*STAIRS 2014*
*U. Endriss and J. Leite (Eds.)*


*doi:10.3233/978-1-61499-421-3-81*

# Learning Probabilistic CP-nets from Observations of Optimal Items

Damien BIGOT [a], Jérôme MENGIN [a] and Bruno ZANUTTINI [b]
[a] *IRIT, Université Paul Sabatier, Toulouse, France*
[b] *GREYC, Université de Caen Basse-Normandie, France*

**Abstract.** Modelling preferences has been an active research topic in Artificial Intelligence for more than fifteen years. Existing formalisms are rich and flexible enough to capture the behaviour of complex decision rules. However, for being interesting in practice, it is interesting to learn not a single model, but a probabilistic model that can compactly represent the preferences of a group of users – this model can then be finely tuned to fit one particular user. Even in contexts where a user is not anonymous, her preferences can depend on the value of a non controllable state variable. In such contexts, we would like to be able to answer questions like "What is the probability that $o$ is preferred to $o'$ by some (unknown) agent?", or "Which item is most likely to be the preferred one, given some constraints?"

We study in this paper how Probabilistic Conditional Preference networks can be learnt, both in off-line and on-line settings.

**Keywords.** PCP-net, Learning, preference, recommandation

## 1. Introduction

The development of recommender systems and other interactive systems for supporting decision-making has highlighted the need for models capable of using a user's preferences to guide her choices. Modelling preferences has been an active research topic in Artificial Intelligence for more than fifteen years. In recent years, several formalisms have been proposed that are rich enough to describe in a compact way complex preferences of a user over combinatorial domains. When the user's preferences are qualitative, and have a "simple" structure, conditional preference networks (CP-nets, [5]) and their variants [4,6] are popular representation frameworks. In particular, CP-nets come with efficient algorithms for finding most preferred items (*item optimisation problem*).

Existing formalisms are rich and flexible enough to capture the behaviour of complex decision rules. However, for being interesting in practice, these formalisms must also permit fast elicitation of a user's preferences, involving a reasonable amount of interaction only. Anonymous recommendation systems, preference-based search [17], or configuration of combinatorial products in *business-to-customer* problems [15] are good examples of decision problems in which the user's preferences are not known *a priori*. In such applications, a single interaction with the user must typically last at most 0.25 s, and the whole session must typically last at most 20 minutes, even if the item to be recommended to the user is searched for in a combinatorial set.

Recently there have been several interesting proposals for learning preferences, many of them where presented in [11]. The approaches range from learning numerical ranking functions [12,18,1] to learning qualitative, structured preference rules [13,10,14,2]. These works assume that a set of rankings or pairwise comparisons is given or elicitated, in order to build a model that generalises these rankings or comparisons.

However, in several settings, it is interesting to learn not a single model, but a probabilistic model that can compactly represent the preferences of a group of users. In a next step, this model can then be finely tuned to fit one particular user. Even in contexts where a user is not anonymous, her preferences are usually ill-known, because they can depend on the value of a non controllable state variable. In such contexts, we would like to be able to answer questions like "What is the probability that $o$ is preferred to $o'$ by some (unknown) agent?", or "Which item is most likely to be the preferred one, given some constraints?"

*Probabilistic Conditional Preference networks* (or PCP-nets for short) [9,8] enable the user to compactly represent a probability distribution over some partial orderings and answer such queries. Specifcially, a PCP-net specifies a probability distribution over a family of CP-nets. There is a close connection between CP-nets and Bayesian networks: [7] proves that the problem of finding the most probable optimal item is similar to an optimisation problem in a Bayesian network. However, a PCP-net encodes a probability distribution over partial orders, not just on a list of items.

We study in this paper how PCP-nets can be learnt, both in off-line and on-line settings. Appart from the probabilistic approach, one difference with the works mentioned above is that we do not assume that we have, or elicitate, a set of rankings or pairwise comparisons. Instead, we suppose that we have a list of items which, it is assumed, are or have been optimal for some user or in some context. Such a list can be, for instance, a list of items that have been sold. We prove that such information is sufficient to learn a partial order over the set of possible items, when these have a combinatorial structure.

