Summarizing Conditional Preference Networks

A Thesis
Submitted to The Faculty of Graduate Studies and Research
in Partial Fulfillment of the Requirements
For the Degree of

Master of Science
in
Computer Science
University of Regina

By
Abu Mohammad Hammad Ali

Regina, Saskatchewan
April, 2019

© Copyright 2019: A.M.Hammad Ali
Abu Mohammad Hammad Ali, candidate for the degree of Master of Science in Computer Science, has presented a thesis titled, *Summarizing Conditional Preference Networks*, in an oral examination held on April 16, 2019. The following committee members have found the thesis acceptable in form and content, and that the candidate demonstrated satisfactory knowledge of the subject material.

External Examiner: Dr. Allen Herman, Department of Mathematics & Statistics

Supervisor: Dr. Sandra Zilles, Department of Computer Science

Committee Member: Dr. Malek Mouhoub, Department of Computer Science

Committee Member: Dr. Howard Hamilton, Department of Computer Science

Chair of Defense: Dr. Brian Sterenberg, Department of Chemistry & Biochemistry
Abstract

Preference modeling has been studied extensively in the literature, and has applications in recommender systems and automated decision-making. The eventual objective of working with preference models is to be able to reason about preferences over objects, often referred to as outcomes. In most of the literature, each outcome is described as an assignment of values to a set of attributes. Representing and reasoning about preferences over outcomes calls for efficient preference models. In this thesis, we focus on one such model, Conditional Preference Networks (CP-nets). A CP-net is a graphical model that captures the preferences of an individual using a directed graph, with vertices representing attributes and edges representing dependency relations between attributes. Information about the preferential dependence/independence between attributes can be leveraged to efficiently order outcomes without exhaustively comparing all attributes in a pair of outcomes. In most existing studies, it is assumed that each individual user has their unique CP-net representing their preferences. In this thesis, we propose an approach to aggregate the preferences of multiple users via a single CP-net, while minimizing disagreement with individual users. We assume that each user has their preferences represented via a separable CP-net, i.e., a CP-net without any edges between attributes. Our goal is to represent the preferences of a group of users using a single CP-net, referred to as a summary CP-net. We present two algorithms that assume all the input CP-nets are separable, with results on correctness
and complexity for each algorithm. We also present a discussion on some important properties of CP-nets and the impact these have on our algorithms.
Acknowledgements

I would like to thank Dr. Sandra Zilles for her constant guidance and patience, and her invaluable mentoring during the last two years. I am also grateful to Dr. Howard Hamilton for his many comments and insightful questions that have helped me ask better questions, probe deeper into answers, and in general made me a better researcher. Thanks are also due to Dr. Malek Mouhoub for his early comments while I was choosing a research topic to focus on. Lastly, I gratefully acknowledge Dr. Mumit Khan, whose guidance and counsel in my college days keep me going to this day in my academic aspirations.
Contents

Abstract .......................................................... i

Acknowledgements ............................................... i

Contents .......................................................... ii

List of Figures ................................................... v

List of Tables ...................................................... vi

1 Introduction .................................................... 1

1.1 Preference Modelling ....................................... 1

1.2 Our Problem ................................................ 4

1.3 Approach and Contributions of This Thesis ............. 5

2 Background .................................................... 7

2.1 CP-net Syntax ............................................... 7
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.2</td>
<td>Absolute Distance</td>
<td>59</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Sum of Squares</td>
<td>60</td>
</tr>
<tr>
<td>5.1.4</td>
<td>Square of Sums</td>
<td>60</td>
</tr>
<tr>
<td>5.2</td>
<td>Absolute Distance</td>
<td>63</td>
</tr>
<tr>
<td>5.2.1</td>
<td>The Algorithm</td>
<td>63</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Proof of Optimality</td>
<td>65</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Complexity Analysis</td>
<td>68</td>
</tr>
<tr>
<td>5.3</td>
<td>Sum of Squares Distance</td>
<td>70</td>
</tr>
<tr>
<td>5.3.1</td>
<td>The Algorithm</td>
<td>70</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Proof of Optimality</td>
<td>72</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Complexity Analysis</td>
<td>75</td>
</tr>
<tr>
<td>5.4</td>
<td>Discussion on Algorithm 3</td>
<td>77</td>
</tr>
<tr>
<td>6</td>
<td>Conclusions and Future Work</td>
<td>85</td>
</tr>
<tr>
<td>References</td>
<td></td>
<td>88</td>
</tr>
</tbody>
</table>
List of Figures

2.1 An example CP-net ........................................... 8

3.1 Example CP-net $N_a$ ........................................ 21

3.2 Example CP-net $N_b$ ........................................ 21

3.3 $N_1$ is one of the CP-nets used to illustrate Algorithm 1 .... 24

3.4 $N_2$ is one of the CP-nets used to illustrate Algorithm 1 .... 25

4.1 Example dependency graph for a CP-net with Unique Topological Ordering ........................................... 42

4.2 The CP-net $N$ discussed in Example 4.3.1 ................. 52

4.3 The CP-net $N_s$ discussed in Example 4.3.1 ............... 52

4.4 The CP-net $N'_s$ discussed in Examples 4.3.2 and 4.3.3 .... 52

5.1 The output CP-net ........................................... 73
List of Tables

3.1 Possible CP-net responses and interpretation . . . . . . . . . . 19
3.2 Outcome pairs and ordering by $N_a, N_b$. Sequence of improving
flips is shown for each non-swap comparison. . . . . . . . . . . 21
3.3 CPTs after Phase 1 . . . . . . . . . . . . . . . . . . . . . . . 30
5.1 Pairwise disagreement values for $N_x$ with absolute distance. The
rows represent input CP-nets and the columns represent at-
tributes. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 65
5.2 Pairwise disagreement values $d_{ij}^2$ for $N_x$ with sum of squares
distance. The rows represent input CP-nets and the columns
represent attributes. . . . . . . . . . . . . . . . . . . . . . . . 72
Chapter 1

Introduction

In this chapter, we discuss the importance of preference representation and common features of most work that has been done to date in the field. We then present our problem and the motivation behind our attempt to solve this problem.

1.1 Preference Modelling

Preference modeling has been studied extensively in the literature, and has applications in recommender systems and automated decision-making. The eventual objective of working with preference models is to be able to reason about preferences over objects, referred to as outcomes. In most of the literature, each outcome is described as an assignment of values to a set of attributes. Representing and reasoning about preferences over outcomes calls
for efficient preference models. A common approach in the preference representation literature is to describe outcomes in terms of multiple attributes, each with a domain of possible values. For example, if we want to represent preferences over different laptop models, we begin by decomposing each laptop as an assignment of values to a set of attributes. These attributes could be brand, model, memory, CPU speed, colour, size, and weight. Specific laptop configurations are outcomes, each of which is defined as an assignment of values to all the attributes. This gives us an outcome space exponential in the number of attributes. Obtaining a preference ordering over this outcome space is intractable without efficient means of preference representation. Several models for representing preferences have been proposed in the literature, such as Lexicographic Preference Trees [1], Conditional Preference Networks [2], utility based models such as the one presented in [3], and other variances of these.

In this thesis, we focus on Conditional Preference Networks (CP-nets). A CP-net is a graphical preference model designed to allow concise and efficient preference representation and reasoning. As mentioned above, we assume outcomes are represented as an assignment of values to attributes. Each attribute has a domain, with dependencies between attributes where applicable. Preferences over attribute domains are interpreted under a ceteris paribus assumption, which means “all else being equal”. For instance, if an individual says they prefer a black laptop over a red one, we take it to mean that for two laptops identical in all other attributes, the user prefers the colour black over the colour red. The preferences over an attribute are stated in a Conditional
Preference Table (CPT) for that attribute.

In our laptop example, each attribute would be represented with a vertex with a corresponding domain of values. For example, the attribute brand could have the domain \{Toshiba, Sony, Dell, Apple\} and the attribute colour could have the domain \{Black, White, Silver, Grey, Red, Blue\}. If a user usually prefers the colour red over black, but prefers black over red if the brand is Sony, then the attribute brand is a parent of the attribute colour, meaning that the preferences over colours depends on the brand. This would be denoted by a directed edge from brand to colour, and the CPT for colour would show the preference ordering over colours conditioned on each possible value of brand.

In theory, there is no upper limit on the number of parents an attribute may have. However, in practice it is unlikely to be arbitrarily large. While the directed graph for a CP-net can be cyclic, the literature often focuses on acyclic graphs because this guarantees the existence of a consistent preference ordering over the outcome space, as mentioned in [2]. The acyclicity assumption has been made in several studies, such as [2], [4], [5] and [6].

Consider a cyclic CP-net \(N\) with attributes \(A\) and \(B\), with \(A\) and \(B\) as parents of each other. Suppose the CPT for \(A\) in \(N\) is \(CPT(N,A) = \{b' : a \succ a', b : a' \succ a\}\), and the CPT for \(B\) in \(N\) is \(CPT(N,B) = \{a' : b' \succ b, a : b \succ b'\}\). Here the symbol \(\succ\) denotes the preference relation, and the statement \(b' : a \succ a'\) would mean “given that the value of \(B\) is \(b'\), the user prefers \(a\) over \(a'\) as a value of \(A\)”. The preference ordering induced by this CP-net is inconsistent, because it induces both \(a'b' \succ a'b\) and \(a'b \succ a'b'\). Note
that this is a consequence of the specific CPTs in this example. In general, cyclic CP-nets may also be consistent. However, unlike for acyclic CP-nets, we cannot guarantee consistency \cite{2}.

In our thesis, we propose an algorithm to measure the disagreement over a pair of CP-nets. This algorithm makes no assumptions about the structure of the CP-net and works with any arbitrary graph structure. The two other algorithms we propose to approximate a group of assume that all the CP-nets we aim to approximate are separable. This means that the underlying graph has no edges.

1.2 Our Problem

In most existing literature, each user is assumed to have a CP-net representing their preferences. In this thesis, we aim to aggregate the preferences of a group of users using a single CP-net called a summary CP-net. Each user has their preferences represented by a CP-net called an input CP-net. Our goal is to find a summary CP-net that best approximates the preferences of the group. The primary motivation for our work lies in possible applications of CP-nets in recommender systems. If each user has a unique CP-net, a large number of CP-nets will need to be stored for a popular recommender system. Further, it is highly likely that there are subgroups of users with very similar preferences. We therefore propose representing a group of input CP-nets with a single summary CP-net approximating each input CP-net in the group. We want to ensure that the summary CP-net agrees with each
of the input CP-nets as closely as possible. To ensure this, we propose two distance measures, described in Chapter 5. We base this measure on a notion of pairwise disagreement, which has applications in clustering similar CP-nets together. In connection to this last point, we can think of our summary CP-net as the centroid for a cluster of CP-nets.

1.3 Approach and Contributions of This Thesis

We begin by defining a notion of pairwise disagreement between CP-nets. This notion helps quantify the degree of similarity between a pair of CP-nets. We propose an algorithm to compute this disagreement in time quadratic in the size of the CP-nets. We then build on the pairwise disagreement to propose two distance measures quantifying how well a summary CP-net represents the preferences of a group of users. Our eventual objective is to find a summary CP-net minimizing the distance from the set of input CP-nets. We propose two algorithms to solve this problem, one for each of the distance measures.

A separable CP-net is one in which the dependency graph has no edges. This means that the preferences over the domain of any given attribute are independent of the values in all the other attributes. Separable CP-nets offers several simplifications that make it a particularly easy case to begin our study with. Our proposed algorithms assume all input CP-nets are separable, and find the optimal summary CP-net in time linear in the number of input CP-
nets and the number of attributes. We also provide a proof of correctness for both algorithms.

The rest of this thesis is structured as follows. In Chapter 2 we present details on the notation and terminology used in the rest of the thesis. We also present a brief discussion of related work. Chapter 3 presents essential definitions, some preliminary results, and a first formal problem statement. Chapter 4 provides some results that justify our focus on computing the disagreement between a pair of CP-nets for only those outcome pairs that differ on the value assigned to exactly one attribute. In Chapter 5, we discuss potential choices for distance measures and present our algorithms to find summary CP-nets relative to these measures with examples, analyses of complexity, and proofs of correctness. Chapter 6 concludes this thesis with a discussion of the limitations of our approach and potential future directions of research.
Chapter 2

Background

In this chapter we introduce the syntax and semantics of CP-nets we will be using for the rest of the thesis. We also discuss other studies that focus on learning CP-nets or aggregating preferences, and how they apply to our problem.

2.1 CP-net Syntax

This section is based largely on the notation and examples used in [2]. A CP-net $N$ is a directed acyclic graph $(V, E)$. Each vertex $V_i \in V$ represents an attribute, with $\text{Domain}(V_i) = \{v_i, v'_i\}$. An edge $e \in E$ from vertex $V_j$ to vertex $V_i$ indicates that the user’s preferences over $V_i$ depend on the value assigned to $V_j$. Preference over an attribute $V_i$ is defined as a total order over

\footnote{Note here that, for simplicity, we assume that all attribute domains are binary.}
the values in \( \text{Domain}(V_i) \). We refer to \( V_j \) as a parent of \( V_i \). It is possible for \( V_i \) to have multiple parents. The set of parents of \( V_i \) in a CP-net \( N \) is denoted by \( \text{Parent}(N,V_i) \). Each \( V_i \) is annotated with a Conditional Preference Table (CPT), denoted by \( CPT(N,V_i) \), which gives the user’s preference order over the values in \( \text{Domain}(V_i) \). Each preference ordering rule is of the form \( v_i \succ v'_i \), indicating that the value \( v_i \) is preferred over the value \( v'_i \). In a complete CP-net \( N \), for attribute \( V_i \) with \( k \) parents, \( CPT(N,V_i) \) has an ordering for each possible instantiation of \( \text{Parent}(N,V_i) \). Formally, if \( |\text{Parent}(N,V_i)| = k \), then \( CPT(N,V_i) \) has \( 2^k \) entries. A given instantiation of the parent set is also referred to as a context for the parent set. The preference statements are also referred to as CPT rules. In practice, CP-nets might often be incomplete. In that case, the CPTs of some attributes will not have all the possible CPT rules for that attribute. This is usually due to lack of information about the user’s preferences. In our study, we assume all CP-nets are complete.

In Figure 2.1 we see an example CP-net \( N \) with attributes \( A \), \( B \) and

![Figure 2.1: An example CP-net](image)
C. Each attribute has a binary domain, and the preferences on C are conditional on the values assigned to A and B. Each attribute has a CPT showing the preference order over its domain values. For A and B, the preference is unconditional, so their CPTs have only one ordering rule. Since Parent(N,C) = \{A, B\}, CPT(N,C) has one ordering rule for each possible assignment of values to A and B.

As mentioned, outcomes are represented as an assignment of values to attributes. In the next section, we see how preferences over attributes and outcomes are represented and reasoned with by CP-nets.

2.2 CP-net Semantics

Preferences over attribute domains are interpreted under a ceteris paribus assumption, which means “all else being equal”. For instance, if an individual says they prefer a black laptop over a red one, we take it to mean that for two laptops identical in all other attributes, the user prefers the colour black over the colour red. The ceteris paribus assumption is natural, and appears in several real-life e-commerce applications [2].

