Research Article

Trusted QoS Assurance Approach for Composite Service

Meng Cai ¹, Ying Cui ², Yang Yu ¹, Ying Xue ¹, and Han Luo ¹

¹School of Humanities and Social Sciences, Xi'an Jiaotong University, Xi'an 710049, China

²School of Mechano-Electronic Engineering, Xidian University, Xi'an 710071, China

Correspondence should be addressed to Meng Cai; mengcai@mail.xjtu.edu.cn

Received 18 March 2021; Revised 16 April 2021; Accepted 29 April 2021; Published 17 May 2021

Academic Editor: Hao Peng

Copyright © 2021 Meng Cai et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The characteristics of service computing environment, such as service loose coupling, resource heterogeneity, and protocol independence, propose higher demand for the trustworthiness of service computing systems. Trust provisioning for composite services has become a hot research spot worldwide. In this paper, quality of service (QoS) planning techniques are introduced into service composition-oriented QoS provisioning architecture. By QoS planning, the overall QoS requirement of the composite service is decomposed into separate QoS requirement for every constituent atom service, the QoS level of which can subsequently be satisfied through well-designed service entity selection policies. For any single service entity, its QoS level is variable when the deployment environment or the load of service node changes. To mitigate the uncertainty, we put forward QoS preprocessing algorithms to estimate the future QoS levels of service entities with their history execution data. Then, based on the modeling of composite service and QoS planning, we design three algorithms, which include the time preference algorithm, cost/availability (C/A) preference algorithm, and Euclidean distance preference algorithm, to select suitable atomic services meeting the user's requirements. Finally, by combining genetic algorithm and local-search algorithm, we propose memetic algorithm to meet the QoS requirements of composite service. The effectiveness of the proposed methods by which the QoS requirements can be satisfied up to 90% is verified through experiments.

1. Introduction

With the enhancement of the interconnection, intercommunication, and interoperability of the information processing platform, as well as the improvement of the degree of integration, higher requirements are put forward for the reliable serviceability and service quality assurance of the service computing system [1]. The general information processing platform provides a consistent operating environment for all kinds of general information processing services and supports the intensive operation of all kinds of information services. At the same time, various information systems on the same platform, different services in the same information system, different service requests of different users, and different requests of the same user all have various requirements of service quality [2]. On the one hand, the unique characteristics of service computing, such as loose coupling, location transparency, and protocol independence, make it an ideal model for building various complex

data processing and information service systems in the network environment. On the other hand, factors such as the looseness, independence, and heterogeneity of the computing environment also bring new technical challenges to the guarantee of nonfunctional attributes such as real-time performance, security, reliability, and availability of service computing. In the open and dynamic network environment, the composite service-oriented computing model needs to implement a reliable software architecture with high reliability, adaptability, and self-management capability [3].

Trusted computing integrates real-time, safe, reliable, and available computing theories in the traditional category and studies the interaction of system behaviors under nonfunctional attributes in a unified way, so that the behavior state can be monitored, the behavior results can be evaluated, and the abnormal behaviors can be controlled [4]. Quality of service (QoS) is a comprehensive index describing the nonfunctional characteristics of computer systems and communication networks, which is used to measure the

satisfaction degree of using a service [5]. Both trusted computing and QoS aware computing focus on the multidimensional nonfunctional characteristics of the system, so it can be considered to represent the trusted attribute as the multidimensional QoS demand of the system, to provide credibility guarantee for the network service. It has become an important direction in the field of trusted computing to use the service quality theory to provide a guarantee for the trusted demand of service [6].

As the number of Web services shared over the network increases dramatically, the traditional choice of user services is based on QoS values published by vendors. In reality, the network environment shared by Web services is characterized by openness, instability influenced by the external environment, and unreliability of QoS value of services influenced by business competition. The incredibility of the QoS value of Web services and the uncertainty of the invocation of Web services make users unable to obtain Web service combinations according to actual requirements. Therefore, in the case of complex network environment and massive Web services, how to efficiently select a group of Web services that can meet users' functional requirements and have credibility from a large number of candidate services is a key scientific and technical problem to be solved in the field of service computing.

In this paper, we proposed an approach composed of QoS modeling, QoS prediction, QoS planning, and service selection, to guarantee the QoS of composite service. Firstly, we built the models to describe the trusted QoS of atomic service and trusted QoS of composite service, respectively. Secondly, we introduced the exponential smoothing method to evaluate the abilities of atomic services and found that the prediction of QoS information of atomic service by the exponential smoothing method was accurate and feasible. Based on the QoS estimation, we modeled QoS calculation rules of composite service. In addition, we proposed a series of service selection algorithms and used computational simulation to demonstrate that the Euclidean distance preference algorithm had a more balanced and excellent characteristic. Finally, we compared the memetic algorithm with random selection to meet the multidimensional QoS requirement of composite service.

2. Related Works

With the popularization and development of Web service technology, on the one hand, more and more users begin to complete various business processes by combining Web service, and the business process pattern is being promoted to the broader users. On the other hand, users no longer need specific Web service as the executors of tasks in the process owing to the appearance of a large number of Web services with the same or similar functions rather than automatically selecting Web service according to the needs and constraints of users.

To meet the specific application background and requirements, we usually need to combine multiple Web services according to certain granularity to form a service process with a specific structure and realize complete

business logic in dealing with complex business. Facing this situation makes it important to evaluate and ensure the process of QoS.

When users select Web service, in addition to functional attributes, they can also set up constraints on some non-functional attributes, that is, by constraining QoS attributes to be selected. QoS can include attributes such as price, response time, availability, and reliability [7]. Providers of Web service also tend to give a range of QoS attribute values of the service, thus providing a reference for potential users.

In QoS sensitive business processes, the QoS of the entire process is also a feature of user concern except requiring the process to complete predefined tasks [8]. Therefore, how to make an effective choice in the alternative Web service to meet the requirements of users, so that the selected Web service can not only complete the task assigned by the process and meet the local constraints, but also cooperate with the Web service which completes other tasks in the process, is an urgent issue that needs to be solved and it is often referred to as QoS sensitive Web service composite issue.

At present, many scholars have studied the trusted computing technology and QoS technology in the complex distributed computing environment and achieved some results [9]. Existing research is often based on the idea of QoS guarantee, from the perspective of reliability, security, and other nonfunctional attributes of service computing research, especially through the combination of atomic services to maximize the needs of users [10, 11]. For Web service composition in the study, the researcher proposed the service selection method based on mathematical programming; service selection method based on genetic algorithm (GA); service selection method based on particle swarm optimization (PSO) algorithm, based on swarm intelligence computing; service selection method based on global QoS constraint decomposition; service selection method based on artificial intelligence theory; and many other kinds of service composition methods [12–14].

