Completion of User Preference based on CP-nets in Automated Negotiation

Jianlong Cai, Jieyu Zhan* and Yuncheng Jiang
Guangzhou Key Laboratory of Big Data and Intelligent Education, School of Computer Science, South China Normal University, Guangzhou, China

Keywords: CP-nets, Automated Negotiation, Preference Completion, Incomplete Information.

Abstract: Automated negotiation is a process in which autonomous agents negotiate with the opponents to achieve some specific purposes for their users, such as maximising the users' benefits. CP-net is one of the most important representations of user preferences in automated negotiation due to its ability and flexibility to express interdependent relationship among issues. In order to be able to negotiate better on behalf of users, a negotiating agent needs to fully understand its user's preferences, so that it can adopt suitable negotiation strategies and obtain ideal negotiation results. However, the preference information provided to the negotiating agents by the users is often incomplete. Hence, based on partial preference information provided, this paper proposes a module in negotiation framework to complete user total preferences that are represented by CP-nets. The experimental results show the validity of CP-nets structure learning algorithm in the proposed module and confirm that the module can help users achieve better agreements in negotiation.

1 INTRODUCTION

Negotiation is a behavioral activity for both parties to solve problems, which is ubiquitous in human life. With the rapid development of computer technology, people have begun to be interested in automated negotiation (Jennings et al., 2001; Kumar and Mastorakis, 2009), that is, a negotiating agent can help or represent human users to negotiate with other agents or human opponents. Because of the characteristics of intelligence and efficiency, negotiating agents have begun to be widely employed in supply chain, e-commerce and other fields (Tsimpoukis et al., 2018; Sanchez-Anguix et al., 2021).

Modelling preferences is a necessary condition for any step of decision analysis in the field of automated negotiation. Therefore, different kinds of representations of user preference have been proposed in the previous studies (Lafage and Lang, 2000; Boutilier et al., 2004; Amor et al., 2016). However, most of the studies ignore the situation of incomplete user preferences. In practical negotiation situations, the assumption of complete user preference is often difficult to meet due to many reasons. Firstly, users may not be willing to spend much time to describe the utilities of all outcomes when the issues are interdependent and the outcome space of negotiation is huge. Secondly, perhaps due to the lack of information, users cannot give clear preferences to all outcomes of negotiation. Thirdly, some users may only provide partial preference information for privacy reasons. Therefore, it is very necessary to study how to reason and complete user preference in negotiation scenarios with incomplete user preference information.

Although there are many kinds of preference representations studied in the field of automated negotiation, in this paper we focus on the one based on Conditional preference networks (CP-nets) by considering the following advantages. Firstly, CP-nets can represent qualitative preferences and tolerates partial ordering, which is suitable in some scenarios where it is difficult for the user to assess their preferences in a quantitative form. Secondly, CP-nets allow representation of conditional preferences, which can express interdependent relationship among issues in reality. Finally, due to the mass of outcome space, it can be indecisive for users to provide complete information about their preferences of outcomes one by one, while CP-nets provide a relatively convenient and intuitive way to represent preferences.

This paper focuses on automated negotiation with incomplete user preference information. The contribution of this paper is twofold. Firstly, we extend au-

*Corresponding author

Cai, J., Zhan, J. and Jiang, Y.
Completion of User Preference based on CP-nets in Automated Negotiation.
DOI: 10.5220/0010909200003116
In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022) - Volume 1*, pages 383-390
ISBN: 978-989-758-547-0; ISSN: 2184-433X

tomated negotiation frameworks by proposing a module for reasoning and completing user preferences represented by CP-nets. Secondly, we propose an algorithm to learn structures of corresponding CP-nets from partial pairwise comparisons provided by users.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 presents background knowledge on automated negotiation and CP-nets. Section 4 describes the methods for user preference modelling and completion; Section 5 discusses the experimental setup and analyses the results. Finally, Section 6 concludes the paper with future work.

2 RELATED WORK

In this section, we will discuss related work about both negotiations with incomplete information and learning CP-nets. Incomplete information in automated negotiations have been studied in many different forms, most of which focus on the uncertainty of opponent model including preferences modelling and strategies. Compared with opponent modelling, the research on user modelling with incomplete information is relatively less. (Baarslag and Gerding, 2015) put forward a method to solve the problem of preference elicitation in negotiation. Using the idea of greed, they proposed an optimization algorithm with a time complexity of $O(nlogn)$. Further than the original algorithm, the following research (Baarslag and Kaisers, 2017) supported the situation of stochastic utility information. (Haddawy et al., 2003) proposed a method for eliciting user preference models. The method is based on the Knowledge Based Artificial Neural Network (KBANN) pioneered by (Towell and Shavlik, 1994). When domain knowledge is available, even in the form of weak and inaccurate hypotheses, much less data is required to construct an accurate user preference model. However, the method for user preference elicitation in (Haddawy et al., 2003) is seen as a kind of supervised learning methods that is not genuinely suitable for using in automated negotiation scenarios.

