Efficient and Reliable Service Selection for Heterogeneous Distributes Software Systems

Shangguang Wang[1], Lin Huang[1], Lei Sun[1], Ching-Hsien Hsu[2], Fangchun Yang[1]
[1]State Key Laboratory of Networking and Switching Technology; [2]Department of Computer Science and Information
[1]Engineering Beijing University of Posts and Telecommunications; [2]Chung Hua University
[1]Beijing 100876, China; [2]Hsinchu 707, Taiwan
{sgwang; huanglin; sunlei}bupt.edu.cn; chh@chu.edu.tw; fcyang@bupt.edu.cn

Abstract—The service-oriented paradigm is emerging as a new way to heterogeneous distributes software system that are composed of services locally or remotely accessed by middleware technology. How to select the optimal composited service from a set of functionally equivalent services but different QoS attributes has become a hot research in service community. However existing middleware solutions or approaches are inefficient as they search all solution spaces. More importantly, they neglect the QoS inherently uncertainty due to the dynamic network environment. In this paper, based on a service composition middleware framework, we propose an efficient and reliable service selection approach that attempts to select the best reliable composited service on the basis of filtering low reliable services by computing QoS uncertainty. The approach first employs information theory and probability theory to abandon high QoS uncertainty services and downsize the solution spaces. A reliability fitness function is then designed to select the best reliable service for composited services. We experimented with real-world and synthetic datasets and compared our approach with other approaches. Our results show that our approach is not only fast, but also find more reliable composited services.

Keywords-*service selection; service composition; QoS uncertainty; Entropy; Variance*

1. INTRODUCTION

Services are commonly regarded as black boxes with well-defined interfaces that can be recursively aggregated into new service by *service composition* technology [1]. An important aspect of service composition is the finding and binding of services in order to compose them into a composite application. Service composition has become the kernel technology in the domain of service-oriented architecture (SOA) which is able to meet the business requirements of heterogeneous distributes software systems.

According to the SOA paradigm, composite applications are specified as abstract processes composed of a set of abstract services (called *service class*). Then, at service run time, for each service class, a concrete service (called *service candidate*) is selected and invoked. This case ensures loose coupling and flexibility of the design for many business applications distributed within and across organizational boundaries [2].

As well known, QoS (e.g. response time, reliability and throughput) plays an important role in determining the performance of selected services for service composition middleware [3]. Traditional service discovering and matching approaches (e.g., UDDI, Bluetooth) only focus on searching services with functionalities. But with the violently growth number of services, typically, there are a lot of service which is functionally equivalent providing for users, leading to users do not know which service should be selected. Hence, QoS-based service selection approach is proposed aiming at finding the best combination of services that satisfy a set of end-to-end QoS constraints from user's requests.

Some notable service selection approaches include Hybrid [4], GA [5], Replanning [6], CAR & AR [7], MIP [8], and Heuristic [9]. Although these approaches have been shown to perform well in their respective context, it turns out that they are not suited for composited services because of violent QoS fluctuation of services (e.g. the response time of service changes over time). They are lacking in considering the uncertainty of QoS so that they cannot provide reliable services for users in composition system due to dynamic service environments. Generally, service candidates participating in service selection widely distribute in the network. These services come from different organizations/systems and run on different platforms. Hence, any slight changes in location, network environment, service requirement time, and other aspects will affect the reliability of these service candidates [10]. Therefore, it is worth noting that a service with consistently good QoS is typically more reliable than a service with a large Variance on its QoS. Therefore, consistency should be considered as an important criterion for reliable service selection.

In addition, there is an old or new question we have to face. Are there really massive services with same functional attribute but different QoS? The statistics published by the Web services search engine Seekda! indicated that the number of Web services increased exponentially over the recent years. Before Cloud computing, many researchers want to know whether they can be used as service candidates of each service class. Some researchers may be pessimistic. But now, the pay-per-use business model promoted by the Cloud Computing paradigm may enable service providers to offer massive services (e.g., infrastructure as a service, platform as a service, and software as a service) to public, private or hybrid Cloud platforms [11]. Hence, in a vision for the future, there will be massive services. However, most existing approaches suffer from a concentrated workload with the increasing number of services, which causes poor real-time. The main reason is that they focused too much on the optimization of

selection approaches to reduce time cost within service selection process. They neglected a basic principle: how to reduce the search space of service candidates (called *solution space*) is more important than only focusing on the seeking or optimization of service selection approaches.

Different from most existing approaches, this paper, we propose an efficient and reliable approach do not only consider the QoS uncertainty of services, but also pay close attention to downsize the solution spaces of service selection process. The QoS uncertainty is used to filter low reliable services by information theory and probability theory. The higher the QoS uncertainty of a service is, the lower the reliability of the service is, and then it must be filtered from the service candidates. Why do we use the information theory and probability theory in this paper? Entropy is used to measure the expectations of a random variable and its numerical value can excellently reflect the degree of a service's disorder. Also, the main role of the Variance is a measure of the stability of a sample. Using these two aspects to prune the low reliable services could help us make up the defects in existing service selection approaches. Compared with previous QoS-based service selection approaches, our main contributions can be summarized as follows.

Middleware Framework: Aimed at efficient and reliable service selection in heterogeneous distributes software systems, a service composition framework is presented with three distinct components, i.e., *Discovery Engine*, S*election Engine*, and *Composition Engine*.

High Reliability. We adopt Entropy and Variance to compute the uncertainty of QoS. Then the low reliable services will be pruned, and high reliable service can be selected by our designed reliability fitness function for composited services.

Low Computation Time. Because many low reliable services are filtered, the solution space of service selection downsize sharply. This is lower computation time in service selection process than existing techniques.

Extensive Experiments. We have implemented our approach and experimented with real-world 5825 services and 10000 synthetic services. Our results show than our approach is superior to other approaches. We also report results on the parameter study of our approach.

The remainder of this paper is organized as follows. In Section 2, we introduce the background of service selection, including related definitions and related work. Section 3 introduce proposed service composition framework. Section 4 describes our approach in detail including computing QoS uncertainty, uncertain services filtering, and service selection process. The evaluation in Section 5 demonstrates the benefits of our approach. Finally, Section 6 concludes the paper.

2. BACKGROUND

2.1 Related concepts

In this section, we will explain some related concepts about service selection and service composition. The purpose of a composition service is to achieve a particular function which can satisfy user's requirements and preferences. It is obtained by combining a plurality of service candidates which are selected from each service class (which have a number of service candidates). We can understand the concepts of composite service deeply through the following example. In a composite service $S = \{ s_1 , s_2 , \dots , s_n \}$, any $s_i \in S$ and $s_i = \{ s_{i1}, s_{i2}, ..., s_{il} \}$ refers to a service class and it contain $l\ (l > 1)$ functionally equivalent service candidates with different QoS values. The QoS value on a specific Web service are $\{ q_1, q_2, ..., q_l \}$.