The literature [15] carries out a survey and research review in the field of Web services technology, clarifies the research motivation of service credibility evaluation, discusses the importance of atomic Web services credibility in the construction of service composition strategy, and classifies the credibility evaluation methods. Wang et al. [16] divided the process of service composition into three stages: service planning, service selection, and service binding, to reasonably and effectively evaluate the credibility of services, and investigated the credibility of services in each stage. Ardagna and Pernici [17] designed a service composition method based on mixed-integer programming when considering the possible local and global QoS constraints in service selection. Sun and Zhao [18] decomposed global QoS constraints into local QoS constraints through a mixed-integer programming method, thus transforming the service composition problem belonging to the global optimization problem into the local optimal service selection problem and reducing the complexity of the problem. Surianarayanan et al. [19] proposed a service composition method of constraint decomposition method to calculate the utility of

composite services through the utility of component services and the utility constraints of component services of composite service constraints. The literature [20] decomposed service composition into two stages: constraint decomposition and service selection. Another study [21] proposed a service optimization combination method based on correlation graph and 0-1 linear programming. The service optimization combination method based on mathematical programming can effectively solve the problem of service optimization combination to a certain extent. However, when the problem scale is large, the calculation time of such a method is relatively long, and it cannot meet the real-time requirements of service composition.

Few researchers in the above studies distinguish the concept of service and service entity from the perspective of trusted QoS guarantee architecture. When making QoS planning, planners are faced with the underlying atomic service entity that provides specific service functions. In this system view, the requirement of trusted QoS for composite services can only be met through QoS planning based on the choice of atomic service entities. Moreover, the QoS embodied by the service entity at run time is often inconsistent with the QoS declared by the service entity, which makes this QoS planning method limited in guaranteeing the trusted QoS of the composite service at run time.

3. Trusted QoS for Service Modeling

In this paper, we refer to the composite service with trusted constraints of QoS described jointly by BPEL language and state diagrams as processes. We focus more on service entities than service for the process; what directly determines the process QoS is not the service entity of QoS, but the QoS; the planning of QoS is also based on the service (abstract service) of QoS, not the service entity of QoS.

Service composition is to combine some relatively simple services according to certain business process logic into complex composite services, thus providing more powerful and complete business functions. Based on the loose service combination, the customization of business processes can be realized, and personalized services can be provided to users, so that the business can adapt to changes [22].

Entities are defined as people, units, various computer hardware and software modules, or their multiple sets. Multiple sets are linear sum of integral coefficients of various elements.

Atomic service/service entity is defined as one entity that works or operates for another entity called service. Here, the work is for the service provider; the operation (computer and information receiving, sending, transforming, etc.) is for the computer hardware and software module that provides the service.

Composite service is defined as merging multiple small granularity services into one large granularity service. In the network architecture, the service combination is a combination of service providers.

Abstract service is defined as service functions provided by service entities. It can also be understood as an abstract proxy composed of atomic service entities with the same

service function (but the nonfunctional capabilities of QoS will be different), which is used to represent all atomic services with this function.

With this concept, in the planning process of QoS, we can put forward reasonable requirements of QoS for each abstract service in the composite service, because the abstract service represents the operation status and QoS capability of all atomic service entities with the same function. In this paper, unless otherwise specified, we call abstract service "service", which is different from atomic service (service entity).

When the business process changes, it is only necessary to reorganize the services that make up the business process or dynamically adjust the component services that make up the business process to quickly respond to the changes of the business process [23]. In the business process realized by service composition, both developers and end-users can combine the existing services to obtain new and complex services and form new or even dynamic business processes. This promotes the utilization of service, improves the reusability of service, and accelerates the development of application projects [24].

3.1. Trusted QoS for Atomic Service Modeling. There are many methods for modeling the trusted properties of atomic services. Generally speaking, it is divided into real time, availability, reliability, cost, and so on. In this paper, we use the following methods to model:

- (i) Real time: it is used to measure the time characteristics of a service. A service call time t defined as follows: on the service caller side, the time difference between t_1 moment and t_2 moment from sending service request to receiving returned result is $|t_2 - t_1|$. t includes network transmission time. The real-time performance of service will be described by ternary < minimum. Minimum, average, and maximum are all elements of R^+ . Average represents the average service call time to access the service in history, and maximum represents the minimum service call time to access the service in history. The average access time can be calculated using the following formula: $T(s) = (\sum_{i=1}^n T_i(s)/n)$, where n is the number of times the service was observed in history, and $T_i(s)$ is the time at which the service was called in history.

The reason why this arithmetic average method has a predictive function is that it can always eliminate to a certain extent the influence of random changes caused by accidental interference on the QoS capability of atomic service. However, this method also conceals the fluctuating trend in the development of the atomic service itself.

Aiming at the limitation of this modeling method, we will use a weighted average method named exponential smoothing in the planning process of QoS. It means that the recent QoS characteristics of the atomic service have a greater impact on the

future and the more up-to-date information reflected; the importance of historical data which predict the QoS capability of the atomic service should decrease from the close to the distant in time.

- (ii) **Availability:** the availability of a service describes the probability that the service can be accessed correctly. The availability of service S , namely $A(S)$, is defined as a real number on $[0, 1]$, $A(S) \in [0, 1]$. Use the following method to measure the availability of services: $A(S) = T_\theta(S)/\theta$, where θ is a time constant in seconds and $T_\theta(S)$ is the time at which the service can be accessed correctly in the past θ seconds. In practical application, different θ values are selected for observation according to the access frequency of different services. For services with high frequency of access, we choose a smaller period of time for investigation. For services with low frequency of access, we choose a larger period of time for investigation.
- (iii) **Cost:** it is the expense of using a service. The trusted cost of a service is described as a positive integer. The cost of atomic services is defined as an element on $\mathbb{I}^+$. In practical applications, the cost of a service is given by the provider of the service (or the cost is determined according to the time spent by the service user to calculate the resources and the use of the resources).