The preference representation used in most automated negotiation frameworks is represented as linear additive utility functions (Baarslag, 2016), which is simple and intuitive to compute. However, in practical situations, there is likely to be interdependence among the issues of negotiation. CP-nets allow representation of conditional preferences, which is more intuitive to be used to indicate preferences (Boutilier et al., 2004). Since CP-net represents a partial order over outcome space, learning CP-nets can be seen as a ranking learning of preferences from known pairwise comparisons (Liu et al., 2018). (Goldsmith et al., 2008) investigated the computational complexity of testing dominance and consistency in CP-nets, and proved that complexity of testing dominance and ordering queries are in general NP-hard. CP-nets can be cyclic or acyclic, but for the sake of intuition and consistency, the CP-nets we describe default to acyclic CP-nets. (Liu et al., 2018) learn structure of CP-nets by calculating dependent degree among attributes. However, most studies of structure learning only focus on binary-valued CP-nets, which is limited in the field of automated negotiation.

(Dimopoulos et al., 2009) proposed an algorithm that converts pairwise comparisons to binary-clauses to learn structure of CP-nets. For employing it in automated negotiation scenarios, we have extended the algorithm by three aspects that will be described in detail in Section 4. (Aydoğan and Yolum, 2010) first apply CP-nets as preference representations to utility-based automated negotiation, and extend the algorithms in (Aydoğan et al., 2015). However, this is an estimate of the utility of preferences only if the CP-nets is known, while our work focuses on learning unknown CP-nets in automated negotiation scenarios.

3 PRELIMINARIES

In this section, we give a brief overview of the basic knowledge. We adopt a basic bilateral multi-issue negotiation scenarios, which has been widely used in the field of negotiation. $\mathcal{V} = \{X_1, \ldots, X_n\}$ is the set of issues in negotiation domain. Each issue X_i is associated with a domain of values $DOM(X_i) = \{x_{i1}, \ldots, x_{im}\}$. The negotiation domain $\Omega = DOM(X_1) \times \cdots \times DOM(X_n)$ is the set of all possible negotiation outcomes o. $o[X]$ denotes the assignment of issue X. The goal of negotiation is to reach an agreement which is an acceptable outcome for all parties. We denote the set of all assignments to $\mathcal{X} \subseteq \mathcal{V}$ by $Asst(\mathcal{X})$. For example, we set $\mathcal{C} = \{C_1, C_2\}$, then $Asst(\mathcal{C}) = Dom(C_1) \times Dom(C_2)$. For the decision maker, preference relationship follows a strict partial order, which is indicated by the symbol $\prec$. If one prefers o_2 to o_1, then it can be denoted by $o_1 \prec o_2$.

Next, we introduce the concepts of CP-nets and preference graph (Boutilier et al., 2004).

Definition 1 (CP-nets). *A CP-net is a directed acyclic graph (DAG) model G, in which nodes represent issues and edges represent the dependencies between issues. If there is a directed edge from issue X_j to issue X_i, then the preference of X_i involves X_j for the user, which is denoted by $Pare(X_i) = \{X_j\}$. Each issue $X_i \in \mathcal{V}$ holds a condition preference table $CPT(X_i)$,*

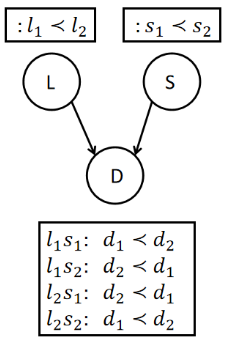


Figure 1: A sample CP-net for travel domain with issues location(L), season(S), duration(D).

which represents the preference ordering of assignments of X for different assignments of $Pare(X_i)$ in the G. If $\forall X_i \in \mathcal{V}$, $DOM(X_i)$ is binary, it is called binary-valued CP-net. If not, i.e., $\exists X_i \in V$, the amount of possible assignments of $DOM(X_i)$ is more than 2, it is called multi-valued CP-net.

The following is an example for a CP-net.

Example 1 (Travel Domain). *Figure 1 illustrates a CP-net that shows Justin's preferences to travel with three issues: Location, Season and Duration. In this example, l_1 denotes Paris and l_2 denotes Hawaii; s_1 denotes Winter and s_2 denotes Summer; d_1 denotes 3days and d_2 denotes 7days. When Justin makes a choice about the duration of the travel, the preference for 3days and 7days is conditional. If Paris and Winter are selected or Hawaii and Summer are selected, he prefers 7days to 3days; If Paris and Summer are selected or Hawaii and Winter are selected, he prefers 3days; Note that, comparing the preference of two outcomes, ancestral values dominate over descendant values. In this example, no matter D takes d_1 or d_2, all outcomes selecting l_2 and s_2 have priority over outcomes selecting l_1 and s_1.*

Definition 2 (Preference Graph.). *Preference graph is a DAG induced from the CP-net G. Nodes in preference graph stand for all outcomes in a negotiation domain, in which the root node presents the worst outcome for the user and the best outcome placed at the leaf node. Each directed edge represents an improving flip. By transitivity, if there exists a path in a preference graph from o_i to o_j, then o_i and o_j are comparable, and $o_i \prec o_j$, otherwise o_i and o_j are non-comparable or called indifferent.*

Figure 2 is a preference graph of the CP-net described in Example 1. An edge from $o_1 = l_1s_1d_1$ to $o_2 = l_1s_1d_2$ in this preference graph denotes an improving flip on $DOM(D)$, meaning that the user prefers o_2 to o_1 ($o_1[D] = d_1$ and $o_2[D] = d_2$).

