Preference recommendation for personalized search

Article *in* Knowledge-Based Systems · February 2016

DOI: 10.1016/j.knosys.2016.02.016

5 authors, including:

Xuan Zhou

70 PUBLICATIONS **716** CITATIONS

SEE PROFILE

Cheng Wan

Nanjing Medical University

10 PUBLICATIONS **53** CITATIONS

SEE PROFILE

Athman Bouguettaya

University of Sydney

302 PUBLICATIONS **4,586** CITATIONS

SEE PROFILE

Knowledge-Based Systems 100 (2016) 124–136

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Preference recommendation for personalized search

Hongbing Wang[a,b,*], Shizhi Shao[a,b], Xuan Zhou[c], Cheng Wan[a,b], Athman Bouguettaya[d]

[a] School of Computer Science and Engineering, Southeast University, SIPAILOU 2, NanJing 210096, China
[b] Key Laboratory of Computer Network and Information Integration, Southeast University, SIPAILOU 2, NanJing 210096, China
[c] DKE Lab, Renmin University, China
[d] School of Computer Science and Information Technology, RMIT, Australia

ARTICLE INFO

Article history:
Received 27 June 2015
Revised 10 February 2016
Accepted 20 February 2016
Available online 27 February 2016

Keywords:
CP-nets
User preferences
Recommender systems

ABSTRACT

Conditional Preference Networks (CP-nets) are widely used to express qualitative preferences. As users are sometimes reluctant or unable to specify complete CP-nets, it prohibits personalized search from being effectively conducted. In this article, we present an approach to perform personalized search using incomplete CP-nets. We propose a preference recommendation scheme for complementing a user's CP-nets, so as to improve the accuracy of personalized search. We have conducted extensive simulation and user study to demonstrate the effectiveness of our approach.

1. Introduction

With the enrichment of products and services on the Internet, service consumers are faced with an increasing spectrum of choices. Decision making in face of a large number of choices is always a challenge, especially when decision makers have diverse tastes or criteria. Personalized search [1–5] is a technology aiming to help users identify desirable products or services based on their personal preferences. It elicits a user's preferences, expresses them in standard models, and let computer systems automatically identify products interested by the user.

In recent years, a number of models, with varying expressiveness and complexity, have been proposed to describe user preferences for personalized search. For instance, Matthijs et al. [6] presented a personalization approach that builds user profiles using users' complete browsing behavior. Kim et al. [7] proposed a tag-based personalized search model to enhance the accuracy and coverage of information retrieval. Among these models, Conditional Preference Networks (CP-nets) [8–10] is a widely used one. It is not only able to concisely express users' qualitative preferences, but also able to specify the scopes of user preferences. This makes CP-nets a suitable model for personalized search systems.

Most of the existing search systems based on CP-nets [11,12] assume that users should completely specify their preferences prior to search. This assumption may not hold in many real word circumstances. As preference elicitation and description are usually tedious and error-prone processes [13], it is not always realistic to acquire complete CP-nets expressions of users' preferences. To make personalized search work, it is essential that a system could cope with incomplete CP-nets expressions and make the best of the available information to retrieve user desired products or services.

Some relevant technologies were introduced in our previous work [14,15], which attempts to utilize incomplete CP-nets to select Web services [14]. Furthermore, we have introduced a framework based on collaborative filtering [15] to perform preference complementation. In this paper, we extend our previous work and propose a complete system for conducting personalized search using incomplete CP-nets. Firstly, we broaden the application of our approach from service selection to object search. Secondly, we propose a new and complete framework to manage the process of this method. Thirdly, based on the principle of collaborative filtering [16], we propose a preference recommendation scheme for users to incrementally complement incomplete CP-nets and resolve conflicting CP-nets during the course of personalized search. Three new algorithms are proposed and applied in different steps of personalized search. Finally, new sets of simulation and user study were conducted, to compare our approach against other approaches.

The remainder of the paper is organized as follows. Section 2 reviews some background on CP-nets and collaborative filtering.

* Corresponding author at: School of Computer Science and Engineering, Southeast University, SIPAILOU 2, NanJing 210096, China. Tel.: +86 2552090861; fax: +86 25520908801, +862552090861.
E-mail addresses: hbw@seu.edu.cn (H. Wang), shizhishao@seu.edu.cn (S. Shao), zhou.xuan@outlook.com (X. Zhou), chengwan@seu.edu.cn (C. Wan), athman.bouguettaya@rmit.edu.au (A. Bouguettaya).

http://dx.doi.org/10.1016/j.knosys.2016.02.016
0950-7051/

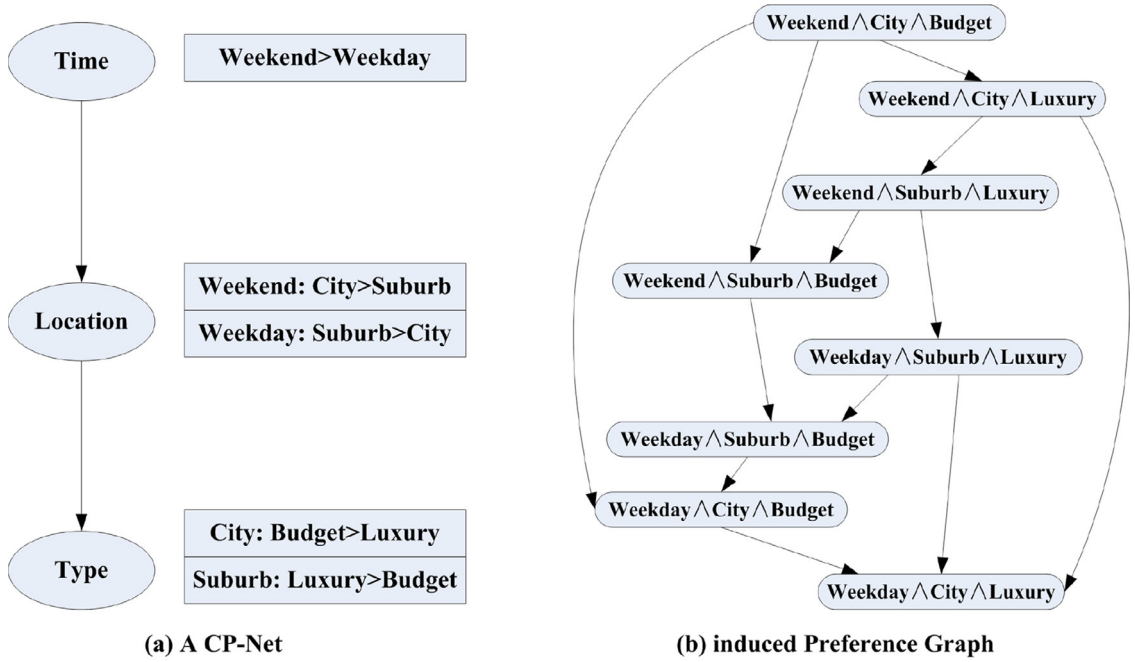


Fig. 1. CP-nets example.

Section 3 presents the personalized search system based on preference recommendation. Section 4 introduces a set of novel personalized search methods. Section 5 presents the results of the experimental evaluation and user study. Finally, Section 6 concludes the paper and provides direction for future work.

2. Background and related work

In this section, we give a brief introduction of CP-nets and collaborative filtering.

2.1. CP-nets

Conditional Preference Networks (CP-nets) [8,17] were designed for representing qualitative user preferences. The model can be defined as follows.

Definition 1 (CP-nets)**.** Let $V = \{X_1, \ldots, X_n\}$ be the set of attributes of a type of objects. The CP-nets over V is a directed graph G over $X_1,\ldots, X_n$ (which is called *dependency graph*), whose nodes are annotated with conditional preference tables, denoted by $CPT(X_i)$ for each $X_i \in V$. Each conditional preference table $CPT(X_i)$ associates a total order of X_is values with each instantiation of X_is parents.

We illustrate the semantics of CP-nets using an example in Fig. 1. Suppose Alice plans to visit Sydney and need to book a hotel on the Internet. She is then faced with a number of choices, such as *time* (i.e., on a weekend or weekday), *location* of hotel (i.e., in the city or a suburb), and *type* of hotel (i.e., a budget hotel or a luxury one). As shown in Fig. 1(a), Alice has an unconditional preference on time. Indicated by the corresponding CPT, she prefers to spend weekend in Sydney rather than weekday. Alice's preference on location, however, depends on the when she visits Sydney. If it is a weekend, she would like to stay in the city, as a lot of activities take place in the city on weekend. If it is a weekday, she prefers to stay in a suburb. Moreover, Alice's preference on the type of hotel depends on the area she is staying. If she lives in the city, she would like to book a budget hotel, as she will spend most of her time in the shopping malls rather in the hotel. If she lives in a suburb, she prefers a luxury hotel with quality service. Based on the CP-nets presentation of Alice's references, we can induce the detailed preference graph of her, which is shown in Fig. 1(b). A budget hotel in the city booked for a weekend will be Alice's first choice.

