Learning Conditional Preference Networks

Frédéric Koriche, Bruno Zanuttini

Learning Conditional Preference Networks

Frédéric Koriche[a], Bruno Zanuttini[b]

[a] *LIRMM, CNRS UMR 5506, Université Montpellier II, France*
[b] *GREYC, CNRS UMR 6072, Université de Caen Basse-Normandie, ENSICAEN, France.*
Corresponding author: Frédéric Koriche, LIRMM, 161, rue Ada, 34095 Montpellier Cedex 5, France.
Phone: +33 (0)4.67.41.85.85. Fax: +33 (0)4.67.41.85.00

Abstract

Conditional preference networks (*CP-nets*) have recently emerged as a popular language capable of representing ordinal preference relations in a compact and structured manner. In this paper, we investigate the problem of learning CP-nets in the well-known model of exact identification with equivalence and membership queries. The goal is to identify a target preference ordering with a binary-valued CP-net by interacting with the user through a small number of queries. Each example supplied by the user or the learner is a preference statement on a pair of outcomes.

In this model, we show that acyclic CP-nets are not learnable with equivalence queries alone, even if the examples are restricted to *swaps* for which dominance testing takes linear time. By contrast, acyclic CP-nets are what is called *attribute-efficiently* learnable when both equivalence queries and membership queries are available: we indeed provide a learning algorithm whose query complexity is linear in the description size of the target concept, but only logarithmic in the total number of attributes. Interestingly, similar properties are derived for tree-structured CP-nets in the presence of arbitrary examples. Our learning algorithms are shown to be *quasi-optimal* by deriving lower bounds on the VC-dimension of CP-nets. In a nutshell, our results reveal that *active* queries are required for efficiently learning CP-nets in large multi-attribute domains.

Key words: preference elicitation, conditional preference network (CP-net), query-directed learning, attribute-efficient learning

1. Introduction

The spectrum of AI applications that resort on the ability to reason about preferences is extremely wide, ranging from configuration software and recommender systems to autonomous agents and group decision-making. Since many, if not most, of these applications are defined over large, multi-attribute domains, a key challenge in preference research is to develop representation languages and elicitation techniques that cope with the exponential size of the outcome space.

Among the different languages that have been devised in the literature for representing and reasoning about preferences, conditional preference networks (*CP-nets*) have received a great deal of attention by providing a compact and natural representation of ordinal preferences in

Email addresses: Frederic.Koriche@lirmm.fr (Frédéric Koriche), Bruno.Zanuttini@info.unicaen.fr (Bruno Zanuttini)

multi-attribute domains [8–12, 17–19, 22]. Briefly, a CP-net is a graph in which each node is labeled with a table describing the user's preference over alternative values of this node given different values of the parent nodes. For example, the entry $J_b \wedge P_b : S_r \succ S_b$ might state that, all other things being equal, I prefer a red shirt to a black one if the color for both the jacket and the pants is black. The semantics of a CP-net is defined by a dominance ordering on the outcome space, derived from such reading of entries in the tables. Based on this relation, a key reasoning task is *dominance testing*: given a CP-net N and a pair of outcomes (o, o'), determine whether o dominates o', according to the dominance ordering induced by N.

Ideally, in preference elicitation with CP-nets, the learner should simply "fill the tables" by asking the user how her preference over the values of one node depends on the values of its parents. Yet, in practice, eliciting preferences is far from being easy because the dependency graph is generally not known in advance: the learner must therefore seek the interdependencies between attributes and identify a minimal set of parents for each target node. The problem is exacerbated still further by the fact that real-world applications typically involve many irrelevant attributes. For instance, it is not uncommon in recommender systems to describe products using thousands of variables, with the expectation that only a small fraction of these are crucial for specifying preferences [6, 34]. The learner is thus required to select, within a large collection of attributes, a relatively small subset over which the network will be specified.

Such considerations bring into sharp focus the need for *query learning algorithms* that aim at extracting CP-nets under the guidance of the user through an appropriate sequence of queries. A widely adopted framework for studying this issue is the model of exact learning with equivalence and membership queries, introduced by Angluin [1]. Briefly, a *membership query* allows the learner to ask the classification of any example, while an *equivalence query* allows the learner to ask whether its hypothesized concept is the correct one; in case of mistake, the learner is given a counterexample, that is, an instance for which the hypothesis and the target concept give different classifications. The utility of this model lies in the fact that rich concept classes, including Horn theories [3], decision trees [13], some description logics [21], and several fragments of first-order logic [4, 25], have been shown to be learnable with both equivalence queries and membership queries, while in weaker versions one can prove superpolynomial lower bounds.

In the setting of preference elicitation, the target concept is a dominance ordering on the outcome space. Each example is a preference statement on some pair of outcomes. For a membership query, the learner supplies an example (o, o') to the user and is told whether o dominates o' (is a member of the target preference relation), or not. For an equivalence query, the learner presents a CP-net N, and either is told that N correctly identifies the target concept, or it is given a counterexample (o, o'). The goal is to identify the target concept using as few resources as possible, where resources refer both to the runtime and the number of queries.

In essence, equivalence queries capture a form of *passive* elicitation which can be simulated in several ways. For instance, in a batch or *offline* scenario, each equivalence query can be simulated by cycling through a given set of examples supplied by the user until we find a counterexample to our current CP-net. If no counterexamples are found, then the elicitation process terminates. In an *online* scenario, each equivalence query can be simulated by observing the user behavior: a prediction mistake occurs if she makes a choice among several outcomes that contradicts our current CP-net. Importantly, these simulations can never require more cycles, or make more prediction mistakes, than the worst-case number of equivalence queries in the learning algorithm. By contrast, membership queries capture a form of *active* elicitation by allowing us to ask about examples of our own choice. In both scenarios, membership queries can be used to revise the current CP-net in light of the observed counterexample.

From a practical perspective, one must take into account the fact that outcomes are typically *not* comparable with an equivalent cost. Indeed, as observed by Green and Srinivasan [23], users can meaningfully compare outcomes if they differ only on very few attributes. Similarly, for the learner, this task can be arduous because dominance testing is generally NP-hard, even for acyclic CP-nets. Based on these considerations, our learnability results are defined in terms of a *concept class* in which the target concept is chosen, and an *instance class* that circumscribes the set of examples used by equivalence and membership queries.

The key message to be gleaned from this paper is that *active* learning is required for correct and efficient extraction of preference networks in binary-valued domains. On the one hand, acyclic CP-nets are *not* learnable with equivalence queries alone, while on the other, they are learnable with equivalence and membership queries, provided that the instance class is restricted to simple outcome pairs for which dominance testing takes linear time, namely, pairs of outcomes which differ over only one attribute. Interestingly, a similar property holds for tree-structured CP-nets by extending the instance class to arbitrary examples. When membership queries are available, we provide *attribute-efficient* learning algorithms for which the query complexity is *linear* in the size of the minimal CP-net that identifies the target concept, and *logarithmic* in the total number of attributes. By establishing lower bounds on the Vapnik-Chervonenkis (VC) dimension of acyclic CP-nets and tree-structured CP-nets, our query learning algorithm are shown to be optimal up to a logarithmic number of queries. In a nutshell, such encouraging results pave the way for fast elicitation techniques capable of extracting "small" CP-nets in "large" domains.

This paper is organized as follows. After introducing the necessary background in CP-nets in Section 2 and query-directed learning in Section 3, we examine the learnability issue of acyclic CP-nets and tree CP-nets in Section 4 and 5, respectively. The quasi-optimality of our query learning algorithms is shown in Section 6, where we derive lower bounds on the VC-dimension of preference networks. We conclude in Section 7 by discussing related work and mentioning some further results and open problems.

2. Conditional Preference Networks

Throughout this study, we shall concentrate on preference learning problems where the user's domain is described by a set of Boolean variables $X_n = \{x_1, \cdots, x_n\}$, with $n > 0$.

As usual, we refer to x_i and $\overline{x}_i$ as *literals*. Given a literal p, we denote by $\overline{p}$ the opposite of p. For example, if p_i is $\overline{x}_i$, then $\overline{p}_i$ is x_i. A *term* t is a conjunction of literals. By $var(t)$ we denote the set of variables occurring in t. We say that a term t is *maximal* for a subset of variables $Y \subseteq X_n$ if $var(t) = Y$. For example, the term $x_1\overline{x}_2$ is maximal for $\{x_1, x_2\}$, but not for $\{x_1, x_2, x_3\}$.

2.1. Syntax of CP-nets

A conditional preference rule (*CP-rule*) on a variable x is an expression of the form $t : p \succ \overline{p}$, where p is a literal of x and t is a term such that $x \notin var(t)$. Such a rule captures the statement "given that t holds, the value p is preferred to the value $\overline{p}$ for the variable x, all other things being equal". Borrowing the usual terminology of production rule systems, the preference $p \succ \overline{p}$ and the term t are respectively called the *head* and the *body* of the CP-rule $t : p \succ \overline{p}$.

A conditional preference table (*CP-table*) on a variable x with respect to a set $Y \subseteq X_n \setminus \{x\}$ is a set of CP-rules on x that associates *at most* one rule $t : p \succ \overline{p}$ to each maximal term t for Y. The CP-table is *complete* if *exactly* one rule $t : p \succ \overline{p}$ is associated to each maximal term t for Y. For example, the set $\{x_1 : x_2 \succ \overline{x}_2\}$ is an incomplete CP-table on x_2 with respect to $\{x_1\}$, while the extended set $\{x_1 : x_2 \succ \overline{x}_2,\ \overline{x}_1 : \overline{x}_2 \succ x_2\}$ is a complete CP-table on x_2 with respect to $\{x_1\}$.

Variable	Literals
Coat	trench (x_1) and parka ($\overline{x}_1$)
Hat	large (x_2) and small ($\overline{x}_2$)
Jacket	black (x_3) and white ($\overline{x}_3$)
Pants	black (x_4) and white ($\overline{x}_4$)
Shirt	red (x_5) and white ($\overline{x}_5$)
Shoes	heels (x_6) and flats ($\overline{x}_6$)

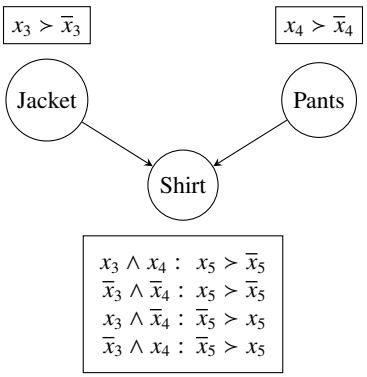

Figure 1: The domain and CP-net of "evening dress"

A conditional preference network (*CP-net*) over X_n is a labeled digraph N in which the set $var(N)$ of nodes is a subset of X_n, and such that each node $x \in var(N)$ is annotated with a CP-table $cpt(x)$ on x with respect to the set $par(x)$ of parents of x in the graph. The variables in $var(N)$ are called *relevant*, and the variables in $X_n \setminus var(N)$ are called *irrelevant*. The CP-net is *complete* if every relevant variable is annotated with a complete CP-table.

A CP-net is *acyclic* if its digraph is acyclic, and *tree-structured* if its digraph forms a forest, that is, a disjoint union of trees. Note that a tree-structured CP-net is an acyclic preference network where each node has at most one parent.

Recall that the size of a graph is defined by the number e of its edges. By extension, we define the *size* $|N|$ of a CP-net N to be $r + e$, where r is the total number of rules occurring in N, and e is the number of edges in the graph of N.

These different notions are illustrated in the following examples.

Example 1. Let us consider a variant of the *evening dress* domain [9]. Susan is a robopsychologist who spends most of her time at the robot manufactory. Occasionally, she is invited to evening ceremonies, but she is always late at work and must quickly change her outfits at home before getting to the ceremony. Fortunately, her domestic robot can help her to choose, among the available clean clothes, a combination that she would like the most. The robot does not know *a priori* which are Susan's outfit preferences, but it is equipped with a learning module that can extract these preferences by observing her behavior and asking simple questions.

The variables standing for the different clothes are described in the left part of Figure 1. Only three of them are relevant to Susan's preferences: they are associated to the jacket, pants, and shirt. Susan unconditionally prefers black to white as the color of both the jacket ($x_3 \succ \overline{x}_3$) and the pants ($x_4 \succ \overline{x}_4$), while her preference for a red shirt versus a white one depends on the combination of jacket and pants. Namely, a red shirt (x_5) brings a touch of color if the jacket and pants are the same colour, but a white shirt ($\overline{x}_5$) appears to be more sober if they are different.

The target CP-net N is depicted on the right part of Figure 1. N is defined on 3 relevant variables, and it is acyclic and complete. Since it contains 2 edges and 6 rules, its size is $|N| = 8$.

