Online learning of acyclic conditional preference networks from noisy data

Fabien Labernia, Bruno Zanuttini, Brice Mayag, Florian Yger, Jamal Atif

Online learning of acyclic conditional preference networks from noisy data

Fabien Labernia*, Bruno Zanuttini†, Brice Mayag*, Florian Yger* and Jamal Atif*
*Paris-Dauphine, PSL Research University, CNRS, UMR 7243, LAMSADE
75016 PARIS, FRANCE
Email: {fabien.labernia,brice.mayag,florian.yger,jamal.atif}@lamsade.dauphine.fr
†GREYC, UMR 6072, UNICAEN/CNRS/ENSICAEN
14000 CAEN, FRANCE
Email: bruno.zanuttini@unicaen.fr

***Abstract*—We deal with online learning of acyclic Conditional Preference networks (CP-nets) from data streams, possibly corrupted with noise. We introduce a new, efficient algorithm relying on (i) information-theoretic measures defined over the induced preference rules, which allow us to deal with corrupted data in a principled way, and on (ii) the Hoeffding bound to define an asymptotically optimal decision criterion for selecting the best conditioned variable to update the learned network. This is the first algorithm dealing with online learning of CP-nets in the presence of noise. We provide a thorough theoretical analysis of the algorithm, and demonstrate its effectiveness through an empirical evaluation on synthetic and on real datasets.**

I. Introduction

Preference representation, reasoning (e.g. [19]–[21]), and learning (e.g. [4], [10], [13]), are receiving increasing attention in the literature, in particular within the context of recommender systems. In this paper, we address the problem of learning **combinatorial preferences**, and more precisely, learning acyclic **conditional preference networks** (CP-nets) [3].

CP-nets constitute a preference representation language based on **ceteris paribus** ("all other things being equal") comparisons, that is, comparisons of objects which differ over only one attribute. Such comparisons are arguably cognitively simpler than general ones, and CP-nets leverage this principle for factorizing the preferences over each attribute (or **variable**), leading to compact graphical representations.

Learning CP-nets, even by adding acyclicity constraints, is an NP-Complete problem [4]. Some works try to solve this problem by considering consistent preferences[1], e.g. regression-based learning [8], [9], [14], learning by reduction to 2-SAT [6], [18] and learning using user queries [4], [11], [13]. Recently, [15], [16] considered the problem of learning CP-nets from noisy data. We call **noise** (also called **incoherence** or **inconsistency** in other papers) an observed preference opposite to the real one, that is, observing "the user prefers $\mathbf{a}$ to $\mathbf{b}$" whereas her true preference is "I prefer $\mathbf{b}$ to $\mathbf{a}$". Noise can be caused by many factors: distracted user, corrupted data, unobserved variables, etc.

[1]i.e. the opposite preference is never observed (in the binary case).

In addition to noise, we consider an **online setting**, where observations (data) arrive as a stream and cannot be memoized, as in recommender systems where they could be generated by user clicks. In that sense, our approach differs from most of the literature: we consider the target CP-net to reflect the global preferences of a set of users, with variations among different users' preferences seen as noise w.r.t. this global relation.

Noise and onlineness of course make the problem more complex, and call for an algorithm that can maintain a best hypothesis about the target CP-net, given the observations received so far. The only online algorithm we know for solving this problem is a query-based algorithm [11]. We give experimental results comparing our approach to this one.

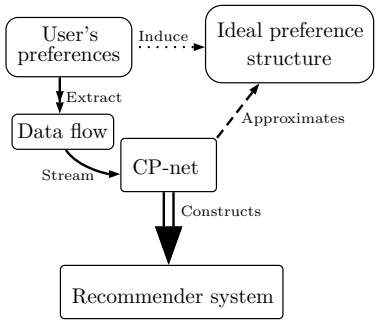

Fig. 1. Global scheme of a learning procedure for CP-nets.

To summarize, we propose a new, efficient, and robust to noise **online** algorithm for learning acyclic CP-nets on binary variables, as illustrated in Fig. 1. Our aim is to come up with a learning procedure that can be used in a recommender system. The algorithm observes a stream of pairwise preferences between two *ceteris paribus* objects, and learns an acyclic CP-net which maximizes the agreement with the observations. We support our claim by theoretical guarantees and experimental results on synthetic and real data.

II. Conditional preference networks (CP-nets)

Let $\mathbf{V}$ be a set of binary **variables**, with each $V \in \mathbf{V}$ associated with a **domain** $Dom(V) = \{v, v'\}$. For $\mathbf{V}' \subseteq \mathbf{V}$, we write $Dom(\mathbf{V}') = \prod_{V \in \mathbf{V}'} Dom(V)$, and call **state** an

element $\mathbf{v}'$ of $Dom(\mathbf{V}')$. Complete states, that is, elements $\mathbf{o}$ of $Dom(\mathbf{V})$, are called **outcomes**, and constitute the objects over which preferences are expressed.

For states $\mathbf{v}' \in Dom(\mathbf{V}'), \mathbf{v}'' \in Dom(\mathbf{V}'')$ with $\mathbf{V}' \cap \mathbf{V}'' = \emptyset$, we write $\mathbf{v}'\mathbf{v}'' \in Dom(\mathbf{V}' \cup \mathbf{V}'')$ for their concatenation. We also use the notation $v'\mathbf{v}''$ for the extension of state $\mathbf{v}'' \in Dom(\mathbf{V}'')$ by the value $v' \in Dom(V')$ (where $V' \in \mathbf{V}$, and $\mathbf{V}'' \subseteq \mathbf{V} \setminus \{V'\}$). Finally, $\mathbf{o}[\mathbf{V}']$ denotes the projection of the outcome $\mathbf{o}$ onto the variables in $\mathbf{V}'$.

A **(strict) preference relation** is a partial order $\succ$ on $Dom(\mathbf{V})$. $\mathbf{o} \succ \mathbf{o}'$ should be read "outcome $\mathbf{o}$ is **strictly preferred** to outcome $\mathbf{o}'$". We call **ceteris paribus pair** (CP-pair), denoted by $(\mathbf{o}, \mathbf{o}')_V$, a pair of outcomes which differ only over V (in formulas, $\mathbf{o}[\mathbf{V} \setminus \{V\}] = \mathbf{o}'[\mathbf{V} \setminus \{V\}]$).

Let $\succ$ be a strict preference relation on $Dom(\mathbf{V})$ and $X, Y \in \mathbf{V}$. Variable X is said to be a **parent** of variable Y if there is a context in which X alone changes the preference over the values of Y; precisely, if there is a state $\mathbf{z} \in Dom(\mathbf{V} \setminus \{X, Y\})$ satisfying $\mathbf{z}xy \succ \mathbf{z}xy'$ and $\mathbf{z}x'y' \succ \mathbf{z}x'y$ (or $\mathbf{z}xy' \succ \mathbf{z}xy$ and $\mathbf{z}x'y \succ \mathbf{z}x'y'$).

