Optimized Artificial Bee Colony Algorithm for Web Service Composition Problem

Shudong Zhang, Yaru Shao, and Lijuan Zhou

***Abstract*—With the proliferation of web services offering similar functionality, how to choose the optimal service composition that meet consumer demands based on Qos-aware has become increasingly difficult. In this paper, we study the web service composition problem in a sequential composition model. In this work, we propose an optimized artificial bee colony algorithm (OABC) to make it more suitable for the problem. On one hand, this algorithm uses the group initialization strategy based on opposition learning to enhance the diversity of the initial population. On the other hand, in order to improve the exploration and exploitation ability of the algorithm, we dynamically adjust the neighborhood search range during the hired bee phase and introduce a global optimal bee behavior in the scout bee phase. Our experimental results demonstrate the good performance of the solutions obtained by OABC algorithm in fitness and execution time when compared with other existing algorithms.**

***Index Terms*—Web service composition, Qos-aware, artificial bee colony (ABC), optimized artificial bee colony (OABC).**

I. Introduction

Today, with the Internet, especially the Web, becoming the fastest growing collection of data and software programs, and service computing, cloud computing, big data and other technologies are gradually maturing. A growing number of software and some generalized service capabilities appear on the Internet in the form of services. At the same time, various industries gradually emerge large-scale intelligent service resources, and consumers' needs become more demanding and complex. It has gradually become a research hotspot on how to efficiently combine services in a large number of cross-organizational resources to meet the needs of consumers.

Service-oriented architecture (SOA) represents an architecture model suitable for composite services, enabling even dynamically, combinations of a variety of independently developed services to form a distributed software-intensive system. Web services are the basic unit of SOA, and they are considered as self-describing, self-contained, modular applications that can be published, located, and invoked across the network using standard technologies, which are reusable and can be operated on different platforms. However, because of single Web service cannot satisfy all the functionality needed by the user, we need to integrate existing web services to create new composite web services to provide solutions. This process is called Web service composition. Service composition optimization can be carried out from both functional and non-functional aspects. The functional aspects mainly realize automatic service discovery, matching, process planning etc. to meet the functional requirements of the service solution. For example, transportation services, public services, restaurant services, etc. that are aggregated by the tourism service system are all abstract functional services. But how do we choose the appropriate services over the many that can provide the same functionality. For example, transportation service providers can provide consumers with the following travel methods: travel by air, train or car etc. so we need to select a set of service combination solutions that can meet the needs of consumers from non-functional options of candidate services such as the quality of services [1].

In the service composition process, we convert user functional requirements into business workflows. This process includes n subtasks, each of which has its own function-related abstract definition. Specifically, the functional attributes of the basic Web service are mapped to subtasks, and then the mapped Web services are integrated into the composite Web service. In fact, the service composition optimization problem can be regarded as an optimization problem to find the best solution in the feasible solution set. When the problem is simple, the algorithm for solving this type of problem is not complicated, but as the number of tasks and the number of basic Webs increase, the number of possible solutions increases exponentially. Therefore, it is not feasible to find a solution in polynomial time by some accurate methods [2], [3].

In this problem, some people have proposed using swarm intelligence optimization algorithms to find the best solution. Artificial bee colony algorithm (ABC) is a new swarm intelligence optimization algorithm proposed by Karaboga [4]. This algorithm performs better in solving optimization problems [4]-[6]. Considering the superiority of ABC and the problems existing in solving service optimization problems. In this paper, we proposed a new heuristic-based optimized artificial bee colony algorithm (OABC) for service selection. Related work is described in Section II, Section III describe the definition and mathematical model of Qos-aware service composition. Section IV briefly reviews the classical ABC and its shortcomings in service composition problems, and then we give the proposed OABC algorithm. The experimental results and analysis are presented in Section V. Finally, conclusions are given in Section VI.

Manuscript received February 15, 2020; revised December 10, 2020. This research was supported by The National Key Research and Development Program of China (2017YFB1400803, 2018YFB1402900).

