Service Composition in Cyber-Physical-Social Systems

S. Wang, *Senior Member*, *IEEE*, A. Zhou, M. Yang, L. Sun, C. Hsu*, *Senior Member*, *IEEE*, F. Yang, *Senior Member*, *IEEE*

Abstract—Cyber-physical-social systems comprise physical, cyber, and social worlds with various resources as services. The operation and configuration of these services require service composition approaches that can be used to integrate essential components in these worlds, and organize and share relevant information as needed. Although the present service-oriented architecture can effectively support service composition, there still exist many challenges such as high time cost and low reliability for cyber-physical-social systems. In this paper, we propose a fast and reliable service composition approach to integrate the physical network, cyberspace, and social network. In this approach, first, skyline component computation is performed to reduce the solution space; then, the coefficient of variation is employed to filter the components with high quality of service (QoS) fluctuation and finally the best optimal components are selected by maximizing the fitness function based on users' end-to-end QoS requirements. We applied our approach to real-world dataset and the results showed that our proposed approach has better reliability as well as lower time cost than other approaches.

Index Terms—cyber-physical-social system; service composition; component; QoS; coefficient of variation

1. Introduction

Cyber-Physical-Social System (CPSS) is a new research topic in which physical, cyber, and social worlds (related to human factors) are integrated based on the interactions between these worlds in real time[1]. CPSSs rely on communication, computation, and control infrastructures that commonly consist of several levels of the three worlds with various resources as services. The operation and configuration of CPSSs require approaches that can be used for managing the variability at design time and the dynamics at runtime, which are caused by a multitude of component types and changing application environments[2]. Service composition approach is one of the most important among these approaches, and it plays an important role in integrating essential components in these worlds and also in organizing and sharing relevant information as required, such as virtual reality 3D glasses.

It is well known that a CPSS performs parallel execution and self-synchronization, and also influences the physical, information, cognitive, and social domains[1]; this provides a new way to transform command and control organizations by the fusion of CPSS internal essential components from the three worlds. In service-oriented architecture (SOA), services correspond to abstract components realized by an API interface[3,4] and are platform independent, self-contained, and programmable. The true capacity of SOA can only be achieved when multiple services are integrated into more capable and powerful applications[5] for achieving cross call and sharing of resources under different domains and organizations. Service composition allows domains to form alliances, outsource functionalities, and perform self-synchronization. From the perspective of a CPSS software developer, service composition dramatically reduces the cost and risks of building new CPSS applications such that existing business logics can be represented as components and be reused. Thus, service composition approaches provide an ideal paradigm for designing and constructing command and control organizations for CPSSs.

Service composition is a process in which existing components are dynamically selected, integrated, and invoked according to certain requirements. It provides a value-added service as it comprises varied components[6]. According to the SOA paradigm, service composition is a combination of a series of services (also known as abstract components) and at run time, for each service, concrete components are integrated and invoked[7]. In service composition, besides considering the component functions to match CPSS users' request, the quality of service (QoS), which is an aggregated indicator of component quality, should also satisfy CPSS users' requirements. The QoS attributes of every component (e.g., response time, throughput, reliability,

- *S. Wang, A. Zhou, M. Yang, L. Sun and F. Yang are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. E-mail: {sgwang; aozhou; mingzheyang; leisun; fcyang@bupt.edu.cn}.*
- *C. Hsu is with the Department of Computer Science and Information Engineering, Chung Hua University, Hsinch, Taiwan. E-mail: chh@chu.edu.tw*
  ** The corresponding author*

and availability) are very important factors because they can influence the performance and stability of CPSSs. Hence, effectively selecting the best services and integrating them is critical for CPSSs.

Although plenty of service composition approaches such as MIP[8], Heuristic[9], Hybrid[10] and LOEM[11] have excellent performance in Web service environments, there are two challenges for the service composition of CPSSs, which are as follows.

- **Violent QoS fluctuation**: Traditional approaches pay little attention to the violent QoS fluctuations of components such as the response time of changes in the components over a period of time; hence, these approaches cannot provide reliable components to the users since these components are derived from the dynamic and different domains in CPSSs. Although QoS fluctuation also reduces the reliability of traditional service composition schemes, we believe the decrease in reliability of the service composition in CPSSs is higher than that of traditional schemes due to the expansion of service domains. Generally, components may be dispersed in different domains, may be obtained from different organizations or networks, or may operate on different platforms. Slight changes, such as those in the location, network environment, requirement time, will affect the QoS consistency of components and the reliability of service composition. For example, a component with continuous and high quality QoS attributes history has a better reliability than a component with high violent QoS attribute history. Hence, because of the dynamic nature, diversity, and cross-domain of CPSSs, there are inherent uncertainties of the component QoS and it leads to a large degree of error between running results and the actual results, which may lead to service composition failure. Therefore, it is worth noting that a component with high QoS consistency is typically more reliable than a component with a large coefficient of variance in terms of its QoS.
- **High time cost**: Service composition in CPSSs has a higher time cost than traditional schemes. With the development of CPSSs, more components with the same functions but significant differences in terms of the QoS attributes will be published in the three worlds. Similar to traditional service composition, for the service composition in CPSSs, the obvious way to obtain the best result is enumerating all possible compositions, which will lead to an NP-hard problem[9]. Therefore, selecting components of each service based on the QoS attributes for meeting user demands will result in huge time cost for service composition. For traditional service composition schemes, reducing the time cost is a research issue as well; however, most traditional services are Web services and their numbers are limited for supporting service composition. When the number of components is very low, the time cost is acceptable. However, with the rapid increase in the number of components in CPSSs, it becomes unrealistic to enumerate all possible components, which will lead to high time cost. Hence, to avoid high computation time, reducing the search space of components reliably is an important principle for service composition in CPSSs.

Note that traditional service composition schemes also consider the time cost and reliability but these two issues are not very important: 1) the number of Web services with composition ability is few as IT companies compete with each other; 2) the QoS fluctuation is not drastic due to most services are from one world. However, different from traditional schemes, the time cost and reliability of service composition have become important issues in CPSSs due to the characteristic (massive services are from different world) of CPSSs. Hence, in this paper, we overcome the above two issues and propose an efficient service composition approach by Skyline service computation and QoS fluctuation computation. The contributions of this study are as follows:

1) Unlike traditional approaches, our research not only focuses on reducing the time cost of service composition further but also on guaranteeing the reliability of CPSSs. To the best of our knowledge, this is the first study to consider the service composition problem in CPSSs.
2) We propose a fast and reliable service composition approach in CPSSs, where Skyline service computation is first carried out to reduce the solution space; then, coefficient of variation is employed to filter the components that have high QoS fluctuation and finally the best and reliable components are selected by maximizing the fitness function.
3) We implement our approach and compare the result with four other approaches based on real-world dataset. The experiment results show that our approach performs better service composition for CPSSs.