In real-world settings, a user may not be able to provide a complete CP-nets presentation of his/her preferences, especially when there are a large number attributes describing the products or services. As a result, Alice's preferences for hotels may be incomplete. As shown in Fig. 2(a), some fields in the CPT are missing. In this situation, *Weekend*∧*City*∧*budget, Weekend*∧*City*∧*Luxury* and *Weekday*∧*Suburb*∧*Luxury* become incomparable (see Fig. 2(b)). When preference specifications in a CP-nets are incomplete, personalized search will be less effective, as there can be too many candidates appearing optimal to the user.

The model of CP-nets was first introduced in [8,17], in which the authors present a number of core properties of this model as well as the algorithms for personalized search, such as the algorithms for optimal outcome generation and rankings.

In recent years, the model of CP-nets has been further expanded in some follow-up work [18]. Various techniques have been proposed to apply CP-nets to personalized search and decision making. They include the procedures to conduct CP-nets elicitation [19] and the methods to integrate CP-nets with other decision making supporting tools [20]. Recently, a CP-nets-based model for consumer-centric information service composition has been proposed [21]. However, existing techniques of personalized search rarely deal with the cases of incomplete CP-nets. In this paper, we provide solutions to personalized search with incomplete CP-nets.

2.2. Recommender system and collaborative filtering

Recommender system is a technology aiming to find products or services that can be interested by specific users. It analyzes a user's profile and predicts the user's interests through statistical methods. Then, users interests can be used to identify products. The approaches of recommender system can also be applied to preference elicitation. The most typical technique used in recommender system is collaborative filtering [16,22]. It assumes that like-minded users tend to have similar interests, and utilizes the

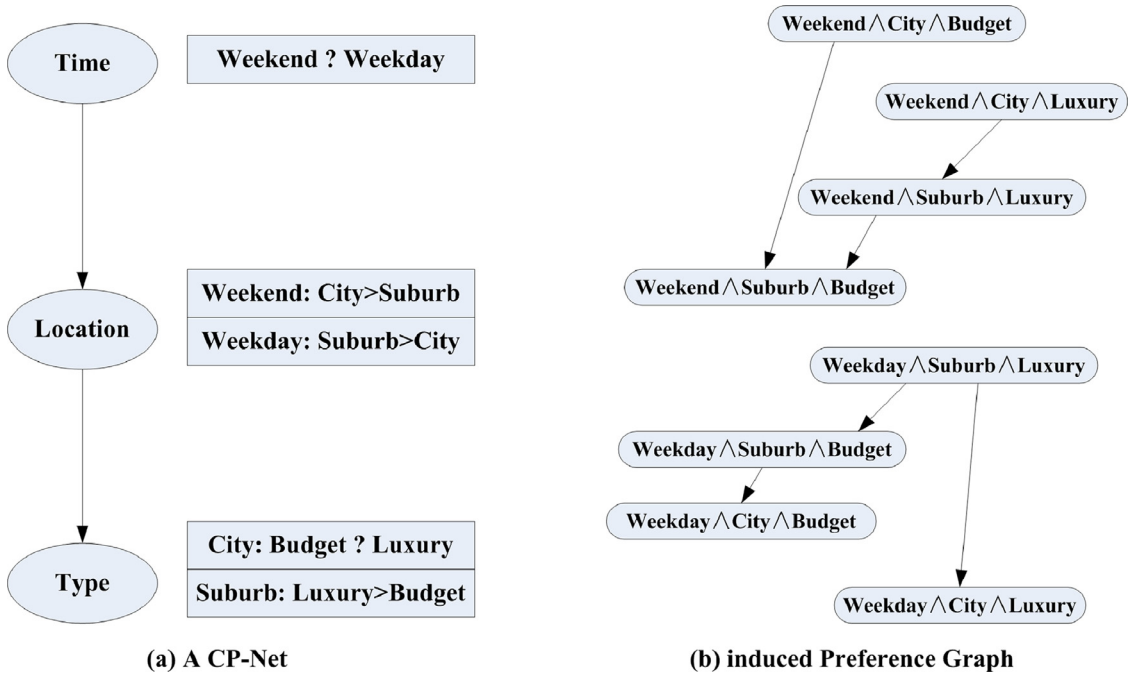


Fig. 2. An example of incomplete CP-nets.

most common pattern among users' preferences to make recommendation. To recommend products to a user, it first identifies the historical user profiles that share similar characteristic with the user. Then it analyzes the interests of the like-minded historical users, and uses their common interests to select products for the active user. The method has been successfully applied to a number of leading commercial Web-sites, such as Amazon and Ebay. To handle incomplete user preferences, we borrow the idea of collaborative filtering. We find historical users who share similar preferences with the active user, and use their preferences to complement the active user's preferences.

There has been a significant body of work on recommender system [23,24] and collaborative filtering [16,25–27]. Zhang et.al. proposed a hybrid recommender system for service selection [28], which combines collaborative filtering with fuzzy set techniques. However, most of the existing approaches work in the item level rather than in user preference level. In other words, it identifies like-minded users by comparing users' ratings on existing products and services. When the number of products and services is large, these approaches will demand a large number of users' ratings. In this paper, we exploit the possibility of applying collaborative filtering directly to user preferences. Our approach compares users' preferences (expressed in CP-nets) to identify like-minded users. Working in the preference level, this approach requires much less information to be effective. It can be applied to product repositories of arbitrary size.

3. The framework

According to the existing work on collaborative filtering [16,26,29] and the observation of our previous work [12,14], like-minded users usually have similar preferences and CP-nets. It is possible to predict a user's preferences using the preferences of his/her like-minded users. Therefore, we propose a complete system of personalized search based on preference recommendation. The framework of this system is shown in Fig. 3.

Upon receiving a user's preferences description, our system first performs *Consistency Checking*. In this step, it checks CP-nets proposed by user to verify if it contains conflict. As a number of algorithms for conflict detection or consistency checking in CP-nets have been proposed in some recent work [9,10], we skip the details of this step.

If user's preferences are consistent, *Personalized Search* will be performed to find desirable products or services for the user. And if the size of result set is acceptable, the personalized search ends. Normally, a threshold is defined to control the number of available services included in the solution set. The threshold often depends on the performance of the system and users experience. After the search, the user's preferences will be stored in the repository of user profiles.

If user's preferences contain conflicts, which are represented as cycles in the induced preference graph, a conflict removal procedure will be launched to remove all conflicts (cycles). When *remove conflicting preferences*, the individual preferences involved in the conflict (cycle) that are least supported by the like-minded users will be removed. The revised consistent CP-nets description will be passed to the personalized selector to retrieve the user's favorite products or services.

In order to improve the performance of conflict resolution, we pre-cluster historical user preferences into several clusters. Each cluster includes users with similar preferences. The step for identifying similar users or *like-minded users* is known as *Similar User Detection*. To measure how much an individual preference is supported by a group of similar users, *Preference Voting* is performed.

If the user's preferences are under-specified, the CP-nets will be incomplete. In such a case, the size of result set might be too large for a user to consume. Similarly, *Preference Voting* is performed to propose preferences to complement the CP-nets. In *Preference Complementation*, candidate preferences will be scored by the likelihood and sensitivity based on the voting results. The conditional preferences with the highest score will be chosen and added to the current CP-nets. Preference complementation is an incremental process, in which one preference is added to the CP-nets at a time. Following that, service selection is performed again to refine the result set. This process will be repeated until the result set is sufficiently small or no more complementation can be made.

