Future Generation Computer Systems 90 (2019) 273–290

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

QoS-aware cloud service composition using eagle strategy

Siva Kumar Gavvala [a], Chandrashekar Jatoth [b], G.R. Gangadharan [c,*], Rajkumar Buyya [d]

[a] University of Hyderabad, Hyderabad, India
[b] Koneru Lakshmaiah Education Foundation, Hyderabad, India
[c] National Institute of Technology, Tiruchirappalli, India
[d] Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne, Australia

ARTICLE INFO

Article history:
Received 14 September 2017
Received in revised form 1 June 2018
Accepted 28 July 2018
Available online 10 August 2018

Keywords:
Quality of Service (QoS)
Cloud services
Service composition
Metaheuristic algorithm
Eagle strategy

ABSTRACT

In recent years, several cloud services have proliferated that conspicuously result in providing similar services having same functionality by multiple service providers, but varying in Quality of Service (QoS) properties. Thus, providing a cloud service composition with optimal QoS values that satisfy the requirements of an user becomes complex and challenging in a cloud environment. Several metaheuristics proposed in solving this problem. However, many of them fail to maintain a suitable balance between exploration and exploitation. We propose a novel Eagle Strategy with Whale Optimization Algorithm (ESWOA) that ensures the proper balance between exploration and exploitation.

1. Introduction

Cloud computing is a way of delivering IT enabled capacities to the consumers in the form of 'services' with elasticity and scalability, where consumers can make use of resources, platform, or software on-demand without having to possess and manage the underlying complexity of the technology [1,2]. Nowadays, several cloud services with comparable functionality are available to consumers at different prices and performance levels (together refereed as Quality of Service (QoS) parameters). Consumers often find it difficult to select an optimal cloud service that satisfies their requirements, as a single service cannot make them complacent [3], that result in rejecting the service out rightly. Due to this, it becomes imperative to perform composition of different available services that bridge together to build a composite service that satisfies the requirements of the consumers.

As there are multiple concrete services for each abstract service (task) that provides same functionalities, the decision needs to be taken on which concrete service needs to be selected for the respective abstract service. Selection of a concrete service needs to be done on the basis of Quality of Service (QoS) parameters because these services exhibit same functional properties.

As we wade through the service composition process, we encounter several potential solutions that would take exponential time for processing all solutions. For example, if there are m abstract services and n candidate services, the number of possible service composition would be n^m [4]. Since it is an NP-Hard problem, we need to find an amicable solution i.e., a near optimal solution that satisfies both the functional and non-functional attributes of the service, that turned out to be an optimization problem.

1.1. Metaheuristics

Near-optimal solutions can be found through highly effective algorithms, often called Metaheuristics [5]. These search methods are highly recommended for getting the good solutions, that are optimal and may be sub-optimal in few cases, in polynomial time instead of exponential time which happens when we solve these problems using conventional methods.

Most of the Metaheuristic algorithms are population based and inspired by the natural phenomena. These algorithms capture the intelligent behavior of the species and try to integrate into the domain specific problems that are related to scheduling, planning, finance and engineering design.

1.2. Using metaheuristics for service composition

All the nature inspired metaheuristics algorithms strive to attain the global optimal solution by inspecting the most favorable locations in their domain search space, based on the natural mechanism that the species possess. The process of this inspection varies among various algorithms. Few algorithms may work well for certain problems, while not-so-good for other problems. This is

* Corresponding author.
E-mail addresses: siva.gavvala@gmail.com (S.K. Gavvala), chandrashekar.jatoth@gmail.com (C. Jatoth), ganga@nitt.edu.in (G.R. Gangadharan), rbuyya@unimelb.edu.au (R. Buyya).

https://doi.org/10.1016/j.future.2018.07.062
0167-739X/

due to the randomization or stochastic strategies that are used in solving those NP-Hard problems. As we could expect the presence of multiple local minima or local maxima in our search space, metaheuristic algorithms need to forage and try to explore as many regions as possible to find the global solution.

1.3. Motivation and contribution

Since cloud service composition is an NP-Hard problem, it is essential to use meta-heuristic algorithms to get the near global optimal solutions. Researchers have been using algorithms that are inspired from biological evolution or from nature like using the intelligence behavior of a group (swarm) of animals and achieved considerably good results to these complex problems.

However, though these algorithms work better in certain conditions, there are few intrinsic issues from the existing ones when they fail to maintain the proper balance between two important yet contrasting components exploration and exploitation. A decent degree of diversity has to be preserved in a population of solutions to reflect exploration and exploitation in the progressing population. Issues such as *premature convergence,* occur when an intensive local exploitation is applied if the population has very low diversity, thus losing the possibility of reaching a globally optimal solution. Also, if the diversity of population solutions is too high, it may be good for global exploration but results in *slow convergence rate* [6]. Hence it is critical to maintain the balance between these components to ensure better results [5,7].

In this work, we propose an algorithm that uses Eagle Strategy allowing us to authorize both exploration and exploitation in an effective way to balance the foraging process. Our experimental results show that our method outperforms the standard algorithms like GA (Genetic Algorithm), WOA (Whale Optimization Algorithm), DGABC (Discrete Guided Artificial Bee Colony), HGA (Hybrid Genetic Algorithm), and GRASP (Greedy Randomized Adaptive Search Procedure).

The remainder of this paper is organized as follows. In Section 2, we present the related works on QoS-aware cloud service composition using bio-inspired and meta-heuristic algorithms. Section 3 discusses modeling of service composition using QoS attributes and related concepts. Section 4 presents our proposed Eagle Strategy with Whale Optimization Algorithm (ESWOA) to achieve the balance between exploration and exploitation. Section 5 reports the experimental analysis of our proposed algorithm, followed by concluding remarks in Section 6.

2. Related work

Many approaches have been proposed for QoS-aware cloud service composition problem in cloud and web services. These approaches generally use optimization methods based on bio-inspired and meta-heuristics algorithms.

Canfora et al. [8] introduced Genetic Algorithm (GA) for performing QoS-aware web service composition. Maolin and Feng [9] proposed a hybrid genetic algorithm by considering conflict and dependency constraints within services to solve web service composition. Yue et al. [10] proposed an improved Genetic Algorithm for web service composition that results in an enhanced convergence rate. Zhi-peng et al. [11] utilized Simulated Annealing and Genetic Algorithm to propose a new approach QQDSGA (QoE/QoS driven simulated annealing based genetic algorithm) to perform QoS-aware web service composition. Gao et al. [12] and YangYu et al. [13] developed tree-coded genetic algorithms to solve QoS-aware web service composition. Yilmaz et al. [14] proposed improved GA-based approaches such as GA with Simulated Annealing and GA with Harmony search for QoS-aware service composition. Bao et al. [15] developed an orthogonal GA for QoS-aware web service composition, where the initial population is designed based on an orthogonal design and crossover operator. Quanwang et al. [16] designed a novel method to optimize the overall QoS values using a backtracking-based algorithm and an extended GA for QoS-aware web service composition. Although genetic algorithms are widely used in finding QoS-aware web service composition, genetic algorithms have some inherent shortcomings including a low premature convergence rate in local optimum and increase the search space due to fixed predetermined crossover and mutation rates.

Particle Swarm Organization (PSO) and its variants are applied for solving service composition by several researchers [17]. Wang et al. [18] developed a Chaos PSO method supporting global constraints to solve QoS-aware service composition. Liu et al. [19] developed a hybrid quantum PSO algorithm to solve the combinatorial optimization problem for web service composition. Zhao et al. [20] designed an improved discrete immune optimization method based on PSO that combines proportional clone and proliferation of immune optimization algorithm for QoS-aware web service composition. However, these PSO based methods are trapped at local optima in some circumstances and are not often good at diversification. Parejo et al. [21] developed a hybrid greedy randomized adaptive search procedure (GRASP) and Path-Relinking algorithm to solve QoS aware web service composition problem. It attempts at merging good solutions found earlier into the current search iteration. However, it takes more execution time than other algorithms.

Huo et al. [22] proposed a discrete gbest-guided artificial bee colony algorithm to find out the optimal composition path. Seghir et al. [23] proposed a hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition. Where GA is used for global search and Fruit fly optimization is used for local search. Karimi et al. [24] proposed an approach for composition that reduces the search space while performing local service selection by using association rules and service clustering and genetic algorithm to get the global optimal solution. Liu et al. [25] proposed a method that simulates the human intelligence evolution process, named as specific Social Learning Optimization (S-SLO) to solve QoS-aware cloud service composition. Younes et al. [26] proposed a model that integrates the strengths of Memetic algorithm which is gene-based and PSO which is swarm based, to a form Shuffled Frog Leaping Algorithm that accomplishes the task of finding global optimum. Seghir et al. [27] proposed a discrete imperialistic competitive algorithm using artificial bee colony algorithm for global exploration. However, the said approaches still have some intrinsic defects such as low premature convergence rate and increase the search space in local optima due to an exponential increase in Pareto front size due to number of cloud services in composition. Zhang et al. [28] proposed a GA with improved crossover and mutation operator to solve QoS-aware Service Composition for geo-distributed Multi-Cloud environment. Yu et al. [29] presented a greedy based QoS-aware web service composition based on greedy and ant colony optimization algorithms to solve service composition in cloud computing environments. Kurdi et al. [30] discuss a novel method COMbinatorial optimization algorithm for cloud service COMposition (COM2) to solve multiple cloud service composition. Jula et al. [31] proposed improved Imperialist competitive algorithm for service time optimization in cloud service composition. Jula et al. [32] developed a hybrid method of an improved Gravitational attraction search with an Imperialist Competitive Algorithm for cloud computing service composition. Wang et al. [33] proposed a GA to solve QoS-aware service composition in geo-distributed cloud environment.

2.1. Meta-heuristics

Recently, many researchers proposed several exciting meta-heuristics algorithms which work aggressively to find out global optima. A comprehensive review of various versions of KH algorithm had been listed out by [34]. One of those algorithms is Krill algorithm proposed by [35] based on herding behavior of the Krill individuals. The position of the new Krill will be calculated based on the factors that simulates their behavior such as (i) krill individual movement under the influence of other krills (ii) its foraging activity and (iii) random dispersion of krills. Wang et al. [36] proposed a Chaotic particle swarm krill herd algorithm based on mutation operator from the APSO [37], integrated to enhance the global convergence speed while maintaining its true flavor of original algorithm. Wang et al. [38] proposed a Chaotic krill herd algorithm that incorporates chaotic strategies to update the inertia weights into the KH optimization process with the aim of accelerating its global convergence speed. The chaotic KH method adjusts the three main movements of the krill in the optimization process. Further, Wang et al. [39] proposed a novel chaotic cuckoo search (CCS) optimization algorithm by incorporating chaotic theory into CS algorithm. In this algorithm, chaos characteristics are combined with the CS for further enhancing its performance. Then, the elitism scheme is incorporated into CCS to find the best cuckoos. Guoa et al. [40] proposed a new Improved krill herd (IKH) algorithm that concentrates on improving the process of updation of Krill individual new positions. In IKH, Levy flight distribution and elitism mechanism are integrated for calculating the KH motion among the top krills that in turn enhances the convergence speed. Wang et al. [41] proposed a new improved firefly algorithm for global numerical optimization. In this algorithm, the functionality of information exchange between the top fireflies is incorporated during the process of the light intensity updating unlike in standard firefly algorithm. Wang et al. [42] developed a novel hybrid approach combining KH and Quantum based PSO to enhance the capability of local search, there by hindering the chances of premature convergence. Wang et al. [43] proposed a technique to surmount the bad exploitation phase by introducing a hybrid differential evolution technique while KH does the exploration bit, balancing the two vital yet contrasting components in the stochastic world. Wang et al. [44] developed a Mouth Search algorithm (MS) based on phototaxis and Lévy flights of the moths. Wang et al. [45] developed a multi-stage krill herd (MSKH) algorithm based on the standard KH and the local mutation, crossover (LMC) operator to balance the exploration and exploitation stages. Wang et al. [46] developed a Lévy-flight krill herd algorithm for solving global optimization problem. A new Opposition based KH with Cauchy mutation and position clamping is introduced into for avoiding the trapping into local minima for some complex benchmark functions [36]. Simulated Annealing based KH algorithm is proposed to fine tune the performance of KH by adopting new krill selecting (KS) operator and the elitism strategy [47]. A Stud Krill Herd (SKH) technique is proposed by [48], where stud krill shares optimal information to all krills instead of randomly selected krills. Wang et al. [49] developed a method for tuning better parameter settings and finding out the effective coefficients for three krill motions and for crossover and selection operators that hinders the chances for trapped in local optimum. Gaige et al. [50] proposed a novel hybrid KH and harmony search to solve global optimization problem. Wang et al. [51] proposed a biogeography-based krill herd algorithm to solve complex optimization tasks consisting of a new Krill migration operator for updating the krill individuals. Heqi et al. [52] introduced a hybrid KH and ABC (KHABC) algorithm to solve global optimization.