The elicitation of probabilistic CP-nets is discussed by [16]. However, the authors did not give a precise semantics to their CP-nets.

The next section sums up the main properties of CP-nets and PCP-nets. We then describe how it is possible to learn PCP-nets, off-line and on-line. Finally, we show the results of some experiments that simulate an on-line learning setting.

## 2. Background on probabilistic CP-nets

We consider combinatorial objects defined over a set of $n$ variables $\mathscr{V}$. Variables are denoted by uppercase letters $A,B,X,Y,\ldots$. In this paper, we suppose that variables are Boolean; we consistently write $x$ and $\bar{x}$ for the two values in the domain $\underline{X}$ of $X$.

For a set of variables $U \subseteq \mathscr{V}$ , $\underline{U}$ denotes the Cartesian product of their domains. Elements of $\underline{\mathscr{V}}$ are called *items*, denoted by $o,o',\ldots$. Elements of $\underline{U}$ for some $U \subseteq \mathscr{V}$ are denoted by $u,u',\ldots$. Given two sets of variables $U,V \subseteq \mathscr{V}$ and $v \in \underline{V}$, we write $v[U]$ for the restriction of $v$ to the variables in $U \cap V$.

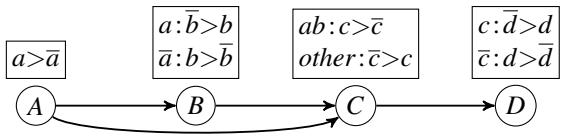
Preferences are encoded in CP-nets using rules of the form $X,u:>$, where $X \in \mathscr{V}$, $u$ is an instantiation of variables in a set $U \subseteq \mathscr{V}$ that does not contain $X$, and $>$ is a total strict order over the domain of $X$: either $x>\bar{x}$ or $\bar{x}>x$. Informally, the rule $X,u:x>\bar{x}$ can

be read:"Whenever $u$ is the case, then $x$ is preferred to $\overline{x}$, *ceteris paribus* (all other things being equal)."

A rule $X,u:>$, with $u \in \underline{U}$, indicates that the preference over the values of $X$ depends on the values of the variables in $U$. Associated to every CP-net is a directed graph $G$ over $\mathscr{V}$: there is an edge $(Y,X)$ whenever the preference over the values of $X$ depends on the values of $Y$; $G$ is called the *structure* of the CP-net. We write $\mathsf{pa}(X)$ for the set of variables on which the order over $\underline{X}$ depends, called the parents of $X$: $\mathsf{pa}(X) = \{Y \in \mathscr{V} \mid (Y,X) \in G\}$. It is generally assumed that a CP net contains a rule $(X,u:>)$ for every $X \in \mathscr{V}$ and every $u \in \mathsf{pa}(X)$; the set of the rules that order the domain of $X$ is called the *conditional preference table*, or CPT, for $X$. When $X$ is clear from the context, we write $u:>$ instead of $(X,u:>)$ A CP-net specifies a partial ordering $\succ$ over the set of items: $\succ$ is the transitive closure of the set of the pairs of items $(o,o')$ such that there is a rule $(X,u:>)$ with $o[X] > o'[X]$ and $o[U] = o'[U]$ and $o[Y] = o'[Y]$ for every $Y \notin (U \cup \{X\})$. When needed, we will distinguish the partial ordering associated with a particular CP-net $N$ using a subscript: $\succ_N$.

CP-nets are most interesting when there is no cyclic preferential dependency between the variables, that is, when the graph $G$ does not contain any (directed) cycle. In this case, [5] proved that the relation $\succ$ is a (partial) strict order.

**Example 1** *An example of an acyclic CP-net over 4 binary variables $A,B,C,D$ is:*




*The rule $ab:c>\overline{c}$ implies that $abcd \succ ab\overline{c}d$. We also have that $ab\overline{c}d \succ ab\overline{c}\overline{d}$ because of the rule $\overline{c}:d>\overline{d}$, thus, by transitivity, $abcd \succ ab\overline{c}\overline{d}$.*

*Optimization* can be done in time linear in the size of an acyclic CP-net: choose an ordering $X_1,\ldots,X_n$ of the variables in $\mathscr{V}$ that is compatible with the dependency graph (if $X_i \in \mathsf{pa}(X_j)$ then $i < j$), and assign in turn to every $X_j$ its most preferred value, given the values that have already been chosen for its parents. The resulting item is the unique optimal (undominated) item of the ordering specified by the CP-net. For instance, the order represented by the CP-net above has exactly one optimal item, which is $a\overline{b}\overline{c}d$ ( ie there is no object which is preferred to $a\overline{b}\overline{c}d$).

