A Novel Method for Optimizing Multi-User Service Selection
Hua Jin, Hua Zou, Fangchun Yang, Rongheng Lin, Tianping Shuai

Abstract
Web service selection technology based on quality of service (QoS) has been a research focus for a while. Current researchers are trying to find a service selection solution for single user to get the optimal QoS utility value. However, there is a condition in which multiple users may raise their requests with same functional requirements. These requests will cause a heavy load on service nodes that can provide better QoS. In addition, service composite solutions for users are various because each user’s personal preferences are different. From the perspective of network, network operator wishes to increase the QoS utility of overall network, instead of just satisfying few people’s needs. Because traditional single user service selection is weak on solving above problem, this paper proposes a novel method based on Ant Colony System and KM algorithm (ACK) to meet all above requirements. This method first decomposes the global QoS constrains into local constrains of each service class. Secondly, each local constrain is taken as a global constraint for users and further decomposed in user dimension. After above two steps, a global selection problem is transformed into a local selection problem. Finally, the overall optimal service solution for users will be found out based on decomposed constraints. Experimental results show that our method outperforms traditional solutions in terms of computation time while achieving close-to-optimal results. Results demonstrate that our method is particularly suitable for situation of large-scale users and services selection.

Keywords: Multi-User, Service Selection, Ant Colony System, Bipartite Graph, Converged Network

1. Introduction

Web Service technologies provide a promising solution for the seamless integration of business applications to create new value-added services. As more and more services are emerging on converged network, how to select a composite service to meet user’s requirements both functional and non-functional has become a popular research area. Based on user requirements, service selection mechanism selects service candidates from service classes and composes service candidates into a service flow. Existing works mainly focus on how to provide optimal service selection solution for single user request [1-6]. However, multiple requirements may arrive in a short time, in real life. These requirements are function similar with different QoS preferences. For example, students will plan their trip when school vacation is arriving. John has a trip request and the functionality of this request is composed of tour planning service, room booking service and ticket booking service. Trip requests of other students from different schools are also composed by same services. Using traditional tactics of single user for multiuser service selection will cause problems. Those students won’t be able to be aware of other students’ choice because they don’t know each other. Owing to this, conflicts may happen when a number of users request for same service candidate. In addition, such conflicts will cause service overload because request number exceeds service capability limitation, resulting in a QoS dropping. Actually, it is unnecessary for all users to select service that can provide the best QoS attribute, since different people will have various QoS preferences. What users need are services that best fit to their personal requirements, that is, the higher QoS utility value is, the better. Besides, from the perspective of network operator, quality of experience (QoE) of users may decline due to the drop of service QoS. Therefore, it is an urgent and important issue of making overall service selection solution for multi-user
to get the overall optimal QoS utility value, while still meeting user’s personal preferences and global constraints. IV

The remainder of the paper is organized as follows: Section 2 provides an overview of current related work. Section 3 gives a motivated description about multiuser service selection and introduces global constraint decomposition. Section 4 provides mathematics description on multiuser service selection problem and then decomposes the problem into three steps. Specific algorithms to get final results are detailed in Section 5. Experimental results are presented and analyzed in Section 6. Finally, Section 7 gives conclusions and an outlook on possible continuations of future work.

2. Related work

Various approaches have been proposed to solve web service composition problems. Researchers in this area have focused on different sub-topics and proposed their own assertions and results.

Zeng et al. [1][2] proposed that multiple factors, including QoS attributes, global constraints and preferences set by users, should be considered in service composition. Furthermore, they presented a middleware platform for service composition. They used QoS utility value to represent user’s satisfaction and formulated the composition process as an optimization problem. Maximilien and Singh [3] [4] proposed an agent-based architecture for service selection, considering trust factor in selection process. And they [5] also gave a framework model and ontology for dynamic service selection, based on the information of service quality and trustworthiness. Yu and Lin [6] proposed a service selection model with end-to-end QoS constraints by maximizing an application specific utility function under the end-to-end QoS constraints and to solve the model, algorithms based on Multiple Choice Knapsack Problem (MCKP) and Multi-dimension Multi-choice 0-1 Knapsack Problem (MMKP) were presented. Wu and Wang [7] proposed a QoS-aware Web service global selection algorithm based on multi-objective genetic algorithm. The proposed algorithm can optimize multiple objectives simultaneously on the premise of satisfying multiple constraints and finally obtain a set of constrained Pareto optimum composite service solutions.

Alrifai M et al. [8] first proposed a quality level partition method for global constrains decomposition, using mixed integer program (MIP) to decompose global constrains into local constrains of each service class. Then they used distributed local selection to find best web service that satisfied local constrains, by which way the total time complexity was highly declined. Wang et al. [9] also decomposed global constraints by quality level, using an improved quality level determination method, which was based on fuzzy logic and adaptive adjustment. Additionally, Wang used particle swarm algorithm to search optimal solution, which made improvement on computation time and close-to-optimal results. However, due to global QoS constrains of all users’ solution, summation of each single user’s best QoS utility may not be the overall optimal QoS utility value.

To solve problem of group users service selection, present researchers take qualitative or quantitative method to process multi-users’ preference. By this way, they got a uniform preference to represent all group users and selected services based on this uniform preference. Zhang et al.[10] first to aggregate group users into different cluster, and then computed each cluster’s preferences according to the importance of each cluster. Finally, it used a uniform global QoS preference to get an ideal solution for entire group. Zhou et al.[11] focused on QoS-based qualitative service selection. They used Rank mechanism to reason group users’ QoS preference and computed global solution that satisfied most of users. Because each user in group gets the same solution, users’ personal preference in above researches cannot be well satisfied.

