Improved Hunting Search Algorithm for Web Service Composition Optimization

Q J Wang 1,a, Q L Zhang1,b, J T Li2,c, X Y Song1,d* and Z Y Liu1,e

1 Department of Computer Science and Technology, Dalian Neusoft University of Information, Dalian, Liaoning, China;
2 Software Technology Institute, Dalian Jiaotong University, Dalian, Liaoning, China;
aemail: wangqingjun@neusoft.edu.cn, bemail: zhangqilong@neusoft.edu.cn,
cemail: 285477831@qq.com, demail: songxiaoying@neusoft.edu.cn,
eemail: liuzhenyu@neusoft.edu.cn,
*Corresponding author: demail: songxiaoying@neusoft.edu.cn

Abstract. With the rapid development and application of Web services, the problem of Web service composition optimization based on Quality of Service (QoS) has become a research hotspot in the field of service computing. There are many researches on hunting search algorithm in continuous optimization problems, however, there are few researches using it to solve Web service composition optimization. In this paper, we propose an improvement on the hunting search algorithm (IHuS). It introduces an elite guidance strategy to accelerate the convergence of the algorithm, and at the same time it learns from the mutation strategy in the GA algorithm to improve the diversity of combined services. Compared with the standard HuS and GA algorithms, the experimental results demonstrate that IHuS has faster convergence speed and better optimization performance in solving large-scale Web service composition problems.

1. Introduction
With the rapid development of high-tech technologies such as big data and cloud computing, a large number of candidate web services with the same functions but different QoS have appeared on the network. Service selection and composition among the set of candidate Web services will form an "explosion" of the composition scheme, which makes Web service composition selection an NP-hard problem [1]. In a large-scale and complex network environment, it has become one of the research hotspots that how to efficiently select a service composition with meeting the user’s QoS requirements and with the global optimal service quality from a large number of composition solutions in the service computing field. In recent years, domestic and foreign scholars have proposed a variety of solutions to the problem of Web service composition based on service quality. It can be roughly divided into three categories: exhaustive methods, artificial intelligence methods and intelligent optimization algorithms. The exhaustive method is to enumerate all possible solutions and select the best one. The time complexity of the algorithm is large, it is not recommended to use in the case of large-scale service composition. Artificial intelligence methods have been applied in the field of Web service composition optimization by some scholars.
In [2], it used a deep reinforcement learning to solve the problem of low efficiency of service composition optimization. However, the scalability and applicability of Web combination services did not be taken account in complex network environments. In the selection of Web service composition, intelligent optimization algorithms are the most widely used. In [3], the algorithm used discrete particle swarm algorithm to solve the problem of dynamic Web service composition. It introduces Skyline technology to eliminate redundant candidate services, and enhances the global search ability of the algorithm by improving the diversity of particle swarms. However, the PSO algorithm is generally easy to fall into the local optimal problem. In [4], it applied the improved artificial bee colony (ABC) algorithm to Web service composition optimization in a large-scale service environment. In [5], it proposed an improved ant colony optimization algorithm with dynamic pheromone update, and applied it in the field of service composition optimization. The algorithm showed a good optimization effect. In [6], it applied the improved flower pollination algorithm to Web service composition optimization in a large-scale environment. However, it did not consider the throughput of Web service composition in large-scale environments.

In this paper, an improved hunting search algorithm is proposed to solve the optimization problem of large-scale Web service composition. An elite guidance strategy and composition mutation operation are added to the hunting search algorithm. Hence, it accelerates convergence speed and enhances the algorithm optimization ability to improve the performance of Web service composition.

2. Web Service Composition

2.1 Web Service Composition problem
Assuming that a Web service composition process includes n subtask nodes, the sequence of subtask nodes is \(T=\{T_1, T_2, \ldots, T_n\}\). Each subtask node corresponds to a candidate service set \(S_i = \{ws_{i1}, ws_{i2}, \ldots, ws_{im}\}\). \(S_i\) is a set of services with the same function but different service quality and QoS. \(ws_i\) is a specific service entity, and \(m\) is the number of candidate services in \(T_i\).

The optimization problem of Web service composition can be described as: A specific service entity is selected from the candidate service set of each subtask, and the service composition formed by the selected service entities is required to meet the customer's QoS requirements and have a globally optimal or near-optimal service quality. In Web service composition, there are generally four composition models: sequential model, parallel model, selection model and cycle model. Since parallel, selection, and loop models can all be transformed into sequential models [7], we only discuss sequential models in this paper.