For the remainder of this thesis, we will use the word instantiation of a set to refer to an assignment of values to each attribute in the set. For a given CP-net, we denote the set of all instantiations of a set \(X \subseteq V\) by \(Inst(X)\). An outcome \(o\) is an instantiation of all the attributes in \(V\). \(Inst(V)\) represents the set of all possible outcomes.
Suppose $V = \{ A, B, C \}$, with $\text{Domain}(A) = \{ a, a' \}$, $\text{Domain}(B) = \{ b, b' \}$, $\text{Domain}(C) = \{ c, c' \}$. If $X = \{ A, C \}$, then $\text{Inst}(X) = \{ a'c', a'c, ac', ac \}$, where each member of this set is obtained by choosing one value from the domain of $A$ and one from the domain of $C$ respectively. A sample outcome $o \in \text{Inst}(V)$ would be $a'bc'$. The size of a CP-net is defined as the sum of the number of CPT rules in the CPTs for all the attributes [2]. In Figure 2.1, the size of the CP-net is $1+1+4 = 6$.

The real power of CP-nets comes from the notions of preferential independence and conditional preferential independence. We now provide a brief overview of these terms. Details can be found in [2].

A set of attributes $X \subseteq V$ is preferentially independent of $Y = V - X$ if and only if, for all $x_1, x_2 \in \text{Inst}(X)$ and for all $y_1, y_2 \in \text{Inst}(Y)$, we have

$$x_1 y_1 \succ x_2 y_1 \iff x_1 y_2 \succ x_2 y_2$$

Intuitively, the preference relation over assignments to $X$ is not influenced by the assignment to any attribute in $Y$.

Absolute preferential independence as outlined above is rare. More often, we have conditional preferential independence, defined as follows.

Let $X, Y$ and $Z$ be nonempty sets that partition $V$. $X$ is conditionally preferentially independent of $Y$ given an assignment $z \in \text{Inst}(Z)$ to $Z$ if and only if, for all $x_1, x_2 \in \text{Inst}(X)$ and for all $y_1, y_2 \in \text{Inst}(Y)$, we have
Intuitively, the preference relation over assignments to the set $X$ is not influenced by assignments to any attribute in $Y$, given the fixed assignment $z$ to the attributes in $Z$.

We now show how a given CP-net imposes a preference order over the set of outcomes, and how changing the structure of a CP-net can induce a different preference order.

Consider a CP-net $N$ with the set of attributes $\{A, B\}$, $\text{Parent}(N, A) = \emptyset$ and $\text{Parent}(N, B) = \{A\}$, $\text{Domain}(A) = \{a, a'\}$ and $\text{Domain}(B) = \{b, b'\}$. $\text{CPT}(N, A) = \{a' \succ a\}$, and $\text{CPT}(N, B) = \{a' : b' \succ b, a : b \succ b'\}$. Each CPT rule is used to order an outcome that differs on the value assigned to exactly one attribute. For example, with outcomes $ab'$ and $ab$, we use $\text{CPT}(N, B)$ to see that when $A$ is assigned $a$, $b \succ b'$ and thus $ab \succ ab'$. Similarly, we get $a'b \succ ab$ and $a'b' \succ a'b$. Taking the transitive closure of these orderings provides us with the preference order over the entire set of outcomes, shown below.

$$a'b' \succ a'b \succ ab \succ ab'$$

Each pair in the sequence above is referred to as an improving/worsening flip, since moving from one outcome of the pair to the other improves/worsens the value of a single attribute. A sequence of improving/worsening flips can be used to witness that one outcome is better/worse than another.
If we consider a second CP-net $N'$ which is identical to the one above except that $CPT(N', B) = \{a' : b \succ b', a : b' \succ b\}$, the preference order changes. For this CP-net, based on $CPT(N', B)$, we get $a'b \succ ab$ and in turn, $a'b' \succ a'b$ and $a'b \succ a'b'$. Taking the transitive closure gives us

$$a'b \succ a'b' \succ ab \succ ab'$$

which is the preference order induced by this second CP-net.

 Preferential independence reduces the computational effort needed to find a preference ranking by allowing us to assign values to some attributes knowing that their preferences are not influenced by certain other attributes. CP-nets offer a concise way of representing preferences over attributes while leveraging the idea of preferential independence. Once the CP-net has been constructed, we can carry out reasoning tasks like finding the best outcome(s) or comparing a pair of outcomes. These tasks are referred to as outcome optimization and outcome comparison, respectively. Algorithms for these tasks are described in [2].

### 2.3 Related Work

While CP-nets offer an efficient means of preference representation, these networks have to be constructed before we can use them for reasoning with preferences. One way to build a CP-net would be to ask users to list all attributes that influence their preferences, identify the parent set for each, and specify
the required CPTs. Obviously, this is a tedious process. Further, users are often unaware of their exact preferences. A more practical approach would be to extract the underlying CP-net by observing users make choices over outcomes. For this, we consider pairs of outcomes, indicating which of these is the preferred one. Such examples can be obtained either by observing user choices over time, or by asking the user specific queries.

The two major approaches to learning CP-nets are passive learning and active learning. Approaches using a static set of examples are referred to as passive learning approaches. The alternative is to learn by making interactive queries to the user. Making a query could involve, for example, presenting the user with specific outcome pairs and asking them to state which outcome is preferred. In this case, the learner is not limited to a static set of examples and can actively choose particularly informative pairs to present to the user as a query. In particular, the user can be asked equivalence and membership queries. In an equivalence query, the user is presented a CP-net and asked whether it represents their preferences accurately. The user either responds with an affirmative answer or an outcome pair that is not correctly classified by the presented CP-net. With a membership query, the user is presented an outcome pair and asked to identify the preferred outcome. Such approaches are referred to as active learning. Details on active concept learning can be found in [7] and [8].

Both approaches for learning CP-nets have been attempted in the past. One of the first passive learning approaches can be found in [9]. This study shows, using correspondence with the SAT problem, the general problem of
learning CP-nets is intractable. This justifies their focus on learning separable CP-nets. In [10], the authors extend their work to learn a CP-net consistent with the largest possible subset of examples. The authors in [11] also focus on passive learning. However, they focus on a PAC-learning algorithm that may potentially fail to learn and return an empty CP-net.

The first study to have attempted active learning can be found in [4], where the authors propose an algorithm to learn binary CP-nets using a combination of equivalence and membership queries. They also assume that the target CP-net is represented by either an acyclic or a tree-structured graph, with separate algorithms for the two cases. Another active learning approach can be found in [5], which proposes an online learning algorithm that is guaranteed to return a complete CP-net in polynomial time, provided there is an upper bound on the the number of parents for any attribute.

Some other studies on learning CP-nets are found in [12], [13], [14], [15], and [16]. Results on the learning complexity of CP-nets can be found in [6].

Most of the learning attempts mentioned above assume that each individual user has their own CP-net and is able to give consistent responses to queries about preferences over outcomes. More recently, studies like [17] and [18] attempt query-based learning of acyclic CP-nets from noisy data. One of the reasons the authors give for noisy data is attempting to build a CP-net that represents the preferences of a group of users, some of whom might have contradictory preferences. Their proposed algorithm first tries to resolve such contradictory preferences by searching for a possible parent to the relevant
attribute. However, if the search for such a parent fails, all they do is list a set of outcome pairs for which their CP-net does not have a consistent response. On the other hand, we build a CP-net that approximates each of the input CP-nets to some degree, and minimizes the distance measure we propose in Chapter 4.

A third study, [19], is more relevant to the problem we attempt to solve in this thesis. In this study, the authors propose an algorithm for the online learning of an acyclic CP-net from noisy data. Unlike the previous studies, this study uses information-theoretic measures to deal with the noisy data and construct an acyclic CP-net that maximizes agreement with observed outcome pairs. Due to the online setting, this study assumes the input consists of outcome pairs in the form of a stream of pairwise comparisons. At any given time instant, their objective is to have a CP-net that maximizes agreement with pairs observed up to that time instant.

The study in [19] comes closest to solving essentially the same problem as ours, but there are still some differences. Firstly they are dealing with an online setting, while we assume all the input CP-nets are available to us at once. We assume the inputs are separable CP-nets, as opposed to pairs of outcomes with their preference order. We also focus mostly on the problem of clustering similar CP-nets, which can then be represented by a summary CP-net as the centroid for the cluster. We also include a discussion on some interesting properties of CP-nets, and how these properties can affect the attempt to find a summary CP-net.
A handful of studies have focused on preference aggregation, our eventual goal. The authors in \cite{20} propose the notion of mCP-nets, aimed at working with the preferences of \( m \) users. However, their suggested approach works with a set of partial CP-nets and various voting schemes to find the outcome order that best represents the collective preference of the group. Another study, \cite{21}, also attempts preference aggregation by aiming for collective decision-making based on social choice theory. In contrast, our approach outputs a summary CP-net. The biggest advantage of this form of output is that all known theoretical results about CP-nets apply to our output, and existing algorithms can be used for reasoning with the preferences expressed in the summary CP-net.

More recently, studies like \cite{22}, \cite{23} and \cite{24} have proposed a notion of CP-net distance. These studies define the notion of distance between CP-nets by generalizing on the Kendall-Tau Distance between partial orders. However, it should be mentioned here that while they propose approximate distance functions, their study focuses on using this to quantify the differences over pairs of CP-nets and not for group preference aggregation in the same way that we do.
As stated in Chapter 1, most literature on CP-nets assumes each user has a unique CP-net. However, we hypothesize that CP-nets for certain groups of users may have more in common with each other than with those in other groups. We are essentially claiming that a set of CP-nets can be partitioned into similarity-based clusters. Given such a cluster, it would be useful if all CP-nets in a given cluster could be summarized by a single representative CP-net. This would avoid over-fitting and make it easier to model the preferences of a new user by finding the representative CP-net that best approximates their preferences. Further, being able to represent a group of CP-nets with a single CP-net would enable us to implement agglomerative hierarchical clustering.
Thus, it would be useful to design an algorithm that finds a summary CP-net given a set of input CP-nets. We want to formalize this problem statement. In order to do so we first define some terms.

### 3.1 Measures of CP-net Similarity

We begin by discussing the possible responses dictated by the semantics of a CP-net when presented with an outcome ordering query. Given an outcome pair \((o, o')\) and a CP-net \(N\), there are three possible responses according to dominance testing. \(N\) could entail \(o \succ o'\), or \(o' \succ o\), or find the two outcomes incomparable. We want to define the disagreement between two CP-nets based on their responses to these queries. It is easy to define disagreement for the first two responses. The third response is trickier, since at least one of the CP-nets cannot order the outcomes either way. Incomparability between two outcomes can have multiple interpretations. For our purposes, we choose to interpret incomparability as the user being indifferent between the outcomes in the pair. This interpretation forms the basis of how we define our disagreement measure.

Let us use \(N\) to denote one of the input CP-nets, and \(N_s\) to denote the summary CP-net. If \(N\) cannot compare an outcome pair \((o, o')\) but \(N_s\) can, we will count this as an agreement. The intuition is that since \(N\) is indifferent, it is acceptable for \(N_s\) to impose an ordering to better approximate other CP-nets in the input set. This is similar to the concept of weak dominance over outcome pairs. On the contrary, if \(N\) has a concrete response to the ordering
Table 3.1: Possible CP-net responses and interpretation

<table>
<thead>
<tr>
<th>$N$</th>
<th>$N_s$</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$o_1 \succ o_2$</td>
<td>$o_1 \succ o_2$</td>
<td>agree</td>
</tr>
<tr>
<td>$o_1 \succ o_2$</td>
<td>$o_2 \succ o_1$</td>
<td>disagree</td>
</tr>
<tr>
<td>incomparable</td>
<td>$o_1 \succ o_2$</td>
<td>agree</td>
</tr>
<tr>
<td>$o_1 \succ o_2$</td>
<td>incomparable</td>
<td>disagree</td>
</tr>
</tbody>
</table>

query but $N_s$ finds the pair incomparable, we consider this a disagreement. Clearly in this case $N_s$ is losing information about preference ordering and should be penalized. We summarize how pairs of responses are interpreted in Table 3.1.

Next we define a function for the disagreement between a pair of CP-nets for a given outcome pair $(o, o')$. We denote this by $\delta(N, N_s)(o, o')$, and define it as follows.

$$
\delta(N, N_s)(o, o') = \begin{cases} 
0 & \text{if } N \text{ entails neither } o \succ o' \text{ nor } o' \succ o \\
0 & \text{if } N \text{ and } N_s \text{ entail the same order over } (o, o') \\
1 & \text{otherwise}
\end{cases}
$$

We now define the absolute and relative disagreement between a pair of CP-nets. In what follows, $X$ denotes the set of all possible outcome pairs for a given CP-net. Similarly, $X_{\text{swap}}$ denotes all possible swap outcome pairs. For both these sets, we include exactly one of outcome pairs $(o_1, o_2)$ and $(o_2, o_1)$.
Definition 1 Let $N$ and $N_s$ be two CP-nets over the same set of attributes $V$. The Absolute Overall Disagreement of $N$ and $N_s$, denoted by $\Delta_{\text{overall}}(N, N_s)$, is defined as
\[
\Delta_{\text{overall}}(N, N_s) = \sum_{(o, o') \in X} \delta(N, N_s)(o, o')
\]
and the Relative Overall Disagreement of $N$ and $N_s$, denoted by $\delta_{\text{overall}}(N, N_s)$, is defined as
\[
\delta_{\text{overall}}(N, N_s) = \frac{\sum_{(o, o') \in X} \delta(N, N_s)(o, o')}{|X|}
\]

We use the term swap to refer to outcome pairs that differ on the value assigned to exactly one attribute. For example, the outcome pair $(abc, abc')$ is a swap that differs only on the value assigned to $C$.

Definition 2 Let $N$ and $N_s$ be two CP-nets over the same set of attributes $V$. The Absolute Swap Disagreement of $N$ and $N_s$, denoted by $\Delta_{\text{swap}}(N, N_s)$, is defined as
\[
\Delta_{\text{swap}}(N, N_s) = \sum_{(o, o') \in X_{\text{swap}}} \delta(N, N_s)(o, o')
\]
and the Relative Swap Disagreement of $N$ and $N_s$, denoted by $\delta_{\text{swap}}(N, N_s)$, is defined as
\[
\delta_{\text{swap}}(N, N_s) = \frac{\sum_{(o, o') \in X_{\text{swap}}} \Delta(N, N_s)(o, o')}{|X_{\text{swap}}|}
\]

Definition 3 Let $N$ and $N_s$ be two CP-nets over the same set of attributes $V$. The Relative Overall Agreement between $N$ and $N_s$ is given by $1 - \delta_{\text{overall}}(N, N_s)$. Analogously, the Relative Swap Agreement of $N$ and $N_s$ is given by $1 - \delta_{\text{swap}}(N, N_s)$.
Table 3.2: Outcome pairs and ordering by $N_a, N_b$.
Sequence of improving flips is shown for each non-swap comparison.

<table>
<thead>
<tr>
<th>$o_1$</th>
<th>$o_2$</th>
<th>$N_a$</th>
<th>$N_b$</th>
<th>$\delta(N_a, N_b)(o_1, o_2)$</th>
<th>$\delta(N_b, N_a)(o_1, o_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a' b'$</td>
<td>$a' b$</td>
<td>$o_2 &gt; o_1$</td>
<td>$o_2 &gt; o_1$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$a' b'$</td>
<td>$a' b'$</td>
<td>$o_2 &gt; o_1$</td>
<td>$o_2 &gt; o_1$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$a' b'$</td>
<td>$a b$</td>
<td>$o_2 &gt; o_1$</td>
<td>$o_2 &gt; o_1$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$a b$</td>
<td>$a b'$</td>
<td>Incomparable</td>
<td>$o_2 &gt; o_1$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$a' b$</td>
<td>$a b$</td>
<td>$o_2 &gt; o_1$</td>
<td>$o_2 &gt; o_1$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$a' b$</td>
<td>$a b$</td>
<td>$o_2 &gt; o_1$</td>
<td>$o_2 &gt; o_1$</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$^{a}a' b' \rightarrow a b' \rightarrow a b$
$^{b}a' b' \rightarrow a b' \rightarrow a b$
$^{c}a' b \rightarrow a' b' \rightarrow a b$

As the reader may notice, our disagreement measure is not symmetric. Given a pair of CP-nets $N_a$ and $N_b$, $\Delta_{overall}(N_a, N_b)$ might not be equal to $\Delta_{overall}(N_b, N_a)$. We illustrate this using an example.