Example 2. Let us turn to a variant of the *flight reservation* domain [12] using our favorite character. As a research scientist, Susan often assists to conferences in different countries, by taking a flight from the USA. In this context, Susan's domestic robot can select a travel that

Variable	Literals
Airline	British Airways (x_1) and Delta ($\overline{x}_1$)
Ticket class	business (x_2) vs economy ($\overline{x}_2$) class
Departure day	one day (x_3) vs two days ($\overline{x}_3$) before
Seat	window (x_4) and aisle ($\overline{x}_4$)
Stop-over	direct (x_5) vs indirect ($\overline{x}_5$) flight
Departure time	day (x_6) vs night ($\overline{x}_6$) flight

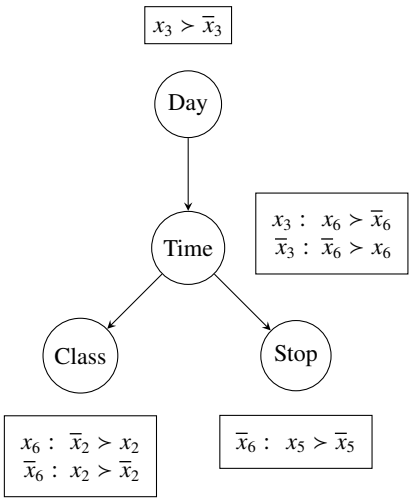

Figure 2: The domain and CP-net of "flight reservation"

optimizes her preference over the flight options, given the available resources at the reservation moment. Again, the robot does not know *a priori* which are Susan's preferences, but it can learn them by observing previous flight reservations and asking few questions.

The attributes standing for the different flight options are described in the left part of Figure 2. The relevant variables are the ticket class, departure day, departure time, and stop-over. Susan unconditionally prefers to take a flight leaving one day, instead of two days, before the conference ($x_3 \succ \overline{x}_3$). If Susan leaves two days before the conference, then she prefers an evening flight ($\overline{x}_6$) because she can work longer during the departure day. On the other hand, if she leaves one day before the conference, then she prefers a day flight (x_6) because she would like to rest few hours in the hotel before the conference opening. On a night flight, Susan prefers a business class (x_2) and a direct flight (x_5) because she expects to sleep uninterruptedly in a comfortable seat. Yet, on a day flight, Susan prefers an economy class ($\overline{x}_2$) because she is awake and does not need a large seat to read papers, but she doesn't have strict preferences on the stop-over: sometimes she likes to stretch her legs a bit in a transit airport during long travels, while at other times she prefers a direct flight in order to minimize the overall travel time.

The resulting CP-net N is depicted on the right part of Figure 2. N is defined on $|var(N)| = 4$ relevant variables, and it is tree-structured and incomplete. Because it contains 3 edges and 6 rules, the size of the network is $|N| = 9$.

2.2. *Semantics of CP-nets*

An *outcome* o is a maximal term for $X_n = \{x_1, \cdots, x_n\}$ or, equivalently, a map that assigns a Boolean value to each variable x_i in X_n. The space of all outcomes generated from X_n is denoted by O_n, and we write $\mathbf{0}$ (resp. $\mathbf{1}$) for the outcome in O_n that assigns 0 (resp. 1) to every variable in X_n. Given a variable x_i and an outcome o, we write $o(i)$ for the value which o assigns to x_i.

Given an outcome o and a term t, we write $o[t]$ for the outcome obtained by making o agree with t, that is, for all i, $o[t](i) = t(i)$ if $x_i \in var(t)$, and $o[t](i) = o(i)$ otherwise. For example, if $o = x_1x_2x_3$ and $t = x_1\overline{x}_2$, then $o[t] = x_1\overline{x}_2x_3$. An outcome o *satisfies* a term t if $o = o[t]$.

A *preference relation* $\succ$ is an irreflexive and transitive binary relation on $\mathcal{O}_n$. Based on these notions, the expression $o \succ o'$ states that o is preferred to o' according to $\succ$.

The *ceteris paribus* semantics of a CP-rule $t : p \succ \overline{p}$ on a variable x can be described as follows: if an outcome o satisfies t and assigns the value p to x, then it is preferred to the outcome o' which differs from o only in that is assigns the value $\overline{p}$ to x. In formal terms, we say that o is *preferred* to o' for $t : p \succ \overline{p}$ if $o = o[t \wedge p]$ and $o' = o[\overline{p}]$. In this case, the pair (o, o') is called a *model* of the rule $t : p \succ \overline{p}$.

By extending this semantics to preference networks, we say that o *dominates* o' for a given CP-net N if there is a sequence $(o_1, \ldots, o_m)$ such that $o_1 = o'$, $o_m = o$ and for each $i : 1 \leq i < m$, (o_{i+1}, o_i) is a model of some CP-rule of N. In this case, (o, o') is called a *model* of N, and $(o_1, \ldots, o_m)$ an *improving sequence* from o' to o.

For example, in the "evening dress" scenario, we observe that if Susan is wearing black pants (x_4) then she prefers a black jacket (x_3) and a white shirt ($\overline{x}_5$) to a white jacket ($\overline{x}_3$) and a red shirt (x_5), all other things being equal. In particular, we can infer the improving sequence:

$$(x_1 x_2 \overline{x}_3 x_4 x_5 x_6,\ x_1 x_2 \overline{x}_3 x_4 \overline{x}_5 x_6,\ x_1 x_2 x_3 x_4 \overline{x}_5 x_6)$$

The set of all models of N is denoted by $\succ_N$. A CP-net N is *consistent* if there is no outcome o which dominates itself, i.e. $o \succ_N o$. If N is consistent, then $\succ_N$ is a preference relation over the outcome space $\mathcal{O}_n$. In general, preference networks are not always consistent because they can induce cycles in improving sequences. However, as observed by Boutilier *et al.* [9], any acyclic CP-net is guaranteed to be consistent.

Finally, given two CP-nets N and N', we say that N *subsumes* N' if for any CP-rule $t' : p \succ \overline{p}$ in N', there is a CP-rule $t : p \succ \overline{p}$ in N such that $t' \subseteq t$. Clearly, if N subsumes N' then we must have both $var(N') \subseteq var(N)$ and $|N'| \leq |N|$. Yet, it is *not* generally the case that $\succ_{N'}$ is included in $\succ_N$ (or vice-versa) when N subsumes N', because missing rules remove some models, while shorter rules add some. A CP-net N is *minimal* if there is no distinct CP-net N' subsumed by N and for which $\succ_N = \succ_{N'}$. For example, the CP-nets specified in Figures 1 and 2 are minimal.

3. Exact Learning with Queries

The learning criterion expected in this study is that of *exact identification*, which is achieved if the learner can infer a CP-net that correctly represents the target concept. In this setting, the learning problems under consideration are specified by an instance class and a concept class.

An *example* of size $n > 0$ is a couple of outcomes $(o, o') \in \mathcal{O}_n \times \mathcal{O}_n$. An *instance class* is a subset $\mathcal{I}_n$ of $\mathcal{O}_n \times \mathcal{O}_n$. By $\mathcal{I} = \bigcup_{n>0} \mathcal{I}_n$, we denote the instance class graded by example size n. For example, the instance class of *swaps* (also known as *flips*) is the family of all outcome pairs that differ in the value of exactly one variable. In other words, (o, o') is a swap if $o' = o[p]$ and $o = o'[\overline{p}]$ for some literal p.

A *concept* is a preference relation $\succ$ over the outcome space $\mathcal{O}_n$. A *representation class* is a set $\mathcal{N}_n$ of CP-nets over X_n. A concept $\succ$ is *representable* by $\mathcal{N}_n$ if there is a CP-net $N \in \mathcal{N}_n$ such that $\succ_N = \succ$. The *concept class* of $\mathcal{N}_n$ is the set $\mathcal{C}^n_{\mathcal{N}}$ of all concepts that are representable by $\mathcal{N}_n$. The *description size* of a concept $\succ$ in $\mathcal{C}^n_{\mathcal{N}}$ is the minimum of $|N|$ over all representations N of $\succ$ in $\mathcal{N}_n$. By $\mathcal{C}_{\mathcal{N}} = \bigcup_{n>0} \mathcal{C}^n_{\mathcal{N}}$, we denote the concept class graded by example size n. For example, the class $\mathcal{C}_{\text{ACY}}$ of *acyclic CP-nets* is the family of all preference relations that are representable by an acyclic CP-net.

Now, let $C_{\mathcal{N}}$ be a concept class, $\mathcal{I}$ be an instance class, and $\succ$ be a target concept in $C^n_{\mathcal{N}}$. The learner may extract information about $\succ$ using two types of queries. A *membership query* MQ over $\mathcal{I}$ takes an example $(o, o') \in \mathcal{I}_n$ and returns `Yes` if $o \succ o'$, and `No` otherwise[1]. An *equivalence query* EQ over $\mathcal{I}$ takes a CP-net $N \in \mathcal{N}_n$, and returns `Yes` if N is a representation of $\succ$, or returns a counterexample $(o, o') \in \mathcal{I}_n$ otherwise. The counterexample (o, o') is *positive* if $o \succ o'$ (and hence $o \not\succ_N o'$), and *negative* if $o \not\succ o'$ (and hence $o \succ_N o'$). It is important to note that in general, $o \not\succ o'$ does *not* imply $o' \succ o$ (o and o' may be incomparable).

A minimal requirement for a learning algorithm is to identify the target concept using a polynomial number of queries.

Definition 1 (query learning)**.** An algorithm A is a *query learning algorithm* for a concept class $C_{\mathcal{N}}$ with respect to an instance class $\mathcal{I}$ if, for any input dimension $n > 0$, there are two polynomials p and q such that, for any target concept $\succ$ in $C^n_{\mathcal{N}}$, after $p(n, s)$ membership and equivalence queries over $\mathcal{I}_n$, and total running time in $q(n, s)$, A outputs a representation $N \in \mathcal{N}_n$ of $\succ$, where s is the description size of $\succ$ in $C^n_{\mathcal{N}}$. The polynomial p is called the *query complexity* of A.

A more selective requirement, which arises naturally when the number of irrelevant attributes is large, is that the number of queries be polynomial in the description size of the target concept rather than in the total number of attributes [14].

Definition 2 (attribute efficiency)**.** A query-learning algorithm is *attribute-efficient* if its query complexity depends polynomially on the description size s of the target concept $\succ$, but only polylogarithmically on the input dimension n.

We say that a concept class $C_{\mathcal{N}}$ is *attribute-efficiently learnable* with respect to an instance class $\mathcal{I}$, if there is an attribute-efficient query learning algorithm for $C_{\mathcal{N}}$ with respect to $\mathcal{I}$.

Clearly, the strength of query-directed learning lies in membership queries, which model not only the interaction with a user, but also the careful crafting of experiments by a learner in order to observe the response of the user.

In order to show that a concept class of CP-nets is *not* learnable with equivalence queries only, we use the technique of *approximate fingerprints* introduced by Angluin [2].

Intuitively, a concept class C has approximate fingerprints if it includes a set Γ such that for each hypothesis N in C supplied by the learner, the user can choose a counterexample for N that eliminates only a superpolynomially small fraction of candidate concepts in Γ. By repeating this process, the learner cannot be certain of the target concept in Γ after only polynomially many equivalence queries.

Formally, let $C_{\mathcal{N}}$ be a concept class, $\mathcal{I}$ be an instance class, and $\succ$ be a target concept in $C^n_{\mathcal{N}}$. Additionally, let $\Gamma^n_{\mathcal{N}}$ be a nonempty subset of $C^n_{\mathcal{N}}$. Then an example (o, o') in $\mathcal{I}_n$ is called an *α-fingerprint* of $\Gamma^n_{\mathcal{N}}$ according to $\succ$ if the proportion of hypotheses in $\Gamma^n_{\mathcal{N}}$ which agree with $\succ$ on (o, o') is less than α, i.e.

$$\frac{|\{\succ' \in \Gamma^n_{\mathcal{N}} : o \succ' o' \text{ iff } o \succ o'\}|}{|\Gamma^n_{\mathcal{N}}|} < \alpha$$

Intuitively, an adversary will use α-fingerprints as poorly informative counterexamples to equivalence queries.

[1] We follow the literature on learning in using the term "membership queries", asking whether the couple (o, o') is in the target preference relation, but they can also be seen as "dominance queries".

We can now apply the notion of approximate fingerprints to the setting of CP-nets.[2]

Definition 3 (approximate fingerprints)**.** A concept class C_N has *approximate fingerprints* with respect to an instance class $\mathcal{I}$ if, for any polynomial $p(n)$, C_N^n includes a subset Γ_N^n such that for any sufficiently large n, Γ_N^n contains at least two concepts, and for all concepts $\succ$ in C_N^n of description size bounded by $p(n)$, there is an example in $\mathcal{I}_n$ which is a $\frac{1}{p(n)}$-fingerprint of Γ_N^n according to $\succ$.