CP-nets are built on this notion. A **CP-rule** for variable V with domain $Dom(V) = \{x, x'\}$ is a rule of the form $r = (\mathbf{u} : v \succ v')$, or $\bar{r} = (\mathbf{u} : v' \succ v)$, with $\mathbf{u} \in Dom(\mathbf{U})$ for some $\mathbf{U} \subseteq \mathbf{V}$. The semantics of r is that $\mathbf{o} \succ \mathbf{o}'$ must hold for all CP pairs $(\mathbf{o}, \mathbf{o}')_V$ such that $\mathbf{o}[\mathbf{U}] = \mathbf{o}'[\mathbf{U}] = u$ and $\mathbf{o}[V] = x, \mathbf{o}'[V] = x'$; the semantics of $\bar{r}$ is the opposite. A **CP-table** for V is a collection of CP-rules for V, all sharing the same parent set $\mathbf{U}$, with at most one rule $r_{\mathbf{u}}$ or $\bar{r}_{\mathbf{u}}$ per state $\mathbf{u} \in Dom(\mathbf{U})$. The set $\mathbf{U}$ is also written $Pa(V)$, as it indeed corresponds to (a redundant superset of) the set of parent variables of V in the induced relation. Finally, given two opposite rules $r_{\mathbf{u}}$ and $\bar{r}_{\mathbf{u}}$, we define the notation of a global rule as follow.

Definition 1 (Rule schema)**.** *Let $r_{\mathbf{u}}$ and $\bar{r}_{\mathbf{u}}$ be two opposite rules on $V \in \mathbf{V}$. $R = (\mathbf{u} : V)$ denotes the common **rule schema** of $r_{\mathbf{u}}$ and $\bar{r}_{\mathbf{u}}$, where the preference over the assignment of V is not specified.*

Definition 2 (Conditional preference network)**.** *A **conditional preference network** (CP-net) is a triple $\mathcal{N} = (\mathbf{V}, A, CPT(V)_{V \in \mathbf{V}})$, where $(\mathbf{V}, A)$ is a directed graph on the set of variables $\mathbf{V}$, and for all V, $CPT(V)$ is a CP-table for V with $Pa(V) = \{X \in \mathbf{V}, (X, V) \in A\}$.*

The preference relation $\succ_{\mathcal{N}}$ induced by a CP-net $\mathcal{N}$ is the transitive closure of the relation induced by all rules in the CP-tables of $\mathcal{N}$. It is known that if the digraph $(\mathbf{V}, A)$ associated with $\mathcal{N}$ is acyclic, then the relation $\succ_{\mathcal{N}}$ is consistent (irreflexive). We focus on such **acyclic** CP-nets in this work.

We say that a binary CP-net is **complete** if $\forall V \in \mathbf{V}$, $|CPT(V)| = 2^{|Pa(V)|}$. Otherwise, we say that the CP-net is **incomplete**.

Fig. 2 depicts an example of a CP-net, which contains four variables A, B, C, D with three sets of independent variables $\{A, D\}, \{B, D\}, \{C, D\}$, and one variable B conditioned by A and C. We have, for example, $a'b'cd \succ a'bcd$ (bottommost rule) and $a'bcd \succ abcd$ (top left rule), and hence $a'b'cd \succ abcd$ (by transitivity).

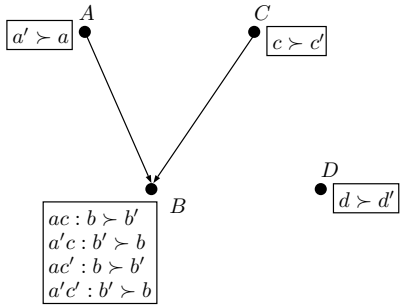

Fig. 2. A complete CP-net with four variables.

III. The learning problem

We consider the problem of learning a CP-net from observations of the form $\mathbf{o} \succ \mathbf{o}'$, where $(\mathbf{o}, \mathbf{o})_V$ forms a CP pair. Such preferences are natural since they correspond to statements like, *e.g.*, "I would prefer this car in blue color than in red color". We consider a stream Ω of observations, and our aim is to learn, at all times t (*i.e.*, after receiving t observations), an acyclic learned CP-net $\mathcal{N}_L^t$ maximizing an accuracy measure $\mathcal{L}$ w.r.t. Ω^t (the stream restricted to its first t observations).

We choose the following, natural **agreement** measure, which is also used in [16]:

$$\mathcal{L}(\Omega^t, \mathcal{N}_L^t) = \frac{|\{(\mathbf{o} \succ \mathbf{o}') \in \Omega^t \mid \mathbf{o} \succ_{\mathcal{N}_L^t} \mathbf{o}'\}|}{t}. \qquad (1)$$

In words, we seek to maximize the number of comparisons in the stream which the learned CP-net correctly predicts.

Since Ω may contain some noise, i.e. observations $\mathbf{o} \succ \mathbf{o}'$ and $\mathbf{o}' \succ \mathbf{o}$ for the same CP pair $(\mathbf{o}, \mathbf{o}')_V$, we cannot hope to achieve perfect accuracy (Eq. (1)) in general. This would only fit the noise and result in an inconsistent (reflexive) learned preference relation. Furthermore, the problem of testing the consistency of a cyclic CP-net is known to be PSPACE-hard [13]. For these reasons, we require acyclicity, and hence consistency, for the learned CP-net, which can be seen as a regularization bias. Fig. 3 gives a formal definition of our problem.

Fig. 3. The learning problem.

Input:	a data stream Ω of *ceteris paribus* pairwise comparisons
Ouput:	an acyclic CP-net $\mathcal{N}_L$
Measure:	maximize Eq. (1) (under acyclicity)

IV. Proposed algorithm for learning a CP-net

To cope with noise and maximize agreement with the observed data, the algorithm chooses, for each rule schema $R = (\mathbf{u} : V)$, the rule $r_{\mathbf{u}}$ or the rule $\bar{r}_{\mathbf{u}}$ to be the one

which corresponds to the greatest number of observations received so far. Hence our algorithm relies on identifying good rule schemas and for each one, on maintaining estimates of the number of received observations in favor of $r_{\mathbf{u}}$ and $\bar{r}_{\mathbf{u}}$. For the latter objective, we use estimated counters which are maintained online.

The high-level procedure is shown in Algorithm 1 (the learned CP-net $\mathcal{N}_L$ is a **global variable** that is not included as a parameter). This procedure processes the observed CP-comparisons online: for each new one, it generates a CP-rule (which always exists) by using the received swap (from which we deduce the preference), and the parents of the current variable in the CP-net (the conditioned part of the CP-rule). Then, it updates all counters with respect to the comparison and decides whether it is necessary to add a new parent variable or not (Line 5). Algorithm 1 also needs to set the maximum number of parents for each conditioned variable. If so, it finds the best new parent variable, and updates all the counters (Line 6). We describe all these steps in details in the rest of this section.

Algorithm 1: `learningCPNet()`

Data: A data flow Ω, a parent bound k, and a trust probability δ.
Result: A learned CP-net $\mathcal{N}_L$.

1 **Initialize** $\mathcal{N}_L$ and all counts to 0;
2 **for** $(\mathbf{o}, \mathbf{o}')_V \in \Omega$ **do**
3 Let $r = (\mathbf{u} : \mathbf{o}[V] \succ \mathbf{o}'[V])$ be the CP-rule induced by $(\mathbf{o}, \mathbf{o}')_V$ with $\mathbf{u} \in Dom(Pa(V))$ and R its associated rule schema;
4 `update_CPTable`$(V, R, r, \mathbf{o})$;
5 **if** *Eq.* (6) *and* $|Pa(V)| < k$ **then**
6 `addParent`$(CPT(V))$;

For the sake of clarity, we use the following notation. For a rule schema $R = (\mathbf{u} : V)$, a new parent variable $P \in \mathbf{V} \setminus (\{V\} \cup Pa(V))$ and a value $p \in Dom(P)$, we write R^p for the "extended" rule schema $(\mathbf{u}p : V)$. Similarly, given a CP-rule $r = (\mathbf{u} : v \succ v')$ (resp. $\bar{r} = (\mathbf{u} : v' \succ v)$), we write r^p for the rule $(\mathbf{u}p : v \succ v')$ (resp. $\bar{r}^p$ for $(\mathbf{u}p : v' \succ v)$). Finally, we denote the set of all nonparent variables of a variable V by $\overline{Pa}(V)$ (*i.e.*, $\overline{Pa}(V) = \mathbf{V} \setminus (Pa(V) \cup \{V\})$).

