Reasoning with Conditional Preferences across Attributes

Chen, S.; Buffett, Scott; and Fleming, Michael

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC·CNRC

*Reasoning with Conditional Preferences Across Attributes**

Chen, S., Buffett, S., and Fleming, W.
2007

* 20th Canadian Conference on Artificial Intelligence. May 28, 2007.
NRC 49292.

Canada

Reasoning with Conditional Preferences across Attributes

Shaoju Chen[1], Scott Buffett[2], and Michael W. Fleming[1]

[1] University of New Brunswick, Fredericton, NB, E3B 5A3
{shaoju.chen,mwf}@unb.ca
[2] National Research Council Canada, Fredericton, NB, E3B 9W4
Scott.Buffett@nrc.gc.ca

Abstract. Before an autonomous agent can perform automated negotiation on behalf of a user in an electronic commerce transaction, the user's preferences over the set of outcomes must be learned as accurately as possible. This paper presents a structure, a Conditional Outcome Preference Network (COP-network), for modeling preferences directly elicited from a user. The COP-network then expands to indicate all preferences that can be inferred as a result. The network can be easily checked for consistency and redundancy, and can be used to determine quickly whether one outcome is preferred over another. An important feature of the COP-network is that conditional preferences, where a user's preference over outcomes depends on whether particular attribute values are included, can be modeled and inferred as well. If the agent also knows the user's utilities for some of the possible outcomes, then these can be considered in the COP-network as well. Three techniques for estimating utilities based on the specified preferences and utilities are described. One such technique, which works by first estimating utilities for long chains of outcomes for which preferences are known, is shown to be the most effective.

1 Introduction

The widespread use of the Internet today allows people to engage in more communication, interaction, and transactions online than ever before. Opportunities for automated negotiation between agents over the Internet are abundant, and the use of such technologies becomes more and more feasible as the speed of communication and processing increases. Buyers can negotiate with sellers over the price or other terms of a potential exchange. Web users can negotiate the terms of websites' policies for handling private information. Even the terms of use and configuration of web services can be negotiated. However, before negotiation can commence, an agent representing a user must know the user's utilities for potential agreements. Utility elicitation techniques can help the agent to learn some of the user's preferences, but it is typically infeasible to learn all of them due to the exponential number of outcomes. Therefore, an agent must gather as much information as possible from a few utility elicitation queries, and attempt to estimate or infer utilities over outcomes that cannot be elicited directly.

In the realm of privacy, as well as many other application areas, utilities for outcomes cannot typically be computed as an additive function of the user's utilities for individual aspects or attributes of those outcomes. This is due to the highly dependent nature of the attributes. Matters are complicated when a user specifies conditional preferences. For example, a user may not mind releasing information which identifies his place of employment, nor would he mind exposing his job title. However, he may have strong reservations when it comes to giving away both of these particular items of information, as it may personally identify him. So perhaps his utility for exposing his job title is dependent on whether his place of employment is also part of the final outcome.

One can envision other situations where this may be the case, such as in the stock market. Investors often prefer to create a balanced portfolio, where risks are hedged against each other and the chance of overall growth is maximized. Here, adding particular items to the portfolio will be more or less preferred, depending on which others are included. Another example is the case of determining options to be included in a new car. Perhaps air conditioning is important to a particular buyer, but becomes less important if a convertible roof is included.

Given this, one can see that it is quite complex to determine a global utility function that is consistent with all preferences that can be derived given the known interdependencies. In order to determine such utilities, a preference structure is needed. Boutilier et al [1, 2] present a structure, known as a CP-network, for reasoning about conditional preferences over values within a single attribute. Consider a car example with attributes "Make" and "Colour". A user may specify preferences for "Make" such as "Pontiac is preferred over Volkswagen", or "Colour" such as "Black is preferred over silver". From this, the reasoning technique can infer that black Pontiacs are preferred over silver Volkswagens, all else equal. Additionally, conditional preferences can be used. For example, consider a buyer that only likes Pontiacs that were made after 2002. Then the preference for "make" is conditional on the outcome for "year".

