TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 10/11 pp90-99
Volume 20, Number 1, February 2015

An Optimization Algorithm for Service Composition Based on an Improved FOA

Yiwen Zhang, Guangming Cui, Yan Wang, Xing Guo, and Shu Zhao*

Abstract: Large-scale service composition has become an important research topic in Service-Oriented Computing (SOC). Quality of Service (QoS) has been mostly applied to represent nonfunctional properties of web services and to differentiate those with the same functionality. Many studies for measuring service composition in terms of QoS have been completed. Among current popular optimization methods for service composition, the exhaustion method has some disadvantages such as requiring a large number of calculations and poor scalability. Similarly, the traditional evolutionary computation method has defects such as exhibiting slow convergence speed and falling easily into the local optimum. In order to solve these problems, an improved optimization algorithm, WS_FOA (Web Service composition based on Fruit Fly Optimization Algorithm) for service composition, was proposed, on the basis of the modeling of service composition and the FOA. Simulated experiments demonstrated that the algorithm is effective, feasible, stable, and possesses good global searching ability.

Key words: service composition; Fruit Fly Optimization Algorithm (FOA); Quality of Service (QoS) index

1 Introduction

In the late 1990s, with the development of distributed technology and eXtensible Markup Language (XML) technology, the service composition technology appeared. Its core idea is to serve as the basic unit, and then rapidly construct a loosely coupled, distributed application by combining basic services[1]. Service composition is a process that assembles a certain number of services in accordance with business logic. It achieves business objectives through the implementation of the combined service. Web service absorbed the advantages of distributed computing, grid computing, and XML technologies, and was applied successfully to solving the problem of heterogeneous distributed computing, as well as the reuse of code and data[2]. In the meantime, it also has high interoperability, platform independence, loose coupling, and highly integrated capabilities. However, a single web service provides limited functionality. Service-Oriented Computing (SOC) enables the composition of these services in a loosely coupled way in order to achieve complex functionality.

While academia and business have paid much attention to service composition technologies, numerous executable service composition schemes have been produced. These can be described as the combination of a service function and Quality of Service (QoS), in which researchers mostly adopt the exhaustive and evolutionary algorithms as the calculation method based on QoS attributes. Exhaustive methods have poor extension and require a large amount of calculation[3]. They are generally applicable to static situations and have small-scale service composition. As the number of web service publications raises sharply, evolutionary computation algorithms provide a better solution. In particular, the Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) algorithm are two typical algorithms that are currently used in the research of service composition optimization. Compared with the former, PSO has the positive characteristics of

• Yiwen Zhang, Guangming Cui, Yan Wang, Xing Guo, and Shu Zhao are with the School of Computer Science and Technology, Key Laboratory of Intelligent Computing and Signal Processing, Ministry of Education, Anhui University, Hefei 230601, China. E-mail: zhaoshuzs2002@hotmail.com.
* To whom correspondence should be addressed.
Manuscript received: 2014-12-07; accepted: 2014-12-25

possessing few parameters and fast convergence speed. It shows good search capabilities in many optimization problems, but has the defect of easily falling into the local optimum[4,5]. Based on this, the paper puts forward an optimization algorithm for Web Service composition based on the improved Fruit Fly Optimization Algorithm (FOA), denoted as WS_FOA. Compared with PSO algorithm, this algorithm has a few advantages: fewer parameters, faster convergence speed, and less running time. Further, WS_FOA has good global searching ability.

2 FOA

FOA is a new method for finding global optimization based on the foraging behavior of the fruit fly. The fruit fly itself is superior to other species in sensing and perception, especially in osphresis and vision. Based on the fruit fly and its foraging process, FOA was proposed[6]. The food finding behavior of the fruit fly is illustrated in Fig. 1 and the implementation of FOA follows.

(1) Initialize parameters: given the size of swarm (Sizepop) and the maximum of iteration times (Maxgen), initialize randomly fruit fly swarm location (x_{axis}, y_{axis}).

(2) Give the direction and distance (Value) for individual fruit fly searching for food using osphresis, which is random. (Value zone is [−1, 1] in the FOA test).

$$\begin{cases} x_i = x_{\text{axis}} + \text{Value}; \\ y_i = y_{\text{axis}} + \text{Value} \end{cases} \tag{1}$$

(3) Since the food location cannot be known, the distance (Dist_i) to the origin is thus estimated first. Then the smell concentration judgment value (Smell_i) is calculated, and this value is the reciprocal of distance.

$$\text{Dist}_i = \sqrt{x_i^2 + y_i^2} \tag{2}$$

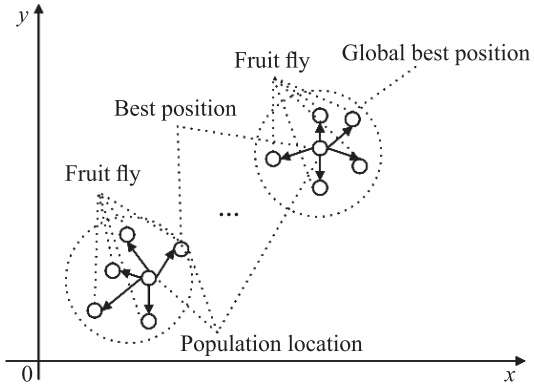


Fig. 1 The food finding behavior of the fruit fly.

