User Grouping for Sharing Services with Capacity Limit

Xiping Liu, Wanchun Dou, and Xin Wang

Abstract—Sharing a service among multiple users could bring benefit to users by reducing their service price and also benefit service providers by allowing them to make more profit. Shared services usually have a capacity limit. To construct a win-win situation between users and providers through service sharing, it is necessary to reasonably organize users with similar service requests into groups under the limits of group sizes. This paper explores methods to enable user grouping for sharing services with the limit on the group size. Based on user request descriptions and evaluation, a connection relation is built to present which two users could be in a group. Semi-groups can be derived from the connection relation and two grouping schemes satisfying the group size limit are further determined through proposed algorithms to meet different service expectations. Finally, a case study and a simulation are given to demonstrate the effectiveness of the proposed methods in grouping users to provide higher service benefits to both users and providers.

Index Terms—Service sharing, grouping, services with capacity limit, asymmetric similarity

1 Introduction

Service Computing is an important enabling technique for various services desired by business activities and scientific applications [1]. Similar to web services, more and more services in the physical world involve service computing [2], [3]. With a smart device connected to Internet, a user could place a service request [4], [5], [6], [7], or select a service from ones published by providers [2], [6], [7], [8], [9]. Example services can range from ones involving web access such as online music downloading to offline ones such as a container shipment.

Some services can be shared by multiple users to bring benefits to both users and providers. Users could get a lower price and obtain more personalized services through group advantages [2], [3], [10], [11], [12], [13], while providers of the shared services could possibly gain higher profits by attracting more users to reduce the cost and develop new customized products [14], [15]. For example, when multiple users share a taxi, each user can pay much lower fee compared to taking the taxi alone. During the peak hours, there are not enough taxies to serve user requests. By offering a discount, a taxi could attract more users to share the service and make more money. In this scenario, sharing may bring in more profit to a taxi company and even the whole taxi industry. Many mobile transportation platforms such as Uber, Lyft, and Didi provide customers the option for carpooling. As another example, multiple tourists could require a company to make a customized trip plan rarely provided for a single user and such a group service could be found at Ctrip, FiendlyPlanetTravel, BobNeffTours, etc. Other sharing examples with similar benefits could be found in the data center with co-leasing services, container shipping services, group purchasing services, product customization services, and so on. Generally speaking, sharing a service among multiple users means this service could be taken by them during the same period. Services to be shared usually have a capacity limit and the total service capacity could be divided into multiple sharing units.

The significance and advantage of sharing services among multiple users have been explored in some recent literature. Wang et al. [2] propose a scheduling algorithm to match customer requirements with multiple service resources for different sharing types, with the goal of maximizing the requirement satisfaction degree and service utilization rate while reducing the service sharing cost. Dou et al. [11] propose a multi-criteria decision method, named AHP, to translate multiple individual preferences into a unified group weight vector to quantitatively evaluate services for the group and select one with the largest group benefit. Salamó et al. [16] present nine different consensus strategies to recommend different product lists for a group of users based on user preferences to candidate products.

Existing academic research and industry technologies on service sharing mainly focus on how to provide a satisfying service for a given group, while ignoring the important issue of assembling users into groups. Although Apps such

- *X. Liu is with the Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu 210023, China. E-mail: liuxp@njupt.edu.cn.*
- *W. Dou is with the State Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing, Jiangsu 210023, China. E-mail: douwc@nju.edu.cn.*
- *X. Wang is with the Wireless Networking and System Lab, Department of Electrical and Computer Engineering, Stony Brook University New York, New York, NY 11794 USA. E-mail: x.wang@stonybrook.edu.*

Manuscript received 12 Aug. 2016; revised 21 Nov. 2017; accepted 14 Feb. 2018. Date of publication 27 Feb. 2018; date of current version 7 Apr. 2021. (Corresponding authors: Xiping Liu and Wanchun Dou.)
Digital Object Identifier no. 10.1109/TSC.2018.2810120

as Uber, Lyft, and Didi can group users with the same routes by matching their sources and destinations, few requirements on service quality are considered in the grouping, which affects the user experience to some extent. To share a service among multiple users, it is necessary to group users based on their request similarity, where a request should include the requirements on QoS features.

Although there are many studies on service clustering and user clustering, these methods are not generally based on user requests and hard to apply in user grouping for service sharing. Service clustering [17], [18], [19] partitions candidate services into multiple classes based on functional or/and non-functional descriptions of services to promote effective and efficient service discovery and selection. A service is generally described through multiple features with each represented by a single value and the similarity between services is not relevant to user preferences. In a service sharing scenario, however, multiple acceptable values could be provided for a single feature and user preferences are very important for similarity computing. User clustering [20], [21], [22], [23] is mainly used to gather users into communities based on user profiles such as consuming history and preferences but not specific service requests. As users in the same community may have different requirements on services and may not want to share the service, the methods proposed for user clustering can hardly be used for service sharing.

Despite the potential of service sharing in providing significant benefits to both users and providers, there lacks a method to effectively group users with similar requests for sharing services with capacity limit. The aim of this work is to develop such a grouping method to decrease the price for users grouped and increase the profit for providers through service sharing. Specifically, a connection relation can be used to guide the user grouping. For a pair of users, the connection relation indicates if the two users can accept each other as a group member. It can be obtained based on user request descriptions and evaluation. In this paper, we propose two algorithms to group users based on the connection relation for different benefits.

The rest of this paper is organized as follows: Section 2 analyzes the related work based on a basic grouping procedure. Section 3 gives some necessary definitions to describe the problem. Section 4 presents a user request evaluation method to build a connection relation. Section 5 puts forwards algorithms to obtain final grouping schemes based on the connection relation. Section 6 provides a way to get group consensus. We illustrate our method through a case study and present our performance evaluations in Section 7. Finally, Section 8 concludes this paper.

2 Basic Procedure for User Grouping and Related Work

Generally, a service broker can run an algorithm to group users based on their requests, which can be sent from computers or smart devices over wired or wireless networks. The broker can select a suitable service according to the service requirement of a group from registered services and send the information back to users so that they can share the selected service online or offline. Depending on the company business need and IT infrastructure, the service broker can run in a local server or in a cloud belonging to the third party. Our service grouping algorithm is designed to be simple to implement.

With the input of requests from different users, the final grouping result should indicate which users form a group and provide a unified requirement description for each group to select a shared service. Based on user requests, the grouping mainly involves three basic steps.

- Step I: Evaluate similarity between user requests;
- Step II: Group users based on the similarity;
- Step III: Achieve a consensus for each group.

In a service sharing scenario, a user request generally consists of two parts, service requirements [2], [5], [7], [24] related to the shared service and grouping context related to group members who share the service. Specifically, a service requirement includes expected values and importance levels on each QoS feature; and grouping context includes member requirements and user information. Due to the complexity in request descriptions, it is hard to employ current similarity computing and clustering methods on user grouping for service sharing. The related work is analyzed as follows.

2.1 Similarity Evaluation

Two users could be in a group only if they have similar service requirements and compatible grouping context. The grouping context of two users is considered to be compatible only if one satisfies the requirements of the other member and vice versa, which could be easily checked based on user requests. However, similarity computing between service requirements is much more complicated.

The similarity between two objects described by a set of features is usually measured through Minkowski distance, Mahalanobis distance, Cosine similarity, Jaccard similarity, etc. Euclidean distance [19], [25], [26] is one of most commonly used metrics, where two objects described by n features are considered as two vectors in the Euclidean space. In [19], QoS similarity among a group of candidate services is computed based on different feature values and the same weights to get service clusters for more efficient service selection. In [25], QoS similarity between candidate services and an initial service is evaluated to get an alternative service. In [26], QoS similarity between values from providers and consumers is calculated to get the reputation of services. Pearson correlation coefficient [22], [23] is another common way to evaluate the similarity, where a user can predict the quality of an unused service based on the feedbacks from users who share similar opinions for other used services.

The similarity between service requirements, however, is hard to be measured through above methods. A service requirement is described through expected values and different importance levels on each QoS feature and both of them should be considered for the similarity evaluation. Unfortunately, most existing research computes the similarity only based on feature values [27], [28].

In an example scenario with two QoS features $(f_1 \text{ and } f_2)$, user A and user B have exactly the same expected values on f_1 but totally different values on f_2. Considering the different importance levels that A and B would place on the two features, the similarity has many possibilities. If both A and B think f_1 is much more important than f_2, they would feel that they have similar requirements. In contrast, if they both take f_2 as the more important one, they would think their requirements are not similar. Moreover, the similarity would be asymmetric if users have opposite opinions on importance level of a feature. Suppose that A think f_1 is more important while B thinks f_2 is more important, then A would feel their requirements are similar while B would not.

Despite various existing measures in evaluating the similarity, they [19], [25], [26], [27], [28] did not consider the difference in setting importance levels of features for two objects, which cannot effectively capture the service requirement similarity as shown in the above cases. Moreover, distance or similarity is mostly computed based on features with a single value [19], [25], [26], [27], [28], while it is highly possible that the value requirement on a QoS feature lies within a range which makes these methods fail.

We propose in Section 4 a similarity evaluation method based on both expected values and importance levels of QoS features. For each pair of users, two similarity values are obtained to describe their feelings about the similarity to each other and this is a challenge for clustering.

2.2. User Grouping

Besides the challenge brought by two similarity values between two users, the capacity limit of a service is another issue we should handle. Generally, there is a restriction on the maximum number of members in a group for service sharing. For taxi sharing, four is usually set as the maximum number. Moreover, the determination of the minimum group size needs to consider the profit. For example, a travel company may customize its trip plan only for a group more than ten people.

Existing research provides lots of valuable clustering methods while few of them solve the two critical issues, the asymmetric similarity and the group size limit. K-means clustering is one of the most commonly used methods [20], [28], [29], [30] to get k clusters by minimizing the sum of the squared distance between objects and their corresponding cluster centroids. Fuzzy c-means clustering (FCM) is similar to K-means but allows an object to belong to more than one cluster [22], [31], [32]. Membership level of an object in a cluster is applied in FCM to calculate weighted distance in the objective function. Spectral clustering [20], [33] views objects as vertices in the weighted graph and reformulates the clustering as a graph partitioning problem, where the weights of edges correspond to the similarity between two objects. Hierarchical clustering [20], [23], [27], [28] organizes objects into a hierarchical structure through divisive methods or agglomerative methods, where the former decomposes objects into smaller clusters from up to down and the latter merges objects into bigger clusters from down to up. Density-based clustering [19], [34], [35] classifies objects into core points and border points with respect to parameters Eps and MinPts and then starts from the core points to find clusters through density-reachability and density-connectivity.

These studies cluster most similar objects with or without a limitation on the number of clusters, paying little attention to the limit of the cluster size necessary to obey for service sharing. Moreover, the two similarity values between each pair of users make these methods fail too. To address this issue, we consider setting proper thresholds to identify high similarity and then transfer the asymmetric similarity into a unified Boolean value. A connection relation could be achieved based on Boolean similarity and compatibility. Based on a group size limit, we propose two grouping algorithms with different benefits to obtain reasonable grouping results from the connection relation in Section 5.

