Journal of Database Management
Volume 34 • Issue 1

An Enhanced Energy-Efficient Web Service Composition Algorithm Based on the Firefly Algorithm

Yifei Xue, Northeast Forestry University, China
Jian Wang, Northeast Forestry University, China*
Weipeng Jing, Northeast Forestry University, China

ABSTRACT

Numerous web services with the same function but different service qualities are constantly emerging on the network. Optimizing web service composition based on multiple candidate services sets an urgent problem in the service composition neighborhood. This paper modifies the traditional Firefly algorithm and adds exchange and mutation mechanisms to optimize the Web service composition efficiently in multiple candidate service sets. Meanwhile, it discretizes the continuous space of its solution set and better adapts to the service composition optimization problem. Experimental results show that compared with the GA, IA, SA, ACO, FACO, and EFACO algorithms, this algorithm has better optimization performance, faster speed, and higher energy efficiency for solving service composition optimization problems in the case of large-scale data. The higher the combined complexity of the solution, the stronger the performance compared to other algorithms. It can better deal with the increasingly complex situation of Web service composition problems.

KEYWORDS

composition optimization, energy efficient, firefly algorithm, Web service, web services composition

INTRODUCTION

With the development of the information age, data is generated faster and faster, forming big data, which has brought a revolutionary product to many industries. Compared with the previous relational database management system, big data processing and storage are complicated, and the system's energy consumption increases a lot, which burdens the system (Meyer & Weske, 2006; Yaghoubi et al., 2020). As the infrastructure layer of the extensive data system, the cloud computing platform can meet this challenge to a certain extent. Cloud services are quantifiable IT services provided externally under the cloud computing architecture (Meyer & Weske, 2006; Yaghoubi et al., 2020). Cloud services are more convenient, low deployment cost, and highly open compared to traditional

DOI: 10.4018/JDM.321740 *Corresponding Author

services (Ramalingam & Mohan, 2021). Web services are one of the cloud services. Web service is a kind of Web application, which is self-contained, self-describing, and modular, and can be published, searched, and invoked through the Web (Tsalgatidou & Pilioura, 2002). Using Web services can realize cross-platform interaction or integration of heterogeneous applications on the Internet, which provides excellent convenience for existing service providers and users (Al-Masri & Mahmoud, 2008). More and more enterprises and users are now beginning to deploy their software, systems, computing resources as Web services.

Web services adopt a service-oriented architecture(SOA), which implements service invocation through the interaction between service providers, requesters, and registry entities (Atkinson et al., 2002; Li et al., 2007; Marcelo Fantinato et al., 2021). The service provider (server) provides services that satisfy the requirements based on the initial query of the service requester (client or user) in the "service discovery" phase. These services are published in the registry; then, the service requester is registered. The center finds and binds the "best" Web service that fits its conditions (Haddad et al., 2010). The service providers consider the user's needs and select the "best" Web service. More importantly, consider the quality of service (QoS) of Web services.

In the natural environment, many Web services can provide the same function. The difference is that their quality of service is uneven, good and bad. For example, when booking a flight or downloading music, many service providers can support similar services. However, their quality is different, and users will choose a service with fast response speed, high reliability, and low energy consumption. High-quality Web service will contribute to solid competitiveness for service providers.

Nowadays, the demands of users are becoming more and more complex. Web services with a single function can no longer fit the requirements of users. Service providers can combine multiple Web services to achieve business purposes by the reuse of services, and it does reduce not only business costs but also meets customer needs. However, Web service composition is a challenging problem. Among the massive amounts of data, the task of retrieving Web services is enormous, and each service has QoS indicators as restrictions. There may be conflicts between QoS indicators. Choosing the best combination of web services requires considering the QoS attributes of each type of web service (Dahan et al., 2021; Rajaram & Selvi, 2022; Vidyasankar & Vossen, 2011).

Therefore, how to efficiently combine various Web services has become a research hotspot in the field. The service composition problem mainly studies how to realize the requirements of complex tasks by combining existing services. Web service composition is becoming more and more complicated, and it is becoming increasingly challenging to select sub-services that meet the requirements. Many experts and scholars have provided their research ideas for this. Swarm intelligence optimization algorithms are more suitable for solving the NP problem of Web service composition (Abbasi et al., 2018), such as genetic algorithm, particle swarm algorithm, ant colony algorithm, firefly algorithm.

BACKGROUND

There are many ideas to solve the problem of Web service composition. Exhaustive algorithms can be used. However, exhaustive algorithms are computationally expensive and take a long time. Therefore, heuristic algorithms are widely used to solve such problems. Meyer proposes for the first time a Web service automatic composition algorithm based on heuristic search (Meyer & Weske, 2006). This algorithm supports the parallel creation of service composition and the creation of variables in the service composition process. However, the search space is too broad, and the running time is extended. Later, some scholars introduced swarm intelligence algorithms into this field. The swarm intelligence optimization algorithm is to find the optimal solution by imitating the habits of organisms and using the interaction between individuals (Zhang et al., 2010).

A genetic algorithm is an algorithm that simulates the process of biological genetics and evolution and has good global search capabilities. Literature (Amiri & Serajzadeh, 2010) modifies the function of generating chromosomes in the genetic algorithm to enhance the algorithm's speed, but

this function is random, and the result of the algorithm is unstable. References (Seghir & Khababa, 2018; Yang et al., 2019; Zhang et al., 2012) apply a multi-objective genetic algorithm to analyze multiple objective functions to solve the Web service composition problem. However, the algorithm is challenging to operate, and it is easy to fall into the local optimum, and it is not easy to achieve the global optimum solution. Therefore, some scholars develop the genetic algorithm by integrating other methods. Literature (Yilmaz & Karagoz, 2014) updates the exchange strategy of a genetic algorithm by combining it with a simulated annealing algorithm and chorus search algorithm. To decrease the algorithm's search space and advance the performance of the genetic algorithm, literature (Sivaram et al., 2019; Ye & Guan, 2016) alters the local optimization method of the candidate service set.

Compared with the genetic algorithm, the particle swarm algorithm has fewer parameters and accelerated convergence speed. Literature (Pourpanah et al., 2022; Wen et al., 2013) proposes a particle swarm algorithm based on the circular orbit of sub- particles and zero inertia weight to optimize Web service composition. Literature (Liao et al., 2014; Pourpanah et al., 2022) introduces a particle swarm optimization algorithm, which searches for the optimal solution according to the density of feasible solutions of the subpopulation, and increases the search ability of the population. Nevertheless, particle swarm algorithm generally has premature convergence, and it is obvious to fall into the local optimum.

Ant colony algorithm is another population intelligent optimization algorithm. Its concept is simple, easy to understand, and has fewer parameters than the particle swarm algorithm. Literature (Yang et al., 2019) recommends a dynamic ant colony algorithm that transforms web service composition into a directed acyclic graph and finds the shortest path among them. Literature (Yang et al., 2019; Zhang et al., 2010) joins genetic algorithm and ant colony algorithm, obtains the local solution through the improved ant colony algorithm, utilizes the local solution to initialize the genetic algorithm population, and finally gains the optimal solution through the genetic algorithm. Literature (Dahan et al., 2017; Yang et al., 2019) offers a flying ant colony optimization algorithm. The optimal ant will share pheromone with the surrounding ants, promoting the quality of the optimal solution but extending the running time. Literature (Dahan et al., 2019) states an improved flying ant colony algorithm, which adjusts the number of adjacent nodes receiving pheromone according to the quality of the solution, which better balances the exploration and development of the ant colony. Literature (Dahan et al., 2021) adds three different optimization methods based on FACO to improve the searchability of the algorithm. This method is the EFACO algorithm.

