{y}$;*
– weakly consistent by completion with N *if every example $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{T}$ is individually consistent by completion with N.*

Lastly, we will say that $\mathcal{T}$ is *strongly / weakly compatible* if it is strongly / weakly consistent by completion with at least one CP-net. Clearly, strong compatibility implies weak compatibility. Moreover, since an example $(\boldsymbol{x}, \boldsymbol{y})$ is consistent by completion with a CP-net N if and only if $(\boldsymbol{y}, \boldsymbol{x})$ is not implied by N, implicative compatibility implies strong compatibility.

As it stands, it turns out to be significantly more difficult to search for a CP-net strongly or weakly consistent with a set of examples than to search for a CP-net implicatively consistent with it. Therefore, we mainly focus on implicative compatibility; strong and weak compatibility will only be discussed in the context of *separable* CP-nets, that is, CP-nets where all variables are independent.

4 Learnability of CP-nets

In this section we investigate the theoretical limits concerning the learnability issue of CP-nets. In essence, the Vapnik-Chervonenkis dimension of a class gives upper bounds on the difficulty to learn, in terms of numbers of examples, while the approximate fingerprint property gives lower bounds.

4.1 Vapnik-Chervonenkis dimension

The *Vapnik-Chervonenkis (VC) dimension* of a class of concepts is a fundamental complexity measure used in theoretical machine learning. Intuitively, the VC-dimension of $\mathcal{C}$ is the maximum number of informative examples which can be received by the learner, where an “informative” example is an observation that helps the learner reducing the number of consistent hypotheses.

Formally, let $\mathcal{C}$ be a concept class defined over some representation class $\mathcal{R}$. A set of instances $\mathcal{T}$ is said to be *shattered* by $\mathcal{C}$ if, whatever the partition of $\mathcal{T}$ into $\mathcal{T}^+ \cup \mathcal{T}^-$, there is a concept $N \in \mathcal{C}$ which admits all instances in $\mathcal{T}^+$ as positive examples and all instances in $\mathcal{T}^-$ as negative examples. The *Vapnik-Chervonenkis dimension of* $\mathcal{C}$, denoted $VC(\mathcal{C})$, is the maximum size of a set of examples $\mathcal{T}$ which is shattered by $\mathcal{C}$. The intuition is that for such a set $\mathcal{T}$, as long as the learner ignores the label of at least one example, at least two concepts in $\mathcal{C}$ are consistent with the labels it has seen so far.

When the learner has access only to a certain kind of example (e.g., swap examples), it makes sense to adapt the notion of VC-dimension. So if $\mathcal{E}$ is a class of instances, we write $VC_{\mathcal{E}}(\mathcal{C})$ for the VC-dimension of $\mathcal{C}$ *with respect to* $\mathcal{E}$, that is, the maximum size of a $\mathcal{T} \subseteq \mathcal{E}$ which is shattered by $\mathcal{C}$. Clearly, if $\mathcal{E} \subseteq \mathcal{E}'$, then $VC_{\mathcal{E}}(\mathcal{C}) \leq VC_{\mathcal{E}'}(\mathcal{C})$.

Observe that, as there are 2^m partitions of a set of m examples into positive and negative examples, each of which must be captured by a different concept, $VC(\mathcal{C}) \leq \log_2 |\mathcal{C}|$ always holds (whatever the class of examples).

We now give the VC-dimension of the class of all CP-nets which have a fixed graph. The intuition here is that since the parents of each variable are known, the quantity of information needed to characterize a CP-net is exactly 1 per possible rule in the CP-net, namely, one pair of outcomes which dictates the conclusion of the rule.

Call *subgraph* of G a graph with the same vertices as G but whose set of edges is included in that of G.

Proposition 2. *Let G be a graph, and let $\mathcal{C}_G$ be the class of all concepts which are representable by a binary complete CP-net whose graph is G or a subgraph of G. Then the VC-dimension of $\mathcal{C}_G$ with respect to swap examples is exactly the number of conditional preference rules in any such CP-net. If CP-nets are possibly incomplete, then the VC-dimension (w.r.t. swap examples) is still the number of rules in any complete CP-net on (a subgraph of) G.*

This property can help us finding an upper bound of the VC-dimension of acyclic CP-nets. Intuitively, the number of acyclic graphs with indegree at most k is lower bounded by $|\mathcal{C}_G|$ for a convenient graph with k roots and $n-k$ vertices with indegree exactly k, and upper bounded by $(n-1)^{n(k+1)}$ (for each vertex, choose at most k parents). Since any binary-valued CP-net built over an acyclic graph of degree at most k allows at most $n2^k$ entries, there are at most $(n-1)^{n(k+1)}2^{n2^k}$ binary-valued acyclic CP-nets with degree at most k. We therefore obtain the following result.

Corollary 1. *Let $k \in o(n)$, and let $\mathcal{C}^k_{\text{ACY}}$ be the class of all concepts which are representable by a possibly incomplete binary CP-net whose graph is acyclic and with indegree at most k. Then, the VC-dimension of $\mathcal{C}^k_{\text{ACY}}$ with respect to swap or arbitrary examples is in $\tilde{\Theta}(n2^k)$.*

Finally, without any restriction over the degree, it can be shown that the VC-dimension grows as $\Theta(2^n)$, which is still much below the VC-dimension of the class of all possible CP-nets.

Corollary 2. *Let $\mathcal{C}_{\text{ACY}}$ be the class of all concepts which are representable by a possibly incomplete binary CP-net whose graph is acyclic. Then, the VC-dimension of $\mathcal{C}_{\text{ACY}}$ with respect to swap examples is in $\Theta(2^n)$.*

4.2 Approximate fingerprints

Approximate fingerprints are a powerful tool for obtaining non learnability results in active learning. Intuitively, a class of concepts $\mathcal{C}$ has the approximate fingerprint property if there is a subset $\mathcal{C}^*$ of $\mathcal{C}$ such that for any concept $N \in \mathcal{C}$, there is an example with which N is consistent, but with which only a superpolynomially small fraction of the concepts in $\mathcal{C}^*$ are also consistent.