We will introduce the definition of preference database, which will be used in our algorithms.

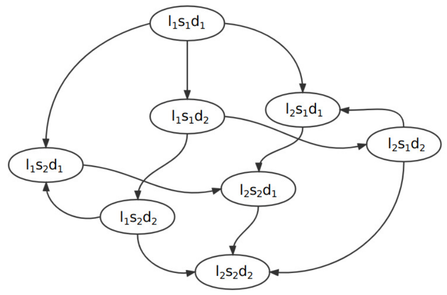


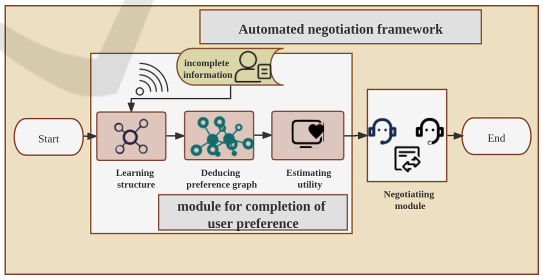
Figure 2: Preference graph induced from Figure 1.

Definition 3 (Preference Database). *Given two outcomes o and o' in a CP-net G, if o and o' are comparable, they make up a comparable pairwise comparison stored in the preference database $\mathcal{P}$. If $o \prec o'$, it denotes (o, o'), otherwise, it denotes (o', o). When all comparable pairwise comparisons derived from the preference graph which is induced from given CP-net are in the preference database, we call it standard preference database. When the set of comparable pairwise comparisons derived from the CP-net learned by given preference database, we call that learned preference database.*

In this section, we outlined the basics required for this paper. In the next section, we will introduce the module we propose and algorithms used in this paper.

4 USER MODELLING

4.1 Module for Preference Completion



Automated negotiation framework
incomplete information
Start
Learning structure
Deducing preference graph
Estimating utility
module for completion of user preference
Negotiating module
End

Figure 3: The automated negotiation framework with module for completion of user preference.

Due to the incomplete information, the agent needs to estimate the user preference model before conducting automated negotiation. We propose a module to complete this task shown in Figure 3. Since we assume that users preferences are represented by CP-nets, the first thing we address is structure learning of CP-nets. The structure of CP-net is learned from a set of pairwise comparisons provided from the user, and then the preference graph is induced from the learned structure. There may be some offers that are not comparable based on the preference graph. However, most negotiation strategies and opponent modelings are based on quantitative preferences leading to a totally ordered set of outcomes. Therefore, we use the appropriate heuristics method (Aydoğan et al., 2015) to transform qualitative preferences into quantitative preferences.

4.2 Structure Learning of CP-nets

Algorithm 1: *learn*(Issue set $\mathcal{V}$, preference database $\mathcal{P}$).

```
Output: a learned CP-net $G$ with respect to $\mathcal{P}$
 1: $G \Leftarrow \emptyset; \mathcal{R} \Leftarrow \emptyset; \mathcal{X} \Leftarrow \mathcal{V}; k \Leftarrow 0$
 2: Add all issues in $\mathcal{V}$ with empty CPT into $G$.
 3: while $k < |\mathcal{V}|$ do
 4:     if $k = 0$ then
 5:         $\mathcal{R} \Leftarrow findRoots(\mathcal{P}, \mathcal{X}, G)$
 6:         $\mathcal{X} \Leftarrow \mathcal{X} \setminus \mathcal{R}$
 7:     else
 8:         $added \Leftarrow extendNetwork(\mathcal{P}, \mathcal{X}, \mathcal{V}, k, G)$
 9:         $\mathcal{X} \Leftarrow \mathcal{X} \setminus added$
10:     end if
11:     if $\mathcal{X} = \emptyset$ then
12:         return $G$
13:     end if
14: end while
15: if $\mathcal{X} \neq \emptyset$ then
16:     $learningWithMaxSAT(\mathcal{P}, \mathcal{X}, \mathcal{R}, G)$
17: end if
18: return $G$
```

We present an algorithm to tackle the problem of structure learning, which extends the algorithm proposed by (Dimopoulos et al., 2009) to better accommodate automated negotiation scenarios. As shown in Algorithm 1, main procedure *learn*() takes a preference database $\mathcal{P}$ provided from user and the issue set $\mathcal{V}$ as input, and outputs an acyclic CP-net that satisfies the standard preference database as much as possible. At first, the algorithm maintains an empty CP-net G and a set $\mathcal{X}$ containing issues to be learned, k denotes the number of provisional parents of issue X, where the process loops until either $k = |\mathcal{V}|$ or the CP-net G is constructed. Since the assignment of ancestor nodes (issues) impact the preferences of descendant issues, accurate derivation of the root issues in CP-nets is crucial. However, the original algorithm is essentially an approximate learning algorithm, which is not accurate enough. For the above reasons, first we extend the original algorithm as follows.