Xiaoqing Liu and Kenneth K et al.[12] raised the problem that conflicts may occur when too many requester select the same best web service. They used Euclidean distance with weights to measure degree of matching of services based on QoS. A 0-1 integral programming model for maximizing the sum of matching degree is created to solve problem of optimal service selection for multiple requesters (GOSSMR). But they only focused on atomic web service selection. In fact, user request may have to be satisfied by composite service under certain global constrains.

Therefore, this paper proposes a service selection method for multiuser condition. Each user’s request is considered as a composite service. This method mainly focuses on those users who have same functional requirements and will gain solutions for each user according to his personal preferences. Besides, method raised by this paper is able to get the overall optimal QoS utility value under global constrains.

This research is supported by the National Basic Research and Development Program (973 program) of China under Grant No. 2009CB320406; the National High Technology Research and Development Program of China under Grant No.2011A01A102; the Foundation for Innovative Research Groups of the National Natural Science Foundation of China under Grant No. 60821001.
3. Problem description

Current service selection mechanism preferred to select service candidates that can provide the best QoS utility value. As Fig. 1 illustrated, each user’s selection is independent with others. With the increments of user requirements for same service class, service candidates that can provide better QoS value will be selected by more and more users. However, resources in a service candidate, such as computation resource or storage resource are not unlimited. A service candidate may not be able to guarantee its QoS provision due to requests overloading. When a single service candidate takes requests more than its threshold, QoS value may decline rapidly, resulting in a bad QoE for users. For example, supposing the overall computation resource of service candidate $s_i$ is $C$ and each user request consumes $c_u$ units of computation resource, the computation threshold of $s_i$ can be calculated as $I_{c_i}^L = \lfloor C / c_u \rfloor$. When the number of user requests processed simultaneously by service candidates $s_i$ exceeds its threshold $I_{c_i}^L$, response delay may increase due to lacking of computation resource.

In the perspective of network operators, they wish to enhance the satisfaction of all users accessed to the network while ensure meeting certain global constraints (such as the overall reliability must be more than 0.8). Because of the existence of some global constraints, summation of each user’s best QoS utility may not be the overall optimal utility value. Besides, another operators’ wish is to avoid QoS declining caused by service overloading. It means that the total number of user requests that a service instance is processing cannot exceed its threshold. Operators have to find a way to make services selection plan from a global perspective for all users, balancing requests load among service candidates. Therefore, the purpose of multiuser service selection is to gain an overall optimal QoS utility value within some global constraints, including QoS constraint and threshold constraint, and satisfy each user’s preference, as depicted in Fig. 2.

Traditional single user service selection has been proved to be an $NP$-hard problem [13]. Multiuser service selection problem can be seen as constituted by a number of single user service selection problems. It’s not difficult to prove that multiuser service selection is also an $NP$-hard problem.

3.1. QoS aggregation

Present service selection implementations are mainly separated by functional requirements and non-functional requirements. This paper supposes all user have same functional requirements but with different QoS preferences. The QoS values provided by one service candidate can be represented as a vector $Q_s = \{ q_1(s), ..., q_r(s) \}$, where $q_i(s)$ means the $i$-th QoS attribute of service candidate $s$. QoS attributes can be divided into two types according to their characteristics [1]: positive attributes and negative attributes. Positive attributes means higher value will bring better service quality, such as reputation and availability. While negative attributes means higher attribute value will cause worse service quality, such as response time or execution time. Different attributes mentioned above can be obtained by different ways, i.e., Response Time can be got by system monitor while Reputation can be got by user feedbacks.

Different composition model or QoS attribute types will lead to different way of QoS utility computation. Here we only consider sequential model to compute QoS utility, other models, such as
parallel or loop, can be transformed into sequential by appropriate process [14]. Because QoS attributes have diverse dimensions and ranges, we normalize all QoS attributes in the same range between [0,1]. The normalization of positive attributes and negative attributes are as follows:

\[
U_{\text{negative}} = \frac{Q_{\text{max}}(j,k) - q_i(s_{i,j})}{Q_{\text{max}}(j,k) - Q_{\text{min}}(j,k)} \quad U_{\text{positive}} = \frac{q_i(s_{i,j}) - Q_{\text{min}}(j,k)}{Q_{\text{max}}(j,k) - Q_{\text{min}}(j,k)}
\]

Here \(Q_{\text{max}}(j,k) = \text{Max}_q(s_{j,k}), \forall s_{j,k} \in S_j\), is the maximum value of \(k\)-th QoS attribute among all service candidates that belongs to service class \(S_j\). So \(Q_{\text{min}}(j,k)\) is the minimum value of \(k\)-th QoS attribute in service class \(S_j\). Let \(w_i\) be the weight of each attribute to represent user’s preference, satisfying \(\sum_{i=1}^{w_i} = 1\). Therefore, the utility of a service candidate \(s_{i,j}\) for user \(u\) can be calculated as follows:

\[
U(s_{i,j}) = \sum_{k=1}^{Q_{\text{max}}(j,k) - q_i(s_{i,j})} w_i + \sum_{k=1}^{Q_{\text{max}}(j,k) - Q_{\text{min}}(j,k)} w_i
\]

The final service composition solution for \(User_u\) can be denoted as \(CS_u = \{s_{1,u}, s_{2,u}, \ldots, s_{m,u}\}\), and the utility of this solution is computed as

\[
U(CS_u) = \sum_{k=1}^{Q_{\text{max}}(k) - Q_{\text{min}}(k)} w_i + \sum_{k=1}^{Q_{\text{max}}(k) - Q_{\text{min}}(k)} w_i
\]

where \(q_i(c_{x,k})\) represents the aggregated value of \(k\)-th QoS attribute in one possible solution, \(Q_{\text{max}}(k)\) and \(Q_{\text{min}}(k)\) represents the maximum and minimum aggregated value of \(k\)-th QoS attribute in all service classes.