Shudong Zhang, Yaru Shao, and Lijuan Zhou are with the College of Information Engineering, Capital Normal University, Beijing, 100089, China (Corresponding author: Lijuan Zhou; e-mail: zhangshudong@cnu.edu.cn, 2181002051@cnu.edu.cn, zhoulijuan@cnu.edu.cn).

doi: 10.18178/ijmlc.2021.11.5.1056

II. Related Work

In recent years, a lot of research has been carried out on the optimization of service composition at home and abroad. Currently, the nonfunctional service composition optimization is mainly focused on the quality of service (QoS). Qos-aware service composition is a challenging problem because it has proven to be an NP problem as the number of tasks and services increases [7]. Traditional combinatorial optimization methods, such as branch-definition method [8], mixed integer programming [9], etc., are very sensitive to the number of candidate services, and their computational complexity shows an exponential increase with increasing scale.

To solve this problem, many researchers have proposed swarm intelligence optimization algorithms (such as the ABC algorithm). Its core concept is to explore space based on the nature of the population. In this process, learning strategies are used to acquire information to effectively find approximate optimal solutions. For example: genetic algorithm, differential evolution algorithm, simulated annealing algorithm, ant colony algorithm, particle swarm algorithm, etc. [10] are all used to solve the service composition optimization problem based on large-scale candidate services. These algorithms have been popular in the field of web service combination because of their advantages of self-adaptive, self-organization, cooperation, strong robustness and good distributed parallelism. ABC algorithm has proven to be superior in many optimization problems [11], [12] because of its simplicity, robustness, ease of implementation and good convergence. In order to improve the performance of ABC both in exploration and exploitation [13], researchers have proposed many variants of ABC in recent years. In [14], Kousalya G *et al*. used the Bees algorithm (BA). BA is based on random search and neighborhood search to obtain the best service combination. In [15], Ying *et al.* proposed a discrete Gbest-guided artificial bee colony algorithm (DGABC), which added attenuation functions to the service composition model and formalized the service composition into a nonlinear integer programming problem to improve the accuracy of the evaluation. In [16], Xiaolong *et al.* established a mathematical model for QoS attributes, and they proposed a representative quantitative method and aggregation method for Qos characteristics. Then they used a taboo strategy and chaotic factors to improve ABC. In [17], Liu R *et al.* proposed a classification algorithm based on 4.5t to set the optimal parameters of ABC. In [18], Wang X *et al.* proposed an ABC approximation method based on a greedy search strategy. The first algorithm is called a threshold-based algorithm (TBA) and the second algorithm is called a distance-based algorithm (DBA). In [19], inspired by bee and cuckoo search, respectively, Chifu V R and others proposed two selection methods to compose a graph to search the best solution. In [20], Dahan F *et al.* proposed the enhanced artificial bee colony algorithm (EABC), which performs neighbor selection based on Euclidean distance to control the convergence efficiency of the algorithm.

Compared with the classical ABC, the above improved artificial bee colony algorithm has better performance, but there are still some shortcomings. Currently, existing research work only adopts a single strategy to improve the performance of ABC without considering the entire exploration and exploitation process of the algorithm. In this paper, we proposed a new heuristic-based optimized artificial bee colony algorithm (OABC) for service composition. On one hand, this algorithm uses the group initialization strategy based on opposition learning to enhance the diversity of the initial population. On the other hand, in order to improve the exploration and exploitation ability of the algorithm, we dynamically adjust the neighborhood search range during the hired bee phase and introduce a global optimal bee behavior in the scout bee phase. Our experimental results indicate that compared with other existing algorithms, our proposed algorithm can find better solutions and shorten the execution time.

III. Problem Description and Definition

In this section, we provided a definition of the web services composition problem and related mathematical modeling. Table I explained the notations frequently appeared in this paper.