Monarch butterfly optimization (MBO) is a new metaheuristic algorithm that mimics the migration of butterflies [53]. Wang et al. [54] proposed a new version of MBO that uses a crossover operator and a greedy strategy that ensures only better individuals to move to next generation. Yanhong et al. [55] solved the famous 0–1 knapsack problem by proposing the novel binary monarch butterfly optimization method. Wang et al. [56] developed a hybrid differential evolution accelerated particle swarm optimization (DPSO) algorithm to solve numerical optimization problems. Wang et al. [57] proposed a hybrid model HS/CS to solve numerical optimization algorithm. Cui et al. [58] presented an oriented cuckoo search algorithm to improve DV-Hop performance for cyber–physical systems. Gaige et al. [59] proposed a hybrid method harmony search (HS) with biogeography based optimization to solve global numerical optimization. [60] presented an improved version of bat algorithm in combination with a differential evolution to solve the uninhabited combat air vehicles (UCAV) 3-D path planning. Guohua et al. [61] developed a novel variable reduction for derivative unconstrained optimization and constrained optimization problems with equality and active inequality constraints.

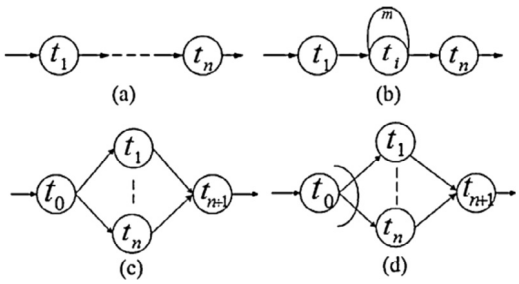

Fig. 1. A Workflow of service composition.

Rizk-Allah et al. [62] proposed a multi-objective fruit fly optimization algorithm for addressing multi objective optimization. Rizk-Allah et al. [63] developed a parallel hurricane optimization algorithm for solving economic emission load dispatch problem in modern power systems. Liu et al. [64] proposed a multi-objective optimization model for gesture segmentation based on a two-phase estimation of distribution algorithm. Guohua et al. [65] presented a multi-population based framework to realize the ensemble of multiple DE variants. Wang et al. [66] proposed an evolutionary multi-objective optimization algorithms (B*-tree and a multistep simulated annealing) to solve high performance computing for cyber–physical social systems. By combining the Earth observation resources, a coordinated architecture is planned to enhance the utilization of its resources by proposing the highest-weight-first allocated and tabu list based simulated annealing algorithms [67]. Zhang et al. [68] developed a new bat algorithm with mutation and used for enhancing the global convergence speed in the image matching problem. Rui et al. [69] proposed a novel reference-inspired coevolutionary algorithm to solve many optimization models. Wang et al. [70] developed a self-adaptive extreme learning machine to solve classification problems.

Wang et al. [71] proposed six information feedback models based on improved metaheuristic algorithms. Wang et al. [72] presented a hybrid method based on cuckoo search and krill herd algorithms to solve global optimization problems. Feng et al. [73] proposed a binary moth search algorithm to solve Discounted 0-1 Knapsack Problem. Wang et al. [74] presented a novel elephant herding optimization and a novel Earthworm optimization algorithm [75] to solve global optimization problems. Feng et al. [76] proposed a method chaotic monarch butterfly optimization algorithm with Gaussian mutation to solve 0–1 knapsack problems. Yi et al. [77] discussed a variant of probabilistic neural network with self-adaptive strategy to solve transformer fault diagnosis problem. Rui et al. [78] proposed a multiobjective evolutionary

Table 1
List of QoS attributes and their description.

S.no	Attribute name	Description
1	Price	It defines the amount needed to pay by user for requesting a service
2	Reliability	It tells how frequent a service is available during the user request
3	Response Time	Time taken in seconds to serve a request once the request is received
4	Reputation	Based on user experience, a service is evaluated in terms of certainty
5	Throughput	In certain time period, the number of requests served is defined as Throughput
6	Security	It defines on how much the service is offering privacy by verifying the users involved in it

algorithm based on decomposition (MOEA/D) to solve optimization problems. Rui et al. [79] presented a multiobjective evolutionary algorithm based on decomposition LWS (MOEA/D-LWS) for many-objective optimization. Guohua et al. [80] proposed a novel population-based across neighborhood search to solve numerical optimization. Wang et al. [81] proposed a hybrid firefly-inspired krill-herd optimization algorithm based on firefly and krill herd algorithms to solve optimization problems.

3. Modeling of QoS-aware cloud service composition

3.1. Service composition model

In service oriented computing (SOC), service providers often find difficulty in providing a service to the request of an user due to its complexity. In such cases, the complex task is decomposed into sub-tasks. Those candidate services that perform sub-tasks is later composed to accomplish the initial request.

Let T be the initial user request. Now, $T = \{T_1, T_2, T_3, \ldots, T_n\}$ where n is the number of sub-tasks. Each sub-task is performed by the abstract service S_i which is the collection of candidate services $\{ws_{i,1}, ws_{i,2}, ws_{i,3}, \ldots, ws_{i,n_i}\}$ where n_i refers to the number of candidate services for that particular S_i. These candidate services possess different Quality of Services (QoS) values but same functionalities.

QoS attributes play a vital role in deciding the candidate service for each abstract service as they exhibit similar roles. These QoS attributes for each candidate service is represented as $\text{QoS}(ws_{i,j}) = \{q_1, q_2, q_3, \ldots, q_r\}$, where q_k denotes the kth number of QoS attribute, i denotes the ith abstract service and j denotes the jth candidate service for S_i. We denote the QoS Attributes set of a composite service by $Q(cs)$, which is calculated by using the aggregation functions as shown in Table 3.

This aggregation is done based on structure of a workflow [4, 82]. A typical workflow that comprises of various structures is shown in Fig. 1. It has four basic structures such as (a) sequential, (b) loop, (c) parallel, and (d) conditional. In a sequential structure, all the tasks are processed in sequential order. In a parallel structure, all parallel tasks are executed to proceed to the subsequent tasks. In a conditional structure, at least one task is executed among multiple branches to proceed further. In a loop structure, a task executes multiple times. If we were to compose different services into a single composite service (CS), their respective QoS values need to be aggregated according to the aggregate functions by using a fitness function (Section 3.2). The final value that we obtain after applying composite service to fitness function is the final fitness value ($csQoS$), that needs to be optimized. This value is used to evaluate the importance of the composite service with respect to other composition paths.

A single abstract service has multiple candidate services that has the same functionalities with different QoS values. Hence, it becomes tedious to select an optimal service that fulfills the requirements of user. To overcome this issue, QoS attributes help to rescue, as each candidate service is associated with the QoS attributes such as Availability, Response time, Reliability, Throughput etc. These QoS attributes are used to select a service which contributes in attaining an optimal candidate service for each abstract service. The QoS attributes and their description are presented in Table 1.

Table 2
QoS attributes.

Positive attributes	Negative attributes
Availability	Response time
Throughput	Cost
Reliability	Execution time

While calculating the value of csQoS, we need to examine the composition workflow that comprises of sequential, parallel, conditional, and loop structures, as shown in Fig. 1. We have considered only sequential structure in this work, while other structures can be effortlessly transformed to sequential. For instance, we could transform a conditional structure by using the conditional probability to a sequential structure, and a loop structure can also be easily remodeled by using loop peeling, as explained in [4].

There are two categories in QoS attributes, such as positive and negative attributes [82]. If a high value is desirable for a QoS value, it is termed as the positive attribute, and if a low value is desirable, it is termed as the negative attribute. The classification of QoS attributes is shown in Table 2. Based on the characteristics of QoS attributes, they are aggregated using aggregations functions. The formula of these functions for particular QoS attributes in various composition structures are shown in Table 3. Where n is the number of abstract services in a composition path and m is the number of parallel cloud services. In conditional composition, we have $\sum_{i=1}^{n} Pr_i = 1$ where n is the number of choices and Pr_i is the probability of selecting a conditional service. As these QoS values range widely, we need to normalize all the attribute values to fetch them in the same range [0, 1], so that no attribute dominates the other attributes. As we have two categories of attributes, we need to manage them individually while being normalized.

The formula for normalization of positive attributes is as follows:

$$UniQ_k = \begin{cases} \dfrac{Q_k - minQ_k}{maxQ_k - minQ_k} & maxQ_k - minQ_k \neq 0 \\ 1 & maxQ_k - minQ_k = 0 \end{cases} \quad (1)$$

The formula for normalization of negative attributes is as follows:

$$UniQ_k = \begin{cases} \dfrac{maxQ_k - Q_k}{maxQ_k - minQ_k} & maxQ_k - minQ_k \neq 0 \\ 1 & maxQ_k - minQ_k = 0 \end{cases} \quad (2)$$

where $maxQ_k$ and $minQ_k$ denote the highest and lowest QoS value of the kth dimension attribute for all the composition paths. The normalized value will be 1 if they are same.

3.2. Fitness function

This function is used to evaluate the fitness (csQoS) of the particular composition path. If we are solving the standard maximization problem, then we have to be more inclined to find out the composition path that has the maximum fitness value. Else if we are solving the standard minimization problem, then we try

Table 3
QoS aggregation functions with various composition structures.

QoS attributes	Sequence	Parallel	Loop (call ws_i^j service k times)	Conditional(Pr_i)
Availability	$\prod_{i=1}^{n}(q_{avail,i})$	$\prod_{i=1}^{m}(q_{avail,i})$	$k * \prod_{i=1}^{n}(q_{avail,i})$	$\prod_{i=1}^{n} q_{avail,i} * Pr_i$
Response Time	$\sum_{i=1}^{n}(q_{rt,i})$	$min(q_{rt,i})$	$k * \sum_{i=1}^{n}(q_{rt,i})$	$\sum_{i=1}^{n} q_{rt,i} * Pr_i$
Reliability	$\prod_{i=1}^{n}(q_{rel,i})$	$max(q_{rel,i})$	$k * \prod_{i=1}^{n}(q_{rel,i})$	$\prod_{i=1}^{n} q_{rel,i} * Pr_i$
Throughput	$min(q_{tp,i})$	$min(q_{tp,i})$	$k * \prod_{i=1}^{n} q_{tp,i}$	$\prod_{i=1}^{n} q_{tp,i} * Pr_i$

to find out the composition path that has the least fitness value when compared to others. In this way, we use this function to select our required composition path. The formula for fitness function is stated in Eq. (3).

To find the value of csQoS, we employ the SAW (Simple Additive Weighting) method as used in [20,83], based on $Q(cs)$. Initially, different weights are assigned to each QoS attribute based on its level of importance.

Let w_k be the weight of the kth QoS attribute. Now, $\sum_{k=1}^{r} w_k = 1$, where w_k is fixed by the user within the range [0, 1] based on their interests. The service composition model for maximization of csQoS is defined as follows:

$$\textbf{Max}\ csQoS = \sum_{k=1}^{r} UniQ_k * w_k \tag{3}$$

where r denotes the number of QoS attributes. Thus, the efficiency of our algorithm is calculated by the quality of the composition path thrown by our method. This quality is often termed as 'fitness' of the whole path that is formed by selecting certain concrete services from the respective abstract services.