Since then, swarm intelligence optimization algorithms have developed swiftly, and various experts and scholars have recommended various swarm intelligence optimization algorithms to solve Web service composition optimization. Literature(Zhao et al., 2019) suggests an improved immune algorithm that applies heuristic algorithm ideas to reduce the search space and raise the algorithm's speed. References (F.-C. Pop et al., 2011; Qi et al., 2018) adjust the differential evolution algorithm, introduce structural knowledge, and submit new coding methods. Although the algorithm's optimization ability is improved, it is unsuitable for large-scale Web ser- vice composition scenarios. They aimed at the large-scale Web service portfolio, literature (Chandra & Niyogi, 2019; Huo et al., 2015) advanced artificial bee colony algorithms, and differential evolution algorithm to advance search efficiency. Literature (Attiya et al., 2020; Liu et al., 2008) combines the simulated annealing algorithm and the local greedy algorithm to comply the Web service composition. In 2008, Yang first introduced the firefly algorithm (Yang, 2009). Then Literature (Jiang et al., 2017; C. B. Pop et al., 2011; Rajeswari & Kanniappan, 2021) submitted a hybrid firefly algorithm, which consolidates the firefly algorithm with genetic operators and encodes all solutions through the planning graph structure to find the optimal solution.

Although the above algorithms have effectively solved Web service composition, they all have some problems, such as slow solution speed and effortless to fall into local optimum. For the optimization problem of Web service composition, this paper offers an enhanced firefly algorithm, which appends random disturbances when calculating the moving distance, which is better suited to explain the problem of discrete solution space. At the same time, the exchange and mutation mechanism in the genetic algorithm is added to make it converge faster and perform adequately.

METHOD

Web Service Composition Description

Definite a Web service be expressed as *WS=(S, F, QoS)*, where *S* is the service overview, *F* is the function implemented by the service, and *QoS* is the service quality. Each Web service often has multiple QoS attributes. Based on the different emphasis on the composite service, different QoS attributes can be selected. This paper mainly considers the five widely adopted QoS attributes: cost, response time, energy, availability, and reliability, and *QoS=(C, RT, E, A, R)*. There into, *C* is the cost, which refers to the fee that needs to be paid to the service provider to use the Web service. *RT* is the response time, which is when it takes to initiate a request to the Web service until it returns the result. *E* is the energy, which is the energy consumed by performing web services. *A* is availability, which refers to the degree to which the Web service is ready for rapid use. And *R* is reliability, which represents the possibility of fault-free execution of the Web service.

Suppose a complex composition of Web services is composed of n tasks, and there are m candidate Web services in each type of task, then $WSC = (G_1, G_2, ..., G_n) = (WS_{11}, WS_{12}, ..., WS_{nm})$ **Figure 1**. There are usually four structures for Web service composition: sequence, loop, parallelism, and selection, as shown in **Figure 2** (Dahan et al., 2017).

In practical applications, Web service composition may also have a more complex structure to resolve elaborate assignments. However, complicated structures can be decomposed into sequential structures in the end, so this paper only considers the composition of Web services with sequential structures **Figure 3**, and the comprehensive QoS attribute calculation equation is shown in **Table 1**(Dahan et al., 2017; Ibrahim et al., 2020).

Different algorithms have different dimensions, so the following equation is used to standardize the fitness of each algorithm(Alayed et al., 2019):

$$\mathrm{StandardScaler}\left(\mathrm{i}\right) = \frac{ALGO_{Resulti}}{\sum_{i=1}^{n} ALGO_{Resulti}} \tag{1}$$

$$i \in \left[\mathrm{EFA}, ACO, FACO, \mathrm{EFACO}, \mathrm{IA}, \mathrm{SA}, GA\right]$$

Figure 1.
Web service composition structure

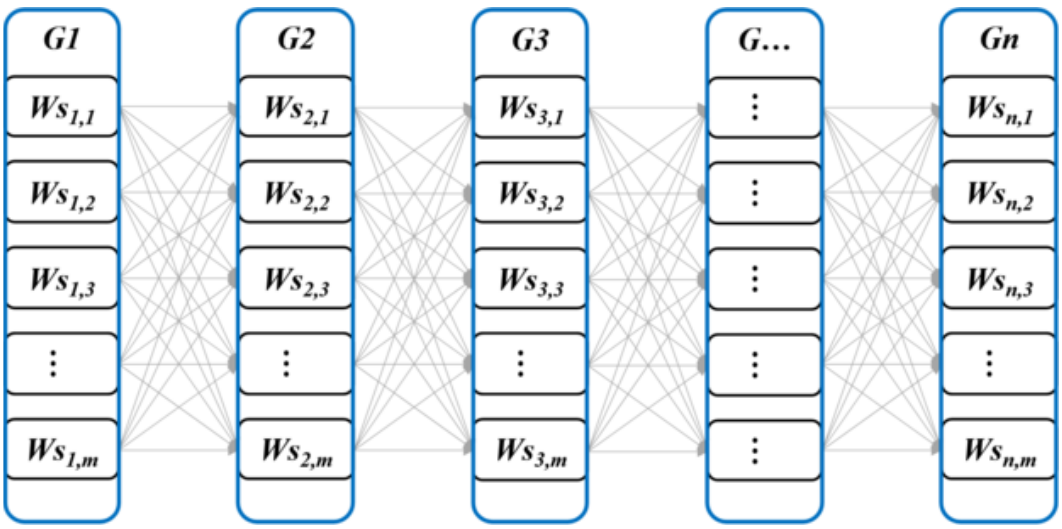


Figure 2.
The basic structure of web service composition

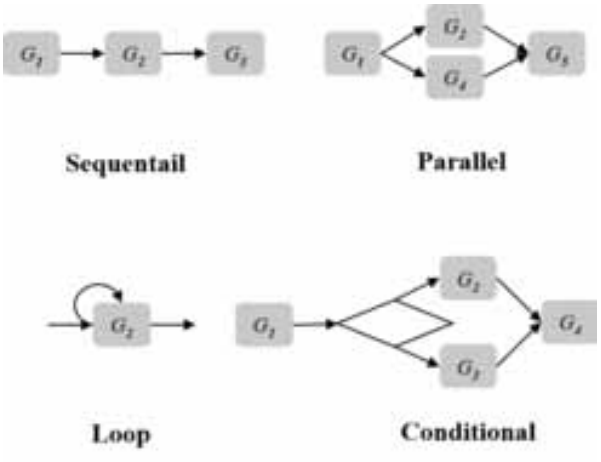
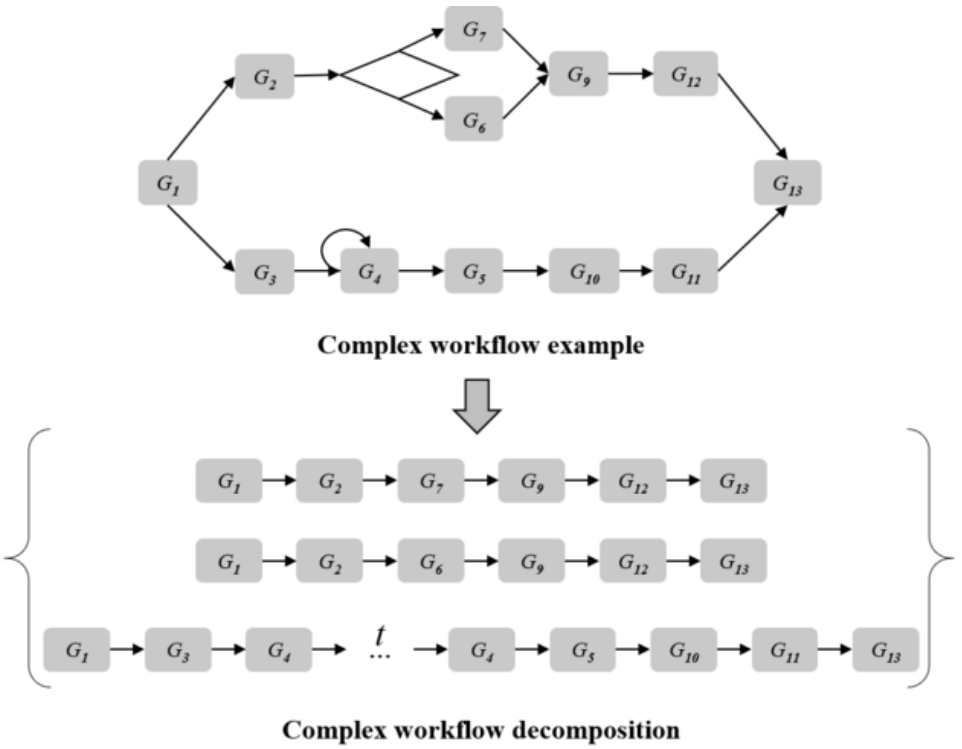