4. Learning Acyclic CP-nets with Queries

Acyclic CP-nets take a central part in preference research by providing the right level of expressivity for many real-world applications, while remaining tractable for certain reasoning tasks such as outcome optimization [9, 18]. We write C_{ACY} (resp. C_{ACC}) for the class of concepts which are representable by an acyclic CP-net (resp. a complete acyclic CP-net).

Before exploring the learnability issue of acyclic CP-nets, we introduce two useful properties related to their structure. Recall that two outcomes form a swap if they differ in the value of exactly one variable. Such examples correspond to simple situations of the form "I prefer this car red than white", where the color is one of the multiple attributes describing cars.

The first property states that, in acyclic CP-nets, the preference on swaps can be retrieved in linear time by simple rule matching.

Lemma 1. Let N be an acyclic CP-net and (o, o') be a swap. Then $o \succ_N o'$ if and only if there is a CP-rule r in N such that (o, o') is a model of r.

Proof. The *if* direction is immediate. Conversely, suppose $o \succ_N o'$. Then, by definition of $\succ_N$, there is an improving sequence from o' to o in N. Assume without loss of generality that o satisfies x_1 and $o' = o[\overline{x}_1]$. Using the suffix fixing rule [9, Section 5.1], we can also assume that the sequence affects only x_1 and its ascendants in N. By acyclicity, one of those ascendants, say x_k, is modified but has none of its parents modified in the sequence. However, x_k cannot be modified back to its original value, because otherwise this would imply the existence of two opposite rules $t : \overline{x}_k \succ x_k$ and $t : x_k \succ \overline{x}_k$ in the CP-table of x_k. By applying the same strategy to all remaining ascendants of x_1, it follows that only x_1 is modified, which implies that N includes a rule of the form $t : x_1 \succ \overline{x}_1$, as desired. □

The second property states that acyclic CP-nets have a canonical representation, which is minimal with respect to size.

Lemma 2. Let $\succ$ be a concept in C_{ACY}. Then there is a unique acyclic CP-net N that represents $\succ$ and has minimal size $|N|$.

Proof. Let N be a representation of $\succ$ satisfying the following condition: for any variables x and y, if y is a parent of x in N, then there is a literal p of x and an example (o, o') such that exactly one of $(o[y \wedge p], o'[y \wedge \overline{p}])$ and $(o[\overline{y} \wedge p], o'[\overline{y} \wedge \overline{p}])$ is not a model of N. This amounts to say that y is a *relevant* parent of x. Clearly, such a representation exists: take any representation and remove all irrelevant parents.

[2] In the original definition [2], approximate fingerprints are defined according to two polynomials p_1 and p_2, which respectively bound the size of target concepts and the size of examples. So here $p_1(n) = p(n)$ and $p_2(n) = n$.

We now show that any representation N' of $\succ$ subsumes N, from what the claim will follow since $|N|$ is monotonic with respect to subsumption. So, let $r \in N$ be a rule of the form $t : p \succ \overline{p}$. Any pair (o, o') for which $o = o[t \wedge p]$ and $o' = o[\overline{p}]$ is a model of r. Since $\succ_N = \succ_{N'}$, by Lemma 1, (o, o') must be a model of some rule r' in N' of the form $t' : p \succ \overline{p}$. If $t \not\subseteq t'$, then there is a variable $y \in var(t) \setminus var(t')$ such that $(o[y], o'[y])$ and $(o[\overline{y}], o'[\overline{y}])$ are both models of r', hence of N'. But by definition of N, exactly one of these pairs is not a model of N, contradicting the fact that $\succ_N = \succ_{N'}$. Therefore $t \subseteq t'$, and hence, N' subsumes N. □

4.1. Learning Acyclic CP-nets with Equivalence Queries Alone

Based on the above properties, we show that complete acyclic CP-nets have approximate fingerprints, even if the supplied examples are restricted to swaps.

Theorem 1. The class C_{acc} has approximate fingerprints with respect to swap examples.

Proof. For $n > 0$, let Γ^n_{acc} be the class of all concepts represented by a CP-net N^* with[3] $\log n$ root nodes x_j pointing to the same fixed child node x_1. Each table $cpt(x_j)$ has the rule $x_j \succ \overline{x}_j$. The table $cpt(x_1)$ includes one rule $s : x_1 \succ \overline{x}_1$, where s is the conjunction of all positive literals in $par(x_1)$, and $n - 1$ rules $s' : \overline{x}_1 \succ x_1$, where s' is any maximal term of $par(x_1)$ with at least one negative literal. Clearly N^* is acyclic and complete. Furthermore, $|\Gamma^n_{\text{acc}}| = \binom{n-1}{\log n}$.

Now, let $p(n) = n^k$ for some constant k, and consider any concept $\succ_N$ in C^n_{acc} for which the size of the minimal representation N is at most $p(n)$ (in particular, N has at most $p(n)$ rules). The fingerprint (o, o') is defined as follows.

First, if N does not include any rule of the form $t : x_1 \succ \overline{x}_1$, then o is the outcome $\mathbf{1}$ and o' is set to $o[\overline{x}_1]$. Clearly, $o \not\succ_N o'$ but $o \succ o'$ for any concept $\succ$ in Γ^n_{acc}. Therefore, (o, o') is an α-fingerprint of Γ^n_{acc} for any $\alpha > 0$.

Now, if N includes a rule of the form $t : x_1 \succ \overline{x}_1$, then o is any outcome satisfying $t \wedge x_1$ and containing $k \log n$ positive literals excluding x_1, and o' is set to $o[\overline{x}_1]$. Note that $|t| \leq k \log n$. Indeed, we know that $|N| \leq n^k$, and because N is complete, its CP-table on x_1 contains $2^{|t|}$ entries. Thus, o can be constructed by satisfying t first and filling the rest as necessary (for n large enough to ensure $n - k \log n \geq k \log n$ in case t is, say, all 0). Clearly, $o \succ_N o'$ and o has $k \log n$ positive literals (excluding x_1). Hence, the number of concepts $\succ$ in Γ^n_{acc} which agree with N on (o, o') is $\binom{k \log n}{\log n}$. Using the fact that $\frac{a-i}{b-i} \leq \frac{a}{b}$ for $b \geq a$, $i \geq 0$,

$$\begin{aligned}
\frac{|\{\succ \in \Gamma^n_{\text{acc}} : o \succ o'\}|}{|\Gamma^n_{\text{acc}}|} &= \binom{k \log n}{\log n} \Big/ \binom{n-1}{\log n} \\
&= \frac{(k \log n)!}{(k \log n - \log n)!} \frac{(n - 1 - \log n)!}{(n-1)!} \\
&= \prod_{i=0}^{\log n - 1} \frac{k \log n - i}{n - 1 - i} \\
&\leq \frac{(k \log n)^{\log n}}{(n-1)^{\log n}}
\end{aligned}$$

Taking logarithms, this proportion is less than $\frac{1}{n^k}$ if and only if $\frac{n-1}{\log n} > k 2^k$, which is true for sufficiently large n. □

[3] Notation log with always refer to base 2.

Thus, by applying Angluin's result [2, Theorem 1], we derive the following corollary.

Corollary 1. Complete acyclic CP-nets are *not* learnable with equivalence queries only.

4.2. Learning Acyclic CP-nets with Equivalence and Membership Queries

When membership queries are available, we can provide an attribute-efficient algorithm for learning acyclic, and possibly *incomplete*, CP-nets, provided that the supplied counterexamples are restricted to swaps. Importantly, observe that when there is a counterexample, there is always a *swap* counterexample (this follows from the semantics of CP-nets).

As specified in Algorithm 1, the learner starts from the empty CP-net $N = \varnothing$, and iteratively updates N by asking equivalence queries. On seeing a counterexample (o, o'), the learner first checks whether N includes a rule that covers (o, o'). If this is indeed the case, then (o, o') is negative, and the body of that rule is refined using membership queries. In all other cases, (o, o') must be positive, and a new rule is added to the table of x_i in order to cover this example. If, in addition, the swap (o, o') "violates" some rule r in N, that is, (o', o) is a model of r, then again a new parent of x_i is found using membership queries.

In our algorithm, each rule r of the form $t : p_i \succ \overline{p}_i$ in N is associated with an outcome o_r, called the *support* of r, and such that $(o_r[p_i], o_r[\overline{p}_i])$ is a model of r. This support keeps in cache the positive counterexample from which the rule was built.

The key routine SearchParent finds a new parent of some misclassifying rule r, using only a logarithmic number of membership queries. Using the support o_r of r and the last counterexample (o, o'), it operates a binary search on the sequence $(o_1, \cdots, o_n)$ where o_j is formed by the first j literals occurring in o_r and the last $n - j$ literals occurring in o. The invariant maintained by the algorithm is that $o_a[p_i] \not\succ o_a[\overline{p}_i]$ but $o_b[p_i] \succ o_b[\overline{p}_i]$ in the target concept. So, when the search has been narrowed to $a = b - 1$, flipping the value of x_b in o_a is enough for changing the preference, from what it follows that x_b is a parent of the variable of p_i.

Interestingly, we observe that membership queries are restricted to swap examples, so as to minimize the cognitive effort spent by the user in comparing outcomes. These considerations are illustrated in the following example.

Example 3. Let us carry on with the "evening dress" scenario introduced above. Suppose that each time Susan is invited to a ceremony, she informs her domestic robot to choose a suitable outfit. As a helpful assistant, the robot selects an "optimal" combination given (1) the CP-net that it has learnt so far, and (2) the fact that for some categories of clothing, only one option is available because the other is not clean enough.

When Susan gets home, she cannot afford the time to inspect every aspect of the combination before going to the ceremony. Notably, if the combination is not suitable, Susan changes only one item of clothing and wears the resulting outfit. Notice that in this case, the swap (o, o') formed by the initial combination o' and the resulting outfit o is a counterexample to the robot's hypothesis N. Then, during the following morning, the robot is allowed to ask a few membership queries in order to refine its hypothesis.

Digressing momentarily from the problem of learning Susan's preferences, it is important to keep in mind that the task of constructing an optimal extension o' of a set of predetermined literals t given an acyclic CP-net N can be performed in linear time [9, Section 3.1]: start from t and extend it by sweeping through the network N from top to bottom, setting each variable to its preferred value given the instantiation of its parents, and choosing arbitrarily if the rule is absent; finally, for each remaining (irrelevant) attribute, take an arbitrary value of the variable.

Algorithm 1: Learning Acyclic CP-nets

$N \leftarrow \varnothing$
while EQ(N) $\neq$ `Yes` do
  let (o, o') be the counterexample returned by EQ(N)
  let x_i be the variable such that $p_i \in o$ and $\overline{p}_i \in o'$
  if (o, o') is a model of some rule $\langle r, o_r \rangle$ in N then
    /* *The counterexample is negative* */
    $x_j \leftarrow$ SearchParent$(x_i, p_i, o, o_r, 0, n)$
    $par(x_i) \leftarrow par(x_i) \cup \{x_j\}$
    expand the body of each rule $\langle r', o_{r'} \rangle$ in $cpt(x_i)$ with the literal of x_j in $o_{r'}$
  else
    /* *The counterexample is positive* */
    add the rule $\langle t : p_i \succ \overline{p}_i, o \rangle$ to N where t is the projection of o onto $par(x_i)$
    if (o', o) is a model of some rule $\langle r, o_r \rangle$ in N then
      $x_j \leftarrow$ SearchParent$(x_i, p_i, o', o_r, 0, n)$
      $par(x_i) \leftarrow par(x_i) \cup \{x_j\}$
      expand the body of each rule $\langle r', o_{r'} \rangle$ in $cpt(x_i)$ with the literal of x_j in $o_{r'}$
return N

Procedure SearchParent(x_i, p_i, o, o', a, b)

if $a = b - 1$ then return x_b
$j \leftarrow \lfloor (a + b)/2 \rfloor$
$o_j \leftarrow o[t_j]$ where t_j is the term formed by the first j literals occurring in o'
if MQ$(o_j[p_i], o_j[\overline{p}_i])$ = `Yes` then
  return SearchParent(x_i, p_i, o, o', a, j)
else
  return SearchParent(x_i, p_i, o, o', j, b)

We now return to our learning problem: suppose that the current hypothesis N consists in a single rule $x_3 \succ \overline{x}_3$ stating that a black jacket is preferred to a white one (Figure 3a). In addition, suppose that during the day of the ceremony all clothes are clean, except the white pants ($\overline{x}_4$). An optimal extension o' of $t = x_4$ with respect to N is $\overline{x}_1\overline{x}_2x_3x_4\overline{x}_5\overline{x}_6$. Clearly, o' does not match Susan's preferences because a red shirt (x_5) is preferred to a white one ($\overline{x}_5$) when the colors of the jacket and the pants are identical. The modified outcome o is thus $\overline{x}_1\overline{x}_2x_3x_4x_5\overline{x}_6$. The resulting swap (o, o') is therefore a positive counterexample to N, and hence, the robot expands its hypothesis with the rule $x_5 \succ \overline{x}_5$ (Figure 3b), and stores the support $o_r = \overline{x}_1\overline{x}_2x_3x_4x_5\overline{x}_6$.

