Web Service Selection with Incomplete or Inconsistent User Preferences

Hongbing Wang[1], Shizhi Shao[1], Xuan Zhou[2], Cheng Wan[1], and Athman Bouguettaya[2]

[1] School of Computer Science and Engineering, Southeast University, China
{hbw,szs,chw}@seu.edu.cn
[2] CSIRO ICT Centre, Australia
{xuan.zhou,athman.Bouguettaya}@csiro.au

Abstract. Web service selection enables a user to find the most desirable service based on his / her preferences. However, user preferences in real world can be either incomplete or inconsistent, such that service selection cannot be conducted properly. This paper presents a system to facilitate Web service selection in face of incomplete or inconsistent user preferences. The system utilizes the information of historical users to amend the active user's preference, so as to improve the results of service selection. We present a detailed design of the system and verify its efficiency through extensive experiments.

1 Introduction

As an increasing number of Web services have been deployed on the Web, service selection is becoming an important technique for helping users identify desirable Web services. To conduct effective service selection, we need (1) a model to adequately describe users' requirements or preferences over the nonfunctional properties of services, such as Quality of Web Service, and (2) an intelligent algorithm to select services according to a user's preferences. In recent years, a number of solutions have been proposed to address these two issues.

Most of existing solutions perform service selection based on quantitative criteria, such as a utility function [1,2]. These quantitative approaches are computationally efficient. However, they offer limited usability to end users, as it is difficult for users to express their preferences using quantitative metrics [2], such as *Utility(Qantas Airline)=0.9* and *Utility(Thai Airline)=0.7*. In many cases, users tend to express their preferences in a qualitative way, such as "I prefer *Qantas Airline* to *Thai Airline*". To obtain better usability, a number of qualitative methods [3,4] have recently been proposed to model users' preferences and to perform service selection.

Qualitative Web service selection is faced with a number of challenges as well. On the one hand, users may not provide complete descriptions of their preferences, such that service selection may produce too many results. On the other hand, as users are not completely rational, they may provide inconsistent

L. Baresi, C.-H. Chi, and J. Suzuki (Eds.): ICSOC-ServiceWave 2009, LNCS 5900, pp. 83–98, 2009.

descriptions of their preferences, such that no result will be obtained. According to [5,6], these cases are quite common in real life. To perform effective service selection, we need an intelligent system that is able to automatically complement users' incomplete preferences and remove inconsistencies.

This paper proposes a system for conducting qualitative Web service selection in face of incomplete or conflicting user preferences. To enable effective service selection, it finds a number of historical users with similar preferences, and uses their preferences to amend the preference of the active user. Then, it conducts service selection using the amended preferences to obtain improved results. This approach is in spirit similar to that of recommender systems [7,8]. We present a detailed design of this service selection scheme, which include the technique for finding similar users and the scheme for preference amendment. An experimental evaluation has been conducted to verify its efficiency and effectiveness.

The rest of the paper is organized as follows. Section 2 gives some background on qualitative service selection and recommender system. Section 3 presents our general service selection framework. Section 4 presents the heuristics and the algorithms for amending users' preferences. Section 5 gives the results of our experimental evaluation. Finally, Section 6 provides a conclusion.

2 Background

We first give a brief overview of Web service selection, and proceed to review the technologies of Conditional Preference Network (CP-Net) [10] and Recommender System.

2.1 Web Service Selection

In a typical scenario of service discovery, a user describes a desired service, and an agent identifies the relevant services or service compositions that satisfies the user's requirements. The entire process actually consists of two steps, as illustrated in Fig. 1. First, an abstract service or abstract service composition is identified, which offers the conceptual functionality required by the user. For example, if a user requests a service to store a data set, this step would returns an abstract service called Data Storage. If the user requires that her information be stored securely, this step would return an abstract composition consisting of a Data Encryption service and a Data Storage service. While the abstract service or composition is correct in functionality, it is not executable. In the second step, a set of concrete services are selected to turn the abstract service or composition into executable process. For example, either *Faidu File System* or *Doogle Database* can be selected to provide Data Storage service. Either *Universal Protection* or *PGP Cypher* can be selected to provide the Data Encryption service. *Service selection*, also known as service optimization [9], refers to the second step. Its objective is to select the concrete services which can best satisfy the user. The level of satisfaction of a user is mostly determined by a service's nonfunctional features, such as reliability, latency and etc. Therefore, service selection always

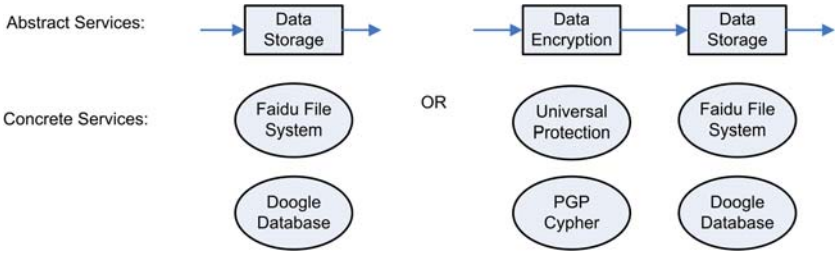


Fig. 1. Example of Service Selection

focuses on comparing the nonfunctional attributes of concrete services. However, as different users have different options on services' goodness, users' preferences are important information for conducting effective service selection.

