Conditionally Acyclic CO-Networks for Efficient Preferential Optimization

Pierre-François Gimenez, Jérôme Mengin

ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230352

Conditionally Acyclic CO-Networks for Efficient Preferential Optimization

Pierre-François Gimenez [a;*] **and Jérôme Mengin** [b;**]

[a]CentraleSupélec, Univ. Rennes, IRISA, France
[b]IRIT, Université Paul Sabatier, CNRS, France

Abstract. This paper focuses on graphical models for modelling preferences in combinatorial space and their use for item optimisation. The preferential optimisation task seeks to find the preferred item containing some defined values, which is useful for many recommendation settings in e-commerce. We show that efficient (i.e., with polynomial time complexity) preferential optimisation is achieved with a subset of cyclic CP-nets called conditional acyclic CP-net. We also introduce a new graphical preference model, called Conditional-Optimality networks (CO-networks), that are more concise than conditional acyclic CP-nets and LP-trees but have the same expressiveness with respect to optimisation. Finally, we empirically show that preferential optimisation can be used for encoding alternatives into partial instantiations and vice versa, paving the way towards CO-nets and CP-nets unsupervised learning with the minimal description length (MDL) principle.

1 Introduction

Online shopping services, like video-on-demand streaming services and product configurators for computers, cars, or kitchens, rely on recommendation and customisation of the user experience to boost sales [29]. Recommendations are essential in large, combinatorial product spaces, where the number of alternatives can lead to overchoice confusion [17]. In such a case, a user is overwhelmed by the possibilities and cannot choose. A common tool in configurators is optimal completion, where the configurator automatically completes a partially configured product by maximising the product's utility for the user. Recommendation, and optimal completion, in particular, are typically based on a modelling of user preferences. However, except when the number of attributes is very small, it is intractable to represent a linear order over the space of all possible alternatives in extension, so for a decision-maker to give their preferences, some structure is needed. Several types of graphical representations of preferences have been studied in the literature. Combinatorial preferences can be modelled with numerical models, such as GAI-nets [14] and ensemble ranking function [12], or by ordinal graphical models, such as lexicographic preferences trees (LP-trees [11]) and conditional preferences networks (CP-nets [4]). We focus on the latter models because the optimal completion query can be answered in polynomial time with LP-trees and acyclic CP-nets with the Forward Sweep algorithm [4].

* Corresponding Author. Email: pierre-francois.gimenez@centralesupelec.fr
** Corresponding Author. Email: Jerome.Mengin@irit.fr

LP-trees are graphical representations of a linear (total) order based on the relative importance of attributes: to compare two alternatives, their value for the most important attribute is compared. If they are different, then we can conclude which alternative is preferred. Otherwise, the second most important attribute is used for comparison, and so on. CP-nets are based on *ceteris paribus* preferences and only encode the comparison between two outcomes that differ by the value of only one variable. Therefore, some alternatives may be incomparable, even by transitivity. However, since CP-nets do not encode attribute importance, they are more compact than LP-trees. So, these two model classes have a trade-off: LP-trees describe total orders, and CP-nets are more compact. However, CP-nets are more general in the sense that they can represent, albeit partially, any preference relation, whereas LP-trees can only represent (generalised) lexicographic preference relations.

Our contributions can be summarised as follow:

- we show that conditional acyclic CP-nets, introduced by [28], are exactly the CP-nets for which Forward Sweep, a polytime algorithm for preferential optimisation, can be applied to;
- we introduce an even more compact version of CP-nets, called CO-nets (conditional optimality networks), that can be used to answer optimisation queries (under some structural condition) about, in particular, but not restricted to, generalised lexicographic preferences;
- we show that preferential optimisation can be used for encoding and decoding data effectively in a *Minimum Description Length* approach to learning preferences with an empirical comparison to popular compression algorithms, paving the way towards unsupervised learning of CP-nets with the (MDL) principle.

Section 2 gives some background about preferences models. Section 3 proves the the characterisation of conditional acyclic CP-nets as the set of CP-nets which efficient preferential optimisation. Section 4 presents conditional optimality networks and their associated optimisation algorithm. Section 5 studies the relationship between different subclasses of LP-trees and CP-nets concerning the optimisation query. Finally, Section 6 presents a new encoding and decoding technique based on preferential optimisation experimentally compared to popular compression algorithms. Section 7 concludes.

2 Background and notations

Combinatorial Domain We consider a combinatorial domain over a finite set $\mathcal{X}$ of discrete attributes that characterise the possible alternatives. Each attribute $X \in \mathcal{X}$ has a finite set of possible values $\underline{X}$.

$\underline{\mathcal{X}}$ denotes the Cartesian product of the domains of the attributes in $\mathcal{X}$, its elements are called *alternatives*. We often use the symbols o, $o', o_1, o_2, \ldots$ to denote alternatives. In the following, n is the number of attributes in $\mathcal{X}$, and d is a bound on the size of the domains of the attributes: for every $X \in \mathcal{X}$, $2 \leq |\underline{X}| \leq d$.

For a subset U of $\mathcal{X}$, we will denote by $\underline{U}$ the Cartesian product of the domains of the attributes in U, every $u \in \underline{U}$ is an instantiation of U, or partial instantiation (of $\mathcal{X}$). If v is an instantiation of some $V \subseteq \mathcal{X}$, $v[U]$ denotes the restriction of v to the attributes in $V \cap U$. We say that instantiations $u \in \underline{U}$ and v are compatible if $v[U \cap V] = u[U \cap V]$, written $u \sim v$. If $U \subseteq V$ and $v[U] = u$, we say that v extends u, also written $u \subseteq v$.

Preference relations In this paper, we consider only preference relations that do not allow for indifference: such a preference relation is a linear order over $\underline{\mathcal{X}}$, that is, a total, transitive, irreflexive binary relation over $\underline{\mathcal{X}}$, often denoted with curly symbol $\succ$. For alternatives $o, o' \in \underline{\mathcal{X}}$, $o \succ o'$ indicates that o is strictly more preferred to o'.

In many settings, one is essentially interested in finding some alternative that is optimal in some restricted set of alternatives, in the sense that no other alternative "beats" it / is strictly more preferred in that set. In particular, in interactive configuration settings, given a partial instantiation $u \in \underline{U}$ already built by a user, it can be useful to show the user what is the best completion of u, according to her preferences. We denote $opt(u, \succ)$ the most preferred – according to $\succ$ – alternative compatible with u; it exists and is unique in a linear order; in later sections we will study some incomplete representations of preferences, and we will use the same notation $opt(u, \succ)$ when the most $\succ$-preferred alternative compatible with u exists and is unique. We say that alternative o, such that $o[U] = u$, is *u-undominated* if and only if there is no alternative o' such that $o'[U] = u$ and $o' \succ o$. o is *undominated* if there is no alternative o' such that $o' \succ o$.

Graphical models It has long been observed that, given the exponential size of $\underline{\mathcal{X}}$, for any practical purpose one must make the assumption that the preference relations of interest exhibit some *structure*. Several models that have been studied in the AI literature to represent preference relations separate the information into three components: 1) a graph (or sometimes a hypergraph), representing some relationship between attributes; 2) some local information about preferences, in tables associated with the nodes of the graph; 3) a rule to aggregate the local preferences into a global binary relation over $\underline{\mathcal{X}}$. We focus below on two such models: *Conditional Preference Networks* (CP-nets) and *Lexicographic Preference Trees* (LPTs).