The QoS affects the performance of a Web service, is the non-functional attribute of Web Service. A service's QoS has many attributes, such as response time, reliability, throughput, delay, availability and so on. Generally, it can be divided into two categories: positive QoS attributes and negative QoS attributes. The positive QoS attributes (e.g., *reliability, availability*) means that the larger the attribute value is, the better the quality of Web service is. Conversely, the negative QoS attributes (e.g., *response time, delay*) is as low as possible. In this paper, we consider both positive and negative QoS attributes.

Generally, a service's QoS contains multiple attributes. We could get the corresponding attribute value through quantitative calculation. For example, the service s_{ij} has r attributes and its attribute vector can be expressed as $Qs_{ij} = \{ q_1(s_{ij}) , q_2(s_{ij}) , \dots , q_r(s_{ij}) \}$ where the value of $q_k(s_{ij})\ \ (1 < k < r)$ represents the k-th attribute value in service s_{ij} . Similarly, the composite service's attribute vector can be expressed as $QS = \{ q_1(S) , q_2(S) , \dots , q_r(S) \}$ where the value of $q_k(S)$ is aggregated by the k-th attribute values from all the selected service candidates. Table 1 lists the QoS aggregation functions of the sequential composition model. Other models (e.g., parallel, conditional and loops) can be transformed to the sequential model using techniques described in existing papers [12].

Table 1. QoS aggregation functions

QoS Attributes	QoS aggregated functions
Response time	$q(S) = \sum_{i=1}^{n} q(s_i)$
Throughput	$q(S) = min_{i=1}^{n} q(s_i)$
Reputation	$q(S) = \frac{1}{n} \sum_{i=1}^{n} q(s_i)$
Reliability	$q(S) = \prod_{i=1}^{n} q(s_i)$

In a Web service composition, generally, each service candidate contains multiple QoS attributes which leading to the different of the unit or scope of its QoS attributes. This is not helpful for service selection. Therefore, the QoS utility functions are used to solve the global QoS attribute values of each candidate service. The QoS utility function is usually employed to map the vector of QoS values Qs_{ij} into

a single real value Us_{ij} . Then, we would launch the service selection by sorting and ranking each candidate of global QoS aggregated value Us_{ij} . As we all known, in Web service compositions, users generally expect the lower negative QoS attributions values and the higher positive QoS attributions values. So for the negative QoS attributes, a minimum value should be obtained and for the positive QoS attributes, a maximum value should be obtained. The QoS utility function in this paper is similar to [4, 9]. It scales all attributes value in domain [0, 1] for uniform computing on multi-dimensional service attributes, then it adds the user's requests on each attribute. Here we list the QoS utility function definition.

Definition 1 (QoS Utility Function): Suppose there are x negative QoS attributions to be maximized and y positive QoS attributions to be minimized. The QoS utility functions for a Web service $s_{ij} \in s_i$ and composited service S is defined as follows:

$$U(s_{ij}) = \sum_{\alpha=1}^{x} \frac{Q_{i,\alpha}^{max} - q_\alpha(s_{ij})}{Q_{i,\alpha}^{max} - Q_{i,\alpha}^{min}} . \omega_\alpha + \sum_{\beta=1}^{y} \frac{q_\beta(s_{ij}) - Q_{i,\beta}^{min}}{Q_{i,\beta}^{max} - Q_{i,\beta}^{min}} . \omega_\beta \tag{1}$$

$$U(S) = \sum_{\alpha=1}^{x} \frac{Q_{\alpha}^{max} - q_\alpha(S)}{Q_{\alpha}^{max} - Q_{\alpha}^{min}} . \omega_\alpha + \sum_{\beta=1}^{y} \frac{q_\beta(S) - Q_{\beta}^{min}}{Q_{\beta}^{max} - Q_{\beta}^{min}} . \omega_\beta \tag{2}$$

with

$$\begin{cases} Q_\alpha^{max} = \sum_{i=1}^{x} Q_{i,\alpha}^{max} \left(Q_{i,\alpha}^{max} = \max_{\forall s_{ij} \in s_i} q_\alpha(s_{ij}) \right) \\ Q_\beta^{max} = \sum_{i=1}^{y} Q_{i,\beta}^{max} \left(Q_{i,\beta}^{max} = \max_{\forall s_{ij} \in s_i} q_\beta(s_{ij}) \right) \end{cases} \tag{3}$$

$$\begin{cases} Q_\alpha^{min} = \sum_{i=1}^{x} Q_{i,\alpha}^{min} \left(Q_{i,\alpha}^{min} = \min_{\forall s_{ij} \in s_i} q_\alpha(s_{ij}) \right) \\ Q_\beta^{min} = \sum_{i=1}^{y} Q_{i,\beta}^{min} \left(Q_{i,\beta}^{min} = \min_{\forall s_{ij} \in s_i} q_\beta(s_{ij}) \right) \end{cases} \tag{4}$$

Where ω_α and ω_β represents user's preferences, is the weights for each QoS attributes and satisfy $\sum_{\alpha=1}^{x} \omega_\alpha + \sum_{\beta=1}^{y} \omega_\beta = 1 (0 < \omega_\alpha, \omega_\beta < 1)$; $Q_{i,k}^{max}$ ($0 < k < \alpha$ or $0 < k < \beta$) is the maximum value of the k-th attribute in all service candidates of the service class s_i , and similarly $Q_{i,k}^{min}$ is the minimum value in class s_i ; Q_k^{max} is the summation of each $Q_{i,k}^{max}$ in the composition service S and similarly Q_k^{min} is the summation of each $Q_{i,k}^{min}$ in the composition service S .

In Definition 1, for negative QoS attributes, we compare the distance $Q_{i,k}^{max} - q_k(s_{ij})$ between the maximum value in a service class s_i and the value of a service candidate s_{ij} with the distance $Q_{i,k}^{max} - Q_{i,k}^{min}$ between the maximum and minimum in a service class s_i . For positive QoS attributes, conversely, we compare the distance $q_k(s_{ij}) - Q_{i,k}^{min}$ between the value of a service candidate s_{ij} and the minimum value in a service class s_i with the distance $Q_{i,k}^{max} - Q_{i,k}^{min}$ between the maximum and minimum in a service class s_i . All QoS attributes are weighted by user's preferences so that the QoS utility function does not rely on any attribute, but rely on user's preferences. The weight ω_k is a very important factor which represents the user's preference to the k-th attribute in a composite service. It is usually assigned any value within [0, 1] and all the weight values satisfy $\sum_{k=1}^{r} \omega_k = 1$. The larger the numerical value is, the more attention from users to the requirement of this attribute, and this attribute will occupy the more important position in the service selection process. Therefore, we should set the value of ω_k according to the user's preference to the k-th attribute. Of course, when a user does not known how to assign the ω value of each attribute, the users can assign their weight ω on each attribute through linguistic terms, such as very unimportant, unimportant, medium, important, and very important. The distribution of specific level is shown in Table 2 [13]. If the users do not specified the weight ω of each QoS attribute, in this paper, we will allocate an average ω value of each attribute.