2.2 CP-Net

Conditional Preference Network (CP-Net) [10,11] is a widely used model for qualitatively representing users' preferences. This model can be briefly defined as follows.

Definition 1 (CP-net). Let $V = \{X_1, ..., X_n\}$ be a set of attributes of Web services. A CP-net over V is a directed graph G (called *dependency graph*) over $X_1, ..., X_n$, in which each node is annotated with a Conditional Preference Table, denoted by $CPT(X_i)$. Each conditional preference table $CPT(X_i)$ associates a total order of X_i's values with each instantiation of X_i's parents. □

We illustrate the semantics of CP-net using the example in Fig. 2. A Data Storage service can be described by a number of attributes. They include *Platform*, which can be a file system or a database, *Location*, which can be USA or China, and *Provider*, which can be a private company or a public organization. As shown in Fig. 2 (a), the user has an unconditional preference on *Platform*. As indicated by the corresponding CPT, she always prefers databases to file systems. The user's preference on *Location*, however, depends on the platform she chose. As a file system offer less data processing capability than a database, the user may consume much more I/O traffics when using a file system. If the user is located in China, most likely she would like the file system to be located in China too. On the other hand, if the platform is a database, she prefers it to be located in USA, as she believes that database technologies in USA are more sophisticated. Moreover, the user's preference on *Provider* depends on the location of the service. For service providers in USA, she believes that private companies are more trustworthy than public organizations. For service providers in China, she believes that public sectors are more trustworthy than private companies. Based on the CP-net presentation of the user's reference, we can deduce the detailed preference graph of her, which gives the user's explicit preferences among all

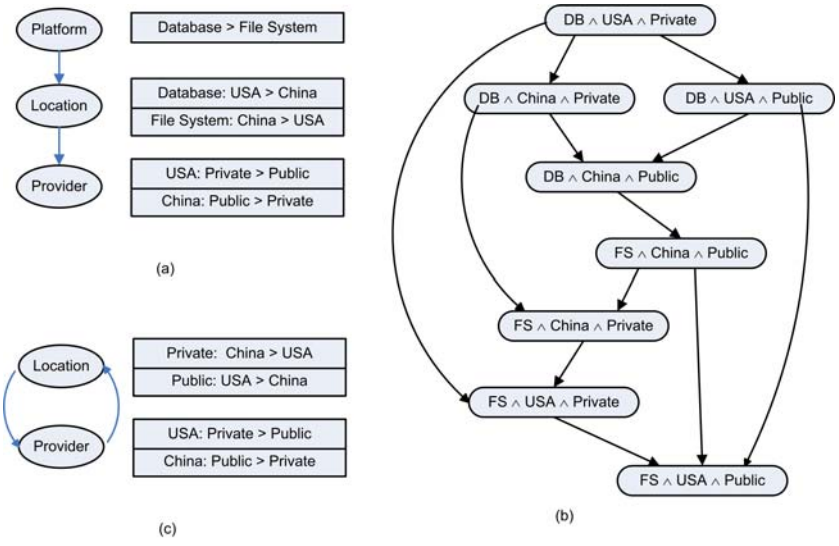


Fig. 2. Examples: (a) CP-net, (b) Induced Preference Graph, (c) Inconsistent CP-net

types of services. This induced preference graph is shown in Fig. 2 (b). Database services provided by private companies in USA are the user's first choice.

In real-world settings, a user may not want or be able to give a complete CP-net presentation of her preferences. For instance, the user's preference over platform in Fig. 2 (a) can be missing. In this situation, $FS \wedge China \wedge Public$ and $DB \wedge USA \wedge Private$ become incomparable. When preference specifications in a CP-net are sparse, service selection may not be useful anymore, as there can be too many candidate services that are possibly optimal. In a worse case, a user's specifications in the CP-net can be semantically inconsistent. As illustrated in Fig. 2 (c), a user may specify that the attributes Location and Provider are mutually dependent, and give four conditional preferences. However, based on the user's specification, we find conflicts in the induced preferences. (We can deduce both $China \wedge Public \succ USA \wedge Private$ and $USA \wedge Private \succ China \wedge Public$.) In this case, no optimal service can be found. According to [5,6], as users are not complete rational, such cases are very common.

Existing techniques for service selection are unable to deal with the above scenarios. In this paper, we provide solutions to service selection when information of user preferences is incomplete or conflicting.