3.2. Trusted QoS for Composite Service Modeling. The trusted nature of the process has the same trusted aspect and description domain as the trusted nature of the service. The credibility of the composite service is determined by the credibility of its subservices and its composition mode. The operation function of composite service is composed of multiple subtasks in a certain way, and there are control dependency and data dependency between subtasks. For the execution process of composite services S , we can decompose it with reference to the method in [10] and describe the control dependence in the execution process by using state diagram. On this basis, we can use the method of [8] to generate a directed acyclic graph to describe its execution process. The node represents the subtask, and the direction of the edge describes the execution order of the task. Different subtasks may be provided by different subservices. The execution plan of a service S is defined as $\langle G, P \rangle$, where G is a directed acyclic graph representing its control dependency during execution. $P = \{\langle t_1, s_1 \rangle, \langle t_2, s_2 \rangle, \dots, \langle t_n, s_n \rangle\}$ describes the corresponding relationship between subtasks and subservices, and $\langle t_i, s_i \rangle$ indicates that t_i is completed by the service s_i .

Figure 1 is a typical implementation plan diagram. The process of service S can be broken down into multiple subtasks. The control dependency between tasks is described by the directed acyclic graph. The starting state is t_i , two parallel subtasks t_1 and t_2 are triggered simultaneously, the concurrent activities are synchronized at t_4 , then task t_5 is started, and finally t_f is finished. The functions of these subtasks are provided by several services. $\langle t_i, s_i \rangle$ describes

FIGURE 1: A schematic diagram of the typical implementation plan.

the corresponding relationship between subtasks and services. Among them, $\langle t_i, t_1, t_2, t_3, t_4, t_f \rangle$ is a critical path (shown by the green squares in Figure 1 in the execution process), and the execution time of the critical path determines the execution time of the whole operation.

For a more complex business process of Web service, we can decompose it into recursive combinations of several basic institutions. Table 1 gives the QoS calculation rules for the flow of several basic structures. In this paper, we will explain and verify the proposed method with three common QoS attributes: time, cost, and availability.

Sequence structure (Sequence): the service process of sequence structure $\mathbf{W}$ is composed of service by m ($s_1, s_2, s_i, \dots, s_m$) in a certain order.

Time (T): the response time $T(\mathbf{W})$ of the service process $\mathbf{W}$ is composed of the sum of the response time of each service that constitutes the service process.

$$T(\mathbf{W}) = \sum_{i=1}^m T(s_i). \quad (1)$$

Cost (C): the cost $C(\mathbf{W})$ of service process $\mathbf{W}$ is the sum of the costs of each service that constitutes the service process.

$$C(\mathbf{W}) = \sum_{i=1}^m C(s_i). \quad (2)$$

Availability (A): the availability $A(\mathbf{W})$ of service process $\mathbf{W}$ is composed of the availability product of each service that constitutes the service process.

$$A(\mathbf{W}) = \prod_{i=1}^m A(s_i). \quad (3)$$

Choice structure (Choice): the service process of choice structure $\mathbf{W}$ is composed of service by m ($s_1, s_2, s_i, \dots, s_m$) in a certain order.

For the selection structure, each selection branch is marked with the probability of being selected. For example, a process selection structure has two branches, its prices c_1 and c_2 , respectively. The probability of being selected is p and q , respectively. The cost of the whole structure is calculated as follows: $pc_1 + qc_2$, s.t. $p + q = 1$.

TABLE 1: QoS calculation methods for composite service.

QoS dimension	Workflow			
	Sequence	Choice	Parallel	Loop
Time (T)	$\sum_{i=1}^m T(s_i)$	$\sum_{i=1}^m P_i * T(s_i)$	$\text{Max}\{T(s_i)\}, i \in (1, 2, \dots, m)$	$m * T(s)$
Availability (A)	$\sum_{i=1}^m A(s_i)$	$\sum_{i=1}^m P_i * A(s_i)$	$\prod_{i=1}^m A(s_i)$	$A(s)^m$
Cost (C)	$\sum_{i=1}^m A(s_i)$	$\sum_{i=1}^m P_i * C(s_i)$	$\sum_{i=1}^m C(s_i)$	$m * C(s)$

The initialization value of probability can be determined by the designer of business process, and then the information obtained by monitoring the process execution process is updated continuously. Accordingly, for the selection structure 1, ..., m with N of branches, the probability of each branch is $\{p_1, p_2, \dots, p_i, \dots, p_m\}$, of which $\sum_{i=1}^m p_i = 1$. Usually, the QoS of this choice structure is the attribute value of each task and its corresponding branch probability, and then the sum.

Time (T): the response time $T(W)$ of the service process W is composed of the weighted sum of the response time probabilities of the service that constitutes the service process.

$$T(W) = \sum_{i=1}^m P_i * T(s_i). \quad (4)$$

Cost (C): the cost $C(W)$ of service process W is composed of the weighted sum of the cost probabilities of the service that constitutes the service process.

$$C(W) = \sum_{i=1}^m P_i * C(s_i). \quad (5)$$

Availability (A): the availability $A(W)$ of service process W is composed of the weighted sum of the cost probabilities of the service that constitutes the service process.

$$A(W) = \sum_{i=1}^m P_i * A(s_i). \quad (6)$$

Parallel structure (Parallel): the service process of parallel structure W is composed of service by m ($s_1, s_2, s_i, \dots, s_m$) in synchronization.

Time (T): the response time $T(W)$ of the service process W is composed of the maximum value of the response time that constitutes the service process. For a concurrent structure, the time response of the process is limited by the branch with the largest time response.

$$T(W) = \text{Max}\{T(s_i)\}, \quad i \in (1, 2, \dots, m). \quad (7)$$

Cost (C): the cost $C(W)$ of service process W is composed of the sum of the costs of the services that constitute the service process.

$$C(W) = \sum_{i=1}^m C(s_i). \quad (8)$$

Availability (A): the availability $A(W)$ of service process W is composed of the availability product of each service that constitutes the service process.

$$A(W) = \prod_{i=1}^m A(s_i). \quad (9)$$

Loop structure (Loop): the service process W of loop structure is composed of m times repetition of loop body service s . For a given service s and the number of cycles m , it can also be equivalent to a sequential structure which is composed of m of the same service s .

If a loop trip costs c_1 , the estimated total cost of the cyclic structure is $m * c_1$. Compared with the method of expanding the loop, this method can calculate the QoS of the whole process more quickly and accurately.

Time (T): the response time $T(W)$ of the service process W is m times the response time of the loop body service s .

$$T(W) = m * T(s). \quad (10)$$

Cost (C): the cost $C(W)$ of service process W is m times the cost of loop service s .

$$C(W) = m * C(s). \quad (11)$$

Availability (A): the availability $A(W)$ of service process W is the m -th power of s availability for the loop body.

$$A(W) = A(s)^m. \quad (12)$$

4. Data Preprocessing for Atomic Service

Before service selection, the QoS planning module must know the QoS information of the atomic services in the service portfolio, which should conform to the atomic services trust modeling approach described in Section 3.1. For an atomic service, its QoS capabilities may be manifested at different levels over time. Simply put, multiple executions of an atomic service will necessarily reflect different QoS, just as a Web service will not have the same response time every time. In the face of such a large amount of historical information, it is very important for the accuracy of QoS planning to make a reasonable analysis and estimate the QoS capability of the atomic service in the next execution.