In the privacy example, attributes of outcomes correspond to items of personal information to be exchanged. Each attribute can then take on one of two values: "included in the agreement" or "not included in the agreement". Under Boutilier's model, the user can only specify preferences such as for "e-mail address" , "not included" is preferred over "included" (and likewise for "phone number"). In areas such as privacy where sets of items are being negotiated, a richer model is needed where preferences can be expressed across attributes, such as the value "included" is preferred for "phone number" over the value "included" for "e-mail address" (i.e., phone number is preferred over e-mail).

In this paper, we develop a preference structure that will indicate all preferences that can be derived, given the conditional and unconditional utilities elicited from the user. This structure is referred to as a Conditional Outcome Preference Network (COP-network or COPN). The COP-network is a directed graph where, given that the information elicited from the user is accurate, if an outcome o in the graph precedes another outcome o', then the user's true utility

for o is higher than that for o'. This network is then used to estimate the user's utilities for all outcomes for which the utility has not been elicited.

The paper is organized as follows. In Section 2 we briefly review the CP-network, and in Section 3 we introduce our COP-network. In Section 4 we demonstrate how the COPN can be used to estimate utilities over outcomes, and give a simple example of how this would be applied. In Section 5 we discuss the results of our tests for accuracy of our technique, and in Sections 6 and 7 we offer our conclusions and discuss directions for future work.

2 Conditional Preference Networks (CP-nets)

Boutilier et al. [1, 2] explore a representation referred to as a conditional preference network (CP-network) for structuring user preferences. The representation is based on the dependence and conditional preferential independence between attributes. A CP-network over a set of n attributes is graphical, where a node is created for every attribute. For each attribute, the user must identify a set of parent attributes whose values can influence the user's preference over values for the attribute. Each node has an associated table describing the user's preferences over values for the attribute given every combination of parent values. The theory works on the concept of *ceteris paribus* (all else being equal), where preferences are defined and accepted as being true, given the conditions, all else equal. Let X, Y and Z (non-empty) partition the set of attributes. X is said to be *conditionally preferentially independent* of Y given Z if, for any assignments x_1, x_2, y_1, y_2 and z to those sets of attributes:

$$x_1 y_1 z \preceq x_2 y_1 z \text{ iff } x_1 y_2 z \preceq x_2 y_2 z$$

Thus if Z is the set of parent attributes of X (i.e. the conditions imposed on the preferences in X), then given the conditions, preferences in X are said to hold *ceteris paribus*, meaning that values for attribute values for Y are irrelevant. We employ the *ceteris paribus* assumption in our model as well.

For example, suppose that there are four attributes, A, B, C, and D, and that each attribute has binary values (a and $\overline{a}$ are values for attribute A, b and $\overline{b}$ for B, *etc.*); and that attribute A has no parent, A is the parent of B, and B is the parent of C and D. Suppose that the conditional preferences are as follows:

$A:$	$\overline{a} \succ a$	($\overline{a}$ is preferred over a)
$B:$	$\overline{a} : \overline{b} \succ b$	(given $\overline{a}$, $\overline{b}$ is preferred over b)
	$a : b \succ \overline{b}$	(given a, b is preferred over $\overline{b}$)
$C:$	$\overline{b} : \overline{c} \succ c$	(given $\overline{b}$, $\overline{c}$ is preferred over c)
	$b : c \succ \overline{c}$	(given b, c is preferred over $\overline{c}$)
$D:$	$\overline{b} : d \succ \overline{d}$	(given $\overline{b}$, d is preferred over $\overline{d}$)
	$b : \overline{d} \succ d$	(given b, $\overline{d}$ is preferred over d)

The CP-network is shown in Figure 1.

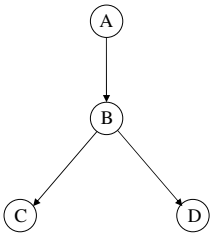

Fig. 1. An example CP-net.

A dominance checking algorithm is then used to determine whether one outcome is preferred over another. The idea here is that preferences higher in the CP-net are more important than those that are lower. That is, if outcome o has a violation in the user's preference for attribute X (i.e. the less preferred value is present), and outcome o' has a violation in attribute X', then if X is an ancestor of X' in the CP-net, then o is preferred over o'. In the example, $\overline{a}\overline{b}\overline{c}\overline{d}$ has a violation in D and $abc\overline{d}$ has a violation in A. Since A is an ancestor of D, the violation in $abc\overline{d}$ is more damaging than the violation in $\overline{a}\overline{b}\overline{c}\overline{d}$, and thus $\overline{a}\overline{b}\overline{c}\overline{d}$ is preferred over $abc\overline{d}$. This can be shown by a sequence of "flips":