2.3 Recommender System

Recommender system [7,8] is a technology attempting to select information items that are likely to be interesting for users. It analyzes a user's profile and predicts the user's interests through statistical methods. We found that similar approaches can be applied to complement a user's incomplete preferences or to

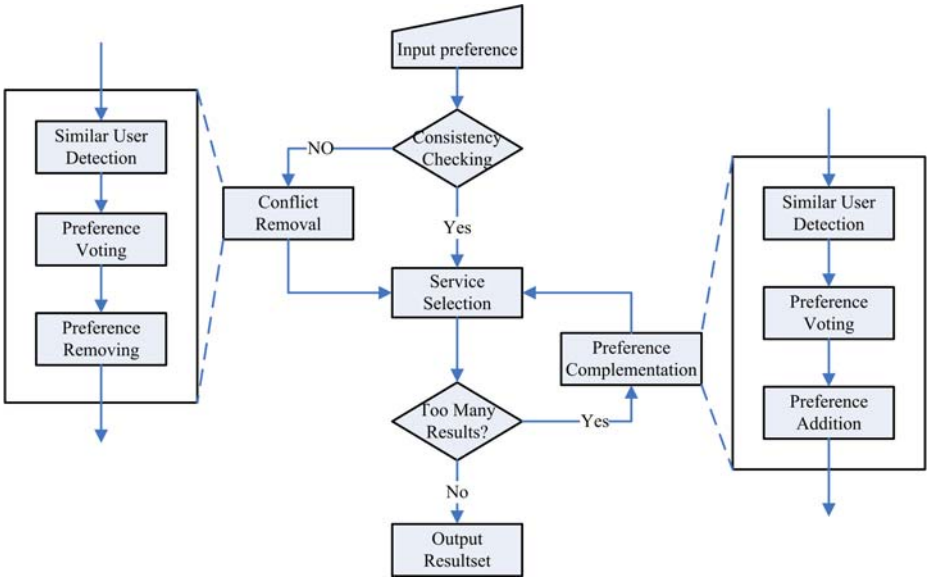


Fig. 3. Process of Service Selection

fix a user's inconsistent preferences. The most typical technique used in recommender system is collaborative filtering [12]. Collaborative filtering utilizes the regular pattern that like-minded users tend to have similar interest. It compares users' profiles to select users who share similar characteristic with the active user. Then it aggregates the interests of these like-minded users to predicate the possible interests of the active user. The method has been successfully applied to a number of leading commercial Web-sites, such as Amazon and Ebay. To solve the problems in service selection, we borrow the idea of collaborative filtering. We find historical users who share similar preferences with the active user, and use their preferences to amend the active user's preferences, such that service selection can be successfully conducted. In the following, we present a detailed design of this approach.

3 Service Selection Framework

The complete process of service selection in our system is shown in Figure 3. Upon receiving a user's preference description, the system first checks its consistency. If it contains conflicts, which are represented as cycles in the induced preference graph, a conflict removal process is conducted to remove all conflicts (cycles). The amended preference description is then passed to the service selector to retrieve the user's favorite Web services. If the result set is too big to be handled by the user, which means that the user's preferences is under-specified, the preference description is passed to the preference complementation process,

which will find some additional preferences the user would probably agree. Following that, service selection is performed again to refine the result set. This process can be repeated until the result set is manageable or no more complementation can be made.

The figure also shows the sub-steps of conflict removal and preference complementation. Both processes utilize the profiles of historical users. To remove a conflict, it first finds the users that are most similar to the active user. Then, a voting process is conducted among these users to identify the most *unimportant* preference involved in the conflict. This unimportant preference is thus removed to break the conflict. To complement a user's preferences, instead, the most *important* preference is selected and added to the active user's preferences.

4 Preference Amendment

Our approach of service selection is based on a single pattern – similar users tend to have similar preferences. Hence, the key issues of preference amendment include (1) how to find similar users and (2) how to amend a user's preferences based on the other users' profiles.

4.1 Similar User Detection

To identify similar users, we compare the current user's preferences against the preferences of other users. The users with the most similar preferences are selected. As introduced in Section 2, we describe a user's preferences using CP-net. Thus the similarity between two users can be measured by the similarity between their CP-nets. An intuitive measure of this similarity is defined as follows.

Definition 2 (Distance between CP-nets). Let A and B be two CP-nets of an abstract service composition. Let $P(A)$ and $P(B)$ be the induced preference graphs of A and B respectively. Let e denote an edge in a preference graph. Thus, the distance from B to A is calculated as:

$$Dis(A:B) = \frac{|\{e : e \in P(A) \wedge e \in P(B)\}|}{|\{e : e \in P(A) \vee e \in P(B)\}| - |\{e : e \in P(A) \wedge e \in P(B)\}|}$$

□

According to Definition 2, the distance between CP-net B and CP-net A can be measured by the size of the overlap between A and B's induced preference graphs (as illustrated in Fig. 2 (b)) divided by the size of the non-overlapping parts. While this measure of distance is intuitive, its computation can be very expensive. According the definition of CP-net, the size of an induced preference graph grows exponentially with the number of attributes of services. Therefore, when a large number of attributes are considered in service selection, it will be infeasible to use Definition 2 to compute users' similarity. Fortunately, we can largely reduce the cost by utilizing the characteristics of CP-nets.

