Web Service Selection for Multiple Agents with Incomplete Preferences

Hongbing Wang, Jie Zhang†, Cheng Wan, Shizhi Shao, Robin Cohen†, Junjie Xu, and Peicheng Li
School of Computer Science and Engineering, Southeast University, Nanjing, China
†School of Computer Science, University of Waterloo, Waterloo, Canada
{hbw@seu.edu.cn}

***Abstract*—A qualitative way is desirable for web service selection according to agents' preferences on non-functional Quality of Service (QoS) attributes of services. However, it is challenging when the decision has to be made for multiple agents with preferences on attributes that may be incomplete. In this paper, we first use a qualitative graphical representation tool called CP-nets to describe preference relations in a relatively compact, intuitive and structured manner. We then propose algorithms based on this representation to select web services for multiple agents despite the presence of incompleteness in their preference orderings. Our experimental results indicate that this method can always obtain some optimal outcomes which closely satisfy all agents.**

I. Introduction

Web service selection has drawn more and more attention [1], [2], [3]. The service that satisfies an agent the most among a set of services needs to be selected based on the agent's preferences on non-functional QoS attributes of services. In most existing solutions, an utility function is used to represent an agent's preferences, which is a powerful quantitative approach to knowledge representation. In many cases, it is however desirable to assess preferences in a qualitative rather than quantitative way because a quantitative approach may induce errors from agents in identifying their utilities and thus make wrong selection of services. In addition, a decision may have to be made for a group of agents with different preferences on QoS attributes of web services [4]. Their preferences may also be incomplete. Incompleteness of preferences represents an absence of knowledge about the relationship between certain pairs of outcomes. It arises naturally when we have not fully elicited agents' preferences or when agents have privacy concerns which prevent them from revealing their complete preference orderings. It then becomes difficult to comprehensively consider all the agents' preferences and select services which satisfy them the most.

In this paper, we use CP-nets [5] to represent qualitative preference relations in a relatively compact, intuitive and structured manner under conditional ceteris paribus (all else being equal) preference statements. Based on this representation, we propose a preference reasoning algorithm to first construct a derivation tree from a CP-net for each agent and then generate all service patterns for this agent. We also rank the service patterns by voting semantics [6]. Finally, we merge ranked service patterns for all agents and select a set of services that satisfy these agents the most. The processes of service selection for multiple agents with incomplete preferences are demonstrated by a concrete example, and the performance is also evaluated by a wide set of artificially generated QoS attributes of services and values of attributes. Our method is able to accurately select the most optimal services for agents in different simulation scenarios, within acceptable execution time.

II. An Example Scenario

In a typical scenario of service selection between a group of users, each user describes her preferences. The agent acting on behalf of the user will identify the relevant services that satisfy this user the most. The agent will also communicate and negotiate with other agents to reach an agreement on services that closely satisfies all these users.

A motivating real life example for our work is the field of enterprise information management. In this field, the most widely used application by different departments is probably the data storage and access service. These services need to meet the needs of different departments. For example, a company's branches need to choose a proper data storage and access service when they conduct joint marketing activities. Each branch expresses its preferences over QoS attributes of services. The branch A may prefer security over other attributes (i.e. response time and price). Branch B may be concerned more with the attribute of response time. Other branches may prefer some other quality attributes (e.g. the platform). If no branch can persuade other branches, no service can satisfy all these branches. In addition, some branches may express their preferences on only a part of the attributes. We focus on service selection for finding services that closely satisfy these branches.

III. Preliminaries

We begin with a brief outline of relevant notions from decision theory, CP-nets introduced by Boutilier et al. [5], and mCP-nets and voting semantics proposed by Rossi et al. [6], which are the fundamental concepts and methods for the proposal of our algorithms of service selection for multiple agents with incomplete preferences.

A. Preference Logic

Assume that the world can be in one of a set of states S, and in each state s there are a number of actions A_s

that can be performed. Each action in one state denotes a specific outcome. The set of all outcomes is denoted by O. A preference ranking is a total preorder over the set of outcomes: $o_1 \succeq o_2$ means that outcome o_1 is equal to or more preferred than outcome o_2; $o_1 \succ o_2$ means that outcome o_1 is strictly more preferred than outcome o_2; while $o_1 \sim o_2$ denotes that the decision maker is indifferent between o_1 and o_2.

Assume a set of variables (attributes) $V = \{X_1, ..., X_n\}$ with domains $D(X_1), ..., D(X_n)$. An assignment x of values to a set $X \subseteq V$ of variables (also called an instantiation of X) is a function that maps each variable in X to an element of its domain: if $X = V$, x is a complete assignment, otherwise x is a partial assignment [4]. We denote by $Asst(X)$ the set of all assignments to X. If x and y are assignments to disjoint sets X and Y ($X \cap Y = \emptyset$), respectively, we denote the combination assignment set of x and y by xy. For any outcome o, we denote by $o[X]$ the value $x \in D(X)$ assigned to variable X by that outcome. A subset of variables X is preferentially independent of its complement $Y = V - X$ iff for all $x_1, x_2 \in Asst(X)$ and $y_1, y_2 \in Asst(Y)$, we have: $x_1y_1 \subseteq x_2y_1$ iff $x_1y_2 \subseteq x_2y_2$. That is, the structure of the preference relation over assignments to X does not change and can be assessed as other attributes vary. Let X, Y, and Z be a partition of V into three disjoint non-empty sets. X is conditionally preferentially independent of Y given an assignment $z \in Asst(Z)$ iff for all $x_1, x_2 \in Asst(X)$ and $y_1, y_2 \in Asst(Y)$, we have: $x_1y_1z \subseteq x_2y_1z$ iff $x_1y_2z \subseteq x_2y_2z$. In other words, if X is conditionally preferentially independent of Y for all $z \in Asst(Z)$, then X is conditionally preferentially independent of Y given the set of variables Z.