$$\text{Smell}_i = \frac{1}{\text{Dist}_i} \tag{3}$$

(4) Substitute smell concentration judgment value (Smell_i) into smell concentration judgment function (also called fitness function) to find the smell concentration (Fitness_i) of the individual location of the fruit fly.

$$\text{Fitness}_i = \text{Function}(\text{Smell}_i) \tag{4}$$

(5) Find the fruit fly with minimal smell concentration (finding the minimal value) among the fruit fly swarm.

$$[\text{bestFitness bestindex}] = \min(\text{Fitness}_i) \tag{5}$$

(6) Record and keep the best smell concentration value and its coordinate (x, y), and at this moment, the fruit fly swarm will use vision to fly towards that location.

$$\begin{cases} \text{Fitnessbest} = \text{bestFitness}; \\ x_{\text{axis}} = x(\text{bestindex}); \\ y_{\text{axis}} = y(\text{bestindex}) \end{cases} \tag{6}$$

(7) Enter the iterative optimization process to repeat the implementation of Steps 2–5. Then, determine if the smell concentration is superior to the previous iterative smell concentration and the current number of iterations is smaller than the maximum number of iterations (Maxgen). If so, implement Step 6.

3 The Problem of Modeling

The optimization algorithm proposed in this paper is based on two specific experimental models. One is the service composition model, and the other is the WS_FOA algorithm model. By constructing the web service model, various attributes of web service can be clearly defined and used and a quantitative index can be specified in the experiment. Through the WS_FOA algorithm model, the practical significance in service composition of every parameter in FOA can be clearly described. This is convenient for algorithm implementation.

3.1 Service composition modeling

Definition 1 QoS

Let $\text{QoS} = \{A, T, C, R\}$, where A represents availability, T represents response time, C represents cost, and R represents reputation[7].

(1) Availability (A)

Availability refers to the probability that a web service is accessible, namely, the proportion of successful web service visits in the period of time t.

Let $A = t_{\text{success}}/t$, where the t_{success} represents the time of successfully accessing the service within the time period t. A is a number in the range [0, 1].

(2) Response time (T)

Response time refers to the expected delay between the time when a service requester sends a service request and the time when the result is obtained. It mainly includes the web service execution time T_{exe}, network transmission time T_{trans}, and other time consumption T_{oth}. Hence, the response time is evaluated as $T = T_{\text{exe}} + T_{\text{trans}} + T_{\text{oth}}$.

(3) Cost (C)

Cost refers to the fee that a service requester has to pay to the service provider for the service invocation. It mainly includes the requisite basis fee C_{serv}, such as software (SaaS), hardware (IaaS), and platform (PaaS)[8], and the service management fee C_{manag}. It can be described as $C = C_{\text{serv}} + C_{\text{manag}}$.

(4) Reputation (R)

Reputation refers to the trustworthy measure of the web service, which is mainly based on the experience of users after using the web service. Different users may have different evaluations on the same web service. Let $R = \sum_{i=1}^{n} R_i/n$, where R_i represents the evaluation of web services of the i-th end-user. R is a number in the range [0, 1].

Definition 2 Web service (s)

Web service is a four-tuple. Let $s = \{\text{ID}, \text{Source}, \text{Function}, \text{QoS}\}$, where ID is the unique identification of a web service, Source is the describing information of service name and publisher, etc., Function is a functional description of the Web service, and QoS is the description of the quality of the web service.

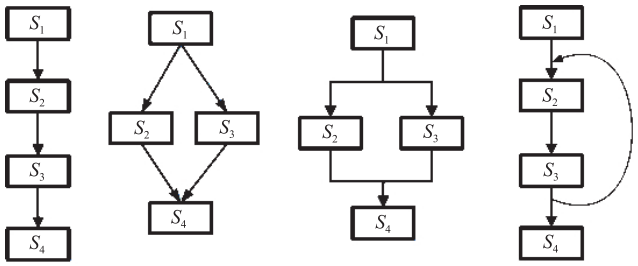
Definition 3 Service Composition (SC)

Let $\text{SC} = (s_1, s_2, s_3, \cdots, s_{n-1}, s_n)$, where $s_1 - s_n$ represent each web subservice respectively, and $s_1 \in S_1, s_2 \in S_2, s_3 \in S_3, \cdots, s_{n-1} \in S_{n-1}, s_n \in S_n$. At the same time, S_1 to S_n are called sets of subservices of service composition. Subservices in S_i have the same function, but have different service qualities. Thus, it is concluded that $\text{SC} \in S_1 \times S_2 \times S_3 \times \cdots \times S_{n-1} \times S_n$.

The execution paths of a composite service can be composed of four basic composition patterns, such as sequential, selection, parallel, and loop (see Fig. 2)[9]. The formulas of all patterns are as follows.

(1) Sequential model (Fig. 2a). QoS is found by Eqs. (7).

(2) Selection model (Fig. 2b). Suppose that the selected probability of each service S_i is λ_i, then $\sum_{i=1}^{n} \lambda_i = 1$. Then, QoS is found by Eqs. (8).



