Hindawi
Mobile Information Systems
Volume 2022, Article ID 4277909, 13 pages
https://doi.org/10.1155/2022/4277909

Research Article

Web Service Composition Optimization with the Improved Fireworks Algorithm

Bo Jiang, Yanbin Qin, Junchen Yang, Hang Li, Liuhai Wang, and Jiale Wang

School of Computer Science and Information Engineering, Zhejiang Gongshang University, Hangzhou 310018, China

Correspondence should be addressed to Bo Jiang; nancybjiang@zjgsu.edu.cn and Jiale Wang; wjl8026@zjgsu.edu.cn

Received 21 October 2021; Revised 20 January 2022; Accepted 1 March 2022; Published 12 March 2022

Academic Editor: Claudio Agostino Ardagna

Even though the number of services is increasing, a single service can just complete simple tasks. In the face of complex tasks, we require composite multiple services to complete them. For the purpose of improving the efficiency of web service composition, we propose a service composition approach based on an improved fireworks algorithm (FWA++). First, we use the strategy of random selection to keep $N-1$ individuals for the next generation, and the purpose is to speed up the convergence speed of the FWA++ and enhance the search ability. Second, we randomize the total number of sparks and maximum amplitude of sparks for each generation. In this way, the search ability and the ability to jump out of the local optimal solution are dynamically balanced throughout the execution of the algorithm. Our experimental results show that compared with other existing approaches, the approach proposed in this paper is more efficient and stable.

1. Introduction

With the emergence of a large number of web services, enterprises and individuals can select multiple services to build enterprise applications and software systems through the technology of service composition [1, 2], which is called Service-Oriented Computing (SOC). Service composition is one of the core technologies of SOC, which flexibly aggregates the required resources and realizes service reuse [3]. However, with the convergence of a large number of web services with the same function, quality of service (QoS) has increasingly become an essential factor that needs to be considered in the process of selecting services from these function-equivalent services for composition. Therefore, many QoS-based web service composition methods have been proposed.

Heuristic and metaheuristic algorithms are often used to solve the web service composition problem. Metaheuristic algorithms have the characteristics of good optimization effect and less time consumption. With the attention of researchers at home and abroad, more and more metaheuristic algorithms have been proposed, including particle swarm optimization (PSO) [4], artificial bee colony (ABC) [5], Bacterial Foraging Optimization (BFO) [6], fireworks algorithm (FWA) [7], Fruit Fly Optimization (FOA) [8], Moth Search Algorithm (MSA) [9], Harris Hawks Optimization (HHO) [10], Slime Mould Algorithm (SMA) [11], and Colony Predation Algorithm (CPA) [12].

Among them, different variants of FWA have been proposed to improve the performance of the traditional FWA. Zheng et al. [13] proposed an Enhanced Fireworks Algorithm (EFWA); in their work, four strategies were utilized to improve the original FWA. First, the minimum explosion amplitude parameter was used to avoid the case of 0 amplitude. Second, the generation strategy of explosion sparks was modified to enhance explosion ability. Third, the generation strategy of Gaussian sparks and mapping rules were modified to avoid the degradation of optimization performance caused by the objective function being far from the origin. Fourth, a random selection strategy was adopted to reduce the algorithm's running time. However, the explosion amplitude is static during the execution of the algorithm, which limits the scope of application of the algorithm. Zheng et al. [14] and Li et al. [15] proposed the Dynamic Search

Fireworks Algorithm (dynFWA) and the Adaptive Fireworks Algorithm (AFWA). The two algorithms were enhanced on the basis of the EFWA algorithm and improved the setting of explosion amplitude so that the explosion amplitude can be adaptive in the whole algorithm. Thus, the algorithm can adapt to different optimization objectives and search stages. The IFWA algorithm proposed by Zhang et al. [16] was improved on the aspect of optimization strategies. The strategy has stronger robustness and applicability by using the optimal fireworks to generate sparks instead of the explosion amplitude adjustment method of dynFWA and AFWA. However, the optimal fireworks to mutate reduces the diversity of the population, and the elite selection strategy makes the fireworks population close to the local optimal position. Yu et al. [17] proposed another elite strategy based on EFWA. Each firework and the sparks it produces are used to calculate a gradient-like vector, and then use it to estimate the convergence point, and may replace the worst firework individuals in the next generation.

In addition to improving the operator, the hybrid FWA combined with other metaheuristic algorithms is also an important research direction. Zheng et al. [18] combined the differential evolution (DE) algorithm with the FWA. A feasible solution is generated by the DE operator to the selected individual. If the fitness value of the feasible solution is better than the original individual, the original solution should be replaced. Zhang et al. [19] introduced the biogeographic optimization algorithm (BBO) into the EFWA to form a new algorithm (BBO-FWA), in which BBO provides an idea of cross-migration of firework individuals according to fitness value, and the lower the fitness value, the higher the probability of cross-migration.

What is more, FWA has been widely used in a wide range of real-world problems. Bolaji et al. [20] used FWA to train the parameters of the feedforward neural network and applied it to the classification problem. Zare et al. [21] introduced two effective cross-generation mutation operators into FWA to solve discrete and multiconstrained problems. Furthermore, this method was applied to solve the problem of multiregional economic dispatch. For more information about the applications of the FWA, please refer to the following literature [22, 23].

Although the above research has improved the FWA's performance, according to the no free lunch theorem [24], there is no specific metaheuristic algorithm that can positively affect various types of optimization problems. So, it is encouraged to improve and apply the FWA to solve the optimization problems. In addition, the balance between the search ability and the ability to jump out of the local optimum in the FWA has not been well solved. Therefore, we propose the improved fireworks algorithm (FWA++) to solve the optimization problem of service composition. First, we cluster the services with the same function. A WDSL document records a lot of information related to the service, so we extract the service name, port type, information, document, and operation from it and transform them as an embedding. Then, the k-means algorithm is used to cluster multiple services which are transformed as embeddings. Second, we improve the original FWA in two aspects: (1) The selection strategy in the original FWA requires a lot of computation time, so we randomly keep $N - 1$ individuals to the next generation under the premise of ensuring the optimization direction. It reduces computation time. (2) The algorithm falling into the local optimum was avoided, and strong search ability was kept. In each generation, we randomize the total number of sparks and maximum amplitude of sparks. Finally, we apply the FWA++ to service composition, and the results show that our algorithm is effective.

Our main contributions are as follows:

(i) In order to speed up the convergence speed of the FWA++ and enhance the local search ability, we use the strategy of random selection to keep individuals for the next generation

(ii) For each generation, randomize the total number of sparks and maximum amplitude of sparks. In this way, the search ability and the ability to jump out of the local optimal solution are dynamically balanced throughout the execution of the algorithm

(iii) We experimented with our approach in a real-world data set which includes 9 QoS attribute values of 2500 real web services. The results show that compared with several existing approaches, the performance of our proposed approach is better

The rest of this paper is organized as follows: Section 2 summarizes the related work of web service composition. Section 3 presents mathematical modeling for the web service composition problem and a diagram to illustrate the process of service composition. Section 4 introduces the framework of approach and design details. Section 5 introduces simulation experiments and performance evaluation. Section 6 reviews and summarizes this paper.

2. Related Work

In this section, we will refer to some literature to describe the related work of service composition. Generally, to solve the optimization problems of service composition, heuristic and metaheuristic algorithms are mainly used. Nevertheless, the metaheuristic algorithms are the essential solution, so we will focus on them in this section.