A. Counters

Definition 3 (Rule counter)**.** *Let $V \in \mathbf{V}$ be a variable and $R \in CPT(V)$ a rule schema. We call **rule counter** the number of observed swaps that support the CP-rule r (resp $\bar{r}$), which is denoted by $c(r)$ (resp. $c(\bar{r})$).*

Definition 4 (Nonparent counter)**.** *Let $V \in \mathbf{V}$ be a variable, $R \in CPT(V)$ a rule schema, and $X \in \overline{Pa}(V)$. We call **nonparent counter** the number of observed swaps $(\mathbf{o}, \mathbf{o}')_V$ that support the CP-rule r with $\mathbf{o}[X] = x$ (resp $\mathbf{o}[X] = x'$), which is denoted by $c(x_r)$ (resp. $c(x'_r)$). We define in the same way the nonparent counter for the opposite rule $\bar{r}$, with $c(x_{\bar{r}})$ (resp. $c(x'_{\bar{r}})$).*

In an online setting, we cannot memorize each received observation, not even a counter for each of them since the data stream can be infinite. Hence we use rule counters (Definition 3) and nonparent counters (Definition 4) that are maintained online. Precisely, for each rule schema $R = (\mathbf{u} : V)$ in the current learned CP-net $\mathcal{N}_L$, it maintains an estimate $c(r)$ (resp. $c(\bar{r})$) of the number of observed CP-comparisons that support the rule $r = (\mathbf{u} : v \succ v')$ (resp. $\bar{r} = (\mathbf{u} : v' \succ v)$), where $\mathbf{o} \succ \mathbf{o}'$ supports $(\mathbf{u} : v \succ v')$ if $(o, o')_V$ is a CP-pair and $o[Pa(V)] = \mathbf{u}, \mathbf{o}[V] = v, \mathbf{o}'[V] = v'$ hold. Moreover, for each nonparent variable $X \in \overline{Pa}(V)$ with domain $\{x, x'\}$, the algorithm maintains estimates $c(x_r)$, $c(x'_r)$, $c(x_{\bar{r}})$, and $c(x'_{\bar{r}})$). For instance, the counter $c(x_r)$ estimates the number of observed comparisons $o \succ o'$ that support r and satisfy $o[X] = o'[X] = x$. Observe that we only need $m(4n + 2)$ counters for a CP-net $\mathcal{N}_L$ containing m rules over n variables. The estimation of the updated counters is described in the sequel.

B. Parent decision phase

The parent decision phase consists in deciding whether a variable needs a new parent (for maximizing the agreement), and draws inspiration from the stochastic multi-armed bandit problem [2].

To each variable V we associate a value of "information gain", representing to what extent it would be worth adding a new parent to V. Intuitively, V needs a new parent when there are many states $\mathbf{u} \in Dom(Pa(V))$ such that there are the same number of observations supporting the CP-rule $\mathbf{u} : v \succ v'$ and the rule $\mathbf{u} : v' \succ v$. More precisely, for each rule schema R for V (over its current set of parents $Pa(V)$) with associated CP-rules $r, \bar{r}$, we call R **impure** if both $c(r)$ and $c(\bar{r})$ are nonzero and equal (otherwise, the rule schema is pure when just one of the counters is equal to zero. Then the current set $Pa(V)$ is clearly sufficient). We then measure a degree of impurity using **Shannon's entropy** [5]. Let $f(r) = \frac{c(r)}{c(r)+c(\bar{r})}$ and $f(\bar{r}) = \frac{c(\bar{r})}{c(r)+c(\bar{r})}$ measure the relative (estimated) frequencies of observations supporting r and $\bar{r}$. Then we define

$$E(R) = \begin{cases} 0, \text{ if } c(r) = 0 \text{ or } c(\bar{r}) = 0, \\ -f(r)\log(f(r)) - f(\bar{r})\log(f(\bar{r})), \text{ otherwise.} \end{cases} \tag{2}$$

To aggregate impurity of all rules for a given variable V together, we simply define the "information gain" of V to be:

$$G(V) = \sum_{R \in CPT(V)} f_R E(R), \tag{3}$$

with $f_R = \frac{c(r)+c(\bar{r})}{\sum_{R' \in CPT(\mathbf{V})} c(r') + c(\bar{r}')}$ the estimated frequency of the rule schema R among all rule schemas of $\mathcal{N}$ (computed from the counters).

Obviously, in an online and noisy setting, we do not want to add one parent at each step, since the algorithm would overfit noise. Intuitively, the decision needs to be supported by sufficiently many observations, and for this we use **Hoeffding's bound** (also called "additive Chernoff's bound") [7], [12], [17]. Let $|Y|$ be the range of a (real-valued) random variable Y, assume that we have observed m realizations of Y's observations, and write

$$\epsilon = \sqrt{\frac{|Y|^2 \ln\left(\frac{1}{\delta}\right)}{2m}}. \tag{4}$$

Then, Hoeffding's bound states that the mean of the observations is in $[\tilde{y} - \epsilon, \tilde{y} + \epsilon]$ with a probability $1 - \delta$, where $\tilde{y}$ is the true mean of the values in Y.

Back to our algorithm, we consider the mean of the information gains of a variable V as computed after seeing each of m comparisons on V:

$$\tilde{G}_{m_V}(V) = \frac{1}{m_V} \sum_{i=1}^{m_V} G_i(V), \tag{5}$$

with m_V the number of times we have observed comparisons on V (*i.e.*, CP-pairs $(o, o')_V$), and $G_i(V)$ the information gain computed for V after seeing the i^{th} comparison on V (we can see this mean as a way to give more importance to the current gain). Following Hoeffding's bound, if V is a variable such that it has the highest observed $\tilde{G}_{m_V}(V)$ after seeing m_V comparisons, we decide to select V as a conditioned variable if and only if

$$\tilde{G}_{m_V}(V) > \epsilon_{m_V} \tag{6}$$

holds, with $|Y| = \log(2)$, $m = m_V$ and $\delta \in]0, 1]$ a hyperparameter (i.e. a parameter which is fixed by the user) in Eq. 4. That is, if observations are drawn i.i.d. and up to the estimation induced by the counters, variable V is indeed the best choice with a probability of $1 - \delta$. The intuition behind Eq. 4 is to look for a conditioned variable which has a sufficient entropy while enough swaps have been seen. For more details, the reader can refer to [7, Section 2].

Algorithm 2: `update_CPTable`$(V, R, r, \mathbf{o})$

Data: A variable V, a CP-rule r with its associated rule schema R, and an outcome $\mathbf{o}$.
Result: Update each corresponding counter.