Figure 3.
Complex workflow example and decomposition



According to its QoS attribute, definite the objective function(Dahan et al., 2017; Ibrahim et al., 2020) of the web service composition be:

Table 1.
Calculation equations of QoS attribute in service composition

QoS attributes	Calculation formula
Cost (C)	$\sum_{i=1}^{n} C\left(WS_{ij}\right) \quad j \in m$
Response time (RT)	$\sum_{i=1}^{n} RT\left(WS_{ij}\right) \; j \in m$
Energy(E)	$\sum_{i=1}^{n} E\left(WS_{ij}\right) \; j \in m$
Availability (A)	$\prod_{i=1}^{n} A\left(WS_{ij}\right) \quad j \in m$
Reliability (R)	$\prod_{i=1}^{n} R\left(WS_{ij}\right) \; j \in m$

$$argmax_{k\in Pop}\left(F\right) = \frac{\left(W_a \prod_{i=1}^{n} A_{ix}^{k} + W_r \prod_{i=1}^{n} R_{ix}^{k}\right)}{\left(W_c \sum_{i=1}^{n} C_{ix}^{k} + W_{rt} \sum_{i=1}^{n} RT_{ix}^{k} + W_e \sum_{i=1}^{n} E_{ix}^{k}\right)} \tag{2}$$

Among them, *Wa, Wr, Wc, Wrt,* and *We* represent the weights of five QoS attributes of Web service availability, reliability, cost, response time, and energy, respectively.

Firefly Algorithm

Encode each firefly as a set of solutions as follows:

$$x_i = [ws_1,\ ws_2, ws_3 \ldots ws_n] \tag{3}$$

The ws_n is a Web service in task *n*.

In order to guarantee the orderliness of the candidate Web services in each task, the following equation is used to estimate and rank the comprehensive QoS attributes of each Web service(Dahan et al., 2017; Ibrahim et al., 2020) .

$$Q_k = \frac{w_a A_k + w_r R_k}{w_c C_k + w_{rt} RT_k + w_e E_k} \tag{4}$$

Then the candidate Web services in the k-th task are sorted as follows:

$$T_k = [ws_{k1}, ws_{k2}, ws_{k3} \ldots ws_{km}] \tag{5}$$

The *m* is the number of candidate Web services in each task.

Assuming that any two fireflies *a* and *b* in the solution space, their position variables x_a = $[x_{a1},x_{a2},x_{a3},...,x_{an}]$ and x_b = $[x_{b1},x_{b2},x_{b3},...,x_{bn}]$ are n-dimensional vectors. If the brightness *Ia* of the firefly *a* is greater than the brightness I_b of the firefly *b*, then the firefly *b* is attracted by the firefly *a* and moves to *a*, and the attraction β_{ab} can be expressed as:

$$\beta_{ab} = \beta_0 e^{-\gamma d_{ab}^2} \tag{6}$$

In the formula: β_0 is the attraction of the firefly at the light source (d = 0), generally taken as 1; γ is the light absorption coefficient, and the theoretical value range is [0, ¥); d_{ab} is the distance between the firefly a and the firefly b; *F* is calculated according to equation (2).

$$d_{ab} = 1 / \left(F_a - F_b\right) \tag{7}$$

In order to prevent the optimization from falling into the optimal local solution, random disturbance(Wang et al., 2017) is added to the attraction function:

$$\beta_{ab} = \begin{cases} rand1, & if \ \ rand2 < 0.5 \\ \beta_{ab} & else \end{cases} \tag{8}$$

In equation (8), *rand1* and *rand2* represent two random numbers with a range of [0, 1].

The firefly b moves to the firefly a under the action of the attraction β_{ab}, and its position update equation is as follows(Li et al., 2015; Wang et al., 2017; Yang, 2009):

$$x_a^{t+1} = x_b^t + \beta_{ab} * \left(x_a^t - x_b^t\right) + \pm\left(\mathrm{t}+1\right) * rand \tag{9}$$

In the equation: *t* is the number of iterations of the algorithm, α is the step size factor, rand Î *[−NWS/10, NWS/10]* is a random number, and *NWS* is the number of candidates for Web services in each type of task. In order to better approximate the optimal solution, adaptively modify the step size according to the number of iterations(Wang et al., 2017), where G_{max} represents the maximum number of iterations of the algorithm:

$$\pm\left(\mathrm{t}+1\right) = \left(1 - \frac{\mathrm{t}}{G_{max}}\right) * \pm\left(\mathrm{t}\right) \tag{10}$$

Suppose that after the firefly population moves, the optimal solution in the population is not updated. In that case, this iteration is considered invalid, and some fireflies are selected for Exchange and Mutation. Exchange refers to selecting the fireflies with the top 1/4 of the objective function value, calculating the current Web service appearance probability of each firefly, and exchanging 1-3 Web services with low appearance probability to Web services with high appearance probability. The mutation is to select the fireflies whose objective function value is ranked as the last 1/2, calculate

the probability of each web service task, and mutate 1-3 web services with a higher chance into any web service.

Algorithm Flowchart

This paper introduces an enhanced firefly algorithm for the Web service composition to better solve this problem. The specific algorithm implementation flow chart is shown in Figure 4. The specific steps of the firefly algorithm:

Step 1: Input data, load the QoS attributes of all candidate Web services, store them according to their task categories, use equation (4) to calculate and sort the Q values of the Web services in each task.
Step 2: Initialize the firefly algorithm parameters: population size n, iteration size Maxiteration, light intensity absorption coefficient g, maximum attraction b0, and step factor a.
Step 3: Initialize the firefly population and assign a random solution vector to each firefly.
Step 4: Calculate the current optimal solution in the firefly population based on the existing population, use equations (2) to calculate the fitness value F of each solution, and record the maximum F value of the population and its corresponding solution.
Step 5: Individuals in the firefly population move to the corresponding firefly according to their respective brightness. Use equations (6)(7)(8) to calculate the attractiveness bab between firefly a and b, and then use equations (9)(10) Update the solution represented by the moving firefly and use equation (2) to update the fitness value F of the moving firefly.
Step 6: Calculate the optimal solution of the current population and determine whether the population's optimal solution has been improved after the firefly population has relocated and otherwise exchange the better 1/4 individuals in the population. Mutate the worst 1/2 individual in the firefly population. If yes, go to step 7.
Step 7: Determine whether the algorithm termination condition is met, output the optimal solution if it is satisfied, otherwise skip to step 4 to continue execution.

