Web Intelligence and Agent Systems: An International Journal 5 (2007) 1–5
IOS Press

Qualitative Preference-Based Service Selection for Multiple Agents

Hongbing Wang [a,*], Jie Zhang [b], Cheng Wan [a], Shizhi Shao [a] and Robin Cohen [c]
[a] *Science and Engineering, Southeast University, China*
E-mail: hbw@seu.edu.cn
[b] *School of Computer Engineering, Nanyang Technological University, Singapore*
E-mail: zhangj@ntu.edu.sg
[c] *School of Computer Science, University of Waterloo, Canada*
E-mail: rcohen@cs.uwaterloo.ca

AbstractService selection is one of the important problems in applications of multi-agent systems. A qualitative way is desirable for 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 a preference reasoning algorithm to construct a derivation tree from a CP-net for each agent and then generate all service patterns for this agent. The Rank semantic is used together with the Lex semantic to provide the ordering of different service patterns. After that, we merge ranked service patterns for all agents and select a set of services that satisfy these agents the most. Finally, we also propose a semantic validation algorithm to show the satisfactory degree of the best service patterns according to other voting semantics. Experimental results indicate that this method can always obtain optimal outcomes which closely satisfy all agents, within acceptable execution time.

Keywords: Multi-agent Service Selection, Qualitative Preference, CP-nets, mCP-nets, Incomplete Preference

1. Introduction

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. For example, an agent may express its preference that it prefers value a over value b for one attribute of some type of services, instead of using a numeric value to represent its preferences on value a and value b respectively. Besides, a quantitative approach may induce errors from agents in identifying their utilities and thus make wrong selection of services, as it is sometimes not straightforward to assign an utility value to an attribute value.

Research studies on services selection based on preferences mainly focus on a single agent's preferences. But in many real situations, a decision may have to be made for a group of agents with different preferences on QoS attributes of web services [4]. In addition, the agents' preferences may also be incomplete [5]. 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

*Corresponding author. E-mail: hbw@seu.edu.cn.

agents' preferences and select services which satisfy them the most.

In this paper[1], we use CP-nets [7] 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 the Rank semantic [8]. Finally, we merge ranked service patterns for all agents based on the Lex semantic, and select a set of services that satisfy these agents the most. In addition, a semantic validation algorithm is proposed to show the satisfactory degree of the best service patterns according to the voting semantics of Pareto, Majority and Max. The processes of service selection for multiple agents with incomplete preferences are demonstrated by a concrete example, involving three agents with different preferences.

The performance of our method is also evaluated by a wide set of artificially generated QoS attributes of services and values of attributes. More specifically, we verify the effectiveness of our algorithms and compare it with the quantitative approach. We also record down the execution time of our algorithms. For these experiments, we vary the total number of available services, the number of attributes for services, the number of agents involved in service selection, as well as the number of possible attribute constraint values. Experimental results confirm that our method is able to more effectively select the most optimal services for agents than the quantitative approach in different simulation scenarios. The execution time of our method is also acceptable.

The remainder of the paper is organized as follows. A motivation scenario is given in Section 2. Section 3 introduces the notions and concepts of preference logic, CP-nets, mCP-nets, and voting semantics. In Sections 4 and 5, the representing and reasoning techniques of preferences and the illustrative examples are presented. The experimental results are provided in Section 6. Section 7 summarizes and compares our method with the related work. Conclusions and future work are finally given in Section 8.

[1]This paper is the extended version of our previous work in [6].

2. 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 satisfy all these users.

A motivating real life example for our work is the field of enterprise information management [9]. 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.

3. Preliminaries

We begin with a brief outline of relevant notions from decision theory, CP-nets introduced by Boutilier et al. [7], and mCP-nets and voting semantics proposed by Rossi et al. [8], which are the fundamental concepts and methods for the proposal of our algorithms of service selection for multiple agents with qualitative preferences that may be incomplete.

3.1. 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 nonempty 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.

3.2. CP-nets

CP-nets introduced by Boutilier et al. [7] 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.

3.3. 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-nets [8] 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-nets is an assignment of the values to the attributes in all the partial CP-nets in their domains. Note that a CP-net is also a special mCP-nets where m=1. Thus, service selection based on mCP-nets inherits complexity of and is more complicated than that based on a single CP-net.

3.4. Voting Semantics

We reason about a mCP-nets 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 [8], described as follows:

- In Pareto, outcomes are often incomparable. An outcome is Pareto optimal iff no other outcome is better.
- Majority and Max are the weaker criteria, and many agents often vote in favor or for incomparability. Two outcomes are majority OR max incomparable iff they are not ordered either way.
- In the Lex semantic, agents are ordered by their importance. Two outcomes are lexicographically incomparable iff there exists some distinguished agent such that all agents higher in the ordered are indifferent between the two outcomes and the outcomes are incomparable to the distinguished agent.
- In the Rank semantic, each agent ranks each outcome. Two outcomes are ranked indifferent iff the sums of the ranks assigned to them are the same.

The Pareto, Lex and Rank semantics define strict orderings. By comparison, neither the Majority nor Max semantics induce a strict ordering. The five voting semantics are also embedded by Rossi et al. [8] in the context of mCP-nets, but they did not provide actual algorithm designs. We extend and implement the Rank semantic together with the Lex semantic to address multiple agents' preferences on QoS attributes

in service selection, by considering the case where agents may have different preferences and they may be weighted differently in making decisions on services. We also validate the selected best services according to the other three voting semantics (Pareto, Majority and Max).

4. Preference Representation and Reasoning

We first describe the representation for incomplete preferences of an agent using CP-nets. The example scenario in Section 2 will be used throughout the current section. We then present the processes and algorithms of our reasoning about possibly incomplete qualitative preferences of multiple agents for service selection.