Let S represents the abstraction of tasks that perform different functions in the service composition. Each abstract service class S_i is a set of web services that provide the same functionality, i.e., $S_i = \{s_i^1, s_i^2, \dots, s_i^m\}$, here $i = 1, \dots, n$, $j = 1 \dots m$, So s_i^j denotes the j^{th} web service in class i.

As a single web service is difficult to meet the increasingly complex requirements, we need to integrate existing web services to create new composite web services to provide the best solution for task requirements. This process is called web service composition. This solution may be a complex workflow that contains the basic web services selected from each candidate set of abstract services. The workflow defines the arrangement between services: sequential, parallel, circular, etc. Since workflows can be broken down into a series of simple sequential structures. In this paper, we only consider sequential composition processes. As shown in Fig. 1.

TABLE I: Notations Descriptions Used in This Paper

Notations	Description
s	Atomic web services
S	Abstract service class
m	Number of web services in the abstract service class S
n	Number of tasks participating in the service
F	Fitness function value
CWS	Composite web service

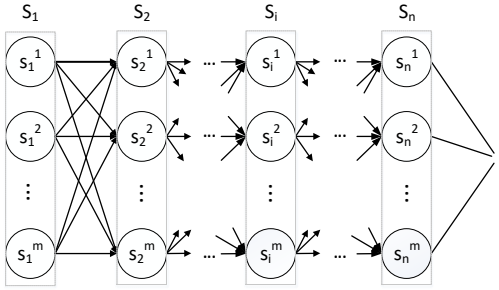


Fig. 1. Complex service composition.

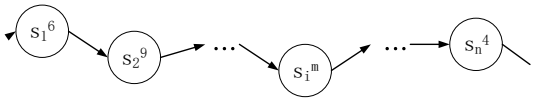


Fig. 2. Optimal solution.

Based on consumer preferences, we aim to find a set of service composition solutions that satisfy the global best value of Qos. The final possible results are shown in Fig. 2. For each solution, we use the formulas in Table II to calculate the value of each QoS function. Here we consider four Qos attributes frequently used by services: response time, reliability, throughput, and cost. In the formula, n denotes the number of tasks and s_i^j represents the j^{th} candidate web service of the i^{th} task.

TABLE II: QOS AGGREGATION FORMULAS

Qos Attributes	Calculation Formula
Response Time (RT)	$\sum_{i=1}^{n} RT(s_i^j)$
Reliability(R)	$\prod_{i=1}^{n} R(s_i^j)$
Throughput(T)	$\prod_{i=1}^{n} T(s_i^j)$
Cost(C)	$\sum_{i=1}^{n} C(s_i^j)$

Service attributes are generally divided into positive Qos and negative Qos. The higher the value of positive Qos, the higher the value they provide, such as service reliability and throughput. On the other hand, for negative Qos, the smaller the value, the higher the utility value, such as service response time and cost. To facilitate the calculation of the objective function value of the composite service, we normalize the attribute values of different web services to a number between 0 and 1.

For positive Qos, equation (1) is used to normalize:

$$Q_k^{norm}(s) = \begin{cases} \dfrac{Q_k(s) - Q_k^{min}(s)}{Q_k^{max}(s) - Q_k^{min}(s)}, & Q_k^{max}(s) \neq Q_k^{min}(s) \\ 1, & Q_k^{max}(s) = Q_k^{min}(s) \end{cases} \quad (1)$$

For negative Qos, equation (2) is used to normalize:

$$Q_k^{norm}(s) = \begin{cases} \dfrac{Q_k^{max} - Q_k(s)}{Q_k^{max}(s) - Q_k^{min}(s)}, & Q_k^{max}(s) \neq Q_k^{min}(s) \\ 1, & Q_k^{max}(s) = Q_k^{min}(s) \end{cases} \quad (2)$$

where $Q_k^{max}(s)$ and $Q_k^{min}(s)$ are the maximum and minimum values of the k^{th} Attribute of service s. Equation (3) is used to evaluate the fitness of the solution.