The *forward sweep* procedure above can also be used to find an item that is optimal among those that satisfy a given conjunction of constraints of the form $Y_k = y_k$ – one only has to find, for each $X_i$, the optimal admissible value given the value of its parents. For instance, the most optimal item, among those that have $b$ as value for $B$, is $ab\overline{c}d$. Conversely, if we know the structure of a CP-net and the optimal item $o$, then we can immediately induce some of the rules of the CP-net: for every $X \in \mathscr{V}$, the CP-net contains the rule $(X,o[\mathsf{pa}(X)] : o[X] > \overline{o[X]})$.

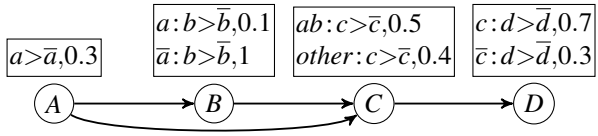
Note that the *dominance* problem, that is deciding, given a CP-net and two items $o$ and $o'$, if $o \succ o'$, is an NP-hard problem, even for acyclic CP-nets. Yet, a weaker *ordering query* can be answered in time linear in the number of variables as follows: given an acyclic CP-net and two items $o$ and $o'$, choose again an ordering $X_1,\ldots,X_n$ that is compatible with $G$, and consider the variables one after the other as long as $o[X_i] = o'[X_i]$; let

now $i$ be the first $i$ such that $o[X_i] \neq o'[X_i]$: if $o[X_i] > o'[X_i]$ (resp. $o[X_i] < o'[X_i]$) given the values of the parents of $X_i$ in $o$ and $o'$, then $o' \not\succ o$ (resp. $o \not\succ o'$).

### 2.1. Probabilistic CP-nets

Uncertainty about the ordering over the items can be represented by associating probabilities to a given preference structure $G$ [9,8]: for each pair $(X,u)$, with $X \in \mathcal{V}$ and $u \in \mathsf{pa}(U)$ a probability distribution is defined over the set of possible orderings over $\underline{X}$; in the case of boolean variables, this distribution is entirely defined by the probability that the ordering is $x > \overline{x}$. In the sequel, we write $p(X,u : x > \overline{x})$ for this probability. A *probabilistic CP-net*, or PCP-net for short, is entirely defined by its structure $G$ and the probabilities $p(X,u : x > \overline{x})$.

**Example 2** *A probabilistic CP-net with the same structure as the CP-net of Example 1:*




Suppose that a PCP-net is given, with structure $G$. Assuming that the orderings over the domains of the variables are probabilistically independent from one another, the probability that a given CP-net $N$, with the same structure $G$, occurs, can be defined as:

$$P(N) = \prod_{(X,u)} p(X,u : >^N_{X,u})$$

where the product is taken over all variables $X \in \mathcal{V}$ and assignments $u \in \mathsf{pa}(X)$, and where $>^N_{X,u}$ denotes the ordering over $\underline{X}$ that occurs in $N$ when $u$ is the case.

So, a PCP-net $\mathcal{N}$ is not intended to represent a preference relation. Rather, it represents a probability distribution over a set of CP-nets, namely, those which have the same structure as $\mathcal{N}$: we say that they are *compatible* with the PCP-net. We will write $N \propto \mathcal{N}$ to indicate that the CP-net $N$ has the same structure as $\mathcal{N}$.

Note that, in a PCP-net, the probabilities of the rules are independent from one another. So, for instance, it would not be possible to represent with a PCP-net a probability distribution over CP-nets with the structure of Example 2 if the rules over $B$ and $C$ were dependent; if the preferred values for $B$ and $C$ were always $b$ and $c$ together, or $\overline{b}$ and $\overline{c}$. An important topic for further research is to generalize the approach to allow for such dependencies.