$c(r) \leftarrow c(r) + 1$;
for $X \in \overline{Pa}(V)$ **do**
  if $\mathbf{o}[X] = x$ **then** $c(x_r) \leftarrow c(x_r) + 1$;
  else $c(x'_r) \leftarrow c(x'_r) + 1$;
  Update the nonparent variable relevance Eq. (7);
if $c(r) > c(\bar{r})$ *and* $\bar{r} \in CPT(V)$ **then**
  $CPT(V) \leftarrow (CPT(V) \setminus \{\bar{r}\}) \cup \{r\}$;
if $c(r) < c(\bar{r})$ *and* $r \in CPT(V)$ **then**
  $CPT(V) \leftarrow (CPT(V) \setminus \{r\}) \cup \{\bar{r}\}$;

C. Parent search phase

Once the algorithm has decided to add a parent to a variable V with domain $\{v, v'\}$, the parent search phase consists of finding the best new parent. By definition, a parent variable P of V is characterized by having, in at least one context $\mathbf{u}$, value p when v is preferred to v', and the other value p' when v' is preferred. Hence for $r = (\mathbf{u} : v \succ v')$ and $Dom(P) = \{p, p'\}$ we should observe either (i) many comparisons supporting $r^p = (p\mathbf{u} : v \succ v')$ and many supporting $\bar{r}^{p'} = (p'\mathbf{u} : v' \succ v)$, or (ii) many supporting $r^{p'}$ and many supporting $\bar{r}^p$. Then, using the counters, we define the **relevance** of each nonparent variable X of V:

$$rel(X, V) = \frac{1}{|CPT(V)|} \sum_{R \in CPT(V)} \frac{\max(C, \bar{C})}{c(r) + c(\bar{r})}, \tag{7}$$

where $C = c(x_r) + c(x'_{\bar{r}})$ and $\bar{C} = c(x'_r) + c(x_{\bar{r}})$, with $c(r) + c(\bar{r})$ a normalization value. This relevance can be gradually updated without browsing all CP-rules (Line 5 of Algorithm 2).

We can see that $\frac{1}{2} \leq rel(P, V) \leq 1$. However, the worst case for a parent candidate P is, for each of these assignments and each type of a CP-rule, to observe the same counters, i.e. $rel(P, V) = \frac{1}{2}$. Then, the algorithm only accepts variables that have $rel(P, V) > \frac{1}{2}$. Finally, the best parent variable for V is defined by

$$\begin{array}{ll} \underset{P \in \text{Acy}(\overline{Pa}(V))}{\text{argmax}} & rel(P, V), \\ \text{s.t.} & rel(P, V) > \frac{1}{2}, \end{array} \tag{8}$$

(breaking ties arbitrarily), where Acy restricts the candidate to those whose addition induces no cycle in the digraph of $\mathcal{N}_L$ (which can be computed in time linear in the size of the digraph).

D. Updating rules and counters

When no parent is added, updating the counters simply consists of incrementing by 1 those supported by the just received observation, and for each rule schema R in $\mathcal{N}_L$ we only need to check which of $c(r)$ and $c(\bar{r})$ is highest.

Contrastingly, when P is chosen as a new parent of V, we need to update all rule schemas in $CPT(V)$ and their corresponding counters. Let $R = (\mathbf{u} : V)$ be a schema with rules $r, \bar{r}$, and let P be a new parent variable for V. By definition of $c(p_r)$, we can compute the new counter for r^p by

$$c(r^p) = c(p_r), \tag{9}$$

and similarly for $\bar{r}^p, r^{p'}, \bar{r}^{p'}$. We however also need to compute counters $c(b_{r^p}), c(b'_{r^p}), c(b_{r^{p'}}), \ldots$ for all remaining nonparent variables B of V (with domain $\{b, b'\}$) and all newly introduced rules. Due to the loss of information in online methods, it is obviously difficult to find the exact new counter for the remaining nonparent variables. The problem of estimating the new counters can be viewed as follows: consider a known set of objects $\mathbf{E}$ and a known size of two subsets $\mathbf{E}_1 \subseteq \mathbf{E}$ and $\mathbf{E}_2 \subseteq \mathbf{E}$. How can we estimate the size of the intersection

$|\mathbf{E}_1 \cap \mathbf{E}_2|$? Such an intersection can be bound between the Łukasiewicz t-norm (lower bound), denoted here by $c^-(.)$, and the Nilpotent minimum (upper bound), denoted here by $c^+(.)$. Then, we obtain for an assignment $b \in Dom(B)$ and a new rule r^p:

$$c^-(b_{r^p}) = \max\left(c(b_r) - c(r) + c(p_r), 0\right), \quad (10)$$

$$c^+(b_{r^p}) = \min\left(c(b_r), c(p_r)\right). \quad (11)$$

These bounds bring us to obtain $c^-(b_{r^p}) \leq c^*(b_{r^p}) \leq c^+(b_{r^p})$. We conservatively choose the lower bound $c^-(.)$ (Equation (10)), which is the estimated new counter on the overlap between the counter for the nonparent variable B as associated to R, and the new parent P as associated to R (and similarly for value b' of B and rules $r^{p'}, \bar{r}^p, \bar{r}^{p'}$). This estimation is the price to pay for the online setting. However, all counters are used after seeing m_V swaps (Eq. (4)) for the variable V, then the error made by this conservative bound is statistically corrected with a sufficient probability (see Section V). Moreover, this is the only place where an estimation occurs, and we show in our experiments that this works well in practice.

Algorithm 3 summarizes the whole procedure.

Algorithm 3: `addParent`$(CPT(V))$

Data: The CP-table $CPT(V)$.
Result: Update, if it is possible, the parents of V.

Choose P by applying Eq. (8);
if P *exists* **then**
  $Pa(V) \leftarrow Pa(V) \cup \{P\}$;
  for $R \in CPT(V)$ **do**
    Replace R by R^p and $R^{p'}$;
    Update all new CP-rules (Eq. (9)) and nonparent variables (Eq. (10)) counts;
    $CPT(V) \leftarrow CPT(V) \setminus \{r, \bar{r}\}$;
    if $c(r^p) > c(\bar{r}^p)$ **then**
      $CPT(V) \leftarrow CPT(V) \cup \{r^p\}$;
    if $c(r^p) < c(\bar{r}^p)$ **then**
      $CPT(V) \leftarrow CPT(V) \cup \{\bar{r}^p\}$;
    if $c(r^{p'}) > c(\bar{r}^{p'})$ **then**
      $CPT(V) \leftarrow CPT(V) \cup \{r^{p'}\}$;
    if $c(r^{p'}) < c(\bar{r}^{p'})$ **then**
      $CPT(V) \leftarrow CPT(V) \cup \{\bar{r}^{p'}\}$;
  $rel(V, V') \leftarrow 0, \forall V' \in \overline{Pa}(V)$;

a) Example: Let A, B, C be three variables and Ω be the following database:

1	$abc \succ abc'$
2	$a'bc' \succ a'bc$
3	$a'b'c' \succ a'b'c$

We fix our hyperparameter δ to 0.7. We first handle $abc \succ abc'$. After updating all counters, $\mathcal{N}_L$ contains only the CP-rule $\{\emptyset : c \succ c'\}$ and Ineq. (6) is `False`, so no parent is added. Then we handle $a'bc' \succ a'bc$. Let $r = (\emptyset : c \succ c')$ and $\bar{r} = (\emptyset : c' \succ c)$. We have the best information gain with variable C, with $\tilde{G}_2(C) \approx 0.15 > \epsilon \approx 0.13$. Hence we add a parent to C so as to split the CP-rule into two distinct ones. We have $rel(A, C) = 1$ and $rel(B, C) = \frac{1}{2}$, hence we choose A and update the counters of all new CP-rules using Eq. (9) and Eq. (10). The learned CP-net $\mathcal{N}_L$ now contains two CP-rules: $\{a : c \succ c', a' : c' \succ c\}$. We finally handle $a'b'c' \succ a'b'c$ and update all counters. Ineq. (6) is `False` and the algorithm stops. We obtain a CP-net which contains two CP-rules, which (in this small example) perfectly fit the observations.

V. Theoretical results

We remind that we denote by Ω the dataset which contains a set of swaps. This section begins by a proof of the soundness and the completeness of our algorithm.