When $k = 0$, we call Algorithm 2 $findRoots()$ to find root issues in the CP-nets. Considering every X in $\mathcal{X}$, X will be a root issue if there exists an *ordering* of values of X that satisfies comparisons set $\{\mathcal{P} \mid o[I] = o'[I]\}$, where I represents a subset of $\mathcal{X}$ except X. That is, the preference ordering of X is not conditioned by any assignments of the other issues. Finally it creates the $CPT(X)$ based on *ordering*. Once the $CPT(X)$ is created, X will be removed from the set $\mathcal{X}$.

Algorithm 2: $findRoots(\mathcal{P}, \mathcal{X}, G)$.

```
Output: issue set $\mathcal{R}$
1: $\mathcal{R} \Leftarrow \emptyset$
2: for $X$ in $\mathcal{X}$ do
3:     $I \Leftarrow \mathcal{X} \setminus X$
4:     if exist an ordering of values of $X$ that satisfies $Comparisons = \{\mathcal{P} \mid o[I] = o'[I]\}$ then
5:         $\mathcal{R}$.add($X$)
6:         $createCPT(X, ordering)$
7:     end if
8: end for
9: return $\mathcal{R}$
```

Algorithm 3: $extendNetwork(\mathcal{P}, \mathcal{X}, \mathcal{V}, k, G)$.

```
Output: issue set $added$
 1: $added \Leftarrow \emptyset$
 2: for $X$ in $\mathcal{X}$ do
 3:     $\mathcal{U} \Leftarrow provisionalParentSet(X, \mathcal{X}, G)$
 4:     $\mathcal{K} \Leftarrow parentCombinations(\mathcal{U}, k)$
 5:     for $K$ in $\mathcal{K}$ do
 6:         $I \Leftarrow \mathcal{X} \setminus (X \cup K)$
 7:         $parentOrNot(K, X, I)$
 8:     end for
 9:     if One of $K \in \mathcal{K}$ is the parent of $X$ then
10:         $added$.add($X$)
11:     end if
12: end for
13: return $added$
```

When $k > 0$, we call $extendNetwork()$ shown in Algorithm 3, traversing X in the set $\mathcal{X}$ to extend the G by completing $CPT(X)$. During the traversing, $provisionalParentSet()$ returns a provisional parent set of X, denoted by $\mathcal{U}$, which guarantees that the learned CP-net is acyclic. $parentCombinations()$ returns a set $\mathcal{K}$ of combinations of k elements selected from $\mathcal{U}$. During iterating through K in $\mathcal{K}$, $parentOrNot()$ decides whether K can be the parent combination of X in G. $parentOrNot()$ examines pairwise comparisons $(k_i x_i i_i, k_j x_j i_j) \in \mathcal{P}$, where $k_i, k_j \in Asst(\mathcal{K})$, $x_i \neq x_j$, $x_i, x_j \in Asst(X)$, $i_i = i_j$, $i_i, i_j \in Asst(I)$, and chooses the suitable clauses in $k_i : x_i \prec x_j$ and $k_j : x_i \prec x_j$ leading to no contradiction among all clauses selected. Taking an example of contradiction, $a_i : b_i \prec b_j$, $a_i : b_j \prec b_k$ and $a_i : b_k \prec b_i$ have been selected, then the $CPT(B)$ is induced as

$a_i : b_i \prec b_j \prec b_k \prec b_i$. The above problem can be converted into Boolean satisfiability problem (SAT). Since the number of clauses in the element of the set are no more than 2, the SAT instance above is the 2-SAT, which could be solved in polynomial time. In this paper, we employ solvers (Morgado et al., 2014; Sörensson and Een, 2005) for solving SAT. Each unit can be separated into two parts to be encoded, one is the assignment of K, and the other one is the relationship between assignment of X. As an example, $a_1 : b_1 \prec b_2$ can be encoded as α_1, $a_1 : b_2 \prec b_1$ can be encoded as $\neg\alpha_1$, and $a_2 : b_1 \prec b_2$ is encoded as β_1. Finally, it solved the 2-SAT and writes into the $CPT(X)$ with the decoded results. We take an example for describing this process.

