**Global Optimization Algorithm for Cloud Service Composition**

Hongwei YANG†, Member, Fucheng XUE†, Dan LIU†, Li LI†(a), and Jiahui FENG†, Nonmembers

**SUMMARY** Service composition optimization is a classic NP-hard problem. How to quickly select high-quality services that meet user needs from a large number of candidate services is a hot topic in cloud service composition research. An efficient second-order beetle swarm optimization is proposed with a global search ability to solve the problem of cloud service composition optimization in this study. First, the first beetle antennae search algorithm is introduced into the modified particle swarm optimization algorithm, initialize the population by using a chaotic sequence, and the modified nonlinear dynamic trigonometric learning factors are adopted to control the expanding capacity of particles and global convergence capability. Second, modified secondary oscillation factors are incorporated, increasing the search precision of the algorithm and global searching ability. An adaptive step adjustment is utilized to improve the stability of the algorithm. Experimental results founded on a real data set indicated that the proposed global optimization algorithm can solve web service composition optimization problems in a cloud environment. It exhibits excellent global searching ability, has comparatively fast convergence speed, favorable stability, and requires less time cost.

**key words:** web service composition, beetle antennae search algorithm, particle swarm optimization algorithm, global optimization algorithm

### 1. Introduction

With the rapid development of cloud computing, increased computing resources, and constantly updated products from web service providers, the scale of cloud computing’s resource pool has been growing continuously, and the requirements for the service’s computing ability are becoming astronomical. Meanwhile, the utilization ratio of these resources is decreasing gradually. Service composition technology was proposed to increase the utilization ratio of resources and meet the complex and varied task requests of users. The flowchart of service composition is given in Fig. 1.

Composing existing web services in the cloud environment is highly significant for reducing cloud service costs and increasing cloud service quality. The five-dimensional quality of service (QoS) model proposed by AgFlow [1] is frequently adopted in web service composition. QoS is mainly used to indicate the nonfunctional attributes of web service and to distinguish services with the same or similar functions. Typical web service composition methods are classified and systematically discussed service web composition models in [2]. Problems exist in service composition optimization studies with an excessively large scale and poor composition efficiency. Their weak composition organizational compatibility leads to low performance. Moreover, the problem of increasing QoS in service composition has not been effectively solved. Efficiently selecting high-quality service composition with huge composition schemes to meet users’ requirements has become a key problem in the service optimization field, and which plays a leading role in enhancing the development of cloud computing.

Achieving service composition through QoS attribute perception and obtaining composition service with the maximum QoS value via global optimization service is classic NP-hard problems [3]. Thus, the evaluation index of web service composition in the cloud environment can be established following its QoS value. The computation of QoS value has grown up to be a multi-objective optimization problem.

The second part of this paper introduces related works. The third part describes the cloud service composition problems. The fourth part provides a detailed description of the global optimization algorithm proposed in this study. The fifth part presents the experiment and result in analyses of the algorithm. The sixth part summarizes the study.

### 2. Related Works

To solve the problem of web service composition optimization, researchers have proposed different intelligent algorithms. An artificial bee colony algorithm (DGABC) is presented to search for the best cloud service combination solution by stimulating bee foraging [4]. Compared with particle swarm optimization (PSO), DGABC has more parameters and lower convergence ability. An improved...
algorithm is developed to combine a differential evolution algorithm and a social cognitive optimization algorithm to solve the problem of QoS-aware cloud service composition [5]. However, the convergence ability of this combined algorithm is slow, and its local convergence ability is insufficient. A dynamic service composition method is introduced based on QoS by using the ant colony algorithm in the process of service composition optimization [6]. However, the ant colony algorithm requires a large amount of calculation. Setting the parameters improperly will result in gradual solution speed and poor solution set quality. A hybrid approach is proposed for the automatic composition of Web services that generates semantic input-output based compositions with optimal end-to-end QoS, minimizing the number of services of the resulting composition [7]. A mutation-based harmonious search (MBHS) algorithm is therefore proposed to select Web services and compose with the fewest defects [8], but they did not consider the algorithm’s global search capability and algorithm’s stability. TAN Wenan [9] proposed an improved Flower Pollination Algorithm (IFPA) to solve the problem of Web service composition. The genetic algorithm (GA) and PSO are two comparatively typical algorithms in the field of service composition optimization [10]. A dual-objective genetic optimization algorithm suggests in [11], which solved the Web service composition problem in MCE from the perspective of cost and risk. Seghir F [12] proposed a hybrid genetic algorithm (HGA) that combines two phases to perform the evolutionary process search, including the genetic algorithm phase and fruit fly optimization phase. In the genetic algorithm phase, a novel roulette wheel selection operator is therefore proposed to enhance the efficiency and the exploration search. An Improved Genetic Algorithm is applied to Web service composition [13]. An improved genetic algorithm (CGA) [14], which uses a real coding method to solve the QoS-based service selection problem, avoids the negative effects of long chromosomes in the algorithm. However, the genetic algorithm has a slower convergence speed than the particle swarm optimization algorithm, and the CGA does not take into consideration the stability of the algorithm. However, compared with the PSO, GA exhibits several disadvantages, such as poor convergence ability, numerous parameters, and insufficient local convergence ability. A highly efficient service composition technique is put in place to apply in cloud computing. Built on the proxy method, this technique can compose services by identifying QoS parameters, based on the fitness function, it can select the best service by using PSO [15]. An evolutionary algorithm [16], called harmony PSO, which can provide an effective composition with better performance and execution time. Hosseini Shirvani [17] proposed a bi-objective time-varying particle swarm optimisation algorithm for Web service composition. The improved PSO algorithm does not take into consideration the performance and accuracy of the PSO algorithm in high-dimensional solution space and the stability of the optimization ability of this algorithm. Accordingly, we propose a second-order beetle swarm optimization algorithm (SBSO) for web service composition optimization in this study. By expanding the beetle antennae search (BAS) algorithm to population, the PSO’s problem of easily falling into the local best value is solved. By using the trigonometric controlling learning factor, global convergence capability, and rate are increased. By using the chaotic sequence to initialize the population, the solution space range of the algorithm in the initial iteration is expanded. By introducing the modified secondary oscillation factors, the algorithm’s global searching ability is strengthened when dealing with multi-objective optimization problems. The algorithm’s execution time is also shortened. An adaptive step adjustment is utilized to improve the stability of the algorithm.