4.1. 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.

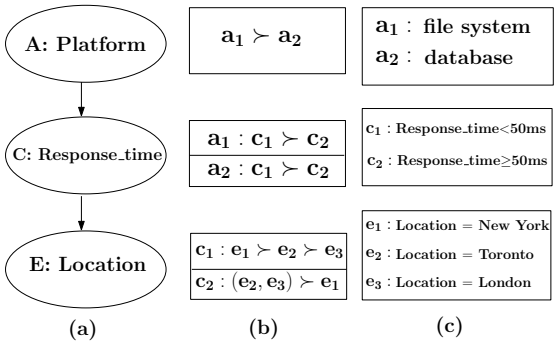

Figure 1. (a, b) CP-net for Agent X, (b) CPT, (c) Preference Statements

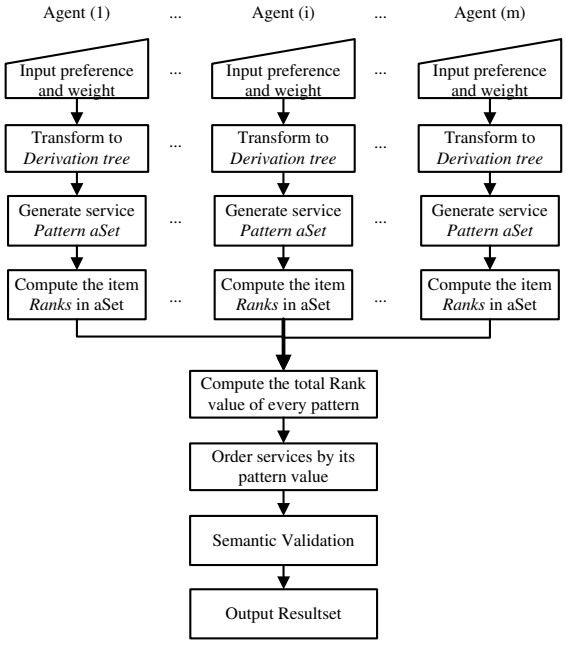

Figure 2. Service Selection Processes

4.2. 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 based on the Lex semantics. The available services will be divided into different patterns according to their quality attribute values. We also provide the results of the other semantics (i.e. Pareto, Ma-

jority and Max) for the best patterns. Finally, the services that match the best patterns will be chosen for all agents. The process of service selection in our system is shown in Figure 2.

4.3. 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 the algorithm for validating the selected service patterns according to the Pareto, Majority and Max semantics. We finally discuss complexity analysis for these algorithms that will also be confirmed by the experimental results in Section 6.

Algorithm 1 Generating Derivation Tree

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

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

4.3.1. 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 3. Nodes a_1 and a_2 are in the upper level of the tree because attribute **A** is unconditional. Node a_1 is a left child because it is more preferred than node a_2.

4.3.2. 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.* $a_1b_1c_1d_1e_1$.

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.*

As shown in Figure 3, 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 3 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 accurateness. 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.

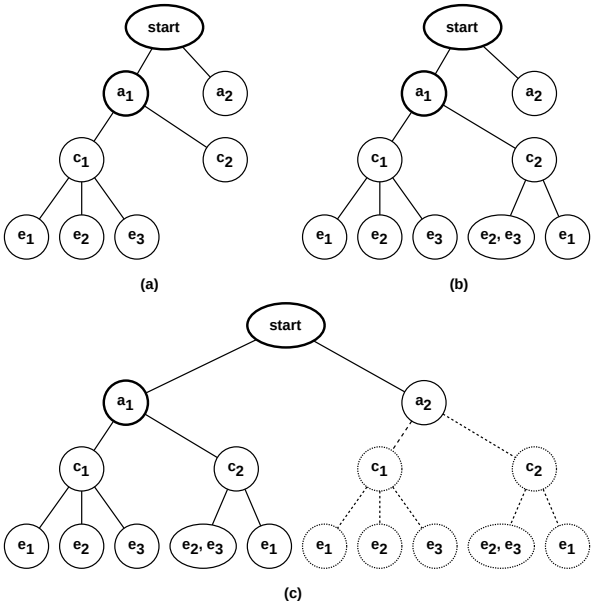

Figure 3. Derivation Tree of CPT for Agent X

Algorithm 2 Generating Service Pattern aSet

Input:
Preference statements;
Derivation tree Tr;
Output:
Service pattern aSet T

```
1: int i = 0; aSet T = NULL;
2: for each path in tree Tr from left to right do
3:    i = i + 1;
4:    Combine preference statements in all levels;
5:    Sub_aSet T[i] = NULL;
6:    Add the combination into T[i] as an item;
7:    Add T[i] into aSet T;
8: end for
9: for each remaining attribute do
10:   if attribute value has strict priority then
11:      Order preference statements by priority;
12:   end if
13:   Add them into each Sub_aSet in T;
14: end for
```

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 5, 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.

Algorithm 3 Ranking Service Patterns in aSet

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

```
1: int i = 0;
2: for each Sub_aSet in T by priority do
3:    i = i + 1;
4:    j = 0;
5:    for each service pattern in T[i] by priority do
6:       j = j + 1;
7:       Rank(pattern[j]) = i × setWeight + j;
8:    end for
9: end for
```

4.3.3. 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.

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:

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

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.

Algorithm 4 Merging 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;

```
1: if no weight of agents provided then
2:    Weight of each agent = 1/M;
3: end if
4: Initialize Rank_total of each pattern = 0;
5: Compute Rank_total using Equation 1;
6: Order service patterns according to Rank_total
7: for each set of patterns with same Rank_total do
8:    Compute weighted distance using Equation 2;
9:    Order them according to weighted distance;
10: end for
```

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.