B. CP-nets

CP-nets introduced by Boutilier et al. [5] is a tool for compactly representing qualitative preference relations under the ceteris paribus assumption. A CP-net over variables $V = \{X_1, ..., X_2\}$ is a directed graph G over $X_1, ..., X_2$ whose nodes are annotated with conditional preference tables CPT(X_i) for each $X_i \in V$. Each conditional preference table CPT(X_i) associates a total order $\succ_u^i$ with each instantiation u of X_i's parents $Pa(X_i) = U$. For each variable X_i, we ask the user to identify a set of parent variables $Pa(X_i)$ that can affect her preference over various values of X_i. Formally, given $Pa(X_i)$ we have that X_i is conditionally preferentially independent of $V - (Pa(X_i) \cup \{X_i\})$. Given this information, we ask the user to explicitly specify her preferences over the values of X_i for all instantiations of the variable set $Pa(X_i)$ to generate the CP-net and CPT.

C. mCP-nets

We first introduce partial CP-nets as a CP-net in which certain attributes may not be ranked. This implies that the agent is indifferent to the values of these attributes. We put together several partial CP-nets to represent the incomplete preferences of multiple agents. A mCP-net [6] is a set of m partial CP-nets which may share some attributes, such that every attribute is ranked by at least one of the partial CP-nets. An outcome for a mCP-net is an assignment of the values to the attributes in all the partial CP-nets in their domains.

D. Voting Semantics

We reason about a mCP-net by querying each partial CP-net and then merging the results, which can be seen as each agent "voting" whether an outcome dominates another. There are five different voting semantics: Pareto, Majority, Max, Lex, and Rank [6]. In Pareto, outcomes are often incomparable. Majority and Max are the weaker criteria, and many agents often vote in favor or for incomparability. In Lex, agents are ordered by their importance. In the Rank semantic, each agent ranks each outcome. It is also embedded by Rossi et al. [6] in the context of mCP-nets, but they did not provide actual algorithm designs. We extend and implement the Rank semantic to address multiple agents' preferences on QoS attributes in web service selection, by considering the case where agents may have implement preferences and agents may be weighted differently in making decisions on services.

IV. Preference Representation and Reasoning

We first describe the representation for incomplete preferences of an agent using CP-nets. The example scenario in Section II will be used throughout the current section. We then present the processes and algorithms of our reasoning about incomplete preferences of multiple agents for service selection.

A. Representing Incomplete Preference

Assume that the data storage and access service of a company can be described by a number of QoS attributes, including Platform (**A**: a file system or a database), Security (**B**: low or high level), Response_time (**C**: 0 - 100ms), Price (**D**: \$0 - \$100), and Location (**E**: New York, Toronto or London). Let $V = \{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}\}$ be the set of the five attributes.

Definition 1: ***Attribute Constraint****:* is used to describe the constraints on quality attributes defined by agents. For example, an agent may define constraints on Security as b_1: low and b_2: high.

Definition 2: ***Preference Statement****:* is used to describe a preference about one quality attribute. For example, an agent may prefer high level security for a service. The preference statement is $b_2 \succ b_1$.

If an agent X proposes its incomplete preference sequence: Platform $\succeq$ Response_time $\succeq$ Location, the partial CP-net is shown in Figure 1. The arrows denote that one attribute dominates another. This partial CP-net consists of only three variables **A**, **C**, and **E** because the agent does not provide the full preference information. The agent has an unconditional preference on Platform, and it prefers a file system for storing data. In this example, no matter which preference statement is met between a_1 and a_2, c_1 is always better than c_2. The agent's preference on Location, however, depends on Response_time. For example, if the response time is longer than 50ms, agent X is indifferent between e_2 and e_3, but e_1 is less preferred than e_2 and e_3 by X.

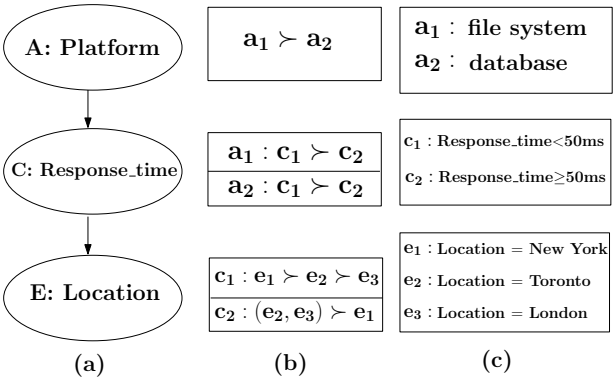

Fig. 1. (a, b) CP-net for Agent X, (b) CPT, (c) Preference Statements

B. Service Selection Processes

The service selection processes are as follows. For each agent, we first transform its CPT of preference description to a derivation tree. We then generate a set of service patterns (called aSet) that satisfy the agent's preferences. The rank value of each item in the aSet will be computed based on the Rank semantic. The total rank values of each service pattern will then be computed by merging rank values of service patterns from each agent. The available services will be divided into different patterns according to their quality attribute values. Finally, the services that match the best patterns will be chosen for all agents.