The remainder of the paper is organized as follows. Section 2 introduces the problem definition and related work. Section 3 describes our approach in detail including Skyline component computation, QoS

fluctuation computation, and service composition algorithm. Section 4 evaluates the benefits of our approach based on the experiment results. Finally, Section 5 concludes the paper and discusses future work.

2. Problem Definition and Related Work

2.1 Problem Definition

Definition 1 (Service Composition). Let *S* denote a service composition in a CPSS that consists of *n* services with different functions; that is, $S = \{ s_1 , s_2 , \dots , s_n \}$, where $s_i \ (0 < i \le n)$ denotes the *i*-th service. Then, service s_i contains many components, that is, $s_i = \{ s_{i1}, s_{i2}, ..., s_{il} \}$, where $l \ (l > 1)$ denotes the number of components. Hence, service composition involves the selection of components from each service and assembling a new application through workflows[12] with QoS constraints for CPSSs. Similar to other studies [8][9][10][11][14,35-37,39], our study also focuses on sequential service composition.

The QoS constraints of a component play a significant role in service composition because QoS expresses the numerical reference point of a component and affects the performance of the component in CPSSs. For example, if component s_{ij} possesses *r* types of QoS attributes, the QoS attribute vector can be expressed as $Qs_{ij} = \{ q_1(s_{ij}) , q_2(s_{ij}) , \dots , q_r(s_{ij}) \}$, where $q_k(s_{ij}) \ (1 \le k \le r)$ denotes the *k*-th attribute value of component s_{ij}. Similarly, the QoS attribute vector of service composition *S* can be expressed as $QS = \{ q_1(S) , q_2(S) , \dots , q_r(S) \}$, where $q_k(S) \ (1 \le k \le r)$ denotes the *k*-th attribute value of service composition *S,* which is an aggregation of the *k*-th attribute values from all selected components using QoS aggregation functions. For example, the aggregation functions of response time and reliability attribute are as follows, respectively (more functions can be found in [13-16]):

$$q(S) = \sum_{i=1}^{n} q(s_i) \tag{1}$$

$$q(S) = \prod_{i=1}^{n} q(s_i) \tag{2}$$

Generally, in CPSSs, a component has a number of different types of QoS attributes having a different concrete unit and scope; this causes difficulties and leads to obstacles in service composition in CPSSs.

Definition 2 (QoS Utility Function). Therefore, we need to utilize scope of multiple QoS attributes by using QoS utility function [9,10,13-16]; we can then define the function of component $s_{ij} \in s_i$ and service composition *S* in CPSSs as follows:

$$U(s_{ij}) = \sum_{k=1}^{r} \frac{Q_{i,k}^{max} - q_k(s_{ij})}{Q_{i,k}^{max} - Q_{i,k}^{min}} \cdot \omega_k \tag{3}$$

$$U(S) = \sum_{k=1}^{r} \frac{Q_k^{max} - q_k(S)}{Q_k^{max} - Q_k^{min}} \cdot \omega_k \tag{4}$$

with

$$\begin{cases} Q_k^{max} = \sum_{i=1}^{r} Q_{i,k}^{max} \ (Q_{i,k}^{max} = \max\limits_{\forall s_{ij} \in s_i} q_k(s_{ij})) \\ Q_k^{min} = \sum_{i=1}^{r} Q_{i,k}^{min} \ (Q_{i,k}^{min} = \min\limits_{\forall s_{ij} \in s_i} q_k(s_{ij})) \end{cases} \tag{5}$$

where ω_k is the weight of each QoS attribute and satisfies $\sum_{k=1}^{r} \omega_k = 1 (0 < \omega_k < 1)$. $Q_{i,k}^{max} (0 < k \le r)$ is the maximum value of the *k*-th attribute in all components of service s_i, and similarly, $Q_{i,k}^{\min}$ is the minimum value in service s_i; Q_k^{max} is the summation of all $Q_{i,k}^{max}$ in service composition *S.* Similarly, $Q_k^{\min}$ is the summation of all $Q_{i,k}^{\min}$ in service composition *S* in CPSSs.

Definition 3 (QoS Constraints). During service composition, QoS constraints must be considered for satisfying users' requirements such as the total response time ≤3 s. Then, QoS constraints can be definite as a vector $C = \{C_1 , C_2 , ..., C_r\}$, where every QoS constraint $C_i \ (0 < i \le \mathrm{r})$ represents the maximum or minimum boundary of an aggregated QoS attribute.

Definition 4 (Reliability). Reliability is the deviation degree of the variance of integrated components with the maximum and minimum variance in terms of service composition by

$$rel(S) = \sum_{k=1}^{r} \frac{V_k^{max}(S) - \sum_{i=1}^{n} v_k(s_i)}{V_k^{max}(S) - V_k^{min}(S)} \cdot \omega_k \times 100\% \tag{6}$$

with

$$\begin{cases} V_k^{max}(S) = \sum_{i=1}^{n} V_{i,k}^{max}(s_i) \\ V_k^{min}(S) = \sum_{i=1}^{n} V_{i,k}^{min}(s_i) \end{cases} \tag{7}$$

where $V_k^{max}(S)$ is the maximum aggregated variation of all integrated components, $V_k^{min}(S)$ is the minimum aggregated variation of all integrated components, and the other parameters are the same as in Eqs. 3-5.

Selecting optimal components in terms of QoS constraints for reliable service composition *S* is an optimization problem in CPSSs, which requires the following two conditions to be met: 1) guaranteeing QoS constraints $q(S) \le C_i$; 2) maximizing QoS utility value $U(S)$.

2.2 Related Work

Regarding the service composition problem in CPSSs, although the work done is insufficient for conducting a survey, many service composition methods have been proposed in Web service environments.