4.3.4. Semantic Validation of Best Service Patterns

As mentioned in Section 3.4, there are mainly five voting semantics in service selection. In the above algorithms, the Rank semantic is used together with the Lex semantic to produce the order of different service patterns. But, in real world applications, other semantics may also be concerned when selecting services based on preferences of each member agent in a decision group. We thus propose to make use of the other three voting semantics (i.e. Pareto, Majority and Max) as additional validation for the best service patterns that are ranked according to the Rank and Lex semantics. We provide for each of the best service patterns

Algorithm 5 Service Selection

Input:
Service patterns ordered by total rank;
A set of available services;
Output:
Service patterns ordered by total rank;

```
1: for each available service do
2:    Find matching pattern based on attribute values;
3: end for
4: boolean flag = true;
5: while flag do
6:    for each ordered service pattern do
7:       Find services that match the pattern;
8:       if found matched services then
9:          flag = false;
10:      end if
11:   end for
12: end while
13: return the found services;
```

a satisfactory degree calculated using Pareto, Majority and Max respectively.

Pareto-Satisfactory Degree (PSD) of a service pattern is a percentage value to describe the satisfactory degree of the service pattern according to the Pareto voting semantic. Let one of the best service patterns (i.e. patterns with the smallest rank values according to Algorithm 4) be P_j and $|P_j|$ be the number of attribute constraints (see Definition 1) in this pattern. If one of the attribute constraints in the pattern is preferred by all agents according to the agents' preference statements, the attribute constraint is called a Pareto-satisfactory attribute constraint.

Let $|P_j{}_{\shortmid Pareto}|$ be the number of Pareto-satisfactory attribute constrains in the pattern P_j. We have the Pareto-Satisfactory Degree of P_j as follows:

$$PSD(P_j) = \frac{|P_j{}_{\shortmid Pareto}|}{|P_j|} * 100\% \quad (3)$$

If the number of agents preferring an attribute constraint in P_j is larger than the number of the rest of the agents, the attribute constraint is called the Majority-satisfactory attribute constraint. Let $|P_j{}_{\shortmid Majority}|$ be the number of Majority-satisfactory attribute constraints in the pattern P_j. Accordingly, the Majority-Satisfactory degree ($MajSD$) can be defined as follows:

$$MajSD(P_j) = \frac{|P_j{}_{\shortmid Majority}|}{|P_j|} * 100\% \quad (4)$$

If the number of agents preferring an attribute constraint in P_j for an attribute is larger than the maximum number of agents that prefer any of the other attribute constraints for the same attribute, and also larger than the number of agents that disclaim the attribute, the attribute constraint is called the Max-satisfactory attribute constraint. Let $|P_j \upharpoonright_{Max}|$ be the number of Max-satisfactory attribute constraints in the pattern P_j. Accordingly, the Max-Satisfactory degree ($MaxSD$) can be defined as follows:

$$MaxSD(Pat_j) = \frac{|P_j \upharpoonright_{Max}|}{|P_j|} * 100\% \quad (5)$$

According to the definition of the above three voting semantics, we can derive two important characteristics of the relationships between the three voting semantics. First, if an attribute constraint is called the Pareto-satisfactory attribute constraint, it will satisfy all the agents' requirements. So the number of agents that do not prefer the attribute constraint is zero. Obviously, if an attribute constraint satisfies the Pareto voting semantic, it will also satisfy the Majority voting semantic and the Max voting semantic. Second, the number of agents preferring a Majority-satisfactory attribute constraint is larger than the total number of agents that do not prefer any of the outcome, which is larger than the maximum number of agents that prefer any of the other outcomes. So, if a preference outcome satisfies the Majority voting semantic, it will also satisfy the Max voting semantic. Based on the above relationship characteristics of voting semantics, we can simplify the calculation of $|P_j \upharpoonright_{Max}|$ and $|P_j \upharpoonright_{Majority}|$.

The pseudo code for semantic validation is summarized in Algorithm 6. Note that, in the algorithm, to check whether an agent prefers an attribute constraint in a service pattern, we can traverse the agent's derivation tree to see whether the attribute constraint appears on the left branch of the attribute. If the attribute constraint appears on the left branch, it means that the agent prefers the attribute constraint. If the attribute constraint appears on the right branch, it means that the agent prefers other attribute constraints for the preference. If the attribute does not appear in the derivation tree, it means that the agent disclaims the attribute and the attribute constraint.

4.3.5. 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

Algorithm 6 Semantic Validation

Input:
Top N service patterns ordered by total rank;
Derivation tree of each agent;
Output:
Semantic satisfactory degrees of the service patterns;

```
1: int nPrefer, nPreferOther, nDisclaim;
2: //number of agents preferring the attribute constraint, other constraints and disclaiming attribute, respectively
3: for each service pattern P_j do
4:   for each attribute constraint in P_j do
5:     nPrefer = nPreferOther = nDisclaim=0;
6:     for each agent with a derivation tree do
7:       if it prefers the attribute constraint then
8:         nPrefer ++;
9:       else
10:        if it prefers other constraints then
11:          nPreferOther ++;
12:        else
13:          if it disclaims the attribute then
14:            nDisclaim ++;
15:          end if
16:        end if
17:      end if
18:    end for
19:    if nPreferOther=nDisclaim=0 then
20:      |P_j ↾_Pareto| ++; |P_j ↾_Majority| ++;
21:      |P_j ↾_Max| ++;
22:    else
23:      if nPrefer>nPreferOther+nDisclaim then
24:        |P_j ↾_Majority| ++;
25:        |P_j ↾_Max| ++;
26:      else
27:        if (nPrefer>nPreferOther) AND
           (nPrefer>nDisclaim) then
28:          |P_j ↾_Max| ++;
29:        end if
30:      end if
31:    end if
32:  end for
33:  Compute PSD(P_j), MajSD(P_j), MaxSD(P_j);
34: end for
35: return semantic satisfactory degree of each pattern;
```

