QoS-Based and Network-Aware Web Service Composition across Cloud Datacenters

Dandan Wang[1], Yang Yang[1] and Zhenqiang Mi[1]
[1]School of Computer and Communication Engineering, University of Science and Technology Beijing
Beijing, 100083, China
[e-mail: wdd_ustb@163.com, yyang@ustb.edu.cn, mizq@ustb.edu.cn]
*Corresponding author: Yang Yang

Received May 25, 2014; revised July 16, 2014; revised September 17, 2014; accepted January 21, 2015; published March 31, 2015

Abstract

With the development of cloud computing, more and more Web services are deployed on geo-distributed datacenters and are offered to cloud users all over the world. Through service composition technologies, these independent fine-grain services can be integrated to value-added coarse-grain services. During the composition, a number of Web services may provide the same function but differ in performance. In addition, the distribution of cloud datacenters presents a geographically dispersive manner, which elevates the impact of the network on the QoS of composite services. So it is important to select an optimal composition path in terms of QoS when many functionally equivalent services are available. To achieve this objective, we first present a graph model that takes both QoS of Web services and QoS of network into consideration. Then, a novel approach aiming at selecting the optimal composition path that fulfills the user's end-to-end QoS requirements is provided. We evaluate our approach through simulation and compare our method with existing solutions. Results show that our approach significantly outperforms existing solutions in terms of optimality and scalability.

Keywords: Network, QoS, Service composition, Web services, Cloud datacenters

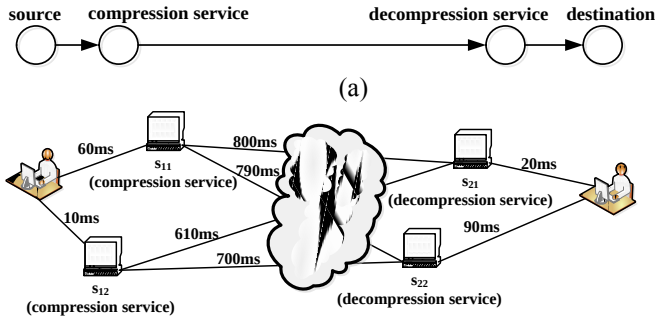
This work was supported by the National Science Foundation of China (Grant No. 61202508，61472033，61272432, 61370092) and Fundamental Research Funds for the Central Universities (FRF-TP-14-045A2).

http://dx.doi.org/10.3837/tiis.2015.03.008 **ISSN : 1976-7277**

1. Introduction

The development of cloud computing is making users increasingly accustomed to using the Internet to gain software resources in the form of Web services. Web services are self-describing software applications that can be advertised, located and utilized across the Internet following a set of standards such as SOAP, WSDL, and UDDI [1, 2]. In cloud computing, Web services are deployed in datacenters which are usually geographically dispersive and communicate with each other through Internet. Through service composition technologies, loosely-coupled services that are deployed dispersively can be integrated to complex and value-added composite services as long as each component service's interface specification is subject to standard protocols.

Consider an example of the large data transmission shown in **Fig. 1(a)**. A connection is established for large data transmission between two end systems which are located in different domains. However, the network bandwidth between two domains is limited. Therefore, compression and decompression are needed in the source and destination domains respectively. Then the composite service of large data transmission is composed of a compression service and a decompression service.



Service	Response time
s_{11}	80ms
s_{12}	100ms
s_{21}	80ms
s_{22}	70ms

(c)

Fig. 1. Example of service composition

1.1 QoS-Based Service Composition

QoS of Web services refers to various nonfunctional characteristics such as response time, throughput, availability, and reliability [3, 4]. Given an abstract representation of a composition request, a number of candidate services that provide the same function but differ in QoS can be obtained. For example, in **Fig. 1(b)**, two compression services and two decompression services have been discovered. **Fig. 1(c)** presents the response time of the four services. So there are four different combinations to compose the service, and each generates different QoS.

1.2 Network-Aware Service Composition

With the development of Software as a Service (SaaS) and distributed cloud environment, more and more Web services from different datacenters all over the world are offered. The QoS of a composite service is impacted by the network environment among cloud datacenters, especially the degree of distribution of the services [5]. For example, **Fig. 1(b)** presents the

network delays among users and services. If we only consider the QoS of Web services, s_{11} and s_{22} would be selected because their combined QoS shown in **Fig. 1(c)** are optimal, and the total response time is 150ms. But the fact is that the network delay that is shown beside each communication links in **Fig. 1(b)** would add 940ms to the total response time. The overall time of executing s_{11} and s_{22} is 1090ms. On the other hand, if s_{12} and s_{21} are selected, the overall execution time would only be 820ms, with respect to the fact that the QoS of s_{12} and s_{21} are not optimal. So QoS of network is a noticeable parameter of service composition.

1.3 Contributions

In this paper, we establish a composition model based on graph theory. Our model considers not only the QoS of Web services but also the QoS of network among cloud datacenters. We specify a QoS model that allows us to compute QoS of composite services.

An efficient algorithm is proposed for selecting Web services that meets the user's QoS constraints, followed by a strategy to handle general composition structure such as parallel, conditional and loop.

Simulations show that our model is more realistic. The simulation results also show that our strategy performs better than existing strategy in terms of solution optimality and scalability with respect to different parameters.

The rest of this paper is organized as follows: In Section 2, related work is discussed and summarized. In Section 3, we introduce the problem and describe our model for sequential composition structure. Our algorithm for sequential composition structure is presented in Section 4. Section 5 demonstrates how to apply our algorithm to service composition of general structure. Section 6 shows the simulation results. Finally, Section 7 gives conclusions and an outlook on possible continuations of our work.

2. Related Work

Quality of Services (QoS) is a challenge to cloud computing. In [6], the authors present a QoS management model for mobile cloud computing based on Fuzzy Cognitive Map (FCM). In [7], to continuously support the QoS requirement of an application after data corruption, the authors propose two QoS-aware data replication algorithms in cloud computing systems. QoS of composite Web service has also attracted great interest in the research community. The work in [8] gives details on how to collect networking level criteria of Web service's execution time, availability and reliability. In [9, 10] the authors present approaches to compute and evaluate the QoS of Web services. In [11-13], the QoS-based service management has also been widely discussed. These work focus on QoS management and provide a basis for QoS driven service composition.

Web service composition should not only consider the functional requirements of users, but also take into account the multiple nonfunctional criteria (e.g. response time). Two general models exist for Web service composition: the combinational model defining the problem as a multi-dimension multi-choice knapsack problem (MMKP) and the graph model defining the problem as a multi-constrained optimal path (MCOP) problem [14].

The work in [15] uses combinational model to find the optimal selection of component services. The authors use linear programming technique to solve the problem. But with the increasing scale of problems, the computation amount of this method increases exponentially. The work of Mahammad Alrifai addresses the problem by combining global optimization with local selection methods [16, 17]. By decomposing the optimization problem into small sub-problems, his approach is able to solve the problem in a distributed manner. The work in

[18] extends the methods above. The authors present a strategy to further reduce the search space by examining only subsets of the candidate services since the number of candidate services for a composition may be too large.