2.3 Group Consensus Achievement

For most clustering studies [20], [27], [28], [36], a final result is a grouping scheme where all objects are clustered into disjoint groups. For user grouping oriented to service sharing, however, a consensus on a unified group service requirement should be further determined for each group, where a group service requirement also consists of both expected values and importance levels of each feature.

The way to achieve a consensus for a group of users is discussed in many studies. The authors in [11] put much effort on the weight or preference determination of each feature for achieving a group consensus while they discuss little about expected values. Reference [16] presents nine different strategies to get a consensus on a recommendation list, while the object to evaluate is a whole product but not detailed feature descriptions. Reference [37] groups users to obtain a consensus, considering both values and importance levels. However, the value of a QoS feature is represented through a trapezoidal fuzzy number which is hardly used for all feature types of QoS. Moreover, the importance level is represented through the preference order, which can only provide relative importance (i.e., feature A is more important than feature B), but is hard to represent how much more important A is than B.

As current research generally cannot provide sufficient support to achieve a consensus on both expected values and importance levels of QoS feature, we introduce a negotiation mechanism in Section 6 to obtain an acceptable consensus on the group service requirement for each group.

3. Grouping Framework Based on User Request Descriptions

In order to form efficient service sharing groups, it is important for users to properly present their service requests. In this section, we introduce some basic rules to describe user requests and then provide a grouping framework.

3.1 Describing User Requests

A request consists of two major parts, a service requirement and grouping context. Different service requirements and grouping context would bring users into multiple different groups.

A service requirement may include multiple features, each containing a value range. For example, the price of a service may range from fifteen to thirty dollars and the requirement of a user could range from fifteen to eighteen; the location where a service is executed may be among several buildings of a block and the requirement of a user could be on certain floor of a building; etc. We use $SF = \{sf_i \mid i = 1..n\}$ to represent the set of service quality features. Moreover, $SFVR = \{Sfvr_i \mid i = 1..n\}$ represents the value ranges of features in SF. The actual value range of

$Sfvr_i$ is considered as a set and could be determined based on data published by service providers.

For the set of users who need to share a certain type of service, it is represented as $U = \{u_p \mid p = 1..t\}$. A user may provide several acceptable values and an importance level for each sf_i and such a service requirement is defined as follows.

Definition 1 (SR). $SR = \{Sr_p = <V_p, K_p, W_p> \mid p = 1..t\}$ *represents service requirements for each user in* U, *where*

- $V_p = <V_{p1}, V_{p2}, \ldots V_{pi} \ldots, V_{pn}>$, V_{pi} *represents the range in which* u_p *expects the value of* sf_i *of the shared service to be,* $V_{pi} \subseteq Sfvr_i$, $i = 1..n$.
- $K_p = \{sf_i \mid sf_i$ *is a key feature and must be satisfied for* u_p, $sf_i \in SF\}$.
- $W_p = <w_{p1}, w_{p2}, \ldots w_{pi} \ldots, w_{pn}>$, w_{pi} *represents the importance level of* sf_i *for* u_p, $w_{pi} \in [0, 1]$, $\sum_{i=1}^{n} w_{pi} = 1, i = 1..n$.

If u_p and u_q are in the same group, they must have consistent expected values on K_p and K_q. For example, if $K_p \cap K_q = \{sf_1\}$, $V_{p1} = \{\text{on}\}$, $V_{q1} = \{\text{off}\}$, u_p and u_q cannot be in a group as no service can satisfy both of them.

As analyzed in Section 2, the similarity between two service requirements is described as two asymmetric values. For a user u_p, there is usually a threshold on the acceptable similarity, i.e., u_p can accept u_q as his/her group member only when the similarity of Sr_q to Sr_p is above the given threshold.

Definition 2 (THR). $THR = \{thr_p \mid thr_p \in [0, 1], p = 1..t\}$ *represents similarity thresholds for each user in* U.

A default threshold value could be set based on the mean value for the users who do not provide thresholds. Besides, the user can increase the threshold if the group consensus does not satisfy him/her or decrease it if he cannot get grouped.

Besides *SR* related to the shared service, a user may have requirements on members who share the service. The set of user features is represented as $UF = \{uf_i \mid i = 1..m\}$ and $UFVR = \{Ufvr_i \mid i = 1..m\}$ represents the value ranges of features in UF.

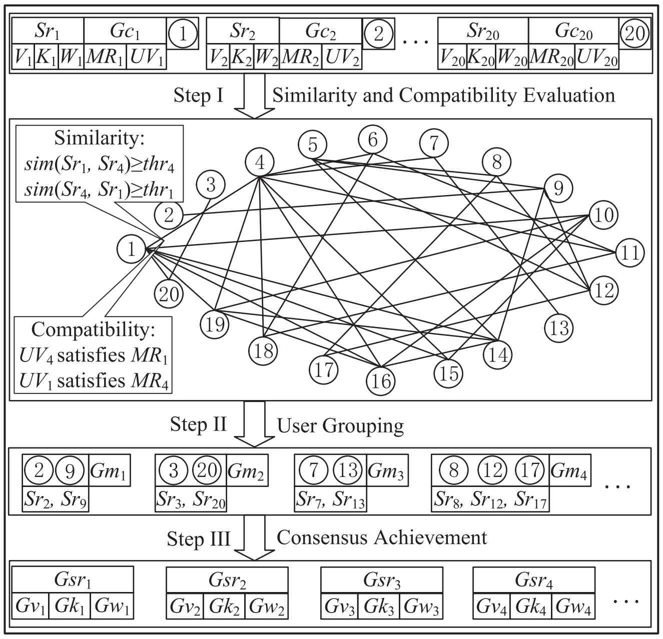
Definition 3 (GC). $GC = \{Gc_p = <MR_p, UV_p> \mid p = 1..t\}$ *represents the grouping context for each user in* U, *where*

- $MR_p = <Mr_{p1}, Mr_{p2}, \ldots Mr_{pi} \ldots, Mr_{pm}>$, Mr_{pi} *means the range in which* u_p *expects the value of* uf_i *of users in the same group to be,* $Mr_{pi} \subseteq Ufvr_i, i = 1..m$.
- $UV_p = <Uv_{p1}, Uv_{p2}, \ldots Uv_{pi} \ldots, Uv_{pm}> \mid Uv_{pi}$ *represents the value of* u_p *on* uf_i, $Uv_{pi} \subseteq Ufvr_i$, $i = 1..m$.

Sometimes, a user may refuse to provide some personal information for reasons such as privacy concerns, which might make this user lose the chance of joining some groups.

3.2 User Grouping Framework

Based on user requests described through *SR*, *THR*, and *GC*, a final result *GROUP* could be obtained based on the limit of the group size. *minN* and *maxN* are used to represent the minimum and maximum numbers of members in a group, which are usually set according to the limitation on the profit and the capacity of services.



Sr_1 Gc_1 1 V_1 K_1 W_1 MR_1 UV_1
Sr_2 Gc_2 2 V_2 K_2 W_2 MR_2 UV_2
Sr_{20} Gc_{20} 20 V_{20} K_{20} W_{20} MR_{20} UV_{20}
Step I Similarity and Compatibility Evaluation
Similarity: $sim(Sr_1, Sr_4) \geq thr_4$ $sim(Sr_4, Sr_1) \geq thr_1$
Compatibility: UV_4 satisfies MR_1 UV_1 satisfies MR_4
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Step II User Grouping
2 9 Gm_1 Sr_2, Sr_9
3 20 Gm_2 Sr_3, Sr_{20}
7 13 Gm_3 Sr_7, Sr_{13}
8 12 17 Gm_4 Sr_8, Sr_{12}, Sr_{17}
Step III Consensus Achievement
Gsr_1 Gv_1 Gk_1 Gw_1
Gsr_2 Gv_2 Gk_2 Gw_2
Gsr_3 Gv_3 Gk_3 Gw_3
Gsr_4 Gv_4 Gk_4 Gw_4

Fig. 1. The framework of user grouping.

Definition 4 (GROUP). $GROUP = <GM, GSR>$ *represents the final result for grouping users in* U, *where*

- $GM = \{Gm_w = \{u_p \mid p \in [1, t]\} \mid Gm_w$ *represents the set of members from the* w*th group, where each member in* Gm_w *satisfies that the similarity to* Sr_p *is higher than* thr_p *and the corresponding grouping context is compatible with* u_p's; $Gm_w \cap Gm_{w'} = \varnothing (w \neq w')$, $minN \leq |Gm_w| \leq maxN$, $w = 1..z\}$.
- $GSR = \{Gsr_w = <Gv_w, Gk_w, Gw_w> \mid Gsr_w$ *represents the group service requirement of the* w*th group,* $w = 1..z\}$

The type of services we consider is the one that a user won't benefit from multiple service times or resources. Thus, we set the constraint that groups in *GM* are mutually exclusive, and each user can join only one group to get a shared service. Note that the union of all groups in *GM* may not equal *U* as some users may not be in any groups. Also, z is not a constant and its value depends on the grouping process. Fig. 1 provides a framework to get a final grouping result based on well-defined user requests. As requests usually come at different time, the grouping can be activated for a given time window or when reaching a pre-set number of requests. Generally, grouping can generate more benefits when there atre enough normal requests besides t sharing requests, such as taxi sharing during the peak hours. With a grouping scheme, a subset of services may target for group users while the remaining services could still be provided to single users.

In Step I, A connection relation could be built based on the evaluation on similarity and compatibility between each pair of users. The compatibility of grouping context could be easily checked by matching *UV* to *MR*, while the similarity of service requirements is more complicated. As analyzed in Section 2, both V_p and W_p should be considered for the similarity between u_p and another user. Thus, the similarity between every two users is described through two values, which makes the general clustering methods based on a single value fail. Based on given thresholds from users, the two similarity values between every two users could be transferred to a Boolean value, which indicate whether these two users accept each other as a member. The detailed evaluation method is presented in Section 4.

In Step II, given a connection relation and a group size limit, there may be multiple candidate grouping schemes. We present two grouping algorithms in Section 5 for different grouping benefits.

In Step III, with the determined grouping scheme *GM*, the consensus on *GSR* need to be achieved and the detailed method is discussed in Section 6.

4 Building A Connection Relation Based on Request Evaluation

According to given *SR*, *THR* and *GC* of *t* users, the service requirement similarity and the grouping context compatibility could be obtained for each user pair and a connection relation on *U* could be built. Then grouping operations could be further performed.

The similarity of service requirements between two users could be computed through two steps, the consistency evaluation on key features and similarity computing between consistent user pairs.

The service requirement consistency between users u_p and u_q is determined based on their requirements on common key features. If V_{pi} equals V_{qi} for each sf_i in $K_p \cap K_q$, service requirements of u_p and u_q are consistent. Users with conflicting service requirements should not be in a group as it is hard to select a service satisfying all their requirements on key features, as analyzed in Section 3.