$maxPS$. 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 $O(M \times sizeT \times log(sizeT))$, where M is the total number of agents. The complexity of Algorithm 5 is dependent on $sizeT$ and the number of available services for selection. Because Algorithm 6 is used to validate the semantic satisfaction of selected service patterns, only top N service patterns are considered in the algorithm. The complexity of Algorithm 6 is determined by the number of attributes in one pattern and the number of agents. Furthermore, although we use three semantics to validate the patterns, we only need to execute one round of loop because of the relationships between the three semantics discussed earlier. So the complexity of Algorithm 6 is $O(M \times attNum)$. Our experimental results in Section 6 indicate that the runtime of our algorithms is tolerable for a large number of available services.

5. Example Demonstration

In this section, we follow up with the example scenario described in Section 2 and the preferences of agent X in Section 4.1, and add preferences of other two agents, Y and Z. We demonstrate the processing results of our service pattern generation, ranking, merging, as well as semantic validation algorithms.

The preferences of agents Y and Z are represented by CP-nets, as shown in Figures 4 and 5. 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 4 (b) and Figure 5 (b). As described in Section 4.3, 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 1, 2 and 3.

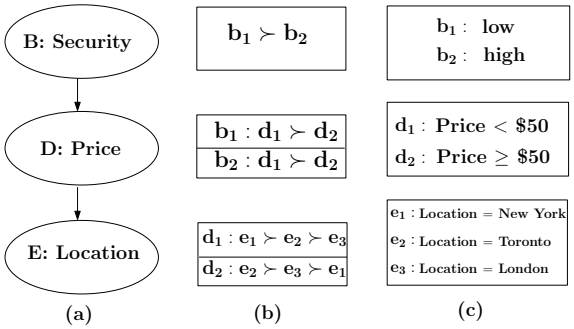

Figure 4. (a,b) CP-net for Agent Y, (b) CPT, (c) Preference Statements

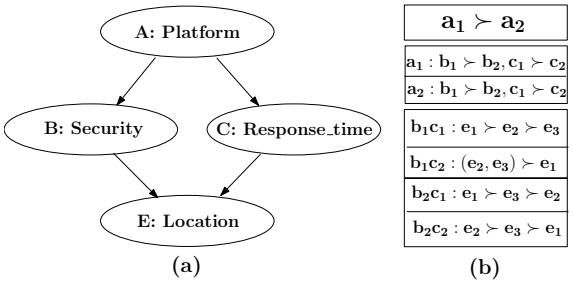

Figure 5. (a,b) CP-net for Agent Z, (b) CPT

From Tables 1, 2 and 3, 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 4 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.

From Table 4, we can see that service pattern $a_1b_1c_1d_1e_1$ is indeed ranked the best by our algo-

Table 1
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 2
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 3
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

Table 4
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

rithms. 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 5. 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 5
Agents with Different Weights

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

Comparing Tables 4 and 5, 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 4, but it now becomes more preferred in Table 5 when agents have different weights. The service pattern $a_1b_1c_1d_2e_2$ is ranked the fifth in Table 5 but was not even in the top 10 list of Table 4. Different weights of agents' decisions do affect the final ranking of service patterns. Our algorithms are able to capture this effect.

Furthermore, we make use of the semantic validation algorithm to validate the ranking results under different voting semantics. According to the preferences of the three agents X, Y and Z, the semantic voting results are shown in Table 6. Here, we use "1" to indicate that an agent prefers the attribute constraint, "-1" to indicate that the agent prefers other attribute constraints, and "0" to indicate that the agent disclaims the attribute. Based on the agents' voting results, the satisfied voting semantics are also presented for each attribute constraint in a service pattern in the table. Note that as discussed in Section 4.3.4, if an attribute satisfies Pareto, it also satisfies both Majority and Max. If it satisfies Majority, it also satisfies Max. The semantic "Null" means that the attribute constraint does not satisfy any voting semantic.

Table 6
Semantic Validation of Top 3 Service Patterns

Pattern	Constraint	X	Y	Z	Semantic
$a_1b_1c_1d_1e_1$	a_1	1	0	1	Majority
	b_1	0	1	1	Majority
	c_1	1	0	1	Majority
	d_1	0	1	0	Null
	e_1	1	1	1	Pareto
$a_1b_1c_1d_1e_2$	a_1	1	0	1	Majority
	b_1	0	1	1	Majority
	c_1	1	0	1	Majority
	d_1	0	1	0	Null
	e_2	-1	-1	0	Null
$a_1b_1c_1d_2e_1$	a_1	1	0	1	Majority
	b_1	0	1	1	Majority
	c_1	1	0	1	Majority
	d_2	0	-1	0	Null
	e_1	1	-1	1	Majority

Table 7
Semantic Validation Results

#	Pattern	PSD	MajSD	MaxSD
1	$a_1b_1c_1d_1e_1$	20%	80%	80%
2	$a_1b_1c_1d_1e_2$	0%	60%	60%
3	$a_1b_1c_1d_2e_1$	0%	80%	80%
4	$a_1b_1c_1d_1e_3$	0%	60%	60%
5	$a_1b_1c_2d_1e_2$	0%	60%	60%
6	$a_1b_1c_2d_1e_3$	0%	60%	60%
7	$a_1b_1c_2d_1e_1$	0%	40%	40%
8	$a_1b_1c_1d_2e_3$	0%	60%	60%
9	$a_1b_1c_2d_2e_2$	20%	60%	60%
10	$a_1b_2c_1d_1e_1$	20%	60%	60%

The semantic validation results for the top 10 service patterns are shown in Table 7. Generally, the patterns with higher ranking scores have better semantic validation results. In Table 7, we can see that the first pattern is still the best according to the semantic validation. But the third pattern is better than the second pattern according to the voting semantics of Pareto and Majority. In this case, it is up to the agents to decide which voting semantics they concern more in order to choose the most suitable service patterns.