### 3.2. Global QoS constraints decomposing

For traditional service selection, if global QoS constraints are decomposed into local constraints, each service class will be able to select their best service candidate independently based on local constraints and user preferences. Not only can this method ensure the satisfaction of global constraints, but also gets a composition service solution in a much faster way.

Generally speaking, a simple decomposition way would be to divide global constraint \(C_j\) into \(n\) local constraints, each local constraint \(c_{i,j}\) equals to \(\frac{C_j}{n}\). Here \(n\) is the number of service classes. However, as different service classes may have different QoS value ranges, such simple division won’t help searching optimal composition solution. By considering all above, global QoS constraint decomposition problem is modeled as an optimization problem. The goal of the problem is to find the a set of local constraints for each service class \(S_j\) that covers as many as possible service candidates, while make attribute summation doesn’t violate any of global constraints.

However, multiuser service selection adds a user dimension compared to single user service selection. Therefore, further constraint decomposition in user dimension is needed after decomposing constraint in service class dimension.

### 3.3. Quality level partition

Quality Level Partition is a method to decompose global QoS constraints used in single user service selection problem [8][9], which can help solving problem much faster. Quality level is a set of discrete values that divide the quality range of each QoS attribute in a service class and each quality level is mapped as a local constraint. After decomposing global constraint into local constraint for each service class, the final composition solution can be got by distributed local selection in each service class. By this method, a composition solution can be selected in a much faster way without violating any of the global constraints.
Quality levels are initialized for each service class $S_j$ by dividing the value ranges of each QoS attribute $q_k$ into a set of $d$ discrete quality values as depicted in Fig. 3. Each sub-range value is denoted as a quality level $q_{jk}$.

$$Q_{\text{min}}(j, k) \leq q_{jk}^1 \leq q_{jk}^2 \leq \cdots \leq q_{jk}^d \leq Q_{\text{max}}(j, k)$$

Here $d$ quality levels will divide the range of $q_k$’s attribute value into $d$ sub-ranges. Then we assign each quality level a utility value $z_{jk}$, which represents the benefit of using $q_{jk}$ as local constraint. The value $z_{jk}$ indicates how many web services would qualify if the $z$-th level was used as local constraint for the $k$-th QoS attribute in service class $S_j$, and estimates the highest obtainable utility value for that class. Because utility calculation method of quality level is closely related to problem description, detailed computation method will be given in section 4.

### 4. Motivation

A user’s request can be decomposed by functionality and taken as a composite service, denoting as $CS = \{S_1, S_2, \ldots, S_n\}$. Each $S$ represents a specific function, and every $S_j = \{s_{j,1}, s_{j,2}, \ldots, s_{j,m}\}$. For users who have same request, their composite service solutions $CS_u, CS_v, \ldots$ are exactly the same. The objective of multiuser service selection is to get each user’s service solution and maximize the summation value of every user’s QoS utility. In addition, each solution has to meet user’s preferences and satisfy global constrains. Such target can be transformed into a constraint-optimization problem below:

$$\begin{align*}
\text{Max}U(CS_m) &= \sum_{j=1}^{n} \sum_{i=1}^{m} Q_{\text{min}}(k) - \sum_{j=1}^{n} \sum_{i=1}^{m} q_k(s_{j,i}) x_{j,i,u} w_{u,i} \\
&+ \sum_{j=1}^{n} \sum_{i=1}^{m} q_k(s_{j,i}) x_{j,i,u} - Q_{\text{max}}(k) w_{u,i}
\end{align*}$$

$s.t. \sum_{j=1}^{n} x_{j,i,u} = 1, 1 \leq j \leq n$

$$\sum_{j=1}^{n} \sum_{i=1}^{m} q_k(s_{j,i}) x_{j,i,u} \leq \text{Limit}$$

$x_{j,i,u}$ is a binary decision variable indicating user’s choice. $x_{j,i,u} = 1$ means that user $u$ selects service instance $s_{j,i}$ while $x_{j,i,u} = 0$ means not. Limit is the maximum request number that a service candidate is able to process well simultaneously. $C_k$ represents the global constrains on $k$-th QoS attribute.

Considering the complexity and characteristics of multiuser service selection, we decompose the problem into three steps and reduce problem complexity gradually in each step, as illustrated in Fig. 4.
In step1, global constraints $C_k$ of $k$-th QoS attribute is decomposed into $n$ local constraints $c'_{k1}, ..., c'_{kn}$, where $n$ is the number of service class. In service class dimension $c'_{kj}$ is a local constraint. As for users who select service instances in same class $S_j$, $c'_{kj}$ is a global constraint that the summation of $k$-th QoS attribute of selected service instances cannot violate. Therefore, in step2, each $c'_{kj}$ can be further decomposed into $t$ local constraints $c''_{kj1}, ..., c''_{kjt}$ in user dimension, where $t$ is the number of users. $c''_{kj}$ is the local constraint for user $i$ when he selects service candidate from service class $S_j$.