3. Description of Service Composition Problems in the Cloud Environment

With the development of cloud computing, the number of services in the cloud environment is increasing continually, and a large quantity of cloud services with the same or similar functions but different quality has appeared. When users choose among these services, they consider both their semantic meanings and nonfunctional factors. Therefore, QoS value has grown up to be a key factor that influences service composition. The common key indicators of QoS are availability, reliability, success rate, and execution time [1]. The QoS calculation formula is given in Table 1.

Table 1 QoS calculation formula.

| QoS Indicators | Sequence Structure | Parallel Structure | Selective Structure | Loop Structure |
|----------------|--------------------|--------------------|--------------------|---------------|
| Response Time  | $\sum_{i=1}^{n} T_i$ | $\sum_{i=1}^{n} T_i$ | $\sum_{i=1}^{n} (P_iT_i)$ | $\sum_{i=1}^{n} T_i$ |
| Success Ability| $\prod_{i=1}^{n} S_i$ | $\prod_{i=1}^{n} S_i$ | $\prod_{i=1}^{n} (P_iS_i)$ | $\prod_{i=1}^{n} S_i$ |
| Reliability    | $\prod_{i=1}^{n} R_i$ | $\prod_{i=1}^{n} R_i$ | $\prod_{i=1}^{n} (P_iR_i)$ | $\prod_{i=1}^{n} R_i$ |
| Availability   | $\prod_{i=1}^{n} A_i$ | $\prod_{i=1}^{n} A_i$ | $\prod_{i=1}^{n} (P_iA_i)$ | $\prod_{i=1}^{n} A_i$ |
Equation (1) represents the QoS attributes of the optional services, which can meet users’ needs as Matrix \( Q \). The rows of the matrix represent various QoS attribute values, and the columns of the matrix represent the values of individual attributes. The matrix can be expanded appropriately. Whenever an attribute is added, a column should be added to the matrix.

QoS attributes are divided into two types following their values’ influence on QoS levels: positive and passive QoS attributes. In passive QoS attributes, the larger the values of a QoS attribute, the higher its QoS level will be. In positive QoS attributes, the larger the values of a QoS attribute, the lower its QoS level will be. For example, reliability, success rate, and availability are passive QoS attributes; and execution time and service charge are positive QoS attributes.

Given that the evaluation of each attribute is not of the same order of magnitude, the formulas will be normalized as follows:

\[
Q_j = \begin{cases} 
\frac{Q_{j}^\text{max} - Q_{j}^\text{min}}{Q_{j}^\text{max} - Q_{j}^\text{min}} & Q_{j}^\text{max} \neq Q_{j}^\text{min} \text{ passive type,} \\
1 & Q_{j}^\text{max} = Q_{j}^\text{min}
\end{cases}
\]  

(2)

\[
Q_j = \begin{cases} 
\frac{Q_{j} - Q_{j}^\text{min}}{Q_{j}^\text{max} - Q_{j}^\text{min}} & Q_{j}^\text{max} \neq Q_{j}^\text{min} \text{ positive type,} \\
1 & Q_{j}^\text{max} = Q_{j}^\text{min}
\end{cases}
\]  

(3)

where \( Q_j^\text{max} \) and \( Q_j^\text{min} \) are the maximum and minimum values of the \( j \)-th QoS attribute. With increasing and decreasing cloud services, their values will change dynamically. Passive QoS attribute values are computed and transformed as benefit types in Eq. (2). Meanwhile, positive QoS attribute values are computed and transformed as cost types in Eq. (3).

Definition 2: Service composition is a five-tuple \( SP = (n, m, QoS, w, F) \), \( m = \{m_1, m_2, \ldots, m_i\} \) represents the number of tasks. The value \( n \) represents the number of candidate services. QoS is the quality of service composition. \( w = \{w_1, w_2, \ldots, w_n\} \) is the corresponding weight of each QoS attribute, and simultaneously, it meets \( \sum_{i=1}^{n} w_i = 1 \). \( F \) is the objective computing function. The objective computing function of service composition in this study is defined as follows:

\[
F = \sum_{i=1}^{n} Q_i \times W_i
\]  

(4)

Where \( W_i \) is the weight value of each parameter. The attribute value after standardization is owned by the same order of magnitude. Thus, the maximum value of the objective function is the service of the highest quality of service.

4. The Detail of the Proposed Global Optimization Algorithm

4.1 Swarm Intelligence Algorithm

(1) Standard PSO algorithm

PSO [18] is a heuristic algorithm that is derived from the intelligent behavior simulation of simplified social groups. Each solution of the optimization problem is a position of particles in the space search. By changing particle speed, the flight distance and direction of particles are changed accordingly. Each particle finds the individual optimal \( P_{best} \) and particle in the iterative process. The optimal position of all the particles in the subgroup is \( G_{best} \). The PSO algorithm has a few parameters, fast convergence speed, strong search capability, and good effect on solving high-dimensional multi-objective optimization problems. The updating formulas for particle swarm speed and position are as follows:

\[
v_i^{k+1} = \omega \cdot v_i^k + c_1 \cdot r_1 \cdot (P_{best} - x_i^k) + c_2 \cdot r_2 \cdot (G_{best} - x_i^k)
\]  

(5)

\[
x_i^{k+1} = x_i^k + v_i^{k+1}
\]  

(6)

where \( v_i^{k+1} \) is the velocity of the \( i \)-th particle after the \( k \)-th iteration; \( x_i^{k+1} \) is the position of the \( i \)-th particle after the \( k \)-th iteration; \( r_1, r_2, r_3 \) is a random number with a value of [0, 1]; \( \omega \) is the inertia weight; and \( P_{best} \) and \( G_{best} \) are the individual and group optimal values, respectively.