6. Experimental Results

In this section, we show experimental results from four aspects. We first carry out experiments to verify the effectiveness of our method and compare it with a quantitative approach. We then evaluate our method based on a large number of random QoS attribute values. After that, the execution time of our method is tested for different numbers of generated candidate services, QoS attributes, agents and possible values for each attribute, respectively. Finally, we use the three voting semantics of Pareto, Majority and Max to verify the performance of the top 5 Sub_aSets generated by our method.

6.1. Effectiveness Comparison with a Quantitative Approach

We begin with a set of experiments to verify the effectiveness of our approach and compare with a quantitative approach. In these experiments, the good outcome of an approach means that the approach correctly ranks candidate services and remains their relative positions according to agents' preferences, no matter what other services are. The effectiveness of the approach is then evaluated as the ratio of the number of good outcomes over all tests.

Table 8
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 9
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

The first experiment involves six service attributes (**B**, **C** and **D** mentioned in Section 4.1, 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 8. 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 9. Although the rank of each service is different in each test in the experiment, our approach remains the relative positions of candidate services according to agents' preferences. It gives five good outcomes in all five tests. So the effectiveness of our approach in this experiment is 1.

In the second experiment, we evaluate the effectiveness 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 4.1 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 10. We can see that the ranking of the 5 best Sub_aSets generated by our approach correctly matches their average satisfaction ratios, which confirms that our approach accurately ranks services.

Table 10
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

The third experiment is carried out to demonstrate that a classic quantitative approach (i.e. of [10]) 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 11 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 11
Incorrect scores of Another 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

6.2. Further Test Based on Random QoS Attribute Values

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 6(a), 6(b) and 6(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.

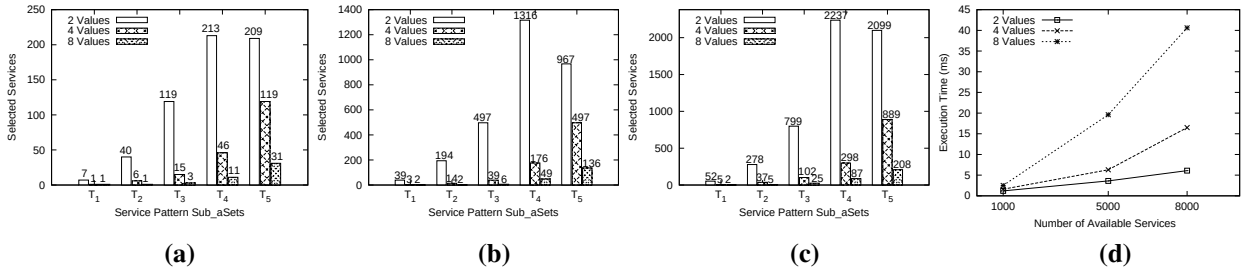

Figure 6. (a) Service Selection from 1000 Services; (b) Service Selection from 5000 Services; (c) Service Selection from 8000 Services; (d) Runtime for Different Numbers of Candidate Services;

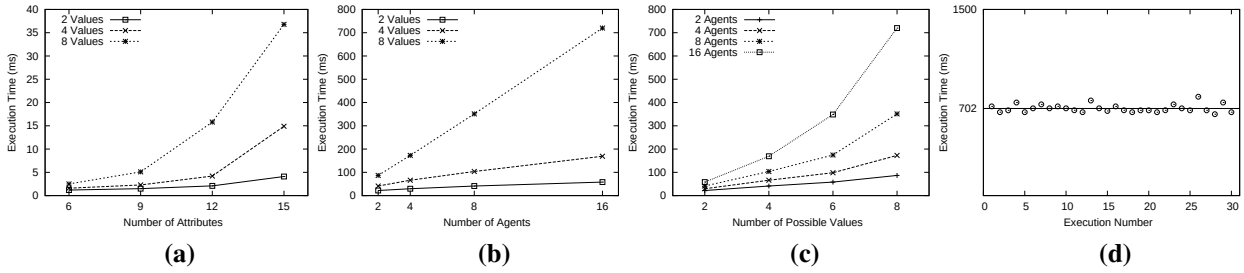

Figure 7. (a) Runtime for Different Numbers of Attributes; (b) Runtime for Different Numbers of Agents; (c) Runtime for Different Numbers of Possible Values; (d) Average Execution Time

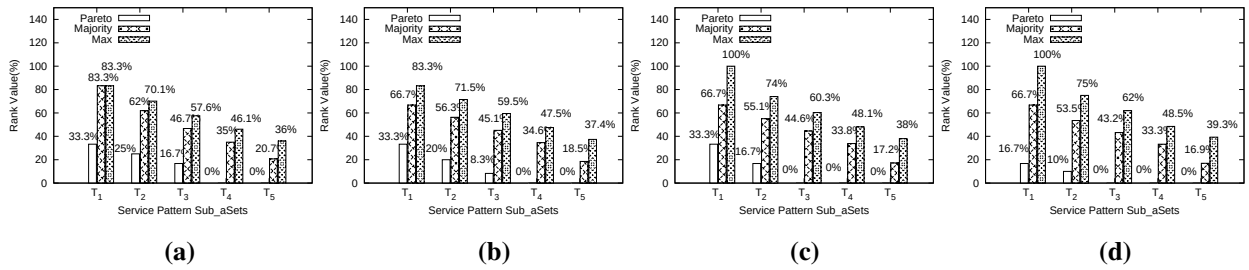

Figure 8. The value of PSD, MajSD, MaxSD under different numbers of agents.(a) 4 Agents; (b) 8 Agents; (c) 16 Agents; (d) 32 Agents