Given a PCP-net $\mathcal{N}$, which represents a probability distribution on a set of deterministic CP-nets, reasoning tasks consist in computing probabilities associated with interesting events.

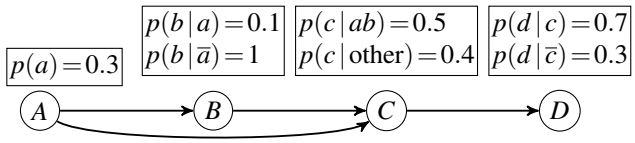
#### 2.1.1. Probabilistic optimization

Let, for any item $o$, "opt$=o$" denote the set of compatible CP-nets that have $o$ as unique optimal item. Then $P(\text{opt}=o)$ is, by definition, the sum of the probabilities $P(N)$ of the CP-nets $N$ that are compatible with $\mathcal{N}$ and such that $o$ is optimal for $N$.

Interestingly, considering acyclic CP-nets, we mentioned earlier that an item $o$ is optimal in $N$ if and only if for every variable $X$, $N$ contains the rule $o[\mathsf{pa}(X)]:o[X]>\overline{o[X]}$, therefore

$$P(\text{opt}=o)=\prod_{X\in\mathscr{V}} p(X,o[\mathsf{pa}(X)]:o[X]>\overline{o[X]}).$$

This formula indicates that the probabilities of optimality can be encoded in a Bayesian Network associated to $\mathscr{N}$ [8]: let $BN(\mathscr{N})$ denote this network, it structure is the same oriented graph as that of $\mathscr{N}$, and, for every binary variable $X$ and assignment $u\in\mathsf{pa}(X)$, the conditional probability table for $X$ contains $p(x\,|\,u)=p(X,u:x>\bar{x})$. (For non binary variables, $p(x\,|\,u)$ would be the sum of the probabilities of the local orderings that have $x$ at the top.) For instance, the probabilities of optimality of the PCP-net of Example 2 are encoded in the following Bayesian network:




In particular, computing the item that has the highest probability of being optimal is a #P-hard problem; If G is a tree, this item can be computed in linear time by using a bottom-up procedure [9].

Also, we can express the probability that a given value $x$ for one variable $X\in\mathscr{V}$ appears in the optimal item as follows:

$$P(\text{opt}[X]=x)=\sum_{u\in\underline{U}} p(X,u:x>\bar{x})\times P(\text{opt}[U]=u)$$

where $\text{opt}[U]=u$ denotes the event that the optimal item has values $u$ for the variables in $U$. More generally, the probability that a partial assignment is optimal in terms of the probabilities of the rules of the PCP-net is:

$$P(\text{opt}[U]=u)=\sum_{\substack{a\in\underline{\mathsf{asc}(U)}\\ a[U]=u}}\prod_{Y\in\mathsf{asc}(U)} p(Y,a[\mathsf{pa}(Y)]:a[Y]>\overline{a[Y]})$$

where $\mathsf{asc}(U)$ denotes the set of ascendants of the variables in $U$, including $U$. (More precisely, $\mathsf{asc}(U)$ is the smallest set that contains $U$ and all the parents of each of its elements.) This equation does not give a practical mean of computing $P(\text{opt}[U]=u)$ unless the number of parents and the height of the PCP-nets are bounded, since the size of $\underline{\mathsf{asc}(U)}$ is exponential in the size of $\mathsf{asc}(U)$.

### *2.1.2. Probability of dominance*

Let, for any two items $o,o'$, "$o\succ o'$" denote the set of compatible CP-nets $N$ such that $o\succ_N o'$. Then $P(o\succ o')$ is, by definition, the sum of the probabilities $P(N)$ of the CP-nets $N$ that are compatible with $\mathscr{N}$ and such that $o\succ_N o'$. [9] show that computing this probability is #P-complete, even when considering acyclic PCP-nets or polytrees.