$$F(CWS) = \sum_{k=1}^{h} Q_k^{agg}(CWS) \quad (3)$$

where $Q_k^{agg}(CWS)$ Is the normalized sum of each service attribute in the composite service, and h is the number of service attributes

IV. ALGORITHM IMPROVEMENT

A. Classic ABC

The classical ABC was proposed by Karaboga [5] based on the bee colony foraging behavior. There are three important roles: hired bees, onlooker bees, and scout bees. The corresponding three important behavior modes are: hired bees to search for food sources, onlooker bees to choose food sources, and scout bees to determine whether to abandon food sources. The main idea is: the hired bees target the food source to collect available information; onlooker bees to select high-quality food sources through the food source information collected by the hired bees; If a food source is abandoned after being mined multiple times, the hired bee corresponding to the food source will be turned into a scout bee to find a new food source. In service composition optimization problems, the food source corresponds to the solution, and the quality of the food source corresponds to the quality of the solution.

This algorithm consists of the following four steps:

1) Initialization of food sources

The initial food source is randomly generated, as in (4). There are SN initial food sources, which can also be called initial populations, i.e., $X = \{X_1, X_2, \ldots X_i, \ldots, X_{SN}\}$. Each food source X_i Represents a feasible solution in d-dimension, i.e., $X_i = \{X_{i1}, X_{i2}, \ldots, X_{ij}, \ldots, X_{id}\}$.

$$X_{ij} = \underline{X_j} + \varphi_{ij}\left(\overline{X_j} - \underline{X_j}\right), \varphi_{ij} \in (0\sim1) \quad (4)$$

where j is the dimension, $j \in \{1,2,3,\ldots,d\}$, $\overline{X_j}$ And $\underline{X_j}$ Denote the upper and lower limits of the j^{th} Dimension of the food source, respectively.

2) Hired bee phase

Each hired bee searches in the neighborhood of the current food source as in (5), and compares the new food source V_i With the old food source X_i. Get better candidate solutions by greedy selection.

$$V_{ij} = X_{ij} + \varphi_{ij}(X_{ij} - X_{kj}) \ \varphi_{ij} \in (-1\sim1) \quad (5)$$

where k is a food source, $k \in \{1,2,\ldots,SN\}, j \in \{1,2,\ldots d\}$, and $k \neq i$.

3) Onlooker bee phase

After the hired bee completes the neighborhood search, the onlooker bee uses the roulette algorithm to select a more suitable food source according to the information collected by the hired bee, as in (6).

$$P_i = \frac{fit_i}{\sum_{i=1}^{SN} fit_i} \quad (6)$$

where fit_i Denotes the fitness of the i^{th} Food source. After making a choice, onlooker bee searches the neighborhood of the food source again as in (5), and then greedily selects a better food source.

4) Scout bee phase

If the quality of a food source is not updated after searching for a predefined limit, indicating that the food source is trapped in a local optimum. The food source will be abandoned, and the hired bee corresponding to the food source will be turned into a scout bee.

Obviously, ABC is a process of exploring and exploiting. However, the classical ABC algorithm was not capable of early exploration and later exploitation. Specifically, the initial food source is randomly generated. Moreover, the hired bee blindly searches for the food source neighborhood in a certain dimension. After the food source falls into local

optimization, scout bees are randomly initialized. This uncertainty directly leads to the slow convergence rate of ABC and the solution is easy to fall into local optimization.

In view of the shortcomings of the classic ABC mentioned above, researchers have proposed many methods to improve the search efficiency of ABC. For example, in [18], Wang X et al. Proposed the ABC approximation method based on greedy search strategy, TBA and DBA. Experimental results showed that the solution improves in quality, but the time complexity is high. In [20], Dahan F *et al*. Proposed the EABC algorithm, which selects a fixed number of neighbors based on Euclidean distance in order to control the algorithm's later convergence speed. However, experimental results showed that the number of neighbors under different conditions is difficult to fix. In summary, none of the methods mentioned above take the comprehensive search mechanism into consideration.