SIMULATION EXPERIMENT

Experimental Environment and Data

All the calculation examples in this article are completed under the Ubuntu operating system. The computer CPU model is Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz, the quantity is 2, and the memory is 32GB. The algorithm's code implementation is based on Python 3.8.0, and the Numpy library version 1.20.3 is adopted in the calculation.

Database1, Database2, Database3, and Database4 employed in the experiment are real data sets constructed based on the real QWS data sets(Al-Masri & Mahmoud, 2007a, 2007b, 2008) QWS1 and QWS2. Among them, QWS1 and QWS2 contain 365 and 2507 essential web services, respectively, each in the data set Web services to include nine QoS attributes. This experiment selects three types of QoS attributes: response time, availability, and reliability. Due to its lack of cost and energy attributes, they are randomly generated. The cost is in the range of 45-50. And the energy is in a normal distribution with a mean of 25 and a variance of 5. Refer to **Table 2** for the number of tasks in the dataset and Web services in the tasks.

In order to compare the pros and cons of the algorithms, the experiments selected genetic algorithm(GA)(Amiri & Serajzadeh, 2010), immune algorithm(IA)(Zhao et al., 2019), simulated annealing algorithm (SA)(Attiya et al., 2020), ant colony algorithm(ACO)(Dahan et al., 2019; Zhang et al., 2010), flying ant colony algorithm(FACO)(Alayed et al., 2019; Dahan et al., 2019) and the improved flying ant colony algorithm(EFACO)(Dahan et al., 2021) six algorithms as a

Figure 4.
Flowchart of EFA algorithm

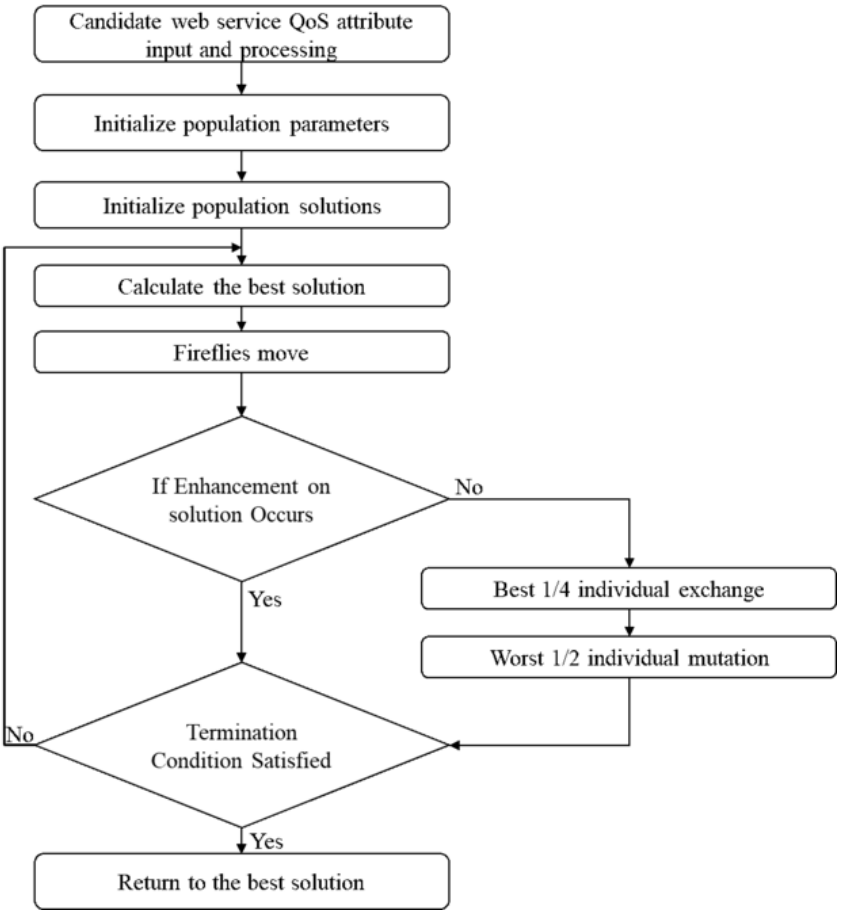


Table 2.
Web service data

ID	Tasks	Web services
Database1	10	30
Database2	20	15
Database3	30	75
Database4	50	25

comparison. Because of the experiment's accuracy and effectiveness, the combination optimization algorithms(Dahan et al., 2021) were tested in the same data set, and each group of experiments was repeated 50 times. To explain how the enhanced firefly algorithm (EFA) performs in different task quantities and under different Web services, expand the number of Web services or tasks based on a fixed number of tasks of 30 and a Web service of 30, respectively. Each combination is repeated 20 times based on the same data set experiment.

Parameter Set

In the experiment, the population size of each algorithm is set to 30, and the weights of Web service QoS attributes *A*, *R*, *C*, *RT* and *E* are set to: *Wa: 0.15, Wr: 0.15, Wc: 0.2, Wrt: 0.2, We:0.3* please refer to other parameter settings **Table 3**.

Results

Figure 5 results from seven algorithms running 50 times on the simulation data set Database1 with 10 tasks and 30 candidate Web services. There are nine segments: the average running time of the algorithm(mean-T), the average fitness(mean-V), the maximum running time of the algorithm(max-T), the maximum fitness of the algorithm(max-V), the coefficient of variation of the algorithm running time(cv-T), the coefficient of variation of the fitness(cv-V), the energy of the web service composition (E), the energy efficiency of the fitness(mean-V/E), the energy efficiency of the running time(mean-T/E). **Figure 6**, **Figure 7**, **Figure 8** are the comparison results of the EFA and the other six algorithms under the data sets with different task numbers and different candidate Web service numbers.

In summary, as shown in the above four figures, the EFA has no apparent advantage in running time, and adaptability is preferable when the data set is small. And, it has lower energy consumption and higher energy efficiency in time and fitness. Meantime, it has lower energy consumption and higher energy efficiency in time and fitness. When the data set is considerable, the EFA runs less time to optimize web service combinations. The result is more stable; the fitness value is significantly better than other algorithms; the maximum value is considerably higher than other algorithms; the coefficient of variation is the smallest; the energy efficiency of the fitness is exceptionally superior to other algorithms. Comparing **Figure 5** and **Figure 8**, when the number of tasks in the Web service composition is more numerous than the number of Web services, the EFA runs faster than the ACO, the FACO, and the EFACO. It is better than the SA in the coefficient of variation of the running time. It is significantly better than the other six algorithms in the solution of the best web service combination. At the same time, the coefficient of variation of the result is the smallest, and the energy efficiency is the highest.

Figures 9 and **10** show the energy consumption of the seven algorithms when the number of fixed web services or tasks is 30. As shown in the figure, as the number of tasks or web services rises, the energy consumption of EFA is at a comparatively high level.

Figures 11 and **12** show the comparison of runtime energy efficiency (T/E) among the seven algorithms when the number of fixed candidate Web services for web service combination is 30 or when the number of set tasks is 30. As the number of tasks or candidate Web services improvements, compared with ACO, FACO, and EFACO, the time required for the EFA algorithm to consume unit energy, the energy consumption rate gradually decreases as the number of tasks or the number of candidate Web services increase. The reduction is much smaller than the ACO, the FACO, and the EFACO. The overall trend is similar to the GA and the IA.