The position formula is regarded as a function of the number of iterations \( k \) to the position \( x \). Let \( \xi_1 = c_1 r_1, \xi_2 = c_2 r_2, \) and \( \omega = 1 \). The velocity renewal formula, Eq. (11), is regarded as the form of the velocity formula \( V_{r+1} = V_r + a \times \Delta T \). The following equation is derived:

\[
a = \xi_1 (P_{best} - x(k)) + \xi_2 (G_{best} - x(k)) = x(k)''
\]  

(7)

which can be represented as

\[
x(k)'' = (\xi_1 + \xi_2) x(k) - (\xi_1 P_{best} + \xi_2 G_{best}) = 0
\]  

(8)

Equation (8) is a second-order differential equation. By solving this equation, the position path formula for search position in generation \( k \) is as follows:

\[
X(k) = C_1 \cos(\sqrt{\xi_1 + \xi_2}k) + C_2 \sin(\sqrt{\xi_1 + \xi_2}k)
\]  

(9)

Equation (9) shows that the position function \( X(k) \) fluctuates in a fixed interval. That is, finding the position of each particle after \( k \) iterations in the interval, and the position function \( X(k) \) has no oscillation convergence. Thus, the algorithm easily falls into the local optimum.

(2) Beetle antennae search algorithm

The beetle antennae search algorithm (BAS) [19], [20] is an intelligent algorithm for searching for an optimal solution. The creation of this algorithm’s principle was inspired by how beetles find their food. The BAS algorithm can be implemented in various fields. Fan [21] proposed a proportional composite-integral-derivative (PID) controller
that combines the BAS algorithm with PID strategies. Thus, the BAS algorithm was implemented to PID. The beetle antennae search algorithm applies to the micro-grid for energy management [22]. The BAS algorithm is utilized to optimize a back-propagation neural network and predict cement strength [23]. A beetle swarm evolutionary competitive algorithm is proposed in [24], which introduces the swarm evolutionary competition mechanism into BAS to optimize bridge sensors. The BAS algorithm is mostly used in multi-objective optimization problems. Compared with traditional heuristic algorithms, the BAS algorithm exhibits the characteristics of fast convergence speed, small computation amount, few adjustment parameters, and good convergence in low-dimensional problems [25]. However, the performance of this algorithm is poor when solving high-dimensional problems, and it easily falls into the local optimum. Therefore, the current study combines PSO with BAS and applies the resulting algorithm to the cloud service composition problem of multi-objective optimization. The model of the BAS algorithm is as follows.

(1) The left and right antennae of the beetle is set on the two sides of its center of mass.
(2) The proportion of the beetle’s step to the distance between two antennae (d0) is a fixed constant.
(3) When the beetle flies to the next step, the orientation of its head is random. The orientation of the beetle’s head is random in every step. That is, the manner in which the right antenna points to the left antenna is random and is defined as follows:

\[
\vec{b} = \frac{\text{rand}(k, 1)}{||\text{rand}(k, 1)||}
\]

where rand() is a random function, and k is the space dimension. The position of the two antennae is defined as follows:

\[
x_l = x' - d_l \vec{b}
\]
\[
x_r = x' + d_l \vec{b}
\]

where xl and xr are the positions of the left and right antennae, respectively; x’ is the position of the beetle’s center of mass at moment t; dl is the distance between two antennae at moment t. Then, the moving direction and distance of the beetle’s next step are confirmed as follows:

\[
x' = x^{t-1} + \delta \vec{b} \cdot \text{sign}(f(x_r) - f(x_l)) \tag{13}
\]

where \( f(\cdot) \) is a fitness function and \( \delta \) represents a search step at moment t. A large initial step is convenient for global searching. With the increasing number of iterations, the step will be shortened gradually. This condition is beneficial for a detailed search as follows:

\[
\delta = k \delta^{-1} \tag{14}
\]

4.2 Second-Order Beetle Swarm Optimization

The problems exist in the particle swarm algorithm is solved by the BAS algorithm in this study, the proposed algorithm is named second-order beetle swarm optimization algorithm (SBSO). The initial position and velocity of each particle are of the same process as the standard PSO algorithm. Besides, dynamic factors and a modified secondary oscillation link are employed to increase the particle population’s diversity and optimize the population’s food searching parameter. The proposed SBSO algorithm has the same computing process as the traditional PSO algorithm. First, an individual beetle is expanded to a population and its position update formula is as follows:

\[
v_b = -\delta \cdot \vec{b} \cdot \text{sign}(f(x_r) - f(x_l)) \tag{15}
\]

\[
v_i^{k+1} = \omega \cdot v_i^k + c_1 \cdot r_1 \cdot (P_{\text{best}} - x_i^k) + c_2 \cdot r_2 \cdot (G_{\text{best}} - x_i^k) + c_3 \cdot r_3 \cdot v_{b1} \tag{16}
\]

\[
x_i^{k+1} = x_i^k + v_i^{k+1} \tag{17}
\]

where \( v_i^{k+1} \) is the i-th beetle particle’s velocity after k times of iterations. \( x_i^{k+1} \) is the i-th beetle particle’s position after k times of iterations. \( v_b \) is the update rate of the beetle population. \( c \) is the learning factor. \text{sign}() is a sign function. \( r_1, r_2, \) and \( r_3 \) are the random numbers evaluated among [0, 1]. \( \omega \) is the inertia weight. \( P_{\text{best}} \) and \( G_{\text{best}} \) are the personal best of a particle and the global best of a population, respectively.

To solve the BSO algorithm’s problem of easily falling into the local optimum, convergence speed is accelerated and global searching speed is strengthened. A secondary oscillation link is introduced to modify the velocity update equation of the BSO algorithm. Then, the equation of the secondary oscillation link is modified, providing the BSO algorithm with faster convergence speed and stronger global searching ability.

