IFOA4WSC: a quick and effective algorithm for QoS-aware service composition

Yiwen Zhang, Guangming Cui* and Shu Zhao

Department of Computer Science and Technology,
Anhui University,
Hefei, Anhui, China
Email: zhangyiwen@ahu.edu.cn
Email: cuiguangming2010@qq.com
Email: zhaoshuzs2002@hotmail.com
*Corresponding author

Jie Tang

Department of Computer Science and Technology,
Tsinghua University,
Beijing, China
Email: jietang@tsinghua.edu.cn.

Abstract: In the service-oriented environment, large-scale Service Composition (SC) has become an important research, where Quality of Service (QoS) has been applied widely. QoS is generally used to represent non-functional properties of web services and differentiate them with the same functionality. Studying how to select appropriate services for a composition from a very large number of similar candidate services is an NP-hard problem. However, there are limitations among existing methods, e.g. poor scalability and massive calculation in the exhaustion method, slow convergence speed and easy to fall into local optimum in the traditional evolutionary computation method. Thus, in this paper, an improved fruit fly optimisation algorithm called IFOA4WSC is proposed, which is quick and effective for web service composition. We analyse the convergence of fruit fly swarms in IFOA4WSC algorithm, and carry out a large number of simulation experiments in the common and random data sets. Experimental results show that IFOA4WSC is effective and highly efficient, and has better stability against classical PSO, GA and FOA.

Keywords: IFOA4WSC; service composition; QoS; quality of service; convergence analysis.

Reference to this paper should be made as follows: Zhang, Y., Cui, G., Zhao, S. and Tang, J. (2016) 'IFOA4WSC: a quick and effective algorithm for QoS-aware service composition', *Int. J. Web and Grid Services*, Vol. 12, No. 1, pp.81–108.

Biographical notes: Yiwen Zhang is an Associate Professor in the School of Computer Science and Technology, Anhui University. He received the Master's degree in Computer Software and Theory in 2006 and the PhD degree

in Management Science and Engineering in 2013 from the Hefei University of Technology. His research interests include services computing, cloud computing, e-commerce and swarm intelligence.

Guangming Cui is a Master's degree Candidate in the School of Computer Science and Technology, Anhui University. His current research interests include services computing, cloud computing and swarm intelligence.

Shu Zhao is an Associate Professor in the Department of Computer Science and Technology, Anhui University. She received her PhD degree in Computer Science from Anhui University in 2007. Her current research interests include social network, granular computing and machine learning.

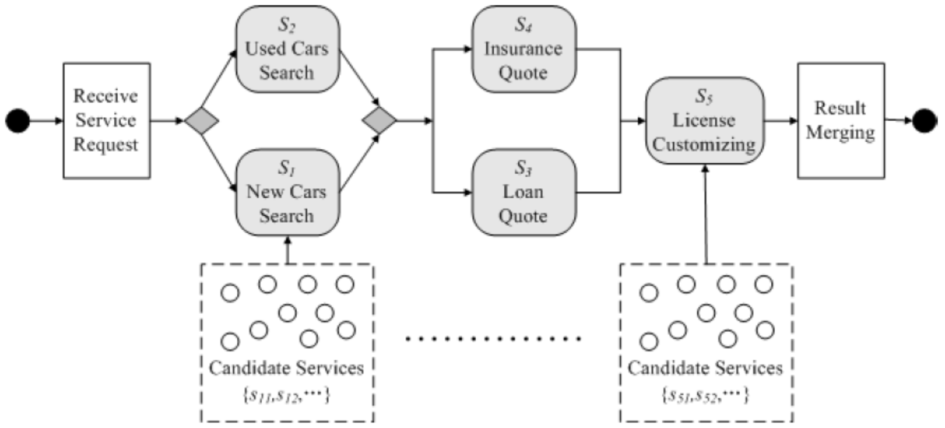
Jie Tang is an Associate Professor in the Department of Computer Science and Technology, Tsinghua University. His main research interests include data mining algorithms and social network theories. He has been Visiting Scholar at Cornell University, Chinese University of Hong Kong, Hong Kong University of Science and Technology, and Leuven University. He has published over 100 research papers in major international journals and conferences including *KDD*, *IJCAI*, *WWW*, *SIGMOD*, *ACL*, *Machine Learning Journal*, *TKDD*, *TKDE* and *JWS*.

1 Introduction

With the development of distributed technology and Extensible Markup Language (XML) technology, web service technology appeared. Web service absorbs the advantages of distributed computing, grid computing and XML technology, and has been applied successfully to heterogeneous distributed computation, as well as reuse of code and data. In the meantime, it also has merits of high inter-operability, platform independence, loosely coupling and highly integrated capability. Nevertheless, a single web service cannot often meet the needs of users. Consequently, web service technology aims to enable the inter-operation of heterogeneous systems, reuse of distributed functions in an unprecedented scale and create value-added business applications composed by dynamically selected individual services (Rao and Su, 2004; Zeng et al., 2004). This technology is called Service Composition (SC), and it has achieved significant success.

However, selecting web services for a composition is not an easy problem because there may be many web services equal in functionality but different in Quality of Service (QoS). One of the basic concerns is to optimise the overall QoS value of the composition. Figure 1 shows an example of a web application for finding the best car offers. This example originates from He et al. (2014) and is adapted to the characteristics of this research. In this example, for each sub-task (shown in the grey box), a very large number of candidate services are available to provide the required functionality but with different QoS values. Users are typically unaware of the involved services. They specify their QoS requirements in the Service-Level Agreement (SLA) in accordance with QoS constraints (e.g. maximum availability, minimum response time, minimum cost, maximum reputation). The goal of QoS-aware service composition is to select one service for each sub-task such that the aggregated QoS values satisfy all QoS constraints at the application level. Key issue in optimising the overall QoS value is time complexity and it has proved to be an NP-hard problem (He et al., 2014; Yilmaz and Karagoz, 2014; Wang et al., 2014).

Figure 1 Example of service composition


Receive Service Request
S_2 Used Cars Search
S_1 New Cars Search
S_4 Insurance Quote
S_3 Loan Quote
S_5 License Customizing
Result Merging
Candidate Services $\{s_{11}, s_{12}, \cdots\}$
Candidate Services $\{s_{51}, s_{52}, \cdots\}$

In addition, with the arrival of the era of cloud computing, the concept of 'everything as a service' makes service computing even more popular. Users' needs increase continuously and consequently the service computing system takes on notable characteristics including dynamics, loose coupling and large scale. At the same time, more and more service suppliers begin to provide web services that have similar functions but different levels, which leads to the increase of the number and scale of service. Therefore, the problem of non-functional parametric SC receives more and more attention. Nevertheless, the SC problem is complex, multi-polar, strong constraint, dynamical and high dimensional. It is difficult for the traditional optimisation techniques [such as Integer Programming (IP) (Hossain et al., 2012), Dynamic Programming (DP) (Uc-Cetina et al., 2014), Linear Programming (LP) (Gabrel et al., 2014) and graph algorithm (Chen and Yan, 2014; Lin et al., 2012] to be applied to this new application scenario. Although exhaustive method can guarantee gaining the optimal solution, it is only suitable for small-scale application scenarios. As a classical optimisation algorithm, DP is good at solving a problem with two key factors: optimal substructure and overlapping sub-problem. The runtime of DP depends on the total number of the sub-problems and the numbers chosen in sub-problems. Compared with the exhaustive method, DP improves the efficiency greatly, but its runtime is still pseudo-polynomial. LP combines the SC problem for the multidimensional purpose by an abstract optimisation model generally following the linear fitting. However, it is difficult for the LP method to find the optimal solution to the problem efficiently when the dimension of the problem is large. SC problem can also be abstracted as a graph-theoretic problem, which can be solved by graph algorithm. Graph algorithm is more intuitional, but the problem of SC needs to be converted to obtain topology among the services at first. In addition, the acquisition of topology information will consume substantial time. There are also many intelligent optimisation algorithms such as Genetic Algorithm (GA) (Niewiadomski et al., 2014), Particle Swarm

Optimisation (PSO) (Fang et al., 2014; Zhang, 2014) Ant Colony Optimisation (ACO) (Yu et al., 2015; Rostami et al., 2014) and Fruit Fly Optimisation Algorithm (FOA) (Pan, 2012). Among them, FOA is the simplest but a relatively powerful one. It has only three control parameters, and its procedure is simple to understand and implement (Pan, 2012; Shan et al., 2013; Li et al., 2013).

However, these optimisation algorithms still have some shortcomings for practical applications. For instance, GA is strongly dependent on the initial population and is time-consuming. PSO relies heavily on the parameter settings and may fall into local optimum easily. ACO is easy to lapse into stagnation and converges slowly. FOA is still not successfully applied to web SC optimisation.

We propose a novel and quick algorithm IFOA4WSC to meet the large-scale SC scenarios which remains effective and efficient when the candidate services scenario scales up. In particular, our main contributions can be summarised as follows.

1 An improved quick and effective optimisation algorithm, IFOA4WSC, for QoS-aware large-scale SC, is proposed, which adopts dynamic step, information sharing and elitist strategy. Furthermore, the motion and position equation of fruit fly swarm are presented.

2 The convergence condition of fruit fly swarm in IFOA4WSC is analysed. It postulates that random value is constant, which provides theoretical foundation for further improving the effectiveness and the efficiency of IFOA4WSC.

3 A comprehensive comparison of state-of-the-art heuristic optimisation algorithms is presented. These benchmark algorithms are often used to evaluate the performance of various new SC optimisation methods.

In our evaluations, we show that our algorithm is effective, highly efficient and more stable than classical PSO, GA and FOA for QoS-aware service composition.