## 3. Learning a PCP-net from probabilities of optimality

In many settings, like a recommender system, the system can record a list of items that can be assumed to have been optimal for some user at some point. It can be, for instance, a list of sold / rented items. Let $\mathscr{L}$ denote this list. Assuming that the users' preferences correspond to some PCP-net $\mathscr{N}$, the frequencies of the items in this list correspond to the probabilities of optimality of items in the corresponding Bayesian network. This suggests that this PCP-net can be induced from this list of sold items.

### 3.1. Off-line learning

*Learning the parameters* Let us assume that we know the structure of the hidden PCP-net, and that we want to estimate the probabilities in the tables. When the variables are binary, these probabilities are exactly the probabilities that appear in the tables of the Bayesian network that encodes the probabilities of optimality.

In particular, observing the probabilities of optimality can be sufficient to estimate the probabilities of the rules: for any binary variable $X \in \mathscr{V}$, for every $u \in \underline{\mathsf{pa}(X)}$, we have:

$$p(X,u:x>\bar{x}) = P(\mathrm{opt}[X]=x \,|\, \mathrm{opt}[U]=u) \sim |\{o \in \mathscr{L}, o[UX]=ux\}| / |\{o \in \mathscr{L}, o[U]=u\}|$$

when $\{o \in \mathscr{L}, o[U]=u\}$ is not empty, that is, when $P(\mathrm{opt}[U]=u) \neq 0$ in the hidden PCP-net.

More generally, we can use methods that have been used for Bayesian networks to learn these probabilities.

If $P(\mathrm{opt}[U]=u)=0$, we may still be able to estimate $p(X,u:x>\bar{x})$ from the probabilities of sub-optimal items, if we have a list of items that have been chosen by some users under some constraint, for instance a list of items sold during a period of a time where some options were not available. Assuming that the preferences of the user remain the same, the only effect of such constraint is to restrict the domain of some variables, but does not change the probabilities of other variables for the remaining combinations of values of the parents. Let $\mathscr{L}_{V=v}$ be a list of items optimal under the constraint $V=v$ for some $V \subseteq \mathsf{asc}(U)$ and $v \in \underline{V}$ such that $v[U]=u[V]$, and let "$\mathrm{opt}_{V=v}[U]=u$" denote the event that the item that is optimal among those that have values $v$ for the variables in $V$, has values $u$ for the variables in $U$, then

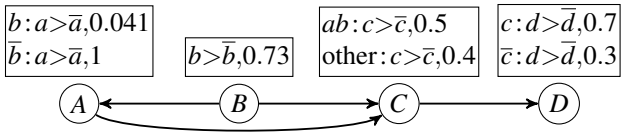
$$\begin{aligned} p(X,u:x>\bar{x}) &= P(\mathrm{opt}_{V=v}[X]=x \,|\, \mathrm{opt}_{V=v}[U]=u) \\ &\sim |\{o \in \mathscr{L}_{V=v}, o[UX]=ux\}| / |\{o \in \mathscr{L}_{V=v}, o[U]=u\}| \end{aligned}$$

For instance, consider a PCP-net that has the same structure as the PCP-net of Example 2. The probability of the rule $D,c:d>\bar{d}$ can be estimated as follows:

1. if $P(\mathrm{opt}[C]=c) \neq 0$:
   $p(D,c:d>\bar{d}) \sim |\{o \in \mathscr{L}, o[CD]=cd\}| / |\{o \in \mathscr{L}, o[C]=c\}|$;
2. if $P(\mathrm{opt}[C]=c)=0$, which is the case for instance if $p(a>\bar{a})=p(b>\bar{b})=1$ and $p(ab:c>\bar{c})=0$:
   $p(D,c:d>\bar{d}) \sim |\{o \in \mathscr{L}_{C=c}, o[D]=d\}| / |\mathscr{L}_{C=c}|$;
3. or, still if $P(\mathrm{opt}[C]=c)=0$, but $p(\bar{a}b:c>\bar{c}) \neq 0$:
   $p(D,c:d>\bar{d}) \sim |\{o \in \mathscr{L}_{A=\bar{a}}, o[CD]=cd\}| / |\{o \in \mathscr{L}_{A=\bar{a}}, o[C]=c\}|$;

Equation 2 and 3 above give two different ways of computing $p(D,c:d>\overline{d})$ when the probability of having $C=c$ in a optimal item is zero, corresponding to two different observations: 2. corresponds to the observation of optimal items when $C=c$ is forced, and 3. to the observation of optimal items when $A=\overline{a}$ is forced.