Table 2. The weight values

Linguistic terms	Weight values
Very unimportant	0.125
Unimportant	0.25
Medium	0.5
Important	0.75
Very important	1

In a Web service selection, in order to obtain the optimal composite service, we should add the global QoS constraints to filter Web services that are not satisfied user's QoS requirements. This is helpful to shorten the computation time and improve the efficiency of service selection. We can obtain different optimal composite services with the difference constraint sets. For example, for a given vector of global QoS constrains $C = \{C_1, C_2, ..., C_m\}$ $(0 < m < r)$. Each constrain can be expressed in terms of upper or lower bounds for the aggregated QoS values. We could take the advantage of these m constraints to select the optimal composite service.

In this paper, we consider both the negative attributes and positive attributes in the service selection process for composite services.

2.2 Related work

There are plenty of covert channel approaches in literature. Here, we will only review some notable work.

Ran [14] proposed a new Web service discovery model by combining functional with non-functional requests for

service discovery. The model opens up a new wide research area that select Web services based on QoS. From the beginning of the research, many researchers have paid more attention to the QoS-based Web service selection and composition [5, 6, 15, 16]. For example, Liu et al. [15] proposed an open, fair and dynamic QoS computation model for Web services selection, but this model existed a shortcoming that it only considered the QoS attributes without user's requirements and preferences.

Integer Programming [8, 9, 17, 18] was often used to solve the selection of the optimal composite service. In [8], the authors proposed a novel service selection optimization approach. This approach contained the following three main steps. First, loops peeling are adopted in the optimization. Second, if a feasible solution for the web service composition problem does not exist, negotiating QoS parameters is performed in order to determine new quality values for web service invocations. Finally, a new class of global constraints, which allows the execution of stateful web service components, is introduced. In [17] , the authors presented a middleware platform for Web services composition by solving the max of utility functions over QoS attributes meanwhile satisfy user's requirements. They used Integer Programming to solve the optimal utility values. Our previous work [18] employed Cloud Model for pruning the redundant services, then using Mix Integer Programming to select the optimal services.

Moreover, some researchers [19, 20] considered the impact of QoS dependencies in service selection processes. Baraka et al. [19] presented a correlation-aware service selection approach for handing QoS dependencies among Web services and improved the composition quality. The approach first models the quality dependencies among Web services and then uses correlation-aware search space reduction techniques to eliminate uninteresting compositions from the search space before selection. Also, Feng et al. [20] proposed a novel approach that considered the QoS-aware service composition problem in the presence of service-dependent QoS, user-provided topological and QoS constraints. The approach effectively handles service-dependent QoS by directly integrating it into the composition process rather than after the composition and significant improves the performance on real life scenarios with complex service and QoS dependencies.

Although the above-mentioned approaches perform well in service selection process, because they did not consider the uncertainty of QoS, they are impossible to guarantee the reliability of the solution. Different from our another previous work [21], this paper designs a service composition middleware framework to support service selection in heterogeneous distributes software systems, and then propose an efficient and reliable service selection approach by computing QoS uncertainty. Moreover, this paper focuses on new and more comparison experiments to evaluate the proposed approach

3. MIDDLEWARE FRAMEWORK

Aimed at service selection in heterogeneous distributes software systems, a service composition framework is presented as shown in Fig. 1. There are three distinct components in our framework, i.e., Discovery Engine, Selection Engine, and Composition Engine.

Discovery Engine: its main function is obtaining valid service lists, user's QoS constraints and preferences. Service providers from heterogeneous distributed systems publish their services in a UDDI [22] (service registry) where they can be found by users or service requestors based on their functional and nonfunctional properties. Given an abstract composition request according a user's requests, the Discovery Engine uses UDDI to locate available services with service providers for each task. UDDI uses syntactic or semantic functional matching between the tasks and service descriptions to find a list of candidate services for each task. QoS constraints and user's preferences can be obtained from Users [23].

Selection Engine: it is the core of the middleware framework. Its function is finding selection results by aggregated QoS functions. In this engine, Computing QoS Uncertainty model and Uncertain Services Filtering model are used to prune unreliable services and reduce the solution space, respectively. Service Selection model adopts a 0-1 Integer Programming to find and select the optimal services according user preferences and OoS constraints.

Composition Engine: Having found suitable services, Composition Engine can bind against that concrete services and invoke the selected concrete services one by one on the basis of selection results from Selection Engine.

For easy understanding, Fig. 3 shows a seven-step process of service selection with service composition middleware. Then based on the proposed service composition middleware, we propose an efficient and reliable service selection approach by computing QoS uncertainty

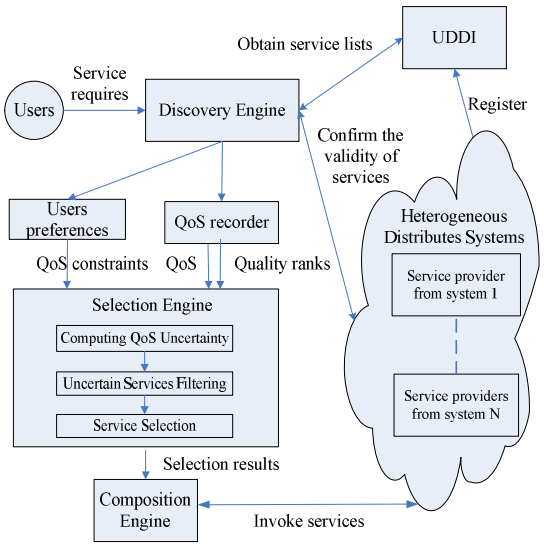

Figure 1. The framework for service composition middleware.

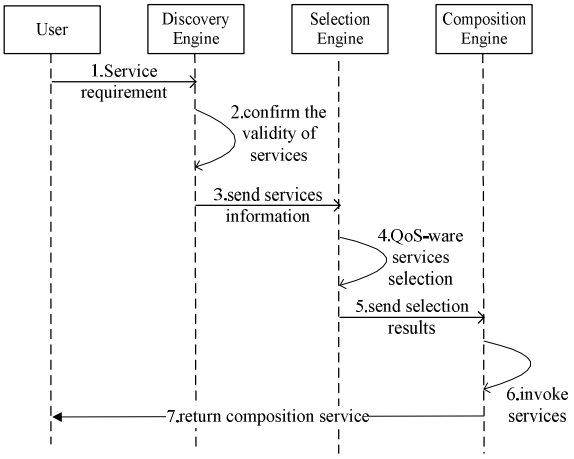

Figure 2. Process of service selection.