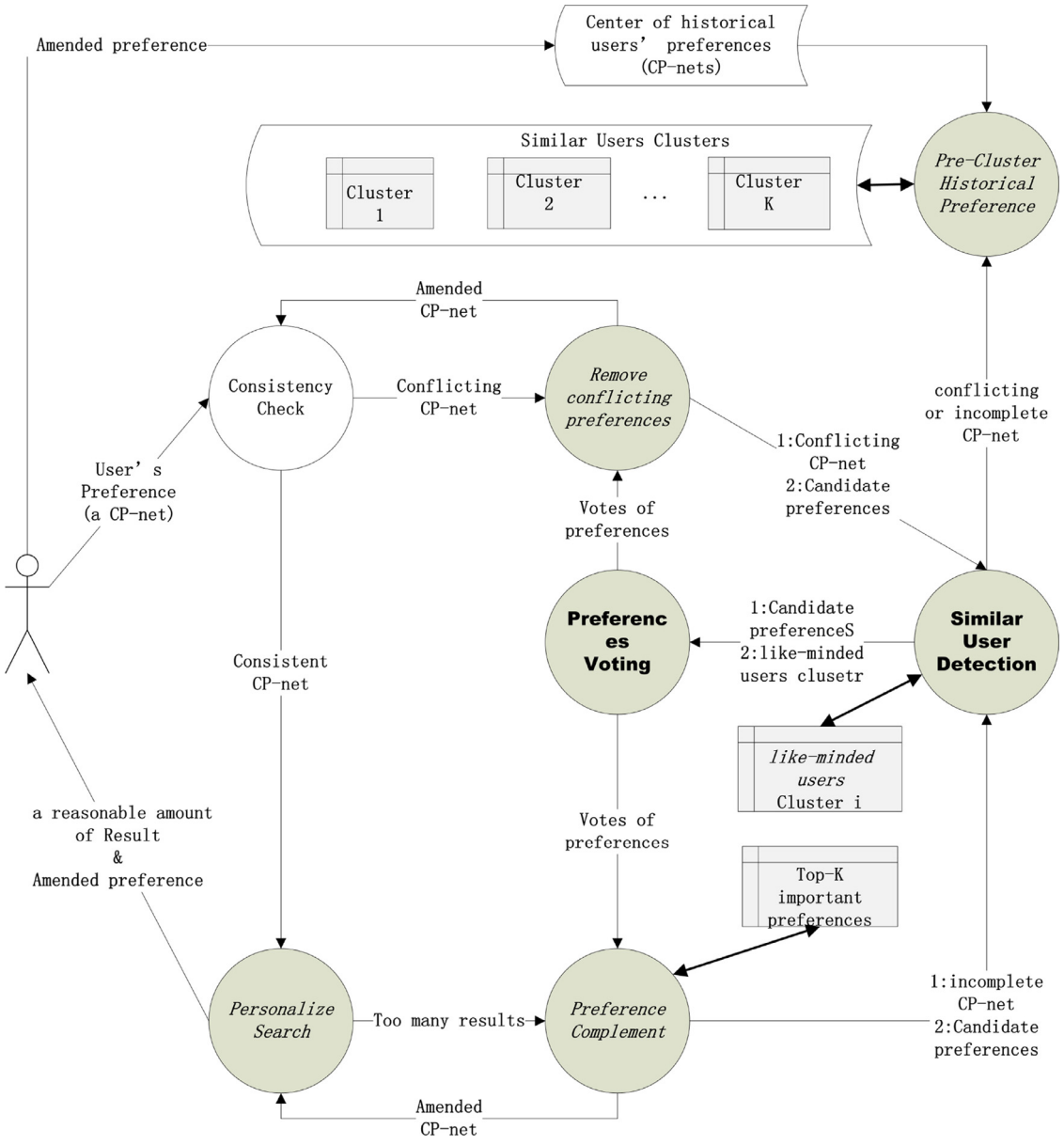


Fig. 3. The framework.

In our previous work, the algorithm for identifying similar users is complex, and the cost of preference voting can be large when confronted with a large number of historical users. Therefore, we improved the method of preference complementation in this paper. Firstly, we utilize a step of *Historical Preference Pre-Clustering* to decrease the computational complexity for identifying similar users. Secondly, we allow preference voting to return the votes of a list of preferences. Then, in the step of preference complementation, the top-K (e.g $K = 5$) preferences will be added to the active user's preferences one by one. The next round of preference voting will not be activated until the top-K preferences have been used.

4. The methods

In this section, the methods of preference recommendation scheme based on the principle of collaborative filtering [16] are presented. These methods help users incrementally complement their incomplete CP-nets or resolving their conflicting CP-nets during personalized search.

4.1. Identification of similar preference

As introduced in Section 2, we describe a user's preferences using a CP-nets. 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 object requested by users. 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 the definition, the distance between CP-nets B and CP-nets A is the size of the overlap between A and B's induced preference graphs (as illustrated in Fig. 1(b)) divided by the size of

the non-overlapping parts. To simplify the calculation of this distance, we assume that different users' CP-nets share the same dependency graph. This assumption does not necessarily hold. However, when users specify different dependency graphs in their CP-nets, we can still create a common artificial dependency graph for them. This can be easily achieved by conjoining their dependency graphs into a common super graph. Then the users' CPTs can be adjusted accordingly to fit the common dependency graph, without altering their semantics. When CP-nets share a common dependency graph, their distances can be directly calculated from their CPTs.

Lemma 1. *Let $\{X_1,\ldots, X_n\}$ 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 value range of X_i. Then, given a CP-nets, each conditional preference in $CPT(X_i)$ forms $\prod_{X_j \notin D(X_i)} |R(X_j)|$ edges in the induced preference graphs.*

The value of an attribute may be discrete or continuous. If the attribute value of X_i is discrete, $R(X_i)$ is the set of values that can be assigned to X_i. Otherwise, the continuous value will be replaced with a series of values at fixed intervals. $|R(X_j)|$ is the cardinality of the set $R(X_j)$. It is the number of members of $R(X_j)$. For instances, in Fig. 1, $D(Time) = \varnothing$, $X_j \notin D(Time) = \{Location, Type\}$, $|R(Location)| = 2$, $|R(Type)| = 2$. Accordingly, the preference *Weekend* $\succ$ *Weekday* determines four edges in the induced preference graph, while the preference *Weekend: City* $\succ$ *Suburb* determines two edge in the induced preference graph.

Theorem 1. *Let $\{X_1,\ldots, X_n\}$ 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 using the following Eq. (1).*

$$Dis(A:B) = \frac{\sum_{X_i}(|CPT_A(X_i) \cap CPT_B(X_i)| \times \prod_{X_j \notin D(X_i)} |R(X_j)|)}{\sum_{X_i}(|CPT_A(X_i) \cup CPT_B(X_i) - CPT_A(X_i) \cap CPT_B(X_i)| \times \prod_{X_j \notin D(X_i)} |R(X_j)|)} \quad (1)$$

As the size of an induced preference graph grows exponentially with the number of attributes of a CP-nets, it is expensive to compute the distance 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 in linear proportion to the number of conditional preferences in the CPTs.

4.2. Clustering historical users' preferences

Using distances between CP-nets, we can identify users with similar preferences. To speed up the process, our recommendation scheme conducts offline clustering over the CP-nets of historical users. All users within a cluster are considered like-minded users. When a new user declares his/her preferences, the system compares the distances between the user and the existing clusters, and adds the user to the cluster of the closet distance. Our algorithm (shown in Algorithm 1) of clustering is K-Means.

The inputs of the algorithm includes: the set of historical users' preferences ($\overrightarrow{\mathbf{CP}} = \{CP_i\}$), the cluster number (K) of like-minded users, and the number (T) of iterations. Let $\overrightarrow{\mathbf{CP}} = \{CP_i\}$ denote the set of CP-nets of the historical users. Firstly, K CP-nets will be selected randomly as the center of initial clusters. Secondly, an iterative process will be used to dynamically build the clusters of like-minded users until the number of iterations goes beyond T or the members in each cluster are stable. The clusters are regarded stable if the centers of clusters stop changing in the clustering process. Finally, the stable clusters of like-minded users will be returned.

Algorithm 1: Clustering.

Input: $\overrightarrow{\mathbf{CP}} = \{CP_i\}$: the set of historical users' preferences, K: the cluster number of like-minded users, T: the iterative number;
Output: $\overrightarrow{\mathbf{C}} = \{C_i, 0 < i \le K\}$:$K$ clusters of like-minded users;

```
Randomly select K users' CP-nets;
iterative=0;
Let these K CP-nets as the centers of clustering;
while (iterative < T) do
    for each CP-nets CP_i un-clustered do
        Calculate the distances between CP_i and the K centers of clustering according to Eq. (1);
    end for
    Choose the largest distance and add CP_i into the corresponding cluster C_x;
    CenterDisSumCluster=the sum of distances between the center and rest CP-nets in the cluster C_x according to Eq. (1);
    for each CP-nets CP_j in cluster C_x do
        DisSum=the sum of distances between CP_j and rest CP-nets according to Eq. (1);
        if DisSum < CenterDisSumCluster then
            set CP_j as the new center;
            CenterDisSumCluster=DisSum;
        end if
    end for
    if the new centers are same as the old ones in last round then
        return C = {C_i}, 0 < i ≤ K;
    end if
end while
```

The computational complexity of the algorithm depends on three factors: the size of the data set D(|D|), the number of clusters K and the number of iterations T. In most cases, $K << |D|$, $T << |D|$. Therefore, the computational complexity is $O(K*T*|D|)$. Considering the constraints on response time of personalized search, this part of computation could be performed regularly offline.

4.3. Similar users detection

In personalized search, the detection of similar users needs to be quick and accurate. In this section, an efficient detection algorithm based on statistics and clustering is proposed. The algorithm made a tradeoff between computational complexity and accuracy. It utilizes the pre-clustered historical preferences and statistics of the CP-nets.

In Algorithm 2 , the current user's preferences will be firstly compared against the clusters of the historical users preferences. The distance between the current user's preferences and the centers of every cluster will be calculated. And the users in the cluster whose center is closest to the current user are selected as the candidate similar users. In order to ensure accuracy, *T*-test will be used to calculate the significant difference between the nearest distance and the mean of the distances with other candidates. If there is significant difference ($sg < 0.05$), it means that the selected cluster can well represent the characteristics of the current user. Then, the users in the selected cluster will be returned as the similar users. Otherwise, it indicates that the selected cluster is not representative. In this case, the algorithm will search the whole set

Algorithm 2: Similar users detection

Input: $\overrightarrow{\mathbf{C}} = \{C_i\}, 0 < i \leq K$: K clusters of like-minded users, cp: current user's CP-net;
Output: Like-minded users cluster, $\overrightarrow{\mathbf{C_x}}$;

```
for each cluster C_i do
    Calculate the distances between the center of C_i and the current cp;
end for
C_x = the cluster with the nearest distance;
nearestDis=the largest distance of clusters;
restDis=mean of the distance of rest clusters;
sg= significant difference between the nearest distance and the rest distance;
if sg < 0.05 then
    return C_x; /*the selected nearest cluster represents significant difference*/
else
    CS=the size of C_x;
    C_x = ∅;
    for each CP-nets in C do
        PairDis=the distances between the CP-nets of CP_i and the current cp;
        if PairDis is in the most CS nearest distances then
            Add this CP-nets into C_x;
        end if
    end for
    return C_x; /*users which preference are top-N nearest to the current user;*/
end if
```

of historical users and return the top-N users (N is the size of the resulting cluster) that are nearest to the current user. The top-N users come from different clusters.