When common secondary oscillation is added to the BSO algorithm, its velocity evolution formula is presented as follows:

\[
v_i^{k+1} = \omega \cdot v_i^k + \xi_1 (P_{\text{best}} - (1 + \beta_1)x_i^k - \beta_1 x_i^{k-1}) + \\
\xi_2 (G_{\text{best}} - (1 + \beta_2)x_i^k - \beta_2 x_i^{k-1}) \tag{18}
\]

\[
x_i^{k+1} = x_i^k + v_i^{k+1} \tag{19}
\]

where \( \xi_1 = c_1 r_1 \) and \( \xi_2 = c_2 r_2 \) when iteration times \( k < G_{\text{max}}/2 \), and \( \beta_1 < \xi_1, \beta_2 < \xi_2 \). The proposed algorithm exhibits a strong ability to search for the best solution. When iteration times \( k \geq G_{\text{max}}/2 \), and \( \beta_1 \geq \frac{2 \sqrt{\xi_1 - 1}}{\xi_1}, \beta_2 \geq \frac{2 \sqrt{\xi_2 - 1}}{\xi_2} \), the algorithm has rapid convergence speed. The value of \( G_{\text{max}} \) is the maximum iteration time.

To resolve the shortcomings of the BAS and PSO algorithms, the author of this study combined the two algorithms, introduced secondary oscillation, and modified the secondary oscillation. According to the literature [26], compared with the common secondary oscillation, the modified secondary oscillation changes the evaluations of the four \( \beta \) and defines them as four different parameters. Moreover, the relationships among the four parameters and their...
value range are determined. Thus, the algorithm achieves stronger searching ability, higher precision, faster convergence speed, and a more diverse population. The modified Eqs. (18) and (19) can be presented as Eqs. (20) and (21):

\[
v_{i}^{k+1} = \omega \cdot v_{i}^{k} + \xi_{1} \left( P_{\text{best}} - (1 + \beta_{1})x_{i}^{k} - \beta_{2}x_{i}^{k-1} \right) + \xi_{2} \left( G_{\text{best}} - (1 + \beta_{3})x_{i}^{k} + \beta_{4}v_{i}^{k-1} \right)
\]

\[
x_{i}^{k+1} = x_{i}^{k} + v_{i}^{k+1}
\]

When iteration times \( k < G_{\text{max}}/2 \), the algorithm exhibits a strong ability to search for the best solution. The value range and relationships of the algorithm’s oscillation convergence \( \beta_{1}, \beta_{2}, \beta_{3}, \beta_{4} \) are presented as the following formula:

\[
0 < \beta_{2} < \frac{1 + \beta_{1}}{2}, \quad 0 < \beta_{4} < \frac{1 + \beta_{3}}{2}, \quad 0 < \beta_{1} < 1,
\]

\[
0 < \beta_{3} < 1
\]

When iteration times \( k \geq G_{\text{max}}/2 \), the algorithm presents asymptotic convergence and its precision is increased. The value range and relationships of \( \beta_{1}, \beta_{2}, \beta_{3}, \beta_{4} \) are presented as the following formula:

\[
\beta_{1} < \beta_{2} - 1, \quad \beta_{3} < \beta_{4} - 1, \quad 0 < \beta_{2} < 1, \quad 0 < \beta_{4} < 1
\]

The learning factors that focus on a constant value will cause the velocity and position of the local particles to exhibit regular linear changes. Thus, the algorithm’s ability to search for the best solution becomes limited. Reference [4] presented nonlinear dynamic learning factors based on a trigonometric function to control the particles’ expanding capacity and global convergence capability. \( c_{1} \) and \( c_{2} \) are defined as the following equations:

\[
c_{1} = \rho_{1} + \cos^{2}(\rho_{1} \times \frac{t}{G_{\text{max}}})
\]

\[
c_{2} = \rho_{1} + \sin^{2}(\rho_{1} \times \frac{t}{G_{\text{max}}})
\]

where \( t \) is the current iteration times, \( \rho_{1} \) is a constant, and \( G_{\text{max}} \) is the maximum iteration times.

4.3 Adaptive Step-Size Factor

A step factor is key to controlling the convergence speed of an algorithm [27]. Step size is closely related to step factor \( \alpha \). The specific reasons are as follows.

(1) If the decay of step size is sufficiently slow, then the global search ability is strong, but convergence speed is too slow.

(2) If the decay of step size is too fast, then the global optimal solution may not be obtained.

The larger the step factor (tends to be 1), the slower the convergence speed and the stronger the global search capability. By contrast, the smaller the step factor (tends to be 0), the faster the convergence speed, but the proposed algorithm easily falls into the local extremum. However, the step factor of the basic search algorithm is fixed in the optimization process. To enable the algorithm to obtain better optimization ability, this study proposes an improved method of changing the step factor dynamically. In particular, during the early stage of optimization, a large step factor should be used to expand the overall search range in the solution space and accelerate the search speed. During the late stage of optimization, the search solution tends to be stable. To make the solution accuracy, the step factor should be reduced. Also, the smaller the initial step factor, the easier it will fall into the local extreme value. Thus, a high initial value, such as 0.95, should be selected. From the preceding considerations, the following adjustment mechanism is established:

\[
e = \alpha - 0.2 * ((i + 1)/G_{\text{max}} + 0.5), \quad f_{i} < f_{p\text{best}}
\]

\[
e = \alpha, \quad f_{i} \geq f_{p\text{best}}
\]

where \( f_{i} \) is the current fitness value, \( f_{p\text{best}} \) is the historical optimal fitness value, \( i \) is the current number of iterations, \( \alpha \) is the default step factor, and \( e \) is the current step. In the formula, when the current fitness value is greater than the historical optimal fitness value, the current performance is good. At this moment, the default value of the step factor should be maintained to ensure a global search. Otherwise, the optimization performance becomes poor, and the step factor should be reduced to accelerate convergence. Simultaneously, the maximum fitness value tends to be stable with an increase in iteration time, and the step factor should be reduced such that the scale is enlarged.

4.4 Dynamic Factor

The SBSO proposed in this study adopted modified versions of the aforementioned formulas for trigonometric dynamic controlling learning factors. Then \( c_{1} \) and \( c_{2} \) should be recomputed and two square calculations are necessary for each iteration process. Thus, the author of this study modified the aforementioned formulas for trigonometric dynamic controlling learning factors as follows:

\[
c_{1} = \alpha_{1} + \alpha_{2} \cos(t \times p_{i})/G_{\text{max}}
\]

\[
c_{2} = \alpha_{1} - \alpha_{2} \cos(t \times p_{i})/G_{\text{max}}
\]

Based on the original formulas, a weight factor \( \alpha \) is added to control the influence of trigonometric transformation on learning factors \( c_{1} \) and \( c_{2} \). This trigonometric transformation can reduce square calculations. Besides, the particles’ expanding capacity, global convergence capability, and global searching ability are strengthened.

