The Complexity of Learning Acyclic Conditional Preference Networks

Eisa Alanazi\textsuperscript{1}, Malek Mouhoub\textsuperscript{2}, and Sandra Zilles\textsuperscript{2}

\textsuperscript{1}Department of Computer Science
Umm Al-Qura University
Makkah, Saudi Arabia
\textsuperscript{2}Department of Computer Science
University of Regina
Regina, SK, Canada S4S 0A2

Abstract

Learning of user preferences, as represented by, for example, Conditional Preference Networks (CP-nets), has become a core issue in AI research. Recent studies investigate learning of CP-nets from randomly chosen examples or from membership and equivalence queries. To assess the optimality of learning algorithms as well as to better understand the combinatorial structure of classes of CP-nets, it is helpful to calculate certain learning-theoretic information complexity parameters. This article focuses on the frequently studied case of learning from so-called swap examples, which express preferences among objects that differ in only one attribute. It presents bounds on or exact values of some well-studied information complexity parameters, namely the VC dimension, the teaching dimension, and the recursive teaching dimension, for classes of acyclic CP-nets. We further provide algorithms that learn tree-structured and general acyclic CP-nets from membership queries. Using our results on complexity parameters, we assess the optimality of our algorithms as well as that of another query learning algorithm for acyclic CP-nets presented in the literature. Our algorithms are near-optimal, and can, under certain assumptions, be adapted to the case when the membership oracle is faulty.

1 Introduction

Preference learning has become a major branch of AI, with applications in decision support systems in general and in e-commerce in particular \cite{1}. For instance, recommender systems based on collaborative filtering make predictions on a single user’s preferences by exploiting information about large groups of
users. Another example are intelligent tutoring systems, which learn a student’s preferences in order to deliver personalized content to the student.

To design and analyze algorithms for learning preferences of a single user, one needs an abstract model for representing user preferences. Some approaches model preferences quantitatively, thus allowing for expressing the relative magnitude of preferences between object pairs, while others are purely qualitative, expressing partial orders or rankings over objects [2, 3].

Most application domains are of multi-attribute form, meaning that the set of possible alternatives (i.e., objects, or outcomes) is defined on a set of attributes and every alternative corresponds to an assignment of values to the attributes. Such combinatorial domains require compact models to capture the preference information in a structured manner. In recent years, various models have been suggested, such as Generalized Additive Decomposable (GAI-net) utility functions [4], Lexicographic Preference Trees [5], and Conditional Preference Networks (CP-nets) [6].

CP-nets provide a compact qualitative preference representation for multi-attribute domains where the preference of one attribute may depend on the values of other attributes. The study of their learnability [7, 8, 9, 10, 11, 4] is an ongoing topic in research on preference elicitation.

For example, Koriche and Zanuttini [7] investigated query learning of \( k \)-bounded acyclic CP-nets (i.e., with a bound \( k \) on the number of attributes on which the preferences for any attribute may depend). Their successful algorithms used both membership and equivalence queries, cf. [12], while they proved that equivalence queries alone are not sufficient for efficient learnability. CP-nets have also been studied in models of passive learning from examples, both for batch learning [8, 9, 10, 4] and for online learning [11, 13].

The focus of our work is on the design of methods for learning CP-nets through interaction with the user, and on an analysis of the complexity of such learning problems. In particular, we study the model of learning from membership queries, in which users are asked for information on their preference between two objects. To the best of our knowledge, algorithms for learning CP-nets from membership queries only have not been studied in the literature yet. We argue below (in Section 7) why algorithms using membership queries alone are of importance to research on preference learning. In a nutshell, membership queries seem to be more easily deployable in preference learning than equivalence queries and, from a theoretical point of view, it is interesting to see how powerful they are in comparison to equivalence queries. The latter alone are known to be insufficient for efficient learning of acyclic CP-nets [7]. Therefore, one major part of this article deals with learning CP-nets from membership queries only. A potential application of our approach is in recommender systems. Many recommender systems request item ratings from users, i.e., they pose queries to the user in the form of an item, and the user is expected to rate the item on a fixed scale. It has been shown though that replacing such queries with queries in which users are asked to express a preference between two items may be a more useful approach in recommender systems [14].

In every formal model of learning, a fundamental question in assessing learn-
ing algorithms is how many queries or examples would be needed by the best possible learning algorithm in the given model. For several models, lower bounds can be derived from the Vapnik Chervonenkis dimension (VCD, [15]). This central parameter is one of several that, in addition to yielding bounds on the performance of learning algorithms, provide deep insights into the combinatorial structure of the studied concept class. Such insights can in turn help to design new learning algorithms.

A classical result states that \( \log_2(\frac{4}{3})d \) is a lower bound on the number of equivalence and membership queries required for learning a concept class whose VCD equals \( d \) [16]. Likewise, it is known that a parameter called the teaching dimension [17] is a lower bound on the number of membership queries required for learning [18]. Therefore, another major part of this article deals with calculating exact values or non-trivial bounds on a number of learning-theoretic complexity parameters, such as the VCD and the teaching dimension. All these complexity parameters are calculated under the assumption that information about user preferences is provided for so-called swaps, exclusively. A swap is a pair of objects that differ in the value of only a single attribute. Learning CP-nets over swap examples is an often studied scenario [7, 8, 19, 20, 13], which we adopt here for various reasons detailed in Section 4.

Our main contributions are the following:

(a) We provide the first study that exactly calculates the VCD for the class of unbounded acyclic CP-nets, and give a lower bound for any bound \( k \). So far, the only existing studies present a lower bound [7], which we prove incorrect for large values of \( k \), and asymptotic complexities [21]. The latter show that \( \text{VCD} \in \Theta(2^k) \) for \( k = n - 1 \) and \( \text{VCD} \in \Theta(n 2^k) \) when \( k \in o(n) \), in agreement with our result that \( \text{VCD} = 2^n - 1 \) for \( k = n - 1 \), and is at least \( \text{VCD} \geq 2^k - 1 + (n - k)2^k \) for general values of \( k \). It should be noted that both previous studies assume that CP-nets can be incomplete, i.e., for some attributes, preference relations may not be fully specified. In our study, we first investigate the (not uncommon) assumption that CP-nets are complete, but then we extend each of our results to the more general case that includes incomplete CP-nets, as well. Further, some of our results are more general than existing ones in that they cover also the case of CP-nets with multi-valued attributes (as opposed to binary attributes.)

As a byproduct of our study, we obtain that the VCD of the class of all consistent CP-nets (whether acyclic or cyclic)\(^1\) equals that of the class of all acyclic CP-nets. Hence, the class of acyclic CP-nets is less expressive than that of all consistent CP-nets, but may (at least in some models) be as hard to learn.

(b) We further provide exact values (or, in some cases, non-trivial bounds) for two other important information complexity parameters, namely the teaching dimension [17], and the recursive teaching dimension [22].

(c) AppendixB gives an in-depth study of structural properties of the class of all complete acyclic CP-nets that are of importance to learning-theoretic studies.

\(^1\)A consistent CP-net is one that does not prefer an outcome \( o \) over another outcome \( o' \) while at the same time preferring \( o' \) over \( o \). Acyclic CP-nets are always consistent, but cyclic ones are not necessarily so.
(d) We present a new algorithm that learns tree-structured CP-nets (i.e., the case \(k = 1\)) from membership queries only and use our results on the teaching dimension to show that our algorithm is close to optimal. We further extend our algorithm to deal with the general case of \(k\)-bounded acyclic CP-nets with bound \(k \geq 1\).

(e) In most real-world scenarios, one would expect some degree of noise in the responses to membership queries, or that sometimes no response at all is obtained. To address this issue, we demonstrate how, under certain assumptions on the noise and the missing responses, our algorithm for learning tree CP-nets can be adapted to handle incomplete or incorrect answers to membership queries.

(f) We re-assess the degree of optimality of Koriche and Zanuttini’s algorithm for learning bounded acyclic CP-nets, using our result on the VCD. This article extends a previous conference paper [23]. Theorem 4 in this conference paper included an incorrect claim about the so-called self-directed learning complexity of classes of acyclic CP-nets; the incorrect statement has been removed in this extended version.

2 Related Work

This section sets the problems studied in the present paper into the context of the existing literature, both in terms of methods for learning CP-nets and in terms of studies of learning-theoretic complexity parameters in general.

2.1 Learning CP-Nets

The problem of learning CP-nets has recently gained a substantial amount of attention [8, 24, 9, 7, 21, 10, 11, 25, 26, 27, 4].

Both in active and in passive learning, a sub-problem to be solved by many natural learning algorithms is the so-called consistency problem. This decision problem is defined as follows. A problem instance consists of a CP-net \(N\) and a set \(S\) of user preferences between objects, in the form of “object \(o\) is preferred over object \(o'\)” or “object \(o\) is not preferred over object \(o'\)”. The question to be answered is whether or not \(N\) is consistent with \(S\), i.e., whether the partial order \(\succ\) over objects that is induced by \(N\) satisfies \(o \succ o'\) if \(S\) states that \(o\) is preferred over \(o'\), and satisfies \(o \not\succ o'\) if \(S\) states that \(o\) is not preferred over \(o'\). The consistency problem was shown to be NP-hard even if \(N\) is restricted to be an acyclic \(k\)-bounded CP-net for some fixed \(k \geq 2\) and even when, for any object pair \((o, o')\) under consideration, the outcomes \(o\) and \(o'\) differ in the values of at most two attributes [8]. Based on this result, Dimopoulos et al. [8] showed that complete acyclic CP-nets with bounded indegree are not efficiently PAC-learnable, i.e., learnable in polynomial time in the PAC model. The authors, however, then showed that such CP-nets are efficiently PAC-learnable from examples that are drawn exclusively from the set of so-called transparent entailments. Specifically, this implied that complete acyclic \(k\)-bounded CP-nets
are efficiently PAC-learnable from swap examples. Michael and Papageorgiou [26] then provided a comprehensive experimental view on the performance of the algorithm proposed in [8]. Their work also proposed an efficient method for checking whether a given entailment is transparent or not. These studies focus on learning approximations of target CP-nets passively and from randomly chosen data. By comparison, all algorithms we propose below learn target CP-nets exactly and they actively pose queries in order to collect training data, following Angluin’s model of learning from membership queries [12].

Lang and Mengin [9] considered the complexity of learning binary separable CP-nets, in various learning settings. The literature also includes results on learning CP-nets from noisy examples, namely, via statistical hypothesis testing [25], using evolutionary algorithms and metaheuristics [28, 4], or by learning the induced graph directly, which takes time exponential in the number of attributes [10]. These results cannot be compared to the ones presented in the present paper, as (i) they focus on approximating instead of exactly learning the target CP-net, and (ii) the noise models on which they build differ substantially from the settings we consider. In our first setting, there is no noise in the data whatsoever. The second setting we study is one in which the membership oracle may corrupt a certain number of query responses, but there is no randomness to the process. Instead, one analyzes learning under an adversarial assumption on the oracle’s choice of which answers to corrupt, and then investigates whether exact learning is still possible [29, 30, 31]. To the best of our knowledge, this setting has not been studied in the context of CP-nets so far.

As for active learning, Guerin et al. [11] proposed a heuristic online algorithm that is not limited to swap comparisons. The algorithm assumes the user to provide explicit answers of the form “object o is preferred over object o′”, “object o′ is preferred over object o”, or “neither of the two objects is preferred over the other” to any query (o, o′). Labernia et al. [13] proposed another online learning algorithm based on swap observations where the latter can be noisy. It is assumed that the target CP-net represents the global preference for a group of users and the noise is due to variations of a user’s preference compared to the global one. The authors formally proved that their algorithm produces a close approximation to the target CP-net and analyzed the algorithm empirically under random noise. Again, as in all the related literature discussed above, the most striking difference to our setting is that these works focus on approximating the target CP-net rather than learning it exactly.

To the best of our knowledge, the only studies of learning CP-nets in Angluin’s query model, where the target concept is identified exactly, are one by Koriche and Zanuttini [7] and one by Labernia et al. [19]. Koriche and Zanuttini assumed perfect oracles and investigated the problem of learning complete and incomplete bounded CP-nets from membership and equivalence queries over the swap instance space. They showed that complete acyclic CP-nets are not learnable from equivalence queries alone but are attribute-efficiently learnable

---

2 A CP-net is separable if it is 0-bounded, i.e., the preferences over the domain of any attribute are not conditioned on the values of other attributes.
from membership and equivalence queries. Attribute-efficiency means that the number of queries required is upper-bounded by a function that is polynomial in the size of the input, but only logarithmic in the number of attributes. In the case of tree CP-nets, their results hold true even when the equivalence queries may return non-swap examples. The setting considered in their work is more general than ours and exhibits the power of membership queries when it comes to learning CP-nets. Labernia et al. [19] investigated the problem of learning an average CP-net from multiple users using equivalence queries alone. However, neither study addresses the problem of learning complete acyclic CP-nets from membership queries alone, whether corrupted or uncorrupted. We provide a detailed comparison of our methods to those by Koriche and Zanuttini in Section 7. In a nutshell, our algorithm improves on theirs in that it does not require equivalence queries, but has the downside of not being attribute-efficient. The latter is not an artefact of our algorithm—we argue in Section 7 why attribute-efficient learning of CP-nets with membership queries alone is not possible.

2.2 Complexity Parameters in Computational Learning Theory

The Vapnik Chervonenkis Dimension, also called VC dimension [15], is one of the best studied complexity parameters in the computational learning theory literature. Upper and/or lower sample complexity bounds that are linear in the VC dimension are known for various popular models of concept learning, namely for PAC learning, which is a basic model of learning from randomly chosen examples [32], for exact learning from equivalence and membership queries, which is a model of learning from active queries [16], and, in some special cases even for learning from teachers [33, 34].

Because of these bounds, knowledge of the VC dimension value of a concept class $\mathcal{C}$ can help in assessing learning algorithms for $\mathcal{C}$. For example, if the number of queries consumed by an algorithm learning $\mathcal{C}$ exceeds a known lower bound on the query complexity by a constant factor, we know that the algorithm is within a constant factor of optimal. For this reason, the VC dimension of classes of CP-nets have been studied in the literature. Koriche and Zanuttini [7], for the purpose of analyzing their algorithms for learning from equivalence and membership queries, established a lower bound on the VC dimension of the class of complete and incomplete $k$-bounded binary CP-nets. Chevaleyre et al. [21] gave asymptotic estimates on the VC dimension of classes of CP-nets. They showed that the VC dimension of acyclic binary CP-nets is $\Theta(2^n)$ for arbitrary CP-nets and $\tilde{\Theta}(n2^k)$ for $k$-bounded CP-nets. Here $n$ is the number of attributes in a CP-net. The results by Chevaleyre et al. are in agreement with our results, stating that VCD is $2^n - 1$ for arbitrary acyclic CP-nets and at least $(n - k)2^k + 2^k - 1$ for $k$-bounded ones.

Our work improves on both of these contributions. Firstly, we correct a mistake in the lower bound published by Koriche and Zanuttini. Secondly, compared to asymptotic studies by Chevaleyre et al., we calculate exact values...
or explicit lower bounds on the VC dimension. Thirdly, we calculate the VC dimension also for the case that the attributes in a CP-net have more than two values.

To the best of our knowledge, there are no other studies that address learning-theoretic complexity parameters of CP-nets. The present paper computes two more parameters, namely the teaching dimension [17] and the recursive teaching dimension [22], both of which address the complexity of machine teaching. Recently, models of machine teaching have received increased attention in the machine learning community [35, 36], since they try to capture the idea of helpfully selected training data as would be expected in many human-centric applications. In our study, teaching complexity parameters are of relevance for two reasons.

First, the teaching dimension is a lower bound on the number of membership queries required for learning [17], and thus a tool for evaluating the efficiency of our learning algorithms relative to the theoretic optimum.

Second, due to the increased interest in machine teaching, the machine learning community is looking for bounds on the efficiency of teaching, in terms of other well-studied parameters. The recursive teaching dimension is the first teaching complexity parameter that was shown to be closely related to the VC dimension. It is known to be at most quadratic in the VC dimension [37], and under certain structural properties even equal to the VC dimension [33]. However, it remains open whether or not it is upper-bounded by a function linear in the VC dimension [34]. With the class of all unbounded acyclic CP-nets, we provide the first example of an “interesting” concept class for which the VC dimension and the recursive teaching dimension are equal, provably without satisfying any of the known structural properties that would imply such equality. Thus, our study of the recursive teaching dimension of classes of CP-nets may be of help to ongoing learning-theoretic studies of teaching complexity in general.

3 Background

This section introduces the terminology and notation used subsequently, and motivates the formal settings studied in the rest of the paper.

3.1 Conditional Preference Networks (CP-nets)

We largely follow the notation introduced by Boutilier et al. [6] in their seminal work on CP-nets; the reader is referred to Table 1 for a list of the most important notation used throughout our manuscript.

Let $V = \{v_1, v_2, \ldots, v_n\}$ be a set of attributes or variables. Each variable $v_i \in V$ has a set of possible values (its domain) $D_{v_i} = \{v_{i1}, v_{i2}, \ldots, v_{im}\}$. We assume that every domain $D_{v_i}$ is of a fixed size $m \geq 2$, independent of $i$. An assignment $x$ to a set of variables $X \subseteq V$ is a mapping for every variable $v_i \in X$ to a value from $D_{v_i}$. We denote the set of all assignments of $X \subseteq V$ by $O_X$.
| notation | meaning |
|----------|---------|
| $n$      | number of the domain of each variable |
| $m$      | size of the domain of each variable |
| $k$      | upper bound on the number of parents of a variable in a CP-net |
| $V$      | set of $n$ distinct variables, $V = \{v_1, v_2, \ldots, v_n\}$ |
| $v_i$    | variable in $V$, for $1 \leq i \leq n$ |
| $D_{v_i}$| domain $\{v_{i1}, v_{i2}, \ldots, v_{im}\}$ of variable $v_i \in V$ |
| $O_X$    | set of vectors (outcomes over $X$) that assign each $v_i \in X$ a value in $D_{v_i}$ |
| $O$      | set of all outcomes over the full variable set $V$; equal to $O_V$ |
| $o \succ o'$ | outcome $o \in O$ is strictly preferred over outcome $o' \in O$ |
| $o[X]$   | projection of outcome $o \in O$ onto a set $X \subseteq V$; $o[v_i]$ is short for $o[\{v_i\}]$ |
| $x(= (o, o'))$ | swap pair of outcomes |
| $V(x)$   | swapped variable of $x$ |
| $x.1$ and $x.2$ | $o$ and $o'$ respectively, where $x = (o, o')$ is a swap |
| $x[\Gamma]$ | projection of $x.1$ (and also of $x.2$) onto a set $\Gamma \subseteq V \setminus \{V(x)\}$ |
| $Pa(v_i)$ | set of the parent variables of $v_i$; note that $Pa(v_i) \subseteq V \setminus \{v_i\}$ |
| $\succeq^u_{v_i}$ | conditional preference relation of $v_i$ in the context of $u$, where $u \in O_{Pa(v_i)}$ |
| CPT($v_i$) | conditional preference table of $v_i$ |
| $\text{size}(\text{CPT}(v_i))$ | size (number of preference statements) of CPT($v_i$); note $\text{size}(\text{CPT}(v_i)) \leq m|Pa(v_i)|$ |
| $E$      | set of edges in a CP-net, where $(v_i, v_j) \in E$ iff $v_i \in Pa(v_j)$ |
| $c$      | a concept |
| $\mathcal{C}$ | a concept in a concept class |
| $\mathcal{X}_{\text{swap}}$ | instance space over which a concept class is defined |
| $\mathcal{X}_{\text{swap}}^{\text{swap}}$ | instance space of swap examples (without redundancies) |
| $\mathcal{X}_{\text{swap}}^{\text{swap}}$ | instance space of swap examples (with redundancies) |
| $c(x)$   | label that concept $c$ assigns to instance $x$ |
| $\text{VCD}(\mathcal{C})$ | VC dimension of concept class $\mathcal{C}$ |
| $\text{TD}(\mathcal{C})$ | teaching dimension of concept class $\mathcal{C}$ |
| $\text{TD}(c, \mathcal{C})$ | teaching dimension of concept $c$ with respect to concept class $\mathcal{C}$ |
| $\text{RTD}(\mathcal{C})$ | recursive teaching dimension of concept class $\mathcal{C}$ |
| $\mathcal{C}_{\text{ac}}^{\kappa}$ | class of all complete acyclic $k$-bounded CP-nets, over $\mathcal{X}_{\text{swap}}$ |
| $\mathcal{C}_{\text{ac}}^{\kappa}$ | class of all complete or incomplete acyclic $k$-bounded CP-nets, over $\mathcal{X}_{\text{swap}}$ |
| $\mathcal{M}_k$ | $(n-k)k + \binom{n-1}{2} (\leq nk)$; maximum number of edges in a CP-net in $\mathcal{C}_{\text{ac}}^{\kappa}$ |
| $\mathcal{M}_k$ | $(n-k)m^k + \frac{m^k}{m^k - 1}$; maximum number of statements in a CP-net in $\mathcal{C}_{\text{ac}}^{\kappa}$ |
| $\mathcal{U}$ | smallest possible size of an $(m, n - 1, k)$-universal set |
| LIM      | strategy to combat a limited oracle |
| MAL      | strategy to combat a malicious oracle |
| $E^*(x)$ | set of all swap instances differing from $x$ in exactly one non-swapped variable |

Table 1: Summary of notation.
and remove the subscript when $X = V$. A preference is an irreflexive, transitive binary relation $\succ$. For any $o, o' \in O$, we write $o \succ o'$ (resp. $o \not\succ o'$) to denote the fact that $o$ is strictly preferred (resp. not preferred) to $o'$, where $o$ and $o'$ are incomparable w.r.t. $\succ$ if both $o \not\succ o'$ and $o' \not\succ o$ holds. We use $o[X]$ to denote the projection of $o$ onto $X \subset V$ and write $o[v_i]$ instead of $o[[v_i]]$.

The CP-net model captures complex qualitative preference statements in a graphical way. Informally, a CP-net is a set of statements of the form $\gamma : v_1^{i(1)} \succ \ldots \succ v_m^{i(m)}$ which states that the preference over $v_i$ with $D_{v_i} = \{v_1', v_2', \ldots, v_m'\}$ is conditioned upon the assignment of $\Gamma \subseteq V \setminus \{v_i\}$, where $\sigma$ is some permutation over $\{1, \ldots, m\}$. In particular, when $\Gamma$ has the value $\gamma \in O_T$ and $s \leq t$, $v_s^{i(f)}$ is preferred to $v_t^{i(f)}$ as a value of $v_i$ ceteris paribus (all other things being equal). That is, for any two outcomes $o, o' \in O$ where $o[v_i] = v_s^{i(f)}$ and $o'[v_i] = v_t^{i(f)}$, the preference holds when i) $o[\Gamma] = o'[\Gamma] = \gamma$ and ii) $o[Z] = o'[Z]$ for $Z = V \setminus (\Gamma \cup \{v_i\})$. In such case, we say $o$ is preferred to $o'$ ceteris paribus. Clearly, there could be exponentially many pairs of outcomes $(o, o')$ that are affected by one such statement.

CP-nets provide a compact representation of preferences over $O$ by providing such statements for every variable. For every $v_i \in V$, the decision maker\footnote{This can be any entity in charge of constructing the preference network, i.e., a computer agent, a person, a group of people, etc.} chooses a set $\text{Pa}(v_i) \subseteq V \setminus \{v_i\}$ of parent variables that influence the preference order of $v_i$. For any $\gamma \in O_{\text{Pa}(v_i)}$, the decision maker may choose to specify a total order $\succ_{\gamma}^{v_i}$ over $D_{v_i}$. We refer to $\succ_{\gamma}^{v_i}$ as the conditional preference statement of $v_i$ in the context of $\gamma$. A Conditional Preference Table for $v_i$, $\text{CPT}(v_i)$, is a set of conditional preference statements $\{\succ_{\gamma_1}^{v_i}, \ldots, \succ_{\gamma_k}^{v_i}\}$.

**Definition 1** (CP-net [6]). Given, $V$, $\text{Pa}(v)$, and $\text{CPT}(v)$ for $v \in V$, a CP-net is a directed graph $(V, E)$, where, for any $v_i, v_j \in V$, $(v_i, v_j) \in E$ iff $v_i \in \text{Pa}(v_j)$.

A CP-net is acyclic if it contains no cycles. We call a CP-net $k$-bounded, for some $k \leq n - 1$, if each vertex has indegree at most $k$, i.e., each variable has a parent set of size at most $k$. CP-nets that are 0-bounded are also called separable; those that are 1-bounded are called tree CP-nets. When speaking about the class of “unbounded” acyclic CP-nets, we refer to the case when no upper bound is given on the indegree of nodes in a CP-net, other than the trivial bound $k = n - 1$.

**Definition 2.** $\text{CPT}(v_i)$ is said to be complete, if, for every element $\gamma \in O_{\text{Pa}(v_i)}$, the preference relation $\succ_{\gamma}^{v_i}$ is defined, i.e., $\text{CPT}(v_i)$ contains a statement that imposes a total order on $D_{v_i}$ for every context $\gamma$ over the parent variables. By contrast, $\text{CPT}(v_i)$ is incomplete, if there exists some $\gamma \in O_{\text{Pa}(v_i)}$ for which the preference relation $\succ_{\gamma}^{v_i}$ is empty. Analogously, a CP-net $N$ is said to be complete if every $\text{CPT}$ it poses is complete; otherwise it is incomplete.