In the first case, when the detected cluster is representative enough, the computational complexity of the algorithm will be in proportion with the number of clusters K. In the second case, the algorithm will have to traverse the whole user set. Accordingly, the computational complexity will be in proportion with the size of the user set D ($|D|$).

4.4. Preference voting

According to the above Section 4.3, using distances between CP-nets, we can find users with similar preferences. When a user's preferences are incomplete or inconsistent, it can be amended using the preferences of his/her like-minded users. Similar to collaborative filtering, if a user's preferences, i.e., the contents in her CPTs, are incomplete, we will add to it some additional preferences which are most supported by the like-minded users. If a user's preferences contain a conflict (cycle), we firstly find the individual preferences involved in the conflict (cycle), 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.

In preference voting, the system checks if the candidate preferences can be included in the CP-nets based on their popularity in like-minded users. If one candidate preference can be found in one like-minded user's preferences, the candidate preference will get one vote. The computational complexity of the preference voting is the product of the number of candidate preferences and the number of like-minded users.

In the end, the preferences with the most votes are used complement an incomplete CP-net. On the contrary, the preference with the least votes is removed to break conflicts.

4.5. Removing conflicting preferences

Once a cycle/conflict in the induced preference graph is detected, the corresponding conditional preferences involved in the cycle will be the candidates to be removed from the CP-net. They are known as *candidate conflicting preferences*. As shown in Section 4.4, the preference voting scheme is applied to determine which candidate conflicting 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 need to consider two factors. Firstly, the preference should be supported by as less like-minded user as possible. Secondly, the preference should correspond to as less edge in the induced preference graph as possible. The first condition indicates that the preference is likely to be an incorrect one, as most like-minded users do not have it. The second condition ensures that removal of the preference would not affect the user's preferences graph too much. Let P be a conditional preference in the CPT of the attribute X. Let $R(X)$ be the attributes X depends on. Let $Votes(P)$ be the number of votes P receives from the like-minded users. The weighted score shown in Eq. (2) is used to measure the suitability of removing P from the CP-net. The preference with the lowest score will be removed.

$$Score(P) = Votes(P) \times \prod_{X_j \notin D(X_i)} \left|R(X_j)\right| \tag{2}$$

An example of preference voting are shown in Fig. 4. The CP-nets in the figure is inconsistent, as its induced preference graph contains a cycle, which is shown on the 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. These preferences are then considered to be removed from the CP-net. The edge $e1$ gets 3 votes, which means there are 3 similar users who share this preference. And the edges $e2$, $e3$ and $e4$ get 3, 2 and 5 votes respectively.

Based on Eq. (2), the scores of the preferences are:
$Score(B1 : C1 \succ C2) = 3 \times 2 = 6$,
$Score(A2, C2 : B1 \succ B2) = 3 \times 1 = 3$,
$Score(B2 : C2 \succ C1) = 2 \times 2 = 4$,
$Score(A2, C1 : B2 \succ B1) = 5 \times 1 = 5$.

Thus, $A2$, $C2$: $B1 \succ B2$ 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 significant preference, the conflict removal algorithm did not choose to remove it.

4.6. Preference complementation

When an active user's CP-nets is incomplete, we can then recommend him/her with the preferences of his/her like-minded users. As there could be a large number of preferences offered by the like-minded users, selection of right preferences for recommendation is an important issue. Applying the principle of collaborative filtering, if a preference is shared by more like-minded users, it is more likely that this preference is also shared by the active user. Therefore, our system lets the like-minded users to vote for the candidate preferences. If a candidate preference can be deduced from a user's CP-net, our system regards that the user vote for this preference. In the end, the preferences with more votes are more likely to be a complementary preference for the active user.

Fig. 5 shows an example of preference voting. Three conditional preferences, that is, $A1$, $C1$: $B1?B2$, $A2$, $C1$: $B1?B2$ and $B2$:

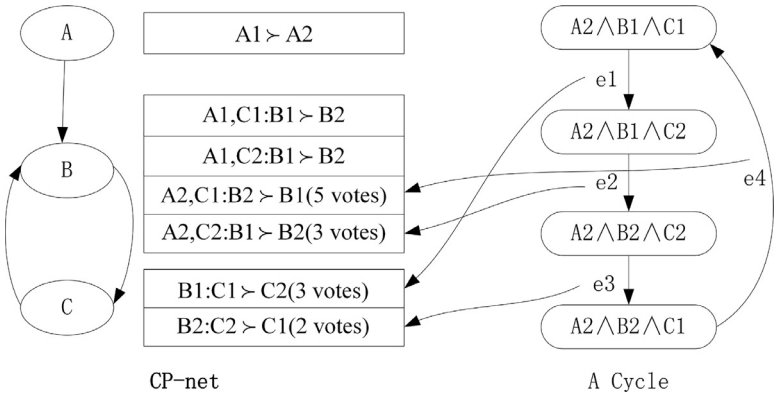


Fig. 4. Example of negative preference voting.

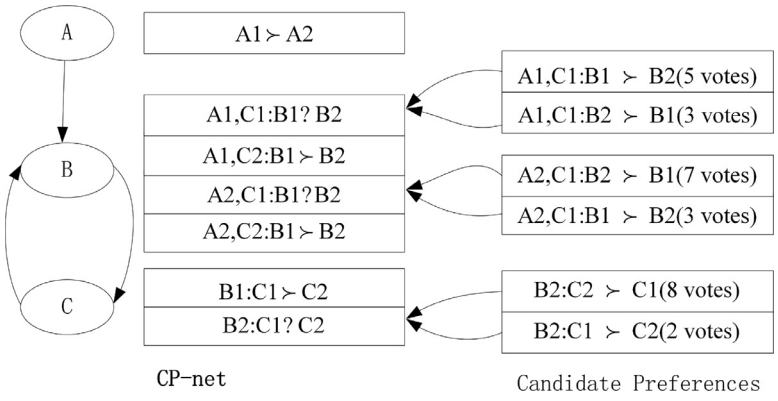


Fig. 5. Example of preference voting.

$C1?C2$, are missing in the active user's CPTs. Based on the voting results, which are shown on the left of Fig. 5, $B2$: $C2 \succ C1$ is the most common preference shared by the like-minded users. Therefore, it would be the most probable preference also liked by the active user. As $A2$, $C1$: $B2 \succ B1$ is conflicting with the existing CP-net, we have to ignore it, even though it gets the second highest votes. Instead, $A1$, $C1$: $B1 \succ B2$ is the second most likely preference that could hold for the active user. The conflict can be proven as follows:

$$\because A2, C2 : B1 \succ B2 \Rightarrow \therefore A2B1C2 \succ A2B2C2$$
$$\because B2 : C2 \succ C1(8votes), B2 : C1 \succ C2(2votes)$$
$$\therefore B2 : C2 \succ C1 \, is \, selected$$
$$\therefore A2B1C2 \succ A2B2C2 \succ A2B2C1$$
$$if \, A2, C1 : B2 \succ B1$$
$$then \, A2B1C2 \succ A2B2C2 \succ A2B2C1 \succ A2B1C1$$
$$\therefore A2B1C2 \succ A2B1C1$$
$$\therefore B1 : C2 \succ C1 \, conflict$$

The results of preference voting are used to rank candidate preferences according to their likelihood of matching a user's mind. However, likelihood is not the only factor that determines the suitability of a preference in recommendation. A preference can be highly likely, whereas it may not help the user in search if it cannot reduce the search results. Therefore, selectivity of preference should also be a factor considered in recommendation. Our recommendation scheme using the following function to assess the suitability of each candidate preference. Preferences are ranked based on the scores and recommended to users progressively.

$$score(P) = \alpha \times likelihood(P) + (1 - \alpha) \times selectivity(P) \quad (3)$$

In Eq. (3), $likelihood(P)$ measures the likelihood of the preference P in matching the user's mind. It is calculated by the number of the votes from the like-minded users. Namely,

$$likelihood(P) = \frac{votes_p}{votes_{total}} \quad (4)$$

where $votes_P$ represents the number of votes P receives, and $votes_{total}$ represents the total number of votes a preference can receive. $selectivity(P)$ measures the fraction of objects that can be pruned by applying the preference P to search. N_p denotes the number of service or products satisfying the user's preferences. And N denotes the total number of available service or products. The selectivity is calculated as

$$selectivity(P) = \frac{N_p}{N} \quad (5)$$

As $selectivity(P)$ is independent from any particular conduction of search, it can be computed in the pre-processing phase. Finally, α is a tuning factor for balancing likelihood and selectivity. It is an empirical value which enables a user to maximally prune the search results by checking the minimum recommended preferences. According to our experiments, we suggest to set α to 0.5.