Consider the pair of CP-nets $N_a, N_b$ shown in Figures 3.1 and 3.2. $N_a$ cannot compare between the outcomes $a' b$ and $ab'$, but $N_b$ can. Table 3.2 shows all possible outcome pairs, the ordering for each pair by both $N_a$ and $N_b$, and the value of the disagreement function for each outcome pair. Summing
these values for all possible pairs gives us the overall disagreement between this pair of CP-nets.

Based on the definitions provided earlier in this chapter, $\Delta_{\text{overall}}(N_a, N_b) = 1/6$ and $\Delta_{\text{overall}}(N_b, N_a) = 2/6$. This difference occurs because the case $o_1 = a'b$ and $o_2 = ab'$ in Table 3.2 is not symmetric. One of these is considered to have lost information and thus counts as a disagreement.

Swap disagreement is symmetric. Swaps can never be incomparable because swaps differ on the value of exactly one attribute. All other attributes have identical values in either outcome, and the CPT of the swapped attribute can always tell us the preference order for the pair.

We will often refer to a swap being ordered by a given CPT. We explain what we mean by this phrase. Consider the outcome pair $(ab'c', a'b'c')$. This is a swap, with $A$ as the swapped attribute. Since attributes $B$ and $C$ have the same value in both outcomes, any preference for one outcome over the other can depend only on $A$. In such a case, we say that the CPT for $A$ orders the swap $(ab'c', a'b'c')$.

We now have all the terminology needed to propose a first problem statement.

**Problem Statement 1** Given a set of CP-nets $T = \{N_1, N_2, ..., N_t\}$ as input, find a summary CP-net $N_s$ that minimizes the overall disagreement of this summary CP-net with each CP-net in $T$.

Given that overall disagreement is defined between a pair of CP-nets, an
obvious question is what it means for a summary CP-net to minimize overall
disagreement with an entire set of CP-nets. One option would be to minimize
the maximum of all pairwise disagreements and another option would be to
minimize the average pairwise disagreement. We will address this question
later in this thesis, after making further refinements to our problem statement.

3.2 Efficient Computation of Swap Disagreement

In this section, we propose a procedure for computing the swap disagreement
between an arbitrary pair of CP-nets without enumerating all swaps. Avoiding
enumeration is important because the number of swaps is exponential in the
number of attributes and can be exponential in the size of the CP-nets.

There are two reasons why we focus on computing swap disagreement
rather than overall disagreement. Firstly, the correct ordering for swaps can
be determined by looking at the CPT of a single attribute. In contrast, the
ordering for arbitrary outcome pairs depends on multiple CPTs. Secondly, we
demonstrate in Chapter 4 that there is a relationship between swap and over-
all disagreement for some classes of CP-nets, such that minimizing the swap
disagreement is necessary when attempting to minimize the overall disagree-
ment.

When dealing with swaps, the outcome ordering depends only on the CPT
for the swapped attribute. We can therefore compute the disagreement over
all swaps by comparing the CPT rules in the two CP-nets. As mentioned previously, the total number of CPT rules gives the size of the corresponding CP-net. Thus, our proposed algorithm can compute the swap disagreement in time quadratic in the product of the sizes of the two CP-nets, as we will see below.

The CPT rule for any attribute has two components — an instantiation of the parent set of the attribute, henceforth referred to as the context, and the preference ordering over the domain of the attribute. For attributes whose preference is not conditioned on any other attributes, the parent set is empty. The CPTs for such attributes only contain one unconditional ordering rule, without any context.

We use the CP-nets $N_1$ and $N_2$, which are shown, in Figures 3.3 and 3.4 respectively, to illustrate the procedure. The pseudo-code is given Algorithm
1. Before we explain the pseudo-code, we first establish some fundamental properties.

$\text{Parent}(N,V_i)$ denotes the set of parents of $V_i$ in CP-net $N$. $\text{CPT}(N,V_i)$ represents the CPT for $V_i$ in $N$. Consider attribute $V_i$ in $N$, $|\text{Parent}(N,V_i)| = k$. Suppose $s$ is a rule in $\text{CPT}(N,V_i)$. We refer to the number $2^n-k-1$ as the weight of $s$. We now justify the use of this term and show how it helps compute the swap disagreement. Recall that for each possible swap $(o,o')$, we count exactly one of $(o,o')$ and $(o',o)$. Since we focus on complete CP-nets, knowing the correct ordering for either pair gives us the correct ordering for the other.

**Lemma 1** Let $N$ be a CP-net over a set of attributes $V$, with $|V| = n$. Let $V_i \in V$ be an attribute in $N$ with $k$ parents. Then each preference rule in $\text{CPT}(N,V_i)$ correctly determines the ordering of $2^{n-k-1}$ swaps.

**Proof.** For swaps of any attribute $V_i$, the $k$ parents have a fixed context, and $V_i$ takes on different values in each outcome of the swap. This leaves $n-k-1$
other attributes that can be assigned arbitrary values. Since we are dealing
with swaps of \( V_i \), these \( n - k - 1 \) attributes must be assigned the same value
in both outcomes. Each possible assignment of values corresponds to a swap.
Given that \( V_i \) is the only attribute with different values in each outcome of
the pair, all such instances are correctly ordered using \( CPT(V_i) \). There are
exactly \( 2^{n-k-1} \) such instances. 

In Figures 3.3 and 3.4, for attribute \( A \), \( N_1 \) and \( N_2 \) have the same CPT,
which helps order four swaps, given by the four possible ways to assign val-
ues to \( B \) and \( C \). \( CPT(N_1,B) \) and \( CPT(N_2,B) \) are also identical. Each rule
has a fixed context for the parent attribute \( A \), and corresponds to the two
swaps given by the two possible values for \( C \). For instance, when the con-
text for \( A \) is fixed to \( a \), with \( B \) as the swapped attribute we can create two
swaps, by assigning \( C \) the value of either \( c \) or \( c' \). Lastly, for \( CPT(N_1,C) \) and
\( CPT(N_2,C) \), exactly half the rules match. Each rule has a fixed context for
the parent attributes \( A \) and \( B \). Thus, each rule corresponds to only one swap.

By Lemma 1 in a CP-net \( N \) and for an attribute \( V_i \) with \( k \) parents, there
are \( 2^{n-k-1} \) swaps ordered using each rule in \( CPT(N,V_i) \). The lemma thus
justifies why we refer to this number as the weight of the CPT rule — the
higher the weight of a CPT rule, the greater the number of swaps ordered
using this rule.

**Definition 4** Let \( N \) be a CP-net over a set of attributes \( V \), with \( |V| = n \). Let
\( V_i \in V \) be an attribute in \( N \) with \( k \) parents. Let \( s \) be a rule in \( CPT(N,V_i) \)
with context \( \beta \). Then the weight of \( s \), denoted \( w(s) \), is the number of swaps
whose ordering is determined by \( s \), which is \( 2^{n-k-1} \) by Lemma \[3\]. Likewise, we use the term weight of \( \beta \), denoted \( w(\beta) \), to refer to the same number.

We now introduce some notation and make two observations critical for the design of our algorithm to compute swap disagreement. \( S(N) \) is the set of all CPT rules for a CP-net \( N \). Each rule \( s_i \in S(N) \) has a corresponding weight \( w(s_i) \).

**Definition 5** Let \( N, N' \) be two arbitrary CP-nets over the same set of attributes \( V \). Let \( V_i \in V \). Then \( R(N, N', V_i) \) denotes the set of all contexts over \( \text{Parent}(N, V_i) \cup \text{Parent}(N', V_i) \) for which \( N \) and \( N' \) provide different preference orders over \( V_i \). Each context \( \beta \in R(N, N', V_i) \) has a corresponding weight \( w(\beta) \). Moreover, we define \( R(N, N') = \bigcup_{V_i \in V} R(N, N', V_i) \).

For example, consider attribute \( A \), \( \text{Parent}(N, A) = \emptyset \) and \( \text{Parent}(N', A) = \{B, C\} \). Suppose \( \text{CPT}(N, A) = \{a \succ a'\} \) and \( \text{CPT}(N', A) = \{bc : a \succ a', b'c' : a \succ a', bc' : a' \succ a, b'c' : a' \succ a\} \). Then \( R(N, N', A) = \{bc', b'c\} \).

We use \( X_{\text{swap}} \) to denote the set of all swaps. As mentioned above, we count swaps like \((o, o')\) and \((o', o)\) exactly once.

**Observation 1** The sum of the weights of all CPT rules across all the CPTs in a CP-net \( N \) gives us the total number of swaps for CP-net \( N \).

\[
|X_{\text{swap}}| = \sum_{s \in S(N)} w(s)
\]
Observation 2  The sum of the weights of the contexts for which a pair of CP-nets \((N, N')\) provide different preference orderings gives us the number of swaps on which they disagree. This is referred to as the absolute swap disagreement, according to Definition 2.

\[
\Delta_{\text{swap}}(N, N') = \sum_{\beta \in R(N, N')} w(\beta)
\]

Algorithm 1 gives a procedure to compute the swap disagreement between two CP-nets. Let us begin by walking through Algorithm 1 for CP-nets \(N_1\) and \(N_2\) as shown in Figures 3.3 and 3.4. \(N_1\) and \(N_2\) have one structural difference, namely that \(\text{Parent}(N_1, C) = \{A, B\}\) and \(\text{Parent}(N_2, C) = \{A\}\).

For each attribute we take the union of its parent sets in steps 4-6. At the end of this process, the union parent sets are \(\text{Parent}(A) = \phi, \text{Parent}(B) = \{A\}\), and \(\text{Parent}(C) = \{A, B\}\). Next we check each CP-net for attributes whose parent sets do not match the combined parent sets obtained after the union. This is done in phase 1 of our algorithm.

Every attribute in \(N_1\) has the same parent set as the corresponding union parent set, so no changes are made. For \(N_2\), \(A\) and \(B\) have the same sets, but \(C\) has a different parent set. So the two rules in \(CPT(N_2, C)\) are removed, and appropriate CPT rules are added for each possible instantiation of the union parent set. The CPTs obtained after phase 1 are shown in Table 3.3.

In phase 2, we iterate over each attribute. For each attribute, we compute the weight of its CPT rules in line 21. In line 22, \(z\) denotes the number of
Algorithm 1: A procedure to compute the relative swap disagreement of two CP-nets

\begin{algorithm}
\begin{algorithmic}[1]
\State \textbf{ComputeSwapDisagreement} \((N, N')\);
\State \textbf{Input} : Two CP-nets \(N_1\) and \(N_2\)
\State \textbf{Output}: \(\delta_{\text{swap}}(N_1, N_2)\)
\State \textbf{weightsum} = 0
\State \textbf{disagreement} = 0
\For \(V_i \in V\)
\State \textbf{Parent}(\(V_i\)) = \textbf{Parent}(N_1, V_i) \cup \textbf{Parent}(N_2, V_i)
\EndFor
\State \textbf{Beginning of Phase 1}
\For \(i \in \{1, 2\}\)
\For all \(V_i \in V\) with \textbf{Parent}(\(V_i\)) \(\neq\) \textbf{Parent}(\(N_i, V_i\))
\For each rule \(\beta : a \succ a' \in \text{CPT}(N_i, V_i)\)
\State remove \(\beta : a \succ a'\) from \(\text{CPT}(N_i, V_i)\)
\For each \(\beta' \in \text{Inst}(\text{Parent}(V_i) - \text{Parent}(N_i, V_i))\)
\State add \(\beta' : a \succ a'\) to \(\text{CPT}(N_i, V_i)\)
\EndFor
\EndFor
\EndFor
\EndFor
\State \textbf{End of Phase 1}
\State \textbf{Beginning of Phase 2}
\For all \(V_i \in V\)
\State \textbf{weight} = \(2^{|V| - |\text{Parent}(V_i)|} - 1\)
\State \(z = |\text{CPT}(N_1, V_i) \cap \text{CPT}(N_2, V_i)|\)
\State \textbf{weightsum} = \textbf{weightsum} + \textbf{weight} \cdot |\text{CPT}(N_1, V_i)|
\State \textbf{disagreement} = \textbf{disagreement} + \textbf{weight} \cdot |\text{CPT}(N_1, V_i)| - z
\EndFor
\State \(\delta_{\text{swap}}(N_1, N_2) = \text{disagreement}/\text{weightsum}\)
\State \textbf{End of Phase 2}
\end{algorithmic}
\end{algorithm}
rules on which the two CPTs agree. We use the variable \textit{weightsum} to store the sum of weights of all rules. For a given attribute, the sum of weights of all its rules is computed and added to the total in line 23. In line 24, we compute the sum of the weights of the rules on which the two CPTs disagree and add this to the total disagreement so far. We get the number of disagreements by subtracting \( z \) from the size of either CPT, since after phase 1 the CPTs in both CP-nets have the same size.

Attributes \( A \) and \( B \) have the same CPTs and no disagreement. \( A \) has one rule of weight 4. \( B \) has two rules, each of weight 2. These weights add up to 8 and a disagreement of 0. \( \text{CPT}(N_1, C) \) and \( \text{CPT}(N_2, C) \) both have four rules, and two of them do not agree. Each rule has weight 1, resulting in a total of 2 swaps on which \( N_1 \) and \( N_2 \) disagree. Thus, the absolute swap disagreement is 2. The sum of all weights, and thus the number of swaps, is 12. The relative swap disagreement is 2/12.
3.2.1 Proof of Correctness

In this section, we prove the following lemma.

**Lemma 2** Algorithm 1 correctly computes the relative swap disagreement between a pair of CP-nets.

**Proof.** Algorithm 1 runs in two phases. In phase 1, we ensure that both input CP-nets have the same number of CPT rules. In phase 2, we compare the corresponding CPT rules and count disagreements.

We first prove that at the end of phase 1, $N_1$ and $N_2$ have the same number of CPT rules.

In lines 4-6, we create a combined parent set for each attribute by taking the union of their respective parent sets. Let us suppose that $|\text{Parent}(N_1, V_i) \setminus \text{Parent}(N_2, V_i)| = p_1 (\geq 0)$ and $|\text{Parent}(N_2, V_i) \setminus \text{Parent}(N_1, V_i)| = p_2 (\geq 0)$.

Before phase 1, $CPT(N_1, V_i)$ has $2|\text{Parent}(N_1, V_i)|$ rules and $CPT(N_2, V_i)$ has $2|\text{Parent}(N_2, V_i)|$ rules. Algorithm 1 replaces each rule in $CPT(N_1, V_i)$ by $2p_2$ rules, one for every context $\beta'$ over $|\text{Parent}(N_2, V_i) \setminus \text{Parent}(N_1, V_i)|$. Symmetrically, it also replaces each rule in $CPT(N_2, V_i)$ by $2p_1$ rules, one for every context $\beta'$ over $|\text{Parent}(N_1, V_i) \setminus \text{Parent}(N_2, V_i)|$.