Suppose that during the day of the next ceremony, all clothes are clean, except the black pants (x_4). Here, an optimal extension o' of $\overline{x}_4$ with respect to N is $x_1x_2x_3\overline{x}_4x_5x_6$. Again, o' is not suitable because a white shirt ($\overline{x}_5$) is preferred to a red one (x_5) when the colors of the jacket and the pants are different. The modified outcome o is therefore $x_1x_2x_3\overline{x}_4\overline{x}_5x_6$. We remark here

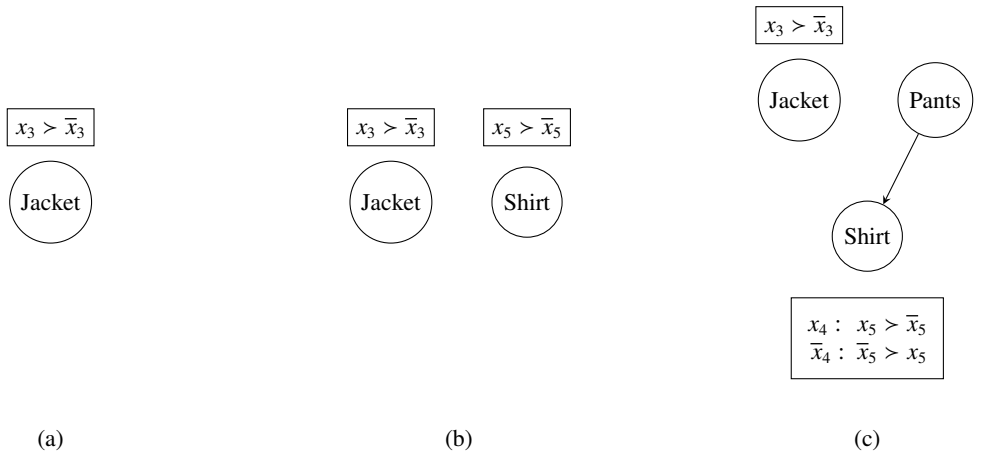

Figure 3: Learning the CP-net of "evening dress"

that the pair (o, o') is a positive counterexample to N, but the reverse pair (o', o) is a model of $x_5 \succ \overline{x}_5$. Thus, the robot must refine the body of this rule. In doing so, it can identify the parent x_4 of x_5 using only two membership queries. Indeed, a first call of the routine SearchParent with the outcomes o' and o_r, the literal x_5, and the bounds $a = 0$ and $b = 6$, gives $j = 3$, $t_j = \overline{x}_1 \wedge \overline{x}_2 \wedge x_3$, $o_j = o'[t_j] = \overline{x}_1\overline{x}_2x_3\overline{x}_4x_5x_6$, hence a membership query is asked on the swap:

$$(\overline{x}_1\overline{x}_2x_3\overline{x}_4x_5x_6, \overline{x}_1\overline{x}_2x_3\overline{x}_4\overline{x}_5x_6)$$

After receiving a negative answer, the routine SearchParent is called again with the bounds $a = 3$ and $b = 6$, yielding a membership query on the swap

$$(\overline{x}_1\overline{x}_2x_3x_4x_5x_6, \overline{x}_1\overline{x}_2x_3x_4\overline{x}_5x_6)$$

Since the answer is positive, a last call to SearchParent takes place, with the bounds $a = 3$ and $b = 4$, and leaves us with the variable x_4. Based on this new parent of x_5, the robot can refine its rule $x_5 \succ \overline{x}_5$ into $x_4 : x_5 \succ \overline{x}_5$, using the support o_r, and add the rule $\overline{x}_4 : \overline{x}_5 \succ x_5$ using the example (o, o') (Figure 3c).

As shown in the lemma below, the routine SearchParent is guaranteed to find a new parent in the rule r, by maintaining the invariant that for each explored subsequence $(o_a, \cdots, o_b)$, we have both $o_a[p_i] \not\succ o_a[\overline{p}_i]$ and $o_b[p_i] \succ o_b[\overline{p}_i]$ in the target concept.

Lemma 3. Let $\succ$ be a concept of C_{ACY}, o_a, o_b be two outcomes, and $p_i, \overline{p}_i$ be a pair of opposite literals for some variable x_i. If we have $o_a[p_i] \not\succ o_a[\overline{p}_i]$ and $o_b[p_i] \succ o_b[\overline{p}_i]$, then there is a parent x_j of x_i in the minimal representation N^* of $\succ$ whose value is different in o_a and o_b.

Proof. Consider the sequence of outcomes $(o_0, \cdots, o_n)$ such that $o_j = o_a[t_j]$, and where t_j is the conjunction of the first j literals in o_b. Since $o_0 = o_a$ and $o_n = o_b$, there is some $j > 0$ such that $o_{j-1}[p_i] \not\succ o_{j-1}[\overline{p}_i]$ and $o_j[p_i] \succ o_j[\overline{p}_i]$. Consequently, there is a rule $t : p_i \succ \overline{p}_i$ in N^* such that o_j satisfies t but o_{j-1} does not. Since they differ only on x_j, it follows that $x_j \in var(t)$. □

We now prove that our query learning algorithm is correct, by showing that it maintains the invariant that the learned hypothesis is subsumed by the target representation.

Lemma 4. Let $\succ$ be a target concept in the class C_{ACY}. Then Algorithm 1 maintains an acyclic CP-net N which is always subsumed by the minimal representation N^* of $\succ$.

Proof. Initially, $N = \varnothing$, so the property holds. Now, consider an iteration of the main loop and suppose by the inductive hypothesis that N is subsumed by N^* before calling EQ.

If N includes a rule r of the form $t : p_i \succ \overline{p}_i$ for which $o = o[t \wedge p_i]$ and $o' = o[\overline{p}_i]$, then by Lemma 1 $o \succ_N o'$ holds, so the counterexample is negative ($o \not\succ o'$). By construction, for the support o_r of r we also have $o_r = o_r[t \wedge p_i]$, and $o_r \succ o_r[\overline{p}_i]$. So by Lemma 3, SearchParent returns a parent x_j of x_i in N^*. Now, let r' be any rule of the form $t' : p'_i \succ \overline{p}'_i$ in N. Since the support $o_{r'}$ of r' satisfies $o_{r'}[p'_i] \succ o_{r'}[\overline{p}'_i]$, by the inductive hypothesis, there is a rule $t^* : p'_i \succ \overline{p}'_i$ in N^* such that $t' \subseteq t^*$ and $o_{r'}$ satisfies t^*. This together with the fact that x_j is a parent of x_i in N^* ensures $t' \cup \{o_{r'}(j)\} \subseteq t^*$, so the invariant is preserved.

Conversely, if N includes no rule as above, then by Lemma 1 $o \succ_N o'$ does not hold, so the counterexample is positive. Consider the rule $t : p_i \succ \overline{p}_i$ added to N. Since (o, o') is a positive counterexample, by Lemma 1, it follows that N^* includes a rule $t^* : p_i \succ \overline{p}_i$ for which $o = o[t^* \wedge p_i]$ and $o' = o[\overline{p}_i]$. So $t \subseteq t^*$, and the invariant is preserved.

Finally, if o satisfies t for some rule $t : \overline{p}_i \succ p_i$ in N, then (o', o) is a negative counterexample for this rule, so SearchParent returns a parent x_j of x_i in N^* just as in the first case, and again the invariant is preserved. □

With these properties in hand, we can now turn to the complexity of Algorithm 1. Concerning equivalence queries, each counterexample allows us to find a new rule $t : p_i \succ \overline{p}_i$ or a new parent x_j of some variable in the minimal representation N^* of the target concept. Because the hypothesis N is always subsumed by N^*, this can happen at most $|N^*|$ times. For membership queries, at most $\lceil \log n \rceil$ of these are used in each call of SearchParent, which always uncovers a new parent of some variable. So the number of these queries is at most $e\lceil \log n \rceil$.

Finally, because the running time of our algorithm is essentially linear in the number of queries and the number of variables, we can derive the following result.

Theorem 2. Acyclic CP-nets are attribute-efficiently learnable from equivalence and membership queries over swaps: for any $n > 0$, any target concept $\succ$ in C^n_{ACY} can be identified in polynomial time using at most $|N^*| + 1$ equivalence queries and $e\lceil \log n \rceil$ membership queries, where e is the number of edges in the minimal representation N^* of $\succ$.

5. Learning Tree CP-nets with Queries

Binary-valued tree-structured CP-nets constitute a restricted, yet important, class of preference networks for which dominance testing on *arbitrary* pairs of outcomes is solvable in quadratic time using a backtrack-free search technique [9]. It is therefore legitimate to study the learnability issues of this class in the general setting where the examples supplied to the learner are arbitrary preference situations.

In the following, we write C_{TREE} for the class of all concepts representable by a tree-structured CP-net (or *tree* CP-net for short) .

5.1. Learning Tree CP-nets with Equivalence Queries Alone

We first show that in the presence of arbitrary examples, tree CP-nets have approximate fingerprints, even if they are restricted to a single chain.

Theorem 3. The class C_{TREE} has approximate fingerprints with respect to arbitrary outcome pairs.

Proof. We assume without loss of generality that n is even to avoid floors and ceilings. To each permutation π of $(x_1, \cdots, x_n)$ we associate the smallest set of rules N_π defined as follows: $x_{\pi(1)} \succ \overline{x}_{\pi(1)}$ is in N_π, and for each $i > 1$, $x_{\pi(i-1)} : x_{\pi(i)} \succ \overline{x}_{\pi(i)}$ is in N_π. Let Γ^n_{TREE} be the class of all concepts represented by some N_π specified as above. Clearly, $|\Gamma^n_{\text{TREE}}| = n!$.

Now, let $p(n) = n^k$ for some constant k and let $N \in C_{\text{TREE}}$ of size at most $p(n)$.[4] The fingerprint (o, o') is defined in the following way.

First, suppose that there is an outcome o_1 containing at least $n/2$ ones and such that $(o_1, \mathbf{0})$ is a model of N. Then, there is an improving sequence from $\mathbf{0}$ to o_1 in N, and since variables are flipped one by one, it must contain an outcome o with exactly $n/2$ ones. Moreover, by construction $o \succ_N \mathbf{0}$ holds. Let $o' = \mathbf{0}$. We claim that (o, o') is an $\frac{1}{n^k}$-fingerprint of Γ^n_{TREE} with respect to $\succ_N$. Indeed, a concept N_π in Γ^n_{TREE} has (o, o') as a model if and only if the first $n/2$ variables according to π are exactly those assigned 1 by o. Otherwise, any improving sequence in N_π should flip at least one variable assigned 0 by both o and o', with no way back, a contradiction. It follows that there are $(n/2)!(n/2)!$ concepts in Γ^n_{TREE} with (o, o') as a model, and hence

$$\frac{|\{\succ \in \Gamma^n_{\text{TREE}} : o \succ o'\}|}{|\Gamma^n_{\text{TREE}}|} = \frac{\left(\frac{n}{2}!\right)^2}{n!} = \binom{n}{\frac{n}{2}}^{-1}$$

Using the binomial theorem, we know that $2^n = \sum_{i=0}^{n} \binom{n}{i}$. This, together with the fact that the largest term in the sum is given for $i = \frac{n}{2}$, we have $2^n \leq (n+1)\binom{n}{\frac{n}{2}}$. Therefore,

$$\frac{|\{\succ \in \Gamma^n_{\text{TREE}} : o \succ o'\}|}{|\Gamma^n_{\text{TREE}}|} \leq \frac{n+1}{2^n}$$

This ratio is clearly less than $\frac{1}{n^k}$ for sufficiently large n.

Now, assume that there is no o_1 as above. Let $o = \mathbf{1}$ and $o' = \mathbf{0}$. So $o \not\succ_N o'$, but $o \succ o'$ holds for every concept $\succ$ in Γ^n_{TREE}. Therefore, (o, o') is an α-fingerprint of Γ^n_{TREE} for any $\alpha > 0$. □

Corollary 2. Tree CP-nets are not learnable with equivalence queries alone.