For each atomic service with the same functionality, its ability to provide services is different, which is not only restricted by its characteristics but also shows different QoS characteristics due to the environment it is deployed in and the resources it allocates. Therefore, for each atomic service, it is necessary to estimate the QoS capability of the services it currently provides. This process is called “QoS prediction” of atomic services. The main purpose of QoS prediction of atomic services is to meet the demand after QoS planning by selecting appropriate atomic services as far as possible, that is, to meet the user’s QoS demand at the same time, to avoid the adjustment of the resource layer. A forecast is an estimate of the current QoS capabilities of atomic services that, if accurate, may avoid the additional overhead of resource layer control. If there is no special explanation, this paper takes the QoS prediction of the time dimension as an example for illustration.

4.1. Exponential Smoothing Prediction. The exponential smoothing method requires less data, so it is a simple and practical forecasting method for QoS prediction [25]. The method of exponential smoothing takes the weighted average of a sequence of historical events as a forecast of the future. It is a special case of the weighted moving average method, where we select only one weight, the weight of the nearest observed value. The weights of other data can be calculated automatically and will get smaller over time. The basic model of the exponential smoothing method is as follows.

$$F_{t+1} = \alpha Y_t + (1 - \alpha)F_t. \quad (13)$$

In (13), F_{t+1} is the predicted value of time series in the period $t + 1$, Y_t is the actual value of time series in the period t , F_t is the predicted value of time series in the period t , and α is the smoothing constant ($0 \leq \alpha \leq 1$).

Formula (13) shows that the predicted value for the period $t + 1$ is the weighted average of the actual value for the period t and the predicted value for the period t . In fact, we can show that the predicted value for the exponential smoothing method for any period is also the weighted average of all the historical actual data for the time series, as in $F_4 = \alpha Y_3 + \alpha(1 - \alpha)Y_2 + (1 - \alpha)^2 Y_1$.

Although the exponential smoothing method provides a weighted average of all historical observations, we do not need to store all historical data on the computer for the next period. Once the smoothing constant α is selected, we need only two items of information to calculate the predicted value. The formula shows that if α is given, we only need to know the actual value and the predicted value of the time series in the period of t , namely, Y_t and F_t ; then, we can calculate the predicted value in the period of $t + 1$.

Forecast accuracy. Although any value of α up to 0 to 1 is acceptable, some values of α produce more accurate predictions than others. We rewrite (13) in the following form:

$$F_{t+1} = F_t + \alpha(Y_t - F_t). \quad (14)$$

Therefore, the new forecast (F_{t+1}) is equal to the historical forecast (F_t) plus an adjustment, which is equal to α

times the most recent forecast error ($Y_t - F_t$). In other words, by adjusting the predicted value of the t period and some prediction errors, we can get the predicted value of the $t + 1$ period. If the time series contains a large number of random variations, we tend to use a smaller smoothing constant. The reason for this choice is that many forecast errors are due to random variation, and we do not want to overreact to forecasts and adjust them too quickly. For the time series with small random variation, a larger smoothing exponential constant can be selected. The advantage of this method is that the condition can be changed quickly to adjust the error when the prediction error occurs. For the selection of the most appropriate α , the Mean Square Error (MSE) analysis should be carried out through the analysis of historical data or experiments.

4.2. QoS Prediction Experiment for Random Changes. Figure 2 shows the QoS capability estimation diagram of atomic service based on the exponential smoothing method, where the red broken line on the vertical axis represents the response time of a Web service executed for 50 consecutive times, and the blue asterisk represents the response time of these 50 atomic services. The abscissa represents the number of tests based on the time estimate from the exponential smoothing method. Besides, we use $T(Y_i)$ to represent the actual measured value of the i -th Web service response time, and $T(F_i)$ is used to represent the exponential smoothing estimate of the i -th Web service response time, where $i \in (1, 2, \dots, 50)$.

As can be seen from the figure, the exponential smoothing method has a good prediction effect on the estimation of QoS capability of atomic service and smooths the capability deviation displayed by the service during its execution, which can eliminate interference errors to a certain extent. These errors can be caused by inaccuracies in measurement or by the environment in which the atomic service itself is deployed. The exponential smoothing method can be used to describe to a considerable extent the true level of service quality of the atomic service itself, which is an essential characteristic of the atomic service, as opposed to the performance characteristics that are affected by the environment in which it is deployed and the allocation of resources. Besides, the QoS estimation of exponential smoothing cannot guarantee the accurate prediction of QoS level for the next execution of the atomic service. In essence, it is the weighted average of the historical value of the service on the QoS indicator, which is an improvement of the time average of the trusted QoS modeling method of the atomic service in Section 3.1.

What we need to notice is that the value of each blue asterisk $T(F_i)$ is given by the weighted average of $T(Y_j)$ [$j \in (1, 2, \dots, i - 1)$], which is less than i in the time coordinate. Of course, in practical application, to reduce the storage capacity, the actual value stored in the data center is the actual measured value and the estimated value at the last execution of the Web service which is calculated by (14), rather than recording all measurements of the historical execution time of the atomic service.

FIGURE 2: QoS estimation for random variation.

Linear fitting was conducted between the actual and estimated values of 50 measurements in this experiment, as shown in Figure 3, where the abscissa represents the actual value, the ordinate represents the estimated value, and $(T(Y_j), T(F_i))$ constitutes the coordinate points in the figure. Figure 3(a) represents the original data, and Figure 3(b) represents the results of actual and estimated values sorted from small to large (to reflect the fluctuation range of QoS value). The results show that both the actual value and the estimated value can reflect the response time of the Web service around 100, while the estimated value has a smaller standard variance. The fitting equation in Figure 3(b) is $T(F) = 0.29T(Y) + 71$, and the correlation coefficient $R^2 = 0.97$. R^2 is a real number between 0 and 1, which is used to evaluate the fitting degree of the regression equation. The greater the value, the better the fitting degree.