(a) Sequential (b) Selection (c) Parallel (d) Loop

Fig. 2 Four basic composition patterns for service composition.

$$\begin{cases} A = \prod_{i=1}^{n} A_i; \\ R = \sum_{i=1}^{n} R_i/n; \\ C = \sum_{i=1}^{n} C_i; \\ T = \sum_{i=1}^{n} T_i \end{cases} \tag{7}$$

$$\begin{cases} A = \prod_{i=1}^{n} (A_i \times \lambda_i); \\ R = \sum_{i=1}^{n} (R_i \times \lambda_i); \\ C = \sum_{i=1}^{n} (C_i \times \lambda_i); \\ T = \sum_{i=1}^{n} (T_i \times \lambda_i) \end{cases} \tag{8}$$

(3) Parallel model (Fig. 2c). Suppose n services are executed in parallel. Then, QoS is found by Eqs. (9).

(4) Loop model (Fig. 2d). Suppose the loop model is carried out θ times. Then, QoS is found by Eqs. (10).

$$\begin{cases} A = \prod_{i=1}^{n} A_i; \\ R = \sum_{i=1}^{n} R_i/n; \\ C = \sum_{i=1}^{n} C_i; \\ T = \max(T_i), i \in [1, n] \end{cases} \tag{9}$$

$$\begin{cases} A = \prod_{i=1}^{n} A_i; \\ R = \sum_{i=1}^{n} R_i / n; \\ C = \theta \times \sum_{i=1}^{n} C_i; \\ T = \theta \times \sum_{i=1}^{n} T_i \end{cases} \tag{10}$$

Definition 4 QoS pretreatment

Different types of indexes have different dimensions. In order to eliminate the incommensurability stemming from different dimensions and different dimensional units, all indexes need to be normalized to a dimensionless interval according to a certain utility function (usually it is normalized to [0,1]) before the cooperative evaluation. This paper only considers the discrete indexes and normalizes them according to the following formula.

$$q_{ij} = \begin{cases} (q_j^{\max} - q_{ij})/(q_j^{\max} - q_j^{\min}), & \text{benefit\_type;} \\ (q_{ij} - q_j^{\min})/(q_j^{\max} - q_j^{\min}), & \text{cost\_type} \end{cases} \tag{11}$$

where $q_j^{\max}$ and $q_j^{\min}$ represent the current maximum and minimum value of the j-th evaluation index respectively. Their value will change dynamically with the joining or withdrawing from a web service.

3.2 Algorithm modeling

3.2.1 Individual fruit fly

In the experiment, service composition is a five-tuple, namely each record of SC contains five subservice processes. Therefore, when each subservice is regarded as a single individual fruit fly, each SC contains five individual fruit flies, which belong to different subservice sets, respectively. In this way, we can do an iteration with the five flies separately by using FOA, and then find the optimal solution by using the fitness function value of each composition. Finally, the location of the five flies in the optimal service set is just the location of each subservice.

3.2.2 Code of individual fruit fly and motion equation of fruit fly

The code for the individual fruit fly uses a discrete method; it adopts the method of integer encoding. The integer represents the location of the fruit fly in its set of subservices. Motion equation (F) is the direction and the size according to which the fruit fly swarm can optimally move. That is, $F = V \times t, (t \in [-1, 1])$. V denotes the mobile step, meaning the range where the fruit fly could move once. Thus, the position equation of the individual fruit fly is as follows.

$$P_k^{i+1,j} = P^{i,j_{\text{best}}} + F \tag{12}$$

The $P_k^{i+1,j}$ represents the location of the k-th fruit fly of the j-th fly swarm in the $(i+1)$-th iterative process. $P^{i,j_{\text{best}}}$ represents the optimal location of the j-th fly swarm in the i-th iterative process, which is recorded as the seed for the individual fruit fly of the j-th fly swarm in the i-th iterative process. In other words, it is the location of the fruit fly swarm in the next iterative process.

3.2.3 Definition of distance

In each iteration of FOA for each fruit fly, use the reciprocal of the offset which is the location of the fruit fly in its set of subservices related to the 0-th record as the distance. After that, the value of smell concentration judgment is calculated by using the relation between distance and smell concentration judgment value, and then the fitness function(Fitness) is calculated. The distance (Dist) and the smell concentration judgment value (Smell) are defined as

$$\begin{cases} \text{Dist}_k^{i+1,j} = 1/P_k^{i+1,j}; \\ \text{Smell}_k^{i+1,j} = 1/Dist_k^{i+1,j} \end{cases} \tag{13}$$

$P_k^{i+1,j}$ is the location where the individual fruit fly has shifted in its set of subservices. $\text{Dist}_k^{i+1,j}$ denotes the distance of the k-th fruit fly of the j-th fly swarm in the $(i+1)$-th iterative process. $\text{Smell}_k^{i+1,j}$ represents the smell concentration judgment value of the k-th fruit fly of the j-th fly swarm in the $(i+1)$-th iterative process.