4.7. Personalized search using incomplete CP-nets

In this paper, CP-nets is used to represent and model user preferences. In this section, we introduce an algorithm to generate the optimal outcome using CP-nets. The algorithm is shown in Algorithm 3 .

The inputs of the algorithm include: (1) an acyclic CP-nets which represents user's preference, (2) the attribute set for describing the objects, and (3) the set of available solutions. The algorithm is based on the idea of multi-way tree building. It starts

Algorithm 3: Personalized search.

Input: an acyclic CP-net; $AS = \{as_i, 0 < i < |AS|\}$: available solutions set;$V = \{X_1, ..., X_n\}$: attributes set of objects;
Output: R:result set of optimal solutions;

```
Build a multi-way tree T with only the root.
for each variable X ∈ V with no parent do
    Let D(X) be the value domain of X
    Get the preference order of D(X):{x_1 ≻ x_2 ≻ ... ≻ x_n};
    if T has no leaf nodes then
        Get the partition of available solutions set {AS_1, ..., AS_n},
        AS_i(1 ≤ i ≤ n) is the set of available solutions satisfied
        the value of x_i;
        Add the partition as leaf nodes into T according to the
        priority from left to right;
        Remove the X from V;
    else
        for each leaf nodes, lf, in T do
            ls=the number of leaf nodes;
            Get the partition of solutions in the node {AS_1, ..., AS_ls},
            AS_i(1 ≤ i ≤ ls) is the set of available solutions satisfied
            the value of x_i;
            Add the partition as new leaf nodes of lf into T;
        end for
    end if
end for
R=the available solutions in the most left of the non-empty
leaf nodes
return R
```

with an empty root of a multi-way tree. The variables without a parent in the acyclic CP-nets will be selected to build the partition of available solutions. One solution is a service or an object that satisfies the user's preference. The available solutions will be put into different leaf nodes in the multi-tree. And the order of leaf nodes shows user's preferences on services or products. The available solutions in the most left non-empty leaf nodes will be returned as the set of optimal solutions.

The time complexity of this algorithm is exponential w.r.t. the size of CP-net, because the entire set of non-dominated feasible solutions will grow exponentially with respect to the number of variables. However, the current set of solutions in the algorithm is always valid, which is an important property of this algorithm. Therefore, the user starts receiving results as soon as the very first solution is generated.

5. Experimental evaluation

User study and simulations as performed to evaluate the effectiveness of our approach. Firstly, we evaluate the performance of our methods in a simulated scenario. Secondly, we compare our work with other personalized search approaches. Finally, we conduct a user study to demonstrate the effectiveness of our approach in real world.

5.1. Performance evaluation

5.1.1. Experiment context

In order to make sure that the results of our experiments are not biased, we simulated the scenario of personalized search using randomly generated objects and user preferences. The number of attributes and the number of possible values of each attribute were varied to simulate different types of objects. In the experiments, 10,000 objects with different attribute values were randomly generated for each type. CP-nets were generated randomly to simulate user preferences. As mentioned previously, for each type of objects, all users' CP-nets share a common dependency graph. A random graph is generated to represent each of the dependency graph. Then, 5000 historical users were created using randomly generated CPTs. Each CPT was filled with random conditional preferences, represented by random orders of attribute values. To simulate real-world situations, the users were clustered into 10 groups. The CP-nets of each user group were variations of a single complete CP-net. Variations of a single complete CP-nets means that some CP-nets are generated from one complete CP-nets by changing a few preferences. Basically, one single CP-nets was duplicated for 500 times, and 20% of the conditional preferences of each duplicate were randomly varied, to obtain 500 CP-nets for the users in a group. These complete CP-nets were used as the users' latent real preferences. Then the conditional preferences in each duplicate were randomly removed to obtain 500 incomplete CP-nets. These incomplete CP-nets were used as the user explicit preferences. As a result, the 10 group of users formed 10 natural clusters. The CP-nets in a single group were similar to each other, and the CP-nets from different groups were different.

To perform personalized search, we picked an object type, randomly selected a user from the 5000 historical users, and used the user's explicit CP-nets to perform the search. To evaluate the performance of preference recommendation, our simulation program automatically selected the recommended preferences that existed in the user's latent complete CP-net, and observed how it pruned the research results.

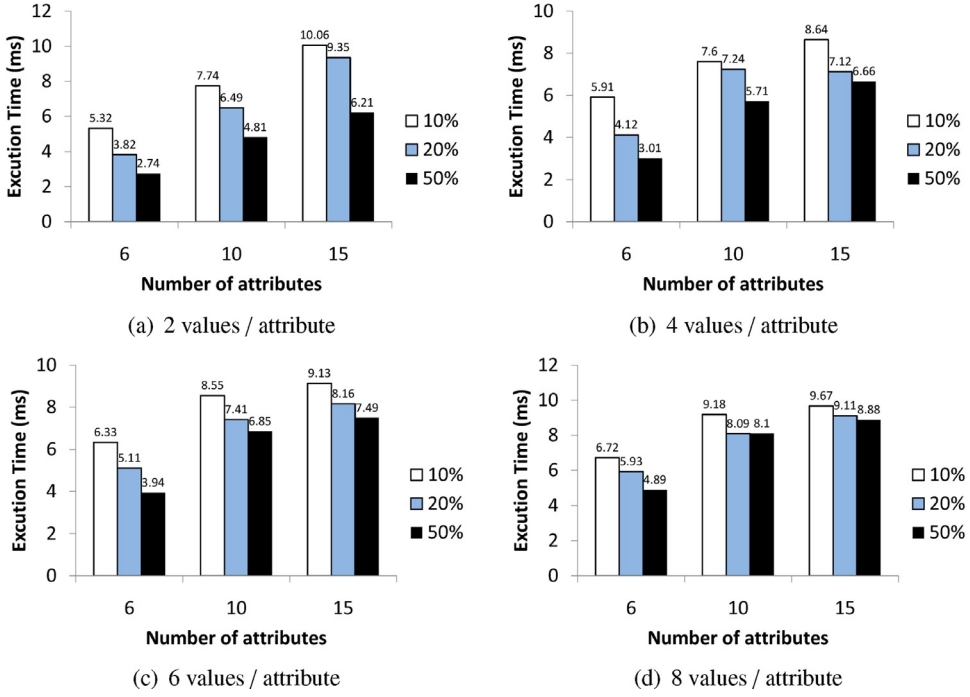
The personalized search system was implemented using Java. The simulation was conducted in a personal computer with a CPU of 2.66 GHz and a RAM of 4G.

5.1.2. Results of evaluation

In the first set of experiments, the efficiency of search recommendation were evaluated. The number of attributes in CP-nets was varied among 6, 10 and 15. The number of attribute values was varied from 2 to 16. The degree of completeness of user preferences was varied among 10%, 20% and 50%. The process of personalized search was repeated for 100 times. The average execution time of generating the recommended preferences was recorded.

Fig. 6 shows the efficiency of preference recommendation. The processes of preference recommendation consist of two steps, that is, identification of similar users and selection of recommended preferences. According to the experiments, when the attribute number and the number of attribute values are same, the more completeness is, the shorter execution time is. As the cost of both steps is proportional to the size of CP-net, the results in Fig. 6 are expected. It can be concluded that our algorithm of preference recommendation is scalable with the number of attributes and the number of attribute values.

In the second set of experiments, the effectiveness of preference recommendation in pruning search results was evaluated. The number of attributes was fixed to 8, and the number of values per attribute was set to 10. The number of objects was varied from 1000 to 125,000, and preference recommendation was applied in searching results. Fig. 7 shows the variation of average number of results when users incrementally added the recommended preferences to their CP-nets. In Fig. 7, the X axis represents the number of objects to be chosen, and the Y axis represents the number of selected objects recommended by our personalized searching. The first legend "before" annotates the results before preference complementation. The second legend "one preference" annotates the results after adding one preference recommended by our method, and so on. As we can see, the result set can be significantly reduced by incorporating additional preferences. When there were



(a) 2 values / attribute (b) 4 values / attribute

(c) 6 values / attribute (d) 8 values / attribute

Fig. 6. Execution time of preference recommendation.

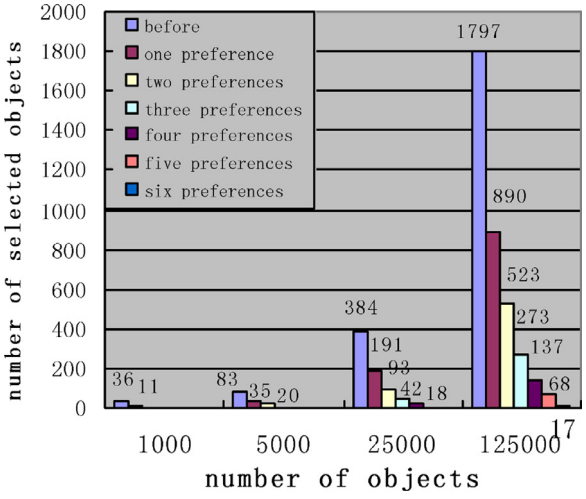


Fig. 7. Results pruning by adding preferences.