4. Eagle Strategy with Whale Optimization Algorithm (ESWOA)

This section describes our proposed Eagle Strategy with Whale Optimization Algorithm (ESWOA).

4.1. Eagle strategy

Eagle strategy technique was developed by Yang and Deb [84] that does optimization in two phases, preserve the balance between exploration and exploitation. In this strategy, the exploration is done similar to how an eagle searches for its prey initially. Once the prey is found the eagle changes its behavior in chasing the prey to intensive attacking. This has been imitated by this strategy in the exploitation phase, by integrating an optimization technique that does a rigorous local search such as downhill simplex or Nelder–Mead method [85]. Evidently, we could use various efficient meta-heuristic algorithms like Particle Swarm Organization, Firefly Algorithm [86], Differential Evolution or Artificial Bee Colony to do a strenuous local search. In [84], Levy walk and Firefly algorithm have been coupled to draft a Eagle Strategy technique. The pseudocode for Eagle strategy is presented in Algorithm 1.

The parameter P_e enables us to authorize in an iterative manner between the exploration and exploitation. To start with, initial solutions are mounted from a large search space as these solutions often constitute high diversity. These instances undergo an evolution by an intensive metaheuristic algorithm that leads to a converged state, the state in which solutions have low diversity. Subsequently, a new set of solutions are acquired again from the larger search space, that again comprises of high diversity, for another round of intensive iteration stage [87]. In a similar manner, exploration and exploitation have been utilized to preserve the superior degree of diversity in the entire population.

Algorithm 1: Eagle Strategy.

```
Objective function f(x)
Initialization of sample space
While(t < maximumnumberofiterations)
    Do Global Exploration by randomization
    Fitness Evaluation and finding a promising solution
    if(P_e < rand), do
        Local exploitation by efficient local optimizer
      if(a better solution is found)
          current best solution is updated
      end
    end
    t = t + 1;
end
```

Eagle strategy has been popularly used by researchers to enhance the efficiency of metaheuristic algorithms. Eagle strategy with cuckoo search is proposed in [88] for the optimal balance between intensification and diversification. They provided a practical estimate based on the intermittent search theory. Similarly, Eagle Strategy using Flower Algorithm [89] has been proposed to prove the effectiveness of technique. An improved artificial bee colony algorithm with two stage eagle strategy (ETABC) is proposed in [87] where cuckoo search has been used in the first stage as it uses Levy Flights.

4.2. Whale optimization algorithm (WOA)

WOA is one of the latest metaheuristic algorithms introduced by Mirjalili et al. [90] in 2016. WOA algorithm mimics the behavior of the Humpback whales. Usually, these whales wander in groups and often behave intelligently to catch their prey, which is called the bubble-net attacking method as shown in Fig. 2, a strategy that hunts small fishes by trapping them in self-created distinctive bubbles along a circle or 9-shaped path. This foraging has been mimicked in the WOA algorithm.

To replicate this behavior in WOA, there is 50% probability that the whale follows any of the two paths (circle or 9-shaped paths) mentioned to update the position of the whale during optimization. The mathematical representation of the WOA strategy is given as follows:

4.2.1. Encircling prey mechanism

In WOA, we make an assumption that the current best solution is the target prey or near optimal solution, while other whales try to update their positions in the direction of this current best solution. This is represented as follows:

$$\vec{D} = |\vec{C} \cdot \vec{X}^*(t) - \vec{X}(t)| \tag{4}$$

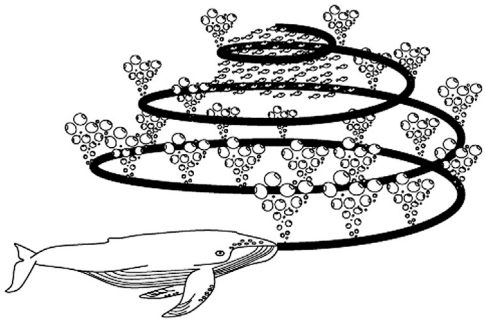

Fig. 2. Bubble-net hunting behavior of humpback whales.

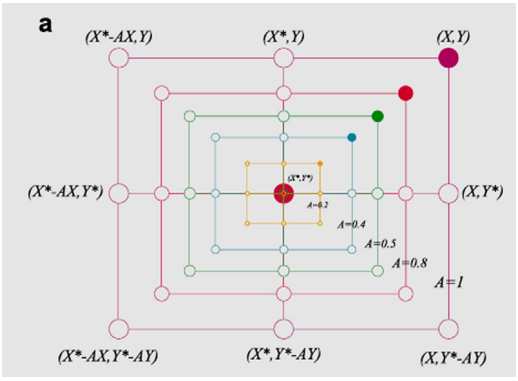

Fig. 3. Shrinking encircle mechanism.

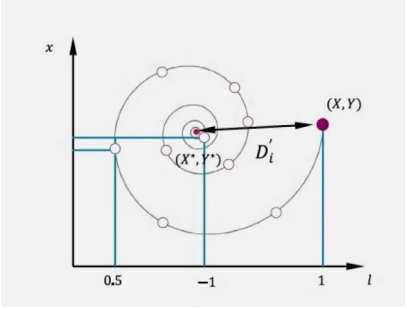

Fig. 4. Spiral updating position.

$$\vec{X}(t+1) = |\vec{X}^*(t) - \vec{A} \cdot \vec{D}| \tag{5}$$

where $\vec{X}^*$ denotes the position of best whale found till now, $\vec{X}$ is the position of the whale at current iteration and t represents the current iteration.

$$\vec{A} = 2.\vec{a} \cdot \vec{r} - \vec{a} \tag{6}$$

$$\vec{C} = 2 \cdot \vec{r} \tag{7}$$

$\vec{a}$ will be decreased from 2 to 0 linearly in every iteration and $\vec{r}$ is the random number in the range [0, 1].

4.2.2. Bubble net attacking method (exploitation)

- **Shrinking Encircling:** We decrease the value of a linearly that in turn decreases $\vec{A}$ value, as it ensures that the newly updated whale will fall in the range of $[-a, a]$. Now, the new possible positions of the individual whale that will be determined in between the current whale position and best whale position is shown in Fig. 3 in a 2-Dimensional space. where (X, Y) denotes the position of current selected whale and (X^*, Y^*) represents the best whale position.
- **Spiral Updating Position:** The current whale follows the best whale by moving in an helix shaped path as shown in Fig. 4. This figure also shows the new possible positions of whale that is going to be updated. A mathematical equation represented for this helix shaped behavior, which is used to update the position between the current whale and the target whale (current best found till now) is stated in Eq. (8).

$$\vec{X}(t+1) = \vec{D}' \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t) \tag{8}$$

where l represents a random number within the range $[-1, 1]$, b ensures the logarithmic shape and $\vec{D}'$ is the distance the between current whale and the prey.

Table 4
Mapping between Whale Foraging and service composition.

Whale Foraging	Service composition
Search Agent (Whale)	Service composition solution
Leader Whale	Optimal service composition solution
Speed of searching and foraging	Speed of algorithm optimization
Fitness of whale	Fitness of composite service

4.2.3. Search for prey (exploration)

Unlike in encircling prey mechanism that updates the current whale position based on the best whale, here updating is done based on the random whale which is decided by the vector $\vec{A}$. Exploration follows the Eqs. (9) and (10).

$$\vec{D} = |\vec{C} \cdot \vec{X}_{Rand} - \vec{X}| \tag{9}$$

$$\vec{X}(t+1) = |\vec{X}_{Rand} - \vec{A} \cdot \vec{D}| \tag{10}$$

where $\vec{X}_{Rand}$ is the random whale selected. This whale is used as a reference to update the current selected whale. The pseudocode of Whale Optimization Algorithm is presented in Algorithm 2.

Algorithm 2: Whale Optimization Algorithm (WOA)

```
Generate initial whale population randomly;
Determine fitness of every search agent;
Store X* as best search agent;
while t < MAX_Iter do
  for Each search agent do
    Update a, A, C, l, p;
    if p < 0.5 then
      if |A|< 1 then
        Update individual position using
          shrinking encircling mechanism (Equation (4));
      else if |A|>= 1 then
        Select a random whale;
        Update current whale by Equation (7);
      end if
    else if p >= 0.5 then
      Update the whale position through spiral updating mechanism
      through (Equation (5));
    end if
  end for
  Update X* if there is a better search agent;
  Update t = t + 1;
  Stop, if Halting condition is TRUE;
end while
```

Before proceeding to proposed method, we present the correspondence (Mapping) between Whale Foraging and service composition for better understanding as shown in Table 4.

Table 5
Initial population.

S.no	Population	csQoS
1	[1, 1, 3, 4, 3]	0.9097
2	[3, 2, 1, 2, 2]	1.5791
3	[1, 3, 2, 4, 1]	1.1330
4	[2, 4, 3, 3, 2]	1.4181
5	[4, 2, 4, 3, 3]	1.4714
6	[3, 2, 3, 1, 2]	1.5587

4.3. Eagle Strategy with Whale Optimization Algorithm (ESWOA)

In our proposed technique, we apply the exploitation process of the WOA algorithm (Section 4.2.2) to perform only the local search, that ensures the intensification part of the Eagle strategy. We apply a new approach for the exploration purpose (described in Section 4.3.4) that performs the diversification part of our Eagle Strategy. We explain the phases starting from encoding to obtaining optimal solution as follows.

4.3.1. Encoding

In traditional WOA algorithm, the whale position represents the feasible solution to an optimization problem that is denoted as a m-dimensional vector. The whale that has the highest fitness represents as a Leader Whale, and the fitness is denoted by the Leader score (X^*), and its position is represented as the Leader Position (LP). We need to maintain the ranges of each dimension for our candidate solution depending on the requirement, with respect to our service composition problem. Therefore, the encoding of whale position should consider the procedure for producing the candidate solutions according to our necessity.

The main goal of QoS-aware cloud service composition is to select a service from a pool of candidate services $ws_{i,1}$, $ws_{i,2}$, $ws_{i,3}$, …, ws_{i,n_m} of each abstract service S_i, obeying the QoS constraints, which finally result in the maximizing the quality of composite service. We adopted an integer encoding scheme similar to [22], as depicted in Fig. 5 that maps an integer to the concrete service. A whale position x_d is represented as a m-dimension vector $x_d = \{x_d^1, x_d^2, x_d^3, \ldots, x_d^m\}$. In this array of numbers, each element x_j^i represents the value of the jth concrete service from the ith abstract service. This value is bound to be in the range of [lb, ub], where lb is 1 and ub is number of concrete services in the pool S_i.

4.3.2. Initialization

In WOA, we generate SN number of whale positions randomly, represented as $\{x_1, x_2, x_3, \ldots, x_{SN}\}$ according to the Eq. (11).

$$x_d^i = lb + \lfloor rand(0, 1) * (ub - lb) \rfloor \tag{11}$$

where SN represents the number of candidate solutions in our search space (population). Now, for every candidate solution, we calculate their fitness values $csQoS$ using the Eq. (3) (Line 1 and 2 in Algorithm 3). We will get SN fitness values, out of which the highest fitness value is treated as Leader score (X^*) (Line 3 in Algorithm 3) and the corresponding whale position is treated as Leader position.

Let us assume that we have $m = 5$ and $n_i = 5$ where $1 \leq i \leq 5$ and SN is 6. The sample population is generated randomly satisfying the said constraints and their respective fitness values ($csQoS$) are calculated by Eq. (3), as in Table 5. For example [3, 2, 1, 2, 2] means the selection of $ws_{1,3}$, $ws_{2,2}$, $ws_{3,1}$, $ws_{4,2}$, $ws_{5,2}$. From these, the highest csQoS is 1.5791 and is stored in X^* and its corresponding position is stored as the Leader position.

4.3.3. Iteration process

After initialization, the candidate solutions will undergo exploration and exploitation processes for MAX_Iter times in search of better solutions than X^*. Let MAX_Iter be the maximum number of iterations that the algorithm runs.