3.2.4 Fitness function

The fitness function is used to calculate the fitness of each SC. The advantages and disadvantages of each SC are judged by the fitness values. Combined with the results of QoS after preprocessing in Eqs. (11), the fitness function can be defined as

$$\text{fitness} = \sum_{j=1}^{n}\sum_{k=1}^{m} C_j W_k Q_{jk} = \sum_{j=1}^{n}\sum_{k=1}^{m} C_j W_k \text{Smell}_k^{i,j} \tag{14}$$

Where $\text{Smell}_k^{i,j}$ represents the k-th component index of the i-th fruit fly of the j-th swarm in the i-th iterative process. C_j represents the weight of the j-th subservice in the SC. Q_{jk} represents the k-th component index of the j-th subservice. W_k represents the weight of the k-th component index. n denotes the number of the subservice. m denotes that each subservice contains m QoS index.

4 Implementation of WS_FOA

4.1 Experimental scheme

WS_FOA is executed based on the iterative operation of multiple fruit fly swarms. Iterative operation of each fruit fly swarm stands for the correlative operation of each set of subservices, including the position change of fruit flies and the optimization process. For example, in the experiment, an SC is constituted of five subservices, which are marked as $s_1 - s_5$. Therefore, there are five fruit fly swarms labeled as $G_1 - G_5$. G_i is a swarm whose practical background and significance are S_i and WS_FOA algorithm is executed accordingly with G_i. In the WS_FOA optimization process, WS_FOA algorithm is executed using the swarms $G_1 - G_5$ respectively, and then each record of SC is constituted of the corresponding position in $G_1 - G_5$. For example, the n-th individual fruit fly is chosen in $G_1 - G_5$ respectively, so the combined record for the service composition is $\text{SC} = (S_1^{G_1}, S_2^{G_2}, S_3^{G_3}, S_4^{G_4}, S_5^{G_5})$. $S_i^{G_i}$ denotes the subservice cataloged as G_i in the i-th set of subservices. Then, calculate each SC consisting of $G_1 - G_5$ according to Eq. (14); the optimal SC in the iterative process contains the subservice which is regarded as the seeded individual fruit fly in the corresponding swarm. Finally, calculate the next fruit fly swarm and proceed to the next iteration according to Eq. (12), until the iteration is over or the end condition is met.

4.2 WS_FOA algorithm

The established QoS index system has different dimensions. In order to eliminate the incommensurability stemming from the different dimensions and dimensional units, combined with Definition 4, the experimental data should be preprocessed before the experiment. Then, the data is optimized according to the WS_FOA algorithm. This algorithm includes two parts. The main steps are described as follows and their implementation procedures are illustrated in Figs. 3 and 4.

(1) QoS preprocessing algorithm

Input: The original index information of set of subservices

Output: The normalized index information of set of subservices

Step 1: Obtain the index information of each set of subservices.

Step 2: Obtain the maximum value and minimum value of each index in the set of subservices.

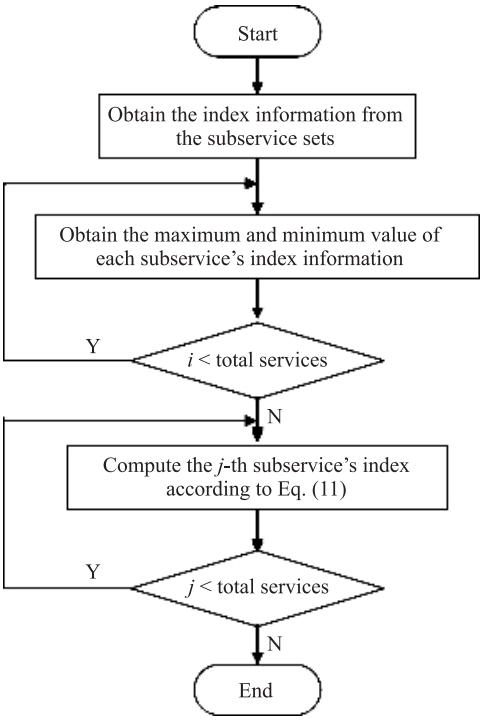


Fig. 3 The implement procedure for QoS preprocessing.

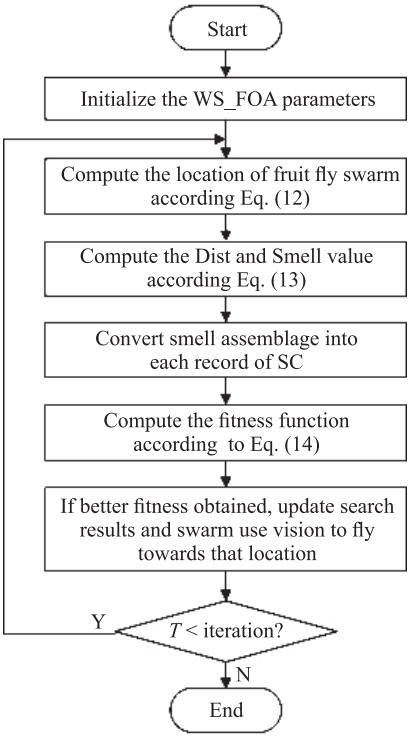


Fig. 4 The implement procedure of the WS_FOA.

Step 3: For each subservice record in set of subservices, A and R indexes are calculated using Eqs. (11) according to the equation of benefit, and C and T indexes are calculated using Eqs. (11) according

to the equation of cost.