5.2. Learning Tree CP-nets with Equivalence and Membership Queries

As further evidence for the utility of membership queries, we now give an attribute-efficient algorithm for eliciting tree CP-nets in the presence of *arbitrary* examples. Let $\triangle(o, o')$ be the set of all variables whose value differ in two outcomes o and o'. For example, if $o = x_1x_2x_3\overline{x}_4$ and $o' = \overline{x}_1x_2x_3x_4$, then $\triangle(o, o') = \{x_1, x_4\}$.

Algorithm 2 uses the fact that considering only variables in $\triangle(o, o')$ and their ascendants is enough for finding an improving sequence from o' to o (suffix fixing rule). Thus, on seeing a counterexample (o, o'), the learner computes the tables for each such variable. Because any variable has at most one parent, its table can be found using few membership queries.

From a practical perspective, it is important to emphasize that the examples used in membership queries are restricted to swaps in order to minimize the cognitive effort spent by the user in comparing outcomes. Furthermore, the outcomes $\mathbf{1}$ and $\mathbf{0}$ used in the routine Propagate can naturally be replaced by any suitable pair $(o, \overline{o})$ for which $\overline{o} = \{\overline{p} : p \in o\}$.

[4] For a tree-structured CP-net N, $|N| \leq 3n$ always holds (at most two rules and one parent per variable), so as soon as $p(n) \geq 3n$, N can be any tree-structured CP-net.

Algorithm 2: Learning tree CP-nets

$N \leftarrow \varnothing$
while EQ(N) $\neq$ `Yes` do
    let (o, o') be the (positive) counterexample returned by EQ(N)
    for each $x \in \triangle(o, o')$ do PROPAGATE(x)
return N

Procedure PROPAGATE(x)

if $x \in var(N)$ then return
$var(N) \leftarrow var(N) \cup \{x\}$
foreach $o \in \{\mathbf{0}, \mathbf{1}\}$ and $p \in \{x, \overline{x}\}$ do $pref(o, p) \leftarrow$ MQ($o[p], o[\overline{p}]$)
if $pref(\mathbf{0}, p) = pref(\mathbf{1}, p) =$ `Yes` for some $p \in \{x, \overline{x}\}$ then
    $par(x) \leftarrow \varnothing$
else if $pref(\mathbf{0}, p) =$ `Yes` for some $p \in \{x, \overline{x}\}$ then
    $par(x) \leftarrow \{$SEARCHPARENT$(x, p, \mathbf{1}, \mathbf{0}, 0, n)\}$
else if $pref(\mathbf{1}, p) =$ `Yes` for some $p \in \{x, \overline{x}\}$ then
    $par(x) \leftarrow \{$SEARCHPARENT$(x, p, \mathbf{0}, \mathbf{1}, 0, n)\}$
else
    $par(x) \leftarrow \varnothing$
add to N a table for x w.r.t. $par(x)$ and with the rules validating the answers $pref(o, p)$
if $par(x) = \{y\}$ then PROPAGATE(y)

Example 4. Let us consider again the *flight reservation* domain introduced in Example 2. Suppose that when Susan is invited to a conference, her domestic robot first contacts several travel agencies through the Web, next collects the available flights, and then presents these options to Susan in some order that is consistent with the flight preferences that it has learnt so far.

In the setting suggested by this scenario, we say that an ordering $(o_1, \ldots, o_m)$ of outcomes is *consistent* with a CP-net N if there is no pair (o_i, o_j) for which $1 \leq i < j \leq m$ and $o_j \succ_N o_i$. The task of finding a consistent ordering for a set of m outcomes can be accomplished in $O\left(|var(N)|m^2\right)$ time, provided that N is acyclic [9, Section 4.1]. Indeed, for each outcome pair (o_i, o_j) it suffices to check whether there exists a variable x such that o_i and o_j assign the same values to all ancestors of x in N, and o_i assigns a more preferred value to x than that assigned by o_j; if this is the case, then we know that $o_j \not\succ_N o_i$, and hence o_i is consistently orderable over o_j.

Returning to our scenario, when Susan is presented an ordering $(o_1, \ldots, o_m)$ of the available flights, she selects the best compromise o_i and makes the reservation. If $i \neq 1$, then (o_i, o_1) is a positive counterexample to the robot's hypothesis N because $o_i \succ o_1$ but $o_i \not\succ_N o_1$. In this case, the robot is allowed to ask membership queries when Susan comes back from the conference.

Let us take an instance of this scenario by assuming that the robot has already learnt the rule: "Susan prefers to take a flight one day (x_3), instead of two days ($\overline{x}_3$), before the conference". The corresponding CP-net N is depicted in Figure 4a. Now, suppose that after contacting the agencies, the robot has collected three possible reservations:

$$o_1 = x_1\overline{x}_2x_3x_4x_5\overline{x}_6$$
$$o_2 = x_1x_2x_3x_4x_5x_6$$
$$o_3 = \overline{x}_1x_2\overline{x}_3\overline{x}_4x_5x_6$$

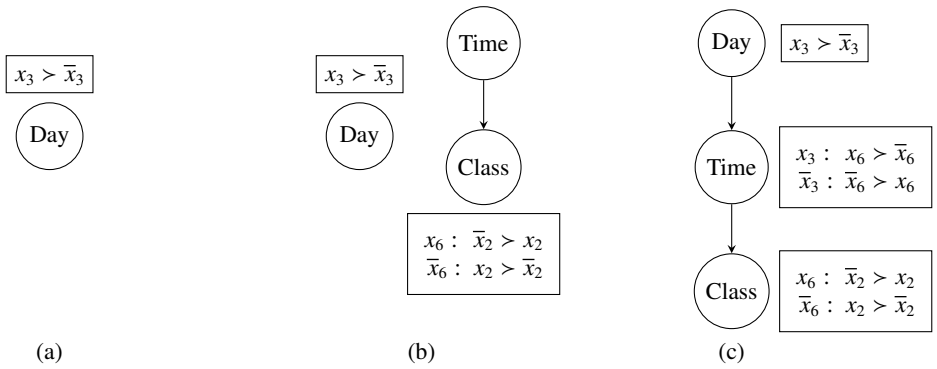

Figure 4: Learning the CP-net of "flight reservation"

The first two options suggest to take a British Airways (x_1) direct flight (x_5) with window seat (x_4) by leaving one day before the conference (x_3). The first option is a night flight ($\overline{x}_6$) in economy class ($\overline{x}_2$), while the second option is a day flight (x_6) in business class (x_2). The third option differs from the second one by taking a Delta flight ($\overline{x}_1$) with aisle seat ($\overline{x}_4$), and leaving two days before the conference ($\overline{x}_3$).

Now, assume that the robot presents these options in the order (o_1, o_2, o_3), which is consistent with N. However, because o_2 dominates o_1 according to the target CP-net presented in Example 2, Susan chooses o_2 and so, (o_2, o_1) is a positive counterexample. Based on the fact that $\triangle(o_2, o_1) = \{x_2, x_6\}$, the robot picks the first variable x_2 and calls the routine PROPAGATE on it, according to Algorithm 2. Thus, the robot makes four membership queries, which give rise to the corresponding answers:

$$\begin{aligned}
&\mathit{pref}(\mathbf{0}, x_2) \leftarrow \mathrm{MQ}(\overline{x}_1 x_2 \overline{x}_3 \overline{x}_4 \overline{x}_5 \overline{x}_6, \overline{x}_1 \overline{x}_2 \overline{x}_3 \overline{x}_4 \overline{x}_5 \overline{x}_6) = \texttt{Yes} \\
&\mathit{pref}(\mathbf{0}, \overline{x}_2) \leftarrow \mathrm{MQ}(\overline{x}_1 \overline{x}_2 \overline{x}_3 \overline{x}_4 \overline{x}_5 \overline{x}_6, \overline{x}_1 x_2 \overline{x}_3 \overline{x}_4 \overline{x}_5 \overline{x}_6) = \texttt{No} \\
&\mathit{pref}(\mathbf{1}, x_2) \leftarrow \mathrm{MQ}(x_1 x_2 x_3 x_4 x_5 x_6, x_1 \overline{x}_2 x_3 x_4 x_5 x_6) = \texttt{No} \\
&\mathit{pref}(\mathbf{1}, \overline{x}_2) \leftarrow \mathrm{MQ}(x_1 \overline{x}_2 x_3 x_4 x_5 x_6, x_1 x_2 x_3 x_4 x_5 x_6) = \texttt{Yes}
\end{aligned}$$

Based on this information, the robot can find the parent of x_2 and fill its CP-table using only three additional membership queries. Indeed, because $\mathit{pref}(\mathbf{0}, x_2) = \texttt{Yes}$ and $\mathit{pref}(\mathbf{1}, x_2) = \texttt{No}$, a first call of the routine SEARCHPARENT yields:

$$(\overline{x}_1 x_2 \overline{x}_3 x_4 x_5 x_6, \overline{x}_1 \overline{x}_2 \overline{x}_3 x_4 x_5 x_6)$$

Making a membership query with this swap and finding that this is a negative example, a second call of SEARCHPARENT (with $a = 3$ and $b = 6$) gives:

$$(\overline{x}_1 x_2 \overline{x}_3 \overline{x}_4 x_5 x_6, \overline{x}_1 \overline{x}_2 \overline{x}_3 \overline{x}_4 x_5 x_6)$$

A membership query with this swap shows that this is also a negative example, hence SEARCHPARENT is called with $a = 4$ and $b = 6$, resulting in the following membership query:

$$(\overline{x}_1 x_2 \overline{x}_3 \overline{x}_4 \overline{x}_5 x_6, \overline{x}_1 \overline{x}_2 \overline{x}_3 \overline{x}_4 \overline{x}_5 x_6)$$

Again this is a negative example, implying that x_6 is the parent of x_2. The resulting table emerges from the fact that $(\mathbf{0}[x_2], \mathbf{0}[\overline{x}_2])$ is a model of the rule $\overline{x}_6 : x_2 \succ \overline{x}_2$, and $(\mathbf{1}[\overline{x}_2], \mathbf{1}[x_2])$ is a model of the rule $x_6 : \overline{x}_2 \succ x_2$. The intermediate hypothesis is depicted in Figure 4b.

Now, because x_6 is a parent of x_2, the routine Propagate is recursively called on x_6. Based on a similar strategy, the robot can fill the table of x_6 using only seven membership queries: four queries are required for revealing that x_6 has a parent, and three additional queries are sufficient to identify that parent: x_3. Since the table of x_3 has already been filled, the propagation is stopped, and the robot picks x_6 in $\triangle(o_2, o_1)$. Yet, because x_6 has already been processed, the robot is left with the hypothesis displayed in Figure 4c.

Lemma 5. Let N^* be a minimal representation of the target concept in C_{TREE}. Then Propagate is called only on variables x in $var(N^*)$, and always extracts the table of x in N^*.

Proof. By construction, when Propagate(x_i) is called by Algorithm 2, we must have $x \in \triangle(o, o')$ and $o \succ o'$. So $x \in var(N^*)$, because otherwise it would have no table in N^* and hence, its value could not change along any improving sequence from o' to o.

Now, given $x \in var(N^*)$, we show that Propagate computes the right set of parents for x. First, suppose that $\text{MQ}(\mathbf{0}[p], \mathbf{0}[\overline{p}]) = \text{MQ}(\mathbf{1}[p], \mathbf{1}[\overline{p}]) = \texttt{Yes}$. By Lemma 1 there is a rule $t : p \succ \overline{p}$ in N^* such that both $\mathbf{0}$ and $\mathbf{1}$ satisfy t. So t is empty, and hence, x has no parent in N^*. Second, suppose that $\text{MQ}(\mathbf{0}[p], \mathbf{0}[\overline{p}]) = \texttt{Yes}$ and $\text{MQ}(\mathbf{1}[p], \mathbf{1}[\overline{p}]) = \texttt{No}$. By Lemma 3 there is a parent y of x in N^*, which is found by SearchParent. The third case is symmetric. In the last case, all queries answer No, so there is no rule on x in N^*, implying that x has no parent in N^*.

Consequently, in all cases Propagate computes the right set of (at most one) parents. Because each possible rule is validated by one of the queries $\text{MQ}(o[p], o[\overline{p}])$, the table computed for x is the correct one. Furthermore, since each recursive call of Propagate is on a variable y which is the parent of some variable in $var(N^*)$, we have $y \in var(N^*)$. □

By Lemma 5, it follows that all counterexamples supplied to the learner are positive. Moreover, from the structure of the algorithm it follows that after treating (o, o'), the hypothesis N contains the correct tables for all ascendants of all variables in $\triangle(o, o')$. This, together with the suffix fixing principle [9, Section 5.1] ensures that N now agrees with $o \succ o'$, and hence, the algorithm is guaranteed to converge.