After evaluating the service requirement consistency between u_p and other users, the service requirement similarity between u_p and ones whose key features are consistent with K_p could be further computed. If there is another user u_q whose V_q and W_q are exactly the same as V_p and W_p, u_p and u_q would be perfect group partners. However, such a case is infrequent. More common cases are users have similar but not the same V_p and W_p. In these cases, it is necessary to compute the similarity with the considerations of both expected values and weights of each feature.

For a feature sf_i, the requirement similarity between u_p and u_q could be computed by Equation (1). If $Sfvr_i$ is a set represented as [lower value, upper value], the size of the intersection and the union of V_{pi} and V_{qi} could be computed according to each set range, respectively.

$$sim_{p-q,i} = sim_{q-p,i} = |V_{pi} \cap V_{qi}| / |V_{pi} \cup V_{qi}|, i = 1..n. \quad (1)$$

For example, suppose that sf_1 means price and its value range is [10, 50], $V_{p1} = [20, 30]$ and $V_{q1} = [10, 25]$ are two possible expected value ranges for sf_1 and the corresponding $sim_{p-q,1} = (25-20)/(30-10) = 5/20 = 0.25$.

Given V_p and V_q, W_p and W_q, the service requirement similarity of u_q to u_p, could be computed as:

$$sim_{q,p} = \sum_{i=1}^{n} w_{pi} sim_{q-p,i}. \quad (2)$$

Despite that $sim_{p-q,i}$ and $sim_{q-p,i}$ are equal, the similarity between u_p and u_q has many possibilities because of different W_p and W_q. For example, given $sim_{p-q,1} = 1$ and $sim_{p-q,2} = 0$, general similarity computing methods without weight consideration do not take this case as the similar one, while it could be similar requirements with certain weight settings. Table 1 presents some possible cases. In case ①, u_p and u_q would think they have similar requirements and are willing to be in a group. Even if the selected shared service may dissatisfy one of them on sf_2, it is acceptable as they both get satisfied on the much more important feature sf_1 and take the advantage of grouping.

TABLE 1
Similarity Between Users Considering the Weight

	$sim_{q,p}$	$Sim_{p,q}$		$sim_{q,p}$	$Sim_{p,q}$
①$W_p = <0.9, 0.1>$ $W_q = <0.8, 0.2>$	0.9	0.8	②$W_p = <0.9, 0.1>$ $W_q = <0.2, 0.8>$	0.9	0.2
③$W_p = <0.1, 0.9>$ $W_q = <0.8, 0.2>$	0.1	0.8	④$W_p = <0.1, 0.9>$ $W_q = <0.2, 0.8>$	0.1	0.2

Generally, users have different weight settings on features and the similarity between two users is described through two different values, as Table 1 shows. With given thresholds, users can find acceptable partners based on the asymmetric similarity.

On the other hand, users with similar service requirements cannot be grouped together if they do not have compatible grouping context. Based on the similarity and the compatibility between users, a connection relation denoting which two users could be in a group can be determined.

Definition 5 (CR). *$CR = \{<u_p, u_q> \mid sim_{p,q} \geq thr_q, sim_{q,p} \geq thr_p, com_{p,q} = com_{q,p} = 1, p, q \in [1, t]\}$ represents the connection relation on U.*

Definition 6 (SU). *$SU = \{Su_p = \{u_q \mid <u_p, u_q> \in CR\} \mid p = 1..t\}$ represents the candidate group members for each user.*

Obviously the binary relation *CR* is symmetric but not transitive. As Fig. 1 shows, $Su_2 = \{u_9\}$ and $Su_9 = \{u_2, u_5, u_{12}, u_{14}\}$, u_9 could group with u_2 and u_5 respectively but u_2 and u_5 cannot be in a group.

We provide Algorithm 1 (time complexity is $O(t^2 \max(n^2, m))$ to obtain Su_p for each user and *CR* can be easily obtained based on *SU*. To get a t^*t similarity matrix, the consistency between any two users u_p and u_q is first evaluated by comparing V_{pi} and V_{qi} for each sf_i in both K_p and K_q. Then the similarity is computed only for the consistent user pairs according to Equations (1) and (2). To get a t^*t compatibility matrix, each pair of users is checked by matching grouping context. There are two situations that u_p does not satisfy u_q's requirement on certain uf_i. One is that the values u_p provided does not satisfy Mr_{qi}; the other is that u_p refuses to provide relevant values i.e., $Uv_{pi} = \emptyset$ and u_q has special requirement on uf_i i.e., $Mr_{qi} \neq Ufvr_i$.

5 User Grouping Based on the Conncection Relation

In the graph corresponding to *CR*, users in each group should be connected to one another and each group is actually a complete subgraph of the *CR* graph. Though lots of grouping schemes could be obtained for a given *CR*, it is very hard to get a reasonable one by directly finding groups from the *CR* graph.

Algorithm 1. User Requests Evaluation and *SU* Building

```
In: $t$, $n$, $m$, $SR$, $GC$, $THR$
Out: $SM = \{sim_{p,q} \mid sim_{p,q} \in [0, 1], p, q = 1..t\}$, $Su_p = \{u_q \mid q \in 1..t\}$
//Evaluate key requirement consistency between $u_p$ & $u_q$
for $p \leftarrow 1$ to $t$
  for $q \leftarrow p + 1$ to $t$
    if $K_p \neq \emptyset$ and $K_q \neq \emptyset$
      $rc_{p,q} \leftarrow$ true
      for each $sf_i$ in $K_p \cap K_q$
        if $V_{pi} \neq V_{qi}$ $rc_{p,q} \leftarrow$ false, $rc_{q,p} \leftarrow$ false, break
      if $rc_{p,q} =$ true $rc_{q,p} \leftarrow$ true
    else $rc_{p,q} \leftarrow$ true, $rc_{q,p} \leftarrow$ true
//Compute service requirement similarity between $u_p$ & $u_q$
for $p \leftarrow 1$ to $t$
  $sim_{p,p} \leftarrow 1$
  for $q \leftarrow p + 1$ to $t$
    if $rc_{p,q} =$ false $sim_{p,q} \leftarrow 0$, $sim_{q,p} \leftarrow 0$
    else
      for $i \leftarrow 1$ to $n$
        compute $sim_{p-q,i}$ according to Equation (1)
      compute $sim_{p,q}$ and $sim_{q,p}$ according to Equation (2)
//Check grouping context compatibility between $u_p$ & $u_q$
for $p \leftarrow 1$ to $t$
  for $q \leftarrow p + 1$ to $t$
  $com_{p,q} \leftarrow 1$
  for each $uf_i$ in $MF$
    if $((Mr_{qi} \neq Ufvr_i)$ and $(Uv_{pi} = \emptyset))$or
    $((Mr_{pi} \neq Ufvr_i)$ and $(Uv_{qi} = \emptyset))$
      $com_{p,q} \leftarrow 0$, $com_{q,p} \leftarrow 0$, break
    else
      if not $Uv_{pi} \subseteq Mr_{qi}$ or not $Uv_{qi} \subseteq Mr_{pi}$
        $com_{p,q} \leftarrow 0$, $com_{q,p} \leftarrow 0$, break
  if $com_{p,q} = 1$ $com_{q,p} \leftarrow 1$
//Finding candidate group members for each $u_p$ $(Su_p)$
for $p \leftarrow 1$ to $t$
  $Su_p \leftarrow \emptyset$
for $p \leftarrow 1$ to $t$
  for $q \leftarrow p + 1$ to $t$
    if $sim_{q,p} \geq thr_p$ and $sim_{p,q} \geq thr_q$ and $com_{p,q} = 1$
      $Su_p \leftarrow Su_p \cup \{u_q\}$, $Su_q \leftarrow Su_q \cup \{u_p\}$
```

As any non-empty subset of the vertices of a complete graph could still form a complete graph, the grouping process could be deployed in two steps. Step one is to find all semi-groups containing more than *minN* members from the *CR* graph, i.e., the sets of complete subgraphs each with the maximum number of users (which cannot be a complete graph any longer if one more user joins in). Step two is to determine the final groups satisfying the group size limit from these semi-groups, where each group would be a subset of certain semi-groups. In this step, we provide two strategies and corresponding algorithms to group users with different benefit considerations.

5.1 Finding Semi-Groups

Definition 7 (SemiG). *$SemiG = \{Sg_v = \{u_p \mid p \in [1, t]\} \mid Sg_v \times Sg_v \subseteq CR \cup IR$ (IR is identity relation on U), $|Sg_v| \geq minN$, $(\forall v)(\forall v')Sg_v \not\subset Sg_{v'}$, $v = 1..y\}$ represents the set of semi-groups.*

Definition 8 (InSemig). *$InSemig = \{In_p = \{v \mid v \in [1, y]\} \mid In_p$ represents the semi-groups in which u_p is, $p = 1..t\}$.*

Here, y is not a constant and its value depends on the grouping process. In Fig. 1, if $minN = 2$, 15 semi-groups can be found, such as $\{u_1, u_4, u_{16}, u_{19}\}$, $\{u_1, u_4, u_{15}\}$, $\{u_1, u_{10}, u_{16}, u_{19}\}$. If $minN = 4$, however, only 4 semi-groups exist and 11 users are not in any semi-groups.

Each group must be a certain subset of certain semi-groups. If a user is not in any semi-groups, he/she cannot join any groups. In this case, modifying the request description or decreasing the threshold might bring the user more opportunity.

A semi-group is in fact a maximal clique [38] in the *CR* graph. The semi-group searching problem based on *CR* is very similar to maximal clique generation problem in an undirected graph, which is a classic NP-hard problem and has some valuable solving methods [38], [39], [40]. As the algorithm in reference [40] can get a solution with the worst-case time complexity of $O(3^{n/3})$, we modify it to get all semi-groups, i.e., maximal cliques with members more than *minN*.

In Algorithm 2, the procedure *findSG*(*SubG_*, *Cand_*) is defined to find semi-groups through a depth-first search, where an empty *SubG* means a new maximal clique *Semig* is found and it could be a semi-group as long as it contains more than *minN* users.