Note that we do not allow strictly partial orders as preference statements; a preference relation in a CPT must either be empty or impose a total order. In the
case of binary CP-nets, which is the focus of the majority of the literature on CP-nets, this restriction is irrelevant, since every order on a domain of two elements is either empty or total. In the non-binary case though, the requirement that every CPT statement be either empty or a total order is a proper restriction.

It would be possible to also study non-binary CP-nets that are incomplete in the sense that some CPT statements impose proper partial orders, but this extension is not discussed below.

Lastly, we assume CP-nets are defined in their minimal form, i.e., there is no dummy parent in any CPT that actually does not affect the preference relation.

Example 1. Figure 1a shows a complete acyclic CP-net over \( V = \{A, B, C\} \) with \( D_A = \{a, \bar{a}\} \), \( D_B = \{b, \bar{b}\} \), \( D_C = \{c, \bar{c}\} \). Each variable is annotated with its CPT. For variable \( A \), the user prefers \( a \) to \( \bar{a} \) unconditionally. For \( C \), the preference depends on the values of \( A \) and \( B \), i.e., \( Pa(C) = \{A, B\} \). For instance, in the context of \( ab, \bar{c} \) is preferred over \( c \). Removing any of the four statements in \( CPT(C) \) would result in an incomplete CP-net.

Two outcomes \( o, \hat{o} \in \mathcal{O} \) are swap outcomes (‘swaps’ for short) if they differ in the value of exactly one variable \( v_i \); then \( v_i \) is called the swapped variable [6]. The size of a preference table for a variable \( v_i \), denoted by \( size(CPT(v_i)) \), is the number of preference statements it holds which is \( m|Pa(v_i)| \) if CPT \( (v_i) \) is complete. The size of a CP-net \( N \) is defined as the sum of its tables’ sizes. \(^4\)

Example 2. In Figure 1, \( abc, \bar{a}bc \) are swaps over the swapped variable \( A \). The size of the CP-net is \( 1 + 2 + 4 = 7 \).

We will frequently use the notation \( M_k = \max\{size(N) \mid N \text{ is a } k\text{-bounded acyclic CP-net}\} \), which refers to the maximum number of statements in any \( k\)-bounded acyclic CP-net over \( n \) variables, each of domain size \( m \). Note that a CP-net has to be complete in order to attain this maximum size. It can be verified that \( M_k = (n - k)m^k + \frac{m^{k-1}}{m-1} \).

Lemma 1. The maximum possible size \( M_k \) of a \( k\)-bounded acyclic CP-net over \( n \) variables of domain size \( m \) is given by \( M_k = (n - k)m^k + \frac{m^{k-1}}{m-1} \).

Proof. We first make the following claim: any \( k\)-bounded acyclic CP-net of largest possible size has (i) exactly 1 variable of indegree \( r \), for any \( r \in \{0, \ldots, k-1\} \) and (ii) exactly \( n - k \) variables of indegree \( k \).

For \( k = 0 \), i.e., for separable CP-nets, there is no \( r \in \{0, \ldots, k - 1\} \), so the claim states the existence of exactly \( n \) vertices of indegree 0, which is obviously correct. Consider any \( k\)-bounded acyclic CP-net \( N \) of largest possible size, where \( k \geq 1 \). Since \( N \) is acyclic, it has a topological sort. W.l.o.g., suppose \( (v_1, \ldots, v_n) \) is the sequence of variables in \( N \) as they occur in a topological sort.

\(^4\) It might seem more adequate to define the size of a CPT to be \( (m - 1) \) times the number of its preference statements, as each preference statement consists of \( m - 1 \) pairwise preferences. In the binary case, i.e., when \( m = 2 \), this makes no difference. As this technical detail does not affect our results, we ignore it and define the size of a CPT and of a CP-net simply by the overall number of its statements.
A » a
a: b » b
a: b » b

C

(a) The CP-net.

Figure 1: An acyclic CP-net (cf. Def. 1) and its induced preference graph (cf. Def. 4).

Clearly, \( v_1 \) must have indegree 0. If \( v_2 \) were also of indegree 0, then \( N \) would not be of maximal size since one could add \( v_1 \) as a parent of \( v_2 \) without violating the indegree bound \( k \). The resulting CP-net would be of larger size than \( N \), since the size of CPT(\( v_2 \)) would grow by a factor of \( m \) without changing the sizes of other CPTs. Hence \( v_2 \) has indegree 1 in \( N \). With the same argument, one can prove that \( v_i \) has indegree \( i - 1 \) in \( N \), for \( 1 \leq i \leq k + 1 \). For the variables \( v_{k+2}, \ldots, v_n \), one can apply the same argument but has to cap their indegrees at \( k \) because \( N \) is \( k \)-bounded. Hence \( v_{k+1}, \ldots, v_n \) all have indegree \( k \). This establishes the claim.

It remains to count the maximal number of statements in a CP-net \( N \) of this specific structure. The maximal number of statements for a given CP-net graph is obviously obtained when the CP-net is complete, i.e., when the CPT for any variable \( v \) has \( m\lvert Pa(v)\rvert \) rules. Summing up, we obtain \( \sum_{i=0}^{k-1} m^i = \frac{m^k-1}{m-1} \) statements for the first \( k \) variables in the topological sort, plus \( (n-k)m^k \) statements for the remaining \( n-k \) variables.

From this lemma, we also know that the maximum possible number of edges in a \( k \)-bounded acyclic CP-net is \( (n-k)k + \sum_{i=0}^{k-1} i = (n-k)k + \binom{k}{2} \). We will use the notation \( e_{max} \) to refer to this quantity.

**Definition 3.** For given \( n \geq 1 \) and \( k \in \{0, \ldots, n-1\} \), let \( e_{max} = (n-k)k + \binom{k}{2} \) denote the maximum possible number of edges in a \( k \)-bounded acyclic CP-net over \( n \) variables.

Note that \( e_{max} \leq nk \).

The semantics of CP-nets is described in terms of improving flips. Let \( \gamma \in O_{Pa(v_i)} \) be an assignment of the parents for a variable \( v_i \in V \). Let \( \gamma_{v_i}^j = v_i^1 > \ldots > v_i^j \) be the preference order of \( v_i \) in the context of \( \gamma \). Then, all else being equal, going from \( v_i^j \) to \( v_i^k \) is an improving flip over \( v_i \) whenever \( k < j \leq m \).

**Example 3.** In Figure 1a, \( (\bar{a}bc, abc) \) is an improving flip with respect to the variable \( B \).
Figure 2: An example of a consistent cyclic CP-net.

For complete CP-nets, the improving flip notion makes every pair \((o, \hat{o})\) of swap outcomes comparable, i.e., either \(o \succ \hat{o}\) or \(\hat{o} \succ o\) holds [6]. The question “is \(o \succ \hat{o}\)?” is then a special case of a so-called dominance query and can be answered directly from the preference table of the swapped variable. Let \(v_i\) be the swapped variable of a swap \((o, \hat{o})\). Let \(\gamma\) be the context of \(Pa(v_i)\) in both \(o\) and \(\hat{o}\). Then, \(o \succ \hat{o}\) iff \(o[v_i] \succ_{\gamma}^v \hat{o}[v_i]\). A general dominance query is of the form: given two outcomes \(o, \hat{o}\) \in \(O\), is \(o \succ \hat{o}\)? The answer is yes, iff \(o\) is preferred to \(\hat{o}\), i.e., there is a sequence \((\lambda_1, \ldots, \lambda_n)\) of improving flips from \(\hat{o}\) to \(o\), where \(\hat{o} = \lambda_1\), \(o = \lambda_n\), and \((\lambda_i, \lambda_{i+1})\) is an improving flip for all \(i \in \{1, \ldots, n-1\}\) [6].

**Example 4.** In Figure 1b, \(abc \succ \bar{abc}\), as witnessed by the sequence \(\bar{abc} \rightarrow abc \rightarrow abc\) of improving flips.

**Definition 4 (Induced Preference Graph [6]).** The induced preference graph of a CP-net \(N\) is a directed graph \(G\) where each vertex represents an outcome \(o \in O\). An edge from \(\hat{o}\) to \(o\) exists iff \((o, \hat{o}) \in O \times O\) is a swap w.r.t. some \(v_i \in V\) and \(o[v_i]\) precedes \(\hat{o}[v_i]\) in \(\succ_{Pa(v_i)}^v\).

Therefore, a CP-net \(N\) defines a partial order \(\succ\) over \(O\) that is given by the transitive closure of its induced preference graph. If \(o \succ \hat{o}\) we say \(N\) entails \((o, \hat{o})\). \(N\) is consistent if there is no \(o \in O\) with \(o \succ o\), i.e., if its induced preference graph is acyclic. Acyclic CP-nets are guaranteed to be consistent while such guarantee does not exist for cyclic CP-nets; the consistency of the latter depends on the actual values of the CPTs [6]. Lastly, the complexity of finding the best outcome in an acyclic CP-net has been shown to be linear [6] while the complexity of answering dominance queries depends on the structure of CP-nets: PSPACE-complete for arbitrary (cyclic and acyclic) consistent CP-nets [38] and linear in case of trees [39].

**Example 5.** Figure 2 shows an example of a cyclic CP-net that is consistent while Figure 3 shows an inconsistent one. Note that both share the same CPTs except for CPT(C). The dotted edges in the induced preference graph of Figure 3 represent a cycle.
3.2 Concept Learning

The first part of our study is concerned with determining—for the case of acyclic CP-nets—the values of information complexity parameters that are typically studied in computational learning theory. By information complexity, we mean the complexity in terms of the amount of information a learning algorithm needs to identify a CP-net. Examples of such complexity notions will be introduced below.

A specific complexity notion corresponds to a specific formal model of machine learning. Each such learning model assumes that there is an information source that supplies the learning algorithm with information about a hidden target concept \( c^* \). The latter is a member of a concept class, which is simply the class of potential target concepts, and, in the context of this paper, also the class of hypotheses that the learning algorithm can formulate in the attempt to identify the target concept \( c^* \).

Formally, one fixes a finite set \( \mathcal{X} \), called instance space, which contains all possible instances (i.e., elements of) an underlying domain. A concept \( c \) is then defined as a mapping from \( \mathcal{X} \) to \{0, 1\}. Equivalently, \( c \) can be seen as the set \( c = \{ x \in \mathcal{X} \mid c(x) = 1 \} \), i.e., a subset of the instance space. A concept class \( \mathcal{C} \) is a set of concepts. Within the scope of our study, the information source (sometimes called oracle), supplies the learning algorithm in some way or another with a set of labeled examples for the target concept \( c^* \in \mathcal{C} \). A labeled example for \( c^* \) is a pair \((x, b) \in \mathcal{X} \times \{0, 1\}\) where \( x \in \mathcal{X} \) and \( b = c(x) \). Under the set interpretation of concepts, this means that \( b = 1 \) if and only if the instance \( x \) belongs to the concept \( c^* \). A concept \( c \) is consistent with a set \( S \subseteq X \times \{0, 1\} \) of labeled examples, if and only if \( c(x) = b \) for all \((x, b) \in S\), i.e., if every element of \( S \) is an example for \( c \).

In practice, a concept is usually encoded by a representation \( \sigma(c) \) defined based on a representation class \( \mathcal{R} \) [40]. Thus, one usually has some fixed representation class \( \mathcal{R} \) in mind, with a one-to-one correspondence between the concept class \( \mathcal{C} \) and its representation class \( \mathcal{R} \). We will assume in what follows that the representation class is chosen in a way that minimizes the worst case size of the representation of any concept in \( \mathcal{C} \). Generally, there may be various interpretations of the term “size;” since we will focus on learning CP-nets, we
use CP-nets as representations for concepts, and the size of a representation is simply the size of the corresponding CP-net as defined above.

At the onset of a learning process, both the oracle and the learning algorithm (often called learner for short) agree on the representation class $\mathcal{R}$ (and thus also on the concept class $\mathcal{C}$), but only the oracle knows the target concept $c^\ast$. After some period of communication with the oracle, the learner is required to identify the target concept $c^\ast$ either exactly or approximately.

Many learning models have been proposed to deal with different learning settings \cite{40,12,41,42}. These models typically differ in the constraints they impose on the oracle and the learning goal. One also distinguishes between learners that actively query the oracle for specific information content and learners that passively receive a set of examples chosen solely by the information source. One of the best known passive learning models is the Probably Approximately Correct (PAC) model \cite{42}. The PAC model is concerned with finding, with high probability, a close approximation to the target concept $c^\ast$ from randomly chosen examples. The examples are assumed to be sampled independently from an unknown distribution. On the other end of the spectrum, a model that requires exact identification of $c^\ast$ is Angluin’s model for learning from queries \cite{12}. In this model, the learner actively poses queries of a certain type to the oracle.

In this paper, we consider specifically two types of queries introduced by Angluin \cite{12}, namely membership queries and equivalence queries. A membership query is specified by an element $x \in \mathcal{X}$ of the instance space, and it represents the question whether or not $c^\ast$ contains $x$. The oracle supplies the learner with the correct answer, i.e., it provides the label $c^\ast(x)$ in response to the membership query for $x$. In an equivalence query, the learner specifies a hypothesis $c$. If $c = c^\ast$, the learning process is completed as the learner has then identified the target concept. If $c \neq c^\ast$, the learner is provided with a labeled example $(x, c^\ast(x))$ that witnesses $c \neq c^\ast$. That means, $c^\ast(x) \neq c(x)$. Note that $x$ in this case can be any element in the symmetric difference of the sets associated with $c$ and $c^\ast$.

A class $\mathcal{C} \subseteq 2^\mathcal{X}$ over some instance space $\mathcal{X}$ is learnable from membership and/or equivalence queries via a representation class $\mathcal{R}$ for $\mathcal{C}$, if there is an algorithm $\mathcal{A}$ such that for every concept $c^\ast \in \mathcal{C}$, $\mathcal{A}$ asks polynomially many adaptive membership and/or equivalence queries and then outputs a hypothesis $h$ that is equivalent to $c^\ast$. By adaptivity, we here mean that learning proceeds in rounds; in every round the learner asks a single query and receives an answer from the oracle before deciding on its subsequent query. The number of queries to be polynomial means that it is upper-bounded by a polynomial in $\text{size}(c^\ast)$ where $\text{size}(c^\ast)$ is the size of the minimal representation of $c^\ast$ w.r.t. $\mathcal{R}$.

The above definition is concerned only with the information or query complexity, i.e., the number of queries required to exactly identify any target concept. Moreover, $\mathcal{C}$ is said to be efficiently learnable from membership and/or equivalence queries if there exists an algorithm $\mathcal{A}$ that exactly learns $\mathcal{C}$, in the above sense, and runs in time polynomial in $\text{size}(c^\ast)$. Every one of the query strategies we describe in Section 7 gives an obvious polynomial time algorithm in this regard, and thus we will not explicitly mention run-time efficiency of
The combinatorial structure of a concept class $C$ has implications on the complexity of learning $C$, in particular on the sample complexity (sometimes called information complexity), which refers to the number of labeled examples the learner needs in order to identify any target concept in the class under the constraints of a given learning model. One of the most important complexity parameters studied in machine learning is the Vapnik-Chervonenkis dimension (VCD). In what follows, let $C$ be a concept class over the (finite) instance space $X$.

**Definition 5.** [15] A subset $Y \subseteq X$ is shattered by $C$ if the projection of $C$ onto $Y$ has $2^{|Y|}$ concepts. The VC dimension of $C$, denoted by $\text{VCD}(C)$, is the size of the largest subset of $X$ that is shattered by $C$.

For example, if $X$ contains 5 elements and $C$ is the class of all subsets of $X$ that have size at most 3, then the VC dimension of $C$ is 3. Clearly, no subset $Y$ of $X$ of size 4 can be shattered by $C$, since no concept in $C$ would contain all 4 elements of $Y$. That means, one obtains only 15, not the full 16 possible concepts over $Y$ when projecting $C$ onto $Y$. However, any subset $Y' \subset X$ of size 3 is indeed shattered by $C$, as every subset of $Y'$ is also a concept in $C$.

The number of randomly chosen examples needed to identify concepts from $C$ in the PAC-learning model is linear in $\text{VCD}(C)$ [43, 32]. By contrast to learning from random examples, in teaching models, the learner is provided with well-chosen labeled examples.

**Definition 6.** [17, 44] A teaching set for a concept $c^* \in C$ with respect to $C$ is a set $S = \{(x_1, \ell_1), \ldots, (x_z, \ell_z)\}$ of labeled examples such that $c^*$ is the only concept $c \in C$ that satisfies $c(x_i) = \ell_i$ for all $i \in \{1, \ldots, z\}$. The teaching dimension of $c$ with respect to $C$, denoted by $\text{TD}(c, C)$, is the size of the smallest teaching set for $c$ with respect to $C$. The teaching dimension of $C$, denoted by $\text{TD}(C)$, is given by $\text{TD}(C) = \max\{\text{TD}(c, C) \mid c \in C\}$.

Consider again the class $C_3$ of all subsets of size at most 3 over a 5-element instance space. Any concept $c$ containing 3 instances has a teaching set of size 3 in this class: the three positively labeled examples referring to the elements contained in $c$ uniquely determine $c$. However, concepts with fewer than 3 elements do not have teaching sets smaller than 5, since any set consisting of 2 positive and 2 negative examples agrees with at least two different concepts in $C_3$, and so does every set of 1 positive and 3 negative examples and every set of 4 negative examples.

$\text{TD}_{\text{min}}(C) = \min\{\text{TD}(c, C) \mid c \in C\}$ denotes the smallest TD of any $c \in C$. In the class $C_3$, the value for TD is 5, while the value for $\text{TD}_{\text{min}}$ is 3.

A well-studied variation of teaching is called recursive teaching. Its complexity parameter, the recursive teaching dimension, is defined by recursively removing from $C$ all the concepts with the smallest TD and then taking the maximum over the smallest TDs encountered in that process. For the corresponding definition of teachers, see [22].
Table 2: The class $\mathcal{C}$ of all singletons and the empty concept over a set of $t$ instances, along with the teaching dimension value of each individual concept.

| $\mathcal{C}$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | ... | $x_t$ | TD |
|---------------|-------|-------|-------|-------|------|-------|----|
| $c_0$         | 0     | 0     | 0     | 0     | ... | 0     | $t$|
| $c_1$         | 1     | 0     | 0     | 0     | ... | 0     | 1  |
| $c_2$         | 0     | 1     | 0     | 0     | ... | 0     | 1  |
| $c_3$         | 0     | 0     | 1     | 0     | ... | 0     | 1  |
| ...           | ...   | ...   | ...   | ...   | ... | ...   |    |
| $c_t$         | 0     | 0     | 0     | 0     | ... | 1     | 1  |

Definition 7. [22] Let $\mathcal{C}_0 = \mathcal{C}$ and, for all $i$ such that $\mathcal{C}_i \neq \emptyset$, define $\mathcal{C}_{i+1} = \mathcal{C}_i \setminus \{c \in \mathcal{C}_i \mid \text{TD}(c, \mathcal{C}_i) = \text{TD}_{\text{min}}(\mathcal{C}_i)\}$. The recursive teaching dimension of $\mathcal{C}$, denoted by $\text{RTD}(\mathcal{C})$, is defined by $\text{RTD}(\mathcal{C}) = \max\{\text{TD}_{\text{min}}(\mathcal{C}_i) \mid i \geq 0\}$.

As an example, consider $\mathcal{C}' = \{c_1, c_2, \ldots, c_t\}$ to be the class of singletons defined over the instance space $\mathcal{X} = \{x_1, x_2, \ldots, x_t\}$ where $c_i = \{x_i\}$ and let $\mathcal{C} = \mathcal{C}' \cup \{c_0\}$, where $c_0$ is the empty concept, i.e., $c_0(x) = 0$ for all $x \in \mathcal{X}$. Table 2 displays this class along with TD$(c, \mathcal{C})$ for every $c \in \mathcal{C}$. Since distinguishing the concept $c_0$ from all other concepts in $\mathcal{C}$ requires $t$ labeled examples, one obtains TD$(\mathcal{C}) = t$. However, RTD$(\mathcal{C}) = 1$ as witnessed by $\mathcal{C}_0 = \mathcal{C}'$ (each concept in $\mathcal{C}'$ can be taught with a single example) and $\mathcal{C}_1 = \{c_0\}$ (the remaining concept $c_0$ has a teaching dimension of 0 with respect to the class containing only $c_0$). Note that also VCD$(\mathcal{C}) = 1$, since there is no set of two examples that is shattered by $\mathcal{C}$.

Similarly, one can verify that RTD$(\mathcal{C}_2^0) = 3$.

As opposed to the TD, the RTD exhibits interesting relationships to the VCD. For example, if $\mathcal{C}$ is a maximum class, i.e., its size $|\mathcal{C}|$ meets Sauer’s upper bound $(\binom{|\mathcal{X}|}{0}) + (\binom{|\mathcal{X}|}{1}) + \ldots + (\binom{|\mathcal{X}|}{\text{VCD}(\mathcal{C})})$ [45], and in addition $\mathcal{C}$ can be “corner-peeled”\(^5\), then $\mathcal{C}$ fulfills RTD$(\mathcal{C}) = \text{VCD}(\mathcal{C})$ [33]. The same equality holds if $\mathcal{C}$ is intersection-closed or has VCD 1 [33]. In general, the RTD is upper-bounded by a function quadratic in the VCD [37].

4 Representing CP-Nets as Concepts

Assuming a user’s preferences are captured in a target CP-net $N^*$, an interesting learning problem is to identify $N^*$ from a set of observations representing the user’s preferences, i.e., labeled examples, of the form $o \succ o'$ or $o \nprec o'$ where $\succ$ is the relation induced by $N^*$ [8, 7]. In order to study the complexity of learning CP-nets, we model a class of CP-nets as a concept class over a fixed instance space.

\(^5\)Corner peeling is a sample compression procedure introduced by Rubinstein and Rubinstein [46]; the actual algorithm or its purpose are not of immediate relevance to our paper.
The first issue to address is how to define the instance space. A natural approach would be to consider any pair \((o, o')\) of outcomes as an instance. Such instance would be contained in the concept corresponding to \(N^*\) if and only if \(N^*\) entails \(o \succ o'\). In our study however, we restrict the instance space to the set of all swaps.

On the one hand, note that our results, due to the restriction to swaps, do not apply to scenarios in which preferences are elicited over arbitrary outcome pairs, as is likely the case in many real-world applications.

On the other hand, for various reasons, the restriction to swaps is still of both theoretical and practical interest and therefore justified. First, CP-net semantics are completely determined by the preference relation over swaps, so that no information on the preference order is lost by restricting to swaps. In particular, the set of all swaps is the most compact instance space for representing the class of all CP-nets or the class of all acyclic CP-nets. Second, many studies in the literature address learning CP-nets from information on swaps, see [7, 8, 19, 20, 13], so that our results can be compared to existing ones on the swap instance space. Third, in the learning models that we consider (teaching and learning from membership queries) learning becomes harder when restricting the instance space. For example, a learning algorithm may potentially succeed faster when it is allowed to enquire about preferences over any pair of outcomes rather than just swaps. Since our study restricts the information presented to the learner to preferences over swaps, our complexity results thus serve as upper bounds on the complexity of learning in more relaxed settings. Fourth, for the design of learning methods, it is often desirable that the learner can check whether its hypothesis (in our case a CP-net) is consistent with the labeled examples obtained. It is known that the time complexity of checking whether a CP-net is consistent with an example is linear in the case of swap examples but NP-hard for non-swap examples [6]. Fifth, there are indeed potential application scenarios in which the information presented to a learner may be in the form of preferences over swaps. This is due to the intuition that in many cases preferences over swaps would be much easier to elicit than preferences over two arbitrary outcomes. For example, a user may be overwhelmed with the question whether to prefer one laptop over another if each of them has a nice feature that the other does not have. It is likely much easier for the user to express a preference over two laptops that are identical except in a single feature like their color.

One may argue that one of the parameters that we study, namely the VC dimension, should be computed over arbitrary instances rather than just swap instances, since the VC dimension captures how difficult a concept class is to learn when the choice of instances is out of the learner’s (or teacher’s) control. However, for the following two reasons, computing the VC dimension over swap instances is of importance to our study:

- We use our calculations on the VC dimension in order to assess the optimality of one of Koriche and Zanuttini’s algorithms [7] for learning acyclic CP-nets with nodes of bounded indegree from equivalence and member-
ship queries. It is well-known that \( \log_2(4/3)\text{VCD}(C) \) is a lower bound on the number of membership and equivalence queries required for learning a concept class \( C \) [16]. Since Koriche and Zanuttini’s algorithm that we assess is designed over the swap instance space, an optimality assessment using the VC dimension necessarily requires that the VC dimension be computed over the swap instance space as well.

- Although the VC dimension is best known for characterizing the sample complexity of learning from randomly chosen examples, namely in the model of PAC learning, existing results exhibit a broader scope of applicability of the VC dimension. Recently it was shown that the number of examples needed for learning from benevolent teachers can be upper-bounded by a function quadratic in the VC dimension [37]. When studying the number of swap examples required for teaching CP-nets, thus again also the VC dimension over swap examples becomes interesting.