This property can be used to show that in an interactive learning setting, a hypothesis $\widehat{N} \in \mathcal{C}$ may fail on an example which only gives clues about superpolynomially few candidate hypotheses. Hence, if the learner only gets information from such failures, in the worst case it *necessarily* makes an exponential number of errors before correctly identifying the target concept. We refer the reader to [2] for formal details.

As for the VC-dimension, the definition of approximate fingerprints can be restricted to instance classes $\mathcal{E}$. Observe that if $\mathcal{E} \subseteq \mathcal{E}'$ and $\mathcal{C}$ has the approximate fingerprint property with respect to $\mathcal{E}$, then it also has this property with respect to $\mathcal{E}'$.

Proposition 3 ([21]). *Let $\mathcal{C}_{\text{ACY}}$ be the class of all concepts which are representable by a binary complete CP-net whose graph is acyclic. Then $\mathcal{C}_{\text{ACY}}$ has the approximate fingerprint property with respect to swap examples.*

Proposition 4 ([21]). *Let $\mathcal{C}_{\text{TREE}}$ be the class of all concepts which are representable by a binary complete CP-net whose graph is a tree. Then $\mathcal{C}_{\text{TREE}}$ has the approximate fingerprint property with respect to arbitrary examples.*

5 Passive learning of CP-Nets

In this section, we investigate *passive* learning of CP-nets. In this setting, the only information about the target concept available to the learner is a set of examples. We shall concentrate here on the widely studied *Probably Approximately Correct* (PAC) learning model introduced by [28]. The intent of this model is to obtain with high probability a representation that is a good approximation of the target concept. To formalize the notion of a good approximation, we need to assume that there is some fixed, but unknown, probability distribution $\mathcal{D}$ defined on the example space $\mathcal{E}$, from which the available examples were drawn. In our case, $\mathcal{D}$ would define a probability over each instance $(\boldsymbol{x}, \boldsymbol{y})$. Given a target concept $\succ$, we then define the *error* of an hypothesized CP-net $\widehat{N}$ as the probability that $\succ$ and $\succ_{\widehat{N}}$ disagree on an example:

$$\mathit{error}(\widehat{N}) = \mathbf{Pr}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}} \left[\left(\boldsymbol{x} \succ \boldsymbol{y} \text{ and } \boldsymbol{x} \not\succ_{\widehat{N}} \boldsymbol{y}\right) \text{ or } \left(\boldsymbol{x} \not\succ \boldsymbol{y} \text{ and } \boldsymbol{x} \succ_{\widehat{N}} \boldsymbol{y}\right) \right]$$

How does one generate a *good* approximation? In the PAC model, one does this by looking at an example set, in which each example $(\boldsymbol{x}, \boldsymbol{y})$ has been drawn independently at random from the distribution $\mathcal{D}$, and labeled with "+" (positive) if $\boldsymbol{x} \succ \boldsymbol{y}$ and with "−" (negative) if $\boldsymbol{x} \not\succ \boldsymbol{y}$.

Thus, in the PAC setting, training and testing use the same distribution, and there is no noise in either phase. A learning algorithm is then a computational procedure that takes a sample of the target concept $\succ$, consisting of a sequence of independent random examples of $\succ$, and returns a hypothesis. We can define PAC learnability of CP-nets as follows.

Definition 5 (PAC learning). *A concept class $\mathcal{C}_{\mathcal{N}}$ is* PAC learnable *by an example class $\mathcal{E}$ if there is a polynomial time learning algorithm A and a polynomial $p(\cdot,\cdot,\cdot)$ such that for any target concept $\succ$ in $\mathcal{C}_{\mathcal{N}}$ over n variables, any probability distribution $\mathcal{D}$ over $\mathcal{E}$, and any parameters $\delta, \epsilon \in (0,1)$, if the algorithm A is given at least $p(n, \frac{1}{\epsilon}, \frac{1}{\delta})$ independent random examples of $\succ$ drawn according to $\mathcal{D}$, then with probability at least $1-\delta$, A returns a hypothesis $\widehat{N} \in \mathcal{N}$ with $error(\widehat{N}) \leq \epsilon$. The smallest such polynomial p is called the* sample complexity *of the learning algorithm A.*

The intent of this definition is that the learning algorithm must process the examples in polynomial time, and must be able to produce a good approximation of the target concept with high probability using only a reasonable number of training examples.

It is important to emphasize that our learnability results are defined over specific instance classes. In particular, if $\mathcal{E}$ is the class of all swap instances, then any distribution $\mathcal{D}$ over $\mathcal{E}$ will assign a zero probability to any "non-swap" instance. This restriction has deep consequences on the predictive power of CP-nets. Namely, even if a positive learnability result with swap instances guarantees that the hypothesized CP-net $\widehat{N}$ is expected to correctly classify "swap" instances drawn independently at random according to the distribution $\mathcal{D}$, such a result does *not* ensure that the learner will correctly classify arbitrary outcome pairs. Indeed, even if the probability of making a mistake on swaps is low, the probability of making a mistake on an arbitrary instance $(\boldsymbol{x}, \boldsymbol{y})$ may increase along an improving sequence from $\boldsymbol{y}$ to $\boldsymbol{x}$.

Many positive learnability results in the PAC model are obtained by showing that (1) there is an efficient algorithm capable of finding a hypothesized representation that is *consistent by implication* with a given sample of the target concept (called a *consistent algorithm*), and (2) the sample complexity of any such algorithm is polynomial.