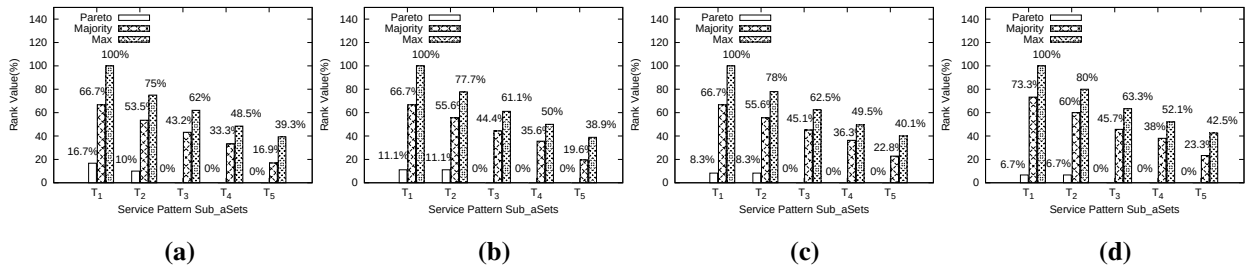

Figure 9. The value of PSD, $MajSD$, $MaxSD$ under different numbers of attribute:(a) 6 attributes; (b) 9 attributes; (c) 12 attributes; (d) 15 attributes

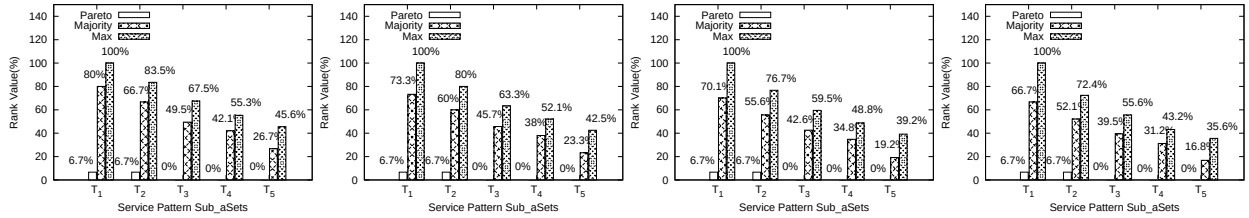

Figure 10. The value of PSD, $MajSD$, $MaxSD$ under different numbers of value:(a) 2 values; (b) 4 values; (c) 6 values; (d)8 values

6.3. Experiments on Execution Time

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 6(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 7(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 4.3.5. 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 7(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 7(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 7(d) show that the average execution time of our algorithms for this extreme case is less than 1 second, which is still acceptable for users of 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.

6.4. Experiments on Semantic Validation

In this section, we show the results of semantic validation for the top 5 Sub_aSets when varying the number of agents, attributes and attribute constraint values. The first set of experiments is to show the semantic validation for the top 5 Sub_aSets under the voting semantics of Pareto, Majority and Max when the number of agents is 4, 8, 16 and 32 respectively. These experiments involve 6 attributes and each attribute has 4 constraint values. As some Sub_aSets (for example, T2) may contain more than one service patterns, we compute the average value of PSD, $MajSD$ and $MaxSD$ for service patterns in those Sub_aSets. The values of PSD, $MajSD$ and $MaxSD$ for the top 5 service pattern Sub_aSets are shown in Figure 8. From the results, we can see that when the number of agents is 4, service patterns in T1 have PSD=33.3%, $MajSD$=83.3% and $MaxSD$=83.3%, while service patterns in T5 have PSD=0%, $MajSD$=20.7% and $MaxSD$=36%. The values of PSD, $MajSD$ and $MaxSD$ are from high to low for T1 to T5. This trend is also true when the number of agents is 8, 16 and 32 respectively, which demonstrates that the ranking for the top 5 Sub_aSets is reliable.

The second set experiments aims to verify the top 5 Sub_aSets under the voting semantics of Pareto, Majority and Max when the number of attributes is 6, 9,

12 and 15 respectively. It involves 32 agents, and each attribute has 4 constraint values. The values of PSD, $MajSD$ and $MaxSD$ for the top 5 Sub_aSets's are shown in Figure 9. From the results, we can see that under 6 attributes, service patterns in T1 have PSD = 16.7%, $MajSD$ = 66.7% and $MaxSD$ = 100%, while those in T5 have PSD = 0%, $MajSD$ = 16.9% and $MaxSD$ = 39.3%. The values of PSD, $MajSD$ and $MaxSD$ are from high to low for T1 to T5. This trend is also true for the cases of 9 attribute, 12 attribute, 15 attribute, which validates that the ranking for the top 5 Sub_aSets is reliable.

The third set of experiments is used to verify the top 5 Sub_aSets under Pareto, Majority and Max when varying the number of attribute constraints. These experiments involve 32 agents and 15 attributes, and the number of constraint values of each attribute varies from 2 to 8. The semantic validation for the top 5 Sub_aSets is shown in Figure 10. From the results, we can see that under 2 constraint values, T1 has PSD = 6.7%, $MajSD$ = 80%, and $MaxSD$ = 100%, while T5 has PSD = 0%, $MajSD$ = 26.7%, and $MaxSD$ = 45.6%. The values of PSD, $MajSD$, and $MaxSD$ are from high to low for T1 to T5 when the number of constraint values is 2. This trend is also true for the cases where the number of attribute values is 4, 6, and 8 respectively. This confirms that the ranking of the top 5 Sub_aSets is reliable. The additional semantic validation also provides more information for agents to select best service patterns, in order to find the most satisfactory services.