C. Preference Reasoning Algorithms

We describe the algorithms that transform CP-nets to derivation trees, generate service pattern sets from the derivation trees, rank service patterns in each set, and merge rank values of service patterns for service selection. We also provide complexity analysis for these algorithms that will also be confirmed by the experimental results in Section VI.

1) Derivation Tree: Based on an agent's preferences represented in a CP-net, an algorithm is proposed to generate a derivation tree. The agent's preference statements of the attributes that are conditioned by a smaller number of or no other attributes will be in the upper level of the tree. Nodes are conditioned by their parents. In the same level of the tree, the preference statements (nodes) will be listed according to their priorities from left to right. For the children of the same parent, the left children are more preferred than the right children. The pseudo-code summary for generating a derivation tree is shown in Algorithm 1, and the derivation tree for agent X's preferences in Figure 1 is shown in Figure 2. a_1 and a_2 are in the upper level of the tree because attribute **A** is unconditional. a_1 is a left child because it is more preferred.

2) Service Patterns and Rank Semantic: Service patterns of an agent are generated from its derivation tree. We then use the Rank semantic method to separate the service patterns into different sets in order to find the best patterns.

Definition 3: ***Service Pattern****:* is a combination of attribute constraints for all attributes of QoS, i.e. $a_1 b_1 c_1 d_1 e_1$.

Input : CP-net of an agent; root of the tree Tr
Define : RUNTIME: allowed runtime of program
Output: Derivation tree Tr of the agent

```
1  Add constraints of unconditional attribute in CP-net as
   children of root, left to right according to priority;
2  foreach other attribute preference in CP-net do
3      Find attribute constraints on condition of leftmost
       node in upper level;
4      Add as children, left to right according to priority;
5  foreach node m ≠ leaf, from bottom up do
6      if preference on condition of m = left node n then
7          Duplicate subtree of n as m's subtree
8      else
9          Find constraints on condition of m;
10         Add as children, left to right based on priority;
11     if run time > RUNTIME then
12         break;
```

Algorithm 1: Generating Derivation Tree

Definition 4: ***Rank of Service Pattern****:* is a number expressing the degree of the service pattern to meet agents' preferences.

Definition 5: ***aSet****:* short for an analogous set is a set of items ordered according to their values. An item in an aSet can also be an aSet called Sub_aSet.

Definition 6: ***Service Pattern aSet****:* is an aSet whose items are service patterns ordered based on their rank values.

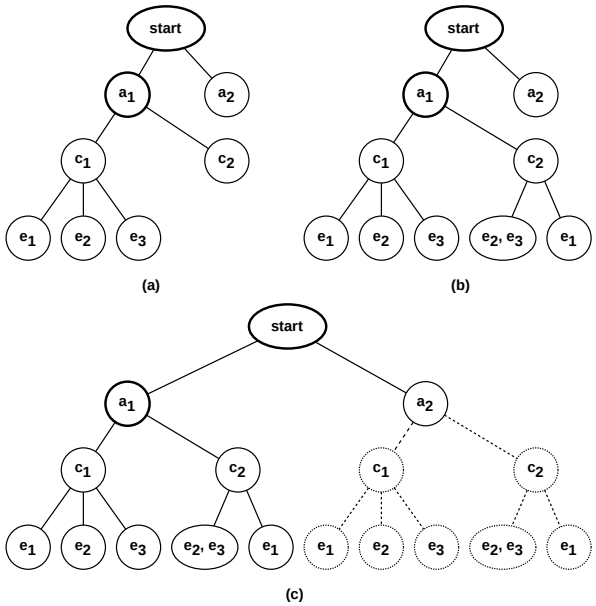

Fig. 2. Derivation Tree of CPT for Agent X

As shown in Figure 2, there may be many paths from the top to the bottom of the derivation tree. The left paths are more preferred than the right ones (see Algorithm 1). The combination of all paths will become an aSet T. For

example, the aSet from Figure 2 is $T = \{T_1, T_2, ..., T_{10}\}$, and $T_1 = a_1c_1e_1$, $T_2 = a_1c_1e_2$ and etc. If the agent's preferences are incomplete, the attributes that are not in the CP-net (called remaining attributes) should be added to each path to become a service pattern. If the remaining attributes have their own priority order, the order will be added into patterns for integrity and accuracy. Suppose that the attributes of Security (**B**) and Price (**D**) have strict priorities, $b_1 \succ b_2$ and $d_1 \succ d_2$. T_1 becomes a service pattern Sub_aSet as $T_1 = \{\{a_1c_1e_1b_1d_1\} \succ \{a_1c_1e_1b_1d_2,\ a_1c_1e_1b_2d_1\} \succ \{a_1c_1e_1b_2d_2\}\}$. The pseudo code for generating all service patterns is shown in Algorithm 2.