Concerning the complexity of our learning algorithm, the number of equivalence queries is at most $|var(N^*)| + 1$, because each counterexample allows the learner to treat at least one new variable in N^*. Likewise, the routine Propagate treats each variable in $var(N^*)$ exactly once, using at most 4 membership queries for collecting the answers $pref(o, p)$ plus $\lceil \log n \rceil$ for identifying a parent. Finally, the hypothesis maintained by the learner is always a subtree of N^*, and hence, dominance testing can be evaluated in quadratic time.

Theorem 4. Tree CP-nets are attribute-efficiently learnable from equivalence and membership queries over arbitrary outcome pairs: for any $n > 0$, any $\succ$ in C^n_{TREE} can be identified in polynomial time using at most $|var(N^*)|+1$ equivalence queries and $4|var(N^*)|+e\lceil \log n \rceil$ membership queries, where N^* is the minimal representation of $\succ$, and e is the number of edges in N^*.

We mention in passing that our algorithm uses $O(n \log n)$ membership queries for making its hypothesis cover a positive example (o, o'), while the length of a shortest improving sequence from o' to o may be in $\Theta(n^2)$ [9, Theorem 13]. Thus, in essence, the learner does not need to reconstruct a proof that (o, o') is positive, neither is the user required to supply one.

6. Lower bounds

In this section, we show that our learning algorithms are quasi-optimal, by deriving lower bounds on the query complexity of their target concept classes. To this point, it is well-known that the Vapnik-Chervonenkis (VC) dimension of a concept class provides a lower bound on the query complexity of that class [5, 28]. Based on this key result, we shall assess the quasi-optimality of our learning algorithms by identifying lower bounds on the VC-dimension of acyclic CP-nets and tree-structured CP-nets.

Intuitively, the VC-dimension of a concept class $\mathcal{C}$ is a measure of the "richness" of $\mathcal{C}$: it captures the maximum number d of examples that can be labeled as either positive or negative in all the 2^d ways using the concepts in $\mathcal{C}$.

In the setting of preference networks, let $\mathcal{C}_\mathcal{N}$ be the concept class induced by some representation class $\mathcal{N}$, and let $\mathcal{I}$ be an instance class. Given a set of examples $O \subseteq \mathcal{I}$, a *labeling* of O is a map f from O to $\{0, 1\}$. Any outcome pair $(o, o') \in O$ for which $f(o, o') = 1$ is called a *positive* example, and dually, any pair $(o, o') \in O$ for which $f(o, o') = 0$ is called a *negative* example. Clearly, there are $2^{|O|}$ distinct labelings of O.

Borrowing the terminology of concept learning, we say that a CP-net N is *consistent* with a labeling f of O if $o \succ_N o'$ for every example (o, o') such that $f(o, o') = 1$, and $o \not\succ_N o'$ for every example (o, o') such that $f(o, o') = 0$. In addition, we say that O is *shattered* by $\mathcal{C}_\mathcal{N}$ if for each distinct labeling f of O, there is a representation $N \in \mathcal{N}$ that is consistent with f. Based on these notions, the *VC-dimension* of $\mathcal{C}_\mathcal{N}$ with respect to $\mathcal{I}$, denoted $\mathrm{VCdim}(\mathcal{C}_\mathcal{N}, \mathcal{I})$, is defined to be the maximum size of any subset O of $\mathcal{I}$ that is shattered by $\mathcal{C}_\mathcal{N}$.

6.1. VC-dimension of single CP-tables

Given two integers n and k such that $0 \leq k < n$, we denote by $\mathcal{C}^{n,k}_{\mathrm{CPT}}$ the class of preference orderings over n-dimensional outcomes that are representable by a single CP-table with 2^k entries. In other words, each concept in $\mathcal{C}^{n,k}_{\mathrm{CPT}}$ can be represented by a CP-net involving $k+1$ vertices chosen among the n possible variables; the first k vertices are root nodes, each with an empty CP-table, that are pointing to the $(k+1)$th vertex with a complete CP-table. Our prime concern in this section is to examine the VC-dimension of this class, from which we can easily derive lower bounds on the VC-dimension of more expressive classes.

Because our learning algorithms use swap examples as membership queries, we are looking for a set of swaps (as large as possible) which is shattered by $\mathcal{C}^{n,k}_{\mathrm{CPT}}$. In brief, we construct a set of swaps O over $1 + kq$ variables: the first variable is the one over which the outcomes in each swap differ, and the kq remaining variables are the candidate parents. The set O will be shattered by $\mathcal{C}^{n,k}_{\mathrm{CPT}}$ because for each possible labeling of O, our construction will ensure that there is a set of k parents for the first variable, among the kq candidates, over which there is a CP-table consistent with the labeling. The key idea is to duplicate each parent variable q times, and to build examples so that the learner necessarily hesitates about at least one parent (*i.e.*, between the true parent and one of its copies) until it has seen the labeling of all examples in O.

Formally, let $Y \subseteq X_n$ be a subset of variables of size $q \leq n$, and $T = \{t_1, \cdots, t_m\}$ a set of terms that are maximal for Y. By taking each term $t_i = t_{i1} \cdots t_{iq}$ as a "row" vector of Boolean entries t_{ij}, T can be viewed as an mq-Boolean matrix, where m and q are respectively the number of rows and the number of columns in the matrix. Based on this observation, we say that T is a *shattered block* if any m-dimensional Boolean vector is a column of T, that is, for every $u \in \{0, 1\}^m$, there is an index $j \in [1, q]$ such that $u = t_{1j} \cdots t_{mj}$.

	Y_1	Y_2	Y_3
O_1	00010111	11111111	1
	00101011	11111111	1
	01001101	11111111	1
O_2	11111111	00010111	1
	11111111	00101011	1
	11111111	01001101	1
O_3	00000000	00000000	1

(a)

	f	Y_1	Y_2	Y_3
O_1	1	$\cdots\cdots1\cdot\cdot$	$\cdot\cdot1\cdots\cdots$	$\cdot$
	0	$\cdots\cdots0\cdot\cdot$	$\cdot\cdot1\cdots\cdots$	$\cdot$
	1	$\cdots\cdots1\cdot\cdot$	$\cdot\cdot1\cdots\cdots$	$\cdot$
O_2	0	$\cdots\cdots1\cdot\cdot$	$\cdot\cdot0\cdots\cdots$	$\cdot$
	1	$\cdots\cdots1\cdot\cdot$	$\cdot\cdot1\cdots\cdots$	$\cdot$
	0	$\cdots\cdots1\cdot\cdot$	$\cdot\cdot0\cdots\cdots$	$\cdot$
O_3	1	$\cdots\cdots0\cdot\cdot$	$\cdot\cdot0\cdots\cdots$	$\cdot$

(b)

Figure 5: Construction (a) and labeling (b) of O for $k = 2$

Clearly, if T is a shattered block with m rows, then the numbers of columns in T is at least 2^m. For example, the following set is a shattered block with $m = 3$ rows and $q = 9$ columns:

$$\overline{x}_1\overline{x}_2\overline{x}_3x_4\overline{x}_5x_6x_7x_8x_9$$
$$\overline{x}_1\overline{x}_2x_3\overline{x}_4x_5\overline{x}_6x_7x_8x_9$$
$$\overline{x}_1x_2\overline{x}_3\overline{x}_4x_5x_6\overline{x}_7x_8x_9$$

Let O be a set of swaps of the form $(o[x], o[\overline{x}])$ for some variable $x \in X_n$, and let $Y \subseteq X_n \setminus \{x\}$. Then, the *projection* of O onto Y, denoted $O(Y)$ is the set of terms $\{o(Y) : (o, o') \in O\}$ where $o(Y)$ is the projection of o onto Y. For example, if O consists of two swaps $(x_1\overline{x}_2x_3x, x_1\overline{x}_2x_3\overline{x})$ and $(\overline{x}_1x_2\overline{x}_3\overline{x}, \overline{x}_1x_2\overline{x}_3x)$, then the projection of O onto $\{x_1, x_2\}$ is $\{x_1\overline{x}_2, \overline{x}_1x_2\}$.

We are now in position to construct the target sample for the class $C^{n,k}_{\text{CPT}}$. The key idea is to select an arbitrary variable x in X_n, and to build $k + 1$ sets of swaps on x. The first k sets are shattered blocks, each associated with a parent of x, and the last set is used to instantiate the variables in X_n which do not occur in the shattered blocks.

Definition 4. Let k, n be integers such that $0 \leq k < n$. Let X_n be a set of Boolean variables, x be an arbitrary variable in X_n and $\{Y_1, ..., Y_{k+1}, \{x\}\}$ be a partition of X_n with Y_{k+1} possibly empty. Then, any set O of examples is said to be *generated* by $\{Y_1, ..., Y_{k+1}, \{x\}\}$ if $O = O_1 \cup \cdots \cup O_{k+1}$, where each O_i is a set of swaps of the form $(o[x], o[\overline{x}])$ satisfying the following conditions:

1. if $i \in [1, k]$, then
   (a) $O_i(Y_i)$ is a shattered block with $\lfloor \log |Y_i| \rfloor$ rows,
   (b) $O_i(Y_j) = \{\mathbf{1}\}$ for every $j \in [1, i-1] \cup [i+1, k]$,
   (c) $O_i(Y_{k+1}) = \{\mathbf{1}\}$;
2. if $i = k + 1$, then for each swap $(o_i[x], o_i[\overline{x}]) \in O_i$
   (a) for every $j \in [1, k]$, $o_i(Y_j) = \{\mathbf{0}\}$ or $o_i(Y_j) = \{\mathbf{1}\}$, and the number of indices j in $[1, k]$ for which $o_i(Y_j) = \{\mathbf{0}\}$ is at least $\min(k, 2)$,
   (b) $o_i(Y_{k+1}) = \{\mathbf{1}\}$.

The construction is illustrated in Figure 5a for $k = 2$ and $n = 18$. Let $X_n = \{x_1, \cdots, x_{17}, x\}$. The set O of examples encoded in the Boolean matrix is generated by a partition $\{Y_1, Y_2, Y_3, \{x\}\}$ of X_n, where $|Y_1| = |Y_2| = 8$ and $|Y_3| = 1$. Any projection $O_i(Y_j)$ is depicted by a corresponding block in the Boolean matrix. Each row in the matrix encodes a swap $(o[x], o[\overline{x}])$ in O, where o is formed by simply expanding the row with x.

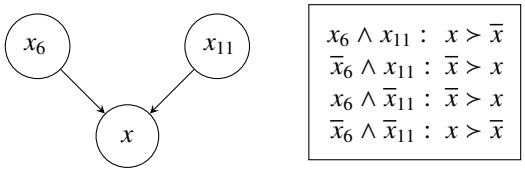

Figure 6: A CP-net in $C^{n,k}_{\text{CPT}}$ consistent with the labeling of O

A labeling of O is given by the second column of the table in Figure 5b. As a key property of our construction, f is guaranteed to match at least one column of each shattered block $O_i(Y_i)$. Here, f matches the sixth column of $O_1(Y_1)$ and the third column of $O_2(Y_2)$. Thus, by taking x_6 and x_{11} as parents of x, we can easily build a CP-net (Figure 6) that is consistent with f.

In a nutshell, any labeling of O is captured by at most one parent for x in each Y_i for $i \leq k$, while Y_{k+1} plays no other role than gathering together the variables not in other Y_i's.

Lemma 6. Let $O = O_1 \cup \cdots \cup O_{k+1}$ be a set of examples generated by a partition $\{Y_1, ..., Y_{k+1}, \{x\}\}$ of X_n, where $0 \leq k < n$. Then $O_1, \cdots, O_k$ and O_{k+1} are mutually disjoint.

Proof. By construction, we know that any row in a shattered block must include at least one variable valued to 1, and at least one variable valued to 0. Based on this observation, let i be an index in $[1, k]$. We know that $O_i(Y_i)$ is a shattered block with m rows, and $O_j(Y_i) = \{\mathbf{1}\}$ for all $j \in [1, i-1] \cup [i+1, k]$. Since $O_i(Y_i)$ does not include the term $\mathbf{1}$, it follows that O_i is disjoint from $O_1 \cup \cdots \cup O_{i-1} \cup O_{i+1} \cup \cdots \cup O_k$. Furthermore, we know that $O_{k+1}(Y_i) = \{\mathbf{1}\}$ or $O_{k+1}(Y_i) = \{\mathbf{0}\}$. Therefore, O_i is disjoint from O_{k+1} because $O_i(Y_i)$ contains neither $\mathbf{0}$ nor $\mathbf{1}$. By applying the same strategy for all indexes $i \in [i, k]$, the result follows. □

Lemma 7. Let $O = O_1 \cup \cdots \cup O_{k+1}$ be a set of examples generated by a partition $\{Y_1, ..., Y_{k+1}, \{x\}\}$ of X_n, where $0 \leq k < n$. Then O is shattered by $C^{n,k}_{\text{CPT}}$.