*Learning the structure* Here again, methods used to learn Bayesian networks can be used. A major hurdle, however, is that several PCP-nets with different structures may give rise to the same probabilities of optimality: this is linked to the fact that the preferential dependencies encoded in PCP-net are oriented, whereas, in a Bayesian network, probabilistic dependencies are symetric. Consider for instance the Bayesian network that encodes the probabilities of optimality for the PCP-net of Example 2: it also encodes the probabilities of optimality of the PCP below, obtained from that of Example 2 by reversing the edge between $A$ and $B$.




Therefore, methods for learning Bayesian networks will not completely identify a hidden PCP-net. However, quite a lot of information is obtained in this way. If a topological ordering of the otherwise unknown PCP-net is known, then the correct direction of each edge can be inferred, and the parameters of the hidden PCP net can be computed from the Bayesian network. Observe that there are natural situations in which such a general, total order might be known in advance. In particular, it is the case of interactive configuration, if the order of the variables to be configured is fixed (the system asks to choose a value for $X_1$, then one for $X_2$, etc.), then one can assume that the preferences will be expressed wrt this order, or at least, it makes sense to approximate these preferences on this order. Otherwise, the correct direction of the edges can be elicited, as described in the next section.

### 3.2. On-line learning

*Structure elicitation* Assuming that a Bayesian network encoding the probabilities of optimality has been computed, in an active learning setting some queries can lead to a quick identification of the correct orientation of the edges of the PCP-net. Assume for instance that the Bayesian network has an edge between variables $X$ and $Y$: either the preferences over the domain of $X$ depend on the value of $Y$, or the vise versa (but not both, since we assume an acyclic PCP net). In order to determine the orientation of the edge, one can submit to users the queries $x:y?\overline{y}$ and $\overline{x}:y?\overline{y}$, if the frequencies of the two possible answers to these queries converge to a common value over time, then $y$ is preferentially independent of $X$. Otherwise, the preferences over the values of $Y$ depend on the value of $X$, and $X$ must be preferentially independent of $Y$.

*Parameters update* Finally, assume that a system has a current PCP net (maybe learnt from a list of past optimal items), and can now observe some users and their optimal items: it may be sensible to update the parameters (probabilities) of the PCP net according to what is being observed.

For instance, if the current PCP net corresponds to a group of past users, and a new user connects to the system, her preferences may not be exactly that of the group, and we may want to modify the parameters of the PCP net so that it incorporates the preferences of this particular user. Or it may be the case that the PCP net represents probabilities of preferences at a given time, or in a given location, and that these probabilities need to be updated in a new context.

Since the probabilities that we want to update directly correspond to the probabilities of some items being optimal, the observations made by the system must be about optimality. Every time our system is presented an optimal item $o$, it will update its parameters, that is, the probabilities of its current PCP-net $\mathscr{N}$ as follows:

For every observed optimal item $o$ do: for every $X \in \mathscr{V}$ do:

1. let $u = o[\mathsf{pa}(U)]$;
2. let $X_o = 1$ if $o[X] = x$, 0 otherwise;
3. $p(X,u:x>\bar{x})$ += $\eta_t (X_o - p(X,u:x>\bar{x}))$.

Note that we only update the probabilities of some of the rules (step 1.): the rules that make the given item optimal. The update rule is common in such a stochastic learning setting (see e.g. [3]). The parameter $\eta_t$ is the *learning rate*, it may vary over time, generally decrease in order to ensure convergence: convergence is guaranteed if $\sum_t \eta_t = \infty$ and $\sum_t \eta_t^2 < \infty$. In our experiments we took $\eta_t = 1/k(t,X,u)$ where $k(t,X,u)$ is the number of times the rule corresponding to $(X,u)$ has been updated so far (making the learning rate a function of the pair $(X,u)$). In this case, at any time, $p(X,u:x>\bar{x})$ is just the frequency with which optimal items $o$ with $o[U] = u$ and $o[X] = x$ have been encountered so far.