$abc\overline{d} \prec \overline{a}bc\overline{d}$ (since $\overline{a} \succ a$, a is flipped to $\overline{a}$)
$\overline{a}bc\overline{d} \prec \overline{a}\overline{b}c\overline{d}$ (since $\overline{a} : \overline{b} \succ b$, b is flipped to $\overline{b}$)
$\overline{a}\overline{b}c\overline{d} \prec \overline{a}\overline{b}\overline{c}\overline{d}$ (since $\overline{b} : \overline{c} \succ c$, c is flipped to $\overline{c}$)

3 Conditional Outcome Preference Networks

In this section we define a structure for representing the specified preferences in such a way that new preferences that can be directly inferred will be immediately evident. The structure is a directed graph that represents preferences over the set of outcomes, and is referred to as a *Conditional Outcome Preference Network (COP-network)*. The main aim in using the network is to (1) determine whether one outcome is preferred over another, and (2) estimate utilities for the entire set of outcomes.

3.1 Creating an initial COP-network

Users can specify preferences in many formats. For example, a preference could be specified as a comparison of values from the same attribute or across different attributes, with or without condition, and over two or more than two values. While the CP-networks described in Section 2 are restricted to representing preferences over values within a single attribute, COP-networks can also handle preferences across different attributes.

To create an initial COP-network, each given preference is transformed to a standard format referred to as a *preference rule*. A preference rule $\overline{a_1} \succ \overline{a_2}$

for a set A of attributes is defined as a specification that represents that one assignment $\overline{a_1}$ to the attributes in A is preferred over another assignment $\overline{a_2}$. Before the COP-network is constructed, all preferences specified by the user are transformed to preference rules. For example, the two conditional preferences $\overline{a} : b\overline{c} \succ \overline{b}c$ and $a : \overline{b}c \succ b\overline{c}$ would be transformed into the two preference rules $\overline{a}b\overline{c} \succ \overline{a}\overline{b}c$ and $a\overline{b}c \succ ab\overline{c}$.

A COP-network is represented by a directed graph, where every outcome is represented by a node, and for nodes n and n' representing outcomes o and o', respectively, if n is a proper ancestor of n' then o is necessarily preferred over o', given the specified preferences and the *ceteris paribus* assumption. Initially, for every specified preference $o \succ o'$, an arc is inserted from the node representing o to the node representing o'. In subsequent sections, we discuss consistency checking and removal of redundant edges. Such a graph, without consistency checking and reduction, is referred to as an *initial* COP-network.

Example 3.1 Suppose that there is a set $\{A, B, C\}$ of attributes, and that each attribute has binary values (a and $\overline{a}$ are values for attribute A, b and $\overline{b}$ for B, c and $\overline{c}$ for C), and that there are the following preferences:

$$\overline{a} \succ a \quad , \quad \overline{b} \succ b \quad , \quad \overline{c} \succ c \quad , \quad a\overline{b} \succ \overline{a}b \quad , \quad \overline{a} : b\overline{c} \succ \overline{b}c \quad , \quad a : \overline{b}c \succ b\overline{c}$$

To structure a COP-network with the above preferences, all feasible outcomes are listed: $\overline{a}\overline{b}\overline{c}$, $a\overline{b}\overline{c}$, $\overline{a}b\overline{c}$, $ab\overline{c}$, $\overline{a}\overline{b}c$, $a\overline{b}c$, $\overline{a}bc$, abc. Next, preference rules as dictated by the given preferences are applied to the outcomes, and a set of preferences over the outcomes is generated. For example, by applying the preference rule $\overline{a} \succ a$, we determine that outcome $\overline{a}bc$ is preferred over outcome abc. The final step is to build a directed graph by creating a node for every outcome and adding a directed edge from node n_i to node n_j if the preference $o_i \succ o_j$ holds for the corresponding outcomes. The resulting graph is shown in Figure 2, where Table 1 denotes which outcome is represented by each node.