CP-nets CP-nets have been introduced by [4] as a tool to make explicit a particular kind of structure, called *preferential (in)dependence*. We give below a slightly more general definition[1] of preferential independence than that of [4]:

Definition. *Attribute X is said to be* preferentially independent *of attribute $Y \neq X$ given $u \in \underline{U}$ for some $U \subseteq \mathcal{X} \setminus \{X, Y\}$ with respect to preference relation $\succ$ if for every $x, x' \in \underline{X}, y, y' \in \underline{Y}, v \in \underline{\mathcal{X} \setminus (U \cup \{X, Y\})}$, $uvxy \succ uvx'y$ if and only if $uvxy' \succ uvx'y'$. We write that X is preferentially independent of Y if X is preferentially independent of Y given the empty assignment $u = \top$.*

[1] Although the initial definition of CP-nets allows for indifference in conditional preference table, [4] also point out that this leads to some difficulty in the semantics of CP-nets.

Note that preferential independence is not necessarily symmetric. A CP-net is a structure that captures / represents the preferential independencies inherent in a given preference relation. Figure 1a depicts a CP-net φ_0. More generally, a CP-net is a triple $\varphi = (\mathcal{X}, \mathsf{Pa}, \mathsf{CPT})$, where:

- Pa associates to every attribute $X \in \mathcal{X}$, a subset $\mathsf{Pa}(X)$ of $\mathcal{X} \setminus \{X\}$, thus Pa defines a directed graph over $\mathcal{X}$, where there is an edge (X, Y) if and only if $X \in \mathsf{Pa}(Y)$. $\mathsf{Pa}(Y)$ is the set of *parents* of Y;
- CPT is a set of *conditional preference tables*, one table $\mathsf{CPT}(X)$ for every attribute X: $\mathsf{CPT}(X)$ contains, for every instantiation u of $\mathsf{Pa}(X)$, a *rule* $u : >$, where $>$ is a linear order over $\underline{X}$.

Example 1. *For the CP-net φ_0 of figure 1a, $\mathsf{Pa}(A) = \{\}$ and $\mathsf{CPT}(A) = \{a > \bar{a}\}$, $\mathsf{Pa}(B) = \{A, C\}$ and $\mathsf{CPT}(B) = \{a \vee \bar{c} : b > \bar{b}, \bar{a}c : \bar{b} > b\}$.*

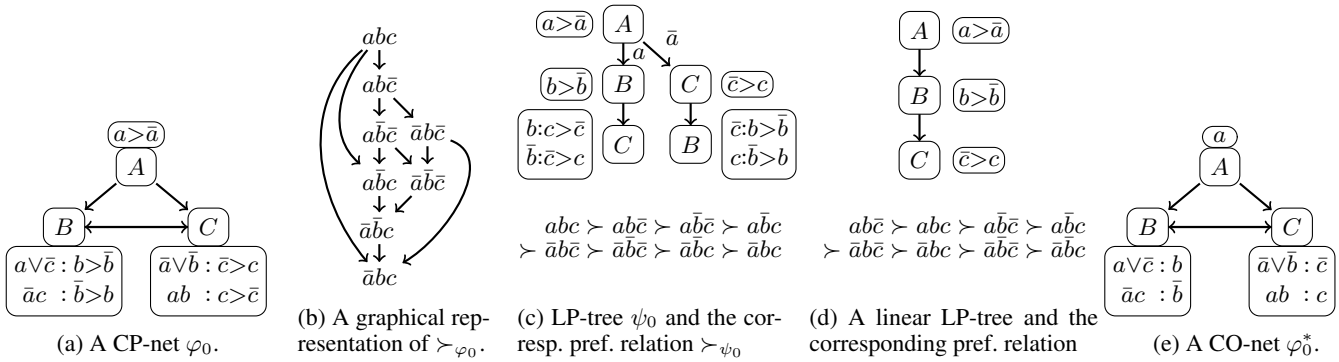
Let us call *swap* any pair of alternatives that have identical values for every attribute except one. A CP-net φ orders every swap $\{o, o'\}$ as follows: let X be the only attribute such that $o[X] \neq o'[X]$, let $u = o[\mathsf{Pa}(X)] = o'[\mathsf{Pa}(X)]$, let $u : >$ be the corresponding rule in $\mathsf{CPT}(X)$, then (o, o') is a *worsening swap* (w.r.t. φ) if and only if $o[X] > o'[X]$. The transitive closure of all the worsening swaps sanctioned by φ is, by definition, transitive, and we denote it by $\succ_\varphi$. It is not necessarily irreflexive, and not complete in general. Figure 1b depicts $\succ_{\varphi_0}$: edges $o \rightarrow o'$ represent the worsening swaps sanctioned by φ_0. Some of them are redundant since implied, by transitivity of $\succ_{\varphi_0}$, by other swaps: for instance the fact that $abc \succ_{\varphi_0} \bar{a}bc$ is implied by the worsening swaps $(abc, ab\bar{c})$, $(\bar{a}b\bar{c}, ab\bar{c})$, $(\bar{a}b\bar{c}, \bar{a}bc)$.

CP-net induced by a preference relation Given a preference relation (linear order) $\succ$ over $\underline{\mathcal{X}}$, it is possible to define a CP-net $\varphi_\succ = (\mathcal{X}, \mathsf{Pa}_\succ, \mathsf{CPT}_\succ)$ that captures the preferential independencies between attributes that are inherent in $\succ$: for every pair of attributes $X, Y \in \mathcal{X}$, $X \in \mathsf{Pa}_\succ(Y)$ if and only if Y is not preferentially independent of X (w.r.t. $\succ$). Then, for every $X \in \mathcal{X}$ and every $u \in \underline{\mathsf{Pa}_\succ(X)}$, $\mathsf{CPT}_\succ(X)$ contains the rule $u : >$, where $>$ is the linear order over $\underline{X}$ such that $x > x'$ if and only $xuv \succ x'uv$ for every $v \in \underline{\mathcal{X} - (\mathsf{Pa}_\succ(X) \cup \{X\})}$.

Example 2. *The CP-net induced by the linear order $abc \succ ab\bar{c} \succ a\bar{b}\bar{c} \succ a\bar{b}c \succ \bar{a}b\bar{c} \succ \bar{a}\bar{b}\bar{c} \succ \bar{a}\bar{b}c \succ \bar{a}bc$ is φ_0. For instance, whatever the values x and y given to B and C respectively, it holds that $axy \succ \bar{a}xy$, thus A does not preferentially depend on B, nor on C, and $\mathsf{CPT}(a) = \{a > \bar{a}\}$.*

Lexicographic Preference Trees LP-trees generalise lexicographic orders, which have been widely studied in decision making [10]. As an inference mechanism, they are equivalent to search trees used by [5], and formalised by [27, 28]. As a preference representation, and elicitation, language, slightly different definitions for LP-trees have been proposed by [2, 7, 9].