Figure 13 and **Figure 14** display the comparison of the energy efficiency of the service quality(V/E) among the seven algorithms when the number of fixed candidate Web services for web service combination is 30 or when the fixed task is 30. With the increase of the number of tasks or Web services, the energy efficiency of the EFA algorithm is more reliable than that of the other six algorithms.

In summary, in optimizing web service composition, the EFA has achieved more dependable calculation results with less energy investment. The EFA has a more excellent appearance in solving this problem.

Table 3.
Algorithm parameters

Algorithm ID	Parameter	Value
GA	*Mutation rate*	*0.8*
	$Max_{iteration}$	*400*
IA	*Mutation rate*	*0.8*
	T	*0.7*
	α	*0.95*
	$Max_{iteration}$	*400*
SA	*T_max*	*1500*
	T_min	*1*
	q	*0.99*
	$Max_{iteration}$	*300*
EFA	β_0	*1*
	α	*0.5*
	γ	*0.075*
	Fireflies	*30*
	$Max_{iteration}$	*120*
ACO	β	*1*
	α	*2*
	ρ	*0.9*
	τ_0	*0.1*
	Ants	*30*
	$Max_{iteration}$	*150*
FACO	β	*1*
	α	*2*
	ρ	*0.9*
	τ_0	*0.1*
	Ants	*30*
	$Max_{iteration}$	*150*
	k	*5*
	FR	*0.9*
EFACO	β	*1*
	α	*2*
	ρ	*0.9*
	τ_0	*0.1*
	Ants	*30*
	$Max_{iteration}$	*150*
	k	*5*
	FR	*0.9*

Figure 5.
The experimental results of Database1 maximizing the objective function

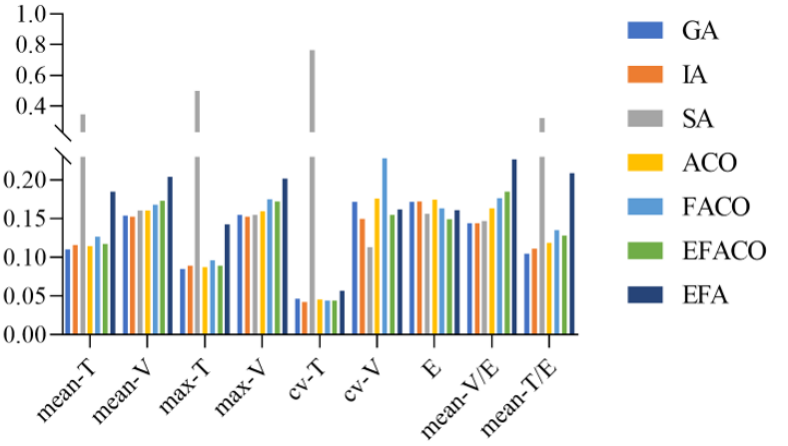
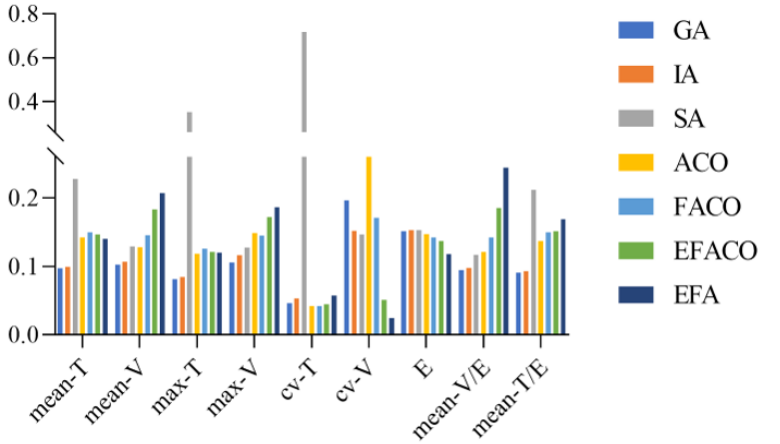


Figure 6.
The experimental results of Database2 maximizing the objective function



DISCUSSION

Web service portfolios introduce a new paradigm for distributed applications. However, the rapid growth of web service provisioning leads to the availability of many web services (WSs) that can perform the same task. It is challenging to construct a good web service portfolio workflow from many optional web services in a limited time. Here, we propose an enhanced firefly algorithm based on the QoS property of web services. We are demonstrating the implementation of the firefly algorithm through a custom transformation to adapt it to the web service combination (WSC) problem to achieve faster convergence and better merit search results.

Xin She Yang initially developed the Firefly algorithm as a nature-inspired metaheuristic algorithm for solving continuous optimization problems. However, FA can also solve discrete permutation problems such as the TSP problem and the flow shop scheduling problem. However, few studies have been conducted on applying the FA algorithm to the web service composition

Figure 7.
The experimental results of Database3 maximizing the objective function

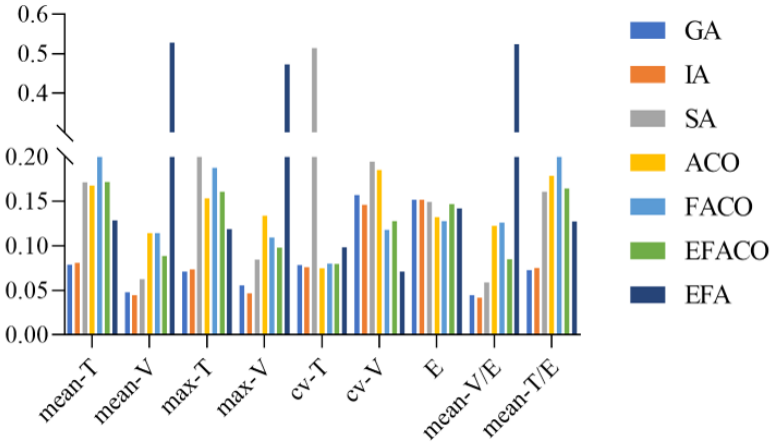
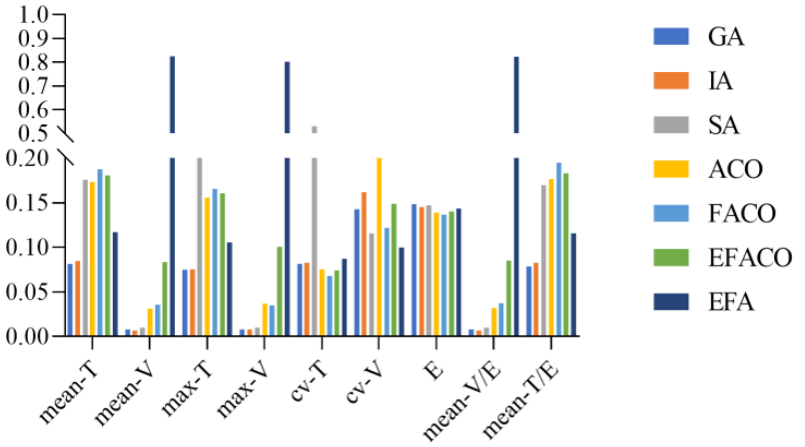


