

Relational Networks of Conditional Preferences

Frédéric Koriche

LIRMM, Université Montpellier II, France
frederic.koriche@lirmm.fr

Abstract. Like relational probabilistic models, the need for relational preference models naturally arises in real-world applications involving multiple, heterogeneous, and richly interconnected objects. On the one hand, relational preferences should be represented into statements which are natural for human users to express. On the other hand, relational preference models should be endowed with a structure that supports tractable forms of reasoning and learning. This paper introduces the framework of *conditional preference relational networks (CPR-nets)*, that maintains the spirit of the popular “CP-nets” by expressing relational preferences in a natural way using the *ceteris paribus* semantics. We show that *acyclic* CPR-nets support tractable inference for optimization and ranking tasks. In addition, we show that in the online learning model, *tree-structured* CPR-nets (with bipartite orderings) are efficiently learnable from both optimization tasks and ranking tasks, using linear loss functions. Our results are corroborated with experiments on a large-scale movie recommendation dataset.

1 Introduction

A recurrent issue in AI is the development of intelligent agents capable of tailoring their actions and recommendations to the preferences of human users. The spectrum of applications that resort on this ability is extremely wide, ranging from adaptive interfaces and configuration softwares, to recommender systems and group decision-making [7]. In essence, the crucial ingredients for addressing this issue are *representation*, *reasoning* and *learning*. In complex domains, we need a representation that offers a compact encoding of preference relations defined over large outcome spaces. We also need to be able to use this representation effectively in order to answer a broad range of queries. And, since the performance of decision makers is dependent on their aptitude to reflect users’ preferences, we need to be able to predict and extract such preferences in an automatic way.

Among the different preference models that have been devised in the literature, *conditional preference networks (CP-nets)* have attracted a lot of attention by providing a compact and intuitive representation of qualitative preferences [2,4,8,14]. By analogy with Bayesian networks, CP-nets are graphical models in which nodes describe variables of interest and arcs capture preferential dependencies between variables. Each node is labeled with a table expressing the preference over alternative values of the node given different values of the parent nodes under a *ceteris paribus* (“all else being equal”) assumption. For example, in a CP-net for movie recommendation, the preference rule $\text{Genre} : \text{comedy} > \text{drama} \mid \text{Date} = \text{fifties}$ might state that, for a film released in the fifties I prefer a comedy to a drama, provided that all other properties are the same.

Despite their popularity, CP-nets are intrinsically limited to “attribute-value” domains. Many applications, however, are richly structured, involving objects of multiple types that are related to each other through a network of different types of relations. For example, movie recommender systems are usually defined over large databases involving various objects, such as movies, actors, directors, writers, critics and users, each entity being specified with its own attributes and related to others using appropriate types of references. Such applications pose new challenges for devising relational preference models endowed with expressive representations, efficient inference engines, and fast learning algorithms.

In this paper, we introduce the framework of *conditional preference relational networks (CPR-nets)* that extends the paradigm of ceteris paribus preferences to relational domains. Briefly, a CPR-net is a template over a relational schema which specifies a ground CP-net for each particular database of objects. Based on the ceteris paribus semantics, the representations provided by CPR-nets are *transparent*, in that a human expert can easily capture their meaning. For example, in a CPR-net for movie recommendation, the rule:

$$\text{Movie.Genre : action} > \text{drama} \leftarrow \text{mode}(\text{Movie.Audience.User.Age}) = \text{teen}$$

might capture the stereotype that, all other things being equal, action movies are preferable to dramas if the majority of people in the audience are teenagers.

A key feature of our framework is that the class of *acyclic* CPR-nets supports efficient inference for two sorts of reasoning tasks of practical interest. Namely, in an *outcome optimization task*, the decision maker is given an outcome in which several attributes are left unspecified; the goal is to find a maximally preferred extension of this outcome. For an acyclic CPR-net, such an extension can be found in linear time. In an *outcome ranking task*, the decision maker is given a set of outcomes, and the goal is to rank them in some non-decreasing order of preference. Again, such a ranking can be found in polynomial time using an acyclic CPR-net.

The learnability of CPR-nets is analyzed within the *online learning setting* [10] which has become the mainstream theoretical model for structured prediction. In this setting, the decision maker observes instances of a reasoning task in a sequential manner. On round t , after observing the t th instance, the decision maker attempts to predict the solution associated with this instance. The prediction is formed by a hypothesis chosen from a predefined class \mathcal{N} of CPR-nets. The decision maker can use this information to choose another hypothesis from the class \mathcal{N} before proceeding to the next round. As a common thread in online learning, we make no assumption regarding the sequence of instance-solution pairs. This setting is thus general enough to capture agnostic situations in which the “true” preference model is not necessarily an element of the predefined class \mathcal{N} .

To measure the performance of the decision maker, we consider two standard metrics. The first, called *regret*, measures the difference in cumulative loss between the decision maker and the optimal hypothesis in \mathcal{N} . The second metric is computational complexity, i.e. the amount of computer resources required to choose hypotheses and to predict solutions. Based on these metrics, we show that the class of tree-structured CPR-nets (with bipartite orderings) is *efficiently learnable* from both optimization tasks and ranking tasks, using linear loss functions. Our online learning algorithm is an extension of the *Hedge* algorithm [11] that exploits the Matrix-Tree Theorem [18] for generating directed spanning trees at random.

The paper is organized as follows. After introducing the syntax and semantics of CPR-nets in Section 2, the theoretical results concerning reasoning with acyclic CPR-nets and learning with tree-structured CPR-nets are presented in sections 3 and 4, respectively. In Section 5 we illustrate the learning potential of our framework with experiments on a large movie recommendation dataset. Finally, in Section 6, we compare our framework with related work and list some perspectives of further research. For the sake of clarity, detailed proofs are left in appendix.

2 Language and Semantics

On the surface, our representation for relational preferences is similar to a probabilistic relational model [13]: the representation is structured in a graphical way by exploiting conditional independencies. However, the nature of connections between nodes in the graph is different: whereas conditional probabilities are *quantitative* and specify a probability measure over the outcome space, conditional preferences are *qualitative* and specify a partial ordering between outcomes.

2.1 Language

The basic building block of our framework is an (object oriented relational) *schema* that specifies a database structure. In order to clarify dependencies among the attributes of interconnected objects, the schema is represented as a digraph \mathcal{S} .

The nodes of \mathcal{S} are separated into *class names* and *attribute names*. Intuitively, a class name denotes a type of objects, and an attribute name captures an elementary property that can be attached to a class name. Each attribute name A is associated with two predefined components: a finite domain $D(A)$ and an *aggregator* γ_A that maps any vector of values in $D(A)$ into a single value of $D(A)$. In what follows, we will omit the subscript A of γ when it is clear from the context. Common aggregators include the mode (most frequently occurring value), the maximum or minimum (if values are ordinals), and the mean (if values are cardinals). The *domain size* \mathcal{S} is given by the maximum of the sizes of its domains.