In the formal model proposed by [2], a *lexicographic preference tree*, or LP-tree for short, is composed of two parts: a rooted tree indicating the relative importance of the attributes, and tables indicating how to compare outcomes that agree on some attributes. An example of a LP-tree is depicted in Figure 1c. Each node of the importance tree is labelled with an attribute $X \in \mathcal{X}$, and is either a leaf of the tree, or has one single, unlabelled outgoing edge, or has



(a) A CP-net φ_0. (b) A graphical representation of $\succ_{\varphi_0}$. (c) LP-tree ψ_0 and the corresp. pref. relation $\succ_{\psi_0}$ (d) A linear LP-tree and the corresponding pref. relation (e) A CO-net φ_0^*.

Figure 1: Examples of a CP-net, two LP-trees and a CO-net.

$|\underline{X}|$ outgoing edges, each one being labelled with one of these values. No attribute can appear twice in a branch. For a given node N, $\mathsf{Anc}(N)$ denotes the set of attributes that label nodes above N. The values of attributes that are at a node above N with a labelled outgoing edge influence the preference at N. We denote by $\mathsf{Inst}(N)$ the set of nodes above N with a labelled outgoing edge and $\mathsf{inst}(N)$ the tuple of values of the edge labels between the root and N. Also, we define $\mathsf{NonInst}(N) = \mathsf{Anc}(N) \setminus \mathsf{Inst}(N)$.

Moreover, one *conditional preference table* $\mathsf{CPT}(N)$ is associated to each node N of the tree: if X is the attribute that labels N, then the table contains rules of the form $v : >$, where $>$ is a linear order $>$ over $\underline{X}$, and $v \in \underline{V}$ for some $V \subseteq \mathsf{NonInst}(N)$. The rules in $\mathsf{CPT}(N)$ must be consistent: given two rules $v : >$ and $v' : >'$, it must be the case that v and v' are not compatible. Moreover, we impose that LP-trees are *complete*: every attribute must appear exactly once on every branch, and for any $u \in \mathsf{NonInst}(N)$, there must be one rule $v : > \in \mathsf{CPT}(N)$ such that u and v are compatible.

Every LP-tree φ induces a linear order over $\underline{\mathcal{X}}$, denoted $\succ_\varphi$: for any node N labelled by X, consider a pair of alternatives o and o' such that $o[\mathsf{Inst}(N)] = o'[\mathsf{Inst}(N)] = \mathsf{inst}(N)$ and $o[X] \neq o'[X]$: N is said to *decide* the pair (o, o'); furthermore, there must be unique rule $v : >$ in $\mathsf{CPT}(N)$ such that $o[\mathsf{NonInst}(N)] = o'[\mathsf{NonInst}(N)]$ is compatible with v: then $o \succ_\varphi o'$ if and only if $o[X] > o'[X]$.

Figure 1c also shows the preference relation induced by the depicted LP-tree ψ_0, which is also that of example 2, so $\varphi_0 = \varphi_{\succ_{\psi_0}}$.

An LP-tree is said to have *unconditional preferences* if, for a given attribute X, every node labelled with X contains a unique, identical rule $\top : >$. We denote $\mathsf{UP\text{-}LPT}$ the class of such LP-trees. In particular, figure 1d depicts a *linear LP-tree*: it has unconditional preferences, and a single branch. It is a strong restriction on expressiveness: linear LP-trees represent the usual, unconditional lexicographic preference relations.

Brauning et al. [7, 6] extend the expressiveness of LP-trees by allowing to label a node with a set of attributes, considered as a single high-dimensional attribute; we do not consider such LP-trees here.

3 Conditional acyclicity and the Forward Sweep procedure

Boutilier et al. [4] proved that, when a CP-net φ is acyclic, it has a unique undominated alternative, which can be computed in polynomial time with the *Forward Sweep* procedure. We prove in this section that this approach characterises the class of *conditionally acyclic* CP-nets, introduced by [28], that strictly contains the class of those CP-nets, the graph of which is acyclic.

Definition ([28]). *CP-net φ is* conditionally acyclic *if there is some LP-tree ψ such that $\succ_\psi$ extends $\succ_\varphi$.*

The CP-net φ_0 of figure 1a is conditionally acyclic, since $\succ_{\varphi_0}$ is contained in $\succ_{\psi_0}$ where ψ_0 is the LP-tree of figure 1c. In fact, ψ_0 is the only LP-tree which has φ_0 as induced CP-net.

We introduce the following notation: given CP-net $\varphi = (\mathcal{X}, \mathsf{Pa}, \mathsf{CPT})$, for every attribute $X \in \mathcal{X}$ and every partial instantiation $u \in \underline{U}$ for some $U \subseteq \mathcal{X}$, $\mathsf{Pa}_\varphi(X|u)$ is the set of attributes of which X is not independent given u.

For the CP-net on figure 1a, although $\mathsf{Pa}(C) = \{A, B\}$, $\mathsf{Pa}(C|\bar{a}) = \emptyset$: whatever the value of B, when $A = \bar{a}$, the ordering over $\underline{C}$ is: $\bar{c} \geq c$.

Algorithm 1 below is a generic Forward Sweep algorithm. It initialises an empty instantiation inst at line 1, and iteratively expands inst by choosing an attribute at line 3, choosing a value for that attribute at line 4, and adding this value to inst. $\mathsf{FS}^{\mathsf{gen}}$ is non-deterministic, as it leaves open the choices of attribute and value at every iteration.

Algorithm 1: Generic Forward Sweep ($\mathsf{FS}^{\mathsf{gen}}$) from [4]

Data: φ, a CP-net
Result: $o \in \underline{\mathcal{X}}$
1 $\mathsf{inst} \leftarrow \top$
2 **while** *possible* **do**
3 choose $X \in \mathcal{X} - \mathsf{Var}(\mathsf{inst})$, s.t. $\mathsf{Pa}_\varphi(X|\mathsf{inst}) = \emptyset$
4 choose $x \in \underline{X}$
5 $\mathsf{inst} \leftarrow \mathsf{inst} \cdot x$
6 **if** *Var(inst)* $= \mathcal{X}$ **then return** inst;
7 **else return** FAILURE;

We will adapt this algorithm in later sections to perform various tasks of interest, and in particular to compute the undominated alternative of some conditionally acyclic CP-net as the Forward Sweep procedure proposed by [4].

We close this section with a proposition that shows that algorithm $\mathsf{FS}^{\mathsf{gen}}$ characterises the class of conditionally acyclic CP-nets.

Proposition 1. *Algorithm $\mathsf{FS}^{\mathsf{gen}}$ always succeeds, whatever the choices of variables or values at lines 3 and 5, if and only if φ is conditionally acyclic.*

In particular, if the input CP-net φ is not conditionally acyclic, there is at least one sequence of choices at lines 3 and 4 that will lead to failure: at some point, there is no more $X \in \mathcal{X} - \mathsf{Var}(\mathsf{inst})$,

s.t. $\mathsf{Pa}_\varphi(X|\mathsf{inst}) = \emptyset$. Note that the algorithm is not intended to be a practical tool to check if a given CP-net is conditionally acyclic, as checking that the algorithm never fails requires a number of trials that is exponential in the n: this problem is PSPACE-complete [28].