The structure of this paper is as follows. Section 2 gives the related works of QoS-aware SC and Section 3 elaborates the formulation of SC problem. The IFOA4WSC algorithm, including its modelling, convergence analysis and implementation, is discussed in Section 4. Section 5 compares our algorithm with PSO, GA and FOA. Experimental results in Section 6 demonstrate the benefits of our algorithm. Conclusions are drawn in Section 7.

2 Related works

There are many researches applying intelligent optimisation algorithm to the problem of QoS-aware SC. Wang et al. (2013) designed a genetic algorithm for web service selection based on QoS awareness by the relational matrix coding. Vasumathi and Moorthi (2012) presented an algorithm called ANN-PSO which combines adaptive neural network with PSO. El Hadad et al. (2010) proposed a service selection algorithm to meet the user preference. User preferences can be used as the weight of QoS standard and the risk level to make semantic definition of transactional demand. Alrifai et al. (2010) reduced the number of candidate services and selected services more efficiently based on skyline. Wang et al. (2010) introduced a QoS-aware service selection model based on fuzzy linear programming. The model can distinguish the differences of candidate services and assist consumers to opt the optimal service according to their expectations and preferences. Liu et al. (2010) used ACO in generating fitter individuals in initial population of GA to improve the convergence rates. Yilmaz and Karagoz (2014) proposed two hybrid GAs and adopted the heuristics Simulated Annealing (SA) (Giedrimas and Sakalauskas, 2012) and Harmony Search (HS) (Geem et al., 2001) as smart mutation operator.

Off-the-shelf researches play a great role in developing the theory and application of SC. Nevertheless, the existing methods generally reduce the solution scale based on some constraints (such as user preferences) or operations (such as Skyline) and then obtain the optimal solution. They sacrifice some identical or similar web services and have very strong subjectivity. Especially, there are still some defects such as prematurity and long response time in applications of QoS-aware large-scale SC. In addition, the common disadvantages of these stochastic algorithms are complicated computational processes which cause difficulty for beginners to understand their principles. Hence, a quick and effective service composition is indispensable to achieve close-to-optimal results within an acceptable period of time.

FOA proposed by Pan (2012) is a novel, evolutionary computation and optimisation technique. This new optimisation algorithm has the advantages of being easy to understand and to be written into program code which is not too long compared with other algorithms. Therefore, FOA has higher optimisation efficiency than other classical PSO, GA for the same number of iterations. However, FOA also has some disadvantages: (1) it may fall into local optimal or be premature in multi-model optimisation problems (Yuan et al., 2014) and (2) it cannot solve complex optimisation problems effectively (Shan et al., 2013). Therefore, the design of high-performance intelligent optimisation algorithm is imperative for solving the large-scale SC problem with QoS-aware multidimensions efficiently.

3 Problem formulation

The QoS-aware service composition problem is briefly defined as follows (Liu et al., 2014): given a fixed service process consisting of a set of sub-tasks and the structural relationships between the sub-tasks, the algorithm selects one concrete service component for each sub-task from its candidate service set, and the final result satisfies the QoS constraints raised by a user. The QoS attributes we consider here include availability, response time, cost and reputation.

Definition 1 [web service (s)]: *Web service is a four-tuple. s = {ID, Source, Function, QoS}, where ID is the unique identification of web service, Source is the fundamental information including service name and publisher; Function is a function describing web services and QoS is the quality of web service. s ∈ S, where S represents web service set, which has identical functions but different non-functional attributes (QoS).*

Definition 2: QoS can be expressed as a four-tuple, $QoS = \{Ava, RT, Cost, Rep\}$, where Ava, RT, $Cost$ and Rep represent the availability, response time, cost and reputation, respectively.

(1) Availability (*Ava*). Availability refers to the time when a web service can provide a specific service, i.e. the proportion of the time when web service is accessed successfully in the time period *t*:

$$Ava = \frac{t_{success}}{t}$$

where $t_{success}$ represents the time of accessing service successfully within the time period *t*.

(2) Response time (*RT*). Response time is the time from a user submitting service request to service execution being complete and returning results. It mainly includes web service execution time RT_{exe}, network transmission time RT_{trans} and other time consumption RT_{oth}:

$$RT = RT_{exe} + RT_{trans} + RT_{oth}$$

(3) Cost (*Cost*). Cost means the total fee from a user submitting service request to service execution being complete and returning results. It mainly includes service basic cost $Cost_{serv}$ such as software (*SaaS*), hardware (*IaaS*) and platform (*PaaS*), and service management cost $Cost_{mang}$:

$$Cost = Cost_{serv} + Cost_{manag}$$

(4) Reputation (*Rep*). Reputation measures the degree of the reliability of a web service, mainly based on the user's experience of using web service. Different users may have different evaluations on the same web service:

$$Rep = \sum_{i=1}^{n} \frac{Rep_i}{n}$$

where Rep_i represents the evaluation that the *i*th terminal user has on web service.

Definition 3: *Service composition is a multi-tuple.* $SC = \left\{ s_i^j \middle| s_i^j \in S_i \wedge i \in [1,n] \wedge j \in [1,m_i] \right\}$, *where (1) S_i denotes the set of candidate services of the ith sub-task, (2) s_i^j represents the jth web service of candidate service set in the ith sub-task and (3) n and m_i represent the number of sub-tasks and candidate services in the ith sub-task, respectively. Thus, we can conclude that* $SC \in S_1 \times S_2 \times \ldots \times S_{n-1} \times S_n$.

Any execution path of web SC is composed of four fundamental patterns (Al-Helal and Gamble, 2014; Liu et al., 2004) (Figure 2), including (a) sequence, (b) selection, (c) parallel and (d) loop.

(1) *Sequential model (Figure 2a).* The computation formula for *QoS* is listed in Table 1 (a).

(2) *Selection model (Figure 2b).* Supposing that the probability of each service S_i being selected is λ_i. $\sum_{i=1}^{n} \lambda_i = 1$. The computation formula for *QoS* is listed in Table 1 (b).

(3) *Parallel model (Figure 2c).* Each service S_i is executed in parallel and the computation formula for *QoS* is listed in Table 1 (c).

(4) *Loop model (Figure 2d).* Supposing that circulation model is executed θ times. The computation formula for *QoS* is listed in Table 1 (d).

Figure 2 Four fundamental patterns for *SC*

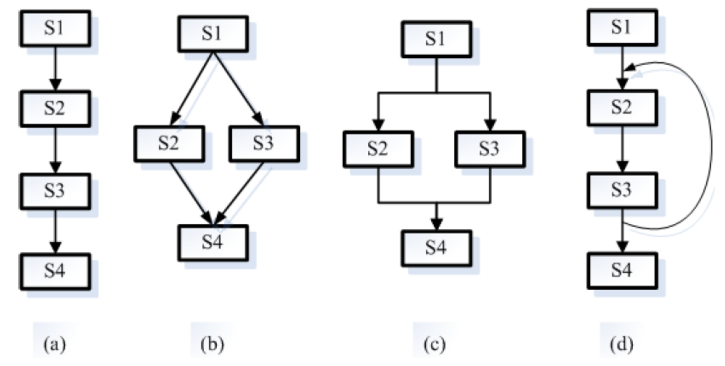


Table 1 The QoS computation formula of four basic composition patterns

Aggregation function	*Attribute*			
	Availability (Ava)	*Response time (RT)*	*Cost (Cost)*	*Reputation (Rep)*
Sequential model (a)	$\prod_{i=1}^{n} Ava_i$	$\sum_{i=1}^{n} RT_i$	$\sum_{i=1}^{n} Cost_i$	$\prod_{i=1}^{n} Rep_i$
Selection model (b)	$\prod_{i=1}^{n} Ava_i \times \lambda_i$	$\sum_{i=1}^{n} RT_i \times \lambda_i$	$\sum_{i=1}^{n} Cost_i \times \lambda$	$\prod_{i=1}^{n} Rep_i \times \lambda_i$
Parallel model (c)	$\prod_{i=1}^{n} Ava_i$	$MAX(RT_i), i \in [1,n]$	$\sum_{i=1}^{n} Cost_i$	$\prod_{i=1}^{n} Rep_i$
Loop model (d)	Ava^{θ}	$RT \times \theta$	$Cost \times \theta$	Rep^{θ}
Dimension type	Increasing	Decreasing	Decreasing	Increasing
Constraint type	Upper	Lower	Lower	Upper

Assuming that the other three non-sequential structures can be converted to the sequential, this paper adopts the sequential structure model as the basis to research issue of web SC optimisation.

Definition 4 (QoS pretreatment): *Different types of indexes have different dimensions. It is necessary to eliminate the incommensurability stemming from different dimensions and different dimensional units. Therefore, all indexes need to be normalised to a dimensionless interval according to a certain utility function (usually it is normalised to [0, 1]) before the cooperative evaluation. This paper only considers the discrete indexes and normalises them according to the following equation*:

$$q_{ij} = \begin{cases} \begin{cases} \dfrac{q_j^{\max} - q_{ij}}{q_j^{\max} - q_j^{\min}} & q_j^{\max} \neq q_j^{\min} \\ 1 & q_j^{\max} = q_j^{\min} \end{cases} & benefit\_type \\ \begin{cases} \dfrac{q_{ij} - q_j^{\min}}{q_j^{\max} - q_j^{\min}} & q_j^{\max} \neq q_j^{\min} \\ 1 & q_j^{\max} = q_j^{\min} \end{cases} & \cos t\_type \end{cases} \tag{1}$$

where $q_j^{\max}$ and $q_j^{\min}$ represent the current maximum and minimum values of the *j*th evaluation index, respectively. Their values will change dynamically with the joining or withdrawal of web service.

4 IFOA4WSC: the proposed algorithm

IFOA4WSC is based on the fruit flies' foraging characteristics, which adopt the dynamic step and elitist strategy to obtain the optimal SC. Our discussion will be as follows. Firstly, the IFOA4WSC algorithm is modelled according to the problem formulation of SC, including individual fruit fly coding, distance and smell concentration judgement value definition. Secondly, the fitness function is defined to judge the pros and cons of SC. Thirdly, the convergence of fruit fly swarm is analysed and then the IFOA4WSC algorithm is implemented finally.