Node:	n_0	n_1	n_2	n_3	n_4	n_5	n_6	n_7
Outcome:	ϕ	a	b	ab	c	ac	bc	abc

Table 1. Node representation for Figure 2. Bar values are removed (e.g. $a\overline{b}\overline{c} \Rightarrow a$)

3.2 Consistency

In decision making, an outcome cannot be preferred over itself. For any given set of preferences, it can be determined whether an outcome is preferred over itself by building an initial COP-network and checking whether there is a cycle in the network. An outcome is preferred over another outcome if there is a path from a node for the first outcome to another node for the second outcome. An initial COP-network is said to be consistent if and only if there is no outcome that is preferred over itself - *i.e.*, if and only if the network is acyclic. If a COP-network corresponding to a given set of preferences is found to have a cycle, then the user must be consulted in order to correct the inconsistency.

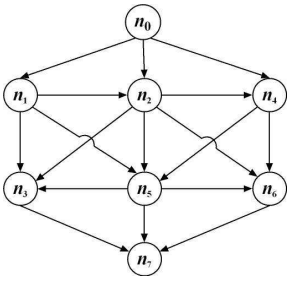

Fig. 2. The initial COPN.

3.3 Reducing an initial COP-network

For nodes n_i, n_j and n_k in an initial COP-network, if there are two paths $n_i \rightarrow n_j \rightarrow \cdots \rightarrow n_k$ and $n_i \rightarrow n_k$, the second path (i.e. the arc from n_i to n_k) is not necessary since preferences that are reflected by the first path include the preference that the second path reflects. Thus, the arc (n_i, n_k) is said to be *redundant* and can be removed. This results in a transitive reduction of the initial COP-network. The aim of reducing the network is to make it easy to compute a utility function based on the network.

The graph from Figure 2 can be reduced to the graph shown in Figure 3.

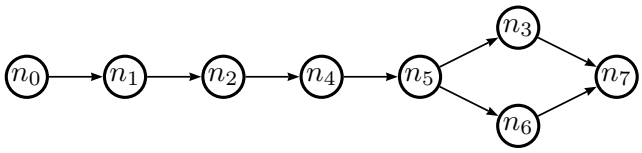

Fig. 3. The reduced COP-network.

3.4 Preference Checking

If each outcome is associated with an offer in a negotiation, the ability to show that one outcome is preferred over another should help the decision making in the negotiation. In a COP-network, the outcome corresponding to a node is preferred over the outcome associated with any proper descendant. For any pair of outcomes, preference checking is quite easy and efficient. For example, for any pair of outcomes o_i and o_j with corresponding nodes n_i and n_j:

- If n_i is a proper ancestor of n_j, o_i is preferred over o_j ($o_i \succ o_j$):
- If n_i is a proper descendant of n_j, o_j is preferred over o_i ($o_j \succ o_i$);
- If n_i is neither an ancestor nor a descendant of n_j, neither of o_i and o_j is known to be preferred over the other

Consider Example 3.1. Outcome o_0 is preferred over outcome o_1 since n_0 is the parent of n_1. Outcome o_1 is preferred over outcome o_7 since n_1 is an ancestor of n_7. Neither of outcomes o_3 and o_6 is known to be preferred over the other since n_3 is neither an ancestor nor a descendant of n_6.

4 COPN Utility Functions

In addition to obtaining a set of preferences from the user during the elicitation stage, our model also allows for querying about specific utilities for outcomes. This can be done by asking standard gamble questions (see Keeney and Raiffa [8]), or by initially treating utility for an outcome as a random variable from a known distribution, and querying the user to sufficiently reduce the uncertainty in the utility estimate (see Chajewska et al. [5]), to cite some examples. Note that there will always be at least two outcomes for which utility is known, since we employ the convention of assigning a utility of 1 to the most preferred outcome (the topmost node in the network), and a utility of 0 to the least preferred outcome (the bottommost node in the network). Based on the COP-net derived from the specified preferences, and the partial utility function $u : O' \rightarrow \Re$ specifying utilities for a subset O' of outcomes, a utility function $\hat{u}$ over the entire set O is produced. This is done in such a way as to preserve the preference ordering specified by the COP-net. Specifically, let n and n' represent outcomes o and o'. If n is a proper ancestor of n', then $\hat{u}(o) > \hat{u}(o')$.

This section demonstrates three techniques for computing $\hat{u}$: The Bounded method, the Random-Path method and the Longest-Path method. Each method is then tested for accuracy against an existing method.