4. OUR APPROACH

The proposed approach in this paper contains three phases. The first phase is QoS uncertainty computing, in which we adopt information theory and probability theory to transform the QoS values into two qualitative concepts. They represent the stability of a service, aiming to rank services. The Second phase is uncertain services filtering, in which we prune the uncertain service candidates according to the two qualitative concepts, aiming to reduce the solution space of service selection. The third phase is service selection, in which, we design a reliability fitness function to find the most reliable composite service with low computation time.

4.1 Computing QoS uncertainty

We first normalize the quantitative QoS values into the domain [0, 1], which is convenient for data processing and uniform QoS attribute values. Then we employ information theory and probability theory to compute the QoS uncertainty by transforming QoS quantitative values into two QoS qualitative concepts. Then according to the two qualitative concepts, a service with consistently good QoS can be distinguished from other services. For illustration, we firstly give the following relevant concepts.

1) Data normalizing

Through the normalization process, limiting the values within a certain range (e.g., [0-1]), is convenient for QoS utility function. Data normalizing means that the original QoS values will be scaled proportionally. There are many ways for normalizing, such as linear conversion, logarithmic conversion, cotangent conversion, etc. In this paper, we adopt the linear conversion to normalize the QoS value. The specific formula is as follows:

$$y = (x - Minvalue) / (Maxvalue - Minvalue) \quad (5)$$

Where x and y represents the corresponding values before and after QoS data conversion, respectively; *Maxvalue* and *Minvalue* represent the maximum and minimum of the original data, respectively .

2) Entropy

In information theory, Entropy is used to measure the expectation value of a random variable which shows the average uncertainty of overall information source (IS). For a particular IS, the Entropy value is changed with the different of statistical properties. In general, the greater the uncertainty of the variable is, the larger the Entropy value is (this means the greater disorder of its corresponding IS). In this paper, we consider the real-world QoS historical values for a service as a discrete IS, and then we employ Entropy to filter services by the following definition of the Entropy.

Definition 2 (Entropy): Let X be the random variable of IS and $H(X)$ be the Entropy value of X. $\{X_1, X_2, ..., X_n\}$ is the range of X. Then, in this case, the $H(X)$ can be work out by the following:

$$H(X) = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i) \quad (6)$$

Where $p(x_i)$ represents x_i 's probability and $p(x_i) \geq 0$ and $\sum_{i=1}^{n} p(x_i) = 1$. Note that the Entropy value $H(X) \geq 0$.

3) Variance

In probability theory, Variance is used to measure the deviation between the random variables and its mathematical Expectation. The larger the Variance value is, the more dispersed the random variable's value relative to the Expectation is, and the greater the disorder degree of the sample data is. Since the Variance's characteristics can fully reflect the stability of the IS, so we could adopt Variance to filter the uncertain services in composite services. The following gives the definition of the Variance.

Definition 3 (Variance): Let X be the random variable of IS. Let EX be the mathematical Expectation of X, and DX be the Variance of the X. Then, the EX and DX can be work out by following:

$$E(X) = \sum_{i=1}^{n} x_i p(x_i) \quad (7)$$

$$D(X) = E(X^2) - (E(X))^2 \quad (8)$$

Where $p(x_i)(p(x_i) \geq 0)$ represents x_i 's probability and $\sum_{i=1}^{n} p(x_i) = 1$.

Since the Entropy and Variance both fully reflects the stability of the IS, why do we adopt both and not just adopt one of them? Considering that the existence of the same two entropy values or two same variances which represent two ISs, respectively. In this situation, if we only filter the uncertain services according to one of them, we will not obtain the ideal results. For example, there are two random variables: X_1={0.01, 0.01, 0.06, 0.06, 0.09, 0.09} and X_2={0.04, 0.04, 0.05, 0.05, 0.06, 0.06}, respectively, represent the response times of six same users accessing two different services (WS_1, WS_2). According to (6), the Entropy values can be calculated, but we find that $En(X_1) = En(X_2)$. Then we could not prune a service according its Entropy value. Furthermore, we calculate their Variances *Dx(X1), Dx(X2)*. Since $Dx(X_1) > Dx(X_2)$, it indicates that the WS_2 is

more stable and reliable than WS_1, and then we should prune WS_1. Hence, this example demonstrates intuitively the benefit of using both.

Currently, Entropy and Variance have been applied to many fields, such as financial market, risk investment, etc. They have achieved lots of good results which provide a basis and reference for our approach. By using the Entropy and Variance for abandoning uncertain services will help us select the lowest uncertain services.

4.2 Uncertain services filtering

Through the above QoS uncertainty computing, we could use the two Entropy *En* and Variance *Dx* to filter the uncertain services. The *En* will help us to filter the services coarse-grained. Suppose there are l functionally equivalent services. The top l_1 ($l_1 < l$) smallest value of *En* will be selected and the rest will be discarded. We then filter the l_1 services by *Dx*. We will select the top of smallest l_2 services from l_1. Finally, we will obtain l_2 low uncertain services.

For service filtering, we take three services WS_1, WS_2 *and* WS_3 that offer the similar hotel service as an example to illustrate the different implications. In the example, the performance of WS_1, WS_2 *and* WS_3 is recorded by a series of transaction logs, which helps capture the actual QoS delivered by each provider in practical application. Because the dynamic environment in which these service providers operate causes the uncertainty of their performance, this can be reflected by the fluctuation among different transactions. For the easy of illustration, although the actual number of transactions should be much larger, we consider only 10 transactions with the four services, respectively. In Table III, it gives these transactions with a focus on the attribute of response time and each value represents the response time when a user invokes a service. Then the aggregated QoS values (S_1, S_2 *and* S_3) obtained by averaging all transactions are given in the last row of Table 3.

Table 3. A Set of Service Transactions

WS_1		WS_2		WS_3	
ID	***Respone time(ms)***	**ID**	***Respone time(ms)***	**ID**	***Respone time(ms)***
S_{10}	12	S_{20}	23	S_{30}	18
S_{11}	31	S_{21}	29	S_{31}	16
S_{12}	15	S_{22}	24	S_{32}	31
S_{13}	32	S_{23}	22	S_{33}	32
S_{14}	14	S_{24}	28	S_{34}	19
S_{15}	32	S_{25}	29	S_{35}	33
S_{16}	31	S_{26}	25	S_{36}	34
S_{17}	36	S_{27}	28	S_{37}	17
S_{18}	34	S_{28}	23	S_{38}	20
S_{19}	13	S_{29}	27	S_{39}	33
S_1	***24.9***	***S_2***	***25.8***	***S_3***	***25.3***

From Table 3, the aggregated QoS values of S_1 is less than that of S_2 and S_3, i.e., $S_1 < S_2$, $S_1 < S_3$. In traditional service selection approach, service S_1 frequently is selected as a service component in service composition because of 24.9<25.8 and 24.9<25.3. However, after analyzing each transaction of these three services in great depth, we find the following three important facts that may be ignored by some existing service selection approaches:

1) In service WS_1, five transactions are in interval [31, 35], four transactions are in interval [11, 15] and one transaction is in interval [36, 40]. In service WS_2, five transactions are in interval [21, 25] and the other five transactions are in interval [26, 30]. Similarly, in service WS_3, five transactions are in interval [15, 20] and the other five transactions are in interval [31, 35]. That means although the average response time of service WS_1 is slightly less than that of WS_2 and WS_3, the response time of service WS_1 is larger than that service WS_2, WS_3 in most transactions.