4.1 Algorithm modelling

4.1.1 Individual fruit fly

In the experiment, service composition is a five-tuple, i.e. each record of *SC* contains five sub-service processes. Therefore, when each sub-service is regarded as a single individual fruit fly, each *SC* contains five individual fruit flies. Each fruit fly belongs to different sub-service sets. In this way, IFOA4WSC iterates with the five files separately, and then the optimal solution is obtained by using the fitness function value of each composition. Finally, the location of the five flies in the optimal service set is just the location of each sub-service.

4.1.2 Coding of individual fruit fly and motion equation

The code for individual fruit fly adopts a discrete method called integer encoding. The integer represents the location of the fruit fly in its set of sub-service. Motion equation is defined by the direction and the size according to which fruit fly swarm can move optimally. Thus, the motion equation is as follows:

$$V_{bn}^{t+1} = \omega V_{bn}^{t} + C_1\delta\left(P_{bn}^{t} - X_{bn}^{t}\right) + C_2\eta\left(P_{gn}^{t} - X_{bn}^{t}\right) \tag{2}$$

The further position equation is

$$x_{i,n}^{t+1} = X_{bn}^{t} + V_{bn}^{t+1} \times \theta \quad \theta \in [-1,1] \tag{3}$$

where P_{bn}^{t} and P_{gn}^{t} represent the optimal location of fruit fly swarm in the *t*th iterative process and the global optimal position which has been found after *t* iterations, respectively. X_{bn}^{t} represents the optimal location of individual fly in the *t*th iterative process and the location of fruit fly swarm in the next iterative process, V_{bn}^{t+1} represents the movement speed of fruit fly swarm in the (*t* + 1)th iterative process and $x_{i,n}^{t+1}$ represents the *n*th-dimensional location of the *i*th fruit fly in the (*t* + 1)th iterative process. C_1 and C_2 are non-negative constants, denoting learning factor. ω denotes inertia weight. δ, η denotes random values uniformly distributed in [0, 1]. C_1, C_2 and ω are defined as

$$\begin{cases} C_1 = C_{1,\max} - \dfrac{t \times \left(C_{1,\max} - C_{1,\min}\right)}{iter} \\ C_2 = C_{2,\max} - \dfrac{t \times \left(C_{2,\max} - C_{2,\min}\right)}{iter} \\ \omega = \omega_{\min} + \dfrac{\left(iter - t\right) \times \left(\omega_{\max} - \omega_{\min}\right)}{iter} \end{cases} \tag{4}$$

where $\omega_{\max}$ and $\omega_{\min}$ are the maximum and minimum values the inertia weight can take, respectively. $C_{1,\max}$, $C_{1,\min}$, $C_{2,\max}$, $C_{2,\min}$ are the maximum and minimum values the learning factor can take, respectively. t and *iter* denote the current evolutional algebra and the total evolutional algebra, respectively.

4.1.3 Distance (Dist) and smell concentration judgement value (Smell)

All of the individuals are one-dimensional values in the background of actual swarm when defining individual fruit fly. Consequently, in each iteration of IFOA4WSC for each fruit fly, the reciprocal of the offset which is the location of the fruit fly in its own set related to the 0th record is used as the distance. After that, the value of smell concentration judgement is calculated by using the relation between *Dist* and *Smell*, and then fitness is calculated. The *Dist* and the *Smell* are defined as

$$Dist_{j,k}^{t+1} = \frac{1}{x_{j,k}^{t+1}} \qquad Smell_{j,k}^{t+1} = \frac{1}{Dist_{j,k}^{t+1}} \tag{5}$$

where $x_{j,k}^{t+1}$ represents the location of the kth fruit fly of the jth fruit fly swarm in the (t+1)th iterative process by equation (3). It is the location where the individual fruit fly is shifted in its set of sub-services. $Dist_{j,k}^{t+1}$ denotes the distance of the kth fruit fly of the jth fruit fly swarm in the (t+1)th iterative process. $Smell_{j,k}^{t+1}$ represents the smell concentration judgement value of the kth fruit fly of the jth fruit fly swarm in the (t+1)th iterative process.

4.1.4 Fitness function

Fitness function calculates the fitness of each SC. The pros and cons of each SC are judged by fitness values of different SC. Combined with the results of QoS after pre-processing using equation (1), the fitness function can be defined as

$$\begin{aligned} fitness &= \sum_{j=1}^{n}\sum_{k=1}^{m} Sw_j W_k Q_{jk} = \sum_{j=1}^{n}\sum_{k=1}^{m} Sw_j W_k Smell_{j,k}^{t} \\ &= W_1 \prod_{j=1}^{n} Sw_j Smell_{j,k}^{t}\left(Ava_j\right) + W_2 \sum_{j=1}^{n} Sw_j Smell_{j,k}^{t}\left(RT_j\right) \\ &\quad + W_3 \sum_{j=1}^{n} Sw_j Smell_{j,k}^{t}\left(Cost_j\right) + W_4 \prod_{j=1}^{n} Sw_j Smell_{j,k}^{t}\left(Rep_j\right) \end{aligned} \tag{6}$$

where $Smell_{j,k}^{t}$ represents the *k*th component index of individual fruit fly of the *j*th swarm in the *t*th iterative process. Sw_j, Q_{jk} and W_k represent the weight of the *j*th sub-service in the *SC*, the *k*th component index in the *j*th sub-service and the weight of the *k*th component index, respectively. *n* and *m* denote the number of sub-services and *QoS* indexes in each sub-service, respectively.

4.2 Convergence analysis of fruit fly swarm

First, we assume that the random values in equations (2) and (3) are constant. A detailed analysis of the convergence of fruit fly swarm in IFOA4WSC is as follows.

Analysis 1: simplification of equation

For the convenience of analysis, random values δ, η, θ are substituted with φ_1, φ_2, φ_3, so equations (2) and (3) could be simplified as

$$V^{t+1} = \omega V^t + \varphi_1 \left(P^t - X^t\right) + \varphi_2 \left(P_g^t - X^t\right) \tag{7}$$

$$X^{t+1} = X^t + \varphi_3 V^{t+1} \tag{8}$$

From equation (7), we obtain

$$X^t = \frac{\omega V^t + \varphi_1 P^t + \varphi_2 P_g^t - V^{t+1}}{\varphi_1 + \varphi_2} \tag{9}$$

From equation (8), we can also obtain

$$V^{t+1} = \frac{X^{t+1} - X^t}{\varphi_3} \tag{10}$$

Equations (7)–(10) are combined together and then the following equations are obtained:

$$V^{t+2} = \omega V^{t+1} + \varphi_1 \left(P^{t+1} - X^{t+1}\right) + \varphi_2 \left(P_g^{t+1} - X^{t+1}\right) \tag{11}$$

$$X^{t+2} = X^{t+1} + \varphi_3 V^{t+2} \tag{12}$$

$$X^{t+1} = \frac{\omega V^{t+1} + \varphi_1 P^{t+1} + \varphi_2 P_g^{t+1} - V^{t+2}}{\varphi_1 + \varphi_2} \tag{13}$$

$$V^{t+2} = \frac{X^{t+2} - X^{t+1}}{\varphi_3} \tag{14}$$

Substituting equations (10) and (14) into (11), we can get

$$X^{t+2} - \left(\omega + 1 - \varphi_1\varphi_3 - \varphi_2\varphi_3\right) X^{t+1} + \omega X^t - \varphi_1\varphi_3 P^{t+1} - \varphi_2\varphi_3 P_g^{t+1} = 0 \tag{15}$$

Substituting equations (9) and (13) into (10), we get

$$\begin{aligned} &\left(\varphi_1\varphi_3 + \varphi_2\varphi_3 + 1\right) V^{t+2} - \left(\omega + 1\right) V^{t+1} \\ &+ \omega V^t + \varphi_1 P^t - \varphi_1 P^{t+1} + \varphi_2 P_g^t - \varphi_2 P_g^{t+1} = 0 \end{aligned} \tag{16}$$

Assuming that $P^t = P^{t+1} = Q_1, P_g^t = P_g^{t+1} = Q_2$, equations (15) and (16) together give

$$X^{t+2} - \left(\omega + 1 - \varphi_1\varphi_3 - \varphi_2\varphi_3\right) X^{t+1} + \omega X^t - \varphi_1\varphi_3 Q_1 - \varphi_2\varphi_3 Q_2 = 0 \tag{17}$$

$$\left(\varphi_1\varphi_3 + \varphi_2\varphi_3 + 1\right) V^{t+2} - \left(\omega + 1\right) V^{t+1} + \omega V^t = 0 \tag{18}$$

As mentioned previously, fruit fly swarm fluctuates with the number of iterations. Location and speed are converted into second-order difference equation. Below we will discuss motor process and peculiarity of fruit fly swarm in search space at the micro-level through analysis of equations (17) and (18).

Analysis 2: analysis on convergence of speed

Equation (18) is converted by *Z*-transform and then we can obtain

$$V(z) = \frac{\Psi z^2 V(0) + \Psi z V(1) - z(\omega+1)V(0)}{\Psi z^2 - z(\omega+1) + \omega} \tag{19}$$

where $\Psi = \varphi_1\varphi_3 + \varphi_2\varphi_3 + 1$.

