2017

Learning Conditional Preference Networks from Optimal Choices

Cory Siler
University of Kentucky, jcsi225@g.uky.edu
Digital Object Identifier: https://doi.org/10.13023/ETD.2017.470

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Cory Siler, Student

Dr. Judy Goldsmith, Major Professor

Dr. Miroslaw Truszczynski, Director of Graduate Studies

Learning Conditional Preference Networks from Optimal Choices

THESIS

A thesis submitted in partial
fulfillment of the requirements for
the degree of Master of Science in
the College of Engineering at the
University of Kentucky

By
Cory Siler
Lexington, Kentucky

Director: Dr. Judy Goldsmith, Professor of Computer Science
Lexington, Kentucky 2017

ABSTRACT OF THESIS

Learning Conditional Preference Networks from Optimal Choices

Conditional preference networks (CP-nets) model user preferences over objects described in terms of values assigned to discrete features, where the preference for one feature may depend on the values of other features. Most existing algorithms for learning CP-nets from the user's choices assume that the user chooses between pairs of objects. However, many real-world applications involve the the user choosing from all combinatorial possibilities or a very large subset. We introduce a CP-net learning algorithm for the latter type of choice, and study its properties formally and empirically.

KEYWORDS: artificial intelligence, preference learning

Author's signature: Cory Siler

Date: December 5, 2017

Learning Conditional Preference Networks from Optimal Choices

By
Cory Siler

Director of Thesis: Judy Goldsmith

Director of Graduate Studies: Miroslaw Truszczynski

Date: December 6, 2017

ACKNOWLEDGMENTS

I am grateful to the professors who served on my committee: Dr. Judy Goldsmith, my advisor, has invested a lot into my growth as a professional, and working with her has been delightful. Dr. Mirek Truszczynski has helped me secure many opportunities over the years, and it was his teaching that first made me appreciate the richness of computer science as an academic discipline. Dr. Brent Harrison joined the faculty during my last semester and I wish our stays at the university could have overlapped more, but he has inspired me a lot in the time I have known him.

This material is based upon work partially supported by the National Science Foundation under Grant No. IIS-1646887. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1 Introduction

Modern computer applications that deliver content to users — search engines, media and gaming platforms, social networking sites, and virtual assistants, to name a few — increasingly incorporate recommender systems that personalize the content to each individual user. Recommender systems tend to base their recommendations on one (or both) of two general sources of information: *Collaborative* recommendation predicts what content will be most relevant to the user by observing the selections of similar users, while *content-based* recommendation explicitly incorporates knowledge about properties of the content itself and predicts relevant content based on the user's apparent preferences over those properties [10].

The focus of this thesis, *conditional preference networks* (*CP-nets*) [8], is a formal preference representation for modeling the user in content-based recommender systems and other decision-making applications. While CP-nets have been widely studied academically, we are not aware of their use in commercial applications — yet we argue that there are some compelling reasons they should be considered.

One is the ease of preserving *privacy* in a CP-net-based recommender system: Whereas collaborative recommendation requires the service provider to collect potentially-sensitive data from many users, content-based recommendation can be done in isolation for each user, and the compactness of CP-nets in particular makes it feasible to store the user model locally on the user's own machine.

Another is *explainability*: While content-based recommendation systems often determine content's relevance to the user with sophisticated classifiers that obscure the reasons that the system chose particular content, CP-nets encode preferences with intuitive qualitative rules; inferences are made by generating formal proofs from these rules.

In the rest of this chapter, we review the CP-net formalism and existing methods for learning a CP-net from user choices. We also propose a set of learning assumptions that differs from what previous works have explored. In Chapter 2, we present an algorithm for learning a CP-net under these assumptions and prove its correctness. In Chapter 3, we test the algorithm on simulated user choices to study its effectiveness at predicting unseen preferences. In Chapter 4, we interpret the results of our experiments and discuss future directions for CP-net learning.

1.1 CP-nets

Consider a situation where we have a set of features,[1] each of which has a set of discrete values it can take; this gives rise to exponentially many outcomes,[2] each defined by a complete assignment of features to their values. For instance, a pizza might be defined in terms of the inclusion or exclusion of each available topping;

[1] Some works in the literature use different terminology: *attributes*, *variables*, etc.

[2] Also known as *alternatives*, *items*, etc.

for n topping choices, there are 2^n possible pizzas. Like many related works in the literature, we will make the simplifying assumption that the features are binary and treat them as Boolean variables,[3] though the methods we introduce are compatible with multivalued features.

To model a user's preferences over a potentially-huge set of outcomes, it is important for the sake of computational efficiency to have a compact representation. A popularly-studied formalism for this, and the focus of this work, is *conditional preference networks*, or *CP-nets* [8]. As the name suggests, CP-nets are based on the idea of conditionality of preferences: The user may have a general preference for one value of a feature over the other (e.g., "I always prefer pepperoni on my pizza"), or the preferred value of one feature may depend on the chosen values for others (e.g., "I prefer peppers on my pizza if and only if there is pepperoni").

A CP-net consists of a directed graph where the nodes correspond to features and an edge from one node to another indicates that the user's preference for the second feature depends on the value assigned to the first; we say that the first feature is a *parent* of the second. Each node is annotated with a *CP-table*, which contains a set of statements specifying what those feature-value preferences are. Let V be a feature with values $\{v, \neg v\}$, and let Par be V's (potentially empty) set of parent features; each CP-table statement takes the form $ParAssn : v \succ \neg v$ or $ParAssn : \neg v \succ v$, where $ParAssn$ is an assignment to the features in Par.

Figure 1.1 illustrates a CP-net with the above pizza scenario: The user's preferences for pepperoni are independent of the other features, so the node has no parents and a CP-table with a single unconditional preference statement; the preference for peppers depends on the presence or absence of pepperoni, so the node for peppers has the node for pepperoni as a parent, and its CP-table contains statements conditioned on the assignment to the pepperoni feature.

The maximum number of statements in a CP-table is equal to the number of possible assignments to the feature's parent set, which is exponential in the number of parents. To keep the size of the CP-net polynomial in the number of features, many works including ours assume a bound on maximum in-degree as a model parameter. We also allow for *incompleteness* in the CP-tables: A feature's CP-table is incomplete if not all possible parent assignments have a corresponding preference statement. This is illustrated with the mushrooms feature in Figure 1.1 — there is no known preference for mushrooms given $\neg pepperoni, peppers$.

The preference orderings encoded by CP-nets are defined according to *ceteris paribus semantics*, where *ceteris paribus* is Latin for "all else being the same"; a CP-table statement shows which of two outcomes is preferred when the values for one feature differ and the values for all others are identical. That is, let o and o' be outcomes where each feature is assigned to the same value in o and o', except $V = v$ in o and $V = \neg v$ in o' for some feature V. Let CP-net N contain the statement $ParAssn : v \succ \neg v$, where $ParAssn$ is an assignment to V's parent features such that the parent assignment is instantiated in o and o', i.e., $ParAssn \subset o$ and

[3]We will sometimes treat outcome assignments as sets with the assigned values as elements; a subset of an outcome denotes a partial assignment.