Figure 4 shows the residual analysis between the 50 measurement results and the estimated results. All the values are evenly distributed between them, belonging to the standard normal shape, which further confirms the correctness of linear fitting in Figure 3. In this experiment, the prediction effect of QoS is not good enough because QoS varies randomly. Nevertheless, the evaluation of atomic service QoS capability by using the exponential smoothing method is relatively accurate. It is essentially a dynamic estimation of the average value of the atomic service QoS capability and a weighted average of the historical measurement value. It explores the essential characteristics of the service. The QoS ability of atomic service will show random fluctuations with the white noise interference. The results of experiments show that the standard deviation of the estimated value 2.75 is significantly smaller than that of the actual value 9.19 (shown in Figures 2 and 3). The estimated value is embedded into small volatilities around real the QoS ability value 100. This means that the estimated value based on exponential smoothing method could reflect the nature of the QoS ability of atomic service. Therefore, this approach

is consistent with the QoS modeling for atomic services in Section 3.1.

4.3. Prediction of QoS with a Linear Trend. Studies have shown that the QoS supply level of a service entity is related to the load of the service node where the service entity is located. For example, in [26], through a large number of experimental tests on database services, it is found that the average service response time of the service is correlated with the CPU load of the service nodes. This relationship can be intuitively understood as follows: in general, a lower load on the service node means that the service node has fewer service invocation requests in the local service queue, and the service request of the service entity will be processed faster. Furthermore, when a load of service nodes is low, the task scheduling overhead caused by service process switching and file association and the interaction between processes of different service entities will be reduced, which will also improve the response time and availability of service entities.

The time response of an atomic service may change with the node load, and this change has a linear trend. In other words, in a certain time series, the response time of an atomic service will increase with the increase of node load and decrease with the decrease of node load. Although there is not necessarily a strictly linear relationship between them, such a monotonous trend does exist.

Based on the theory of the relationship between response time and node load, we conducted relevant tests again, as shown in Figure 5. For QoS estimation with a rising trend, the exponential smoothing method can also have a good tracking effect, and the estimation of the actual measured value is accurate, which fully reflects the time response characteristics of the atomic service. However, for a real system, the response time that increases as the node load increases should be a value that varies more slowly and is closer to that shown in Section 4.2.

Figure 6 shows the fitting between the actual and estimated values of QoS with an upward trend. Figure 6(a) shows the raw data, and Figure 6(b) shows the actual and estimated values sorted from smallest to largest. The results show that the QoS estimation with the linear trend is more accurate than that with the random change, and can reflect the changing trend of QoS. The R^2 of the linear equation fitted in Figures 6(a) and 6(b) is greater than 0.95, and the correlation coefficient is greater than 0.9. The RMSE value and SSE value of fitting results in Figure 6(a) are 14.76 and 10460, respectively, while the RMSE value and SSE value of fitting results in Figure 6(b) are 7.875 and 2977, respectively. This indicates that there is a strong correlation between the actual value and the estimated value and that the ratio between $T(F)$ and $T(Y)$ is close to 1, indicating that the estimated value is very accurate in predicting the actual measured value.

The above two experiments fully prove that the prediction of atomic service QoS information by the exponential smoothing method is accurate and feasible. The estimated data obtained by the exponential smoothing

(a)

(b)

FIGURE 3: QoS fitting of random variation. (a) The actual value. (b) The actual value.

method is a group of QoS information with relatively slow changes, which can fully reflect the reliable QoS level of the atomic service. Moreover, the algorithm is simple and easy to operate, and the consumption of system resources is small. Therefore, it can be fully applied to the preprocessing of QoS data of atomic service for reliable QoS planning.

5. Service Selection Based on QoS Planning

By preprocessing the history data of atomic service, the multidimensional trusted QoS information of each atomic service could be obtained. The typical information would be applied to trusted QoS planning, which is equivalent to decomposing the process QoS requirements of the application into QoS requirements for each service in this composite service.

QoS planning is the decomposition of the QoS requirements of the application process proposed by the user into the QoS requirements that should be satisfied by each service that makes up the process (the process is shown in Figure 7). Based on the results of QoS planning, the system selects the atomic services for each service that can guarantee the QoS requirements of that service according to a certain service selection strategy. The selection of appropriate atomic services to form a composite service is the key to meeting the QoS requirements of users. For verification, this paper uses three dimensions of the QoS metrics for illustration: time (T), cost (C), and availability (A). Based on the results of QoS planning, three service selection strategies are proposed by the author.

FIGURE 4: Residual analysis diagram.

5.1. Time Preference Algorithm. For each service in the composite service, the atomic service with the shortest response time (the value after data preprocessing) is selected to complete the service according to the requirements of the time dimension in the results of QoS planning (average time complexity is $O(n^2)$). This method maximally satisfies the user's requirements for the QoS time dimension, because for each atomic service, even if the QoS capability is estimated using data preprocessing, there is still no precise guarantee that the QoS characteristics of the atomic service will be the

FIGURE 5: Estimates of QoS with an upward trend.

(a)

(b)

FIGURE 6: QoS fitting with an upward trend. (a) The actual value. (b) The actual value.

estimated value at the next execution. In the case when the time characteristics of the atomic service are significantly different from the estimated value, the user's requirements for the time dimension could be satisfied to the maximum extent with the time preference policy, because it leaves the maximum amount of redundancy in time.

5.2. Cost/Availability (C/A) Preference Algorithm. The cost/availability (C/A) preference algorithm is a service selection algorithm based on QoS planning. The basic idea, shown as follows, is to satisfy the user's demand in the time dimension at the lowest possible cost. (Algorithm 1).

Algorithm description:

FIGURE 7: Flowchart of the QoS planning algorithm.

- (1) Based on the results of QoS planning, for each service, the atomic services that satisfy its real-time requirements T_{pj} and form a sequence A_j are selected. Note that the atomic services in the sequence are out of order at this point.
- (2) For each atomic service in the sequence A_j , it is backed up to form a service group until the availability meets the QoS planning requirements A_{pj} (if the availability of the atomic service already meets the requirements, no backup is needed); the cost of the atomic service group is $C_{ji} \cdot k$, and k is the number of atomic services in the service group.
- (3) Sort all the atomic service groups in increasing order of cost/availability (C/A), with the services of different functions in one column, to obtain the order A_j .
- (4) Select the first service group from the sequence A_j , i.e., the service S_j^1 with the smallest C/A, to form the service flow.

This service selection strategy allows users to achieve the required QoS requirements for each dimension with a minimal cost. The average time complexity is $O(\sum s_{ji} + B \cdot N_A + N_A^2)$, where B is the average backup times of A_j and N_A is number of atomic services in A_j . However, for atomic services whose real-time QoS estimates deviate significantly from the actual measurements, the real-time performance may not be satisfied.