5000 objects, around 380 objects were selected using a user's incomplete CP-net. This number was reduced to 18, after 4 additional preferences were added to the CP-net. When there were 125,000 objects, by adding 6 additional preferences, the average number of results was reduced from 1797 to 17.

The recommendation scheme is designed to quickly prune search results. To assess the quality of recommendations, we measured how much user interaction the system would be incurred to reduce the result sets to a certain size (10 in our experiments). The interaction cost can be measured by two factors. The first factor is the number of recommended preferences a user has to inspect or view until he / she finds enough suitable preferences to prune the results. The second is the number of recommended preferences a user selects in the end. The less the inspected or selected preferences, the less time a user will spend in search, and thus the more effective the recommendations. In the experiments, we measured the interaction cost for all the 1000 stereotypical users, with the number of objects varying from 5000 to 125,000. Fig. 8 plots the distribution of the users over the two measure factors. The legends of this figure are as those in Fig. 7.

As Fig. 8 shows, when there are 1000 objects, the majority of users needs to inspect 2–4 recommended preferences and select 1–2 of them to reduce the result set to less than 10. When the number of objects increases, the interaction cost increases accordingly. For instance, when there are 125,000 objects, the majority of users needs to inspect 6–9 recommended preferences and select 5–6 of them to reduce the result set to less than 10. However, the increase of interaction cost is far slower than the increase of the size of object set. As indicated by the four charts in Fig. 8, when the object set grows exponentially (increase from 1000 to 125,000), the interaction cost grows only linearly (increase from 1 to 6). This is because our recommendation scheme considers the selectivity of preferences, such that each recommended preference could always filter out a certain fraction of results.

In the third set of experiments, the influence of the tuning factor α of Eq. (3) on the quality of reference recommendations is evaluated. The α was varied from 0 to 1, and the average interaction cost of search was measured. Fig. 9(a) and (b) shows the varying interaction cost on object sets with sizes of 5000 and 25,000, respectively. The figures clearly show that a user needs to inspect more preferences when selectivity is weighted over likelihood (i.e., α is small). By contrast, when likelihood is weighted over selectivity (i.e., α is large), a user may have to select more preferences to sufficiently reduce the result set. According to the experiment results, α should be set to approximately 0.5 to minimize interaction cost. The results also show that both the likelihood and the selectivity of preferences should be considered to ensure the effectiveness of preference recommendation.

5.2. Comparison with commonly used methods

Personalized search is mainly faced with two challenges. On the one hand, users may under-specify their preferences. On the other hand, users may provide inconsistent descriptions of their

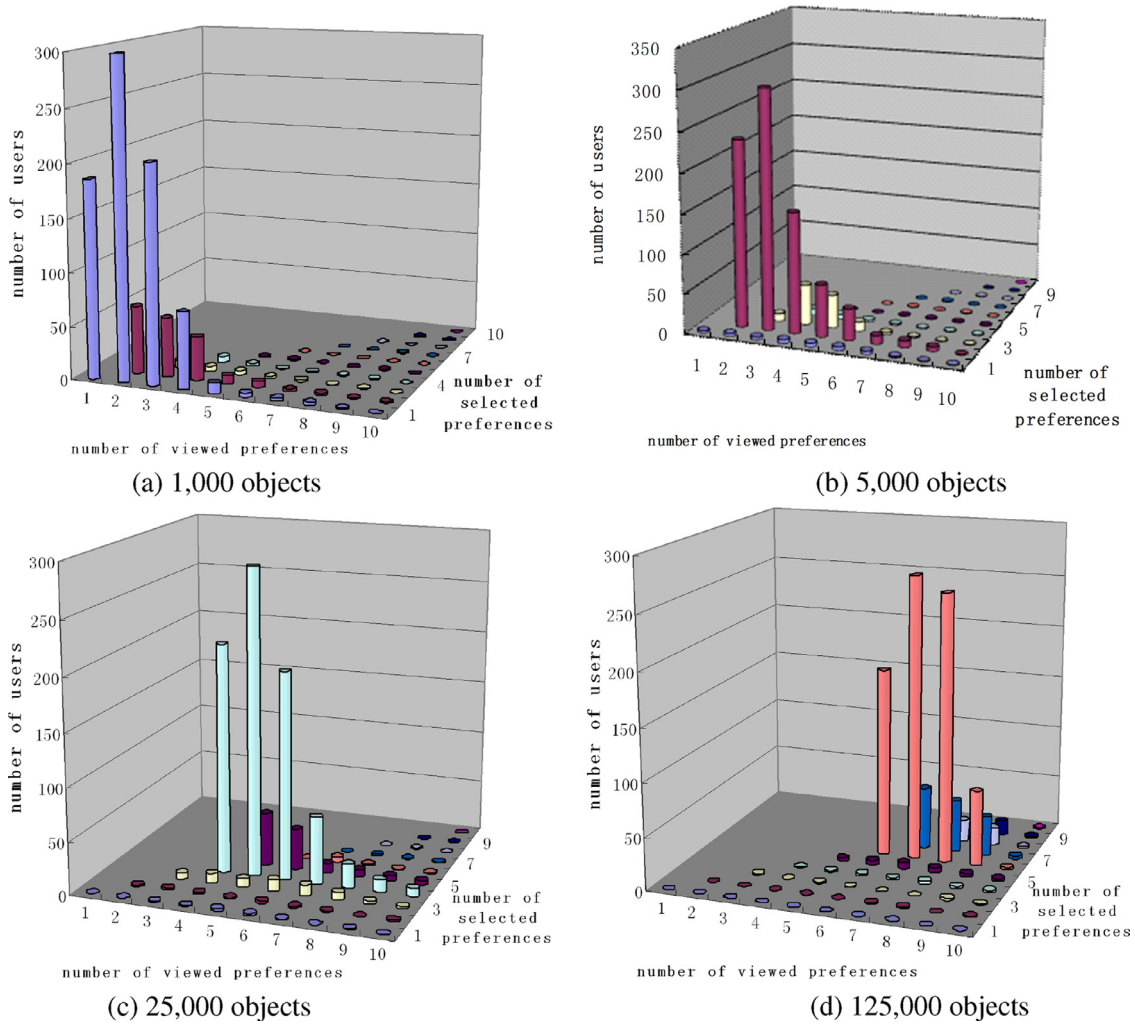


Fig. 8. Interaction cost.

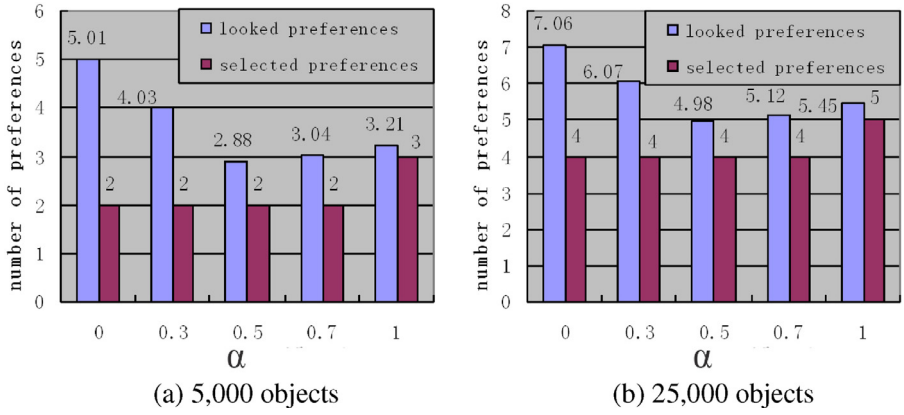


Fig. 9. Influence of α on interaction cost.

preferences. In the second set of experiments, we compared our methods against other commonly used methods in search quality when user's preferences are incomplete or inconsistent.

Existing personalized search methods based on user's preferences mostly includes two parts: modeling and searching. These two parts are integrated and indivisible. In the experiment, each method was compared with the optimal results as baseline when the user's preferences are well described (complete and consistent).

5.2.1. Approaches to compare

There are a number of commonly used methods for personalized search that share our basic ideas. They include the Clustering method using the Weighted Preference (CWP) [30], Bayesian Categorization based on Rating Matrix of user preference (BCRM) [31], the Top-k Iteration based on Utility Function (TIUF) [32], etc.