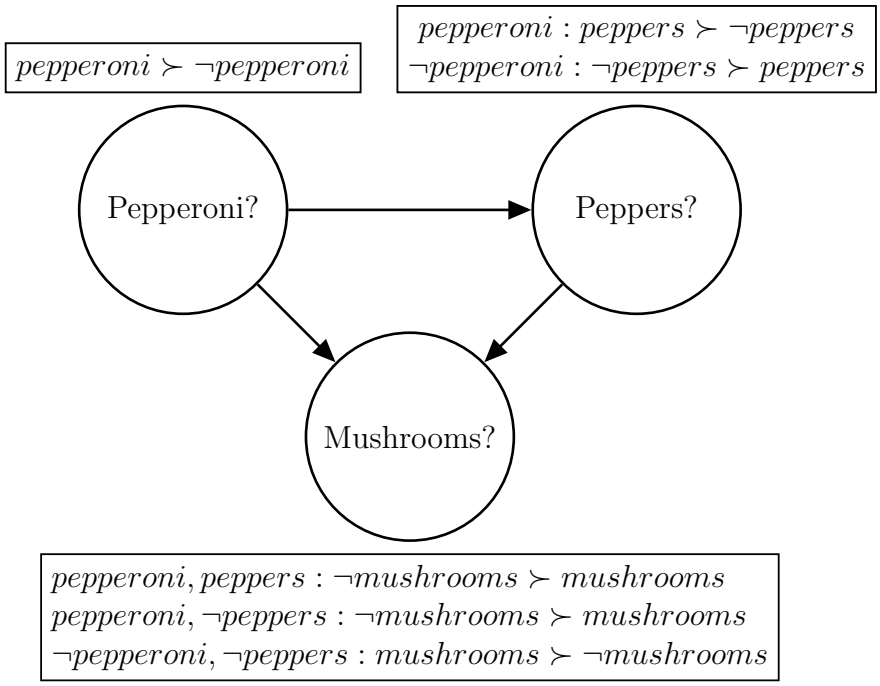

Figure 1.1: A CP-net

$ParAssn \subset o'$. Then we say that $o \succ o'$, i.e., o is preferred to or *dominates* o'.

Furthermore, the dominance relation is transitive; even if o differs from o' by several feature values, $o \succ o'$ if there exists a *series* of single-feature changes from less-preferred to more-preferred according to the CP-table statements — an *improving flipping sequence* — that transforms o' into o.

For instance, in the Figure 1.1 CP-net, we could ask: Is a pepperoni-only pizza preferred to a mushroom-only pizza? Starting with the mushroom-only outcome $\neg pepperoni, \neg peppers, mushrooms$, we can use the fact that $pepperoni$ is always preferred to $\neg pepperoni$ and flip the assignment to $\boldsymbol{pepperoni}, \neg peppers, mushrooms$. Now, the parent assignment for the mushroom feature has changed, and so has its preferred value; we can use the statement $pepperoni, \neg peppers : \neg mushrooms \succ mushrooms$ to flip the assignment to $pepperoni, \neg peppers, \boldsymbol{\neg mushrooms}$. This gives us an improving flipping sequence from the mushroom-only pizza to the pepperoni-only pizza:

$$\begin{aligned} \neg pepperoni, \neg peppers, mushrooms \prec \\ pepperoni, \neg peppers, mushrooms \prec \\ pepperoni, \neg peppers, \neg mushrooms. \end{aligned}$$

This proves that $pepperoni, \neg peppers, \neg mushrooms \succ pepperoni, \neg peppers, mushrooms$ according to the CP-net.

The dominance relation creates a strict partial order over the outcomes; for two outcomes o and o', the CP-net entails either $o \succ o'$, $o' \succ o$, or neither. In the last case, we say that o and o' are *incomparable*. (This may represent the fact that the user

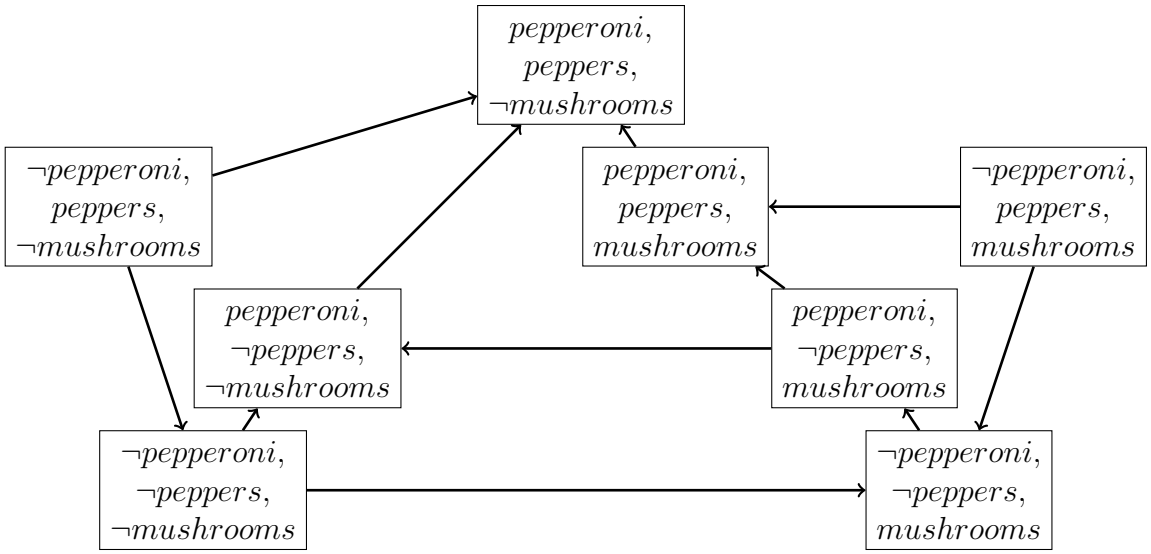

Figure 1.2: Induced preference graph of the CP-net from Figure 1.1

lacks a preference between o and o' or that the preference is unknown [1], or simply that the CP-net model is too restrictive to represent the user's complete preference ordering [21] — a necessary tradeoff of expressivity for compactness.) The dominance relation can be expressed in graph form with the outcomes as nodes, improving flips as directed edges, and the existence of a path between nodes indicating that the descendant outcome dominates the ancestor; this is called the CP-net's *induced preference graph*. Figure 1.2 illustrates an example, showing the dominance relation of the Figure 1.1 CP-net.

1.2 Learning from Optimal Choices

CP-nets were designed to be an intuitive model that allows direct elicitation — i.e., the user themself providing a CP-net to describe their preferences. However, recent studies suggest that nonexpert users often struggle to provide a CP-net that matches even their claimed preferences, let alone their actual choices [3, 15]; we expect that learning a CP-net, either by asking the user simple questions (active learning) or avoiding direct querying at all and inferring from the user's natural behavior (passive learning), will be a more reliable way of acquiring the model in a human-centric application. The algorithm we introduce in this work specifically learns passively, allowing minimal disruptiveness to the user while sacrificing the ability to control what information is received.

Most existing algorithms for passively learning CP-nets learn from sets of what we will refer to as *paired-choice examples*. Each of these consists of a single comparison $o \succ o'$; the implication is that the user has been observed to choose o over o' when presented with both, and that the learned CP-net should entail the preference.