Algorithm 2. Semi-Groups Searching

```
In: $t$, $minN$, $SU$
Out: $SemiG = \{Sg_v = \{u_p \mid p \in [1, t]\} \mid v = 1..y\}$
     $InSemig = \{In_p = \{v \mid v \in [1, y]\} \mid p = 1..t\}$
findSG(SubG, Cand)
  if $SubG = \emptyset$
    if $|Semig| \geq minN$
      $v \leftarrow v + 1$, $Sg_v \leftarrow Semig$, $SemiG \leftarrow SemiG \cup \{Sg_v\}$
      for each $u_q$ in $Sg_v$
        $In_q \leftarrow In_q \cup \{v\}$
  else
    find $u_p$ in SubG that maximizes $|Cand \cap Su_p|$
    while $Cand - Su_p \neq \emptyset$
      choose $u_q$ from $Cand - Su_p$
      $Semig \leftarrow Semig \cup \{u_q\}$
      $SubG\_ \leftarrow SubG \cap Su_q$, $Cand\_ \leftarrow Cand \cap Su_q$
      $findSG(SubG\_, Cand\_)$
      $Cand \leftarrow Cand - \{u_q\}$, $Semig \leftarrow Semig - \{u_q\}$
$v \leftarrow 0$, $Semig \leftarrow \emptyset$, $SemiG \leftarrow \emptyset$, $InSemig \leftarrow \emptyset$
for $p \leftarrow 1$ to $t$ $In_p \leftarrow \emptyset$
$findSG(U, U)$
$y \leftarrow v$
for $p \leftarrow 1$ to $t$
  $InSemig \leftarrow InSemig \cup \{In_p\}$
```

All semi-groups can be found through algorithm 2 and any subsets of a semi-group with *minN* to *maxN* users can form a final group. For example, If $minN = 2$ and $maxN = 4$, 11 final groups could be derived from the semi-group $\{u_1, u_4, u_{16}, u_{19}\}$ shown in Fig. 1, such as $\{u_1, u_4\}$, $\{u_1, u_{16}, u_{19}\}$, $\{u_1, u_4, u_{16}, u_{19}\}$, etc. Generally, for a set of semi-groups, multiple grouping schemes can satisfy the limit of the group size. It is very hard to find the "best" one among so many candidate grouping schemes. First, it is hard to quantify the "best" by providing some criteria suitable for any scenarios and preferences. Second, even there are quantitative criteria for the best one, it is unrealistic to list all possible schemes and evaluate them to choose the one with the best values, especially for the scenario with a great number of users.

With different benefit focuses, we provide two strategies to form the final groups from obtained semi-groups.

5.2 Determining Final Groups to Have More Users Grouped

Service providers and users involved in service sharing may have different opinions on which grouping scheme is better. When publishing services to share, providers often give detailed pricing policies with a discount for a group. Given t user requests, the total profit of a provider generally increases along with the number of users grouped. On the other hand, the price for a single user usually gets lower when a group gets bigger. So service providers prefer putting more users into groups to share their services, and a single user prefers to be in a group with more members to share a service at lower price.

It is very hard to provide a grouping scheme to satisfy all providers and users. Considering the preferences of providers and users, there are two possible grouping options: to put more users into groups, and to establish bigger groups with more members. For example, suppose $SemiG = \{\{Sg_1 = \{u_1, u_2, u_3, u_4\}, Sg_2 = \{u_4, u_5, u_6, u_7\}, Sg_3 = \{u_1, u_8, u_9\}\}$ is in taxi service sharing ($minN = 3$, $maxN = 4$). With the strategy of getting more users grouped, $GM_1 = \{\{u_1, u_8, u_9\}, \{u_2, u_3, u_4\}, \{u_5, u_6, u_7\}\}$ might be the final scheme. With the strategy of getting bigger groups, however, $GM_2 = \{\{u_1, u_2, u_3, u_4\}, \{u_5, u_6, u_7\}\}$ might be the result, where some users grouped could get lower price than GM_1 while u_8 and u_9 are not in any group and get no grouping benefit.

For providers who prefer to make more profit by attracting as many consumers as possible and users who prefer a lower risk of not being in groups, the grouping method mainly focuses on having more users grouped. A user who is first grouped can have more options on group members and have higher possibility of successfully joining a group. The user grouping order is the key issue for a result with more users grouped, which is determined through the following three basic principles.

- Prin. I: the user with the least $|In_p|$ first.
- Prin. II: the user with the fewest conflict semi-groups first.
- Prin. III: the user in a semi-group with the least number of members first.

In Prin. I, users in the least semi-groups are taken as the candidates to be first grouped, as they have less chance of selecting group members and may not find any group to join if they are grouped late. Taking Fig. 1 as an example, $In_3 = 1$ and u_3 can group only with u_{20}. If u_3 is grouped after u_1 and u_1 get u_{20} as a group member, u_3 cannot be in any groups. On the contrary, if u_3 is grouped first and forms group with u_{20}, u_1 in six semi-groups still has many options on members except u_{20}. Sometimes, there is more than one user with the same least $|In_p|$. If they are all in disjoint semi-groups, they can join in groups by picking members in respective semi-groups. If multiple users are in intersecting semi-groups, however, Prin. II needs to be taken for higher possibility of getting more users grouped.

In Prin. II, The user with the fewest conflict semi-groups should be grouped first since it has less impact on the grouping for other users. Given u_p in Sg_v and u_q in $Sg_{v'}$, if Sg_v and $Sg_{v'}$ have common members except for u_p and u_q, $Sg_{v'}$ is a conflict semi-group for u_p, as u_p needs to compete for these common members with u_q. Taking above $SemiG = \{\{Sg_1 = \{u_1, u_2, u_3, u_4\}, Sg_2 = \{u_4, u_5, u_6, u_7\}, Sg_3 = \{u_1, u_8, u_9\}\}$ for example, all users except u_1 and u_4 are in only one semi-group. As $Sg_1 \cap Sg_2 = \{u_4\}$ and $Sg_1 \cap Sg_3 = \{u_1\}$, u_2 and u_3 in Sg_1 have two conflict semi-groups while others have only one. If u_2 and u_3 are grouped first taking u_1 and u_4 as group members, u_8 and u_9 cannot be in any group if $minN \geq 3$.

In Prin. III, users in semi-groups with more members have more options and grouping users with fewer options first can increase the probability of being grouped. In the case of the above $SemiG$, Sg_2 is bigger than Sg_3. If group $\{u_4, u_5, u_6, u_7\}$ forms first, $\{u_2, u_3\}$ or $\{u_8, u_9\}$ cannot be grouped. Finally, with these three principles, GM_1 including every user can be successfully obtained.

With the grouping order determined based on above principles, Algorithm 3 (time complexity is $O(t^4 y^3)$) is proposed to obtain a grouping scheme with more users grouped by determining one final group for a selected user at a time. *findU*() is used to select one user ungrouped and *groupU*() is used to group the selected user. The new emerged group obtained through *groupU*() will bring some changes, which could be set through *renew*(). The grouping process will terminate when *SemiG* is empty. If *Ungrouped* is empty, all users in *U* are partitioned into groups; otherwise, the users still in *Ungrouped* cannot be grouped anymore since they cannot form any semi-groups.

In *groupU*(), a final group would be determined from the corresponding semi-group Sg_v. If u_x in Sg_v also belongs to another conflict semi-group, the average similarity in different semi-groups needs to be computed as Equation (3) to decide if u_x should stay in Sg_v to group with u_p.

$$avesim_{v,x} = \sum_{q \in Sg_v, q \neq x} (sim_{q,x} + sim_{x,q})/(|Sg_v| - 1). \quad (3)$$

If the member number of T, i.e., Sg_v excluding those users who prefer another semi-group, is not between the range of *minN* and *maxN*, the group member should be determined by line 36 to 40. Taking the above *SemiG* for example again, to group u_8 (the first selected user) in Sg_3, $avesim_{1,1}$ and $avesim_{3,1}$ are computed to decide whether u_1 should be grouped in Sg_1 or Sg_3. In this case, as Sg_3 only has *minN* members, u_1 must group with u_8 even though $avesim_{1,1}$ may be higher than $avesim_{3,1}$.

Algorithm 3. Determining *GM* with more Users Grouped

```
In: t, SemiG, InSemig, SM, minN, maxN, U
Out: $GM = \{Gm_w = \{u_p \mid p \in [1, t]\} \mid w = 1..z\}$, Ungrouped
1  findU () //find the user to be grouped next from Cand
2    find users with the least $|In_p|$ as elements of Cand (Prin. I)
3    $T \leftarrow Cand$
4    for each $u_p$ in $T$
5      $min_p.no \leftarrow$ first element in $In_p$, $min_p.num \leftarrow |SemiG|$
6    for each $u_p$ in $T$
7      for each $Sg_v$ in $In_p$
     //find $Conf_{p,v}$ i.e, semi-groups conflict with $Sg_v$ for $u_p$
8        for each $u_q (q \neq p)$ in $T$
9          for each $Sg_{v'}$ in $In_q$
10           if $v \neq v'$ and $Sg_v \cap Sg_{v'} - \{p, q\} \neq \varnothing$
             // $Sg_{v'}$ is a semi-group conflict with $Sg_v$ for $u_p$
11             add $Sg_v$ into $Conf_{q, v'}$
12             if $Sg_{v'}$ is already in $Conf_{p, v}$
             // another user in T and $Sg_{v'}$ is checked before $u_q$
13               $Confsgm_{v,v'} = Confsgm_{v',v} = Confsgm_{v,v'} - \{q\}$
14             else
15               add $Sg_{v'}$ in $Conf_{p, v}$,
16               $Confsgm_{v,v'} = Confsgm_{v',v} = Sg_v \cap Sg_{v'} - \{p, q\}$
17       if $|Conf_{p, v}| < min_p.num$
         //find the semi-group with the fewest conflicting
         semi-groups for $u_p$
18         $min_p.no \leftarrow v$, $min_p.num \leftarrow |Conf_{p, v}|$
19     delete $u_p$ from $T$
20   $groupfor \leftarrow$ null, $m \leftarrow t$
21   for $i \leftarrow 0$ to $|SemiG|$
22     if $groupfor \neq$ null break
23     for each $u_p$ in Cand
24       if $min_p.num = i$ (Prin. II)
25         $v \leftarrow min_p.no$
26         if $|Sg_v| < m$ (Prin. III)
27           $groupfor \leftarrow p$, $m \leftarrow |Sg_v|$
28 groupU() //group for the found user to get a $G_w$
29   $u_p \leftarrow groupfor$, $v \leftarrow min_p.no$, $T \leftarrow Sg_v$
30   if $Conf_{p, v} \neq \emptyset$
     ///delete users who prefer not to group in $Sg_v$
31     for each $Sg_{v'}$ in $Conf_{p, v}$
32       for each $u_x$ in $Confsgm_{v, v'} \cap T$
33         compute $avesim_{v, x}$ and $avesim_{v', x}$ according to
           Equation (3)
34         if $avesim_{v, x} < avesim_{v', x}$
35           delete $u_x$ from $T$
36   if | T |<minN
37     $Gm_w \leftarrow T\cup$ set of users with most minN- | T | $avesim_{v, x}$
       in $Sg_v - T$
38   else
39     if $|T| > maxN$
40       $Gm_w \leftarrow T -$ set of users with the least
         $|T| - maxN$ $avesim_{v, x}$ in $T$
41     else
42       $Gm_w \leftarrow T$
43   $w \leftarrow w + 1$
44 renew() //modify variables after getting a new $Gm_w$
45   $Ungrouped \leftarrow Ungrouped - Gm_w$
46   for each $Sg_v$ in SemiG
47     if $Sg_v \cap Gm_w \neq \varnothing$
48       $Sg_v \leftarrow Sg_v - Gm_w$
49       if $|Sg_v| < minN$ or $Sg_v \subseteq Sg_{v'}$ $(v \neq v')$
50         for each $u_p$ in $Sg_v$
51           delete $v$ from $In_p$
52         delete $Sg_v$ from SemiG
53 $Ungrouped \leftarrow U$, $w \leftarrow 1$
54 while $SemiG \neq \emptyset$
55   findC(), findU(), groupU(), renew()
56 $z \leftarrow w$
```

5.3 Determining Final Groups to Get Bigger Groups

A main purpose for users to share a service with others is to get the service at a lower price. This is impacted by the number of members in a group, i.e., group size. Generally, a single user hopes to join a bigger group, but not all users can really be in such a group and some may not be able to join any groups.