Certain previous service composition approaches [17-20] typically ignore the user's end-to-end QoS constrains, and most of them are inefficient owing to the large amount of time consumed. To obtain the most optimal service composition with global QoS constraint requirements, many globe optimal schemes[8,20,21] have been proposed. For example, L. Zeng et al.[20] focused on dynamic and quality-driven service composition with importance on multiple attributes and global constraints. The key purpose is proposing the QoS aggregation functions based on the weights of QoS attributes and adequately considering the importance of the users' weight in the service composition process. Subsequently, L. Zeng et al.[21] proposed a middleware platform for service composition by obtaining the maximum of utility functions over QoS attributes while also satisfying the user requirements. Soon after, D. Ardagna and B. Pernici[8] utilized a stripping and negotiation mechanism to obtain a feasible solution for overcoming the problems faced in service composition. The abovementioned schemes yield good results; however, on implementing them in CPSSs, because there is a need for enumerating all possible combinations of components, they cannot efficiently handle the excessive time cost with increasing number of components.

To reduce the time cost, many near-to-optimal methods for service composition[10,11] have been proposed. For example, Alrifai M. and Risse T.[10] combined global optimization with local selection methods for the selection of integrated services. By using mixed integer programming (MIP), this approach first finds the optimal decomposition of global QoS constraints into local constrains, and then using local selection, it finds the optimal composited services. Soon after, Qi L. et al.[11] proposed a QoS-based service composition approach by combining local optimization and enumeration. Unlike [10], in this work [11], the global QoS constraints are converted into local constrains for selecting parts of components in each service; then, MIP is used to enumerate all possible compositions for selecting a near-to-optimal solution. In addition, many heuristic optimization algorithms were proposed to solve the time cost problem, such as genetic algorithm[22,23], particle swarm optimization algorithm, [24], and others[25,26].

Moreover, Skyline services [27-29] and QoS dependence services[30,31] were also proposed to reduce the time cost of service composition. For example, Alrifai M. et al. [29] first filtered out a large number of service candidates by selecting Skyline service candidates from each service class and applied the K-means clustering algorithm to achieve service composition. Yu Q. et al. [27] proposed a Skyline computation approach for service composition and enabled users to optimally and efficiently access sets of service as an integrated service package. Moreover, this approach [27] simultaneously analyzes sequential and parallel models, constantly updates Skyline services, eliminates useless service composition, and applies mixed integer programming to find the best solution. Baraka et al. [31] presented a correlation-aware service composition approach for handing QoS dependencies among services and improved the composition quality. This approach [31] first models the quality dependencies among services and then prunes uninteresting compositions based on the search space narrow perception mechanism before selection. In addition, Feng et al. [30] considered the QoS-aware service composition problem in the presence of service dependent QoS, user provided topological and QoS constraints, and effectively handled the problem of service-dependent QoS by directly integrating it into the composition process instead of after the composition.

Although the abovementioned approaches perform well on their respective contexts with constant QoS values, because the QoS value of each component is dynamically changed, they cannot find the most reliable solution. To solve this problem, several service composition schemes have been proposed recently. For example, C. Wan and H. Wang [32] presented an uncertainty-aware QoS match-making model that realizes uncertain attribute-based service composition and improves the reliability of the composition service. S. Y. Hwang et al.[33] proposed two dynamic service composition strategies to select a component for accomplishing an incoming operation of the composite service by using aggregated reliability and achieving a high probability of finishing the remaining operation. Y. Tao and K. J. Lin [34] proposed a broker-based selection architecture to promote dynamic integration and adaptation of QoS-aware services with end-to-end QoS constraints. Moreover, our previous work [14,35-37] also focused on the time cost and reliability of service composition; however, since the QoS value of each component is dynamically changed and because of violent fluctuations in CPSSs, with increasing number of services, they still failed in finding the best reliability solution with low time cost.

3. Our Proposed Approach

In this paper, we propose a fast and reliable service composition approach in CPSSs, also known as FRSkyline, by QoS fluctuation computation and Skyline component computation.

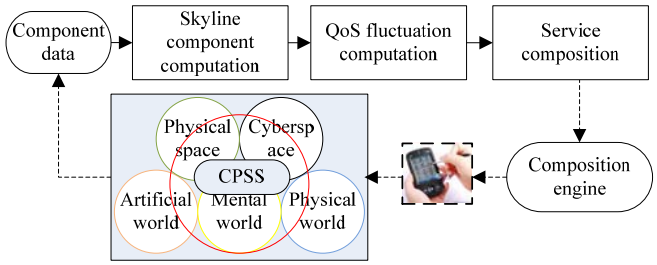

Fig. 1. Overview of our service composition approach

As shown in Fig. 1, FRSkyline contains three phases. The first phase is skyline component computation, in which we employ the concept of skyline component to consider the components not dominated by any other components as effective components, aiming to reduce the solution space required in service composition. The second phase is QoS fluctuation computation, in which we adopt the coefficient of variation to transform the QoS values into a qualitative concept, and it represents the degree of QoS uncertainty of a component. Then, we rank components according to the values and prune the high uncertainty components, aiming to ensure the reliability of service composition and further reduce the search space required in service composition. The final phase is service composition, in which we use the MIP algorithm to select the most reliable component with lower time cost for users in CPSSs through the use of a composition engine [12].

3.1 Skyline Component Computation

Service composition in CPSSs is intended at finding a set of components from each service that maximizes the overall utility value, while satisfying all the global QoS constraints. We could not just select the maximum utility value of each service because it does not guarantee satisfying all the global QoS constraints. In addition, the number of components is very large; thus, as an NP-hard problem, the time cost of enumerating each possible composition is not acceptable. Hence, to avoid the scarcely endurable time cost, the first priority is reducing the redundant components [21]. Hence, in this phase, certain redundant components must be filtered by Skyline component computation.

In this study, component redundancy denotes dominance relationships between components based on their QoS attributes and Skyline components according to Skyline [30].

Definition 1(dominance relationships): Consider a service s_i, two components $x, y \in s_i$, and each component has a QoS attribute vector *Q*. *x* dominant *y*, denoted as $x \prec y$, if *x* is as good and better than *y* in all QoS attributes and better in at least one QoS attribute in *Q*(i.e., $\forall k \in [1,|Q|]: q_k(x) \le q_k(y); \exists k \in [1,|Q|]: q_k(x) < q_k(y)$.)

Definition 2(Skyline components): The Skyline components in a service s_i, denoted by *SL*, are those that are not dominated by any other component in s_i (i.e., $SL = \{ x \in s_i \mid \neg \exists y \in s_i : y \prec x \}$).