Proof. [28] proves that a CP-net φ is equivalent to a set of *CP-statements* that essentially corresponds to the union of the local preference rules in the conditional preference tables in φ for all attributes. [8] gives an algorithm that computes a complete LP-tree that is consistent with a given conditionally acyclic sets of CP-statements. The algorithm is as follows: [2]

Algorithm 2: Build complete LP tree (from [8])

Data: φ set of CP-statements
Result: LP-tree ψ complete, s.t. $\succeq_\psi \supseteq \succeq_\varphi$, or FAILURE
1 $\psi \leftarrow$ {an unlabelled root node}
2 **while** ψ *contains some unlabelled node* **do**
3 choose unlabelled node N of ψ
4 $(X, >) \leftarrow \mathsf{chooseAttribute}(N, \varphi)$
5 **if** $X = \mathit{FAILURE}$ **then** return FAILURE;
6 label N with $(X, >)$
7 **if** $\mathit{Anc}(N) \cup X \neq \mathcal{X}$ **then**
8 **for** $x \in \underline{X}$ **do**
9 add new unlabelled node to ψ
10 attach this node to N with edge labelled with x
11 **return** ψ

Essentially, given a yet unlabelled node N, the algorithm calls at step 4 the helper function chooseAttribute that returns an attribute X and a linear order over $>$ over $\underline{X}$, and expands the tree at step 8 with a child for N for every value in $\underline{X}$. [8] prove that the algorithm succeeds if and only if φ is conditionally acyclic.

The reader is referred to [8] for details about the helper function chooseAttribute in the context of sets of CP-statements. It is not difficult to see that in the case of CP-nets, this condition amounts to picking an attribute X such that all rules $u : > \in \mathsf{CPT}_\varphi(X)$, where u is consistent with the instantiations made above N, specify the same order over $\underline{X}$; that is, this condition amounts to the fact that $\mathsf{Pa}(X|\mathsf{inst}) = \emptyset$.

Thus line 4 in the algorithm above is similar to lines 3 and 4 in $\mathsf{FS}^{\mathsf{gen}}$, except that in $\mathsf{FS}^{\mathsf{gen}}$ a single value x is associated to X at line 4, instead of a linear order $>$ at line 4 in the above algorithm.

Moreover, whereas the above algorithm expands the current node at line 8 with one subtree for every possible value of the chosen attribute X, $\mathsf{FS}^{\mathsf{gen}}$ only expands inst at line 5 in a unique way with the chosen value x.

Therefore, building a tree with the above algorithm amounts to several runs of the $\mathsf{FS}^{\mathsf{gen}}$ algorithm to build all branches in parallel. Thus, $\mathsf{FS}^{\mathsf{gen}}$ always succeeds, for all possible choices at lines 2 and 5, if and only if the above algorithm succeeds, if and only if φ is conditionally acyclic. □

4 Conditional Optimality Networks

As far as optimisation is concerned, it turns out that the only information that is needed, in a conditionally acyclic CP-net, is the optimal values in the preference tables. Taking that into account, in this section, we define a "lightweight" version of CP-nets, that we call *Conditional Optimality Networks*, or CO-nets for short. They are similar to CP-nets, but only contains information about optimality. We will prove that this information is sufficient to reason about a given linear order, provided that the induced CO-net is conditionally acyclic.

Definition. *A CO-net over $\mathcal{X}$ is a tuple $N = (\mathcal{X}, \mathsf{Pa}, \mathit{COT})$, where Pa defines a directed graph over $\mathcal{X}$, and where COT is a conditional optimality table, such that, for every attribute X, and every $u \in \mathsf{Pa}(X)$, $\mathit{COT}(X, u)$ contains a single values of $\underline{X}$, the* optimal value *for X given u.*

For example, figure 1e shows the CO-net induced by the LP-tree ψ_0. Given a preference relation $\succ$ over $\mathcal{X}$, there is a unique CO-net that captures the information about conditional optimality contained in $\succ$, let us denote it by $\varphi^*_\succ = (\mathcal{X}, \mathsf{Pa}, \mathsf{COT})$: $X \in \mathsf{Pa}(Y)$ if Y is not independent of X for optimality, according to the next definition, and COT in $\varphi^*_\succ$ contains, for every attribute $Y \in \mathcal{X}$ and every $u \in \mathsf{Pa}(Y)$, the only undominated value of $\underline{Y}$, given u.

Definition. *Attribute Y is said to be* independent for optimality *of attribute $X \neq Y$ given $u \in \underline{U}$ for some $U \subseteq \mathcal{X} \setminus \{X, Y\}$ with respect to preference relation $\succ$ if for every $x, x' \in \underline{X}$, $y \in \underline{Y}$, $v \in \mathcal{X} \setminus \{X, Y\}$, $uvxy$ is uvx-undominated if and only if $uvxy'$ is uvx'-undominated. We write that Y is independent of X for optimality if Y is independent of X for optimality given the empty assignment $u = \top$.*

Note that, given a preference relation $\succ$, preferential independence implies independence for optimality. Therefore, the graph of the CP-net induced by $\succ$ contains the graph of the CO-net induced by $\succ$; furthermore, since, in the induced CP-net, the CPTs contain linear orders over domains of the variables, they also contain the information about optimal values. Thus the induced CO-net is weaker than the induced CP-net. Obviously, in the case where all variables are binary, the induced CO-net is equivalent to the induced CP-net.

We now consider an instantiation of the Forward Sweep algorithm that specifically computes the only undominated completion of a partial instantiation u, given a conditionally acyclic CP-net or CO-net.

Algorithm 3: Forward Sweep for optimisation ($\mathsf{FS}^{\mathsf{opt}}$)

Data: φ = CP-net or CO-net, $U \subseteq \mathcal{X}$, $u \in \underline{U}$
Result: $\mathit{opt}(u, \varphi)$
1 inst $\leftarrow \top$
2 **while** *possible* **do**
3 choose $X \in \mathcal{X} - \mathsf{Var}(\mathsf{inst})$, s.t. $\mathsf{Pa}_\varphi(X|\mathsf{inst}) = \emptyset$
4 **if** $X \in U$ **then** inst $\leftarrow$ inst $\cdot\, u[X]$;
5 **else** inst $\leftarrow$ inst $\cdot\, \mathsf{opt}(\underline{X}|\mathsf{inst})$;
6 **if** $\mathit{Var}(\mathit{inst}) = \mathcal{X}$ **then return** inst;
7 **else return** FAILURE;

Definition. *We say that a CO-net is* conditionally acyclic *if the $\mathsf{FS}^{\mathsf{opt}}$ always succeeds.*

The next proposition shows that in the case where the CO-net induced by some preference relation is conditionally acyclic, then it contains all necessary information to compute $\mathsf{opt}(\cdot, \succ)$.

Proposition 2. *Let $\succ$ be a preference relation over $\mathcal{X}$. If the induced CO-net, $\varphi^*_\succ$, is conditionally acyclic, then, for every $u \in \underline{U}$ for every $U \subseteq \mathcal{X}$, $\mathsf{opt}(u, \succ) = \mathsf{FS}^{\mathsf{opt}}(u, \varphi^*_\succ)$.*

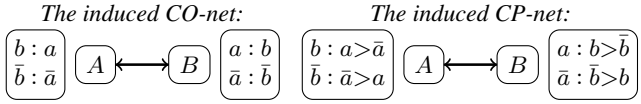
[2] The algorithm proposed by [8] is more general in that it takes another input which is a bound on the number of attributes allowed at each node of the tree; the algorithm that we give here is a restriction where this bound is set to 1, that, we build LP-trees with exactly one attribute at each node.