Step 4: Repeat Steps 2–4, until each set of subservices is totally disposed.

Step 5: Output the index information of each set of subservices after normalization.

(2) Description of WS_FOA algorithm

Input: the normalized set of subservice

Output: the location of individual fruit fly in optimal SC

Step 1: Initialize the location ($P^{0,1_{\text{best}}} - P^{0,n_{\text{best}}}$) of seeded fruit fly in each set of subservices.

Step 2: Calculate the location of fruit fly swarm according to Eq. (12).

Step 3: Calculate distance and smell value of individual fruit fly in each set of subservices according to Eq. (13).

Step 4: Convert smell assemblage into each record of SC.

Step 5: Calculate each record of SC according to Eq. (14). Record the optimal SC in the iterative process, and treat each subservice in the combined record as seeded fruit fly $P^{i,j_{\text{best}}}$ in its corresponding swarm. Compare the optimal value in this iteration with the global optimal value and reserve the best location.

Step 6: Repeat Steps 2–5, until the iteration is over or the demand for accuracy is met.

Step 7: Output the optimal composition.

5 Comparative Study

5.1 PSO algorithm

The PSO algorithm is a heuristic evolutionary algorithm, which was proposed by Eberhart and Kennedy in 1995 and inspired by the social behavior of bird flocking or fish schooling[10]. Each particle's position represents a solution of optimization problems in PSO. The particle's position changes according to two extreme values: One is the historical optimal solution of the particles (pbest) and the other is the global optimal solution of species[11].

$$v_i^d(t+1) = w(t)v_i^d(t) + C_1\text{rand}_i(t)(\text{pbest}_i^d - x_i^d(t)) + C_2\text{rand}_i(t)(\text{gbest}^d - x_i^d(t)) \quad (15)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (16)$$

where $\text{pbest}_i = (\text{pbest}_i^1, \text{pbest}_i^2, \cdots, \text{pbest}_i^n)$ and $\text{gbest} = (\text{gbest}^1, \text{gbest}^2, \cdots, \text{gbest}^n)$ represent the best previous position of the i-th particle and the best previous position among all of the particles in the population respectively. $\text{rand}_i(t)$ is a uniform random variable within the interval [0, 1] at time t. C_1 and C_2 are two acceleration constants. The initialization process for the velocities and positions of the particles is conducted by using vectors of random numbers that are in the corresponding range.

5.2 Comparison of algorithms

Both PSO and FOA belong to a class of intelligent optimization algorithms. They are evolutionary algorithms that are deduced from animal foraging rules in nature. They also have high optimization efficiency in the search space and obtain the optimal solution on the basis of the next evolutionary particle position change. However, there are many differences between them that are described as follows.

(1) Different background. The FOA is a new method for global optimization, based on the foraging behavior of fruit flies. Instead, the PSO algorithm is an evolutionary algorithm based on group social behavior inspired by bird flocking or fish schooling.

(2) Different optimization path. When a fruit fly finds a better position in the FOA, the whole population moves to that location, while in the PSO algorithm each individual dynamically changes their position based on the optimization position and global optimization position.

(3) Different motion equation. The FOA produces individual fruit flies by using population location, which is combined with random values whithin a certain range, while the PSO algorithm changes the speed and position of individual particles with Eqs. (15) and (16).

(4) Different parameter. The FOA calculates the smell using the distance between an individual fruit fly and its origin, using only a few parameters. However, the PSO algorithm contains many more parameters, such as inertia weight, learning factor, etc.

6 Experimental Analysis

6.1 Experiment settings and data sets

This experiment adopted the extendible QoS model proposed by Liu et al.[12] and the QWS data set provided by Zeng et al.[13,14] The value of C_j in Eq. (14) and is 1, namely each subservice accounts for the same percentage. The value of W_k is {0.2, 0.3, 0.2, 0.3}, which corresponds to the four QoS indexes in Definition 1. The experiment is carried out in the Windows 7 OS, and the Visual Studio 2010 compiler environment. Data processing is accessed with the text document

as carrier. Another experimental dataset is a random data set (RWS), where the random function produces a whole number in the range of [0, 1000]. Then the whole number is divided by ten, so the data is in the range of [0, 100], including a decimal. After repeating this many times, we obtain 2500 groups of experimental data. The dataset is divided equally into five candidate service sets that are used as subservice sets. Finally the candidate service sets are normalized in the following experiment.

6.2 Comparison of experiment results

In order to verify the performance of WS_FOA, efficiency, feasibility, and stability experiments have been carried out and the results were compared with those of the PSO algorithm. The results of the experiments are described as follows.

6.2.1 Efficiency

In order to analyze the efficiency of the WS_FOA algorithm, we make two comparisons: the average runtime of optimizing 50 times (Fig. 5) over different population sizes and the same number of iterations (500), and the average runtime of optimizing 50 times (Fig. 6) over the same population size (100) and different numbers of iterations.

Shown in Figs. 5 and 6, the gradual increase of population size and the number of iterations increases the disparity of time between WS_FOA and the PSO algorithm. Thus, when the population size or the number of iterations is large, WS_FOA exhibits the characteristics that PSO does not possess. Meanwhile, hosts of experiments have indicated that in the case of the same population size and the number of iterations, the WS_FOA algorithm has higher operational efficiency than PSO.