- Clustering method using the Weighted Preference (CWP): In this method, researchers described a clustering method using

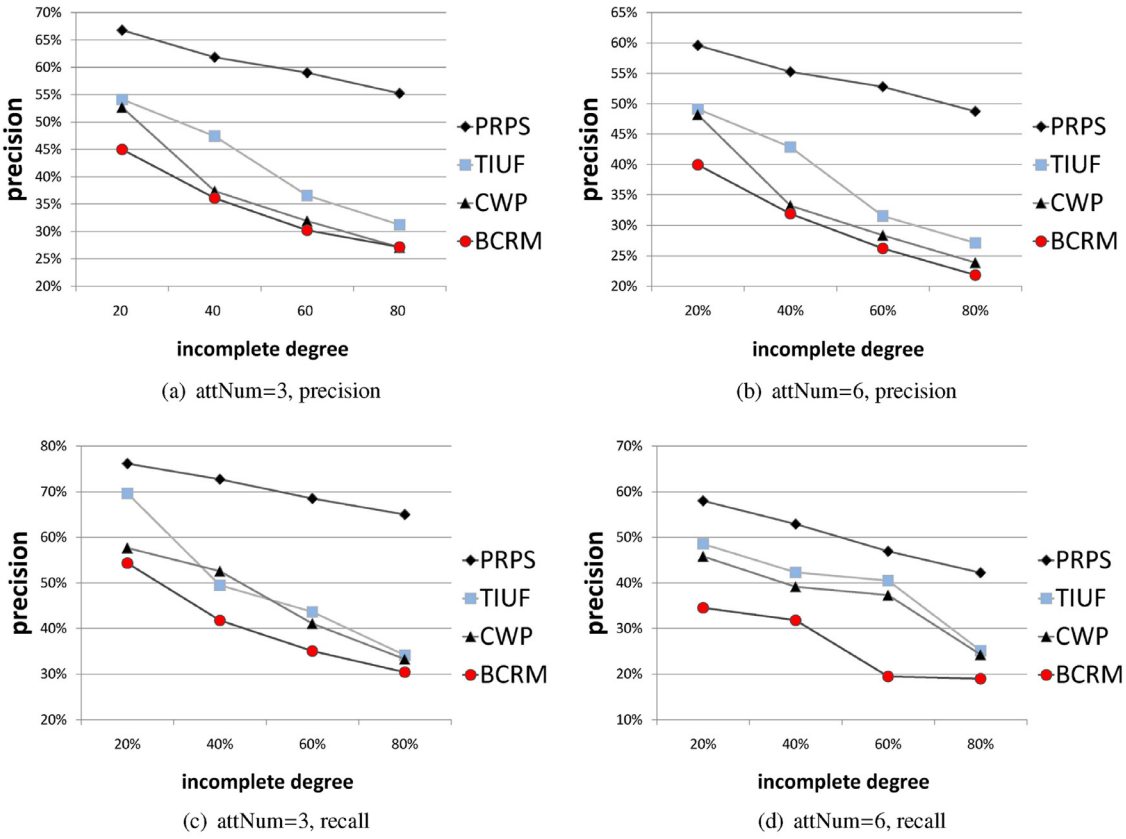


Fig. 10. Comparison: impact of incomplete degree.

the weighted preference based on RFM (Recency, Frequency, Monetary) score. The RFM score of customer is computed so as to reflect the attributes of the customer. It consists of three attributes(R, F, M), each with five bins divided by 20% exact quintile. The system recommends items according to the preferences in the clusters determined by demographic variable and customer's RFM score.

- Bayesian Categorization for recommendation based on Rating Matrix of user preference (BCRM): In this work, users rate the preference for the items. Preference levels are represented on a scale of 0 1.0 in increments of 0.2, a total of six degrees. Only when the value is higher than 0.5, is the user classified as showing interest. By applying the Pearson correlation coefficient, user similarity is defined. The prediction made for new user utilizes the similarity, by applying the Bayesian estimated value to the Pearson coefficient correlation.
- Top-k Iteration based on Utility Function (TIUF): The constraint tuple is used to describe user's preferences on all QoS attributes. Preference weight is used to weigh a user's preferences on different QoS attributes. Preference weight is discretized based on the expected discretized number provided by a simple length-proportional preference weight discretization technique. For each preference weight tuple, a top-k iteration composition process with threshold indicator is used to find the optimal or near-optimal services.

In the experiments, the data set was a randomly generated. As normalization can be used for data preprocessing, in the experiment, the value range of attributes was all set to 0–1. The incompleteness degree was varied to evaluate the above methods under different scenarios of preference missing.

Table 1
Statistical analysis of comparison.

	Metric	attNum	PRPS	TIUF	CWP	BCRM
Means	Precision	3	60.72%	42.35%	37.28%	34.62%
		6	54.11%	37.67%	33.43%	29.99%
	Recall	3	70.61%	49.25%	46.17%	40.42%
		6	50.04%	39.14%	36.66%	26.22%
StD	Precision	3	0.0486	0.1035	0.1111	0.0785
		6	0.0454	0.1012	0.1058	0.0783
	Recall	3	0.0487	0.1501	0.1103	0.1040
		6	0.0688	0.0993	0.0901	0.0813
T test	PRPS	vs	TIUF	0.0047*		
	PRPS	vs	CWP	0.0038*		
	PRPS	vs	BCRM	0.0004**		

* sig < 0.05.
** sig < 0.0005.

5.2.2. Evaluation metrics

The search results are divided into four classes: True Positive (TP): if an object belonging to the optimal results set in the baseline is recommended. True Negative (TN): if an object not belonging to the optimal results set in the baseline is not recommended. False Positive (FP): if an object belonging to the optimal results set in the baseline is not recommended. False Negative (FN): if an object not belonging to the optimal results set in the baseline is recommended.

More than ten kinds of metrics have been applied in recommendation evaluation. In order to verify the effectiveness of our method, precision and recall were used as the evaluation metrics. Precision is defined as $precision = \frac{TP}{TP+FP}$. Recall is defined as $recall = \frac{TP}{TP+FN}$. Preference Recommendation for Personalized Search (PRPS) represents our method.

Table 2
Accuracy of recommended preferences.

Product type	Random recommendations			Selected recommendations		
	Recommended preference	Selected preference	Accuracy	Recommended preference	Selected preference	Accuracy
Meal	5	1.25	25%	5	3.3	65.8%
Mobile phone	6	1.7	27.7%	6	4.3	71.4%
TV program	5	1.2	23.2%	5	3.2	63.4%

5.2.3. Comparison results

In the experiment, the incompleteness degree was set to 20%, 40%, 60%, 80%, respectively. And the attribute number of object was set to 3 or 6. Each experiment was executed 100 times for different sample data.

Fig. 10 shows the search quality of the four methods in 4 subsets of the experiments on the first dataset. The *X* axis denotes the incompleteness degree, and the *Y* axis denotes the values of different metrics. Fig. 10(a) and (b) shows the variation of the precision values under different incompleteness degrees. Fig. 10(c) and (d) shows the variation of the recall values.

From the perspective of precision, the four methods performed differently when the incompleteness degree changes. The mean of PRPS's precision are 60.72%(attNum=3) and 54.11%(attNum=6), which is higher than the other methods. With the growth of incompleteness degree, unlike other methods, PRPS's precision does not decline quickly. When the incomplete degree is 80%, the PRPS's precision is kept to more than 50%, which is better than the other methods. PRPS's recall values show similar trends, as shown in Fig. 10(c) and (d).

Statistical analysis is used to evaluate the performance of the methods. The *Means* and *Standard Deviation(StD)* of every methods are listed in Table 1. The means of PRPS's recall are clearly higher than the other methods. The means of PRPS's recall is 70.61%(attNum=3) and 50.04%(attNum=6). The results of *T-test* prove that the difference is significant ($sig < 0.05$). Table 1 shows that the results of PRPS are stable when the incompleteness degree is changing. In short, our method can generate more accurate recommendation than the other methods.

5.3. User study

As mentioned in Section 4.6, the proposed recommendation scheme builds upon a single principle, which assumes that liked minded users tend to have similar preferences. The user study is conducted to assess the validity of the assumption as well as the effectiveness of our recommendation scheme.

5.3.1. Setup

Our study used three types of commodities widely used in people's daily lives. They include meal, mobile phone and TV program. Each type of objects is described using 4–6 attributes. Each attribute can have 3–5 values. These attributes and the values are detailed in the Appendix A.

There are two phases in the user study. In the first phase, we asked 50 users, who are mostly university students, to build their CP-nets for the three types of objects. We urged them to give as much information as possible, so that we could get nearly complete CP-nets. These 50 users were treated as historical users by our recommendation scheme. In the second phase, we asked another 50 users to evaluate the recommendations generated by our scheme. We asked each user to create a partial CP-net, composed of around 6 conditional preferences. Then we ran our system to recommend each user with a set of new conditional preferences (10 for meal, 12 for mobile phone and 10 for TV program). We asked them to select the preferences that match their thoughts. To evaluate the effectiveness of our recommendation scheme, we generated only a half of the recommendations using our scheme, and picked the other half randomly from the historical users' CP-nets. We randomly intermixed the recommended preferences and presented them to the users.

5.3.2. Results

The results of our user study are summarized in Table 2. It compares the randomly generated recommendations against the recommendations selected by our scheme. The qualities of the two types of recommendations are significantly different. Out of the 5–6 randomly generated preferences, users usually only selected less than 2 of them. The accuracy of recommendation was around 25% In contrast, out of the 5–6 preferences generated by our recommendation scheme, more than 3 of them were selected by our users. The accuracy reached 65%. Although our user study was limited to only three types of objects and a single type of users (university students), the results indicate our that recommendation scheme can be quite effective in identifying users' preferences.