Algorithm 3: Eagle Strategy With WOA (ESWOA)

```
1: Generate initial population randomly;
2: Determine fitness of all individuals using Eq. (6);
3: Store X*=best fitness individual;
4: while t < MAX_Iter do
5:   for Each individual in population do
6:     Calculate probability prob according to (12);
7:     Generate another random number q;
8:     if q < prob then, do
9:       Update individual position by Eq. (13);
10:    end if
11:    Update X* if there is a better solution;
12:  end for
13:  Determine random number rand;
14:  if P_e < rand then {Here P_e=0.2}
15:    Do Local Search, go to STEP 19;
16:  else
17:    Do Global Search, go to STEP 30
18:  end if
19:  for Each individual in population do
20:    Update a, A, C, l, p
21:    if p < 0.5 then
22:      if |A|< 1 then
23:        Update the position of individual using
24:        shrinking encircling mechanism
25:      end if
26:    else if p >= 0.5 then
27:      Update the individual position by spiral updating mechanism
28:    end if
29:  end for
30:  Update X* if there is a better solution
31:  Update t = t + 1
32:  Stop, if Halting condition is TRUE
33: end while
```

4.3.4. Exploration phase in eagle strategy

In this phase of our proposed algorithm, each dimension of the whale position is changed according to the probability $prob$ as mentioned in the Eq. (12).

$$prob = 0.3(1 - \frac{iter}{MAX\_Iter}) \tag{12}$$

where $iter$ is the current iteration number and MAX_Iter is the maximum number of iterations specified initially.

Now, for a currently selected whale, we generate a random number within the range of [1, dim], to determine which dimension should be randomly changed. Another random number q is extracted within the interval [0, 1] and is compared with the probability $prob$. If q<$prob$, then a modification is done to the dimension according to the Eq. (13).

$$X_j = X_{jmin} + rand.(X_{jmax} - X_{jmin}) \tag{13}$$

where $rand$ is a random number generated within the range [0, 1]. X_j is the selected dimension that is to be altered, X_{jmax} is the maximum of dimensions from the currently selected whale, X_{jmin} is the minimum of dimensions.

For this modified whale that we get after performing Eq. (10), its fitness value is calculated based on the Eq. (3). We compare it with the best value(X^*), and update X^* if the new fitness value is greater than the earlier value and its corresponding Leader Position too. This process is illustrated in Algorithm 3 from lines 5–12.

To illustrate the said process, we take the earlier example. For initial iteration, $iter$ value will be 0 ($MAX\_Iter = 500$). So, the probability generated from the Eq. (9) would be 0.3 ($prob = 0.3$). Let us assume that the fourth candidate solution(x_4) is the current selected whale. Now, we generate a new random number, in this case, say $q = 0.2038$. Since $q < prob$, we have to update the randomly generated dimension which is within the range [1, 5] through the Eq. (5). So, the new whale position after updation

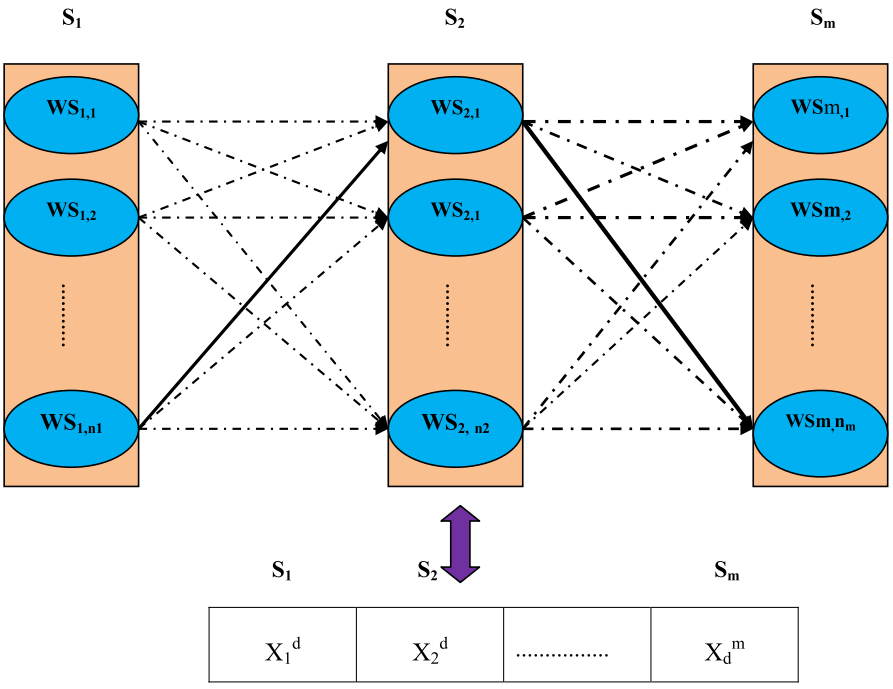

Fig. 5. Integer coding scheme.

Table 6
Population after exploration phase.

S.NO	Population	csQoS
1	[1, 1, 3, 4, 3]	0.9097
2	[3, **1**, 1, 2, 2]	**1.5815**
3	[1, 3, 2, 4, 1]	1.1330
4	[2, 4, 3, 3, 2]	1.4181
5	[4, 2, 4, 3, **2**]	**1.6760**
6	[3, 2, 3, 1, 2]	1.5587

Table 7
Population after exploitation phase.

S.NO	Population	csQoS
1	[1, 1, 3, 4, 3]	0.9097
2	[3, 1, 1, 2, 2]	1.5815
3	[1, 3, 2, 4, 1]	1.1330
4	[2, 4, 3, 3, 2]	1.4181
5	[4, 2, 4, 3, 2]	1.6760
6	[**4**, 2, **4**, **2**, 2]	**1.6785**

would be $x_4 = [4, 2, 4, 3, 2]$ and its fitness value is fitness $(x_4) = 1.6760$. As it is greater than the previous fitness value, we will update the fitness value and its position in population.

After all the whales are exploited, the new population with their respective fitness along with the updated information (see bold and underlined numbers) is shown in Table 6. Now, X^* will be 1.6760.

Then, we need to fix the parameter P_e that authorizes the Exploration and Exploitation phases. We assign P_e value as 0.2, after examining numerous experimental results from [88,89]. Now, a random number is generated within the range [0, 1] and compared with P_e. If P_e is less than the random number generated, then we will perform the exploitation phase (Section 4.3.5), else we increase the iteration count and execute exploration phase (Section 4.3.4). This process is described in Algorithm 3 from lines 13–18.

4.3.5. Exploitation phase in eagle strategy

As we have seen two maneuvers (Shrinking encircling mechanism and Spiral updating as in Section 4.2.2) by a whale that inherently performs exploitation in the standard WOA algorithm from Eqs. (5) and (8), we use the same approach for searching the best solution whale during exploitation phase of our proposed method. As there is an equal probability for selection between the two approaches, we use a random variable p to decide.

$$\vec{X}(t+1) = \begin{cases} \vec{X}^*(t) - \vec{A}.\vec{D} & \text{if } p < 0.5 \\ \vec{X}(t+1) = \vec{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t) & \text{if } p \geq 0.5 \end{cases} \tag{14}$$

After updating the whale position from the Eq. (14), we calculate the fitness of this modified whale and update to X^* if it has a greater value and its corresponding Leader Position too. After performing *MAX_Iter* number of iterations, our proposed method produces the final Leader Position that has the best optimal fitness value (*csQoS*) and treated as an *optimal service composition path*. This process is described in Algorithm 3 from lines 19–30.

We illustrate the exploitation process by the following example. As first, we calculate the parameters a, A, C, l, p (as given in line 20 of Algorithm 3). To perform the shrinking encircling mechanism, p should be less than 0.5 ($p < 0.5$) and $|A| < 1$. Let us assume that the first whale is selected for updation. After performing this mechanism by the Eqs. (4) and (5), the updated whale position is [3, 1, 2, 1, 1] with the fitness value of 1.2731, which is less than X^*. So csQoS is not updated. Considering the Encircling mechanism, p should be greater than or equal to 0.5 ($P \geq 0.5$). Let us assume that the sixth whale has chosen to perform the encircling mechanism. After updating its position by using Eq. (8), its new position will be [4, 2, 4, 2, 2] with the fitness value of 1.6785, which is greater than X^*. So, the new csQoS value will be updated as $X^* = 1.6785$. The updated population at current iteration is shown in Table 7.

Similarly, we perform these two maneuvers for all the whales to update their fitness and positions if any whale is found to be greater than the leader whale's fitness value (X^*). The population after completing all iterations (*MAX_Iter*) and their respective fitness values is shown in Table 8. The best optimal composition path for our example is [4, 1, 1, 1, 2] and its best fitness value X^* is 1.6923.

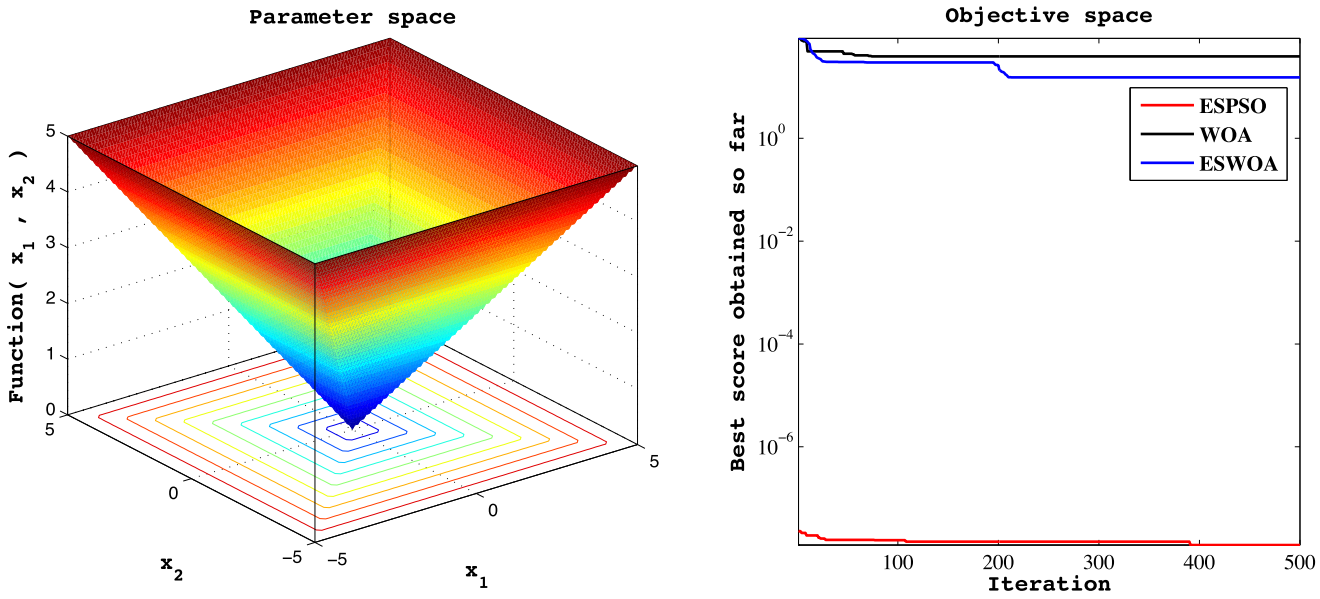

Fig. 6. Convergence curves for Function-F1.

Table 8
Final population.

S.NO	Population	csQoS
1	[1, 1, 3, 4, 3]	0.9097
2	[4, 2, 1, 3, 2]	1.6890
3	[2, 2, 1, 1, 2]	1.5806
4	[2, 1, 1, 1, 2]	1.5816
5	[4, 2, 4, 3, 2]	1.6760
6	[4, 1, 1, 1, 2]	**1.6923**

Contrast to other algorithms, ESWOA implements exploration and exploitation phases by using P_e till we get the possible optimum globally or the algorithm meets the end criterion. The main intention of using Eagle Strategy for our approach is due to its unique capability in generating a new search space for each iteration making the population more diverse in nature. This feature enhances the probability of reaching the global optimum. Because in an iteration while reaching optimum solutions, we can sense the search space (population) getting more and more intensified towards the local optimum (optimal solution in current search space) that prohibits in reaching the other search space bounded optimum solutions, which exactly is the problem with the contemporary algorithms.