Therefore after phase 1, the number of rules in $CPT(N_1, V_i)$ is

$$2|\text{Parent}(N_1, V_i)| \cdot 2^{p_2} = 2^{|\text{Parent}(N_1, V_i)|+p_2}$$
Similarly, the number of rules in $CPT(N_2, V_i)$ is

$$2^{|Parent(N_2, V_i)|} \cdot 2^{p_1} = 2^{|Parent(N_1, V_i)| + p_1}$$

It can be easily checked that these two numbers are equal. In particular, we have that $|Parent(N_1, V_i)| + p_2 = |Parent(N_2, V_i)| + p_1 = |Parent(N_1, V_i) \cup Parent(N_2, V_i)|$. Thus, after phase 1 both CP-nets have the exact same number of CPT rules.

In phase 2, for the CPT of each attribute, we compute the weight of the rules in the CPT and the number of disagreements. Using these, we can compute the sum of the weights of all rules in the CPT, and that of the number of rules that disagree. Since $CPT(N_1, V_i)$ and $CPT(N_2, V_i)$ have the same size, the size of their intersection set gives us the number of rules on which the two CPTs disagree. Subtracting this from the CPT size gives us the number of disagreements. In line 23 we maintain the sum of the weights of all rules for CPTs seen so far. In line 24 we maintain the sum of the weights of the rules that disagree among the CPTs seen so far. Based on Observations 1 and 2 we see that at the end of all iterations, $weightsum$ stores the total number of swaps for all attributes, and $disagreement$ stores the number of swaps that are ordered differently, which is the absolute swap disagreement. Line 26 then gives us the proportion of swaps on which the CP-nets disagree, which is the relative swap disagreement. This concludes our proof.
3.2.2 Complexity Analysis

We now establish the runtime complexity of Algorithm 1.

**Lemma 3** Algorithm 1 runs in time quadratic in the product of the sizes of the input CP-nets.

**Proof.** Algorithm 1 runs in two major phases. In phase 1, the outermost loop only runs twice. The second loop runs once for each attribute, and for each run the third loop scans all rules in the CPT of the attribute. For each attribute $V_i$ in the second loop, the third loop thus runs $|CPT(V_i)|$ times. Together, the second and third loop run $\sum_{V_i \in V} |CPT(V_i)|$ times. By definition, this expression gives the size of the CP-net. In the worst case, the fourth loop runs once for each possible instantiation of the parent set, for a total of $2^k$ times, where $k$ is the number of parents for the attribute under consideration. We assume that $k$ is small compared to $n$. Thus the runtime of phase 1 is dominated by the second and third loop, which run in time quadratic in the sum of the sizes of the two CP-nets.

Phase 2 consists of just one loop that runs in time linear in $|V|$. All statements inside this loop other than line 22 run in constant time. With an efficient hashtable implementation, line 22 can be executed in time linear in the size of the CPTs. Since the loop runs once for each attribute, this takes time at most quadratic in CP-net size, just like phase 1.

The runtime of Algorithm 1 is therefore at most quadratic in the CP-net size. Phase 1 of Algorithm 1 ensures that all CPTs in $N_1$ and $N_2$ have the
same size, and the worst case CPT size is bounded by the product of the corresponding CPT sizes in $N_1$ and $N_2$. Since the sum of the CPT sizes is the size of the CP-net, this proves Lemma 3.

We now state the following theorem, which is a direct consequence of Lemmas 2 and 3.

**Theorem 1** Algorithm 1 computes the swap disagreement between a pair of CP-nets correctly in time quadratic in the sizes of the individual CP-nets.
Chapter 4

Relationship Between Swap and Overall Disagreement

In the previous chapter, we proposed a polynomial time algorithm to compute the swap disagreement over a pair of CP-nets. Since no polynomial time algorithm to compute overall disagreement is known at present, we will be using swap disagreement to define our distance measure in the following chapter. Given that this distance measure will be used as our objective function, this raises the question of whether swap disagreement is a good predictor of overall disagreement. In this chapter, we explore the relationship between these two parameters. For the purposes of constructing a summary CP-net that is a good approximation to each of the input CP-nets, it would be sufficient to show that minimizing the pairwise swap disagreement of two CP-nets also minimizes their pairwise overall disagreement. It would also be useful to provide a non-trivial upper bound on overall disagreement as a function of swap
disagreement. In this chapter, we present some results on the relationship be-
tween swap and overall disagreement, classified based on structural properties
of the dependency graph of the CP-nets under consideration.

4.1 CP-nets with Unique Topological Orderings

In this section, we discuss the relationship between swap and overall disagree-
ment for a class of CP-nets that share a structural property related to their
dependency graph. We define this property, and then present results that
apply to this class of CP-nets.

4.1.1 Unique Topological Ordering Property

We say that a CP-net $N$ has a unique topological ordering if the underlying
dependency graph for $N$ has a unique topological ordering. Such CP-nets are
interesting because their attributes have a unique order of importance.

If an attribute $V_i$ appears in the topological order before another attribute
$V_j$, we say that $V_i$ is more important than $V_j$. If $V_i$ is more important than
$V_j$, in situations where both $V_i$ and $V_j$ cannot be assigned their most preferred
value, the CP-net semantics dictates that it is more important to optimize the
assignment for $V_i$. This is because the algorithm for ordering a pair $(o_1, o_2)$
presented by the authors in \cite{2} compares each attribute according to the topo-
logical order of the attributes, until it finds one that has been assigned different
values in $o_1$ and $o_2$. Then, the outcome which has been assigned the preferred
value for this attribute is the preferred outcome. The correctness of this algo-
rithm has been proven in [2].

From this, it is easy to see why attributes that come earlier in the topolog-
ical ordering would be more important to optimize. Suppose in outcome pair
$(o_1, o_2)$, for the values to their respective parent sets, the CPTs of $V_i$ and $V_j$
dictate that $v_i \succ v'_i$ and $v_j \succ v'_j$. Further, suppose that $o_1$ assigns $v_i$ and $v'_i$ to
$V_i$ and $V_j$ respectively, while $o_2$ assigns $v'_i$ and $v_j$. Lastly, all other attributes
are assigned identical values in each outcome in the pair. Then $o_1 \succ o_2$ because
$o_1$ assigns the preferred value to $V_i$, which appears before $V_j$ in the topological
order and is thus more important.

The above example also demonstrates what happens when a CP-net does
not have a unique topological ordering. Since the order is not unique, there
is at least one pair of attributes $V_i$ and $V_j$ whose relative positions in the
topological order is not unique. This means that the relative importance of
$V_i$ and $V_j$ is not specified. Thus, any outcome pair $(o_1, o_2)$ where $o_1$ optimizes
only $V_i$ and $o_2$ optimizes only $V_j$ cannot be ordered by this CP-net. We say
that the outcome pair $(o_1, o_2)$ calls for a trade-off between $V_i$ and $V_j$, and that
the CP-net cannot support this trade-off between $V_i$ and $V_j$.

In general, suppose we have a set of attributes \{${V_i, V_j, \ldots, V_n}$\} such that
$i < j < \ldots < n$ in the topological order. Then attribute $V_i$ is more important
than $V_j$, which is in turn more important than the next attribute, so on down
to $V_n$. Then, any outcome pair $(o_1, o_2)$ that can only optimize $V_i$ by setting less preferred values for more than one of the other attributes also calls for a trade-off between $V_i$ and those other attributes. Even CP-nets with unique topological ordering are unable to order such outcome pairs. Further, CP-nets in general are unable to order such outcome pairs. This limitation of CP-nets has been discussed in [25], where an extension to the basic CP-net model to support trade-offs is proposed. It should be noted though, that even this extended model does not completely solve the problem.

Consider a CP-net $N$ with $n$ attributes that has a unique topological ordering. We label the attributes $V_1, V_2, ..., V_n$ in the order of their topological ordering. This lets us state the following properties:

- Whenever $i < j$, attribute $V_i$ is more important than attribute $V_j$. For an outcome pair $o_1, o_2$, if $o_1$ assigns a preferred value to $V_i$ only and $o_2$ assigns a preferred value to $V_j$ only, it is more important to optimize $V_i$ than to optimize $V_j$.

- Whenever $i < j$ and $i < k$, attribute $V_i$ is more important than attribute $V_j$ and $V_k$. For an outcome pair $o_1, o_2$, if $o_1$ assigns a preferred value to $V_i$ only and $o_2$ assigns a preferred value to $V_j$ and $V_k$ only, the CP-net is unable to order $o_1, o_2$.

In this section, we focus our discussion on CP-nets with unique topological ordering. This lets us assume that outcome pairs that require trade-off between two attributes can always be ordered according to the CP-net seman-
tics, which then enables us to propose a relationship between swap and overall disagreement for such CP-nets.

4.1.2 Connection with Swap Disagreement

We now discuss how, for a subclass of CP-nets with unique topological ordering, there is a relationship between swap and overall disagreement. We also provide bounds on overall disagreement as a function of swap disagreement for the general class of CP-nets with unique topological ordering.

Consider an attribute $V_j$ with the same parent set in CP-nets $N_1$ and $N_2$ but different CPTs in them. We denote the set of parents by $\text{Parent}(V_j)$. The set of all other attributes $Y$ is then represented by $Y = V \setminus (\{V_j\} \cup \text{Parent}(V_j))$. The exact nature of the relationship between the swap and overall disagreement then depends on the attributes in $Y$.

Let us consider an outcome pair $(o_1, o_2)$ that assigns the same set of values to all attributes in $\text{Parent}(V_j)$, and different values to $V_j$. Lastly, the pair assigns arbitrary values to the attributes in $Y$ for each outcome.

The question we need to answer is whether $(o_1, o_2)$ is ordered using a rule on which $CPT(N_1, V_j)$ and $CPT(N_2, V_j)$ disagree. If they do, then $(o_1, o_2)$ is an outcome pair on which $N_1$ and $N_2$ disagree. If not, this outcome pair is ordered the same way by both $N_1$ and $N_2$ and does not contribute to the disagreement measure. Whether $(o_1, o_2)$ is ordered using $CPT(N_1, V_j)$ and $CPT(N_2, V_j)$ depends on the set $Y$. If $Y$ contains at least one attribute which is more important than $V_j$, and this attribute is assigned different values in $o_1$
and $o_2$, then the ordering does not depend on $V_j$. We now explore this point.

Let us assume that $Y' \subseteq Y$ is the set that contains those attributes in $Y$ that are more important than $V_j$. The relationship between swap and overall disagreement depends on the size of and the values assigned to attributes in $Y'$.

For $(o_1, o_2)$, we already know that $\text{Parent}(V_j)$ has the same context in both outcomes, and that $V_j$ takes on different values in $o_1$ and $o_2$. However, whether $(o_1, o_2)$ is ordered using the CPT of $V_j$ depends on the values assigned to attributes in $Y'$. Since any attribute in $Y'$ is more important than $V_j$, the ordering only depends on $V_j$ when all attributes in $Y'$ have the same values in $o_1$ and $o_2$. The $|Y'|$ attributes can be assigned the same value in both outcomes in $2^{|Y'|}$ ways. The remaining $|Y| - |Y'|$ attributes can be assigned values arbitrarily for each outcome, in $2^{2(|Y| - |Y'|)}$ ways. This gives us a total of $2^{|Y'|} \cdot 2^{2(|Y| - |Y'|)}$ outcome pairs. Simplifying this last expression we get $2^{|Y|} \cdot 2^{2(|Y| - |Y'|)}$.

If $|\text{Parent}(V_j)| = k$, then $|Y| = n - k - 1$. Thus, the numerator of our expression is precisely the square of the weight of the corresponding rule in the CPT of $V_j$, defined in Definition 4. In other words, the numerator is a function of the swap disagreement. This allows us to state the following lemma for the special case when $|Y'| = 0$ and the denominator is 1.

**Lemma 4** Consider a pair of CP-nets $N_1$ and $N_2$ with unique topological ordering and an attribute $V_i$, where $Y = V \setminus (\{V_j\} \cup \text{Parent}(V_j))$. If the swap disagreement between $N_1$ and $N_2$ for a context $\beta \in \text{Parent}(N_1, V_i) \cup \text{Parent}(N_2, V_i)$ is $w(\beta)$, and no attributes in $Y$ are more important that $V_i$, then...
then the overall disagreement due to \( \beta \) is \((w(\beta))^2\). When \(|Y'| > 0\), the denominator is no longer a constant, and the overall disagreement no longer depends only on the swap disagreement. However, we can give a range of values in which the overall disagreement lies, as a function of the swap disagreement.

Given our result in Lemma 4, we now focus on the case when \(0 < |Y'| \leq |Y|\). The overall disagreement due to \(V_i\) lies in the range \((2^{Y}, 2^{2|Y| - 1})\). This lets us state the following lemma.

**Lemma 5** Consider a pair of CP-nets \(N_1\) and \(N_2\) with unique topological ordering and an attribute \(V_i\). If the swap disagreement between the two CP-nets for a context \(\beta \in Parent(N_1, V_i) \cup Parent(N_2, V_i)\) is \(w(\beta)\), then the overall disagreement due to \(\beta\) lies in the range \((2^{w(\beta)}, 2^{(w(\beta))^2 - 1})\).

We now give a few example CP-nets with attributes at both extremes of the range given in Lemma 5.

### 4.1.3 Examples

Consider a CP-net \(N\) with \(n\) attributes and a unique topological ordering. We label the attributes \(V_1, V_2, V_3, ..., V_{n-1}, V_n\) in their topological order, as shown in Figure 4.1. For each attribute \(V_i\) for \(2 \leq i \leq n\), \(Parent(N, V_i) = V_{i-1}\) and \(Parent(N, V_1) = \emptyset\). Thus the CP-net graph is a path of attributes, with \(V_1\) at the root and \(V_n\) at the end. Each attribute has at most one parent, but any two attributes differ in the number of ancestors. We use different
attributes in this CP-net to show how the relation between swap and overall disagreement changes based on the number of ancestors for the attribute. For the rest of this discussion, we assume we have a pair of CP-nets with the dependency graph mentioned above, but different CPTs. Below, we use the notation $Y_i = V \setminus (\{V_j\} \cup \text{Parent}(V_j))$ and use $Y' = \{V_j|1 \leq j \leq i-2\}$ to denote the set of attributes in $Y_i$ that are more important than $V_i$.

Suppose attribute $V_2$ has at least one rule on which the two CP-nets disagree. $|Y_2| = n - 2$. Further, none of the attributes in $Y_2$ are ancestors of $V_2$, so $|Y'_2| = 0$. Thus, for a swap disagreement of $2^{|Y_2|}$, we have an overall disagreement of $2^{|Y_2|}$, as mentioned in Lemma 4.

Now if attribute $V_3$ has a disagreement, $|Y_3| = n - 2$. However, $V_1 \in Y_3$ is also an ancestor of $V_3$. In fact, it is the only ancestor of $V_3$ other than its parent. Thus, $V_1 \in Y'_3$ and $|Y'_3| = 1$. In this case, for a swap disagreement of $2^{|Y_3|}$, the overall disagreement is $2^{|Y_3|}-1$. This is also the highest possible overall disagreement due to the difference in $CPT(N, V_3)$ when $|Y'_3| \neq 0$.

Intuitively, the further down the topological ordering we go, the more an-
cestors $V_i$ has. The most extreme case is with $V_n$, with parent $V_{n-1}$. All attributes from $V_1$ to $V_{n-2}$ are ancestors of $V_n$. Thus, $|Y_n| = |Y_n'| = n - 2$. In this case the swap disagreement and overall disagreement are both $2^{\|Y_n\|}$. This is because once the parent and all attributes in $Y_n'$ have the same value, we are left with only the swaps. Thus, for this case, the swap and overall disagreement due to $CPT(N,V_n)$ are exactly the same. This is also the lowest possible value of overall disagreement, given the swap disagreement.