## 4. Experiments

We ran some experiments to evaluate the update rule that we proposed for an on-line learning setting. We assume a given acyclic structure over a given set of $n$ variables. We have some hidden PCP-net $\mathscr{N}^*$ with this structure, and start from an initial PCP-net with the same structure, where the probabilities of all rules are $1/2$. At each step, we update some rules of the current PCP-net $\mathscr{N}$.

We observed how the distance between the target PCP-net $\mathscr{N}^*$ and the current one $\mathscr{N}$ evolved. As measures of distances we used:

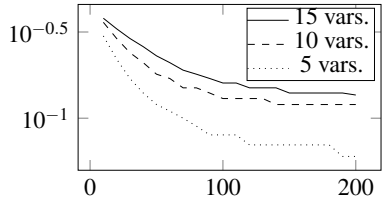
- the sum of the squared differences of the parameters

$$d_2(\mathscr{N},\mathscr{N}^*) = \sum_{(X,u)} \left( p_{\mathscr{N}}(X,u:x>\bar{x}) - p_{\mathscr{N}^*}(X,u:x>\bar{x}) \right)^2;$$

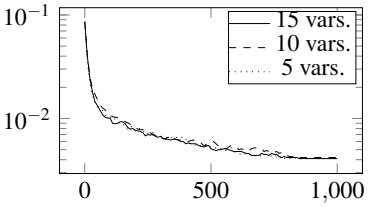
- the Kullback-Leibler divergence, or relative entropy, of the two probability distributions of optimality defined by $\mathscr{N}$ and $\mathscr{N}^*$:

$$d_{KL}(\mathscr{N} \| \mathscr{N}^*) = \sum_{o \in \underline{\mathscr{V}}} \log\left( P_{\mathscr{N}}(\mathrm{opt}=o) / P_{\mathscr{N}^*}(\mathrm{opt}=o) \right) \times P_{\mathscr{N}}(\mathrm{opt}=o).$$

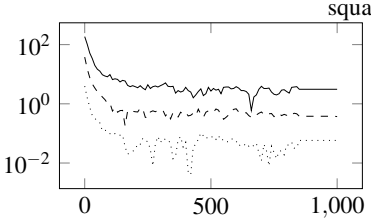
In fact, we computed an estimate of this distance by sampling the set of items, assuming a uniform distribution.




First protocol:Sum of squared diff. of param.




Second protocol:Sum of squared diff. of param.




Second protocol – KL divergence

**Figure 1.** Plots showing the evolution, as the number of observations of optimal outcomes grows ($x$-axis), of a measure of the distance between the target PCP-net and the learnt one ($y$-axis)

We have experimented with two protocols to generate optimal items at each time step.

*First protocol* The idea is to simulate an interactive setting, in which, for some pair of items $o_1$ and $o_2$, we observe for a while which is most frequently optimal, and update our current PCP-net if it does not give the same result. More precisely, for each new period we generate two items as follows: we generate two CP-nets $N_1$ and $N_2$ according to the distribution defined by our current hypothesis $\mathscr{N}$ (this is achieved by choosing, for each combination $(X,u)$, the rule $X,u:x>\bar{x}$) according to the current probability $p(X,u:x>\bar{x})$; let $o_1$ and $o_2$ be the respective optimal items of $N_1$ and $N_2$. Generating the two items in this manner will favor rules that are more probable so far. We can compute the probabilities that $o_1$ and $o_2$ are optimal according to our current PCP-net $\mathscr{N}$: let $p_i = P_{\mathscr{N}}(\text{opt} = o_i)$. We then "observe" which of $o_1$ and $o_2$ is most frequently optimal according to the hidden PCP-net $\mathscr{N}^*$: in fact, we compute $p_i^* = P_{\mathscr{N}^*}(\text{opt} = o_i)$. Eventually, if $p_1 > p_2$ whereas $p_2^* > p_1^*$, we update $\mathscr{N}$ so as to increase the probability that $o_2$ is optimal: we use the update algorithm above with $o = o_2$.