We therefore consider the set \( \{(o, o') \in O \times O \mid (o, o') \text{ is a swap}\} \) as an instance space. The size of this instance space is \( nm^n(m-1) \): every variable has \( m^{n-1} \) different assignments of the other variables and fixing each assignment of these we have \( m(m-1) \) instances. For complete acyclic CP-nets, however, half of these instances are redundant as if \( c((o, o')) = 0 \) then we know for certain that \( c((o', o)) = 1 \), and vice versa. By contrast, in the case of incomplete CP-nets, \( c((o, o')) = 0 \) does not necessarily mean \( c((o', o)) = 1 \) as there could be no relation between the two outcomes, i.e., \( o \) and \( o' \) are incomparable, corresponding to both \( c((o, o')) = 0 \) and \( c((o', o)) = 0 \).

Consequently, the choice of instance space in our study will be as follows:

- Whenever we study classes of CP-nets that contain incomplete CP-nets, we use the instance space \( X_{\text{swap}} = \{(o, o') \in O \times O \mid (o, o') \text{ is a swap}\} \).

- Whenever we study classes of only complete CP-nets, we use an instance space \( X_{\text{swap}} \subset X_{\text{swap}} \) that includes, for any two swap outcomes \( o, o' \), exactly one of the two pairs \( (o, o'), (o', o) \). Note that \(|X_{\text{swap}}| = nm^n\binom{m}{2} = \frac{m^nn(m-1)\cdot}{2} \). When we say that a learning algorithm, either passively or through active queries, is given information on the label of a swap \( (o, o') \) under the target concept, we implicitly refer to information on either \( (o, o') \) or \( (o', o) \), depending on which of the two is actually contained in \( X_{\text{swap}} \).

We sometimes refer to \( X_{\text{swap}} \) as the set of all swaps without “redundancies”, since for the case of complete CP-nets half the instances in \( X_{\text{swap}} \) are redundant. Of course, for incomplete CP-nets they are not redundant.

For \( x = (o, o') \in X_{\text{swap}} \), let \( V(x) \) denote the swapped variable of \( x \). We refer to the first and second outcomes of an example \( x \) as \( x.1 \) and \( x.2 \), respectively. We use \( x[\Gamma] \) to denote the assignments (in both \( x.1 \) and \( x.2 \)) of \( \Gamma \subseteq V \setminus \{V(x)\} \).

\(^6\)All our results are independent on the mechanism choosing which of two pairs \( (o, o'), (o', o) \) to include in \( X_{\text{swap}} \). We assume that the selection is prescribed in some arbitrary fashion.
Table 3: The concepts $c_1$ and $c_2$ represent the CP-nets in Figures 1 and 2, respectively, over $\mathcal{X}_{\text{swap}}$.

Note that $x[\Gamma]$ is guaranteed to be the same in $x.1$ and $x.2$, otherwise $x$ will not form a swap instance.

Now if $N$ is any CP-net and induces the (partial) order $\succ$ over outcomes, then $N$ corresponds to a concept $c_N$ over $\overline{\mathcal{X}}_{\text{swap}}$ (over $\mathcal{X}_{\text{swap}}$, respectively), where $c_N$ is defined as follows, for any $x \in \overline{\mathcal{X}}_{\text{swap}}$ (any $x \in \mathcal{X}_{\text{swap}}$, respectively.)

$$c(x) = \begin{cases} 1 & \text{if } x.1 \succ x.2 \\ 0 & \text{otherwise} \end{cases}$$

In such case, we say that $c$ is represented by $N$. Since no two distinct CP-nets induce exactly the same set of swap entailments, a concept over the instance space $\overline{\mathcal{X}}_{\text{swap}}$ cannot be represented by more than one CP-net, and a concept over the instance space $\mathcal{X}_{\text{swap}}$ cannot be represented by more than one complete CP-net. Therefore, in the context of a specific instance space, we identify a CP-net $N$ with the concept $c_N$ it represents and use these two notions interchangeably.

Consequently, we say that a concept $c$ contains a swap pair $x$ iff the CP-net representing $c$ entails $(x.1, x.2)$. By $\text{size}(c)$, we refer to the size of the CP-net that represents $c$.

Table 3 shows two concepts $c_1$ and $c_2$ that correspond to the complete CP-nets shown in Figures 1 and 2, respectively, along with one choice of $\mathcal{X}_{\text{swap}}$. It is important to restate the fact that $c(x)$ is actually a dominance relation between $x.1$ and $x.2$, i.e., $c(x)$ is mapped to 1 (resp. to 0) if $x.1 \succ x.2$ (resp. $x.2 \succ x.1$) holds. Thus, we sometimes talk about the value of $c(x)$ in terms of the relation between $x.1$ and $x.2$ ($x.1 \succ x.2$ or $x.2 \succ x.1$).

In the remainder of this article, we fix $n \geq 1$, $m \geq 2$, $k \in \{0, \ldots, n-1\}$, and consider the following two concept classes:

- The class $C_{\text{ac}}^k$ of all complete acyclic $k$-bounded CP-nets over $n$ variables of domain size $m$. This class is represented over the instance space $\mathcal{X}_{\text{swap}}$.
- The class $\overline{C}_{\text{ac}}^k$ of all complete and all incomplete acyclic $k$-bounded CP-nets over $n$ variables of domain size $m$. This class is represented over the instance space $\overline{\mathcal{X}}_{\text{swap}}$.

5 The Complexity of Learning Complete Acyclic CP-Nets

In this section, we will study the information complexity parameters introduced above, for the class $C_{\text{ac}}^k$ of all complete acyclic $k$-bounded CP-nets over $n$ vari-
ables of domain size $m$. In Section 6, we will extend our results on the VC dimension and the teaching dimension also to the class $C_{ac}$ of all complete and incomplete acyclic $k$-bounded CP-nets. It turns out though that studying the complete case first is easier.

Table 4 summarizes our complexity results for complete acyclic $k$-bounded CP-nets. The two extreme cases are unbounded acyclic CP-nets ($k = n - 1$) and separable CP-nets ($k = 0$).

To define the value $U_k$ used in this table, we will first introduce the notion of $(z, k)$-universal set, which is typically used in combinatorics, cf. [47, 48].

**Definition 8.** Let $S \subseteq \{0, 1\}^z$ be a set of binary vectors of length $z$ and let $k \leq z$. The set $S$ is called $(z, k)$-universal if, for every set $Z = \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, z\}$ with $|Z| = k$, the projection

$$\{(s_{i_1}, \ldots, s_{i_k}) \mid (s_1, \ldots, s_z) \in S\}$$

of $S$ to the components in $Z$ is of size $2^k$, i.e., it contains all binary vectors of length $k$.

In other words, a $(z, k)$-universal set $S$ is a set of concepts over an instance space $X$ of size $z$, such that every subset of $X$ of size $k$ is shattered by $S$.

**Example 6.** Consider $z = 10$ and $k = 3$, and let $S$ be the set of all binary vectors of length 10 that contain no more than 3 components equal to 1. This set is $(10, 3)$-universal, since projecting it to any set of three components yields all eight binary vectors of length 3. It is not $(10, 4)$-universal since projections onto four components do not exhibit the binary vector consisting of four 1s.

The role of $(z, k)$-universal sets in our study will become evident later on when we investigate the teaching dimension of classes of CP-nets. Here we present a generalization of $(z, k)$-universal sets to the non-binary case in order to define the quantity $U_k$ used in Table 4.

**Definition 9.** Let $S \subseteq \{1, \ldots, m\}^z$ be a set of vectors of length $z$ with components in $\{1, \ldots, m\}$. Let $k \leq z$. The set $S$ is called $(m, z, k)$-universal if, for every set $Z = \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, z\}$ with $|Z| = k$, the projection

$$\{(s_{i_1}, \ldots, s_{i_k}) \mid (s_1, \ldots, s_z) \in S\}$$

of $S$ to the components in $Z$ is of size $m^k$, i.e., it contains all vectors of length $k$ with components in $\{1, \ldots, m\}$.

Thus the classical $(z, k)$-universal sets correspond to $(2, z, k)$-universal sets in our notation. To date, no formula is known that expresses the exact size of a smallest possible $(z, k)$-universal set (or of a smallest possible $(m, z, k)$-universal set, respectively) in dependence of $z$ and $k$ (in dependence of $m$, $z$, and $k$, respectively,) though some useful bounds on this quantity have been established [48]. The quantity is of importance for our results on the teaching dimension.
Table 4: Summary of complexity results for classes of complete CP-nets. \( M_k = (n-k)m^k + \frac{n^k}{m^k} \), as detailed in Lemma 1; \( \epsilon_{\text{max}} = (n-k)k + \binom{n}{2} \leq nk \); the value \( U_k \) is defined in Definition 10.

| Class       | VCD               | TD                  | RTD                  |
|-------------|-------------------|---------------------|----------------------|
| \( C_{ac}^n \) | \( (m-1)M_k \)   | \( n(m-1)M_k \)   | \( (m-1)M_k \)       |
| \( C_{ac}^{n-1} \) | \( m^n - 1 \)    | \( n(m-1)m^{n-1} \)| \( m^n - 1 \)          |
| \( C_{ac}^0 \)   | \( (m-1)m \)      | \( (m-1)m \)       | \( (m-1)m \)         |

**Definition 10.** Let \( n \geq 1, m \geq 2, k \leq n-1 \). When studying CP-nets over \( n \) variables, each of which has a domain of size \( m \), we denote by \( U_k \) the smallest possible size of an \((m, n-1, k)\)-universal set.

The following simple observations on universal sets will be useful for our studies.

**Lemma 2.** The smallest sizes for \((m, n-1, k)\)-universal sets, for \( k \in \{0, 1, n\} \) are as follows.

1. \( U_0 = 1 \).
2. \( U_1 = m \).
3. \( U_{n-1} = m^{n-1} \).

**Proof.** The equality \( U_0 = 1 \) is immediate by definition. \( U_1 = m \) is obvious since, independently of \( n \), the \( m \) distinct vectors \((1, \ldots, 1), (2, \ldots, 2), \ldots, (m, \ldots, m) \) form an \((m, n-1, 1)\)-universal set, and no smaller \((m, n-1, 1)\)-universal set can exist. Finally, to realize all possible assignments to the full universe of size \( n-1 \), exactly \( m^{n-1} \) vectors are required, yielding \( U_{n-1} = m^{n-1} \). ■

A striking observation from Table 4 is that VCD and RTD are equal for all values of \( m \) in \( C_{ac}^n \). Further, when \( m = 2 \) (the best-studied case in the literature), we have that TD(\( C_{ac}^{n-1} \)) equals the instance space size \( n2^{n-1} \). A close inspection of the case \( m = 2 \), as discussed in AppendixB, shows that \( X_{\text{swap}} \) has only \( n \) instances that are relevant for \( C_{ac}^n \), and \( C_{ac}^0 \) corresponds to the class of all concepts over these \( n \) instances. Thus the values of VCD, TD, and RTD are trivially equal to \( n \) in this special case.

The remainder of this section is dedicated to proving the statements from Table 4.

### 5.1 VC Dimension

We begin by studying the VC dimension, with the following main result.

**Theorem 1.** For fixed \( n \geq 1, m \geq 2 \), and any given \( k \leq n-1 \), the following statements hold for the VC dimension of the class \( C_{ac}^k \) of all complete acyclic \( k \)-bounded CP-nets over \( n \) variables with \( m \) values each, over the instance space \( X_{\text{swap}} \).
1. \( \text{VCD}(c_{ac}^{m-1}) = m^n - 1 \).
2. \( \text{VCD}(c_{ac}^0) = (m-1)n. \)
3. \( \text{VCD}(c_{ac}^k) \geq (m-1)nM_k = (m-1)(n-k)m^k + m^k - 1. \)

The proof of Theorem 1 relies on decomposing \( C_{ac}^k \) as a product of concept classes over subsets of \( X_{\text{swap}} \).

**Definition 11.** Let \( C_i \subseteq 2^{X_i} \) and \( C_j \subseteq 2^{X_j} \) be concept classes with \( X_i \cap X_j = \emptyset \). The concept class \( C_i \times C_j \subseteq 2^{X_i \cup X_j} \) is defined by \( C_i \times C_j = \{ c_i \cup c_j \mid c_i \in C_i \text{ and } c_j \in C_j \} \). For concept classes \( C_1, \ldots, C_r \), we define \( \prod_{i=1}^r C_i = C_1 \times \cdots \times C_r = (\cdots ((C_1 \times C_2) \times C_3) \times \cdots \times C_r). \)

It is a well-known and obvious fact that \( \text{VCD}(\prod_{i=1}^t C_i) = \sum_{i=1}^t \text{VCD}(C_i) \), see for example [33, Lemma 16]. For any \( v_i \in V \) and any \( \Gamma \subseteq V \setminus \{v_i\} \), we define \( C_{\text{CPT}(v_i)}^\Gamma \) to be the concept class consisting of all preference relations corresponding to some \( \text{CPT}(v_i) \) where \( Pa(v_i) = \Gamma \) and \( |\Gamma| \leq k \); here the instance space \( X \) is the set of all swap pairs \( x \in X_{\text{swap}} \) with \( V(x) = v_i \). Now, if we fix the context of \( v_i \) by fixing an assignment \( \gamma \in \Omega_\Gamma \) of all variables in \( \Gamma \), we obtain a concept class \( C_{\gamma \gamma_{v_i}}^\Gamma \), which corresponds to the set of all preference statements concerning the variable \( v_i \) conditioned on the context \( \gamma \). Its instance space is the set of all swaps \( x \) with \( V(x) = v_i \) and \( x[\Gamma] = \gamma \).

Recall that \( V = \{v_1, \ldots, v_n\} \). By \( S_n \) we denote the class of all permutations of \( \{1, \ldots, n\} \). Using this notation, we first establish two lemmas before proving Theorem 1.

**Lemma 3.** Under the same premises as in Theorem 1, we obtain
\[
C_{ac}^k = \bigcup_{\sigma \in S_n} \bigcup_{i=1}^n \bigcup_{\Gamma \subseteq \{v_{\sigma(i)}, \ldots, v_{\sigma(i-1)}\}, |\Gamma| \leq k} \bigcup_{\gamma \in \Omega_\Gamma} C_{\gamma \gamma_{v(i)}}^\Gamma.
\]

**Proof.** By definition, for \( v \in V \) and \( \Gamma \subseteq V \setminus \{v\} \), the class \( C_{\text{CPT}(v)}^\Gamma \) equals \( \prod_{\gamma \in \Omega_\Gamma} C_{\gamma \gamma_{v}}^\Gamma \).

Any concept corresponds to choosing a set \( \Gamma_v \) of parent variables of size at most \( k \) for each variable \( v \), which means \( C_{ac}^k \subseteq \bigcup_{i=1}^n \bigcup_{|\Gamma_v| \leq k} C_{\text{CPT}(v)}^\Gamma \).

By acyclicity, \( v_j \in Pa(v_i) \) implies \( v_i \notin Pa(v_j) \), so that for each concept \( c \in C_{ac}^k \) some \( \sigma \in S_n \) fulfills
\[
c \in \bigcup_{i=1}^n \bigcup_{\Gamma \subseteq \{v_{\sigma(i)}, \ldots, v_{\sigma(i-1)}\}, |\Gamma| \leq k} C_{\text{CPT}(v_{\sigma(i)})}^\Gamma.
\]

---

7 This kind of decomposition is a standard technique; the reader may look at [33, Lemma 16] for an example of its usage in the computational learning theory literature.

8 Any concept representing a preference table for \( v \) with \( Pa(v) = \Gamma \) corresponds to a union of concepts each of which represents a preference statement over \( D_v \), conditioned on some context \( \gamma \in \Omega_\Gamma \).
Thus, \( C^{k}_{ac} \subseteq \bigcup_{\sigma \in S_m} \prod_{i=1}^{m} I_{\{v_{\sigma(1)}, \ldots, v_{\sigma(i-1)}\}} \cdot \bigcap_{|\Gamma| \leq k} \bigcap_{\gamma \in \mathcal{O}_\Gamma} C^{\Gamma_{\gamma}}_{\succ^{\Gamma_{\gamma}}_{\gamma}} \).

Similarly, one can argue that every concept in the class on the right hand side represents an acyclic CP-net with parent sets of size at most \( k \). With \( C^{\Gamma_{\gamma}}_{\text{CPT}(v_i)} = \prod_{\gamma \in \mathcal{O}_\Gamma} C^{\Gamma_{\gamma}}_{\succ^{\Gamma_{\gamma}}_{\gamma}} \), the statement of the lemma follows.

\[ \text{Lemma 4. Using the notation introduced before Lemma 3, we obtain } \text{VCD}(C^{\Gamma_{\gamma}}_{\succ^{\Gamma_{\gamma}}_{\gamma}}) = m - 1 \text{ for any } v_i \in V, \ , \Gamma \subseteq V \setminus \{v_i\}, \text{ and } \gamma \in \mathcal{O}_\Gamma. \]

\[ \text{Proof. Let } v_i \in V, \Gamma \subseteq V \setminus \{v_i\}, \text{ and } \gamma \in \mathcal{O}_\Gamma. \text{ We show that } C^{\Gamma_{\gamma}}_{\succ^{\Gamma_{\gamma}}_{\gamma}} \text{ shatters some set of size } m - 1, \text{ but no set of size } m. \text{ Note first that, by definition, } C^{\Gamma_{\gamma}}_{\succ^{\Gamma_{\gamma}}_{\gamma}} \text{ is simply the class of all total orders over the domain } D_{v_i} \text{ of } v_i. \]

To show \( \text{VCD}(C^{\Gamma_{\gamma}}_{\succ^{\Gamma_{\gamma}}_{\gamma}}) \geq m - 1 \), choose any set of \( m - 1 \) swaps over \( \Gamma \cup \{v_i\} \) with fixed context \( \gamma \), in which the pairs of swapped values in \( v_i \) are \((v_1^i, v_2^i), \ldots, (v_{m-1}^i, v_m^i)\). Fix any set \( S \subseteq X_{\text{swap}} \) of \( m \) swaps over \( \Gamma \cup \{v_i\} \) with fixed context \( \gamma \). To show that \( S \) is not shattered, consider the undirected graph \( G \) with vertex set \( D_{v_i} \) in which an edge between \( v_i^j \) and \( v_i^k \) exists if \( S \) contains a swap pair flipping \( v_i^j \) to \( v_i^k \) or vice versa. \( G \) has \( m \) vertices and \( m \) edges and thus contains a cycle. The directed versions of \( G \) correspond to the labelings of \( S \); therefore some labeling \( \ell \) of \( S \) corresponds to a cyclic directed version of \( G \), which does not induce a total order over \( D_{v_i} \). Hence the labeling \( \ell \) is not realized by \( C^{\Gamma_{\gamma}}_{\succ^{\Gamma_{\gamma}}_{\gamma}} \), so that \( S \) is not shattered by \( C^{\Gamma_{\gamma}}_{\succ^{\Gamma_{\gamma}}_{\gamma}} \).

These technical observations will help to establish a lower bound on the VC dimension of classes of bounded acyclic CP-nets. For our upper bound, we need one more lemma. The lemma is a straightforward generalization of an observation made by Booth et al. \cite{5} (their Proposition 3), which states that any concept class corresponding to a set of transitive and irreflexive relations (such as a class of acyclic CP-nets) over \( \{0,1\}^n \) has a VC dimension no larger than \( 2^n - 1 \). We generalize this statement to relations over \( \{0, \ldots, m\}^n \) and, for our purpose, formulate it in CP-net terminology. Note that our statement applies to any class of consistent CP-nets, whether containing incomplete CP-nets or not, and whether acyclic or not.

\[ \text{Lemma 5 (based on \cite{5}). Let } n \geq 1, \ m \geq 2. \text{ Let } X \subseteq X_{\text{swap}} \text{ with } |X| = m^n \text{ be any set of } m^n \text{ swaps over } n \text{ variables of domain size } m. \text{ Let } C \text{ be any class of consistent CP-nets over the same domain } X_{\text{swap}}. \text{ Then } C \text{ does not shatter } X. \text{ In particular, } \text{VCD}(C) \leq m^n - 1. \]

\[ \text{Proof. Suppose } C \text{ shatters } X. \text{ Then each labeling of the swaps in } X \text{ is consistent with some CP-net in } C. \]

Observe first that the induced preference graph of any CP-net in \( C \) has exactly \( m^n \) vertices. Each labeling of a swap corresponds to a directed edge in the induced preference graph of a CP-net.\(^9\) Therefore, each labeling of all

\(^9\)For example, in Figure 2, the edge from ab\(c\) to ab\(c\) corresponds to the labeling 0 for the swap (ab\(c\), ab\(c\)), and, likewise, to the labeling 1 for the swap (ab\(c\), ab\(c\)).
elements in $X$ corresponds to a set of $m^n$ directed edges in the induced preference graph. When ignoring the direction of the edges, the edge sets resulting from these labelings are all identical, and we obtain a set of $m^n$ undirected edges over a set of exactly $m^n$ vertices. Consequently, these edges form at least one cycle. Obviously, there exists a labeling $\ell$ of the elements in $X$ that will turn this undirected cycle into a directed cycle. Since $\mathcal{C}$ shatters $X$, it contains a concept $c$ that is consistent with the labeling $\ell$, i.e., whose induced preference graph contains the directed edges corresponding to the labeling $\ell$. In particular, this concept $c$ has a cycle in its induced preference graph. This contradicts the fact that $c$ is a consistent CP-net.

Therefore, $\mathcal{C}$ does not shatter $X$. Since $X$ was chosen arbitrarily, $\mathcal{C}$ shatters no set of size $m^n$, so that $VCD(\mathcal{C}) \leq m^n - 1$. \hfill $\blacksquare$

We are now ready to give a proof of Theorem 1, which we restate here for convenience.

**Theorem 1.** For fixed $n \geq 1$, $m \geq 2$, and any given $k \leq n - 1$, the following statements hold for the VC dimension of the class $\mathcal{C}_{ac}^k$ of all complete acyclic $k$-bounded CP-nets over $n$ variables with $m$ values each, over the instance space $\mathcal{X}_{\text{swap}}$.

1. $VCD(\mathcal{C}_{ac}^{m-1}) = m^n - 1$.
2. $VCD(\mathcal{C}_{ac}^0) = (m - 1)n$.
3. $VCD(\mathcal{C}_{ac}^k) \geq (m - 1)m^k + m^k - 1$.

**Proof.** Lemma 3 states that

$$\mathcal{C}_{ac}^k = \bigcup_{\sigma \in S_n} \prod_{i=1}^n \bigcup_{\Gamma \subseteq \{v_{\sigma(1)}, \ldots, v_{\sigma(i-1)}\}, |\Gamma| \leq k} \bigcup_{\gamma \in \mathcal{O}_\Gamma} \mathcal{C}_{\Gamma \rightarrow v_{\sigma(i)}}^\gamma,$$

which yields the bound

$$VCD(\mathcal{C}_{ac}^k) \geq \max_{\sigma \in S_n} \sum_{i=1}^n \max_{\Gamma \subseteq \{v_{\sigma(1)}, \ldots, v_{\sigma(i-1)}\}, |\Gamma| \leq k} \sum_{\gamma \in \mathcal{O}_\Gamma} VCD(\mathcal{C}_{\Gamma \rightarrow v_{\sigma(i)}}^\gamma).$$

By Lemma 4, we have $VCD(\mathcal{C}_{\Gamma \rightarrow v_{\sigma(i)}}^\gamma) = m - 1$, independent of $\Gamma$ and $\gamma$, so that one obtains, for any $\sigma \in S_n$,

$$VCD(\mathcal{C}_{ac}^k) \geq (m - 1) \sum_{i=1}^n \max_{\Gamma \subseteq \{v_{\sigma(1)}, \ldots, v_{\sigma(i-1)}\}, |\Gamma| \leq k} |\mathcal{O}_\Gamma|$$

$$= (m - 1) \sum_{i=1}^n \max_{\Gamma \subseteq \{v_{\sigma(1)}, \ldots, v_{\sigma(i-1)}\}, |\Gamma| \leq k} m^{|\Gamma|}$$

$$= (m - 1)M_k.$$

It remains to verify $VCD(\mathcal{C}_{ac}^k) \leq (m - 1)M_k$ for $k \in \{0, n - 1\}$. 

24
Figure 4: Three networks each of which is subsumed by the ones to its left.

For $k = 0$, we have $M_k = n$, so let us consider any set $Y$ of size greater than $(m - 1)n$ and argue why $Y$ cannot be shattered by $C_{ac}^0$. Clearly, there exists a variable $v_i$ that is swapped in at least $m$ instances in $Y$. In order to shatter these $\geq m$ instances with the same swapped variable, a concept class of CP-nets would need to contain CP-nets in which some variables have non-empty parent sets, which is not the case for $C_{ac}^0$. Thus $Y$ is not shattered, i.e., $VCD(C_{ac}^k) \leq (m - 1)M_k$.

For $k = n - 1$, the upper bound $VCD(C_{ac}^k) \leq m^n - 1 = (m - 1)M_{n-1}$ follows immediately from Lemma 5.

### 5.2 Recursive Teaching Dimension

For studying teaching complexity, it is useful to identify concepts that are “easy to teach.” To this end, we use the notion of subsumption [7]: given CP-nets $N, N'$, we say $N$ subsumes $N'$ if for all $v_i \in V$ the following holds: If $y_1 \succ y_2$ is specified in CPT($v_i$) in $N'$ for some context $\gamma'$, then $y_1 \succ y_2$ is specified in CPT($v_i$) in $N$ for some context containing $\gamma'$. If in addition $N \neq N'$, we say that $N$ strictly subsumes $N'$.