Given a particular abstract service or service composition, we assume that different users' CP-nets share the same dependency graph. This assumption is based on two facts. First, the dependencies among the attributes of a certain service type are usually determined by the inherent characteristics of these attributes themselves. For instance, as illustrated in Fig. 2, the dependency between Location and Provider is determined by the correlation between the quality of service and these two attributes. In contrast, it is difficult to argue that a dependency exists between Location and Platform. As another example, *Destination* and *Hotel* are two attributes of a Travel service. It is easy to understand that Hotel depends on Destination, as a tourist a choice of hotel usually depends on where he is visiting. However, it is difficult to justify that Destination depends on Hotel. Second, even when users specify different dependency graphs in their CP-nets, we can create a common dependency graph for them by combining their dependency graphs into one. The users' CP-nets can be adjusted accordingly to fit the more complex common dependency graph, without varying their semantics. When CP-nets share a common dependency graph, their distances can be directly calculated from their CPTs.

Lemma 1. Let $\{X_1, ..., X_n\}$ be the attributes of an abstract service S. Let $D(X_i)$ denote the set of attributes which X_i depends on. Let $R(X_i)$ be the set of values that can be assigned to X_i. Then, given a CP-net, each conditional preference in $CPT(X_i)$ forms $\prod_{X_j \notin D(X_i)} |R(X_j)|$ edges in the induced preference graphs. □

For instances, in Fig. 2, the preference *Database* $\succ$ *File System* determines four edges in the induced preference graph, while the preference *China: Public* $\succ$ *Private* determines two edge in the induced preference graph. According Lemma 1, we can compute the distance between two CP-nets using the following formula.

Theorem 1. Let $\{X_1, ..., X_n\}$ be the attributes of an abstract service S. Let A and B be two CP-nets of S which share the same dependency graph. Let $D(X_i)$ denote the set of attributes which X_i depends on. Let $R(X_i)$ be the set of values that can be assigned to X_i. Then, the distance from B to A can be calculated by:

$$Dis(A:B) = \frac{\sum_{X_i} \left(|CPT_A(X_i) \cap CPT_B(X_i)| \times \prod_{X_j \notin D(X_i)} |R(X_j)| \right)}{\sum_{X_i} \left(|CPT_A(X_i) \cup CPT_B(X_i) - CPT_A(X_i) \cap CPT_B(X_i)| \times \prod_{X_j \notin D(X_i)} |R(X_j)| \right)}$$

□

As discussed previously, it is expensive to compute the distance between CP-nets by counting the overlapped edges in the induced preference graphs. By applying Theorem 1, the computational cost can be reduced to the order of the size of CP-nets. Specifically, the cost is linear with the number of conditional preferences in the CPTS.

4.2 Preference Voting

Using distances between CP-nets, we can identify users with similar preferences. When a user's preference is incomplete or inconsistent, it can be amended using the preferences of his / her like-minded users. As we assume that different users' CP-nets share a common dependency graph, by incompleteness or inconsistency, we always refer to the conditional preferences in the CPTs. We apply the idea of collaborative filtering. If a user's preferences, i.e., the conditional preferences in her CPTs, is incomplete, we add to it some additional preferences which are most supported by the like-minded users. If a user's preferences contain a conflict, we find all the preferences involved in the conflict, and remove the one that is least supported by the like-minded users. To measure how much an individual preference is supported by a group of users, an voting scheme is utilized. If the preference can be deduced from a user's CP-net, we regard that the user votes for this preference. In the end, the preferences with the most votes are candidates for complementing an incomplete CP-net. The preferences with the least vote are candidates to be removed to break a conflict.

4.3 Conflict Removal

To remove conflicts from a CP-net, we need to first identify conflicts, which are actually cycles in the induced preference graph of the CP-net. As a number of algorithms for conflict detection or consistency check in CP-nets have been proposed [13,14], our system directly reuses them to detect conflicts (cycles). Once a cycle in the induced preference graph is detected, we go through its edges to find the corresponding conditional preferences in the CPTs. These preferences are candidates to be removed from the CP-net. Finally, our voting scheme is applied to determine the final preference to be removed.

According to Lemma 1, a conditional preference in a CPT can correspond to more than one edges in the induced preference graph. When choosing the most suitable conditional preference to remove, we consider two factors. First, the preference should be supported by as less like-minded users as possible. This indicates that the preference is likely to be a incorrect one, as most like-minded users do not have it. Second, the preference should correspond to as less edge in the induced preference graph as possible. This ensures that removal of the preference would not affect the user's preference graph too much. Let P be a conditional preference in the CPT of the attribute X. Let $R(X)$ be the attributes which X depends on. Let $Votes(P)$ be the number of votes P receives from the like-minded users. We use the following score to measure the suitability of removing P from the CP-net.

$$Score(P) = Votes(P) \times \prod_{X_j \notin D(X_i)} |R(X_j)| \tag{1}$$

The score is actually the production of the two factors mentioned above. Our system always chooses the preference with the lowest score to remove.