Postulate further that φ_1, φ_2, φ_3 are constants and the characteristic equation of equation (19) is computed as

$$\Psi z^2 - z(\omega+1) + \omega = 0 \tag{20}$$

According to analysis on the stability of discrete system, the necessary and sufficient condition for the stability of system is that all zero points of the characteristic equation are located in a unit circle centred at the origin in plane *z*. Therefore, equation (20) is converted by bilinear transformation. Postulating that $z = \frac{\mu+1}{\mu-1}$ in equation (20) leads to

$$\begin{aligned} &\left(\frac{\mu+1}{\mu-1}\right)^2 \Psi - \frac{\mu+1}{\mu-1}\left(\omega+1\right) + \omega = 0 \\ &\Rightarrow \Psi\mu^2 + \left(2\Psi - 1\right)\mu + \left(\Psi - 2\omega - 1\right) = 0 \end{aligned} \tag{21}$$

According to *Routh* criterion, the necessary and sufficient condition for stability of the second-order linear system is that all coefficients are positive in characteristic equation. The stability condition of differential equation (18) is computed as

$$\begin{cases} \varphi_1\varphi_3 + \varphi_2\varphi_3 + 1 > 0 \\ 2\varphi_1\varphi_3 + 2\varphi_2\varphi_3 + 1 \geq 0 \\ \varphi_1\varphi_3 + \varphi_2\varphi_3 - 2\omega \geq 0 \end{cases} \Rightarrow \begin{cases} \varphi_1\varphi_3 + \varphi_2\varphi_3 \geq -0.5 \\ \varphi_1\varphi_3 + \varphi_2\varphi_3 \geq 2\omega \end{cases} \tag{22}$$

Therefore, when the condition in equation (22) is met, it can be concluded that $V^t = \lim_{z\to 0}\left((z-1)V(z)\right) = 0$ through the final value theorem of *Z*-transform. Namely, when φ_1, φ_2, φ_3 are constants, the speed of IFOA4WSC converges to 0.

Analysis 3: analysis on convergence of location

Similarly, equation (17) is converted by Z and it can be seen that

$$X(z)=\frac{\left(X(0)z^2+X(1)z-\Phi X(0)+\varphi_1\varphi_3Q_1+\varphi_2\varphi_3Q_2\right)(z-1)}{\left(z^2-\Phi z+\omega\right)(z-1)} \tag{23}$$

where $\Phi = \omega + 1 - \varphi_1\varphi_3 - \varphi_2\varphi_3$.

Postulate further that φ_1, φ_2, φ_3 are constants and the characteristic equation of equation (23) is computed as

$$\left(z^2-\Phi z+\omega\right)(z-1)=0 \tag{24}$$

Equation (24) is converted by bilinear transformation. Postulating that $z=\frac{\mu+1}{\mu-1}$ in equation (24) leads to

$$\begin{gathered}\left(\left(\frac{\mu+1}{\mu-1}\right)^2-\left(\omega+1-\varphi_1\varphi_3-\varphi_2\varphi_3\right)\frac{\mu+1}{\mu-1}+\omega\right)\frac{2}{\mu-1}=0\\ \Rightarrow\left(\varphi_1\varphi_3+\varphi_2\varphi_3\right)\mu^2+(2-2\omega)\mu+2\omega+2-\varphi_1\varphi_3-\varphi_2\varphi_3=0\end{gathered} \tag{25}$$

Similarly, the stability condition of location is computed as

$$\begin{cases}\varphi_1\varphi_3+\varphi_2\varphi_3>0\\ 1-\omega\geq 0\\ 2\omega+2\geq\varphi_1\varphi_3+\varphi_2\varphi_3\end{cases} \tag{26}$$

when the condition of equation (26) is met, after applying the final value theorem of Z-transform, we can obtain

$$X^t=\lim_{z\to 0}\left((z-1)X(z)\right)=\frac{\varphi_1\varphi_3Q_1+\varphi_2\varphi_3Q_2}{\varphi_1\varphi_3+\varphi_2\varphi_3}.$$

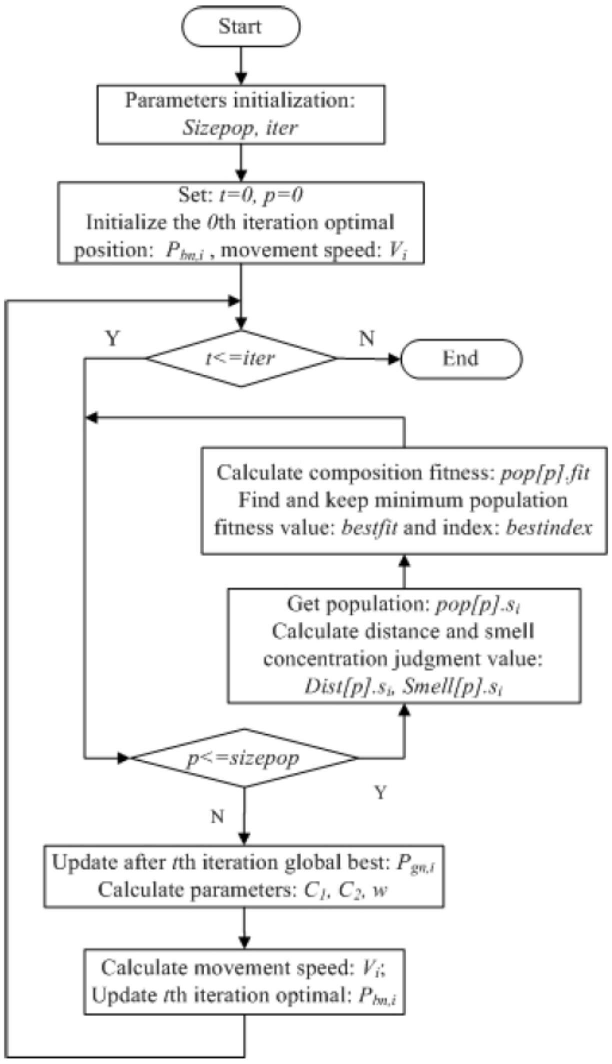
That is to say, the speed of IFOA4WSC converges to $\frac{\varphi_1\varphi_3Q_1+\varphi_2\varphi_3Q_2}{\varphi_1\varphi_3+\varphi_2\varphi_3}$ when φ_1, φ_2, φ_3 are constants.

According to detailed analysis on motor process of fruit fly swarm in IFOA4WSC, the condition equation which parameters select and adjust in IFOA4WSC is obtained. It provides a theoretical basis for the improvement of convergence and efficiency of the IFOA4WSC algorithm.

4.3 Implementation of algorithm

It is necessary to eliminate the incommensurability stemming from the different dimensions and dimensional units given that the established *QoS* index system has different dimensions. Consequently, the experimental data should be pre-processed before the experiment by combining with Definition 4. Then, the pre-processed data are optimised according to IFOA4WSC. This algorithm includes two parts: *QoS* pre-processing algorithm and IFOA4WSC. The main steps are described as follows and their implementation procedures are illustrated in Figure 3.

Figure 3 The implementation procedure of the IFOA4WSC


Start
Parameters initialization: Sizepop, iter
Set: t=0, p=0
Initialize the 0th iteration optimal position: $P_{bm,i}$, movement speed: V_i
t<=iter
Y
N
End
Calculate composition fitness: pop[p].fit
Find and keep minimum population fitness value: bestfit and index: bestindex
Get population: pop[p].s_i
Calculate distance and smell concentration judgment value:
Dist[p].s_i, Smell[p].s_i
p<=sizepop
Y
N
Update after tth iteration global best: $P_{gn,i}$
Calculate parameters: C_1, C_2, w
Calculate movement speed: V_i;
Update tth iteration optimal: $P_{bm,i}$

Algorithm 1 Procedure of the *QoS* pre-processing algorithm

Require: The original index information of sub-services

Ensure: The normalised index information of sub-services

```
1: for each $S_i \in S$, $i \in [1,n]$ do
2:   for each $s_i^j \in S_i, j \in [1, m_i]$ do
3:     $Get(MAX(Ava), MAX(Rep))$;
4:     $Get(MAX(Cost), MAX(RT))$;
5:     $Get(MIN(Ava), MIN(Rep))$;
```

```
6:     $Get(MIN(Cost), MIN(RT))$;
7:   end for
8: end for
9: for each $s_i^j \in S_i, j \in [1, m_i]$ do
10:  $s_i^j(Ava) = \frac{MAX(Ava) - s_i^j(Ava)}{MAX(Ava) - MIN(Ava)}$;
11:  $s_i^j(Rep) = \frac{MAX(Rep) - s_i^j(Rep)}{MAX(Rep) - MIN(Rep)}$;
12:  $s_i^j(Cost) = \frac{s_i^j(Cost) - MIN(Cost)}{MAX(Cost) - MIN(Cost)}$;
13:  $s_i^j(RT) = \frac{s_i^j(RT) - MIN(RT)}{MAX(RT) - MIN(RT)}$;
14: end for
```

where *Get*(*MAX*(*Ava*), *MAX*(*Rep*)) represents the maximum value of indexes *Ava*,*Rep* in the obtained sub-service set S_i, *Get*(*MAX*(*Cost*), *MAX*(*RT*)) represents the maximum value of indexes *Cost*,*RT* in the obtained sub-service set S_i. The minimum value is obtained similarly. $s_i^j(Ava) = \frac{MAX(Ava) - s_i^j(Ava)}{MAX(Ava) - MIN(Ava)}$ denotes calculating index value of attribute *Ava* of the *j*th individual fruit fly in sub-service set S_i. The others are calculated similarly.