Now let $C \subseteq C_{ac}^k$. A concept $c \in C$ is maximal in $C$ if no $c' \in C$ strictly subsumes $c$. The size of maximal concepts in $C_{ac}^k$ equals $M_k$, by definition.

**Example 7.** Let $C_{ac}^2$ be the class of all unbounded complete acyclic CP-nets over three variables $V = \{A, B, C\}$. Consider the three CP-nets in Figure 4. Clearly, $N_3$ is subsumed by $N_2$ and $N_1$, and $N_2$ is subsumed by $N_1$. Further, $N_1$ is maximal with respect to $C_{ac}^2$ and thus also maximal with respect to $\overline{C_{ac}^2}$. Each of the three CP-nets also strictly subsumes any CP-net resulting from removal of some of its statements. Since subsumption is transitive, the CP-net $N_1$ strictly subsumes any CP-net resulting from $N_3$ after removal of statements.

The following two lemmas formalize the intuition that maximal concepts are “easy to teach.” Lemma 6 gives an upper bound on the size of a smallest teaching set of any maximal concept, while Lemma 8 implies that a maximal concept is never harder to teach than any concept it subsumes.

**Lemma 6.** Let $n \geq 1$, $m \geq 2$, and $k \leq n - 1$. Let $C \subseteq C_{ac}^k$ be a subclass of the class of all complete acyclic $k$-bounded CP-nets over $n$ variables of domain size $m$. For any maximal concept $c$ in $C$, we have $TD(c, C) \leq (m - 1)\text{size}(c)$, i.e.,
Lemma 7. Let \( C \) in the class \( C_m \) context. The number of statements in a maximal concept equals \((m - 1)\text{size}(c)\) for each statement in its CPTs, so as to determine the preference \( s \) for each \( C \) concept in \( A \) teaching set for a maximal concept must contain \( m - 1 \) positively labeled swap examples in a set \( T \). For \( 1 \leq j \leq m - 1 \), the \( j \)th such example labels a pair \( x = (x_1, x_2) \) of swap outcomes with \( V(x) = v_i \), the projection of \( x \) onto \( \{v_i\} \) is \((y_j, y_{j+1})\), and the projection of \( x \) onto the remaining variables contains \( \gamma \). The set \( T \) then has cardinality \((m - 1)\text{size}(c)\) and is obviously consistent with \( N \).

It remains to show that no other CP-net in \( C \) is consistent with \( T \). Suppose some CP-net \( c' \in C \) is consistent with \( T \). Note that, for each \( i \) and each context \( \gamma \) occurring in CPT\((v_i)\) in \( c \), the information in \( T \) determines a strict total order on \( D_{v_i} \). The corresponding preferences are all entailed by \( c' \) as well, since \( c' \) is consistent with \( T \). Therefore, \( c' \) subsumes \( c \). Since \( c \) is a maximal concept, this implies that \( c' \) does not strictly subsume \( c \). Hence, \( c' = c \), i.e., \( c \) is the only concept in \( C \) that is consistent with \( T \).

Next, we determine the teaching dimension of maximal concepts in the class \( C_{ac}^k \).

Lemma 7. Let \( n \geq 1 \), \( m \geq 2 \), and \( k \leq n - 1 \). Further, let \( c \) be a maximal concept in the class \( C_{ac}^k \) of all complete acyclic \( k \)-bounded CP-nets over \( n \) variables of domain size \( m \). Then \( TD(c, C_{ac}^k) = (m - 1)M_k = (m - 1)(n - k)m^k + m^k - 1 \).

Proof. A teaching set for a maximal concept must contain \( m - 1 \) examples for each statement in its CPTs, so as to determine the preferences for each context. The number of statements in a maximal concept equals \( M_k \), which yields \( TD(c, C_{ac}^k) = TD(c, C_{ac}^k) = (m - 1)M_k \). Lemma 6 then completes the proof.

The following lemma then shows that no concept \( c' \) in \( C_{ac}^k \) has a smaller teaching dimension than any maximal concept in \( C_{ac}^k \) that subsumes \( c' \).

Lemma 8. Let \( n \geq 1 \), \( m \geq 2 \), and \( k \leq n - 1 \), and consider the class \( C_{ac}^k \) of all complete acyclic \( k \)-bounded CP-nets over \( n \) variables of domain size \( m \). Then each non-maximal \( c' \in C_{ac}^k \) is strictly subsumed by some \( c \in C_{ac}^k \) such that \( TD(c', C_{ac}^k) \geq TD(c, C_{ac}^k) \). In particular, if \( c \) is any maximal concept in \( C_{ac}^k \), then \( TD(c', C_{ac}^k) \geq TD(c, C_{ac}^k) \) for each \( c'' \in C_{ac}^k \) subsumed by \( c \).

Proof. From the graph \( G' \) for \( c' \), we build a graph \( G \) by adding the maximum possible number of edges to a single variable \( v \). As \( c' \) is not maximal, it is possible to add at least one edge without violating the indegree bound \( k \). The CP-nets corresponding to \( G \) and \( G' \) differ only in CPT\((v)\). Let \( c \) be the concept representing \( G \) and \( z \) be the size of its CPT for \( v \). A smallest teaching set \( T' \) for \( c' \) can be modified to a teaching set for \( c \) by replacing only those examples that refer to the swapped variable \( v \); \((m - 1)z \) examples suffice. To distinguish \( c' \) from \( c \), \( T' \) must contain at least \((m - 1)z \) examples referring to the swapped variable \( v \) \((m - 1)\) for each context in CPT\((v)\) in \( c \). Hence \( TD(c', C_{ac}^k) \geq TD(c, C_{ac}^k) \).
As a consequence, the easiest to teach concepts in $C^k_{ac}$ are the maximal ones.

Using these lemmas, one can determine the recursive teaching dimension of the concept classes $C^k_{ac}$, for any $k$.

**Theorem 2.** Let $n \geq 1$, $m \geq 2$, and $k \leq n - 1$. For the recursive teaching dimension of the class $C^k_{ac}$ of all complete acyclic $k$-bounded CP-nets over $n$ variables of domain size $m$, we obtain:

$$\text{RTD}(C^k_{ac}) = (m - 1)M_k = (m - 1)(n - k)m^k + m^k - 1.$$  

As before, here $C^k_{ac}$ is defined over the instance space $X_{\text{swap}}$.

**Proof.** The upper bound on the recursive teaching dimension follows from Lemma 7 and then repeated application of Lemma 6. The lower bound follows from Lemma 8 in combination with Lemma 7—these two lemmas state that the smallest teaching dimension of any concept in $C^k_{ac}$ is that of any maximal concept, and that this teaching dimension value equals $(m - 1)M_k$.

\[\blacksquare\]

### 5.2.1 An Example Illustrating Teaching Sets

For the case of complete CP-nets, we will now illustrate our results on teaching sets through examples. Let us consider the case where $V = \{A, B, C\}$, $D_A = \{a, \bar{a}\}$, $D_B = \{b, \bar{b}\}$, and $D_C = \{c, \bar{c}\}$. We consider the class $C^{n-1}_{ac}$ of all complete acyclic CP-nets defined over $V$. Figure 4 shows three concepts, $N_1$, $N_2$, and $N_3$, from this class.

In order to illustrate Lemma 6, let us first compute an upper bound on the teaching dimension of the maximal concept $N_1$. The claim is that $\text{TD}(N_1, C^{n-1}_{ac})$ is less than or equal to the size of $N_1$, which is 7. Consider a set of entailments $\mathcal{E}$ corresponding to the teaching set $T$ as described in the proof of Lemma 6. One possibility for $\mathcal{E}$ is the set consisting of the entailments $abc \Rightarrow \bar{a}bc$, $abc \Rightarrow \bar{ab}c$, $\bar{a}bc \Rightarrow \bar{ab}c$, $abc \Rightarrow \bar{ab}c$, $abc \Rightarrow \bar{ab}c$, $\bar{a}bc \Rightarrow \bar{ab}c$, and $abc \Rightarrow \bar{ab}c$. $\mathcal{E}$ is obviously consistent with $N_1$. A closer look shows that it is also a teaching set for $N_1$ with respect to $C^{n-1}_{ac}$. To see why, consider the partition $\mathcal{E} = \mathcal{E}_A \cup \mathcal{E}_B \cup \mathcal{E}_C$, where

- $\mathcal{E}_A = \{abc \Rightarrow \bar{abc}\}$,
- $\mathcal{E}_B = \{abc \Rightarrow \bar{a}bc, \bar{a}bc \Rightarrow \bar{ab}c\}$, and
- $\mathcal{E}_C = \{abc \Rightarrow \bar{ab}c, \bar{ab}c \Rightarrow \bar{a}bc\}$.

The set $\mathcal{E}_C$ shows that $C$ has at least two parents, which, in our case, have to be $A$ and $B$. As a result, $Pa(C) = \{A, B\}$ and $\text{CPT}(C)$ is defined precisely: it is the CPT with the maximum possible parent set $\{A, B\}$ and $\mathcal{E}_C$ contains a statement for every context over this parent set. Similarly, $\mathcal{E}_B$ shows that there must be at least one parent for $B$. Given the fact that $B \in Pa(A)$ we conclude that $Pa(B) = \{A\}$ as there is no other way to explain $\mathcal{E}_C$ and $\mathcal{E}_B$ together. Therefore, we have identified $\text{CPT}(B)$ precisely. In an acyclic CP-net, thus $\text{CPT}(A)$ must
be unconditional (which is consistent with $E_A$ having one entailment only). Thus, $E$ is a teaching set for $N_1$ and $TD(N_1,C_{ac}^{n-1}) \leq size(N_1) = 7$.

Now, let us illustrate Lemma 8. For this purpose, consider the teaching dimension of $N_3$ which is not maximal w.r.t. $C_{ac}^{n-1}$. According to Lemma 8, there exists a concept $N$ that subsumes $N_3$ and for which $TD(N_3,C_{ac}^{n-1}) \leq TD(N,C_{ac}^{n-1})$. We follow the proof of Lemma 8. If $G'$ is the graph of $N_3$, we construct a new graph $G$ as follows: $G$ results from $G'$ by selecting the variable $C$ and adding two incoming edges (i.e., the maximum possible number of edges) to the node labeled by this variable. A concept with such graph is $N_1$. Let $E$ be a set of entailments corresponding to a teaching set for $N_3$ with size equal to $TD(N_3,C_{ac}^{n-1})$. We claim that the number of entailments in $E$ whose swapped variable is $C$ has to be greater than or equal 4. To see this, consider the CP-net $N$ with graph $G$ where CPT($A$) and CPT($B$) are identical to the corresponding conditional preference tables in $N_3$. Moreover, for CPT($C$) in $N$, for every entailment in $e \in E$ whose swapped variable is $C$, a statement $u : c \triangleright \bar{c}$ or $u : \bar{c} \triangleright c$ is created in agreement with $e$, where $u$ provides the values of $A$ and $B$ in $e$. For instance if the subset of $E$ referring to swapped variable $C$ is $\{abc \triangleright ab\bar{c}, \bar{a}bc \triangleright \bar{a}bc\}$, then one possible table for CPT($C$) in $N$ is $\{b : c \triangleright \bar{c}, \bar{b} : \bar{c} \triangleright c\}$.

Note that $TD(N_3,C_{ac}^{n-1}) \leq TD(C_{ac}^{n-1})$. By Table 4, $TD(C_{ac}^{n-1})$ equals $(m-1)nm^{n-1}$ which, for $m = 2$ and $n = 3$ evaluates to 12.

In the sum, we discussed why $TD(N_1,C_{ac}^{n-1}) \leq 7$ and $TD(N_1,C_{ac}^{n-1}) \leq TD(N_3,C_{ac}^{n-1}) \leq 12$. It is actually the case that $TD(N_1,C_{ac}^{n-1}) = 7$, $TD(N_2,C_{ac}^{n-1}) = 9$ and $TD(N_3,C_{ac}^{n-1}) = 10$. Figure 5 shows one possible example of minimum teaching sets for the three networks. In the displayed choice of teaching sets, we selected 10 instances for teaching $N_3$, use all but one of them (some with flipped labels) to teach $N_2$ and use seven of them (some with flipped labels) to teach $N_1$. While this is not the only choice of smallest possible teaching sets for these concepts, it illustrates that some subset of the instances that are used to teach some concept, with appropriate labeling, can also be used to teach a concept that subsumes it. In Figure 5, we show which instances are removed from the teaching set of $N_3$ by crossing them out. For instance, the entailment $ab\bar{c} \triangleright \bar{a}bc$ is not needed in the teaching set of $N_2$ as the remaining examples provide enough evidence to conclude that $A \in Pa(B)$ and $A \in Pa(C)$; thus one entailment for CPT($A$) suffices for teaching CPT($A$).

5.2.2 Structural Properties of Interest to Learning-Theoretic Studies

The class $C_{ac}^{n-1}$ is interesting from a structural point of view, as its VC dimension equals its recursive teaching dimension, see Table 4. In general, the VC dimension can exceed the recursive teaching dimension by an arbitrary amount, and it can also be smaller than the recursive teaching dimension [33]. Simon and Zilles [34] posed the question whether the recursive teaching dimension can

10Alternatively, the CPT for $C$ in $N_1$ can also be used.
be upper-bounded by a function that is linear in the VC dimension. So far, the best known upper bound on the recursive teaching dimension is quadratic in the VC dimension [37].

The computational learning theory literature knows of a number of structural properties under which the VC dimension and the recursive teaching dimension coincide. In Appendix B, we study these structural properties and show that they do not apply to the class of complete unbounded acyclic CP-nets.

5.3 Teaching Dimension

Computing the teaching dimension is somewhat more involved. We will introduce some terminology first. In a teaching set, the purpose of a conflict pair for a variable \( v_i \) given a variable \( v_j \) is to demonstrate that \( v_j \) is a parent of \( v_i \).

**Definition 12.** Let \( n \geq 1 \), \( m \geq 2 \). Let \( N \) be a CP-net over \( n \) variables of domain size \( m \), and let \( v_i, v_j \) be variables in \( N \). A conflict pair for \( v_i \) given \( v_j \) is a pair \((x, x')\) of swaps such that (i) \( V(x) = V(x') = v_i \), (ii) \( x.1 \) and \( x'.1 \) differ only in \( v_j \), (iii) \( x.2 \) and \( x'.2 \) differ only in \( v_j \), and (iv) \( N \) entails \( x \) but not \( x' \).

Obviously, the variable \( v_j \) is a parent of the variable \( v_i \) if and only if a conflict pair for \( v_i \) given \( v_j \) exists. For instance, in the CP-net \( N_1 \) in Figure 4, the variable \( C \) has a conflict pair given \( A \), since \( A \) is a parent of \( C \). An example of a conflict pair here would be \(((\bar{a}bc, \bar{a}bc), (abc, abc))\). The CP-net \( N_1 \) entails \((\bar{a}bc, \bar{a}bc)\), but it does not entail \((abc, abc)\). In the same figure, in the CP-net \( N_3 \), the variable \( C \) has no parents, so that each context over \( A \) and \( B \) results in the same preference for \( C \). In other words, \( C \) has no conflict pair given \( A \) and no conflict pair given \( B \). The variable \( B \) in \( N_3 \) has a conflict pair given \( A \) but none given \( C \).

\[
\begin{array}{ccc}
abc \succ \bar{a}bc & abc \succ \bar{a}bc & abc \succ \bar{a}bc \\
\bar{a}bc \succ \bar{a}bc & \bar{a}bc \succ \bar{a}bc & a \succ \bar{a}bc \\
abc \succ abc & abc \succ abc & abc \succ abc \\
\bar{a}bc \succ \bar{a}bc & \bar{a}bc \succ \bar{a}bc & \bar{a}bc \succ \bar{a}bc \\
abc \succ abc & abc \succ abc & \bar{a}bc \succ \bar{a}bc \\
\end{array}
\]

Figure 5: Teaching sets for the three networks in Figure 4 w.r.t. the class \( \mathcal{C}_{ac}^{n-1} \).
When a variable \( v_i \) in a CP-net \( c \) has the largest possible number of parents, conflict pairs for each parent are an efficient way of determining the parent set of \( v_i \). However, to teach the learner the absence of parents, e.g., when the parent set of \( v_i \) is empty, the previously introduced notion of universal set becomes relevant. The idea is to use swaps over \( v_i \) in which the contexts form an \( (m, n - 1, k) \)-universal set in order to show that \( v_i \) has no parents at all. If however \( v_i \) does have parents, then such sets of swaps can help determine the preference relations for those parents.

**Definition 13.** Let \( n \geq 1, m \geq 2, \) and \( 0 \leq k \leq n - 1 \). Let \( V = \{ v_1, \ldots, v_n \} \) be a set of \( n \) distinct variables, where each \( v_i \in V \) has the domain \( \{ v_1^i, \ldots, v_m^i \} \) of size \( m \). Fix any \( (m, n - 1, k) \)-universal set \( U \), and let \( v_i \in V \). Then define the set \( U_{v_i} \) of contexts over \( V \setminus \{ v_i \} \) as follows.

\[
U_{v_i} = \{(v_j^i, \ldots, v_{i-1}^j, v_{i+1}^j, \ldots, v_j^n) | (j_1, \ldots, j_{i-1}, j_{i+1}, \ldots, j_n) \in U\}.
\]

We call \( U_{v_i} \) the context set imposed by \( U \) and \( v_i \).

A set \( F \subseteq X_{\text{swap}} \) of swaps is called a swap expression of \( U_{v_i} \), if \( |F| = (m - 1)|U_{v_i}| \) and, for each \( (v_{j_1}^i, \ldots, v_{j_{i-1}}^i, v_{j_{i+1}}^i, \ldots, v_{j_n}^i) \in U_{v_i} \) there is an order \( a_1 \succ a_2 \succ \ldots \succ a_m \) over the domain \( D_{v_i} = \{a_1, \ldots, a_m\} \) such that

\[
((v_{j_1}^i, \ldots, v_{j_{i-1}}^i, a_t, v_{j_{i+1}}^i, \ldots, v_{j_n}^i), (v_{j_1}^i, \ldots, v_{j_{i-1}}^i, a_{t+1}, v_{j_{i+1}}^i, \ldots, v_{j_n}^i)) \in F
\]

for all \( t \) with \( 1 \leq t \leq m - 1 \).

For example, consider \( n = 4, m = 2, \) and \( k = 2 \), and consider four variables \( A, B, C, \) and \( D \) with domains \( \{a, \bar{a}\}, \{b, \bar{b}\}, \{c, \bar{c}\}, \) and \( \{d, \bar{d}\}, \) respectively. A \((2, 3, 2)\)-universal set of vectors is, for instance

\[
U = \{(0, 0, 0), (1, 0, 1), (0, 1, 1), (1, 1, 0)\},
\]

since any projection of these vectors onto two of their three components yields all four binary vectors of length 2. The context set \( U_A \) imposed by \( U \) and \( A \) is then

\[
U_A = \{bcd, \bar{c}d, b\bar{c}d, \bar{b}\bar{c}d\}.
\]

Finally, a swap expression of \( U_A \) is given as follows.

\[
F = \{(abcd, ab\bar{c}d), (abcd, \bar{a}\bar{b}d), (ab\bar{c}d, \bar{a}\bar{b}d), (ab\bar{c}d, \bar{a}\bar{b}d)\}.
\]

If \( A \) had a third value \( \bar{a} \) in its domain, then \( F \) would be twice as large; for each swap \( (ab\bar{c}d, \bar{a}\bar{b}c^d) \) in \( F \), one would also include, for instance, \( (\bar{a}b\bar{c}d, \bar{a}\bar{b}c^d) \).

**Lemma 9.** Let \( n \geq 1, m \geq 2, \) and \( 0 \leq k \leq n - 1 \). Let \( N \) be any \( k \)-bounded CP-net over \( n \) variables of domain size \( m \). Let \( v_i \in V \) be any variable in \( N \).

The variable \( v_i \) has at least one parent in \( N \) if and only if, for every \( (m, n - 1, k) \)-universal set \( U \), the imposed context set \( U_{v_i} \) contains two distinct contexts \( \gamma, \gamma' \in \mathcal{O}_{v_i \setminus \{v_i\}} \) such that \( N \) entails \( \gamma : a \succ \bar{a} \) but not \( \gamma' : a \succ \bar{a} \), for some values \( a, \bar{a} \in D_{v_i} \).
Proof. First, suppose \( Pa(v_i) \neq \emptyset \) in \( N \) and let \( U \) be any \((m, n - 1, k)\)-universal set. Then there are \( a, \bar{a} \in D_{v_i} \) and two distinct contexts \( \alpha, \alpha' \in O_{Pa(v_i)} \) such that \( N \) entails \( \alpha : a \triangleright \bar{a} \) but not \( \alpha' : \bar{a} \triangleright a \). Since \( U \) is an \((m, n - 1, k)\)-universal set and the length of \( \alpha \) and \( \alpha' \) is at most \( k \), there are some contexts \( \beta, \beta' \in O_{\cup \{Pa(v_i)\}} \), such that the contexts \( \gamma, \gamma' \in O_{\cup \{v_i\}} \) belong to \( U_{v_i} \), where \( \gamma, \gamma' \) result from \( \alpha, \alpha' \) by extension with \( \beta \) and \( \beta' \), respectively. Clearly, \( N \) entails \( \gamma : a \triangleright \bar{a} \) but not \( \gamma' : \bar{a} \triangleright a \), since no variable over which \( \beta \) and \( \beta' \) are defined has an influence on the preference over \( v_i \).

Second, suppose that there are two distinct contexts \( \gamma, \gamma' \in O_{\cup \{v_i\}} \) such that \( N \) entails \( \gamma : a \triangleright \bar{a} \) but not \( \gamma' : \bar{a} \triangleright a \). This means that the preference over \( D_{v_i} \) is conditional, i.e., \( v_i \) has a parent in \( N \).

The following lemma gives upper and lower bounds on the teaching dimension of any single concept in the class of complete acyclic \( k \)-bounded CP-nets.

Lemma 10. \( \text{Let } n \geq 1, \ m \geq 2, \text{ and } 0 \leq k \leq n - 1. \text{ Let } C_{ac}^k \text{ be the class of all complete acyclic } k \text{-bounded CP-nets over } n \text{ variables of domain size } m, \text{ over the instance space } X_{\text{swap}}. \text{ Fix } c \in C_{ac}^k \text{ and let } e_c \text{ denote the number of edges in } c. \text{ The teaching dimension of the concept } c \text{ with respect to the class } C_{ac}^k \text{ is bounded as follows.} \)

\[(m - 1)M_k \leq TD(c, C_{ac}^k) \leq e_c + n(m - 1)U_k.\]

Proof. The lower bound follows from Lemma 8 in combination with Lemma 7—these two lemmas state that the smallest teaching dimension of any concept in \( C_{ac}^k \) is that of any maximal concept, and that this teaching dimension value equals \((m - 1)M_k\).

For the upper bound, it suffices to show that \( c \) has a teaching set of size \( e_c + n(m - 1)U_k \). Consider a set \( T \) of labeled examples defined in the following way. Fix any smallest \((m, n - 1, k)\)-universal set \( U \). For each variable \( v_i \) and each context \( \gamma \in U_{v_i} \), the set \( T \) includes \( m - 1 \) examples determining the preference order over \( D_{v_i} \) under the context \( \gamma \). Note that these examples form a swap expression of \( U_{v_i} \). In addition to these \( n(m - 1)U_k \) examples, for each variable \( v_i \) with \( Pa(v_i) \neq \emptyset \) and for each \( v \in Pa(v_i) \), the set \( T \) contains two swaps \( x, x' \) over the swapped variable \( v_i \) that form a conflict pair for \( v_i \) given \( v \). By Definition 12, the contexts in these two swaps are identical except for the variable \( v \), which is assigned in a way that the two swaps display a difference in the preference over \( D_{v_i} \). Any conflict pair \((x, x')\) for \( v_i \) given \( v \) will be sufficient for this purpose. In particular, we can choose \( x \) and \( x' \) in a way such that \( x \) is already contained in the swap expression of \( U_{v_i} \) included in \( T \). Thus, in total, we add one labeled swap for each edge in \( c \) on top of the \( n(m - 1)U_k \) examples from the swap expressions.

Clearly, the size of \( T \) is \( e_c + n(m - 1)U_k \). To show that \( T \) is a teaching set for \( c \), let \( c' \) be any concept in \( C_{ac}^k \) that is consistent with \( T \), and let \( v_i \) be any variable.

\(^{11}\text{Recall that variables do not have dummy parents, i.e., every parent listed in the parent set of } v_i \text{ must affect the preference over } D_{v_i}.\)
Let $P$ be the parent set of $v_i$ in $c$, and $P'$ the parent set of $v_i$ in $c'$. The conflict pairs in $T$ imply that $P \subseteq P'$. Now, for every context $\alpha$ over $P$, the swap expression of $U_{v_i}$ contained in $T$ shows that all context extensions of $\alpha$ to contexts of length up to $k$ give rise to the same preference order over $v_i$ in $c'$. Therefore, $P'$ cannot be a strict superset of $P$—otherwise all elements of $P' \setminus P$ would be dummy parents. So $P = P'$.