From Observation 2, we know that the swap disagreement of a pair of CP-nets is the sum of the weights of the CPT rules on which they disagree. A similar approach does not work for overall disagreement. If we add the number of overall outcome pairs on which the two CP-nets disagree due to all rules $r_i \in R$, some outcome pairs will be counted more than once. Simply summing the number of disagreements for each rule gives us an upper bound on the overall disagreement as a function of the swap disagreement. We give this relation below. Recall here that, for two CP-nets $N_1$ and $N_2$, the set $R(N_1,N_2)$ contains all contexts $\beta \in Parent(N_1,V_i) \cup Parent(N_2,V_i)$, for any $V_i \in V$, under which $N_1$ and $N_2$ provide a different preference ordering on $V_i$.

**Observation 3** *The overall disagreement between a pair of CP-nets $N_1$ and $N_2$ is bounded from above by the sum of squares of the swap disagreement for each context in $R(N_1,N_2)$ on which the two CP-nets disagree, i.e.,*

$$\Delta_{\text{overall}}(N,N') \leq \sum_{\beta \in R(N_1,N_2)} (w(\beta))^2$$

*where, for each $\beta \in R(N_1,N_2)$, $w(\beta)$ denotes the weight of the*
We conclude this section by pointing out that for CP-nets with a unique topological ordering, there is one special case where the minimum value for swap disagreement also ensures the minimum value for the overall disagreement. However, in other cases, two CP-nets with the swap disagreement may have different overall disagreements. Therefore, we need a procedure which, among multiple summary CP-nets with the minimum swap disagreement, can choose the one that also has the minimum overall disagreement.

In the next two sections, we present the relationship between swap and overall disagreement when all the input CP-nets are separable. This is particularly interesting since the two algorithms presented in this thesis focus on the case of separable CP-nets. First, we look at the case when \( N_s \) is also separable.

4.2 Separable CP-nets with Separable Summary CP-net

In this section, we will describe the relationship between swap disagreement and overall disagreement for a pair of separable CP-nets. In particular, we want to prove the following theorem.

**Theorem 2** Suppose an input CP-net \( N \) and the summary CP-net \( N_s \) are both separable and have a swap disagreement value of \( m \cdot 2^{n-1} \), \( m \leq n \). Then
the overall disagreement for $N$ and $N_s$ is

$$\sum_{l=1}^{m} \sum_{j=1}^{l} \binom{m}{j} \cdot \binom{n-m}{l-j} \cdot 2^{n-l}$$

Theorem 2 tells us that given the swap disagreement over a pair of CP-nets $N$ and $N_s$, it is possible to compute the overall disagreement between $N$ and $N_s$ without enumerating all outcome pairs. It also tells us that a lower swap disagreement between a pair of CP-nets can lead to a lower overall disagreement. Lastly, this means that the overall disagreement measure is symmetric when restricted to separables, since swap disagreement is symmetric. We now present some additional results, working up to a proof for Theorem 2.

We begin by pointing out that for separable CP-nets, several outcome pairs are going to be incomparable. Since there are no parent-child relationships in such CP-nets, all attributes are interpreted as equally important. Thus, whenever an outcome pair calls for a trade-off between two attributes, separable CP-nets are unable to order this outcome pair. This is important because, given how we define disagreement, whenever $N$ finds an outcome pair incomparable, this counts as an agreement regardless of what $N_s$ does. This makes computing overall disagreement for separable CP-nets significantly easier.

Before we proceed with details, we want to define a few terms and deal with two special cases.

**Definition 6** For a pair of arbitrary CP-nets $N$ and $N_s$ and a given attribute $V_i$, the CPTs for $V_i$ may agree on all rules, disagree on all rules, or agree on
some rules and disagree on the others. We define the degree of disagreement between $N$ and $N_s$ for attribute $V_i$, denoted by $z(V_i)$, as the proportion of parent contexts for which the two CP-nets disagree for a given attribute $V_i$. Symmetrically, the degree of agreement between $N$ and $N_s$, denoted by $z'(V_i)$, is the proportion of parent contexts for which the two CP-nets agree for $V_i$.

Clearly when $N$ and $N_s$ are both separable, $z(V_i) \in \{0, 1\}$. For all CP-nets in general, $z(V_i) + z'(V_i) = 1$.

If the relative swap disagreement between a pair of CP-nets is 0 or 1, then the relative overall disagreement between them is also 0 or 1 respectively, because the preference order induced by a CP-net is the transitive closure of its preference order over the swaps. Thus, complete disagreement over the swaps translates to complete disagreement over all outcome pairs, and symmetrically for complete agreement.

We now focus on the general case where the relative swap disagreement lies in the interval $(0, 1)$.

Essentially, we want to count the number of outcome pairs on which $N$ and $N_s$ disagree. As mentioned above, this can be done by focusing on just the outcome pairs comparable by $N$. In order to do this, we partition the outcome pairs by the number of attributes on which the two outcomes disagree.

We use $X^{(l)}$ to denote the set of all outcome pairs $(o_1, o_2) \in X$ such that $o_1$ and $o_2$ differ on exactly $l$ attributes. $X^{(1)}$ then denotes the set of all swaps, i.e., $X^{(1)} = X_{\text{swap}}$. We first state results about the size of the set $X^{(l)}$, and some interesting subsets.
Lemma 6  Let $X^{(l)}$ be the set of all outcome pairs $(o_1, o_2)$ such that $o_1$ and $o_2$ differ on exactly $l$ attributes. Then

$$|X^{(l)}| = \binom{n}{l} \cdot 2^{n-1}$$

Proof. Out of the $n$ attributes, we can choose which $l$ are assigned different values in $\binom{n}{l}$ ways. The remaining $n - l$ attributes have the same value in both outcomes, which can be done in $2^{n-l}$ ways. Lastly, the $l$ attributes chosen to differ can be assigned $2^l$ possible combinations of values in $o_1$. From this, we can assign the corresponding values in $o_2$, which must differ from those on $o_1$. This can be done in $2^l$ ways, however it counts each outcome pair twice. Thus, we can assign values to the $l$ chosen attributes in $2^{l-1}$ ways. This gives a total of $\binom{n}{l} \cdot 2^{n-l} \cdot 2^{l-1}$ outcome pairs, simplifying which gives us $\binom{n}{l} \cdot 2^{n-1}$. \[\square\]

Of the outcome pairs in $X^{(l)}$, the comparable pairs are those that do not need trade-off over two or more attributes. This is only possible when, without loss of generality, all $l$ swapped attributes have their preferred value assigned in $o_1$. This leads to the following lemma.

Lemma 7  Let $X_c^{(l)}$ denote the set of comparable outcome pairs that differ on $l$ attributes. Then

$$|X_c^{(l)}| = \binom{n}{l} \cdot 2^{n-l}$$

We want to count the number of outcome pairs in $X_c^{(l)}$ on which $N$ and $N_s$ disagree. The two CP-nets disagree on an outcome pair only when at least one of the $l$ swapped attributes in $o_1$ and $o_2$ also has different CPTs in the
two CP-nets. Thus, we count how many of the comparable pairs differ on one of the attributes for which \( N \) and \( N_s \) have different CPT rules. We provide a formal statement for this problem.

**Problem Statement 2** Let \( N \) and \( N_s \) be a pair of separable CP-nets, where \( V \) denotes the set of all attributes and \( V' \) denotes the set of attributes with different CPTs in \( N \) and \( N_s \), \( V' \subseteq V \), \(|V| = n\) and \(|V'| = m\). For each \( V_i \in V' \), \( z(V_i) \in \{0, 1\} \). We want to count the number of pairs in \( X_c^{(l)} \), for \( 2 \leq l \leq n \), that differ on at least one attribute in \( V' \).

**Lemma 8** Using the notation of Problem Statement 2, the number of pairs in \( X_c^{(l)} \), for \( 2 \leq l \leq n \), that differ on at least one of the \( m \) attributes in \( V' \) is given by

\[
\sum_{j=1}^{l} \binom{m}{j} \cdot \binom{n-m}{l-j} \cdot 2^{n-l}
\]

**Proof.** Any arbitrary outcome pair in \( X_c^{(l)} \) differs on exactly \( l \) attributes. We want to count the number of outcomes where at least one of these \( l \) attributes belongs to one of the \( m \) attributes in \( V' \), the other swapped attributes belong to one of the \( n - m \) attributes in \( V \setminus V' \), and all other attributes are assigned identical values in \( o_1 \) and \( o_2 \). This is exactly what our expression counts. \( \square \)

Lemma 8 gives us, for a given \( l \), the number of outcome pairs that differ on \( l \) attributes and on which \( N \) and \( N_s \) disagree. We can now state the following lemma for the number of all outcome pairs that differ on at least one attribute in \( V' \).

**Lemma 9** Using the notation of Problem Statement 2, the number of all out-
come pairs that differ on at least one of the \( m \) attributes in \( V' \) is given by

\[
\sum_{l=1}^{n} \sum_{j=1}^{l} \binom{m}{j} \cdot \binom{n-m}{l-j} \cdot 2^{n-l}
\]

To simplify the proof of Theorem 2, we first prove one further lemma.

**Lemma 10** For a pair of separable CP-nets \( N \) and \( N_s \) that differ on exactly \( m \) attributes, the swap disagreement between \( N \) and \( N_s \) is \( m \cdot 2^{n-1} \).

**Proof.** Since \( N \) and \( N_s \) are both separable, we have \( z(V_i) = 1 \) if \( CPT(N, V_i) \neq CPT(N_s, V_i) \) and \( z(V_i) = 0 \) otherwise.

If \( |V'| = m \), then the swaps of the \( m \) attributes \( V_i \) for which \( z(V_i) = 1 \) are exactly the swaps on which \( N \) and \( N_s \) disagree. Further, \( N \) and \( N_s \) do not disagree on any of the swaps for the \( n-m \) attributes for which \( z(V_i) = 0 \). Since each \( V_i \in V \) has \( 2^{n-1} \) swaps, this gives us a swap disagreement of \( m \cdot 2^{n-1} \).

Combining the results of Lemma 9 and 10 completes the proof of Theorem 2. This establishes that there is a correspondence between the swap disagreement and the overall disagreement between a pair of CP-nets. In particular, given the swap disagreement \( m \cdot 2^{n-1} \) between \( N \) and \( N_s \), we can compute the overall disagreement between \( N \) and \( N_s \). We can also see that the overall disagreement increases with increasing values of \( m \).
4.3 Separable CP-nets With Arbitrary Summary CP-net

In this section, we focus on the relationship between swap and overall disagreement when \( N_s \) is not a separable CP-net. We begin by giving an upper bound on the overall disagreement between \( N \) and \( N_s \), based on our results in the previous section.

As we mentioned above, Theorem 2, which gives a one-to-one correspondence between swap and overall disagreement, only applies when both \( N \) and \( N_s \) are separable. This is because we assume that any pair \((o_1, o_2)\) assigning different values to an attribute in \( V' \) will be ordered differently by \( N \) and \( N_s \). This assumption is valid for separable CP-nets, where an attribute \( V_i \) having different CPTs necessarily means \( z(V_i) = 1 \). However, when \( N_s \) is not separable, \( z(V_i) \) lies in the interval \((0, 1)\). Thus, in general, the number of pairs ordered differently by \( N \) and \( N_s \) is some fraction of the expression in Theorem 2. This gives us the following theorem.

**Theorem 3** Suppose the input CP-net \( N \) is separable but the summary CP-net \( N_s \) is not, and exactly \( m \) attributes have different CPTs in \( N \) and \( N_s \). Then the overall disagreement between \( N \) and \( N_s \) is bounded from above by

\[
\sum_{l=1}^{n} \sum_{j=1}^{l} \binom{m}{j} \cdot \binom{n-m}{l-j} \cdot 2^{n-l}
\]

In the rest of this section, we will show a few examples of how the overall
disagreement between a pair of CP-nets can be calculated, given their swap
disagreement. This will help establish that there is a connection between the
swap and overall disagreement. In Chapter 5, following our algorithms, we
will show how we can guarantee that our non-separable summary CP-net \( N_s \)
will always minimize the overall disagreement with the set of input CP-nets.

Recall that we use \( V' \) to denote the set of attributes for which \( N \) and \( N_s \)
have different CPTs. Based on the intuition presented in Section 4.2, outcome
pairs that differ only on attributes that are not in \( V' \) will be ordered identically
by \( N \) and \( N_s \). When \( N_s \) is not separable, some proportion of the outcome pairs
that differ on attributes in \( V' \) will also be ordered identically. This proportion
depends on the degree of difference for the attribute or attributes in \( V' \) that
are swapped attributes in the outcome pair. We can compute this proportion
by computing, for outcome pairs that differ on at least one attribute in \( V' \),
the proportion of such pairs that correspond to matching CPT rules in both
\( N \) and \( N_s \). From this we can easily obtain the proportion of such pairs that
correspond to at least one CPT rule on which \( N \) and \( N_s \) disagree. We now
give a few examples of how overall disagreement can be computed given the
swap disagreement.

\textbf{Example 4.3.1}

Consider the CP-nets \( N \) and \( N_s \), as shown in Figure 4.2 and Figure 4.3.
\( CPT(N, A) = CPT(N_s, A) = \{a' > a\} \) and \( CPT(N, C) = CPT(N_s, C) = \{c' > c\} \).
\( CPT(N, B) = \{b' > b\} \) and \( CPT(N, D) = \{d' > d\}. \) \( Parent(N_s, B) \)
= \{A\} and \( CPT(N_s, B) = \{a': b' > b, a : b > b'\}. \) Similarly, \( Parent(N_s, D) \)
= \{C\} and \( CPT(N_s, D) = \{c' : d' > d, c : d > d'\}. \)
Figure 4.2: The CP-net $N$ discussed in Example 4.3.1

Figure 4.3: The CP-net $N_s$ discussed in Example 4.3.1

Figure 4.4: The CP-net $N_s'$ discussed in Examples 4.3.2 and 4.3.3
For \( j = 1 \), we consider the outcome pairs that have exactly one of \( B \) or \( D \) as the swapped attribute – the swaps of \( B \) and \( D \). Each has 8 possible swaps, and \( \frac{1}{2} \) of them disagree because \( z(B) = z(D) = \frac{1}{2} \). Thus, the disagreement due to each of \( B \) and \( D \) is 4, for a total of 8.

Next we compute the disagreement for outcome pairs differing on both \( B \) and \( D \). We see \( \text{Parent}(N_s, B) \cap \text{Parent}(N_s, D) = \emptyset \). We essentially want to find the proportion of outcome pairs with fixed contexts for \( \text{Parent}(N_s, B) \) and \( \text{Parent}(N_s, D) \) and with \( B \) and \( D \) swapped, that are ordered differently by \( N \) and \( N_s \). One way to do this is to find the proportion of such outcome pairs that are ordered identically by \( N \) and \( N_s \). This happens when the parent contexts correspond to a CPT rule on which \( N \) and \( N_s \) agree. In our case this proportion is \( \frac{1}{4} \), since \( z'(B) = z'(D) = \frac{1}{2} \), and the outcome pairs on which \( N \) and \( N_s \) agree are the ones where \( A = a' \) and \( C = c' \), and \( B \) and \( D \) are swapped \(^1\). From this, we know that the proportion of outcome pairs on which \( N \) and \( N_s \) disagree is \( \frac{3}{4} \). There are four outcome pairs with fixed contexts for \( \text{Parent}(N_s, B) \) and \( \text{Parent}(N_s, D) \) and with \( B \) and \( D \) swapped. These are the outcome pairs whose order depends on \( \text{CPT}(N_s, B) \) and \( \text{CPT}(N_s, D) \). Multiplying the number of such outcome pairs with the degree of disagreement gives us the number of outcome pairs on which \( N \) and \( N_s \) disagree due to the CPTs of \( B \) and \( D \), which is 3. Adding this to the case for \( j = 1 \), we get a total overall disagreement of 11.