The arcs of \mathcal{S} capture functional constraints between nodes: they are separated into *attributes* and *references*. Specifically, an attribute is an arc of the form (X, A) , also denoted $X.A$, where X is a class name and A an attribute name. A reference is an arc of the form $X.Y$ where X and Y are class names. A *chain* is an undirected path in \mathcal{S} of the form $X.R$ where X is a class name and R is a (possibly empty) sequence of class names. A *term* is an expression of the form $X.R.A$ where $X.R$ is a chain and A is an attribute name connected to the last class name of $X.R$. The sets of all attributes and all terms in \mathcal{S} are denoted $\mathcal{A}(\mathcal{S})$ and $\mathcal{T}(\mathcal{S})$, respectively.

There is a natural correspondence between our representation and that of relational databases. Each class name X is associated with a table and each of its adjacent nodes is associated with a column in the table. For an attribute $X.A$, the entries in the corresponding column are values in $D(A)$, and for a reference $X.Y$, the entries are foreign keys, each identifying an object in Y . We note in passing that this representation does not prevent us from having complex relationships between entities: each k -ary relationship can be captured by introducing a new class name associated with k references, in which each object corresponds to a row in the relationship. These notions are clarified in the following example.

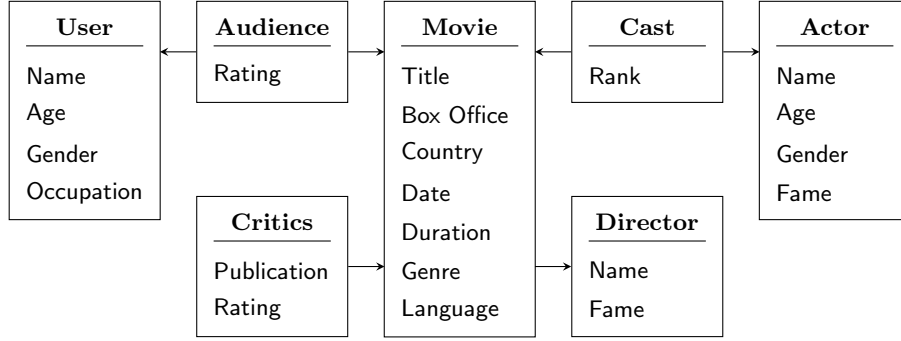


Fig. 1. A relational schema for the movie domain.

Example 1. Suppose we would like to design a movie recommender system that periodically suggests a list of movies to each subscriber. The relational schema, described in Figure 1, is composed of movies, actors, directors, critics, and users. Each box specifies a class name with its adjacent attribute names, and each arrow denotes a reference between class names. For instance, the term `User.Audience.Movie.Genre` refers to all genres of movies watched by a user.

A *preference ordering* over a domain D is an irreflexive, asymmetric and transitive relation $>$ on D . The *cover graph* of $>$ is the digraph for which the node set is D and the arc set is formed by all pairs $(v, v') \in D \times D$ where $v > v'$ and such that there is no intermediate value $v'' \in D$ satisfying $v > v'' > v'$. Note that the cover graph of any preference ordering is acyclic. A preference ordering is *bipartite* if its cover graph is bipartite.

Definition 1. For a set $\mathcal{A} \subseteq \mathcal{A}(S)$ and a set $\mathcal{T} \subseteq \mathcal{T}(S)$, a *conditional preference relational network (CPR-net)* is a pair $N = (par, cpt)$ such that:

- *par* associates a *parent set* to each attribute $X.A$ in \mathcal{A} ; the parent set $par(X.A)$ is a collection $\{X.R_1.A_1, \dots, X.R_p.A_p\}$ of terms in $\mathcal{T} \setminus \{X.A\}$ rooted at X .
- *cpt* associates a *conditional preference table* to each attribute $X.A$ in \mathcal{A} ; the table $cpt_{X.A}$ maps each vector \mathbf{u} in $D(A_1) \times \dots \times D(A_p)$ into a preference ordering $cpt_{X.A}(\mathbf{u})$ over $D(A)$.

By $\mathcal{N}[\mathcal{A}, \mathcal{T}]$ we denote the class of all CPR-nets defined over the set \mathcal{A} of attributes taking parents in the set \mathcal{T} of terms. Each attribute in \mathcal{A} is said to be *controllable*. For example, in a movie recommender system, it is legitimate to consider that movie attributes, including the genre, the release date and the duration of a film, are controllable. On the other hand, user attributes such as the age, the gender and the occupation of a person, are typically uncontrollable.

Given a CPR-net $N \in \mathcal{N}[\mathcal{A}, \mathcal{T}]$, the *dependency graph* of N is the digraph $\mathcal{G}(N)$ with node set \mathcal{A} and such that there is an arc from $X.A$ to $X'.A'$ if and only if $X'.A'$ is the suffix of some term in $par(X.A)$. A CPR-net N is *acyclic* if its dependency graph is acyclic. N is *tree-structured* if its dependency graph is a directed forest. Finally, N is *bipartite-ordered* if each of the entries of its preference tables is a bipartite preference ordering.

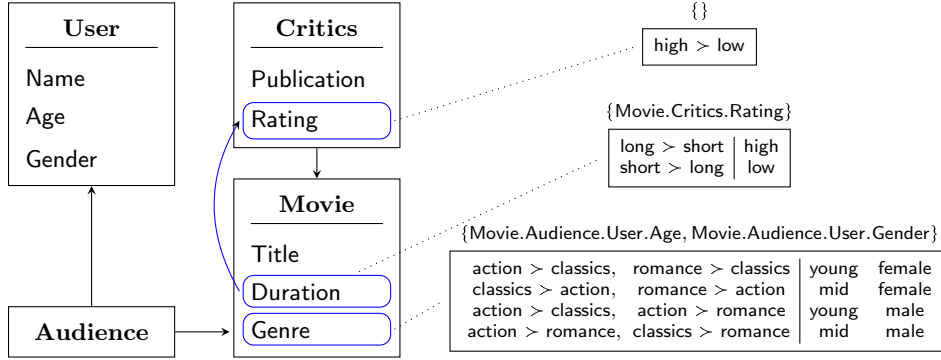


Fig. 2. A tree-structured CPR-net for the movie domain.

The *parent size* of N is the maximum number p of parents per attribute in N . It is important to keep in mind that the parent size of a CPR-net does not necessarily coincide with the in-degree of its dependency graph. In particular, a tree-structured CPR-net N can have parent sets composed of multiple terms, provided that at most one suffix is in \mathcal{A} . Because the size of conditional preference tables grows exponentially with the number of parents, we will assume that the parent size p of any CPR-net under consideration in this study is *constant*.

Example 2. Consider a restricted view of our movie recommendation domain, described in Figure 2. The CPR-net N is defined over the set \mathcal{A} of controllable attributes including Critics.Rating, Movie.Duration and Movie.Genre. The dependency graph of N is depicted in the left part of the figure, while the parent sets and preference tables of N are presented in the right part. We can observe that N is both tree-structured and bipartite-ordered. Notably, the parents of Movie.Genre are defined over uncontrollable attributes.