The heuristic algorithms are constructed through the experience of specific optimization problems [25]. Klein et al. [26] presented a method based on hill climbing to find the best solution. In this method, the search space is limited. So, the algorithm's time complexity is much less than that of the linear problem when finding an optimal solution. Liu et al. [27] proposed a service composition method based on a branch constraint execution plan. The algorithm is divided into two stages: in the first stage, the composite service state is transformed into a state transition graph; then, the dynamic process of composite service execution can be analyzed. The second stage uses the web service execution language to find the best solution. Lin et al. [28] presented a relaxable QoS-based service composition approach. In this approach, the optimal solution is subject to local and global

constraints. Although the heuristic algorithm can get an approximate solution in a reasonable time and data scale, the designs of the heuristic algorithms mainly depend on the experience of specific optimization problems, so this limits its scope of application. Moreover, when dealing with the problem of large-scale data, the effect is often not guaranteed.

The metaheuristic algorithms are also approximation algorithms. They are no longer designed for specific optimization problems and have the characteristics of a good optimization effect and less time consumption. Many scholars used metaheuristics to solve service composition problems. Canfora et al. [29] tackled the service composition problem by the genetic algorithm (GA). Although GA is slower than integer programming, it is an effective method to handle service composition optimization. Furthermore, Tang and Ai [30] proposed a hybrid genetic algorithm for the optimal web service composition problem, which performs better than other algorithms. However, when service-oriented applications are complex, GA is not suitable. Ludwig [31] addressed the problem of service composition by introducing particle swarm optimization (PSO). The results show that it performs very well. However, the problem of premature convergence of PSO needs to be solved.

Chandra and Niyogi [32] proposed a modified artificial bee colony (mABC) algorithm. In this algorithm, a chaotic-based opposition learning method is used to initialize the population, and differential evolution (DE) is used to enhance exploitation. mABC has robust scalability and high convergence speed. However, service composition is a discrete optimization problem; mABC may fall into the local optimum. Xu et al. [33] proposed an approach based on artificial bee colony (ABC) algorithms for the service composition problem. In this approach, the author improved the neighborhood search of the ABC algorithm to adapt to the discretization of service composition. At the same time, three algorithms are proposed to maintain the performance and simplicity of the approach. The results show that this approach has high accuracy and avoids local optimization. However, it is time-consuming when this approach replaces multiple component services at the same time.

Li et al. [34] introduced an elite evolutionary strategy (EES). Furthermore, it was applied to HHO to improve the convergence speed and capacity of jumping out of the local optimum. Moreover, a hybrid algorithm that combined EES and HHO was presented. This algorithm has a fast convergence speed and strong robustness. However, this approach may fall into the local optimum because service composition is a discrete optimization problem. Li et al. [35] presented a novel approach CHHO to find an optimal solution by incorporating logical chaotic sequence into the Harris Hawks Optimization (HHO) algorithm. In this approach, the neighborhood relations of concrete services were constructed to form a continuous space, which avoids CHHO falling into the local optimum. Chaotic sequences have ergodic and chaotic features, which help CHHO improve the capacity to jump out of local optimization. In large-scale scenarios, CHHO has less computation time. However, CHHO's performance will not be good when the QoS attributes are not independent.

3. Problem Statement

Before proposing our approach, we will provide a diagram to illustrate the process of service composition and the mathematical modeling for the web service composition problem.

Figure 1 shows the process of service composition by using the integer coding method and the sequential combination pattern. First, input a composite service $S = \{T_1, T_2, \cdots, T_n\}$. Here, each task corresponds to an abstract service, for example, T_1 corresponds to abstract service S_1. And each abstract service has m number of concrete services; for example, S_1 has m number of concrete services with the same function. Second, select n number of concrete services from the corresponding abstract service as a composite service. In theory, there are $\prod_{i=1}^{n} m_i$ number of composite services. Calculate the QoS value of the composite services to obtain an optimal value that reaches the minimal objective value of Equation (3). Finally, output an optimal composite service.

In this mathematical modeling, response time and price are negative attributes; the smaller the better. Reputation and availability are positive attributes; the larger the better. Therefore, in order to unify metrics and calculations, it is necessary to normalize the QoS values. The normalization methods of QoS value are defined as follows:

$$q_i = \begin{cases} \dfrac{q_i - q_i^{\min}}{q_i^{\max} - q_i^{\min}}, & q_i^{\min} \neq q_i^{\max}, \\ 1, & q_i^{\min} = q_i^{\max}, \end{cases} \tag{1}$$

$$q_i = \begin{cases} \dfrac{q_i^{\max} - q_i}{q_i^{\max} - q_i^{\min}}, & q_i^{\min} \neq q_i^{\max}, \\ 1, & q_i^{\min} = q_i^{\max}. \end{cases} \tag{2}$$

$q_i^{\max}$ and $q_i^{\min}$ are the maximum value and minimum value of the ith QoS attribute of composite services, respectively; Equation (1) is used to normalize the negative attributes, such as response time and price; Equation (2) is used to normalize the positive attributes, such as reputation and availability.

We model the service composition problem as a minimization problem, and the optimization model is formulated as follows:

$$\text{minimize } \frac{\sum_{k=1}^{r} \left(q_k^{\text{agg}} \times w_k\right)}{\sum_{j=1}^{l} \left(q_j^{\text{agg}} \times w_j\right)}. \tag{3}$$

In a composite service, r and l are the numbers of negative and positive QoS attributes of each service, respectively. w_k and w_j represent the weights of the kth negative and jth positive attributes, respectively. q_k^{agg} is the sum of value of the k th negative attribute of each service, and q_j^{agg} is the sum of value of the jth positive attribute of each service.

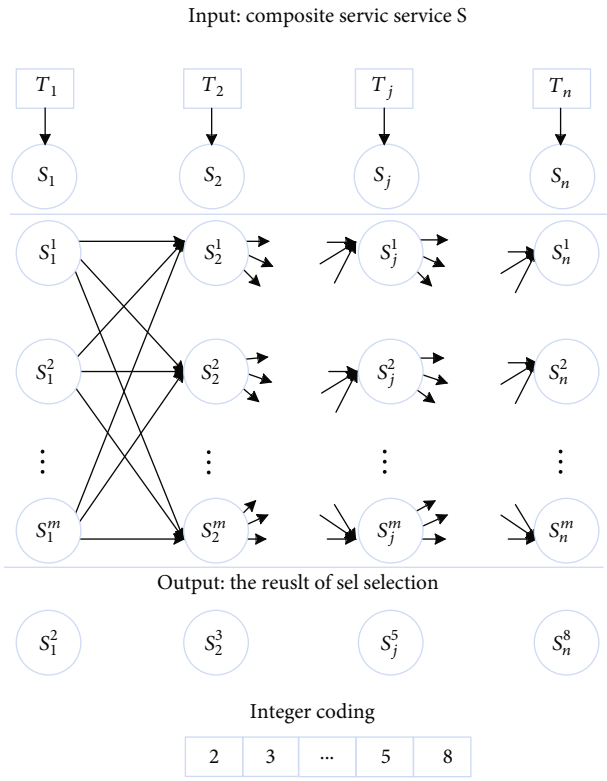

Figure 1: The process of service composition.

4. Proposed Approach

In this section, we will introduce the overall framework of the approach then explain the details of our approach.