As just stated, the swap expression of $U_{v_i}$ contained in $T$ determines the preference order over $v_i$ for each context $\alpha$ over $P$ and each extension of such $\alpha$ to contexts over $k$ variables. Since every parent set in $c'$ has size at most $k$, the CPT of $v_i$ in $c'$ is thus fully determined. Consequently, $c = c'$, i.e., $T$ is a teaching set for $c$. This concludes the proof.

We illustrate the teaching set construction in the proof of Lemma 10 with an example.

**Example 8.** Consider the case $n = 4$, $m = 2$, and $k = 2$, and suppose the CP-net from Figure 6 is the target CP-net. Note that a smallest $(3, 2)$-universal set has exactly four elements, which will be the cardinality of each set $\mathcal{F}$ considered for each variable. To construct a teaching set, each variable will be treated separately.

For variable $A$, one selects a set of contexts for $A$ that are imposed by a smallest $(2, 3, 2)$-universal set. One such choice is $\{bcd, b\bar{c}d, \bar{b}c\bar{d}, \bar{b}\bar{c}\bar{d}\}$. This leads to the following swap expression $\mathcal{F}$.

$$\mathcal{F} = \{(abcd, ab\bar{c}d), (\bar{a}\bar{b}cd, \bar{a}b\bar{c}d), (ab\bar{c}d, \bar{a}\bar{b}cd), (\bar{a}\bar{b}\bar{c}d, \bar{a}\bar{b}cd)\}.$$  

The set $T$ will contain all the examples in $\mathcal{F}$, together with their correct labels, which will show that the target CP-net entails $a\gamma > \bar{a}\gamma$ for all $\gamma \in \mathcal{O}_{V \setminus \{A\}}$. This part of the teaching set $T$ hence determines that $A$ has no parents and that $a > \bar{a}$ is the only statement in $\text{CPT}(A)$.

For variable $B$, the same construction would yield a swap expression

$$\mathcal{F} = \{(abcd, ab\bar{c}d), (\bar{a}\bar{b}cd, \bar{a}b\bar{c}d), (ab\bar{c}d, \bar{a}\bar{b}cd), (\bar{a}\bar{b}\bar{c}d, \bar{a}\bar{b}cd)\}.$$
Including these swaps with their correct labels in $T$ reveals that $abcd \succ \bar{abcd}$, $\bar{abcd} \succ \bar{abc}$, $abcd \succ \bar{abc}$, $\bar{abcd} \succ \bar{abc}$. It is clear from these examples already that $B$ must have at least one parent, but the parent set itself is not yet determined. In order to show that $A$ is a parent of $B$, we select the swap $x = (abcd, \bar{abcd})$ from $T$ and pick an appropriate swap $x'$ so that $(x, x')$ forms a conflict pair for $B$ given $A$; in this case $x' = (\bar{abc}, \bar{abcd})$. We add $x'$ with its label to $T$. Now we claim that $T$ fully determines $CPT(B)$. To see this, partition $\mathcal{F}$ according to the values in $A$:

$$\mathcal{F}_{A=a} = \{(abcd, \bar{abcd}), (\bar{abc}, \bar{abcd})\} \text{ and } \mathcal{F}_{A=\bar{a}} = \{(\bar{abc}, \bar{abcd}), (\bar{abcd}, \bar{abc})\}.$$ 

Each set displays a single preference order over $D_B$, so that $B$ has no further parents. The set $T$ thus reveals that $Pa(B) = \{A\}$ and that $CPT(B)$ contains the two statements $a : b \succ b$ and $\bar{a} : b \succ b$.

For variable $C$, the construction from the proof of Lemma 10 works similarly as for $B$. Two conflict pairs reveal that $A$ and $B$ are parents of $C$, and partitioning the initial swap expression set into four subsets (according to the four possible assignments to the variables $A$ and $B$) will show the same preference order over $D_C$ in each of the four parts, which will fully determine $CPT(C)$.

Finally, variable $D$ can be treated by analogy to the case of variable $A$.

Our main result on the teaching dimension can be stated as follows.

**Theorem 3.** Let $n \geq 1$, $m = 2$, and $0 \leq k \leq n - 1$. The teaching dimension of the class of all complete acyclic $k$-bounded CP-nets over $n$ variables of domain size $m$, over the instance space $X_{\text{swap}}$, is bounded as follows.

$$n(m-1)m^k \leq n(m-1)\mathcal{U}_k \leq \text{TD}(C^{k}_{ac}) \leq \epsilon_{\max} + n(m-1)\mathcal{U}_k \leq nk + n(m-1){n-1\choose k}m^k.$$ 

For the values $k = 0, k = 1$, and $k = n - 1$, the following holds:

1. $\text{TD}(C^{0}_{ac}) = (m-1)n$.
2. $\text{TD}(C^{1}_{ac}) = (m-1)mn$.
3. $\text{TD}(C^{n-1}_{ac}) = (m-1)nm^{n-1}$.

**Proof.** To obtain a lower bound of $n(m-1)\mathcal{U}_k$ on the teaching dimension, let us first determine the teaching dimension of any complete separable CP-net with respect to the class $C^{k}_{ac}$. Consider any unconditional CPT, i.e., a CPT of the form $CPT(v_i) = \{a_1 \succ \cdots \succ a_m\}$. To distinguish a complete separable CP-net $c$ with this CPT from all other concepts in $C^{k}_{ac}$, examples are needed that show that $v_i$ has no parent. Since for CP-nets in $C^{k}_{ac}$, the variable $v_i$ can have up to $k$ parents, a teaching set for $c$ has to demonstrate the following: For each set $R$ of $k$ variables in $V \setminus \{v_i\}$, and for each of the $m^k$ possible contexts over $R$, the target CP-net $c$ entails the same order $a_1 \succ \cdots \succ a_m$. That means, a teaching set of swap pairs for $c$ must “cover” each such context over $R$, and for each
such context provide at least $m - 1$ examples so that the total order over the $m$ domain values of $v_i$ can be determined. In other words, the swap pairs with swapped variable $v_i$ that occur in the teaching set for $c$ must be partitioned into parts of size $m - 1$, where each part forms an $(m, n - 1, k)$-universal set. The same applies to each of the $n$ variables in $V$, since knowledge of the CPT for any subset of $V$ still leaves open the possibility of up to $k$ parents for each of the remaining variables. Consequently, to teach $c$, at least $n(m - 1)\mathcal{U}_k$ examples are needed. Thus, $TD(C_{ac}^k) \geq n(m - 1)\mathcal{U}_k$.

The upper bound of $e_{max} + n(m - 1)\mathcal{U}_k$ on the teaching dimension is immediate from Lemma 10.

The upper and lower bounds on $\mathcal{U}_k$ are not hard to obtain. First, $m^k \leq \mathcal{U}_k$ because an $(m, n - 1, k)$-universal set must yield $m^k$ distinct projections onto any set of at least $k$ components. Second, $\mathcal{U}_k \leq \binom{n-1}{k}m^k$ because an $(m, n - 1, k)$-universal set can always be obtained as follows: for each $k$-element subset of the $n - 1$-element universe, include $m^k$ vectors, namely one for each possible assignment of values in $\{1, \ldots, m\}$ to each of the $k$ elements in the subset.

Concerning the values for $k = 0$, $k = 1$, and $k = n - 1$, note that $\mathcal{U}_0 = 1$, $\mathcal{U}_1 = m$, and $\mathcal{U}_{n-1} = m^{n-1}$ by Lemma 2, which yields the desired lower bounds on the teaching dimension values for $C_{ac1}^0$, $C_{ac}^1$, and $C_{ac}^{n-1}$, respectively.

For $k = 0$, we have $e_{max} = 0$, so that the upper and lower bounds on the teaching dimension coincide, yielding $n(m - 1)$.

For $k = 1$, we will show an upper bound of $(m - 1)mn$, matching the lower bound. Consider any variable $v_i$ in the target CP-net. If it has no parents, then its CPT can be taught with $(m - 1)\mathcal{U}_1 = (m - 1)m$ examples. If it has a parent, then $(m - 1)m$ suitably chosen examples suffice to teach the target preference relation over $D_{v_i}$ for every possible value of the parent variable. Since $k = 1$, it is then clear that $v_i$ has no other variable, so that CPT($v_i$) is fully determined. Applying the same reasoning to every variable, we obtain a teaching set of size $n(m - 1)m$.

For $k = n - 1$, a teaching set that specifies a preference order over $m$ values using $m - 1$ suitably chosen swaps over $v$, for each variable $v$ and for each of the $m^{n-1}$ contexts over the remaining variables, uniquely determines the target concept, so that the lower bound of $(m - 1)mn^{n-1}$ is attained.

Theorem 3 implies that, for $C_{ac}^{n-1}$, the ratio of TD over instance space size $|\mathcal{X}_{swap}|$ is $\frac{2}{m}$. In particular, in the case of binary CP-nets (i.e., when $m = 2$), which is the focus of most of the literature on learning CP-nets, the teaching dimension equals the instance space size. However, maximal concepts have a teaching dimension far below the worst-case teaching dimension.

6 The Complexity of Learning Potentially Incomplete Acyclic CP-Nets

In this section, we revisit our results on the VC dimension and the teaching dimension for the case that CP-nets are potentially incomplete. This extension
Table 5: Summary of complexity results for classes of potentially incomplete CP-nets. $M_k = (n-k)m^k + \frac{m^k}{m-1}$, as detailed in Lemma 1; $\epsilon_{\text{max}} = (n-k)k + \binom{k}{2} \leq nk$; the value $U_k$ is defined in Definition 10.

| class       | VCD                        | TD                           |
|-------------|----------------------------|------------------------------|
| $\mathcal{C}_{\text{ac}}^k$ | $\geq (m-1)M_k$          | $2n(m-1)U_k \leq \text{TD} \leq \epsilon_{\text{max}} + 2n(m-1)U_k$ |
| $\mathcal{C}_{\text{ac}}$  | $n^m - 1$                  | $2n(m-1)m^{n-1}$             |
| $\mathcal{C}_{\text{ac}}$  | $(m-1)n$                   | $2(m-1)n$                   |

is useful for several reasons.

- In practice, it is likely that users will sometimes be indifferent for some attributes (under certain contexts) or that not enough information is available to completely determine user preferences. This makes the restriction to complete CP-nets unrealistic.

- We will introduce algorithms that learn both complete and incomplete CP-nets from membership queries. In order to be able to assess the optimality of these algorithms in terms of the number of queries required, we will use the value of the teaching dimension of the corresponding concept class. Likewise, we will reassess an algorithm by Koriche and Zanuttini [7] that uses equivalence and membership queries for learning potentially incomplete CP-nets; for this purpose we need to calculate the VC dimension of the corresponding concept class.

- Our extension of Theorem 1 to the case of potentially incomplete CP-nets substantially improves on (and corrects) a result by Koriche and Zanuttini [7], who present a lower bound on $\text{VCD}(\mathcal{C}_{\text{ac}}^k)$; their bound is in fact incorrect unless $k \ll n$.

The main results of this section are summarized in Table 5.

### 6.1 VC Dimension

We will first establish that Theorem 1 remains true when we allow incomplete CP-nets and use the larger instance space $\mathcal{X}_{\text{swap}}$. In particular, the VC dimension of the class of all complete acyclic $k$-bounded CP-nets (over $\mathcal{X}_{\text{swap}}$) equals the VC dimension of the class of all complete and incomplete acyclic $k$-bounded CP-nets (over $\mathcal{X}_{\text{swap}}$).

**Theorem 4.** For fixed $n \geq 1$, $m \geq 2$, and any given $k \leq n - 1$, the following VC dimension values are equal:

1. $\text{VCD}(\mathcal{C}_{\text{ac}}^k)$, i.e., the VC dimension of the class of all complete acyclic $k$-bounded CP-nets over $n$ variables with $m$ values each, over the instance space $\mathcal{X}_{\text{swap}}$. 


2. $\text{VCD}(\mathcal{C}_{ac}^k)$, i.e., the VC dimension of the class of all complete and incomplete acyclic $k$-bounded CP-nets over $n$ variables with $m$ values each, over the instance space $X_{\text{swap}}$.

Proof. Since $\text{VCD}(\mathcal{C}_{ac}^k) \geq \text{VCD}(\mathcal{C}_{ac}^k)$ is immediate from the definitions, it suffices to show that $\text{VCD}(\mathcal{C}_{ac}^k) \geq \text{VCD}(\mathcal{C}_{ac}^k)$.

Suppose $X \subseteq X_{\text{swap}}$ is a set of swap instances that is shattered by $\mathcal{C}_{ac}^k$. We need to show that there is a set $X' \subseteq X_{\text{swap}}$ such that $|X'| = |X|$ and $X'$ is shattered by $\mathcal{C}_{ac}^k$. Since $X$ is shattered by $\mathcal{C}_{ac}^k$, there is no $(o, o')$ such that both $(o, o')$ and $(o', o)$ belong to $X$. Otherwise there would be a $c \in \mathcal{C}_{ac}^k$ such that $c(o, o') = c(o', o) = 1$, which is an impossible inconsistency.

The fact that there is no $(o, o')$ such that both $(o, o')$ and $(o', o)$ belong to $X$ has two implications:

(i) Either $X \subseteq X_{\text{swap}}$ or swapping the order of some pairs in $X$ results in a set $X'$ such that $|X'| = |X|$ and $X'$ is a subset of $X_{\text{swap}}$.

(ii) Every incomplete CP-net $c \in \mathcal{C}_{ac}^k$ can be turned into a complete CP-net in $\mathcal{C}_{ac}^k$ by adding statements to some of its CPTs without changing the value of $c(x)$ for any $x \in X$. (This is because every incomplete CP-net results from a complete one by removing some CPT statement(s).) This holds in particular for the set of $2^{|X|}$ CP-nets in $\mathcal{C}_{ac}^k$ that witness the shattering of $X$: each of them can be completed in a way so that the $2^{|X|}$ completions still shatter $X$.

Combining (i) and (ii), we obtain that there is a set $X' \subseteq X_{\text{swap}}$ such that $|X'| = |X|$ and $X'$ is shattered by $\mathcal{C}_{ac}^k$.  

From Theorems 1 and 4, we immediately obtain the following corollary.

**Corollary 1.** For fixed $n \geq 1$, $m \geq 2$, and any given $k \leq n - 1$, the following statements hold for the VC dimension of the class $\mathcal{C}_{ac}^k$ of all complete and all incomplete acyclic $k$-bounded CP-nets over $n$ variables with $m$ values each, over the instance space $X_{\text{swap}}$.

1. $\text{VCD}(\mathcal{C}_{ac}^{n-1}) = m^n - 1$.

2. $\text{VCD}(\mathcal{C}_{ac}^0) = (m - 1)n$.

3. $\text{VCD}(\mathcal{C}_{ac}^k) \geq (m - 1)M_k = (m - 1)(n - k)m^k + m^k - 1$.

It is worth observing that, for the case $k = n - 1$, we can generalize our result on the VC dimension to the class of all consistent (acyclic or cyclic) CP-nets. Lemma 5 immediately implies that the VC dimension of the class of all consistent CP-nets is at most $m^n - 1$. Our Theorem 1 shows that a subclass of this class already has a VC dimension of $m^n - 1$, so that we obtain the following corollary.

**Corollary 2.** Over the instance space $X_{\text{swap}}$, the VC dimension of the class $\mathcal{C}_{cons}^k$ of complete and incomplete consistent unbounded CP-nets over $n$ variables of domain size $m$ is $m^n - 1$.  

36
Sauer’s Lemma \[45\] then bounds the number of consistent CP-nets from above by
\[
\sum_{i=0}^{m^{n-1}} \binom{|\mathcal{X}_{\text{swap}}|}{i}.
\]

One possible interpretation of Theorem 1 and Corollary 2 is that acyclic CP-nets, while less expressive, are in some sense as hard to learn as all consistent CP-nets.

To conclude our discussion of the VC dimension, we would like to remark that our results contradict a lower bound on the VC dimension of the class of all complete and incomplete acyclic CP-nets with bounded indegree that was presented in \[7\]. In Appendix A, we argue why the latter bound is incorrect.

### 6.2 Teaching Dimension

It is not difficult to adapt our previous teaching dimension results to the case of learning both complete and incomplete acyclic CP-nets, for various indegree bounds \(k\).

**Theorem 5.** Let \(n \geq 1\), \(m = 2\), and \(0 \leq k \leq n - 1\). The teaching dimension of the class of all complete and incomplete acyclic \(k\)-bounded CP-nets over \(n\) variables of domain size \(m\), over the instance space \(\mathcal{X}_{\text{swap}}\), is bounded as follows.

\[
2n(m-1)k \leq \text{TD}(\mathcal{C}_k^{ac}) \leq e_{\max} + 2n(m-1)k \leq nk + 2n(m-1) \binom{n-1}{k} m^k.
\]

For the values \(k = 0\), \(k = 1\), and \(k = n - 1\), the following holds:

1. \(\text{TD}(\mathcal{C}_0^{ac}) = 2(m-1)n\).
2. \(\text{TD}(\mathcal{C}_1^{ac}) = 2(m-1)mn\).
3. \(\text{TD}(\mathcal{C}_{n-1}^{ac}) = 2(m-1)nm^{n-1}\).

**Proof.** For the lower bound, consider the empty CP-net, i.e., a separable CP-net without any statements in any of its CPTs. We use the same argument as in the proof of Theorem 3 to show that one needs at least \(n\) swap expressions of contexts imposed by universal sets to teach this CP-net. Note however, that every swap \((o, o')\) in the resulting teaching set will be labeled 0, as the target CP-net entails no preferences. This would leave open the option that the reverse swap \((o', o)\) could be labeled 1 by the target CP-net. To exclude this option, the teaching set must contain the reverse swap \((o', o)\) as well, with the correct label 0. Therefore, the lower bound is exactly twice as large as the lower bound in Theorem 3.

For the upper bound, our argument is similar to that in the proof of Lemma 10. The construction of the teaching sets remains essentially the same: one includes (i) a swap expression of a context set imposed by a universal set, in order to
teach preferences once the parents are known, as well as to determine when a variable has no parents (on top of the already found ones), and (ii) a conflict pair \((x, x')\) for each edge in the target CP-net, where \(x\) is chosen from the swap expression set. The only difference is that we make the swap expression set twice as large by including the reverse swap \((x.2, x.1)\) to every \((x.1, x.2)\) already included. This will make sure that, in the case of a swap \((o, o')\) labeled 0, one can find out whether the reverse swap \((o', o)\) is also labeled 0 (which would mean that the target CP-net entails no preference between \(o\) and \(o'\)) or whether the reverse swap is labeled 1 (which would mean that the target CP-net entails the preference \(o' \succ o\)).

The trivial upper and lower bounds on the value \(U_k\), namely \(\binom{n-1}{k}m^k\) and \(m^k\), respectively, were already established in Theorem 3.

For the special cases of \(k = 0\), \(k = 1\), and \(k = n - 1\), one only needs to adapt the argument in the proof of Theorem 3 by forming the symmetric closure of the set of swaps used in the teaching sets, i.e., always adding the reverse swaps. ■

7 Learning from Perfect Membership Queries

In this section, we investigate the problem of learning complete CP-nets from membership queries alone, in contrast to the setting considered by Koriche and Zanuttini, where the learner asks both membership and equivalence queries [7].

A membership query, represented by a pair \((o, o')\) of objects, corresponds to asking the user directly whether they prefer \(o\) over \(o'\) or not. In an idealized model, it is expected that the user will always answer truthfully. An equivalence query, represented by a CP-net \(N\), corresponds to asking the user whether \(N\) correctly captures their preferences. The user is expected to answer correctly, where a negative response is accompanied by a witness, i.e., a pair \((o, o')\) of objects for which \(N\) entails a preference opposite to the user’s.

While in computational learning theory, beginning with Angluin’s seminal work on learning regular languages [49], the combination of equivalence and membership queries has been the most commonly investigated query scenario, we have several reasons for investigating learning from membership queries only.

Firstly, from the cognitive perspective, answering membership queries of the form “is \(o\) better than \(o'\)?” is more intuitive and poses less burden upon the user than comparing a proposed CP-net to the true one (which is the case when answering equivalence queries). Considering that membership queries can at times be perceived as too intrusive by the user, equivalence queries do not seem reasonable at all. To the best of our knowledge, the only “real-world” implementation of equivalence queries is in the area of formal methods, where an equivalence query corresponds to a guess on the semantics of a program in a given class of target programs, and a formal verification procedure (comparable to a model checker) can be run in order to verify the correctness of the guess. This works well for fully automated reasoning about finite-state systems, where such verification procedures can be implemented, but it is unreasonable to assume that users of an e-commerce system (or of some similar kind of application)
can verify equivalence queries about their preferences.\textsuperscript{12}

Secondly, Koriche and Zanuttini showed that membership queries are powerful in the sense that CP-nets are not efficiently learnable from equivalence queries alone but they are from equivalence and membership queries [7]. Thus, an immediate question is whether membership queries alone are powerful enough to efficiently learn CP-nets. In particular, we can consider learning from membership queries as an extreme case of limiting the allowable number of equivalence queries in order to investigate how much information can be obtained from the (intuitively less costly) membership queries.

Thirdly, direct queries to users, asking them to express preferences between two objects, were shown to be useful for information gathering in recommender systems, possibly more so than the typical queries asking users to rate items on a fixed scale [14]. Thus, recommender systems are a potential application of our studies on learning CP-nets from membership queries.

The complexity results presented in Sections 5 and 6 have interesting consequences on learning CP-nets from membership queries alone. In particular, it is known that the query complexity of the optimal membership query algorithm is lower-bounded by the teaching dimension of the class [18]. Therefore, in this section, we propose strategies to learn CP-nets and use the TD results to assess their optimality. In what follows, we show near-optimal query strategies for tree CP-nets and generally for classes of bounded acyclic CP-nets.

7.1 Tree CP-Nets

Koriche and Zanuttini [7] present an algorithm for learning a binary tree-structured CP-net $N^*$ that may be incomplete (i.e., it learns the superclass $C^{\text{ac}}_{m^*}$ for $m = 2$). Their learner uses at most $n_{N^*} + 1$ equivalence queries and $4n_{N^*} + e_{N^*} \log_2(n)$ membership queries, where $n_{N^*}$ is the number of relevant variables and $e_{N^*}$ the number of edges in $N^*$. We present a method for learning any CP-net in $C^{\text{ac}}_{m^*}$ (i.e., complete tree CP-nets) for any $m$, using only membership queries. Later, we will extend that method to cover also the case of incomplete CP-nets.

Recall that, for a CP-net $N$, a conflict pair w.r.t. $v_i$ is a pair $(x, x')$ of swaps such that (i) $V(x) = V(x') = v_i$, (ii) $x.1$ and $x'.1$ agree on $v_i$, (iii) $x.2$ and $x'.2$...
agree on \( v_i \), and (iv) \( N \) entails one of the swaps \( x, x' \), but not the other, cf. Definition 12. If \( v_i \) has a conflict pair \((x, x')\), then \( v_i \) has a parent variable \( v_j \) whose values in \( x \) and \( x' \) are different. Such a variable \( v_j \) can be found with \( \log_2(n) \) membership queries by binary search (each query halves the number of candidate variables with different values in \( x \) and \( x' \)) [47].

We use this binary search to learn tree-structured CP-nets from membership queries, by exploiting the following fact: if a variable \( v_i \) in a tree CP-net has a parent, then a conflict pair w.r.t. \( v_i \) exists and can be detected by asking membership queries to sort \( m \) “test sets” for \( v_i \). Let \((v_1^i, \ldots, v_m^i)\) be an arbitrary but fixed permutation of \( D_{v_i} \). Then, for all \( j \in \{1, \ldots, m\} \), a test set \( I_{i,j} \) for \( v_i \) is defined by \( I_{i,j} = \{v_j^1, \ldots, v_j^{j-1}, v_j^r, v_j^{j+1}, \ldots, v_j^m\} \mid 1 \leq r \leq m \}. \) Since \( v_i \) has no more than one parent, determining preference orders over \( m \) such test sets of size \( m \) is sufficient for revealing conflict pairs, rather than having to test all possible contexts in \( \mathcal{O}_{V \setminus \{v_i\}} \).