Proof. The proof is similar to that of lemma 3 in [4]. Let $U \subseteq \mathcal{X}$, $u \in \underline{U}$. By definition of conditional acyclicity, we know that $\mathsf{FS}^{\mathsf{opt}}$ always succeeds; let $o^*(u)$ be the alternative returned by the algorithm when called with (U, u). Let X_i be the attribute chosen at the ith iteration, x_i^* the value taken by the variable X_i at the ith iteration, and inst_i the value of inst after the ith iteration. We have $\mathsf{inst}_1 = x_1^*$, $\mathsf{inst}_2 = x_1^* x_2^*$, ..., and $\mathsf{inst}_n = o^*(u)$. Let o be another alternative such that $o[U] = u$. Define sequence of alternatives $(o_i)_{i=n,\dots,1}$ as follows: $o_n = o^*(u)$, and for every $i = n, n-1, \dots, 1$: $o_{i-1} = o_i$ except if $X_i \notin U$, in which case o_{i-1} is identical to o_i except that $o_{i-1}[X_i] = o[X_i]$. Then $o_0 = o$. Moreover, for every $i = n, n-1, \dots, 1$, since $o^*[X_i] = x_i$ is the optimal value for X_i given inst_{i-1}, we have that $o^*[X_i] > o[X_i]$ and $o_i \succ o_{i-1}$, or $o^*[X_i] = o[X_i]$ and $o_i = o_{i-1}$; because $o \neq o^*(u)$, for at least one i it must be the case that $o_i \succ o_{i-1}$. Thus $o^*(u) \succ o_0$. □

The important point here is that, if one wants to elicitate or learn preferences to support a decision maker in tasks that only involve optimisation queries, then all what is needed is a conditionally acyclic CO-net. Note that the above result does not hold anymore for non-conditionally acyclic CO-nets, as shown on the next example.

Example 3. *Consider two binary attributes A and B, and the following preference relation: $ab \succ \bar{a}\bar{b} \succ a\bar{b} \succ \bar{a}b$, with one undominated alternative ab. The induced CO-net, depicted below, is not conditionally ayclic; the partial order $\succ_\varphi$ defined by the induced CP-net φ has two undominated alternatives : ab and $\bar{a}\bar{b}$.*

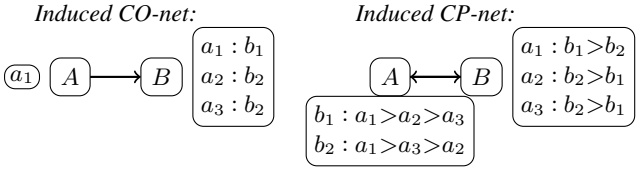
The induced CO-net: *The induced CP-net:*



Besides, since the independence for optimality is weaker than the preferential independence, it is possible for cyclic CP-nets to have associated conditionally acyclic CO-nets. Remark that this can only happen if there exist non-binary attributes.

Example 4. *Consider two attributes A, B with $\underline{A} = \{a_1, a_2, a_3\}$, $\underline{B} = \{b_1, b_2\}$, and the following preference relation: $a_1b_1 \succ a_1b_2 \succ a_3b_2 \succ a_2b_2 \succ a_2b_1 \succ a_3b_1$. The induced CP-net is not conditionally acyclic but the induced CO-net is acyclic.*

Induced CO-net: *Induced CP-net:*



5 Expressiveness

In this section, we explore the relationship between some classes of CO-nets and the classes of preference relations that induces them.

Separable preferences We start with some results about *preferential separability*. A preference relation is said to be *separable* (resp. *opt-separable*) if for every pair of attributes (X, Y), X is preferentially independent of Y (resp. independent of Y for optimality).

By definition of CP-nets (resp. CO-nets), a preference relation is separable (resp. opt-separable) if and only if the induced CP-net (resp. CO-net) has no edge. Moreover, the preference relation defined by an LP-tree ψ is separable if and only $\psi \in \mathsf{UP\text{-}LPT}$; in other words: the preference relation defined by ψ is separable if and only if ψ has unconditional rules. Conversely, given a CP-net or a CO-net φ with no edge, there are $n!$ linear LP-trees that induce φ (defined by the possible orderings of the attributes along the single branch of linear LP-trees), and even more LP-trees with unconditional preferences but conditional importance (that is, LP-trees with several branches). Note that there are also preference relations that induce φ and that are not represented by any LP-tree.

In settings where one needs to learn the preferences of a decision maker for the sole purpose of optimisation, and if it can be assumed that the decision-maker's preferences are separable, the above remark, combined with that of Prop. 2 formalises the unsurprising fact that one only has to learn the optimal value for every attribute, independently from the values of the other attributes.

Importantly, it should be noted that this does not extend to settings where one would need to compare two alternatives: even if the unknown preference relation is separable, comparing two alternatives may require information that is *not* captured by the induced CP-net.

Unconditional importance It has long been recognised that CP-nets alone cannot capture all the information needed to represent most preference relations. LP-trees on the other hand completely represent some preference relations, with some information about the relative *importance* of the attributes: in a given branch, every attribute is more important than all attributes below it in that branch, for comparing alternatives that are decided in that branch. Figure 1c shows a LP-tree where the relative importance of attributes B and C are conditioned by the value for attribute A: for comparing alternatives that have value a for A, B is more important than C; but C is more important than B for comparing alternatives that have value $\bar{a}$ for A. Let $\mathsf{UI\text{-}LPT}$ denote the class of LP-trees with unconditional importance, that is, where the ordering of the attributes is the same in every branch; such an LP-tree is always equivalent to an LP-tree with a single branch.

Proposition 3. *Given $\psi \in \mathsf{UI\text{-}LPT}$, the CP-net and CO-net induced by $\succ_\psi$ are acyclic; and, given an acyclic CP-net or CO-net φ, there exists some $\psi \in \mathsf{UI\text{-}LPT}$ such that φ is induced by $\succ_\psi$.*

Proof. Consider some LP-tree $\psi \in \mathsf{UI\text{-}LPT}$ with unconditional order of importance $X_1, X_2, \dots, X_n$: in every branch, the node at depth i is labelled with attribute X_i, and the preference rules at that node can only depend on $X_1, \dots, X_{i-1}$; thus in the induced CP-net / CO-net, $\mathsf{Pa}(X_i) \subseteq \{X_1, \dots, X_{i-1}\}$. Therefore the graph is acyclic. Conversely, given an acyclic CP-net φ, consider any topological ordering $X_1, \dots, X_n$ of the attributes, an LP-tree ψ such that $\succ_\psi$ induces φ can be built that has a single branch, with the nodes labelled from root to bottom with the attributes with in order $X_1, \dots, X_n$, and with the same CPT at node labelled with X_i in ψ as at node labelled with X_i in φ. If φ is a CO-net, an LP-tree can be constructed in the same way, but one must also complete the preference table at node labelled with X_i with any local orderings over $\underline{X_i}$ that have the correct optimal value, given the values of the parents of X_i in φ. □

Here again, given an acyclic CP-net / CO-net φ, there are preference relations that induce φ but cannot be represented by any LP-tree. However, the result above shows that in settings where one needs to learn the preferences of a decision maker for the sole purpose of optimisation, and if it can be assumed that the decision-maker's preferences can be represented with an LP-tree with unconditional importance, one can safely search for an acyclic CO-net. This is important from the point of view of machine learning, since it puts a strong bias on the search space, and also limits the amount

of information that must be induced. For example, for n binary attributes, there are 2^n different CO/CP-nets but $2^n \times n!$ linear LP-trees, and, in a conditionally acyclic CO/CP-nets, there are n nodes and at most $O(n^2)$ edges, while in a LP-tree there are at most $O(2^n)$ nodes and edges.