5. Experiments

5.1. Experimental setup and dataset

In our proposed work, the standard QWS dataset [91] has been used for performing experiments, that comprises of various QoS attributes such as Reliability, Availability, Response time, Throughput, etc. All the experiments have been carried out in a PC with Intel Core i5, 8GB Ram, Windows 8.1 (64-bit system) and MATLAB R2013a version.

We have evaluated the performance of our proposed algorithm in the following ways:

1. Validation against Benchmark functions
2. Comparing ESWOA with contemporary algorithms
3. Statistical Analysis of ESWOA with contemporary algorithms.

Table 9
Parameter settings.

Variables	Values
Population size	30
Number of runs	30
No of Iterations per run	500
P_e value in ESWOA	0.2
ESPSO	
$c1, c2$	2
w_{min}	0.4
w_{max}	0.9
λ	2

5.2. Validation against benchmark functions

We have tested our proposed algorithm ESWOA with certain benchmark functions which include unimodal and multimodal functions [90]. The unimodal functions test the exploitation capability of our proposed algorithm as it has only a single global optimum. The multimodal functions will test the exploration capability of our proposed algorithm and how it escapes the local optimum as it eventually approaches to a global optimum.

These benchmark results have been compared with traditional WOA algorithm [90] and ESPSO [92]. The parameter settings used in these algorithms are stated in Table 9. The list of all benchmark functions and their specifications are presented in Table 10. The statistical results such as mean and standard deviation obtained for these benchmark functions are listed in Table 11.

5.2.1. Comparison of convergence curves

The convergence curves of our proposed algorithm and the compared algorithms are shown in Figs. 6–16 for F1–F11 benchmark functions. In the said Figures, y-axis indicates the best fitness value obtained so far and x-axis represents the maximum number of iterations. Among the chosen benchmark functions, F1–F3 are unimodal functions and F4–F11 are multimodal functions. We observe that our proposed approach converges to global optimum rapidly when compared with other algorithms, that shows the high ability to escape local optimum.

Evaluation of exploitation capability

As F1–F3 are unimodal functions, these functions test the exploitation capability of algorithms. It is evident from Figs. 6 to 8

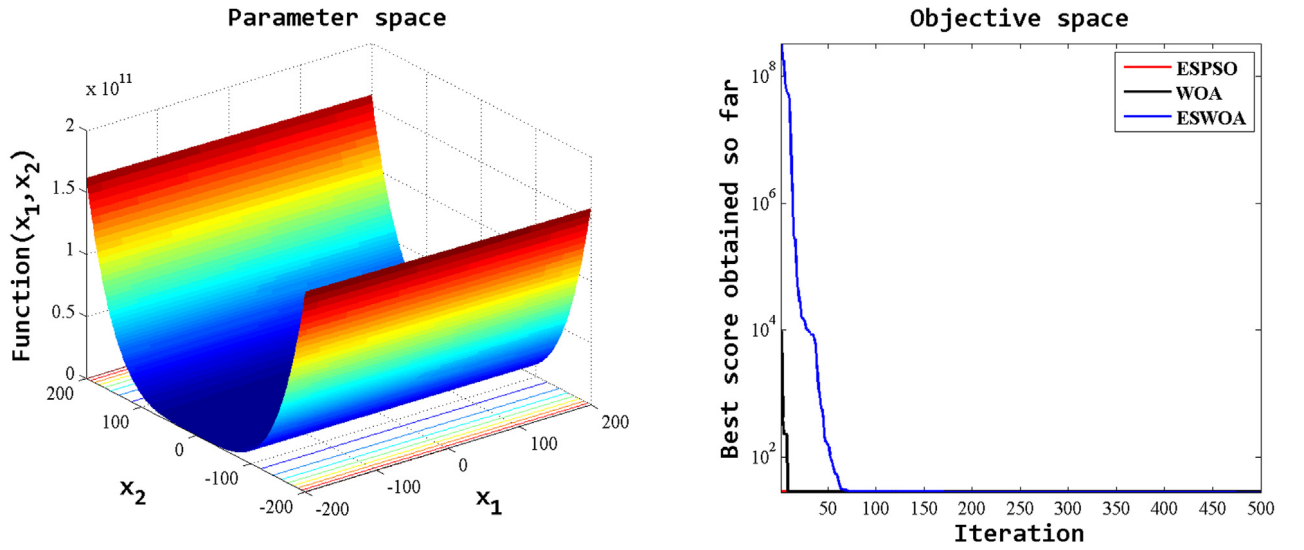

Fig. 7. Convergence curves for Function-F2.

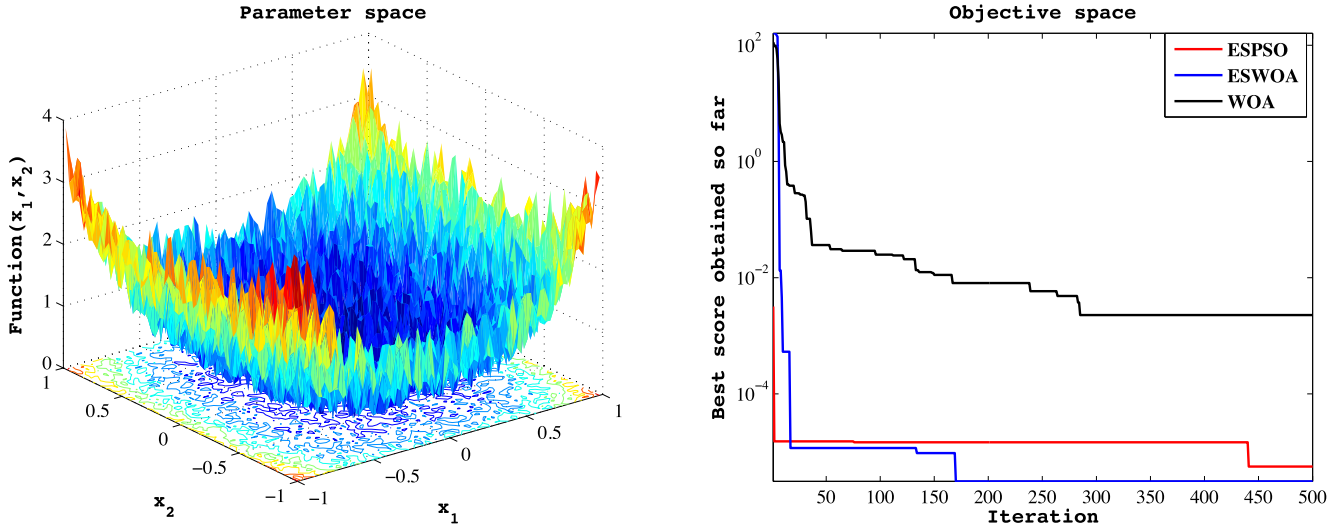

Fig. 8. Convergence curves for Function-F3.

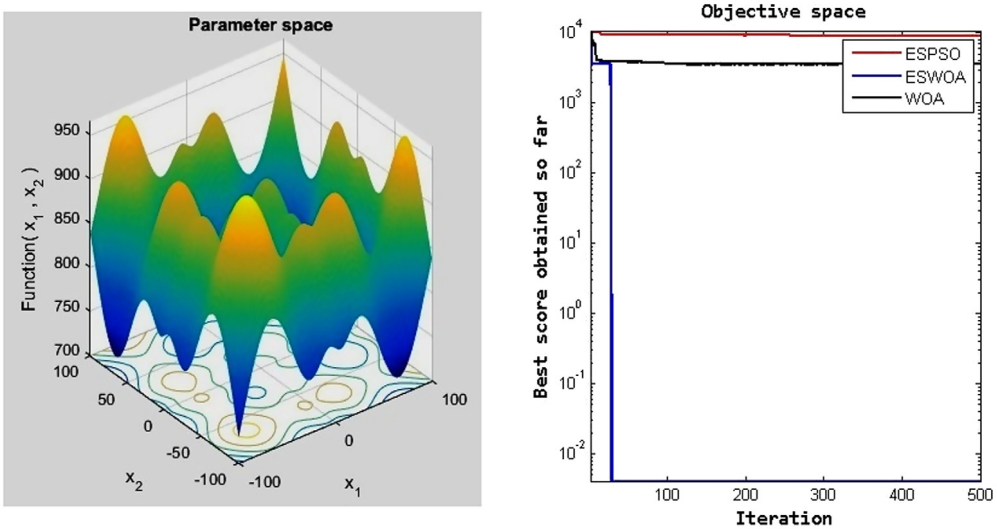

Fig. 9. Convergence curves for Function-F4.

Table 10
List of benchmark functions.

Function	Dim	Range	f_{min}
$F_1(x) = max_i\{\|X_i\|, 1 \le i \le n\}$	30	[−100,100]	0
$F_2(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x^2)^2 + (x_i - i)^2]$	30	[−30,30]	0
$F_3(x) = \sum_{i=1}^{n} ix_i^4 + random(0, 1)$	30	[−1.28,1.28]	0
$F_4(x) = 418.9829 * dim - \sum_{i=1}^{dim} x_i \sin(\sqrt{\|x_i\|})$	30	[−100,100]	0
$F_5(x) = -20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} X_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	30	[−32,32]	0
$F_6(x) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(X_i, 10, 100, 4)$	30	[−50, 50]	0
$F_7(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\}+$ $\sum_{i=n}^{n} u(X_i, 5, 100, 4)$	30	[−50, 50]	0
$F_8(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	2	[−4.5, 4.5]	0
$F_9(x) = Sin^2(\pi w_1) + \sum_{i=1}^{d-1}(w_i - 1)^2[1 + 10sin^2(\pi w_i + 1)] + (w_d - 1)^2[1 + sin^2(2\pi w_d)]$	30	[-10, 10]	0
$F_10(x) = -\sum_{i=1}^{d} sin(x_i)sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$	30	$[0, \pi]$	2
$F_11(x) = \frac{1}{2}\sum_{i=1}^{d}(x_i^4 - 16x_i^2 + 5x_i)$	30	[−5, 5]	0

Table 11
Benchmark functions results.

Function	WOA		ESPSO		ESWOA	
	Average	Std. Dev	Average	Std. Dev	Average	Std. Dev
F1	45.322	24.2993	1.028	0.0667	38	22.2199
F2	28.1684	0.3753	1021.6175	193.6183	28	0
F3	0.0032	0.0039	50.0307	10.4801	8.0316E−06	8.5354E−06
F4	10734.139	115.456	11891.7006	112.389	10661.25	7.40034E−12
F5	5.15143E−15	2.70406E−15	3.4687	0.1258	8.88178E−16	0
F6	0.0179	0.0086	0.5463	0.0769	1.57054E−32	0
F7	0.5584	0.3194	3.1224	0.2916	0.4033	0.2709
F8	38.0716	19.1832	8.0381E−04	8.0124E−04	6.2140E−11	5.9013E−12
F9	40.3801	21.4131	52.1318	24.8014	4.3148E−03	3.1816E−04
F10	2.7432E−0.6	1.8132E−0.2	1.8110E−01	0.1238	3.6183E−10	2.9061E−10
F11	0.8413	0.6186	0.5414	0.2189	2.6018E−03	1.8143E−04

that ESWOA gives reasonably good results when compared with other algorithms. ESWOA converges faster than other algorithms for F2 and F3.

Evaluation of exploration capability

Functions F4–F11 are multimodal functions that test the exploration capability of algorithms in reaching multiple global optima solutions. We observe that ESPSO often gets stuck at local optimum where as our proposed approach performs well in overcoming the multiple local optima present in search space and trying to reach global optima as quickly as possible. Our proposed Eagle Strategy technique enhances the diversity of population in every iteration that helps in reaching global optimum faster.

5.3. Comparison with contemporary algorithms

We implemented some standard intelligence optimization algorithms, such as Genetic Algorithm [8], Discrete Gbest-guided Artificial Bee Colony (DGABC) [22], Standard WOA algorithm [90], Hybrid Genetic Algorithm (HGA) [23], and Greedy Randomized Adaptive Search Procedure (GRASP) [21] and compared the results with our algorithm.