2) By comparing service WS_2 with WS_3, we could find the transactions are evenly distributed in two intervals. But the service WS_3 has a larger span distribution than service WS_2, which means service WS_2 is more stable than service WS_3.

3) The response time of service WS_1 and WS_3 is more volatile than that of service WS_2, i.e., service WS_1 and WS_3 with a large variance on its QoS, but service WS_2 with consistently good QoS.

Hence, according to the three facts, if service WS_1 or WS_3 is selected as a service component, the actual execution result of service WS_1 or WS_3 may deviates from their average response time, which will result in poor composition service QoS or service selection failure. It is obvious that service WS_2 is more stable than other two services. So how to compute the uncertainty of service that is used to distinguish a service with a consistently good QoS from other services with large variance on its QoS, is an important issue.

In this study, we adopt uncertain service filtering to Table III. Then the Entropy and Variance {*En, Dx*} of services WS_1, WS_2 *and* WS_3 can be calculated, i.e., NS_1 = {1.361, 106.25}, NS_2= {1, 6.25} and NS_3= {1, 56.25}. Since the WS_1's *En* is higher than WS_2 and WS_3 (i.e., 1.361>1), the uncertainty level of service WS_1 is smaller than that of service WS_2, WS_3. In addition, the WS_3's *Dx* is higher than WS_2 (i.e., 56.25 > 6.25), so the uncertainty level of service WS_3 is also smaller than that of service WS_2. This means that the QoS of service WS_2 is consistently good but service WS_1, WS_3 with a large variance on its QoS. Thus, the service WS_2 should be selected as a service candidate rather than the other two services, which is different from some traditional approaches. Then by setting different threshold parameters of *En* and *Dx* according to different service environment, the services with a large variance on its QoS and the services with a consistently good QoS can be distinguished. Then the latter will be as service candidates prior to the former for reliable service selection.

By this way, our approach can filter the uncertain services, thereby reduce the solution space of service selection and shorten the computation time in service composition.

4.3 Service Selection

After QoS uncertainty computing and uncertain services filtering, service candidates with consistently good QoS can be discovered in each service class. Then, a service selection solution has to be used to find the most reliable service of

each class with global QoS constraints. In this paper, a 0-1 Integer Programming model is used to solve the optimization problem of service selection based on the filtered services. Recently, the Integer Programming has been used to solve the service composition problem by several researches [8, 9, 17, 18] and achieved good results.

In this paper, we propose a reliability fitness function definition to reflect the reliability of service selection. The larger the reliability fitness value is, the more reliable the solution of service selection is. .

Definition 4 (Reliability Fitness Function): Suppose there are *x* negative QoS attributions and *y* positive QoS attributions. The reliability fitness function is defined as follows:

$$F(S)=\sum_{\alpha=1}^{x}\frac{Q_\alpha^{max}-\sum_{i=1}^{n}\sum_{j=1}^{l}x_{ij}\cdot q_\alpha(s_{ij})}{(Q_\alpha^{max}-Q_\alpha^{min})\cdot\sigma_\alpha}\cdot\omega_\alpha+\sum_{\beta=1}^{y}\frac{\sum_{i=1}^{n}\sum_{j=1}^{l}x_{ij}\cdot q_\beta(s_{ij})-Q_\beta^{min}}{(Q_\beta^{max}-Q_\beta^{min})\cdot\sigma_\beta}\cdot\omega_\beta \quad (9)$$

Where x_{ij} is a binary decision variable for representing the service candidate whether is selected; a service candidate s_{ij} is selected in the optimal composition service if its corresponding variable *x* is set to 1 and 0 otherwise; σ is the standard deviation of a sample that consists of all the service candidates from all service classes and it can reflect the overall volatility of all service candidates; σ_k ($0<k<\alpha$ or $0<k<\beta$) is the standard deviation of a sample that consists of *k-th* attribute values for all the service candidates from all service classes and it can reflect the overall volatility of all service candidates; $q_k(s_{ij})$ represents the *k*-th attribute value in service s_{ij} and Q_α^{max} , Q_α^{min} , Q_β^{max} and Q_β^{min} can be calculated by formula (4).

In the phase of service selection, we can get the most reliable composite service that satisfies all global QoS constraints. So the 0-1 integer programming model can be formulated as follows:

$$Max \quad F(S) \quad (10)$$

$$Subject\ to \begin{cases} \sum_{i=1}^{n}\sum_{j=1}^{l}q_k(s_{ij})\cdot x_{ij}\le(\ge)C_k, 1\le k\le m\le r \\ \sum_{j=1}^{l}x_{ij}=1, 1\le i\le n, x_{ij}\in\{0,1\} \end{cases} \quad (11)$$

Where *r* is the number of QoS attributes; *m* is the number of QoS constraints; *n* is the number of service class; C_k represents the *k*-th global QoS constrains with respect to the *k*-th QoS attribute.

By solving (10) and (11), a list of reliable services are obtained and returned from each class to service broker providing a composition service for users.

5. EXPERIMENTS

We have implemented our approach and experimented with a real-world dataset and a synthetic dataset. We compare our approach with Global approach [8] and Skyline approach [3] by conducting several groups of experiments. Moreover, we also studied the parameters of our approach. The experiments indicate that the solution of our approach is more reliable than other approaches, and shows that the computation time of our approach is much shorter than other approaches.

5.1 Experiment Setup

We conduct experiments using two types of datasets. The first is a real-world Web service QoS dataset named WSDream dataset from [24, 25]. WSDream dataset contains nearly 2 million real-world QoS Web service invocation records. Values of two QoS attributes (i.e., *Response time*, *Throughput*) are collected by 339 service users on 5825 Web services. It also contains the information of these 5825 Web services and 339 users.

In order to make sure the experiments results of our approach are not biased on the used WSDream dataset, we also conduct experiments with a second dataset which is a randomly generated dataset (named Random dataset) that contains 10,000 services with two QoS attributes.