For users who prefer a lower price to the lower risk of not being in any groups, the grouping method mainly focuses on increasing the group size. If a semi-group includes more than *maxN* members, a biggest final group could generated from this semi-group. "Semi-groups with more members first" is a basic principle to get bigger groups. For example, given $SemiG = \{\{Sg_1 = \{u_1, u_2, u_3, u_4\}, Sg_2 = \{u_4, u_5, u_6, u_7\}, Sg_3 = \{u_1, u_8, u_9\}\}$, a final group $\{u_1, u_2, u_3, u_4\}$ could be first obtained but not $\{u_1, u_8, u_9\}$. Based on this principle, Algorithm 4 (time complexity is $O(ty^2)$) is designed to find a scheme with a bigger average group size. Although more users may not be in any groups, this algorithm could bring lower prices for the users who can join groups.

Algorithm 4. Determining GM with Bigger Groups

```
In: t, SemiG, InSemig, SM, minN, maxN, U
Out: $GM = \{Gm_w = \{u_p \mid p \in [1, t]\} \mid w = 1..z\}$, Ungrouped
1  $Ungrouped \leftarrow U$, $w \leftarrow 1$
2  while $SemiG \neq \emptyset$
3  //find semi-groups with most members as candidates
4    $Cand \leftarrow \emptyset$
5    for $i \leftarrow t$ to minN
6      for each $Sg_v$
7        if $|Sg_v| = i$ add $Sg_v$ into Cand
8      if $Cand \neq \emptyset$ Break
9  //find the semi-group to be grouped next from Cand
10   for each $Sg_v$ in Cand
11     compute average semi-group similarity according to
       Equation (4)
12   select the $Sg_v$ with biggest avesgsim
13 //group for the selected semi-group in Cand to get a $Gm_w$
14   if $|Sg_v| > maxN$
15     $Gm_w \leftarrow maxN$ members with maxN $avesim_{v, x}$ in $Sg_v$
16   else
17     $Gm_w \leftarrow Sg_v$
18   $w \leftarrow w + 1$
19   renew()//modify variables after getting a new $G_w$
20 $z \leftarrow w$
```

Similar to Algorithm 3, the grouping is done by determining one final group at a time and the same *renew*() needs to be called when a new final group emerges; also, the terminating condition is the same (*SemiG* is empty). Differently, the grouping is deployed from the perspective of a semi-group but not a user. The semi-group with most members would be selected to form a final group in each round. If there are many semi-groups with the same most members, the one with the biggest average semi-group similarity computed with Equation (4) would be selected to provide a more consistent group service requirement.

$$Avesgsim_v = \sum_{x \in Sg_v} avesim_{v,\,x}/|Sg_v. \tag{4}$$

After the semi-group is selected, it is very easy to form a final group with *maxN* or all members in this semi-group, where the members are sorted by the average similarity computed through Equation (3) to keep the similarity in a group as high as possible.

6 Group Consensus Achievement

When multiple users assemble into a group to share a service, their requirements need to be orchestrated into a unified group service requirement acceptable to each member for selecting a shared service. The group service requirement of the wth group (Gsr_w) includes a requirement vector Gv_w representing the required value range of each feature, a set Gk_w denoting key features, and a weight vector Gw_w indicating importance levels for each feature.

To achieve the consensus on the vector of the value range of each feature, it would be straightforward to take the intersection of requirements from all the members as the group requirement description. However, members in a group might have different and even conflicting requirements on certain feature though they have synthetically high similarity. It is possible that such an intersection does not exist for a certain feature, which means that the expected values on this feature unavoidably conflicts with certain users' requirements. In this case, a negotiation strategy is desired to provide a reasonable feature description acceptable to all the members. First, it is necessary to check whether this feature is a key feature which must be satisfied without negotiation. The requirements on a key feature of certain users should be set as the group requirement. If no user takes this feature as a key feature, taking the requirement of the user with the highest weight on this feature as the group requirement would be comparatively acceptable to members.

To achieve the consensus on the key features, any member's key feature should be taken as a key feature of the group service requirement, which is acceptable to users without key feature settings.

To achieve the consensus on the weight vector, the weight for each feature is set as the average weight of all the members. As each requirement in the group should be equally treated, it is comparatively fair to take the average weight. In fact, two users with highly similar service requirements generally have comparatively high weight on features with similar value ranges and low weight on features without similar value ranges, so the average weights for each feature are close to the weights of each user to some degree.

Based on above strategies, Algorithm 5 (time complexity is $O(wn^2t)$) is applied to achieve a wide-acceptable unified group service requirement from multiple service requirements. The achieved group service requirement should be similar to each individual service requirement of users. Specifically, the service requirement similarity of a group to a member could be computed through Equation (5) and used to evaluate whether the group service requirement is acceptable to this member.

$$gusim_{w,p} = \sum_{i=1}^{n} w_{pi} sim_{Gm_w - u_p,\, i},$$

where

$$sim_{Gm_w - u_p,\, i} = |Gv_{wi} \cap V_{pi}|/|Gv_{wi} \cup V_{pi}|, u_p \hat{I} G_w. \tag{5}$$

As a group service requirement is acceptable to all members, it could be directly used to select a suitable service for the whole group. Besides, if no candidate service satisfies the requirement, there is a possibility that this group could negotiate with some providers on the service customization.

Algorithm 5. Achieving *GSR* Consensus

```
In: GM, SF, SFVR, SR
Out: $GSR = \{Gsr_w = <\ Gv_w, Gk_w, Gw_w\ >\ |\ w = 1..z\}$
1  for each $Gm_w$ in GM
2    for each $sf_i$ in SF
3      $Gv_{wi} \leftarrow Sfvr_i$
4      for each $u_p$ in $Gm_w$
5        $Gv_{wi} \leftarrow Gv_{wi} \cap V_{pi}$
6      if $Gv_{wi} = \emptyset$
7        for each $u_p$ in $Gm_w$
8          if $sf_i \in K_p$ $Gv_{wi} \leftarrow V_{pi}$, break
9        if $Gv_{wi} = \emptyset$
10         $max \leftarrow -1$
11         for each $u_p$ in $Gm_w$
12           if $w_{pi} > max$ $max \leftarrow w_{pi}, x \leftarrow p$
13         $Gv_{wi} \leftarrow V_{xi}$
14     for each $u_p$ in $Gm_w$
15       if $sf_i \in K_p$ add $sf_i$ into $Gk_w$
16     $gw_{wi} \leftarrow 0$
17     for each $u_p$ in $Gm_w$
18       $gw_{wi} \leftarrow gw_{wi} + w_{pi}$
19     $gw_{wi} \leftarrow gw_{wi}/|Gm_w|$
```

7 Case Study and Simulation

In this section, the proposed grouping methods are illustrated in details through a taxi service sharing case. Simulations are further designed based on this case to evaluate the grouping effect. More discussions about related work are also presented.

7.1 A Case Study and Discussion

Our methods to group users with similar service requests could be applied to many service sharing cases. For example, multiple tourists to the same destination with different requirements on hotel rating, vehicle mode, guiding language, etc. could be gathered into groups to get more customized service; multiple merchants could share a container or a warehouse based on the same preservation requirements to reduce the cost; multiple consumers in close locations could bargain with merchants to get a local service with a lower price. As long as their service requests are described and handled based on the methods in this paper, users with similar requests could be grouped and obtain the service online or offline to get benefits from

TABLE 2
Service Request Descriptions of 20 Users

	sf_1	sf_2	sf$_3$	sf_4		uf_1	uf_2		uf_1	uf_2
Sr_1	n/0.6*	4-5/0.1	4-5/0.3	a/0	MR_1	a	a	UV_1	f	y
Sr_2	y/0.8*	3-5/0.2	a/0	a/0	MR_2	m	a	UV_2	m	n
Sr_3	n/0.2	a/0	5/0.2	C/0.6*	MR_3	a	n	UV_3	m	n
Sr_4	n/0.4	a/0	5/0.4	AB/0.2	MR_4	a	a	UV_4	f	y
Sr_5	y/0.5	4-5/0.2	a/0	AB/0.3	MR_5	a	a	UV_5	f	n
Sr_6	n/0.2	5/0.1	5/0.5*	B/0.2	MR_6	a	a	UV_6	f	y
Sr_7	n/0.1	a/0	5/0.4	A/0.5*	MR_7	a	a	UV_7	m	n
Sr_8	y/0.6*	4-5/0.1	3-5/0.2	D/0.1	MR_8	f	a	UV_8	f	y
Sr_9	y/0.3	a/0	4-5/0.4	AB/0.3	MR_9	a	a	UV_9	m	n
Sr_{10}	n/0.7*	3-5/0.3	a/0	a/0	MR_{10}	a	a	UV_{10}	f	y
Sr_{11}	a/0	a/0	5/0.3	B/0.7*	MR_{11}	a	a	UV_{11}	m	y
Sr_{12}	y/0.6*	4-5/0.1	a/0	A/0.3	MR_{12}	a	a	UV_{12}	f	y
Sr_{13}	y/0.2	4-5/0.1	5/0.2	A/0.5*	MR_{13}	a	n	UV_{13}	m	n
Sr_{14}	a/0	4-5/0.4	3-5/0.4	AB/0.2	MR_{14}	a	a	UV_{14}	m	n
Sr_{15}	a/0	3-5/0.3	4-5/0.4	AB/0.3	MR_{15}	f	y	UV_{15}	f	y
Sr_{16}	n/0.6*	3-5/0.2	3-5/0.2	a/0	MR_{16}	a	a	UV_{16}	m	y
Sr_{17}	y/0.6*	a/0	5/0.4	a/0	MR_{17}	f	y	UV_{17}	f	y
Sr_{18}	n/0.1	5/0.1	5/0.4	B/0.4	MR_{18}	a	a	UV_{18}	f	n
Sr_{19}	n/0.5	4-5/0.2	3-5/0.3	a/0	MR_{19}	a	a	UV_{19}	f	n
Sr_{20}	a/0	5/0.2	4-5/0.4	C/0.4	MR_{20}	a	a	UV_{20}	f	n