However, it is common to encounter situations where users choose from an entire space of combinatorial possibilities rather than just a pair of outcomes. Revisiting

the pizza example from earlier — while a user at a friend's party may make choices between two types of pizza, a user ordering from a restaurant is free to select any combination of toppings that are currently in stock. Just as a system could treat a paired choice as evidence that the user prefers the chosen outcome to the non-chosen one, it should be able to treat this second type of choice as evidence that the chosen outcome is the user's most-preferred available option.

This work focuses on learning from sets of this latter type of example, which we will refer to as an *optimal-choice example*. Formally, we define an optimal-choice example as a pair $(cond, opt)$, where the condition $cond$ is an assignment to a (possibly-empty) subset of the feature set and the optimum opt is an outcome such that $cond \subseteq opt$. This is interpreted as meaning that opt is the most-preferred outcome among those containing the assignments in $cond$ (i.e., opt is not dominated by any other outcome containing $cond$); when this holds in a CP-net, we will say that the CP-net is *consistent with* the example.

For instance, since the pepperoni-and-pepper pizza is the most-preferred in Figure 1.1, the CP-net is consistent with the optimal-choice example having $cond = \emptyset$ and $opt = pepperoni, peppers, \neg mushrooms$. However, if the restaurant is out of pepperoni, a mushroom pizza becomes the best available option, hence the CP-net is also consistent with $cond = \neg pepperoni, opt = \neg pepperoni, \neg peppers, mushrooms$.

While we are the first (to our knowledge) to discuss *learning* from this type of information, the inverse operation — starting with a CP-net and determining an optimal outcome given some set of preassigmnents — is discussed in a paper by Boutilier et al. [8]; they give a linear-time algorithm for finding the optimum. (Their later work [9] explores outcome optimization allowing for a more general class of constraints on the feature assignments.)

In theory, assuming there is only one optimal outcome per condition,[4] we could expand an optimal-choice example $(cond, opt)$ into a set of paired-choice examples, $opt \succ o$ for every outcome o where $cond \subset o$ and $o \neq opt$. For instance, from the optimal-choice example with $cond = \neg pepperoni, opt = \neg pepperoni, \neg peppers, mushrooms$, we could derive all of the following:

- $\neg pepperoni, \neg peppers, mushrooms \succ \neg pepperoni, peppers, mushrooms$,
- $\neg pepperoni, \neg peppers, mushrooms \succ \neg pepperoni, \neg peppers, \neg mushrooms$, and
- $\neg pepperoni, \neg peppers, mushrooms \succ \neg pepperoni, peppers, \neg mushrooms$.

However, the number of paired-choice examples we would need to derive to be equivalent to an optimal-choice example is exponential in the number of nonconditioned features, so learning from optimal-choice examples by first converting them into paired-choice examples would be intractable (or require sampling, and thus loss of information); to make practical use of the full information provided by optimal-choice

[4]This is the case with acyclic CP-nets, which we focus on in this work. Cyclic CP-nets may have multiple optima for a given condition, or even none at all. If the latter happens, it is because the CP-net's preference ordering is *inconsistent*, having cycles in the dominance relation and thus outcomes that dominate themselves by transitivity.

examples, we need learning algorithms designed to work with them directly. We will present such an algorithm in Chapter 2.

1.3 Related Work

We survey several existing works related to CP-net learning here, starting with passive learning (i.e., noninteractive learning that relies on preexisting data).

Lang and Mengin [21] investigate the passive learning of *separable* CP-nets, i.e., CP-nets with no edges, representing preference relations where the preferences for all features are unconditional. They consider decision problems asking whether such a CP-net exists (i.e., determining representability of the user's preferences by a separable CP-net) given a set of paired-choice examples. They show that deciding the existence of a separable CP-net that entails every paired-choice example is in P. They also discuss weaker notions of learning that increase the complexity: If, for each example $o \succ o'$ the CP-net is allowed to merely entail $o' \not\succ o$ rather than $o \succ o'$ itself, deciding the existence of the CP-net is NP-complete.

Dimopoulos et al. [13] give a passive learning algorithm for paired-choice examples that is polynomial-time given a fixed maximum in-degree. Successful learning is guaranteed only for example sets that are entailed by CP-nets with a specific type of entailment (*transparent entailment*). The authors prove PAC-learning guarantees about how closely the learned CP-net is likely to replicate the user's true preference ordering, assuming the true preferences are also CP-net-structured and the example distribution meets certain assumptions. Chevaleyre et al. [11] also study the PAC-learning complexity of CP-nets.

Liu et al. [22] consider "noisy" paired-choice examples: An observation $o \succ o'$ has some probability of being the user's true preference, and otherwise it is incorrect and the user actually has the preference $o' \succ o$. Their learning approach uses statistical hypothesis testing to determine whether a dependency holds between a feature and a candidate parent set (the null hypothesis being that the members of the set are not the feature's parents, and the alternative hypothesis being that they are). In another work [23], they consider sets of possibly-inconsistent paired-choice examples with associated weights; their goal is to maximize the total weight of comparisons entailed by the CP-net. They present an algorithm that uses branch-and-bound search to find a consistent preference graph (dominance relation) over the outcome from the examples such that the total weight of examples entailed by the relation is maximized; it then constructs a CP-net whose induced preference graph has the computed preference graph as a subgraph.

Allen et al. [5] explore the use of local search to optimize a CP-net's adherence to a set of noisy/inconsistent paired-choice examples (focusing on tree-structured CP-nets, for which dominance-based optimization metrics can be evaluated in polynomial time). Another metaheuristic approach to CP-net learning is that of Haqqani and Li [18], who use evolutionary algorithms. Michael and Papageorgiou [24] also propose optimization-based learning, namely, modifying the algorithm of Dimopoulos et al. [13] to formulate CP-table selection as a maximum satisfiability problem.

We now turn our attention to *active* learning of CP-nets (i.e., building CP-nets through interactive user querying). Guerin et al. [17] give active CP-net learning algorithm that runs in two phases: The first initializes a separable CP-net by asking the user the "default" preferences for each feature, and the second asks paired-choice queries to determine parent sets. Koriche and Zanuttini [19] and Labernia et al. [20] each present active learning algorithms based on "equivalence queries". An equivalence query asks whether the current learned CP-net is equivalent to the "true" CP-net. If not, it gets a paired-choice counterexample — specifically, a "swap" where the compared outcomes differ on only one feature — that is missing or violated in the current CP-net, which must be repaired.

Related to the notion of learning a CP-net from scratch is that of completing a partial CP-net. In recommender systems research, this has been explored in terms of collaborative filtering, i.e., guessing the rest of a user's CP-net by the preferences of similar users. Wang et al. [25] consider incompleteness in CP-tables, assuming a shared graph structure between agents' CP-nets. Their approach computes similarity between agents' CP-nets using a measure of distance between induced preference graphs; the shared structure of the CP-nets allows this measure to be computed without enumerating the induced preference graphs themselves. Missing CP-table statements are filled in using a "vote" among similar CP-nets. The El Fidha and Amor [14] use an alternative similarity measure based directly on the CP-nets rather than the induced preference graphs, allowing comparison between CP-nets with differing structure; they iterate through sufficiently similar (as determined by clustering) users' CP-nets, copying over child nodes as well as CP-table statements when able.

Note that in all but the last works, the learning relies on choices between pairs of outcomes. To our knowledge, the only prior work based on learning from optimal choices is that of Bigot et al. [7], whose algorithms take as input a set of unconditional optimal-choice examples with probabilities that they are optimal. The algorithms infer a PCP-net, i.e., *probabilistic CP-net* [6, 12], a model that encodes a probability distribution over CP-nets by associating with each CP-table statement a probability that the statement holds.