The critical defect of combinational model is that the QoS of network cannot be reflected in the model. As the number of Web services increases rapidly, distributing the load to global datacenters will be effective in providing stable services [19, 20]. Major cloud service providers have been deploying and operating geographically dispersive datacenters to serve the globally distributed cloud users. The network performance among datacenters has been attracting more and more attention in the area of service composition.

Some researchers apply graph model to service composition problem. In [21], Jin Xiao et al. investigate the composition of QoS-aware network communication path across large scale multi-domain networks. Adrian Klein et al. [5] propose a network model for service composition in the cloud. The authors estimate the network latency between arbitrary network locations of services or users and propose a genetic algorithm to find services that will result in low latency for certain communication patterns. However, their work only focuses on optimizing composite service's execution time. In real applications, QoS constraints defined in a service level agreement usually involve multiple QoS criteria (such as price, availability). Unlike Adrian Klein's work, we use Simple Additive Weighting (SAW) technique for multiple criteria computation and use Multi-Constrained Optimal Path (MCOP) to model the problem. The work most closely related to our own is [14]. Based on the algorithm of single-source shortest paths in directed acyclic graphs, they propose a heuristic algorithm (MCSP_K) to handle the constraints requirements. However, their work does not give details of how to specify QoS of network. Additionally, their algorithm cannot achieve both optimality and scalability simultaneously. Our work handles above problem, and our model is applicable to geo-distributed cloud datacenters.

3. System Model

Fig. 2 gives a conceptual overview of Web service composition in cloud computing. In cloud computing, Web services are deployed in SaaS layer. There are brokers, which can be centralized or distributed, managing all the information of Web services in SaaS layer. Web services are registered to brokers by providers in order to be discovered. Upon receiving a composition request from the user, the composition engine will first translate the request to a workflow. Then based on the service information and network information, a composition plan will be generated through the process of service selection in accordance with the predefined workflow. At last, the selected Web services will be executed according to the composition plan. Our proposed algorithm of service selection will be described in Section 4. In this section, we focus on modelling sequential composition structure. Other structures (e.g., parallel, conditional and loop) can be transformed to sequential structure which will be introduced in Section 5.

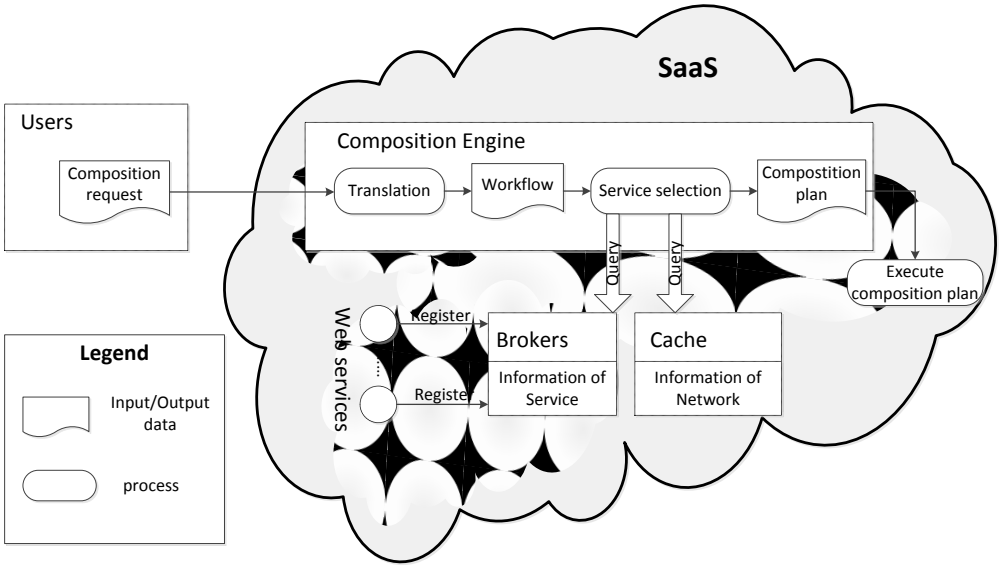


Fig. 2. Overview of Web service composition in cloud computing

3.1 QoS-Based Service

The following definitions are used in this article.

Definition 1: Atomic Service (s). An atomic service is an independent unit to solve a particular task in a service computing system. For example, in **Fig. 1(b)**, s_{11} is an atomic service.

Definition 2: Service Set (S). A service set is a collection of atomic services with the same function but different QoS levels. For example, in **Fig. 1(b)**, $S_1=\{s_{11}, s_{12}\}$ is a service set.

QoS describes the non-functional properties. QoS of atomic services can be provided by providers, computed based on execution and monitored by the users, or collected via users' feedback in terms of the characteristic of each QoS criterion [22]. In this paper, we focus on four typical QoS criteria of atomic services shown in **Table 1**.

Table 1. Typical QoS criteria of atomic services

QoS Criterion	Description	Unit
response time (st)	The execution duration between the moment when a request is arrived and the moment when the result is obtained.	ms
availability (sa)	The probability that a service is accessible.	percent
price (sp)	The money that the requester has to pay to the service provider for the use of service.	dollar
reputation (sr)	A measure of services' trustworthiness.	percent

We convert the atomic services' relationship in **Fig. 1(b)** into a multistage graph shown in **Fig. 3**. In order to simplify the problem, all properties on nodes are migrated to corresponding edges in the graph. The QoS criteria of an atomic service are added to its incoming edges. For example, the QoS criteria marked in **Fig. 3** belong to s_{21}.

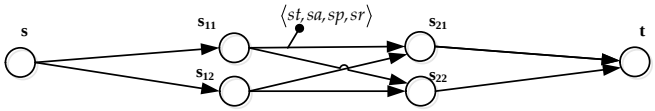


Fig. 3. Atomic services graph

3.2 QoS of Network among Datacenters

We can distinguish the QoS of network into two kinds: QoS of network between services and QoS of network between service and user.

The QoS of network between two atomic services is mainly decided by the network environment between datacenters. For example, if two atomic services are deployed on the same datacenter, then the QoS of network between them is negligible. In contrast, if the network between them is a noticeable parameter, then the QoS of network (such as network delay) must be considered in Web service composition. The network environment among cloud datacenters is measurable and predictable because that the number of datacenters for certain cloud provider is limited and stable. Cloud provider can storage the network parameters among datacenters in cache for facilitating usage.

The QoS of network between service and user is mainly determined by the network environment between them. It can be obtained from the feedback of network and the information of execution monitoring. Many studies deal with point-to-point QoS of network, such as [23][24][25][26], which is not the focus of our paper.

In this paper, we focus on two typical QoS criteria of network shown in **Table 2**.