The sample complexity of a consistent learning algorithm is usually measured using the VC-dimension of the concept class $\mathcal{C}_{\mathcal{N}}$. Indeed, it is shown in [4] that the sample complexity of a consistent learning algorithm is at most

$$\frac{1}{\epsilon(1-\sqrt{\epsilon})}\left(2\mathrm{VC}(\mathcal{C}_{\mathcal{N}})\ln\frac{6}{\epsilon} + \ln\frac{2}{\delta}\right) \tag{1}$$

A preliminary work on passive learning of CP-nets is [12]. They give an algorithm which, given a set of positive examples, outputs a CP-net that implies them, under some conditions. It is not entirely clear yet which class of CP-nets is learned by this algorithm.

5.1 PAC Learning of Acyclic CP-Nets

We first investigate PAC learnability of various classes of acyclic CP-nets when the examples provided to the learner are swaps. Recall that for such examples, the dominance test with acyclic CP-nets is linear-time solvable (simple lookup in the conditional preference table), contrary to the general case.

We first show that even for the restricted class of concepts which are representable by a CP-net whose graph is a chain, the consistency problem is NP-complete. It follows directly (unless P=NP) that this class is not PAC-learnable with the very weak restriction that the produced hypothesis classifies correctly the examples received.

Proposition 5. *Deciding whether there exists a binary complete CP-net whose graph is a chain and which implies a given set of swaps is NP-complete. The result holds even if all examples are positive.*

A proof based on a reduction from the Hamiltonian path problem can be found in [11]. Another interesting class of CP-nets is that of acyclic *singly-connected* CP-nets [5], that is, those acyclic CP-nets in whose graph each pair of vertices is connected by at most one directed path. Unfortunately, again we have a negative result for PAC learnability of such CP-nets.

Proposition 6. *Deciding whether there exists a binary complete CP-net whose graph is acyclic singly-connected and which implies a given set of swaps is NP-complete. The result holds even if all examples are positive.*

This result [11] can be proven with a reduction from propositional satisfiability (SAT). We conjecture that a similar negative result holds for more general classes of acyclic CP-nets.

We now turn to positive results with swaps. Observe that if $\mathcal{C}_{\mathcal{N}'}$ is PAC learnable with swaps and $\mathcal{C}_{\mathcal{N}} \subseteq \mathcal{C}_{\mathcal{N}'}$, then $\mathcal{C}_{\mathcal{N}}$ is not necessarily PAC learnable with swaps as well. So our positive results do not contradict the negative ones. The main result is that tree CP-nets are PAC-learnable from swaps.

Proposition 7. *The class of all concepts which are representable by a (possibly incomplete) tree binary CP-net is PAC-learnable from swap examples.*

In addition, learning the structure of such a CP-net can be reduced to finding a spanning tree in a directed graph [11]. Since in general there may be several CP-nets which imply a given set of examples, it is interesting to impose some restrictions, e.g., on the degree of the forest (maximum number of children of a node). The next result states that the class of CP-nets whose graph is a forest with degree at most k is *improperly* PAC-learnable (in quasi-polynomial time) [11]. That is, it is "PAC-learnable", but the hypothesis may be in a larger representation class than the target concept.

Proposition 8. *There is a quasi-polynomial time algorithm which, given a set of swaps $\mathcal{T}$ over n variables implied by a (possibly incomplete) binary-valued CP-net whose graph is a forest of degree k, computes a binary-valued CP-net which implies $\mathcal{T}$ and whose graph is a forest of degree at most $k + \log n$.*

Finally, we give a more general result about tree CP-nets with a bounded number of tables on *arbitrary* examples. By Cayley's formula, we know that there are k^{k-1} rooted trees with k vertices. Each root is labeled by an unconditional rule of the form $p \succ \overline{p}$, and all other nodes are labeled by conditional rules of the form $p' : p \succ \overline{p}$, where p and p' are literals. There are $2n$ tables with no condition per rule and $6n(n-1)$ (possibly incomplete) tables with one condition per rule. It follows that the number C_k of CP-trees with at most k tables is bounded by $\sum_{i=0}^{k} 2n(6kn(n-1)+1)^{k-1}$, which is indeed polynomial in k. So the VC-dimension of such CP-trees is polynomial in n.

Based on this result, we can use a simple consistent algorithm specified as follows. Start with the hypothesis set $\mathcal{N}$ of all CP-trees with at most k tables. For each example $(\boldsymbol{x}, \boldsymbol{y})$ in $\mathcal{T}$, remove any hypothesis N in $\mathcal{N}$ that is inconsistent with $(\boldsymbol{x}, \boldsymbol{y})$, that is, any hypothesis N for which the dominance test over $(\boldsymbol{x}, \boldsymbol{y})$ disagrees with its label. If the resulting set $\mathcal{N}$ is empty then $\mathcal{T}$ is not consistent with N. Otherwise, pick an arbitrary tree from $\mathcal{N}$. Because the dominance test is quadratic in the number of variables for binary-valued CP-trees, the running time is polynomial in C_k.

Proposition 9. *The class $\mathcal{C}^k_{\text{TREE}}$ of all concepts representable by a (possibly incomplete) binary-valued CP-tree with at most k tables is PAC learnable from arbitrary examples.*

5.2 PAC Learning of Separable CP-nets

We now consider the task of learning a CP-net of the simplest form: the variables are independent of each other. With binary variables, this means that if the possible values for variable X are x and $\overline{x}$, and the target CP-net is complete, then the preference table for X contains either $x \succ \overline{x}$ or $\overline{x} \succ x$.