4.5 Chaos Initialization

The chaotic sequence is added to the initial position and velocity of the BSO algorithm to disturb, the purpose is to prevent the BSO algorithm from converging prematurely and avoid entering the local optimal situation. The initial positions of \( m \) particles can be iterated through chaotic using formula (30), and then the chaotic sequence is mapped using formula (31):

\[
x(t + 1) = \mu x(t)(1 - x(t)), \quad t = 0, 1, 2, \cdots, n
\]
\[ x_i^t = \text{Round}(1 + c_t(w_i - 1)) \]

The detail of the SBSO algorithm is listed as follows:

Input: Maximum number of iterations \( G_{\text{max}} \), learning factors \( c_1 \) and \( c_2 \), population pop, initial step \( d_0 \)

Output: Individual optimal \( G_{\text{best}} \)

Step1: Initialization of the SBSO algorithm: Initialize the size, speed, and position of the beetles’ population by using chaos sequence.

Step2: Calculate the initial fitness value: Calculate the fitness value of each beetle, evaluate the optimal fitness value of the individual and the optimal fitness value of the group.

Step3: Calculate the optimal fitness value: Update the individual optimal value and the group optimal value through the BAS algorithm.

Step4: To ensure the best optimization results: Update the learning factor through formulas (28) and (29), update the velocity and position of the beetles by formulas (20) and (21).

Step5: Update the optimal fitness value: Update the current individual optimal fitness value and group optimal fitness value.

Step6: For precise optimum value: According to the current fitness value, the beetle step size can be adjusted by formulas (26) and (27) adaptively.

Step7: Iteration condition: If the maximum number of iterations is met, the SBSO algorithm ends; otherwise, go to step 2.

The SBSO algorithm flow chart is as follows:

Integer length coding is used in the problem of service composition optimization. The position of the \( i \)-th particle \( X_i = (X_{i1}, X_{i2}, \ldots, X_{in}) \) represents a service composition mode. \( n \) represents the number of subtasks and the dimension of particles. \( X_{ij} \) represents the selection of the candidate service with no. \( j \) for the \( i \)-th task. A real data set is imported to solve the fitness function of the proposed cloud service composition model using the developed algorithm.

## 5. Experiment and Simulation Result Analyses

### 5.1 Experimental Environment

The SBSO algorithm proposed in this study exhibits quicker convergence speed, stronger global searching ability, and less time cost when dealing with web service composition optimization problems in the cloud environment. The author of this study conducted a large number of experiments on a sequential structure using QWS real data sets [28]–[30], a random data set RDS, and WS-Dream data sets. QWS data sets are web service data sets compiled by Professor Eyhab Al-Masri of Guelph University. All the data sets in QWS are real data sets collected from various service websites. This data set contains 2507 real services and 9 QoS values; The RDS dataset is to make certain that the experimental results acquired by the proposed algorithms are not biased for the used real dataset. It is a wide dataset of randomly generated atomic web services denoted as RDS is considered. The random lower and upper values of QoS attribute values in each experiment were generated randomly in the ranges \([0, 1]\), respectively. In this dataset, it is easy to extend the QoS model by adding new QoS criteria without modifying the proposed algorithms for solving the service composition problems. Some data in QWS are shown in Table 2.

We added the WS-Dream datasets to verify the SBSO’s effectiveness in solving service composition problems. It is a real dataset collected by Zheng et al. [32], [33]. Currently, it has been applied in many studies [34], [35]. The WS-Dream dataset contains various datasets:

**Dataset 1:** The size of the web service QoS data set is 150*100. It monitors 100 Web services by using 150 distributed computer nodes located all over the world.

**Dataset 2:** The size of the web service QoS data set is 339*5825. It is Real-world QoS evaluation results from 339 users on 5,825 Web services.

**Dataset 3:** It contains 102*3568 Web service data. The value in the data set is standardized to verify the

| Table 2 | Part of the data set. |
|---------|-----------------------|
| Responses Time | Availability | Successability | Reliability | WSDL Address |
| 302.75 | 89 | 90 | 73 | http://xml.assessment.com/service/MAPPMatching.asmx?wsdl |
| 482 | 85 | 95 | 73 | http://www.msnaptopinter.op.org/asmx/WS/339/trasnportJ2/wsd1 |
| 3321.4 | 89 | 96 | 73 | http://www.strkio.tar/webservices/uidadda.asmx?wsdl |
| 126 | 98 | 100 | 67 | http://www.holidaywservice.com/Holidays/GBNR/Date/GBNIRHolidaysDate.asmx?WSDL |
| 107 | 80 | 81 | 67 | http://galax.tnsc.edu/casjobs/CasUers.aspx?WSDL |
| 255 | 98 | 99 | 67 | http://www.bhi.ebi.ac.uk/soaplab/emboss/services/llmhm.echmpfam.deriv.ed?wsdl |
| 136 | 76 | 76 | 60 | http://www.embelhi.ac.uk/tools/webservices/wsdl/WSO6feteh.wsd1 |
| 102 | 91 | 97 | 67 | http://trial.serviceobjects.com/p/PackTrack.asmx?wsdl |
The part of standardized data is shown in Table 3.