2.2 Semantics

Given a schema \mathcal{S} , a *skeleton* for \mathcal{S} is a map κ that assigns to each class name X a finite set $\llbracket X \rrbracket_\kappa$ of *objects*, and to each reference $X.Y$ a function $\llbracket X.Y \rrbracket_\kappa$ from $\llbracket X \rrbracket_\kappa$ into $\llbracket Y \rrbracket_\kappa$. Based on the standard semantics of inverse and composition operations, any skeleton determines a relation to any chain in the schema. A *ground chain* is an expression of the form $o.R$ where $X.R$ is a chain and o is an object in $\llbracket X \rrbracket_\kappa$. The notions of *ground term* and *ground attribute* are defined similarly. Given a ground chain $o.R$, we denote by $\llbracket o.R \rrbracket_\kappa$ the set of objects $\{o_1, \dots, o_n\}$ such that $(o, o_i) \in \llbracket X.R \rrbracket_\kappa$ for $1 \leq i \leq n$. For a collection of attributes \mathcal{A} , we denote by \mathcal{A}_κ the set of ground attributes $\{o.A : X.A \in \mathcal{A}, o \in \llbracket X \rrbracket_\kappa\}$.

An *interpretation* or *outcome* is a map I that extends a skeleton κ by assigning to each attribute $X.A$ a function $\llbracket X.A \rrbracket_I$ from $\llbracket X \rrbracket_\kappa$ into $D(A)$. By \mathcal{I}_κ , we denote the space formed by all interpretations extending κ . Given a ground attribute $o.A$, the value assigned by I to $o.A$ is denoted $\llbracket o.A \rrbracket_I$. More generally, given a ground term $o.R.A$ where $\llbracket o.R \rrbracket_\kappa = \{o_1, \dots, o_n\}$, we denote by $\llbracket o.R.A \rrbracket_I$ the vector \mathbf{v} in $D(A)^n$ such that $v_i = \llbracket o_i.A \rrbracket_I$ for $1 \leq i \leq n$.

With these notions in hand, we are now ready to examine the *ceteris paribus* semantics of relational preference networks. Consider a CPR-net N defined over a set of attributes \mathcal{A} , and a skeleton κ . A pair (I, J) of interpretations extending κ is called a *flip* on a ground attribute $o.A \in \mathcal{A}_\kappa$ if they are everywhere identical on \mathcal{A}_κ , excepted for $o.A$. Given an attribute $X.A$ with parent set $\{X.R_1.A_1, \dots, X.R_p.A_p\}$ and a flip I, J on $o.A$, we say that I *dominates* J in N if the value $\llbracket o.A \rrbracket_I$ is preferred to the value $\llbracket o.A \rrbracket_J$ in the entry of the table $cpt_{X.A}$ associated to the vector:

$$\llbracket par(o.A) \rrbracket_I = (\gamma(\llbracket o.R_1.A_1 \rrbracket_I), \dots, \gamma(\llbracket o.R_p.A_p \rrbracket_I))$$

By extension, given any pair (I, J) of interpretations in \mathcal{I}_κ , we say that I *dominates* J in N , and write $I \succ_N J$, if there is a sequence (I_1, \dots, I_n) of flips such that $I_1 = I$, $I_n = J$, and I_i dominates I_{i+1} in N_κ for $1 \leq i < n$. A CPR-net N is *coherent* if, for any skeleton κ , the relation \succ_N is a preference ordering over \mathcal{I}_κ .

Theorem 1. *Any acyclic CPR-net is coherent.*

Example 3. Consider two interpretations I_1 and I_2 for the schema given in Figure 2. Based on the tables specified below, we remark that (I_1, I_2) is a flip on the movie genre. Using the preference table of Movie.Genre it follows that $I_1 \succ_N I_2$.

		User			Critics			Movie		
		Name	Age	Gender	Pub.	Title	Rating	Title	Dur.	Genre
I_1		Mary	mid	female	IMDb	Big Fish	high	Big Fish	long	romance
I_2						GoodFellas		GoodFellas		action

3 Preference Reasoning

In the following, we will assume a prefixed and known schema \mathcal{S} of domain size d , in which any aggregator can be implemented by a procedure that runs in $\mathcal{O}(n \log d)$ time, where n is the maximum number of objects per class name assigned by a skeleton. For a class of CPR-nets, a *reasoning task* consists in a set of instances and a set of solutions. Of particular interest here is the class $\mathcal{N}_{\text{acy}}[\mathcal{A}, \mathcal{T}]$ of acyclic CPR-nets. The size of \mathcal{A} is denoted a , and the maximum of the sizes of terms in \mathcal{T} is denoted k . The next lemma states that the parent values of an attribute can be retrieved in quasi-linear time using standard join and projection operations.

Lemma 1. *Let N be a CPR-net in $\mathcal{N}_{\text{acy}}[\mathcal{A}, \mathcal{T}]$. Then, for any interpretation I and any ground attribute $o.A$, the vector of parent values $\llbracket par(o.A) \rrbracket_I$ can be constructed in $\mathcal{O}(t)$ time, where $t = kn \log_2(n) + n \log_2(d)$.*

3.1 Preference Optimization

A *partial outcome* is an interpretation I that assigns the value “*” (unknown) to some ground attributes. A *completion* of I with respect to a set of attributes \mathcal{A} , is a map that extends I by replacing the unknown value of each attribute in \mathcal{A}_κ with a value of appropriate type. J is *optimal* for a CPR-net N if there is no distinct completion J' of I such that $J' \succ_N J$. An *outcome optimization task* for $\mathcal{N}[\mathcal{A}, \mathcal{T}]$ is a reasoning task in which instances are partial outcomes and solutions are outcome completions with respect to \mathcal{A} . Given a CPR-net N and an instance I , the task is to find a completion J of I that is optimal for N .

For acyclic CPR-nets, such a completion can be found in polynomial time using the *forward sweep* algorithm [4], adapted to relational domains. Given a CPR-net N defined over an attribute set \mathcal{A} a partial outcome I , we first construct a linear extension of the ground attributes in the dependency graph $\mathcal{G}(N)$, where κ is the skeleton of I . Then, starting from $J = I$, we instantiate each $o.A \in \mathcal{A}_\kappa$ in turn to a root value in the preference ordering $\text{cpt}_{X.A}(\llbracket \text{par}(o.A) \rrbracket_I)$.

Theorem 2. *Let N be a CPR-net in $\mathcal{N}_{\text{acy}}[\mathcal{A}, \mathcal{T}]$. Then, for any partial outcome I , finding an optimal completion of I for N can be done in $\mathcal{O}(\text{ant})$ time.*

Example 4. Suppose that Ann is a young woman. Based on the tree-structured CPR-net in Figure 2, what would be her favorite romance movies? Starting from the partial outcome I in which only Ann’s attributes are known, we can derive two optimal completions J_1 and J_2 of I . For both completions, the value of Critics.Rating is set to high, and the value of Movie.Duration is set to long. The value of Movie.Genre is action for J_1 and romance for J_2 .

3.2 Preference Ranking

For a skeleton κ , an *outcome set* is any finite collection $S = \{I_1, \dots, I_m\}$ of interpretations in \mathcal{I}_κ . A *ranking* of S is a permutation π over $[m] = \{1, \dots, m\}$. The ranking π is *consistent* with a CPR-net N if, for any pair I_i, I_j in S , $I_i >_N I_j$ implies $\pi(i) > \pi(j)$. An *outcome ranking task* is a reasoning task for which any instance is an outcome set of size $m > 1$ and any solution is a permutation over m elements. Given a CPR-net N and an instance S , the problem is to find a ranking π of S that is consistent with N .