In our assessments, few parameters are fixed to all algorithms that control the execution time and specify the amount of population. We set the population size as 30, and each experiment is executed continuously for 30 times, and the mean of each run has been noted duly. The weights used for each QoS attributes were: $w_C = 0.25$, $w_{Rs} = 0.25$, $w_A = 0.20$, $w_{Re} = 0.15$, $w_{Th} = 0.15$ based on user preferences and AHP with the MNV (mean of normalized values) method.

For DGABC [22], the limit value was set to 5, which means a food source that was not updated to 5 trails is abandoned by its employee bee. For GA [8], the parameters that have been set were crossover probability=0.7 and the mutation probability=0.05. The random selection procedure was used to select chromosomes for crossover and mutation. For HGA [23], the parameters that have been set in this algorithm were crossover probability = 0.9, the mutation probability = 0.20. The roulette wheel selection operator was used to select chromosomes for crossover and mutation. In GRASP [21], the parameters were set as: $\alpha = 0.25$, $\mu = 18$, $\delta = 8$, and other parameters as used in [21]. For WOA [90] the parameters were set as: Dim = 30, Lb = 1, Ub = 30, p = [0, 1], search agents = 30, Maxiter = 500. We have evaluated the performance of our proposed approaches in the following ways:

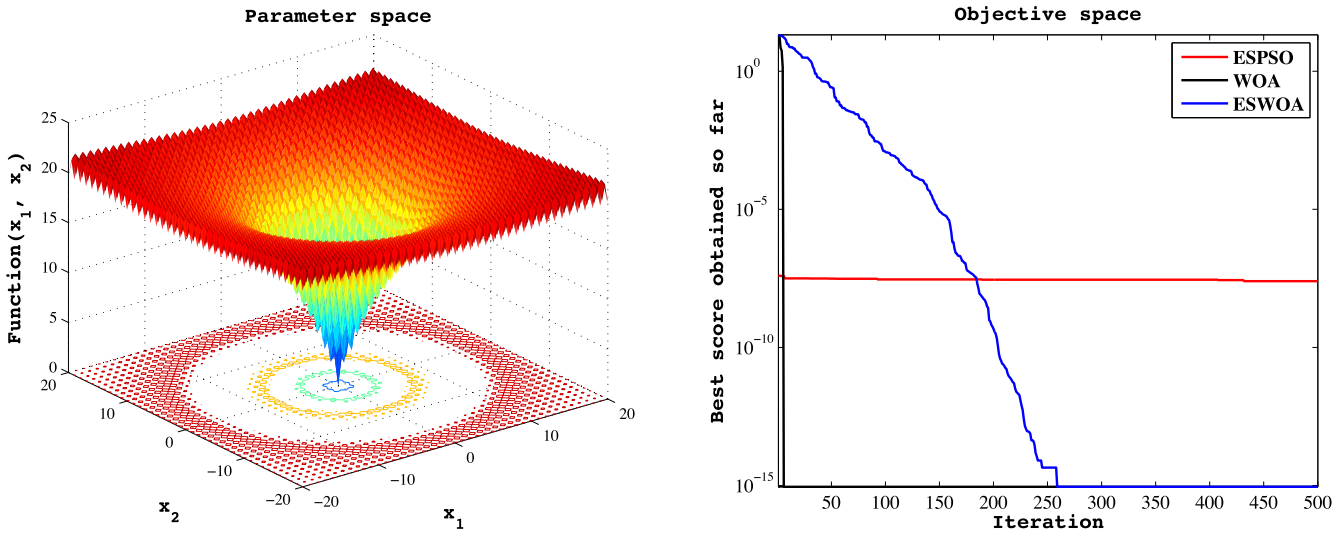

Fig. 10. Convergence curves for Function-F5.

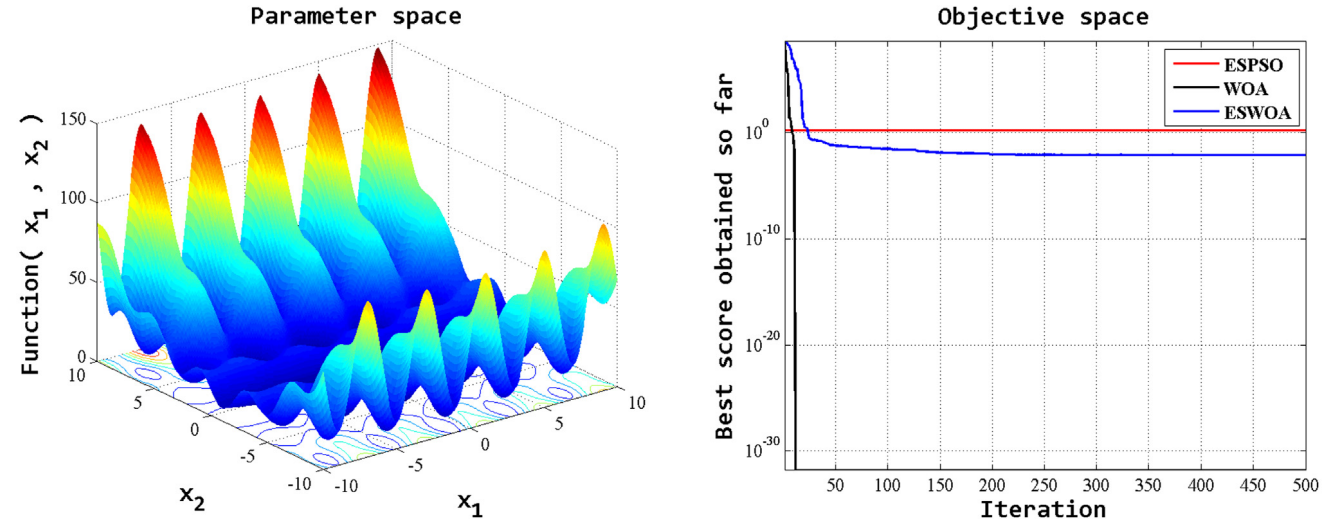

Fig. 11. Convergence curves for Function-F6.

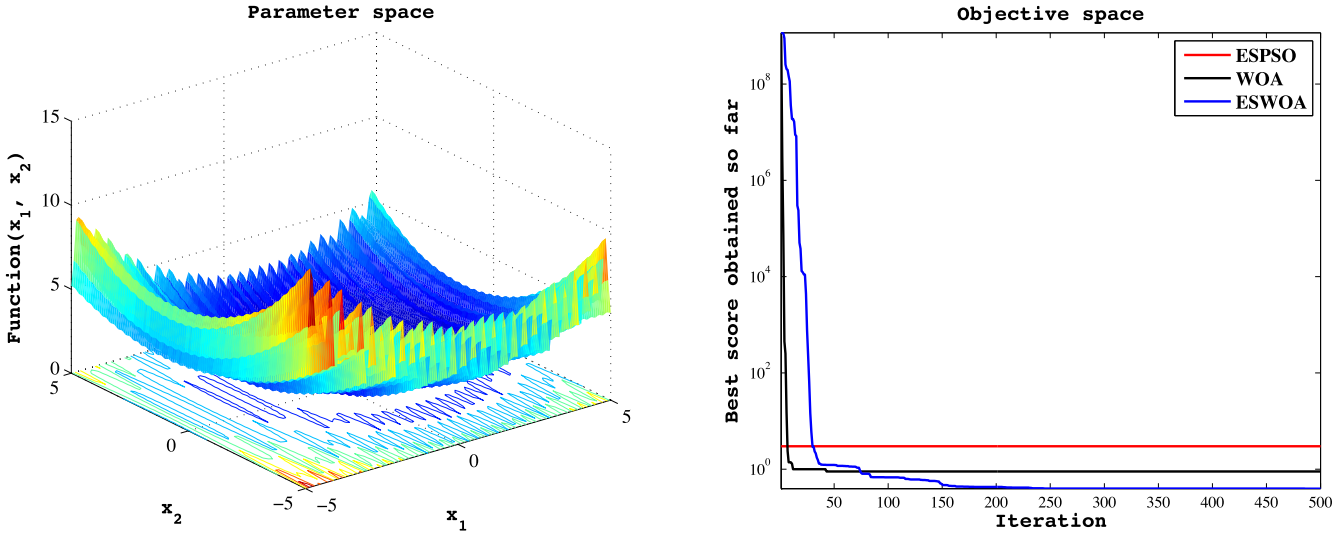

Fig. 12. Convergence curves for Function-F7.

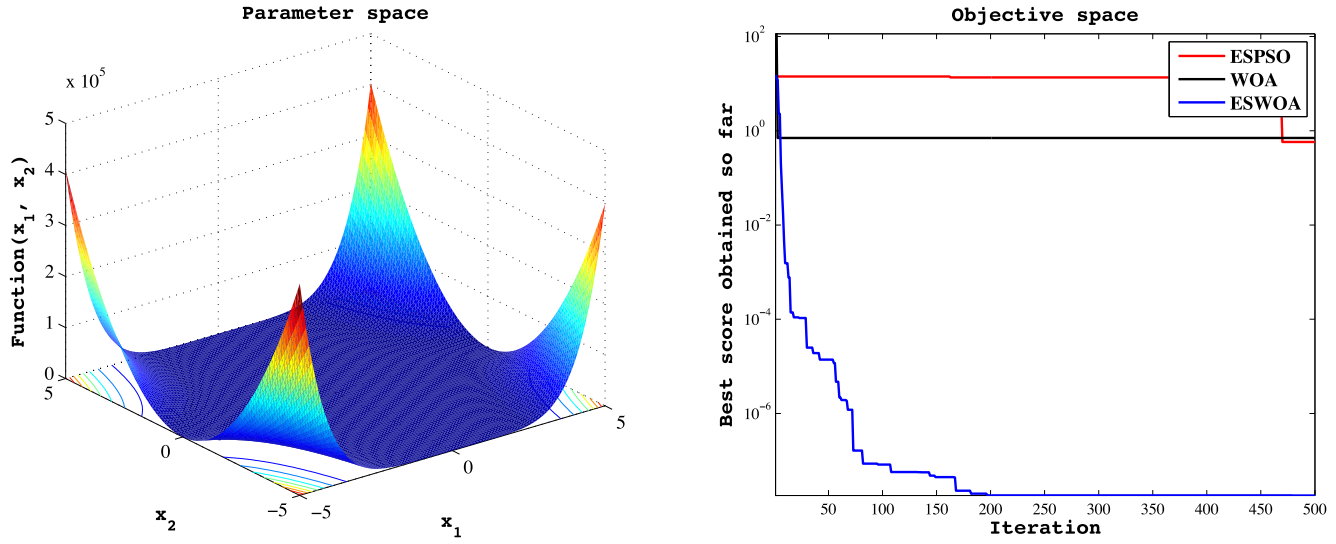

Fig. 13. Convergence curves for Function-F8.

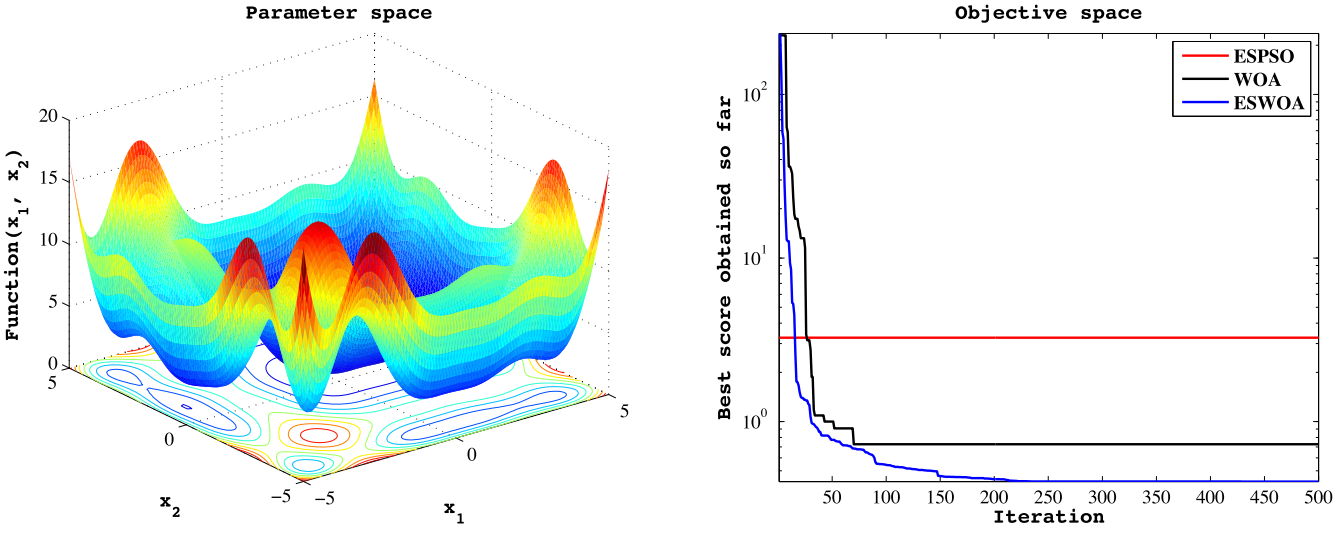

Fig. 14. Convergence curves for Function-F9.