**Example 9.** Consider the set of variables \( V = \{A, B, C\} \) where \( D_A = \{a, a', a'', a'''\} \), \( D_B = \{b, b', b'', b'''\} \), and \( D_C = \{c, c', c'', c'''\} \). The following is one possible collection of test sets for the variable \( A \):

\[
\begin{align*}
I_{A,1} &= \{abc, a'bc, a''bc, a'''bc\} \\
I_{A,2} &= \{ab'c', a'b'c', a''b'c', a'''b'c'\} \\
I_{A,3} &= \{ab''c'', a'b''c'', a''b''c'', a'''b''c''\} \\
I_{A,4} &= \{ab'''c''', a'b'''c''', a''b'''c''', a'''b'''c'''\}
\end{align*}
\]

Clearly, a complete target CP-net imposes a total order on every \( I_{i,j} \), which can be revealed by posing enough membership queries selected from the \( \binom{m}{2} \) swaps over \( I_{i,j} \); a total of \( O(m \log_2(m)) \) comparisons suffice to determine the order over \( I_{i,j} \). This yields a simple algorithm for learning tree CP-nets with membership queries:

**Algorithm 1.** For every variable \( v_i \), determine \( Pa(v_i) \) and \( CPT(v_i) \) as follows:

1. For every value \( j \in \{1, \ldots, m\} \), ask \( O(m \log_2(m)) \) membership queries from the \( \binom{m}{2} \) swaps over \( I_{i,j} \) to obtain an order over \( I_{i,j} \). Note that, for \( m = 2 \), a single query is enough to obtain an order over \( I_{i,j} \).
2. If for all \( j_1, j_2 \in \{1, \ldots, m\} \) the obtained order over \( I_{i,j_1} \) imposes the same order on \( D_{v_i} \) as the obtained order over \( I_{i,j_2} \) does, i.e., there is no conflict pair for \( v_i \), then \( Pa(v_i) = \emptyset \). In this case, \( CPT(v_i) \) is fully determined by the queries in Step 1, following the order over \( D_{v_i} \) that is imposed by the order over any of the \( I_{i,j} \).
3. If there are some \( j_1, j_2 \in \{1, \ldots, m\} \) such that the obtained order over \( I_{i,j_1} \) imposes a different order on \( D_{v_i} \) than the obtained order over \( I_{i,j_2} \) does, i.e., there is a conflict pair \((x, x')\) for \( v_i \), then find the only parent of \( v_i \) by \( \log_2(n) \) further queries, as described by Damaschke [47]. From these queries, together with the ones posed in Step 1, \( CPT(v_i) \) is fully determined.
The procedure described by Damaschke [47] is a binary search on the set of candidates for the parent variable. Let \((x, x')\) be the conflict pair over variable \(v_i\), as found in Step 3, where

\[
x = (a_1 \ldots a_{i-1}a_ia_{i+1} \ldots a_n, \ a_1 \ldots a_{i-1}a'_ia_{i+1} \ldots a_n),
\]
\[
x' = (a'_1 \ldots a'_ia'_i a_{i+1} \ldots a'_n, \ a'_1 \ldots a'_{i-1}a'_ia'_{i+1} \ldots a'_n).
\]

Initially, each variable other than \(v_i\) is a potential parent. The set of potential parents is halved recursively by asking membership queries for swaps \((o, o')\) over \(v_i\), with \(o(v_i) = a_i, \ o'(v_i) = a'_i\), and half of the potential parent variables in \(o\) and \(o'\) having the same values as in \(x\), while the other half of the potential parent variables has values identical to those in \(x'\). (The variables that have been eliminated from the set of potential parent variables will all be assigned the same values as in \(x\).)

**Example 10.** Suppose the queries on the test sets revealed a conflict pair \((x, x')\) for the variable \(v_5\), where

\[
x = (a_1a_2a_3a_4a, \ a_1a_2a_3a_4\bar{a}),
\]
\[
x' = (a'_1a'_2a'_3a'_4a, \ a'_1a'_2a'_3a'_4\bar{a}),
\]

and

\[
a_1a_2a_3a_4a \succ a_1a_2a_3a_4\bar{a},
\]

while

\[
a'_1a'_2a'_3a'_4a \succ a'_1a'_2a'_3a'_4\bar{a}.
\]

To find the only parent of \(v_5\), Damaschke’s procedure will check whether

\[
a_1a_2a'_3a'_4a \succ a_1a_2a'_3a'_4\bar{a}.
\]

If yes, then either \(v_1\) or \(v_2\) must be the parent of \(v_5\), and one will next check whether

\[
a_1a'_2a_3a_4a \succ a_1a'_2a_3a_4\bar{a}.
\]

If yes, then \(v_1\) is the parent of \(v_5\), else \(v_2\) is the parent of \(v_5\). If, however, \(a_1a_2a'_3a'_4\bar{a} \succ a_1a_2a'_3a'_4a\), then the second query would have been to test whether \(a_1a_2a'_3a_4a \succ a_1a_2a'_3a_4\bar{a}\), in order to determine whether the parent of \(v_5\) is \(v_3\) or \(v_4\).

We formulate a result on learning complete tree CP-nets.

**Theorem 6.** Let \(n \geq 1, m \geq 2\). Algorithm 1 learns every complete tree CP-net \(N^* \in \mathcal{C}_{ac}^1\) over \(n\) variables of domain size \(m\) with

\[
O(nm^2 \log_2(m) + e_{N^*} \cdot \log_2(n))
\]

membership queries over swap examples in \(X_{\text{swap}}\), where \(e_{N^*}\) is the number of edges in \(N^*\). In particular, for \(m = 2\), Algorithm 1 requires at most \(2n + e_{N^*} \cdot \log_2(n)\) queries.
Proof. Step 1 consumes $O(m^2 \log_2 (m))$ queries and is run $n$ times, and Step 3 consumes an additional $\log_2 (n)$ for every variable $v_i$ that has a parent. In total, this sums up to $O(nm^2 \log_2 (m) + e_{N^*} \log_2 (n))$ queries. When $m = 2$, Step 1 requires only $m$ queries for each variable $v_i$, namely a single query per test set, for 2 test sets. This results in a total of $2n + e_{N^*} \log_2 (n)$ queries.

The correctness of Algorithm 1 follows from the properties of conflict pairs and tests sets. The existence of a conflict pair is equivalent to the existence of a parent, and the orders over the test sets correspond to CPT statements for all relevant contexts. The $m$ test sets as specified above are sufficient since each variable has at most one parent. Thus, the responses to the queries asked by Algorithm 1 uniquely determine the target tree CP-net.

As Theorem 6 states, for the binary case, Algorithm 1 requires $2n + e_{N^*} \log_2 (n)$ queries at most, i.e., compared to Koriche and Zanuttini’s method, when focusing only on tree CP-nets with non-empty CPTs, our method reduces the number of membership queries by a factor of 2, while at the same time dropping equivalence queries altogether.

It is a well-known fact that the teaching dimension of a concept class $C$ is a lower bound on the worst-case number of membership queries required for learning concepts in $C$ [17]. We have proven above that $\text{TD}(C_{ac}^1) = n(m - 1)\mathcal{U}_1 = nm(m - 1)$. That means that our method uses no more than on the order of $\log_2 (m) + e_{N^*} \log_2 (n)$ queries more than an optimal one, which means, asymptotically, it uses at most an extra $e_{N^*} \log_2 (n)$ queries when $m = 2$.

For $m = 2$, i.e., when CP-nets are binary, it is not hard to extend our result to the case of potentially incomplete CP-nets, at the cost of just doubling the number of queries. Perhaps not coincidentally, the teaching dimension also essentially doubles when switching from complete CP-nets to both complete and incomplete CP-nets, cf. Theorems 3 and 5.

**Theorem 7.** Let $n \geq 1$, $m = 2$. There exists an algorithm that learns every (complete or incomplete) tree CP-net $N^* \in C_{ac}^1$ over $n$ binary variables with at most $4n + 2e_{N^*} \log_2 (n)$ membership queries over swap examples in $\overline{X}_{\text{swap}}$, where $e_{N^*}$ is the number of edges in $N^*$.

Proof. The desired algorithm is a simple modification of Algorithm 1; whenever Algorithm 1 asks a query for a swap $x = (x.1, x.2)$, the new algorithm will ask two queries, namely one for $x = (x.1, x.2)$ and one for $x' = (x.2, x.1)$ (since it is no longer guaranteed that $c(x') = 1$ whenever $c(x) = 0$.) Other than that, the algorithm proceeds the same way as Algorithm 1. If both queries $x$ and $x'$ are answered with 0, then the target CP-net does not specify a preference between $x.1$ and $x.2$. Determining orders over test sets then works with at most twice the number of queries as before, even if those orders are empty. Conflict pairs still imply the existence of a parent, and still every variable has at most one parent, which can still be found with binary search. The claim then follows from Theorem 6.

Let us compare the algorithm provided by Koriche and Zanuttini [7] to ours. While our algorithm does not require equivalence queries, uses the same number
of queries in total when learning both complete and incomplete CP-nets, and is described for not necessarily binary CP-nets, Koriche and Zanuttini’s has two desirable properties that our method does not have:

- It learns CP-nets with nodes of arbitrary bounded indegree $k$, not just for $k = 1$.
- It is attribute-efficient, i.e., the number of queries it poses is polynomial in the size of the target CP-net, but only logarithmic in the number $n$ of variables.

The attribute-efficiency of their algorithm is possible only because equivalence queries are allowed. In particular, consider the incomplete CP-net without edges in which every CPT is empty. This CP-net has size 0, but to learn it only with membership queries over swap examples requires to ask queries regarding every one of its $n$ CPTs.

As for allowing arbitrary values of $k$, we will refine Algorithm 1 and present a new procedure in Section 7.3, which will efficiently learn any $k$-bounded acyclic CP-net when $m = 2$, i.e., in the binary case.

### 7.2 Re-assessment of Koriche and Zanuttini’s algorithm

To evaluate their algorithm, Koriche and Zanuttini compare its query consumption to the value $\log_2(4/3)VCD(C)$, which is a lower bound on the required number of membership and equivalence queries, known from fundamental learning-theoretic studies [16]. In lieu of an exact value for $VCD(C)$, Koriche and Zanuttini plug in a lower bound on $VCD(C)$, cf. their Theorem 6. We show in Appendix A that this lower bound is not quite correct; consequently, here we re-assess the query consumption of Koriche and Zanuttini’s algorithm.

For any $k$, their algorithm uses at most $s_{N^*} + e_{N^*} \log_2(n) + e_{N^*} + 1$ queries in total, for a target CP-net $N^*$ with $s_{N^*}$ statements and $e_{N^*}$ edges. In the worst case $s_{N^*} = M_k \leq VCD(C)$ and $e_{N^*} = \binom{k}{2} + (n - k)k$ (i.e., $N^*$ is maximal w.r.t. $C$). This yields $M_k + e_{N^*}(\log_2(n) + 1)$ queries for their algorithm, which exceeds the lower bound $\log_2(4/3)VCD(C)$ by at most $\log_2(3/2)VCD(C) + e_{N^*} \log_2(n)$. This is a more refined assessment compared to the term $e_{N^*} \log_2(n)$ that they report, and it holds for any value of $k$.

### 7.3 Bounded Binary Acyclic CP-nets

So far, our arguments for bounded acyclic CP-nets in general have been information-theoretic with no investigation of their query complexity. In this section, we show that the teaching dimension results for $C_{ac}^k$ (cf. Theorem 3) immediately yield a general strategy for learning complete acyclic CP-nets from membership queries alone, when $m = 2$. Recall that, in the binary case, $\nu_k \leq TD(C_{ac}^k) \leq e_{max} + \nu_k$, where $\nu_k$ is the size of an $(n - 1, k)$-universal set $U$ of minimum size.
For $k \geq 2$, the quantity $U_k$ is known to be $\Omega(2^k \log_2(n-1))$ and $O(k2^k \log_2(n-1))$ [50]. Thus, using $e_{\text{max}} \leq nk$, we obtain

$$\Omega(n2^k \log_2(n-1)) \ni \text{TD}(C^k_{\text{ac}}) \in O(nk2^k \log_2(n-1)).$$

Our proposed method for learning $C^k_{\text{ac}}$ from membership queries reuses the idea of universal sets in teaching sets. For each variable $v_i$, one selects a swap expression $F$ of the context set $U_{v_i}$ imposed by $U$. Then, for every variable $v_i$, one queries the elements of such set $F$. The rest of the method we propose is an adaptation of Algorithm 1. For simplicity, in the description of the algorithm, we assume all variables have the same domain $\{0, 1\}$.

**Algorithm 2.** For every variable $v_i$, determine $Pa(v_i)$ and $\text{CPT}(v_i)$ as follows:

1. Ask membership queries for all the elements of $F$. *This step amounts to $U_k$ membership queries.*

2. Initialize $P = \emptyset$: the set $P$ will always contain all the parents of $v_i$ found so far. If $|P| = k$, $Pa(v_i)$ must equal $P$, since no variable has more than $k$ parents. In this case, go to Step 7, else go to Step 3.

3. If all of the queried instances for $F$ show the same preference over the values of $v_i$, then $v_i$ has no further parents, i.e., $Pa(v_i) = P$. (This is because there are at most $k$ parents for $v_i$ and the same statement appears in every context of all potential parent sets of size $k$, cf. Lemma 9.) In this case, go to Step 7.

   Otherwise, there exist two contexts $\gamma, \gamma' \in \mathcal{O}(V \setminus (P \cup \{v_i\}))$ for which $\gamma : v'_1 \succ v'_2$ and $\gamma' : v'_2 \succ v'_1$ are answers obtained from the queries in Step 1. Clearly, $Pa(v_i)$ contains an element of $V \setminus (P \cup \{v_i\})$.

4. Apply Damaschke’s binary search procedure [47] to determine one new variable $v \in Pa(v_i)$, by analogy to the procedure used in Algorithm 1. Add $v$ to the set $P$. *This step amounts to at most $\log_2(n-1)$ membership queries.*

5. Partition $F$ into two sets $F_{v=0}$ and $F_{v=1}$. Here $F_{v=z}$ is the set of all swaps in $F$ in which the value of $v$ is $z$. Note that these swaps have all been queried in Step 1 already.

6. If all of the queried instances for $F_{v=0}$ show a single preference over the values of $v_i$, and likewise all of the queried instances for $F_{v=1}$ show a single preference over the values of $v_i$, then $Pa(v_i) = P$; go to Step 7.

   Otherwise, recursively add parents to $P$ by calling Step 3 once with $F_{v=0}$ in place of $F$ and once with $F_{v=1}$ in place of $F$, both times excluding the elements of $P$ from the binary search in Step 4.

7. From the queries posed above, $\text{CPT}(v_i)$ is fully determined.
Example 11. To illustrate Algorithm 2, consider the case \( n = 4, m = 2, \) and \( k = 2, \) and suppose the CP-net from Figure 6 is the target CP-net. Note that a smallest \((3,2)\)-universal set has exactly four elements, which will be the cardinality of each set \( F \) considered for each variable. Algorithm 2 will consider each variable in turn.

For variable \( A, \) it will first ask four queries, one for each swap pair in, say, the following set

\[
F = \{(abcd, \overline{abcd}), (abcd, \overline{abcd}), (ab\overline{cd}, \overline{ab\overline{cd}}), (ab\overline{cd}, \overline{ab\overline{cd}})\}.
\]

(Here the \((3,2)\)-universal set realizing all possible 2-dimensional context sub-vectors within 3-dimensional context vectors would be \( \{\overline{bcd}, \overline{bcd}, \overline{bcd}\}. \) The responses show that \( a\gamma \succ a\gamma \) for all \( \gamma \in O_{V\setminus A}. \) The algorithm hence detects that \( A \) has no parents and that \( a \succ \overline{a} \) is the only statement in \( \text{CPT}(A). \)

For variable \( B, \) the same procedure applied to

\[
F = \{(abcd, \overline{abcd}), (\overline{abcd}, \overline{abcd}), (\overline{ab\overline{cd}}, \overline{ab\overline{cd}}), (\overline{ab\overline{cd}}, \overline{ab\overline{cd}})\}
\]

reveals that \( ab\overline{cd} \succ ab\overline{cd}, \overline{ab\overline{cd}} \succ ab\overline{cd}, \overline{ab\overline{cd}} \succ ab\overline{cd}, \overline{ab\overline{cd}} \succ ab\overline{cd}. \) Two contexts \( \gamma \) and \( \gamma' \) for which the preference over \( D_B \) differs are, for example, \( \gamma = \overline{ac}d, \gamma' = \overline{ac}d. \) Hence \( B \) must have at least one parent among \( A, C, D, \) and applying Damaschke’s binary search to the context \( \gamma \) yields that the preference given \( \overline{ac}d \) differs from the preference given \( \overline{ac}d. \) So \( A \) must be a parent of \( B. \) Partitioning \( F \) yields

\[
F_{A=a} = \{(abcd, \overline{abcd}), (\overline{abcd}, \overline{abcd})\} \text{ and } F_{A=\overline{a}} = \{(\overline{abcd}, \overline{abcd}), (\overline{ab\overline{cd}}, \overline{ab\overline{cd}})\}.
\]

Each set displays a single preference order over \( D_B \) in the queries originally asked, so that \( B \) has no further parents. The algorithm has determined that \( \text{Pa}(B) = \{A\} \) and that \( \text{CPT}(B) \) contains the two statements \( a : b \succ \overline{b} \) and \( \overline{a} : \overline{b} \succ b. \)

For variable \( C, \) queries for pairs in the set

\[
F = \{(abcd, \overline{abcd}), (\overline{abcd}, \overline{abcd}), (abcd, \overline{abcd}), (\overline{abcd}, \overline{abcd})\}
\]

reveal that \( ab\overline{cd} \succ ab\overline{cd}, \overline{ab\overline{cd}} \succ ab\overline{cd}, \) and \( ab\overline{cd} \succ ab\overline{cd}, \) while \( ab\overline{cd} \succ ab\overline{cd}. \) In the same way as illustrated above for variable \( B, \) the algorithm finds a first parent of \( C, \) say \( A. \) This time, partitioning \( F \) into two sets \( F_{A=a} \) and \( F_{A=\overline{a}} \) will show that \( C \) must have another parent, since the two contexts \( \overline{ab\overline{d}} \) and \( \overline{abd} \) in \( F_{A=\overline{a}} \) give rise to two different preference orders over \( D_C. \) Another binary search is started, so that the parent \( B \) is found. Since \( k = 2, \) the variable \( C \) has no further parents. Consequently, the algorithm correctly infers that \( \text{Pa}(C) = \{A, B\} \) and that \( \text{CPT}(C) \) contains four statements showing that \( \overline{c} \succ c \) under any context containing \( \overline{ab}, \) while \( c \succ \overline{c} \) under the other contexts possible for \( C. \)

Finally, for variable \( D, \) Algorithm 2 would, similarly to the case of variable \( A, \) use four queries to learn that \( D \) has no parents and that \( d \) is unconditionally preferred over \( \overline{d} \) in the target CP-net.
Theorem 8. Let \( n \geq 1, m = 2, 0 \leq k \leq n - 1 \). Algorithm 2 learns every complete binary \( k \)-bounded CP-net \( N^* \in C_{ac}^k \) over \( n \) variables with
\[
O(nU_k + e_{N^*} \log_2(n))
\]
membership queries over swap examples in \( X_{\text{swap}} \), where \( e_{N^*} \) is the number of edges in \( N^* \).

Proof. The number of queries made by Algorithm 2 can be upper-bounded as follows. The only steps in which queries are made are Steps 1 and 3. Step 1 is run once for each of the \( n \) variables, each time costing \( U_k \) queries, for a total of \( nU_k \) queries. Step 3 is run once for each edge in the target CP-net, each time costing at most \( \log_2(n) \) queries, for a total of \( O(e_{N^*} \log_2(n)) \) queries. In the sum, the query consumption of Algorithm 2 is in \( O(nU_k + e_{N^*} \log_2(n)) \).

We will next prove the correctness of Algorithm 2. Note that the set \( F \) in Step 1 of the algorithm is a swap expression of a context set imposed by a \((2, n - 1, k)\)-universal set. By Lemma 9, Algorithm 2 decides correctly in Step 2 whether or not \( v_i \) has a parent. The same lemma can be applied to any part of the set \( F \) as defined in Step 5. Hence, Algorithm 2 correctly determines the parent set of each variable. Since \( F \) is a swap expression of a context set imposed by a \((2, n - 1, k)\)-universal set, and the domain is of size 2, the responses to queries in \( F \) determine a preference order for each of the relevant contexts. Consequently, Algorithm 2 is able to infer all CPT statements from those responses.

Thus, one can identify any concept \( c \in C_{ac}^k \) with \( nU_k + e_c \log_2(n) \) membership queries, where \( U_k \in O(nk2^k \log_2(n - 1)) \). A universal set of such size was proven to exist by a probabilistic argument with no explicit construction of the set [48]. However, a construction of an \((n - 1, k)\)-universal set of size \( 2^k \log_2(n - 1)k^{O(\log_2(k))} \) is reported in the literature [50]. Therefore, one can effectively learn any concept in \( C_{ac}^k \) with a number of queries bounded by \( n2^k \log_2(n - 1)k^{O(\log_2(k))} + e \log_2(n) \), which is at most a factor of \( k^{O(\log_2(k))} \) away from the teaching dimension, and thus at most a factor of \( k^{O(\log_2(k))} \) away from optimal. Moreover, when \( k^{2^k} \leq \sqrt{n} \), there is an explicit construction of a \((n, k)\)-universal set of size \( n \) [48], e.g., for \( n = 800 \) this construction is guaranteed to work with an indegree up to 3 and for \( n = 1500 \) with an indegree up to 4. This can be utilized if one is interested in learning sparse CP-nets over a large number of variables.

Finally, a simple adaptation of Algorithm 2 allows for efficient learning of all \( k \)-bounded acyclic CP-nets, whether complete or incomplete. As in the proof of Theorem 7, this comes at a cost of a factor of 2 in the number of queries.

Theorem 9. Let \( n \geq 1, m = 2, 0 \leq k \leq n - 1 \). There exists an algorithm that learns every (complete or incomplete) binary \( k \)-bounded CP-net \( N^* \in C_{ac}^k \) over \( n \) variables with
\[
O(nU_k + e_{N^*} \log_2(n))
\]
membership queries over swap examples in \( X_{\text{swap}} \), where \( e_{N^*} \) is the number of edges in \( N^* \).
Proof. The desired algorithm simulates Algorithm 2; whenever Algorithm 2 asks a query for a swap \( x = (x.1, x.2) \), the new algorithm will ask two queries, namely one for \( x = (x.1, x.2) \) and one for \( x' = (x.2, x.1) \) (since it is no longer guaranteed that \( c(x') = 1 \) whenever \( c(x) = 0 \).) Other than that, the algorithm proceeds the same way as Algorithm 2. If both queries \( x \) and \( x' \) are answered with 0, then the target CP-net does not specify a preference between \( x.1 \) and \( x.2 \). The rest of the proof is identical to that of Theorem 8.

8 Learning from Corrupted Membership Queries

So far, we have assumed that all membership queries are answered correctly. This assumption is unrealistic in many settings, especially when it comes to dealing with human experts that may have incomplete knowledge of the target function. For instance, when eliciting CP-nets from users, it could be the case that the user actually does not know which of two given outcomes is to be preferred or does not want to respond to a query. When recommender systems explicitly request information from users, be it in the form of item ratings or in the form of preference queries, not all queries will be answered and some will be answered incorrectly.

In this section, we consider two models of learning from membership queries in which the oracle does not always provide a correct answer. In particular, we consider the situation in which there is a fixed set \( L \) of instances \( x \) for which the oracle does not provide the true classification \( c^*(x) \). The set \( L \) is assumed to be chosen in advance by an adversary. There are two ways in which the oracle could deal with queries to elements in \( L \):

- **A limited oracle** returns “I don’t know” (denoted by \( \perp \)) when queried for any \( x \in L \), and returns the true label \( c^*(x) \) for any \( x \notin L \) [29, 31].
- **A malicious oracle** returns the wrong label \( 1 - c^*(x) \) when queried for any \( x \in L \), and returns the true label \( c^*(x) \) for any \( x \notin L \) [29, 30].

In either case, the oracle is persistent in the sense that it will return the same answer every time the same instance is queried. A concept class \( \mathcal{C} \) is learnable with membership queries to a limited (malicious, resp.) oracle if there is an algorithm that exactly identifies any target concept \( c^* \in \mathcal{C} \) by asking a limited (malicious, resp.) oracle a number of membership queries that is polynomial in \( n \), the size of \( c^* \), and \( |L| \) [29, 31]13.

Consider for instance the application scenario of a recommender system that explicitly asks users for preferences over pairs of items. The limited oracle then corresponds to a user who chooses to ignore some queries or who does not know which of the two presented items to prefer (perhaps since the user doesn’t know the items well enough to express a preference.) The malicious oracle is a simplified model of a user who misunderstands some queries and answers them

---

13This corresponds to the strict learning model discussed in [29, 31].
incorrectly, or who gets annoyed with queries and answers some of them on purpose incorrectly.

At first, the assumption that the oracle is persistent may seem unrealistic. However, in practice, this assumption translates into never asking the same query twice, since the system cannot possibly gain information from repeating queries to a persistent user. Repeating queries to the user is often undesirable, as it bears a high risk of annoying the user.

For practical applications, one might prefer models of probabilistic noise and learning algorithms that infer approximations to the true target based on randomness in the noise. Instead, we pursue the idealized goal of exactly identifying the target concept in the face of partial or incorrect information, which is useful for studying strict limitations of learning.

It is obvious that exact learning is not possible unless some restrictions are made on the set $L$. Note though that we assume that the learner has no prior information on the size or content of $L$. Instead, we will study under which conditions on $L$ learning complete acyclic CP-nets from limited or malicious oracles is possible even if the learner has no information on these conditions.

We will restrict our analysis to the case of binary CP-nets. Further, we will make use of the trivial observation that exact learnability with membership queries to a malicious oracle implies exact learnability with membership queries to a limited oracle.

### 8.1 Limitations on the Corrupted Set

We first establish that learning the class $C_{\text{ac}}^k$ of all complete acyclic binary $k$-bounded CP-nets over $n$ variables is impossible, both from malicious and from limited oracles, when $|L| \geq 2^{n-1-k}$.

**Proposition 1.** Let $m = 2$ and $1 \leq k \leq n-1$. If $|L| \geq 2^{n-1-k}$, then the class $C_{\text{ac}}^k$ is not learnable with membership queries to a limited oracle and not learnable with membership queries to a malicious oracle.