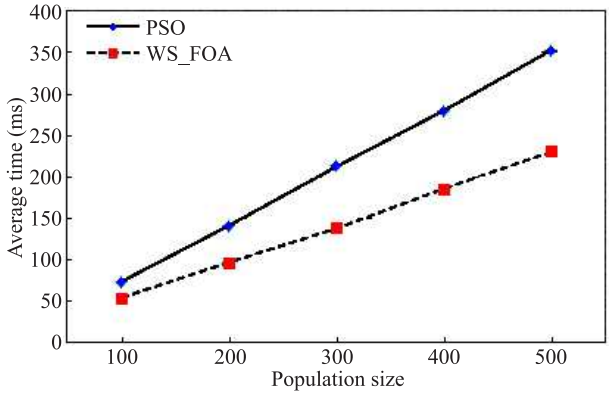


Fig. 5 Average runtimes over different population sizes and the same number of iterations.

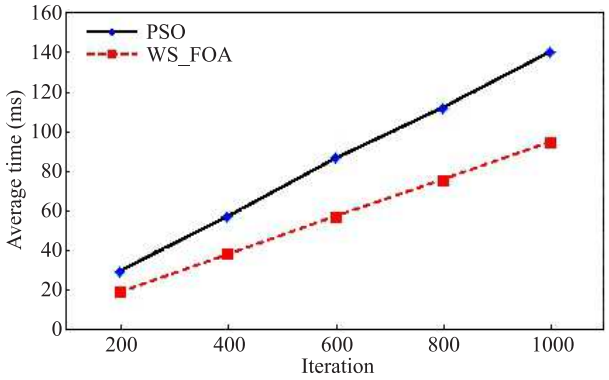
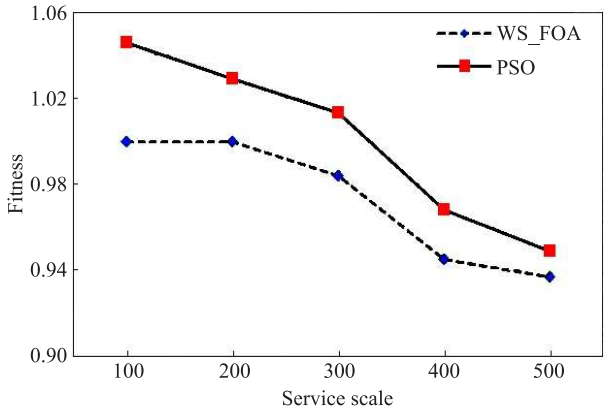


Fig. 6 Average runtimes over same population size and the different numbers of iterations.

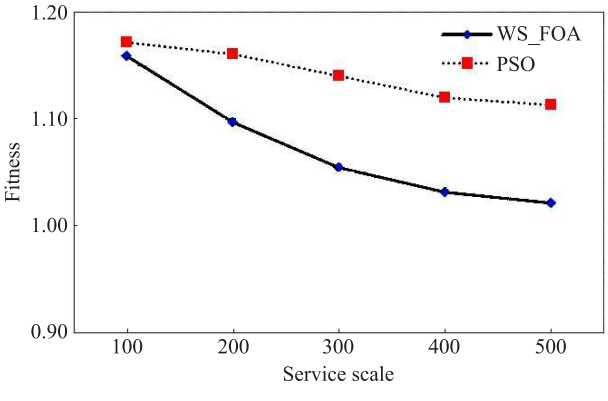
6.2.2 Feasibility

In order to validate the global searching capability of the WS_FOA algorithm, we compare it to the optimizing result of the PSO algorithm over the same population size (120) and iterations (500), but with different service scale (Fig. 7).

It is clear that the WS_FOA optimization is superior to that of traditional PSO in the same service scale.



(a) QWS



(b) RWS

Fig. 7 Comparative diagram of optimizing results on the same service scale.

This paper shows that not only does WS_FOA possess higher searching efficiency, but also possesses better optimizing effects than those of traditional PSO. Therefore, WS_FOA has fine feasibility.

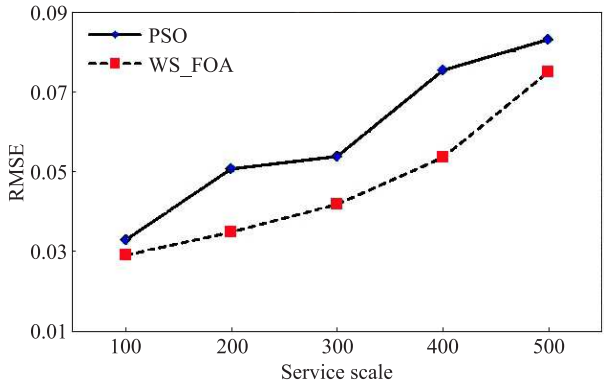
6.2.3 Stability

In order to compare the degree of dispersion of the results over 50 trials in repeated experiments between the WS_FOA and PSO algorithms, the Root-Mean-Square Error (RMSE) has been proposed[15]. The formula is as follows:

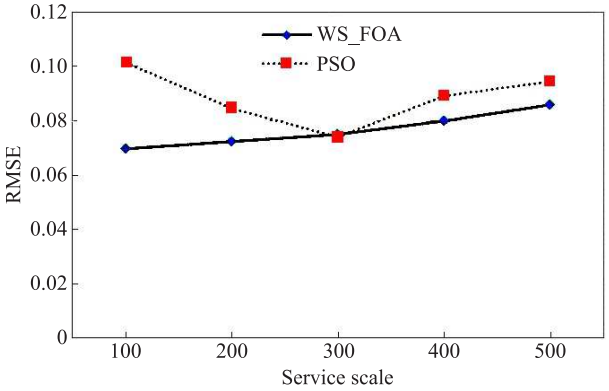
$$\mathrm{RMSE} = \sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2/n} \qquad (17)$$

The $\overline{X}$ represents the average of n times repeated test results; X_i represents the i-th test result. According to Eq. (17), the result is shown in Fig. 8.

Obviously, it is easy to see the stability of WS_FOA is higher than that of PSO in the two data sets above. Combined with Fig. 7, the results show that the WS_FOA's optimization performance is significantly better than PSO.



(a) QWS



(b) RWS

Fig. 8 Analysis result of RMSE.

7 Related Work

A number of studies of SC have been published in recent years. In this section, we survey the current approach for SC. Conventional SC approaches can be classified into the following three categories, all of which assume a predefined composition workflow pattern, exhaustive search, classic optimization decision method, and evolution approximation algorithm.

(1) Exhaustive search. This approach[16,17] tries to enumerate all possible combinations by using candidate web services for each task. As a result, a service composition with the optimal QoS value for a predefined workflow pattern can be selected, if one exists and satisfies all global QoS constraints. However, the time complexity of this approach is unacceptably high, i.e., $O(m^n)$, where m and n are the maximum number of candidate services for a task and the number of tasks in a workflow, respectively. Thus, this approach is only suitable for small-scale service composition situations.

(2) Classic optimization decision method. SC is a problem in which there are many potential solutions, among which, one or a limited number of solutions are optimal. Thus, SC is known as an optimization problem. Some types of classic optimization decision methods, such as the Analytic Hierarchy Process (AHP)[18], Linear Programming (LP)[19], dynamic programming[20], and TOPSIS[21], can be used to solve this problem. In general, for generating optimized service composition using this method, a several phase or multi-layer hierarchical structure should be divided. Then a multiple criteria decision-making process is applied to calculate the QoS value of candidate services, using the weights assigned to each QoS criterion. Although this approach is locally optimal and efficient with a low time complexity, it does not guarantee to satisfy global QoS constraints. Thus, using classic algorithms to solve optimization problems is possible only with some modifications and improvements for decreasing the execution time.

(3) Evolution approximation algorithm. As the number of distributed services is rising rapidly, particularly in the cloud, selecting the best fit services for a given task becomes more challenging. To decrease the time complexity, researchers model the SC problem as multi-objective combinatorial optimization problem, which is an NP-complete problem. An evolution approximation algorithm can be used to generate a

composite service with a suboptimal QoS value, such as genetic algorithm[22], PSO[23], parallel optimization algorithm[24], etc. This approach can obtain the approximate optimal solution within an acceptable range of time, however, it is easy to fall into a local optimum.

8 Conclusions

An improved optimization algorithm for service composition, WS_FOA, is put forth in this paper. Compared with the traditional PSO algorithm, the WS_FOA has faster running speed. The WS_FOA algorithm reflects the irreplaceable advantages that PSO does not possess, especially in the case of large service scale. Meanwhile, hosts of experiments indicate that the optimizing effects of the WS_FOA algorithm are better than those of traditional PSO. For the time being, it is far superior to the traditional PSO algorithm. This paper lays a foundation for optimization research of enormous service composition in the big data era.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Nos. 61402006 and 61202227), the Natural Science Foundation of Anhui Province of China (No. 1408085MF132), the Science and Technology Planning Project of Anhui Province of China (No. 1301032162), the College Students Scientific Research Training Program (No. KYXL2014060), and the 211 Project of Anhui University (No. 02303301).

References

[1] M. Papazoglou and D. Georgakopoulos, Service-oriented computing, *Communications of the ACM*, vol. 46, no. 10, pp. 25–28, 2003.

[2] J. F. Chen, H. H. Wang, D. Towey, C. Y. Mao, R. B. Huang, and Y. Z. Zhan, Worst-input mutation approach to web services vulnerability testing based on SOAP messages, *Tsinghua Science and Technology*, vol. 19, no. 5, pp. 429–441, 2014.

[3] T. Wen, G. J. Sheng, Q. Guo, and Y. Q. Li, Web service composition based on modified particle swarm optimization, (in Chinese), *Chinese Journal of Computers*, vol. 36, no. 5, pp. 1031–1045, 2013.

[4] A. Yadav and K. Deep, Shrinking hypersphere based trajectory of particles in PSO, *Applied Mathematics and Computation*, vol. 220, no. 9, pp. 246–267, 2013.