In the problem of service combination, algorithm execution time and service combination scheme quality are important factors influencing user satisfaction. Therefore, in order to make ABC algorithm more adaptable, we proposed an optimized artificial bee colony algorithm after analyzing the shortcomings of the existing algorithm.

B. Our Proposed ABC

In the proposed OABC, we made three major changes to the current classic ABC search mechanism.

1) Use opposition-based initialization

In fact, the distribution of the initial solution in space affects the solution efficiency of the algorithm. In order to increase the diversity of the initial population, we used population initialization method based on opposition learning to make the initial population more evenly distributed in the solution space.

Specifically, after the population is initialized, the opposition population is generated as in (7).

$$oX_{ij} = UB_j + LB_j - X_{ij} \tag{7}$$

We evaluated the original population and its opposite population (*2* • *SN*), and selected better *SN* food sources as the initial population, where UB_j and LB_j represent the upper and lower bounds of the corresponding dimension, respectively.

2) Dynamic adjust search range

In general, at early search stage, we prefer distant neighbors to encourage exploration. As the evolutionary process advances, we prefer to select close neighbors to speed up convergence after determining the approximate orientation of the optimal solution. As in (8), the factor φ is a parameter that controls the search range of the neighborhood. The larger the value of φ, the larger the search range of the neighborhood. In the classic ABC algorithm, φ is randomly generated without control, so we added a factor θ to dynamically adjust the neighborhood search scope.

$$V_{ij} = X_{ij} + \theta\varphi_{ij}\left(X_{ij} - X_{kj}\right), \varphi_{ij} \in (-1\sim1),$$
$$\theta = \varphi_{max} - \frac{iter(\varphi_{max} - \varphi_{min})}{MCN} \tag{8}$$

where φ_{max} and φ_{min} are constants, they are used to control the maximum and minimum range of neighborhood search. This paper takes experience values of *1* and *0.4*, $iter$ is the current number of iterations, and MCN is the maximum number of iterations.

3) Introduce a global optimal bee behavior for scout bee

If a food source falls into a local optimum after limit iterations, then the onlooker bee will be converted into a scout bee to randomly generate a new food source as in (4). Related researches [21], [22] showed that the global optimal solution X_{best} in each iteration of the algorithm record useful information. Investigating the best solution can potentially improve the performance of the algorithm. The improved method for the scout bee to generate a new solution as in (9):

$$X_{ij} = X_{min,j} + \varphi_{ij}\left(X_{max,j} - X_{min,j}\right) + (1 - \varphi_{ij})(X_{ij} - X_{best,j}) \tag{9}$$

where φ_{ij} is a random number between *0* and *1*; $X_{best,j}$ is the j^{th} component of the global optimal solution.

The execution process of OABC algorithm is shown in algorithm 1

Algorithm 1: The OABC algorithm

Input: *SN*-population size, *MCN*- maximum iterations
limits- maximum number of food source iterations
Output: Best solution

1. Initialize the food sources
2. Generate the opposite food sources
3. Evaluate both the initialized and opposite food sources
4. Select the *SN* solutions using elitism principle
5. **for** *iter* = 0 → *MCN*-1
6. **Hired bee phase:**
7. Execute a dynamic neighborhood search
8. Calculate the fitness of V_i and X_i
9. **if** *fitness*(V_i) > *fitness*(X_i) **then**
10. $X_i = V_i$
11. **else**
12. *limit*[*i*] = *limit*[*i*]+1
13. **end**
14. Calculate the $P[X_i]$ of being selected
15. **end**
16. **Onlooker bee phase:**
17. Select solutions according to $P[X_i]$
18. Perform steps 7 to 13 in Algorithm 1
19. **end**
20. **Scout bee phase:**
21. **if** *limit*[*i*]==*limits* **then**
22. Initialize scout bees according to X_{best}
23. *limit*[*i*]=0
24. **end**
25. **end**
26. keep best solution
27. **end**
28. **return** best solution

V. Experimental Result and Discussion

In this section, in order to verify the effectiveness of our proposed OABC algorithm. We compared OABC with the classical ABC, and EABC.