Proposition 1 (Soundness)**.** *Algorithm 1 always returns an acyclic CP-net (which is possibly incomplete) for any dataset* Ω*.*

Proof. By construction: each arc added between two variables corresponds to a parent link (Line 3 of Algorithm 3) with a duplication of all CP-rules of the current variable w.r.t. the values of the new parent (Line 5 of Algorithm 3). Furthermore, the acyclic property is verified once Algorithm 3 adds a parent (Eq. (8)). Finally, the only objects that are added to the CP-tables are CP-rules (Line 6 of Algorithm 3), which verify the definition of an acyclic CP-net, and prove the soundness of Algorithm 1. □

Definition 5 (Restriction of a CP-net)**.** *Let* $(<, \mathbf{V}^2)$ *be an order on the arcs in the digraph of a CP-net* $\mathcal{N}$*. For a current arc* (P_0, V_0)*, we denote by* $\mathcal{N}_{(P_0,V_0)}$ *the* ***restriction*** *of* $\mathcal{N}$ *for the relation* $(P, V) < (P_0, V_0)$*, i.e.* $\mathcal{N}$ *contains only the arcs that have a lower position in* $(<, \mathbf{V}^2)$ *(including* (P_0, V_0)*), where the restriction is defined by:*

(i) $\forall V \in \mathbf{V}$*, if* $Pa_{(P_0,V_0)}(V) = Pa(V)$*, then* $CPT_{(P_0,V_0)}(V) = CPT(V)$*;*
(ii) *if* $Pa_{(P_0,V_0)}(V) \subset Pa(V)$*, then* $(\mathbf{u} : v \succ v') \in CPT_{(P_0,V_0)}(V)$ *if and only if the majority of rules in* $\mathcal{N}$ *has the form* $(\mathbf{uw} : v \succ v')$*, where* $\mathbf{u} \in Pa_{(P_0,V_0)}(V)$*, and* $\mathbf{w} \in Pa(V), \mathbf{w} \notin Pa_{(P_0,V_0)}(V)$*.*

Definition 6 (Exact counter)**.** *We call* ***exact counter*** *for a CP-net* $\mathcal{N}$ *and a set of swaps* Ω *the actual number of times each rule and nonparent counter has been observed on* Ω *so far.*

Definition 7 (Information gain consistency)**.** *Let* $(<, \mathbf{V}^2)$ *be an arc ordering on* $\mathbf{V}$*, and a set of swaps* Ω*.* $(<, \mathbf{V}^2)$ *is* ***information gain consistent*** *w.r.t.* Ω *if, by using the exact rule counters of* $\mathcal{N}_{(P_0,V_0)}$ *and* Ω*:*

(i) (P, V) *is the successor of* (P_0, V) *in* $(<, \mathbf{V}^2)$*, where* V *is the variable that maximizes the information gain among all variables that have missing parents;*
(ii) $\forall (P, V)$ *in the digraph of* $\mathcal{N}$*,* $(P, V) < (P', V')$ *in* $(<, \mathbf{V}^2)$*, where* $P' \notin Pa(V')$ *in* $\mathcal{N}$*;*

(iii) *let* (P_{last}, V_{last}) *be the last arc in* $(<, \mathbf{V}^2)$ *to complete the digraph of* $\mathcal{N}$*, then* $\forall V \in \mathbf{V},\ G(V) = 0$ *for* $\mathcal{N}_{(P_{last}, V_{last})}$.

Lemma 1. *There always exists an order* $(<, \mathbf{V}^2)$ *which is information gain consistent for acyclic CP-nets.*

Proof. We construct an order $(<, \mathbf{V}^2)$ such that $\forall (P, V)$ with $P \in Pa(V)$ and $\forall (P', V')$ with $P' \notin Pa(V')$, $(P, V) < (P', V')$.
The point (i) of Definition 7 is verified by ordering and adding in a decreasing way following the values of information gain of all the arcs (P, V). The arcs (P', V') are considered only after the arcs (P, V): it means that all variables in $\mathbf{V}$ already have their parents, so their information gains are equal to 0, which verify the point (iii). Finally, the point (ii) is verified by the construction of $(<, \mathbf{V}^2)$. □

Intuitively, Lemma 1 says that the order $(<, \mathbf{V}^2)$ corresponds to the order where the algorithm has the highest probability to learn the relations.

The Hoeffding bound gives us a trust probability $(1 - \delta)$ which bounds all of the counters and information gains. Then, Algorithm 1 chooses the new parent variables following $(<, \mathbf{V}^2)$ with the trust probability $(1 - \delta)$. Finally, the algorithm will learn the appropriate structure since it will not add any parent after the arc $(P_{\text{last}}, V_{\text{last}})$ (point (iii) of Definition 7), which exactly corresponds to the set of parents of $\mathcal{N}^*$ (point (ii) of Definition 7).

Definition 8 (relevant consistency)**.** *Let* $(<, \mathbf{V}^2)$ *be an order on arcs in the digraph of a CP-net* $\mathcal{N}$ *and a set of swaps* Ω*.* $(<, \mathbf{V}^2)$ *is* ***relevant consistent*** *w.r.t.* Ω *if, by using the exact nonparents counters into* $\mathcal{N}_{(P_0, V_0)}$*,* (P, V) *is the successor of* (P_0, V_0) *in* $(<, \mathbf{V}^2), \forall (P_0, V_0) < (P_{last}, V_{last})$*, where* P *is the parent that maximizes the value of* $rel(P, V)$*.*

Lemma 2. *There always exists an order* $(<, \mathbf{V}^2)$ *which is information gain and relevance consistent w.r.t. a set of swaps* Ω*.*

Proof. Consider the order constructed in the proof of Lemma 1. We extend this order: $(P_1, V_1) < (P_2, V_1) < \ldots < (P_{k_1}, V_1) < \ldots < (P_0, V_0) < \ldots < (P_{\text{last}}, V_{\text{last}}) < (P', V'), \forall (P', V')$ s.t. $P' \notin Pa(V')$, where k_1 is the k^{th} parent of V_1, and $\forall i, j, l$ s.t. $i < j, rel(P_i, V_l) > rel(P_j, V_k)$. This extension verifies by construction the Definition 8, and is still compatible with Lemma 1 due to the independence of P in the information gain of V. □

Lemma 3. *Let an unnoised set of swaps* Ω *be a dataset which induces the optimal batch CP-net* $\mathcal{N}^*$*. If the counters used in Algorithm 1 are exact, then it learns all arcs* (P, V) *of* $\mathcal{N}^*$ *for an information gain and relevance consistent order* $(<, \mathbf{V}^2)$*.*

Proof. Let $\mathcal{N}_L$ the CP-net learned by Algorithm 1. We denote by $(P, V) \in \mathcal{N}^*$ the arcs that are in the digraph of $\mathcal{N}^*$. Suppose by induction that Algorithm 1 had already learned all arcs before (P, V) following $(<, \mathbf{V}^2)$, and the next arc is (P_0, V_0) (the initialization trivially works if we set $|\mathbf{V}| = 1$). Two cases occur:

- V_0 needs no parent, so $G(V_0) = 0$. Furthermore, Algorithm 1 selects the variable which maximizes Eq. (3), then $\forall V' \in \mathbf{V} \setminus \{V_0\}$ and $\forall P' \in \overline{Pa}(V'), (P_0, V_0) < (P', V')$ and $G(V') = 0$;
- V_0 needs the parent P_0. Then, $rel(P_0, V_0) > \frac{1}{2}$, and Algorithm 1 chooses the variable P which maximizes, after choosing V_0 the value of $rel(P, V_0)$ (Eq. (7)). Then, for $P = P_0$ and $\forall P' \in \overline{Pa}(V_0) \setminus \{P_0\}, rel(P_0, V_0) > rel(P', V_0)$. The arc (P_0, V_0) corresponds to the successor of (P, V) in the order $(<, \mathbf{V}^2)$, which proves the lemma.