We perform several experiments of the QoS-based service composition problem. Each experiment consists of a service composition request with *n* service classes, *l* service candidates per class, and *m* global QoS constraints. By varying the number of these parameters, we collect the results of each experiment, where each unique combination of these three parameters represents one experiment. We first perform the experiment using (9) to find the optimal selection that satisfies all global QoS constraints meanwhile maximizing the reliability fitness value. We record the required computation time t_1 and the obtained reliability fitness value v_1 for each experiment. We then provide the same experiments on all approaches and record the computation time t_2 and the reliability fitness value v_2 for the same experiment, respectively. Then we verify the accuracy of the experiment by comparing t_1 with t_2 and v_1 with v_2 for each experiment.

In our experiments, the number of QoS attributes *r* is set to 2, QoS constraints *m* is also set to 2, the number of service candidates per service class varies from 100 to 1000, and the weight for the two attributes is equally 0.5. For WSDream dataset, the number of service class *n* is fixed to 5 and to 10 for the Random dataset. And, the number of historical transactions is set to 250 for WSDream dataset and to 500 for the Random dataset. In our approach, we use *En* to filter out half of the service candidates in first phase and then in second phase, use *Dx* to select 2/5 service candidates from the selected service candidates in first phase. Then we will get 1/5 service candidates which are relative stability and QoS certainty.

All the experiments are performed on the same computer with Intel(R) Xeon(R) 2.6GHz processor, 32.0GB of RAM, Windows Server 2008 R2, and Matlab R2013a. All experiments performed 20 times.

5.2 Reliability Comprations

The reliability fitness function has a theoretical maximum $F_{max}(S)$ which represents the value when obtaining the optimal service composition in ideal situation.

The $F_{max}(S)$ is not fixed and varies with the difference of weight ω and standard deviation σ . In this paper, we define the concepts of Reliability as follows:

$$Reliability = F_{fitness} / F_{max}(S) \quad (11)$$

Where $F_{fitness}(S)$ represents the reliability fitness value obtained in each experiment; $F_{max}(S)$ as constant is 62.5 for WSDream dataset and 75 for Random dataset. They are obtained on basis of analyzing a larger number of experimental results.

As shown in Fig. 3 and Fig. 4, we compare our approach with Global approach and Skyline approach in terms of the Reliability. We could find that the Reliability obtained by our approach is always larger than obtained by Global approach and Skyline approach with the increasing number of service candidates. These experimental results illustrate that our approach verifies effectively the influence of QoS uncertainty on the quality of composition service and largely improves the reliability of service selection. By using *En* and *Dx* to monitor service's QoS historical transactions, our approach effectively identify that the large variances services on its QoS and prune them. Our approach greatly improves the reliability of service composition.

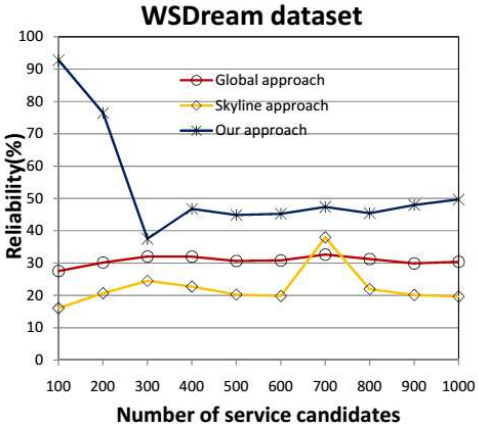

Figure 3. Reliability with WSDream dataset.

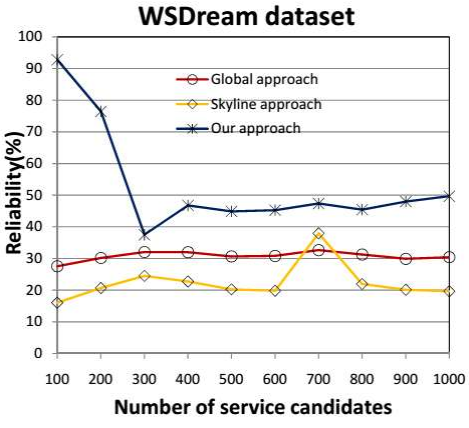

Figure 4. Reliability with WSDream dataset.

5.3 Computation Time Comparisions

In this section, we perform experiments to compare our approach with Global approach and Skyline approach on the computation time.

From Fig. 5 and Fig. 6, we could find that the computation time is growing with the growing of service candidates, and for any service candidates, the computation time of our approach is always shortest in all approaches. The experimental results illustrate that our approach significantly reduces the time cost of service composition because of the searching space is reduced in our approach.

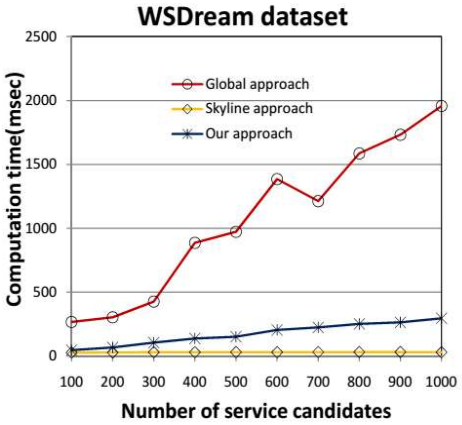

Figure 5. Computing time with WSDream dataset.

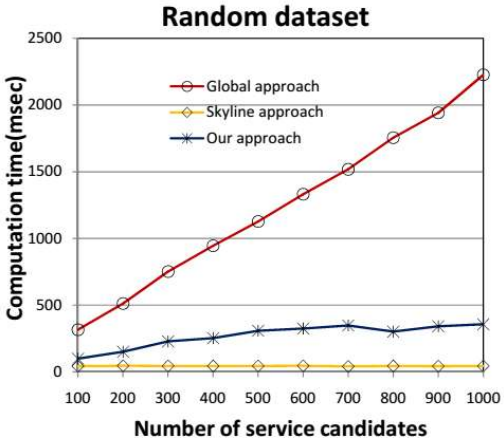

Figure 6. Computing time with Random dataset.

5.4 Parameters Study

5.4.1 The Studies of The Parameters En and Dx

In this section, we study the two parameters *En and Dx*. We fixe one of the two parameters, and then obtain the Reliability and Computation time with the change of another one. Fig. 7 and Fig. 8 list the experimental results (Reliability, Computation time) with the change of *Dx*. We fixed the number of service candidates is 500 and the *En* is 0.5 which represents selecting half of service candidates by *En*. And, the *Dx* is varies from 0.05 to 0.5 which represents the proportion of final selected service candidates.

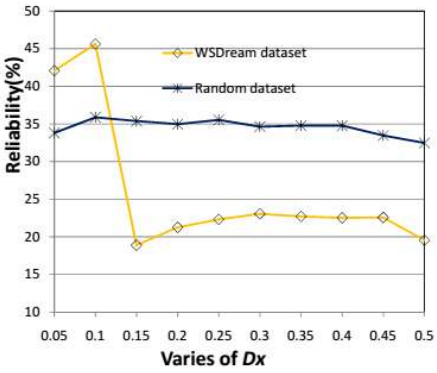

Figure 7. Reliability with *Dx*.