2.2 QoS-based service composition problem model
In this paper, we consider four QoS parameters of Web services, namely: response time (rt), availability (a), throughput (t) and reliability (rel). The QoS attribute set of a single service can be expressed as \(Q(ws_i) = \{q_{rt}, q_{av}, q_{t}, q_{rel}\}\).

The QoS attribute set of the composite service is marked as \(Q(cs) = \{Q_{rt}, Q_{av}, Q_{t}, Q_{rel}\}\), where, Each element is aggregated by the QoS attribute value of the service entity. We refer to [8], the calculation method of the QoS parameters of the service composition sequence model is as follows:
Where, \( n \) is the number of subtasks, \( 1 \leq j \leq m \), and \( m \) is the number of services in Candidate service set. In practical applications, the measurement units and magnitudes of different QoS attributes are not the same. Therefore, it is necessary to normalize each attribute before calculating the user’s QoS satisfaction with the combined service. QoS attributes can be divided into positive attributes and negative attributes.

For positive attributes such as throughput, availability, and reliability, the larger numbers are, the better is. The normalization formula is as follows:

\[
UQ_r = \begin{cases} 
\frac{Q_r - \min Q_r}{\max Q_r - \min Q_r}, & \text{max } Q_r \neq \min Q_r \\
1, & \text{max } Q_r = \min Q_r
\end{cases}
\]

(2)

For negative properties such as response time, the smaller the value is, the better is. The normalization formula is as follows:

\[
UQ_r = \begin{cases} 
\frac{\max Q_r - Q_r}{\max Q_r - \min Q_r}, & \text{max } Q_r \neq \min Q_r \\
1, & \text{max } Q_r = \min Q_r
\end{cases}
\]

(3)

\( Q_r \) is the aggregate value of the attribute of the service composition, and \( \min Q_r \) is the minimum aggregate value of the attribute of all Web service compositions.

Define the user’s QoS satisfaction calculation formula for the service composition is defined as follows:

\[
QSat(cs) = \sum_{r=1}^{n} w_r \ast UQ_r
\]

(4)

Where, \( w_r \) indicates the user’s attribute preference for the \( r \)th QoS of the service and meets the condition \( \sum_{r=1}^{n} w_r = 1 \) and \( 0 < w_r < 1 \). It can be seen that the greater the user QoS satisfaction, the better the combined service plan.

3. **Hunting search algorithm**

Hunting Search (HuS) algorithm was proposed by Ofadeh et al. in 2010. It is a heuristic optimization algorithm that simulates predation and hunting behavior of group animals (lions, wolves, dolphins, etc.) [9]. The hunter surrounds the prey and gradually reduces the encircling circle until the prey is caught. During the hunting process, each hunter will adjust his position according to the position of the prey and the positions of other members. The leader in the group is the hunter who has the best position at the current stage (the current population optimal solution). In the process of the hunter's position change, the leader's identity also changes, and he is always the individual occupying the best position. If the prey escapes the enclosure, the hunters will reorganize and surround the prey again.

The basic hunting search algorithm consists of the following 5 steps:
Step 1: Initialize the optimization problem and algorithm parameters. They include Hunting Group Size (HGS), Maximum Movement toward Leader (MML), Hunting Group Consideration Rate (HGCR), EN, Ra, Reorganization parameters $\alpha$ and $\beta$, and algorithm maximum number of iterations.

Step 2: Initialize the hunting group (HG). The hunting group matrix is filled with feasible randomly generated solution vectors. The leader is defined based on the values of objective functions of the hunters.

Step 3: Moving toward the leader. The new hunters’ positions (new solution vectors) $x' = (x'_1, x'_2, \ldots, x'_n)$ are generated by moving toward the leader (the hunter that has the best position in the group) as follows:

$$x'_i = x_i + \text{rand} \times MML \times (x_i - x^*)$$

where, $\text{MML}$ is the maximum movement toward the leader, $\text{rand}$ is a uniform random number which varies between 0 and 1, and $x_i^*$ is the position value of the leader for the $i$th variable. If its previous position is better than its new position, it comes back to the previous position.