7. Related Work

In recent years, service selection based on QoS for multiple agents has become an important research problem in the area of service computing. The existing investigation is mainly focused on three directions: QoS-oriented service selection, the group decision of multi-agents, and uncertain preferences and factors in service selection. In this section, we provide brief descriptions of the studies in each direction and contrast our work with those studies.

Different quantitative approaches have been proposed for QoS-oriented service selection [1,2,11]. Ardagna and Pernici [2] introduce a mixed integer linear programming modeling approach to the service selection problem. Lamparter et al. [1] uses utility function policies which are drawn 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. Quantitative methods have also been considered for service composition. Quantitative intentional automata (QIA) [12] is proposed to address the composition of service oriented computing. QIA is an extension of constraint automata (CA) that incorporates the influence of a system's environment on its performance. The quantitative constraint automata extends CA with quantitative models to capture such non-functional aspects of a system's behavior. An efficient service selection algorithm [13] is also presented to provide the appropriate ground for QoS-aware composition in dynamic service environments. This algorithm is formed as a guided heuristic, and uses "quality level" to express the QoS attributes of services. It proceeds by exploring a combinatorial search tree built from candidate services according to certain rules. Although quantitative approaches have been widely used, qualitative methods, on another hand, are more general. Garcia et al. [14] present a service selection framework that transforms qualitative preferences into an optimization problem. However, they address only a single agent's complete preferences.

Group decision making of multi-agents is also an important direction of related research work. Herrera-Viedma et al. [15] 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 [16] 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. [17] propose an integration approach to combine multiple attribute decision making with users' preference information on alternatives. [18] allows users to interpret the relationships between their query terms and the query space. Accordingly, it allows the users to take an active role in the information retrieval process.

Some researchers have done work in the combination of service selection and group decision. For example, in [19], Lo et al. propose a Web Services Integration and Processing Language to describe opera-

tions and data sources in data processing and integration to fulfill dynamic business requirements. [20] propose an enhanced I-B and B-MDL algorithm, which tries to reduce constraint computation costs and improve the pruning efficiency by means of sort order for candidate parent nodes. [21] tries to solute the problem that efficient local selection strategy fails short in handling global QoS requirements. The authors consider the quantitative non-functional properties of services, and propose a solution that consists of two steps: 1) a mixed integer programming (MIP) is used to find the optimal decomposition of global QoS constraints into local constraints; 2) distributed local selection is then used to find the best services that satisfy these local constraints. Furthermore, they use the notion of skyline to express the services' quantitative QoS attributes. [22] propose an approach to effectively and efficiently select services for composition, reducing the number of candidate services to be considered. [23] distinctively use the concept of lexicographical preferences for the multi-criteria decision-making of the most suitable information and services that fit user needs. The criterion satisfaction levels are defined with a single threshold that represents a boundary value between acceptable and unacceptable values of attributes of alternatives. [24] propose an algorithm that can recommend a number of suitable services based on the user's QoS requirements. In this work, the service's response time, trust degree and monetary cost are considered. [25] tries to use a semantic method to support the automated discovery, selection, and composition of services. They design an efficient model-driven approach to generate OWL-S ontologies from Unified Modeling Language (UML) models. But, due to the complexity of the OWL-S grammar and UML, it is difficult to realize the service selection and discovery manually. Some researchers try to use empirical analysis to study consumer selection of E-Commerce websites in a B2C environment [26]. They present and empirically examine a model where website value in terms of website quality as well as awareness of the site and consumer differences are key variables in explaining online consumer behavior in their choice of websites despite the existence of price dispersions. [27] propose several multiple criteria programming methods for analyzing customers' behavior and finding the appropriate measures to satisfy the VIP user's requirements. [28] examine a class of WSC problems, attempting to balance the trade-off between offline composition and online information gathering with a view of producing high-quality compositions efficiently and without excessive data gathering. Their investigation is performed in the context of the semantic web employing an existing preference-based Hierarchical Task Network WSC system. [29] propose an interaction model to detect and resolve inconsistencies in evolving service compositions. This work presents a new method that associates web services with agents' capability of communication and reflective process execution. [30] presents a framework for web service composition based on social norms, particularly obligations.

Because of the importance of user preferences during the service selection, some researchers have studied the effect of uncertain preference in service computing. [31] studies the problem of majority-rule-based collective decision-making where the agents' preferences are represented by CP-nets. This paper proposes an efficient SAT-based approach, called MajCP, to compute the majority winning alternatives. [32]studies the problem of dominance testing in CP-nets. The authors propose a heuristic algorithm for dominance testing with arbitrary acyclic CP-nets. Their work is only applicable for acyclic CP-nets. Furthermore, uncertain factors have been considered in the QoS of services. In [33], an approach is proposed to measure the quality of elementary services based on the superposition of uncertain factors. And services' priorities can be determined by a method according to the qualities of elementary services.

Although many researchers have worked on the service selection in the environment of multi-agents, the existing approaches are mostly based on quantitative approaches. But in many cases, user's preference cannot be represented by quantitative approaches. In this case, qualitative approaches can be another choice for the representation of agents' preferences. We apply the compact representation of agents' preferences using CP-nets and propose a list of algorithms by implementing the Rank and Lex semantics that are verified by other voting semantics, to select satisfactory services for multiple agents with incomplete preferences.

8. Conclusions and Future Work

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 ap-

proach 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 be extended to cope with more general group decision making problems with incomplete preference relations, as we will consider more complex preferences for service selection in the future. For future work, we will also consider preference analysis when there is a cycle in CP-nets.

Acknowledgments

This work is partially supported by the NSFC (61232007) and JSNSF of China (No.BK2010417).