Input : Preference statements, derivation tree Tr
Output: Service pattern aSet T

```
int i = 0; aSet T = NULL;
foreach path in tree Tr from left to right do
    i = i + 1;
    Combine preference statements in all levels;
    Sub_aSet T[i] = NULL;
    Add the combination into T[i] as an item;
    Add T[i] into aSet T;
foreach remaining attribute do
    if attribute value has strict priority then
        Order preference statements by priority;
    Add them into each Sub_aSet in T;
```

Algorithm 2: Generating Service Pattern aSet

Input : Service pattern aSet T, setWeight
Output: Ranked service patterns in aSet

```
int i = 0;
foreach Sub_aSet in T by priority do
    i = i + 1; j = 0;
    foreach service pattern in T[i] by priority do
        j = j + 1;
        Rank(pattern[j]) = i × setWeight + j;
```

Algorithm 3: Ranking Service Patterns in aSet

Because T_1 contains the best service patterns, we define the initial rank of items in T_1 as a number *setWeight*. The value of setWeight is larger than the product of the size of aSet T and the maximum size of the Sub_aSets of T. In the example in Section V, we set setWeight to be 100. The initial rank of the items in T_1 is then 100. The rank of each pattern in T_1 will be added by its order number. For example, Rank($a_1c_1e_1b_1d_1$) = 101, Rank($a_1c_1e_1b_1d_2$) = 102, and Rank($a_1c_1e_1b_2d_1$) = 102. $a_1c_1e_1b_1d_2$ and $a_1c_1e_1b_2d_1$ have the same rank value because they are in the same Sub_aSet of T_1. The pseudo code for ranking service patterns is shown in Algorithm 3.

3) Merging and Selection: After we have an aSet for each agent and service patterns with rank values in each aSet, we now merge the aSets for all agents and compute the total rank values of each service pattern in the merged aSet. Service selection for the agents will depend on these total rank values of service patterns.

Input : Total number of agents M, ranked service pattern aSet for each agent, weight of each agent (optional)
Output: Service patterns ordered by total rank

```
if no weight of agents provided then
    Weight of each agent = 1/M;
Initialize Rank_total of each pattern = 0;
Compute Rank_total using Equation 1;
Order service patterns according to Rank_total
foreach set of patterns with same Rank_total do
    Compute weighted distance using Equation 2;
    Order them according to weighted distance;
```

Algorithm 4: Merging Service Patterns

Input : Service patterns ordered by total rank, a set of available services
Output: The best services

```
foreach available service do
    Find matching pattern based on attribute values;
boolean flag = true;
while flag do
    foreach ordered service pattern do
        Find services that match the pattern;
        if found matched services then
            flag = false;
Return the found services;
```

Algorithm 5: Service Selection

Different organizational forms exist among agents. If all the agents have equal weight, the same pattern's rank values of the agents will simply be added. The pattern with the highest total rank is the best pattern for all the agents. However, agents may be weighted differently in making decision on services. In this case, the total rank of service pattern j (j is the id of the pattern) can be computed as follows:

$$Rank\_total(j) = \sum_{i=1}^{M} [Rank_i^j \times Weight(i)] \qquad (1)$$

where i is the id of an agent, M is the total number of agents, and $Rank_i^j$ is agent i's rank for service pattern j. The total weight of all agents is normalized to 1. If some patterns have the same total rank, each of these patterns will be ordered by its weighted distance to the mean of all rank values for this pattern. Note that the mean value can be calculated by Equation 1 because the total weight is 1. The weighted distance can then be computed as follows:

$$Dis(j) = \sum_{i=1}^{M} [Weight(i) \times |Rank_i^j - Rank\_total(j)|^2] \quad (2)$$

The pseudo code for merging service patterns and computing their total rank values is shown in Algorithm 4.

Based on the total rank values of all service patterns, service selection is done by matching available services with service patterns. The services that match the patterns with the smallest rank values will be returned to all agents. The pseudo code for service selection is shown in Algorithm 5.

4) Algorithm Complexity Analysis: Assume that the number of attributes included in a CP-net is $attNum$ and the maximum number of attribute constraints in one preference statement is $maxPS$. The computational complexity of line 1 in Algorithm 1 is lower than $O(maxPS)$. The computational complexity of the first and second For loops is lower than $O(attNum \times maxPS)$. Because the duplicate operator costs only $O(1)$ time, the computational complexity of the second For loop will decrease dramatically depending on the ratio of duplicates. Furthermore, this algorithm gives the agent the right to set the runtime limit. If the runtime limit is reached, the algorithm will stop and return the left part of the derivation tree which represents the comparatively more preferred combination of attributes. Some methods of cutting derivation trees may also be considered in our future work.

Suppose that the number of all service quality attributes is $aNum\_all$ and the maximum number of attribute constraints of one attribute is $maxAC$. The worst case complexity of Algorithms 2 and 3 is $O(aNum\_all^{maxAC})$. But, if the attribute constraints of agents remain stable over time, this step can be processed in advance to improve the efficiency.

If the size of aSet T is $sizeT$ and we adopt the bin sort method to order service patterns, the complexity of Algorithm 4 is $O(M \times sizeT \times log(sizeT))$. The complexity of Algorithm 5 is dependent on $sizeT$ and the number of available services for selection. Our experimental results in Section VI indicate that the runtime of our algorithms is tolerable for a large number of available services.

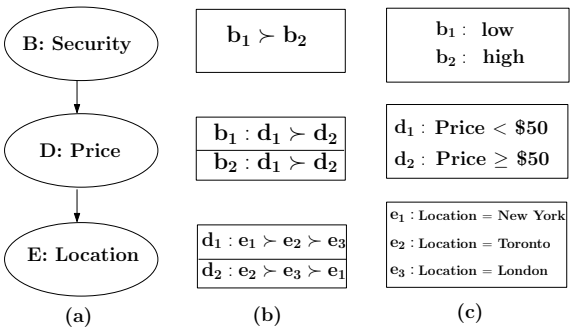

Fig. 3. (a,b) CP-net for Agent Y, (b) CPT, (c) Preference Statements

V. Example Demonstration

In this section, we follow up with the example scenario described in Section II and the preferences of agent X in Section IV-A, and add preferences of other two agents, Y and Z. We demonstrate the processing results of our service pattern generation, ranking and merging algorithms.