□

A direct consequence of Lemma 3 is that for an unnoised dataset Ω, Algorithm 1 can learn the corresponding target CP-net $\mathcal{N}^*$ following the agreement measure defined in Eq. (1). It also proves the completeness of Algorithm 1. Another corollary of Lemma 3 is that the corresponding batch algorithm which receives all swaps between each adding parent phase can also learn the target CP-net $\mathcal{N}^*$, even for a noisy dataset Ω (with a noise probability lower than $\frac{1}{2}$).

Definition 9 (Arc disagreement)**.** *Let* $\mathcal{N}$ *and* $\mathcal{N}'$ *be two CP-nets respectively associated to two different learning algorithms* $\mathcal{A}$ *and* $\mathcal{A}'$*. We define the* ***arc disagreement****, denoted by* $d_a(\mathcal{N}_{(P,V)}, \mathcal{N}'_{(P',V')})$ *as the probability that*

$$d_a(\mathcal{N}_{(P,V)}, \mathcal{N}'_{(P',V')}) = \mathbb{P}(V \neq V')\mathbb{P}(P \neq P')\mathbb{P}(c \neq c'),$$

where (P, V) *and* (P', V') *respectively denote the new arc of* $\mathcal{N}$ *learned by* $\mathcal{A}$*, and the new arc of* $\mathcal{N}'$ *learned by* $\mathcal{A}'$*, and* c, c' *denote the disagreement between the counters of both CP-nets after adding the corresponding arc.*

Intuitively, Definition 9 expresses the probability of having, when the Hoeffding bound is used to find a new parent variable, a difference between two CP-nets: the conditioned variable, the parent variable, and/or the counters can differ. It is then easy, from this probability and the fact that an algorithm can only learn nk different parents ($\frac{n(n-1)}{2}$ with acyclicity constraint), to deduce the whole disagreement which is made by two different learning algorithms on two different CP-nets.

Definition 10 (CP-net disagreement)**.** *The* ***CP-net disagreement*** *between two CP-nets* $\mathcal{N}$ *and* $\mathcal{N}'$*, denoted by* $d_c(\mathcal{N}, \mathcal{N}')$*, corresponds to the sum of arcs disagreement for all possible arcs, i.e.*

$$d_c(\mathcal{N}, \mathcal{N}') = \sum_{i=1}^{nk} \mathbb{P}((P, V) \neq (P', V')),$$

where n *is the number of variables in the CP-nets and* k *the maximum number of parents per variable (modulo the acyclicity constraint).*

Roughly speaking, these two previous definitions express the probability of an algorithm to learn the optimal structure and preferences compared to the optimal learning algorithm. In our case, the batch algorithm is considered as the optimal one, which allows us to deduce the following learning bound.

Theorem 1. *Let $\mathcal{A}$ be the batch algorithm of Algorithm 1, $\mathcal{N}^*$ its learned CP-net, and $\hat{\mathcal{N}}$ the learned CP-net of Algorithm 1. Let Ω be an infinite noisy dataset (set of swaps) with $p < \frac{1}{2}$ the associated noise probability.*

Algorithm 1 has a CP-net disagreement equal to $d_c(\mathcal{N}, \mathcal{N}') \leq \frac{pnk(4n-2)\delta^3 2^{k+1}}{m}$, with n the number of variables, k the maximum number of parents per variable, δ the Hoeffding probability, and m the necessary number of swaps before the adding parent phase (we suppose here that $m_V = m, \forall V \in \mathbf{V}$).

Proof. Let $(\hat{P}, \hat{V})$ and (P^*, V^*) be, respectively, the arc added in $\hat{\mathcal{N}}$ by Algorithm 1, and the arc adding in $\mathcal{N}^*$ by the batch algorithm $\mathcal{A}$.

We firstly compute the probability of the arc disagreement between $\hat{\mathcal{N}}_{(\hat{P},\hat{V})}$ and $\mathcal{N}^*_{(P^*,V^*)}$:

- by the Hoeffding bound and the Lemma 3, we have $\mathbb{P}(\hat{V} \neq V^*) \leq \delta$;
- in the same manner, we have $\mathbb{P}(\hat{P} \neq P^*) \leq \delta$;
- the counters are estimated by the Łukaciewicz norm, but can be refined by the Hoeffding bound: after m observed examples, the mean of seeing each form of CP-rule is $\frac{m}{2^{k+1}}$ multiply by (i) the number of counters in one rule schema $(4n-2)$ and (ii) by the original Hoeffding probability δ. Then we obtain $\mathbb{P}(\hat{c} \neq c^*) \leq (4n-2)\frac{2^{k+1}\delta}{m}$.

We can deduce the probability of the arc distance:

$$\begin{aligned} d_a(\hat{\mathcal{N}}_{(\hat{P},\hat{V})}, \mathcal{N}^*_{(P^*,V^*)}) &= \mathbb{P}(\hat{V} \neq V^*)\mathbb{P}(\hat{P} \neq P^*)\mathbb{P}(\hat{c} \neq c^*) \\ &\leq \delta^2(4n-2)\frac{2^{k+1}\delta}{m} \\ &= \frac{(4n-2)\delta^3 2^{k+1}}{m}. \end{aligned}$$

Then, by using the union bound, we find the whole disagreement for all of the arcs that can be added during the learning phase:

$$\begin{aligned} d_c(\hat{\mathcal{N}}, \mathcal{N}^*) &= \sum_{i=1}^{nk} \mathbb{P}((\hat{P}, \hat{V}) \neq (P^*, V^*)) \\ &= \sum_{i=1}^{nk} \mathbb{P}(\hat{V} \neq V^*)\mathbb{P}(\hat{P} \neq P^*)\mathbb{P}(\hat{c} \neq c^*) \\ &\leq \sum_{i=1}^{nk} \delta^2(4n-2)\frac{2^{k+1}\delta}{m} \\ &= \frac{nk(4n-2)\delta^3 2^{k+1}}{m}. \end{aligned}$$

Finally, we apply the noise probability p of the dataset Ω to obtain the result which proves the theorem:

$$d_c(\hat{\mathcal{N}}, \mathcal{N}^*) \leq \frac{pnk(4n-2)\delta^3 2^{k+1}}{m}.$$

□

Theorem 1 shows that the disagreement polynomially increases following the number of variables n and the maximum number of parents per variable k.

Moreover, it is easy to see that the time complexity of the whole procedure after an observation has been received is in $O(mn)$, where m is the number of rules in the current learned CP-net $\mathcal{N}_L$ (the size of $\mathcal{N}_L$ is in $O(mn)$). Since it is easy, and reasonable, to further impose that the learned CP-net has a digraph with a small degree k (see the experiments), and hence that m is in $O(n^k)$, we finally get an **efficient learning algorithm for acyclic CP-nets in an online and noisy setting** (for more details about the complexity of learning acyclic CP-nets, we refer the reader to [1]).

VI. Experimental results

To evaluate the efficiency of our algorithm, three experimental protocols have been designed: the first one aimed at testing the accuracy with presence of noise, the computational time and the experimental convergence of our procedure. The second one consists in testing our algorithm on real datasets. Finally, the last one concerns the comparison between our algorithm and the online procedure proposed by [11].

We made these experiments on a laptop with an Intel core i7-4600U, 16Go of RAM and Python language[2].