Chapter 2 Algorithm

In this chapter, we introduce the focal algorithm of this thesis. Given a set of optimal-choice examples, it learns a CP-net consistent with all examples — i.e., for each example $(cond, opt)$, the outcome opt will be optimal given the condition $cond$ in the learned CP-net. Successful learning of a CP-net is subject to a few restrictions: A CP-net consistent with all examples must exist, it must be acyclic, and no node's in-degree may exceed a prespecified bound. In this chapter, we will first outline the algorithm itself. Then we will prove the its completeness (the algorithm always returns a CP-net if able), soundness (the returned CP-net is consistent with the examples), and efficiency (the runtime is polynomial in the number of features and examples given a fixed maximum in-degree) in Section 2.2.

2.1 Overview

The main routine of the algorithm, shown in Figure 2.1, is very similar to Dimopoulos et al.'s algorithm [13] for learning from paired examples. The algorithm iteratively builds a CP-net by attempting to add a feature (with its associated CP-table) at a time, choosing its parent set from among the features that have already been added to the CP-net. (Restricting candidate parents to the features already in the CP-net forces the graph to be acyclic; choosing parents from among all features would permit cycles but also potentially result in models whose underlying preference orderings are inconsistent — i.e., having an outcome that dominates itself by transitivity.) Candidate parent sets are considered in increasing order of size (starting with the empty set), which prevents superfluous parents (i.e., those that do not actually affect the preferred value of the feature) from being introduced.[1] If a CP-table cannot be added with the available parent sets, the feature can be revisited later, after more possible parents are introduced. The algorithm terminates when all features have been added, completing the learning process. It will also terminate if none of the remaining features can be added, indicating that a CP-net could not be learned from the example set.

The key difference between this algorithm and Dimopoulos et al.'s, owing to the fact that they process different types of examples, is how the CP-tables are constructed. The subroutine our algorithm uses to try to create a given feature's CP-table is outlined in Figure 2.2. For each assignment $ParAssn$ to the prospective parents of a feature V with values $\{v, \neg v\}$, we must choose at most one of the statements $ParAssn : v \succ \neg v$ or $ParAssn : \neg v \succ v$ to insert into the CP-table. If the first statement holds, it must be the case that any outcomes containing $ParAssn$ must also assign $V = v$ in order to be optimal unless the condition forces $V = \neg v$; similarly, if the second holds, optima with $ParAssn$ must assign $V = \neg v$ if able. For

[1] Allen et al. [2] refer to CP-nets containing superfluous parents as *degenerate*, as the CP-tables map to degenerate Boolean functions when the features are binary.

```
Input: binary feature set AllFeatures; in-degree bound k; set Examples of
       examples of the form (cond, opt)
Output: if possible, an acyclic CP-net with maximum in-degree ≤ k for which
        each example's opt is optimal given its cond
cpnet ← an empty CP-net
RemainingFeatures ← AllFeatures
AddedFeatures ← ∅
repeat
    forall feat ∈ RemainingFeatures do
        for i ← 0 to k do
            forall size-i subsets CandidateParents ⊆ AddedFeatures do
                CPT ← CreateCPT(feat, CandidateParents, Examples)
                if CPT ≠ null then
                    add feat to cpnet with CandidateParents as parents and CPT
                     as the CP-table
                    RemainingFeatures ← RemainingFeatures \ {feat}
                    AddedFeatures ← AddedFeatures ∪ {feat}
                end
            end
        end
    end
until no new features are added to cpnet
if RemainingFeatures = ∅ then
    return cpnet
else
    return null
end
```

Figure 2.1: Passive learning algorithm from optimal examples

each parent assignment, the algorithm checks whether the examples agree on a single preferred value of the feature given the assignment. If they do, the corresponding statement is added to the CP-table; if the examples conflict, the candidate parent set cannot be the feature's true parent set; and if there are no relevant examples (where the parent assignment appears and the feature value is unrestricted), no statement is added but the CP-table construction continues.

The last possibility results in the algorithm allowing *incompleteness* in the CP-tables; some parent assignments may not have corresponding preferred values. This provides some flexibility in an online environment for adding to the CP-net later if new examples come in, but if a complete CP-net is required immediately (e.g., for guarantees about the computational complexity of dominance testing [8]), the missing entries can be filled in arbitrarily (e.g., randomly, or according to some heuristic).

```
Input: feature feat; candidate parent set CandidateParents; example set
       Examples
Output: if possible, returns a CP-table such that feat has its preferred value
        in all examples where it is not preassigned
CPT ← an empty CP-table
forall examples (cond, opt) ∈ Examples for which cond does not assign a value
 to feat do
    Val ← the value of feat in opt
    Assn ← the assignment to CandidateParents in opt
    if CPT does not already contain a statement conditioned on Assn then
        insert the statement Assn : ChosenVal ≻ ¬ChosenVal into CPT
    else if CPT contains the statement Assn : ¬ChosenVal ≻ ChosenVal then
        return null
    end
end
return CPT
```

Figure 2.2: `CreateCPT` subroutine

2.2 Properties

We will start by proving our algorithm's completeness. Note that unlike the paired-choice-example version by Dimopoulos et al. [13], which is only complete for special cases, our optimal-choice-example version is complete for all cases where a CP-net fitting the examples and the model constraints exists.

Theorem 1. *The learning algorithm in Figure 2.1 always returns an acyclic CP-net obeying the given in-degree bound if there exists one consistent with the examples.*

Proof. We will prove this by contradiction. Suppose the example set has at least one CP-net N_{true} fitting the requirements but the algorithm returns null. Let $N_{learned}$ be the CP-net constructed so far by the algorithm when it terminates, let *Added* be the set of features that have been added to $N_{learned}$, and let *Remaining* be the set of features not in *Added*. Because the algorithm returns null, no features in *Remaining* could be added to the $N_{learned}$ with a parent set chosen from *Added*; the only valid parent sets for each feature in *Remaining* would require a parent from *Remaining*. This means that any group of parent sets for all elements of *Remaining* would produce cycles. But this contradicts the fact that since N_{true} is acyclic, it contains a valid group of parent sets for the elements of *Remaining* with no cycles among them. Therefore, it cannot be the case that a CP-net fulfilling the requirements exists but none is returned. □

Now we will prove soundness. This has two components. The first is that the example outcomes will be optimal (undominated) given their conditions. However, learning a CP-net satisfying only this component would be trivial; we could simply

return a CP-net with empty CP-tables, making all outcomes incomparable and therefore undominated. The second component ensures that our learned CP-net is more informative: *No other* outcome will be optimal given an example's condition.

Theorem 2. *If the algorithm in Figure 2.1 returns a CP-net, then the CP-net will be consistent with all examples. (That is, for every example* $(cond, opt)$ *in the example set, opt will be the only optimal outcome given cond — that is, for every outcome o such that* $o \neq opt$ *and* $cond \subset o$*, the CP-net will entail* $opt \succ o$*.)*

Proof. Let $N_{learned}$ be the learned CP-net. First, we will prove that $N_{learned}$ must be consistent with all examples.