6 Forward Sweep for encoding / decoding data

Learning graphical models of preferences has mostly been attempted in settings where the input data is a set of pairwise comparisons, that is, pairs of alternatives (o, o') where o is deemed preferred to o'. In such settings, one attempts to learn a model φ such that $o \succ_\varphi o'$: this leads to an empirical loss function that counts the number of "misordered" pairs in the input data. In the case of CP-nets, checking if $o \succ_\varphi o'$ holds is an NP-complete problem. Besides, and CP-nets do not define a total relation, thus this empirical loss function is ill-defined.

In order to alleviate these problems, works on learning CP-nets often put some restrictions on the graphical structure of the CP-nets, like a bound on the number of parents of each attribute [21, 16, 1, e.g.], and/or restrict the input to some simple types of pairwise comparisons [18, 19, e.g.]. Learning LP-trees is easier [2, 7, 6, 22], but at the cost of a significant loss in expressiveness.

Fargier et al. [9] propose to learn a preference relation from a different kind of data: a set of alternatives that have been chosen by users of some decision-aid system. The idea is that the commoner a value in this set of chosen alternatives, the more it is likely to characterise the preferred alternative(s). We now show how this idea can be combined with the *Minimal Description Length* induction principle to enable a promising new way of learning CO-nets.

The idea of the MDL principle for machine learning is that, given some data D and a class of possible models that may "explain" D, one should choose the model H that enables the lossless compression of D with minimum size [15]. Formally, if $L(D|H)$ denotes the length of the representation that permits to retrieve D knowing H, one can define the minimum description length for D given a class of models $\mathcal{H}$ as $L(D) = \min_{H\in\mathcal{H}} \big(L(H) + L(D|H)\big)$. MDL has been successfully applied in the unsupervised learning of many classes of models, such as Bayesian networks [25, 20], causal networks [23], formal grammars [13], and applied to data mining in graphs [26].

When H is a CO-net, the size of H is simply the sum of the size of the graph and of the size of the conditional optimality table:

$$L(H) = L_{\mathbb{N}}(|\mathcal{X}|) + \sum_{N\in\mathcal{X}} \left(L_{\mathbb{N}}(|Pa(N)|) + \log_2 \binom{|\mathcal{X}|-1}{|Pa(N)|} + |\underline{Pa(N)}| \log_2 |\underline{N}| \right) \quad (1)$$

where $L_{\mathbb{N}}$ is the length of the Rissanen universal integer encoding [24], defined as $L_{\mathbb{N}}(n) = \log^*(1+n) + \log c_0$ where $\log^*$ is the expansion $\log n + \log\log n + \ldots$, including only the positive terms, and c_0 is a constant, and where, slightly abusing notation, $|\underline{N}|$ denotes the domain size of the attribute labelling N. These terms encode, in order: the total number of nodes, and for each node, the number of its parents, its set of parents and the optimal value for each value of its parents.

As a preliminary step towards the application of the MDL principle to learn CO-nets, we propose in the remainder of this section a simple way of "coding" alternatives, given a CO-net. The approach makes use of the optimisation query. Given a strict partial order $\succ$, consider an alternative o, and a partial instantiation u such that

Algorithm 4: Forward Sweep for encoding ($\mathsf{FS}^{\mathsf{enc}}$)

Data: φ = CO-net, $o \in \underline{\mathcal{X}}$
Result: $code(o, \varphi)$
inst $\leftarrow \top$; code $\leftarrow \top$
while *possible* **do**
  choose $X \in \mathcal{X} - \mathsf{Var}(\mathsf{inst})$, s.t. $\mathsf{Pa}_\varphi(X|\mathsf{inst}) = \emptyset$
  if $o[X] \neq \mathit{opt}(X|\mathit{inst})$ **then** code $\leftarrow$ code $\cdot\, o[X]$;
  inst $\leftarrow$ inst $\cdot\, o[X]$
if $\mathit{Var}(\mathit{inst}) = \mathcal{X}$ **then return** code;
else return FAILURE;

$opt(u, \succ) = o$: u, being a partial instantiation, is shorter than the alternative o, so it can be seen as short code for o – if there is a practical algorithm to retrieve o from u, which is the case with conditionally acyclic CO-nets since we know from proposition 2 that we can compute $opt(u, \succ)$ with the $\mathsf{FS}^{\mathsf{opt}}$ algorithm. Given o, there will be in general several partial instantiations u such that $opt(u, \succ) = o$, but if we can uniquely define one such partial instantation for every o, then we have a way of encoding alternatives.

In the following, opt^{-1} denotes the inverse function of opt: given an alternative o, $opt^{-1}(o, \succ) = \{u | opt(u, \succ) = o\}$.

Definition. *Let $\succ$ be a strict partial order over $\underline{\mathcal{X}}$. Suppose that for every o, $opt^{-1}(o, \succ)$ contains a unique u with minimal size. Then we say that $\succ$ is* uniquely encoding, *and we define $\mathit{code}(o, \succ)$ to be this unique minimal u in $opt^{-1}(o, \succ)$.*

Example 5. *Consider again the preference that corresponds to the LP-tree of figure 1c: $abc \succ ab\bar{c} \succ \bar{a}b\bar{c} \succ a\bar{b}\bar{c} \succ \bar{a}\bar{b}\bar{c} \succ \bar{a}bc \succ \bar{a}\bar{b}c \succ \bar{a}bc$. Then $opt(\bar{a}b, \succ) = \bar{a}b\bar{c}$, because it is the most preferred alternative compatible with $\bar{a}b$. In fact, $opt^{-1}(\bar{a}b\bar{c}, \succ) = \{\bar{a}, \bar{a}b, \bar{a}\bar{c}, \bar{a}b\bar{c}\}$. Therefore, $\mathit{code}(\bar{a}b\bar{c}, \succ) = \bar{a}$. It can be checked that $\succ$ is uniquely encoding.*

The linear order $ab \succ \bar{a}\bar{b} \succ a\bar{b} \succ \bar{a}b$ of example 3 is not uniquely encoding: $opt^{-1}(\bar{a}\bar{b}) = \{\bar{a}\bar{b}, \bar{b}, \bar{a}\}$.

We already know that if a preference $\succ$ induces a conditionally acyclic CO-net, then the opt / decoding function can be computed with the $\mathsf{FS}^{\mathsf{opt}}$ algorithm. The main result of this section is that in this case, another instance of the Forward Sweep procedure, called $\mathsf{FS}^{\mathsf{enc}}$, and depicted in Algorithm 4, can be used to compute the code function. We illustrate it on an example.