Table 3  Part of standardized data.

| Response Time | Availability | Success Ability | Reliability |
|---------------|--------------|-----------------|-------------|
| 0.0556        | 0.8817       | 0.8913          | 0.7143      |
| 0.0899        | 0.9456       | 0.9457          | 0.7143      |
| 0.6632        | 0.8817       | 0.9555          | 0.7143      |
| 0.0180        | 0.9785       | 1               | 0.6071      |
| 0.0141        | 0.8602       | 0.9457          | 0.7143      |
| 0.0440        | 0.9785       | 0.9981          | 0.6071      |
| 0.0201        | 0.7419       | 0.7391          | 0.4821      |
| 0.0132        | 0.9032       | 0.9674          | 0.6071      |

Table 4  SBSO setting of parameter values.

| Parameters     | Value |
|----------------|-------|
| $V_{\text{max}}$ | 3     |
| $V_{\text{min}}$ | 0.5   |
| $d_0$           | 0.8   |
| $\text{step}$   | 3     |
| $\omega$        | 0.7   |
| $\omega_{\text{max}}$ | 0.9   |
| $\omega_{\text{min}}$ | 0.4   |
| MaxIteration    | 100   |

The part of the processed data set is shown in Table 3. The part of the processed data set is shown in Table 3.

In this study, the proposed SBSO was compared with the IFPA [9], IGA [13], CGA [14], and BOTV-PSO [17].

The four QoS attributes, response time, availability, success ability, and reliability, were used in this study, and the above-mentioned attribute weights were set as 0.25, 0.25, 0.25, and 0.25, respectively. The algorithm accuracy, execution time, convergence analysis, and algorithm stability are compared in this section. Thus, the performance of the five algorithms in service composition is compared accurately. The experimental environment was as follows: Intel(R) Xeon(R) CPU E3-1220 V2 @ 3.10GHz 3.50GHz, 12 GB ddr3 1333MHz memory, 64 bit Windows 10 OS, and MATLAB R2015b.

$V_{\text{max}}$ and $V_{\text{min}}$ are the maximum and minimum values of velocity, $d_0$ is the distance between the two antennae, step is the initial step, $\omega_{\text{max}}$ and $\omega_{\text{min}}$ are the inertia weights, and $\omega$ is the second-order oscillation factor.

The value of $V_{\text{max}}$ and $V_{\text{min}}$ have an impact on the flying speed of beetle particles. The flying speed of beetle particles determines the search speed of the SBSO. The accuracy of the SBSO can be increased when the $V_{\text{max}}$ and $V_{\text{min}}$ is set to the optimal value, and the execution time of the SBSO can be reduced accordingly. The size of the initial step is related to the convergence speed of the proposed algorithm. To reduce the execution time and accelerate the convergence of SBSO, the initial step size can be set to a large value and vice versa. The secondary oscillation factor can affect the diversity of the SBSO population. The diversity of the population can improve the accuracy of the SBSO and avoid premature convergence. According to Ref. [26], the accuracy of the PSO can reach to maximum when the value $\omega$ is set to 0.7-1. Similarly, The inertia weight $\omega$ can affect the global and local optimization capabilities of the SBSO [31]. This study uses linearly decreasing inertia weights [18]:

$$
\omega = \omega_{\text{min}} + (\omega_{\text{max}} - \omega_{\text{min}})(T_{\text{max}} - t)/T_{\text{max}}
$$

(32)

Fig. 3  Influence of different scale of cloud service composition problems (n = 10)

where, $\omega_{\text{max}}$ is the initial inertia weight value, and $\omega_{\text{min}}$ is the inertia weight value corresponding to the maximum evolution algebra. Based on the literature [9], the inertia weight values are $\omega_{\text{max}} = 0.9$ and $\omega_{\text{min}} = 0.4$.

5.2 Execution Time

In terms of the execution time of the algorithm, the impact of the five algorithms on average execution time under different scale service requests was tested under the condition of the same number of particles and iterations. In this study, the problem size is set to $n \times m$, $n$ is the number of task requests, and $m$ is the number of candidate services. The five algorithms were compared when the iterations were 100 generations. To reduce the chance of the experiment, the mean value was computed after 50 times simulations. The experimental results are presented in Figs. 2–4.

As shown in Fig. 3, the scale of the solution space expands with the expansion of the cloud service composition problem. In the high-dimensional solution space, the execution time of the SBSO is the lowest in all of the comparison algorithms.

As shown in Fig. 4 and Fig. 5, When the problem
dimension is small, it is obvious that DPSO and BOTV-PSO perform better than the other two algorithms. The CGA performs well when the number of candidate services is small. As the number of candidate services increases, the growth of the service scale has less impact on the execution time of SBSO, BOTV-PSO, and IFPA. The execution time of the SBSO is shorter than that of BOTV-PSO and IFPA. Compared with the other two algorithms, the code execution efficiency of SBSO is higher. Therefore, SBSO is suitable for solving large-scale cloud service composition problems.

5.3 Algorithm Accuracy

This study compares the average optimal fitness values of the five algorithms and their accuracy when cloud service composition problems are solved from different angles. To prevent contingency, this study conducts 50 experiments to obtain optimal and average values.

In this study, the algorithm with a high average fitness value represents the algorithm with high accuracy and strong search capability. The accuracy of five algorithms is compared for different cloud service composition scales in Fig.6. The accuracy of IGA is worse than that of BOTV-PSO, CGA, IFPA, and SBSO. The SBSO has the highest accuracy, and the scale of the service has no perceptible effect on the accuracy of SBSO. Meanwhile, the SBSO is more stable than the other four algorithms when solving the problem of cloud service composition.

From the experimental data in Figs. 7 and 8, it can be concluded that SBSO has the highest algorithm accuracy among the five algorithms. Experimental data shows that CGA and IGA are easily trapped in local optimization. The global search capability of IFPA and BOTV-PSO are insufficient. So SBSO has the strongest global search capability among the five algorithms. The accuracy of the SBSO exceeds the other four algorithms when the dimension of the service composition problem increases. In this study, all data sets have been standardized. Therefore, the fitness value reflects the accuracy of the five algorithms and also represents the quality of service. Compared with the other four algorithms, SBSO is more suitable for solving the problem of cloud service composition optimization.

5.4 Algorithm Stability

Robustness indicates the fault tolerance of the algorithm.
The swarm intelligence algorithm has certain randomness. The gap of the solution is greater when the problem is getting more and more complicated. To verify the robustness of the SBSO for solving cloud service combinations of different sizes, the SBSO, IFPAA, IGA, CGA, and BOTV-PSO were run 100 times under the three service scales of $10 \times 10$, $20 \times 20$, and $50 \times 50$, and record the fitness value of each algorithm and calculate its standard deviation.