TABLE 3
Service Requirement Similarity between Each Pair of Users

$sim_{p,q}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	0	.45	.70	.35	.55	.43	0	.55	.90	.32	0	.32	.77	.75	.87	0	.45	.90	.60
2	0	1	.19	.18	.78	.18	.21	.81	.61	0	.23	.74	.43	.61	.61	0	.68	.21	.31	.33
3	.79	.12	1	.80	.08	.72	0	.11	.20	.88	0	.04	0	.29	.38	.79	.40	.52	.68	.64
4	.79	.12	.40	1	.38	.82	.75	.11	.50	.88	.65	.19	.49	.49	.68	.79	.40	.72	.68	.24
5	.22	.93	.04	.28	1	.25	.33	.82	.76	.20	.41	.85	.59	.84	.66	.25	.68	.33	.38	.26
6	.80	.07	.40	.90	.25	1	.50	.12	.35	.80	1	.05	.25	.43	.45	.73	.40	1	.70	.40
7	.79	.12	0	.90	.23	.72	1	.11	.35	.88	0	.34	.74	.39	.53	.79	.40	.52	.68	.24
8	0	.93	.07	.13	.70	.22	.13	1	.57	0	.10	.70	.37	.80	.47	0	.73	.18	.50	.37
9	.34	.92	.10	.40	.88	.37	.45	.77	1	.18	.50	.79	.59	.63	.88	.25	.80	.42	.28	.44
10	.79	0	.39	.58	.28	.38	.31	0	.31	1	.23	0	.23	.61	.61	.92	0	.31	.81	.33
11	.49	.52	0	.70	.48	.82	0	.41	.50	.53	1	.34	0	.39	.53	.49	.70	.87	.43	.24
12	0	.93	.04	.18	.85	.15	.58	.82	.61	0	.06	1	.84	.74	.51	0	.68	.13	.38	.26
13	.25	.93	0	.50	.85	.55	.90	.77	.65	.20	0	1	1	.63	.55	.2	1	.45	.30	.30
14	.60	.53	.17	.53	.75	.42	.43	.60	.72	.55	.45	.55	.52	1	.77	.63	.43	.43	.75	.37
15	.67	.60	.20	.60	.68	.48	.50	.50	.85	.65	.50	.52	.52	.73	1	.63	.50	.48	.58	.47
16	.87	0	.42	.63	.28	.45	.36	0	.42	1	.28	0	.26	.77	.72	1	0	.37	.93	.43
17	0	.92	.35	.50	.73	.57	.52	.73	.65	0	.48	.72	.56	.39	.53	0	1	.52	.18	.34
18	.80	.07	.40	.90	.25	1	.50	.12	.35	.80	1	.05	.25	.43	.45	.73	.40	1	.70	.40
19	.90	.13	.42	.63	.35	.47	.36	.33	.42	.90	.28	.18	.29	.90	.62	.93	.13	.38	1	.47
20	.65	.47	.80	.40	.35	.45	.25	.48	.55	.45	.15	.35	.25	.47	.50	.50	.50	.35	.55	1

grouping, which in many cases could also increase the profit of service providers.

To illustrate our methods in a more concrete way, we consider the case of taxi sharing among 20 users from place A to B for lower fee and less waiting time during the peak hour. Some relevant variables are listed as follows.

$SF = \{sf_1$ the air-conditioner state, sf_2 the driver rating, sf_3 the taxi rating, sf_4 the taxi model$\}$.

$SFVR = \{Sfvr_1 = \{\mathrm{y, n}\},\ Sfvr_2 = [1,\ 5],\ Sfvr_3 = [1,\ 5],\ Sfvr_4 = \{\mathrm{A, B, C, D}\}\}$.

$UF = \{uf_1/\mathrm{gender},\ uf_2/\mathrm{using\ perfume\ or\ not}\}$.

$UFVR = \{Ufvr_1 = \{\mathrm{f, m}\}, Ufvr_2 = \{\mathrm{y, n}\}\}$.

SR and *GC* of the 20 users are presented in details in Table 2, where "expected values/weight" represents V_{pi}/W_{pi} and '*' denotes a key feature. If a user thinks any value on a certain feature is OK, the expected value is set as "a". Users could set different thresholds to accept others with the similarity higher than the threshold as their group members. To focus more on the methods to illustrate, we set a unified threshold $thr = 0.60$ for all users, and set $minN = 2, maxN = 4$.

7.1.1. Connection Relation Building

According to above data and the method described in Section 4, the asymmetric similarity of service requirements between each pair of users are obtained and presented in Table 3, where ".45" denotes "0.45". Also, the compatibility of grouping context can be checked, for example, u_1 is compatible with $u_4 \sim u_{12}$ and $u_{14} \sim u_{20}$. We can get a connection relation among 20 users, as shown in the Fig. 1.

7.1.2. User Grouping

According to the determined *CR*, *SemiG* could be found according to Algorithm 2, as listed in Table 4.

According to above data and Algorithm 3, to have more users grouped, 8 groups including all 20 users could be finally determined, as Table 5 shows.

Given the same semi-groups, another grouping option is to get bigger groups according to Algorithm 4. In this case, $GM = \{Gm_1 = \{u_1, u_{10}, u_{16}, u_{19}\},\ Gm_2 = \{u_4, u_6, u_{11}, u_{18}\},\ Gm_3 = \{u_5, u_8, u_{12}\},\quad Gm_4 = \{u_7, u_{13}\},\quad Gm_5 = \{u_2, u_9\},\ Gm_6 = \{u_3, u_{20}\}\}$ (sorted by the emerging time). In this scheme, the average group size is bigger while 3 users are not in any groups.

7.1.3. Achieving Consensus on Group Service Requirements

After organizing 20 users into groups, a unified group service requirement for each group can be obtained according to Algorithm 5.

Table 5 gives relevant details for the grouping scheme with 8 groups and the details for the other scheme are not given here for the page limit. The service requirement similarity of Gsr_w to Sr_p is listed on the right side, which is high enough and acceptable to group members. If the final grouping result with group service requirements does not satisfy a certain user, the user could appropriately adjust the relevant requirement and present a feedback to get a better acceptable grouping result. This extended operation is out of the scope of this paper.

7.1.4. Comparison and Discussion

As the above case study illustrates, both the proposed grouping algorithms can obtain schemes satisfying the group size limit, where one scheme focuses on putting more

TABLE 4
Details of *SemiG* with 15 Semi-Groups

Sg_1	$\{u_2, u_9\}$	Sg_2	$\{u_3, u_{20}\}$	Sg_3	$\{u_1, u_4, u_{15}\}$
Sg_4	$\{u_1, u_4, u_{16}, u_{19}\}$	Sg_5	$\{u_4, u_6, u_{11}, u_{18}\}$	Sg_6	$\{u_4, u_7\}$
Sg_7	$\{u_5, u_8, u_{12}\}$	Sg_8	$\{u_5, u_9, u_{12}\}$	Sg_9	$\{u_5, u_9, u_{14}\}$
Sg_{10}	$\{u_8, u_{12}, u_{17}\}$	Sg_{11}	$\{u_1, u_{10}, u_{15}\}$	Sg_{12}	$\{u_1, u_{10}, u_{16}, u_{19}\}$
Sg_{13}	$\{u_7, u_{13}\}$	Sg_{14}	$\{u_1, u_{14}, u_{16}, u_{19}\}$	Sg_{15}	$\{u_1, u_{20}\}$

TABLE 5
A Final Grouping Result

	sf_1	sf_2	sf_3	sf_4	sim		sf_1	sf_2	sf_3	sf_4	sim
u_2	y/.8*	3-5/.2	a/0	a/0	1	u_{14}	a/0	4-5/.4	3-5/.4	AB/.2	1
u_9	y/.3	a/0	4-5/.4	AB/.3	1	**G_5: y/.25, 4-5/.3, 3-5/.2, AB/.25**					
G_1: y/.55*, 3-5/.1, 4-5/.2, AB/.15						u_4	n/.4	a/0	5/.4	AB/.2	.9
u_3	n/.2	a/0	5/.2	C/.6*	1	u_6	n/.2	5/.1	5/.5*	AB/.2	.9
u_{20}	a/0	5/.2	4-5/.4	C/.4	.8	u_{11}	a/0	a/0	5/.3	B/.7 *	1
G_2: n/.1, 5/.1, 5/.3, C/.5*						u_{18}	n/.1	5/.1	5/.4	B/.4	1
u_7	n/.1	a/0	5/.4	A/.5*	.9	**G_6: n/.17, 5/.05, 5/.4*, B/.38***					
u_{13}	y/.2	4-5/.1	5/.2	A/.5*	1	u_1	n/.6*	4-5/.1	4-5/.3	a/0	.97
G_3: y/.15, 4-5/.05, 5/.3, A/.5*						u_{10}	n/.7*	3-5/.3	a/0	a/0	1
u_8	y/.6*	4-5/.1	3-5/.2	D/.1	.77	u_{15}	a/0	3-5/.3	4-5/.4	AB/.3	1
u_{12}	y/.6*	4-5/.1	a/0	A/.3	1	**G_7: n/.44*, 3-5/.23, 4-5/.23, AB/.1**					
u_{17}	y/.6*	a/0	5/.4	a/0	1	u_{16}	n/.6*	3-5/.2	3-5/.2	a/0	.93
G_4: y/.6*, 4-5/.07, 5/.2, A/.13						u_{19}	n/.5	4-5/.2	3-5/.3	a/0	1
u_5	y/.5	4-5/.2	a/0	AB/.3	1	**G_8: n/.55, 4-5/.2, 3-5/.25, a/0**					

users into groups and the other focuses on enlarging group sizes. These methods could be applied to not only the taxi service sharing but also to other service sharing cases, where relevant parameters can be set based on practical scenarios. However, the benefits achieved through our grouping methods cannot be obtained through other literature methods.

Grouping needs to first describe user requests through both service requirements and grouping context. Despite the large number of studies on the methods of describing the service requirements [2], [5], [6], [7], [8], [9], [10], [11], few consider the grouping context like ours. Moreover, we present key features to distinguish hard requirements which must be satisfied from soft requirements which could be compromised, while such a situation is hardly expressed by setting a higher weight [2], [8], [11].

It is very important for grouping to evaluate the service requirement similarity and the grouping context compatibility based on user request descriptions. Both the weight and expected multiple values of each feature need to be considered in determining the service requirement similarity, as analyzed in Sections 2 and 4. Unfortunately, the general similarity or distance computing methods for clustering [19], [20], [21], [22], [23], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35] cannot handle this situation, though they have performed well for situations that objects are described through multiple features with each described by a single value and no importance levels of features needs to be considered.

In order to group users for sharing the services with the capacity limit, it needs to consider the group size limit, as well as to take into account the asymmetric similarity and the compatibility between users. This makes it very hard to directly apply current clustering research [20], [27], [28], [36], as most studies on K-means clustering [29], [30], FCM clustering [22], [31], [32], spectral clustering [33], hierarchical clustering [23] and density-based clustering [19], [34], [35] are based on only symmetric similarity and do not take into account the limitation on cluster size.