4.1 The Bounded Method

The Bounded method computes the utility by setting upper and lower bounds for each outcome o for which utility is unknown, and assigns the average of these bounds as the utility value. Let n represent o in the tree, let O_k be the set of outcomes for which the utility is known, and let $O_a, O_d \subseteq O_k$ be the set of outcomes represented by ancestors and descendents of n, respectively. Then the Bounded method computes the utility estimate $\hat{u}_B$ as

$$\hat{u} = \frac{min\{u(o') \mid o' \in O_a\} + max\{u(o') \mid o' \in O_d\}}{2} \tag{1}$$

By selecting the value that lies in the middle of the possible range, the bounded method produces utilities that are, in most cases, not too far from the true utilities. However, when there are paths of outcomes in the COP-net for which the preference ordering is known, if the utilities for the outcomes have the same upper and lower bounds the Bounded method will assign the same utility to each outcome. The next two methods overcome this drawback by assigning utilities that preserve preference orderings.

4.2 The Random-Path Method

Given a COP-network and a set of known utilities, the Random-path technique randomly selects a path of outcomes in the network for which utilities are unknown, and assigns utilities to those outcomes in such a way that preserves this preference ordering. Formally, let $p = (o_1, o_2, \ldots, o_n)$ be a path in the network. This path is a candidate for selection if (1) $\hat{u}$ is known for o_1 and o_n, and $\hat{u}$ is unknown for all other outcome nodes on p, (2) for all paths satisfying (1), $\hat{u}(o_1)$ is minimal, and (3) for all paths satisfying (1) and (2), $\hat{u}(o_n)$ is maximal. Requirements (2) and (3) ensure consistency in the utilities in the graph[3]. Once a suitable path p has been selected, the utility $\hat{u}$ is assigned for each outcome on p, decreasing from o_1 to o_n, by

$$\hat{u}(o_i) = \hat{u}(o_n) + \frac{(n-i)(\hat{u}(o_1) - \hat{u}(o_n))}{n-1} \tag{2}$$

For example if p consisted of four outcomes with $\hat{u}(o_1) = 0.8$ and $\hat{u}(o_4) = 0.2$, then $\hat{u}(o_2)$ and $\hat{u}(o_3)$ would be assigned utilities of 0.6 and 0.4, respectively.

The process of selecting paths at random and assigning utilities in this way continues until all outcomes are considered.

4.3 The Longest-Path Method

The Longest-Path method works in much the same manner as the Random-Path method, except that the longest acceptable candidate path is always selected. Utilities for outcomes are assigned according to Equation 2. Path selection continues until all outcomes have been considered.

4.4 A Simple Example

Consider the COP-network in Figure 4 containing 6 nodes, where each node n_i represents outcome o_i. Initially let $u(o_1) = 0.82$ and $u(o_6) = 0.1$. Each of the three techniques described above computes $\hat{u}$ for outcomes o_2 to o_5 as follows:

Bounded: Since the upper bound for all outcomes o_2 to o_5 is 0.82 and the lower bound is 0.1, $\hat{u}(o_i) = \frac{0.82-0.1}{2} = 0.46$ for all $i = 2$ to 5.
Random-Path: Random-Path will randomly begin with one of two paths: $p_1 = (o_1, o_2, o_3, o_5, o_6)$ or $p_2 = (o_1, o_4, o_5, o_6)$.

1. If p_1 is chosen first, then $\hat{u}(o_2) = 0.64$, $\hat{u}(o_3) = 0.46$ and $\hat{u}(o_5) = 0.28$. Next, path (o_1, o_4, o_5) is chosen and $\hat{u}(o_4) = 0.55$.
2. If p_2 is chosen first, then $\hat{u}(o_4) = 0.58$ and $\hat{u}(o_5) = 0.34$. Next, path (o_1, o_2, o_3, o_5) is chosen and $\hat{u}(o_2) = 0.66$ and $\hat{u}(o_3) = 0.5$.

Longest-Path: Longest-Path will choose $p_1 = (o_1, o_2, o_3, o_5, o_6)$ first (since it is the longer of p_1 and p_2), and compute utilities as in point 1 in Random-Path above.

[3] Refer to Chen [6] for more on ensuring consistency in path selection.