Fig. 2 shows an example of a Skyline component in a certain service. Each component has two QoS attributes, namely response time and throughput. Hence, the services are represented as points in the 2D space. We find that the component *a* is a Skyline component because it is not dominated by any other component (i.e., there is no other component that offers both quicker response time and lower throughput than *a*). In addition, components *f* and *k* are also Skyline components, however, component *b* is not a Skyline component since it is dominated by component *a*. Then, components *a, f, and k* are effective components, however components *b, c, d, e, g, and h* will be pruned from the list of components in a service. Thus, we could use the notable Skyline algorithm[38] to determine which component should be pruned from the list of components in each service.

After Skyline component computation, the search space of service composition is reduced, and this can abruptly shorten the computation time of service composition in CPSSs.

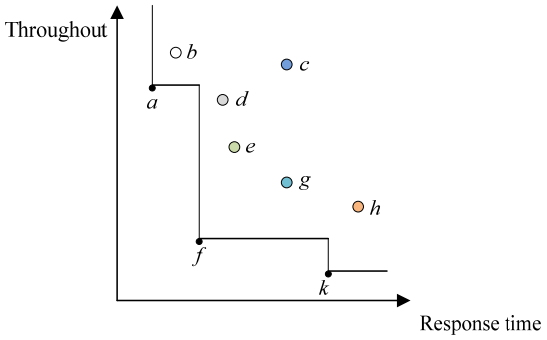

Fig. 2. Example of skyline components

3.2 QoS Fluctuation Computation

3.2.1 QoS Fluctuation

TABLE I. SET OF COMPONENT TRANSACTIONS [14,35-37,39]

Component: *W*		**Component: *T***	
ID	*Response time (ms)*	*ID*	*Response time (ms)*
w_1	32	t_1	20
w_2	39	t_2	52
w_3	42	t_3	42
w_4	35	t_4	21
w_5	34	t_5	44
$\underline{w}$	36.4	$\underline{T}$	35.8

In CPSSs, we first consider two components, *W* and *T*, offering similar components as an example to illustrate the QoS fluctuation. In this example, we consider five

transactions using each service. These transactions are represented as (w_1,…,w_5) and (t_1,…,t_5) as shown in Table I [14,35-37,39]. Table I shows the response time values of these transactions. The aggregated QoS value (*w* and *t*) is obtained by averaging all transactions.

From Table I, the aggregate QoS values of component *W* are greater than those of component *T*, i.e., $\underline{w} > \underline{t}$. In the traditional service composition approach, component *T* is usually selected in the component composition because 35.8 < 36.4. However, after deeply analyzing each transaction of these two components, we found the following two important facts that may be ignored in some existing approaches. (1) The aggregated response time of component *T* is slightly lesser than that of component *W* but the response times of the three transactions (w_2, w_3, w_5) of component *W* are lesser than those of (t_2, t_3, t_5) of component *T*, i.e., $w_2 < t_2$, $w_3 < t_3$ and $w_5 < t_5$. This implies that for most transactions, the response time of component *W* is less than that of component *T*. (2) The response time of component *T* is more volatile than that of component *W*, i.e., the QoS value of component *T* has a large variance, while the QoS value of component *W* is consistent and good relatively.

In the service composition process of CPSSs, if component *T* is selected, the actual composition result of component *T* may deviate from $\underline{t}$, which will lead to poor component quality of the composition or service composition failure. Hence, it may be obvious that component *W* is more stable than component *T* and the selection of *W* as a component may be better for the service composition process. Hence, the computation of the QoS fluctuation for distinguishing a component with consistently good QoS from another with large variance in its QoS is an important issue in service composition of CPSSs.

Hence, for this purpose, we employ the coefficient of variation to compute the QoS fluctuation by transforming QoS quantitative values to obtain a QoS qualitative concept. Then, based on the qualitative concept, a component with consistently good QoS can be distinguished from other components. We first provide the definition of the coefficient of variation in the following.

3.2.2 Coefficient of Variation

In probability theory, the coefficient of variation is a normalized measure of the dispersion degree of a probability distribution. It can facilitate in comparing the discrete degree of two or more random variables especially when their measurement units or average are not same. In general, the greater the uncertainty of the variable, the greater will be the coefficient of variation. In this paper, we consider the historical QoS value for a component as the discrete random variable, and then employ the coefficient of variation to filter the components with high QoS fluctuation in CPSSs. The definition is provided in the following.

Definition 3 (coefficient of variation): Let *X* be the random variable and $\{X_1, X_2, \ldots, X_n\}$ denote the range of *X*. Then, in this case, the coefficient of variation can be obtained by the following:

$$CV = S / \overline{X} \times 100\% \tag{4}$$

with

$$\begin{cases} \overline{X} = \dfrac{1}{n}\sum_{i=1}^{n} x_i \\ S = \sqrt{\dfrac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{X})^2} \end{cases} \tag{5}$$

where *S* represents the standard deviation of *X* and $\overline{X}$ represents the average *X*.

We apply the coefficient of variation to the values in Table I and then these response time values can be seen as quantitative QoS values expressed by five QoS historical values, i.e., (w_1,…,w_5) or (t_1,…,t_5). The QoS fluctuation of each component can be expressed by its eigenvector, $EI = \{\overline{X}, CV\}$. The eigenvectors of components *W* and *T* can be calculated as $EI_W = \{36.40, 11.09\}$ and $EI_T = \{35.80, 40.40\}$. Since $CV_W < CV_T$, the QoS fluctuation of component *W* is lower than that of component *T*. This means that the QoS of component *W* is consistently better than that of *T*. Thus, component *W* should be selected rather than component *T*; this selection is different from traditional approaches.

Through this process, the coefficient of variation can facilitate in further reducing the search space for service composition and thus ensure the reliability of service composition.