Suppose an *opt* is not optimal given its *cond* in $N_{learned}$. This means that Boutilier et al.'s "forward sweep" optimization procedure [8] — visiting all non-*cond* features' nodes in a topological order and assigning them to a most-preferred value given their parents' assignments — could not yield *opt* when performed on $N_{learned}$. This implies that there is a feature in $opt \setminus cond$ that is assigned to its less-preferred value given its parents' assignments. Let V be the feature, let v be its assigned value in *opt*, let P be the parent set of V in $N_{learned}$, and let $ParAssn \subset opt$ be the assignment to P. There must be an entry $ParAssn : \neg v \succ v$ in the feature's CP-table, constructed by `CreateCPT` in the algorithm. However, the example $(cond, opt)$ would have caused the algorithm to try to insert the opposite, $ParAssn : v \succ \neg v$, and trying to insert both would have caused `CreateCPT` to fail, contradicting the fact that P is V's parent set. Therefore, each *opt* must be optimal given its *cond* in $N_{learned}$.

Next, we will prove that no other outcomes are optimal in $N_{learned}$ given the same conditions as the examples.

If multiple outcomes are optimal given a condition, they must by definition of optimality be incomparable to each other (rather than one dominating another). Because $N_{learned}$ is acyclic, incomparability can happen for a pair of optimal outcomes only when they differ on feature for which the relevant preference statement is missing. Suppose we have an example $(cond, opt)$ consistent with $N_{learned}$ and an incomparable outcome opt' that is also optimal given *cond*. Let V be the first feature in a topological sorting of the CP-net nodes for which the outcomes differ; the outcomes have the same assignment $ParAssn$ to the parent set P of V. But when `CreateCPT` created V's CP-table, one of the following must have been the case:

- `CreateCPT` set the value in *opt* as the preferred value given $ParAssn$. But this would contradict the assumption that there is no preference statement for $ParAssn$.
- `CreateCPT` saw another example with the opposite assignment and concluded that P was not in fact V's parent set. But this would contradict the assumption that P is indeed the parent set.

Therefore, each *opt* in the example set must be the sole optimum in $N_{learned}$ given its *cond*. □

Lastly, we will analyze the algorithm's computational complexity. Its tractability relies on a small bound on the learned CP-net's maximum in-degree. (Such a bound is a common assumption in works dealing with CP-nets, since unlimited in-degree would allow for exponential-sized CP-tables.)

Theorem 3. *Let the in-degree bound k be a fixed constant; then the algorithm in Figure 2.1 runs in time polynomial in the number of examples and features.*

Proof. Let n be the number of features, and let m be the number of examples. The outer loop of the algorithm iterates through the set of up to n remaining feature no more than n times, since it reduces the set size each time. For each of these remaining features, the algorithm considers a $\sum_{i=0}^{k} \binom{n}{i} = O(n^k)$ number of candidate parent sets, making a `CreateCPT` call for each; `CreateCPT` is called $O(n^{k+2})$ times in a run of the algorithm. `CreateCPT` iterates through each of the m examples with n features, each time operating on a subset of those features, for $O(m * n)$ per run. This gives the algorithm an overall $O(m * n^{k+3})$ runtime. □

Chapter 3 Experiments

We have demonstrated an algorithm that reliably finds a CP-net in which all provided outcomes are optimal given their conditions, if one exists. However, in learning a CP-net we are not merely storing the optima but rather are building a whole preference model for the user; we are interested in the model's ability not only to reflect known information about the user's preferences but also to predict *unknown* information, assuming that the user's underlying preferences have a CP-net structure.[1]

In particular, we are interested in studying the relationship between the optimal-examples learning paradigm and the traditional paired-examples learning paradigm; we will learn CP-nets from sets of the former and measure how well they can predict the latter.

3.1 Setup

To create example sets on which to test our learning algorithm, we used Allen et al.'s approach [4] to generate 100 random binary-valued acyclic CP-nets over 10 features with in-degree at most 2. These CP-nets represent the true preferences of simulated "users" that we hope to model as closely as possible. Next, we simulated series of "choices" by the users that the system might encounter: For each CP-net, we generated optimal-choice example sets by sampling (with replacement, allowing redundant examples) from each of two different example distributions.

For the first distribution, we chose the examples uniformly at random by sampling conditions from the set of all 3^{10} possible conditions — that is, for each of the 10 features, we randomly chose between assigning it to true in the condition, assigning it to false from the condition, or excluding it from the condition. We then found the CP-net's optimal outcome under the condition (the acyclicity of the CP-net guarantees that there is only one optimal assignment for a given condition).

Since there are $\binom{10}{k} * 2^k$ possible examples conditioned on exactly k features, uniformly random example generation favored somewhat strict conditions; for instance, there were 13440 possible examples conditioned on 7 features (about 26% of all examples), compared to only one example not conditioned on any feature (less than .002% of all examples). See Figure 3.1 for a visualization of the proportion of examples per number of features in the condition.

For the second distribution, we were interested in the performance of the learning algorithm in scenarios where the observed choices tend to have fewer constraints (e.g., the choice of pizzas from a menu where toppings are only subject to sporadic unavailability). As such, our generation method was biased toward less-strict conditions: For each feature, we preassigned it in a given example's condition with only

[1] This is a common assumption in CP-net learning experiments, although the question of how well CP-nets can model real human preferences has only recently started to be examined empirically [3, 15].

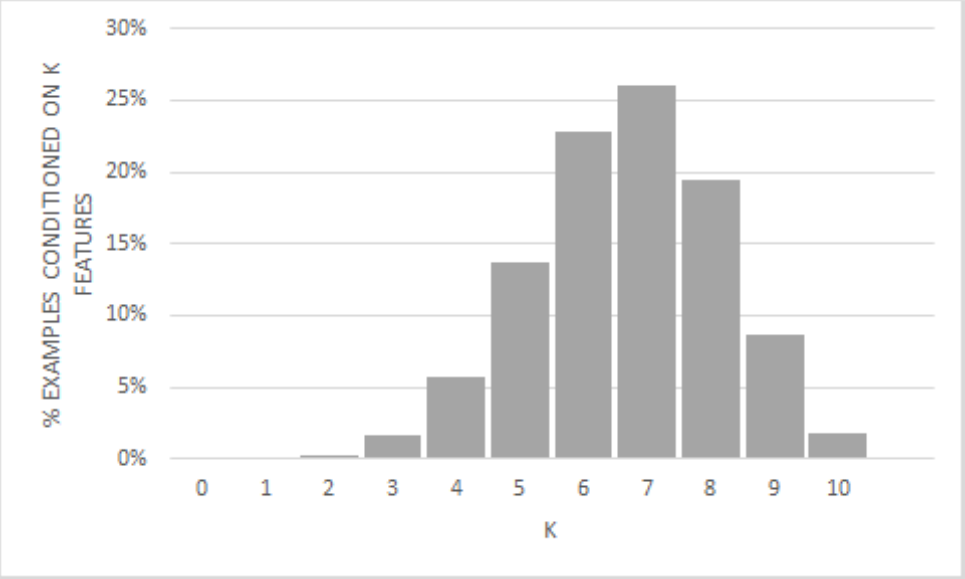

Figure 3.1: Probability of generating an optimal-choice example with k features preset in the condition, uniform distribution