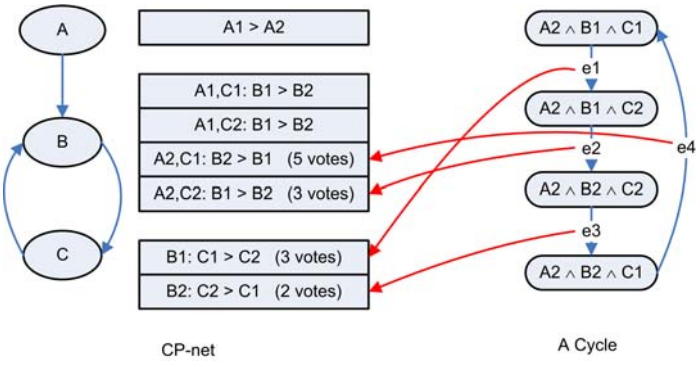


Fig. 4. Example of Conflict Removal

An example of conflict removal is shown in Fig. 4. The CP-net in the figure is inconsistent, as its induced preference graph contains a cycle, as shown on left of Fig. 4. The edges of the cycle, i.e. $e1, e2, e3, e4$, are induced from the conditional preferences $B1 : C1 \succ C2$, $A2, C2 : B1 \succ B2$, $B2 : C2 \succ C1$ and $A2, C1 : B2 \succ B1$, respectively. Thus, these preferences are candidates to be removed from the CP-net. Based on Formula 1, the scores of the preferences are:

$$Score(B1 : C1 \succ C2) = 3 \times 2 = 6,$$
$$Score(A2, C2 : B1 \succ B) = 3 \times 1 = 3,$$
$$Score(B2 : C2 \succ C1) = 2 \times 2 = 4,$$
$$Score(A2, C1 : B2 \succ B1) = 5 \times 1 = 5.$$

Based on the scores, $A2, C2 : B1 \succ B2$ is finally removed from the CP-net. As we can see, even though $B2 : C2 \succ C1$ got the least votes, because it is a more significant preference, our conflict removal algorithm did not choose to remove it.

4.4 Preference Complementation

To complement a CP-net, we consider the unknown conditional preferences in the CPTs. Based on the voting of the like-minded users, the conditional preferences with the most votes is chosen to be added to the current CP-net. When adding a conditional preference in CPTs, it is important to ensure that the resulting CP-net should not contain conflicts. The conditional preferences that will form cycles in the induced preference graph are not considered in preference complementation. Preference complementation is an incremental process, in which one conditional preference is added to the CPT-net at a time. The process stops until the number of services returned by service selection is sufficiently small (e.g., less than 20 services) or no more preference can be added to the current CP-net.