Moreover, the service requirement consensus on each group is hard to directly achieve through current studies [11], [16], [37], as few of them provide negotiation mechanism on both expected multiple values and quantified importance levels for each feature. The above case study shows that our methods can achieve a consensus highly similar to individual service requirements.

TABLE 6
Grouping Results with Different *k*

k	*gno.*	*ugno.*	*k*	*gno.*	*ugno.*	*k*	*gno.*	*ugno.*
2	7	18	5	7	19	8	8	19
3	6	18	6	7	20	9	7	18
4	6	20	7	7	19	10	6	15

To illustrate our contribution more specifically, we try to apply the existing clustering methods to the above case. As general distance or similarity computing methods are not oriented to objects whose features are described through multiple values, the Equation (1) in our method could be employed. For example, the similarity between u_1 and u_2 is computed based on $sim_{1-2,1} = 0$, $sim_{1-2,2} = 0.67$, $sim_{1-2,3} = 0.4$, $sim_{1-2,4} = 1$. Without weight consideration, the similarity is computed as the mean value of four features which is 0.52. In this way, the similarity between two service requirements is a single value and general clustering methods can be applied by setting a limitation on cluster size. More specifically, only the cluster consisting of the number of users between *minN* and *maxN* is considered as a final group and the cluster consisting of more than *maxN* users should be further subdivided. Taking K-means clustering for example, multiple *GM*s with different *k* can be obtained in the above case, where the number of groups (*gno.*) and the number of users grouped (*ugno.*) are listed in Table 6.

K-means method could get a grouping scheme with more users grouped or bigger groups by adjusting the parameter *k*. However, the grouping result obtained without weight consideration is not reasonable. For example, the grouping result is $GM = \{Gm_1 = \{u_1, u_{15}, u_{19}, u_{20}\}, Gm_2 = \{u_3, u_{10}, u_{11}, u_{16}\}, Gm_3 = \{u_4, u_6, u_7, u_{18}\}, Gm_4 = \{u_5, u_9, u_{14}, u_{17}\}, Gm_5 = \{u_{12}, u_{13}\}, Gm_6 = \{u_2, u_8\}\}$ when *k* is set as 4. In Gm_2, u_3 requires sf_3 to be C while u_{11} requires it to be B. As they both take sf_3 as the key feature, it is impossible to get a group consensus to satisfy both of them. Without weight consideration, their similarity is computed as $(0.5 + 1 + 1 + 0)/4 = 0.625$, which makes u_3 and u_{11} wrongly grouped together. In our method, their similarity is 0 as the key features do not match. Even if sf_3 is not set as the key feature, their similarity is only 0.3, which makes it not possible to put u_3 and u_{11} into the same group. Another example of the impact of the weight on the similarity is $sim_{12,17} = 0.68$ and $sim_{17,12} = 0.72$, while the similarity is only 0.46 without considering the weight.

As above case study illustrates, computing the similarity without weight consideration would make users join a group they should not join, like u_3 and u_{11}. Also, it would make users not in a group they could be, like u_{12} and u_{17}. Besides K-means clustering, unfortunately, few general clustering methods (FCM clustering, spectral clustering, etc.) can handle asymmetric similarity between objects caused by the weight consideration. This makes it fail to apply general clustering methods to user grouping for service sharing.

7.2 Simulation Results

The above case study shows how our methods work for grouping 20 users into 7 or 8 taxis. The grouping simulation

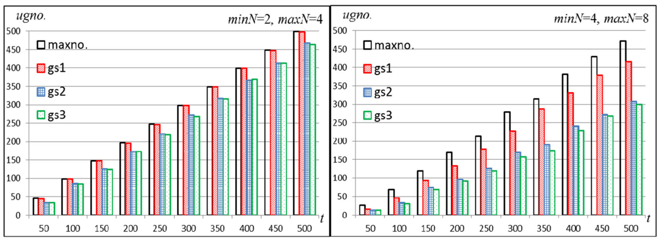


Fig. 2. Total number of users grouped.

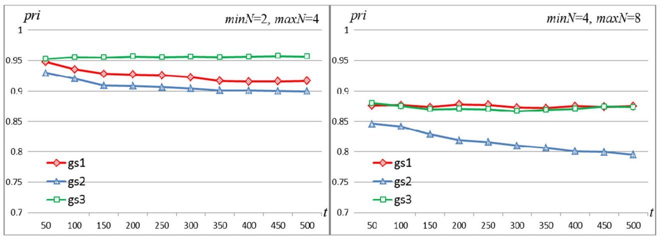


Fig. 4. Average price for each group member.

for more users with uniformly distributed random service requirements and grouping context settings is further given to evaluate the performance of our grouping methods, where t ranges from 50 to 500 and $thr_p = 0.6\,(p = 1..t)$. The corresponding *SFVR* and *UFVR* are the same as those in Section 7.1.

The grouping benefit, i.e., the lower price for users and the higher total profit for providers, is affected by the settings of group size, price function, cost function, etc. Bigger *minN* and *maxN* would increase the average group size and decrease the price, at the cost of higher number of users ungrouped. Moreover, different functions of service price and cost would result in different curves for the total profit of providers and the average price of users. Compared to schemes without grouping, providers could get more profit by setting a discount less than the cost reduction per user derived from the grouping. How to make an appropriate pricing policy based on cost evaluation to get a widely acceptable win-win situation is a very important issue for service sharing, which is out of the scope of this paper and deserves more effort and research.

To present the total profit of providers and the discounted price of users in a more direct and straightforward way, we set a price function and a cost function for all the providers as follows, where *mno.* is the number of members in a group to share a service:

$$price(mno.) = 1 - 0.04 * (mno. - 1), mno. = 1..8, \quad (6)$$

$$cost(mno.) = 0.6 + 0.05 * (mno. - 1), mno. = 1..8. \quad (7)$$

The total profit of providers could be computed by subtracting the total cost from the total payment. For example, if four users get services respectively, they all should pay for 1 unit price and the cost per user is 0.6, so the total profit for providers is $4^*(1 - 0.6) = 1.6$. If they share a certain service, however, they all only need to pay for $1 - 0.04^*(4 - 1) = 0.88$. The cost of the shared service is $0.6 + 0.05^*(4 - 1) = 0.75$ and the cost per user is $0.75/4 = 0.1875$. So the total profit is $4^*0.88 - 0.75 = 2.77$.

Simulation shows all the similarity of a group requirement to a member requirement in both grouping schemes obtained through Algorithms 3 and 4 is over 0.75, which is higher than the given threshold 0.6 and acceptable to users involved in service sharing. More detailed comparison results are presented in Figs. 2, 3, 4, and 5, including the total number of users grouped (*ugno.*), the average group size (*ags*), the average price for each group member (*pri*), and the total profit (*pro*). In Figs. 2, 3, 4, and 5, gs1 denotes the grouping scheme based on Algorithm 3 and gs2 is from Algorithm 4, where all data are obtained based on the average of multiple simulations. As a reference grouping scheme, in gs3, members are randomly selected from ordered semi-groups.

With the random request settings, it is possible that some users have the connection relation with only fewer than *minN* users, which makes them not in any semi-groups and not able to be grouped. The *maxno.* is used to represent the number of users who have the possibility of being grouped, which in fact equals $|InSemig|$. As Fig. 2 shows, gs1 can get most users grouped and *ugno.* of gs1 is always much closer to *maxno.* than those of gs2 and gs3.

As Fig. 3 shows, the average group size of gs2 is up to 40 percent larger than those of gs1 and gs3. According to the price policy, the bigger the average group size, the lower the average price for members. Fig. 4 shows the average price of gs2 is much lower than those of gs1 and gs3.

Between the achieved gs1 with most users grouped and gs2 with the lowest price, gs1 would be the choice of users who are not in any groups in gs2 and gs2 would be the choice of users who have fewer group members in gs1. Before the grouping operation is performed, however, a user does not know if he/she can get a lower price in a certain group or if he/she is the one who cannot join any groups. When the third party builds a platform to organize users into groups for sharing services, both Algorithms 3 and 4 could be applied to provide more options. If a user thinks it is more important to successfully join a group than to get possibly lower price, he/she may choose to participate in the grouping with Algorithm 3. On the other hand, if a user cares more about a lower price than the risk of having no group to join, he/she may choose the grouping with Algorithm 4.

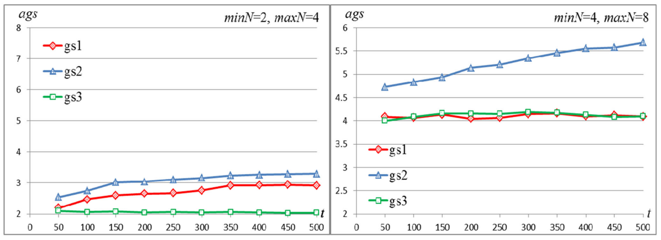


Fig. 3. Average group size.

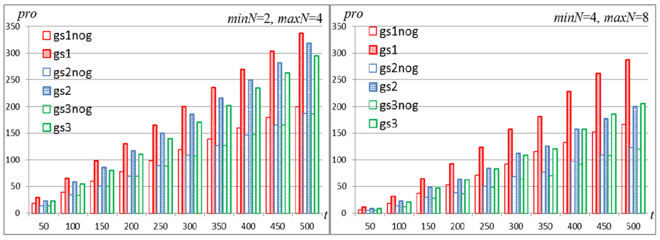


Fig. 5. Total profit of service providers.

A user who chooses to participate in service sharing can generally get the service with a lower price as long as he/she can join a group. On the other hand, can service providers get more profit by providing shared services?

In Fig. 5, 'gs*nog' denotes the total profit of providers if users grouped in 'gs*' are not grouped and get services separately. For example, the total profit of gs1nog for $t = 500$ is computed as users grouped (429) multiplies unit profit $(price(1) - \mathrm{cost}(1) = 0.4)$, that is 171.6; while gs1 gets 96 groups with 4 members, 5 groups with 5 members, 2 groups with 6 members, and 1 group with 8 members, and the total profit is computed as $96^*(0.88^*4 - 0.75) + 5^*(0.84^*5 - 0.8) + 2^*(0.8^*6 - 0.85) + 1^*(0.72^*8 - 0.95) = 295.63$. Obviously, making users in groups to share services can bring much more profit. As Fig. 5 shows, the total profit of grouping schemes increases over 30 percent than the one without grouping, taking advantage of the cost reduction. Moreover, for the same user requests, gs1 gets much more profit than gs2/gs3 as gs1 brings more users into groups.

Accordingly, compared to the grouping result without special strategies, i.e., gs3, the simulation shows that our methods can effectively organize users with highly similar requests into groups to obtain a service with a lower price and bring more profit for providers, which creates a win-win situation.