Table 2. Typical QoS criteria of network

QoS Criterion	Description	unit
network delay (nd)	The transmission duration over the network.	ms
network availability(na)	The probability that the network is accessible.	percent

Therefore, on every edge of a composition graph (except for edges pointing to destination), there should be two kinds of QoS criteria: the QoS of atomic service ($\langle st,sa,sp,sr \rangle$) and the QoS of network ($\langle nd,na \rangle$). We integrate them by the following equations and obtain the QoS $\langle t,a,p,r \rangle$ standing for time, availability, price and reputation in order.

$$t = st + nd \tag{1}$$

$$a = sa + na \tag{2}$$

$$p = sp \tag{3}$$

$$r = sr \tag{4}$$

3.3 Utility Computation of Edge

QoS criteria on edges need to be normalized first for a uniform measurement of multiple criteria independent of units. We use the Simple Additive Weighting (SAW) technique [27], one of the most widely used techniques for multiple dimensions computation. We can distinguish two types of QoS criteria: positive criteria and negative criteria. The increase of values in positive criteria is beneficial for users, such as availability and reputation. The decrease of values in negative criteria is beneficial for users, such as time and price.

The normalization formulations of negative and positive criteria are as eq. (5) and eq. (6), respectively. Where $f_{-}^{x}(e_i)$ and $f_{+}^{x}(e_i)$ are the normalized values of the x-th QoS criterion on edge e_i. e_i is an edge whose end node belongs to S_i and $q^x(e_i)$ represents the x-th QoS criterion on edge e_i. $\max q_i^x$ and $\min q_i^x$ indicate the maximal and minimal value of the x-th QoS criterion on edges whose end nodes belong to S_i, respectively.

$$f_{-}^{x}(e_i) = \begin{cases} \dfrac{\max q_i^x - q^x(e_i)}{\max q_i^x - \min q_i^x}, & \max q_i^x - \min q_i^x \neq 0 \\ 1, & \max q_i^x - \min q_i^x = 0 \end{cases} \tag{5}$$

$$f_{+}^{x}(e_i) = \begin{cases} \dfrac{q^x(e_i) - \min q_i^x}{\max q_i^x - \min q_i^x}, & \max q_i^x - \min q_i^x \neq 0 \\ 1, & \max q_i^x - \min q_i^x = 0 \end{cases} \tag{6}$$

Generally, for users, it is always better to maximize the values of positive criteria and to minimize the values of negative criteria. In addition, users' preference to different criteria should be considered in utility computation. Based on the aforementioned description, the equation of computing utility of an edge is shown in eq. (7). Where α^x reflects the requester's preference ($\sum_{x=1}^{m} \alpha^x = 1$) and the number of QoS criteria is m.

$$U(e) = \sum_{x=1}^{m} \alpha^x \times f^x(e) \tag{7}$$

Based on Section 3.1 to Section 3.3 above, we can model the composition problem as the graph shown in **Fig. 4**.

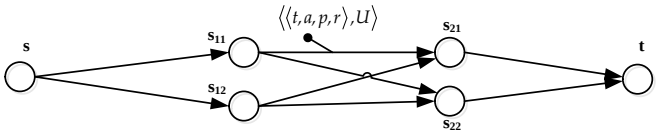


Fig. 4. Composition model graph

3.4 QoS Computation of Composite Service

The QoS of a composite service is up to the QoS and the composition structure. Similar to most work [17, 28, 29], aggregation functions for QoS computation of a sequential composite service are shown in **Table 3**. Where *CS* represents the composite service and e_i represents the component edge of composition path. The number of component edges is n.

Table 3. QoS computation of a sequential composite service

QoS Criterion	response time (t)	availability (a)	price (p)	reputation (r)
Aggregation Function	$t(CS) = \sum_{i=1}^{n} t(e_i)$	$a(CS) = \prod_{i=1}^{n} a(e_i)$	$p(CS) = \sum_{i=1}^{n} p(e_i)$	$r(CS) = \sum_{i=1}^{n} r(e_i) / n$

3.5 Problem Statement

The QoS of a feasible composite service should satisfy the constraints of user's request which are denoted by upper or lower bounds.

The goal of QoS-based and network-aware Web service composition across cloud datacenters is to find one composition path that the overall performance is optimal and user's QoS requirements are satisfied. Therefore, the QoS-based and network-aware service

composition problem can be described as follows: Strategy is needed to find a path with the largest overall utility and meeting the QoS constraints in the model graph.

For a given sequential composition request R= {S_1, S_2 … S_{n-1}} and a given set of QoS constraints C= {C_t, C_a, C_p, C_r}, find a composition path CS= {e_1, e_2 … e_n} such that:

1. The overall utility $U(CS)=\sum_{i=1}^{n}U(e_i)$ is maximized;

2. The aggregated QoS satisfies:

$$t(CS)\le C_t,\ \text{if } C_t\neq \text{null};$$

$$p(CS)\le C_p,\ \text{if } C_p\neq \text{null};$$

$$a(CS)\ge C_a,\ \text{if } C_a\neq \text{null};$$

$$r(CS)\ge C_r,\ \text{if } C_r\neq \text{null}.$$

4. Path Selection for Sequential Structure

4.1 Existing Algorithms

H_MCOP: Korkmaz [30] proposed a heuristic alorithm H_MCOP to solve the path selection problem. H_MCOP aggregates QoS criteria and QoS constraints into a single value based on a non-linear function. Then it adopts the Dijkstria's shortest path algorithm twice to search the path with the minimal function value.

MCSP: In the field of service composition, Tao Yu [14] proposed the MCSP algorithm to handle the constraints requirements. The idea is to topologically sort all nodes in the service candidate graph, then visit nodes in topological order and relax it. In each node, a list of paths from the source to the node meeting QoS constraints is kept.

MCSP_K: In order to and reduce the memory space needed, Tao Yu [14] modifies the MCSP algorithm by keeping only K paths on each node. This algorithm is called MCSP_K. In the new search strategy, paths with K minimum values of relax function are kept at each intermediate node.

4.2 Algorithm Description

We propose an efficient algorithm MCOP_M (Multi-Constrained Optimal Path problem for Multistage graph) based on the branch-and-bound method. Branch-and-Bound is a popular method for handling combinatorial search problems. The basic principle of it is to reduce search space of the problem by pruning unsearched branches which cannot yield better results than already found [31]. MCOP_M attempts to find a feasible path subject to multiple constraints simultaneously, and maximize the utility of the path.

Herein, we use a two-stage strategy for solving the problem. It first exactly finds the largest utility from each node *u* to destination node *t* regardless of the constraints. Then it starts from source *s* and finds composition path meeting the constraints requirements based on the branch-and-bound method. The pseudocode for MCOP_M is shown in **Table 4**. Its inputs are a multistage graph $G = (V, E)$ in which each edge $e(i,j)\in E$ is associated with a utility $U(i,j)$ and QoS criteria $p^k(i,j)$, k=1,2,,...K; a source node s; a destination node t and K constraints c_k, k=1,2,…,K. There are two directions in the algorithm: (a) backward (from t to s) to compute the largest utility of the remaining segment; (b) forward (from s to t) to find the composition

path in terms of constraints and the outcome of backward direction.