4.1. The Whole Framework. The framework of our approach is shown in Figure 2, which consists of two parts: (1) Data preparation: first, the document model is used to process WSDL documents to generate the corresponding embeddings. Then, we use the k-means algorithm to cluster embeddings. Finally, they are divided into four clusters as the input data of the following algorithm. (2) Primary algorithm processing: first, FWA++ initializes firework population N randomly and evaluates their fitness values to find the current best solution. Second, FWA++ enters into the iterations: (a) Calculate the number of explosion sparks and the amplitude of the explosion for each firework according to the fitness value and then generate explosion sparks. (b) Perform Gaussian mutation to generate Gaussian sparks. Furthermore, map the unsatisfied sparks back to the feasible space. (c) According to the selection strategy, keep the current optimal solution and randomly select $N - 1$ individuals from the current sparks and fireworks for the next new generation. Finally, the algorithm termination condition is met and output the optimal solution.

4.2. Data Preparation. In this part, we will describe the process of data preprocessing in detail.

4.2.1. WSDL to Embedding. The significant text information of a service, including message, documents, service name, and operations, is recorded in the WSDL (Web Service Description Language) document. For the purpose of getting service function information, we extract message, documents, service name, and operations from the WSDL document. And then, we digitize the extracted text information. Here, the sentence transformer framework is used to convert text information into embedding. It provides an easy way to calculate dense vector representations of sentences and many models to realize the task of text digitization. We choose the paraphrase-xlm-r-multilingual-v1 model to obtain the embedding. The model is based on transformer networks such as BERT/RoBERTa/XLM-RoBERTa and achieves great performance in the mission. The text information is embedded in the vector space and is close to similar text information. After, each WSDL document is processed by the paraphrase-xlm-r-multilingual-v1 model, which is represented by an embedding. Finally, each service corresponds to an embedding.

4.2.2. Classify Web Services. The k-means algorithm is an unsupervised clustering algorithm that is relatively easy to implement. And it is widely used for good performance. The idea of the k-means algorithm is simple. A sample set is divided into several clusters according to the distance between the samples, and then, the points in the clusters close and the distance between the clusters is as large as possible. In our approach, we use k-means to cluster embeddings generated by the paraphrase-xlm-r-multilingual-v1 model, where each embedding represents each concrete service. In this way, classifying services according to functions is realized.

4.2.3. Merge Data. The QoS information corresponding to each service is merged according to the clustering result of the k-means algorithm. After the merger, each abstract service contains several concrete services. And the QoS information of each concrete service includes four attribute values, including cost, response time, reputation, and availability.

4.3. Fireworks Algorithm. The fireworks algorithm (FWA) is inspired by the fireworks explosion and presented by Tan and Zhu [7]. The idea of the algorithm is simple, but the specific implementation is complicated. It is mainly composed of four parts: explosion operator, mutation operator, mapping rule, and selection strategy. In the explosion phase, explosion sparks will be generated. And the basic principle is that if a firework's fitness value is better than that of other fireworks, it will have a smaller explosion range and generate more explosion sparks. The purpose is to speed up the local search ability near the current optimal solution. On the contrary, if a firework's fitness value is relatively poor, it has an extensive explosion range and generates a small number of explosion sparks. The primary purpose is to enhance the diversity of the population. In the mutation phase, Gaussian sparks will be generated and increase the diversity of the spark population. Meanwhile, unsatisfied Gaussian sparks

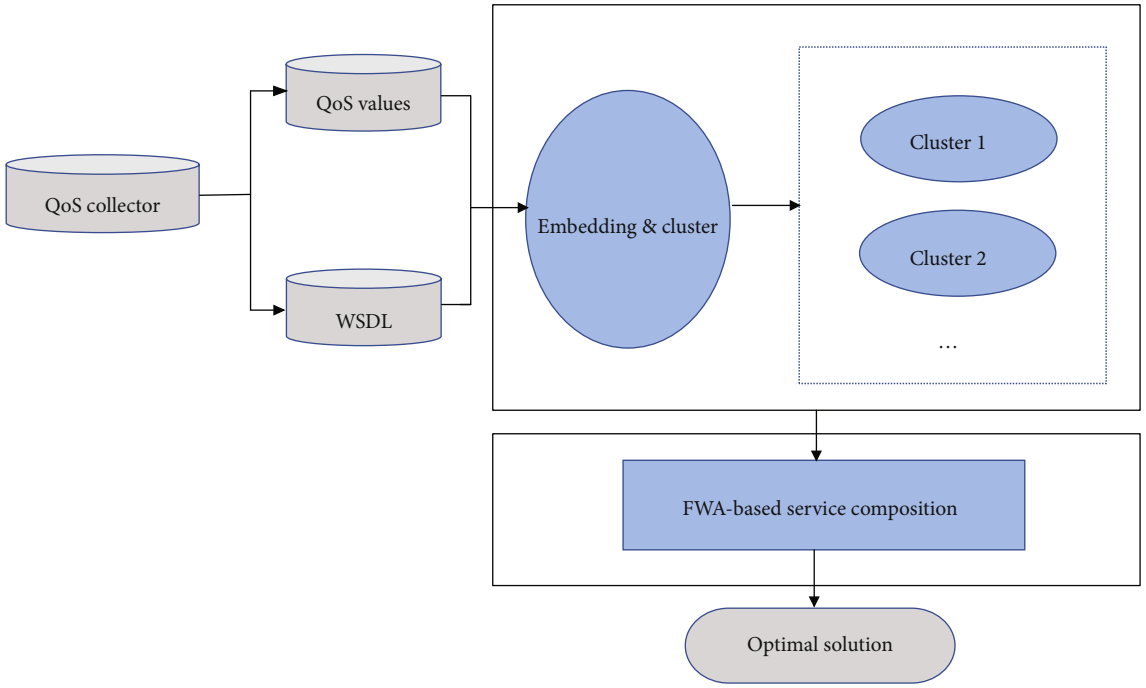

Figure 2: The framework of the proposed web service composition approach.

are mapped to the feasible space by the mapping rule. In the selection phase, the algorithm will refer to the pros and cons of sparks' location and randomly keep a specified number of sparks. The framework of the fireworks algorithm is shown in Figure 3. For each generation of explosion, N number of locations are selected; N fireworks are set off. And then, we obtain the location of sparks and evaluate the quality of the locations. The algorithm stops once the optimal solution is found. On the contrary, we select N other locations from the current fireworks and sparks as the next generation of explosion.

4.4. Feasible Solution Encoding. Before using the algorithm to implement service composition, each concrete service must be coded. The integer coding method is adopted to code them. Here, the concrete service in each abstract service is coded starting from 1. If there are N services in an abstract service, then $[1, 2, 3, 4, \cdots, N]$ is the code of its services, and the remaining abstract services are also used this way. Figure 4 shows a feasible solution [1–3, 5], which means that service 1 is selected from the first abstract service, service 5 is selected from the second abstract service, service 3 is selected from the third abstract service, and service 2 is selected from the fourth abstract service.

4.5. Operator Analysis. Suppose the problem to be optimized is

$$\text{Min } f(x) \in R, \quad x \in \Omega, \tag{4}$$

where Ω is the feasible region of the solution. Below, we will introduce each part in detail.

4.5.1. Explosion Operator. The explosion operation is essential for the fireworks to generate sparks. Generally, fireworks with better fitness values can generate more sparks in a smaller area; it enhances the algorithm's capability of local search. Conversely, fireworks with poor fitness values can only generate fewer sparks in a larger range; it is aimed at increasing the diversity of sparks and improving the algorithm's global search capability.