**Proof.** It suffices to prove non-learnability from limited oracles.

Consider a CP-net $N$ in which all the variables are unconditional, except for one variable $v_i$, which has exactly $k$ parents. Furthermore, let $\text{CPT}(v_i)$ impose the same order on the domain of $v_i$ for all contexts over $Pa(v_i)$, except for one context $\gamma$ over $Pa(v_i)$ for which $\text{CPT}(v_i)$ imposes the reverse order over the domain of $v_i$. Figure 7 shows an example of such a CP-net. Consider the set $X = \{x = (x_1, x_2) \in X_{\text{swap}} \mid V(x) = v_i \text{ and } x_1[Pa(v_i)] = x_2[Pa(v_i)] = \gamma\}$. Each swap instance $x$ in $X$ has the same values in the $k$ parent variables of $v_i$, and there are no choices for its values in the binary swap variable $v_i$ either. For each of the remaining $n - 1 - k$ variables, there are $m = 2$ choices. Hence, the cardinality of $X$ is $2^{n-1-k}$. Since $|L| \geq 2^{n-1-k}$, the adversary can choose $L$ so that $X \subseteq L$.

Then, in communication with a limited membership oracle, the learner will not be able to distinguish $N$ from the CP-net $N'$ that is equivalent to $N$ except for having the orders over the domain of $v_i$ swapped in $\text{CPT}(v_i)$. □
In essence, this negative result is due to the fact that the number of instances supporting a statement in a CPT with $k$ parents is $2^{n-1-k}$—having all these corrupted makes learning hopeless. In particular, when $k = n - 1$, i.e., for unbounded CP-nets, even a single corrupted query will make learning impossible. This is because a CPT of a variable with $n - 1$ parents may have statements that are witnessed by only a single context, and thus by only a single swap pair $x$. The adversary can choose $L = \{x\}$ and not respond to the query $x$, so that the learner cannot figure out the correct preference for the swap pair $x$.

In the remainder of this section, we will study assumptions on the structure of $L$ that will allow a learning algorithm to overcome the corrupted answers from limited or malicious oracles at the expense of a bounded number of additional queries. That means, we will constrain the adversary in its options for selecting $L$, without directly limiting the size of $L$. Roughly speaking, we no longer look at the worst possible choice of the set $L$, but at a case where the corrupted responses are “reasonably spread out.”

The goal is to be able to obtain the correct answer for any query $x$ made by our algorithms that learn from perfect oracles, simply by taking majority votes over a bounded number of additional “verification queries.” If we know that the corrupted oracle will affect only a small subset of these verification queries (small enough so that the majority vote over them is guaranteed to yield the correct label for $x$), we can use the same learning procedures as in the perfect oracle case, supplemented by a bounded number of verification queries.

Suppose that, for each $x \in X_{\text{swap}}$, we could effectively compute a small set $VQ(x) \subseteq X_{\text{swap}}$ such that the limited/malicious oracle would be guaranteed to return the correct label for $x$ on more than half of the queries for elements in $VQ(x)$. In the case of membership queries to a limited oracle, we could then simulate Algorithms 1 or 2 with the following modification:

**LIM** If a query for $x \in X_{\text{swap}}$ made by Algorithm 1 (or 2) is answered with ⊥, replace this response by the majority vote of the limited oracle’s responses to all queries over the set $VQ(x)$.

In the case of learning from malicious oracles, *every* query made by Algorithms 1 or 2 would have to be supplemented by verification queries:

![Figure 7: A CP-net $N \in C_{ac}^k$ that, for $|L| \geq 2^{n-1-k}$, cannot be distinguished from any CP-net $N'$ that differs from $N$ only in CPT($v_i$).](image-url)
MAL If a query for \( x \in X_{\text{swap}} \) is made by Algorithm 1 (or 2), respond to that query by taking the majority vote over the malicious oracle’s responses to all queries over the set \( V_Q(x) \).

If \( q \) is an upper bound on the size of the set \( V_Q(x) \), for any \( x \in X_{\text{swap}} \), the modified algorithms then would need to ask at most \( qz \) membership queries, where \( z \) is the number of queries asked by Algorithms 1 or 2.

It remains to find a suitable set \( V_Q(x) \) of verification queries for any swap pair \( x \), so that, intuitively,

- the size of \( V_Q(x) \) is not too large, and
- it is not too unreasonable an assumption that the corrupted oracle will return the true label for \( x \) on the majority of the swap pairs in \( V_Q(x) \).

To this end, we introduce some notation.

**Definition 14.** Let \( x = (x_1, x_2) \in X_{\text{swap}} \) and \( 1 \leq t \leq n - 1 \). Then we denote by \( F^t(x) \) the set of all swap instances \( x' = (x'_1, x'_2) \) with the swapped variable \( V(x') = V(x) \) and with a Hamming distance\(^{14} \) of exactly \( t \) between \( x_i \) and \( x'_i \) when restricted to \( V \setminus \{V(x)\} \), i.e.

\[
F^t(x) = \{x' \in X_{\text{swap}} \mid V(x) = V(x') = v_s \text{ and } |\{v \in V \setminus \{v_s\} \mid x[\{v\}] \neq x'[\{v\}]\}| = t\}.
\]

For example, if \( x = (abcd, \bar{abcd}) \) where \( D_{v_1} = \{a, \bar{a}\} \), \( D_{v_2} = \{b, \bar{b}\} \), \( D_{v_3} = \{c, \bar{c}\} \), \( D_{v_4} = \{d, \bar{d}\} \), then for \( t = 1 \) we get

\[
F^1(x) = \{\bar{abcd}, \bar{abcd}, (abc\bar{d}, a\bar{b}c\bar{d}), (abcd, a\bar{b}cd)\}.
\]

Each swap in that set has the same swapped variable as \( x \) (namely \( v_2 \)) and has a context that differs from that in \( x \) in exactly one variable.

There is a relationship between the entailment of an instance \( x \) and the entailments of the elements of \( F^t(x) \) for any \( t \): Given a preference table \( \text{CPT}(v_i) \), where \( v_i \) has \( k \) parents, and given \( t \), it is not hard to see that

- \( |F^t(x)| = \binom{n-1}{t} \), and
- the set \( F^t(x) \) contains \( \binom{n-1-k}{t} \) elements with the same entailment w.r.t. \( V(x) \) as in \( x \) itself. These are the instances that share the same values in the parent variables of \( V(x) \) and, hence, their entailments have to be identical.

If most of the elements in \( F^t(x) \) share the same entailment, they could be used to compensate the oracle corruption. However, queries to elements of \( F^t(x) \) again might receive corrupted answers. We will therefore impose a restriction on the overlap between the set \( L \) of instances with corrupted responses and any set \( F^t(x) \), specifically for \( t = 1 \).

We would like to constrain \( L \) so that querying the elements of \( F^1(x) \), and then picking the most frequent answer, yields the true classification of instance
and the entailments of the elements of $F$ complete acyclic will simulate Algorithm 2 with a perfect membership oracle and thus learn any
Suppose Theorem 10. is bounded to be sufficiently small.

Figure 8: An example of the entailments of an instance $x$ and $F^1(x)$ for $x = (\bar{a}bcd, \bar{a}cde)$ with $Pa(v_5) = \{v_1\}$ and CPT($v_5$) is \{a : e > e, \bar{a} : \bar{e} > e\}.

We then obtain the following learnability result for the case that the indegree $k$ is bounded to be sufficiently small.

**Theorem 10.** Suppose $n > 2k + 2$. For each swap pair $x \in X_{\text{swap}}$, let the set $VQ(x)$ of verification queries used by the LIM and MAL strategies be equal to $F^1(x)$.

1. If $|F^1(x) \cap L| \leq n - 2 - 2k$ for every $x \in X_{\text{swap}}$, then the strategy LIM will learn any complete acyclic $k$-bounded binary CP-net over $n$ variables, when interacting with a limited oracle.

2. If $|F^1(x) \cap L| \leq \lfloor \frac{n - 1}{2} \rfloor - k - 1$ for every $x \in X_{\text{swap}}$, learn any complete acyclic $k$-bounded binary CP-net over $n$ variables, when interacting with a malicious oracle.

**Proof.** Note that there are $n - 1$ elements in $F^1(x)$, of which at least $n - 1 - k$ have the same label in the target concept as $x$ and at most $k$ have the opposite label.

First, suppose $|F^1(x) \cap L| \leq n - 2 - 2k$ for every $x \in X_{\text{swap}}$, in the case of a limited oracle. Then, for every $x \in X_{\text{swap}}$, the limited oracle will respond with a label to at least $|F^1(x)| - n + 2 + 2k = 2k + 1$ of the queries for elements in $F^1(x)$. Since at most $k$ of these elements have the opposite label as $x$, the majority of these queries will return the label for $x$. Therefore, the strategy LIM will simulate Algorithm 2 with a perfect membership oracle and thus learn any complete acyclic $k$-bounded binary CP-net over $n$ variables.

Second, suppose $|F^1(x) \cap L| \leq \lfloor \frac{n - 1}{2} \rfloor$ for every $x \in X_{\text{swap}}$, in the case of a malicious oracle. Then, for every $x \in X_{\text{swap}}$, the limited oracle will correctly respond to at least $|F^1(x)| - \lfloor \frac{n - 1}{2} \rfloor + k + 1 = \lfloor \frac{n - 1}{2} \rfloor + k + 1$ of the queries for elements in $F^1(x)$. In the worst case, these $\lfloor \frac{n - 1}{2} \rfloor + k + 1$ correctly answered queries contain all of the $k$ elements of $F^1(x)$ that have the opposite label of $x$. That means that at least $\lfloor \frac{n - 1}{2} \rfloor + 1$ of the $n - 1$ queries over $F^1(x)$ (and thus a majority) return the correct label for $x$. Therefore, the strategy MAL will simulate Algorithm 2 with a perfect membership oracle and thus learn any complete acyclic $k$-bounded binary CP-net over $n$ variables.

We obtain the following corollary.

---

\(^{14}\) The Hamming distance \([51]\) between two vectors is the number of components in which they disagree.
Corollary 3. Suppose \( n > 2k + 2 \).

1. If \( |F^1(x) \cap L| \leq n - 2 - 2k \) for every \( x \in \mathcal{X}_{\text{swap}} \), then the class of all complete acyclic k-bounded binary CP-nets over \( n \) variables is learnable with membership queries to a limited oracle.

2. If \( |F^1(x) \cap L| \leq \lfloor \frac{n-1}{2} \rfloor - k - 1 \) for every \( x \in \mathcal{X}_{\text{swap}} \), then the class of all complete acyclic k-bounded binary CP-nets over \( n \) variables is learnable with membership queries to a malicious oracle.

In either case, the worst-case number of queries can be upper-bounded by \( O(n^2 \mathcal{U}_k + e_{N^*} \log_2(n)) \), which is in \( O(n^2 2^k \log_2(n) k^{O(\log_2(k))} + e_{N^*} n \log_2(n)) \).

Proof. Learnability follows from Theorem 10. The bound on the number of queries is derived from the bound in Theorem 8 by multiplying with a factor of \( O(n) \) due to the verification query set of size \( \binom{n-1}{1} = n-1 \).

In the special case of learning tree CP-nets, i.e., when \( k = 1 \), we obtain the following improved result.

Corollary 4. Suppose \( n > 4 \).

1. If \( |F^1(x) \cap L| \leq n - 4 \) for every \( x \in \mathcal{X}_{\text{swap}} \), then the class of all complete acyclic binary tree CP-nets over \( n \) variables is learnable with membership queries to a limited oracle.

2. If \( |F^1(x) \cap L| \leq \lfloor \frac{n-1}{2} \rfloor - 2 \) for every \( x \in \mathcal{X}_{\text{swap}} \), then the class of all complete acyclic binary tree CP-nets over \( n \) variables is learnable with membership queries to a malicious oracle.

In particular, in either case, the worst-case number of queries is in \( O(n^2 + e_{N^*} n \log_2(n)) \), where \( e_{N^*} \) is the number of edges in the target CP-net \( N^* \).

Proof. Learnability follows from Theorem 10. The bound on the number of queries is derived from the bound in Theorem 6 by multiplying with a factor of \( O(n) \) due to the verification query set of size \( \binom{n-1}{1} = n-1 \).

9 Conclusion

We determined exact values or non-trivial bounds on the parameters VCD and TD for the classes of all (all complete, resp.) k-bounded acyclic CP-nets for any \( k \), and used some of the insights gained thereby for the design of algorithms for learning CP-nets from membership queries, both in a setting where membership queries are always answered correctly and in settings where some answers may be missing or incorrect. The VCD values we determined correct a mistake in [7]. Further, we used the calculated TD values in order to show that our proposed algorithm for learning complete tree CP-nets from membership queries alone is close to optimal, and the calculated VCD values in order to refine an
optimality assessment of Koriche and Zanuttini’s algorithm for learning all $k$-bounded acyclic CP-nets from equivalence and membership queries.

All our results were obtained over the swap instance space. When using membership queries, a restriction of the instance space to swaps actually limits the options of the algorithm; one may want to investigate whether our algorithms can be made more efficient when given access to the full set of outcome pairs.

For fixed $k$, the number of membership queries made by our learning algorithms grows moderately with $n$, with an asymptotic worst-case behavior of $O(n \log_2(n))$ (to be multiplied by $n$ in the corrupted setting), which is at most a factor of $O(\log_2(n))$ above the best achievable, and suggests practical feasibility. In practice, running our algorithms may become prohibitive when $k$ grows too large—as is also evident in the complexity parameters we calculated for the unbounded case, i.e., when $k = n - 1$. Intuitively though, learning preferences that depend on a large number of parent variables is a problem that one might prefer to approach with approximate learning. When the dependency structure between the variables becomes so complex that unbounded CP-nets are needed to model them, one might in practice worry about overfitting the observed preferences—a simpler yet approximate CP-net may in such cases be a more desirable learning target. Knowledge of the number of queries required for exact identification of CPTs and CP-nets will then also be useful in the design of heuristic approaches for approximating a target CPT or CP-net.

On top of the VCD and the TD, we calculated another parameter, RTD, for classes of all complete $k$-bounded acyclic CP-nets. While we did not use our RTD results for optimality assessments of proposed learning algorithms, they are of interest to learning-theoretic studies, as they exhibit, with the class $C_{ac}^{n-1}$ of all complete unbounded acyclic CP-nets, the first known non-maximum class as well as the first not intersection-closed class that is interesting from an application point of view and satisfies $\text{RTD} = \text{VCD}$, cf. AppendixB. Thus further studies on the structure of CP-nets may be helpful toward the solution of an open problem concerning the general relationship between RTD and VCD [34].

Our results may also have implications on the study of consistent CP-nets. Since the class of acyclic CP-nets is less expressive than that of all consistent CP-nets, while having the same information complexity in terms of VCD, it would be interesting to find out whether learning algorithms for acyclic CP-nets can be easily adapted to consistent CP-nets in general.

Acknowledgements

The authors would like to express their gratitude to the reviewers, whose detailed comments helped to improve the quality of this article.

Eisa Alanazi’s research was supported by the Saudi Cultural Bureau in Canada. Malek Mouhoub and Sandra Zilles were supported by the Discovery Grant program of the Natural Sciences and Engineering Research Council
of Canada (NSERC). Sandra Zilles was also supported by the NSERC Canada Research Chairs program.

References

[1] J. Fürnkranz, E. Hüllermeier, Preference Learning: An Introduction, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 1–17.

[2] R. L. Keeney, H. Raiffa, Decisions with multiple objectives: preferences and value trade-offs, Cambridge university press, 1993.

[3] C. Domshlak, E. Hüllermeier, S. Kaci, H. Prade, Preferences in ai: An overview, Artificial Intelligence 175 (7) (2011) 1037 – 1052.

[4] T. E. Allen, C. Siler, J. Goldsmith, Learning tree-structured CP-nets with local search, in: Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference, 2017, pp. 8–13.

[5] R. Booth, Y. Chevaleyre, J. Lang, J. Mengin, C. Sombattheera, Learning conditionally lexicographic preference relations, in: ECAI, 2010, pp. 269–274.

[6] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, D. Poole, CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements, Journal of Artificial Intelligence Research 21 (2004) 135–191.

[7] F. Koriche, B. Zanuttini, Learning conditional preference networks, Artificial Intelligence 174 (11) (2010) 685–703.

[8] Y. Dimopoulos, L. Michael, F. Athienitou, Ceteris paribus preference elicitation with predictive guarantees, in: IJCAI, 2009, pp. 1890–1895.

[9] J. Lang, J. Mengin, The complexity of learning separable ceteris paribus preferences, in: IJCAI, 2009, pp. 848–853.

[10] J. Liu, Y. Xiong, C. Wu, Z. Yao, W. Liu, Learning conditional preference networks from inconsistent examples, IEEE Transactions on Knowledge and Data Engineering 99 (2012) 1.

[11] J. T. Guerin, T. E. Allen, J. Goldsmith, Learning CP-net preferences online from user queries, in: ADT, 2013, pp. 208–220.

[12] D. Angluin, Queries and concept learning, Machine Learning 2 (4) (1988) 319–342.

[13] F. Labernia, B. Zanuttini, B. Mayag, F. Yger, J. Atif, Online learning of acyclic conditional preference networks from noisy data, in: IEEE International Conference on Data Mining, ICDM, 2017, pp. 247–256.
[14] N. Jones, A. Brun, A. Boyer, Comparisons instead of ratings: Towards more stable preferences, in: Proceedings of the 2011 IEEE/WIC/ACM International Conference on Web Intelligence, 2011, pp. 451–456.

[15] V. N. Vapnik, A. Y. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities, Theory of Probability and its Applications 16 (2) (1971) 264–280.

[16] P. Auer, P. Long, Structural results about on-line learning models with and without queries, Machine Learning 36 (3) (1999) 147—181.

[17] S. A. Goldman, M. J. Kearns, On the complexity of teaching, Journal of Computer and System Sciences 50 (1995) 20–31.

[18] D. Angluin, Queries revisited, in: N. Abe, R. Khardon, T. Zeugmann (Eds.), Algorithmic Learning Theory, Vol. 2225 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2001, pp. 12–31.

[19] F. Labernia, F. Yger, B. Mayag, J. Atif, Query-based learning of acyclic conditional preference networks from noisy data, in: DA2PL, 2016.

[20] F. Labernia, F. Yger, B. Mayag, J. Atif, Query-based learning of acyclic conditional preference networks from contradictory preferences, EURO J Decis Process 6 (2018) 39–59.

[21] Y. Chevaleyre, F. Koriche, J. Lang, J. Mégine, B. Zanuttini, Learning ordinal preferences on multiattribute domains: the case of CP-nets, in: Preference Learning, Springer-Verlag, 2010, pp. 273–296.

[22] S. Zilles, S. Lange, R. Holte, M. Zinkevich, Models of cooperative teaching and learning, Journal of Machine Learning Research 12 (2011) 349–384.

[23] E. Alanazi, M. Mouhoub, S. Zilles, The complexity of learning acyclic CP-nets, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI), 2016, pp. 1361–1367.

[24] F. Koriche, B. Zanuttini, Learning conditional preference networks with queries, in: IJCAI, 2009, pp. 1930–1935.

[25] J. Liu, Z. Yao, Y. Xiong, W. Liu, C. Wu, Learning conditional preference network from noisy samples using hypothesis testing, Knowledge-Based Systems 40 (2013) 7–16.

[26] L. Michael, E. Papageorgiou, An empirical investigation of ceteris paribus learnability, in: IJCAI, 2013.

[27] D. Bigot, J. Mengin, B. Zanuttini, Learning probabilistic cp-nets from observations of optimal items, in: STAIRS 2014 - Proceedings of the 7th European Starting AI Researcher Symposium, Prague, Czech Republic, August 18-22, 2014, 2014, pp. 81–90.
[28] M. Haqqani, X. Li, An evolutionary approach for learning conditional preference networks from inconsistent examples, in: Proceedings of the 13th International Conference on Advanced Data Mining and Applications, ADMA 2017, 2017, pp. 502–515.

[29] D. Angluin, M. Krikis, R. H. Sloan, G. Turán, Malicious omissions and errors in answers to membership queries, Machine Learning 28 (2-3) (1997) 211–255.

[30] R. Bennet, N. H. Bshouty, Learning attribute-efficiently with corrupt oracles, Theor. Comput. Sci. 387 (1) (2007) 32–50.

[31] L. Bishl, N. H. Bshouty, L. Khoury, Learning with errors in answers to membership queries, J. Comput. Syst. Sci. 74 (1) (2008) 2–15.

[32] S. Hanneke, The optimal sample complexity of PAC learning, Journal of Machine Learning Research 17 (2016) 38:1–38:15.

[33] T. Doliwa, G. Fan, H. U. Simon, S. Zilles, Recursive teaching dimension, VC-dimension and sample compression, Journal of Machine Learning Research 15 (2014) 3107–3131.

[34] H. U. Simon, S. Zilles, Open problem: Recursive teaching dimension versus VC dimension, in: COLT, 2015, pp. 1770–1772.

[35] X. Zhu, Machine teaching: An inverse problem to machine learning and an approach toward optimal education, in: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015, pp. 4083–4087.

[36] X. Zhu, A. Singla, S. Zilles, A. N. Rafferty, An overview of machine teaching, CoRR abs/1801.05927.

[37] L. Hu, R. Wu, T. Li, L. Wang, Quadratic upper bound for recursive teaching dimension of finite VC classes, in: COLT, 2017.

[38] J. Goldsmith, J. Lang, M. Truszczynski, N. Wilson, The computational complexity of dominance and consistency in cp-nets, J. Artif. Intell. Res. (JAIR) 33 (2008) 403–432.

[39] D. Bigot, B. Zamutini, H. Fargier, J. Mengin, Probabilistic conditional preference networks, in: Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, UAI 2013, Bellevue, WA, USA, August 11-15, 2013, 2013.

[40] M. J. Kearns, U. V. Vazirani, An Introduction to Computational Learning Theory, MIT Press, Cambridge, MA, USA, 1994.

[41] N. Littlestone, Learning quickly when irrelevant attributes abound: a new linear threshold algorithm, Machine Learning 2 (4) (1988) 245—318.
A Revising a Lower Bound on VCD from [7]

Let $\overline{C}_{ac}^{2,k,e}$ be the class of all binary acyclic $k$-bounded CP-nets with at most $e$ edges, where $0 \leq k < n$ and $k \leq e \leq \binom{n}{2}$ (note that this class includes both complete and incomplete CP-nets.) Koriche and Zanuttini [7, Theorem 6] gave a lower bound on $\text{VCD}(\overline{C}_{ac}^{2,k,e})$ over the swap instance space. In particular, setting $u = \lceil \frac{e}{k} \rceil$ and $r = \lceil \log_2 \frac{n-u}{k} \rceil$, they claimed that $\text{VCD}(\overline{C}_{ac}^{2,k,e})$ is lower-bounded by

$$
\text{LB} = \begin{cases} 
1, & \text{if } k = 0, \\
u(r + 1), & \text{if } k = 1, \\
u(2^k + k(r - 1) - 1), & \text{if } k > 1. 
\end{cases}
$$

We claim that this bound is not generally correct for large values of $k$ and $e$. 

A. G. Valiant, A theory of the learnable, Commun. ACM 27 (11) (1984) 1134–1142.

A. Blumer, A. Ehrenfeucht, D. Haussler, M. K. Warmuth, Learnability and the Vapnik-Chervonenkis dimension, Journal of the ACM 36 (4) (1989) 929–965.

A. Shinohara, S. Miyano, Teachability in computational learning, New Generation Computing 8 (4) (1991) 337–347.

N. Sauer, On the density of families of sets, Journal of Combinatorial Theory, Series A 13 (1) (1972) 145–147.

B. I. P. Rubinstein, J. H. Rubinstein, A geometric approach to sample compression, Journal of Machine Learning Research 13 (2012) 1221–1261.

P. Damaschke, Adaptive versus nonadaptive attribute-efficient learning, Machine Learning 41 (2) (2000) 197–215.

S. Jukna, Extremal Combinatorics: With Applications in Computer Science, 1st Edition, Springer Publishing Company, Incorporated, 2010.

D. Angluin, Learning regular languages from queries and counterexamples, Information and Computation 75 (1987) 87–106.

G. Seroussi, N. H. Bshouty, Vector sets for exhaustive testing of logic circuits, IEEE Trans. Information Theory 34 (3) (1988) 513–522.

R. W. Hamming, Error detecting and error correcting codes, The Bell System Technical Journal 29 (1950) 147–160.

S. Moran, M. K. Warmuth, Labeled compression schemes for extremal classes, in: Proceedings of the 27th International Conference on Algorithmic Learning Theory, ALT, 2016, pp. 34–49.
For any given \( k \), it is easy to see that there is a target concept in \( \mathcal{C}_{ac}^{2,k,e} \) whose graph has \( e_{\text{max}} = (n - k)k + \binom{k}{2} \) edges. We can always construct such a graph \( G \) as indicated in the proof of Lemma 1: Let \( V_1 \) and \( V_2 \) be a partition over the \( n \) vertices of \( G \) where \( |V_1| = n - k \) and \( |V_2| = k \). Add an edge from each node in \( V_2 \) to each node in \( V_1 \). This results in \((n - k)k \) edges and \( G \) is clearly acyclic with indegree exactly \( k \) for every element in \( V_1 \). For the remaining \( \binom{k}{2} \) edges, let \( < \) be any total order on \( V_2 \); now the edge \((v, w)\) is added to \( G \) if and only if \( v < w \).