Figure 8.
The experimental results of Database4 maximizing the objective function



problem, which proved to be an NP-hard problem. We improve and optimize FA for solving the web service combination problem. At the same time, our method takes a similar index encoding approach in the EDFA and DFA algorithms. Also, it computes Q_k (**Equation 4**) based on the QoS property of individual web services for ranking, which enhances the ability of the algorithm to avoid local minimal traps to some extent. On the other hand, we improved the diversity of results to search for high-quality solutions by adopting a randomization mechanism in processing the attractiveness β_{ab} of fireflies (**Equation 8**). In addition, the randomization parameter α of the algorithm decreases gradually with the number of EFA iterations. This allows the algorithm to maintain high flexibility in the early iterations to obtain more excellent solutions. Still, as the number of iterations increases and the current optimal solution is already at a high level, the decrease of α helps to reduce the exploration of other sub-optimal results by EFA. This allows EFA to obtain better solutions within a more limited number of iterations and reduce the solution search time. Experimentation of these

Figure 9.
Comparison of the energy consumption of the web service at a fixed number of 30

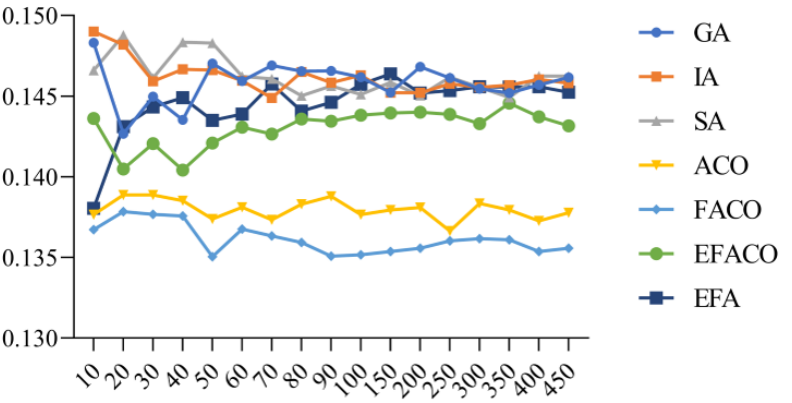
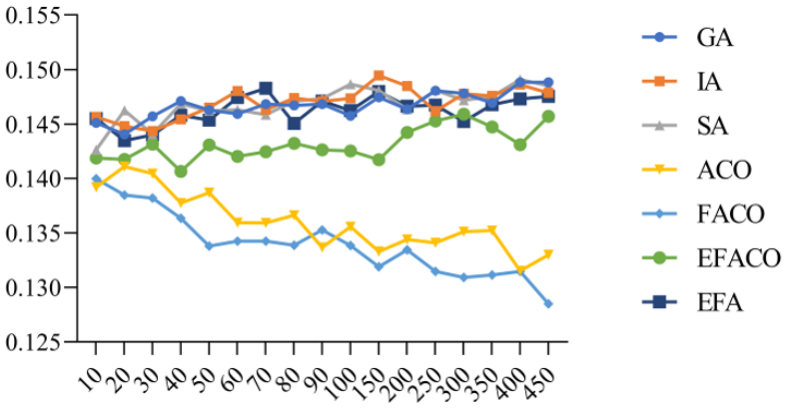


Figure 10.
Comparison of energy consumption when the number of tasks is fixed at 30



methods in EFA shows that the EFA algorithm has superior solving power and less solution time than other metaheuristics (**Figure 6-8**).

In addition, from a higher level, deep learning has its advantages over EFA-like heuristics in solving web service combination problems. Firstly, it has excellent learning ability and excellent solving ability. Secondly, it is highly adaptable and can be applied more easily and quickly in the face of web service combination problems. In addition, because of the data-driven nature of deep learning, the deep learning algorithm's performance will be further improved through the continuous increase of data sets and training iterations, which can better solve the web service combination problem. The research on deep learning in service combinations is also increasing gradually(Moustafa & Zhang, 2013). For example, Huagang Liang used PD-DQN to solve the service combination problem in cloud manufacturing(Liang et al., 2021). Hongbing Wang proposed a new service combination scheme based on the deep reinforcement learning (DRL) POMDP-WSC model(Wang et al., 2019). However, there are some disadvantages of deep-learning methods. High computational effort and high model complexity this back makes the solution time of the algorithm prolonged, which is difficult to meet when facing the needs of time-sensitive tasks(Liu & Lang, 2019). High data dependency, poor interpretability and

Figure 11.
Comparison of the energy efficiency of the running time of the web service at a fixed number of 30

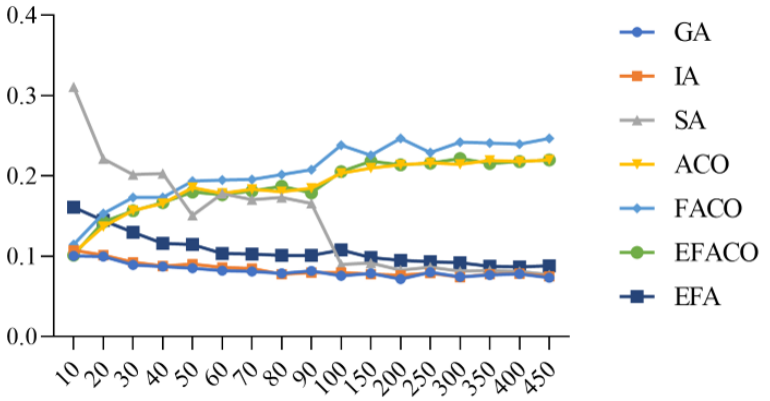
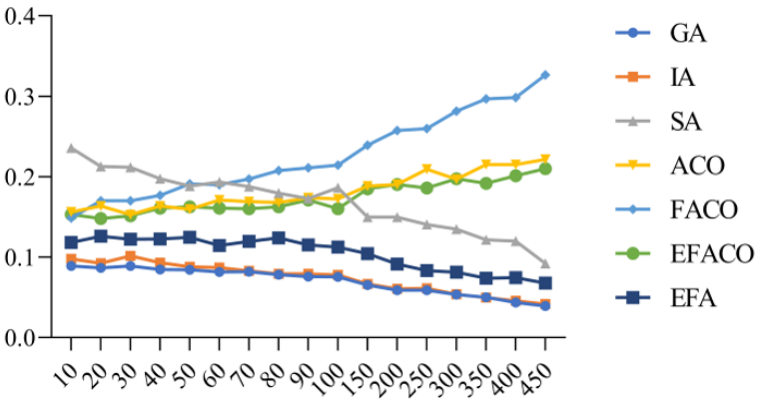


Figure 12.
Comparison of the energy efficiency of the running time when the number of tasks is fixed at 30



possible bias, etc., make it easier for previously trained models to cope with new situations when the environmental data changes(Johnson & Khoshgoftaar, 2019; Shinde & Shah, 2018). Considering their respective advantages and disadvantages, we believe that metaheuristics and deep learning algorithms complement each other. Integrating these two algorithms for application in solving web service problems may be helpful, where better solutions are obtained faster and more efficiently.