Take a simple case as an example, 2 persons (John and Smith) are required to finish the same job and the job is composed by three sub-tasks. The total cost to finish the job cannot exceed 18 dollars. A naive decomposition of cost constraint is to divide these 18 dollars into 6 dollars per sub-task, corresponding to each service class’s local constraint $c''_{kj}$. That means the summation of cost for these 2 people to finish task1 can be no more than 6$. This 6 dollars, which is the constraint of task1, can be further decomposed into 2$ and 4$. That means John has to finish task1 within 2$ and Smith within 4$. Such 2$ or 4$ is corresponding to each user’s constraint $c''_{kj}$.

4.1. Step1: constraints decomposition for service class

In step1, we decompose the global constraints of $k$-th QoS attribute in service class dimension (as the red line depicted in figure4 (a)), taking all users as a unit. $Q_{kj}$ means the quality level for $k$-th QoS attribute of service class $S_j$ and $P_{kj}$ is denoted as its corresponding utility value. The objective function of quality level partition can be expressed as follows:

$$\max \sum_{j=1}^{n} \sum_{k=1}^{r} \sum_{z=1}^{d} \ln(P_{kj}) \cdot v_{kj}$$

$n$ is service class number, $r$ is the number of QoS attributes and $d$ means that QoS attribute of each service class is divided into $d$ sub-ranges by quality levels. Detailed calculation method of $P_{kj}$ will be given in step3. $v_{kj}$ is also a binary decision variable for each quality level $Q_{kj}$. If $v_{kj} = 1$, it means that quality level $Q_{kj}$ was selected to be a local constraint for $k$-th QoS attribute at service class $S_j$ and $v_{kj} = 0$ otherwise. Therefore, we have following allocation constraint for variable $v_{kj}$:
\[ \forall j, \forall k : \sum_{i=1}^{d} v^i_{jk} = 1, 1 \leq j \leq n, 1 \leq k \leq r \]

To make sure the satisfaction of global constraints, one more bound should be taken:

\[ \forall k : \sum_{j=1}^{d} \sum_{i=1}^{d} v^i_{jk} \cdot Q^i_{jk} \leq C_k, 1 \leq k \leq r \]

In step1, \( Q^i_{jk} \) is a quality level of service class \( S_j \), however, it also can be a global constraint from the perspective of users, which will be explained in step2.

4.2. Step 2: user constraints decomposition

The problem is less complex after decomposing global constraint in service class dimension in step1. Its target is to make all \( t \) users get the overall close-to-optimal QoS utility when each user selects his service candidate from the same service class \( S_j \). However, it is still an optimization problem with certain constraints and can be proved to be a NP problem, Constraints are composed of service class \( S_j \)’s QoS constraints and each service candidate’s loading threshold. Problem target and QoS constraint are expressed as follows:

\[ \forall j : \text{Max} \sum_{u=1}^{t} U_u(s_{ju}) \cdot y^u_j \cdot 1 \leq j \leq n \quad (3) \]

\[ \text{s.t. } \forall j : \sum_{u=1}^{t} q_u(s_{ju}) \cdot y^u_j \leq c^i_{jk}, 1 \leq j \leq k, c^i_{jk} = Q^i_{jk} \]

\( y^u_j \) is a binary decision variable indicating whether a user select service instance \( s_{ju} \). \( y^u_j = 1 \) means user \( u \) selects service candidate \( s_{ju} \), and \( y^u_j = 0 \) means not. \( U_u(s_{ju}) \) is the QoS utility value for user \( u \) if he selects service \( s_{ju} \). \( c^i_{jk} \) is a global constraint of service class \( S_j \). When all \( t \) users select their service candidates in service class \( S_j \), the summation value of \( k \)-th QoS attribute of all users’ selected service candidates cannot exceed \( c^i_{jk} \). Here \( c^i_{jk} \) and \( Q^i_{jk} \) have the relationship of \( c^i_{jk} = Q^i_{jk} \), where \( Q^i_{jk} \) is created by global constraint decomposition in step1.

Because above problem is still NP-hard, a further decomposition of constraint \( c^i_{jk} \) can be taken to reduce problem complexity. In service class \( S_j \), let quality level \( q^j_{mu} \) denote the local constraint of user \( u \) for \( k \)-th QoS attribute (as the red line depicted in Fig. 4 (b)), and \( p^j_{mu} \) be the corresponding utility.

The value of \( p^j_{mu} \) can be calculated as

\[ p^j_{mu} = \frac{h(q^j_{mu})}{m_j} \cdot \frac{u(q^j_{mu})}{u_{max}} \quad (4) \]

where \( h(q^j_{mu}) \) represent the number of service candidates in class \( S_j \) would qualify if this level was used as user \( u \)'s local constraint; \( m_j \) is the total number of service candidates in service class \( S_j \); \( u(q^j_{mu}) \) is the highest QoS utility value can be got by quality level of \( q^j_{mu} \); \( u_{max} \) is the highest QoS utility that can be obtained in \( S_j \) by considering all service candidates. Finally, the value of \( p^j_{mu} \) estimates the possible maximum benefit of using quality level \( q^j_{mu} \) as a local constraint, when user \( u \) select service candidates in service class \( S_j \).