In the FWA, the explosion amplitude of each firework and the number of explosion sparks are calculated based on its fitness value relative to other fireworks. For a firework x_i, the formulas for the explosion amplitude A_i and the number of explosion sparks S_i are defined as follows:

$$A_i = \widehat{A} \times \frac{f(x_i) - y_{\min} + \varepsilon}{\sum_{i=1}^{N} (f(x_i) - y_{\min}) + \varepsilon}, \tag{5}$$

$$S_i = M \times \frac{y_{\max} - f(x_i) + \varepsilon}{\sum_{i=1}^{N} (y_{\max} - f(x_i)) + \varepsilon}, \tag{6}$$

where $y_{\min} = \min(f(x_i))$ $(i = 1, 2, \cdots, N)$ is the minimum fitness value in the current firework population and $y_{\max} = \max(f(x_i))$ $(i = 1, 2, \cdots, N)$ is the maximum fitness value in the current firework population. $\widehat{A}$ is a constant used to adjust the amplitude of the explosion, M is also a constant used to adjust the number of explosion sparks, and ε is the minimum positive constant to avoid division by zero. In order to dynamically balance the search ability and the ability to jump out of the local optimal solution of the FWA++, FWA++ randomizes the total number of spark M and amplitude of sparks $\widehat{A}$ in each generation.

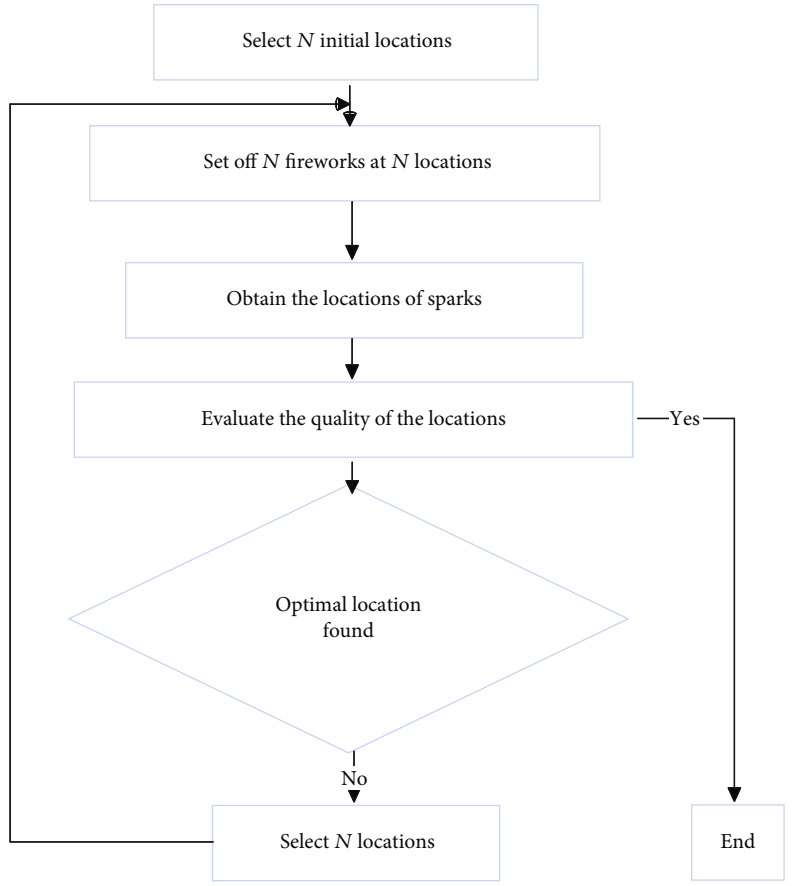

Figure 3: The framework of fireworks algorithm.

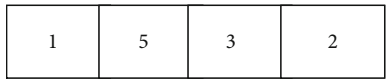

Figure 4: Feasible solution encoding format.

Limit the fireworks with good fitness value to generate too many explosion sparks. At the same time, fireworks with poor fitness value will generate few explosion sparks, and the number of sparks S_i is determined by the following formula:

$$\widehat{S}_i = \begin{cases} \text{round}(a * M), & S_i < aM, \\ \text{round}(b * M), & S_i > bM, a < b < 1, \\ \text{round}(S_i), & \text{otherwise}, \end{cases} \tag{7}$$

where a is lower bound for explosion amplitude, b is upper bound for explosion amplitude, and round$(\cdot)$ is a rounding function based on the rounding principle.

When a firework explodes, the sparks can be affected by the explosion in any z direction. The FWA uses a formula to randomly obtain several dimensions affected by the explosion:

$$z = \text{round}\,(d^* \text{ rand}\,(0, 1)), \tag{8}$$

where d is the dimension of a firework and rand $(0, 1)$ represents a random number function conforming to a uniform distribution on the interval $[0, 1]$.

Assuming that the ith firework is $x_i = (x_1, x_2, \cdots, x_N)$, the formula for generating sparks is

$$\begin{aligned} h &= A_i * \text{rand}\,(-1, 1), \\ ex_k^i &= x_k^i + h. \end{aligned} \tag{9}$$

4.5.2. Mutation Operator. Many sparks can be generated through the explosion operation, but these sparks are mainly around the original firework and have similar properties to the original firework population. In order to keep the diversity of the sparks, FWA introduces a mutation operator to generate Gaussian sparks. The process of generating Gaussian sparks is as follows: FWA selects a firework x_i from the firework population randomly, and then, it randomly selects a certain number of dimensions for the firework to perform Gaussian mutation operation. Performing Gaussian mutation operation on the selected dimension k of a firework x_i is

$$\widehat{x}_{ik} = x_{ik} \times e, \tag{10}$$

where $e \sim n\,(1, 1)$, $n\,(1, 1)$ represents the Gaussian distribution with a mean value of 1 and variance of 1.

4.5.3. Mapping Rules. The explosion sparks and Gaussian sparks may exceed the boundary range of feasible region Ω . When the spark x_i exceeds the boundary in dimension k, it is mapped to a new location through the mapping rule of formula (11):

$$\widehat{x}_{ik} = x_{\mathrm{LB},k} + |\widehat{x}_{ik}|\%(x_{\mathrm{UB},k} - x_{\mathrm{LB},k}), \tag{11}$$

where $x_{\mathrm{UB},k}$ and $x_{\mathrm{LB},k}$ are the upper and lower bounds of the solution space on dimension k, respectively.

4.5.4. Selection Strategy. In FWA, in order to keep the excellent individuals in the firework population to the next generation population, N individuals need to be selected from the candidate set composed of explosion sparks, Gaussian sparks, and fireworks in the current generation. Suppose that the candidate set is k and the population size is N. The individual with the smallest fitness value in the candidate set will be determinedly selected to the next generation as a firework (elite strategy), and the remaining $N-1$ fireworks will be selected from the candidate set. For candidate x_i, the calculation formula of the selected probability is as follows:

$$\begin{aligned} p(x_i) &= \frac{R(x_i)}{\sum_{x_j \in K} R(x_j)}, \\ R(x_i) &= \sum_{x_j \in K} d(x_i - x_j) = \sum_{x_j \in K} \|x_i - x_j\|, \end{aligned} \tag{12}$$

where $R(x_i)$ is the sum of the distances between the current individual x_i and other individuals in the candidate set. $p(x_i)$ is the probability of the individual being selected. In the candidate set, if an individual is far away from other individuals, then the probability of it being selected is high. But the above selection strategy is time-consuming, so FWA++ uses the way of random selection to keep $N-1$ individuals for the next generation.