Table 4. Pseudocode for MCOP_M

MCOP_M($G = (V, E), s, t, c^k, k=1,2,\ldots K$)
//definitions used in the algorithm:
pt: the current search path;
$pt<m>$: the already traveled segment of current search path from s to m;
$Q^k(pt<m>)$: the k-th accumulated QoS criterion along the already traveled segment of current search path from s to m;
L: a list of nodes being candidates for further processing and partitioning.
1: **For** each node $u \in V$, taken in reverse topological order
2: **Backward_Computation**(u)
3: **End For**
4: **Forward_Selection**($G = (V, E), s, t, c^k$)

4.2.1 Initialization

In the backward direction (lines 1-3 in *MCOP_M*), the algorithm finds the largest utility from each node u to destination t without considering QoS constraints. The pseudocode for *Backward_Computation* is shown in **Table 5.** For each node u, the algorithm maintains the label $o(u)$ to record the outcome of *Backward_Computation*.

Table 5. Pseudocode for Backward_Computing

Backward_Computation(u)
1: $o(u)=0$; //initialization
2: **For** each $v \in$ adj[u]
3: **If**($o(v)+U(u,v)>o(u)$)
4: $o(u)= o(v)+U(u,v)$
5: **End If**
6: **End For**

4.2.2 Branching

In the forward direction (line 4 in *MCOP_M*), the algorithm tries to find an optimal path meeting the constraints requirements based on the branch-and-bound method. The bound function is as eq. (8) (n is the precursor node of u). Where pt is the current search path, $pt<u>$ is the already traveled segment of current search path pt from source s to node u. $B(pt<u>)$ is the sum of the actual utility of path $pt<u>$ and the largest utility that may be generated from node u to destination t without considering QoS constraints. $B(pt<u>)$ represents the largest utility that may be generated if the algorithm proceeds along the current search path from s to u.

$$B(pt<u>) = B(pt<n>) - o(n) + U(n,u) + o(u) \tag{8}$$

The pseudocode for *Forward_Selection* is shown in **Table 6**. The algorithm selects the node u with the largest bound function value from list L to be branch node and then transverses all its successor nodes (If there is more than one node with the largest bound function value, the algorithm will choose one randomly). The node that has been selected is deleted from list L. If the QoS of current path meets QoS constraints, then calculate the value of bound function and add current path to list L. The process above is repeated until the selected node from list L is

the destination node.

Table 6. Pseudocode for Forward_Selection

```
Forward_ Selection(G = (V, E),s, t, c^k)
1:  pt<s> = <s> ;            //initialization
2:  B(pt<s>)=o(s);
3:  add <s, pt<s>, B(pt<s>)> to L
4:  While(true)
5:      u← Find node with the largest B value from L
6:      If(u==t)  then
7:          return pt<t> associated with the largest B value
8:      Else
9:          delete <u, pt<u>,B(pt<u>)> from L
10:         For each node m∈adj[u]
11:             compute values of Q^k(pt<m>), k=1,2,….,K
12:             If (Q^k(pt<m>) meets the constraint of c^k ),k=1,2,…,K
13:                 then B(pt<m>)= B(pt<u>) – o(u)+ U(u,m) + o(m)
14:                         add <m, pt<m>, B(pt<m>)> to L
15:             End If
16:         End For
17:     End If
18:     If ( L is null )
19:         return failure
20:     End If
21:  End While
```

5. Approach for Service Composition of General Structure

In real applications, the atomic services' composition order may not be sequential. Other general structures include parallel, conditional and loop operations. In order to simplify the problem we convert general composition structure to sequential structure. Some related works have proposed approaches to the simplification process [14, 32]. Based on the idea of related work, we propose new conversion methods. In the new conversion methods, we take use of virtual nodes and virtual edges, which makes the converted structure applicable to our algorithm MCOP_M. We illustrate the process of QoS computation for parallel, conditional, and loop structure in detail. The conversion is achieved step-by-step. At each step, the nonsequential part in the graph is replaced by a virtual atomic service and sequential edges. The QoS on the edges is aggregated from the QoS on the original edges. This process continues until no further nonsequential part exists [32]. Then a sequential representation of the originally structure is obtained.

5.1 Reduction of Parallel Structure

In parallel structure, multiple atomic services are executed concurrently and merged synchronization. Aggregation functions for the QoS computation of a composite service with parallel structure are illustrated in **Table 7**.

Table 7. QoS computation of a parallel composite service

QoS Criteria	response time (t)	availability (a)	price (p)	reputation (r)
Aggregation Function	$t(CS) = \max_{i=1,\ldots,n} t(e_i)$	$a(CS) = \prod_{i=1}^{n} a(e_i)$	$p(CS) = \sum_{i=1}^{n} p(e_i)$	$r(CS) = \sum_{i=1}^{n} r(e_i) \Big/ n$

In the typical service composition case that involves both sequential and parallel structures as illustrated in **Fig. 5(a)**, we assume that the atomic services gained through the process of

service discovery are as **Fig. 5(b)**. The complex structure of this graph can be transformed into an equivalent sequential structure. Take the parallel path (s_{11}, s_{21}&(s_{31},s_{42}), s_{51}) for example, we assume that the QoS of atomic services have been added to services' incoming edge and integrated with the QoS of network. The process of converting this path shown in **Fig. 6** is as the following three steps:

- Step1: For each sequential branch ((s_{11},s_{21},s_{51}), (s_{11},s_{31},s_{42},s_{51})), calculate the QoS using the equations in **Table 3**.
- Step 2: Based on the above outcome, calculate the QoS ($\langle t,a,p,r \rangle$) of the parallel path using the equations in **Table 7**.
- Step 3: Replace the nodes s_{21}, s_{31} and s_{42} by a virtual node and convert the path to a sequential path.

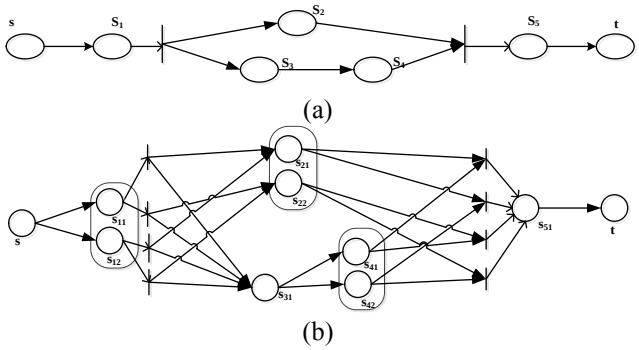


Fig. 5. Typical parallel service composition

We can convert **Fig. 5(b)** to a multistage graph shown in **Fig. 7** through applying the above steps to other parallel paths. Then the parallel structure is converted to sequential structure. We can calculate the utility of every edge using eq. (7) and then apply algorithm MCOP_M to the converted graph.