Because the \( u(q^j_{mu}) \) value computed by function (1) can be biased by local properties leading to local optima instead of global optima. Therefore, we use the distance between the maximum and
minimum overall quality values $Q_{\max}(k) - Q_{\min}(k)$ instead of local distance used in function (1). This scaling method ensures that the evaluation of service candidates is globally valid:

$$U(s_j) = \sum_{i=1}^{d} \frac{Q_{\max}(j,k)-q_{ij}(s_j)}{Q_{\max}(k)-Q_{\min}(k)} + \sum_{i=1}^{d} \frac{q_{ij}(s_j)-Q_{\min}(j,k)}{Q_{\max}(k)-Q_{\min}(k)}$$  \hspace{1cm} (5)$$

Similar to the constraint decomposition in step 1, quality level partition for different users can also be transformed into an optimization problem, expressed as follows:

$$\forall j : \text{Max} \sum_{a=1}^{t} \sum_{k=1}^{d} \sum_{i=1}^{d} \ln(p_{j,a}^*(s)) * v_{j,a}^*, 1 \leq j \leq n \hspace{1cm} (6)$$

Here $t$ is user numbers and $v_{j,a}^*$ is a binary decision variable related to $p_{j,a}^*$. $v_{j,a}^* = 1$ indicates that quality level $q_{j,a}^*$ is chosen to be a local constraint for user $u$, whereas $v_{j,a}^* = 0$ means $q_{j,a}^*$ wasn’t selected. In order to make sure the constraint $Q_j^*$ of service class $S_j$ won’t be violated when $Q_j^*$ is decomposed in user dimension, one more constraint for function (6) needs to be considered:

$$\forall k : \sum_{a=1}^{t} \sum_{i=1}^{d} v_{j,a}^* * q_{j,a}^* \leq Q_j^*, 1 \leq k \leq r$$

4.3. Step 3: best matching of users and service candidates

Each user’s close-to-optimal quality level $q_{j,a}^*$ for same service class can be got after step 2, and service candidates that violate user’s constraint $q_{j,a}^*$ are skipped from selection. As mentioned in section 3, there is a limitation on request number that a service candidate can process simultaneously. QoS quality may decrease when user requests beyond the threshold of a service candidate. The optimization problem in step 3 is as same as the problem described in step 2. Constraint of service loading can be expressed specifically as follows:

$$\forall j, \forall i : \sum_{a=1}^{t} v_{j,a}^* \leq \text{Limit}, 1 \leq j \leq n, 1 \leq i \leq m$$

Where Limit represents the maximum request number a service candidate can process simultaneously. Because each selected service candidate meets each user’s constraint, the summation value of $k$-th QoS attribute of service candidates selected by all users in service class $S_j$ will satisfies $S_j$’s local constraint $Q_j^*$. As depicted in Fig 4(c), problem in step 3 is similar to assignment problem. By means of bipartite graph matching algorithm, the utility value $P_j^*$ of quality level $Q_j^*$ can be calculated as function (7), which is mentioned in step 1.

$$P_j^* = \text{Max} \sum_{a=1}^{t} \sum_{i=1}^{d} U_i(s_j) * y_{j,a}^*$$  \hspace{1cm} (7)$$

4.4. Global optimal QoS utility computation

After processing of above three steps, we can get all service classes’ quality levels and their corresponding utility values. For each service class, there will be $d$ quality levels and utility values. Then ant colony algorithm will be used to get a close-to-optimal quality level solution $QL_{\text{optimal}} = (Q_{1,k}^*, Q_{2,k}^*, \ldots, Q_{d,k}^*)$ that satisfies global constraints. Then we can get the corresponding utility value set $U_{\text{optimal}} = (P_1^*, P_2^*, \ldots, P_d^*)$ according to $QL_{\text{optimal}}$. The summation value $U_{\text{optimal}}$ is the overall optimal QoS utility we are pursuing for multiuser service selection. As we have mentioned
above, each element $Q_{i}$ in $Q_{optimal}$ is a local constraint of service class $S_i$. Furthermore, each $Q_{i}$ is corresponding to an optimal matching solution of users and service instances, and these service instances belong to the same service class $S_i$. In service class dimension, the selection relationship between users and service candidates is computed and recorded in step 3. Therefore, a user’s service selection solution can be got by compositing service instances he has selected in different service classes.

5. Multi-User service selection based on ACK method

5.1. Ant colony for quality level selection

Ant Colony is a meta-heuristic proposed by Marco Dorigo in 1992. Ant Colony algorithm is inspired by nature that ants always find the shortest path between nest and food source. Ants will deposit pheromone on the path they have walked when they find food. The released pheromone can be felt by other ants and influence them on probability of path choice. Because a short path will attract more and more ants, the density of pheromone on this path will become bigger and bigger. Meanwhile, pheromone trails on other paths will progressively decrease by evaporation. After generations of iteration, a short path with strong pheromone trail will form, so that nearly all the ants will follow it to food source [15]. Ant Colony Algorithm has been widely employed on service selection optimization problems [16][17].

In this paper, ant colony system is applied in solving function (2) and (6) to get best quality level partition solution and its corresponding utility value. From the description of function (2) and (6) we can find that $P_{jk}$ and $p_{jk}$ are key elements for results computation. With regard to $p_{jk}$ mentioned in step 2, its value can be computed by function (4) and (5). As for variable $P_{jk}$ in step 1, the value of itself is equal to the objective of function (7). Since function (7) is similar to an assignment problem, we can use $KM$ algorithm to compute its result. Details of $KM$ algorithm will be given in the following part of this section.