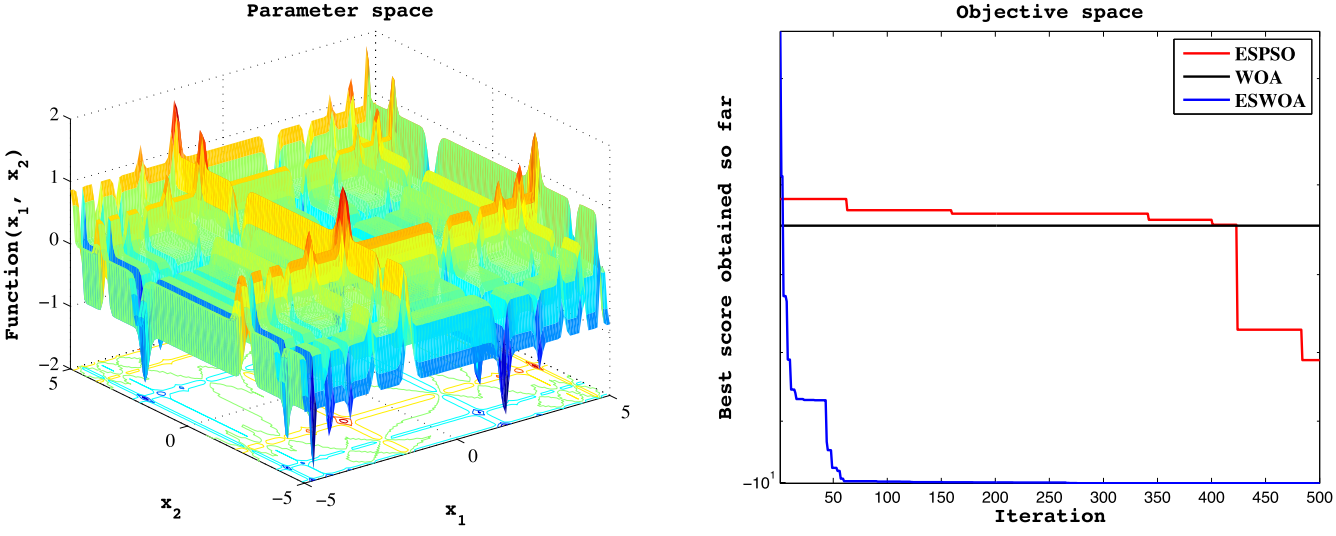

Fig. 15. Convergence curves for Function-F10.

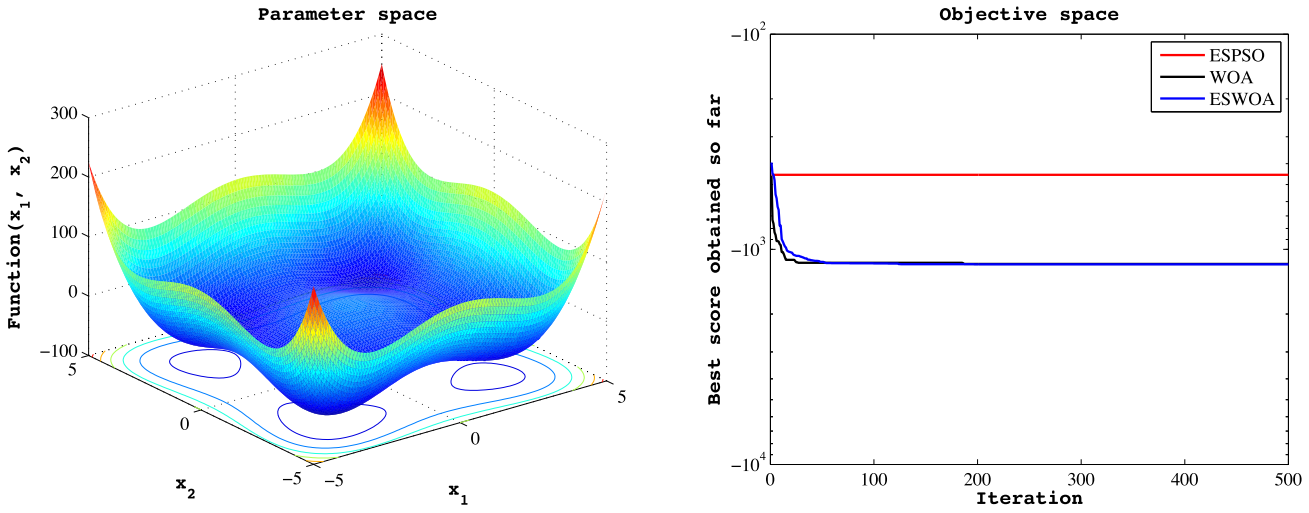

Fig. 16. Convergence curves for Function-F11.

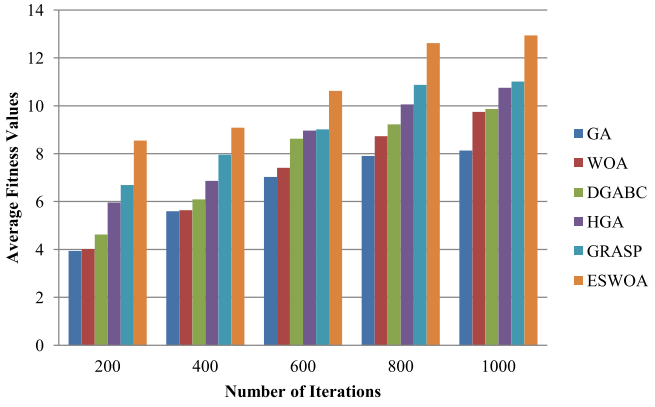

Fig. 17. Comparing Average Fitness values for varying number of iterations.

- Evaluate the average fitness values by varying number abstract and candidate services.
- Evaluate the average fitness values of our proposed algorithms against other compared algorithms in terms of iterations.

Figs. 18–21 illustrate a scenario where a composite service comprising of 25, 50, 75, and 100 abstract services, while for each abstract service, the number candidate services ranges as 25, 50, 75, and 100. From Figs. 18–21, we observed that the average fitness values increase with increase in the number of abstract services but does not affect the solution quality. From Fig. 18, the best average fitness values obtained by our proposed ESWOA for 100 candidate service is 10.08779, whereas the best solutions obtained by GA, WOA, DGABC, HGA, and GRASP are 5.59482, 5.6382, 6.08779, 7.9458, and 8.8941 respectively. From Fig. 19, the best average fitness values obtained by our proposed ESWOA for 100 candidate service is 11.62508, whereas the best solutions obtained by GA, WOA, DGABC, HGA, and GRASP are 7.0281, 7.41052, 8.62508, 9.0245, and 9.9521 respectively. As seen from Fig. 20, the solution obtained by our proposed ESWOA for 100 candidate service is 12.62128, whereas the best solutions obtained by GA, WOA, DGABC, HGA, and GRASP are 7.9038, 8.72924, 9.22749, 10.5631, and 10.986 respectively. From Fig. 21, the solution obtained by our proposed algorithm for 100 candidate service is 12.73428, whereas the best solutions obtained by GA, WOA, DGABC, HGA, and GRASP are 8.1316, 9.7418, 9.8674, 10.3486, 10.962 respectively. Thus, our proposed algorithm is more efficient than other compared algorithms. The details of remaining experiments (for varying combinations of abstract services and candidate services) are omitted, due to the similar experiments.

Table 12
Execution time (in seconds).

No of Services	GA	WOA	DGABC	HGA	GRASP	ESWOA
25	56.35	25.56	30.63	28.12	26.14	**20.56**
50	56.58	20.12	31.45	34.59	34.56	**25.15**
75	56.35	20.56	33.59	48.12	36.14	**29.56**
100	56.35	40.56	48.63	40.12	49.14	**31.56**

To ensure the robustness of our proposed ESWOA, we have executed all the said algorithms for multiple iterations such as 200, 400 and so on. It is to be noted that every iteration number shown in Fig. 17 had been run for 30 times and their mean is noted. This has been done for every method that we are comparing with our algorithm. Based on Fig. 17, we observed that our proposed ESWOA performs better than other algorithms (GA, WOA, DGABC, HGA, and GRASP) regarding the average best fitness value and convergence rate reached. ESWOA has the fastest convergence rate compared to GA, WOA, DGABC, HGA, and GRASP.

Table 12 presents the execution time of different methods for 25, 50, 75, and 100 abstract services with respect to 100 candidate services. From Table 12, the execution time for GA, WOA, DGABC, HGA, and GRASP to evaluate the best solutions were 56.35, 40.56, 48.63, 40.12, and 49.14 s, respectively, and the execution time for ESWOA was 31.56 s. From Table 12, we observe that our proposed method ESWOA is more efficient than the other methods compared.

In GA, HGA for each iteration, a new population is generated, and the individual fitnesses are assessed (based on fixed and predetermined crossover and unguided mutation). These processes cause GA and HGA to converge slowly and to become easily stuck in local maxima. It takes more time because of the higher number of parameter turnings and the single-point crossover. In DGABC, for each iteration, the initial population is generated. This population is repeated for the cycles of the search processes of the employees, onlooker, and scout bees respectively. However, this lacks a centralized processor to guide it towards good solutions, and the time for convergence remains uncertain [83,93]. In WOA, for each iteration, the search process starts with creating a set of random candidate solutions, and the individual fitnesses are evaluated

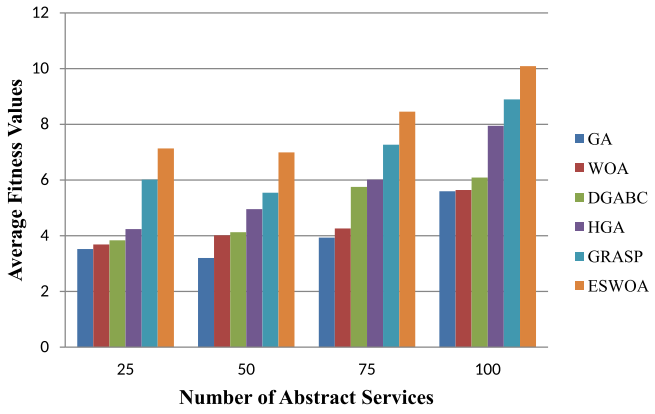

Fig. 18. Average fitness values for 25 abstract services.

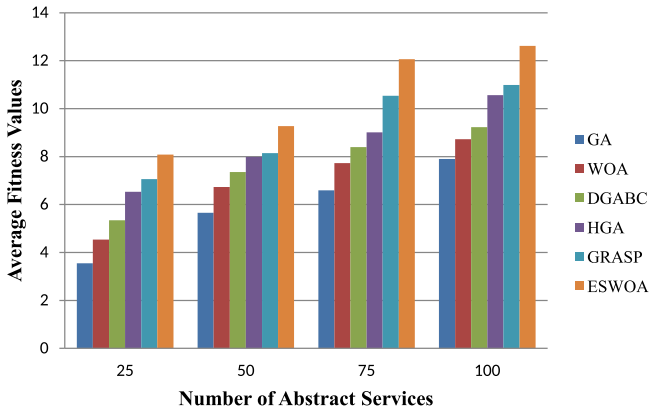

Fig. 20. Average fitness values for 75 abstract services.