Example 6. *Consider again the linear order that corresponds to the LP-tree ψ_0 of figure 1c, whose induced CO-net is depicted on figure 1e. Let $o = \bar{a}b\bar{c}$, and suppose we want to compute $\mathit{code}(\bar{a}b\bar{c}, \succ)$ with algorithm $\mathit{FS}^{\mathit{enc}}$. At the first iteration of the "while" loop, the only variable that has no parent is A, with optimal value $a \neq o[A] = \bar{a}$, thus $\mathit{code} \leftarrow \bar{a}$, and $\mathit{inst} \leftarrow \bar{a}$. At the next iteration, $\mathit{Pa}(C|\bar{a}) = \emptyset$, with optimal value $\bar{c} = o[C]$, so code is not updated, and $\mathit{inst} \leftarrow \bar{a}\bar{c}$. At the last iteration, the optimal value for B given inst is $b = o[B]$, thus the algorithm returns $\bar{a}$.*

Proposition 4. *If the CO-net induced by a given preference relation $\succ$ is conditionally acyclic, then $\succ$ is uniquely encoding, and the coding function can be computed with the $\mathit{FS}^{\mathit{enc}}$ algorithm in Algorithm 4.*

Proof. Let o be any alternative, and let u be the output of Algorithm 4 for o. Let us show that $opt(u, \succ) = o$. As shown earlier, $opt(u)$ can be computed with Algorithm 3. Remark that, in

Dataset	LZMA	PPMd	bzip2	DEFLATE	zstd	LZ4	zpaq	brotli	separable CO-net	CO-net
Small	95.80%	97.90%	97.46%	94.50%	96.22%	93.51%	94.46%	96.42%	92.19%	97.03%
Medium	96.04%	97.98%	97.71%	94.82%	96.45%	93.94%	95.21%	96.58%	91.21%	97.12%
Big	96.40%	97.93%	97.64%	94.90%	97.04%	94.29%	94.73%	97.23%	93.41%	97.67%

Table 1: Space savings for various compression algorithms on the three Renault datasets.

this decoding algorithm, the value of $u[X]$ only affects the output if $u[X] \neq \mathsf{opt}(\underline{X}|\mathsf{inst})$: otherwise, no matter whether u is defined or not on X, x will have the same value. In this regard, Algorithm 4 simply computes the subset of o that have an impact on the decoding, i.e., such that $o[X] \neq \mathsf{opt}(\underline{X}|\mathsf{o})$. Therefore, if we denote u the output of Algorithm 4 for o, then $opt(u, \succ) = o$.

Now, let us show that u (defined on U) is the unique minimal instance such that $opt(u) = o$. Let $v \neq u$ (defined on V) such that $opt(v) = o$. Consider the traces of Algorithm 3 for u and v. First, let us remark there exists an order of attributes selection that is compatible with both the optimisation of u and v. Indeed, since $opt(u) = opt(v)$, if we denote inst_u (resp. inst_v) the content of variable inst at any point of the execution of Algorithm 3 applied to u (resp. v), then $\mathsf{inst}_u \subseteq opt(u)$ and $\mathsf{inst}_v \subseteq opt(v)$. Therefore, inst_u and inst_v are always compatible. For this reason, the set of available attributes for X, that only depends on $\mathsf{Pa}_\varphi(X|\mathsf{inst})$, is the same for u and for v. Let us denote L such an attributes selection order. Let us now prove that $opt(u[V]) = opt(u)$ by comparing the execution of Algorithm 4 on u, v and $u[V]$, denoted T_u, T_v and $T_{u[V]}$, for this attributes selection order L. Let X_i be the variable chosen at iteration i. Since u and v lead to the same optimal alternative, it means that the same value of x_i is chosen at each iteration. Either $u[X_i] = v[X_i]$, and therefore $u[X_i] = u[V][X_i]$, so the same value x_i is chosen in $T_{u[V]}$ and the execution still have the same values of inst. If $u[X_i] \neq v[X_i]$, and because $u \sim v$, at least one of u or v is not defined on X_i. Let us assume (without loss of generality) that u is not defined on X_i. In that case, $u[V]$ is not defined on X_i either and $T_{u[V]}$ the same value for x_i as T_u. Therefore, the execution still have the same values of inst. By recurrence, since the values of inst are the same at each point of the executions, then $opt(u[V]) = opt(u)$. By assumption, u is minimal for cardinality. Therefore $U \subsetneq V$, so $u \subsetneq v$ and $|u| < |v|$, so u is the unique minimum for cardinality. □

We are now able to compute the length of the compression of data D, given some CO-net H that is uniquely encoding:

$$L(D \mid H) = \sum_{o \in D} \left(L_{\mathbb{N}}(|\mathsf{code}(o, H)|) + \log_2 \binom{|\mathcal{X}|}{|\mathsf{code}(o,H)|} + \sum_{x \in \mathsf{code}(o,H)} \log_2(|\underline{X}| - 1) \right) \quad (2)$$

For each outcome o of the dataset, the terms encode, in order: the length of the minimal code of o, the set of variables that are assigned in this code, and the value for each attribute.

Compression experiment While MDL is typically used for model selection and not compression, we propose to experimentally assess the relevance of CO-nets and separable CO-nets in the context of preference representations by comparing the length of the codes of several datasets with implementations of efficient compression algorithms configured for the highest compression ratio. Source code, data and CO-nets are available online[3].

[3] https://github.com/PFGimenez/co-net-ecai23

The datasets are real-world sales history of cars from the Renault car manufacturer: "Small" has 48 variables and is 2.7MB, "Medium" has 44 variables and is 1.4MB and "Big" has 87 variables and is 3.2MB. These files are in a csv format which is a text format. They can be easily compressed, as shown by the space savings in Table 1. The three algorithms with the highest space saving are PPMd, bzip2, and our encoding based on a CO-net. Separable CO-nets have the lowest space saving of all methods in this Table, due to they very limited expressivity, but they still achieve 90%+ space saving. Table 1 does not include snappy and LZ77 algorithms because of their poor compression efficiency: they obtained about 70% and 85% space saving, respectively.

While a direct comparison is unfair because MDL is a theoretical tool that does not need to comply with the technical constraints of file formats and compression speed, we can still conclude that CO-nets can represent the regularities of real-world datasets with an efficiency similar to the most efficient compression algorithms.

7 Conclusion

While ubiquitous in many industrial applications, the preferential optimisation query has been little studied on its own. By focusing on graphical preferences models with efficient (polytime) preferential optimisation, CP-nets, and LP-trees, we showed that these two popular models are, in fact, as expressive for this query, and that conditionally acyclic CP-nets are exactly the CP-nets where Forward Sweep can be applied. Besides, we proposed an even more compact graphical model class, the CO-nets, that can be used for optimisation even though they contain little information about the actual linear order of preferences.

The method proposed by [9] to learn LP-trees can only be applied to models representing total orders, where the rank is defined. LP-trees are useful models for preferences representation, but, as we demonstrated here, one does not need attribute importance when the optimisation query is the only query of interest. On the other hand, CP-nets are just as expressive and much more succinct. However, the learning approach of [9] is inapplicable to CP-nets because they do not represent total orders.