5.3. Euclidean Distance Preference Algorithm. The composite service selected by the Euclidean distance preference algorithm is the closest to the comprehensive quality of service proposed by the user [27]. Besides, the required quality of service for each service after QoS planning is denoted as $\text{QoS}(s) = (T, C, A)$, which represents the conditions that the service should satisfy in three dimensions: real time, cost, and availability.

For example, there are n atomic services s_1, s_2, s_3 , $\text{QoS}(s_1) = (T_1, C_1, A_1)$, $\text{QoS}(s_2) = (T_2, C_2, A_2)$, $\text{QoS}(s_3) = (T_3, C_3, A_3)$, so that the services of the same function could be described as a matrix:

$$Q = \begin{pmatrix} T_1 & C_1 & A_1 \\ T_2 & C_2 & A_2 \\ \dots & \dots & \dots \\ T_n & C_n & A_n \end{pmatrix}. \quad (15)$$

In terms of service quality criteria, there are quality standards where higher values indicate better quality, while there are quality standards where lower values indicate better quality [28]. The former are called positive-quality criteria, such as availability, and the latter are called negative-quality criteria, such as execution time. In addition, in order to prevent the value of a quality criterion from being exceedingly large to influence the final result, it is necessary to concentrate on the values of all quality criteria between $[0, 1]$.

Input:

- (1) Flowchart
- (2) The flowchart contains m tasks, denoted as t_j , ($j \in 1, 2, \dots, m$).
- (3) Each task has n_j service with the same function to complete the task, denoted as s_{ji} , ($i \in 1, 2, \dots, n_j$).
- (4) Each atomic service has three QoS properties: T (real time), A (availability), and C (cost). Each atomic service has its own real-time interval $[T_{ji1}, T_{ji2}]$ and estimated execution time $\overline{T}_{ji}$.
- (5) Each service would have real-time requirements T_{pj} and availability requirements A_{pj} after QoS planning, $j \in 1, 2, \dots, m$.

ALGORITHM 1

For negative service quality, we use (16) for processing; for positive service quality, (17) is used for processing.

$$V_{i,j} = \begin{cases} \frac{Q_j^{\max} - Q_{i,j}}{Q_j^{\max} - Q_j^{\min}}, & \text{if } Q_j^{\max} - Q_j^{\min} \neq 0, \\ 1, & \text{if } Q_j^{\max} - Q_j^{\min} = 0. \end{cases} \quad (16)$$

$$V_{i,j} = \begin{cases} \frac{Q_{i,j} - Q_j^{\min}}{Q_j^{\max} - Q_j^{\min}}, & \text{if } Q_j^{\max} - Q_j^{\min} \neq 0, \\ 1, & \text{if } Q_j^{\max} - Q_j^{\min} = 0. \end{cases} \quad (17)$$

In the above equation, atomic services s_{ij} with the same function constitute service s_j , $Q_j^{\max}$ denotes the maximum value of service s_j on dimension Q , $Q_j^{\min}$ denotes the minimum value of all atomic services s_j on dimension Q , and $Q_{i,j}$ denotes the value of atomic services on dimension Q , where Q denotes the dimension of service quality, e.g., real time T , cost C , and availability A .

The matrix in (15) is processed according to (16) and (17) to obtain

$$Q^* = \begin{pmatrix} V_{11} & V_{12} & V_{13} \\ V_{21} & V_{22} & V_{23} \\ \dots & \dots & \dots \\ V_{n1} & V_{n2} & V_{n3} \end{pmatrix} = \begin{pmatrix} Q_1 \\ Q_2 \\ \dots \\ Q_n \end{pmatrix}. \quad (18)$$

We calculate the demand $QoS(s) = (T, C, A)$ after QoS planning using the above formula also for the comprehensive quality to obtain the comprehensive quality value q . q and Q_1 are considered as two points, and then the Euclidean distance is used to calculate the deviation between q and Q_1 . The value of each service quality of q after processing could be set as (q_1, q_2, q_3) , and then the deviation between q and Q_j is

$$d(q, Q_j) = \sqrt{\sum_{i=1}^3 (q_i - V_{ji})^2}. \quad (19)$$

Then, the smallest one in $d(q, Q_j)$ is the service that is closest to the comprehensive service quality proposed by the user. The average time complexity is $O(n \cdot (m + n + 1))$, where n is the number of atomic services and m is the number of QoS dimensions.

5.4. Simulation Experiment. Figure 8 shows the graph of the results after using trusted QoS planning. In the following, the three algorithms are compared in terms of the total time, total cost, and Euclidean distance metrics, respectively. The horizontal axis indicates the number of simulation experiments and the vertical axis indicates the indicator we examined.

The three algorithms are examined in terms of total time, which is shown in Figure 9.

From the figure, it could be seen that the time (T) preference algorithm has the shortest total time, which is due to the fact that the algorithm uses only the time metric as the basis for service selection, which maximizes the satisfaction of the time requirement. The random selection algorithm and the cost/availability (C/A) preference algorithm exhibit random fluctuations around the expected value (1400) because they do not consider the time metric. The Euclidean distance preference algorithm performs better than the random selection algorithm and the cost/availability (C/A) preference algorithm in terms of time metrics because it considers time, cost, and availability together, but not as well as the time (T) preference algorithm.

The three algorithms are examined in terms of total cost, which is as shown in Figure 10.

From Figure 10, it could be seen that the cost/availability (C/A) preference algorithm is the least costly. This is due to the fact that the algorithm uses cost (C) as a metric for measuring service selection and therefore is able to maximize the cost satisfaction.

The random selection algorithm and the time (T) preference algorithm exhibit random fluctuations around the expected value (250) because the metric of cost is random in both. The Euclidean distance preference algorithm performs better than the random selection algorithm and the time (T) preference algorithm in terms of cost metrics because it considers time, cost, and availability together, but not as well as the cost/availability (C/A) preference algorithm.

The three algorithms are examined in terms of similarity to the expected value (Euclidean distance), as shown in Figure 11.

As can be seen from Figure 11, the Euclidean distance preference algorithm has the best performance in this metric. The algorithm considers a combination of time (T), cost (C), and availability (A). The random selection algorithm has the worst performance in all three metrics because they are chosen randomly. The time (T) preference algorithm performs better than the random selection algorithm

FIGURE 8: Planning result for process QoS.

FIGURE 9: Comparison of the total time of the four algorithms.

FIGURE 10: Comparison of the total cost of the four algorithms.

FIGURE 11: Comparison of the four algorithms with the expected value of the Euclidean distance.

in terms of the Euclidean distance of the expected value because it takes time (T) into account. The cost/availability (C/A) preference algorithm outperforms the time (T) preference algorithm in terms of Euclidean distance because it considers both cost (C) and availability (A), but it is inferior to the Euclidean distance preference algorithm.