[5] S. Panda, B. Mohanty, and P. K. Hota, Hybrid BFOA-PSO algorithm for automatic generation control of linear and nonlinear interconnected power systems, *Applied Soft Computing*, vol. 13, no. 12, pp. 4718–4730, 2013.

[6] W. T. Pan, A new fruit fly optimization algorithm: Taking the financial distress model as an example, *Knowledge Based Systems*, vol. 26, no. 2, pp. 69–74, 2012.

[7] X. Q. Fan, C. J. Jiang, J. L. Wang, and S. C. Pang, Random-QoS aware reliable web service composition, (in Chinese), *Journal of Software*, vol. 20, no. 3, pp. 546–556, 2009.

[8] S. Xiang, B. Zhao, A. Yang, and T. Wei, Dynamic measurement protocol in infrastructure as a service, *Tsinghua Science and Technology*, vol. 19, no. 5, pp. 470–477, 2014.

[9] H. Al-Helal and R. Gamble, Introducing replaceability into web service composition, *IEEE Transaction on Services Computing*, vol. 7, no. 2, pp. 198–209, 2014.

[10] R. Eberhart and J. Kennedy, A new optimizer using particle swarm theory, in *Proceedings of the 7th International Symposium on Micro Machine and Human Science*, IEEE Service Center, Piscataway, NJ, USA, 1995, pp. 39–43.

[11] S. Moein and R. Logeswaran, KGMO: A swarm optimization algorithm based on the kinetic energy of gas molecules, *Information Sciences*, vol. 275, pp. 127–144, 2014.

[12] 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*, New York, USA, 2004, pp. 66–73.

[13] L. Zeng, B. Benatallah, and M. Dumas, Quality driven web service composition, in *Proceedings of the 12th International Conference on World Wide Web*, ACM, New York, USA, 2003, pp. 411–421.

[14] L. Zeng, B. Benatallah, and A. H. Ngu, QoS-aware middleware for web service composition, *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 311–327, 2004.

[15] M. Silic, G. Delac, I. Krka, and S. Srbljic, Scalable and accurate prediction of availability of atomic web services, *IEEE Transaction on Services Computing*, vol. 7, no. 2, pp. 252–264, 2014.

[16] B. Wu, C. Chi, and S. Xu. Service selection model based on QoS reference vector, in *Proceeding of the IEEE Congress Services*, Salt Lake City, UT, USA, 2007, pp. 270–277.

[17] M. Alrifai and T. Risse, Combining global optimization with local selection for efficient QoS-aware service composition, in *Proceeding of World Wide Web*, Madrid, Spain, 2009, pp. 881–890.

[18] A. Ishizaka and A. Labib, Review of the main developments in the analytic hierarchy process, *Expert Systems with Applications*, vol. 38, pp. 14 336–14 345, 2011.

[19] M. S. Hossain, M. M. Hassan, M. Al Qurishi, and A. Alghamdi, Resource allocation for service composition in cloud-based video surveillance platform, in *Proceeding of IEEE Intenational Conference on Multimedia and Expo Workshops*, Melbourne, Australia, 2012, pp. 408–412.

[20] D. Worm, M. Zivkovic, H. van den Berg, and R. van der Mei, Revenue maximization with quality assurance for composite web services, in *Proceeding of the 5th IEEE International Conference on Service-Oriented Computing and Applications*, Taipei, China, 2012, pp. 1–9.

[21] Z. ur Rehman, O. Khadeer Hussain, and F. Khadeer Hussain, Parallel cloud service selection and ranking based on QoS history, *International Journal of Parallel Programming*, vol. 42, no. 5, pp. 820–852, 2014.

[22] A. Klein, F. Ishikawa, and S. Honiden, San-GAA self-adaptive network-aware approach to service composition, *IEEE Transactions on Services Computing*, vol. 7, no. 3, pp. 452–464, 2014.

[23] S. G. Wang, Q. B. Sun, H. Zou, and F. C. Yang, Particle swarm optimization with skyline operator for fast cloudbased web service composition, *Mobile Networks and Applications*, vol. 18, pp. 116–121, 2013.

[24] F. Tao, Y. Laili, L. Xu, and L. Zhang, FC-PACO-RM: A parallel method for service composition optimal-selection in cloud manufacturing system, *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 2023–2033, 2013.

Yiwen Zhang received the master degree in computer software and theory in 2006 and the PhD degree in management science and engineering in 2013 both from the Hefei University of Technology. He is currently an associate professor in the School of Computer Science and Technology at Anhui University. His research interests include service computing, cloud computing, and e-commerce.

Guangming Cui is a master student in the School of Computer Science and Technology, Anhui University. His current research interests include service computing and cloud computing.

Yan Wang is an undergraduate student in the School of Computer Science and Technology at Anhui University. She is majoring in software engineering in Anhui University. Her research interests include service computing and cloud computing.

Xing Guo received the master degree in computer science in 2009 and the PhD degree in 2013 both from Anhui University, China. He is a lecturer in the School of Computer Science and Technology at Anhui University. His research interests include, but are not limited to computer vision, image processing, service computing, and cloud computing.

Shu Zhao received her PhD degree in computer science from Anhui University in 2007. She is now an associate professor in the Department of Computer science and Technology, Anhui University. Her current research interests include quotient space theory, granular computing, and machine learning.