In this case, checking if a given CP-net N entails $\boldsymbol{x} \succ_N \boldsymbol{y}$ for an arbitrary example is easy. Let $\text{Diff}(\boldsymbol{x}, \boldsymbol{y}) = \{x_i \mid (\boldsymbol{x})_i = x_i \text{ and } (\boldsymbol{y})_i = \overline{x_i}\} \cup \{\overline{x_i} \mid (\boldsymbol{x})_i = \overline{x_i} \text{ and } (\boldsymbol{y})_i = x_i\}$. Then $\boldsymbol{x} \succ_N \boldsymbol{y}$ if and only if N contains $x_i \succ \overline{x}_i$ for every $x_i \in \text{Diff}(\boldsymbol{x}, \boldsymbol{y})$ and $\overline{x}_i \succ x_i$ for every $\overline{x}_i \in \text{Diff}(\boldsymbol{x}, \boldsymbol{y})$ (this is a corollary of Theorems 7 and 8 by [5]).

Now, with each example $(\boldsymbol{x}, \boldsymbol{y})$ we associate the clause $C^-_{\boldsymbol{x},\boldsymbol{y}}$ that contains $\neg x_i$ iff $x_i \in \text{Diff}(\boldsymbol{x}, \boldsymbol{y})$ and x_i iff $\overline{x}_i \in \text{Diff}(\boldsymbol{x}, \boldsymbol{y})$. The intended meaning of the literal $\neg x_i$ is that $\overline{x}_i$ is preferred to x_i, whereas the meaning of the literal x_i is that x_i is preferred to $\overline{x}_i$; hence the meaning of the clause $C^-_{\boldsymbol{x},\boldsymbol{y}}$ is that $\boldsymbol{x} \not\succ_N \boldsymbol{y}$ for every separable CP-net N in which at least one of these local preferences is true, by virtue of the lemma above. For instance, if $\boldsymbol{x}{=}\overline{x}_1 x_2 x_3 \overline{x}_4$ and $\boldsymbol{y}{=}x_1 \overline{x}_2 x_3 x_4$ then $\text{Diff}(\boldsymbol{x}, \boldsymbol{y}) = \{\overline{x}_1, x_2, \overline{x}_4\}$ and $C^-_{\boldsymbol{x},\boldsymbol{y}} = x_1 \vee \neg x_2 \vee x_4$. This clause expresses that x_1 is preferred to $\overline{x}_1$, or $\overline{x}_2$ is preferred to x_2, or x_4 is preferred to $\overline{x}_4$.

With each positive example $(\boldsymbol{x}, \boldsymbol{y})$ we can also associate the cube (conjunction of literals) $C^+_{\boldsymbol{x},\boldsymbol{y}} \equiv \neg C^-_{\boldsymbol{x},\boldsymbol{y}}$. Given a set of training examples $\mathcal{T}$, let $\Gamma_{\mathcal{T}} = \bigwedge\{C^*_e \mid e \in \mathcal{T}\}$, where $C^*_{\boldsymbol{x},\boldsymbol{y}} = C^+_{\boldsymbol{x},\boldsymbol{y}}$ if $(\boldsymbol{x}, \boldsymbol{y})$ is a positive example, and $C^*_{\boldsymbol{x},\boldsymbol{y}} = C^-_{\boldsymbol{x},\boldsymbol{y}}$ if the example is negative. Clearly, $\Gamma_{\mathcal{T}}$ is equivalent to a set of clauses.

Now consider the following one-to-one correspondence between truth assignments M over $\{x_1, \ldots, x_n\}$ and separable CP-nets N_M over $\mathcal{V}$: N_M contains the preference $x_i \succ \overline{x}_i$ for every i such that $M \models x_i$ and the preference $\overline{x}_i \succ x_i$ for every i such that $M \models \neg x_i$. For instance, if $M(x_1) = M(x_4) = \top$ and

$M(x_2) = M(x_3) = \bot$, N_M contains the preference tables $\{x_1 \succ \overline{x}_1,\ \overline{x}_2 \succ x_2,\ \overline{x}_3 \succ x_3,\ x_4 \succ \overline{x}_4\}$. Then for an interpretation M, it is easily seen that $M \models C^-_{\boldsymbol{x},\boldsymbol{y}}$ if and only if N_M does not imply $(\boldsymbol{x},\boldsymbol{y})$ or, equivalently, that $M \models C^+_{\boldsymbol{x},\boldsymbol{y}}$ if and only if N_M implies $(\boldsymbol{x},\boldsymbol{y})$.

It follows that a set of examples $\mathcal{T}$ is implicatively consistent with N_M for a given model M if and only if $M \models \Gamma_{\mathcal{T}}$. Therefore, searching for a CP-net which implies a given set of examples amounts to searching for a model of the corresponding set of clauses, the size of which grows polynomially with the size of the set of examples.

This technique easily extends to nonbinary variables: we can use a propositional variable x_i^{kl} for every pair of distinct values $\{x_i^k, x_i^l\}$ for every variable X_i, where the intended meaning of x_i^{kl} is $x_i^k \succ_N x_i^l$, and add clauses to represent the transitivity of the relation $\succ_N$; there is a polynomial number of them (details can be found in [22]).

The one-to-one correspondence given above is a reduction from our learning problem to satisfiability. It is actually possible to find a reduction in the opposite direction (see [22]), from which we get the following result.

Proposition 10. *[22] Deciding whether there is a (binary or non-binary) complete separable CP-net which implies a given set of arbitrary examples is NP-complete. The result holds even if all examples are negative.*

It follows directly that this class is not PAC-learnable with the very weak restriction that the produced hypothesis classifies correctly the examples received. However:

Proposition 11. *[22] Deciding whether there is a binary-valued complete separable CP-net which implies a given set of positive examples can be done in polynomial time.*

Observe that this result can be extended to PAC-learnability of (possibly incomplete) separable CP-nets from positive examples, in the setting of one-sided errors [28]. This is because there is always a unique minimal (in terms of rules) incomplete separable CP-net which implies a given set of positive examples.