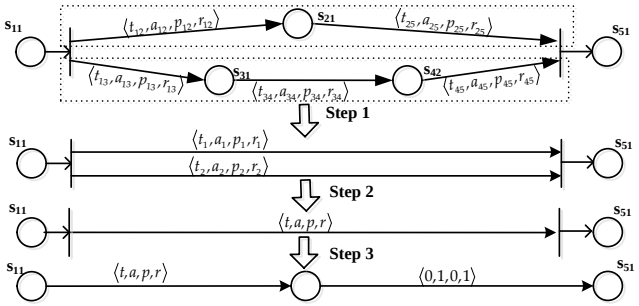


Fig. 6. Reduction of parallel path

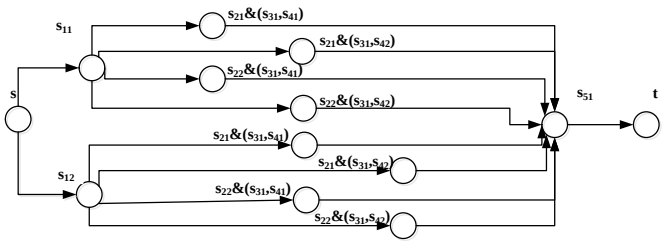


Fig. 7. Converted graph of parallel structure

5.2 Reduction of Conditional Structure

The process of handling conditional structure is similar to that of handling parallel structure except for the calculation of QoS. In a typical conditional service composition path shown in **Fig. 8**, the probability that s_{11} is followed by s_{21} or s_{31} is y_1 and y_2 respectively.

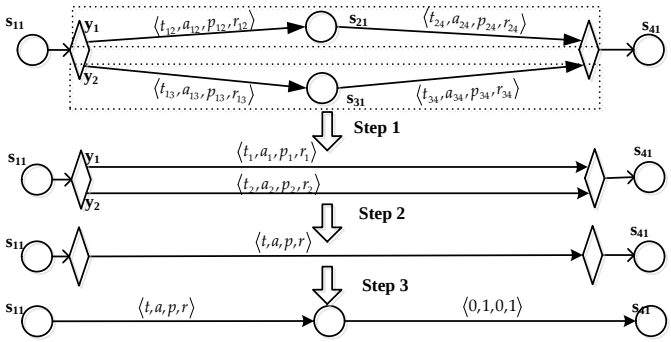


Fig. 8. Reduction of conditional path

We can obtain that the QoS for sequential path (s_{11}, s_{21}, s_{41}) is $\langle t_1, a_1, p_1, r_1 \rangle$, and for sequential path (s_{11},s_{31},s_{41}) is $\langle t_2, a_2, p_2, r_2 \rangle$ based on the equations in **Table 3**. Then $\langle t, a, p, r \rangle$ of the overall conditional path is calculated as **Table 8**.

Table 8. QoS computation of a conditional composite service

QoS Criteria	response time (t)	availability (a)	price (p)	reputation (r)
Aggregation Function	$t = t_1 \times y_1 + t_2 \times y_2$	$p = p_1 \times y_1 + p_2 \times y$	$r = r_1 \times y_1 + r_2 \times y_2$	$U = U_1 \times y_1 + U_2 \times y_2$

5.3 Reduction of Loop Structure

In loop structure, one or more atomic services are executed repeatedly up to a maximum number of k times. In a simple loop composition path shown in **Fig. 9**, the probability that s_{21} is followed by itself is y_1. We assume the QoS of atomic services have been added to services' incoming edge and integrated with the QoS of network. Note that $\langle t_{22}, a_{22}, p_{22}, r_{22} \rangle$ is equal to the QoS of Web service s_{21} ($\langle st_{21}, sa_{21}, sp_{21}, sr_{21} \rangle$) because there is no network cost.

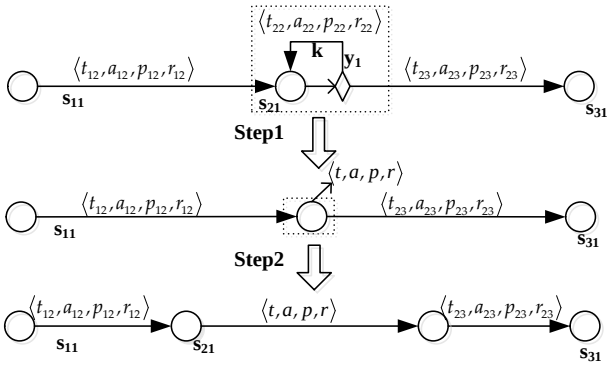


Fig. 9. Reduction of loop path

In the first step we calculate the QoS of loop structure. Aggregation functions for QoS computation of loop structure are based on the statistical scenario where the number of iterations is $k \times y_1$. To compute the QoS of loop structure $\langle t, a, p, r \rangle$, the following formulae are applied:

Table 9. QoS computation of loop structure

QoS Criteria	response time (t)	availability (a)	price (p)	reputation (r)
Aggregation Function	$t = t_{22} \times k \times y_1$	$a=(a_{22})^{k \times y1}$	$p= p_{22} \times k \times y_1$	$r=r_{22}$

In the second step we replace the loop path by a virtual atomic service. The above QoS is added to the edge from s_{21} to the virtual service. The loop then is removed by unfolding the cycles.

6. Simulation

6.1 Simulation Setup

In our simulations, we assumed that all Web services are deployed on eight geo-distributed datacenters. Values of network delay between two datacenters were randomly generated within the interval [2, 800] and values of network availability were within the range of [0.98, 1] so that a good range of realistic values will be covered. We associated QoS of network with each edge after generating graphs. QoS of atomic services were generated randomly. The random values of response time in each service set had a Gaussian distribution and a mean value within the range [20, 1500]. The ranges of atomic services' availability, price and reputation were [0.95, 1], [1, 10] and [0.4, 1] in order. For simulation, the four QoS criteria mentioned above gained equal preference from users. Our simulation study included two parts: (a) study of the effect of network QoS where we run MCOP_M and MCSP_K based on two models considering network QoS and regardless of network QoS, respectively and (b) comparison of our algorithm MCOP_M and other algorithm MCSP_K [4] where we used solution utility and running time as metrics.

6.2 Model Comparison

We created twelve composition instances shown in **Table 10**. The numbers of service sets, atomic services per set were given by parameters m and n. we measured the solution utility and response time by running MCOP_M and MCSP_K based on two composition models. The differences between the two models were that one considered QoS of network and the other neglected QoS of network.

Table 10. Simulation instances

No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
m	5	5	5	10	10	10	15	15	15	20	20	20
n	10	20	30	10	20	30	10	20	30	10	20	30

Fig. 10 shows the result of this simulation. Both MCOP_M and MCSP_K find composite services with lower response time under the model with QoS of network. This is because network delay is considered in the former model. Note that solution utility under the model considering QoS of network is higher than that under the model without QoS of network. A significant utility gain of both algorithm MCOP_M and MCSP_K was achieved when QoS of network was considered in composition model, which demonstrated that our proposed model was more applicable.