In all of the graphics in the sequel, each point corresponds to a simple averaged value according to 30 runs, with a report of the standard deviation as an error bar on these points. We use Eq. (1) as accuracy measure.

A. Learning from synthetic noisy datasets

We consider an uniform random noise (it is appropriate to use the Hoeffding bound) represented by a probability $p \in [0, 1]$.[3] This amounts, for each pairwise comparison, to reverse at random the true preference with a probability p in order to represent the corrupted preferences. To generate our synthetic dataset Ω , we proceed as follow:

1) compute a random acyclic CP-net $\mathcal{N}_T$ with n variables following Algorithm 3 in [11] (we firstly generate random links between variables, then we generate random preferences for each CP-table);
2) generate a random pairwise comparison $(\mathbf{o}, \mathbf{o}')_V$ and deduce the preference from $\mathcal{N}_T$, i.e. $\mathbf{o} \succ_{\mathcal{N}_T} \mathbf{o}'$ or $\mathbf{o}' \succ_{\mathcal{N}_T} \mathbf{o}$;
3) let $\mathbf{o} \succ_{\mathcal{N}_T} \mathbf{o}'$ be the current preference. Generate a random number $x \in [0, 1]$. If $x > p$, then $\Omega \leftarrow \Omega \cup \{\mathbf{o} \succ \mathbf{o}'\}$. $\Omega \leftarrow \Omega \cup \{\mathbf{o}' \succ \mathbf{o}\}$ otherwise;

The first experiment is designed to assess the robustness of the algorithm w.r.t. noise. Fig. 4 depicts the algorithm's accuracy as a function of the amount of noise affecting the dataset. The algorithm shows that the accuracy varies following the given noise rate. We can also observe an overfitting of our structure when the number of parents is too high, with a decreasing accuracy.

Fig. 5 is obtained by learning a CP-net $\mathcal{N}_L$ from a noiseless dataset Ω and computing the error rate $1 - \mathcal{L}(\mathcal{N}_L, \Omega)$ (Eq. (1)).

[2]The code is available here: https://services.lamsade.dauphine.fr/owncloud/s/SW5UtCRk5BBYHHb.

[3]However, when $p > \frac{1}{2}$, Algorithm 1 will clearly learn the noisy preferences because they will be in majority.

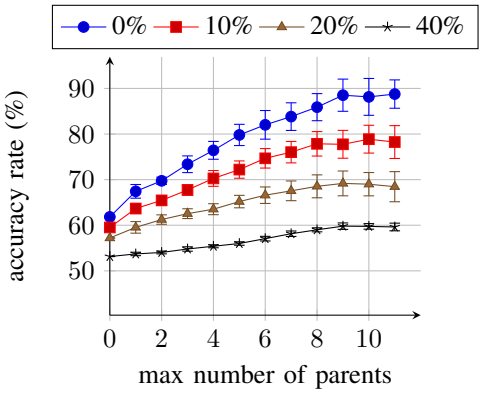

Fig. 4. Learning accuracy according to the maximum number of parents per variable. Dataset is randomly generated (20,000 comparisons, 12 variables, and $\delta = 0.95$) with a varied noise probability (one per graphic line).

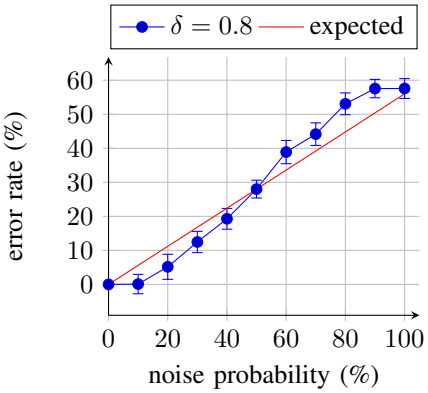

Fig. 5. Error rate between a noiseless and a noisy dataset according to noise level. Dataset is randomly generated (10,000 comparisons, and 12 variables).

We corrupt Ω with various noise levels (denoted by Ω'), and learn a new CP-net $\mathcal{N}'_L$ from it. We compute the error rate using the original dataset, i.e., $1 - \mathcal{L}(\mathcal{N}'_L, \Omega)$. Finally, the error rate from noiseless and the one from noisy dataset are subtracted to obtain the points in Fig. 5. The obtained error rate does not reflect the true one, and induces a bias in the graphic lines because we cannot perfectly learn the CP-net. Again, the algorithm demonstrates a quite good robustness to noise, depending on the hyperparameter value δ.

Fig. 6 reveals a lack of efficiency for values of δ less or equal to 0.8. For higher values, the algorithm allows for adding more new parent variables, which amounts to increase the accuracy scores.

Fig. 7 shows the convergence of our algorithm. Note that we compute here the error rate after each iteration of Line 2 in Algorithm 1. One can observe that this error decreases drastically at the first steps, and continue decreasing smoothly as soon as new pairwise comparisons are acquired.

Finally, the computation time is given by Fig. 8. One can observe a linear growth of the computation time w.r.t. the number of parents. Moreover, this time, and its extreme values, also grow linearly w.r.t. the size of Ω.

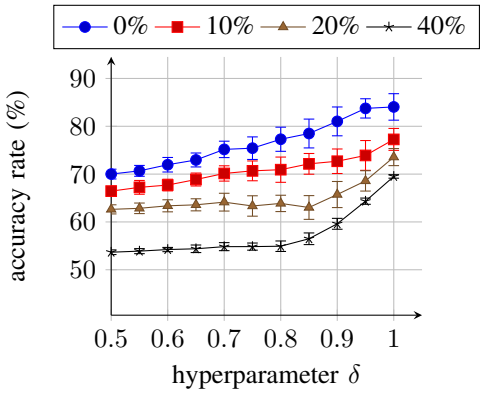

Fig. 6. Learning accuracy according to the hyperparameter δ. Dataset is randomly generated (20,000 comparisons, 15 variables) with a varied noise probability (one per graphic line).

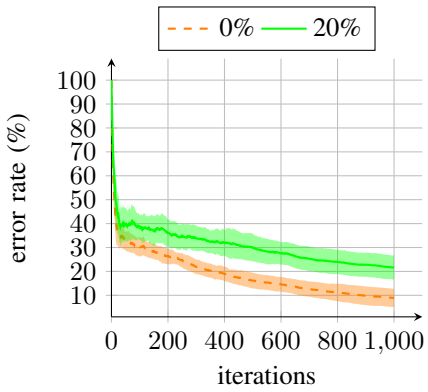

Fig. 7. Experimental convergence of our algorithms. One iteration correspond to a consideration of one pairwise comparison. Dataset is randomly generated (1,000 comparisons, 8 variables, and $\delta = 0.95$) with a varied noise probability (one per graphic line).

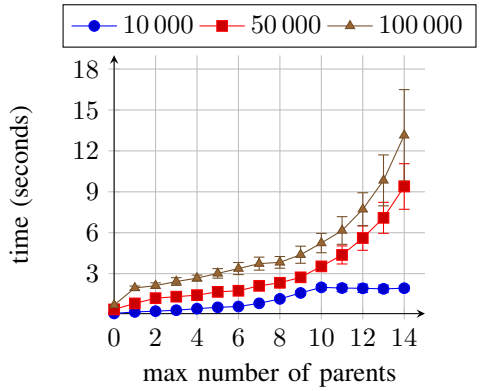

Fig. 8. Learning time according to the maximum number of parents per variable. Dataset is randomly generated (15 variables, $\delta = 0.95$) with varied number of comparisons (one per graphic line).