4.6. Fitness Function. The utility function is used to evaluate the pros and cons of the composite service. In order to calculate the fitness value of the current composite service, the utility function is constructed as follows:

$$\begin{aligned} \text{fitness} &= \frac{\sum_{k=1}^{r} \left(q_k^{\text{agg}} \times w_k\right)}{\sum_{j=1}^{l} \left(q_j^{\text{agg}} \times w_j\right)} - D(p), \\ D(p) &= \frac{t_c}{t_m} \times \sum_{i=1}^{4} \left[w_i \times \left(\frac{\Delta q_i}{q_{\text{con}}^i} \right)^2 \right], \end{aligned} \tag{13}$$

where $D(p)$ is the penalty coefficient, t_c is the current generation, t_m is the maximum generation, w_i is the weight of the ith QoS attribute of the composite service, Δq_i is related to the positive and negative of QoS attributes, and q_{con}^i is the user's constraint on the ith QoS attribute, which is offered by users. The formula for calculating Δq_i is as follows.

If the QoS attributes are positive attributes (such as reputation and availability), then

$$\Delta q^i = \begin{cases} q_{\text{con}}^i - q^i, & q^i < q_{\text{con}}^i, \\ 0, & q^i > q_{\text{con}}^i. \end{cases} \tag{14}$$

If the QoS attributes are negative (such as price and response time), then

$$\Delta q^i = \begin{cases} q^i - q_{\text{con}}^i, & q^i > q_{\text{con}}^i, \\ 0, & q^i < q_{\text{con}}^i. \end{cases} \tag{15}$$

4.7. Pseudocode of FWA++ and Computational Complexity. The pseudocode of FWA++ is reported in Algorithm 1. Meanwhile, in the FWA++, each generation consists of initialization operation, explosion operation, mutation operation, and selection operation. The initialization operation includes initializing the population size and calculating the fitness values, and the computational complexity is $O(N)$ and $O(N)$, respectively. N is the population size. The explosion operation includes calculating the explosion amplitude, calculating the number of explosion sparks, generating the number of explosion sparks, and calculating the fitness values, and the computational complexity is $O(N)$, $O(N)$, $O(\widehat{S}_i)$, and $O(\widehat{S}_i)$, respectively. $\widehat{S}_i$ is the number of explosion sparks. The mutation operation includes generating Gaussian sparks and calculating fitness values, and the computational complexity is $O(M_g)$ and $O(M_g)$, respectively. The select operation chooses N sparks from fireworks, explosion sparks, and Gaussian sparks for the next generation, and the computational complexity is $O(N + M_g + \widehat{S}_i)$. M_g is the number of Gaussian sparks in each generation. Therefore, with the maximum number of generations T, the computational complexity of FWA++ is $O(T \times (5N + 3M_g + 3\widehat{S}_i))$.

5. Experiments and Evaluation

All algorithms were developed in Python, and all experiments were run on a PC equipped with AMD Ryzen 7 5800H CPU, 16 GB memory, and Windows 10 OS.

5.1. Data Set. In order to verify the effectiveness of our approach, we use QWS2.0 real-world data provided by [36]. The data set includes 9 QoS attributes of 2500 real web services. Since the data set does not include the price attribute of services, the attribute value of the service price is generated in a certain range (0.01-1.00) through a random algorithm.

5.2. Baseline Approaches. In this paper, we select the following well-known metaheuristic optimization algorithms to compare with our algorithm:

(1) *Fireworks Algorithm (FWA) [36].* FWA is a global optimization algorithm inspired by exploding fireworks. It has advantages in convergence speed and global solution accuracy.

Input: N- population size,
M- total numbers of sparks generated by the fireworks,
$\widehat{A}$- the maximum explosion amplitude,
M_g - the number of mutation sparks each generation
Output: X_{best} (the best solution)
1: Initialize fireworks population $P = X_i (i = 1, 2, \cdots, N)$;
2: Calculate the fitness value of each firework and store them to the
3: candidate set.
4: **while**($t \leq$ Max number of evolutions)**do**
5: **for** each firework X_i**do**
6: //in Eq. (5), A_i is dynamic in each generation.
7: //in Eq. (6), M is random in each generation.
8: Calculate the explosion amplitude A_i and the number $\widehat{S}_i$ of explosion sparks
9: by Eqs. (5), (6), and (7);
10: Obtain locations of $\widehat{S}_i$ sparks of the firework X_i using Eqs. (6), (7), and (8);
11: Calculate the fitness value of each explosion spark and store them to the
12: candidate set.
13: **end for**
14: **for**$k = 1$ to M_g**do**
15: Randomly select an individual X_i from all fireworks;
16: Generate a Gaussian spark for the firework by Eq. (10);
17: Map the sparks exceeding the search range into the search space by Eq. (11);
18: Calculate the fitness value of each Gaussian spark and store them to the
19: candidate set.
20: **end for**
21: Select the best location and keep it for next explosion generation;
22: Randomly select $N - 1$ locations from the candidate set (two types of sparks
23: and the current fireworks);
24: $t = t + 1$;
25: **end while**
26:Return X_{best}

Algorithm 1: The pseudocode of FWA++.

(2) *Particle Swarm Optimization with Corrective Procedure (CPSO) [38]*. In this approach, the corrective procedure was introduced to upgrade particles effectively. The results show that it greatly improves the problems of premature convergence and local optima.

(3) *Modified Artificial Bee Colony (mABC) Algorithm [32]*. In this algorithm, a chaotic-based opposition learning method is used to initialize the population, and differential evolution (DE) is used to enhance exploitation. mABC has robust scalability and high convergence speed.

5.3. Parameter Settings. The following parameter settings were used in our experiments: For fitness function, user preference q^c_{con}, q^t_{con}, q^r_{con}, and q^a_{con} for four QoS attributes are set to 0.50, 0.30, 0.80, and 0.80, respectively. The preference values are provided by the user; if the user does not provide the preference values, the default values are used. The weights of four attributes are set to 0.2, 0.2, 0.3, and 0.3, respectively. For FWA++, the population size N is set to 5, M is set to a random number between 50 and 80 in each generation, $\widehat{A} = 40$, $a = 0.04$, $b = 0.8$, $M_g = 11$, and the maximum number of generations (iterations) is set to 500. For FWA, the population size N is set to 5, and $M = 5$, $\widehat{A} = 40$, $a = 0.04$, $b = 0.8$, $M_g = 5$, and the maximum number of generations (iterations) is set to 500. For CPSO, the population size N is set to 30; the maximum number of iterations is set to 500. For mABC, the population size N is set to 30, the maximum number of iterations is set to 500, $\mu = 4$, and limit = $DXN/2$ (D is the dimension size).

5.4. Performance Comparison. In the experiments, the mean value of 20 independent runs of each algorithm was obtained to make a reasonable evaluation. And we fix the number of abstract services to 4 with the same number of candidate services and vary total candidate service from 100 to 500 to verify the effectiveness of FWA++.

Figure 5 shows the optimization results of the algorithms that are introduced above where iteration varies from 50 to 500 and the total number of services fixed to 300. We can see that as the iteration increases, the fitness value of the FWA++ is decreasing rapidly. The fitness value of FWA++ is superior to FWA, mABC, and CPSO when iteration reaches 100. The fitness hardly changes when the iteration is between 250 and 500, which means that FWA++ has converged. The result ensures that the performance of FWA++ outperforms other algorithms.