In Section 6, we detailed how Forward Sweep can be used for encoding and decoding alternatives for a given CO-net. Such procedures can be used with the MDL framework to learn CP-nets or CO-nets by minimizing the MDL score. In this context, Prop. 1 is especially important since it shows that conditionally acyclic CP-nets (resp. CO-nets) are exactly the CP-nets (resp. CO-nets) where the efficient Forward Sweep algorithm can be used for encoding and decoding. Polytime encoding and decoding is paramount for scalable MDL learning, generally based on local greedy search. This is a preliminary step towards unsupervised CP-nets and CO-nets learning with minimal description length (MDL) by adapting the polytime Forward Sweep algorithm to encoding and decoding.

Acknowledgements We thank anonymous reviewers for their valuable comments. This work has benefited from the AI Interdisci-

plinary Institute ANITI. ANITI is funded by the French "Investing for the Future – PIA3" program under grant agreement ANR-19-PI3A-0004. This work has also been supported by the PING/ACK project of the French National Agency for Research, grant agreement ANR-18-CE40-0011.

References

[1] Thomas E. Allen, Cory Siler, and Judy Goldsmith, 'Learning tree-structured cp-nets with local search', in *Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2017)*, eds., Vasile Rus and Zdravko Markov, pp. 8–13. AAAI Press, (2017).

[2] Richard Booth, Yann Chevaleyre, Jérôme Lang, Jérôme Mengin, and Chattrakul Sombattheera, 'Learning conditionally lexicographic preference relations', in *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, eds., Helder Coelho, Rudi Studer, and Michael Wooldridge, volume 215 of *Frontiers in Artificial Intelligence and Applications*, p. 269–274. IOS Press, (2010).

[3] Craig Boutilier, ed. *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, 2009.

[4] Craig Boutilier, Romen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole, 'CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements', *Journal of Artificial Intelligence Research*, **21**, 135–191, (2004).

[5] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole, 'Preference-based constrained optimization with cp-nets', *Computational Intelligence*, **20**(2), 137–157, (2004).

[6] Michael Bräuning, Eyke Hüllermeier, Tobias Keller, and Martin Glaum, 'Lexicographic preferences for predictive modeling of human decision making: A new machine learning method with an application in accounting', *European Journal of Operational Research*, **258**(1), 295–306, (2017).

[7] Michael Bräuning and Eyke Hüllermeyer, 'Learning conditional lexicographic preference trees', in *Preference Learning: Problems and Applications in AI. Proceedings of the ECAI 2012 workshop*, eds., Johannes Fürnkranz and Eyke Hüllermeyer, p. 11–15, (2012).

[8] Hélène Fargier and Jérôme Mengin, 'A knowledge compilation map for conditional preference statements-based languages', in *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '21)*, eds., Frank Dignum, Alessio Lomuscio, Ulle Endriss, and Ann Nowé, pp. 492–500. ACM, (2021).

[9] Hélène Fargier, Pierre Francois Gimenez, and Jérôme Mengin, 'Learning lexicographic preference trees from positive examples', in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*, eds., Sheila A. McIlraith and Kilian Q. Weinberger, p. 2959–2966. AAAI Press, (2018).

[10] Peter C. Fishburn, 'Lexicographic orders, utilities and decision rules: A survey', *Management Science*, **20**(11), 1442–1471, (1974).

[11] Niall M Fraser, 'Applications of preference trees', in *Proceedings of SMC'93*, pp. 132–136, (1993).

[12] Yoav Freund, Raj D. Iyer, Robert E. Schapire, and Yoram Singer, 'An efficient boosting algorithm for combining preferences', *Journal of Machine Learning Research*, **4**, 933–969, (2003).

[13] Minos Garofalakis, Aristides Gionis, Rajeev Rastogi, Sridhar Seshadri, and Kyuseok Shim, 'Xtract: learning document type descriptors from xml document collections', *Data mining and knowledge discovery*, **7**, 23–56, (2003).

[14] Christophe Gonzales and Patrice Perny, 'GAI networks for utility elicitation', in *Proceedings of KR'04*, pp. 224–234, (2004).

[15] Peter Grünwald, 'Model selection based on minimum description length', *Journal of mathematical psychology*, **44**(1), 133–152, (2000).

[16] Joshua T. Guerin, Thomas E. Allen, and Judy Goldsmith, 'Learning cp-net preferences online from user queries', in *Proceedings of the Third International Conference on Algorithmic Decision Theory(ADT 2013)*, eds., Patrice Perny, Marc Pirlot, and Alexis Tsoukiàs, volume 8176 of *Lecture Notes in Computer Science*, pp. 208–220. Springer, (2013).

[17] Cynthia Huffman and Barbara E Kahn, 'Variety for sale: mass customization or mass confusion?', *Journal of retailing*, **74**(4), 491–513, (1998).

[18] Frédéric Koriche and Bruno Zanuttini, 'Learning conditional preference networks', *Artificial Intelligence*, **174**(11), 685–703, (2010).

[19] Fabien Labernia, Bruno Zanuttini, Brice Mayag, Florian Yger, and Jamal Atif, 'Online learning of acyclic conditional preference networks from noisy data', in *IEEE International Conference on Data Mining (ICDM 2017)*, eds., Vijay Raghavan, Srinivas Aluru, George Karypis, Lucio Miele, and Xindong Wu, pp. 247–256. IEEE Computer Society, (2017).

[20] Wai Lam and Fahiem Bacchus, 'Learning bayesian belief networks: An approach based on the mdl principle', *Computational intelligence*, **10**(3), 269–293, (1994).

[21] Jérôme Lang and Jérôme Mengin, 'The complexity of learning separable ceteris paribus preferences', In Boutilier [3], pp. 848–853.

[22] Xudong Liu and Miroslaw Truszczynski, 'Learning partial lexicographic preference trees over combinatorial domains', in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015)*, eds., Blai Bonet and Sven Koenig, pp. 1539–1545. AAAI Press, (2015).

[23] Osman A Mian, Alexander Marx, and Jilles Vreeken, 'Discovering fully oriented causal networks', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8975–8982, (2021).

[24] Jorma Rissanen, 'A universal prior for integers and estimation by minimum description length', *The Annals of statistics*, **11**(2), 416–431, (1983).

[25] Joe Suzuki, 'A construction of bayesian networks from databases based on an mdl principle', in *Uncertainty in Artificial Intelligence*, pp. 266–273. Elsevier, (1993).

[26] Bianca Wackersreuther, Peter Wackersreuther, Annahita Oswald, Christian Böhm, and Karsten M Borgwardt, 'Frequent subgraph discovery in dynamic networks', in *Proceedings of the eighth workshop on mining and learning with graphs*, pp. 155–162, (2010).

[27] Nic Wilson, 'Consistency and constrained optimisation for conditional preferences', in *Proceedings of the 16th Eureopean Conference on Artificial Intelligence (ECAI 2004)*, eds., Ramón López de Mántaras and Lorenza Saitta, p. 888–892. IOS Press, (2004).

[28] Nic Wilson, 'Computational techniques for a simple theory of conditional preferences', *Artificial Intelligence*, **175**, 1053–1091, (2011).

[29] Linda L Zhang, 'Product configuration: a review of the state-of-the-art and future research', *International Journal of Production Research*, **52**(21), 6381–6398, (2014).