CONCLUSION

This paper analyzes the optimization of QoS-based Web service composition. It implements a new firefly algorithm with superior performance, EFA, which brings the population solution approaching the optimal solution in the traditional firefly algorithm to the principle of selection, exchange, and mutation in the genetic algorithm. The combination of mechanisms makes the enhanced firefly algorithm more fitting for optimizing Web service composition, diminishing the running time, and promoting the value of the objective function. The algorithm has two main innovations: adding random

Figure 13.
Comparison of the energy efficiency of the service quality when the number of web services is fixed at 30

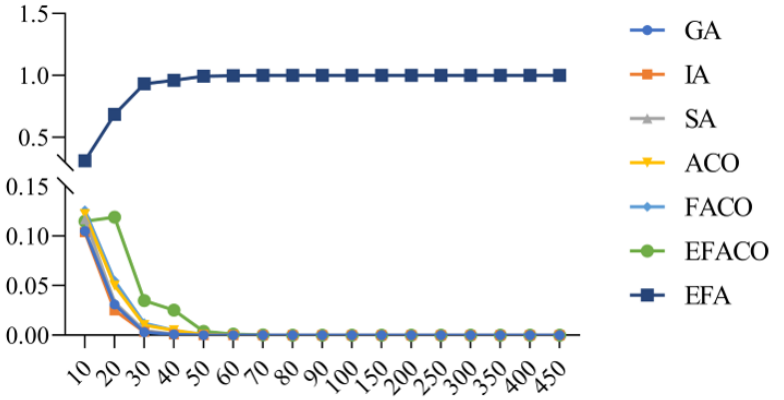
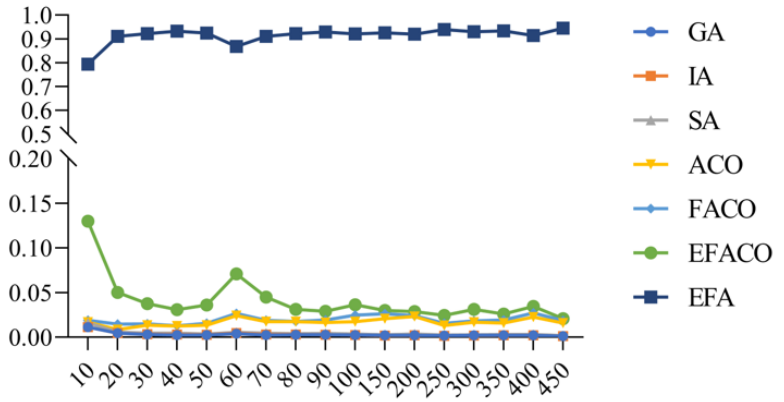


Figure 14.
Comparison of the energy efficiency of the service quality when the number of tasks is fixed at 30



disturbances when calculating the moving distance to adapt to the Web service composition problem; after the iteration is completed, the firefly population is exchanged and mutated. This mutation and exchanged increase the diversity, allow firefly to explore, and obtain a more dependable solution. Experiments have demonstrated that the EFA algorithm is superior to the EFACO, FACO, ACO, SA, IA, and GA algorithms in operating speed, robustness, and energy efficiency. In addition, in the application of large-scale data, compared to other algorithms, the EFA algorithm has more robust superiority.

ACKNOWLEDGMENT

This research was supported by National Natural Science Foundation of China [31770768], Fundamental Research Funds for the Central Universities[2572017PZ04], Heilongjiang Province Applied Technology Research and Development Program Major Project [GA18B301, GA20A301] and China State Forestry Administration Forestry Industry Public Welfare Project [201504307].

REFERENCES

Abbasi, N., Moeini, A., & Gandomani, T. J. (2018). Web Service Candidate Identification Using the Firefly Algorithm. [IJWSR]. *International Journal of Web Services Research*, *15*(4), 45–60. doi:10.4018/IJWSR.2018100103

Al-Masri, E., & Mahmoud, Q. H. (2007a). Discovering the best web service. *Proceedings of the 16th international conference on World Wide Web.* doi:10.1145/1242572.1242795

Al-Masri, E., & Mahmoud, Q. H. (2007b). Qos-based discovery and ranking of web services. *2007 16th international conference on computer communications and networks,* Hawaii, USA. 10.1145/1242572.1242795

Al-Masri, E., & Mahmoud, Q. H. (2008). Investigating web services on the world wide web. *Proceedings of the 17th international conference on World Wide Web*, Beijing, China. doi:10.1145/1367497.1367605

Alayed, H., Dahan, F., Alfakih, T., Mathkour, H., & Arafah, M. (2019). Enhancement of ant colony optimization for QoS-aware Web service selection. *IEEE Access : Practical Innovations, Open Solutions*, *7*, 97041–97051. doi:10.1109/ACCESS.2019.2927769

Amiri, M. A., & Serajzadeh, H. (2010). QoS aware web service composition based on genetic algorithm. *2010 5th International Symposium on Telecommunications,* Tehran, Iran.

Atkinson, B., Della-Libera, G., Hada, S., Hondo, M., Hallam-Baker, P., Klein, J., & Maruyama, H. (2002). *Web services security (WS-Security). Specification*. Microsoft Corporation.

Attiya, I., Abd Elaziz, M., & Xiong, S. J. C. (2020). Job scheduling in cloud computing using a modified harris hawks optimization and simulated annealing algorithm. *Hindawi*.

Chandra, M., & Niyogi, R. (2019). Web service selection using modified artificial bee colony algorithm. *IEEE Access : Practical Innovations, Open Solutions*, *7*, 88673–88684. doi:10.1109/ACCESS.2019.2926155

Dahan, F., El Hindi, K., & Ghoneim, A. (2017). An adapted ant-inspired algorithm for enhancing Web service composition. [IJSWIS]. *International Journal on Semantic Web and Information Systems*, *13*(4), 181–197. doi:10.4018/IJSWIS.2017100109

Dahan, F., El Hindi, K., Ghoneim, A., & Alsalman, H. (2021). An Enhanced Ant Colony Optimization Based Algorithm to Solve QoS-Aware Web Service Composition. *IEEE Access : Practical Innovations, Open Solutions*, *9*, 34098–34111. doi:10.1109/ACCESS.2021.3061738

Dahan, F., El Hindi, K., Mathkour, H., & AlSalman, H. (2019). Dynamic flying ant colony optimization (DFACO) for solving the traveling salesman problem. *Sensors (Basel)*, *19*(8), 1837. doi:10.3390/s19081837 PMID:30999688

Fantinato, M., Peres, S. M., Kafeza, E., Chiu, D. K. W., & Hung, P. C. K. (2021). A review on the integration of deep learning and service-oriented architecture. [JDM]. *Journal of Database Management*, *32*(3), 95–119. doi:10.4018/JDM.2021070105

Haddad, S., Mokdad, L., & Youcef, S. (2010). Selection of the best composite Web service based on quality of service. *INFORMATIK 2010–Business Process and Service Science–Proceedings of ISSS and BPSC*. Research Gate.

Huo, Y., Zhuang, Y., Gu, J., Ni, S., & Xue, Y. (2015). Discrete gbest-guided artificial bee colony algorithm for cloud service composition. *Applied Intelligence*, *42*(4), 661–678. doi:10.1007/s10489-014-0617-y

Ibrahim, G. J., Rashid, T. A., & Akinsolu, M. O. (2020). An energy efficient service composition mechanism using a hybrid meta-heuristic algorithm in a mobile cloud environment. *Journal of Parallel and Distributed Computing*, *143*, 77–87. doi:10.1016/j.jpdc.2020.05.002

Jiang, M., Jiang, L., Jiang, D., Xiong, J., Shen, J., Ahmed, S. H., Luo, J., & Song, H. (2017). Dynamic measurement errors prediction for sensors based on firefly algorithm optimize support vector machine. *Sustainable Cities and Society*, *35*, 250–256. doi:10.1016/j.scs.2017.08.004

Johnson, J. M., & Khoshgoftaar, T. M. (2019). Survey on deep learning with class imbalance. *Journal of Big Data, 6*(1), 1-54.