Now, as soon as $\mathcal{T}$ becomes large with respect to the number of attributes n, the chances that $\mathcal{T}$ is implicatively consistent with a separable CP-net become low. In this case, we may want to determine a separable CP-net that is implicatively consistent with as many examples of $\mathcal{T}$ as possible. This problem amounts to solving a MAXSAT problem, when each example corresponds to exactly one clause of $\Gamma_{\mathcal{T}}$, that is when we have no positive example: the separable CP-net that best fits a set of positive examples corresponds to the interpretation maximizing the number of clauses from $\Gamma_{\mathcal{T}}$ satisfied. In this case, we can reuse algorithms for MAXSAT for computing a separable CP-net that best fits a set of positive examples, as well as polynomial approximation schemes. This extends to nonbinary variables, with the difference that the clauses representing transitivity of the local preference tables are protected.

Lastly, again using the same kind of translation, we easily get the following results.

Proposition 12. *If all variables are binary and all examples in $\mathcal{T}$ differ at most on two variables, then deciding whether there exists a separable CP-net which implies $\mathcal{T}$ can be done in polynomial time; however, the corresponding optimization problem remains* NP*-hard.*

5.3 Learning a complete preference relation

We close this section by providing results about the learning context (B) specified in section 3: the target concept is a linear order, not necessarily representable by a CP-net. Our goal is to find a CP-net that would be a good representation for this relation. Recall that since the target is a linear order, we only need to consider *positive* examples.

In the rest of this section, we investigate in turn the problems of finding a CP-net that is weakly consistent with a given set of examples, then strongly consistent with it. We focus on the problem of finding *separable* CP-nets. A set of examples is said to be *weakly separable* (resp. *strongly separable*) if there exists a separable CP-net with which it is weakly (resp. strongly) consistent.

We start by showing how the search for a separable CP-net that is weakly consistent with a set of examples can be rewritten as an instance of propositional satisfiability (SAT). Recall from Section 5.2 that an example $(\boldsymbol{x}, \boldsymbol{y})$ can be translated into a clause $C^{-}_{\boldsymbol{y},\boldsymbol{x}}$, the models of which correspond to separable CP-nets that are consistent with $(\boldsymbol{x}, \boldsymbol{y})$. Given a set of examples $\mathcal{T}$, let $\Phi_{\mathcal{T}} = \{C^{-}_{\boldsymbol{y},\boldsymbol{x}} \mid (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{T}\}$. Then, given an interpretation M, a set of examples $\mathcal{T}$ is weakly consistent with N_M if and only if $M \models \Phi_{\mathcal{T}}$. As a consequence, $\mathcal{T}$ is weakly separable if and only if $\Phi_{\mathcal{T}}$ is satisfiable.

Example 5. Consider three binary attributes A, B, C, and the set of examples

$$\mathcal{T} = \{(abc\,,\, \overline{a}bc),\ (\overline{a}\overline{b}c\,,\, ab\overline{c}),\ (\overline{a}b\overline{c}\,,\, a\overline{b}c),\ (\overline{a}bc\,,\, \overline{a}\overline{b}\overline{c})\}$$

$\Phi_{\mathcal{T}}$ has a unique model, corresponding to the separable CP-net $N = \{a \succ \overline{a}, b \succ \overline{b}, c \succ \overline{c}\}$. Therefore, N is the unique separable CP-net weakly consistent with N, and $\mathcal{T}$ is weakly separable.

As in section 5.2, a similar translation can be used with non-binary variables, and algorithms for solving MAXSAT can be used to search for a CP-net that is weakly consistent with as many examples as possible.

Proposition 13. *Deciding whether a set of examples over (binary or non-binary) attributes is weakly separable is* NP*-complete.*

Now, let us turn to the notion of strong compatibility. Characterizing such a property is less easy. Indeed, the difference between weak and strong compatibility is that while in weak compatibility we look for a separable CP-net which is consistent with each individual example in $\mathcal{T}$, in strong compatibility we look for a separable CP-net which is consistent with the whole set of examples $\mathcal{T}$.

Example 5, continued *$\mathcal{T}$ is not strongly consistent with N, because $\mathcal{T} \cup \succ_N$*

has the following cycle:

$$a\bar{b}c \succ_N \bar{a}\bar{b}c \succ_{\mathcal{T}} ab\bar{c} \succ_N \bar{a}b\bar{c} \succ_{\mathcal{T}} a\bar{b}c$$

Since $\mathcal{T}$ is not strongly consistent with any separable CP-net other than N (because N is the unique one with which $\mathcal{T}$ is weakly compatible), $\mathcal{T}$ is not strongly separable[6]*.*

Note that all outcomes in the cycle on the example above appear in $\mathcal{T}$. More generally, if we denote by $\mathcal{O}(\mathcal{T})$ the set of outcomes that appear in $\mathcal{T}$, it can be proved that $\mathcal{T}$ is strongly consistent with N if and only if the restriction of $\succ_N \cup \mathcal{T}$ to $\mathcal{O}(\mathcal{T})$ is acyclic. Since this restriction has at most $2\,|\mathcal{T}|$ vertices, checking if it possesses a cycle can be done in polynomial time. Thus, checking whether $\mathcal{T}$ is strongly consistent with a given CP-net N is in P, and we have the following result:

Proposition 14. *[23] Checking whether $\mathcal{T}$ is strongly separable is* NP*-complete.*

Note that although weak and strong separability have the same complexity, weak separability enjoys the nice property that there is a simple solution-preserving translation into SAT (the models of $\Phi_{\mathcal{T}}$ correspond bijectively to the CP-nets that are weakly consistent with $\mathcal{T}$), which allows weak separability to be computed in practice using algorithms for SAT[7]. Contrastingly, in order to compute a separable CP-net strongly consistent with $\mathcal{T}$, we can generate structures N weakly consistent with $\mathcal{T}$, and test for acyclicity of $\succ_N \cup \mathcal{T}$ using graph algorithms.