Example 2. *Consider the issues A, B, C with values* $\{a_1, a_2\}$*,* $\{b_1, b_2, b_3\}$*,* $\{c_1, c_2\}$ *respectively. During the invocation of* $parentOrNot(\{A\}, B, \{C\})$ *(determine whether* $\{A\}$ *is the parent of B with the same assignments of* $\{C\}$*), given pairwise comparisons are as follow:*

(1) $a_1b_1c_1 \prec a_1b_2c_1$, (2) $a_2b_1c_2 \prec a_1b_2c_2$,
(3) $a_1b_2c_1 \prec a_1b_3c_1$, (4) $a_2b_2c_1 \prec a_1b_1c_1$,
(5) $a_1b_1c_2 \prec a_1b_3c_2$, (6) $a_1b_3c_1 \prec a_2b_1c_1$.

Deduce these pairwise comparisons above:

(1) $a_1 : b_1 \prec b_2$, (2) $a_2 : b_1 \prec b_2$ *or* $a_1 : b_1 \prec b_2$,
(3) $a_1 : b_2 \prec b_3$, (4) $a_2 : b_2 \prec b_1$ *or* $a_1 : b_2 \prec b_1$,
(5) $a_1 : b_1 \prec b_3$, (6) $a_1 : b_3 \prec b_1$ *or* $a_2 : b_3 \prec b_1$.

We associate the elements of $\{A\}$ *with component of Boolean variables as follows: the assignment* $a_1 \Rightarrow \alpha$*,* $a_2 \Rightarrow \beta$*. And the relationship* $b_i \prec b_j$ *are associated with subscript. Based on the above rules, the resulting SAT instance is* $(\alpha_1) \wedge (\beta_1 \vee \alpha_1) \wedge (\alpha_2) \wedge (\neg\beta_1 \vee \neg\alpha_1) \wedge (\alpha_3) \wedge (\neg\alpha_3 \vee \neg\beta_3)$*. And the solution of SAT instance above are* $(\alpha_1, \alpha_2, \alpha_3, \neg\beta_1, \neg\beta_3)$*. Considering first three elements* $(\alpha_1, \alpha_2, \alpha_3)$*, they can be decoded as* $a_1 : b_1 \prec b_2$*,* $a_1 : b_2 \prec b_3$ *and* $a_1 : b_1 \prec b_3$ *respectively, which can be deduced an ordering* $a_1 : b_1 \prec b_2 \prec b_3$ *and written into the* $CPT(B)$ *with* $a_1 : b_1 \prec b_2 \prec b_3$*. The remaining only have 2 clauses* $a_2 : b_2 \prec b_1$*,* $a_2 : b_3 \prec b_2$*. According to transferability, it could still deduce* $a_2 : b_3 \prec b_2 \prec b_1$*. Therefore,* $\{A\}$ *is the set of parent of B, and* $CPT(B)$ *is created.*

The second extension is described as follows: Firstly, there may be such a situation that more than one K satisfies being the parent of X, while only one K can be parent of X, in principle. To cope with this situation, we consider the count of clauses, K with maximum count of clauses will be the parent of X. Secondly, once the ordering is incomplete, it is necessary to make a completion. Therefore, We use the method *BordaCount* (Emerson, 2013) to determine the incomplete position in the ordering.

Algorithm 4: $learningWithMaxSAT(\mathcal{P}, \mathcal{X}, \mathcal{R}, G)$.

```
1: for $X$ in $\mathcal{X}$ do
2:     for $R$ in $\mathcal{R}$ do
3:         $fit = Fitness(X, R)$
4:     end for
5:     $R_i = argmax(fit)$
6:     $R_i$ is a parent of $X$, and create $CPT(X)$
7: end for
8: return
```

The following is the third extension of the original algorithm. Back to the main procedure, if there are issues remaining in the set $\mathcal{X}$ after completion of the traversal on K, original algorithm will return *False* to indicate failure of learning, which is not available for automated negotiation scenarios. Therefore, we calls $learningWithMaxSAT()$ shown in Algorithm 4 to learning remaining issues. This function is similar to the $extendNetwork()$ shown in Algorithm 3. For less complexity, the parents of remaining issues $X \in \mathcal{X}$ is considered from $\mathcal{R}$, and only one issue could be the parent of X. The method that decides whether $R \in \mathcal{R}$ can be the parent of X translates the problem into a SAT instance. But we only need to solve for its maximum satisfiable (Max-Sat) solution. We choose the one with the maximum satisfaction rate as the parent of X and create $CPT(X)$. When all $CPT(X), X \in \mathcal{V}$ have been created, the CP-net G is completed and main procedure $learn()$ returns G.