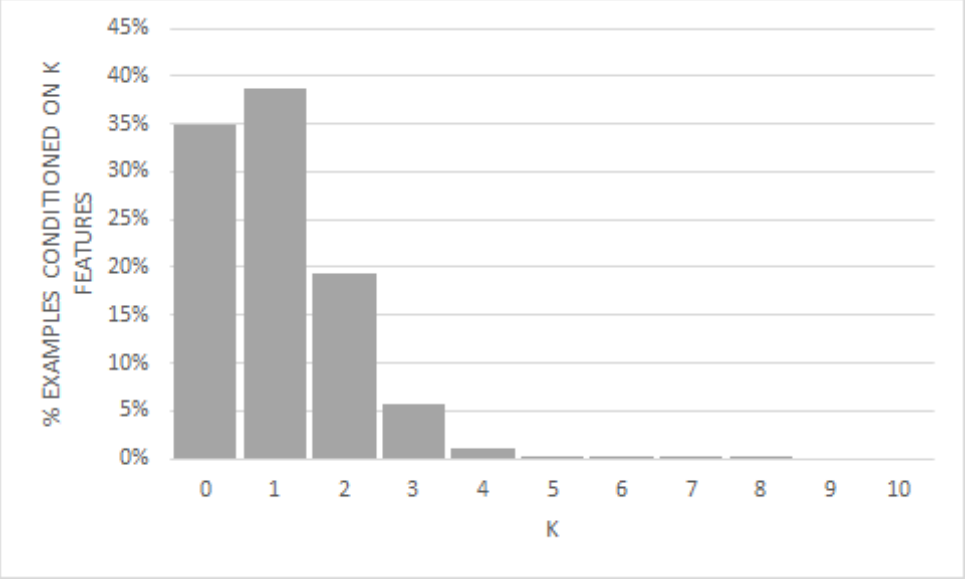

Figure 3.2: Probability of generating an optimal-choice example with k features preset in the condition, biased distribution

10% probability. Figure 3.2 shows the probability of generating an example for each feature count in the condition.

We generated an example set once per example set size (5, 10, 20, 40, and 80), per example distribution, per generated CP-net. After generating an example set, we then ran the learning algorithm from Figure 2.1 to learn an acyclic, in-degree-bounded-to-2, possibly-incomplete CP-net consistent with all examples.

3.2 Results

To measure the quality of the learned CP-net in terms of reproducing the preference ordering of the original CP-net used for example generation, we computed all the comparisons $o \succ o'$ that made up the two CP-nets' preference orderings by traversing the induced preference graph (an intractable task with large numbers of features, but feasible with only 10). As one metric, we computed the proportion of the original CP-net's comparisons that were entailed by the learned CP-net. That is, let P_{true} be the set of comparisons entailed by the simulated user's true CP-net, and let $P_{learned}$ be the CP-net learned from the examples; we used the metric:

$$\frac{|P_{learned} \cap P_{true}|}{|P_{true}|}. \tag{3.1}$$

Additionally, as a metric that incorporates lack of irrelevant entailments along with inclusion of relevant entailments, we computed the Jaccard similarity of the learned and true CP-nets (a measure previously used to compare CP-nets by Wang et al. [25]):

$$\frac{|P_{learned} \cap P_{true}|}{|P_{learned} \cup P_{true}|}. \tag{3.2}$$

In addition to the incomplete learned CP-net, we also derived a complete learned CP-net by randomly filling in the missing (i.e., no preferred value for some parent assignment) statements of the incomplete CP-net's CP-tables. We computed the aforementioned quality metrics for the completed CP-net with respect to the simulated user's true CP-net.

Finally, as a baseline for performance, for each simulated user's CP-net we generated a second random CP-net; we computed the quality metrics for the random CP-net.

Figures 3.3 and 3.4 show the average proportion of entailments learned using the uniformly random and biased-toward-fewer-conditions example sets respectively. Likewise, Figures 3.5 and 3.6 show the average similarity scores for the CP-nets learned from those example sets.

Performance on small example sets was poor in our experiments, with the learning algorithm averaging worse-than-random CP-net quality on the smallest sets (except by the similarity metric with the biased examples). Randomly completing the learned CP-nets improved the performance only marginally.

As the number of examples increased, we saw the learning process converge toward the true CP-net faster with the uniformly random examples than with the biased examples. We can explain this trend by the fact that the biased examples, since they tend to have fewer conditions, are more informative individually — a general example entails several more-specific examples — but as we add more examples, the uniformly random ones continue to be informative while the biased ones are often redundant (either by direct duplication, or by entailment of a more-specific example by a more-general one).

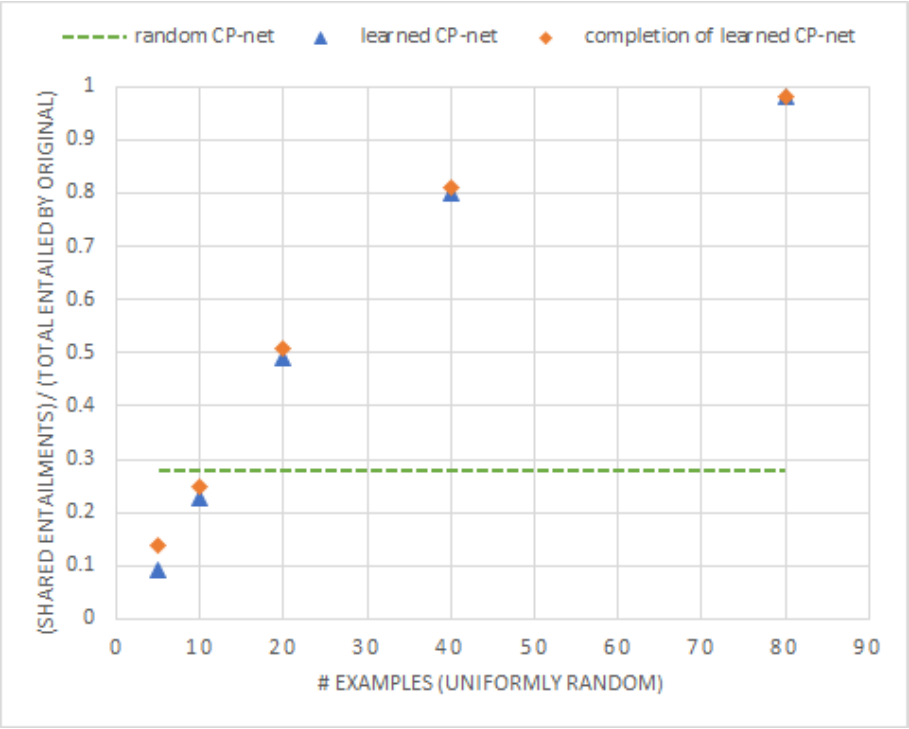

Figure 3.3: Average performance on uniformly random examples, Equation 3.1 metric