Li, M., Ma, J., Zhang, Y., Zhou, H., & Liu, J. (2015). Firefly algorithm solving multiple traveling salesman problem. *Journal of Computational and Theoretical Nanoscience*, *12*(7), 1277–1281. doi:10.1166/jctn.2015.3886

Li, S.-H., Huang, S., Yen, D., & Chang, C.-C. (2007). Migrating Legacy Information Systems to Web Services Architecture. [JDM]. *Journal of Database Management*, *18*(4), 1–25. doi:10.4018/jdm.2007100101

Liang, H., Wen, X., Liu, Y., Zhang, H., Zhang, L., Wang, L. J. R., & Manufacturing, C.-I. (2021). Logistics-involved QoS-aware service composition in cloud manufacturing with deep reinforcement learning. *Scientific Research Publishing*.

Liao, J., Liu, Y., Zhu, X., & Wang, J. (2014). Accurate sub-swarms particle swarm optimization algorithm for service composition. *Journal of Systems and Software*, *90*, 191–203. doi:10.1016/j.jss.2013.11.1113

Liu, H., & Lang, B. J. (2019). Machine learning and deep learning methods for intrusion detection systems. *Survey (London, England)*, *9*(20), 4396. doi:10.3390/app9204396

Liu, Q., Zhang, S.-L., Yang, R., & Lian, X.-J. (2008). Web services composition with QoS bound based on simulated annealing algorithm. [English Edition]. *Journal of Southwest University (Natural Science Edition)*, *24*(3), 308–311.

Meyer, H., & Weske, M. (2006). Automated service composition using heuristic search. *International Conference on Business Process Management,* Berlin, Heidelberg. doi:10.1007/11841760_7

Moustafa, A., & Zhang, M. (2013). Multi-objective service composition using reinforcement learning. Service-Oriented Computing: *11th International Conference, ICSOC 2013*, Berlin, Germany.

Pop, C. B., Chifu, V. R., Salomie, I., Baico, R. B., Dinsoreanu, M., & Copil, G. (2011). A hybrid firefly-inspired approach for optimal semantic web service composition. *Scalable Computing: Practice and Experience*, *12*(3), 363–370.

Pop, F.-C., Pallez, D., Cremene, M., Tettamanzi, A., Suciu, M., & Vaida, M. (2011). Qos-based service optimization using differential evolution. *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, Dublin, Ireland. doi:10.1145/2001576.2001830

Pourpanah, F., Wang, R., Lim, C. P., Wang, X.-Z., & Yazdani, D. J. A. I. R. (2022). *A review of artificial fish swarm algorithms: Recent advances and applications*, 1-37.

Qi, J., Xu, B., Xue, Y., Wang, K., & Sun, Y. (2018). Knowledge based differential evolution for cloud computing service composition. *Journal of Ambient Intelligence and Humanized Computing*, *9*(3), 565–574. doi:10.1007/s12652-016-0445-5

Rajaram, K., & Selvi, K. (2022). CCC-Quality2: Cross-Cloud Service Composition Based on Quality of Clouds and Web Services for Business Applications. In Information and Communication Technology for Competitive Strategies (ICTCS 2021) (pp. 413-422). Springer.

Rajeswari, P., & Kanniappan, J. (2021). Hybrid Metaheuristics Web Service Composition Model for QoS Aware Services. *Computer Systems Science and Engineering*, *41*(2), 511–524. doi:10.32604/csse.2022.020352

Ramalingam, C., & Mohan, P. J. S. (2021). Addressing semantics standards for cloud portability and interoperability in multi cloud environment. *Symmetry, 13*(2), 317.

Seghir, F., & Khababa, A. J. J. (2018). A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition. *Journal of Intelligent Manufacturing, 29*, 1773-1792.

Shinde, P. P., & Shah, S. (2018). A review of machine learning and deep learning applications. *2018 Fourth international conference on computing communication control and automation (ICCUBEA).*

Sivaram, M., Batri, K., Amin Salih, M., & Porkodi, V. (2019). Exploiting the local optima in genetic algorithm using tabu search. *Indian Journal of Science and Technology*, *12*(1), 1–13.

Tsalgatidou, A., & Pilioura, T. (2002). An overview of standards and related technology in web services. *Distributed and Parallel Databases*, *12*(2), 135–162. doi:10.1023/A:1016599017660

Vidyasankar, K., & Vossen, G. (2011). Multi-Level Modeling of Web Service Compositions with Transactional Properties. *Journal of Database Management (JDM), 22,* 1–31. 10.4018/jdm.2011040101

Wang, H., Gu, M., Yu, Q., Tao, Y., Li, J., Fei, H., & Hong, T. J. K.-B. S. (2019). Adaptive and large-scale service composition based on deep reinforcement learning. *Knowledge-Based Sustems, 180*, 75-90.

Wang, H., Zhou, X., Sun, H., Yu, X., Zhao, J., Zhang, H., & Cui, L. (2017). Firefly algorithm with adaptive control parameters. *Soft Computing*, *21*(17), 5091–5102. doi:10.1007/s00500-016-2104-3

Wen, T., Sheng, G.-J., Guo, Q., & Li, Y.-Q. (2013). Web service composition based on modified particle swarm optimization. [Chinese Journal of Computers]. *Jisuanji Xuebao*, *36*(5), 1031–1046. doi:10.3724/SP.J.1016.2013.01031

Yaghoubi, M., Maroosi, A. J. (2020). Simulation and modeling of an improved multi-verse optimization algorithm for QoS-aware web service composition with service level agreements in the cloud environments. *Simul. Model Pract. Theory, 103*, 102090.

Yang, X.-S. (2009). Firefly algorithms for multimodal optimization. *International symposium on stochastic algorithms,* Berlin, Heidelberg. doi:10.1007/978-3-642-04944-6_14

Yang, Y., Yang, B., Wang, S., Liu, F., Wang, Y., & Shu, X. J. (2019). A dynamic ant-colony genetic algorithm for cloud service composition optimization. *The International Journal of Advanced Manufacturing Technology, 102*, 355-368.

Ye, H., & Guan, Y. (2016). QoS-aware web service composition based on local selection and genetic algorithm. *Journal of Chinese Computer Systems*, *37*(7), 1389–1392.

Yilmaz, A. E., & Karagoz, P. (2014). Improved genetic algorithm based approach for QoS aware web service composition. 2014 IEEE international conference on web services, Alaska, USA.

Zhang, W., Chang, C. K., Feng, T., & Jiang, H.-y. (2010). QoS-based dynamic web service composition with ant colony optimization. *2010 IEEE 34th Annual Computer Software and Applications Conference*, Seoul, Korea.

Zhang, Y. Y., Xiong, H. L., & Zhang, Y. C. (2012). An improved genetic algorithm of web services composition with qos. *Advanced Materials Research*.

Zhao, X., Li, R., & Zuo, X. J. (2019). Advances on QoS-aware web service selection and composition with nature-inspired computing. *CAAI Transactions in Intelligence Technology*, *4*(3), 159–174.

Yifei Xue is a graduate student in the School of Information and Computer Science, Northeast Forestry University. Her research focuses on swarm intelligence optimization algorithms, which are applied to web service neighborhoods.

Wang Jian is an associate professor, supervisor of master's degree, and director of communication engineering in the School of Information and Computer Engineering, Northeast Forestry University. His main research interests are the Internet of things, signal and information processing, etc.

Jing Weipeng is a professor (doctoral supervisor) of the School of Information and Computer Engineering of Northeast Forestry University, deputy director of the Key Laboratory of Forestry big data and High-performance Computing of the State Forestry Administration, vice dean of the School of Information and Computer Engineering of Northeast Forestry University, a senior member of Chinese computer Society, member of IEEE, and member of computer Application expert Committee of Chinese computer Society. His main research interests are massive spatial data management, parallel computing, distributed computing, intelligent interpretation of remote sensing images, and intelligent processing of spectral data.