Step 4: Position correction-cooperation between members. After moving toward the leader, hunters (based on other hunter positions and some random factors) choose another position to find better solutions. The formula of position correction is as follows:

$$x'_i \left\{ \begin{array}{ll}
  x'_i & \in \{x_1^i, x_2^i, \ldots, x_{\text{HGS}}^i\} \text{ with probability HGCR} \\
  x'_i = x_i \pm Ra & \text{with probability(1-HGCR)}
\end{array} \right.$$  

where, the parameter HGCR is the probability of choosing one value from the hunting group stored in the HG, 1-HGCR is the probability of doing a position correction, and $Ra$ is the moving distance radius.

Step 5: Reorganizing the hunting group.

The sequence of searches that end with trapping the group in a local minimum or the certain number of searches is called one epoch. The leader keeps its position and the other hunters randomly choose their positions as follows:

$$x'_i = x_i^* \pm \text{rand} \times (\max(x_i) - \min(x_i)) \times \exp(-\beta \times EN)$$

where, $x_i^*$ is the position value of the leader for the $i$th variable. $\text{rand}$ is a uniform random number which varies [0, 1], $\max(x_i)$ and $\min(x_i)$ are the maximum and minimum possible values of variable $x_i$, $\text{EN}$ counts the number of times that the group has been trapped until this step. $\alpha, \beta$ are reorganizing parameters.

4. Web service composition optimization based on improved hunting search algorithm

The standard hunting search algorithm is improved in this paper, and introduces an elite guidance strategy to speed up the algorithm’s global convergence and improve the efficiency of optimization. At the same time, in order to ensure the diversity of the population, the mutation operation in the GA algorithm is introduced. Improved Hunting Search (IHuS) is used to solve the problem of large-scale discrete Web service composition optimization. The algorithm is described as follows:

4.1 Coding method, initial solution and fitness function

Each hunter is regarded as a web service composition solution. Discrete web services use integer encoding. The solution vector corresponding to the service composition is $x_i = (x_{i1}, x_{i2}, \ldots, x_{ik}, L_x, x_{in})$, where, $n$ is the number of tasks. $x_{ik} \in [1, 2, L, m_k]$ denotes Service entity number selected in the $k$th subtask candidate service set of the $i$th service composition. $m_k$ denotes the total number of services in the subtask candidate service set. The problem of Web service composition selection is transformed into an optimal solution problem of $n$-element vectors.

According to the coding rules of the solution, a service entity number is randomly selected in each subtask service set as a component of the solution vector, and an initial solution is finally
generated. Define the fitness function of the solution as follows:

\[ FT(x_i) = Q_{sat}(cs) \]  (8)

It can be seen that the greater the user QoS satisfaction is, the greater the adaptation value is, and the better the solution is.

4.2 Moving toward the leader

Assuming the current population leader is \( x_{\text{best}} = (x_{b,1}, x_{b,2}, \ldots, x_{b,k}, \ldots, x_{b,n}) \), then the calculation formula for the movement of each hunter \( x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,k}, \ldots, x_{i,n}) \) to the leader is as follows:

\[
x_{i,k} = x_{i,k} + \left[ \text{rand} \times \text{MML} \times (x_{b,k} - x_{i,k}) \right]
\]  (9)

where, \( \text{rand} \) is a random number between \([0,1]\), and \( \text{MML} \) is the moving step length. If the fitness value of the new solution is better than the original solution, the hunter moves to the new position, otherwise, returns to the original position.

4.3 Position correction-cooperation between members

The hunter will choose another location based on the location of other hunters in the hunter group and a random location in order to find a better solution. The new position calculation formula for hunting individual \( i \) is as follows:

\[
x_{i,k} \left\{ \begin{array}{ll}
x_{i,k} \in \{x_{1,k}, x_{2,k}, \ldots, x_{\text{HGS},k}\}, & p < \text{HGCR} \\
x_{i,k} \pm R_a, & \text{otherwise}
\end{array} \right.
\]  (10)

where, \( p \) is a random number between \([0,1]\). \( R_a \) is the radius of distance, and it is a random value.

4.4 Elite selection and guidance

In order to improve the convergence performance of the algorithm, an elite guidance strategy is introduced. The elite solution is selected from the current HG to speed up the algorithm's global convergence performance and to improve the efficiency of algorithm optimization.