*Second protocol* Here we just simulate the update of our PCP-net after each newly observed chosen item, assuming that it is optimal for some user: we increase the probabilities of the rules that make this item optimal. Therefore, at each time step, we generate a "user" / PCP-net $N$ according to the target distribution represented by the target PCP-net $\mathscr{N}^*$, compute its optimal item $o$, and run the update algorithm with $o$.

*Results* We have run 500 trials with random target PCP-nets with 5, 10 and 15 variables. For each trial, we generated a target PCP-net, ran our experimental protocol and learning algorithm, and measured the distance between the learnt PCP-net and the target one, every 10 observations of an optimal item. The plots in Figure 1 depict the evolution of this distance: each point is an average over the 500 trials for each number of variables.

As can be noticed, a good approximation of the target PCP-net is reached after around 70 observations.

## 5. Conclusion

We have described in this paper how it is possible to learn a probabilistic representation of the preferences of a group of users over a combinatorial domain, or how we can fine-tune such preferences to fit more precisely one particular user. Since CP-nets in general are good at finding most preferred items, our learning method supposes that the learner can be supplied with a list of past most preferred items: we showed how a probabilistic CP-net can be learnt from such information.

## References

[1] D. Bigot, H. Fargier, J. Mengin, and B. Zanuttini. Using and learning gai-decompositions for representing ordinal rankings. In *Proc. ECAI workshop on Preference Learning*, pages 5–10, 2012.

[2] R. Booth, Y. Chevaleyre, J. Lang, J. Mengin, and C. Sombattheera. Learning conditionally lexicographic preference relations. In *Proc. ECAI 2010*.

[3] L. Bottou. Stochastic learning. In O. Bousquet, U. von Luxburg, and G. Rätsch, editors, *Advanced Lectures on Machine Learning*, LNCS 3176, pages 146–168. Springer, 2004.

[4] C. Boutilier, F. Bacchus, and R. I. Brafman. UCP-networks: A directed graphical representation of conditional utilities. In *Proc. UAI 2001*, pages 56–64. Morgan Kaufmann.

[5] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artificial Intelligence Research*, 21:135–191, 2004.

[6] R. I. Brafman and C. Domshlak. Introducing variable importance tradeoffs into CP-nets. In *Proc UAI 2002*, pages 69–76.

[7] C. Cornelio. Dynamic and probabilistic cp-nets. Master's thesis, University of Padua, 2012.

[8] C. Cornelio, J. Goldsmith, N. Mattei, F. Rossi, and K. B. Venable. Updates and uncertainty in CP-nets. In *Proc. Australasian Joint Conf. on Advances in Art. Intell. 2013*, LNCS 8272 , pages 301–312. Springer.

[9] D. D. Bigot, H. Fargier, J. Mengin, and B. Zanuttini. Probabilistic conditional preference networks. In A. Nicholson and P. Smyth, editors, *Proc. UAI 2013*.

[10] Y. Dimopoulos, L. Michael, and F. Athienitou. Ceteris paribus preference elicitation with predictive guarantees. In *Proc. IJCAI 2009*.

[11] J. Fürnkranz and H. Hüllermeier, editors. *Preference learning*. Springer, 2011.

[12] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. KDD 2002*, pages 133–142..

[13] F. Koriche and B. Zanuttini. Learning conditional preference networks with queries. In *Proc. IJCAI 2009*.

[14] J. Lang and J. Mengin. The complexity of learning separable ceteris paribus preferences. In *Proc. IJCAI 2009*.

[15] D. Mailharro. A classification and constraint-based framework for configuration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 12(4):383–397, 1998.

[16] S. S. de Amo, M. Bueno, G. Alves, and N. Silva. CPrefMiner: An algorithm for mining user contextual preferences based on bayesian networks. In *Proc. ICTAI 2012*, vol. 1, pages 114–121.

[17] P. Viappiani, B. Faltings, and P. Pu. Preference-based search using example-critiquing with suggestions. *J. Artificial Intelligence Research*, 27:465–503, 2006.

[18] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In W. Kraaij, A. P. de Vries, C. L. A. Clarke, N. Fuhr, and N. Kando, editors, *Proc. 30th ACM SIGIR Conf. on Information Retrieval 2007*, pages 391–398.