The preferences of agents Y and Z are represented by CP-nets, as shown in Figures 3 and 4. Agent Y proposes its incomplete preference sequence: Security $\succeq$ Price $\succeq$ Location, and agent Z's preference sequence is more complicated: Platform $\succeq$ Security $\sim$ Response_time $\succeq$ Location. Their preference statements are expressed in Figure 3 (b) and Figure 4 (b). As described in Section IV-C, we first generate derivation trees for these two CP-nets respectively, using Algorithm 1. From each derivation tree, we use Algorithm 2 to generate a service pattern aSet for each agent, and rank these service patterns using Algorithm 3. The 5 best Sub_aSets for each agent from T_1 to T_5 are listed in Tables I, II and III.

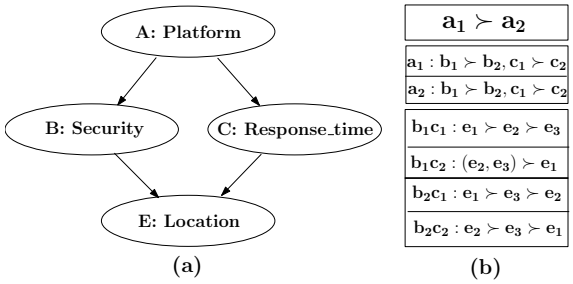

Fig. 4. (a,b) CP-net for Agent Z, (b) CPT

TABLE I
Ranked Service Pattern aSet for Agent X

$\{a_1c_1e_1b_1d_1\}$:101 $\succ$ $\{a_1c_1e_1b_1d_2, a_1c_1e_1b_2d_1\}$:102 $\succ$ $\{a_1c_1e_1b_2d_2\}$:103
$\{a_1c_1e_2b_1d_1\}$:201 $\succ$ $\{a_1c_1e_2b_1d_2, a_1c_1e_2b_2d_1\}$:202 $\succ$ $\{a_1c_1e_2b_2d_2\}$:203
$\{a_1c_1e_3b_1d_1\}$:301 $\succ$ $\{a_1c_1e_3b_1d_2, a_1c_1e_3b_2d_1\}$:302 $\succ$ $\{a_1c_1e_3b_2d_2\}$:303
$\{a_1c_2e_2b_1d_1, a_1c_2e_3b_1d_1\}$:401$\succ$$\{a_1c_2e_2b_1d_2, a_1c_2e_3b_1d_2,$ $a_1c_2e_2b_2d_1, a_1c_2e_3b_2d_1\}$:402$\succ$$\{a_1c_2e_2b_2d_2, a_1c_2e_3b_2d_2\}$:403
$\{a_2c_2e_1b_1d_1\}$:501 $\succ$ $\{a_2c_1e_1b_1d_2, a_2c_1e_1b_2d_1\}$:502 $\succ$ $\{a_2c_1e_1b_2d_2\}$:503

TABLE II
Ranked Service Pattern aSet for Agent Y

$\{b_1d_1e_1c_1a_1, b_1d_1e_1c_1a_2\}$:101$\succ$$\{b_1d_1e_1c_2a_1, b_1d_1e_1c_2a_1\}$:102
$\{b_1d_1e_2c_1a_1, b_1d_1e_2c_1a_2\}$:201$\succ$$\{b_1d_1e_2c_2a_1, b_1d_1e_2c_2a_1\}$:202
$\{b_1d_1e_3c_1a_1, b_1d_1e_3c_1a_2\}$:301$\succ$$\{b_1d_1e_3c_2a_1, b_1d_1e_3c_2a_1\}$:302
$\{b_1d_2e_2c_1a_1, b_1d_2e_2c_1a_2\}$:401$\succ$$\{b_1d_2e_2c_2a_1, b_1d_2e_2c_2a_1\}$:402
$\{b_1d_2e_3c_1a_1, b_1d_2e_3c_1a_2\}$:501$\succ$$\{b_1d_2e_3c_2a_1, b_1d_2e_3c_2a_1\}$:502

TABLE III
Ranked Service Pattern aSet for Agent Z

$\{a_1b_1c_1e_1d_1, a_1b_1c_1e_2d_1\}$:101$\succ$$\{a_1b_1c_1e_1d_2, a_1b_1c_1e_2d_2\}$:102
$\{a_1b_1c_1e_3d_1\}$:201$\succ$$\{a_1b_1c_1e_3d_2\}$:202
$\{a_1b_1c_2e_2d_1, a_1b_1c_2e_3d_1\}$:301$\succ$$\{a_1b_1c_2e_2d_2, a_1b_1c_2e_3d_2\}$:302
$\{a_1b_1c_2e_1d_1\}$:401$\succ$$\{a_1b_1c_2e_1d_2\}$:402
$\{a_1b_2c_1e_1d_1\}$:501$\succ$$\{a_1b_2c_1e_1d_2\}$:502

From Tables I, II and III, we can see that $a_1b_1c_1d_1e_1$ is in the best Sub_aSet of every agent. We can conclude that this service pattern should be the best one for all the agents. Our

Algorithm 4 merges service patterns in the aSet of each agent and assigns a total rank value for each service pattern. The 10 best service patterns are listed in Table IV along with their rank values and their distance from the respective mean rank values averaged over all agents (see Equations 1 and 2). Note that the three agents have equal weight in this case.