3.3 Service Composition

After Skyline component computation and QoS fluctuation computation, some unreliable and redundant components are pruned and components with consistently good QoS can be determined in each service. Then, a service composition solution should be used to determine the most reliable component of each service with global QoS constraints. Recently, mixed integer programming has been used to solve the service composition problem in several researches [8,18,28,31] and it is shown to achieve good results. Hence, in this paper, we use mixed integer programming to solve the optimization problem of service composition based on the filtered components in CPSSs by maximizing the fitness function *F(S)* as follows:

$$Max\ F(S)=\sum_{k=1}^{r}\frac{Q.CV_k^{max}-\sum_{i=1}^{n}\sum_{j=1}^{l}x_{ij}\cdot q_k(s_{ij})\cdot cv_k(s_{ij})}{(Q.CV_k^{max}-Q.CV_k^{min})}\cdot\omega_k \quad (6)$$

with

$$\begin{cases} Q.CV_k^{max}=\sum_{i=1}^{n}(Q_{i,k}^{max}\cdot CV_{i,k}^{max}),\ \ CV_{i,k}^{max}=\max_{\forall s_{ij}\in s_i} cv_k(s_{ij}) \\ Q.CV_k^{min}=\sum_{i=1}^{n}(Q_{i,k}^{min}\cdot CV_{i,k}^{min}),\ \ CV_{i,k}^{min}=\min_{\forall s_{ij}\in s_i} cv_k(s_{ij}) \end{cases} \quad (7)$$

subject to

$$\begin{cases} \sum_{i=1}^{n}\sum_{j=1}^{l}q_k(s_{ij})\cdot x_{ij}\le C_m, 1\le m\le r \\ \sum_{j=1}^{l}x_{ij}=1, 1\le i\le n, x_{ij}\in\{0,1\} \end{cases} \quad (8)$$

where x_{ij} is a binary decision variable for representing whether the component is integrated; a component s_{ij} is integrated if its corresponding variable *x* is set to 1 and if it is 0, it is not; $q_k(s_{ij})$ represents the *k*-th attribute value in component s_{ij}; Q_k^{max}, Q_k^{min} can be calculated by Eq. 3; $CV_{i,k}^{min}$ is the minimum variance value in the service s_i; and CV_k^{max} is the summation of each $CV_{i,k}^{max}$ in the service composition requirement S. Similarly, CV_k^{min} is the summation of each $CV_{i,k}^{min}$; *r* is the number of QoS attributes; *m* is the number of QoS constraints; *n* is the number of services; ω_k is the weight of each QoS attribute and satisfies $\sum_{k=1}^{r}\omega_k=1(0<\omega_k<1)$; C_m represents the *m*-th global QoS constrains with respect to the *r*-th QoS attribute.

By solving Eq. 6 and using mixed integer programming solver methods, a list of reliable components can be obtained and returned from each service to the service composition engine providing a service composition for users in CPSSs, as shown in Fig. 1.

4. EXPERIMENTS

We conducted experiments and compared our approach (FRSkyline) with other approaches in terms of the computation time and reliability. Moreover, we also studied the parameters of our approach.

4.1 Experiment Setup

For evaluating our approach, a real-world service QoS dataset is ideal for performance evaluation; unfortunately, there is not a real-world dataset from any CPSSs. Hence, it is very difficult to perform an objective and scientific experiment. However, in order to evaluate our approach, we need to establish an experiment that is close the objective; hence, we evaluated our approach using the two types of QoS datasets.

The first dataset is a real-world Web service QoS dataset obtained from [40,41], named WSDream. It contains nearly 2 million real-world service invocation records and each record contains two QoS attributes, i.e., response time and throughput. It is collected by 339 service users on 5825 Web services and also contains the information related to these Web services and the users. In order to ensure the experimental results of all approaches are not evaluated only for the WSDream dataset, we also evaluate our approach with a synthetic dataset (named Random dataset) [14,35-37,42,43] as the second dataset. This dataset contains 10,000 services from one CPSS where a random function was employed to create the data of the response time and throughput.

We conducted several service composition experiments for the CPSSs. Each experiment consisted of a composition request with *n* services, *l* components per service, and *m* global QoS constraints. By varying these parameters, we were able to collect results for each experiment; a unique combination of these three parameters was selected for these experiments.

In our experiments, the number of QoS attributes, *r,* was set to 2; the number of QoS constraints, *m,* was also set to 2; the number of components per service varied in the range of 100 to 1000; the weight for each of the two attributes was set to 0.5; and the *CV* was set to 50%. We note that practically the number of components in the service composition is generally less than 10 [50]. Hence, the number of services, *n,* was fixed at 5 for the WSDream dataset and 10 for the Random dataset. Further, the number of historical transactions was set at 250 for the WSDream dataset and 500 for the Random dataset.

All the experiments were performed on the same computer with Intel(R) Xeon(R) 2.6 GHz processor, 32.0 GB of RAM, Windows Server 2008R2, and MATLAB R2013a. In our paper, we compare FRSkyline with the following four approaches

4.1 Compared Approaches

- Global[8]: This approach is a standard global optimization approach with all components. In order to find the best solution, this approach iterates all solutions.
- GlobalCV: This approach considers only half components for which the coefficient of variation is relatively small in the global optimization.
- Skyline[44]: This approach considers only the Skyline components as the available components and other components are discarded.
- CVSkyline: This approach first computes the coefficient of variation and then uses Skyline

component computation to obtain the solution.

4.2 Comparison Results for the Computation Time

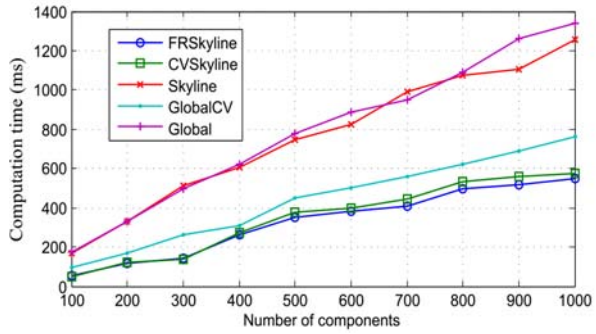

Fig. 3. Computation time for the WSDream dataset.

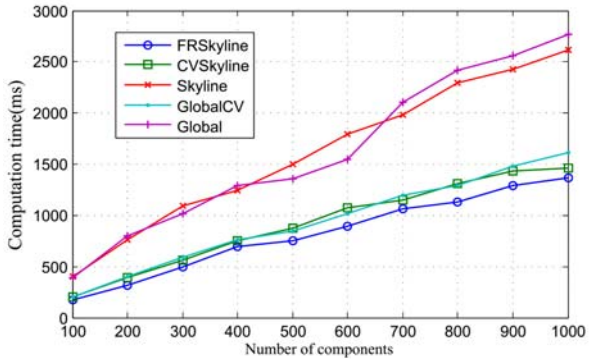

Fig. 4. Computation time for the Random dataset.

In this experiment, we compare FRSkyline with the other four approaches in terms of the computation time.

As shown in Figs. 3 and 4, the computation time of FRSkyline is always lower than that of the other approaches even when the number of components increases. This means that our approach can significantly reduce the time cost of a service composition in CPSSs because its search space is the smallest among all approaches. Note that, because GlobalCV considers only half components for service composition, its computation time is obviously lower than that of Skyline approach.