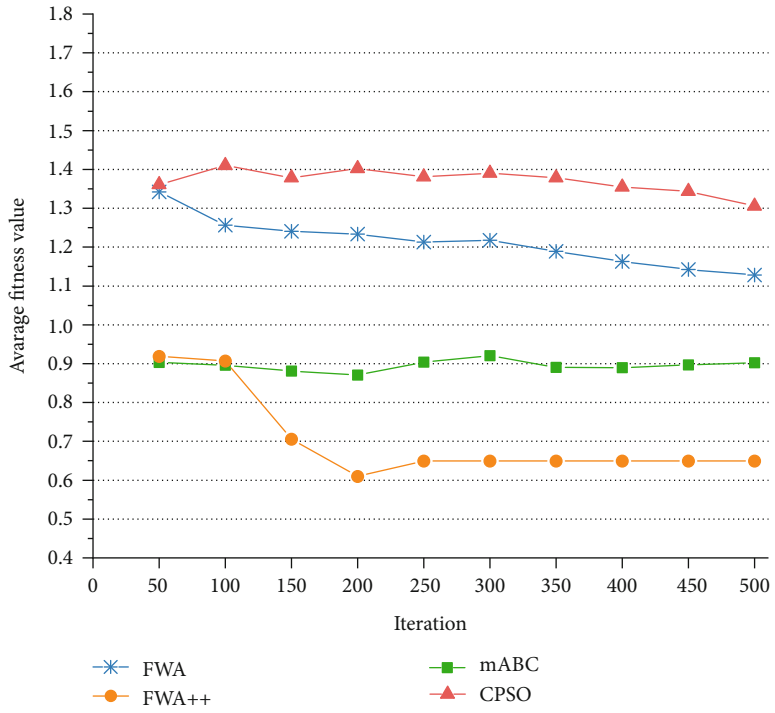

Figure 5: Comparison of optimization performance of different algorithms under different iterations.

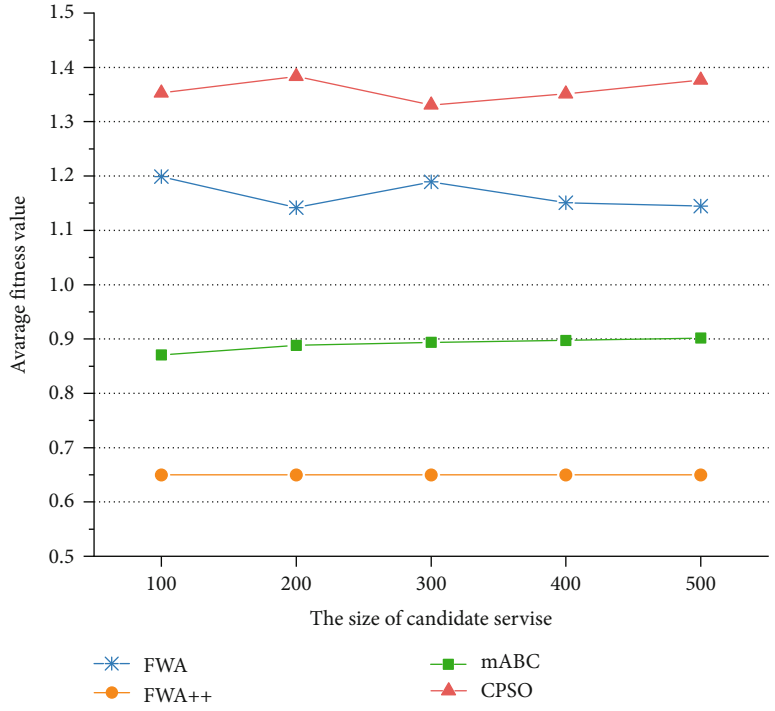

Figure 6: Comparison of optimization performance of different algorithms under the different sizes of candidate service.

From Figure 6, we can observe the optimization results where the total number of candidate services varies from 100 to 500. The fitness values of FWA++ are better than other algorithms in all cases. Meanwhile, all algorithms maintain a certain level of fitness value with the increase of the number of candidate services, which means that the number of candidate services has less influence on the algorithm performance.

From Figure 7, optimization results can be seen obviously where the number of the total candidate service varies

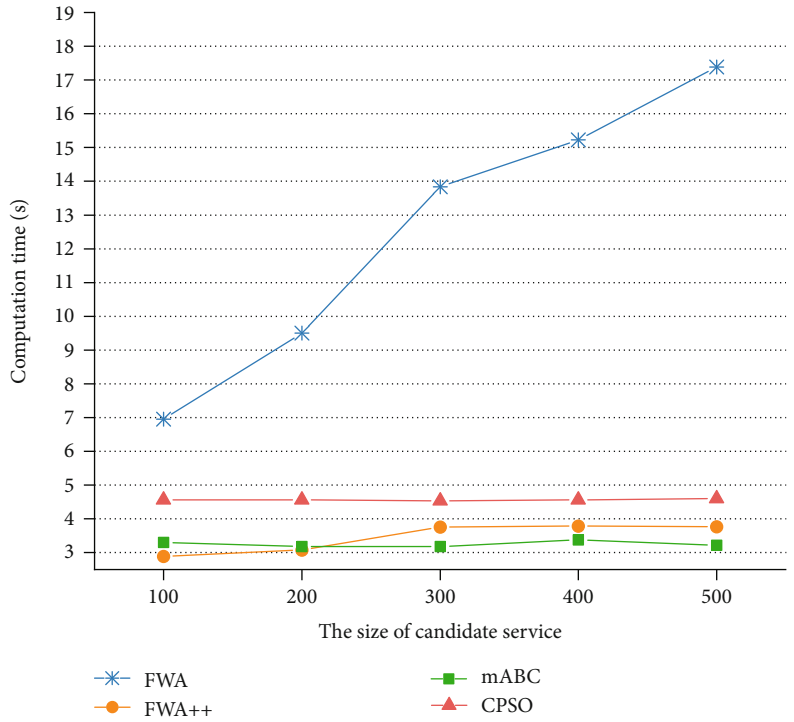

Figure 7: Comparison of computation time of different algorithms under the different sizes of candidate service.

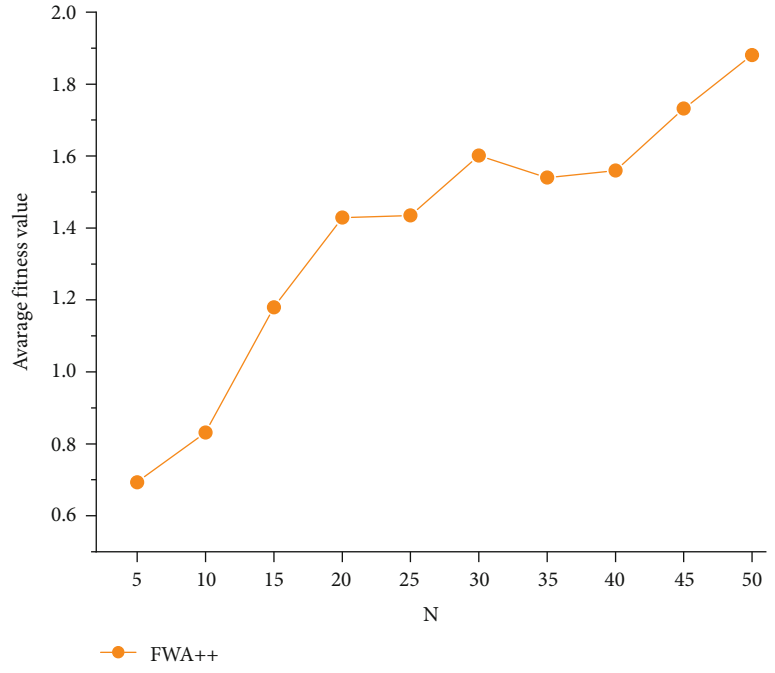

Figure 8: Comparison of optimization performance of different algorithms under the different sizes of the population.

from 100 to 500 and the total number of services fixed to 300. With the increasing number of candidate services, the computation time is maintained at a certain level. What is more, the computation time of FWA++ is least than other algorithms in all of the cases. In addition, FWA needs more computation time for the selection strategy, which is not comparable with other algorithms.