In summary, the Euclidean distance preference algorithm integrates the three indicators into consideration and has a more balanced and excellent characteristic. In practice, different algorithms can be chosen according to different needs.

5.5. Service Selection Based on Heuristic Algorithms. When the computational power is sufficient, service selection can be achieved using heuristic algorithms without going through QoS planning. Service selection for composite service is an NP-hard problem, and this paper propose the memetic algorithm (MA) by combining genetic algorithm (GA) with local search to solve this class of problems [29, 30].

The algorithmic framework of this paper is listed in Algorithm 2. Firstly, various parameters of the genetic algorithm, the flow of the composite service, and the QoS of each atomic service are required to be input. Then the population P is generated by the initialization function `InitialPopulation()`, and next it enters a loop until the number of iterations reaches the set maximum number of iterations. In the loop, firstly, `Selection()` is used to select the parent population to participate in crossover and mutation in a roulette manner, then `GeneticOperation()` is used to perform crossover and mutation operation. `LocalSearch()` is a further search after the crossover and mutation operation so as to find the local optimum, and then `UpdatePopulation()` is used to update the population and get a better

Input: The maximum number of iterations: N_{Imax} . The size of population: S_p . The size of the mating pool: S_{MP} . The size of tournament: S_T . The probability of crossover: P_C . The probability of mutation: P_M . The process of composite service and the QoS of each atomic service.

Output: The selected atomic services and the QoS of composite service.

Steps:

- (1) Initialize $P \leftarrow \text{InitialPopulation}(S_p)$;
- (2) **for** $n = 1$; $n < N_{\text{Imax}} + 1$; $n++$
- (3) $P_{\text{parent}} \leftarrow \text{Selection}(P, S_{\text{MP}}, S_T)$;
- (4) $P_{\text{child}} \leftarrow \text{GeneticOperation}(P_{\text{parent}}, P_C, P_M)$;
- (5) $P_{\text{newchild}} \leftarrow \text{LocalSearch}(P_{\text{child}})$;
- (6) $P \leftarrow \text{UpdatePopulation}(P, P_{\text{newchild}})$;
- (7) **end for**
- (8) **return** P, QoSmax

ALGORITHM 2: MA of service selection.

Input: The chromosome P_{child} . The process of composite service and the QoS of each atomic service.

Output: New chromosome P_{newchild} .

Steps:

- (1) Initialize $P_{\text{newchild}} = P_{\text{child}}$;
- (2) **for** $i = 1$; $i < N_{\text{AbstS}} + 1$; $i++$
- (3) **for** $j = 1$; $j < N_{\text{AtomS}}(i) + 1$; $j++$
- (4) $P_{\text{child}}(i) = S(j)$; //(A atomic service differs from P_{Nchild} ' gene)
- (5) **if** $\text{QoS}(P_{\text{child}}) > \text{QoS}(P_{\text{newchild}})$ //(find an atomic service that improve QoS)
- (6) $P_{\text{newchild}} = P_{\text{child}}$;
- (7) **end if**
- (8) **end for**
- (9) $P_{\text{child}}(i) = P_{\text{newchild}}(i)$;
- (10) **end for**
- (11) **return** P_{newchild}

ALGORITHM 3: Algorithm of local search.

FIGURE 12: Algorithm comparison based on success rate of QoS assurance.

chromosome population. Finally, the result of the calculation is output.

The local-search strategy is given by Algorithm 3 in order to accelerate the convergence. N_{AbstS} in Algorithm 3

denotes the number of abstract services in the composite service, and $N_{\text{AtomS}}(i)$ denotes the number of atomic services in abstract service i . We traverse each gene of the chromosome and determine whether replacing it with

another atomic service improves the QoS. If changing the atomic service of a gene can improve the QoS, the new gene is accepted so that a local optimum is achieved. If QoS could be improved by changing atomic service of the typical gene, this gene could be accepted in order to reach optimal results. The average time complexity of MA algorithm is $O(N_{\text{Imax}} \sum_{N_{\text{Absts}}} N_{\text{AtomS}}(i))$.

The simulation experiments are performed on the composite service flow in Figure 8, using random selection of atomic services and MA selection of atomic services for each experiment, for a total of 100 tests. We set $N_{\text{Imax}} = 1000$, $S_P = 100$, $S_{MP} = 100$, $S_T = 2$, $P_C = 0.1$, and $P_M = 0.9$. Figure 12 shows the number of successes of the above two methods in meeting the QoS requirements proposed by the application in 100 experiments, respectively, where the horizontal coordinate represents the number of experiments n , $n \in 1, 2, \dots, 100$, and the vertical coordinate represents the number of times m the QoS requirements of the composite service are satisfied in n experiments, $m \in 1, 2, \dots, n$. As can be seen from the figure, the number of successes after MA selection of atomic services is significantly greater than the number of successes for random selection of atomic services. Among the 100 experiments conducted, the user's requirements can be satisfied up to 90% because of the QoS of the composite service after MA-selected atomic service, while the randomly selected service has less than 50% chance of success in terms of the QoS requirements proposed by the user, although the user's functional requirements could be satisfied.

6. Conclusion and Future Work

Web composite service technology aims at solving the problem of effective integration of functionally diverse Web service resources on the Internet, and users' multifaceted application requirements could be satisfied by constructing functionally complex and superior composite services. The huge number of candidate services with similar functional attributes and different nonfunctional attributes will increase the complexity of composite service and lead to the problem of Web composite service as NP-hard problem. In the complex network environment, it is difficult for the traditional QoS-based composite service approach to guarantee that the constructed combination solution can meet the user requirements because it cannot measure the trustworthiness of Web services. In this paper, we mainly focus on how to solve the problem of trustworthy QoS guarantee for composite service in the highly complex, dynamic, and untrustworthy Internet environment. We design an operational mechanism to guarantee the trustworthiness requirements of the composite services. The various task segments in the composite services are completed by virtual services, and the quality of the composite services is ensured by QoS planning of the upper layer applications, prediction of the QoS capability of atomic services, and implementation of various service selection algorithms to finally meet the abstract service trustworthiness requirements. Future work will optimize (1) the selection of services with identical functions but differing

interfaces and (2) the impact of interatomic service correlations on composite services.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (71501153), the Innovation Capability Support Project of Shaanxi Province of China (2021KRM135), the Research Fund of Grand Theory and Practical Problem in Philosophy and Social Science of Shaanxi Province of China (2021ND0221), the Research Fund of the Education Department of Shaanxi Province of China (20JG020), and the Natural Science Foundation of Shaanxi Province of China (2019JM-572).