6. Conclusion

In this paper, an approach to personalized search based on incomplete CP-nets was proposed. We introduced an efficient algorithm to enable personalized search based on incomplete CP-nets. Following that, we proposed a preference recommendation scheme for incrementally complementing users' CP-nets, so that the quality of personalized search can be significantly improved. Our approach has two main advantages, compared to the existing ones. First, PRPS can support conditional preferences. Second, PRPS can give stable recommendation when users' preferences are incomplete. The results of our user study and simulation demonstrated the effectiveness of our approach and its potential for real world applications.

This study aims to build a preference recommendation system for personalized search to support web-based products and services selection. We proposed a complete system of personalized search based on users' qualitative preferences. This recommender system can be applied in personalized search if user preferences are consistent. Even when users' preferences are under-specified or conflicting, the system proposed in this paper can automatically predict or optimize the users' preferences based on historical user profiles. Our method is not customized for any specific application domain. Nevertheless, it is easy to incorporate domain specific features, because of the flexibility of CP-nets.

In our future work, we plan to extend PRPS to handle both qualitative and quantitative description of user preference. For example, the interval analysis and the universal gray algebra can be exploit to express users quantitative preference. How to integrate CP-nets and the quantitative models is an interesting problem. Moreover, we will apply our method to different application areas, especially to the domain of health care service recommendation.

Acknowledgment

This work was partially supported by NSFC Key Projects (Nos. 61232007, 61532013), Doctoral Fund of Ministry of Education of the People's Republic of China (No. 20120092110028) and Collaborative Innovation Centers of Novel Software Technology and Industrialization and Wireless Communications Technology. Athman Bouguettaya's research was made possible by NPRP 7-481-1-088 grant from the Qatar National Research Fund (a member of The Qatar Foundation).

Appendix A. Product ontology used in user study

This appendix gives a detailed description of the product types used in our user study. For each type of products, we show its attributes and the values each attribute can take.

Meal (Chinese dishes)	
Attribute:	**Values**
Cuisine:	Canton, Shichuan, Shandong, Huaiyang
Method:	Stewed, Fried, Deep fried, Boiled, Roasted
Meat:	Fish, red meat
Drink:	Liquor, beer
Mobile phone	
Attribute:	**Values**
Brand:	Nokia, Samsung, LG, Motorola
Model:	Clamshell, bar phone
Color:	Silver, blue, black
Format:	TDSCDMA, WCDMA, CDMA2000
Style:	Sports, business, music
TV program	
Attribute:	**Values**
Time:	Morning, noon, evening
Subject:	News, science, sports, original DV
Provider:	Youtube, Kuyou, Tudou
Length:	<10 min, 10–30 min, 30–60 min
Venue:	Online, recorded

References

[1] J. Teevan, S.T. Dumais, E. Horvitz, Personalizing search via automated analysis of interests and activities, in: Proceedings of SIGIR Conference on Research and Development in Information Retrieval, 2005, pp. 449–456.
[2] C. Choi, H. Jeong, A broker-based quality evaluation system for service selection according to the QoS preferences of users, Inf. Sci. 277 (2014) 553–566.
[3] A.V. Dastjerdi, R. Buyya, Compatibility-aware cloud service composition under fuzzy preferences of users, IEEE Trans. Cloud Comput. 2 (1) (2014) 1–13.
[4] G. Liu, Y. Wang, M.A. Orgun, E.-P. Lim, Finding the optimal social trust path for the selection of trustworthy service providers in complex social networks, IEEE Trans. Serv. Comput. 6 (2) (2013) 152–167, doi:10.1109/TSC.2011.58.
[5] Q. Yu, A. Bouguettaya, Efficient service skyline computation for composite service selection, IEEE Trans. Knowl. Data Eng. 25 (4) (2013) 776–789.
[6] N. Matthijs, F. Radlinski, Personalizing web search using long term browsing history, in: Proceedings of the fourth ACM International Conference on Web Search and Data Mining, WSDM'11, ACM, New York, NY, USA, 2011, pp. 25–34, doi:10.1145/1935826.1935840.
[7] H.-N. Kim, M. Rawashdeh, A. Alghamdi, A. El Saddik, Folksonomy-based personalized search and ranking in social media services, Inf. Syst. 37 (1) (2012) 61–76.
[8] C. Boutilier, R.I. Brafman, C. Domshlak, H.H. Hoos, D. Poole, Cp-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements, J. Artif. Intell. Res. 21 (2004) 135–191.
[9] N. Wilson, Extending CP-nets with stronger conditional preference statements, in: Proceedings of AAAI Conference on Artificial Intelligence, 2004, pp. 735–741.
[10] J. Goldsmith, J. Lang, M. Truszczynski, N. Wilson, The computational complexity of dominance and consistency in CP-nets, in: Proceedings of International Joint Conference on Artificial Intelligence, IJCAI, 2005, pp. 144–149.
[11] K. Purrington, E.H. Durfee, Agreeing on social outcomes using individual CP-nets, Multiagent Grid Syst. 5 (4) (2009) 409–425.
[12] H. Wang, X. Zhou, W. Chen, P. Ma, Top-k retrieval using conditional preference networks, in: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM'12, ACM, New York, NY, USA, 2012, pp. 2075–2079, doi:10.1145/2396761.2398576.
[13] L. Chen, P. Pu, Survey of Preference Elicitation Methods, Technical Report, Swiss Federal Institute of Technology in Lausanne (EPFL, 2004.
[14] H. Wang, J. Xu, P. Li, Incomplete preference-driven web service selection, in: Proceedings of IEEE International Conference on Services Computing, SCC (1), 2008, pp. 75–82.
[15] H. Wang, S. Shao, X. Zhou, C. Wan, A. Bouguettaya, Web service selection with incomplete or inconsistent user preferences, in: Proceedings of Joint Conference, ICSOC/ServiceWave, 2009, pp. 83–98.
[16] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, GroupLens: an open architecture for collaborative filtering of netnews, in: Proceedings of Conference on Computer Supported Cooperative Work and Social Computing, CSCW, 1994, pp. 175–186.
[17] C. Boutilier, R.I. Brafman, H.H. Hoos, D. Poole, Reasoning with conditional ceteris paribus preference statements, in: Proceedings of Conference on Uncertainty in Artificial Intelligence, UAI, 1999, pp. 71–80.
[18] H. Wang, J. Zhang, W. Sun, H. Song, G. Guo, X. Zhou, Wcp-nets: a weighted extension to CP-nets for web service selection, in: Service-Oriented Computing, Springer, 2012, pp. 298–312.
[19] Y. Dimopoulos, L. Michael, F. Athienitou, Ceteris paribus preference elicitation with predictive guarantees, in: Proceedings of International Joint Conference on Artificial Intelligence, IJCAI, 2009, pp. 1890–1895.
[20] C. Domshlak, F. Rossi, K.B. Venable, T. Walsh, Reasoning about soft constraints and conditional preferences: complexity results and approximation techniques, in: Proceedings of International Joint Conference on Artificial Intelligence, IJCAI, 2003, pp. 215–220.
[21] Z. Xian-Yuan, Z. Zhao-Hui, A CP-nets-based model and verification for consumer-centric information service composition, in: Proceedings of International Conference on Internet Technology and Applications (iTAP), IEEE, 2011, pp. 1–4.
[22] X. Huang, UsageQoS: estimating the QoS of web services through online user communities, ACM Trans. Web 8 (1) (2013) 1.
[23] G. Adomavicius, A. Tuzhilin, 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) (2005) 734–749.
[24] R.D. Burke, Hybrid recommender systems: survey and experiments, User Model. User-Adapt. Interact. 12 (4) (2002) 331–370.
[25] Q. Yu, QoS-aware service selection via collaborative QoS evaluation, World Wide Web 17 (1) (2014) 33–57.
[26] B.M. Sarwar, G. Karypis, J.A. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of International Conference on World Wide Web, WWW, 2001, pp. 285–295.
[27] M. Meila, D. Heckerman, An experimental comparison of several clustering and initialization methods, in: Proceedings of Conference on Uncertainty in Artificial Intelligence, UAI, 1998, pp. 386–395.
[28] Z. Zhang, H. Lin, K. Liu, D. Wu, G. Zhang, J. Lu, A hybrid fuzzy-based personalized recommender system for telecom products/services, Inf. Sci. 235 (6) (2013) 117–129.
[29] W. Abramowicz, K. Haniewicz, M. Kaczmarek, D. Zyskowski, Architecture for web services filtering and clustering, in: Proceedings of Second International Conference on Internet and Web Applications and Services, ICIW'07, 2007, p. p.18.
[30] Y.S. Cho, C.M. Song, S.P. Jeong, I.B. Oh, K.H. Ryu, Clustering Method Using Weighted Preference Based on RFM Score for Personalized Recommendation System in u-Commerce, Springer, Berlin, Heidelberg, 2014.
[31] K.Y. Jung, User Preference Through Bayesian Categorization for Recommendation, Springer, Berlin, Heidelberg, 2006.
[32] T. Wu, W. Dou, C. Hu, J. Chen, Service mining for trusted service composition in cross-cloud environment, IEEE Syst. J. PP (99) (2014) 1–12, doi:10.1109/JSYST.2014.2361841.