Example 4.3.1 is overly simple to an extent because the two attributes in \( V' \) have disjoint parent sets. In the next example we show how an overlap in

\(^1\text{Recall that } z'(V_i) = 1 - z(V_i).\)
the parent set affects our computation.

Example 4.3.2 \( N \) is the same as in the previous example, but the summary CP-net we consider now is \( N'_s \), shown in Figure 4.4 \( CPT(N'_s, A), CPT(N'_s, B) \), and \( CPT(N'_s, C) \) are identical to the corresponding CPTs in \( N_s \) in Example 4.3.1. The only difference is that we now have \( Parent(N'_s, D) = \{A, C\} \), with \( CPT(N'_s, D) = \{a'c' : d' \succ d, a'c : d \succ d', ac' : d' \succ d, ac : d' \succ d\} \). Thus, \( V' = \{B, D\} \), \( z(B) = \frac{1}{2} \), and \( z(D) = \frac{3}{4} \).

First we consider the swaps of \( B \) and \( D \), each of which has 8 possible swaps. For \( B \), \( \frac{1}{2} \) of them disagree. Thus, the number of swaps of \( B \) ordered differently is four. For \( D \), \( \frac{3}{4} \) of them disagree. Thus, the disagreement over swaps of \( D \) is six. This gives us a total of ten swaps ordered differently.

Next we compute the disagreement for outcome pairs that differ on both \( B \) and \( D \). However, there is a difference. \( Parent(N'_s, B) \cap Parent(N'_s, D) = \{A\} \). Clearly, \( z'(B) = \frac{1}{2} \). On casual inspection, we might conclude that \( z'(D) = \frac{1}{4} \). However, \( Parent(N'_s, B) \cap Parent(N'_s, D) = \{A\} \), so we only consider the proportion of rules in \( CPT(D) \) that assign \( A = a' \), since that is the only value of \( A \) for which the CPTs of both \( B \) and \( D \) have rules on which \( N \) and \( N_s \) agree. This proportion is given by \( z'(D|B) = \frac{1}{2} \). The proportion of outcomes on which \( N \) and \( N_s \) agree is \( \frac{1}{4} \), and thus the proportion on which they disagree is \( \frac{3}{4} \). There are four outcome pairs with identical contexts for \( A \) and \( C \) and \( B \) and \( D \) as swapped attributes. Three of these pairs are ordered differently by \( N_i \) and \( N_s \), and the overall disagreement is thirteen.

We point out here that in our second example, it is clear that the overall
disagreement is no longer a function of just the swap disagreement. Clearly, when a pair of attributes have overlapping parent sets, we need to know more than just the swap disagreement for either of these attributes. We will now present our third example, which further emphasizes the fact that a pair of CP-nets with the same swap disagreement could have different values for the overall agreement.

Example 4.3.3 We present one last example to highlight an important point. We are dealing with the same $N$ as in the previous two examples but we use a different summary CP-net, denoted $N''_s$. $N''_s$ is identical in structure with $N'_s$ from Example 4.3.2, but $CPT(N''_s, D) = \{a'c' : d \succ d', a'c : d \succ d', ac' : d \succ d', ac : d' \succ d\}$. $z(D)$ is still $\frac{3}{4}$, so we might expect this case to be identical to Example 4.3.2. However, it is not, and analyzing why lets us develop a strategy for constructing summary CP-nets.

The problem is that the rule in $CPT(N''_s, D)$ that matches $CPT(N, D)$ assigns $A = a$. But the only rule in $CPT(N''_s, B)$ that matches $CPT(N, B)$ assigns $A = a'$. Thus, there is no way for us to construct an outcome pair with $B$ and $D$ as swapped attributes such that the values assigned correspond to matching rules for both CPTs. Thus, the proportion of outcome pairs ordered differently is 1.

As seen from these examples, for a given attribute, minimizing the swap disagreement is not sufficient to also minimize the overall disagreement due to said attribute. For pairs of attributes with an overlapping parent set, we need to build their CPTs with more care to ensure a lower disagreement when
both attributes are swapped. In Section 5.3, we will present our algorithm to construct an $N_s$ that minimizes swap disagreement with the input set for all attributes. In Section 5.4, we will show how we can always construct $N_s$ so as to also minimize the overall disagreement.
Chapter 5

Minimizing Disagreement Measures

In this chapter, we propose a further refinement to our problem statement. Based on this refined problem statement, we discuss some potential distance functions that could be used as our objective function.

5.1 Distance Measures for Sets of Separable CP-nets

In Chapter 3, we proposed an algorithm that allows us to efficiently compute the swap disagreement between a pair of CP-nets. In Chapter 4, we established that there is a relationship between swap and overall disagreement for CP-nets with some structural properties. Further, we saw that, under cer-
tain circumstances, there is a way to compute the overall disagreement given
the swap disagreement. Given these observations, we now modify Problem
Statement \[ \text{Problem Statement 3} \]

Given a set of CP-nets \( T = \{ N_1, N_2, \ldots, N_t \} \) as input, find a summary CP-net \( N_s \) that minimizes the swap disagreement of \( N_s \) with each \( N_i \in T \).

This still does not answer the question of how to minimize swap disagree-
ment with an entire set of input CP-nets. What we need is a distance function
over the set of input CP-nets and the summary CP-net, based on each individual pairwise agreement. In what follows, we discuss some potential distance
functions.

For any arbitrary pair of CP-nets, the swap disagreement \( d \) lies in the
interval \([0, 1]\). \( d = 0 \) indicates no semantic difference between two CP-nets.
Ideally, for all input CP-nets, the disagreement with \( N_s \) would be 0. Of course
this is not attainable for any but the most trivial inputs. However, we can use
this notion to define a distance function to use as our objective function.

For each input CP-net \( N_i \), let \( d_i \) denote the swap disagreement between
\( N_i \) and \( N_s \). The set of swap disagreements with each input CP-net is then
represented by the vector \( D = (d_1, d_2, \ldots, d_t) \). As mentioned above, the ideal
vector would have all \( d_i = 0 \). While this is not attainable except for some
trivial input sets, we can use the vector \( D \) to define a single parameter to
optimize. The ideal solution would give a vector distance of 0 between \( D \)
and the ideal vector of all 0s. Extending this notion, we can use the distance
between these two vectors as the quantity to be minimized.

5.1.1 Potential Distance Functions

Before we discuss our options for distance function, we decompose the pairwise swap disagreement \( d_i \) further to express it in terms of the pairwise swap disagreement for each attribute. \( d_{ij} \) denotes the pairwise swap disagreement between CP-nets \( N_i \) and \( N_s \) for only the swaps ordered by attribute \( V_j \). This decomposition helps us present different options for our distance function.

5.1.2 Absolute Distance

The most obvious choice for our distance function would be to use the sum of pairwise swap disagreements. We define this distance function as follows.

\[
\sum_{i=1}^{t} \sum_{j=1}^{n} d_{ij}
\]

This distance function is easy to compute. It also suggests that to minimize distance we might want to try and minimize the pairwise swap disagreement for each attribute individually. Absolute distance is amenable to such an approach, since the outer sum adds up disagreement contributions separated for each attribute individually. As we will see later, the optimal solution using this distance function will always summarize an input set of separable CP-nets with a separable CP-net.
5.1.3 Sum of Squares

A second option for distance function could be to use the sum of squares of all the pairwise disagreement values $d_{ij}$. This is defined as follows.

$$\sum_{i=1}^{t} \sum_{j=1}^{n} d_{ij}^2$$

This function is slightly harder to handle than the absolute distance function above. At the same time, it is still easy to compute the total disagreement for only a given attribute. Thus it is still easy to take the approach of minimizing the disagreement for each attribute individually. However, using this distance function often means that $N_s$ is no longer separable even if all input CP-nets are. In fact, as we will see later, in this case $N_s$ is often cyclic even if all the inputs are acyclic CP-nets.

5.1.4 Square of Sums

A third option would be to compute

$$\sum_{i=1}^{t} (\sum_{j=1}^{n} d_{ij})^2$$

A significant drawback of this function is that it is more complicated to compute the total disagreement contributed by only a given attribute. As mentioned already, we want to be able to do this in order to minimize each
\( d_{ij} \) for a given individual \( j \) across all input CP-nets. We therefore do not use this third distance function.

For the rest of this chapter, we present algorithms that use the first two distance functions. We also revise our general problem statement. We will denote the chosen distance function by \( \Delta(T, N_s) \), where \( T \) is the set of inputs and \( N_s \) is a summary CP-net. \( \Delta(T, N_s) \) denotes the distance attained by a CP-net \( N_s \) given a set of inputs \( T \). Our revised problem statement is given below.

**Problem Statement 4** Given a set of CP-nets \( T = \{N_1, N_2, ..., N_t\} \) as input, find a summary CP-net \( N_s \) that minimizes the distance between the vector of all disagreement scores \( D \), and the ideal vector of all 0s. Formally:

\[
\text{minimize} \quad \Delta(T, N_s)
\]

We have now established our problem statement, and identified two interesting distance functions as potential objective functions. We will now propose algorithms for each distance function. We also include discussions of correctness and complexity for both algorithms.

CP-nets can be classified according to the structural properties of the underlying dependency graph. Some interesting classes are

- Separable CP-nets: the dependency graph has no edges.
- Tree-structured CP-nets: the dependency graph is a tree.
• Directed acyclic CP-nets: the dependency graph is a directed acyclic graph.

• Polytrees: the dependency graph is a polytree.

• Singly-connected DAG: all pairs of vertices in the dependency graph have at most one directed path between each other.

• Max $\Delta$-connected DAG: all pairs of vertices in the dependency graph have at most $\Delta$ directed paths between each other.

Finding the optimal summary network for the class of CP-nets in general is a difficult problem, largely due to the fact that an attribute can have an arbitrary parent set with any size and any subset of attributes. The size of and the elements in the parent sets affect the CP-net size, and and also the swap disagreement values, substantially. We therefore take an incremental approach. In this thesis, we present an algorithm to solve our problem under the assumption that all the input CP-nets are separable. This means that there are no edges in the underlying directed graph, and the preferences for any attribute are not dependent on any of the other attributes. This in turn means that all attributes have an empty parent set and one of two possible CPTs. We leverage these properties in our proposed algorithms.

While this restriction to separable input CP-nets is rather extreme, we begin our study here with the hope that we can find provably optimal algorithms for this limited case. These results can then serve as a basis for developing algorithms and theoretical results for more general CP-nets.
In the remainder of this chapter, we first discuss an algorithm for separable CP-nets using absolute distance as the objective function. Following this, we present an algorithm using the sum of squares distance function. We use a running example to show how the second distance function can help achieve a lower disagreement score in certain cases. We also provide proof of optimality and complexity analysis for each algorithm.

5.2 Absolute Distance

In this section, we present our algorithm to find the optimal summary CP-net, when using absolute distance over swap disagreement as our objective function.

5.2.1 The Algorithm

A high level pseudocode is shown in Algorithm 2. Given a set of separable CP-nets $T$ as input, Algorithm 2 constructs a summary CP-net $N_s$ that minimizes absolute distance. We assume all input CP-nets and the output have the same set of attributes $V$, $|V| = n$. We begin with $N_s$ as the empty CP-net and iteratively find the most common CPT rule for each attribute. If a majority CPT rule exists, we assign it to the relevant attribute in $N_s$. If no majority exists, we assign an arbitrary CPT rule. At the end of this process, $N_s$ is the summary network that minimizes the absolute distance with the input set $T$.

We walk through Algorithm 2 using the following example. The input set
Algorithm 2: An algorithm to find an optimal summary for a set of separable CP-nets using absolute distance

1 MinimizeAbsolute
   Input: A set \( T = \{N_1, N_2, \ldots, N_t\} \) of separable CP-nets
   Output: A summary CP-net \( N_s \)
2 Set summary CP-net \( N_s \) to a separable CP-net with empty CPTs
3 for each attribute \( V_i \in V \) do
   4 if \( v_i \succ v'_i \) occurs in strictly more than \( t/2 \) of the CP-nets in \( T \) then
      5 Assign \( v_i \succ v'_i \) to \( CPT(N_s, V_i) \)
   6 else
      7 Assign \( v'_i \succ v_i \) to \( CPT(N_s, V_i) \)
   8 end
4 end

contains four CP-nets, each with \( V = \{V_1, V_2, V_3\} \). We list the input CP-nets below in terms of their CPTs.

- \( CPT(N_1, V_1) = \{v'_1 \succ v_1\}, CPT(N_1, V_2) = \{v'_2 \succ v_2\}, CPT(N_1, V_3) = \{v'_3 \succ v_3\} \)
- \( CPT(N_2, V_1) = \{v'_1 \succ v_1\}, CPT(N_2, V_2) = \{v'_2 \succ v_2\}, CPT(N_2, V_3) = \{v_3 \succ v'_3\} \)
- \( CPT(N_3, V_1) = \{v'_1 \succ v_1\}, CPT(N_3, V_2) = \{v_2 \succ v'_2\}, CPT(N_3, V_3) = \{v'_3 \succ v_3\} \)
- \( CPT(N_4, V_1) = \{v_1 \succ v'_1\}, CPT(N_4, V_2) = \{v'_2 \succ v_2\}, CPT(N_4, V_3) = \{v_3 \succ v'_3\} \)

As we can see, \( CPT(V_1) \) and \( CPT(V_2) \) have the same CPT three times out of four. These are \( v'_1 \succ v_1 \) and \( v'_2 \succ v_2 \) for \( V_2 \). We assign these CPTs to
Table 5.1: Pairwise disagreement values for $N_s$ with absolute distance. The rows represent input CP-nets and the columns represent attributes.

<table>
<thead>
<tr>
<th></th>
<th>$j = 1$</th>
<th>$j = 2$</th>
<th>$j = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i = 1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$i = 2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$i = 3$</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$i = 4$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$N_s, V_1$ and $N_s, V_2$ respectively. For $V_3$, 2 of the inputs have $v_3' \succ v_3$ and the other 2 have $v_3 \succ v_3'$. Based on Algorithm 2, we assign the CPT $v_3' \succ v_3$. As we see shortly, we obtain the same absolute distance even if we assign $v_3 \succ v_3'$ instead. Further, this is also the lowest absolute distance we can obtain. We will prove that Algorithm 2 is optimal in the following section.

### 5.2.2 Proof of Optimality

In this section, we show that Algorithm 2 always finds an optimal summary CP-net. Before presenting our proof, we give a little more detail about how absolute distance is calculated for each input CP-net. This helps us present our proof of optimality.

We begin by describing $N_s$ for the preceding example.

$$CPT(N_s, V_1) = \{v_1' \succ v_1\}, CPT(N_s, V_2) = \{v_2' \succ v_2\}, CPT(N_s, V_3) = \{v_3 \succ v_3'\}$$

The corresponding $d_{ij}$ entries are shown in Table 5.1. For separable CP-nets, the value of $d_{ij}$ is either 0 or 1, denoting the proportion of all swaps of
attribute $V_j$ on which the two CP-nets disagree. Each entry in our table is thus either 0 or 1, depending on whether $N_s$ and $N_i$ have the same CPT for $V_j$. The absolute distance $D$ is 4, obtained by summing all $d_{ij}$ entries.