8 Conclusion

Service sharing could bring benefits to both providers and users through group advantages. To group users for more efficient service sharing, we propose the request-based grouping method to obtain results satisfying the group size limit with different benefits. Unlike the description on a service, the service requirement in a user request is described through multiple acceptable values on each feature and importance levels of features should be considered for similarity computing. This makes general clustering methods fail for the request-based grouping. Our method can handle the asymmetric similarity computing between two service requirements and build a connected relation on users based on the service requirement similarity and the grouping context compatibility. According to two possible strategies that can be selected by users or providers, we propose two algorithms to obtain grouping schemes with more users grouped or bigger group size. We also present a negotiation mechanism to achieve a consensus on a group service requirement. Finally, a taxi sharing case is used to illustrate how our method works for the request-based grouping and why other methods cannot. Moreover, our simulation results show grouping without special strategies will generate much less profit for providers and lead to a higher price for users, while our grouping algorithms could provide better options.

To make our grouping method work well in real service sharing scenarios, there are some limitations to consider. The parameters such as *SF*, *SFVR*, etc. must be carefully set for different scenarios so that users can well express their requirements. Moreover, service providers must set acceptable price functions for shared services based on cost evaluation to ensure more profit. In the future, we will consider a grouping algorithm to maximize the total service revenue by taking into account the idle services when there are not enough user requests in some time periods. Also, adjusting the grouping scheme according to possible feedbacks of users would be another issue to address in our future work.

Acknowledgments

This research is partially supported by the National Science Foundation of US under Grant CNS 1526843, the National Key Research and Development Program of China (No. 2017YFB1400600), the National Science Foundation of China (No. 61672276), the Key Research and Development Project of Jiangsu Province (No. BE2015154 and No. BE2016120), and Jiangsu Oversea Research & Training Program for Univ. Prominent Yong & Middle-aged Teachers and Presidents.

References

[1] Z. Liangjie, "Introduction to IEEE transactions on services computing," *IEEE Trans. Services Comput.*, vol. 6, no. 1, pp. 2–4, Jan.-Mar. 2008.
[2] W. Zhongjie and X. Xiaofei, "A sharing-oriented service selection and scheduling approach for the optimization of resource utilization," *Service Oriented Comput. Appl.*, vol. 6, no. 1, pp. 15–32, 2012.
[3] J. Beihong and H. Jiafeng, "Towards scalable processing for a large-scale ride sharing service, " in *Proc. 9th Int. Conf. Ubiquitous Intell. Comput. Autonomic Trusted Comput.*, Sep. 2012, pp. 940–944.
[4] J. O'Sullivan and D. Edmond, "What's in a service: Towards accurate description of non-functional service properties," *Distrib. Parallel Databases*, vol. 12, no. 2, pp. 117–133, 2002.
[5] C. Ding-Yuan, C. Kuo-Ming, L. Chi-Chun, and T. Chen-Fang, "A user centric service-oriented modeling approach," *World Wide Web-Internet Web Informat. Syst.*, vol. 14, no. 4, pp. 431–459, 2011.
[6] W. Zhi-Jian, L. Zhi-Zhong, Z. Xiao-Feng, and L. Yuan-Sheng, "An approach for composite web service selection based on DGQoS," *Int. J. Adv. Manuf. Technol.*, vol. 56, no. 9-12, pp. 1167–1179, 2011.
[7] D. Mobedpour and C. Ding, "User-centered design of a QoS-based web service selection system," *Serv. Oriented Comput. Appl.*, vol. 7, no. 2, pp. 117–127, 2013.
[8] D. A. Menasce, E. Casalicchio, and V. Dubey, "On optimal service selection in service oriented architectures," *Performance Eval.*, vol. 67, no. 8, pp. 659–675, 2010.
[9] W. Ping, C. Kuo-Ming, and L. Chi-Chun, "On optimal decision for QoS-aware composite service selection," *Expert Syst. Appl.*, vol. 37, no. 1, pp. 440–449, 2010.
[10] M. Sako, "Outsourcing versus shared services," *Commun. ACM*, vol. 53, no. 7, pp. 27–29, 2010.
[11] D. Wanchun, L. Chao, Z. Xunyun, and C. Jinjun, "A collaborative QoS-aware service evaluation method among multi-users for a shared service," *Int. J. Web Services Res.*, vol. 9, no. 1, pp. 30–50, 2012.
[12] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang, "Optimization for dynamic ride-sharing: A review," *Eur. J. Oper. Res.*, vol. 223, no. 2, pp. 295–303, 2012.
[13] G. Manish, "Exploiting the values of shared services," *in Proc. 3rd Int. Conf. Services Emerging Markets*, Dec. 2012, pp. 162–167.
[14] B. Dellaert and P. Dabholkar, "Increasing the attractiveness of mass customization: The role of complementary on-line services and range of options," *Int. J. Electronic Commerce*, vol. 13, no. 3, pp. 43–70, 2009.
[15] T. C. Kuo, "Mass customization and personalization software development: A case study eco-design product service system," *J. Intell. Manuf.*, vol. 24, no. 5, pp. 1019–1031, 2013.
[16] M. Salamó, K. McCarthy, and B. Smyth, "Generating recommendations for consensus negotiation in group personalization services," *Personal Ubiquitous Comput.*, vol. 16, no. 5, pp. 597–610, 2012.
[17] C. B. Pop, V. R. Chifu, I. Salomie, M. Dinsoreanu, T. David, V. Acretoaie, A. Nagy, and C. Oprisa, "Biologically-inspired clustering of semantic web services birds or ants intelligence," *Concurrency Comput. Practice Experience*, vol. 24, no. 6, pp. 619–633, 2012.

[18] D. Skoutas, D. Sacharidis, A. Simitsis, and T. Sellis, “Ranking and clustering web services using multicriteria dominance relationships,” *IEEE Trans. Services Comput.*, vol. 3, no. 3, pp. 163–177, Apr. 2010.

[19] Y. Xia, P. Chen, L. Bao, M. Wang, and J. Yang, “A QoS-aware web service selection algorithm based on clustering,” in *Proc. IEEE 9th Int. Conf. Web Services*, Jul. 2011, pp. 428–435.

[20] P. N. Karamolegkos, C. Z. Patrikakis, N. D. Doulamis, P. T. Vlacheas, and I. G. Nikolakopoulos, “An evaluation study of clustering algorithms in the scope of user communities assessment,” *Comput. Math. with Appl.*, vol. 58, no. 8, pp. 1498–1519, 2009.

[21] Z. Zhang, Q. Li, D. Zeng, and H. Gao, “User community discovery from multi-relational networks,” *Decision Support Syst.*, vol. 54, no. 2, pp. 870–879, 2013.

[22] M. Zhang, X. Liu, R. Zhang, and H. Sun, “A web service recommendation approach based on QoS prediction using fuzzy clustering.” in *Proc. 9th Int. Conf. Services Comput.*, Jun 2012, pp. 138–145.

[23] J. Zhu, Y. Kang, Z. Zheng, and MR. Lyu, “A clustering-based QoS prediction approach for web service recommendation,” in *Proc. IEEE 15th Int. Symp. Object/Component/Service-Oriented Real-Time Distrib. Comput.*, Apr. 2012, pp. 93–98.

[24] Matthias Galster and Eva Bucherer, “A taxonomy for identifying and specifying non-functional requirements in service-oriented development”, in *Proc. IEEE Congress Services*, 2008, pp. 345–352.

[25] M. J. Vishkaei, A. Baraani-Dastjerdi, and K. Jamshidi, “An architecture for web service similarity evaluation based on their functional and QoS aspects,” *Int. J. Web Service Comput.*, vol. 2, no. 2, pp. 1–12, 2011.

[26] S. Zhao, G. Wu, G. Chen, and H. Chen, “Reputation-aware service selection based on QoS similarity,” *J. Netw.*, vol. 6, no. 7, pp. 950–957, 2011.

[27] C. K. Reddy and B. Vinzamuri, “A survey of partitional and hierarchical clustering algorithms,” Charu C. Aggarwal etc. Ed. *Data Clustering: Algorithms and Applications*, Boca Raton, FL, USA: CRC press, 2013, pp.

[28] R. Xu and D. Wunsch, “Survey of clustering algorithms,” *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 645–678, May 2005.

[29] Z. Huang, “Extensions to the K-means algorithm for clustering large data sets with categorical values,” *Data Mining Knowl. Discovery*, vol. 2, pp. 283–304, 1998.

[30] O. M. San, V.-N. Huynh, and Y. Nakamori, “An alternative extension of the k-means algorithm for clustering categorical data,” *Int. J. Appl. Math. Comput. Sci.*, vol. 14, no. 2, pp. 241–247, 2004.

[31] J. C. Bezdek, R. Ehrlich, and W. Full, “FCM: the fuzzy C-means clustering algorithm,” *Comput. Geosciences*, vol. 10, no. 2-3, pp. 191–203, 1983.

[32] R. Hathaway, J. Bezdek, and Y. Hu, “Generalized fuzzy C-means clustering strategies using L_p norm distances,” *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 5, pp. 576–582, Oct. 2000.

[33] U. von Luxburg, “A tutorial on spectral clustering,” *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.

[34] M. Ankerst, M. M. Breunig, H. Kriegel, and J. Sander, “OPTICS: Ordering points to identify the clustering structure,” in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1999, pp. 49–60.

[35] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A densitybased algorithm for discovering clusters in large spatial databases with noise,” in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*, Aug, 1996, pp. 226–231.

[36] M. H. Hasan, J. Jaafar, and M. F. Hassan, “Fuzzy-based clustering of web services’ quality of service: A review,” *J. Commun.* vol. 9, no. 1, pp. 81–90, 2014.

[37] W.-L. Lin, C.-C. Lo, K.-M. Chao, and N. Godwin, “Multi-group QoS consensus for web services,” *J. Comput. Syst. Sci.*, vol. 77, no. 2, pp. 223–243, 2011.

[38] J. Mohsen and H. Abolhassani, “Different aspects of social network analysis.” in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell.*, 2006, pp. 66–72.

[39] K. Makino and T. Uno, “New algorithms for enumerating all maximal cliques,” in *Proc. 9th Scandinavian Workshop Algorithm Theory*, Jul. 2004, vol. 3111, pp. 260–272.

[40] E. Tomita, A. Tanaka, and H. Takahashi, “The worst-case time complexity for generating all maximal cliques and computational experiments,” *Theoretical Comput. Sci.*, vol. 363, pp. 28–42, 2006.

Xiping Liu received the PhD degree in computer applications from Nanjing University. She is an associate professor of College of Computer in Nanjing University of Posts and Telecommunications. Her research interests include service computing and workflow technology.

Wanchun Dou is currently a full professor of Department of Computer Science and Technology in Nanjing University. His research interests include service computing, cloud computing, cooperative computing and workflow technology.

Xin Wang is currently an associate professor of Department of Electrical and Computer Engineering of Stony Brook University. Her research interests include wireless communications, cloud computing, application and service support over wireless networks. She currently serves as the associate editor of IEEE Transctions of Mobile Computing.