6 Active learning of CP-nets

In this section, we investigate the learnability issues of CP-nets in the paradigm of active learning. Recall that in the standard PAC learning model, examples are drawn at random according to an unknown but fixed distribution. This model of learning is merely passive in the sense that the learner has no control over the selection of examples. One can increase the flexibility of this model by allowing the learner to ask about particular examples, that is, the learner makes *membership queries* [1]. This capability appears to increase the power of polynomial-time learning algorithms. For instance, it is known that propositional Horn formulas are PAC-learnable with membership queries [3], but the results of [19] show that without membership queries, Horn formulas are no easier to learn than general CNF or DNF formulas.

In the setting of active preference learning, we assume that the user has in mind a target preference structure $\succ$, but does not know how to represent this

[6] Note that $\mathcal{T}$ is both weakly separable and does not contain any cycles as it was the case for Example 4, yet is not strongly separable.

[7] Such a translation exists for strong separability (which we do not give here), but unfortunately, the set of clauses generated uses $\mathcal{O}(n^2)$ variables (where n is the set of examples), which limits its practical applicability.

structure into a CP-net. However, the user is disposed to help the learner by answering membership queries of the form "does $\boldsymbol{x}$ dominates $\boldsymbol{y}$?", where $\boldsymbol{x}$ and $\boldsymbol{y}$ are outcomes chosen by the learner. A membership query for a target concept $\succ$ is a map MQ that takes as input a pair of outcomes $(\boldsymbol{x}, \boldsymbol{y})$ and returns as output `yes` if $\boldsymbol{x} \succ \boldsymbol{y}$, and `no` if $\boldsymbol{x} \not\succ \boldsymbol{y}$.

From a practical perspective, one must take into account the fact that outcomes are typically *not* comparable with an equivalent cost. As observed in [17], users can meaningfully compare outcomes if they differ only on very few attributes. To this end, we define the *width* of $\mathrm{MQ}(\boldsymbol{x}, \boldsymbol{y})$ to be the number of variables on which $\boldsymbol{x}$ and $\boldsymbol{y}$ differ. A membership query of width 1 is called a *swap membership query.*

Based on these considerations, a minimal requirement behind active learning is to ask as few membership queries as possible. An additional desiderata for minimizing the cognitive effort spent by the user in answering preference queries is to restrict to swap membership queries.

Definition 6 (PAC learning with membership queries). *A concept class $\mathcal{C}_{\mathcal{N}}$ is* PAC learnable with swap membership queries *over an instance class $\mathcal{E}$ if there is a polynomial time learning algorithm A and two polynomials $p(\cdot,\cdot,\cdot)$ and $q(\cdot)$ such that for any target concept $\succ$ in $\mathcal{C}_{\mathcal{N}}$, any probability distribution $\mathcal{D}$ over $\mathcal{I}$, and any parameters $\delta, \epsilon \in (0,1)$, after receiving $p(n, \frac{1}{\epsilon}, \frac{1}{\delta})$ random examples of $\succ$ drawn independently according to $\mathcal{D}$, and asking $q(n)$ swap membership queries, then with probability at least $1-\delta$, A returns a hypothesis $\widehat{N} \in \mathcal{N}$ with error$(\widehat{N}) \leq \epsilon$. The smallest such polynomial q is called the* query complexity *of the learning algorithm A.*

6.1 Active Learning with swap examples

We now investigate the problem of active preference learning, where the target concept can be represented by an *acyclic* CP-net, and the questions are restricted to swap membership queries.

In this setting, we can build an online algorithm for learning actively acyclic CP-nets. Recall that online learning proceeds into trials. Initially the learner chooses an hypothesis $\hat{N}$. During each trial, the learner first receives an example $(\boldsymbol{x}, \boldsymbol{y})$, next predicts the label "+" or "−" of this instance according to its current hypothesis, and then receives the correct label from the user. If the prediction was incorrect then the learner is charged one mistake.

The basic idea underlying our online learning algorithm is to start from the empty CP-net $\hat{N} = \varnothing$ and, during each trial, iteratively revise $\hat{N}$ by maintaining two invariants. The first invariant is that each rule (or entry) in the learner's hypothesis $\hat{N}$ is subsumed by a rule in the minimal representation of the target CP-net N. In other words, for each rule $\hat{\boldsymbol{u}} : x \succ \overline{x}$ in $\hat{N}$, there is a rule $\boldsymbol{u} : x \succ \overline{x}$ in the minimal representation of N, with $\hat{\boldsymbol{u}} \subseteq \boldsymbol{u}$. The second invariant is that each such rule $r = \hat{\boldsymbol{u}} : x \succ \overline{x}$ in $\hat{N}$ is *supported* by an instance $(\boldsymbol{x}_r, \boldsymbol{y}_r)$ of N, that is, $\boldsymbol{x}_r \succ_N \boldsymbol{y}_r$, $\boldsymbol{x}_r$ and $\boldsymbol{y}_r$ satisfy $\hat{\boldsymbol{u}}$, $\boldsymbol{x}_r$ satisfies x, and $\boldsymbol{y}_r$ satisfies $\overline{x}$.