Take quality level partition in step 2 as an example, after global constraints decomposition, we can get a matrix $C_{step2}$ represents users and their quality levels, where each column stands for one user and all his quality levels.

$$C_{step2} = \begin{bmatrix} q_{jk1}^1 & q_{jk2}^1 & \cdots & q_{jk1}^d \\
q_{jk1}^2 & q_{jk2}^2 & q_{jk1}^3 \\
\vdots & \ddots & \vdots \\
q_{jk1}^d & q_{jk2}^d & \cdots & q_{jk1}^d 
\end{bmatrix}$$

As illustrated in Fig. 5, users’ quality levels can be treated as a graph $G=<C,E>$, where $C$ is set of each user’s quality levels and $E$ is the set of edges connecting quality levels between different users. Utility value of each user’s quality level is taken as the weight on edges. We are trying to find a near-optimal quality level solution with the help of $AC$ algorithm. An instance of quality level solution would be like $q_{optimal}^i = (q_{jk1}^1, q_{jk2}^1, \cdots, q_{jk1}^d)$, as the elements connected by lines in Fig. 5.

Figure 5. Graph of users’ quality levels
Each intelligent ant chooses a quality level from next user’s quality level set with a certain probability. This transition probability can be calculated based on two variables. One is the utility value of next use’s quality level. Another is the amount of pheromone deposited on connecting edges. As for optimal service class quality level solution in step 1, its calculation process is similar to above but with a different matrix. Matrix in step 1 is formed by all service classes, their quality level \( z_{jk} \) and corresponding utility value \( z_{jk} \).

Above function (8) is the transition probability calculation method of ants. \( \tau_{q_j q_{j'}} \) represents the pheromone amount between two quality levels and these two quality levels belong to two adjacent users or service classes. \( \alpha \) is the heuristic factor of pheromone and \( \beta \) is the expected heuristic factor. \( \eta_{q_j q_{j'}} \) stands for the utility value of quality level \( q_{j'} \) if \( q_{j'} \) was chosen to be the next element after \( q_j \) in solution. Quality level sets that are allowed to be visited by ants are denoted as \( allowed_m \).

Because of global constraints, we use ant-cycle system to update global pheromone. Once the ant has visited all the nodes on this path at time \( (t+n) \), it will update all the pheromones on all the edges of this path. The pheromone is updated according to formulas below:

\[
\tau_{j}(t+n) = (1 - \rho)\tau_{j}(t) + \rho\Delta\tau_{j}(t,n) , 0 < \rho < 1
\]

\[
\Delta\tau_{j}(t,n) = \sum_{i=1}^{n} \Delta\tau_{j}(t,n)'
\]

\[
\Delta\tau_{j}'(t,n) = \left\{ \begin{array}{ll}
Q \times \sum_{q_{j'} \in allowed} \eta_{q_{j} q_{j'}} & , \text{edge visited} \\
0, & , \text{or else}
\end{array} \right.
\]

\( V \) is the total number of ants, \( (1 - \rho) \) represents the volatile factor of pheromone and \( Q \) represents the pheromone amount deposited by ant during global pheromone updating process. After generations of iteration, optimal path (or solution) will own more pheromones than others and attract more ants. By this way, the optimal quality level partition that satisfies global constraint can be found.

### 5.2. Users-service candidates optimal matching based on KM

Bipartite graph is a kind of special graph and widely used for solving matching problems such as assignment problem [18]. Assignment problem is very common in real life. In its most general form, it can be described as follows: There are a number of people and a number of tasks. Any person can be assigned to perform any task, incurring some cost (or profit) that may vary depending on the person-task assignment. All tasks are required to be finished by assigning exactly one person to one task. The objective of this assignment is to minimize the total cost or maximize total profit. Note that multiuser service selection problem in step 3 is similar to assignment problem except that there is threshold constraint on each service candidate. This paper uses bipartite graph optimal matching to find the best selection relationship between users and service candidates.

For a bipartite graph \( G=<X,Y,E> \) whose vertices can be divided into two disjoint sets \( X \) and \( Y \), where \( X \cap Y = \emptyset \). Each vertex in \( X \) has a edge connects to one in \( Y \) and the set of edges is denoted as \( E \). Users in problem are mapped as \( X \) vertex set and service candidates to be selected are mapped as \( Y \) vertex set. If a service candidate meets user’s local constraint, there is an edge between this user and service candidate, implying that they have a selection relationship. Therefore, the selection relationship between all users and all service candidates are represented as connected edges and mapped as \( E \) set. Weight on edges means the QoS utility value a user will get if one service candidate is selected by a user.
In traditional bipartite graph optimal matching, an $x$ node can only be assigned to one $y$ node and vice versa. But for multi-user multi-service matching, there are situations that a service candidate may be selected by several users simultaneously. Therefore, adjustments are needed so as to employ optimal matching algorithm for solving the problem. In order to guarantee the satisfaction of loading constraint, a service candidate node $y_j$ is mapped into some virtual nodes, denoted as $y_j^1, y_j^2, \ldots, y_j^{\text{Limit}}$. The number of virtual node that a service candidate need to map is the loading threshold value of this service candidate. As for virtual node $y_j^i \in y, y_j^i \in y$, user $u$ will get the same QoS utility value no matter $y_j^i$ or $y_j^i$ is selected.

Table 1 is a matrix represents the relationship between five users and three service candidates. Rows stand for users and columns stand for service candidates. The threshold value for each service candidate is two.