Elite hunter selection adopts tournament strategy. \( h \) individuals \( (0 < h < \text{HGS}) \) are randomly selected in the hunting group, among which the one with the largest fitness value is the elite individual. Components are randomly select from the vectors of the elite individuals, and the components corresponding to the individuals are replaced to guide the optimization, as shown below:

\[
\begin{array}{ccccccc}
\text{individual} & x_{i,1} & x_{i,2} & \ldots & x_{i,k} & \ldots & x_{i,n} \\
\text{elite individual} & x_{e,1} & x_{e,2} & \ldots & x_{e,k} & \ldots & x_{e,n}
\end{array}
\]

Figure 1. Elite guidance

4.5 Mutation operation

We draws on the mutation operation in genetic algorithm to ensure the diversity of the population and avoid the algorithm from falling into the local optimum. The operation is shown as perturbing each component in the individual \( x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,k}, \ldots, x_{i,n}) \), as shown in formula (11).

\[
x_{i,k}^{\text{new}} = \begin{cases} x_{i,k} + \delta, & \text{if } q < \text{MR} \\ x_{i,k}, & \text{otherwise} \end{cases}
\]  (11)

where, \( q \) is a uniform random number between \([0,1]\). \( \text{MP} \) is mutation probability. \( \delta \) is the disturbance factor, which is randomly selected as 1 or -1 with the same probability.
4.6 The process of the algorithm

Step1: Initialize algorithm parameters. Initialize HGS, MML, HGCR, MR, h, c and the maximum number of iterations $G$.

Step2: According to the initial solution generation rule in Section 4.1, hunting group is initialized and the fitness value is calculated, and the individual with the largest fitness value is selected as the leader.

Step3: Moving toward the leader. According to the rules in Section 4.2, each hunter in the hunting group moves to the leader.

Step4: According to the rules in Section 4.2, each hunter in the hunting group moves to the leader. According to (10), the position of the individual is corrected.

Step5: Elite guidance. Select elites and guide individuals according to the rules in section 4.4.

Step6: Mutation operation. According to (11), the mutation operation is performed on the individual. The individual fitness value is recalculated, and if the fitness value is greater than the leader's fitness value, it will be regarded as the leader.

Step7: Judge the end condition, and output the optimal solution if it is satisfied; otherwise, go to step 3.

5. Experimental simulation and result analysis

5.1 Experimental environment and data

In order to verify the effectiveness of the MHuS algorithm in solving the Web service composition optimization problem, we compares the standard hunting search algorithm HuS and the genetic algorithm GA from the three aspects of feasibility, $c$ and response time. The experimental program is written in Python, and the interpreter version is 3.5. The experimental environment is Windows 10 (64 bit), Intel(R) Core(TM) i5-9300H CPU @2.40GHz, 8G RAM. The experiment uses the QWS data set provided by Zeng et al. [10], which contains 2507 Web service information in the real Internet, and each service contains 9 QoS attribute values. We selects four QoS attribute values: response time, availability, throughput, and reliability. In order to simulate the optimization process of Web service composition under real conditions, the experiment set 6 different subtask numbers [5, 10, 15, 20, 25, 30], where, the number of candidate services available for each subtask is 50. Taking into account the randomness of the intelligent optimization algorithm, each group of experiments was run 20 times repeatedly, and the results were averaged.

5.2 Experimental parameter settings

In the experiment, the group size of the three optimization algorithms is set to 50, The maximum number of iterations is set to 300, and the user’s preference for QoS attributes is set to $w_r = w_q = 0.3$, $w_t = w_{re} = 0.2$. Algorithm parameter settings are shown in Table 1.

| Parameter Description                  | Value  |
|----------------------------------------|--------|
| Group size (HGS)                       | 50     |
| Hunting group consideration probability(HGCR) | 0.7    |
| maximum movement toward the leader (MML) | 1      |
| The radius of distance (Ra)            | [-3,3] |
| Number of leading components (c)       | 2      |
| Individual selections in the tournament (h) | 5      |
5.3 Feasibility

In order to verify the feasibility of the IHuS algorithm in the field of Web service composition optimization, the ability of the three optimization algorithms to find the optimal solution under different numbers of subtasks is investigated. The experimental results are shown in Table 2.

Table 2. Compared with fitness value

| Number of subtasks | IHuS | HuS | GA |
|--------------------|------|-----|----|
| 5                  | 0.828| 0.820| 0.751|
| 10                 | 0.839| 0.821| 0.762|
| 15                 | 0.841| 0.822| 0.781|
| 20                 | 0.840| 0.819| 0.769|
| 25                 | 0.829| 0.817| 0.762|
| 30                 | 0.834| 0.812| 0.764|

From Table 2, it can be obtained that the fitness value of the service composition found by IHuS is higher than the other two algorithms under different numbers of subtasks. The IHuS algorithm is not only feasible, but also has better optimization results.