Technically, the algorithm proceeds as follows. On seeing an example $(\boldsymbol{x}, \boldsymbol{y})$, if the learner predicts this instance as negative, while it is positive, then it expands its CP-net with a new rule $\boldsymbol{u} : x \succ \overline{x}$, where the support $(\boldsymbol{x}, \boldsymbol{y})$ is stored. Here, x is the literal in $\boldsymbol{x}$ whose value differs from $\boldsymbol{y}$, and $\boldsymbol{u}$ is the projection of the outcome $\boldsymbol{x}$ onto the current parent set of the variable X in $\hat{N}$. Dually, if the learner predicts $(\boldsymbol{x}, \boldsymbol{y})$ as positive, while it is negative, then it expands the condition $\boldsymbol{u}$ of the misclassifying rule $\boldsymbol{u} : x \succ \overline{x}$ with a new parent. Using the support $(\boldsymbol{x}_r, \boldsymbol{y}_r)$ of this rule, a new parent can be found by asking at most $n-1$ membership queries. To this end, we simply need to incrementally transform $\boldsymbol{x}$ into $\boldsymbol{x}_r$ by iteratively flipping those literals that differ from $\boldsymbol{x}$ and $\boldsymbol{x}_r$, until we find the first literal x_j for which the label of $(\boldsymbol{x}_j[x], \boldsymbol{x}_j[\overline{x}])$ is positive; this literal is kept as a new parent of X. In fact, only $\log_2 n$ membership queries are needed to find this parent, by performing a binary search over this sequence of transformations. A detailed implementation of this algorithm is given in [21].

Example 6. Assume so far the learner has only learnt the rule $x_1 : x_3 > \overline{x}_3$, supported by $x_1x_2x_3x_4x_5 \succ_N x_1x_2\overline{x}_3x_4x_5$.

Now assume it receives the positive example $(\boldsymbol{x}, \boldsymbol{y})$ with $\boldsymbol{x} = \overline{x}_1x_2\overline{x}_3x_4x_5$, $\boldsymbol{y} = \overline{x}_1x_2x_3x_4x_5$. Then it learns the rule $\overline{x}_1 : \overline{x}_3 > x_3$, and stores $(\boldsymbol{x}, \boldsymbol{y})$ as its support.

Finally, assume it receives the positive example $x_1\overline{x}_2\overline{x}_3\overline{x}_4\overline{x}_5 \succ_N x_1\overline{x}_2x_3\overline{x}_4\overline{x}_5$, which contradicts its first rule. Then it searches for a new parent of x_3. Using the support of this rule and the new example, it asks the membership query $x_1x_2\overline{x}_3\overline{x}_4\overline{x}_5 \succ_N x_1x_2x_3\overline{x}_4\overline{x}_5$. Assuming the answer is `yes`, it goes on with the membership query $x_1x_2\overline{x}_3x_4\overline{x}_5 \succ_N x_1x_2x_3x_4\overline{x}_5$. Assuming this one answers `no`, the learner deduces that x_4 is a parent of x_3, hence it updates the previously learnt rules according to their support, resulting in rules $x_1x_4 : x_3 > \overline{x}_3$ and $\overline{x}_1x_4 : \overline{x}_3 > x_3$, and creates a new rule $x_1\overline{x}_4 : \overline{x}_3 > x_3$ so as to cover the new example.

By using a well-known conversion from online learning to PAC-learning [24], we derive the following result.

Proposition 15. *Acyclic (possibly incomplete) binary CP-nets are PAC-learnable with swap membership queries, over the instance class of swaps. There is an online learning algorithm A for this class, such that for any target concept $\succ$ of description size s, the algorithm makes at most s mistakes and uses at most $s \log_2 n$ membership queries.*

6.2 Active Learning with Unrestricted Examples

When the instances supplied to the learner are unrestricted, even predicting their label is a difficult task, because dominance testing is NP-hard for acyclic CP-nets. As observed in the previous section, an important class of concepts for which dominance testing can be accomplished in polynomial time is the class of *tree CP-nets.* In this section, we briefly discuss an online algorithm for learning tree CP-nets.

The algorithm can be specified as follows. Initially, the learner starts from the hypothesis $\hat{N} = \varnothing$ and iteratively expands $\hat{N}$ until it finds the target representation N. An invariant of the algorithm is that $\hat{N}$ is always included in N so the learner can only make mistakes on positive examples $(\boldsymbol{x}, \boldsymbol{y})$. In such cases, the algorithm considers in turn each variable on which $\boldsymbol{x}$ and $\boldsymbol{y}$ differ, and builds its CP-table and that of all its ascendants in the tree. It stops whenever such a variable already has a CP-table in its current hypothesis. Again, to find a parent for each candidate variable, the algorithm can use a binary search strategy.

Proposition 16. *Tree (possibly incomplete) binary CP-nets are PAC-learnable with swap membership queries, over arbitrary examples: there is an online learning algorithm A for this class, such that for any target concept $\succ$ with k nodes, the algorithm makes at most k mistakes and uses $O(k \log_2 n)$ membership queries.*

We conclude this section by emphasizing that some classes of CP-nets are *teachable*, that is, learnable by asking a polynomial number of membership queries, without the need of observing any sample. Thus after making those queries, the learner is guaranteed to correctly predict any instance.

We focus on the class of acyclic CP-nets whose graph has indegree at most k. The learner proceeds by using a levelwise generate-and-test procedure and membership queries for uncovering the set of parents of each variable. Clearly, this approach is acceptable only for small bounded degrees, such as tree CP-nets.