Algorithm 2 Procedure of the IFOA4WSC Algorithm

```
Require:
  The normalised index information of set of sub-services
Ensure:
  the record of optimal service composition
1: $init(P_{bn,i}, V_i), i \in [1,n]$
2: for each $t \leq iter$ do
3:   for each $p \leq sizepop$ do
4:     for each $i \leq n$ do
5:       $pop[p] \cdot s_i = P_{bn,i} + V_i \times \theta\theta \in [-1,1]$;
6:       $Dist[p] \cdot s_i = \frac{1}{pop[p] \cdot s_i}$;
```

```
7:       $Smell[p]\cdot s_i = \frac{1}{Dist[p]\cdot s_i}$;
8:     end for
9:     $pop[p]{\cdot}fit = fitness\_function(Smell[p])$;
10:    if $bestfit \geq pop[p]{\cdot}fit$ then
11:      $bestfit = pop[p]{\cdot}fit$;
12:      $bestindex = p$;
13:    end for
14:  end for
15:  if $P_{gn} \geq pop[bestindex]{\cdot}fit$ then
16:    $P_{gn} = pop[bestindex]{\cdot}fit$;
17:  end if
18:  $C_1 = C_{1,\max} - \frac{t \times (C_{1,\max} - C_{1,\min})}{iter}$;
19:  $C_2 = C_{2,\max} - \frac{t \times (C_{2,\max} - C_{2,\min})}{iter}$;
20:  $\omega = \omega_{\min} + \frac{(\omega_{\max} - \omega_{\min})(iter - t)}{iter}$;
21:  for each $i \leq n$ do
22:    $V_i = \omega V_i + C_1 \delta(pop[bestindex]{\cdot}s_i - P_{bn,i}) + C_2 \eta\,(P_{gn,i} - P_{bn,i})$;
23:    $P_{bn,i} = pop[bestindex]{\cdot}s_i$;
24:  end for
25: end for
```

In Algorithm 2, the number of candidate sub-services adopted is assumed to be n and has four-dimensional indexes, including availability (*Ava*), reputation (*Rep*), response time (*RT*) and cost (*Cost*). $P_{bn,i}$ and $P_{gn,i}$ represent the local extremum and the global extremum in the tth iteration, respectively. i is the ith component of extremum; *pop* is the fruit fly swarm, *sizepop* is the swarm size and *fitness_function* is the fitness function.

5 Comparison of algorithms

To examine how the proposed IFOA4WSC is situated compared to other heuristic optimisation algorithms, an experimental comparison with other algorithms is undertaken. PSO and GA have continued to be used as the standard benchmark algorithms to evaluate the performance of various new optimisation algorithms (Yilmaz and Karagoz, 2014; Wang et al., 2013; Vasumathi and Moorthi, 2012; Tao et al., 2008; Canfora et al., 2005;

Tao et al., 2012). Therefore, in this work, PSO, GA and FOA are selected as the benchmark algorithms against what we evaluate the performance of the proposed IFOA4WSC algorithm. The fundamental characteristics of the PSO, FOA and GA are described below.

5.1 PSO

PSO algorithm proposed by Eberhart and Kennedy (1995) is a heuristic evolutionary computing method and derives from the simplified simulation of the social group's intelligent behaviour. The location of each particle represents a solution to optimisation problem in PSO, which is updated according to two extremums. One is *pbest*, historical optimal solution of the particle itself. The other is *gbest*, global optimal solution which has been computed. The location renewal equation of PSO is as follows (Moein Logeswaran, 2014):

$$\begin{aligned} v_i^d(t+1) = \omega(t)v_i^d(t) + C_1 rand_i(t)\left(pbest_i^d - x_i^d(t)\right) \\ + C_2 rand_i(t)\left(gbest^d - x_i^d(t)\right) \end{aligned} \tag{27}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{28}$$

where $pbest_i = (pbest_i^1, pbest_i^2, \cdots, pbest_i^n)$ and $gbest = (gbest^1, gbest^2, \cdots, gbest^n)$ represent historical extremum of individual particle *i* and historical extremum of particle swarm in the last *t* evolutionary process. $rand_i(t)$ denotes the random value in the *t*th evolutionary range of [0,1], and C_1 and C_2 denote two learning factors.

PSO and IFOA4WSC, belonging to intelligent optimisation algorithms, are deduced from nature animal's rule of foraging and have high searching efficiency in the search space. They could obtain the optimal solution by evolving iteratively according to the change of particle location. Nevertheless, the two algorithms have differences:

- *The proposed backgrounds differ*. IFOA4WSC is a new global optimisation method deduced from foraging behaviour of fruit fly. PSO, an optimisation algorithm deduced from birds' prey, is an evolutionary algorithm based on birds' behaviour.
- *The optimising ways differ*. In IFOA4WSC, a fruit fly finds a better position; then the entire fruit fly swarm move to the position. But in PSO, each individual changes its location dynamically through each individual optimisation and global optimisation.
- *The moving equations differ*. IFOA4WSC uses fruit fly swarm location and movement speed to produce individual fruit fly in a certain range, while PSO shifts speed and location according to equations (27) and (28).
- *The biological characteristics differ*. IFOA4WSC calculates the smell value by the distance between individual and origin of coordinate, and then use the smell value to optimise. However, PSO does not take distance into account.

5.2 FOA

FOA is a new approach for finding global optimisation based on the swarm behaviour of the fruit fly. The fruit fly itself is superior to other species in sensing and perception, especially in osphresis and vision. The FOA is proposed based on the fruit fly and its foraging process (Pan, 2012). Algorithm 3 shows the FOA algorithm.

Algorithm 3 Procedure of the FOA algorithm

```
1: init(x_axis, y_axis);
2: for each $t \leq iter$ do
3:   Get(x,y);
4:   Calculate(Dist,Smell,Fitness);
5:   [bestFitness  bestindex] = min(Fitness);
6:   Set(x_axis, y_axis);
7: end for
```

Although IFOA4WSC is improved based on FOA, it is an attempt to apply FOA to web SC and make plenty of amelioration. The specific differences between FOA and IFOA4WSC are listed below.

- *The adopted models differ*. FOA is an intelligent algorithm based on the foraging behaviour of the fruit fly swarm, while IFOA4WSC is an improved intelligent optimisation algorithm based on FOA according to the optimisation problem of web SC.
- *The moving equations for individual fruit fly differ*. FOA calculates the new location by $x\_axis + V^*\varepsilon, \varepsilon \in [-1,1]$, while IFOA4WSC adopts the more accurate elitist strategy and calculates new individuals by equations (2)–(4).
- *The optimising ways differ*. IFOA4WSC considers the effects that evolutional contemporary optimal particle and optimal particle generated impose on the whole optimisation. However, FOA does not take it into account.
- *The moving step lengths differ*. FOA adopts fixed moving step length V and produces new location by $V^*\varepsilon$, ε is the random value in $[-1, 1]$. IFOA4WSC adopts dynamic adaptation to shift V, and then produces new position by using the altered $V * \varepsilon$.
- *The definitions of distance differ*. FOA adopts the two-dimensional distance equation based on rectangular plane coordinate system, while IFOA4WSC regards the position offset as distance based on one-dimensional coordinate system.

5.3 GA algorithm

Genetic algorithm is a computational model which simulates the natural selection in Darwin's theory of evolution and the biological evolution of genetic mechanism. It is a method of searching the optimal solution in the process of simulating natural evolution. In GA, the initial population is constituted by individual coding. And then, certain operations are applied through the fitness function proposed, such as selection, crossover and mutation, thereby realising the process of the survival of the fittest. This experiment adopts classical GA algorithm shown in Algorithm 4.

Algorithm 4 Procedure of the GA algorithm

```
1: $init(pop, p_c, p_m)$;
2: for each $t \leq iter$ do
3:    $Calculate(Fitness)$;
4:    $[bestFitness\ \ bestindex] = \min(Fitness)$;
5:    $select()$;
6:    $crossover(p_c)$;
7:    $mutation(p_m)$;
8: end for
```

where *select*() denotes selecting operation, preserving individual by roulette. *crossover*(p_c) denotes interlace operation, adopting multiple points crossover. p_c represents the crossover rate. *mutation*(p_m) denotes mutation operation, adopting single point mutation. p_m represents the mutation rate.

6 Experimental results

6.1 Experimental data and experimental environment

The experiment is carried out in Intel Dual Core i3 2.1 GHz CPU, 4G RAM, Windows 7 OS and Visual Studio 2010. All the Sw_j in equation (6) are set to be 1, i.e. each sub-service accounts for the same percentage. The number of sub tasks is 5. W_k is taken from the set {0.2, 0.3, 0.2, 0.3}, corresponding to the four *QoS* indexes in Definition 2. The parameters in equations (2)–(4) and GA experiment are shown in Table 2.

Table 2 Experimental parameter settings

Parameter	*Significance*	*Value*
$\omega_{\max}$	Maximum of inertia weight	0.3
$\omega_{\min}$	Minimum of inertia weight	0.1
$C_{1,\max}$	Maximum of local learning factor	2
$C_{1,\min}$	Minimum of local learning factor	0.5
$C_{2,\max}$	Maximum of global learning factor	2
$C_{2,\min}$	Minimum of global learning factor	0.5
p_m	Mutation rate	0.15
p_c	Crossover rate	0.8

To make the experiment more representative, the experimental data set consists of two parts: the common data set (QWS) and the random data set (RWS). Data set processing is accessed with the text document as carrier. The QWS data set is provided by Zeng et al. (2003, 2004). This experiment divides the QWS data set into five sets of candidate services in accordance with the congruence class relation of module 5. In order to adapt to the actual situation, the random generated RWS data set is constituted by the data which follow $N(0,1)$ normal distribution. The specific generation method is as follows. Postulate that $u,v(u,v \in [-1,1])$ are even-distributed random variables and independent of each other. Supposing that $s = u^2 + v^2$, two random values z_0,z_1 can be generated. The generation equations are

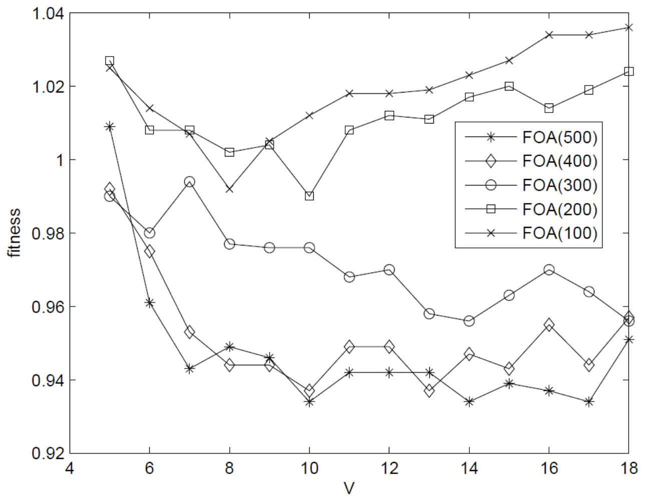
$$\begin{cases} z_0 = u \times \sqrt{\dfrac{-2 \times \ln s}{s}} \\ z_1 = v \times \sqrt{\dfrac{-2 \times \ln s}{s}} \end{cases} \tag{29}$$

To guarantee $\ln s < 0$, it is required that $s < 1 \wedge s \neq 0$. If the condition is met, z_0,z_1 can be generated and follow $N(0,1)$. In the process of generating data, to ensure that the gained data are not less than 0, experimental data can be generated by using normal distribution peculiarity of $P(\mu - 3\sigma < X \leq \mu + 3\sigma) = 99.7\%$ and giving z_0,z_1 a translation 3 units to the right, respectively. In order to possess higher reliability, the multi-peak experimental data are generated and the above-mentioned process is executed using each 100 groups of data. Finally, all the data are normalised unifiedly.