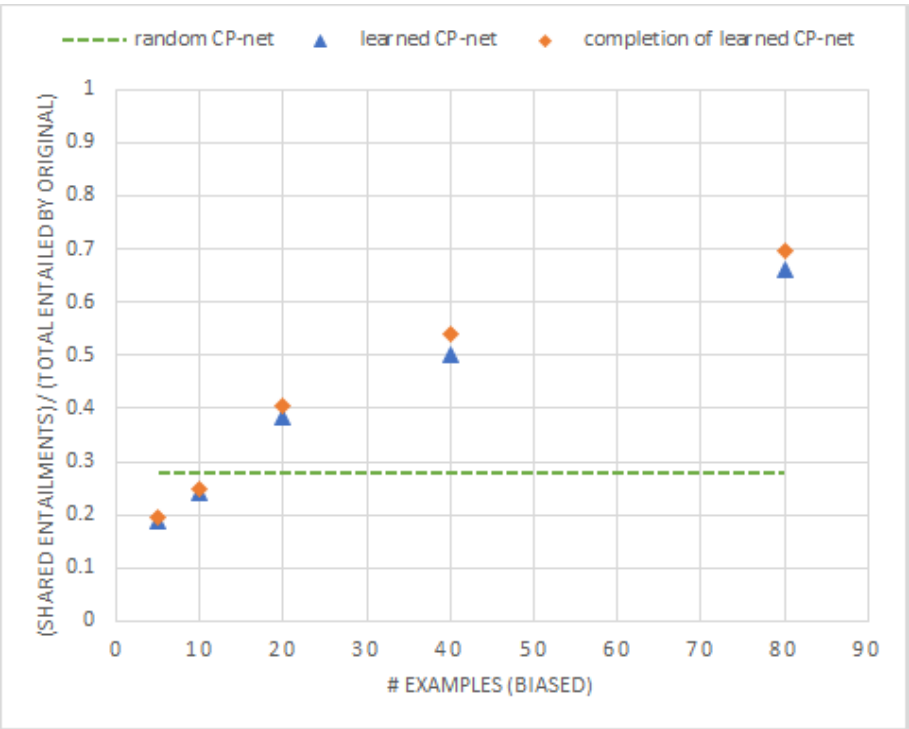

Figure 3.4: Average performance on biased example distribution, Equation 3.1 metric

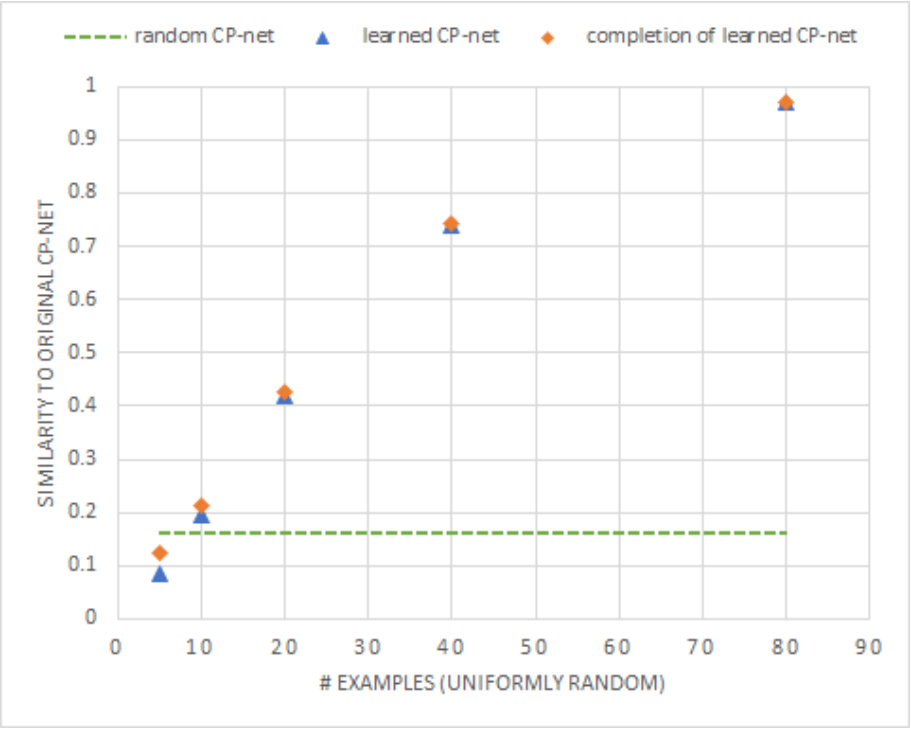

Figure 3.5: Average performance on uniformly random examples, Equation 3.2 metric

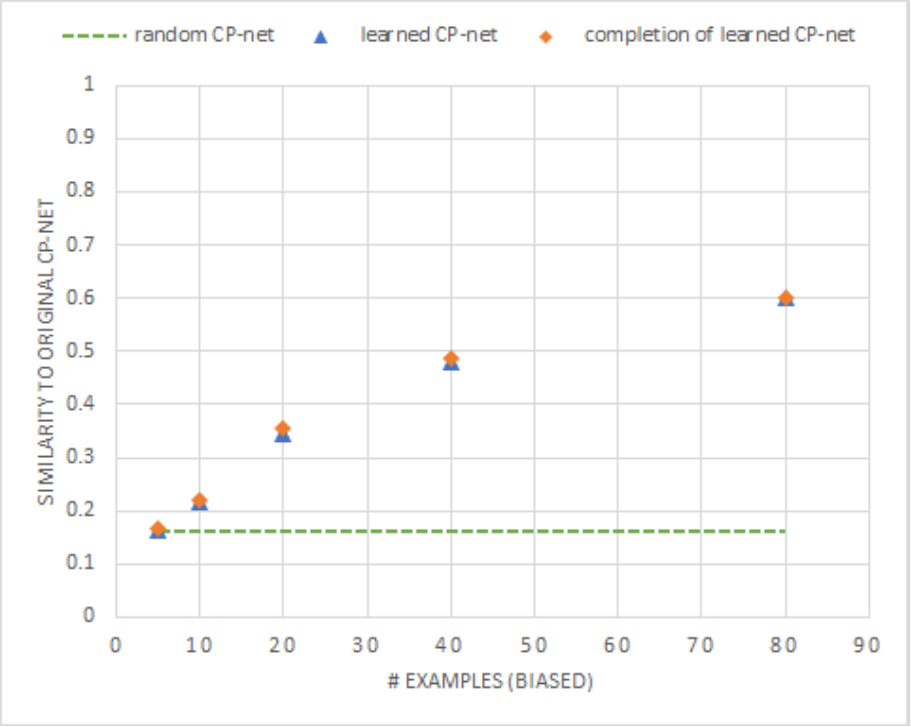

Figure 3.6: Average performance on biased example distribution, Equation 3.2 metric

Chapter 4 Conclusions

In this thesis, we have introduced an algorithm that builds a CP-net fitting an observed set of user choices, where the outcomes were selected from a combinatorial set of options. Because of the large amount of user customization allowed in many applications where we might want to model user preferences, we argue that this type of example is common and should be a greater focus of CP-net learning algorithms. Our experiments test how well the algorithm's constructed models generalize to unseen preferences; as we elaborate below, these experiments shed light on future research directions that could help make learning from optimal-choice examples robust enough for real-world applications.

Our experimental results have negative implications for the plausibility of learning CP-nets from user choices in scenarios like our motivating examples — where we have a fairly sparse set of examples chosen from a very open set of options. On the other hand, our algorithm did come close to learning an ideal CP-net when provided with a larger set of more-constrained examples. However, in large example sets taken from real human users, we are likely to encounter inconsistencies (e.g., due to the user opting for variety and selecting something contrary to their usual preferences) that would cause our exact-learning algorithm to fail.