From this table, we can also see why setting $CPT(N_s, V_3)$ to $\{v'_3 \succ v_3\}$ does not change the absolute distance. Doing so simply flips the 0 and 1 entries in the third column and maintains the same sum. Thus, either CPT attains the same absolute distance. We choose the rule $v'_i \succ v_i$ for the sake of simpler pseudocode. This also demonstrates that for a given input instance, there may be more than one optimal summary CP-net. We now prove that Algorithm 2 always finds one of these optimal CP-nets using absolute distance as our distance function of choice.

**Lemma 11** Algorithm 2 finds an optimal summary CP-net $N_s$ in terms of absolute distance computed over swap disagreement values, given a set $T$ of separable CP-nets.

**Proof.** Suppose that of the $t$ input CP-nets, and for attribute $V_j$, $k$ CP-nets have the $CPT(V_j) = \{v_j \succ v'_j\}$ and $t - k$ have the $CPT(V_j) = \{v'_j \succ v_j\}$. We choose an appropriate rule for $V_j$ depending on the relative sizes of $k$ and $t - k$. We explain how we choose the appropriate CPT rule.

Without loss of generality, we assume that for some choice of CPTs for $N_s$, $k$ of the CP-nets have error $z_1$ and $t - k$ have error $z_2$, where $z_1$ and $z_2$ are the proportion of all swaps of $V_j$ on which the $k$ or $t - k$ CP-nets disagree, respectively. Since there are just two possible CPTs for $V_j$, $z_1 + z_2 = 1$. The total distance, $D$, is given by
\[ D = k \cdot z_1 + (t - k) \cdot z_2 \]

Rewriting \( z_2 = 1 - z_1 \) we have

\[ D = k \cdot z_1 + (t - k) \cdot (1 - z_1) \]

This expression gives us the intuition on how to choose the appropriate CPT. If \( k > t - k \), we want to minimize \( z_1 \). As it happens, we can do this by choosing the majority CPT, the one that occurs in \( k \) of the inputs. This gives us \( z_1 = 0 \), the minimum possible value. Thus we know we cannot do any better than this. The situation is symmetric when \( t - k > k \). In this case we aim for \( 1 - z_1 = 0 \) or equivalently, \( z_1 = 1 \). We obtain this by choosing the CPT that occurs in \( t - k \) of the inputs. Again, this is the maximum possible value of \( z_1 \) and we cannot do any better. Since in either case, we attain the optimal value of \( z_1 \) by choosing the majority CPT, adding more parents to the attribute would not help us improve the score.

Lastly, when \( k = t - k \), the expression for \( D \) simplifies to

\[ k \cdot z_1 + k \cdot (1 - z_1) \]

which yields the constant \( k \) when we cancel out the terms in \( z_1 \). Thus in this case the value remains constant regardless of which CPT we choose. It can also be seen that adding parents to \( V_j \) would not help either, since the final expression is constant and does not depend on \( z_1 \). Since any CPT rule
would attain the same distance value, we assign the CPT $v'_j \succ v'_j$. Since all possible CPTs yield the same distance value, we justify this choice because it is one of the two CPT rules that minimizes the size of $N_s$. This also shows that Algorithm 2 makes the best possible decision for each attribute, and obtains the optimal summary CP-net in terms of the absolute distance function computed based on swap disagreement values for the input CP-nets.

5.2.3 Complexity Analysis

We now establish the runtime complexity of Algorithm 2 using the following lemma.

**Lemma 12** Algorithm 2 runs in time linear in the number of attributes $|V|$ and the size of the input set $|T|$.

**Proof.** The runtime of Algorithm 2 is dominated by the loop in lines 4-10. The loop runs once for each attribute $V_j$ for the input CP-net $N_i$ whose CPT in $V_j$ is largest. For each run, line 5 scans the CPT for this attribute across all input CP-nets, to see if there is a majority. This can be done in time $|T| \cdot |CPT(N_i, V_j)|$. Since we are dealing with separable CP-nets, $|CPT(N_i, V_j)| = 1$. So this can be checked in time $|T|$ for each run of the loop. Thus, the entire loop runs in time $|V| \cdot |T|$. This proves Lemma 12.
with the same distance value. Thus, Algorithm 2 always finds an optimal summary CP-net using absolute distance as an objective function. It also enables us to state the following theorem.

**Theorem 4** When using the absolute distance based on swap disagreement as objective function and limiting the input CP-nets to separable CP-nets, the summary CP-net that minimizes the objective function is also separable.

We conclude this section by stating the following theorem.

**Theorem 5** Algorithm 2 finds an optimal summary CP-net $N_s$, in terms of absolute distance based on swap disagreement, in linear time given a set $T$ of separable CP-nets.

**Proof.** Theorem 5 is a direct consequence of Lemmas 11 and 12.

We point out that the results on optimality we just proved in this section apply to absolute distance computed from the swap disagreement between a pair of CP-nets. As we saw in our proof for Theorem 2 when restricted to separable CP-nets, the overall disagreement is a function of the swap disagreement. In particular, this enables us to say that a lower value for swap disagreement leads to a lower value for overall disagreement. This allows us to state the following corollary.

**Corollary 1** Algorithm 2 finds an optimal summary CP-net $N_s$, in terms of absolute distance based on overall disagreement, in linear time given a set $T$ of separable CP-nets.
5.3 Sum of Squares Distance

We now propose an algorithm that finds an optimal summary network given a set of separable CP-nets as input, using sum of squares distance function based on swap disagreement as the objective function. We use the same example as in Section 4.1 to show how this algorithm works. Lastly, we prove that our algorithm does find an optimal summary CP-net in terms of sum of squares distance computed based on swap disagreement, and give an analysis of its runtime.

5.3.1 The Algorithm

We explain how Algorithm 3 works with our example from Section 4.1. We aim to find the optimal CPT for each attribute. For $V_1$, $k = 1$ of the inputs have the CPT $\{v_i \succ v'_i\}$ and $t - k = 3$ have the CPT $\{v'_i \succ v_i\}$. Thus the optimal value for $z_1$ is $\frac{3}{4}$. The situation with $V_2$ is similar. For both of these attributes, the optimal ratio is $\frac{3}{4}$, which is already reduced. Thus $p = 3$ and $q = 4$. $\log(q) = 2$ which is an integer, so we add two parents to $\text{Parent}(N_s, V_1)$ and $\text{Parent}(N_s, V_2)$. In particular, $\text{Parent}(N_s, V_1) = \{V_2, V_3\}$ and $\text{Parent}(N_s, V_2) = \{V_1, V_3\}$. Note that $\text{Parent}(N_s, V_1) \cap \text{Parent}(N_s, V_2) = \{V_3\}$.

The CPT of either attribute has to contain four CPT rules. Exactly three of these rules are of the form $v'_i \succ v_i$, and the remaining one rule is of the form $v_i \succ v'_i$. This ensures a disagreement of $\frac{3}{4}$ with the $k$ CP-nets that have rules of
Algorithm 3: An algorithm to find an optimal summary CP-net for a set of separable CP-nets using sum of squares distance

1 MinimizeSumSquares
   Input : A set $T = \{N_1, N_2, \ldots, N_t\}$ of separable CP-nets
   Output: A summary CP-net $N_s$
2 Set summary CP-net $N_s$ to a separable CP-net with empty CPTs
3 for each attribute $V_i \in V$ do
4     $k =$ number of occurrences of CPT rule $v_i \succ v'_i$
5     Reduce the fraction $\frac{t-k}{t}$ to $\frac{p}{q}$ where $p$ and $q$ have no common factors
6     if $\text{lg}(q)$ is an integer then
7         Add $\text{lg}(q)$ parents to $\text{Parent}(N_s, V_i)$
8         Build $\text{CPT}(N_s, V_i)$ with exactly $p$ rules $v'_i \succ v_i$
9     end
10    else
11        if $|\text{lg}(q) - \lfloor \text{lg}(q) \rfloor| < |\text{lg}(q) - \lceil \text{lg}(q) \rceil|$ then
12            numParents = $\lfloor \text{lg}(q) \rfloor$
13        end
14        else
15            numParents = $\lceil \text{lg}(q) \rceil$
16        end
17        Add numParents parents to $\text{Parent}(N_s, V_i)$
18        Build $\text{CPT}(N_s, V_i)$ with exactly $p$ rules $v'_i \succ v_i$
19        Complete $\text{CPT}(N_s, V_i)$ with $v_i \succ v'_i$ for the remaining instantiations of the parent set
20     end
21 end
Table 5.2: Pairwise disagreement values $d^2_{ij}$ for $N_s$ with sum of squares distance. The rows represent input CP-nets and the columns represent attributes.

<table>
<thead>
<tr>
<th></th>
<th>$j = 1$</th>
<th>$j = 2$</th>
<th>$j = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i = 1$</td>
<td>0.0625</td>
<td>0.0625</td>
<td>0.25</td>
</tr>
<tr>
<td>$i = 2$</td>
<td>0.0625</td>
<td>0.0625</td>
<td>0.25</td>
</tr>
<tr>
<td>$i = 3$</td>
<td>0.0625</td>
<td>0.5625</td>
<td>0.25</td>
</tr>
<tr>
<td>$i = 4$</td>
<td>0.5625</td>
<td>0.0625</td>
<td>0.25</td>
</tr>
</tbody>
</table>

the form $v_i \succ v'_i$. Note that we can build the CPT in multiple ways. In Section 5.4, we will talk about how we can always build the CPT that minimizes the overall disagreement.

For $V_3$, $k = t - k = 2$, and the optimal value for $z_1 = \frac{2}{4}$, reduced to $\frac{1}{2}$. Thus $p = 1$ and $q = 1$. $\lg(q) = 1$ which is an integer, so we add one parent to $Parent(N_s, V_2)$. Either of the other two attributes will suffice, and we choose $Parent(N_s, V_3) = \{V_2\}$. Note that this means $Parent(N_s, V_1) \cap Parent(N_s, V_3) = \{V_2\}$. Again, we will discuss the best way to construct the corresponding CPT so as to minimize overall disagreement.

Note that if $\lg(q)$ is not an integer, we choose the integer closest to $\lg(q)$ as the number of parents to add. The rest of the process is similar as we add the appropriate number of parents and fill the corresponding CPTs.

5.3.2 Proof of Optimality

In this section, we show that Algorithm 3 always finds an optimal summary CP-net in terms of sum of squares distance based on swap disagreement. Be-
fore presenting our proof, we give details on how sum of squares distance based on swap disagreement is calculated for each input CP-net. This helps us present our proof of optimality.

As in the previous section, we begin by showing how the sum of squares distance is calculated for our example. In line 2, we set $N_s$ to the empty separable CP-net. At the end of our algorithm, $N_s$ is the CP-net shown in Figure 5.1.

The corresponding $d_{ij}^2$ entries are shown in Table 5.2. Since $N_s$ is no longer a separable CP-net, swap disagreement is any number in the interval $[0, 1]$, still denoting the proportion of all swaps of attribute $V_j$ on which the two CP-nets disagree. Each entry in our table is $d_{ij}^2$, the square of the swap disagreement between $N_i$ and $N_s$ for attribute $V_j$. The absolute distance $D$ is 2.5, obtained by summing all entries. We now prove that this is the optimal distance according to the sum of squares function, and that our algorithm always finds the CP-net that minimizes the sum of squares.
Lemma 13 Algorithm 3 finds an optimal summary CP-net $N_s$, in terms of sum of squares based on swap disagreement, given a set $T$ of separable CP-nets.

Proof. Suppose that of the $t$ input CP-nets and for attribute $V_j$, $k$ CP-nets have the $CPT(V_j) = \{v_j \succ v_j'\}$ and $t - k$ have the $CPT(V_j) = \{v_j' \succ v_j\}$. Once we have chosen a CPT for $N_s$, $k$ of the CP-nets will have a disagreement of $z_1$ with $N_s$ and the other $t - k$ will have a disagreement of $z_2$, where $z_1$ and $z_2$ are the proportion of all swaps of $V_j$ on which the $k$ or $t - k$ CP-nets disagree, respectively. Given that there are two possible rules for each instantiation of a CPT in the input CP-nets, we once again have $z_1 + z_2 = 1$ for an attribute $V_j$. The total distance, $D$, is given by

$$D = k \cdot z_1^2 + (t - k) \cdot z_2^2$$

Rewriting $z_2 = 1 - z_1$ and simplifying,

$$D = t \cdot z_1^2 + (2k - 2t) \cdot z_1 + (t - k)$$

This is a second-order polynomial function, and the second derivative is always positive. Thus, this is a convex function. Further, since $z_1 + z_2 = 1$, this function is symmetric about the global minimum. Both of these facts will be important in our proof.

In order to minimize $D$, we take the derivative and solve for $z_1$. This gives us $z_1 = \frac{t-k}{t}$. This is the value for $z_1$ that minimizes $D$. We reduce this fraction.
to $\frac{p}{q}$, where $p$ and $q$ are co-prime. This enables us to obtain the optimal ratio by adding the smallest number of parents that still allow the optimal ratio.

If $\lg(q)$ is an integer, we simply add $\lg(q)$ parents and build the CPT. When building the CPT, we want to ensure that $k$ of the inputs have disagreement $z_1$ and the other $t - k$ have disagreement $z_2$. Since we know that $k$ of the inputs have the CPT $\{v_j \succ v'\}$, this can be achieved by ensuring that $\frac{p}{q}$ proportion of $\text{CPT}(N_k, V_j)$ disagree with those $k$ CP-nets. This is precisely what we do in our algorithm. If the attribute $V_j$ currently under consideration has parent attributes in common with one of the other attributes $V_k$, we also aim to maximize the proportion of rules on which both $V_j$ and $V_k$ have the majority ordering. This will be discussed in more detail in Section 5.4.

If $\lg(q)$ is not an integer, then clearly, we cannot add a fractional number of parents. Thus, the theoretical minimum is not actually attainable. We therefore try to attain the minimum feasible distance. First, we must get the integer closest to $\lg(q)$. This gives us the number of parents to add to $V_j$, following which we build the CPT in line with the same logic and constraints as mentioned above. Since the function for $D$ is symmetric about the minimum, we know that the integer closest to $\lg(q)$ gives us the minimum attainable value. This completes our proof.

### 5.3.3 Complexity Analysis

We now establish the runtime complexity of Algorithm 3 using the following lemma.
Lemma 14 Algorithm 3 runs in time linear in the number of attributes $|V|$ and the size of the input set $|T|$.

Proof. The loop starting at line 4 runs in time $O(|V|)$. Inside the loop, the lines that do not run in constant time are the ones that reduce the fraction $\frac{t-k}{t}$, and the ones that assign an appropriate number of parents and build the corresponding CPTs. We take a look at the complexity of these steps.

Adding $\log(q)$ parents can be done by updating an adjacency list in time $O(\log(q))$. We denote the number of parents to add for an attribute $V_j$ by $\text{Parent}(N_s, V_j)$. Adding the CPT can then be done in time $2^{\text{Parent}(N_s, V_j)}$. The upper bound on $|\text{Parent}(N_s, V_j)|$ is $\log(|T|)$. Lastly, line 3 can be executed in time logarithmic in $t - k$ and $t$ using Euclid’s Algorithm. Thus, the runtime of the loop is dominated by the lines that build the CPT for $V_j$. Thus, the worst-case runtime for Algorithm 3 is $O(|V| \cdot |T|)$.

This allows us to state the following theorem.

Theorem 6 Algorithm 3 finds an optimal summary CP-net $N_s$, in terms of sum of square distance based on swap disagreement, in linear time given a set $T$ of separable CP-nets.

Proof. Theorem 6 is a direct consequence of Lemmas 13 and 14.