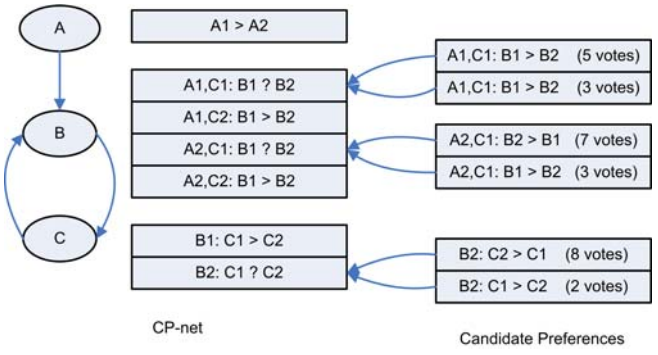


Fig. 5. Example of Preference Complementation

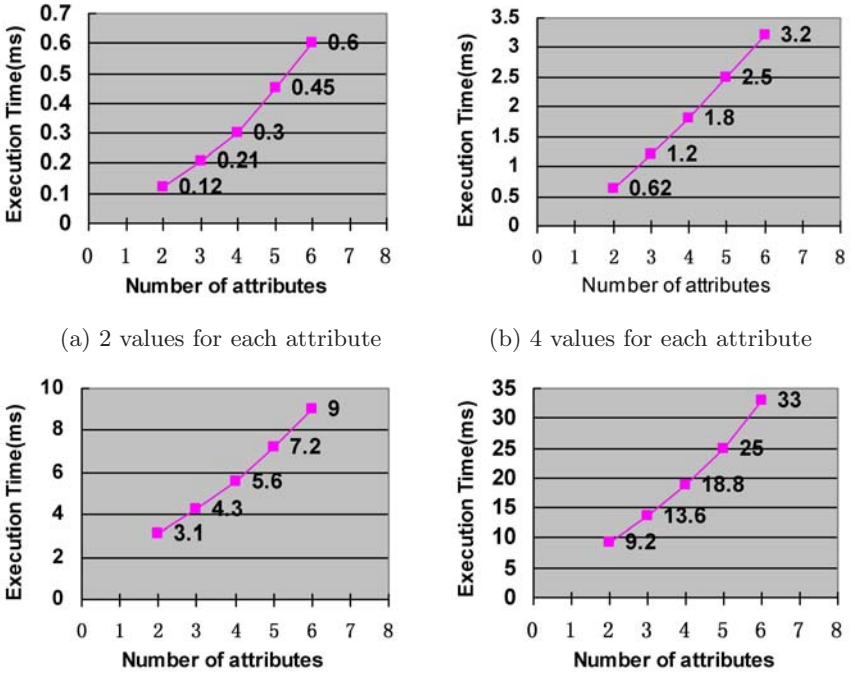
Fig. 5 shows an example of preference complementation. Three conditional preferences, i.e., $A1, C1 : B1?B2$, $A2, C1 : B1?B2$ and $B2 : C1?C2$, are unknown in the CPTs. Based on the voting results, which are shown on the left of Fig. 5, $B2 : C2 \succ C1$ is the most common preference among the like-minded users. Then, it is first chosen to be added to the current CP-net. If the results of service selection are still not satisfactory, the preference with the second highest votes is considered, and so on. As shown in Fig. 4, because $A2, C1 : B2 \succ B1$ will cause conflict, although it has many votes, we have to ignore it in preference complementation. Instead, $A1, C1 : B1 \succ B2$ is used to further complement the CP-net.

5 Experiment

As it is difficult to find sufficient real-world services and user records, we performed simulation to evaluate the efficiency and effectiveness of our approach. This section presents our results.

5.1 Simulation Setup

We simulated the scenario of service selection using randomly generated services and user preferences. To simulate different types of services, we varied the number of attributes and the number of possible values of each attribute. For each type of service we randomly generated 10,000 concrete services, which have different attribute values. To simulate user preferences, we generated random CP-nets. As mentioned previously, for each type of services, all users' CP-nets share a common dependency graph. In our simulation, we generated a random graph to represent each of the dependency graphs. Based on the dependency graph, we generate 5,000 sets of random CPTs to represent 5,000 historical users. Each CPT is filled with random conditional preferences, each of which is a random order of the attribute values. To simulate real-world situations, we



(a) 2 values for each attribute

(b) 4 values for each attribute

(c) 8 values for each attribute

(d) 16 values for each attribute

Fig. 6. Efficiency of Conflict Removal

divided the 5,000 users into 10 groups. Each group of users was based on a single CP-net with complete CPTs. We duplicated the CP-net for 500 times, and randomly varied and removed the conditional preferences in their CPTs, to obtain 500 incomplete CP-nets. Each CP-net then represented a user within that group. As a result, the users in a single group were similar to each other, and those from different groups were different. This enabled our system to easily find like-minded users.

To perform service selection, we randomly picked a service type and randomly selected a user from the 5,000 historical users, and executed the process in Fig. 3 to select the optimal service for that user. We repeated the whole process for multiple times, and recorded the average execution time of each step as well as the statistics of the result sets.

We implemented the service selection system using Java. The processes of conflict removal and preference complementation were based on Section 4. We reused the algorithm of [4] to perform CP-net based service selection. Our simulation was conducted in a personal computer with a CPU of 1.79GHz and a RAM of 768M. The operating system was Windows XP.

5.2 Efficiency of Conflict Removal

In the first set of experiments, we assessed the efficiency of conflict removal. We varied the number of service attributes involved in a conflict from 2 to 6, and the number of attribute values from 2 to 16. We repeated the process of service selection for 100 times and calculated the average execution time for each conflict removal step. The results are shown in Fig. 6.