TABLE IV
MERGED AND RANKED SERVICE PATTERNS

#	Pattern	Rank_total	Distance from Mean
1	$a_1b_1c_1d_1e_1$	101	0
2	$a_1b_1c_1d_1e_2$	168	57.74
3	$a_1b_1c_1d_2e_1$	202	172.63
4	$a_1b_1c_1d_1e_3$	268	207.61
5	$a_1b_1c_2d_1e_2$	301	99.50
6	$a_1b_1c_2d_1e_3$	335	57.45
7	$a_1b_1c_2d_1e_1$	335	207.60
8	$a_1b_1c_1d_2e_3$	368	207.60
9	$a_1b_1c_2d_2e_2$	402	100.00
10	$a_1b_2c_1d_1e_1$	435	304.96

From Table IV, we can see that service pattern $a_1b_1c_1d_1e_1$ is indeed ranked the best by our algorithms. The service patters $a_1b_1c_2d_1e_3$ and $a_1b_1c_2d_1e_1$ have the same rank 335. They are then ranked by their distance to their respective mean rank values. In this case, $a_1b_1c_2d_1e_3$ is closer to its mean and ranked higher than $a_1b_1c_2d_1e_1$.

We also show the results of ranking when agents' decisions are weighted differently. In this example, X's weight is 0.6, Y's weight is 0.2, and Z's weight is also 0.2. We list the 10 best service patterns in Table V. Because no two patterns have the same total rank values, the distance of service patterns from their respective means is not shown in the table.

TABLE V
AGENTS WITH DIFFERENT WEIGHTS

#	Pattern	Rank_total	#	Pattern	Rank_total
1	$a_1b_1c_1d_1e_1$	101	6	$a_1b_2c_1d_2e_1$	232.7
2	$a_1b_1c_1d_2e_1$	131.9	7	$a_1b_1c_1d_1e_3$	291
3	$a_1b_1c_1d_1e_2$	191	8	$a_1b_2c_1d_1e_2$	301.8
4	$a_1b_2c_1d_1e_1$	201.8	9	$a_1b_1c_1d_2e_3$	321.9
5	$a_1b_1c_1d_2e_2$	221.9	10	$a_1b_2c_1d_2e_2$	332.7

Comparing Tables IV and V, we can see that $a_1b_1c_1d_1e_1$ still has the best rank. This service pattern is a dominant one and is not affected by the importance of agents' preferences. However, many other service patterns' positions are changed. For example, $a_1b_1c_1d_2e_1$ was less preferred than $a_1b_1c_1d_1e_2$ in Table IV, but it now becomes more preferred in Table V when agents have different weights. The service pattern $a_1b_1c_1d_2e_2$ is ranked the fifth in Table V but was not even in the top 10 list of Table IV. Different weights of agents' decisions do affect the final ranking of service patterns. Our algorithms are able to capture this effect.

VI. EXPERIMENTAL RESULTS

In this section, we begin with a set of experiments to verify the accuracy of our approach and compare with a quantitative approach. The first experiment involves six service attributes (**B**, **C** and **D** mentioned in Section IV-A, and Integrity, Throughout and Availability) where agents have the consistent preferences over the values of these attributes. For example, every agent prefers a lower price (**D**) and higher security (**B**). This setting allows us to objectively identify a set of services, some of which are strictly more preferred than the others by all agents. The domain of these attributes is normalized to be $[0, 100]$. 5 services are manually generated so that $S_1 \succ S_2 \succ S_3 \succ S_4 \succ S_5$ by all agents as shown in Table VI. Another 25 services are randomly generated. 4 agents with randomly generated preferences are also involved in the experiment. We run the experiment for 5 times and report the ranking of the five services in Table VII. We can see that our approach correctly ranks these services and remains their relative positions no matter what other services are.

TABLE VI
FIVE MANUALLY GENERATED SERVICES

Service	**C**	**D**	**B**	Integrity	Throughout	Availability
S_1	1	0	98	98	98	98
S_2	12	10	86	85	86	86
S_3	34	31	66	65	66	66
S_4	65	62	36	35	36	36
S_5	96	95	6	8	8	8

TABLE VII
RANKING OF THE FIVE MANUALLY GENERATED SERVICES

Service	Test 1	Test 2	Test 3	Test 4	Test 5
S_1	1	1	1	1	1
S_2	2	3	3	3	3
S_3	8	9	8	10	8
S_4	19	22	20	21	21
S_5	23	30	29	30	30

In the second experiment, we evaluate the accuracy of our approach in a more general case. In this experiment and all the later experiments in this section, six QoS attributes are involved, including the five ones mentioned in Section IV-A and the extra one, Availability (**F**: 0 - 100%). Agents may not have consistent preferences for these attributes. Each attribute has 2 values. We generate 500 candidate services. 4 agents are involved in this experiment with randomly generated preferences, some of which are incomplete. This experiment includes two cases. In the first case (Case 1), each agent has the equal weight, while in the second case (Case 2), we assign each agent with a different weight. For each case, our approach generates the best 5 service pattern Sub_aSets. We measure the satisfaction ratio of services in each pattern, and check whether our approach generates ranked service patterns that correctly match their satisfaction ratio. The satisfaction ratio of one service S can be calculated by $\sum_{i=1}^{4} \frac{P_i'}{P_i} Weight(i)$, where P_i is the total number of preference statements for agent i and P_i' is the number of satisfied preference statements by service S for this agent. The average satisfaction ratios of the services in the best 5 service pattern Sub_aSets after running the experiment for 5 times are listed in Table VIII. We can see that the ranking of the 5 best Sub_aSets generated by our approach correctly matches their average satisfaction ratios,