6.2 *Analysis on the influence of moving step length V in FOA*

In FOA, moving equation contains the fixed moving step length V. To test the influence of moving step length V on the algorithm, in the case of different number of candidate services and different value of moving step length V, the statistical result of average fitness value in 50 runs is shown in Figure 4.

Figure 4 Analysis on the influence of the moving step length V under different number of candidate services



It can be seen from Figure 4 that, with the same scale of services, the result of optimisation varies with different values of *V*. With the same value of moving step length *V*, the optimising gap of different service scales is wide, i.e. the optimal effects are out-sync. In IFOA4WSC, moving step length *V* is not fixed and can make adaptive change according to optimising location. Consequently, it is essential for IFOA4WSC to improve the moving step length.

6.3 Comparison of experimental results

In this section, we evaluate the effectiveness, efficiency and stability among PSO, GA, FOA and our algorithm. We use the same data sets for all algorithms. The experimental results of all algorithms are shown in Tables 3 and 4 according to the parameter settings in Table 2. We can see that IFOA4WSC has better optimisation results compared with PSO, GA and FOA. Its time cost is much smaller than PSO and GA, even slightly higher than FOA. Furthermore, our algorithm achieves better stability compared to that of PSO, GA and FOA. This section analyses the performance of the algorithm from three aspects including effectiveness, efficiency and stability, where RMSE denotes root mean square error.

6.3.1 Effectiveness

To analyse the optimisation effectiveness among PSO, FOA, GA and IFOA4WSC, the experiment is implemented 50 times over a population size (120) with 500 iterations. The average solution of optimisation for QWS data set is shown in Figure 5 and the average solution of optimisation for RWS data set is shown in Figure 6.

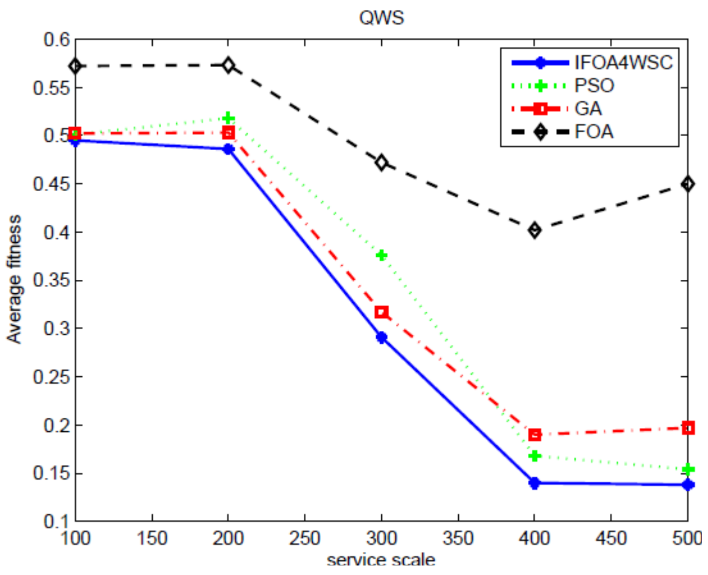
Table 3 Experimental results of different algorithms for fitness value and RMSE

Service scale			*100*	*200*	*300*	*400*	*500*
QWS	IFOA4WSC	Fitness	0.495	0.486	0.291	0.14	0.138
		RMSE	0.005	0.006	0.0092	0.0131	0.0067
	PSO	Fitness	0.501	0.518	0.376	0.168	0.154
		RMSE	0.0134	0.028	0.0452	0.0344	0.0365
	GA	Fitness	0.502	0.503	0.317	0.19	0.197
		RMSE	0.138	0.0164	0.0362	0.0756	0.055
	FOA	Fitness	0.572	0.573	0.472	0.402	0.45
		RMSE	0.0231	0.0284	0.0714	0.0759	0.0908
RWS	IFOA4WSC	Fitness	0.481	0.469	0.467	0.464	0.471
		RMSE	0.0034	0.0153	0.0152	0.0372	0.0409
	PSO	Fitness	0.507	0.504	0.524	0.552	0.553
		RMSE	0.0353	0.0475	0.0622	0.0603	0.0634
	GA	Fitness	0.487	0.492	0.505	0.487	0.494
		RMSE	0.0125	0.0391	0.0363	0.0404	0.0383
	FOA	Fitness	0.612	0.673	0.681	0.677	0.677
		RMSE	0.0477	0.0558	0.0514	0.0607	0.0455

Table 4 Experimental results of different algorithms for time consumption (ms)

		QWS				*RWS*			
		PSO	*IFOA4WSC*	*GA*	*FOA*	*PSO*	*IFOA4WSC*	*GA*	*FOA*
Population scale	100	73.3	51.12	168.02	54.02	76.04	50.82	178.44	50.16
	200	141.74	98.46	535.76	96.6	144.8	101.28	567.46	95.58
	300	213.34	146.72	1145.28	138.92	240.54	148.32	1152.42	147.62
	400	280.56	194.8	1898.64	185.98	281.64	198.12	2026.76	192.98
	500	353.14	245.12	2886.04	231.34	350.48	246.66	2976.72	238.76
Iteration	200	29.64	20.44	67.76	19.24	30.68	18.2	71.6	18.72
	400	57.12	40.3	132.32	38.46	55.92	40.6	142.24	40
	600	86.72	60.72	200.22	57.12	88.54	62	213.2	61.36
	800	112	80.4	263.04	75.58	115.28	82.2	284.54	80.7
	1000	140.58	99.4	329.88	95.12	143.52	101.02	352.9	100.04

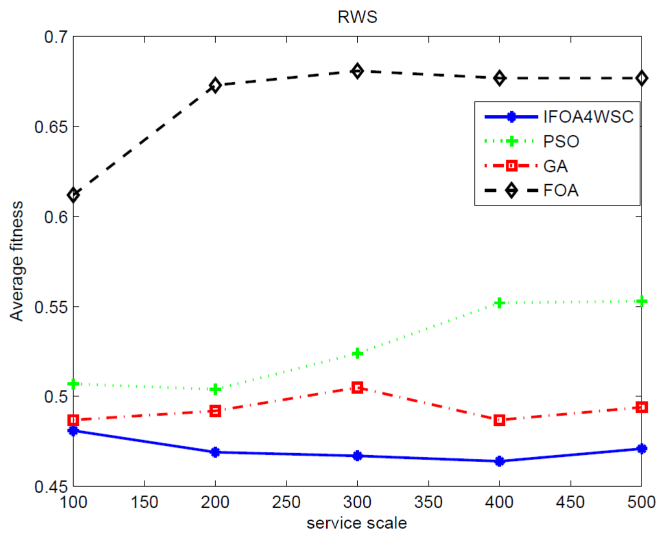
Figure 5 Average optimisation solution of the four algorithms in QWS



Apparently, as shown in Figure 5, in the QWS data set, the optimisation solution of IFOA4WSC is superior to that of PSO and GA. Furthermore, with the expansion of the service scale, optimisation capacity strengthens constantly. Therefore, IFOA4WSC has a good searching capacity.

Figure 6 also clearly shows that, in the RWS data set, random data are independent of each other and lack dependency as real data. Therefore, PSO lacks availability owing to its dependence on the two polar values, i.e. *pbest* and *gbest*. Consequently, the optimisation solution of PSO is slightly worse than that of GA. Nevertheless, the optimisation solution of IFOA4WSC is superior to that of GA, implying that IFOA4WSC has stronger generalisation and better optimisation effectiveness.

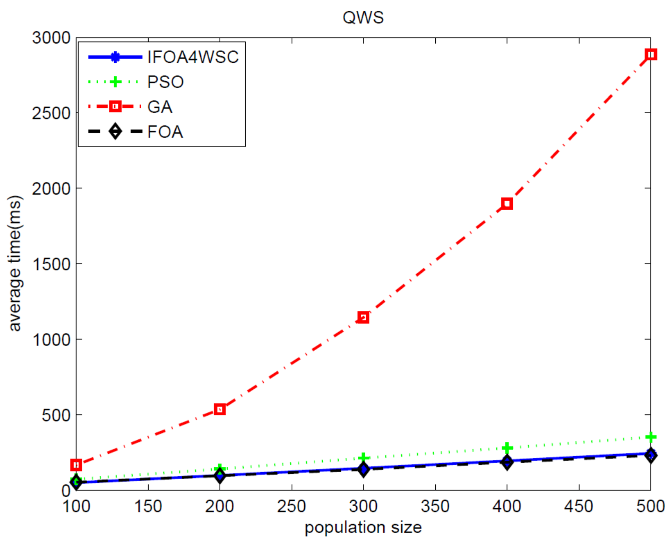
Figure 6 Average optimisation solution of the four algorithms in RWS



6.3.2 Efficiency

For further analysis of time overhead among IFOA4WSC, FOA, PSO and GA, we make two comparisons: the average runtime of optimising 50 times (Figures 7 and 9) with different population sizes and the same number of iterations (500), and the average runtime of optimising 50 times (Figures 8 and 10) with the same population size (100) and different numbers of iterations.