<table>
<thead>
<tr>
<th></th>
<th>$s_1^1$</th>
<th>$s_1^2$</th>
<th>$s_1^3$</th>
<th>$s_1^4$</th>
<th>$s_1^5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td>0.6</td>
<td>0.6</td>
<td>$\ldots$</td>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>$u_2$</td>
<td>0.7</td>
<td>0.7</td>
<td>$\ldots$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$u_5$</td>
<td>0</td>
<td>0</td>
<td>$\ldots$</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>$v_{u_6}$</td>
<td>0</td>
<td>0</td>
<td>$\ldots$</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Because each service candidate’s threshold value is 2, 3 candidates can be mapped into 6 virtual nodes totally, denoted as $(s_1^1, s_1^2, s_1^3, s_2^1, s_2^2, s_2^3)$ in Table 1. Users are represented as $(u_1, u_2, u_3, u_4, u_5, v_{u_6})$, where $v_{u_6}$ is an additional added virtual user node to make balance of user number and service number. Let $A^u_1$ be the set of service candidates that meet user $u_i$’s constraint. If service candidate $s_1 \in A^u_1$, user $u_i$ will obtain the same QoS utility value no matter which virtual node mapped by $s_1$ is selected. Fig 6 illustrated the node mapping of two users and one service candidates. If there is $s_1 \notin A^u_1$, the utility value will be zero and represented as “0” element in matrix.

By this way, a new bipartite graph $G’=<X, Y’, E>$ is created, and the number of nodes $Y’$ is equal to $|Y|*$Limit. For situation of $|X|\neq|Y|$, it is possible to achieve $|X|=|Y|$ by adding some virtual user nodes. Details of making balance can be found in reference [18]. After creating a new bipartite graph $G’$, KM algorithm (Kuhn-Munkres) is applied to compute the maximum value of function (3) mentioned in section 4.

Let $B = A^u_1 \cup A^u_2 \cup \ldots$ be the set of service candidates who meets constraint of users $u_1, u_2, \ldots$. $\text{Num}_s$ represents the number of service candidates in $B$ set and $\text{Num}_u$ is user numbers. $L$ is the load threshold value of a service candidate. If certain conditions are satisfied, we can prove that it is surely to get an optimal matching solution for multiple user-service assignment.
Theorem 1. For a given bipartite graph \( G = \langle X, Y, E \rangle \) representing the relationship between users and service candidates, if \( \forall B_i = A_i^x \cup A_i^y \cup \ldots \), there is \( \text{Num}_x \geq 2 \) and \( \text{Num}_y \times L \geq \text{Num}_x \), then we can say that the bipartite graph \( G = \langle X, Y, E \rangle \) surely exist a complete optimal matching.

Proof: For any given bipartite graph \( G = \langle X, Y, E \rangle \), we can get a balanced bipartite graph by adding virtual user vertexes or service candidate vertexes. For those additional virtual vertexes, their value in matrix will be set to zero. If graph \( G \) satisfies theorem 1, there will be \( |N_x(s)| \geq |S| \) where \( S \in X \). Let \( M \) represent the subset of \( E \) in \( G \), which satisfies \( M \subseteq E \), \( M \) is a perfect matching solution of bipartite graph \( G = \langle X, Y, E \rangle \) according to Hall Theorem [19]. Furthermore, it can be proved that the optimal matching solution computed by KM is a complete optimal matching of above graph \( G \).

6. Experiments results and analysis

The aim of this experiment is to validate our hypothesis that our ACK method achieves close-to-optimal results with a much lower computation time compared to using Ant Colony method directly in solving multiuser service selection problem. Experiments were conducted on a computer with Intel Core2 2.2GHz processor, 2GB of RAM and Windows 7 operating system. Both algorithms were implements and run on Matlab7.0 R2011a Version.

6.1. Simulation datasets

For above purposes, we establish different datasets, each dataset includes \( t \) users and \( n \) service classes. There are \( m \) service candidates in each service class and the maximum number of user requests for a service candidate to process is \( L \). By varying the number of user \( t \) and service candidate \( m \), we create a collection of datasets. Considering the complexity of total computation process, only delay property is considered as a global QoS attribute constraint.

In the following, we use the label “ACK” to our proposed solution and the label “Ant” to refer to using Ant colony solving problem directly. We first solve each dataset using “Ant” approach to find the optimal solution that satisfying all global constraint. We recorded the computation time \( t_{\text{Ant}} \) and the obtained QoS utility value \( u_{\text{Ant}} \) for each test case. We then provided the same dataset to our ACK method and compared its computation time \( t_{\text{ACK}} \) and the final overall QoS utility value \( u_{\text{ACK}} \) with \( t_{\text{Ant}} \) and \( u_{\text{Ant}} \) for the same test dataset respectively. In order to verify the effectiveness of our approach, user number in dataset is changed from 50 to 500 and service candidate number varies from 50 to 100. Loading threshold of service candidate \( L \) changes as user number and service candidate number increasing. Number of service class is assigned with a fix value of 50. QoS attributes, such as delay, price, availability, and each user’s personal preference are created randomly. Specific dates are shown in Table 2.

<table>
<thead>
<tr>
<th>( t )</th>
<th>50</th>
<th>100</th>
<th>150</th>
<th>200</th>
<th>250</th>
<th>300</th>
<th>350</th>
<th>400</th>
<th>450</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m_x )</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>( L_1 )</td>
<td>3</td>
<td>7</td>
<td>10</td>
<td>12</td>
<td>14</td>
<td>18</td>
<td>20</td>
<td>23</td>
<td>26</td>
<td>29</td>
</tr>
<tr>
<td>( m_y )</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>( L_2 )</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>17</td>
<td>19</td>
</tr>
</tbody>
</table>

Parameters in Ant Colony System are set as follows: the heuristic factor of pheromone is \( \alpha = 1 \); the expected heuristic factor is \( \beta = 4 \); volatile factor of pheromone and pheromone amount released by ant are \( \rho = 0.3 \) and \( Q = 0.5 \) respectively. Ant number \( m \) is about 2/3 of service candidate number. Experiment results are averaged by calculating 20 times.
6.2. Computation time comparison