4.3 Transformation of User Preferences

Given a CP-net, the ordering of the relationships among the outcomes is a partial order sequence, so there will be non-comparable pairwise comparisons in most of the time. In order to better use various strategies in automated negotiation scenarios, transforming qualitative preferences into quantitative preferences is useful that agent can get a total ordering of outcomes and know all utility of outcomes in that domain. The method Taxonomic Heuristic (TH) proposed by (Aydoğan et al., 2015) is used to handle this problem. We briefly summarized below. Considering a preference graph induced from a given CP-net, the root node keeps the worst outcome and the leaf node keeps the best outcome for the user. That is, given two outcomes, the dominance between them is determined by their depth (length of the longest path from the root node to this node). The higher depth one is assigned by higher utility.

$$U(o_i) = Max(U(Parent(o_i))) + ran \qquad (1)$$

According to Equation (1) , the outcome's utility is equals the maximum utility among its parents plus a random value $ran \in (0,1]$, which ensures its utility is higher than its ancestors' and maximize leaf node's teutility as well. Although the randomized approach results in an unstable ordering among non-comparable pairwise comparisons, it provides a reference to user for non-comparable pairwise comparisons, and comparable pairwise comparisons are not affected. Hence, the overall impact is not significant.

5 EXPERIMENTS

In order to demonstrate the effectiveness of our module, the experiments were divided into two parts: one is to evaluate the performance of structure learning and the other is to demonstrate the feasibility of our module in negotiation scenarios.

5.1 Performance of Structure Learning

To evaluate the performance of structure learning, we consider experiments of learning CP-nets with 7 issues. We consider 300 randomly generated CP-nets with the number of edges from 1 to 10, where each assignment of edges randomly generates 30 CP-nets and we ensure all CP-nets are acyclic and multi-valued. To test learning performance with 20% of standard database (including noise data), the metrics we selected are the similarity of structure, error and accuracy of learning root issues. The formula of similarity of structure is defined as follows:

$$Similarity = \sum_{n=1}^{|\mathcal{V}|} sim(X_i), \tag{2}$$

where

$$sim(X_i) = \begin{cases} score(X_i) & \text{if } X_i \text{ is root issue,} \\ \dfrac{r_p/(e_p+r_p)}{|\mathcal{V}|} & \text{otherwise,} \end{cases} \tag{3}$$

$$score(X_i) = \begin{cases} \dfrac{1}{|\mathcal{V}|} & \text{if } X_i' \text{ is root issue,} \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

In Equation (3), X_i and X_i' denote an issue in learned CP-net and original CP-net, respectively. r_p denotes the number of parent of X_i learned accurately and e_p denotes the number of parent of X_i learned by error. When $Similarity = 1$, it means the learned structure is exactly the same as the given structure. The *error* is the difference between learned preference database and standard preference database. Equation (5) describes the formula for calculating *error*:

$$error = \frac{num(\text{incorrect pairwise comparisons})}{num(\text{standard preference database})}. \tag{5}$$

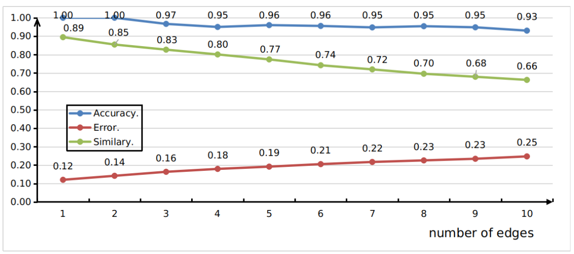


Figure 4: This results of learning CP-nets in different edges.

$num(\text{incorrect pairwise comparisons})$ includes the pairwise comparisons that are in standard preference database but not in learned preference database and are contrary to the facts by standard preference database.

The experiment results of structure learning are shown in Figure 4. The horizontal coordinate indicates the edge number in a CP-net. Considering the dominance of the CP-nets, the performance of learning the root issues accurately may be more important than that of learning other issues in negotiation. The blue line denotes the probability of accurately learning all root issues in the experiment. As we expected, the performance of learning decreases with increasing structural complexity. Results show that the structure of the CP-net can be roughly learned with given data.

5.2 Application in Negotiation Scenario

To demonstrate the effectiveness of our framework, we applied our module to automated negotiation scenarios and tested its performance. We randomly generated five domains with five issues for experiment. The preference profiles are generated with CP-nets model for two dummy users by given domains, but the agents just know 20% of standard preference database. Each domain is used for ten epochs of negotiation. For fairness, the agents will hold preference profile of counterpart to compete repeat, i.e., two times of negotiation per epoch. We investigate three test cases. In each case, two agents (A and B) with the same bidding strategy compete with each other. There are three subcases in each case. In the subcase 1, both agents use our module (denote as *our.vs.our.*). In the subcase 2, agent A uses our module while agent B does not. In the subcase 3, both two agents estimate user model randomly. The classic time dependent tactics proposed by (Faratin et al., 1998) are used for each case. Its curve of utility thread and equation are shown in Figure 5.

The utility threshold varies over time. When $\beta < 0$, it adopts an aggressive static called *Boulware*; When $\beta = 0$, the function curve is a straight line and the static is called *Linear*; When $\beta > 0$, the agent

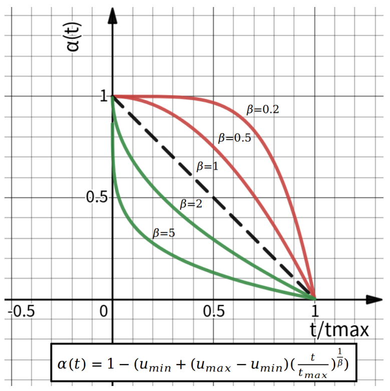


Figure 5: The curve of utility thread.