5.4 Convergence

This group of experiments mainly analyzes the convergence of the application of the IHuS algorithm to the optimization problem of Web service composition. We conduct experiments under the number of subtasks 15, 20, and 25 respectively, and mainly examine the influence of the number of iterations on service portfolio selection. The experimental results are shown in Figure 2.

From Figure 2, it can be obtained that as the number of service composition subtasks increases, the convergence performance of the three algorithms decreases. As IHuS introduces elite guidance strategies and mutation operations, the algorithm basically converges after about 100 iterations. At the same time, the fitness value of the IHuS algorithm is higher than the other two optimization algorithms. Since the GA algorithm does not introduce global information operations, it has the worst convergence.

5.5 Response time

Under large-scale Web service composition, whether the system can respond to user requests or not in real time is one of the important considerations for service composition optimization. This group of experiments compares the response time of the three optimization algorithms under different numbers of subtasks. The experimental results is shown in Figure 3.
From Figure 3, it can be obtained that the response time of the optimal service composition of each algorithm will also increase as the number of service composition subtasks increases. However, the response time of IHuS's optimal composition is the least. With no more than 30 subtasks, the response time of the combined solution for IHuS can be controlled within 500ms, which can meet user requirements in real time.

6. Conclusion

In this paper, we propose a Web service composition method based on improved hunting search algorithm to increase the rate of Web service composition. IHuS introduces elite guidance strategies and mutation operations on the basis of the standard hunting search algorithm, and improves the efficiency of optimization on the premise of ensuring the diversity of compositions. The experimental results verify the feasibility and effectiveness of the IHUS algorithm in this paper. The convergence speed and solution quality of the algorithm are significantly improved compared with HuS and GA algorithms. Malicious service fraud may exist in a complex network environment, hence, the credibility of the web service composition will be the focus of research work in the future.

Acknowledgments

This research is supported by the National Nature Science Foundation of China under Grant NO.61772101, and in part by the Ph.D. Scientific Research Starting Foundation of Liaoning Province, and in part by Innovative Talents in Universities of Liaoning Province under Grant NO.LR2019005 and ZX2020KJ031, and in part by Dalian Association for Science and Technology under Grant NO. ZX2018KJ038.

References:

[1] T Wen, G J Sheng, Q Guo, et al. Web service composition based on modified particle swarm optimization [J]. Chinese Journal of Computers, 2013, 36(5):1031-46.

[2] A Moustafa, T Ito. A deep reinforcement learning approach for large-scale service composition [C], International Conference on Principles and Practice of Multi-Agent Systems, 2018:296-311.

[3] Y P Zhang, Z H Jing Y W Zhang, et al. Dynamic web service composition based on discrete particle swarm optimization [J]. Computer Science, 2015, 42(6):71-75.

[4] M Chandra, R Niyogi. Web service selection using modified artificial bee colony algorithm [J]. IEEE Access, 2019, 99:1-10.
[5] Y Xiao, B Chen, J Chen, et al. Optimizing services composition based on improved ant colony algorithm [J]. Chinese Journal of Computers, 2012, 35(2):270-281.

[6] W A Tan, J k Wu. Optimization of web service composition based on improved flower pollination algorithm [J]. Computer Engineering, 2020, 46(12):67-72.

[7] J Cardoso, A Sheth, J Miller, et al. Quality of service for workflows and web service processes [J]. Journal of Web Semantics, 2004, 1(3):281-308.

[8] H Song, Y L Wang, G Q Liu and B Zhang. Web service composition optimization method based on improved multi-objective artificial bee colony algorithm [J], Journal of Northeastern University (Natural Science), 2019, 40(6):777-782.

[9] R Oftadeh, M J Mahjoob, M Shariatpanahi. A novel meta-heuristic optimization algorithm inspired by group hunting of animals: hunting search [J]. Computers & Mathematics with Applications, 2010, 60(7):2087-2098.

[10] L Zeng, B Benatallah, M Dumas, et al. Quality driven web services composition [C], Proceedings of the 12th International Conference on World Wide Web, 2003:411-421.