Thus, to be useful in realistic applications, an algorithm for learning from optimal-choice example would need to be robust to noisy and inconsistent example sets. The other existing work in learning from optima, that of Bigot et al. [7], handles this by learning a probability distribution over CP-nets from an example set that is also treated as a probability distribution. However, their algorithm assumes only unconditional optima, not making use of the additional information that may be available when the some of the feature values are predetermined. Learning algorithms from *conditionally* optimal choices could make use of techniques that have already been suggested for learning from paired choices. For instance, heuristic search has been proposed for learning from paired-choice examples [5], but its general application is hindered by the fact that dominance testing (and hence evaluating a CP-net with respect to those examples) is computationally hard[1] [8, 16]; it would be even better-suited to learning from optimal-choice examples, as checking consistency of an optimal-choice examples with an acyclic CP-net can be done in linear time [8].

Additionally, if we expect to encounter almost exclusively choices conditioned on few or no features, we should design algorithms that model the overall preference ordering more accurately by acquiring additional information. This suggests the use of active learning to supplement the passive learning described here — i.e., the system asks the user about optima under specific conditions chosen to most effectively fill in knowledge gaps about the user's preferences. Active learning can also be used to

[1]The computational complexity depends on assumptions such as the graph structure of the CP-net and whether the features can be multivalued. The precise complexity of dominance testing for binary-valued, complete, acyclic CP-nets is an open question; Allen et al. [4] conjecture based on empirical evidence that the problem is in NP.

deal with apparent inconsistencies: When the system observes the user make two different choices under the same apparent conditions, it could query the user about which choice should be treated as the "default" and under what circumstances the other choice is preferable.

Bibliography

[1] Thomas E Allen. CP-nets with indifference. In *Allerton Conference on Communication, Control, and Computing*, pages 1488–1495. IEEE, 2013.

[2] Thomas E Allen, Judy Goldsmith, and Nicholas Mattei. Counting, ranking, and randomly generating CP-nets. In *Multidisciplinary Workshop on Advances in Preference Handling: Papers from the AAAI-14 Workshop*, 2014.

[3] Thomas E Allen, Muye Chen, Judy Goldsmith, Nicholas Mattei, Anna Popova, Michel Regenwetter, Francesca Rossi, and Christopher Zwilling. Beyond theory and data in preference modeling: Bringing humans into the loop. In *International Conference on Algorithmic Decision Theory*, pages 3–18. Springer, 2015.

[4] Thomas E Allen, Judy Goldsmith, Hayden Elizabeth Justice, Nicholas Mattei, and Kayla Raines. Uniform random generation and dominance testing for CP-nets. *Journal of Artificial Intelligence Research*, 59:771–813, 2017.

[5] Thomas E Allen, Cory Siler, and Judy Goldsmith. Learning tree-structured CP-nets with local search. In *International Florida Artificial Intelligence Research Society Conference*, pages 8–13. AAAI Press, 2017.

[6] Damien Bigot, Hélène Fargier, Jérôme Mengin, and Bruno Zanuttini. Probabilistic conditional preference networks. In *Conference on Uncertainty in Artificial Intelligence*, 2013.

[7] Damien Bigot, Jérôme Mengin, and Bruno Zanuttini. Learning probabilistic CP-nets from observations of optimal items. In *European Starting AI Researcher Symposium*, volume 264, pages 81–90, 2014.

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

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

[10] Robin Burke, Alexander Felfernig, and Mehmet H Göker. Recommender systems: An overview. *AI Magazine*, 32(3):13–18, 2011.

[11] Yann Chevaleyre, Frédéric Koriche, Jérôme Lang, Jérôme Mengin, and Bruno Zanuttini. Learning ordinal preferences on multiattribute domains: The case of CP-nets. In *Preference Learning*, pages 273–296. Springer, 2010.

[12] Cristina Cornelio, Judy Goldsmith, Nicholas Mattei, Francesca Rossi, and K Brent Venable. Updates and uncertainty in CP-nets. In *Australasian Joint Conference on Artificial Intelligence*, pages 301–312. Springer, 2013.

[13] Yannis Dimopoulos, Loizos Michael, and Fani Athienitou. Ceteris paribus preference elicitation with predictive guarantees. In *International Joint Conference on Artificial Intelligence*, pages 1890–1895. Morgan Kaufmann Publishers Inc., 2009.

[14] Sleh El Fidha and Nahla Ben Amor. Multi-agent conditional preference networks with incomplete and different structure. In *International Conference of Soft Computing and Pattern Recognition*, pages 197–202. IEEE, 2014.

[15] John Fike. A study of human introspection about preferences. Unpublished master's project, University of Kentucky, 2017.

[16] Judy Goldsmith, Jérôme Lang, Miroslaw Truszczynski, and Nic Wilson. The computational complexity of dominance and consistency in CP-nets. *Journal of Artificial Intelligence Research*, 33:403–432, 2008.

[17] Joshua T Guerin, Thomas E Allen, and Judy Goldsmith. Learning CP-net preferences online from user queries. In *International Conference on Algorithmic Decision Theory*, pages 208–220. Springer, 2013.

[18] Mohammad Haqqani and Xiaodong Li. An evolutionary approach for learning conditional preference networks from inconsistent examples. In *International Conference on Advanced Data Mining and Applications*, pages 502–515. Springer, 2017.

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

[20] Fabien Labernia, Florian Yger, Brice Mayag, and Jamal Atif. Query-based learning of acyclic conditional preference networks from contradictory preferences. *EURO Journal on Decision Processes*, 2017.

[21] Jérôme Lang and Jérôme Mengin. The complexity of learning separable ceteris paribus preferences. In *International Joint Conference on Artificial Intelligence*, pages 848–853, 2009.

[22] Juntao Liu, Zhijun Yao, Yi Xiong, Wenyu Liu, and Caihua Wu. Learning conditional preference network from noisy samples using hypothesis testing. *Knowledge-Based Systems*, 40:7–16, 2013.

[23] Juntao Liu, Yi Xiong, Caihua Wu, Zhijun Yao, and Wenyu Liu. Learning conditional preference networks from inconsistent examples. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):376–390, 2014.

[24] Loizos Michael and Elena Papageorgiou. An empirical investigation of ceteris paribus learnability. In *International Joint Conference on Artificial Intelligence*, pages 1537–1543. AAAI Press, 2013.

[25] Hongbing Wang, Jie Zhang, Yangyu Tang, and Shizhi Shao. Collaborative approaches to complementing qualitative preferences of agents for effective service selection. In *IEEE International Conference on Tools with Artificial Intelligence*, pages 51–58. IEEE, 2011.

Vita

Cory Siler

Education

- Bachelor of Science in Computer Science, University of Kentucky, 2016.

Professional Positions

- Research Assistant, University of Kentucky Department of Computer Science, 2017.
- Repperger Research Intern, Air Force Research Laboratory, 2017.

Scholastic and Professional Honors

- Outstanding MS Student in Computer Science, 2016 and 2017.

Professional Publications

- Cory Siler, Luke Harold Miles, and Judy Goldsmith. The complexity of campaigning. In *International Conference on Algorithmic Decision Theory*, 2017.
- Thomas E Allen, Cory Siler, and Judy Goldsmith. Learning tree-structured CP-nets with local search. In *International Florida Artificial Intelligence Research Society Conference*, 2017.
- Cory Siler. Tiered coalition formation games. In *International Florida Artificial Intelligence Research Society Conference*, 2017.