In this study, the standard deviation of the fitness value is used as the evaluation criterion of the algorithm robustness. At the same scale, the smaller the standard deviation of the algorithm fitness value, the stronger the algorithm is robust, and the more stable the algorithm performance.

As shown in Fig. 9, the standard deviations of fitness values of CGA, BOTV-PSO, and IFPA algorithms are nearly the same. However, the standard deviation of SBSO is lower than the other four algorithms. Because SBSO introduced a second-order oscillation factor in the BSO algorithm, the diversity of the population is guaranteed in the initial and iterative process of the algorithm. Meanwhile, the robustness of the algorithm is increased. Adaptive step size adjustment is adopted to avoid the algorithm falling into local optimization and the search efficiency of the algorithm is improved. Experimental results show that the SBSO is more stable and robust than the other four algorithms when solving sequential cloud service composition problems with different scales.

5.5 Convergence Analysis

The convergence of the five algorithms is compared when the number of tasks is set to 10, 15, and 20, respectively. To verify the convergence and divergence of the SBSO, the number of iterations is set to 100, and the RDS data set is selected for the convergence and divergence experiment. The number of candidate services is set to 200 in this experiment. This study obtained the average fitness value after 20 experiments.

In Figs. 10–12, the convergence speed of the five algorithms is decreasing when the dimension of the service composition problem becomes larger. But the accuracy of the SBSO is significantly higher than the other four algorithms. When the number of iterations is 50, the SBSO converges, and the average fitness value is the highest among the five algorithms. The experiment in this section proves that the accuracy and convergence of SBSO is superior to the other four algorithms. Thus, the SBSO is suitable for solving large-scale cloud service composition optimization.
problems.

5.6 Analysis of the Impact of Data Sets on Algorithms

In this section, the performance of the five algorithms are compared under the standardized WS-Dream datasets. The service scale is set as 10*10, 20*20, and 50*50, respectively. The average optimal fitness value is obtained by performing the experiments 20 times.

The average fitness value obtained by SBSO is the best under three different data sets in Fig. 13–15. Data set 1 mainly includes the user's web service interaction time and evaluation. We can see that in Fig. 13, the accuracy of SBSO under this data set is still excellent as the problem increases. Data set 2 contains a large amount of user information processing and server location. In Fig. 14, SBSO can still accurately obtain the optimal value. Data set 3 contains a large amount of service throughput information and various service evaluations. It can be seen from Fig. 15 that SBSO still maintains high accuracy even when the service scale increases. Therefore, the SBSO algorithm can solve the problem of cloud service composition in different environments.

6. Conclusion

The cloud service composition problem is a multi-objective optimization problem. First, this study proposes a QoS-based service composition model. Second, this study proposes an efficient second-order beetle antennae algorithm with global search capability, which improves the global convergence ability by expanding a swarm of beetles from a single beetle. And the improved algorithm uses chaos to initialize the population, increasing the search range of each particle. The improved second-order oscillation formula is added to the SBSO algorithm to update the speed and position of the beetle. Therefore, the convergence and divergence of SBSO have been improved. This study also proposes a new learning factor to control the diversity of the population by improving the trigonometric function formula. Besides, a dynamic step factor is proposed to enhance the stability of SBSO. Finally, this study verifies the SBSO’s performance by comparing the other four algorithms under various data sets. With the expansion of service composition problem dimensions, the SBSO’s execution efficiency and accuracy are significantly better than that of the other four algorithms. The convergence and divergence of SBSO are superior to that of the other four algorithms by using the RDS data set. Therefore, the SBSO is more suitable for solving large-scale cloud service composition optimization problems.

Acknowledgments

This research was funded by the Project of education department of Jilin province, grant number JJKH20200786KJ. The authors express their sincere appreciation to the editors and the anonymous reviewers for their helpful comments.

References

[1] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, “QoS-aware Middleware for Web Services Composition,” IEEE Trans. Softw. Eng., vol.30, no.5, pp.311–327, 2004.
[2] W. Ni, L.-C. Liu, and W. Cheng, “Survey on web services composition methods,” Computer Engineering, vol.34, no.4, pp.79–81, 2008.

[3] S.C. Sadouki and A. Tari, “Multi-objective and discrete elephants herding optimization algorithm for QoS aware web service composition,” RAIRO-Oper. Res., vol.53, no.2, pp.445–459, 2019.

[4] Y. Huo, Y. Zhuang, J. Gu, S. Ni, and Y. Xue, “Discrete guided artificial bee colony algorithm for cloud service composition,” Applied Intelligence, vol.42, no.4, pp.661–678, 2015.

[5] Z.-Z. Liu, D.-H. Chu, C. Song, X. Xue, and B.-Y. Lu, “Social learning optimization (SLO) algorithm paradigm and its application in QoS-aware cloud service composition,” Information Sciences, vol.326, pp.315–333, 2016.

[6] W. Zhang, C.K. Chang, T.M. Feng, and H. Jiang, “QoS-Based Dynamic Web Service Composition with Ant Colony Optimization,” Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual. IEEE, pp.493–502, 2010.

[7] P. Rodriguez-Mier, M. Mucientes, and M. Lama, “Hybrid Optimization Algorithm for Large-Scale QoS-Aware Service Composition,” IEEE Transactions on Services Computing, vol.10, no.4, pp.547–559, 2015.

[8] E. Maythaisong and W. Songpan, “Mutation-Based Harmony Search Algorithm for Hybrid Testing of Web Service Composition,” Computational Intelligence and Neuroscience, vol.2018, pp.1–15, 2018. DOI:10.1155/2018/4759405.

[9] W. Tan and J. Wu, “Optimization of web service composition based on improved flower pollination algorithm,” Computer Engineering, vol.46, no.12, pp.67–72, 2020.

[10] T. Wen, G.-J. Sheng, Q. Guo, and Y.-Q. Li, “Web Service Composition Based on Modified Particle Swarm Optimization,” CHINESE JOURNAL OF COMPUTERS, vol.36, no.5, pp.1031–1046, 2013.