A. Experiments Setting

All experiments were performed in the environment of Microsoft Windows 10, Intel® Core TM i7 CPU 3.60 GHz

and 8GB RAM, and all algorithms were implemented in Java.

We considered four Qos attributes of the service: response time, reliability, throughput, and cost. We simulated real service data and generated two groups of datasets. The dataset contains rows and columns, which represent atomic web services and Qos attributes, respectively. In the first group, we fixed the abstract service class (the number of tasks *n*) to *10* and generated a dataset with the number of candidate web services *m* from *100* to *1000*. In the second group, we fixed the number of candidate services *m* to *500* and generated a dataset of abstract service classes *n* from *5* to *10*.

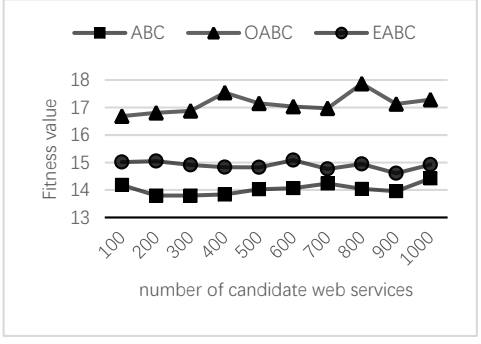
In the experiments, all algorithms used the following parameter settings: The population size (*SN*) is set to *100*, the maximum number of iterations (*maxCycle*) is set to *500*, and the limit number for determining whether the food source is abandoned is set to *100*, EABC algorithm used the best number of neighbors *5*. Since the random behavior of the algorithm may lead to different experimental results, we independently simulated each dataset *20* times. Finally, we evaluated the algorithm's average fitness value and average execution time based on the *20* best solutions.

B. Results

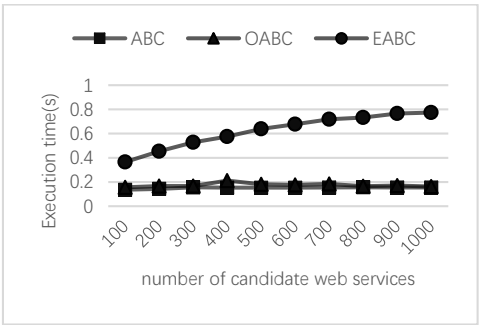
Fig. 3 and Fig. 4 show the comparison of the experimental results of the first and second datasets, respectively.

For the first dataset, we fixed the number of tasks to *10* and the number of candidate web services in the range of 10-1000. Fig. 3a and Fig. 3b show the comparison of the fitness value and execution time of the solution obtained by the three algorithms under different numbers of candidate services respectively.

Obviously, as the number of candidate web services increase, the ABC and EABC fitness values change smoothly, but our proposed OABC fitness value is significantly better than the classic ABC and EABC. The execution time of OABC is slightly longer than that of ABC, but close to ABC. Because our proposed OABC changes the population initialization strategy and the bee colony search strategy based on the classic ABC. But OABC requires less execution time than EABC. This is because in the neighborhood search phase, each iteration of EABC needs to consider a fixed number of neighbors in a random dimension and rank according to the similarity between services. As the number of candidate web services increases, the execution time of EABC increases significantly. OABC only adds factors that can change the search range. Although the obtained solution is not as stable as EABC in terms of fitness, its search range is wide, and it can find better quality solutions in a short execution time.