4.3 Comparison Results for the Reliability

In this experiment, similar to other studies [14,35-37,42,43], the reliability of a service composition for one CPSS is measured as shown below.

Using Definition 4, we calculate the reliability of a service composition as follows:

$$Reliability = \sum_{k=1}^{r} \frac{CV_k^{max} - \sum_{i=1}^{n} cv_k(s_i)}{CV_k^{max} - CV_k^{min}} \cdot \omega_k \times 100\% \quad (9)$$

with

$$\begin{cases} CV_k^{max} = \sum_{i=1}^{n} CV_{i,k}^{max} \\ CV_k^{min} = \sum_{i=1}^{n} CV_{i,k}^{min} \end{cases} \quad (10)$$

where CV_{max} is the maximum aggregated coefficient of the variation of all integrated components, CV_{min} is the minimum aggregated coefficient of variation of all integrated components; r is the number of QoS attributes; and ω_k is the weight of each QoS attribute.

Based on Definition 4, we compare FRSkyline with the other four approaches in terms of the reliability of the service composition.

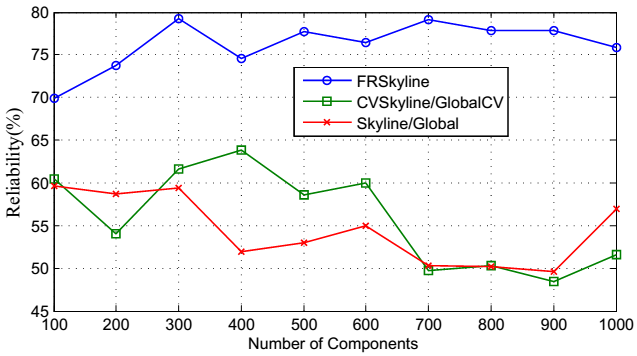

Fig. 5. Reliability for the WSDream dataset.

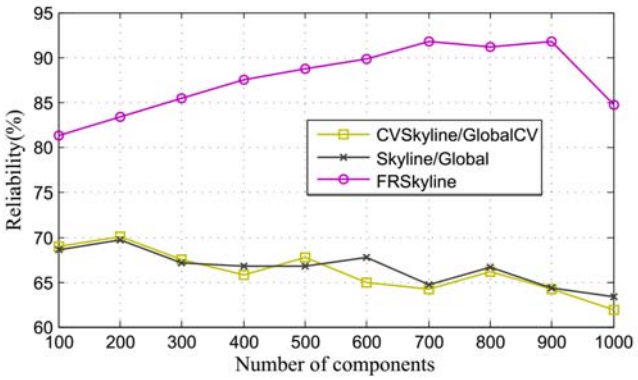

Fig. 6. Reliability for the Random dataset.

Figs. 5 and 6 show that irrespective of the number of components, the reliability of FRSkyline is always higher than that of other approaches. These experimental results illustrate that FRSkyline can effectively overcome the QoS fluctuation to avoid failure of service composition as the reliability of the integrated components is very high. FRSkyline not only uses CV, but also adopts Skyline to perform service composition, which means that its performance is better than only adopting CV or Skyline. Hence, by using *CV* to monitor historical QoS transactions of Skyline components, FRSkyline can effectively identify which components have large variances in terms of their QoS values and then can prune them. Moreover, in the context of this experiment, the reliability of CVSkyline and GlobalCV are the same due to *CV*, and Skyline and Global also have the same reliability as there is no *CV*.

4.5 Parameter Studies

The aim of parameter studies is to (1) show how parameters affect our approach for better understanding of the researchers; (2) show how our approach can conveniently be used by a software engineer (for example, the parameter setting may be different in practical applications).

4.5.1 Parameter CV

In this experiment, we study the computation time and reliability with the change in the parameter *CV* for the WSDream dataset.

Fig. 7 shows the computation time for the WSDream dataset with the variation in parameter *CV*; the computation time for our approach increases as *CV* increases from 10% to 90%. This demonstrates that the computation time increases with increasing *CV* because the component search space grows rapidly. Fig. 8 shows that for the WSDream dataset, the reliability of our approach increases with *CV*.

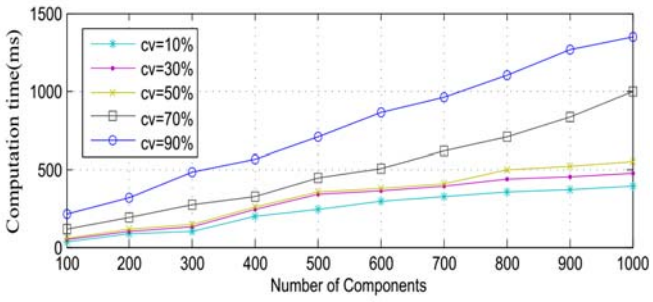

Fig. 7. Computation time with respect to CV for the WSDream dataset.

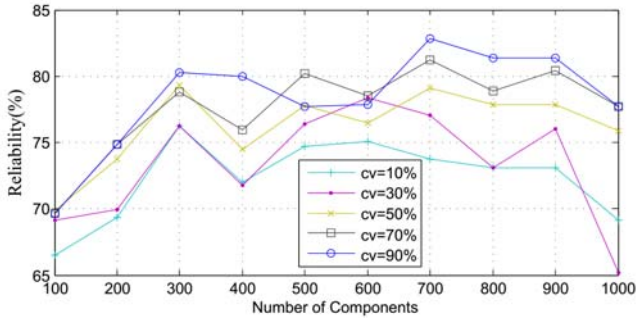

Fig. 8. Reliability with respect to CV for the WSDream dataset.

4.5.2 Parameter ω

In this experiment, we study how parameter ω affects the reliability of a service composition where ω represents the user requests for each QoS attribute for the WSDream dataset. We fixed the number of components as 500 and *CV*=50%. Then, the weight of response time varies from 0.125 to 0.875 and correspondingly the weight of throughout varies from 0.875 to 0.125.

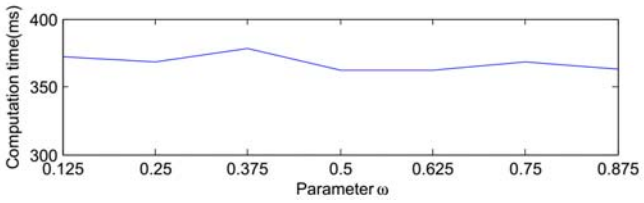

Fig. 9. Computation time with respect to ω.