It can be seen from Fig. 7(a) and Fig. 7(b) that the computation time raises rapidly as user number increases when using \textit{Ant} method. When user number is 500 and the number of service candidates reaches 50, the computation time $t_{\text{Ant}}$ is close to 50 seconds. With user number remains, when service candidate number doubles, computation time $t_{\text{Ant}}$ almost gets to 90 sec. Even if the number of user lower to 50, computation time for 50 and 100 service candidates still consume 6.7s and 10.5s respectively. However, it can be seen that the computation time increase gently when our proposed \textit{ACK} method is employed. When user number doesn’t exceed 200, time to get final result is no more than 1 second. Even for the most complex situation, which is 500 users and 100 service candidates, the computation time $t_{\text{ACK}}$ is only about 13sec. Compared with using traditional \textit{Ant Colony}, our proposed \textit{ACK} method is improved greatly in computation time.

6.3. Overall QoS utility comparison

From the experiments results we find that the final QoS utility value computed by \textit{ACK} method is more than just using \textit{Ant} method. Fig. 8(a) and Fig. 8(b) show the ratio of final QoS utility value calculated by \textit{ACK} method is higher than that of \textit{Ant} method. When the number of users is 50, the final utility value $u_{\text{ACK}}$ gained by \textit{ACK} method is about 11%–15% higher than $u_{\text{Ant}}$ gained by simple \textit{Ant Colony}. With increase of user numbers, gap of results computed by both methods becomes bigger and bigger. When the number of users reaches 500, $u_{\text{ACK}}$ is 21% higher than $u_{\text{Ant}}$. If each service class has 50 service candidates (as showed in Fig. 8(a)), and 23% higher if service candidate number is 100 (as showed in Fig. 8(b)). We also study the performance of both methods with respect to the number of users and the number of users varies from 50 to 500. The curves show that as user number increases, the advantages of \textit{ACK} method become more obvious.
6.4. Computational analysis

The scalability of multiuser service selection is affected by the time complexity of applied algorithm. If being solved by Ant Colony method directly, its maximum complexity is about $O((m^e)^t)$, where $m$ represents the number of service candidates in a service class; $n$ means service class number; $t$ is the number of all users. Since service candidates number $m$, class number $n$ and user number $t$ are far beyond 10, result of $(m^e)^t$ will be huge and computation time will be unbearable for users.

As for ACK method, it first decomposes global constraint into local constraint in step1 and assigns each service class $S_j$ with $d$ quality levels. Then each quality level $Q_{jk}$ will be regarded as a global constraint for all users in step2. When users select service candidates in same service class $S_j$, users have to guarantee that the summation of $k$-th attribute of chosen service candidates will not exceed $Q_{jk}$ value.

Supposing there are $r$ QoS attributes, $n$ service classes totally, and $d$ quality levels in each service class. $W$ different quality levels will be produced after global constraints decomposition in step1, where $W = n \times r \times d$. It means that there will be $W$ constraint decomposition problems need to be computed further, which are as same as the problem in step 2. Because the objective of function (3) can be solved by optimal matching as described in step3, $W$ decomposition problems will bring about $W$ optimal matching computations. From the calculation of $W$, it can be seen that the number of sub-problems in step2 and step3 (that is the $W$) has nothing to do with $m$, where $m$ is the number of service candidates in each service class. In addition, when the number of quality level $d$ satisfies $1 \leq d << \frac{m}{r}$, it is ensured that the scale of the computation complexity won’t increase rapidly after global constraints decomposition.

Therefore, the worst case complexity of ACK method is about $O(2d^n + (3t)^3)$, where $d$ means that QoS attribute value is divided into $d$ sub-ranges. By constraints decomposition, a great number of service candidates that violate local constraints are eliminated in advance, which greatly reduces the amount of candidates for subsequent calculations. In addition, the computation of each quality level’s utility value $P_{jk}$ is independent. All utility value in step1 can be calculated distributed so as to reduce total computation time.

Since Ant Colony Algorithm is a kind of heuristic intelligent algorithm, it cannot guarantee to get the best the result. Meanwhile, efficiency and effect are related closely to parameter settings. As for ACK method, its KM algorithm is a kind of deterministic algorithm. For the given users and service candidates, a best matching can always be found by KM and get its corresponding optimal result. By this way, the ACK method is improved greatly in capability of getting close-to-optimal result. Generally speaking, for solving multiuser service selection problem, the ACK method proposed in this paper outperforms Ant method in both computation time and close-to-optimal result aspects.

7. Conclusion and future work

This paper proposes a novel method which solves the problem of service selection for situation of multiple users and service candidates and maximizes overall QoS utility value. Firstly, we decompose global QoS constrains into local constrains of each service class, treating all users as an entirety. Secondly, each service class’s quality level is considered as a global constraint for users and decomposed in user dimension. Through global constraint decomposition, complexity of the problem is reduced greatly. Finally, an ACK method is used to calculate the overall optimal QoS utility and find suitable solution for each user. By this way, global constraints of QoS attributes and service loading are satisfied. Experimental results show that our method is particularly suitable for selection situation of large-scale users and service candidates and outperforms traditional solutions in terms of computation time while achieving close-to-optimal results.

Right now, a novel quality level determination method based on feedback and adaptive adjustment is under working, hoping to decompose global constrains in a more reasonable way. Besides, efficiency of optimal matching algorithm will be improved in our future work.
8. References