Next, we evaluate the lower bound \( \text{LB} \) for \( k > 1 \) when the target concept has \( e_{\text{max}} \) edges in its graph:

\[
u(2^k + k(r - 1) - 1) = \left\lfloor \frac{e}{k} \right\rfloor (2^k + k(\lceil \log_2 \left( \frac{n - \frac{e}{k}}{k} \right) \rceil - 1) - 1)
\]

(setting \( e = e_{\text{max}} \))

\[
= \frac{k}{(k)} + (n - k)k(2^k + k(\lceil \log_2 \left( \frac{n - \frac{k}{k}}{k} \right) \rceil - 1) - 1)
\]

\[
= \frac{1}{2}(2n - k - 1)(2^k + k(\lceil \log_2 \left( \frac{k + 1}{2k} \right) \rceil - 1) - 1)
\]

\[
= \frac{1}{2}(2n - k - 1)(2^k - 2k - 1)
\]

When \( k = n - 1 \), this term evaluates to \( n2^{n-2} - n^2 + \frac{n}{2} \) which becomes larger than \( 2^n - 1 \) for \( n > 6 \). More generally, given \( k = n - c \) for some constant \( 0 < c < n - 1 \), the term equals

\[
(n + (c - 1))2^{n-(c+1)} - n^2 - (c - 1)n + \frac{(2c - 1)(n + (c - 1))}{2}
\]

which exceeds \( 2^n - 1 \) for small values of \( c \) and \( n > 6 \). Lemma 5, however, states that \( 2^n - 1 \) is an upper bound on the VC dimension of \( \mathcal{C}_{ac}^{n,k,e} \). Consequently, there must be a mistake in the bound \( \text{LB} \). It appears that the mistake was caused by Koriche and Zanuttini assuming that there are acyclic CP-nets with \( \frac{e}{k}2^k \) statements, which is not true for large values of \( k \) and \( e \).

### B Structural Properties of CP-net Classes

The computational learning theory literature knows of a number of structural properties under which the VC dimension and the recursive teaching dimension coincide. The purpose of this section is to investigate which of these structural properties apply to certain classes of acyclic CP-nets. The main result is that the class \( \mathcal{C}_{ac}^{n,k} \) of all complete unbounded acyclic CP-nets does not satisfy any of the known general structural properties sufficient for VCD and RTD to coincide; therefore this class may serve as an interesting starting point for formulating new general properties of a concept class \( \mathcal{C} \) that are sufficient for establishing \( \text{RTD}(\mathcal{C}) = \text{VCD}(\mathcal{C}) \).
We first define the structural properties to be studied in this section. These basic notions from the computational learning theory literature can, for example, also be found in [33].

**Definition 15.** Let \( C \) be any finite concept class over a fixed instance space \( \mathcal{X} \). Let \( d \) denote the VC dimension of \( C \).

1. The class \( C \) is maximum if \( |C| = \sum_{i=0}^{d} \binom{|\mathcal{X}|}{i} \), i.e., if the cardinality of \( C \) meets Sauer’s bound [45] with equality.

2. The class \( C \) is maximal if \( \text{VCD}(C \cup \{c\}) > d \) for any concept \( c \) over \( \mathcal{X} \) that is not contained in \( C \), i.e., if adding any concept \( c \not\in C \) to the class will increase its VC dimension.

3. The class \( C \) is said to be an extremal class if \( C \) strongly shatters every set that it shatters. \( C \) strongly shatters \( S \subseteq \mathcal{X} \) if there is a subclass \( C' \) of \( C \) that shatters \( S \) such that all concepts in \( C' \) agree on the labeling of all instances in \( \mathcal{X} \setminus S \).

4. The class \( C \) is intersection-closed if \( c \cap \bar{c} \in C \) for any two concepts \( c, \bar{c} \in C \).

Intuitively, for a given VC dimension and a given instance space size, maximum classes are the largest possible classes in terms of cardinality, while maximal classes are largest with respect to inclusion. Every maximum class is also maximal but the converse does not hold [40]. Every maximum class is also an extremal class, but not vice versa [52].

It was proven that any finite maximum class \( C \) that can be corner-peeled by the algorithm proposed by Rubinstein and Rubinstein [46] satisfies \( \text{RTD}(C) = \text{VCD}(C) \) [33]. The same equality holds when \( C \) is of VC dimension 1 or when \( C \) is intersection-closed [33]. Therefore, if \( C_{ac}^{n-1} \) were to fulfill any of these structural properties, it would not be surprising that the recursive teaching dimension and the VC dimension coincide for \( C_{ac}^{n-1} \). In this section, we will demonstrate that none of these structural properties are fulfilled for any of the classes \( C_{ac}^{k} \) studied in our manuscript, with the exception of some trivial special cases.

The main results of this section are summarized in Table 6. For the proofs of some of these results, the following definition and lemma will be useful.

**Definition 16.** Let \( c \) be any concept over an instance space \( \mathcal{X} \). Then the complement of \( c \), denoted by \( c^{co} \), is the concept over \( \mathcal{X} \) that contains exactly those instances not contained in \( c \), i.e.,

\[
c^{co}(x) = 1 - c(x) \text{ for all } x \in \mathcal{X}.
\]

\[\text{For example, suppose a concept } c \text{ over an instance space } \{x_1, x_2, x_3, x_4\} \text{ is represented as the vector } (c(x_1), c(x_2), c(x_3), c(x_4)). \text{ Then the concepts } (0, 0, 0, 0), (0, 0, 0, 1), (0, 0, 1, 0), (1, 0, 1, 1) \text{ shatter the set } \{x_3, x_4\}, \text{ but they do not strongly shatter it, because they do not all agree on their labelings in the remaining instances (they disagree in } x_1\text{.) By comparison, } \{(1, 0, 0, 0), (1, 0, 0, 1), (1, 0, 1, 0), (1, 0, 1, 1)\} \text{ strongly shatters the set } \{x_3, x_4\}.\]
Table 6: Structural properties of CP-net concept classes.

| class                  | maximum | maximal | intersection-closed | extremal |
|------------------------|---------|---------|---------------------|---------|
| $C_{ac}^0$ with $m = 2$ (over $X_{sep}$) | yes     | yes     | yes                 | yes     |
| $C_{ac}^0$ with $m > 2$ (over $X_{sep}$ or $X_{swap}$) | no      | no      | no                  | no      |
| $C_{ac}^k$ with $m \geq 2$, $1 \leq k \leq n - 1$ | no      | no      | no                  | no      |

Lemma 11. Let $n \geq 1$, $m \geq 2$, and $k \in \{0, \ldots, n-1\}$. If $c$ belongs to the class $C_{ac}^k$ of all complete acyclic $k$-bounded CP-nets with $n$ variables of domain size $m$, over the instance space $X_{swap}$, then also the complement $c^{co}$ of $c$ belongs to $C_{ac}^k$.

Proof. The CP-net $N^{co}$ corresponding to $c^{co}$ is obtained from the CP-net $N$ corresponding to $c$ by reversing each preference statement in each CPT of $N$. Obviously, in $N^{co}$, each variable has the same parent set as in $N$, so that $c^{co} \in C_{ac}^k$. ■

First of all, we discuss separable CP-nets. Note that there is a one-to-one correspondence between separable CP-nets and $n$-tuples of total orders of $\{1, \ldots, m\}$: since there are no dependencies between the variables, each separable CP-net simply determines an order over the $m$ domain values of a variable, and it does so for each variable independently. This way, for a separable CP-net, the swap example $(v_1 y_2 \ldots v_l \alpha v_{l+1} \ldots v_n, v_1 y_2 \ldots v_l \beta v_{l+1} \ldots v_n)$ will always be labeled exactly the same way as any other swap example $x = (x.1, x.2)$ whose swapped variable is $v_l$ and for which the $l$th positions of $x.1$ and $x.2$ are $\alpha$ and $\beta$, respectively. We may therefore consider separable CP-nets over an instance space that is a proper subset of the set $X_{swap}$ of all swaps without “redundancies”, namely one that contains exactly one swap example for each pair of domain values of each variable. Assuming a fixed choice of such pairs, we denote this subset of $X_{swap}$ by $X_{sep}$ and remark that $|X_{sep}| = \binom{m}{2} n$. Note that, for the class of separable CP-nets, each instance $x \in X_{swap} \setminus X_{sep}$ is redundant in the following sense: there exists some instance $x' \in X_{sep}$ such that

- either $c(x) = c(x')$ for all $c \in C_{ac}^0$,
- or $c(x) = 1 - c(x')$ for all $c \in C_{ac}^0$.

It is now easy to see the following for the binary case.

Proposition 2. Let $n \geq 1$, $m = 2$. Over the instance space $X_{sep}$, the concept class $C_{ac}^0$ of all complete binary separable CP-nets with $n$ variables is maximum (in particular, also maximal and extremal) and intersection-closed.

Proof. Note that we are considering the case $m = 2$ and $k = 0$. Given the instance space $X_{sep}$, the claim is immediate from the fact that $VCD(C_{ac}^0) = (m-1)n = n = \binom{m}{2} n = |X_{sep}|$, which means that $C_{ac}^0$ is the class of all possible concepts over $X_{sep}$. ■
In the non-binary case, we will see below that the situation is different. We start by showing that the class of complete separable CP-nets is not intersection-closed in the non-binary case.

Up to now, a subtlety in the definition of intersection-closedness has been ignored in our discussions. This is best explained using a very simple example. Consider a concept class \(\mathcal{C}\) over \(\mathcal{X} = \{x_1, x_2\}\) that contains the concepts \(\{x_1\}\), \(\{x_2\}\), and the empty concept. Obviously, \(\mathcal{C}\) is intersection-closed. From a purely learning-theoretic point of view, and certainly for the calculation of any of the information complexity parameters studied above, \(\mathcal{C}\) is equivalent to the class \(\mathcal{C}' = \{\{x_2\}, \{x_1\}, \{x_1, x_2\}\}\) that results from \(\mathcal{C}\) simply when flipping all labels. This class is no longer intersection-closed, as it does not contain the intersection of \(\{x_2\}\) and \(\{x_1\}\). Likewise, any two concept classes \(\mathcal{C}\) and \(\mathcal{C}'\) over some instance space \(\mathcal{X}\) are equivalent if one is obtained from the other by "inverting" any of its instances, i.e., by selecting any subset \(X \subseteq \mathcal{X}\) and replacing \(c(x)\) by \(1 - c(x)\) for all \(c \in \mathcal{C}\) and all \(x \in X\).

When defining the instance space \(\mathcal{X}_{\text{swap}}\), we did not impose any requirements, for any swap pair \((o, o')\), as to whether \((o, o')\) or \((o', o)\) should be included in \(\mathcal{X}_{\text{swap}}\). So, in fact, \(\mathcal{X}_{\text{swap}}\) could be any of a whole class of instance spaces, all of which are equivalent for the purposes of calculating the information complexity parameters we studied. Thus, to show that \(\mathcal{C}_{\text{ac}}^0\) is not intersection-closed, we have to consider all possible combinations in which the outcome pairs in \(\mathcal{X}_{\text{swap}}\) could be arranged. In the proof of Proposition 3 this requires a distinction of only two cases, while more cases need to be considered when proving that \(\mathcal{C}_{\text{ac}}^k\) is not intersection-closed for \(k > 0\) (see Proposition 6 below.)

**Proposition 3.** Let \(n \geq 1, m > 2\). Then the class \(\mathcal{C}_{\text{ac}}^0\) of all complete separable CP-nets with \(n\)-variables of domain size \(m\) is not intersection-closed (neither over the instance space \(\mathcal{X}_{\text{sep}}\) nor over the instance space \(\mathcal{X}_{\text{swap}}\)).

**Proof.** Let \(v \in V\) be any variable. Since \(m > 2\), the domain of \(v\) contains three pairwise distinct values \(a_1, a_2,\) and \(a_3\) such that the instance space \((\mathcal{X}_{\text{sep}}\) or \(\mathcal{X}_{\text{swap}}\)) contains swap examples \(x_1, x_2,\) and \(x_3\), each with the swapped variable \(v\), and one of the following two cases holds:

- **Case 1.** The projections \(x_i[v]\) of the swap pairs \(x_i\) to the swapped variable \(v\) are
  \[
  x_1[v] = (a_1, a_2), \quad x_2[v] = (a_1, a_3), \quad x_3[v] = (a_2, a_3).
  \]

- **Case 2.** The projections \(x_i[v]\) of the swap pairs \(x_i\) to the swapped variable \(v\) are
  \[
  x_1[v] = (a_1, a_2), \quad x_2[v] = (a_3, a_1), \quad x_3[v] = (a_2, a_3).
  \]

It remains to show that, in either case, we can find two separable CP-nets whose intersection (as concepts over the given instance space) is not a separable CP-net.

In Case 1, let \(c_1\) be a CP-net entailing \(a_1 \succ a_2, a_1 \succ a_3,\) and \(a_3 \succ a_2,\) while \(c_2\) entails \(a_2 \succ a_1, a_1 \succ a_3,\) and \(a_2 \succ a_3.\) Both these sets of entailments...
can be realized by separable CP-nets. Both $c_1$ and $c_2$ label $x_2$ with 1 (as they both prefer $a_1$ over $a_3$), but they disagree in their labels for $x_1$ and $x_3$. The intersection of $c_1$ and $c_2$ thus labels $x_2$ with 1, while it labels both $x_1$ and $x_3$ with 0. This corresponds to a preference relation in which $a_2$ is preferred over $a_1$, then $a_1$ is preferred over $a_3$, but $a_3$ is preferred over $a_2$. This cycle cannot be realized by a separable CP-net, i.e., $c_1, c_2 \in C^0_{ac}$ while $c_1 \cap c_2 \notin C^0_{ac}$.

In Case 2, let $c_1$ be a CP-net entailing $a_1 \succ a_2, a_3 \succ a_1$, and $a_3 \succ a_2$, while $c_2$ entails $a_2 \succ a_1, a_1 \succ a_3$, and $a_2 \succ a_3$. Both these sets of entailments can be realized by separable CP-nets. The concepts $c_1$ and $c_2$ disagree in their labels for all of $x_1, x_2,$ and $x_3$. The intersection of $c_1$ and $c_2$ thus labels all of $x_1, x_2,$ and $x_3$ with 0. This corresponds to a preference relation in which $a_3$ is preferred over $a_1$, then $a_1$ is preferred over $a_3$, but $a_3$ is preferred over $a_2$. This cycle cannot be realized by a separable CP-net, i.e., $c_1, c_2 \in C^0_{ac}$ while $c_1 \cap c_2 \notin C^0_{ac}$.

Proposition 3 states that the class of non-binary complete separable CP-nets is not intersection-closed, and that this result holds true even when restricting the instance space to the set $X_{sep}$. It turns out that the same class is not maximal or extremal, either, over the same choices for instance spaces. Since maximum classes are always extremal, this also implies that the class of non-binary complete separable CP-nets is not maximum. The proofs of these claims rely on Lemma 11 and establish the same claims for the class $C_{ac}^k$ for any $k \in \{1, \ldots, n - 1\}$ and any $m \geq 2$. Since for $k > 0$, the set $X_{swap}$ is the more reasonable instance space, in the remainder of this section we always assume that a concept class is given over $X_{swap}$. For the class of separable CP-nets though, every proof we provide will go through without modification when $X_{swap}$ is replaced by $X_{sep}$.

First, we show that maximality no longer holds for the class of separable CP-nets, when $m > 2$, and neither for $C_{ac}^k$, when $k > 0$ and $m \geq 2$.

**Proposition 4.** Let $n \geq 1$, $m \geq 2$, and $0 \leq k \leq n - 1$, where $(m, k) \neq (2, 0)$. Then the concept class $C_{ac}^k$ of all complete $k$-bounded acyclic CP-nets over $n$ variables of domain size $m$, over the instance space $X_{swap}$, is not maximal, and, in particular, $C_{ac}^k$ is not maximum.

**Proof.** We need to prove that there exists some concept $c$ over $X_{swap}$ (not necessarily corresponding to a consistent CP-net) such that $\text{VCD}(C_{ac}^k \cup \{c\}) = \text{VCD}(C_{ac}^k)$. We will prove an even stronger statement, namely: for every subset $X \subseteq X_{swap}$ with $|X| = \text{VCD}(C_{ac}^k) + 1$ and every set $C$ of concepts such that $C_{ac}^k \cup C$ shatters $X$, we have $|C| \geq 2$.

Let $X \subseteq X_{swap}$ with $|X| = \text{VCD}(C_{ac}^k) + 1$. Such a set $X$ exists, since $X_{swap}$ is not shattered by $C_{ac}^k$. Moreover, let $\bar{x} = (x^1, \ldots, x^{|X|})$ be any fixed sequence of all and only the elements in $X$, without repetitions. Since $X$ is not shattered by $C_{ac}^k$, there is an assignment $(l_1, \ldots, l_{|X|}) \in \{0, 1\}^{|X|}$ of binary values to $\bar{x}$ that is not realized by $C_{ac}^k$, i.e., there is no concept $c \in C_{ac}^k$ such that $c(x^i) = l_i$ for all $i$. Since no concept in $C_{ac}^k$ realizes the assignment $(l_1, \ldots, l_{|X|})$ on $\bar{x}$, by
Lemma 11, no concept in $C^k_{ac}$ realizes the assignment $(1 - l_1, \ldots, 1 - l_{|X|})$ on $\bar{x}$. Thus, to shatter $X$, one would need to add at least two concepts to $C^k_{ac}$. ■

Second, the following proposition establishes that, under the same conditions as in Proposition 4, the class $C^k_{ac}$ is not extremal.

**Proposition 5.** Let $n \geq 1$, $m \geq 2$, and $0 \leq k \leq n - 1$, where $(m,k) \neq (2,0)$. Then the concept class $C^k_{ac}$ of all complete $k$-bounded acyclic CP-nets over $n$ variables of domain size $m$, over the instance space $X_{swap}$, is not extremal.

**Proof.** Let $X \subseteq X_{swap}$ be a set of instances that is shattered by $C^k_{ac}$, such that $|X| = \text{VCD}(C^k_{ac})$. Since $X_{swap}$ is not shattered by $C^k_{ac}$, we can fix some $\hat{x} \in X_{swap} \setminus X$. Moreover, let $\bar{x} = (x^1, \ldots, x^{|X|})$ be any fixed sequence of all and only the elements in $X$, without repetitions.

Suppose $C^k_{ac}$ were extremal. Then $X$ is strongly shattered, so that, in particular, there exists some $\ell \in \{0,1\}$ such that, for each choice of $(l_1, \ldots, l_{|X|}) \in \{0,1\}^{|X|}$, the labeling $(l_1, \ldots, l_{|X|}, \ell)$ of the instance vector $(x^1, \ldots, x^{|X|}, \hat{x})$ is realized by $C^k_{ac}$. Lemma 11 then implies that, for each choice of $(l_1, \ldots, l_{|X|}) \in \{0,1\}^{|X|}$, the labeling $(1 - l_1, \ldots, 1 - l_{|X|}, 1 - \ell)$ of the instance vector $(x^1, \ldots, x^{|X|}, \hat{x})$ is realized by $C^k_{ac}$. This is equivalent to saying that, for each choice of $(l_1, \ldots, l_{|X|}) \in \{0,1\}^{|X|}$, the labeling $(l_1, \ldots, l_{|X|}, 1 - \ell)$ of the instance vector $(x^1, \ldots, x^{|X|}, \hat{x})$ is realized by $C^k_{ac}$. To sum up, for each choice of $(l_1, \ldots, l_{|X|}) \in \{0,1\}^{|X|}$, both $(l_1, \ldots, l_{|X|}, \ell)$ and $(l_1, \ldots, l_{|X|}, 1 - \ell)$ as labelings of the instance vector $(x^1, \ldots, x^{|X|}, \hat{x})$ are realized by $C^k_{ac}$. This means that $X \cup \{\hat{x}\}$ is shattered by $C^k_{ac}$, in contradiction to $|X| = \text{VCD}(C^k_{ac})$. ■

As a last result of our study of structural properties of CP-net classes, we show that $C^k_{ac}$ is not intersection-closed, when $k \geq 1$ or when $m \geq 3$.

**Proposition 6.** Let $n \geq 1$, $m \geq 2$, and $0 \leq k \leq n - 1$, where $(m,k) \neq (2,0)$. Then the concept class $C^k_{ac}$ of all complete $k$-bounded acyclic CP-nets over $n$ variables of domain size $m$, over the instance space $X_{swap}$, is not intersection-closed.

**Proof.** Let $A$ and $B$ be two distinct variables. Let $a_1$ and $a_2$ be two distinct values in $D_A$, and $b_1$ and $b_2$ two distinct values in $D_B$. Further, let $\gamma$ be any fixed context over the remaining variables, i.e., those in $V \setminus \{A,B\}$. We will argue over the possible preferences over outcomes of the form $ab\gamma$, where $a \in \{a_1,a_2\}$ is an assignment to $A$ and $b \in \{b_1,b_2\}$ is an assignment to $B$.

Without loss of generality, suppose that $X_{swap}$ contains the instance $(a_1 b_1 \gamma, a_1 b_2 \gamma)$ (instead of $(a_1 b_2 \gamma, a_1 b_1 \gamma)$.) If that were not the case, one could rename variables and values accordingly to make $X_{swap}$ contain the instance $(a_1 b_1 \gamma, a_1 b_2 \gamma)$.

There are then various cases to consider for the swap pairs representing the comparisons between outcomes of the form $ab\gamma$, where $a \in \{a_1,a_2\}$ and $b \in \{b_1,b_2\}$. For each case, we provide two concepts $c_1, c_2 \in C^k_{ac}$ for which $c_1 \cap c_2$ is not acyclic and thus is not in $C^k_{ac}$ for any $k$. 

63
Table 7: Cases 1 and 7 in the proof of Proposition 6. Columns 1 through 4 provide binary labels stating which of the four swap instances considered are contained in a concept. Column 5 provides the statements in the corresponding CPT for $A$, while column 6 does the same for $B$. Concepts $c_1$ and $c_2$ belong to $C_{ac}^1$, but $c_1 \cap c_2$ has a cycle, in which $A$ is a parent of $B$ and vice versa.

Case 1. $\mathcal{X}_{\text{swap}}$ contains $(a_1 b_1 \gamma, a_2 b_1 \gamma)$, $(a_2 b_1 \gamma, a_2 b_2 \gamma)$, and $(a_1 b_2 \gamma, a_2 b_2 \gamma)$.

Case 2. $\mathcal{X}_{\text{swap}}$ contains $(a_1 b_1 \gamma, a_2 b_1 \gamma)$, $(a_2 b_2 \gamma, a_2 b_1 \gamma)$, and $(a_1 b_2 \gamma, a_2 b_2 \gamma)$.

Case 3. $\mathcal{X}_{\text{swap}}$ contains $(a_1 b_1 \gamma, a_2 b_1 \gamma)$, $(a_2 b_1 \gamma, a_2 b_2 \gamma)$, and $(a_2 b_2 \gamma, a_1 b_2 \gamma)$.

Case 4. $\mathcal{X}_{\text{swap}}$ contains $(a_1 b_1 \gamma, a_2 b_1 \gamma)$, $(a_2 b_2 \gamma, a_2 b_1 \gamma)$, and $(a_2 b_2 \gamma, a_1 b_2 \gamma)$.

Case 5. $\mathcal{X}_{\text{swap}}$ contains $(a_2 b_1 \gamma, a_1 b_1 \gamma)$, $(a_2 b_1 \gamma, a_2 b_2 \gamma)$, and $(a_1 b_2 \gamma, a_2 b_2 \gamma)$.

Case 6. $\mathcal{X}_{\text{swap}}$ contains $(a_2 b_1 \gamma, a_1 b_1 \gamma)$, $(a_2 b_2 \gamma, a_2 b_1 \gamma)$, and $(a_1 b_2 \gamma, a_2 b_2 \gamma)$.

Case 7. $\mathcal{X}_{\text{swap}}$ contains $(a_2 b_1 \gamma, a_1 b_1 \gamma)$, $(a_2 b_2 \gamma, a_1 b_1 \gamma)$, and $(a_2 b_2 \gamma, a_2 b_2 \gamma)$.

Case 8. $\mathcal{X}_{\text{swap}}$ contains $(a_2 b_1 \gamma, a_1 b_1 \gamma)$, $(a_2 b_2 \gamma, a_2 b_1 \gamma)$, and $(a_2 b_2 \gamma, a_1 b_2 \gamma)$.

Cases 1 and 7 are discussed in Table 7. A violation of the property of intersection-closedness in Cases 2, 3, and 4 can then be immediately deduced from Case 1 by inverting the binary values in the table for column 3, column 4, both columns 3 and 4, respectively. In the same way, Cases 5, 6, and 7 can be handled following Case 7.

To conclude, there are no known structure-related theorems in the literature that would imply $\text{VCD}(C_{ac}^{n-1}) = \text{RTD}(C_{ac}^{n-1})$. Hence, the latter equation, which we have proven in Section 5, is of interest, as it makes the class of complete unbounded acyclic CP-nets the first “natural” class known in the literature for which VCD and RTD coincide. A deeper study of its structural properties might lead to new insights into the relationship between VCD and RTD and might thus address open problems in the field of computational learning theory [34].