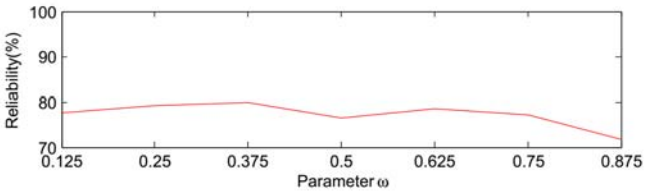

Fig. 10. Reliability with respect to ω.

Figs. 9 and 10 show that irrespective of parameter ω the results obtained by our approach are good. Moreover, we also find that when the average value (0.5) of parameter ω is used, our approach has the lowest computation time.

4.6 Limitations of Our Approach

- Our approach may fail to find a solution when the number of components is less. The higher the number of components, the better the performance of our approach in CPSSs.
- Our approach is not suitable for a new component or the components that are used rarely because the number of historical QoS data records is very low for CPSSs.
- Note that our service composition approach can filter the component with high QoS fluctuation but for the reputation and price attributes, the QoS fluctuation computation cannot be performed for CPSSs.

6 Conclusions

With the increasing number of services in CPSSs, increasing number of components from different domains would be required to be integrated to support new applications such as virtual reality 3D glasses. Hence, in this paper, we presented a fast and reliable service composition approach. Our approach first used Skyline component computation to prune the redundant components and then employed the coefficient of variation to compute the QoS fluctuation to guarantee the reliability. Finally, we used mixed integer programming to find the best and reliable solution with lower computation time. We evaluated our approach using both real-world dataset and randomly generated dataset. The experimental results showed that our approach performs better than other four approaches.

Our future work will focus on service composition based on component QoS prediction and service

composition based on QoS attribute relevance (e.g., service prices) in CPSSs [45-47]. Moreover, comparing our approach with other latest approaches is also the one of our future work.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (61472047 and 61571066).

REFERENCES

[1] Z. Liu, D. s. Yang, D. Wen, W. m. Zhang, and W. Mao, "Cyber-Physical-Social Systems for Command and Control," *IEEE Intelligent Systems, 4,* vol. 26, pp. 92-96, 2011.

[2] A. Smirnov, A. Kashevnik, and A. Ponomarev, "Multi-level Self-organization in Cyber-Physical-Social Systems: Smart Home Cleaning Scenario," *Procedia CIRP,* vol. 30, pp. 329-334, 2015.

[3] M. P. Papazoglou and D. Georgakopoulos, "Introduction: Service-oriented computing," *Communications of the ACM, 10,* vol. 46, pp. 24-28, 2003.

[4] A. Gustavo, F. Casati, H. Kuno, and V. Machiraju, "Web services: concepts, architectures and applications," Springer Berlin, 2004.

[5] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu, "Web services composition: A decade's overview," *Information Sciences,* vol. 280, pp. 218-238, 2014.

[6] M. Chen and Y. Yan, "Redundant service removal in qos-aware service composition," *in: Proc. of IEEE ICWS*, 2012, pp. 431-439.

[7] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, "A framework for QoS-aware binding and re-binding of composite web services," *Journal of Systems and Software, 10,* vol. 81, pp. 1754-1769, 2008.

[8] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *Software Engineering, IEEE Transactions on, 6,* vol. 33, pp. 369-384, 2007.

[9] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for Web services selection with end-to-end QoS constraints," *ACM Transactions on the Web, 1,* vol. 1, p. 6, 2007.

[10] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient QoS-aware service composition," *in: Proc. of WWW*, 2009, pp. 881-890.

[11] L. Qi, Y. Tang, W. Dou, and J. Chen, "Combining local optimization and enumeration for qos-aware web service composition," *in: Proc. of IEEE ICWS,* 2010, pp. 34-41.

[12] F. Wagner, F. Ishikawa, and S. Honiden, "QoS-Aware Automatic Service Composition by Applying Functional Clustering," *in: Proc. of IEEE ICWS,* 2011, pp. 89-96.

[13] S. G. Wang, Q. B. Sun, and F. C. Yang, "Towards Web Service selection based on QoS estimation," *International Journal of Web and Grid Services, 4,* vol. 6, pp. 424-443, 2010.

[14] W. Shangguang, Z. Zheng, S. Qibo, Z. Hua, and Y. Fangchun, "Cloud model for service selection," *in: Proc. of IEEE INFOCOM WKSHP*, 2011, pp. 666-671.

[15] S. Wang, Q. Sun, H. Zou, and F. Yang, "Particle Swarm Optimization with Skyline Operator for Fast Cloud-based Web Service Composition," *Mobile Networks and Applications, 1,* vol. 18, pp. 116-121, 2013/02/01 2013.

[16] S. G. Wang, Q. B. Sun, H. Zou, and F. C. Yang, "Web Service Selection Based on Adaptive Decomposition of Global QoS Constraints in Ubiquitous Environment," *Journal of Internet Technology, 5,* vol. 12, pp. 757-768, 2011.

[17] E. M. Maximilien and M. P. Singh, "A framework and ontology for dynamic web services selection," *IEEE Internet Computing, 5,* vol. 8, pp. 84-93, 2004.

[18] B. Benatallah, M. Dumas, M.-C. Fauvet, F. A. Rabhi, and Q. Z. Sheng, "Overview of some patterns for architecting and managing composite web services," *ACM SIGecom Exchanges, 3,* vol. 3, pp. 9-16, 2002.

[19] Y. Liu, A. H. Ngu, and L. Z. Zeng, "QoS computation and policing in dynamic web service selection," *in: Proc. of WWW*, 2004, pp. 66-73.

[20] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng, "Quality driven web services composition," *in: Proc. of WWW*, 2003, pp. 411-421.

[21] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Transactions on Software Engineering, 5,* vol. 30, pp. 311-327, 2004.

[22] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, "An approach for QoS-aware service composition based on genetic algorithms," *in: Proc. of GEC,* 2005, pp. 1069-1075.

[23] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, "QoS-aware replanning of composite web services," *in: Proc. of IEEE ICWS*, 2005, pp. 121-129.

[24] X. Zhao, Z. Wen, and X. Li, "QoS-aware web service selection with negative selection algorithm," *Knowledge and Information Systems, 2,* vol. 40, pp. 349-373, 2014.