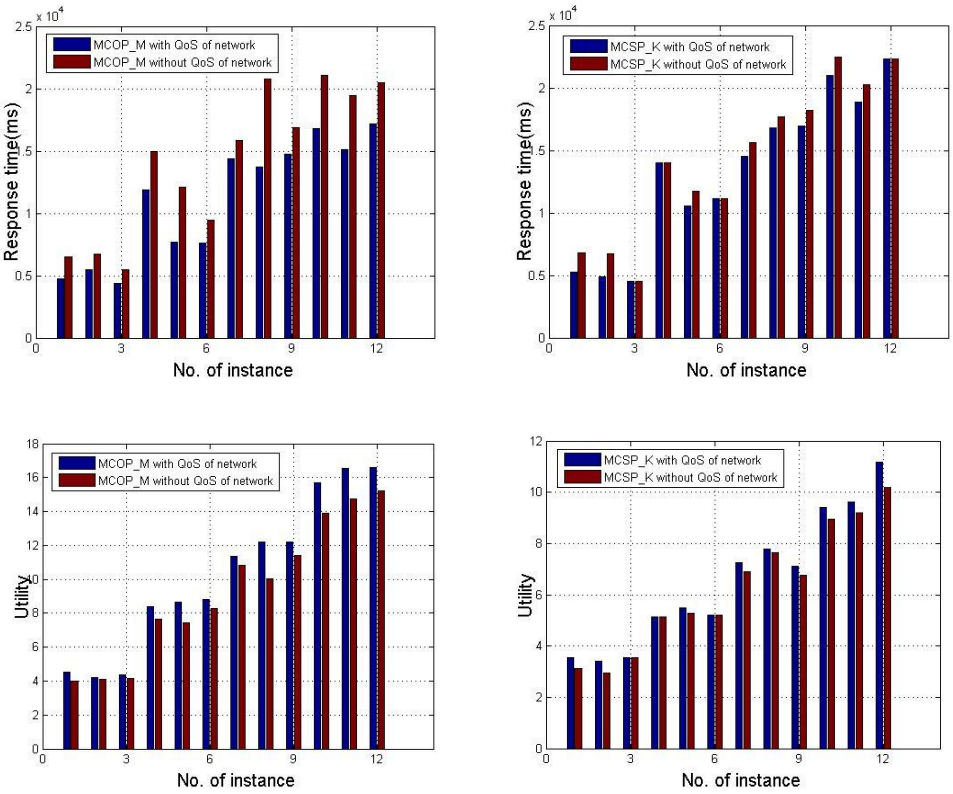


Fig. 10. Model comparison

6.3 Optimality and scalability Comparison

To evaluate the optimality and scalability of our algorithm we compare the solution utility of MCOP_M and MCSP_K versus an increasing problem size.

6.3.1 Comparison for sequential structure

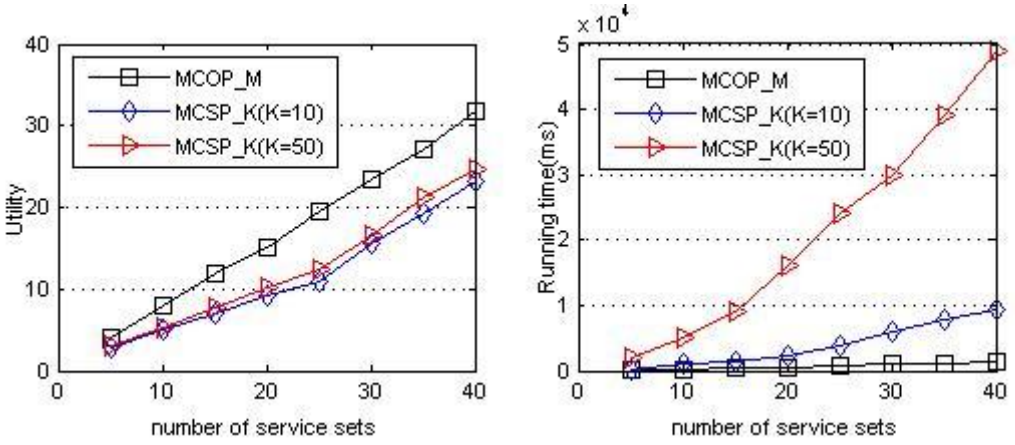
Fig. 11(a) presents the solution utility against an increasing number of service sets. We assigned a fix number 30 to the number of atomic services per set. The number of service sets ranged from 5 to 40. We evaluated the scalability of our approach under the same settings as for solution utility. **Fig. 11(b)** shows the outcome of scalability evaluation. The following findings are revealed.

1. The utility and running time of both algorithms increased with the increasing number of service sets. The solution utility is the sum of the utilities of component edges and the number of component edges is one less than the number of service sets for sequential structure, which explains the result shown in **Fig. 11(a)**. **Fig. 11(b)** shows that the running time increased with the number of service sets increasing, as there are more composition paths to evaluate. But the required running time of MCOP_M increased very slowly compared with MCSP_K.

2. MCSP_K (K=50) found higher solution utility than MCSP_K (K=10). However, the running time of MCSP_K (K=50) was much longer than that of MCSP_K (K=10). Therefore, it seems that MCSP_K with lower value of K runs faster because it fails to improve the solution utility.

3. The solution utilities of MCOP_M were higher than those of both MCSP_K. MCOP_M was able to gain better solution that MCSP_K in all cases. The results also showed that the

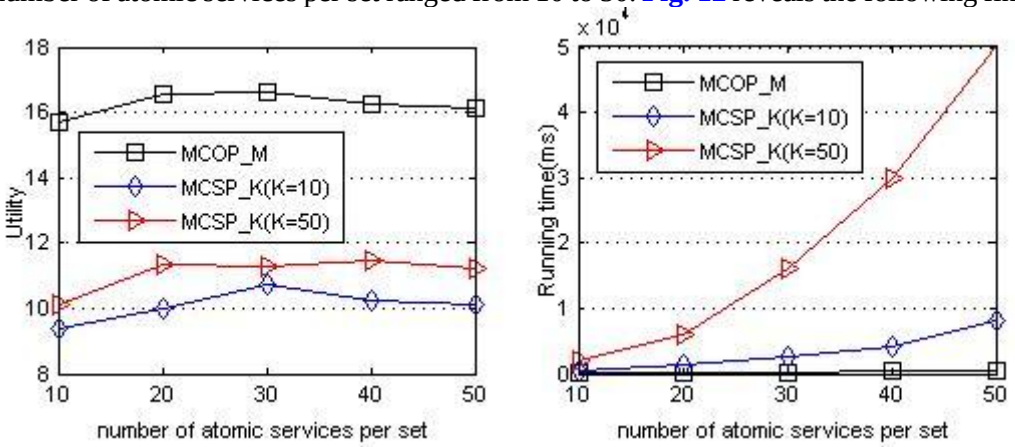
running time of MCOP_M is much shorter than that of MCSP_K. MCOP_M can produce better solutions within shorter time for large composite services.