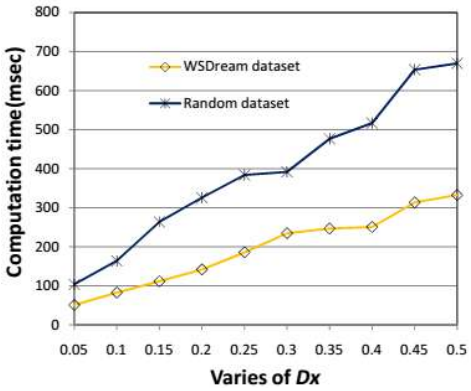

Figure 8. Computation time with *Dx*.

Fig. 9 and Fig. 10 list the experimental results with the change of *En*. We also fixe the number of service candidates is 500 and the *Dx* is 0.2. And, the *En* is varies from 0.55 to 1 which represents the proportion of selected candidate services by *En*. These experiments better describe and enrich our approach and let the readers deeply understand the benefits of our approach.

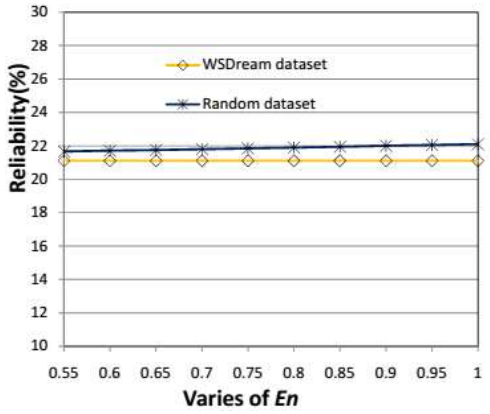

Figure 9. Reliability with *En*.

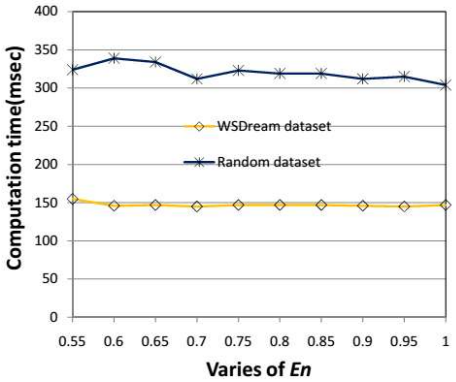

Figure 10. Computation time with *En*.

5.4.2 The Study of The Parameter ω

In this section, we also verify the weight ω for the influence of reliable composition service. The weight ω represents the user's requests for each QoS attribute and it is very important for service composition.

From Fig. 11-14 list the results in both datasets. Each value is the average of 10 results which represent the number of service candidates varies from 100 to 1000. And the weights of Response time is varies from 0 to 1, corresponding, the weights of Throughput is varies from 1 to 0. From the results, we could find that no matter how the ω is allocated, and the results obtained by our approach are better than MIP, i.e. the average of reliability fitness value obtained by our approach is larger than MIP and the average of computation time is smaller than MIP.

All above experimental results demonstrate more fully that our approach greatly improves the reliability of service composition and shortens the computation time of service composition. Our approach can select the optimal composited service spending less time and obtaining greater reliability.

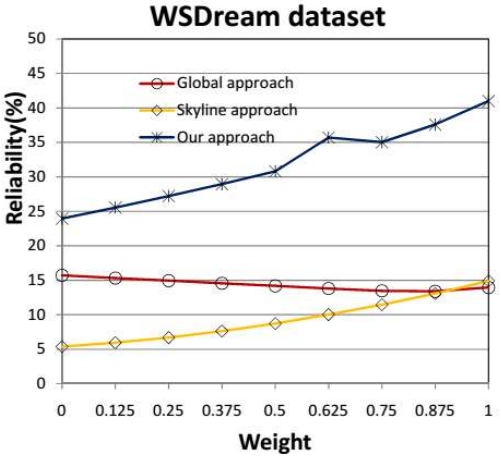

Figure 11. Reliability with ω based on WSDream.

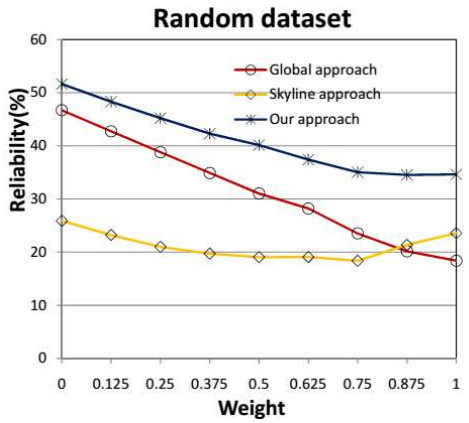

Figure 12. Reliability with ω based on Random dataset.

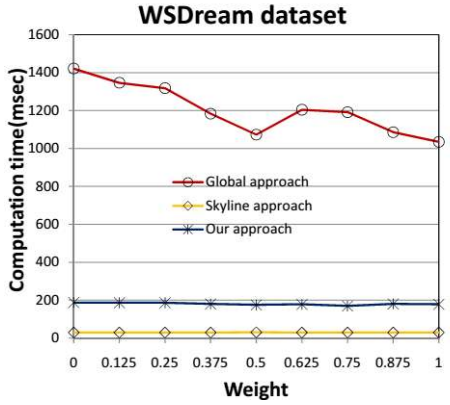

Figure 13. Computation time with ω based on WSDream.

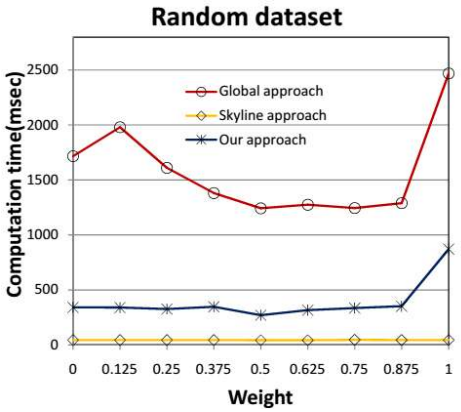

Figure 14. Computation time with ω based on Random dataset.

6. CONCLUSIONS

In this paper, based on a service composition framework, we presented an efficient and reliable service selection approach. Our approach uses the two concepts Entropy and Variance to compute the uncertainty of QoS and then filter these services with high uncertainty. Finally, we design a reliability fitness function to select the most reliable composite services by 0-1 Integer Programming. We evaluate our approach using both real-world and randomly generated service datasets. The result shows that our approach obtains more reliable solution with lower computation time than other approaches. This means that our approach can perform service selection on basis of user's requests more efficiently and effectively.