(a)

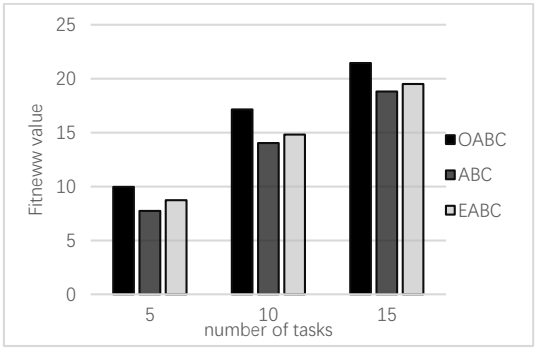


(b)

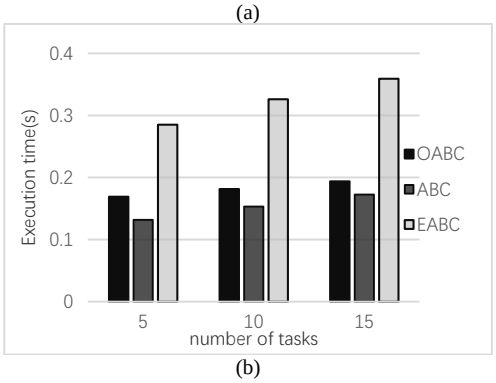
Fig. 3. The performance of the algorithms on varying web services.

For the second dataset, we fixed the number of candidate web services to *500* and extracted the number of tasks *n* = *5,10,15*, and the other parameters were the same as the previous experiment.

Fig. 4a and Fig. 4b respectively show the fitness value and execution time comparison of the three algorithms under different task numbers. It can be seen that the fitness value of OABC is always better than ABC and EABC. Similar to the first datasets, ABC and OABC are close in execution time, but much shorter than the execution time of EABC, so we can conclude that our proposed OABC has better performance.



(a)



(b)

Fig. 4. The performance of the algorithms on varying tasks.

VI. Conclusion

In this paper, we transformed the web service composition optimization problem into a polynomial in a mathematical model to find the optimal solution. Based on the classic ABC algorithm, we proposed an optimized artificial bee colony algorithm (OABC) for QoS-aware web service composition. OABC uses the group initialization strategy based on opposition learning to enhance the diversity of the initial population. On the other hand, in order to improve the exploration and development ability of the algorithm, we dynamically adjust the neighborhood search range during the

hired bee phase and introduce a global optimal bee behavior in the scout bee phase. Experimental results showed that our proposed algorithm outperforms other algorithms in terms of solution fitness and execution time. In the existing algorithms, the neighborhood search strategy only randomly explores one dimension in the candidate solution. The limited one-dimensional information exchange method is easy to fall into a local optimum when dealing with multi-objective problems. In the future, we plan to study new algorithms to solve the problem of multi-dimensional information exchange.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

ShuDong Zhang analyzed the subject status at home and abroad, and then proposed the research content. Yaru Shao realized the algorithm improvement and participated in writing the paper. Lijuan Zhou gave guidance on data analysis and drew conclusions.

Acknowledgment

The authors would like to acknowledge the support of National Key Research and Development Program of China (2017YFB1400803, 2018YFB1402900) and the experimental environment provided by Capital Normal University.