(a) (b)

Fig. 11. Optimality and scalability vs number of service sets

Fig. 12 presents the solution utility and running time against an increasing number of atomic services per service set. We assigned a fix number 20 to the number of service sets. The number of atomic services per set ranged from 10 to 50. **Fig. 12** reveals the following findings.



(a) (b)

Fig. 12. Optimality and scalability vs atomic services per set

1. The running time of both algorithms increased with the increasing number of atomic services per service set, as there were more choices available.

2. The performance of MCSP_K was influenced by the value of K significantly.

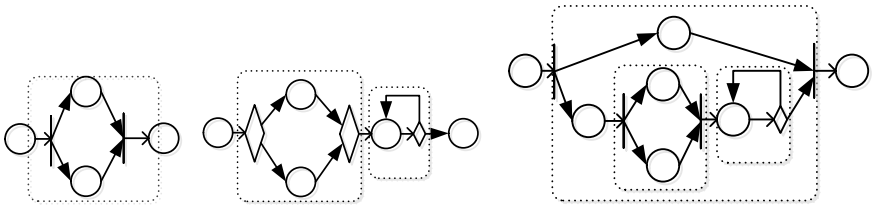
3. The solution utilities of MCOP_M were higher than that of both MCSP_K. In addition, the running time of MCOP_M was significantly lower. By increasing the number of atomic services per set, the required computation time of MCOP_M increased very slowly compared with MCSP_K, which made our solution more scalable.

The reason for above findings is the difference in the design of the two algorithms. MCSP_K traverses all successor nodes of all nodes, which leads the running time increases explosively with the problem scale increasing. However, MCOP_M only traverses successor nodes of the node that may result in the highest utility, which makes MCOP_M processes

problem faster than MCSP_K. On the other hand, MCSP_K gains worse solutions than MCOP_M because that it maintains limited feasible paths on every node.

6.4.2 Comparison for general structure

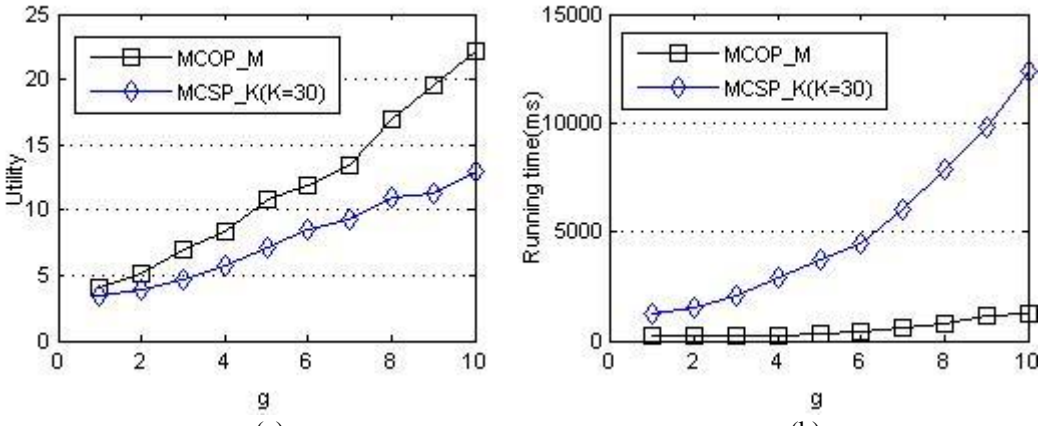
We evaluate the optimality and scalability of our approach against the complexity of composition structure, given by the parameter g. Complexity of composition structure is expressed by the number of the three kinds of structure described in Section 5. We give an example illustrated in **Fig. 13**. In the example shown in **Fig. 13(a)**, there is only one parallel structure, so g equals 1. The composition structure in **Fig. 13(b)** is composed of one conditional component and one loop component, so g equals 2. In **Fig. 13(c)**, there are two parallel components and one loop component, therefore g equals 3. We randomly generated 10 composition structures with g ranged from 1 to 10. We assigned a fix number 20 to the number of atomic services per service set.



(a) (b) (c)

Fig. 13. Example for complexity of composition structure

Fig. 14 presents the solution utility and running time of MCSP_K and MCOP_M. The results of this simulation indicated that the performance of both approaches degraded as the complexity of composition structure increased. The running time increased as the complexity of composition structure increased, which was as expected. However, MCOP_M still outperformed the MCSP_K. Our approach requires less running time because it transforms the complex structure into sequential structure before applying MCOP_M to the problem. MCSP_K requires more time to check the path feasibility because of the complexity of structure.



(a) (b)

Fig. 14. Optimality and scalability vs complexity of composition structure

5. Conclusion

In this paper, we develop a graph model for QoS-based and network-aware Web service composition across cloud datacenters. QoS of Web services and QoS of network are both considered in this model. We model the composition problem to find a composition path that the overall performance is optimized and user's end-to-end QoS constraints are always satisfied. Simulation results show that our model reflects the real-world situations better. We present an algorithm to solve the composition problem, which achieves a better result at the cost of lower running time compared with other strategy. The strategy to handle general composition structure such as parallel and conditional is also proposed. In the current approach, service exceptions and incompatibility during composite service execution are not considered. In our future work, we aim at developing a runtime recovery strategy. This will make our approach more practical and effective.

References

[1] M. Bichler and K.J. Lin, "Service-Oriented computing," *IEEE Computer*, vol. 39, no. 3, pp. 99–101, 2006. Article (CrossRef Link)

[2] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, "Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web services," *IEEE Transactions on Services Computing*, vol. 3, no. 3, pp. 223-235, 2010. Article (CrossRef Link)

[3] D. Menasce, "Composing Web services: a QoS view," *IEEE Internet Computing*, vol. 6, no. 8, pp. 88-90, Dec. 2004. Article (CrossRef Link)

[4] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Qos-aware Web service recommendation by collaborative filtering," *IEEE Transactions on Service Computing*, vol. 4, no. 2, pp. 140-152, 2011. Article (CrossRef Link)

[5] A. Klein, F. Ishikawa, and S. Honiden, "Towards Network-aware Service Composition in the Cloud," in *Proc. of 21th international conference on World Wide Web*, pp. 959-968, 2012. Article (CrossRef Link)

[6] P. Zhang, Z. Yan, "A QoS-aware system for mobile cloud computing," in *Proc. of IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, pp. 518-522, 2011. Article (CrossRef Link)

[7] J. Lin, C. Chen, J. Chang, "QoS-aware data replication for data intensive applications in cloud computing systems," *IEEE Transactions on Cloud Computing*, vol. 1, no.1, pp. 101-115, 2013. Article (CrossRef Link)

[8] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng, "Quality driven Web services composition," in *Proc. of 12th International Conference on World Wide Web*, Budapest, Hungary, pp. 411-421, 2003. Article (CrossRef Link)