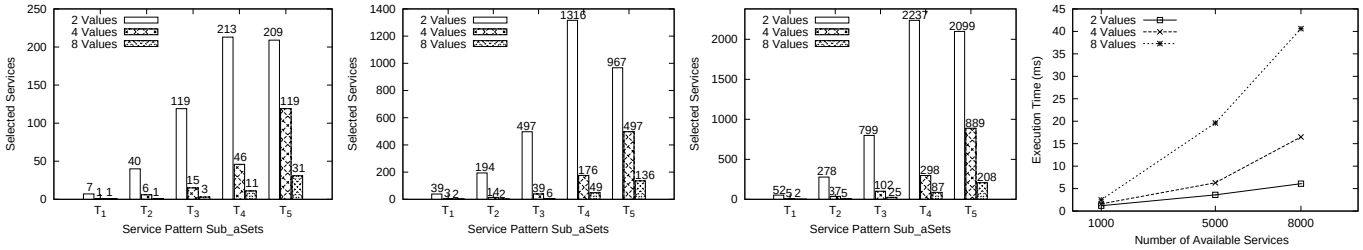

Fig. 5. (a) Service Selection from 1000 Services; (b) Service Selection from 5000 Services; (c) Service Selection from 8000 Services; (d) Runtime for Different Numbers of Candidate Services;

which confirms that our approach accurately ranks services.

The third experiment is carried out to demonstrate that a classic quantitative approach (i.e. of [7]) may have problems when agents cannot accurately identify their quantitative preferences. In this experiment, each agent assigns a score in $[1, 10]$ to each attribute value of a service. However, an agent may incorrectly represent its preference by ± 1, i.e. a score of 9 may be assigned to a attribute value with the correct score of 10. 30 services are randomly generated for the experiment involving 8 agents. Applying the quantitative approach to both correct and incorrect scores of attribute values respectively, we can see from some examples in Table IX that the quantitative approach incorrectly assigns scores to some services, and thus generates wrong ranks for them because of the incorrect scores of service attribute values assigned by agents.

TABLE VIII
AVERAGE SATISFACTION RATIO OF THE BEST 5 SUB_ASETS

Cases	T_1	T_2	T_3	T_4	T_5
Case 1	0.95	0.88	0.83	0.72	0.65
Case 2	0.97	0.88	0.82	0.74	0.66

TABLE IX
INACCURACY OF A QUANTITATIVE APPROACH

Service	Correct Score	Incorrect Score	Incorrect Rank
S_{13}	352	376	3
S_4	368	368	4
S_{16}	400	352	7
S_5	320	296	13
S_{19}	368	280	16

In the second set of experiments, we further test our algorithms based on a wide set of randomly generated QoS attribute values of a large number of candidate services. In some of these experiments, attributes may have a larger number of possible values, which are artificially generated. 1000, 5000, and 8000 candidate services are randomly generated respectively. We also compare the results of service selection for different cases where each attribute has 2, 4, and 8 values respectively. We can see from Figures 5(a), 5(b) and 5(c) that the best Sub_aSet T_1 in each case (i.e. where 1000 services are generated and each attribute has 4 values) has at least one service. These services satisfy the agents the most. As can be seen from these figures, the number of services in each Sub_aSet decreases when attributes have more possible values. This is simply because more possible values for attributes will increase the number of attribute constraints. It becomes more difficult for candidate services to match service patterns in aSets. Comparing the three figures, we can see that the number of services in each Sub_aSet increases when a larger number of candidate services are generated.

In the third set of experiments, we analyze the execution time of our algorithms for different numbers of generated candidate services, QoS attributes, agents and possible values for each attribute, respectively. In the first three cases, each variable has 2, 4, and 8 values respectively. Our analysis has been performed on a 2.13GHz Intel Core2 Workstation with 2GB of RAM. We first look at how the execution time will change when different numbers of candidate services are generated. There are two agents in this experiment. From Figure 5(d), we can see that the execution time of our algorithms will increase exponentially when a larger number of candidate services are generated. We then fix the number of candidate services to 1000. We vary the number of attributes from 6 to 15. The results are shown in Figure 6(a). The execution time of our algorithms also increases exponentially with the increase of the number of attributes. These results comply with our analysis of algorithm complexity in Section IV-C4. However, we can see that the execution time is only 36.8ms for the case where there are 15 attributes in total and each attribute has 8 possible values. This is acceptable for such a large number of generated candidate services.

We then vary the number of agents from 2 up to 16 to compare the execution time. In this case, there are 8000 candidate services. We can see from Figure 6(b) that the execution time increases linearly with the number of agents. Our algorithms scale well with the increase of the number of agents involved in the service selection process. We also fix the number of candidate services to 8000 and the number of attributes to 15, but vary the number of possible values for each attribute. The results in Figure 6(c) show that our algorithms increase exponentially when each attribute has a larger number of possible values. Finally, we test the execution time of our algorithms for the extreme case where there are 8000 candidate services, 15 attributes, and 16 agents, and each attribute has 8 possible values. We run our experiment for 30 times. The results plotted in Figure 6(d) show that the average