Proposition 17. *The class of concepts representable by a binary-valued acyclic CP-net whose graph has degree at most k is teachable: there is an algorithm which outputs such a CP-net using $O(kn^{k+1}2^{k-1})$ membership queries and no other queries or examples, where n is the number of variables and k is the degree of the target concept.*

7 Conclusion and open problems

In this paper we addressed many issues related to learning CP-nets. We argued that a first important problem is whether the CP-net that we aim at learning is such that the user's preference relation coincides with its induced preference relation, or is an approximation by below of the user's complete preference relation. Then we gave a few theoretical results on the learnability of CP-nets, and considered two different learning frameworks: passive learning (from a set of examples), and active learning (by queries).

We hope that our preliminary results in the learnability issue of CP-net open the door to new theoretical results and practical learning algorithms. First of all, we do not have a general method (other than brute-force search) for computing a CP-net which is weakly or strongly consistent with a set of examples in the nonseparable case, nor do we have algorithms for outputting a CP-net realizing an optimal trade-off between simplicity and accuracy.

We already emphasized the lack of expressivity of CP-nets. Although CP-nets are a representation language well-suited to expressing preferential (in)dependencies, they do not allow, for instance, to express statements of relative importance between variables, as lexicographic orders do. We may desire to learn preferences that combine both aspects (preferential dependencies and relative importance). For this, it is necessary to study preference learning with more expressive languages such as TCP-nets [6] or (even more general) conditional preference theories [31].

References

1. D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.
2. D. Angluin. Negative results for equivalence queries. *Machine Learning*, 5:121–150, 1990.
3. D. Angluin, M. Frazier, and L. Pitt. Learning conjunctions of Horn clauses. *Machine Learning*, 9:147–164, 1992.
4. M. Anthony and N. Biggs. *Computational Learning Theory.* Cambridge Univ. Press, 1992.
5. C. Boutilier, R. Brafman, C. Domshlak, H. Hoos, and D. Poole. CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.
6. R. Brafman, C. Domshlak, and S. Shimony. On graphical modeling of preference and importance. *Journal of Artificial Intelligence Research*, 25:389–424, 2006.
7. D. Braziunas and C. Boutilier. Local utility elicitation in GAI models. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 42–49, 2005.
8. C.J.C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, 2005.
9. U. Chajewska, L. Getoor, J. Norman, and Y. Shahar. Utility elicitation as a classification problem. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 79–88, 1998.
10. U. Chajewska, D. Koller, and R. Parr. Making rational decisions using adaptive utility elicitation. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI)*, pages 363–369, 2000.
11. Y. Chevaleyre. A short note on passive learning of CP-nets. Rapport de recherche, Lamsade, mars 2009.
12. Y. Dimopoulos, L. Michael, and F. Athienitou. Ceteris paribus preference elicitation with predictive guarantees. In *Proc. 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, 2009.
13. J. Dombi, C. Imreh, and N. Vincze. Learning lexicographic orders. *European Journal of Operational Research*, 183:748–756, 2007.
14. C. Domshlak and T. Joachims. Efficient and non-parametric reasoning over user preferences. *User Modeling and User-Adapted Interaction (UMUAI)*, 17(1-2):41–69, 2007.
15. J. Doyle, Y. Shoham, and M. Wellman. A logic of relative desire (preliminary report). In *Proceedings of the 6th International Symposium on Methodologies for Intelligent Systems (ISMIS)*, pages 16–31. Springer, 1991.
16. Ch. Gonzales and P. Perny. GAI networks for utility elicitation. In *Principles of Knowledge Representation and Reasoning: Proceedings of the 9th International Conference (KR)*, pages 224–234, 2004.

17. P. Green and V. Srinivasan. Conjoint analysis in consumer research: Issues and outlook. *Journal of Consumer Research*, 5(2):103–123, 1978.
18. V. Ha and P. Haddawy. Problem-focused incremental elicitation of multi-attribute utility models. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 215–222, 1997.
19. M. J. Kearns, M. Li, L. Pitt, and L. G. Valiant. On the learnability of boolean formulae. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 285–295, 1987.
20. R. Keeney and H. Raiffa. *Decision with Multiple Objectives: Preferences and Value Trade-offs.* Wiley, 1976.
21. F. Koriche and B. Zanuttini. Learning conditional preference networks with queries. In *Proc. 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, 2009.
22. J. Lang and J. Mengin. The complexity of learning *ceteris paribus* separable preferences. In *Proc. 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, 2009.
23. J. Lang and J. Mengin. The complexity of learning separable ceteris paribus preferences. Rapport de recherche RR-2009-3-FR, IRIT, Université Paul Sabatier, Toulouse, mars 2009.
24. N. Littlestone. From on-line to batch learning. In *Proceedings of the Second Annual Workshop on Computational Learning Theory*, pages 269–284. Morgan Kaufmann, 1989.
25. M. Sachdev. On learning of ceteris paribus preference theories. Master's thesis, Graduate Faculty of North Carolina State University, 2007.
26. T. Sandholm and C. Boutilier. *Preference Elicitation in Combinatorial Auctions*, chapter 10. In *Combinatorial Auctions*, Cramton, Shoham, and Steinberg Ed., MIT Press, 2006.
27. M. Schmitt and L. Martignon. On the complexity of learning lexicographic strategies. *Journal of Machine Learning Research*, 7:55–83, 2006.
28. L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
29. P. Viappiani, B. Faltings, and P. Pu. Evaluating preference-based search tools: a tale fo two approaches. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, pages 205–210, 2006.
30. P. Viappiani, B. Faltings, and P. Pu. Preference-based search using example-critiquing with suggestions. *Journal of Artificial Intelligence Research*, 27:465–503, 2006.
31. N. Wilson. Extending CP-nets with stronger conditional preference statements. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI)*, pages 735–741, 2004.
32. F. Yaman, Th. Walsh, M. Littman, and M. desJardins. Democratic approximation of lexicographic preference models. In *Proceedings of the 35th International Conference in Machine Learning (ICML),*, pages 1200–1207, 2008.