5.5. Sensitivity Analysis of Parameters

5.5.1. Impact of N. The variable N represents the population size, and the population size is the total number of individuals in any generation. Usually, this parameter is artificially set. The population size has an essential influence on the final solution. In order to obtain a more appropriate population

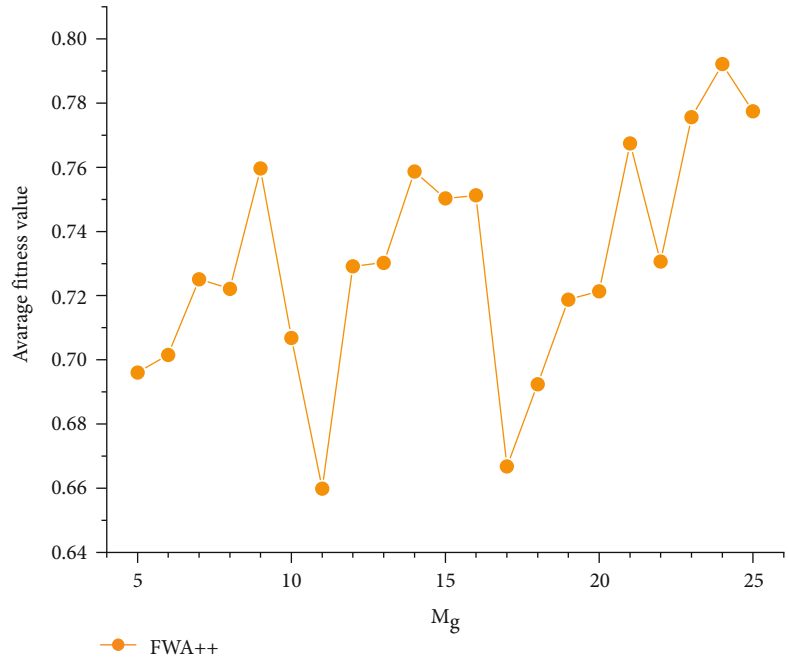

Figure 9: Comparison of optimization performance of different algorithms under the different numbers of Gaussian sparks.

size, we first set N between [5, 50]. By setting different population sizes, we obtain different fitness values. We consider fitness values under different population sizes. Through experiments, the fitness values under different population sizes are shown in Figure 8; we found that the fitness values are quite different under different population sizes. It shows that the population size greatly influences the experimental results. And we can observe that the population size is 5; a relatively small fitness value is obtained. So, we set the population size of FWA++ in our experiments to 5.

5.5.2. Impact of M_g. M_g represents the number of Gaussian sparks. If M_g is too small, FWA++ cannot maintain the diversity of the population. And it may fall into the local optimum and is difficult to obtain the optimal solution. Figure 9 shows the optimization results under different M_g; we can see that M_g is 11 and the fitness value obtained is the smallest, so we set M_g to 11 in our experiments.

5.6. Statistical Analysis. We perform the Friedman test [38–40] method and Wilcoxon signed rank test [41] method for statistical analysis to prove that our proposed approach is statistically significant. Table 1 shows the statistical analysis of the FWA++ compared with FWA, mABC, and CPSO, where "NAC" represents the number of abstract services, "MAC" represents the total number of concrete services, p value < 0.05 represents two approaches have distinct differences, and "+" represents that our approach is preferred to another one. It can be seen now that the p values of FWA++ are particularly low in all of the case (NAC = 4, MAC varies from 100 to 500), which demonstrate that FWA++ have distinct differences with FWA, mABC, and CPSO. Furthermore, we can observe from the "performance" index that the FWA++ algorithm has the best performance among these algorithms. Therefore, FWA++ is significantly effective.

Table 1: Statistical analysis results of the proposed new algorithm (FWA++) and comparison algorithms.

NAC	NCS	FWA++/FWA p value	performance	FWA++/FWA p value	performance	FWA++/FWA p value	performance
4	100	$3.28E-04$	"+"	$7.62E-06$	"+"	$7.62E-06$	"+"
	200	$1.52E-05$	"+"	$7.62E-06$	"+"	$1.52E-05$	"+"
	300	$2.28E-05$	"+"	$7.62E-05$	"+"	$7.62E-06$	"+"
	400	$1.52E-05$	"+"	$7.69E-03$	"+"	$7.62E-05$	"+"
	500	$7.62E-06$	"+"	$1.52E-05$	"+"	$2.33E-03$	"+"

6. Conclusions

Even though the number of services is increasing, it is difficult for a single service to complete complex tasks. Therefore, we need to combine multiple services to complete them. The efficiency of service composition has become a problem to be solved.

This paper proposes a service composition approach based on the improved fireworks algorithm (FWA++). We adopt a random selection strategy, which greatly reduces the running time of the algorithm. We balance the search ability and the ability to jump out of the local optimal solution by dynamically adjusting the total number of sparks and maximum amplitude of sparks for each generation. In this way, the optimization results are more accurate. Compared

with the existing algorithms, our algorithm has better performance.

In the future, we hope to integrate more multidimensional QoS attributes into the model, including waiting time, throughput, and reliability. In addition, we hope that web service composition will be more user-friendly, which can be combined with service recommendations. It can analyze user preferences through historical data and then recommend more personalized composite services to users.

Data Availability

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

Conflicts of Interest

The authors declare no potential conflicts of interest with respect to the research, authorship, and/or publication of this paper.

Acknowledgments

This work was supported in part by the Natural Science Foundation of Zhejiang Province under Grant LY22F020007 and Grant LY21F020002, in part by the Key Research and Development Program Project of Zhejiang Province under Grant 2019C01004 and Grant 2019C03123, and in part by the Commonweal Project of Science and Technology Department of Zhejiang Province under Grant LGF19F020007.

References

[1] W. F. Pan, X. X. Xu, H. Ming, and C. K. Chang, "Clustering Mashups by integrating structural and semantic similarities using fuzzy AHP," *International Journal of Web Services Research*, vol. 18, no. 1, pp. 34–57, 2021.

[2] W. F. Pan and C. L. Chai, "Structure-aware Mashup service clustering for cloud-based Internet of Things using genetic algorithm based clustering algorithm," *Future Generation Computer Systems*, vol. 87, pp. 267–277, 2018.

[3] W. F. Pan, J. L. Dong, K. Liu, and J. Wang, "Topology and topic-aware service clustering," *International Journal of Web Services Research*, vol. 15, no. 3, pp. 18–37, 2018.

[4] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[5] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Applied Soft Computing*, vol. 8, no. 1, pp. 687–697, 2008.

[6] K. M. Passino, "Bacterial foraging optimization," *International Journal of Swarm Intelligence Research (IJSIR)*, vol. 1, no. 1, pp. 1–16, 2010.

[7] Y. Tan and Y. Zhu, *Fireworks Algorithm for Optimization[C]// International Conference in Swarm Intelligence*, Springer, Berlin, Heidelberg, 2010.