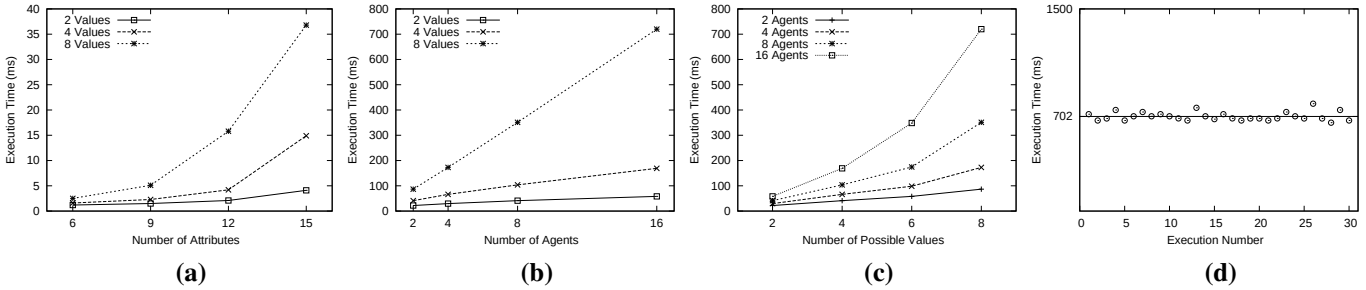

Fig. 6. (a) Runtime for Different Numbers of Attributes; (b) Runtime for Different Numbers of Agents; (c) Runtime for Different Numbers of Possible Values; (d) Average Execution Time

execution time of our algorithms for this extreme case is less than 1 second, which is still acceptable for users of web service selection. Furthermore, the time of each execution is close to the average execution time, indicating that our experimental results in this section are statistically significant.

VII. Related Work

Different quantitative approaches have been proposed for QoS-oriented web service selection [2], [1]. Ardagna and Pernici [2] introduce a mixed integer linear programming modeling approach to the web service selection problem. Lamparter et al. [1] uses utility function policies which draws from multi-attribute decisions theory methods to develop algorithms for optimal service selection. However, these methods require users to provide the exact weight of each attribute. In many situations, users possibly do not know how they should assign weights to attributes in order to maximally meet their preferences. Qualitative methods, on another hand, are more general. Garcia et al. [8] presents a service selection framework that transforms qualitative preferences into an optimization problem. However, they address only a single agent's complete preferences.

Herrera-Viedma et al. [9] present a selection process to deal with group decision making problems with incomplete fuzzy preference relations. They use consistency measures to estimate the incomplete fuzzy preference relations, and propose an iterative procedure to estimate the missing information in incomplete fuzzy preference relations. Xu and Chen [10] develop linear-programming models for dealing with MAGDM (multiple-attribute group decision making) problems, where the information about attribute weights is incomplete and the decision makers have their preferences on alternatives. Zhang et al. [11] propose an integration approach to combine multiple attribute decision making with users' preference information on alternatives. These approaches are also quantitative. Our approach is qualitative. We apply the compact representation of agents' preferences using CP-nets and propose a list of algorithms by implementing the Rank semantic, to select services for multiple agents with incomplete preferences.

VIII. Contributions

Our work allows for service selection based on agent preferences in a qualitative rather than quantitative way. We use CP-nets for representing agents' incomplete preference statements. We also implement the Rank and ceteris paribus semantics for aggregating multiple agents' preferences. We then apply this approach in the design of the algorithms to QoS-based service selection for multiple agents with incomplete preferences. Our experimental results show that our approach is effective in selecting the best services for multiple agents even when they have incomplete preferences, and the execution time of our algorithms is generally acceptable for a large number of agents when many services are available for selection. Our approach may also be extended to cope with more general group decision making problems with incomplete preference relations, as we will consider more complex preferences for web service selection in the future.

References

[1] S. Lamparter, A. Ankolekar, R. Studer, and S. Grimm, "Preference based selection of highly configurable web services," in *Proceedings of the 16th ACM International Conference on World Wide Web*, 2007.

[2] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Transactions on Software Engineering*, vol. 33, no. 6, pp. 369–384, 2007.

[3] P. Xiong and Y. Fan, "Qos-aware web service selection by a synthetic weight," in *Proceedings of the 4th IEEE International Conference on Fuzzy Systems and Knowledge Discovery*, 2007.

[4] C. Boutilier, R. I. Brafman, H. H. Hoos, and D. Poole, "Preference-based constrained optimization with cp-nets," *Computational Intelligence*, vol. 20, no. 2, pp. 137–157, 2004.

[5] ——, "Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements," *Journal of Artificial Intelligence Research*, vol. 21, pp. 137–191, 2004.

[6] F. Rossi, B. Venable, and T. Walsh, "mcp nets: Representing and reasoning with preferences of multiple agents," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2004.

[7] R. L. Keeney and H. Raiffa, *Decision with Mutiple Objectives: Preferences and Value Tradeoffs*. Wiley, 1976.

[8] J. M. Garcia, D. Ruiz, A. Ruiz-Cortes, and J. A. Parejo, "Qos-aware semantic service selection: An optimization problem," in *Proceedings of the 2008 IEEE Congress on Services*, 2008.

[9] E. Herrera-Viedma, F. Chiclana, F. Herrera, and S. Alonso, "Group decision-making model with incomplete fuzzy preference relations based on additive consistency," *IEEE Transactions on Systems Man and Cybernetics Part B (Cybernetics)*, vol. 37, no. 1, pp. 176–189, 2007.

[10] Z. S. Xu and J. Chen, "MAGDM linear-programming models with distinct uncertain preference structures," *IEEE Transactions on Systems Man and Cybernetics Part B (Cybernetics)*, vol. 38, no. 5, pp. 1356–1370, 2008.

[11] Q. Zhang, W.-J. Feng, and D. Shao, "An integration approach to multiple attribute decision making with preference information on alternatives," in *Proceedings of the Conference on Control and Decision*, 2008.