In our future work, we will strengthen our approach and continue to research more efficient service selection approaches. We aim to help users find the optimal composite service according to their QoS requirements and user preferences in the future.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant No. 61202435 and 61272521 and the Natural Science Foundation of Beijing under Grant No.4132048.

REFERENCES

[1] R. Mietzner, C. Fehling, D. Karastoyanova, F. Leymann, Combining horizontal and vertical composition of services, in: IEEE International Conference on Service-Oriented Computing and Applications (SOCA 2010), 2010, pp. 1-8.

[2] G. Canfora, M. Di Penta, R. Esposito, M.L. Villani, A framework for QoS-aware binding and re-binding of composite web services, Journal of Systems and Software, 81 (2008) 1754-1769.

[3] M. Alrifai, D. Skoutas, T. Risse, Selecting skyline services for QoS-based web service composition, in: the 19th international conference on World Wide Web (WWW 2010), 2010, pp. 11-20.

[4] M. Alrifai, T. Risse, Combining global optimization with local selection for efficient QoS-aware service composition, in: the 18th international conference on World Wide Web (WWW2009), 2009, pp. 881-890.

[5] G. Canfora, M.D. Penta, R. Esposito, M.L. Villani, An approach for QoS-aware service composition based on genetic algorithms, in: the 7th annual conference on Genetic and evolutionary computation (GECCO 2005), 2005, pp. 1069-1075.

[6] G. Canfora, M. Di Penta, R. Esposito, M.L. Villani, QoS-aware replanning of composite Web services, in: IEEE International Conference o Web Services (ICWS 2005), 2005, pp. 121-129.

[7] S.-Y. Hwang, E.-P. Lim, C.-H. Lee, C.-H. Chen, Dynamic Web service selection for reliable Web service composition, IEEE Transactions on Services Computing, 1 (2008) 104-116.

[8] D. Ardagna, B. Pernici, Adaptive service composition in flexible processes, IEEE Transactions on Software Engineering, 33 (2007) 369-384.

[9] T. Yu, Y. Zhang, K.-J. Lin, Efficient algorithms for Web services selection with end-to-end QoS constraints, ACM Transactions on the Web, 1 (2007) 1-26.

[10] Y. Qi, A. Bouguettaya, Computing Service Skyline from Uncertain QoWS, IEEE Transactions on Services Computing, 3 (2010) 16-29.

[11] K.S. Candan, W.-S. Li, T. Phan, M. Zhou, Frontiers in information and software as services, in: the 25th IEEE International Conference on Data Engineering (ICDE 2009), 2009, pp. 1761-1768.

[12] J. Cardoso, A. Sheth, J. Miller, J. Arnold, K. Kochut, Quality of service for workflows and web service processes, Web Semantics, 1 (2004) 281-308.

[13] S.S. Yau, Y. Yin, QoS-Based Service Ranking and Selection for Service-Based Systems, in: IEEE International Conference on Services Computing (SCC 2011), 2011, pp. 56-63.

[14] S. Ran, A model for web services discovery with QoS, SIGecom Exchanges, 4 (2003) 1-10.

[15] Y. Liu, A.H. Ngu, L.Z. Zeng, QoS computation and policing in dynamic web service selection, in: the 13th international World Wide Web (WWW 2004), 2004, pp. 66-73.

[16] K. Guosheng, L. Jianxun, T. Mingdong, L. Xiaoqing, K.K. Fletcher, Web Service Selection for Resolving Conflicting Service Requests, in: IEEE International Conference on Web Services (ICWS 2011), 2011, pp. 387-394.

[17] L. Zeng, B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, H. Chang, QoS-Aware Middleware for Web Services Composition, IEEE Transactions on Software Engineering, 30 (2004) 311-327.

[18] W. Shangguang, Z. Zheng, S. Qibo, Z. Hua, Y. Fangchun, Cloud model for service selection, in: 30th IEEE Conference on Computer Communications Workshops on Cloud Computing (INFOCOM WKSHPS), a, 2011, pp. 666-671.

[19] L. Barakat, S. Miles, M. Luck, Efficient Correlation-Aware Service Selection, in: IEEE 19th International Conference on Web Services (ICWS 2012), 2012, pp. 1-8.

[20] F. Yuzhang, N. Le Duy, R. Kanagasabai, Dynamic Service Composition with Service-Dependent QoS Attributes, in: IEEE 20th International Conference on Web Services (ICWS 2013), 2013, pp. 10-17.

[21] L. Sun, S. Wang, J. Li, Q. Sun, F. Yang, QoS Uncertainty Filtering for Fast and Reliable Web Service Selection, in: IEEE International Conference on Web Services (ICWS 2014), 2014, pp. 550-557.

[22] M.P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann, Service-Oriented Computing: State of the Art and Research Challenges, Computer, 40 (2007) 38-45.

[23] F. Li, F. Yang, K. Shuang, S. Su, A Policy-Driven Distributed Framework for Monitoring Quality of Web Services, in: IEEE International Conference on Web Services (ICWS 2008), 2008, pp. 708-715.

[24] Z. Zheng, Y. Zhang, M.R. Lyu, Distributed QoS evaluation for real-world Web services, in: IEEE 8th International Conference on Web Services (ICWS 2010), 2010, pp. 83-90.

[25] Z. Yilei, Z. Zibin, M.R. Lyu, Exploring Latent Features for Memory-Based QoS Prediction in Cloud Computing, in: the 30th IEEE Symposium on Reliable Distributed Systems (SRDS 2011), 2011, pp. 1-10.

Shangguang Wang is an associate professor at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. He received his Ph.D. degree in computer science at Beijing University of Posts and Telecommunications of China in 2011. His PhD thesis was awarded as an outstanding doctoral dissertation by BUPT in 2012. His research interests include Service Computing, Mobile Services, and QoS Management.

Lin Huang received the M.E. degree in computer science and technology from the Institute of Network Technology, Beijing University of Posts and Telecommunications, in 2012. Currently, she is a Ph.D. candidate at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. Her research interests include Reputation measurement, Web service selection.

Lei Sun received the BEng from Qingdao University. He is currently Master student at the Beijing University of Posts and Telecommunications. His research interests include service computing and distributed computing.

Ching-Hsien Hsu is a professor in the department of computer science and information engineering at Chung Hua University, Taiwan. His research includes high performance computing, cloud computing, parallel and distributed systems, and ubiquitous/pervasive computing and intelligence. He has been involved in more than 100 conferences and workshops as various chairs and more than 200 conferences/workshops as a program committee member. He is the editor-in-chief of an international journal on Grid and High Performance Computing and has served on the editorial board for approximately 20 international journals.

Fangchun Yang received his PhD degree in communication and electronic systems from Beijing University of Posts and Telecommunication in 1990. He is currently a professor at the Beijing University of Posts and Telecommunication, China. He has published 6 books and more than 80 papers. His current research interests include network intelligence, services computing, communications software, soft switching technology, and network security. He is a fellow of the IET.