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 (WWW)*, Banff, AB, Canada, 2007, pp. 1013 – 1022.

[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 Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, Haikou, China, 2007, pp. 632–637.

[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] H. Wang, J. Zhang, Y. Tang, and S. Shao, "Collaborative approaches to complementing qualitative preferences of agents for effective service selection," in *Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, Boca Raton, Florida, USA, 2011, pp. 51–58.

[6] H. Wang, J. Zhang, C. Wan, S. Shao, R. Cohen, J. Xu, and P. Li, "Web service selection for multiple agents with incomplete preferences," in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, Toronto, Canada, 2010, pp. 565–572.

[7] 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*, vol. 21, pp. 135–191, 2004.

[8] F. Rossi, K. B. Venable, and T. Walsh, "mcp nets: Representing and reasoning with preferences of multiple agents," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, San Jose, CA, United states, 2004, pp. 729 – 734.

[9] L. Weigang and V. Fracari Branco, "A web information system for determining the controllers of financial entities in central bank of brazil," *Web Intelligence and Agent Systems*, vol. 4, no. 1, pp. 99–116, 2006.

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

[11] Z. Maamar, Q. Sheng, and B. Benatallah, "Towards a conversation-driven composition of web services," *Web Intelligence and Agent Systems*, vol. 2, no. 2, pp. 145–150, 2004.

[12] F. Arbab, T. Chothia, R. Van Der Mei, S. Meng, Y. Moon, and C. Verhoef, "From coordination to stochastic models of qos," in *Proceedings of the 11th International Conference on Coordination*, Lisbon, Portugal, 2009, pp. 268–287.

[13] S. Sohrabi and S. A. McIlraith, "Preference-based web service composition: A middle ground between execution and search," in *Proceedings of the 9th International Semantic Web Conference (ISWC)*, Shanghai, China, 2010, pp. 713–729.

[14] J. M. Garcia, D. Ruiz, A. Ruiz-Cortes, and J. A. Parejo, "Qos-aware semantic service selection: An optimization problem," in *Proceedings of the IEEE Congress on Services (SERVICES)*, Honolulu, HI, United states, 2008, pp. 384–388.

[15] 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.

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

[17] 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 Chinese Control and Decision Conference (CCDC)*, Yantai, Shandong, China, 2008, pp. 3977–3981.

[18] O. Hoeber, X. Yang, and Y. Yao, "Visiq: Supporting visual and interactive query refinement," *Web Intelligence and Agent Systems*, vol. 5, no. 3, pp. 311–329, 2007.

[19] E. Lo, D. Cheung, C. Ng, and T. Lee, "Wsipl: An xml scripting language for integrating web service data and applications," *Web Intelligence and Agent Systems*, vol. 4, no. 1, pp. 25–41, 2006.

[20] J. Ji, C. Liu, J. Yan, and N. Zhong, "An improved bayesian network structure learning algorithm and its application in an intelligent b2c portal," *Web Intelligence and Agent Systems*, vol. 5, no. 2, pp. 127–138, 2007.

[21] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient qos-aware service composition," in *Proceedings of the 18th International World Wide Web Conference (WWW)*, Madrid, Spain, 2009, pp. 881–890.

[22] M. Alrifai, D. Skoutas, and T. Risse, "Selecting skyline services for qos-based web service composition," in *Proceedings of the 19th International Conference on World Wide Web (WWW)*, Raleigh, NC, United states, 2010, pp. 11–20.

[23] R. R. Yager, G. Gumrah, and M. Z. Reformat, "Using a web personal evaluation tool - pet for lexicographic multi-criteria service selection," *Knowledge-Based Systems*, vol. 24, no. 7, pp. 929–942, 2011.

[24] L. Zhao, Y. Ren, M. Li, and K. Sakurai, "Flexible service selection with user-specific qos support in service-oriented architecture," *Journal of Network and Computer Applications*, vol. 35,

no. 3, pp. 962 – 973, 2012.

[25] I.-W. Kim and K.-H. Lee, "A model-driven approach for describing semantic web services: From uml to owl-s," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 39, no. 6, pp. 637–646, 2009.

[26] H. Zo and R. K., "Consumer selection of e-commerce websites in a b2c environment: A discrete decision choice model," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 39, no. 4, pp. 819–839, 2009.

[27] P. Zhang, X. Zhu, Z. Zhang, and Y. Shi, "Multiple criteria programming models for vip e-mail behavior analysis," *Web Intelligence and Agent Systems*, vol. 8, no. 1, pp. 69–78, 2010.

[28] M. S. Sohrabi Shirin, "Preference-based web service composition: A middle ground between execution and search," in *Proceedings of the 9th International Semantic Web Conference (ISWC)*, 2010, pp. 713–729.

[29] R. Elio, E. Stroulia, and W. Blanchet, "Using interaction models to detect and resolve inconsistencies in evolving service compositions," *Web Intelligence and Agent Systems*, vol. 7, no. 2, pp. 139–160, 2009.

[30] J. Gutierrez-Garcia, F. Ramos-Corchado, and J. Koning, "An obligation-based framework for web service composition via agent conversations," *Web Intelligence and Agent Systems*, vol. 10, no. 2, pp. 135–150, 2012.

[31] M. Li, Q. B. Vo, and R. Kowalczyk, "Majority-rule-based preference aggregation on multi-attribute domains with cp-nets," in *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Taipei, Taiwan, 2011, pp. 659–666.

[32] ——, "Efficient heuristic approach to dominance testing in cp-nets," in *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Taipei, Taiwan, 2011, pp. 353–360.

[33] K. Yue, W. Liu, X. Wang, and J. Li, "Approach for measuring quality of web services based on the superposition of uncertain factors," *Computer Research and Development*, vol. 46, no. 5, pp. 841–849, 2009.