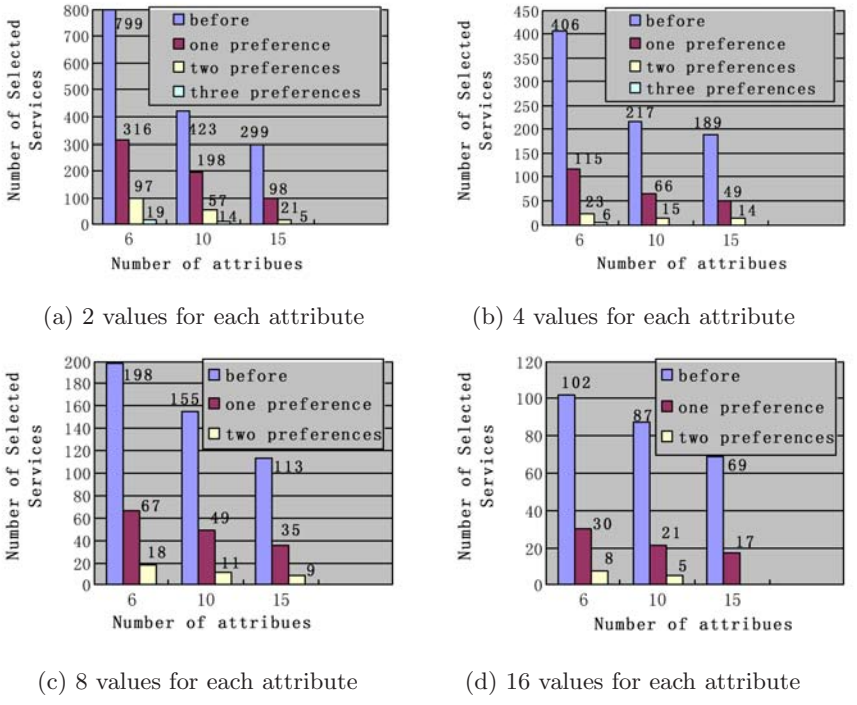
As shown in the results, the performance of conflict removal is scalable with respect to the number of attributes and the number of attribute values. According to Fig. 3, the process of conflict removal consist of two steps, that is, identifying similar users and removing the least supported preference. As discussed in Section 4.1, the cost of computing CP-net distance is linear with the size of CP-net. Thus, the cost of identifying similar users is also linear with the size of CP-net. To remove the least supported preference, the system needs to go through all the conditional preferences involved in the conflict. Its cost is therefore linear with the size of CP-net too. When the number of attributes and the number of possible values increase, the size of CP-net normally does not increase significantly. Therefore, the execution time does not increase significantly too. This justifies the performance shown in Fig. 6.

5.3 Efficiency and Effectiveness of Preference Complementation

In the second set of experiments, we assessed the effectiveness and efficiency of preference complementation. We varied the number of service attributes from 6 to 15, and the number of attribute values from 2 to 16. We also varied the degree of completeness of user preferences. When the number of attributes is 6, we set users' CP-nets to be 50% complete. When the number of attributes is 10, we set users' CP-nets to be 20% complete. When the number of attributes is 15, we set users' CP-nets to be 10% complete. We repeated the process of service selection for 100 times. We recorded the average execution time for each complementation step and the number of selected services after each step.

Fig. 7 shows the numbers of services returned by service selection before and after each step of preference complementation. We assumed that preference complementation stops when less than 20 services are returned. We can see that when a user's preference description is incomplete, the number of services returned by service selection can be too many for the user to evaluate. When additional preferences are added to the description, the result set of service selection can be significantly reduced. As shown in Fig. 7 (a), by adding 3 preferences, the result set were reduced from 800 to only 20. The experiment results indicate that preference complementation is effective in pruning services.

Fig. 8 shows the efficiency of preference complementation. According to Fig. 3, the process of preference complementation consist of two steps, that is, identifying similar users and adding the most supported preference to the user's CP-net. As discussed previously, the cost of both steps is linear with the size of CP-net. When we increase the number of attributes and the number of possible attribute values, the size of CP-net normally does not increase significantly. Therefore, the



(a) 2 values for each attribute

(b) 4 values for each attribute

(c) 8 values for each attribute

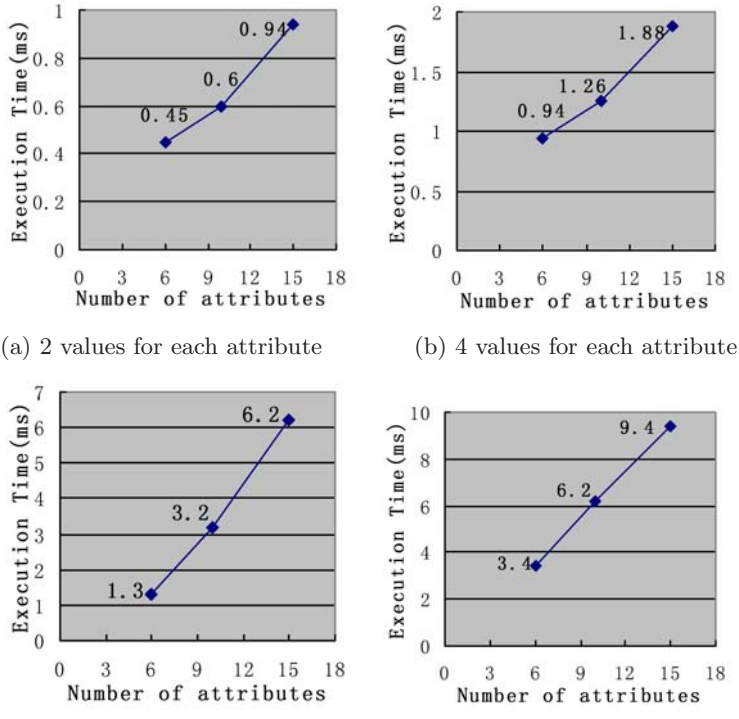
(d) 16 values for each attribute

Fig. 7. Effectiveness of Preference Complementation

performance of preference complementation is scalable with the number of attributes and the number of attribute values.