The outcome ranking problem can be solved in polynomial time for acyclic CPR-nets using a compilation technique inspired from [3,5]. For a skeleton κ , a *utility function* is a map $\phi : \mathcal{I}_\kappa \rightarrow \mathbb{R}$. Any utility function ϕ induces a preference ordering $>_\phi$ over \mathcal{I}_κ such that $I >_\phi J$ if and only if $\phi(I) > \phi(J)$. The function ϕ is *consistent* with a CPR-net N if $>_N$ implies $>_\phi$, that is, every linear extension of $>_\phi$ is a linear extension of $>_N$. Based on these notions, the overall idea of the compilation technique is to map any acyclic CPR-net N and any skeleton κ into a utility function ϕ over \mathcal{I}_κ that is consistent with N . The function ϕ can then be exploited for solving multiple ranking tasks defined over κ .

Theorem 3. *Let N be a CPR-net in $\mathcal{N}_{\text{acy}}[\mathcal{A}, \mathcal{T}]$ and κ be a skeleton. Then, compiling N into a consistent utility function ϕ over \mathcal{I}_κ can be done in $\mathcal{O}(\text{ad}^{p+1})$ time. Based on this utility function ϕ , finding a consistent ranking of any outcome set $S \subseteq \mathcal{I}_\kappa$ of size m can be done in $\mathcal{O}(\text{anmt})$ time.*

Example 5. Consider again the CPR-net N in Figure 2, and suppose that three long movies o_1 , o_2 and o_3 are presented to John, a middle-aged male user; o_1 and o_2 are romance movies, while o_3 is an action movie. The corresponding reviews, denoted c_1 , c_2 and c_3 , are positive for o_1 and o_3 , but quite negative for o_2 . We denote by I_1 (resp. I_2 and I_3) the interpretation associated to the entities John and $\{c_1, o_1\}$ (resp. $\{c_2, o_2\}$ and $\{c_3, o_3\}$). Now, according to the utility function ϕ of N , specified in the proof of Theorem 3 (see Appendix), we have $\phi(I_1) = 1 + 1/2 + 0 = 3/2$, $\phi(I_2) = 1/2$ and $\phi(I_3) = 5/2$. Therefore, the ranking $(3, 1, 2)$ is consistent with N .

4 Preference Learning

By extending our previous considerations, a *prediction class* for a class of CPR-nets \mathcal{N} is a triple $(\mathcal{X}, \mathcal{Y}, \ell)$, where \mathcal{X} is a space of instances, \mathcal{Y} is a space of solutions, and $\ell : \mathcal{N} \times \mathcal{X} \times \mathcal{Y} \rightarrow [0, \lambda]$ is a (bounded) *loss function*.

In the online learning model, the decision maker is a learning algorithm that observes instances of a prediction task in a sequence of rounds. At trial t , the algorithm receives an instance $x_t \in \mathcal{X}$ and is required to predict a corresponding solution $N_t(x_t) \in \mathcal{Y}$ using its current hypothesis $N_t \in \mathcal{N}$. Once the algorithm has predicted, the true solution $y_t \in \mathcal{Y}$ is revealed and the algorithm incurs the loss $\ell(N_t; x_t, y_t)$ that measures the discrepancy between the predicted solution $N_t(x_t)$ and the correct response y_t . In light of this information, the algorithm is allowed to choose a new hypothesis in \mathcal{N} , possibly using a randomized strategy.

The performance of the algorithm is measured relatively to the performance of the best hypothesis in \mathcal{N} . Namely, the *regret* of an online learning algorithm L with respect to a sequence $\{(x_t, y_t)\}$ of T examples is given by the difference between the expected cumulative loss of the algorithm and the cumulative loss of the best hypothesis chosen with the benefit of hindsight. Formally:

$$\text{regret}(L, \{(x_t, y_t)\}) = \mathbb{E} \left[\sum_t \ell(N_t(x_t), y_t) \right] - \min_{N \in \mathcal{N}} \sum_t \ell(N(x_t), y_t)$$

A class of hypotheses \mathcal{N} is *online learnable* with respect to a prediction task $(\mathcal{X}, \mathcal{Y}, \ell)$ if there exists an online learning algorithm L such that, for any sequence of T examples, the regret of L is *sublinear* as a function of T . This condition implies that “on the average” the algorithm performs as good as the best fixed hypothesis in hindsight. If, in addition, the computational complexity of L is polynomial in the dimension parameters associated to \mathcal{N} , \mathcal{X} , and \mathcal{Y} , then \mathcal{N} is *efficiently learnable*.

In this section, any CPR-net is viewed as a set of *components*, whose data structure will be clarified shortly. Let $\mathcal{C}(\mathcal{N})$ be the set of all distinct components generated from a class of hypotheses \mathcal{N} . A loss function ℓ is *linear* for \mathcal{N} if $\ell(N; x, y) = \sum_{C \in \mathcal{N}} \ell(C; x, y)$ for any $N \in \mathcal{N}$, where $\ell(C; x, y)$ denotes the loss incurred by the component C on the example (x, y) .

Our learning algorithm is a variant of the Hedge algorithm [11] adapted to structured models. Following [15], we call this algorithm *Expanded Hedge* (EH). As indicated in Figure 3, the algorithm maintains a parameter vector θ_t over $\mathcal{C}(\mathcal{N})$. On round t , the algorithm predicts with a hypothesis N_t chosen at random according to the exponential distribution \mathbb{P}_{θ_t} induced by θ_t . Then, θ_t is updated using the rule $\theta_{t+1}(C) = \theta_t(C) - \eta_t \ell(C, x_t, y_t)$, where η_t is an adaptive learning rate.

The following result can be derived by a simple adaptation of Hedge’s amortized analysis [10, Theorem 2.2].

Lemma 2. *Let \mathcal{N} be a finite class of CPR-nets, and $(\mathcal{X}, \mathcal{Y}, \ell)$ be a prediction task where ℓ is a λ -bounded linear loss function for \mathcal{N} . Then, for any sequence $\{(x_t, y_t)\}$ of T examples, the regret of the Expanded Hedge algorithm with adaptive learning rate $\eta_t = (2/\lambda)\sqrt{\ln(N)/t}$ satisfies:*

$$\text{regret}(L, \{(x_t, y_t)\}) \leq \lambda \sqrt{T \ln |\mathcal{N}|}$$

Fig. 3: The Expanded Hedge Algorithm

Input:

- a finite class \mathcal{N} of CPR-nets with its component set $\mathcal{C}(\mathcal{N})$,
- a prediction task $(\mathcal{X}, \mathcal{Y}, \ell)$, such that ℓ is linear for \mathcal{N}

Initialization: Set $\theta_1(C) = 0$ for each $C \in \mathcal{C}_{\mathcal{N}}$

Trials: for each round $t = 1, 2, \dots$

- (1) choose N_t according to the distribution $\mathbb{P}_{\theta_t}(N) \sim \exp[\sum_{C \in \mathcal{C}_{\mathcal{N}}} \theta_t(C)]$
 - (2) receive instance $x_t \in \mathcal{X}$
 - (3) predict solution $N_t(x_t) \in \mathcal{Y}$
 - (4) receive response $y_t \in \mathcal{Y}$
 - (5) choose η_t and set $\theta_{t+1}(C) = \theta_t(C) - \eta_t \ell(C, x_t, y_t)$ for each $C \in \mathcal{C}_{\mathcal{N}}$
-

The rest of this section is devoted to the learnability issue of the class $\mathcal{N}_{\text{tree}}$ of tree-structured and bipartite-ordered CPR-nets. To obtain a concrete online learning algorithm for this class, we must determine the components used to encode hypotheses and the linear loss function associated to each reasoning task.