## 5.4 Discussion on Algorithm 3

Now that we have presented Algorithm 3, we resume the discussion that we began in Section 4.3, where we saw that there is a relationship between swap and overall disagreement. We also saw that a pair of CP-nets with the same swap disagreement could still have different overall disagreements. In this section, we will talk about how we can build the CPTs in $N_s$ in a way so as to ensure that, for all attributes, $N_s$ has the optimal value for both swap and overall disagreement. To do this, we first define a few terms.

**Definition 7** Consider a pair $(N, N_s)$ of CP-nets and a pair $(V_i, V_j)$ of attributes. The conditional degree of agreement of $V_i$ with respect to $V_j$ is denoted $z'(V_i|V_j)$. It is defined as the proportion of contexts over $\text{Parent}(N, V_i) \cup \text{Parent}(N_s, V_i)$ for which $\text{CPT}(N, V_i)$ and $\text{CPT}(N_s, V_i)$ have the same preference ordering, given that any attribute in $\text{Parent}(N_s, V_i) \cap \text{Parent}(N_s, V_j)$ has been assigned values for which the corresponding rule in $\text{CPT}(N, V_j)$ and $\text{CPT}(N_s, V_j)$ is one on which $N$ and $N_s$ agree.

Consider a set of attributes $V = \{A, B, C, D\}$, with $V' = \{C, D\}$ being the subset of attributes that have different CPTs in $N$ and $N_s$. $\text{Parent}(N_s, C) = \{A, B\}$, $\text{Parent}(N_s, D) = \{A\}$, and $\text{Parent}(N_s, C) \cap \text{Parent}(N_s, D) = \{A\}$. $\text{CPT}(N, C) = \{c' \succ c\}$ and $\text{CPT}(N, D) = \{d' \succ d\}$.

Let $\text{CPT}(N_s, C) = \{a'b' : c' \succ c, a'b : c \succ c', ab' : c \succ c', ab : c \succ c'\}$ and $\text{CPT}(N_s, D) = \{a' : d' \succ d, a : d \succ d'\}$. Then the degree of agreement satisfies $z'(D) = 1 - z(D) = \frac{1}{2}$ and $z'(C) = \frac{1}{4}$. Suppose we know that an outcome
pair \((o_1, o_2)\) has values assigned such that it corresponds to a matching rule in \(CPT(N, C)\) and \(CPT(N_s, C)\). This must mean that \(A = a'\) since that is the only parent context for which \(CPT(N_s, C)\) and \(CPT(N, C)\) have a matching rule. Thus, \(z'(D|C) = 1\), since when \(A = a'\) the only appropriate rule in \(CPT(N_s, D)\) is one on which \(N\) and \(N_s\) agree.

This notion of conditional degree of disagreement allows us to compute a second term, defined next.

**Definition 8** Consider a pair \((N, N_s)\) of CP-nets and a pair \((V_i, V_j)\) of attributes.

We define the joint degree of agreement of \((V_i, V_j)\), denoted \(z'(V_i \cdot V_j)\), as the proportion of contexts over \(\text{Parent}(N, V_i) \cup \text{Parent}(N_s, V_i)\) for which \(N\) and \(N_s\) have the same preference ordering for the CPTs of both \(V_i\) and \(V_j\) respectively. The notion of joint degree of agreement can be extended naturally to more than two attributes.

In the example presented above, \(z'(C \cdot D) = \frac{1}{4}\). This can be seen from inspecting the two CPTs, or by calculating \(z'(C) \cdot z'(D|C)\). The joint degree of disagreement satisfies \(z(C \cdot D) = 1 - z'(C \cdot D)\).

We now proceed to proving our desired results.

**Theorem 7** For a given input set \(T\) of separable CP-nets, there exists a summary CP-net \(N_s\) which attains the minimum joint degree of disagreement for all pairs of attributes in \(V\). Further, any such \(N_s\) also minimizes the sum of squares distance function computed based on the overall disagreement between...
In order to prove Theorem 7, we present a procedure to compute the overall disagreement, due to a given attribute, between one of the input CP-nets $N$, and the summary CP-net $N_s$. We begin by stating our assumptions, and then give an outline of the computation process.

We assume we know the set $V' \subseteq V$ of attributes that have a different CPT in $N$ and $N_s$, with $|V'| = m$. We also know $CPT(N, V_i)$ and $CPT(N_s, V_i)$ for all $V_i \in V'$. Lastly, for all $V_i \in V'$, $z(V_i) + z'(V_i) = 1$.

**Lemma 15** The overall disagreement between a pair of CP-nets $N$ and $N_s$ is given by the number of outcome pairs $(o_1, o_2)$ ordered differently by $N$ and $N_s$, such that for all swapped attributes $V_i$, $o_1$ is assigned the preferred value according to $CPT(N, V_i)$.

**Proof.** Consider any outcome pair $(o_1, o_2)$ such that at least one swapped attribute has the most preferred value according to $N$ in $o_1$ and at least one swapped attribute has the most preferred value according to $N$ in $o_2$. All other swapped attributes have their preferred value in $o_1$. Then $(o_1, o_2)$ needs a trade-off and is incomparable by $N$. By our definition of disagreement, $(o_1, o_2)$ counts as an agreement regardless of how $N_s$ orders them. Thus, we need not count these pairs when counting disagreement. For all other pairs $(o_1, o_2)$, we count the number ordered differently by $N$ and $N_s$, but to avoid duplicate counting, we consider only those in which all preferred values of swapped attributes are in $o_1$, not in $o_2$. This gives us the overall disagreement.
between $N$ and $N_s$.  

**Algorithm 4:** A procedure to compute the overall disagreement of two CP-nets

```
1 ComputeOverallDisagreement ($N, N_s$);

   **Input:** Two CP-nets $N$ and $N_s$
   **Output:** $OD = \Delta_{overall}(N, N_s)$

2 $n = |V|$  
3 $OD = 0$  
4 for $j = 1$ to $n$ do  
5    Generate each of the $\binom{n}{j}$ possible subsets of $V$ of size $j$  
6        for each subset $U$ do  
7            JDD = Joint degree of disagreement for all $U_i \in U$  
8            count = number of pairs whose ordering depends on $U$  
9            Disagreement = JDD \cdot count  
10           OD = OD + count  
11        end  
12 end
```

We now outline our process for computing the number of arbitrary outcome pairs on which $N$ and $N_s$ disagree. We only need to consider outcome pairs $(o_1, o_2)$ that differ on at least one attribute $V_i \in V'$. It is easy to see that outcome pairs that only differ on attributes in $V \setminus V'$ will be ordered the same way by $N$ and $N_s$.

Outcome pairs $(o_1, o_2)$ that differ on exactly one attribute $V_i \in V'$ are the swaps of $V_i$. The number of such pairs on which $N$ and $N_s$ disagree can be computed by multiplying the number of swaps of $V_i$ by $z(V_i)$.

$N$ and $N_s$ also disagree on all pairs $(o_1, o_2)$ that differ on $j$ of the attributes in $V'$, where $1 \leq j \leq m$. We now discuss how to count the number of such pairs.
Suppose \((o_1, o_2)\) differ on \(j\) attributes, all in \(V'\). We first compute the joint degree of agreement for the \(j\) attributes swapped in \(o_1\) and \(o_2\), from which we obtain the joint degree of disagreement. Then we count the pairs \((o_1, o_2)\) differing on these \(j\) attributes whose ordering depends on the CPTs of those attributes. Multiplying this number by the joint degree of disagreement gives us the number of pairs on which \(N\) and \(N_s\) disagree due to being swapped on these \(j\) attributes. We repeat this process for values of \(j\) in \((1, m)\) and all \(\binom{m}{j}\) subsets of \(V'\). The sum of these values gives us the number of outcome pairs on which \(N\) and \(N_s\) disagree, which is the overall disagreement between \(N\) and \(N_s\).

The joint degree of agreement for any set of attributes is the product of the individual degrees of agreement of the attributes, using conditional degrees of agreement where necessary. This lets us state the following result.

**Lemma 16** For a given input set \(T\) of separable CP-nets, there exists a summary CP-net \(N_s\) that attains the maximum conditional degree of agreement for all attribute pairs \((V_i, V_j)\) in \(V\) such that \(\text{Parent}(N_s, V_i) \cap \text{Parent}(N_s, V_j) \neq \emptyset\). Further, such an \(N_s\) also has the maximum joint degree of agreement for \(V\).

We can obtain such a CP-net \(N_s\) by ensuring that for a given value assigned to the common parents, \(V_i\) and \(V_j\) have the same preference ordering as the majority of input CP-nets. Any \(N_s\) with the maximum joint degree of agreement also has the minimum joint degree of disagreement.

This explains the differences in Examples 4.3.2 and 4.3.3. In Example 4.3.2, when \(A = a'\), the CPTs of both \(B\) and \(D\) have at least one rule that is
the same in $N$ and $N_s$. In Example 4.3.3, there are no such rules and thus no agreement when both $B$ and $D$ are swapped.

While computing each term for a given $j$ is easy, the total number of terms is exponential in $m$. However, this process still allows us to state the following lemmas.

**Lemma 17** For a pair of $N, N_s$ of CP-nets, let $V'$ denote the set of attributes for which $N$ and $N_s$ have different CPTs. Given the set $V'$ and the CPTs for all $V_i \in V'$, Algorithm 4 computes the overall disagreement between $N$ and $N_s$ without enumerating all outcome pairs to check their outcome order iteratively.

**Lemma 18** For a given input set $T$ of separable CP-nets, there exists a summary CP-net $N_s$ which attains the minimum possible value for the joint degree of disagreement for all possible pairs of attributes in $V$. Further, any such $N_s$ also has the minimum possible overall disagreement with the majority CPT for each attribute in $V$.

**Lemma 19** For a given input set $T$ of separable CP-nets, ensuring the minimum degree of disagreement between $N_s$ and the majority CPT for all $V_i \in V$ is necessary but not sufficient to ensure the minimum joint degree of disagreement for all pairs of attributes in $V'$.

**Proof.** When $(o_1, o_2)$ has more than one swapped attribute, we compute the joint degree of disagreement for the set of swapped attributes by computing the joint degree of agreement and subtracting from 1. The joint degree of agreement for a set of attributes $U$ is the product of the individual degrees of
agreement for each $U_i \in U$, using conditional degrees of agreement whenever two attributes have at least one common parent. Clearly, maximizing the degree of agreement is equivalent to minimizing the degree of disagreement. For attributes that do not have any parents in common, minimizing their individual degrees of disagreement also minimizes their joint degree of disagreement. For attributes with an overlapping parent set, the joint degree of disagreement depends on the conditional degree of disagreement, which is different from the individual degree of disagreement. In this latter case we need to minimize the conditional degree of disagreement, and not just the individual degrees of disagreement. This completes our proof. 

Combining the above discussion and lemmas together proves Theorem 7. We now demonstrate how to build CPTs for attributes in $N_s$ in order to achieve this.

We refer back to the example we used for Algorithms 2 and 3. In case of Algorithm 3, attributes $V_1$ and $V_2$ have a common parent, $V_3$. In order to minimize the joint disagreement for $V_1$ and $V_2$, we ensure that $CPT(V_1)$ and $CPT(V_2)$ both have the majority CPT rule for the same value of $V_3$. This is achieved by choosing CPT rules as shown in Figure 5.1.

Similarly, $V_1$ and $V_3$ have a common parent, $V_2$. When building $CPT(V_3)$, we ensure that whenever $V_2$ has the same value in both CPTs and $CPT(V_1)$ has the majority rule, $CPT(V_3)$ also has the majority rule. This ensures that the conditional and joint degree of disagreement are minimized for all attributes.

Now that we have demonstrated how to build the CPTs to minimize the
joint degree of disagreement whenever a pair of attributes have at least one parent in common, we can state the following corollary.

**Corollary 2** Algorithm 3 finds an optimal summary CP-net $N_s$, in terms of sum of squares distance based on overall disagreement, in linear time given a set $T$ of separable CP-nets.

**Proof.** Algorithm 4 gives us a way to compute the overall disagreement between a pair of CP-nets. The computation process demonstrates that the $N_s$ that minimizes joint degree of disagreement with the majority CPT for all attribute pairs, also minimizes the overall disagreement with the majority CPT for all attributes. We have also described a method to ensure that the joint degree of disagreement is minimized between all appropriate attribute pairs, which in turn minimizes the overall disagreement. This proves our corollary.

We conclude this chapter with some final remarks on the algorithms presented here. Algorithm 2 uses the absolute distance function, and is significantly simpler than Algorithm 3. Using Algorithm 2 also lets us claim that when all the inputs are separable CP-nets, the optimal CP-net is also separable. On the other hand, Algorithm 3 uses the sum of squares distance function. The choice of a more complicated distance function leads to a more complicated algorithm. Further, as we saw in our example, the optimal summary CP-net found by Algorithm 3 might be cyclic even when all the inputs are separable CP-nets. Since not all cyclic CP-nets are consistent, we would like to investigate whether there is always a way to build the CPTs such that the resulting CP-net is consistent.
Chapter 6

Conclusions and Future Work

In this thesis, we proposed that a set of CP-nets can be represented by a single summary CP-net that approximates the preference ordering of each individual CP-net. In order to quantify the extent to which a summary CP-net approximates the preferences of one of the input CP-nets, we proposed an efficient algorithm to compute the pairwise swap disagreement between two CP-nets. We then used the pairwise swap disagreement to define two different distance measures, to quantify the semantic difference between a set of CP-nets and their representative summary CP-net. Finally, we presented two algorithms to find an optimal summary CP-net, and included discussions on their correctness and complexity.

As mentioned, we have restricted both the algorithms presented here to the case of separable CP-nets. While this is a significant restriction, we have been able to provide guarantees of optimality and complexity for both algo-
rithms under this assumption. In the future, we would like to focus on more
general CP-nets, using the algorithms discussed here as a starting point. Event-
tually, we hope to use our notion of CP-net similarity and summary CP-nets
to perform hierarchical clustering of CP-nets. Since the output to even a set
of separable CP-nets may be a cyclic CP-net, we will need to design algo-

Another possible future direction is given by our observations on Algorithm
3. The output of Algorithm 3 is often a cyclic CP-net. We cannot guarantee
that a cyclic CP-net will induce a consistent preference ordering. Since the
CPTs can be built in more than one way, we would like to explore whether
Algorithm 3 can always pick a cyclic CP-net which is consistent.

Our interest in ensuring consistency stems from two reasons. Firstly, unless
the CP-net is consistent, the preference ordering induced by this CP-net will
not be consistent either. Secondly, in this thesis we have focused on measuring
the semantic similarity of a pair of CP-nets using swap disagreement. However,
the overall disagreement between a pair of CP-nets is a much better indicator of
their similarity. Our definition of overall disagreement given in this thesis relies
on a consistent preference ordering. Thus, the ability to guarantee consistency
would enable us to use this definition of overall disagreement without any
problems. In the future, we would like to extend our work to use the overall
disagreement between a pair of CP-nets.

Another assumption we have made in this thesis is that all CP-nets are
complete. A CP-net is said to be incomplete if some CPT entries are missing. In practical applications, this could happen because we do not know the preferences of the user for a specific context. When extending to general CP-nets with arbitrary parent sizes, the focus on complete CP-nets becomes more restrictive. This is because attributes with large parent sets will have CPTs with many parent contexts, increasing the likelihood that the user’s preferences might not be known for some of these. Thus, in the future we would like to extend our algorithms to account for incomplete CP-nets. In addition, we would like to review our algorithms and formal results to extend them to the case when the attribute domains are non-binary.
References


[13] Juntao Liu, Yi Xiong, Caihua Wu, Zhijun Yao, and Wenyu Liu. Learn-