Table 1: The results of negotiation in the case *Boulware*.

		Boulware			
		uti.A	uti.B	round	soc.
	our. vs. our.	0.46	0.48	451.20	0.94
Dom.1	our. vs. bas.	**0.61**	0.41	358.35	1.02
	bas. vs. bas.	0.43	0.46	261.50	0.88
	our. vs. our.	0.57	0.61	521.70	1.18
Dom.2	our. vs. bas.	**0.82**	0.64	395.40	1.45
	bas. vs. bas.	0.50	0.56	342.90	1.05
	our. vs. our.	0.62	0.61	509.90	1.23
Dom.3	our. vs. bas.	**0.76**	0.34	368.60	1.10
	bas. vs. bas.	0.38	0.51	301.60	0.89
	our. vs. our.	0.62	0.60	419.15	1.22
Dom.4	our. vs. bas.	**0.78**	0.33	396.35	1.11
	bas. vs. bas.	0.51	0.42	272.00	0.93
	our. vs. our.	0.68	0.66	276.35	1.35
Dom.5	our. vs. bas.	**0.73**	0.60	372.75	1.34
	bas. vs. bas.	0.43	0.45	302.85	0.88

with this static called *Conceder* will make concessions quickly. Each negotiation is limited to 1000 rounds, i.e., $t_{max} = 1000$. The range of utility of each outcome is normalized at $[0, 1]$, i.e., $u_{min} = 0$ and $u_{max} = 1$. The protocol used in this simulation experiments is alternating-offers protocol. Both sides take turns to alternate bids until time runs out or one of agent offers to terminate the negotiation. If one of agent offers to accept opponent's offer, the agreement is reached with the last bid offered.

The results of average utility, average round and average social welfare with $\beta = 0.5$, $\beta = 1$ and $\beta = 2$ are shown in Table 1, Table 2 and Table 3, respectively. It is intuitive that in any subcase, the use of *Conceder* results in the minimum rounds per negotiation. The results of subcase 2 in all cases show that the average utility obtained by agent A is always higher than that obtained by agent B, which verifies the effectiveness of our model for user modelling. By comparing results in subcase 1 and subcase 3, average social welfare obtained by agents in subcase 1 is more than that in subcase 3. The negotiation in subcase 1 takes more time (average round) than that in subcase 3. Combining two analyses above, higher social welfare was obtained despite the decline in utility thresholds with the passage of time, which reflects that agents with our model are willing to spend more time on negotiation with opponent towards higher utility.

Table 2: The results of negotiation in the case *Conceder*.

		Conceder			
		uti.A	uti.B	round	soc.
	our. vs. our.	0.46	0.48	46.95	0.94
Dom.1	our. vs. bas.	**0.57**	0.38	39.25	0.94
	bas. vs. bas.	0.42	0.44	9.85	0.86
	our. vs. our.	0.58	0.63	73.70	1.21
Dom.2	our. vs. bas.	**0.69**	0.57	41.35	1.25
	bas. vs. bas.	0.47	0.48	16.90	0.94
	our. vs. our.	0.64	0.56	59.85	1.21
Dom.3	our. vs. bas.	**0.75**	0.31	26.95	1.06
	bas. vs. bas.	0.46	0.40	9.50	0.85
	our. vs. our.	0.61	0.61	30.35	1.21
Dom.4	our. vs. bas.	**0.71**	0.43	29.80	1.14
	bas. vs. bas.	0.46	0.55	12.30	1.01
	our. vs. our.	0.67	0.67	11.75	1.34
Dom.5	our. vs. bas.	**0.70**	0.59	33.25	1.28
	bas. vs. bas.	0.47	0.48	12.05	0.95

Table 3: The results of negotiation in the case *Linear*.

		Linear			
		uti.A	uti.B	round	soc.
	our. vs. our.	0.49	0.51	218.40	1.00
Dom.1	our. vs. bas.	**0.58**	0.42	369.45	1.00
	bas. vs. bas.	0.44	0.49	76.10	0.93
	our. vs. our.	0.64	0.64	266.95	1.28
Dom.2	our. vs. bas.	**0.77**	0.61	380.45	1.37
	bas. vs. bas.	0.43	0.48	103.05	0.91
	our. vs. our.	0.59	0.63	258.50	1.22
Dom.3	our. vs. bas.	**0.73**	0.35	404.95	1.08
	bas. vs. bas.	0.43	0.42	73.05	0.85
	our. vs. our.	0.58	0.62	168.00	1.20
Dom.4	our. vs. bas.	**0.75**	0.38	401.90	1.13
	bas. vs. bas.	0.45	0.59	77.85	1.04
	our. vs. our.	0.66	0.68	96.95	1.34
Dom.5	our. vs. bas.	**0.72**	0.62	360.00	1.34
	bas. vs. bas.	0.45	0.49	102.35	0.95