4.1 Parameter Learning

For the sake of pedagogy, we first investigate the class $\mathcal{N}_{\text{tree}}[par]$ where the parent set par is prefixed and known. Here, a component is a triplet $C = (X.A, \mathbf{u}, v)$ where $X.A$ is an attribute with parent set $\{X_1.R_1, A_1, \dots, X_p, R_p, A_p\}$, \mathbf{u} is a vector over $D(A_1) \times \dots \times D(A_p)$ and v is a value in $D(A)$. Intuitively, C indicates that v is a maximally preferred value in the preference ordering of $cpt_{X.A}(\mathbf{u})$.

Given a schema of domain size d involving a attributes, the number of distinct components is bounded by ad^{p+1} , and hence, remains polynomial in a and d whenever the parent size p is taken as constant. By contrast, the cardinality of $\mathcal{N}_{\text{tree}}[par]$ is bounded by $(2^d - 1)^{ad^p}$, where $(2^d - 1)$ is the number of bipartite preference orderings over d values. Because this cardinality grows exponentially with a and d , a key computational issue is to generate hypotheses at random according to an exponential distribution over $\mathcal{N}_{\text{tree}}[par]$. Fortunately, this issue can be circumvented by exploiting a useful property of weighted bipartite graphs.

Lemma 3. *For the class $\mathcal{N}_{\text{tree}}[par]$, let θ be a parameter vector over the component set $\mathcal{C}(\mathcal{N}_{\text{tree}}[par])$. Then, generating a hypothesis $N \in \mathcal{N}_{\text{tree}}[par]$ at random according to \mathbb{P}_{θ} can be done in $\mathcal{O}(ad^{p+2})$ time.*

4.2 Structure Learning

Let us turn to the more general class $\mathcal{N}_{\text{tree}}[\mathcal{A}, \mathcal{T}]$ of preference trees with attribute set \mathcal{A} and term set \mathcal{T} . Since the parent structure is unknown, a component is a quadruplet $C = (X.A, par(X.A), \mathbf{u}, v)$, where $X.A$ is an attribute, $par(X.A)$ is a parent set, \mathbf{u} is a vector over the domain of $par(X.A)$ and v is a value in $D(A)$. For a schema of domain size d , the number of distinct components in $\mathcal{N}_{\text{tree}}[\mathcal{A}, \mathcal{T}]$ is bounded by $at^p d^{p+1}$, where $|\mathcal{A}| = a$ and $|\mathcal{T}| = t$. By contrast, the size of $\mathcal{N}_{\text{tree}}[\mathcal{A}, \mathcal{T}]$ is bounded by $a^{a-1} (t^p)^a (2^d - 1)^{ad^p}$, where a^{a-1} is the number of directed trees of order a . Again, this combinatorial barrier can be handled by exploiting a key property of spanning trees captured by the Matrix-Tree Theorem [18].

Lemma 4. For the class $\mathcal{N}_{\text{tree}}[\mathcal{A}, \mathcal{T}]$, let θ be a parameter vector over the component set $\mathcal{C}(\mathcal{N}_{\text{tree}}[\mathcal{A}, \mathcal{T}])$. Then, generating a hypothesis $N \in \mathcal{N}_{\text{tree}}[\mathcal{A}, \mathcal{T}]$ at random according to \mathbb{P}_θ can be done in $\mathcal{O}(a^5 + a^2 t^p d^p + a d^{p+2})$ time.

4.3 Applications

We now have all ingredients in hand to examine the learnability of tree-structured CPR-nets. The linear loss function ℓ_{opt} for optimization tasks is defined as follows: if x is a partial outcome I , y is a completion J of I , and C is a component of the form $(X.A, \text{par}(X.A), \mathbf{u}, v)$, then $\ell_{\text{opt}}(C; x, y) = 1$ if $\llbracket \text{par}(X.A) \rrbracket_J = \mathbf{u}$, $\llbracket X.A \rrbracket_I = *$ and $\llbracket X.A \rrbracket_J \neq v$; otherwise $\ell_{\text{opt}}(C; x, y) = 0$. Based on this feedback rule, we can observe that $\ell(N; x, y)$ is precisely the Hamming distance between the predicted completion $N(x)$ of x and the true completion y of x . Together the fact that ℓ_{opt} is bounded by $\lambda = a$, the composition of Lemmas 2 and 4 yields the following result.

Theorem 4. The class $\mathcal{N}_{\text{tree}}[\mathcal{A}, \mathcal{T}]$ is efficiently online learnable from outcome optimization tasks with regret bound $a^{3/2}(\ln a + p \ln t + d^{p+1})^{1/2} \sqrt{T}$.

The linear loss function ℓ_{rank} for ranking tasks is defined as follows: if x is an outcome set $\{I_1, \dots, I_m\}$, y is a permutation π over $[m]$, and C is a component of the form $(X.A, \text{par}(X.A), \mathbf{u}, v)$, then $\ell_{\text{rank}}(C; x, y)$ is the number of pairs $\{i, j\}$ such that $\llbracket \text{par}(X.A) \rrbracket_{I_i} = \llbracket \text{par}(X.A) \rrbracket_{I_j} = \mathbf{u}$, $\llbracket X.A \rrbracket_{I_i} \neq \llbracket X.A \rrbracket_{I_j} = v$, and $\pi(i) > \pi(j)$. Based on this feedback rule, we can observe that ℓ_{rank} is bounded by $\lambda = a \binom{m}{2}$.

Theorem 5. The class $\mathcal{N}_{\text{tree}}[\mathcal{A}, \mathcal{T}]$ is efficiently online learnable from outcome ranking tasks with regret bound $\binom{m}{2} a^{3/2}(\ln a + p \ln t + d^{p+1})^{1/2} \sqrt{T}$.

Importantly, the regret bounds of preference optimization and preference ranking are independent from the (potentially large) number n of objects in the database.

5 Experiments

After an excursion into the theoretical aspects of our framework, we now present the results of experiments on a recent benchmark in movie recommendation [9]. The database, for which the schema is described in Figure 1, is an integration of the MovieLens, IMDb and Rotten Tomatoes systems. The database includes 3592 movies and 6040 anonymous users, where each user rated at least 20 movies. In our experiments, attributes were formatted according to the specifications in the left table of Figure 4. The set of uncontrollable attributes was formed by User.Age, User.Gender and User.Occupation; the set \mathcal{A} of controllable attributes was formed by all remaining attributes in the table. The set \mathcal{T} of terms was constructed by associating to each attribute name A the shortest path from Movie to A . Finally, users' ratings were grouped into 3 categories: Like, Mixed and Dislike.

Our goal is to analyze the empirical performance of learning tree-structured CPR-nets with the Expanded Hedge algorithm according the type of prediction task (optimization vs. ranking) and the “expressiveness” of tree-structured CPR-nets identified here by their parent size ($p = 1, 2, 3$). Each experiment was conducted by selecting a group of 100 users at random from 4 known occupations.