Proof. We successively examine the cases where $k = 0$, $k = 1$, and $k > 1$.

Case $k = 0$. Observe here that O is reduced to $\{(\mathbf{1}[x], \mathbf{1}[\overline{x}])\}$. So, there are only two possible labelings f of O. Namely, if $f(\mathbf{1}[x], \mathbf{1}[\overline{x}]) = 0$ then $\overline{x} \succ x$ is consistent with f, and dually, if $f(\mathbf{1}[x], \mathbf{1}[\overline{x}]) = 1$ then $x \succ \overline{x}$ is consistent with f. Therefore, O is shattered by $C^{n,0}_{\text{CPT}}$.

Case $k = 1$. By construction, $O = O_1 \cup O_2$, where $O_1(Y_1)$ is a shattered block and $O_2(Y_1) = \{o_2\}$, with $o_2(Y_1) = \mathbf{0}$ and $o_2(Y_2) = \mathbf{1}$. Consider any labeling f of O. If $f(o_2[x], o_2[\overline{x}]) = 0$, then let x_j be a variable in Y_1 for which the column $(x_{1j} \cdots x_{mj})$ in $O_1(Y_1)$ matches $f(O_1)$, that is, $x_{ij} = f(o[x], o[\overline{x}])$ for all swaps $(o[x], o[\overline{x}])$ in O_1. Then, the CP-table on x defined by the entries $x_j : x \succ \overline{x}$ and $\overline{x}_j : \overline{x} \succ x$ is consistent with f on O. Dually, if $f(o_2[x], o_2[\overline{x}]) = 1$, then let x_j be a variable in Y_1 for which the column $(x_{1j} \cdots x_{mj})$ in $O_1(Y_1)$ matches the bitwise complement of $f(O_1)$. Then, the CP-table $\{\overline{x}_j : x \succ \overline{x},\ x_j : \overline{x} \succ x\}$ is consistent with f on O. By combining both results, it follows that O is shattered by $C^{n,1}_{\text{CPT}}$.

Case $k > 1$. Consider any labeling f of O. We start by showing by induction on $i \in [1, k]$ that there is a CP-table N_i, containing $i + 1$ rules on x with at most one negative literal per condition, which is consistent with f on $O_1 \cup \cdots \cup O_i$. First, consider the base case $i = 1$. Let x_{j_1} be a

variable in Y_1 for which the column in $O_1(Y_1)$ matches $f(O_1)$. Then, the CP-table N_1 defined by the entries $x_{j_1} : x \succ \overline{x}$ and $\overline{x}_{j_1} : \overline{x} \succ x$ is consistent with f on O_1.

Now, suppose by induction hypothesis that there exists a CP-table N_{i-1}, containing i rules on x with at most one negative literal per condition, which is consistent with $O_1 \cup \cdots \cup O_{i-1}$. Then, the CP-table N_i is constructed as follows. First, for each of the i rules r occurring in N_{i-1}, expand the condition of r with x_{j_i} (defined as x_{j_1} above) and add the resulting rule in N_i. Then, add $x_{j_1} \cdots x_{j_{i-1}} \overline{x}_j : \overline{x} \succ x$ to N_i. By the inductive hypothesis, the first i rules ensure that N_i is consistent with f on $O_1 \cup \cdots \cup O_{i-1}$. Moreover, one of these rules must be $x_{j_1} \cdots x_{j_i} : x \succ \overline{x}$. This, together with $x_{j_1} \cdots x_{j_{i-1}} \overline{x}_j : \overline{x} \succ x$, ensures that N_i is consistent with f on O_i. Notice that N_i includes $i + 1$ rules with at most one negative literal per condition.

After constructing the table N_k, we build the final CP-table N by conjoining N_k with the new table N_{k+1} defined as follows: for each swap $(o[x], o[\overline{x}])$ in O_{k+1} add the rule $p_{j_1} \cdots p_{j_k} : p \succ \overline{p}$, where p_{j_i} is the projection of o on the variable x_{j_i}, and $p = f(o[x], o[\overline{x}])$. By construction, N_{k+1} is consistent with O_{k+1}, and includes $2^k - k - 1$ rules with at least two negative literals per condition. Therefore, N_k and N_{k+1} have disjoint entries, and hence, N is consistent with O.

By applying the same strategy to all labelings f of O, it follows that O is shattered by $C^{n,k}_{\text{CPT}}$, as desired. □

We are now in position to derive a lower bound on the VC-dimension of CP-tables.

Theorem 5. The VC-dimension of $C^{n,k}_{\text{CPT}}$ with respect to swaps is at least:

- 1 if $k = 0$,
- $m + 1$ if $k = 1$, and
- $2^k + k(m - 1) - 1$ if $k > 1$,

where $m = \left\lfloor \log \frac{n-1}{k} \right\rfloor$.

Proof. Let $q = 1$ if $k = 0$, and $m = \left\lfloor \log \frac{n-1}{k} \right\rfloor$, $q = 2^m$ if $k > 0$.Let x be a variable in X, let $\{Y_1, \cdots, Y_{k+1}, \{x\}\}$ be a partition of X, where $|Y_i| = q$ for $i \in [1, k]$, and let $O = O_1 \cup \cdots \cup O_{k+1}$ be a set of examples generated by $\{Y_1, \cdots, Y_{k+1}, \{x\}\}$. By Lemma 6, we know that $O_1, \cdots, O_k$ and O_{k+1} are mutually disjoint. Based on this property, let us examine the size of O.

- If $k = 0$, then the partition of X is reduced to $\{Y_1\} = \{X \setminus \{x\}\}$, and hence, O is reduced to $O_1 = \{(\mathbf{1}[x], \mathbf{1}[\overline{x}])\}$. So $|O| = 1$.
- If $k = 1$, then O is given by $O_1 \cup O_2$ where $|O_1|$ is a shattered block with m rows and $O_2(Y_1) = \{\mathbf{0}\}$. So $|O| = m + 1$.
- If $k > 1$, then O is formed by the union of the disjoint sets $O_1, \cdots, O_k$ and O_{k+1}, where $|O_1| = \cdots |O_k| = m$, and $|O_{k+1}| = \sum_{i=2}^{k} \binom{i}{k} = 2^k - k - 1$. So $O = 2^k + k(m - 1) - 1$.

Since by application of Lemma 7, we know that O is shattered by $C^{n,k}_{\text{CPT}}$, the result follows. □

6.2. VC dimension of CP-nets

We now turn to the VC-dimension of acyclic CP-nets. Given $n, k, e \in \mathbb{N}$ where $0 \leq k < n$ and $k \leq e \leq \binom{n}{2}$, let $C^{n,k,e}_{\text{ACY}}$ be the class of concepts which are representable by an acyclic CP-net for which the in-degree is at most k and the number of edges is at most e. We derive a lower-bound on the VC-dimension of this class from that of single CP-tables, by considering a collection of such tables organized in a special acyclic graph.

Theorem 6. The VC-dimension of $C^{n,k,e}_{\text{ACY}}$ with respect to swaps is at least:

- 1 if $k = 0$,
- $v(m + 1)$ if $k = 1$, and
- $v(2^k + k(m - 1) - 1))$ if $k > 1$,

where $v = \left\lfloor \frac{e}{k} \right\rfloor$ and $m = \left\lfloor \log \frac{n-v}{k} \right\rfloor$.

Proof. For $k = 0$ (and hence, $e = 0$), the result is a direct application of Theorem 5. For $k \geq 1$, let V be a subset of X containing v variables, and for each variable x in V, let O_x be a set of examples generated by $\{Y_1, ..., Y_{k+1}, \{x\}\}$, where Y_{k+1} includes $V \setminus \{x\}$ and $|Y_1| = \cdots = |Y_k| = \left\lfloor \frac{n-v}{k} \right\rfloor$. Here, the intuition behind the construction of Y_{k+1} is that x cannot have any parent in V because all Y_i's for which $i \in [1, k]$ exclude V. Finally, let $O = \bigcup_{x \in V} O_x$.

Now, consider any labeling f of O. By Lemma 7, we know that for each O_x $(x \in V)$ there is a CP-net N_x consistent with f on O_x, where N_x is formed by k root nodes with an empty table pointing to the same child node with a complete table. Let $N = \bigcup_{x \in V} N_x$. By construction, the sets O_x $(x \in V)$ are mutually disjoint. Furthermore, we know that N is acyclic because for any variable x in V, there is no other variable in V which can occur as a parent node of x in N_x. It follows that N is consistent with f on O, and hence, O is shattered by $C^{n,k,e}_{\text{ACY}}$. Thus, using Theorem 5 to calculate the size of each O_x , we can derive the lower bound on the VC-dimension of $C^{n,k,e}_{\text{ACY}}$ by simply summing over all x's. □

We now have all ingredients in hand to assess the optimality of our algorithms. Based on Auer and Long's result [5, Theorem 2.2], we know that the query complexity of any concept class C with respect to any instance class $\mathcal{I}$ is at least $\log\left(\frac{4}{3}\right) \text{VCdim}(C, \mathcal{I})$, which is a constant factor of the VC dimension of C with respect to $\mathcal{I}$. Therefore, by applying Theorem 6, there is at least one acyclic CP-net N for which the number of queries needed to identify N is at least

$$\log\left(\frac{4}{3}\right)\left(r + e\left(\log \frac{n - e/k}{k} - 1\right) - e/k\right)$$

where k is the in-degree of the graph of N, e is the graph size (number of edges) of N, and $r = e\,2^k/k$ is the number of rules occurring in N. By comparing this lower bound to the number $r + e \log n + e + 1$ of queries used by Algorithm 1, it follows that our learning algorithm is optimal up to a term of order $e \log k$.

In the specific case where $k = 1$, there is at least one tree-structured CP-net N with e relevant variables such that the number of queries needed to identify N is at least $e \log(n - e) + e$. This is now to be compared with the maximum number $e \log n + 5e + 1$ of queries spent by Algorithm 2 in order to identify N.

In a nutshell, both our algorithms are quasi-optimal, even though they do not know in advance the degree k and the graph size e of the target network, and even if Algorithm 2 is able to use arbitrary counterexamples to equivalence queries.

7. Discussion

Along the lines of making query-directed learning applicable to preference elicitation, we have provided a model for learning preference networks from equivalence and membership

queries, together with significant learnability results. Taking into account the cognitive effort required by human users to answer queries, our model is distinguished by the close way in which it integrates learning and dominance testing, and the insistence on having convergence bounds that are polynomial in the minimal description size of the target concept, but only polylogarithmic in the total number of attributes. In essence, our results reveal that membership queries are essential for extracting both acyclic CP-nets from restricted outcome pairs, and tree-structured CP-nets from arbitrary outcome pairs. Importantly, the examples used by these queries can be limited to "swaps" in order to facilitate their comparison by the user.

Our results have interesting consequences in other learning models, including the probably approximately correct (PAC) learning model [32], and the mistake-bound learning model [27]. As follows from a generic conversion method from query-directed learning to PAC learning [1], acyclic CP-nets are attribute-efficiently PAC learnable with respect to swap examples, and tree-structured CP-nets are attribute-efficiently PAC learnable with respect to arbitrary examples, if in both cases membership queries are available. Furthermore, it is well-known that the model of mistake-bound learning and the model of learning with equivalence queries alone are essentially equivalent [27]. Therefore, one corollary of our results is that acyclic CP-nets are not learnable with a polynomial number of mistakes even if the instances to be predicted are restricted to swaps, and tree-structured CP-nets are not mistake-bound learnable if the instances to be predicted are arbitrary outcome pairs. On the other hand, if membership queries (excluding the elements to be predicted) are available, then acyclic CP-nets are attribute-efficiently mistake-bound learnable with respect to swaps, and tree CP-nets are attribute-efficiently mistake-bound learnable with respect to arbitrary instances.

7.1. Related Work

To the best of our knowledge, this work provides the first connection between *active* learning and graphical preference languages. Some authors, though, have recently focused on *passive* learning, where the goal is to extract a CP-net from a set of examples.

Notably, Lang and Mengin [26] consider the special case of *separable* CP-nets, that is, preference networks in which all variables have an empty set of parents. As a key result, they show that the problem of finding a separable CP-net that is consistent with some labeling of an arbitrary set of examples can be solved in time polynomial in the number n of input variables. Because the VC-dimension of separable CP-nets is also polynomial in the input dimension, it follows that separable CP-nets are PAC-learnable.