[11] M.H. Shirvani, “Web Service Composition in multi-cloud environment: A bi-objective genetic optimization algorithm,” 2018 Innovations in Intelligent Systems and Applications (INISTA), 2018. DOI:10.1109/Inista.2018.8466267.

[12] F. Seghir and A. Khababa, “A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition,” Journal of Intelligent Manufacturing, (6): 1-20, 2018.

[13] M. Du, “An improved genetic algorithm for web service composition optimization,” World Scientific Research Journal, vol.6, no.10, 2020.

[14] Y. Li, J. Hu, Z. Wu, C. Liu, F. Peng, and Y. Zhang, “Research on QoS service composition based on coevolutionary genetic algorithm,” Soft Computing, vol.22, pp.7865–7874, 2018. DOI:10.1007/s00500-018-3510-5.

[15] A. Naseri and N.J. Navimipour, “A new agent-based method for QoS-aware cloud service composition using particle swarm optimization algorithm,” Journal of Ambient Intelligence and Humanized Computing, vol.10, pp.1851–1864, 2018. DOI:10.1007/s12652-018-0773-8

[16] H. Fekih, S. Mibaia, and S. Bouamana, “An Efficient User-Centric Web Service Composition Based on Harmony Particle Swarm Optimization,” International Journal of Web Services Research., vol.16, no.1, pp.1–21, 2019.

[17] H.S. Mirzaei, “Bi-objective web service composition problem in multi-cloud environment: a bi-objective time-varying particle swarm optimisation algorithm,” Journal of Experimental & Theoretical Artificial Intelligence, vol.33, no.2, 2021.

[18] R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” Proc. Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp.39–43, 1995.

[19] X. Jiang and S. Li, “BAS: Beetle Antennae Search Algorithm for Optimization Problems,” International Journal of Robotics and Control, vol.1, no.1, 2017. DOI:10.5430/ijrc.v1n1p1

[20] X. Jiang and S. Li, “Beetle antennae search without parameter tuning (BAS-WPT) for multi-objective optimization,” arXiv preprint arXiv:1710.10724, 2017.

[21] Y. Fan, J. Shao, and G. Sun, “Optimized PID Controller Based on Beetle Antennae Search Algorithm for Electro-Hydraulic Position Servo Control System,” Sensors, vol.19, no.12, p.2727, 2019.

[22] Z.Y. Zhu, Z.Y. Zhang, W.S. Man, X. Tong, J. Qiu, and F. Li, “A new beetle antennae search algorithm for multi-objective energy management in microgrid,” 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), Wuhan, China, pp.1599–1603, 2018.

[23] Y. Sun, J. Zhang, G. Li, Y. Wang, J. Sun, and C. Jiang, “Optimized neural network using beetle antennae search for predicting the unconfined compressive strength of jet grouting coalslakes,” International Journal for Numerical and Analytical Methods in Geomechanics, vol.43, no.4, pp.801–813, 2019.

[24] J. Yang and Z. Peng, “Beetle-swarm Evolution Competitive Algorithm for Bridge Sensor Optimal Placement in SHM,” IEEE Sensors J., vol.20, no.15, pp.1558–1748 (c), 2019. DOI: 10.1109/JSEN.2019.2934996.

[25] Y. Zhang, S. Li, and B. Xu, “Convergence analysis of beetle antennae search algorithm and its applications,” arXiv:1904.02397 [cs.NE], 2019.

[26] L. Jiang, R. Ye, C. Liang, and W. Lu, “Improved second-order oscillatory particle swarm optimization,” Computer Engineering and Applications, vol.55, no.9, pp.130–138, 2019.

[27] J. Wang and H. Chen, “BSAS: Beetle swarm antennae search algorithm for optimization problems,” International Journal of Robotics and Control, vol.1, no.1, p.1, 2018.

[28] E. Al-Mari and Q. Mahmoud, “The qws dataset,” 2014. http://www. uoguelph.ca/qm/mahmoud/ qwsindex.html.

[29] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q.Z. Sheng, “Quality driven web services composition,” Proc. 12th International Conference on World Wide Web, pp.411–421, 2003.

[30] L. Zeng, B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, “Qos-aware middleware for web services composition,” IEEE Trans. Softw. Eng., vol.30, no.5, pp.311–327, 2004.

[31] Y. Shi and R. Eberhart, “A modified particle swarm optimizer,” Proc. IEEE International Conference on Evolutionary Computation, Anchorage, AK, pp.69–73, 1998.

[32] Z. Zheng, Y. Zhang, and M.R. Lyu, “Distributed QoS evaluation for real-world web services,” 2010 IEEE International Conference on Web Services, IEEE, pp.83–90, 2010.

[33] Z. Zheng, Y. Zhang, and M.R. Lyu, “Investigating QoS of real-world web services,” IEEE Trans Serv Compu., vol.7, no.1, pp.32–39, 2014.

[34] X. Jian, Q. Zhu, and X. Xia, “An interval-based fuzzy ranking approach for QoS uncertainty-aware service composition,” Optik Int J Light Electron Optics, vol.127, no.4, pp.2102–2110, 2016.

[35] Y. Chen, L. Jiang, J. Zhang, and X. Dong, “A robust service selection method based on uncertain QoS,” Math Prob Eng, vol.2016, 9480769, 2016. https://doi.org/10.1155/2016/9480769.

Hongwei Yang received the master's degree from Changchun University of Science and Technology. He is currently an associate professor at the School of Computer Science and Technology, Changchun University of Science and Technology. His interest area is the internet of things and data prediction.
**Fucheng Xue**  born in February 1995, master's degree candidate of Changchun University of Science and Technology, majoring in distributed system and middleware technology.

**Dan Liu**  received the doctor's degree from Changchun University of Science and Technology. She is currently an associate professor at the School of Computer Science and Technology, Changchun University of Science and Technology, Changchun, China.

**Li Li**  received the master's degree and the doctor's degree from Changchun University of Science and Technology, China. She is currently a professor at the School of Computer Science and Technology, Changchun University of Science and Technology, Changchun, China.

**Jiahui Feng**  born in June 1999, a bachelor degree candidate at Changchun University of Science and Technology. Mainly engaged in cloud computing, data analysis, and mining research.