Attribute	Values
Actor.Fame	2
Actor.Gender	2
Critics.Rating	4
Director.Fame	2
Movie.BoxOffice	4
Movie.Country	6
Movie.Date	6
Movie.Genre	20
User.Age	4
User.Gender	2
User.Occupation	4

	$p = 1$			$p = 2$			$p = 3$		
	100	500	1000	100	500	1000	100	500	1000
$m = 1$	60.5	68.9	74.1	74.5	88.0	90.7	68.9	86.5	95.2
$m = 2$	57.8	65.1	73.4	71.2	87.3	89.5	68.0	85.0	94.8
$m = 4$	47.0	53.5	66.2	64.8	78.1	82.6	51.6	79.4	89.0

Preference Optimization

	$p = 1$			$p = 2$			$p = 3$		
	100	500	1000	100	500	1000	100	500	1000
$m = 2$	62.7	69.4	75.8	73.2	86.1	91.0	70.0	87.3	95.8
$m = 5$	54.7	60.2	67.2	68.8	76.5	85.8	57.7	83.2	89.7
$m = 10$	50.1	56.8	64.0	60.0	73.1	80.5	54.8	77.0	85.4

Preference Ranking

Fig. 4. Experimental setup (left) and results (right) on the movie domain.

The experiment lasts for 1000 rounds, and we measured the performance of the algorithm after 100, 500 and 1000 trials. In doing so, we formed our test set by randomly taking out 10 users from the group, and counted the number of successful predictions on 100 examples generated from these users. The final accuracy results are obtained by averaging the learner’s performance on 20 experiments.

Concerning optimization tasks, each instance x_t was generated as follows: first, select at random a user from the group and fill x_t with her values; next, choose a movie from the user’s Like list, fill x_t with its corresponding values, and set $x'_t = x_t$; then, select at random $m = 1, 2$ or 4 movie attributes and remove their values from x_t . In this setting, the learner is charged one mistake if the predicted completion $N_t(x_t)$ does not match any movie in the user’s Like list. In this case, the response y_t is set to x'_t . For ranking tasks, each instance x_t was generated by first selecting a user, and then by forming an outcome set of $m = 2, 5$ or 10 movies, each chosen at random from the user’s lists. Here, a prediction mistake is made if the ranking $N_t(x_t)$ is inconsistent with the user’s ratings; if so, the response y_t is the closest ranking of $N(x_t)$ that is consistent with the user’s ratings.

The experiments were conducted on a 3.00 GHz Intel Xeon 5570 bi-processor with 8 GB RAM running Windows 7. All procedures were written in C++. Notably, for generating spanning trees we used the Markov chain-based algorithm due to Propp and Wilson [17]. Although this algorithm does not offer the theoretical guarantees of discriminant-based algorithms, it is much simpler to implement.

The accuracy results, reported in Figure 4, corroborate the evidence that ranking tasks are more challenging than optimization tasks for CPR-nets. Regardless of the task predicted, it is apparent that CPR-nets with $p = 3$ outperform CPR-nets with $p = 2$ which, in turn, outperform CPR-nets with $p = 1$. However, the improvement from $p = 1$ to $p = 2$ is more significant than the improvement from $p = 2$ to $p = 3$, indicating that uncontrollable attributes are crucial for prediction. By observing the cumulated loss of each component at the end of the 1000 rounds, the improvement from $p = 2$ to $p = 3$ is essentially due to the presence of rules involving both the user’s gender and the user’s age. Finally, in our experiments, the running time needed for generating tree-structured CPR-nets ranges from less than 1 millisecond with $p = 1$ to 1 second for $p = 3$. The running time needed for optimizing or ranking outcomes with these models is always less than 1 ms.

6 Related Work and Discussion

We have presented a common framework for learning and reasoning with conditional preferences in relational domains. Our main theoretical results state that the acyclic CPR-nets support tractable inference for both preference optimization and preference ranking, and tree-structured CPR-nets can be robustly learned from optimization and ranking tasks using linear loss functions. Our theoretical findings have been complemented by experiments on a large-scale recommendation domain.

To the best of our knowledge, this work provides the first contribution in *model-based relational preference learning*. Although the topic of preferences has recently attracted considerable attention in AI [7] and ML [12], virtually all models for representing preferences are propositional by nature. A notable exception is the work by Brafman [6] on extending generalized additive utility networks (GAI-nets) [1] to relational domains. In essence, CPR-nets are *directed* graphical models while relational GAI-nets are *undirected* graphical models. From a computational viewpoint, the crucial difference between these models lies in the fact that preference optimization can be solved in polynomial time with acyclic CPR-nets, while it is NP-hard for GAI-nets, even in the propositional case. The learnability issue of relational GAI-nets from optimization or ranking tasks remains open.

A direction of research that naturally emerges from our study is to extend the learnability results to larger classes of CPR-nets. Section 4.1 on parameter learning provides a starting point for attacking preference orderings that are more general than bipartite graphs. Orthogonally, Section 4.2 on structure learning might be extended to network structures that are more expressive than forests. Finally, the problem of learning and reasoning with *cyclic* CPR-nets looks challenging.

References

1. Bacchus, F., Grove, A.: Graphical models for preference and utility. In: Proc. of the 11th Conference on Uncertainty in Artificial Intelligence (UAI'95). pp. 3–10 (1995)
2. Binshtok, M., Brafman, R.I., Domshlak, C., Shimony, S.E.: Generic preferences over subsets of structured objects. *J. Artif. Intell. Res.* 34, 133–164 (2009)
3. Boutilier, C., Bacchus, F., Brafman, R.: UCP-networks: A directed graphical representation of conditional utilities. In: Proc. of the 17th Conference in Uncertainty in Artificial Intelligence (UAI'01). pp. 56–64 (2001)
4. Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H., Poole, D.: CP-nets: A tool for representing and reasoning with conditional Ceteris Paribus preference statements. *J. Artif. Intell. Res.* 21, 135–191 (2004)
5. Brafman, R., Domshlak, C.: Graphically structured value-function compilation. *Artificial Intelligence* 172(2-3), 325–349 (2008)
6. Brafman, R.I.: Relational preference rules for control. In: Proc. of the 11th Conference on Knowledge Representation and Reasoning (KR'08). pp. 552–559 (2008)
7. Brafman, R.I., Domshlak, C.: Preference handling - an introductory tutorial. *AI Magazine* 30(1), 58–86 (2009)
8. Brafman, R.I., Domshlak, C., Shimony, S.E.: On graphical modeling of preference and importance. *J. Artif. Intell. Res.* 25, 389–424 (2006)
9. Cantador, I., Brusilovsky, P., Kuflik, T.: 2nd workshop on information heterogeneity and fusion in recommender systems (HetRec'11). In: Proc. of the 5th ACM conference on Recommender systems. RecSys'11 (2011)