In a different direction, Dimopoulos *et al.* [16] investigate the learnability issue of acyclic and complete CP-nets under specific distributions. Recall that in the PAC learning model, the learner uses an example oracle $\textsc{ex}(N, \mathcal{D})$ as its source of labeled examples: when invoked, the oracle returns an example (o, o') chosen randomly according to an arbitrary distribution $\mathcal{D}$, together with the label 1 if $o \succ_N o'$ and the label 0 otherwise. In Dimopoulos *et al.*'s framework, the oracle uses a distribution $\mathcal{D}$ which guarantees that the example chosen at random is *transparently entailed* by the target network N (we refer the reader to the paper for this definition). Under this distribution specific PAC learning model, they demonstrate that acyclic CP-nets are identifiable with a time complexity of $\Theta(n^k)$, where k is the in-degree of N. In particular, since swap examples are always transparently entailed by acyclic and complete CP-nets, it follows that the class of acyclic and complete CP-nets with bounded in-degree k is PAC-learnable with respect to swap examples. This is to be compared with our results which state that, if membership queries are available, then acyclic and possibly incomplete CP-nets with arbitrary in-degree are attribute-efficiently PAC-learnable with respect to swap examples.

Perhaps the closest framework to ours is due to Sachdev [31], who investigates the general issue of learning preference logics [20] with equivalence and membership queries. Although encouraging, his results do not take into account the computational cost of dominance testing, and the query complexity grows exponentially with the size of rules.

7.2. Extensions and Open Problems

Clearly, there are many directions in which one might attempt extensions of this study. We shall concentrate on four of them.

Multiple values. Since we have restricted this study to binary-valued CP-nets, a natural direction of research is to explore the broader class of networks in which any variable x has an arbitrary finite domain D_x.

If we stick to the definition of multi-valued CP-nets suggested by Boutilier *et al.* [9], our learning algorithms can easily be extended to this setting. Here, any rule on some variable x is an expression of the form $t : v_1 \succ v_2 \succ \cdots \succ v_d$, where the body is an assignment of values to the parents of x in the network, and the head is a linear ordering of D_x. For acyclic CP-nets, we simply need to refine Line 9 of Algorithm 1, in which the head of a new rule is constructed, by the linear ordering with a dichotomic search procedure that uses $d \log d$ membership queries. For tree CP-nets, a simple approach is to replace, in Line 8 of Algorithm 2, the computation of $pref(o, p)$ for each $o \in \{\mathbf{0}, \mathbf{1}\}$ and $p \in \{x, \overline{x}\}$ by that of $pref(o, v_i, v_j)$ for each constant outcome[5] $o \in \{\boldsymbol{v_1}, \cdots, \boldsymbol{v_d}\}$ and for each pair of values v_i, v_j in D_x. The query complexity is increased here by a factor of d^3, but more sophisticated algorithms could be conceived to reduce it.

However, we should take into account the evidence that under certain circumstances some values in the domain are irrelevant, and some pairs of values are incomparable. For example, suppose that in the evening dress scenario, Susan has shirts of many different colors. As in Example 1, her preference for the color of the shirt depends on the combination of jacket and pants. Some colors are clearly too flashy for a ceremony, and Susan would never use them with a jacket. So they are irrelevant for the occasion. The remaining colors are not necessarily pairwise comparable: if the jacket and pants are of the same tint, Susan might prefer a blue shirt or green shirt to a white one in order to bring a touch of color, but she would not necessarily have a preference between blue and green.

In this setting, each entry of a CP-table for a relevant variable x would be a rule of the form $t : \{(v_i, v_j)\}$ where the condition is an assignment of values to the parents of x, and the head is a partial ordering on some subset of D_x. In addition to the requirement of attribute efficiency, we should here consider the requirement of *value efficiency*. As an interesting open issue, the problem is to determine whether there exist learning algorithms capable of identifying acyclic CP-nets with swap counterexamples and tree CP-nets with arbitrary counterexamples, and for which the query complexity is only polylogarithmic in the total number of variables *and* the size of the largest domain of these variables.

Indifference. Another direction of research that naturally emerges from our study is to include indifference rules of the form $t : x \sim \overline{x}$. Such a rule expresses the statement "given that t holds and all other things being equal, the values x and $\overline{x}$ are equally preferred". Importantly,

[5] We assume for simplicity of exposition that all variables have the same domain, but the refinement would be similar without this assumption.

indifference must be distinguished from incompleteness, since the latter would be interpreted as "x is not comparable to $\overline{x}$, given that t holds and all other things being equal". Besides the technical issue of extending the semantics of membership and equivalence queries in the presence of indifference, the learnability issue of acyclic CP-nets with indifference is far from being easy, because they are not always guaranteed to be satisfiable.

Compactness. As suggested by Goldsmith *et al.* [22], a natural way to reduce the size of CP-tables within a potentially exponential factor is to use a compact representation.

Specifically, if a table contains two rules of the form $t \wedge p_j : p_i \succ \overline{p}_i$ and $t \wedge \overline{p}_j : p_i \succ \overline{p}_i$, then they are compactly represented as the single rule $t : p_i \succ \overline{p}_i$, even if x_j is a parent of x_i in the network. In this setting, for any relevant variable x in the network, the CP-table of x with respect to $par(x)$ can be viewed as pair $(\varphi_x, \varphi_{\overline{x}})$ of DNF formulas over $par(x)$ which have no model in common. The interpretation of the CP-table is that $o[x] \succ o[\overline{x}]$ (resp. $o[\overline{x}] \succ o[x]$) holds if and only if $o \models \varphi_x$ (resp. $o \models \varphi_{\overline{x}}$).

When learning such a compact representation, *efficiency* is typically evaluated with respect to the size of the *smallest* compact representation of the target CP-net. Unfortunately, when considering possibly incomplete, binary-valued CP-nets, the learning problem turns out to be at least as hard as the problem of learning a DNF formula with the same queries, which is a long-standing open question in machine learning [15, 24, 29].

Theorem 7. The problem of learning a single and possibly incomplete CP-table from equivalence and membership queries over swaps is at least as hard as the problem of learning a DNF formula with equivalence and membership queries.

Proof. Given an algorithm A for learning a compact CP-table $(\varphi^*_x, \varphi^*_{\overline{x}})$ for some variable x, we derive an algorithm B for learning the DNF φ^* using the following transformation. Given an initial hypothesis $(\varphi_x, \varphi_{\overline{x}})$ of Algorithm A, the formula φ_x is used as the first equivalence query of Algorithm B. If B receives a positive counterexample $o \models \varphi_x$, then it forwards to A the positive counterexample $o[x] \succ o[\overline{x}]$. Dually, if B receives a negative counterexample $o \not\models \varphi_x$, then it forwards to A the negative counterexample $o[x] \not\succ o[\overline{x}]$. In both cases, Algorithm B takes the part φ_x of the new hypothesis $(\varphi_x, \varphi_{\overline{x}})$ of Algorithm A and uses it as the next equivalence query, continuing in this fashion until it receives the answer `Yes`. If during the update of $(\varphi_x, \varphi_{\overline{x}})$, Algorithm A must ask a membership query on $(o[x], o[\overline{x}])$, then B simply asks a membership query on $o[x]$ and forwards the answer to A. Since the number of queries and running time are preserved by the transformation, the result follows. □

Cyclicity. Finally, the literature on CP-nets has rapidly flourished in the last years by providing graphical preference languages of increasing expressiveness [7, 10, 12, 22, 30, 33]. In particular, Goldsmith *et al.* [22] argue that acyclic CP-nets are not sufficiently expressive to capture human preferences even in some simple domains. Returning to the evening dress scenario, suppose that Susan prefers white pants given a black jacket, and conversely, she prefers a black jacket given white pants. On the other hand, Susan prefers black pants given a white jacket, and conversely, she prefers a white jacket given black pants. The resulting CP-net defined over only two variables is consistent, and there is no acyclic CP-net giving rise to the same preferences on outcomes.

However, it is easy to see that in this generalized setting our lemmas 1 and 2 do not hold anymore. So, the learnability issue of cyclic CP-nets looks challenging, *even* in the presence of swap examples.

Acknowledgements

This work was supported by the French ANR grant CANAR (ANR-06-BLAN-0383-02).

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. Arias and R. Khardon. Learning closed Horn expressions. *Information and Computation*, 178(1):214–240, 2002.
[5] P. Auer and P. M. Long. Structural results about on-line learning models with and without queries. *Machine Learning*, 36(3):147–181, 1999.
[6] J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *Proceedings of the Twenty-first International Conference (ICML'04), Banff, Alberta, Canada*, pages 9–17. ACM, 2004.
[7] M. Binshtok, R. Brafman, C. Domshlak, and S. Shimony. Generic preferences over subsets of structured objects. *Journal of Artificial Intelligence Research*, 34:133–164, 2009.
[8] C. Boutilier, R. Brafman, H. Hoos, and D. Poole. Reasoning with conditional ceteris paribus preference statements. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI'99), Stockholm, Sweden*, pages 71–80. Morgan Kauffmann, 1999.
[9] 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.
[10] R. Brafman and C. Domshlak. Graphically structured value-function compilation. *Artificial Intelligence*, 172(2-3): 325–349, 2008.
[11] R. Brafman and C. Domshlak. Introducing variable importance tradeoffs into CP-nets. In *Proceedings of the Eighteenth Conference in Uncertainty in Artificial Intelligence (UAI'02), Alberta, Canada*, pages 69–76. Morgan Kaufmann, 2002.
[12] R. Brafman, C. Domshlak, and S. Shimony. On graphical modeling of preference and importance. *Journal of Artificial Intelligence Research*, 25:389–424, 2006.
[13] N. Bshouty. Exact learning Boolean functions via the monotone theory. *Information and Computation*, 123(1): 146–153, 1995.
[14] N. Bshouty and L. Hellerstein. Attribute-efficient learning in query and mistake-bound models. *Journal of Computer and System Sciences*, 56:310–319, 1998.
[15] N. Bshouty, S. Goldman, T. Hancock, and S. Matar. Asking questions to minimize errors. *Journal of Computer and System Sciences*, 52(2):268–286, 1996.
[16] Y. Dimopoulos, L. Michael, and F. Athienitou. Ceteris paribus preference elicitation with predictive guarantees. In Craig Boutilier, editor, *Proc. 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 1890–1895. IJCAI, 2009.
[17] C. Domshlak and R. Brafman. CP-nets: Reasoning and consistency testing. In *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR'02), Toulouse, France*, pages 121–132. Morgan Kaufmann, 2002.
[18] C. Domshlak, R. Brafman, and S. Shimony. Preference-based configuration of web page content. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01), Seattle, Washington*, pages 1451–1456. Morgan Kaufmann, 2001.
[19] C. Domshlak, S. Prestwich, F. Rossi, K. B. Venable, and T. Walsh. Hard and soft constraints for reasoning about qualitative conditional preferences. *Journal of Heuristics*, 12(4–5):263–285, 2006.
[20] J. Doyle, Y. Shoham, and M. Wellman. A logic of relative desire (preliminary report). In *Proceedings of the Sixth International Symposium on Methodologies for Intelligent Systems (ISMIS'91), Charlotte, North Carolina*, pages 16–31. Springer, 1991.
[21] M. Frazier and L. Pitt. Classic learning. *Machine Learning*, 25(2-3):151–193, 1996.
[22] J. Goldsmith, J. Lang, M. Truszczynski, and N. Wilson. The computational complexity of dominance and consistency in CP-nets. *Journal of Artificial Intelligence Research*, 33:403–432, 2008.
[23] P. Green and V. Srinivasan. Conjoint analysis in consumer research: Issues and outlook. *Journal of Consumer Research*, 5(2):103–123, 1978.
[24] L. Hellerstein and V. Raghavan. Exact learning of DNF formulas using DNF hypotheses. *Journal of Computer and System Sciences*, 70(4):435–470, 2005.
[25] R. Khardon. Learning function-free Horn expressions. *Machine Learning*, 37(3):241–275, 1999.

[26] J. Lang and J. Mengin. The complexity of learning separable ceteris paribus preferences. In Craig Boutilier, editor, *Proc. 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 848–853. IJCAI, 2009.

[27] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

[28] W. Maass and G. Turán. Lower bounds methods and separation results for online-learning models. *Machine Learning*, 9:107–145, 1992.

[29] K. Pillaipakkamnatt and V. Raghavan. On the limits of proper learnability of subclasses of DNF formulas. *Machine Learning*, 25(2-3):237–263, 1996.

[30] F. Rossi, K. Venable, and T. Walsh. mCP nets: Representing and reasoning with preferences of multiple agents. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence, San Jose, California*, pages 729–734. AAAI Press / The MIT Press, 2004.

[31] M. Sachdev. On learning of *ceteris paribus* preference theories. Master's thesis, Graduate Faculty of North Carolina State University, 2007.

[32] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[33] N. Wilson. Extending CP-nets with stronger conditional preference statements. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence, San Jose, California*, pages 735–741. AAAI Press / The MIT Press, 2004.

[34] C. Ziegler, G. Lausen, and J. Konstan. On exploiting classification taxonomies in recommender systems. *AI Communications*, 21(2-3):97–125, 2008.