Figure 7 Average runtime of the four algorithms with different population sizes and the same number of iterations in QWS



From Figures 7 and 8, in the QWS data set, it is obvious that, in the aspect of overall time overhead, IFOA4WSC is superior to PSO and GA. Moreover, with the increase of population size and the number of iterations, the disparity between IFOA4WSC and PSO, GA gets wider gradually. Above all, IFOA4WSC shows the ascendancy that PSO and GA cannot replace when the population size and the number of iterations increase. Consequently, IFOA4WSC has strong searching efficiency.

Figure 8 Average runtime of the four algorithms with the same population sizes and different numbers of iterations in QWS

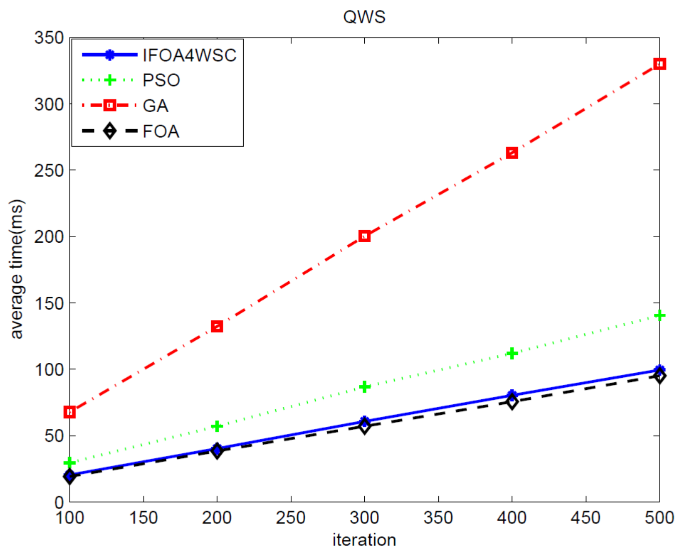
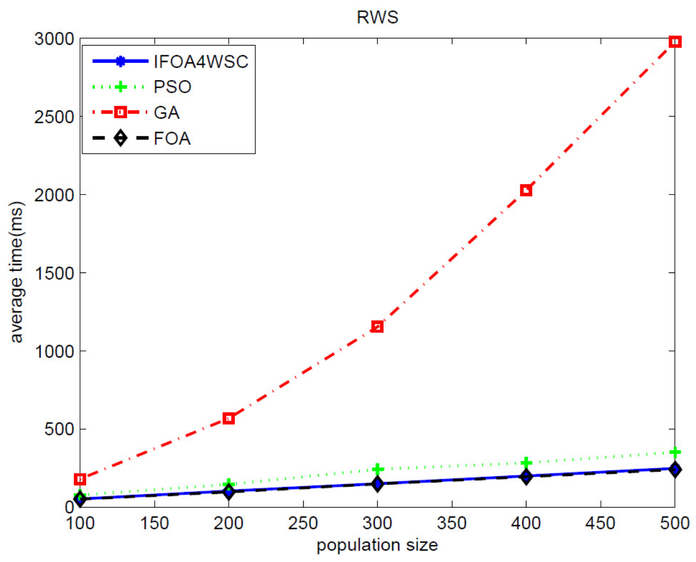
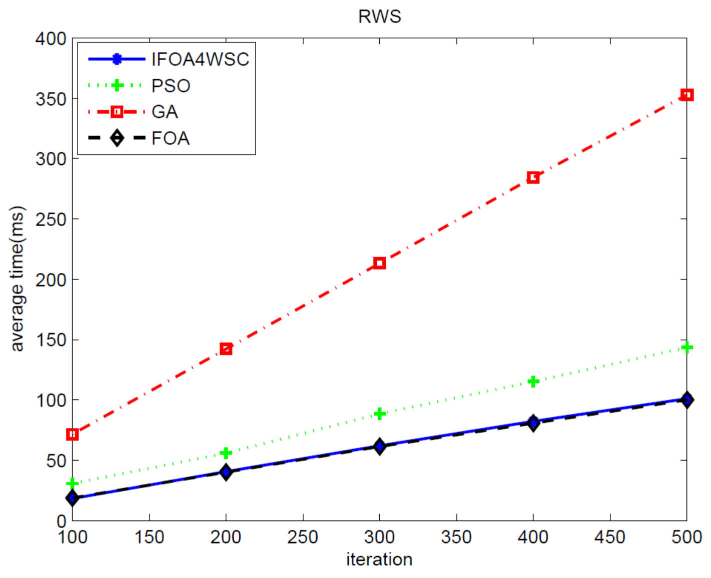


Figure 9 Average runtime of the four algorithms with different population sizes and the same number of iterations in RWS



Figures 9 and 10 show that, in the RWS data set, it is significantly better than that of GA and PSO, while time consumption of IFOA4WSC is similar to that of FOA. Especially when the fruit fly swarm size and the number of iteration is larger and larger, IFOA4WSC shows the ascendancy that PSO and GA cannot replace. For time consumption, FOA is slightly better than that of IFOA4WSC. Nevertheless, from Figure 6, it is obvious that the result of FOA being applied to web SC is far worse than IFOA4WSC. Consequently, IFOA4WSC has high efficiency in time consumption.

Figure 10 Average runtime of the four algorithms with the same population sizes and different numbers of iterations in RWS



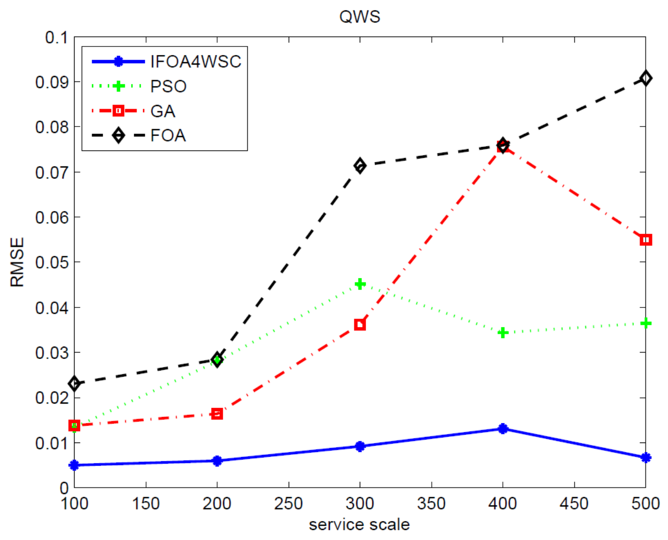
6.3.3 Stability

To compare the stability of different algorithms, dispersion degree of the results is recorded in 50 runs and RMSE (Silic et al., 2014) is used for comparison, whose computational formula is

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n}} \tag{30}$$

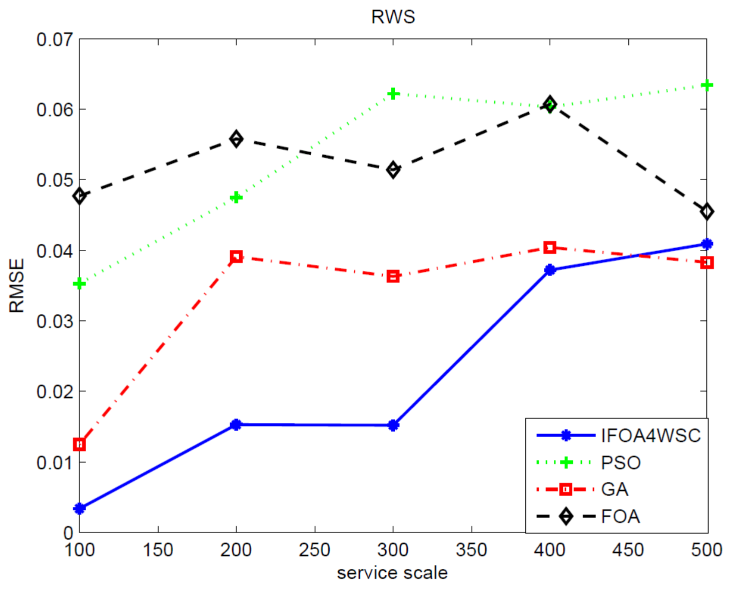
where $\bar{X}$ and X_i denote the average of n times repeated experimental result and the ith optimisation result separately. The comparison among experimental results of IFOA4WSC, FOA, PSO and GA is shown in Figure 11 for the QWS data set and Figure 12 for the RWS data set.

Figure 11 RMSE of the four algorithms in QWS



It can be seen from Figure 11 that, with the different number of services, dispersion degree of IFOA4WSC is lower than that of PSO, FOA and GA obviously. Furthermore, with the increase of services, dispersion degree of IFOA4WSC tends towards stability. It shows that IFOA4WSC has great stability.

Figure 12 RMSE of the four algorithms in RWS



From Figure 12, we can see that, compared with GA, PSO and FOA, the RMSE of IFOA4WSC is stably less than other classical GA and PSO with the increase of service size. Therefore, it can guarantee that optimal composition is obtained, which indicates the times of finding optimal solution is significantly more than that of PSO and GA. Consequently, our algorithm has greater stability.

From what has been discussed above, IFOA4WSC is an efficient, fast and stable intelligent optimisation algorithm and provides a strong support in theory and practice for large-scale web SC.