[25] E. Lee and B. Lee, "An agent-based web service composition using semantic information and QoS," Springer, 2007, pp. 928-937.

[26] L. Wei, L. Junzhou, L. Bo, Z. Xiao, and C. Jiuxin, "Multi-agent based QoS-aware service composition," *in: Proc. of IEEE SMC*, 2010, pp. 3125-3132.

[27] Q. Yu and A. Bouguettaya, "Computing service skylines over sets of services," *in: Proc. of IEEE ICWS,* 2010, pp. 481-488.

[28] K. Benouaret, D. Benslimane, and A. Hadjali, "On the use of fuzzy dominance for computing service skyline

based on qos," *in: Proc. of IEEE ICWS*, 2011, pp. 540-547.
[29] M. Alrifai, D. Skoutas, and T. Risse, "Selecting skyline services for QoS-based web service composition," *in: Proc. of WWWI*, 2010, pp. 11-20.
[30] Y. Feng, L. D. Ngan, and R. Kanagasabai, "Dynamic service composition with service-dependent QoS attributes," *in: Proc. of IEEE ICWS*, 2013, pp. 10-17.
[31] L. Barakat, S. Miles, and M. Luck, "Efficient correlation-aware service selection," *in: Proc. of IEEE ICWS*, 2012, pp. 1-8.
[32] C. Wan and H. Wang, "Uncertainty-aware QoS Description and Selection Model for Web Services," *in: Proc. of IEEE SCC*, 2007, pp. 154-161.
[33] S. Y. Hwang, E. P. Lim, C. H. Lee, and C. H. Chen, "Dynamic Web Service Selection for Reliable Web Service Composition," *IEEE Transactions on Services Computing, 2,* vol. 1, pp. 104-116, 2008.
[34] Y. Tao and K. J. Lin, "A broker-based framework for QoS-aware Web service composition," *in: Proc. of IEEE EEE*, 2005, pp. 22-29.
[35] S. Wang, L. Sun, Q. Sun, X. Li, and F. Yang, "Efficient Service Selection in Mobile Information Systems," *Mobile Information Systems,* vol. 2015, p. 10, 2015.
[36] S. G. Wang, Z. B. Zheng, Q. B. Sun, H. Zou, and F. C. Yang, "Reliable web service selection via QoS uncertainty computing," *International Journal of Web and Grid Services, 4,* vol. 7, pp. 410-426, 2011.
[37] L. Sun, S. Wang, J. Li, Q. Sun, and F. Yang, "QoS Uncertainty Filtering for Fast and Reliable Web Service Selection," *in: Proc. of IEEE ICWS*, 2014, pp. 550-557.
[38] S. Borzsony, D. Kossmann, and K. Stocker, "The skyline operator," *in: Proc. of IEEE ICDE*, 2001, pp. 421-430.
[39] S. Wang, L. Huang, L. Sun, C.-H. Hsu, and F. Yang, "Efficient and reliable service selection for heterogeneous distributed software systems," *Future Generation Computer Systems, doi:10.1016/j.future.2015.12.013, 2015*.
[40] Z. Zheng, Y. Zhang, and M. R. Lyu, "Distributed qos evaluation for real-world web services," *in: Proc. of IEEE ICWS*, 2010, pp. 83-90.
[41] Y. Zhang, Z. Zheng, and M. R. Lyu, "Exploring latent features for memory-based QoS prediction in cloud computing," *in: Proc. of IEEE SRDS*, 2011, pp. 1-10.
[42] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient QoS-aware service composition," *in: Proc. of WWW*, 2009, pp. 881-890.
[43] M. Alrifai, T. Risse, and W. Nejdl, "A Hybrid Approach for Efficient Web Service Composition with End-to-End QoS Constraints," *Acm Transactions on the Web, 2,* vol. 6, pp. 1-31, May 2012.
[44] M. Alrifai, D. Skoutas, and T. Risse, "Selecting skyline services for QoS-based web service composition," *in: Proc. of WWW*, 2010,pp.11-20.
[45] X. Liu, M. Dong, K. Ota, P. Hung, A. Liu, "Service Pricing Decision in Cyber-Physical Systems: Insights from Game Theory," *IEEE Transactions on Services Computing, 2, vol.* 9, pp. 186-198, 2016.
[46] M. Dong, X. Liu, Z. Qian, A. Liu, L. T. Wang, "QoE-ensured price competition model for emerging mobile networks," *IEEE Wireless Communications*, 4, vol. 22, pp. 50-57, 2015.
[47] M. Dong, K. Ota, L. T. Yang, A. Liu, M. Guo, "LSCD: A Low-Storage Clone Detection Protocol for Cyber-Physical Systems," *IEEE Transactions on CAD of Integrated Circuits and Systems*, 5, vol. 35, pp.712-723, 2016.

Shangguang Wang is an Associate Professor at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications (BUPT). He received his Ph.D. degree at BUPT in 2011. He has co-authored more than 100 papers, and played a key role at many international conferences, such as TPC co-Chair of IEEE ICFEC 2017, General Chair of CollaborateCom 2016, General Chair of ICCSA 2016. His research interests include Service Computing, Cloud Computing, and QoS Management. He is a Senior Member of the IEEE.

Ao Zhou received her M.E. in computer science and technology from Beijing University of Posts and Telecommunications in 2012. She is currently a Ph.D. candidate at Beijing University of Posts and Telecommunications. Her research interests include cloud computing and service reliability.

Mingzhe Yang received his bachelor degree in telecommunication and management from Beijing University of Posts and Telecommunications in 2015. He is currently a Ph.D. candidate at Beijing University of Posts and Telecommunications. His research interests include service reliability and mobile cloud computing.

Lei Sun is a postgraduate at Beijing University of Posts and Telecommunications in China. His research interests include service selection and cloud service.

Ching-Hsien Hsu is professor in the Department of Computer Science and Information Engineering at Chung Hua University, Taiwan. His research interests include 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 chair, and more than 200 conferences/workshops as program committee member. He is the

editor-in-chief of the *International Journal of Grid and High Performance Computing*, and has served on the editorial boards of approximately 20 international journals.

Fangchun Yang received his Ph.D.in communications and electronic systems from the Beijing University of Posts and Telecommunication in 1990. He is currently professor at the Beijing University of Posts and Telecommunication, China. He has published six books and more than 80 papers. His current research interests include network intelligence, service computing, communications software, soft-switching technology, and network security. He is a Fellow of the IET.