References

- [1] A. Muñoz and E. B. Fernandez, "TPM: a pattern for an architecture for trusted computing," in *Proceedings of the European Conference on Pattern Languages of Programs*, pp. 1–8, Kloster Irsee, Bavaria, Germany, July 2020.
- [2] A. Nageswaran, A. Revathi, and R. Kaladevi, "Adaptive video streaming with multidimensional quality of service," *European Journal of Molecular & Clinical Medicine*, vol. 7, no. 8, pp. 2098–2105, 2020.
- [3] C. Wang, H. Ma, G. Chen, S. Hartmann, and J. Branke, "Robustness estimation and optimisation for semantic web service composition with stochastic service failures," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 99, pp. 1–16, 2020.
- [4] F. Ibrahim and E. E. Hemayed, "Trusted cloud computing architectures for infrastructure as a service: survey and systematic literature review," *Computers & Security*, vol. 82, pp. 196–226, 2019.
- [5] E. Gelenbe, J. Domanska, P. Fröhlich, M. P. Nowak, and S. Nowak, "Self-aware networks that optimize security, QoS, and energy," *Proceedings of the IEEE*, vol. 108, no. 7, pp. 1150–1167, 2020.
- [6] L. Chuang, W. Yuanzhuo, and T. Liqin, "Development of trusted network and challenges it faces," *ZTE Communications*, vol. 6, no. 1, pp. 13–17, 2020.
- [7] H. Gao, D. Chu, Y. Duan, and Y. Yin, "Probabilistic model checking-based service selection method for business process modeling," *International Journal of Software Engineering and Knowledge Engineering*, vol. 27, no. 6, pp. 897–923, 2017.
- [8] L. Wang, "Architecture-based reliability-sensitive criticality measure for fault-tolerance cloud applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 11, pp. 2408–2421, 2019.
- [9] M. Al-Khafaji, T. Baker, M. Asim et al., "COMITMENT: a fog computing trust management approach," *Journal of Parallel and Distributed Computing*, vol. 137, pp. 1–16, 2020.

- [10] L. Zeng, B. Benatallah, A. Ngu et al., "QoS-aware middleware for web services composition," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 311–327, 2004.
- [11] H. Issa, C. Assi, M. Debbabi, and S. Ray, "QoS-aware middleware for web services composition: a qualitative approach," *Enterprise Information Systems*, vol. 3, no. 4, pp. 449–470, 2009.
- [12] R. Alhajj and J. Rokne, *Encyclopedia of Social Network Analysis and Mining*, Springer Publishing Company, USA, 2014.
- [13] M. Moghaddam and J. G. Davis, *Service Selection in Web Service Composition: A Comparative Review of Existing Approaches*, Web Services Foundations, New York, NY, USA, 2014.
- [14] W. Zhang, Y. Yang, S. Zhang, D. Yu, and Y. Li, "Correlation-aware manufacturing service composition model using an extended flower pollination algorithm," *International Journal of Production Research*, vol. 56, no. 14, pp. 4676–4691, 2018.
- [15] D. Artz and Y. Gil, "A survey of trust in computer science and the semantic web," *Journal of Web Semantics*, vol. 5, no. 2, pp. 58–71, 2007.
- [16] S. Wang, Q. Sun, H. Zou, and F. Yang, "Reputation measure approach of web service for service selection," *IET Software*, vol. 5, no. 5, pp. 466–473, 2011.
- [17] D. Ardagna and B. Pernici, "Global and local QoS guarantee in web service selection," in *Proceedings of the international Conference on Business Process Management*, pp. 32–46, Nancy, France, May 2005.
- [18] S. X. Sun and J. Zhao, "A decomposition-based approach for service composition with global QoS guarantees," *Information Sciences*, vol. 199, pp. 138–153, 2012.
- [19] C. Surianarayanan, G. Ganapathy, and M. S. Ramasamy, "An approach for selecting best available services through a new method of decomposing QoS constraints," *Service Oriented Computing and Applications*, vol. 9, no. 2, pp. 107–138, 2015.
- [20] V. Gabrel, M. Manouvrier, and C. Murat, "Optimal and automatic transactional web service composition with dependency graph and 0-1 linear programming," in *Proceedings of the International Conference on Service-Oriented Computing*, pp. 108–122, Paris, France, November 2014.
- [21] A. J. S. Cardoso, "Quality of service and semantic composition of workflows," Doctoral Dissertation, University of Georgia, Athens, GA, USA, 2002.
- [22] H. Elshaafi and D. Botvich, "Optimisation-based collaborative determination of component trustworthiness in service compositions," *Security and Communication Networks*, vol. 9, no. 6, pp. 513–527, 2016.
- [23] B. Benatallah, M. Dumas, Q. Shen, and A. H. Ngu, "Declarative composition and peer-to-peer provisioning of dynamic web services," in *Proceedings of the 18th International Conference on Data Engineering*, pp. 297–308, San Jose, CA, USA, February 2002.
- [24] M. P. Papazoglou and D. Georgakopoulos, "Introduction," *Communications of the ACM*, vol. 46, no. 10, pp. 24–28, 2003.
- [25] S. Mahajan, L.-J. Chen, and T.-C. Tsai, "Short-term PM2.5 forecasting using exponential smoothing method: a comparative analysis," *Sensors*, vol. 18, no. 10, p. 3223, 2018.
- [26] S. Ranjan and E. Knightly, "High-performance resource allocation and request redirection algorithms for web clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 9, pp. 1186–1200, 2008.
- [27] S. P. Patel and S. H. Upadhyay, "Euclidean distance based feature ranking and subset selection for bearing fault diagnosis," *Expert Systems with Applications*, vol. 154, Article ID 113400, 2020.
- [28] Y. Liu, A. H. Ngu, and L. Z. Zeng, "QoS computation and policing in dynamic web service selection," in *Proceedings of the 13th international World Wide Web Conference on Alternate Track Papers & Posters*, pp. 66–73, New York, NY, USA, May 2004.
- [29] X. Fu, P. Pace, G. Aloï, L. Yang, and G. Fortino, "Topology optimization against cascading failures on wireless sensor networks using a memetic algorithm," *Computer Networks*, vol. 177, Article ID 107327, 2020.
- [30] B. Cao, W. Zhang, X. Wang, J. Zhao, Y. Gu, and Y. Zhang, "A memetic algorithm based on two_Arch2 for multi-depot heterogeneous-vehicle capacitated Arc routing problem," *Swarm and Evolutionary Computation*, vol. 63, Article ID 100864, 2021.