7 Conclusions

An improved optimisation algorithm for web service composition, IFOA4WSC, is put forward in this paper. We propose the formula of variable-velocity and variable-position based on the analysis on the effect of moving step length V in FOA. We also analyse convergence condition of fruit fly swarm in IFOA4WSC postulating that random value is constant, which provides theoretical foundation for improving the performance of IFOA4WSC. A host of simulation experiments on real-world data set and random data set show that IFOA4WSC has a better global searching capacity, higher searching efficiency and stronger stability.

We use the design of weighted fitness function which converts the core algorithm into single objective optimisation. As a future work, multiple objective optimisation can be realised using IFOA4WSC. In addition, the cloud-powered 'everything as a service' (XaaS) megatrend is exploding and the number of web services proliferating. Therefore, parallel IFOA4WSC can be used to improve the accuracy and convergence speed. Our future work will focus on these aspects.

Acknowledgements

This work was supported by the National Key Technology R&D Program (No. 2015BAK24B01), the National Natural Science Foundation of China (Nos. 61402006 and 61202227), the General Research for Humanities and Social Sciences Project of Chinese Ministry of Education (No. 15YJAZH112), the Natural Science Foundation of Anhui Province of China (No. 1408085MF132) and the 211 Project of Anhui University (No. 02303301).

References

Al-Helal, H. and Gamble, R. (2014) 'Introducing replaceability into web service composition', *IEEE Transaction on Services Computing*, Vol. 7, No. 2, pp.198–209.

Alrifai, M., Skoutas, D. and Risse, T. (2010) 'Selecting skyline services for QoS-based web service composition', *Proceedings of the 19th International Conference on World Wide Web*, 26–30 April, Raleigh, NC, pp.26–30.

Canfora, G., Penta, M.D., Esposito, R. and Villlani, M.R. (2005) 'A approach for QoS-aware service composition based on genetic algorithms', *Proceedings of the 7th International Conference on Genetic and Evolutionary Computation*, 25–29 June, Washington, DC, pp.1069–1075.

Chen, M. and Yan, Y. (2014) 'QoS-aware service composition over graph plan through graph reachability', *Proceedings of the IEEE International Conference on Services Computing*, 27 June–2 July, Anchorage, AK, pp.544–551.

Eberhart, R.C. and Kennedy, J. (1995) 'A new optimizer using particle swarm theory', *Proceedings of the 7th International Symposium on Micro Machine and Human Science*, 4–6 October, Nagoya, pp.39–43.

El Hadad, J., Manouvrier, M. and Rukoz, M. (2010) 'TQoS: transactional and QoS-aware selection algorithm for automatic web service composition', *IEEE Transactions on Services Computing*, Vol. 3, No. 1, pp.73–85.

Fang, W., Sun, J., Wu, X.J. and Palade, V. (2015) 'Adaptive web QoS controller based on online system identification using quantum-behaved particle swarm optimization', *Soft Computing*, Vol. 19, pp.1715–1725.

Gabrel, V., Manouvrier, M. and Murat, C. (2014) 'Optimal and automatic transactional web service composition with dependency graph and 0-1 linear programming', *Proceedings of the 12th International Conference on Service-Oriented Computing*, 3–6 November, Paris, France, *LNCS*, Vol. 8831, pp 108–122.

Geem, Z., Kim, J. and Loganathan, G. (2001) 'A new heuristic optimization algorithm: harmony search', *Simulations*, Vol. 76, No. 2, pp.60–68.

Giedrimas, V. and Sakalauskas, L. (2012) 'Simulated annealing and variable neighborhood search algorithm for automated software services composition', *Proceedings of the 35th International Convention on Web Information and Communication Technology, Electronics and Microelectronics*, 21–25 May, Opatija, Croatia, pp.395–399.

He, Q., Yan, J., Jin, H. and Yang, Y. (2014) 'Quality-aware service selection for service-based systems based on iterative multi-attribute combinatorial auction', *IEEE Transactions on Software Engineering*, Vol. 40, No. 2, pp.192–215.

Hossain, M.S., Hassan, M.M., Qurishi, M.A. and Alghamdi, A. (2012) 'Resource Allocation for Service Composition in Cloudbased Video Surveillance Platform', *Proceedings of the IEEE International Conference on Multimedia and Expo Workshops*, 9–13 July, Melbourne, VIC, pp.408–412.

Li, H., Guo, S., Li, C. and Sun, J. (2013) 'A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm', *Knowledge-Based Systems*, Vol. 37, pp.378–387.

Lin, S.Y., Lin, G.T., Chao, K.M. and Lo, C.C. (2012) 'A cost-effective planning graph approach for large-scale web service composition', *Mathematical Problems in Engineering*, Vol. 2012, pp.1–22.

Liu, Y., Ngu, A.H. and Zeng, L.Z. (2004) 'QoS computation and policing in dynamic web service selection', *Proceedings of the 13th International Conference on World Wide Web*, 17–22 May, New York City, NY, pp.66–73.

Liu, H., Zhong, F., Ouyang, B. and Wu, J. (2010) 'An approach for QoS-aware web service composition based on improved genetic algorithm', *Proceedings of the International Conference on Web Information Systems and Mining*, Vol. 1, 23–24 October, Sanya, pp.123–128.

Liu, R., Wang, Z. and Xu, X. (2014) 'Parameter tuning for ABC-based service composition with end-to-end QoS constraints', *IEEE International Conference on Web Services*, 27 June–2 July, Anchorage, AK, pp.590–597.

Moein, S. and Logeswaran, R. (2014) 'KGMO: a swarm optimization algorithm based on the kinetic energy of gas molecules', *Information Sciences*, Vol. 275, pp.127–144.

Niewiadomski, A., Penczek, W. and Skaruz, J. (2014) 'Genetic algorithm to the power of SMT: a hybrid approach to web service composition problem', *Proceedings of the 6th International Conference on Advanced Service Computing*, 25–29 May, Venice, Italy, pp.44–48.

Pan, W.T. (2012) 'A new fruit fly optimization algorithm: taking the financial distress model as an example', *Knowledge-Based Systems*, Vol. 26, No. 2, pp.69–74.

Rao, J. and Su, X. (2004) ‘A survey of automated web service composition methods’, *Proceedings of the 1st International Workshop on Semantic Web Services and Web Process Composition*, 6 July, San Diego, CA, pp.43–54.

Rostami, N.H., Kheirkhah, E. and Jalali, M. (2014) ‘An optimized semantic web service composition method based on clustering and ant colony algorithm’, *International Journal of Web and Semantic Technology*, Vol. 5, No. 1, pp.10–17.

Shan, D., Cao, G.H. and Dong, H.J. (2013) ‘LGMS-FOA: an improved fruit fly optimization algorithm for solving optimization problems’, *Mathematical Problems in Engineering*, Vol. 2013, pp.1–9.

Silic, M., Delac, G., Krka, I. and Srbljic, S. (2014) ‘Scalable and accurate prediction of availability of atomic web services’, *IEEE Transaction on Services Computing*, Vol. 7, No. 2, pp.252–264.

Tao, F., Zhao, D., Hu, Y.F. and Zhou, Z. (2008) ‘Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system’, *IEEE Transactions on Industrial Informatics*, Vol.4, No. 4, pp.315–327.

Tao, F., Qiao, K., Zhang, L., Li, Z. and Nee, A.Y.C. (2012) ‘GA-BHTR: an improved genetic algorithm for partner selection in virtual manufacturing’, *International Journal of Production Research*, Vol. 50, No. 8, pp.2079–2100.

Uc-Cetina, V., Moo-Mena, F. and Hernandez-Ucan, R. (2014) ‘Composition of web services using Markov decision processes and dynamic programming’, *The Scientific World Journal*, Vol. 2015, Article ID 545308.

Vasumathi, B. and Moorthi, S. (2012) ‘Implementation of hybrid ANNPSO algorithm on FPGA for harmonic estimation’, *Engineering Applications of Artificial Intelligence*, Vol. 25, No. 3, pp.476–483.

Wang, P., Chao, K.M. and Lo, C.C. (2010) ‘On optimal decision for QoS-aware composite service selection’, *Expert Systems with Applications*, Vol. 37, No. 1, pp.440–49.

Wang, L., Shen, J., Luo, J. and Dong, F. (2013) ‘An improved genetic algorithm for cost-effective data-intensive service composition’, *Proceedings of the 9th International Conference on Semantics, Knowledge and Grids*, 3–4 October, Beijing, China, pp.105–112.

Wang, S.G., Zhu, X.L. and Yang, F.C. (2014) ‘Efficient QoS management for QoS-aware web service composition’, *International Journal of Web and Grid Services*, Vol. 10, No. 1, pp.1–23.

Yilmaz, A.E. and Karagoz, P. (2014) ‘Improved genetic algorithm based approach for QoS aware web service composition’, *Proceedings of the IEEE International Conference on Services Computing*, 27 June–2 July, Anchorage, AK, pp.463–470.

Yu, Q., Chen, L. and Li, B. (2015) ‘Ant colony optimization applied to web service compositions in cloud computing ’, *Computers & Electrical Engineering*, Vol. 41, pp.18–27.

Yuan, X.F., Dai, X.S., Zhao, J.Y. and He, Q. (2014) ‘On a novel multi-swarm fruit fly optimization algorithm and its application’, *Applied Mathematics and Computation*, Vol. 233, pp.260–271.

Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J. and Sheing, Q.Z. (2003) ‘Quality driven web services composition’, *Proceedings of the 12th International Conference on World Wide Web*, 20–24 May, Budapest, Hungary, pp.411–421.

Zeng, L., Benatallah, B., Ngu, A.H., Dumas, M., Kalagnanam, J. and Chang, H. (2004) ‘QoS-aware middleware for web services composition’, *IEEE Transactions on Software Engineering*, Vol. 30, No. 5, pp.311–327.

Zhang, T.G. (2014) ‘QoS-aware web service selection based on particle swarm optimization’, *Journal of Networks*, Vol. 9, No. 3, pp.565–570.