6 Related Work

Service selection aims at helping user select the optimal service from the list of results returned by service discovery. It is an important process, when (1) users' queries are ambiguous or (2) there are too many services that meet the user's basic requirements. One approach to service selection is to provide interactive interfaces for users to refine their selection criteria. For instance, in [15] the authors proposed form based interfaces that allow user to refine the results of service discovery. In [16], the authors proposed to cluster services based on their various properties, so that users can prune services by choosing appropriate clusters. Another approach to service selection is to rank services according to users' preferences or utilities functions. The work of [1,2,4] as well as our approach fall in the second types of approach. To the best of our knowledge, little work has considered the case when users' qualitative preferences are faulty or incomplete. As this case can be common in real world, this paper proposes techniques to enable service selection in face of inconsistent or incomplete preferences.



(a) 2 values for each attribute (b) 4 values for each attribute

(c) 8 values for each attribute (d) 16 values for each attribute

Fig. 8. Efficiency of Preference Complementation

Application of recommender system in service selection is not new. In [17], the authors proposed a scheme which applies collaborative filtering to facilitate service selection. Their approach directly works on services rather than user preferences. It utilizes users' ratings on various services to identify like-minded users and predicate user desired services. When the number of services is large and users' ratings are insufficient, this approach can be ineffective. In contrast, our approach utilizes users' preferences to identify like-minded users and select services. As user preferences, e.g. CP-nets, are described in the conceptual level, it requires much less information to be effective. Moreover, it can work on arbitrarily large repositories of services.

In [4], the authors proposed an algorithm for performing service selection with incomplete preferences. While the algorithm enable service selection to be correctly conducted using incomplete preferences, it may return too many results to be handled by the user, especially when preferences are under-specified. Our approach utilizes the preferences of historical users to complement an incomplete preference, so as to reduce the result set to a manageable size.

7 Conclusion

In this paper, we present an approach of service selection that can handle incomplete and inconsistent user preferences. Our approach uses CP-nets to model user preferences. It utilizes the preferences of historical users to predicate and amend the active user's preference, so that service selection can be performed properly. We conducted simulation to test our approach. The simulation results verified the effectiveness and efficiency of our techniques in conflict removal and preference complementation.

Acknowledgement. This work is partially supported by NSFC of China (No. 60473091 and No. 60673175).

References

1. Yu, T., Lin, K.J.: Service selection algorithms for composing complex services with multiple qos constraints. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 130–143. Springer, Heidelberg (2005)
2. Lamparter, S., Ankolekar, A., Studer, R., Grimm, S.: Preference-based selection of highly configurable web services. In: WWW, pp. 1013–1022 (2007)
3. Balke, W.T., Wagner, M.: Towards personalized selection of web services. In: WWW (Alternate Paper Tracks) (2003)
4. Wang, H., Xu, J., Li, P.: Incomplete preference-driven web service selection. IEEE SCC (1), 75–82 (2008)
5. Tversky, A.: Contrasting rational and psychological principles of choice. In: Zeckhauser, R.J., Keeney, R.L., Sebenius, J.K. (eds.) Wise Choices. Decisions, Games, and Negotiations, pp. 5–21. Harvard Business School Press, Boston (1996)
6. Mellers, B.A., Schwartz, A., Cooke, A.D.J.: Judgment and decision making. Annual Review of Psychology 49, 447–477 (1998)
7. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. 17(6), 734–749 (2005)
8. Burke, R.D.: Hybrid recommender systems: Survey and experiments. User Model. User-Adapt. Interact. 12(4), 331–370 (2002)
9. Yu, Q., Bouguettaya, A.: Framework for web service query algebra and optimization. TWEB 2(1) (2008)
10. Boutilier, C., Brafman, R.I., Hoos, H.H., Poole, D.: Reasoning with conditional ceteris paribus preference statements. In: UAI, pp. 71–80 (1999)
11. Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H., Poole, D.: Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. J. Artif. Intell. Res. (JAIR) 21, 135–191 (2004)
12. Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: WWW, pp. 285–295 (2001)
13. Wilson, N.: Extending cp-nets with stronger conditional preference statements. In: AAAI, pp. 735–741 (2004)
14. Goldsmith, J., Lang, J., Truszczynski, M., Wilson, N.: The computational complexity of dominance and consistency in cp-nets. In: IJCAI, pp. 144–149 (2005)

15. Sirin, E., Parsia, B., Hendler, J.: Filtering and selecting semantic web services with interactive composition techniques. IEEE Intelligent Systems 19(4), 42–49 (2004)
16. Abramowicz, W., Haniewicz, K., Kaczmarek, M., Zyskowski, D.: Architecture for web services filtering and clustering. In: Second International Conference on Internet and Web Applications and Services, ICIW 2007, p. 18 (2007)
17. Manikrao, U.S., Prabhakar, T.V.: Dynamic selection of web services with recommendation system. In: NWESP 2005: Proceedings of the International Conference on Next Generation Web Services Practices, Washington, DC, USA, p. 117. IEEE Computer Society, Los Alamitos (2005)