10. Cesa-Bianchi, N., Lugosi, G.: Prediction, Learning, and Games. Cambridge (2006)
11. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. 55(1), 119–139 (1997)
12. Fürnkranz, J., Hüllermeier, E.: Preference Learning. Springer (2011)
13. Getoor, L., Friedman, N., Koller, D., Taskar, B.: Learning probabilistic models of link structure. Journal of Machine Learning Research 3, 679–707 (2002)
14. Goldsmith, J., Lang, J., Truszczynski, M., Wilson, N.: The computational complexity of dominance and consistency in CP-nets. J. Artif. Intell. Res. 33, 403–432 (2008)
15. Koolen, W.M., Warmuth, M.K., Kivinen, J.: Hedging structured concepts. In: Proc. of the 23th Conference on Learning Theory (COLT'10). pp. 93–105 (2010)
16. Kulkarni, V.G.: Generating random combinatorial objects. J. Algorithms 11(2), 185–207 (1990)
17. Propp, J.G., Wilson, D.B.: How to get a perfectly random sample from a generic markov chain and generate a random spanning tree of a directed graph. J. Algorithms 27(2), 170–217 (1998)
18. Tutte, W.T.: Graph Theory. Cambridge (1984)

Appendix

Let \mathcal{N} be a CPR-net and κ be a skeleton. By N_κ , we denote the *ground conditional preference network (CP-net)* which associates to each instance $o.A$ of an attribute $X.A$ in N a parent set $par(o.A)$ and a conditional preference table $cpt_{o.A}$. Specifically, if $X.A$ is an attribute with parent set $\{X.R_1.A_1, \dots, X.R_p.A_p\}$ then:

- $par(o.A)$ is the set of all attributes $o_{ij}.A_i$, such that $1 \leq i \leq p$, $1 \leq j \leq n_i$, and $\llbracket o.R_i \rrbracket_\kappa = \{o_{i1}, \dots, o_{in_i}\}$,
- $cpt_{o.A}$ is the map that assigns to each vector $\mathbf{w} = (\mathbf{v}_1, \dots, \mathbf{v}_p)$ in the space $D(A_1)^{n_1} \times \dots \times D(A_p)^{n_p}$ the preference ordering $cpt_{o.A}(\mathbf{w}) = cpt_{X.A}(\mathbf{v})$ where \mathbf{v} is the aggregated vector $(\gamma(\mathbf{v}_1), \dots, \gamma(\mathbf{v}_p))$.

The dependency graph of N_κ , defined in the same way as for N , is denoted $\mathcal{G}(N_\kappa)$.

Proof of Theorem 1. Let N be an acyclic CPR-net, κ be a skeleton, and suppose that $I \succ_N I$ for some $I \in \mathcal{I}_\kappa$. So, there is a sequence (I_1, \dots, I_n) of flips such that $I_1 = I = I_n$ and $I_i \succ_N I_{i+1}$ for $1 \leq i < n$. Let S be the set of all ground attributes $o.A$ such that $\llbracket o.A \rrbracket_{I_i} \neq \llbracket o.A \rrbracket_{I_{i+1}}$ for some flip (I_i, I_{i+1}) . Consider any linear extension of S according to $\mathcal{G}(N_\kappa)$, and take the first element $o.A$ in the ordering. Let i be the first index in the sequence such that $\llbracket o.A \rrbracket_{I_i} \neq \llbracket o.A \rrbracket_{I_{i+1}}$. It follows that $\llbracket o.A \rrbracket_I \neq \llbracket o.A \rrbracket_{I_{i+1}}$. However, because $\mathcal{G}(N_\kappa)$ is acyclic, there is no parent of $o.A$ in S , and hence the ground condition $\llbracket par(o.A) \rrbracket_I$ remains unchanged during all the sequence of flips. It follows that $\llbracket o.A \rrbracket_{I_{i+1}} = \llbracket o.A \rrbracket_{I_{i+2}} = \dots = \llbracket o.A \rrbracket_{I_n}$, and hence $\llbracket o.A \rrbracket_I \neq \llbracket o.A \rrbracket_{I_n}$, which contradicts the assumption that $I \succ_N I$.

Proof of Lemma 1. Consider any arc $X.Y$ in the schema \mathcal{S} . The inverse $\llbracket Y.X \rrbracket_I$ of the relation $\llbracket X.Y \rrbracket_I$ can be computed in $\mathcal{O}(n)$ time. In addition, the tuples in $\llbracket X.Y \rrbracket_I$ (or its inverse) can be ordered lexicographically, which requires $\mathcal{O}(n \log_2 n)$ steps. Based on this ordering, the composition $\llbracket X.Y \rrbracket_I \circ \llbracket Y.Z \rrbracket_I$ can be performed in $\mathcal{O}(n)$ time, in the following way: project $\llbracket X.Y \rrbracket_I$ onto the objects shared by $\llbracket Y.Z \rrbracket_I$ and prune any tuple in $\llbracket Y.Z \rrbracket_I$ that has no match in that projection. Thus, the i th value $\gamma(\llbracket o.R_i.A_i \rrbracket_I)$ of the tuple $\llbracket par(o.A) \rrbracket_I$ can be found in $\mathcal{O}(kn \log_2 n + n \log_2 d)$ time using at most k join operations and one aggregate operation. Since the number of parents per attribute is constant, the result follows.

Proof of Theorem 2. We first establish the correctness of the forward sweep algorithm. Let I be a partial outcome, and J be the completion of I returned by the algorithm. Suppose that $J' \succ_N J$ for some distinct completion J' of I . In this case, there is a sequence of flips from J' to J in \succ_N . Let S be the set of ground attributes for which the value is switched in the sequence. Because $\mathcal{G}(N_\kappa)$ is acyclic, there is at least one $o.A \in S$ for which the values of all parents are fixed by I . Since J and J' are both extensions of I , it follows that $\llbracket \text{par}(o.A) \rrbracket_I = \llbracket \text{par}(o.A) \rrbracket_J = \llbracket \text{par}(o.A) \rrbracket_{J'}$. However, because $\llbracket o.A \rrbracket_J$ is optimal, the value of $o.A$ cannot be switched in the sequence, which contradicts the assumption that $J' \succ_N J$.

Now, consider the computational cost of the algorithm. A linear extension of $\mathcal{G}(N)$ can be constructed in $\mathcal{O}(a)$ time. Based on this ordering, a linear extension of $\mathcal{G}(N_\kappa)$ can be constructed in $\mathcal{O}(an)$ time, by replacing the attribute at position i with its n ground instances at positions i_1, \dots, i_n , where $i_1 > i - 1$ and $i_n < i + 1$. Since the parent assignment of each ground attribute can be found in $\mathcal{O}(t)$ time, the algorithm returns a completion of I in $\mathcal{O}(ant)$ time.

Proof of Theorem 3. The basic ingredients of ϕ are two weight mappings f and g , where f is used to quantify local preferences in the tables of N , while g is used to quantify global dependencies between attributes. Consider an attribute $X.A$ with parent set $\{X_1.R_1.A_1, \dots, X_p.R_p.A_p\}$. Then, for each \mathbf{u} in $D(A_1) \times \dots \times D(A_p)$ and each v in $D(A)$, $f(\mathbf{u}, v)$ is the number of descendants of v in the preference ordering $\text{cpt}_{X.A}(\mathbf{u})$. Note that $f(\mathbf{u}, v) \leq |D(A)| - 1$, and $f(\mathbf{u}, v) = 0$ if v is a leaf.