References

[1] D. A. Menasce, "Composing web services: A QoS view," *IEEE Internet Computing*, 2004, pp. 88-90.
[2] J. Chandrashekar, G. R. Gangadharan, and R. Buyya, "Computational intelligence based qos-aware web service composition: A systematic literature review," *IEEE Transactions on Services Computing*, 2015, pp. 1-1.
[3] P. Emad, Y. Rastegari, P. M. Esfahani, and A. Sala-jegheh, "Web service composition methods: A survey," in *Proc. International MultiConference of Engineers AND Computer Scientists*, vol. 1. 2012.
[4] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," 2005.
[5] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Applied Soft Computing Journal*, 2008, pp. 687-697.
[6] D. Karaboga, B. Gorkemli, C. Ozturk *et al.*, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," *Artificial Intelligence Review*, 2014, vol. 42, no. 1, pp. 21-57.
[7] N. B. Mabrouk, S. Beauche, E. Kuznetsova *et al*., "QoS-Aware service composition in dynamic service oriented environments," presented at the ACM/IFIP/USENIX 10th international conference on Middleware. ACM, 2009.
[8] C. Wan, C. Ullrich, L. Chen, R. Huang, J. Luo, and Z. Shi, "On solving QoS-aware service selection problem with service composition," in *Proc. 2008 Seventh International Conference on Grid and Cooperative Computing*, Shenzhen, 2008, pp. 467-474.
[9] A. Mohammad, T. Risse, and W. Nejdl, "A hybrid approach for efficient Web service composition with end-to-end QoS constraints," *ACM Transactions on the Web*, 2012, pp. 1-31.
[10] C. Rajeswary, "A survey on efficient evolutionary algorithms for Web service selection," *International Journal of Managment It & Engineering*, pp. 177-191, 2012.
[11] K. Dervis and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, pp. 459-471, 2007.
[12] M. S. Kiran, H. Iscan, and M. Giindiiz, "The analysis of discrete artificial bee colony algorithm with neighborhood operator on traveling salesman problem," *Neural Computing and Applications*, pp. 1-23, 2012.
[13] W.-F. Gao and S.-Y. Liu, "A global best artificial bee colony algorithm for global optimization," *Journal of Computational and Applied Mathematics*, vol. 236, pp. 2741-27535, 2012.
[14] G. Kousalya, D. Palanikkumar, and P. R. Piriyankaa, "Optimal web service selection and composition using multi-objective bees algorithm," *International Journal on Information*, vol. 14, no. 14, 2011.
[15] Y. H. Yi *et al*., "Discrete gbest-guided artificial bee colony algorithm for cloud service composition," *Applied Intelligence*, pp. 661-678, 2015.
[16] J. He, L. Chen, X. Wang *et al*., "Web service composition optimization based on improved artificial bee colony algorithm," *Journal of Networks*, 2013, vol. 8, no. 9.
[17] R. Liu, Z. Wang, and X. Xu, "Parameter tuning for ABC-based service composition with end-to-end QoS constraints," in *Proc. 2014 IEEE International Conference on Web Services*, pp. 590-597.
[18] X. Wang, Z. Wang, and X. Xu, "An improved artificial bee colony approach to QoS-aware service selection," in *Proc. the 2013 IEEE 20th International Conference on Web Services*, pp 395–402.
[19] V. R. Chifu, C. B. Pop, I. Salomie *et al.*, "Bio-inspired methods for selecting the optimal web service composition: Bees or cuckoos intelligence?" *International Journal of Business Intelligence and Data Mining*, pp. 321-344, 2011.
[20] D. Fadl, K. E. Hindi, and A. Ghoneim, "Enhanced artificial bee colony algorithm for QoS-aware web service selection problem," *Computing*, pp. 507-517, 2017.
[21] H. Faris, I. Aljarah, M. A. Al-Betar, and S, Mirjalili, "Grey wolf optimizer: a review of recent variants and applications," *Neural Computing and Applications*, pp. 1-23, 2018.
[22] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Transactions on Evolutionary Computation*, 2009, vol. 13, no. 3, pp. 526-53.

Shudong Zhang was born in 1969. He is a professor and now is with the College of Information Engineering, Capital Normal University. He graduated from Beijing Institute of Technology with a bachelor's degree in engineering in 1993; graduated with a master's degree in science from China Academy of Engineering Physics in 1996; graduated from Beijing Institute of Technology with a doctorate in engineering in 2005; From 2005 to 2007, he worked as a postdoctoral researcher in institute of software, Chinese Academy of Sciences. His main research interests include sensor network system and computer software.

Yaru Shao was born in 1996. She is a graduate student in the School of Information Engineering Capital Normal University. Her main research interests include service computing distributed computing and software reliability analysis.

Lijuan Zhou was born in 1969. She is a professor and now is with the College of Information Engineering, Capital Normal University. She graduated from Harbin Engineering University with a major in computer science and technology, obtained a doctorate in engineering and was promoted to a professor in 2004. Her main research interests include data warehouse and business intelligence, data mining and intelligent analysis, database systems and applications.