[8] W. T. Pan, "A new fruit fly optimization algorithm: taking the financial distress model as an example," *Knowledge-Based Systems*, vol. 26, pp. 69–74, 2012.

[9] G. G. Wang, "Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems," *Memetic Computing*, vol. 10, no. 2, pp. 151–164, 2018.

[10] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: algorithm and applications," *Future Generation Computer Systems*, vol. 97, pp. 849–872, 2019.

[11] S. Li, H. Chen, M. Wang, A. A. Heidari, and S. Mirjalili, "Slime mould algorithm: a new method for stochastic optimization," *Future Generation Computer Systems*, vol. 111, pp. 300–323, 2020.

[12] J. Tu, H. Chen, M. Wang, and A. H. Gandomi, "The colony predation algorithm," *Journal of Bionic Engineering*, vol. 18, no. 3, pp. 674–710, 2021.

[13] S. Zheng, A. Janecek, and Y. Tan, "Enhanced fireworks algorithm," in *2013 IEEE congress on evolutionary computation*, pp. 2069–2077, Cancun, Mexico, 2013.

[14] S. Zheng, A. Janecek, J. Li, and Y. Tan, "Dynamic search in fireworks algorithm," in *2014 IEEE congress on evolutionary computation (CEC)*, pp. 3222–3229, Beijing, China, 2014.

[15] J. Li, S. Zheng, and Y. Tan, "Adaptive fireworks algorithm," in *2014 IEEE congress on evolutionary computation (CEC)*, pp. 3214–3221, Beijing, China, 2014.

[16] Y. W. Zhang, J. Wu, S. Zhao, and J. Tang, "Optimization service composition based on improved firework algorithm," *Computer Integrated Manufacturing Systems*, vol. 22, no. 2, pp. 422–432, 2016.

[17] J. Yu, H. Takagi, and Y. Tan, "Accelerating the Fireworks Algorithm with an Estimated Convergence Point," in *International Conference on Swarm Intelligence*, pp. 263–272, Springer, Cham, 2018.

[18] Y. J. Zheng, X. L. Xu, H. F. Ling, and S. Y. Chen, "A hybrid fireworks optimization method with differential evolution operators," *Neurocomputing*, vol. 148, pp. 75–82, 2015.

[19] B. Zhang, M. X. Zhang, and Y. J. Zheng, "A hybrid biogeography-based optimization and fireworks algorithm," in *2014 IEEE congress on evolutionary computation (CEC)*, pp. 3200–3206, Beijing, China, 2014.

[20] A. L. Bolaji, A. A. Ahmad, and P. B. Shola, "Training of neural network for pattern classification using fireworks algorithm," *International Journal of System Assurance Engineering and Management*, vol. 9, no. 1, pp. 208–215, 2018.

[21] M. Zare, M. R. Narimani, M. Malekpour, R. Azizipanah-Abarghooee, and V. Terzija, "Reserve constrained dynamic economic dispatch in multi-area power systems: an improved fireworks algorithm," *International Journal of Electrical Power & Energy Systems*, vol. 126, p. 106579, 2021.

[22] T. Gonsalves, "Feature subset optimization through the fireworks algorithm," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 4, no. 3, pp. 211–218, 2015.

[23] H. A. Bouarara, R. M. Hamou, A. Amine, and A. Rahmani, "A fireworks algorithm for modern web information retrieval with visual results mining," *International Journal of Swarm Intelligence Research (IJSIR)*, vol. 6, no. 3, pp. 1–23, 2015.

[24] Y. C. Ho and D. L. Pepyne, "Simple explanation of the no-free-lunch theorem and its implications," *Journal of Optimization Theory and Applications*, vol. 115, no. 3, pp. 549–570, 2002.

[25] Q. She, X. Wei, G. Nie, and D. Chen, "QoS-aware cloud service composition: a systematic mapping study from the perspective of computational intelligence," *Expert Systems with Applications*, vol. 138, p. 112804, 2019.

[26] A. Klein, F. Ishikawa, and S. Honiden, "Efficient heuristic approach with improved time complexity for QoS-aware service composition," in *2011 IEEE international conference on web services*, pp. 436–443, Washington, DC, USA, 2011.

[27] M. Liu, M. Wang, W. Shen, N. Luo, and J. Yan, "A quality of service (QoS)-aware execution plan selection approach for a service composition process," *Future Generation Computer Systems*, vol. 28, no. 7, pp. 1080–1089, 2012.

[28] C. F. Lin, R. K. Sheu, Y. S. Chang, and S. M. Yuan, "A relaxable service selection algorithm for QoS-based web service composition," *Information and Software Technology*, vol. 53, no. 12, pp. 1370–1381, 2011.

[29] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, "An approach for QoS-aware service composition based on genetic algorithms," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pp. 1069–1075, Washington DC, USA, 2005.

[30] M. Tang and L. Ai, "A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition," in *IEEE congress on evolutionary computation*, pp. 1–8, Barcelona, Spain, 2010.

[31] S. A. Ludwig, "Applying particle swarm optimization to quality-of-service-driven web service composition," in *2012 IEEE 26th international conference on advanced information networking and applications*, pp. 613–620, Fukuoka, Japan, 2012.

[32] M. Chandra and R. Niyogi, "Web service selection using modified artificial bee colony algorithm," *IEEE Access*, vol. 7, pp. 88673–88684, 2019.

[33] X. Xu, Q. Z. Sheng, Z. Wang, and L. Yao, "Novel artificial bee colony algorithms for QoS-aware service selection," *IEEE Transactions on Services Computing*, vol. 12, no. 2, pp. 247–261, 2019.

[34] C. Y. Li, J. Li, H. L. Chen, and A. A. Heidari, "Memetic Harris Hawks optimization: developments and perspectives on project scheduling and QoS-aware web service composition," *Expert Systems with Applications*, vol. 171, p. 114529, 2021.

[35] C. Li, J. Li, and H. Chen, "A meta-heuristic-based approach for QoS-aware service composition," *IEEE Access*, vol. 8, pp. 69579–69592, 2020.

[36] E. Al-Masri and Q. H. Mahmoud, "Qos-based discovery and ranking of web services," in *2007 16th international conference on computer communications and networks*, pp. 529–534, Honolulu, HI, USA, 2007.

[37] B. Borowska, "An improved CPSO algorithm," in *2016 XIth international scientific and technical conference computer sciences and information technologies (CSIT)*, pp. 1–3, Lviv, Ukraine, 2016.

[38] A. Zhu, W. Chen, J. Zhang, X. Zong, W. Zhao, and Y. Xie, "Investor immunization to Ponzi scheme diffusion in social networks and financial risk analysis," *International Journal of Modern Physics B*, vol. 33, no. 11, p. 1950104, 2019.

[39] H. Li, T. Wang, W. F. Pan et al., "Mining key classes in Java projects by examining a very small number of classes: a complex network-based approach," *IEEE Access*, vol. 9, pp. 28076–28088, 2021.

[40] W. F. Pan, B. B. Song, K. S. Li, and K. J. Zhang, "Identifying key classes in object-oriented software using generalized k-core decomposition," *Future Generation Computer Systems*, vol. 81, pp. 188–202, 2018.

[41] W. F. Pan, H. Ming, Z. J. Yang, and T. Wang, "Comments on "using k-core decomposition on class dependency networks to improve bug prediction model's practical performance"," *IEEE Transactions on Software Engineering*, 2022.