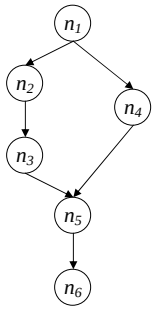

Fig. 4. An example COP-net for computing utilities

5 Analysis

5.1 Accuracy Testing

There are three techniques developed in this paper and discussed in previous sections. Given a set of preferences and given utilities for some of the possible outcomes, each technique constructs a COP-network and develops a utility function to predict all unknown utilities. Experiments were run to compare the accuracy of these three techniques as well as a previously developed technique for determining utilities, which we refer to as the *additive utility*. This technique, which is used by the "MONOLOGUE" automated negotiation system [4], handles interdependencies among attribute values that result from the specified conditional preferences by modifying the amount of utility that each attribute value contributes in a given outcome. For example, if an attribute value a is considered less desirable when attribute value b is present, then a contributes less utility to an outcome including b than it would to an outcome not including b. The overall utility for an outcome is then the sum of these modified utilities.

To test the accuracy of the algorithms, test cases were generated for different numbers of attributes and different numbers of conditional preferences. Tests were then run on these cases to determine how accurately the techniques could estimate a simulated user's true utilities for all outcomes, given a small number of preferences and known utilities. In this experimentation, there were 33 test cases. The numbers of attributes ranged from 4 to 9. The numbers of conditional preferences ranged from 0 to 7.

For each test case, several sets of user preferences were generated, giving a large number of different test COP-networks. For each of these COP-networks, 10,000 trials were run. In each trial, a set of true utilities was generated to be consistent with the given preferences. Each of the four techniques (Bounded, Random-Path, Longest-Path and Additive) was then tested, to determine how accurately it could estimate the true utilities.

In order to determine the accuracy of the four techniques, two measures were considered: the differences between computed utilities and true utilities and the

standard errors of those differences. For each test case, each technique computed the utility for each outcome, and the difference between the computed utility and the true utility was noted. The winning technique was determined for each of the following criteria:

1. **Total difference winner**: The technique with the lowest difference for an outcome is viewed as having the best ability to predict utility for that outcome in the particular test case. This technique is deemed the *difference winner* for the outcome. For all test cases, the technique deemed the difference winner the most often is viewed as having the best ability to predict utility. This technique is deemed the *total difference winner*.
2. **Difference mean winner**: For all test cases, the technique with the lowest *mean* of differences over all outcomes is deemed the *difference mean winner*.
3. **Total standard error winner**: For each trial, the standard error over the set of estimated utilities is measured. The technique with the lowest standard error is deemed the *standard error winner* for the trial. For all test cases, the technique deemed the standard error winner the most often is viewed as having the best ability to predict utility. This technique is deemed the *total standard error winner*.
4. **Standard error mean winner**: For all test cases, the technique with the minimal *mean* of standard errors over all outcomes is deemed to be the *standard error mean winner*.

The accuracy of predicting utility of the four techniques is evaluated by considering the *total difference winner*, the *difference mean winner*, the *total standard error winner*, and the *standard error mean winner*. Table 2 shows the number of times each technique was the winner for each of these four criteria. Clearly, the Longest-Path technique is shown to most accurately predict utility regardless of the numbers of attributes and conditional preferences.

Technique	Total difference winner		Difference mean winner		Standard error winner		Standard error mean winner	
Bounded	0	(0%)	0	(0%)	0	(0%)	0	(0%)
Random-Path	1	(3%)	6.5	(20%)	1	(3%)	5	(15%)
Longest-Path	23	(70%)	25.5	(77%)	32	(97%)	28	(85%)
Additive	9	(27%)	1	(3%)	0	(0%)	0	(0%)

Table 2. Experimental results

5.2 Discussion on Running Time

Since a COP-net contains a node for every possible outcome, run-time for building and traversing the tree is very expensive in the worst case. Let n denote

the number of attributes. If attributes are binary-valued, there are 2^n outcomes. Testing showed that algorithms for computing utilities began to slow significantly at $n = 15$. We envision that, in most practical applications of the technology, 15 attributes is more than sufficient. For example, when negotiating which items will be exchanged in a privacy scenario, or which options will be included in a car, it is difficult to imagine scenarios where both parties have enough concern over so many variables that more than 15 would need to be negotiated. However, in cases where significantly more items are involved, the COP-net can be divided into two or more sub-networks. This is done by partitioning the set of attributes such that dependent attributes are grouped together, and independent attributes are separated. A COP-network is then built for each group. Each such group is unlikely to consist of more than 15 attributes. Utilities can then be computed for outcomes in these COP-nets independently, and a multi-attribute utility function can be used to determine utilities for complete outcomes.