B. *Learning from a real dataset*

We consider in this section the TripAdvisor[4] dataset [22], [23], which contains about 81,000 hotel reviews (from 1,850

[4]http://times.cs.uiuc.edu/~wang296/Data/, "text" folder.

different hotels) represented by seven ratings (value aspect, rooms aspect, location aspect, cleanliness aspect, check in/front desk aspect, service aspect and business service aspect) plus one overall rating (which represents the preference relation). To be able to learn a binary CP-net, we compute, for each hotel, its median review, and we rescale it with the median values in order to obtain the median binary review for each hotel (for more details, see the procedure described in [16, Subsection 5.1]).

One of the limits of CP-nets is that cycles of size 2 between two *ceteris paribus* outcomes cannot be represented (even for cyclic CP-nets). However, such a cycle can occur in real life, and deteriorates the algorithm's learning accuracy. Hence, we do not use these 2-cycles in this experiment.

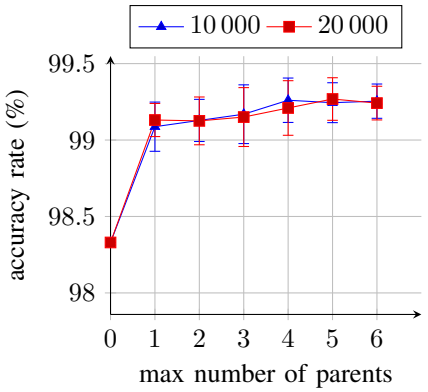

Fig. 9. Learning accuracy according to the maximum number of parent per variable from real datasets. For each dataset, we select two different number of pairwise comparisons.

We can observe from Fig. 9 that without 2-cycles, we can learn the whole structure with about 99% of accuracy (the remaining 1% may correspond to the noisy part of the comparisons). Moreover, we can see that there are not many conditional preferences in this dataset (we have less than 1% of accuracy difference between the unconditional network and the complete CP-net). We also can observe that the size of the TripAdvisor dataset does not affect the accuracy.

C. Comparison with Guerin et al.

In order to reproduce the experiments in [11], we fix the maximum number of parents to 5 and the density of $\mathcal{N}_T$, denoted by $\lambda = \frac{\#\text{arcs}}{n}$, which is a ratio of arcs per node. Furthermore, we consider the indecision case, meaning that one cannot decide whether $\mathbf{o} \succ \mathbf{o}'$ or $\mathbf{o}' \succ \mathbf{o}$, and we use Eq (1) as accuracy measure[5].

Table I shows better results for our algorithm due to the absence of indecision. However, in some cases, the algorithm proposed in [11] has a better disagreement. When the number of variables is small, we can observe close agreement results between both algorithms, when the indecision case does not appear.

[5]disagreement $= 1 -$ (agreement $+$ indecision).

TABLE I
COMPARISON RESULTS BETWEEN OUR ALGORITHM (GRAY CELLS) AND ALGORITHM OF GUERIN *et al.* (WHITE CELLS), WITH $20,000$ COMPARISONS, AND $\delta = 0.95$. THE BEST RESULT IS WRITTEN IN BOLD.

n	Accuracy (%)	Error (%)	Indecision (%)
	$\lambda = 1$		
4	100	0	0
	100	0	0
8	72	3	25
	100	**0**	**0**
12	70	**1**	29
	98	2	**0**
	$\lambda = 3$		
4	**98**	**2**	0
	97	3	0
8	51	**1**	48
	91	9	**0**
12	40	**2**	58
	89	11	**0**

VII. CONCLUSION

We have proposed in this paper a new online learning algorithm of acyclic CP-nets from noisy datasets. This procedure improves the results presented in [11] and provides an asymptotically optimal decision for the choice of conditioned variables. The theoretical and experimental results show the efficiency of its robustness to noise, and a fast convergence during the first iterations.

Future work will concern the management of parent variables in the learned CP-net, when, at some point, these variables become useless and could be deleted. We are also interested about agnostic PAC-learning properties that may guarantee stronger theoretical properties of our algorithm.

REFERENCES

[1] E. Alanazi, M. Mouhoub, and S. Zilles. The complexity of learning acyclic cp-nets. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1361–1367, 2016.

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[3] 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. Artif. Intell. Res.*, 21:135–191, 2004.

[4] Y. Chevaleyre, F. Koriche, J. Lang, J. Mengin, and B. Zanuttini. Learning ordinal preferences on multiattribute domains: The case of cp-nets. In *Preference Learning.*, pages 273–296. 2010.

[5] T. M. Cover and J. A. Thomas. *Elements of information theory (2. ed.)*. Wiley, 2006.

[6] Y. Dimopoulos, L. Michael, and F. Athienitou. Ceteris paribus preference elicitation with predictive guarantees. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 1890–1895, 2009.

[7] P. M. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, MA, USA, August 20-23, 2000*, pages 71–80, 2000.

[8] A. Eckhardt and P. Vojtás. How to learn fuzzy user preferences with variable objectives. In *Proceedings of the Joint 2009 International Fuzzy Systems Association World Congress and 2009 European Society of Fuzzy Logic and Technology Conference, Lisbon, Portugal, July 20-24, 2009*, pages 938–943, 2009.

[9] A. Eckhardt and P. Vojtás. Learning user preferences for 2cp-regression for a recommender system. In *SOFSEM 2010: Theory and Practice of Computer Science, 36th Conference on Current Trends in Theory and Practice of Computer Science, Spindleruv Mlýn, Czech Republic, January 23-29, 2010. Proceedings*, pages 346–357, 2010.

[10] J. Fürnkranz and E. Hüllermeier, editors. *Preference Learning*. Springer, 2010.

[11] J. T. Guerin, T. E. Allen, and J. Goldsmith. Learning cp-net preferences online from user queries. In *Late-Breaking Developments in the Field of Artificial Intelligence, Bellevue, Washington, USA, July 14-18, 2013*, 2013.

[12] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[13] F. Koriche and B. Zanuttini. Learning conditional preference networks. *Artif. Intell.*, 174(11):685–703, 2010.

[14] J. Liu, C. Sui, D. Deng, J. Wang, B. Feng, W. Liu, and C. Wu. Representing conditional preference by boosted regression trees for recommendation. *Inf. Sci.*, 327:1–20, 2016.

[15] J. Liu, Y. Xiong, C. Wu, Z. Yao, and W. Liu. Learning conditional preference networks from inconsistent examples. *IEEE Trans. Knowl. Data Eng.*, 26(2):376–390, 2014.

[16] J. Liu, Z. Yao, Y. Xiong, W. Liu, and C. Wu. Learning conditional preference network from noisy samples using hypothesis testing. *Knowl.-Based Syst.*, 40:7–16, 2013.

[17] O. Maron and A. W. Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. In *Advances in Neural Information Processing Systems 6, [7th NIPS Conference, Denver, Colorado, USA, 1993]*, pages 59–66, 1993.

[18] L. Michael and E. Papageorgiou. An empirical investigation of ceteris paribus learnability. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 1537–1543, 2013.

[19] T. Sandholm. Expressive commerce and its application to sourcing: How we conducted $35 billion of generalized combinatorial auctions. *AI Magazine*, 28(3):45–58, 2007.

[20] A. Tsoukiàs. From decision theory to decision aiding methodology. *European Journal of Operational Research*, 187(1):138–161, 2008.

[21] T. Walsh. Representing and reasoning with preferences. *AI Magazine*, 28(4):59–70, 2007.

[22] H. Wang, Y. Lu, and C. Zhai. Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, July 25-28, 2010*, pages 783–792, 2010.

[23] H. Wang, Y. Lu, and C. Zhai. Latent aspect rating analysis without aspect keyword supervision. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 618–626, 2011.