6 CONCLUSION

This paper studied how to complete user preference with incomplete information in automated negotiation scenarios where user preferences are represented by CP-nets. Firstly, we extended the method of learning CP-nets (Dimopoulos et al., 2009) to reveal the dependencies among issues, in which the extensions including (1) special handling of the root issues to learn more accurately; (2) optimising the choice of provisional parents during learning structure; and (3) avoiding failing to learn CP-nets. Secondly, we estimate utility of offers by TH after completion of CP-nets, making the preference representations of CP-nets can be applied in automated negotiation. Finally, we experimentally demonstrated the feasibility of our module for completion of user preference in negotiation. Much more could be done in the future. For example, due to the structures of CP-nets are interpretative, users can easily express their attitude towards the effect of preference completion, then how to make agents interact with users efficiently to improve the accuracy of completion of preference is one of the most important problems should be address in the future.

ACKNOWLEDGEMENTS

The works described in this paper are supported by the National Natural Science Foundation of China under Grant Nos. 62006085, 61772210 and U1911201; Guangdong Province Universities Pearl River Scholar Funded Scheme (2018); Project of Science and Technology in Guangzhou in China under Grant Nos. 202007040006 and 202102020948; Natural Science Foundation of Guangdong Province in China under Grant No. 2018A030310529; Project of Department of Education of Guangdong Province in China under Grant No. 2017KQNCX048.

REFERENCES

Amor, N. B., Dubois, D., Gouider, H., and Prade, H. (2016). Graphical models for preference representation: An overview. In *Scalable Uncertainty Management*, pages 96–111.

Aydoğan, R., Baarslag, T., Hindriks, K. V., Jonker, C. M., and Yolum, P. (2015). Heuristics for using CP-nets in utility-based negotiation without knowing utilities. *Knowledge and Information Systems*, 45(2):357–388.

Aydoğan, R. and Yolum, P. (2010). Effective negotiation with partial preference information. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 1605–1606.

Baarslag, T. (2016). *Exploring the strategy space of negotiating agents: A framework for bidding, learning and accepting in automated negotiation*. Springer.

Baarslag, T. and Gerding, E. H. (2015). Optimal incremental preference elicitation during negotiation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, page 3–9.

Baarslag, T. and Kaisers, M. (2017). The value of information in automated negotiation: A decision model for eliciting user preferences. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 391–400.

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

Dimopoulos, Y., Michael, L., and Athienitou, F. (2009). Ceteris paribus preference elicitation with predictive guarantees. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 1890–1895.

Emerson, P. (2013). The original borda count and partial voting. *Social Choice and Welfare*, 40(2):353–358.

Faratin, P., Sierra, C., and Jennings, N. R. (1998). Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems*, 24(3-4):159–182.

Goldsmith, J., Lang, J., Truszczynski, M., and Wilson, N. (2008). The computational complexity of dominance and consistency in CP-nets. *Journal of Artificial Intelligence Research*, 33:403–432.

Haddawy, P., Ha, V., Restificar, A., Geisler, B., and Miyamoto, J. (2003). Preference elicitation via theory refinement. *The Journal of Machine Learning Research*, 4:317–337.

Jennings, N. R., Faratin, P., Lomuscio, A. R., Parsons, S., Wooldridge, M. J., and Sierra, C. (2001). Automated negotiation: Prospects, methods and challenges. *Group Decision and Negotiation*, 10(2):199–215.

Kumar, S. and Mastorakis, N. E. (2009). A utility based, multi-attribute negotiation approach for semantic web services. *WSEAS Transaction on Computers*, 8(11):1733–1748.

Lafage, C. and Lang, J. (2000). Logical representation of preferences for group decision making. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning*, pages 457–468.

Liu, Z., Zhong, Z., Li, K., and Zhang, C. (2018). Structure learning of conditional preference networks based on dependent degree of attributes from preference database. *IEEE Access*, 6:27864–27872.

Morgado, A., Dodaro, C., and Marques-Silva, J. (2014). Core-guided maxsat with soft cardinality constraints. In *International Conference on Principles and Practice of Constraint Programming*, pages 564–573. Springer.

Sanchez-Anguix, V., Tunalı, O., Aydoğan, R., and Julian, V. (2021). Can social agents efficiently perform in automated negotiation? *Applied Sciences*, 11(13):6022.

Sörensson, N. and Een, N. (2005). Minisat v1. 13-a sat solver with conflict-clause minimization. *SAT*, 2005(53):1–2.

Towell, G. G. and Shavlik, J. W. (1994). Knowledge-based artificial neural networks. *Artificial intelligence*, 70(1-2):119–165.

Tsimpoukis, D., Baarslag, T., Kaisers, M., and Paterakis, N. (2018). Automated negotiations under user preference uncertainty: A linear programming approach. In *Agreement Technologies - 6th International Conference, AT 2018, Revised Selected Papers*, pages 115–129.