[9] Z. Zheng, Y. Zhang, and M. R. Lyu, "Distributed QoS Evaluation for Real-World Web services," in *Proc. of the International Conference on Web services (ICWS)*, pp. 83-90, 2010. Article (CrossRef Link)

[10] C. Shi, D. Lin, and T. Ishida. "User-Centerd QoS Computation for Web service Selection," in *Proc. of the International Conference on Web services (ICWS)*, pp. 456-463, 2012. Article (CrossRef Link)

[11] F. Li, F. Yang, K. Shuang, S. Su, "Q-Peer: A Decentralized QoS Registry Architecture for Web services," in *Proc. of the International Conference on Service-Oriented Computing* (ICSOC), Springer Berlin Heidelberg, pp. 145-156. 2007. Article (CrossRef Link)

[12] V. Agarwal and P. Jalote, "From Specification to Adaptation: An Integrated QoS-driven Approach for Dynamic Adaptation of Web service Compositions," in *Proc. of the International Conference on Web services (ICWS)*, pp. 275-282, 2010. Article (CrossRef Link)

[13] R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli, "Dynamic qos

management and optimisation in service-based systems," *IEEE Transactions on Software Engineering*. vol. 37, no. 3, pp. 387-409, 2011. Article (CrossRef Link)
[14] T. Yu, Y. Zhang, K. J. Lin, "Efficient algorithms for Web services selection with end-to-end qos constraints," *ACM Transactions on Web*, vol. 1, no. 1, pp. 1-25, 2007. Article (CrossRef Link)
[15] L. Qi, Y. Tang, W. Dou, and J. Chen, "Combining local optimization and enumeration for QoS-aware Web service composition," in *Proc. of the International Conference on Web services (ICWS)*, pp. 34-41, 2010. Article (CrossRef Link)
[16] M. Alrifai, T. Risse, P. Dolog, and W. Nejdl, "A scalable approach for qos-based Web service selection," in *Proc. of Service-Oriented Computing–ICSOC 2008 Workshops*, Springer Berlin Heidelberg, pp. 190-199, 2009. Article (CrossRef Link)
[17] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient qos-aware service composition," in *Proc. of the 18th International Conference on World Wide Web*, pp. 881-890, 2009. Article (CrossRef Link)
[18] M. Alrifai, D. Skoutas, and T. Risse, "Selecting skyline services for qos-based Web service composition," in *Proc. of the 19th International Conference on World Wide Web*, ACM Press, pp. 11–20, 2010. Article (CrossRef Link)
[19] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: automated data placement for geo-distributed cloud services," *NSDI*, pp. 17-32, 2010.
[20] S. Son, G. Jung, and S. C. Jun, "An SLA-based cloud computing that facilitates resource allocation in the distributed data centers of a cloud provider," *The Journal of Supercomputing*, vol. 64, no. 2, pp. 606-637, 2013. Article (CrossRef Link)
[21] J. Xiao and R. Boutaba, "QoS-aware service composition in large scale multi-domain networks," in *Proc. of IFIP/IEEE 9th International Symposium on Integrated Network Management (IM 2005)*, pp. 397-410, 2005. Article (CrossRef Link)
[22] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnamam, and H. Chang, "QoS-Aware middleware for Web services composition," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 311-327, 2004. Article (CrossRef Link)
[23] P. Yalagandula, S. J. Lee, et al., "Correlations in end-to-end network metrics: impact on large scale network monitoring," in *Proc. of Global Internet*, pp. 1-6, 2008. Article (CrossRef Link)
[24] P. Sharma, Z. Xu, S. Banerjee, and S. Lee, "Estimating network proximity and latency," *ACM SIGCOMM CCR*, vol. 36, no. 3, pp. 39–50, 2006. Article (CrossRef Link)
[25] F. Thouin, M. Coates, and M. Rabbat, "Large scale probabilistic available bandwidth estimation," *Computer Networks*, vol. 55, no. 9, pp. 2065-2078, 2011.
[26] J. Jin, J. Liang, et al, "Large-scale qos-aware service-oriented networking with a clustering-based approach," in *Proc. of 16th International Conference on Computer Communications and Networks*, pp. 522-528, 2007. Article (CrossRef Link)
[27] K. P. Yoon and C. L. Hwang, "Multiple attribute decision making: an introduction," *u: Paper Series: Quantitative Applications in the Social Sciences*, 1995.
[28] J. El Hadad, M. Manouvrier and M. Rukoz, "TQoS: Transactional and QoS-aware selection algorithm for automatic Web service composition," *IEEE Transactions on Service Computing*, vol. 3, no. 1, pp.73-85, 2010. Article (CrossRef Link)
[29] F. Tao, Y. J. Laili, L. D. Xu, and L. Zhang, "FC-PACO-RM: Aparallel method for service composition optimal-selection in cloud manufacturing system," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2023-2033, Dec. 2013. Article (CrossRef Link)
[30] T. Korkmaz, M. Krunz, "Multi-constrained optimal path selection," in *Proc. of INFOCOM 2001, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, pp. 834-843, 2001. Article (CrossRef Link)
[31] N. Cherfi, M. Hifi, "A column generation method for the multiple-choice multi-dimensional knapsack problem," *Computational Optimization and Applications*, vol. 46, no. 1, pp. 51-73, 2010. Article (CrossRef Link)
[32] M. Alrifai, T. Risse, W. Nejdl, "A hybrid approach for efficient Web service composition with end-to-end QoS constraints," *ACM Transactions on the Web (TWEB)*, vol. 6, no. 2, article. 7, 2012. Article (CrossRef Link)

Dandan Wang received the B.S degree in 2012 and is working toward the Ph.D degree at the School of Computer and Communication Engineering at University of Science and Technology Beijing, China. Her main research interests include service-oriented architectures, service composition, and scheduling of distributed resources in the cloud.

Yang Yang received B.S. in aotomation from the Department of Automation, Beijing Institute of Iron and Stell, in 1982. Received his Ph.D. in information Engineering, from University of Science and Technology in Lillie, France, in 1988. He has been a professor of University of Science and Technology Beijing since 1988. From 1994, he has been the senior member of many national and provincial technique committees. Prof. Yang Yang's research interests include service science and cloud computing, intelligent control, image processing and pattern recognition, multimedia communication, grid technology. He has co-authored more than 200 refereed journal and conference papers, and several books.

Zhenqiang Mi received B.S. in aotomation and Ph.D. in communication engineering, both from School of Information Engineering, University of Science and Technology Beijing, in year 2006 and 2011, respectively. From 2011, he is assistant professor with the school of computer and communication engineering, University of Science and Technology Beijing. Prof. Mi is IEEE member, and serves as a frequent reviewer in several international journals and TPC member in several international conferences. Prof. Mi has co-authored two books, and more than 20 research papers in international journals and conferences. His research interest includes service computing, multi-robot systems, connectivity in mobile ad hoc networks and cloud computing in mobile environments.