6 Conclusions and Related Work

In this paper, a graphical model referred to as a Conditional Outcome Preference Network (COP-network) is described. Using this model, techniques are developed to infer user preferences and utilities over all possible outcomes, given a small set of known preferences and utilities. Previous preference networks (such as CP-networks [1]) have handled only preferences specified over values for a particular attribute. In this paper, techniques are developed that can infer preferences over outcomes when user preferences are specified for values across attributes as well. As in previous techniques, conditional preferences are also handled. Efficient algorithms have been presented for checking the consistency of a COP-network and for using a given network to determine the user's preferences between any two possible outcomes. Three techniques are presented for estimating utilities for outcomes in the COP-net: Bounded, Random-Path and Longest-Path. Experiments show that the Longest-Path technique achieves the best results of the three, and also outperforms an existing technique.

Preference elicitation is becoming an increasingly popular topic for researchers working in the areas of agents and electronic commerce. Boutilier et al. [3] propose a minimax regret-based approach to preference elicitation. Given a decision problem, choices are made that the user would regret the least should an adversary choose the utility function consistent with the elicited preferences. If regret is higher than some threshold, then more querying is necessary. Determining such a consistent utility function is difficult, especially when conditional preferences exists, so perhaps our work can complement this. Other works on utility elicitation, such as those by Chajewska et al. [5] and Haddawy et al. [7], demonstrate effective ways to estimate utilities based on data obtained on other individuals' preferences or utilities over outcomes. Our work differs from these as we assume that no such data exists.

7 Future Work

For future work, a COP-network capturing a user's known preferences and a set of estimated utilities could be used to make decisions about which preference elicitation questions to ask next. If two nodes in the graph have the property that neither is an ancestor of the other, then it would be reasonable to ask the user a preference elicitation question with the goal of determining which outcome is preferable. However, learning the user's preference over some pairs of outcomes might be more informative than other pairs. For example, due to the structure of the graph, learning that the user prefers outcome o_1 over outcome o_2 ($o_1 \succ o_2$) might also tell the agent that $o_3 \succ o_2$ and that $o_1 \succ o_4$. This then has the potential to be a more useful question than one that provides no such additional preference information.

The approach to this problem will include experimenting with graph-theoretic methods to find a set of candidate edges corresponding to potential preference elicitation questions, and then evaluating the anticipated effect of learning the answer to each of these questions on the expected utility of the agent's strategy (perhaps a negotiation strategy to be used in negotiations with an opposing agent). The question perceived to yield the highest increase in expected utility would be the next question chosen.

Another possible direction for future work is to investigate the feasibility of reducing the space consumption of the current COP-network model by attempting to modify the network so that it contains a node for each attribute rather than each outcome.

References

1. C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research (JAIR)*, 21:135–191, 2004.
2. C. Boutilier, R. I. Brafman, C. Geib, and D. Poole. A constraint-based approach to preference elicitation and decision making. *AAAI Spring Symposium on Qualitative Preferences in Deliberation and Practical Reasoning*, pages 19–28, 1997.
3. C. Boutilier, R. Patrascu, P. Poupart, and D. Schuurmans. Regret-based utility elicitation in constraint-based decision problems. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 929–934, Edinburgh, Scotland, 2005.
4. S. Buffett, L. Comeau, M. W. Fleming, and B. Spencer. Monologue: A tool for negotiating exchanges of private information in e-commerce. In *Third Annual Conference on Privacy, Security and Trust (PST05)*, pages 79–88, 2005.
5. U. Chajewska, D. Koller, and R. Parr. Making rational decisions using adaptive utility elicitation. In *AAAI-00*, pages 363–369, Austin, Texas, USA, 2000.
6. S. Chen. Reasoning with conditional preferences across attributes. Master's thesis, University of New Brunswick, 2006.
7. P. Haddawy, V. Ha, A. Restificar, B. Geisler, and J. Miyamoto. Preference elicitation via theory refinement. *Journal of Machine Learning Research*, 4:313–337, 2003.
8. R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. John Wiley and Sons, Inc., 1976.