The mapping g is constructed in a recursive way using a topological ordering of $\mathcal{G}(N)$. If $X.A$ is the first attribute in the ordering, we set $g(X.A) = 1$. Assuming by induction hypothesis that g is fixed for the first $k - 1$ elements in the ordering, if the k th element $X.A$ has no parents, then $g(X.A) = 1$. Otherwise,

$$g(X.A) = \frac{1}{|D(A)|} \min_{i=1, \dots, p} \left(\frac{g(X_i.A_i)}{\sum_{X_j.A_j \in \text{chi}(X_i.A_i)} \llbracket X_j \rrbracket_\kappa} \right) \quad (1)$$

where $\{X_1.A_1, \dots, X_p.A_p\}$ is the set of all parents of $X.A$ in $\mathcal{G}(N)$, and $\text{chi}(X_i.A_i)$ is the set of all children of $X_i.A_i$ in $\mathcal{G}(N)$. The aim of g is thus to distribute the weight of an attribute evenly between its ground children.

With these ingredients in hand, the utility function ϕ is specified as a sum of sub-utilities $\phi_{X.A}$, each defined for a controllable attribute $X.A$ in N . Namely, if $X.A$ is an attribute with parent set $\text{par}(X.A) = \{X_1.R_1.A_1, \dots, X_p.R_p.A_p\}$, then $\phi_{X.A}$ associates to each value v in $D(A)$ and to each vector \mathbf{u} in $D(A_1) \times \dots \times D(A_p)$ the utility: $\phi_{X.A}(v, \mathbf{u}) = g(X.A)f(\mathbf{u}, v)$. Based on these sub-utility functions, the value of any interpretation I is simply given by:

$$\phi(I) = \sum_{X.A \in \mathcal{A}} \sum_{o \in \llbracket X \rrbracket_\kappa} \phi_{X.A}(\llbracket \text{par}(o.A) \rrbracket_I, \llbracket o.A \rrbracket_I) \quad (2)$$

By a direct application of [5, Theorem 3], it follows that ϕ is consistent with the ground CP-net N_κ . Since $\succ_{N_\kappa} = \succ_N$, ϕ is consistent with the CPR-net N .

Now, consider the computational cost required for constructing ϕ . For each of the values of each preference ordering in N , the weight $f(\mathbf{u}, v)$ can be computed in $\mathcal{O}(d)$ time. Since there are at most a attributes and d^p table entries per attribute, the weight map f can be constructed in $\mathcal{O}(ad^{p+1})$ time. The weight map g can be

obtained in $\mathcal{O}(a)$ time by simply constructing a topological ordering of $\mathcal{G}(N)$ and memorizing the number of ground children per attribute during the traversal.

Finally, the utility $\phi(I)$ of any interpretation I in the outcome set S can be computed in $\mathcal{O}(ant)$ time, as specified in Equation 2. Thus, any ranking of S can be found in $\mathcal{O}(anmt)$ time by labeling each $I \in S$ with ϕ and breaking ties arbitrarily.

Proof of Lemma 3. Let \mathcal{G}_d be the space of all bipartite graphs with node set $[d] = \{1, \dots, d\}$, and \mathbf{w} be a weight vector over \mathbb{R}_+^n , where \mathbb{R}_+ is the set of positive reals. Any graph in \mathcal{G}_d is a pair $G = (D_G, D'_G)$ where D_G is a subset of $[d]$ and $D'_G = [d] \setminus D_G$. The weight of G with respect to \mathbf{w} is defined by the product of weights of elements in D_G . For any subset \mathcal{G} of \mathcal{G}_d , we denote by $\mathbf{w}(\mathcal{G})$ the sum of weights of graphs in \mathcal{G} . Now, let $\mathcal{G}_d[U, V]$ be the subset of \mathcal{G}_d formed by all bipartite graphs G for which D_G covers U and D'_G covers V . We can observe that:

$$\mathbf{w}(\mathcal{G}_d[U, V]) = \prod_{i \in U} w_i \prod_{j \in V \setminus U} (1 + w_j) \quad (3)$$

With this property in hand, we can derive a simple greedy algorithm that generates random bipartite graphs over the distribution induced by $\mathbf{w}(\mathcal{G}_d)$ in $\mathcal{O}(d^2)$ time. Starting from the pair of sets $(U_0, V_0) = (\emptyset, [d])$, for each $i = 1, \dots, d$, we first evaluate the probability p_i defined by the ratio of $\mathbf{w}(G_d[U_{i-1} \cup \{i\}, V_{i-1}])$ to $\mathbf{w}(G_d[U_{i-1}, V_{i-1}])$. Then (U_i, V_i) is set to $(U_{i-1} \cup \{i\}, V_{i-1})$ with probability p_i , and to $(U_{i-1}, V_{i-1} \cup \{i\})$ with probability $1 - p_i$.

Now, let $X.A$ be an attribute with parent set $\{X_1.R_1.A_1, \dots, X_p.R_p.A_p\}$, and \mathbf{u} be a vector in $D(A_1) \times \dots \times D(A_p)$. Without loss of generality, assume that $D(A) = \{v_1, \dots, v_d\}$, and let \mathbf{w} be the weight vector with $w_i = e^{\theta(C_i)}$, where C_i is the component $(X.A, \mathbf{w}, v_i)$. Using the above algorithm we can generate in $\mathcal{O}(d^2)$ time a random bipartite preference ordering for the entry $cpt_{X.A}(\mathbf{u})$. Since there are at most a attributes and d^p tuples of values per attribute, the result follows.

Proof of Lemma 4. Let $X.A$ be an attribute with parent set $par(X.A)$, and \mathbf{u} be a vector in the domain of $par(X.A)$. By $\mathbf{W}[X.A, par(X.A), \mathbf{u}]$, we denote the sum of weights of all bipartite orderings over $D(A)$ corresponding to the set of all components with prefix $(X.A, par(X.A), \mathbf{u})$. By Equation 3, $\mathbf{W}[X.A, par(X.A), \mathbf{u}]$ can be evaluated in $\mathcal{O}(d)$ time. Let $\mathbf{W}[X.A, par(X.A)]$ be the sum of weights of all possible preference tables defined over the attribute $X.A$ with parent set $par(X.A)$. By distributivity, $\mathbf{W}[X.A, par(X.A)]$ can be evaluated in $\mathcal{O}(d^p)$ time by taking the product of weights $\mathbf{W}[X.A, par(X.A), \mathbf{u}]$ for each \mathbf{u} in $D(A_1) \times \dots \times D(A_p)$.

Now, let G be the weighted digraph with node set $\mathcal{A} \cup \{\top\}$; the arc set is formed by all pairs of distinct attributes in \mathcal{A} , together with all pairs of the form $(X.A, \top)$ where $X.A$ is an attribute in \mathcal{A} and \top is the dummy node denoting the absence of parent. The weight $\mathbf{W}[X.A, X'.A']$ is given by the sum of weights $\mathbf{W}[X.A, par(X.A)]$ of all possible parent sets $par(X.A)$ including $q \geq 1$ terms in \mathcal{T} with suffix $X'.A'$ and most $p - q$ terms in \mathcal{T} with uncontrollable suffix. The weight $\mathbf{W}[X.A, \top]$ is defined analogously. Based on the Matrix-Tree Theorem [18], a random spanning tree of G can be obtained in $\mathcal{O}(a^5)$ time using, for instance, a greedy algorithm suggested in [16]. This, together with the fact that the weight of each arc can be obtained in $\mathcal{O}(t^p d^p)$ time yields the result.