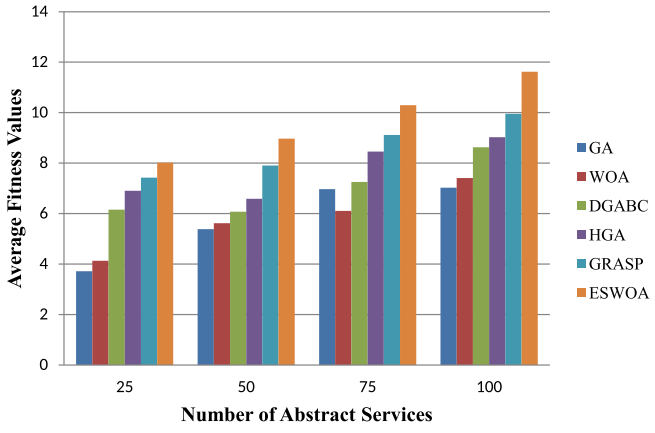

Fig. 19. Average fitness values for 50 abstract services.

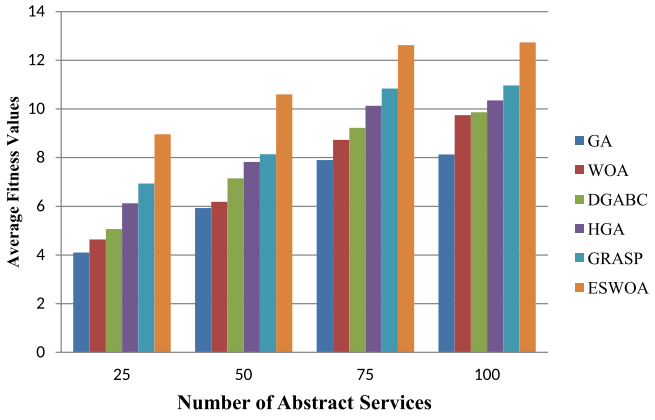

Fig. 21. Average fitness values for 100 abstract services.

(based on a random spiral path around preys and random bubble net attacking method) which gets changed in each iteration. The change decreases the premature convergence rate and increases the search space, which in turn, consumes more time. In GRASP, for each iteration, the new population generated based on greedy search and randomization mechanisms. Hence, each iteration produce a completely new solution, there is no information added to the system when a optimal solution is found. The change decreases the premature convergence rate and increases the search space, which in turn, consumes more time.

In our proposed method, a balance between exploration and exploitation is achieved, to overcome the issues like slow convergence rate or premature convergence to a decent degree. An approach used in exploitation phase of WOA is taken to perform local search in our ESWOA, whereas for exploration, a new method has been proposed in this context. Experiment results show that ESWOA has potential to reach global optimal solutions more seamlessly compared to other contemporary algorithms.

5.4. Statistical analysis

We have performed statistical analysis tests to ensure that the results of our proposed algorithm are statistically significant or not by using parametric and nonparametric tests [94]. Tables 13–15 illustrates the *p*-values computed by the Friedman [95], Friedman Aligned [96], and Quade tests [97] for 25 and 100 abstract services with 100 candidate services. We have performed Bonferroni test [98], Holm test [99], and Holland test [100] to perform the multiple statistical comparison of all algorithms. Based on the

Table 13
Adjusted *p*-value for Friedman test for 25 and 100 abstract services.

	Friedman	Unadjusted	Bonferroni	Holm	Holland
25	GA	0.000314	0.06844	0.07312	0.00234
	WOA	0.086138	0.18132	0.16813	0.05857
	DGABC	0.08638	0.08181	0.07110	0.07110
	GRASP	0.10380	0.15318	0.14913	0.14813
	ESWOA	0.8318	1.0	1.0	0.66418
	Friedman	Unadjusted	Bonferroni	Holm	Holland
100	GA	0.000816	0.07148	0.07813	0.00418
	WOA	0.15813	0.19214	0.18138	0.08161
	DGABC	0.08861	0.09618	0.07919	0.06984
	GRASP	0.20804	0.19498	0.18698	0.17981
	ESWOA	0.92801	1.0	1.0	0.88148

experimental results, we observe that the differences between our approach and GA, WOA, DGABC, HGA, and GRASP are statistically significant.

Table 14 shows a significant improvement of HGA over GA, WOA, DGABC, and GRASP for Holland and Holm methods, except for the Bonferroni–Dunn one for Friedman Aligned test. The Bonferroni–Dunn exhibit the most powerful behavior, reaching the lowest *p*-values in the comparisons.

Table 15 presents a significant improvement of HGA over GA, WOA, DGABC, and GRASP for Holland and Holm methods, except for the Bonferroni–Dunn one for Quade test. The Bonferroni–Dunn shows the most powerful behavior in this category. Similarly, we have conducted experiments on ESWOA over other compared algorithms. Based on the experimental results we observed that our proposed approach statistically significant than other algorithms.

Table 14
Adjusted p-value for Friedman Aligned test for 25 and 100 abstract services.

	Friedman Aligned	Unadjusted	Bonferroni	Holm	Holland
	GA	0.002178	0.08491	0.0791	0.057514
	WOA	0.09913	0.28131	0.2541	0.21636
25	DGABC	0.08389	0.18193	0.17943	0.16918
	GRASP	0.20183	0.19034	0.18931	0.17039
	EAWOA	0.84734	1.0	1.0	0.56673
	Friedman Aligned	Unadjusted	Bonferroni	Holm	Holland
	GA	0.002178	0.08498	0.0796	0.05715
	WOA	0.09918	0.28136	0.2452	0.2361
100	DGABC	0.08390	0.18194	0.17936	0.16920
	GRASP	0.20186	0.19036	0.18936	0.18321
	EAWOA	0.9403	1.0	1.0	0.59966

Table 15
Adjusted p-value for Quade test for 25 and 100 abstract services.

	Quade	Unadjusted	Bonferroni	Holm	Holland
	GA	0.02181	0.18321	0.18321	0.1218
	WOA	0.06284	0.38435	0.40181	0.32181
25	DGABC	0.11961	0.41038	0.9608	0.86432
	GRASP	0.16813	0.48310	1.0	0.80181
	EAWOA	0.98408	1.0	1.0	0.78432
	Quade	Unadjusted	Bonferroni	Holm	Holland
	GA	0.02981	0.20181	0.20181	0.1296
	WOA	0.06484	0.39612	0.39612	0.3301
100	DGABC	0.1282	0.41181	0.42016	0.3301
	GRASP	0.18816	0.49162	1.0	0.8143
	EAWOA	0.9951	1.0	1.0	0.7892

The details of remaining experiments (for varying combinations of abstract services and candidate services) are omitted, due to the similar nature of experiments.

6. Conclusions

Cloud computing delivers IT enables capacities in the form of services to the consumers in an on-demand manner. Due to the seamless proliferation of cloud services delivering with varying quality of service levels. Consumers often find it difficult to select an optimal cloud service to meet their requirements. Hence, choosing the service composition, we need to find a near-optimal solution that satisfies both functional and non-functional requirements. In this paper, Eagle Strategy is used for designing QoS-aware Cloud service composition. By using this, a balance between exploration and exploitation is achieved, to overcome the issues like slow convergence rate or premature convergence to a decent degree. Experiment results show that ESWOA achieves global optimal solutions more seamlessly compared to other contemporary algorithms.

In future, we wish to apply our algorithm for considering the interdependencies and correlations between the cloud services along with QoS factors. In this paper, we assumed that all services come from the same repository. Our future work would be considering multiple service repositories from which composition of services needs to be done by minimizing the communication between the number of clouds.

References

[1] M. Cusumano, Cloud computing and SaaS as new computing platforms, Commun. ACM 53 (4) (2010) 27–29.
[2] R. Buyya, J. Broberg, A.M. Goscinski, Cloud Computing: Principles and Paradigms, in: Wiley Series on Parallel and Distributed Computing, John Wiley, 2010.
[3] M. Luo, L.-J. Zhang, F. Lei, An insuanrance model for guranteeing service assurance, integrity and QoS in cloud computing, in: Proceedings of the IEEE International Conference on Web Services, ICWS, IEEE, 2010, pp. 584–591.
[4] D. Ardagna, B. Pernici, Adaptive service composition in flexible processes, IEEE Trans. Softw. Eng. 33 (6) (2007).
[5] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2003) 268–308.
[6] X.-S. Yang, M. Karamanoglu, T.O. Ting, Y.-X. Zhao, Applications and analysis of bio-inspired eagle strategy for engineering optimization, Neural Comput. Appl. 25 (2) (2014) 411–420.
[7] F. Neri, Diversity management in memetic algorithms, in: Handbook of Memetic Algorithms, Springer, 2012, pp. 153–165.
[8] G. Canfora, M. Di Penta, R. Esposito, M.L. Villani, An approach for QoS-aware service composition based on genetic algorithms, in: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, ACM, 2005, pp. 1069–1075.
[9] M. Tang, L. Ai, A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition, in: Proceedings of the 2010 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2010, pp. 1–8.
[10] Y. Ma, C. Zhang, Quick convergence of genetic algorithm for QoS-driven web service selection, Comput. Netw. 52 (5) (2008) 1093–1104.
[11] Z.-p. Gao, C. Jian, X.-s. Qiu, L.-m. Meng, QoE/QoS driven simulated annealing-based genetic algorithm for web services selection, J. China Univ. Posts Telecommun. 16 (2009) 102–107.
[12] C. Gao, M. Cai, H. Chen, QoS-aware service composition based on tree-coded genetic algorithm, in: Proceedings of the 31st Annual Intl. Conf. Computer Software & Applications, COMPSAC, vol. 1, 2007, pp. 361–367.
[13] Y. Yu, H. Ma, M. Zhang, An adaptive genetic programming approach to QoS-aware web services composition, in: Proceedings of the IEEE Cong. on Evolu. Comp., CEC, 2013, pp. 1740–1747.
[14] E. Yilmaz, P. Karagoz, Improved genetic algorithm based approach for QoS aware web service composition, in: Proceedings of the 2014 IEEE Intl. Conf. on Web Services, ICWS, IEEE, 2014, pp. 463–470.
[15] L. Bao, F. Zhao, M. Shen, Y. Qi, P. Chen, An orthogonal genetic algorithm for QoS-aware service composition, Comput. J. (2016), http://dx.doi.org/10.1093/comjnl/bxw043.
[16] Q. Wu, F. Ishikawa, Q. Zhu, D.-H. Shin, QoS-aware multigranularity service composition: Modeling and optimization, IEEE Trans. Syst., Man, Cybern.: Syst. 46 (11) (2016) 1565–1577.
[17] J. Chandrashekar, G.R. Gangadharan, B. Rajkumar, Computational intelligence based QoS-aware web service composition: A systematic literature review, IEEE Trans. Serv. Comput. 10 (3) (2017) 475–492.
[18] W. Li, H. Yan -xiang, Web service composition based on QoS with chaos particle swarm optimization, in: Proceedings of the 6th Intl. Conf. on Wireless Communications Networking & Mobile Comp., 2010, pp. 1–4.
[19] Y. Liu, H. Miao, Z. Li, H. Gao, QoS-aware web services composition based on HQPSO algorithm, in: Proceedings of the First ACIS/JNU Intl. Conf. on Computers, Networks, Systems and Industrial Engineering, 2011, pp. 400–405.
[20] X. Zhao, B. Song, P. Huang, Z. Wen, J. Weng, Y. Fan, An improved discrete immune optimization algorithm based on PSO for QoS-driven web service composition, Appl. Soft Comput. 12 (8) (2012) 2208–2216.
[21] J.A. Parejo, S. Segura, P. Fernandez, A. Ruiz -Cortés, Qos-aware web services composition using grasp with path relinking, Expert Syst. Appl. 41 (9) (2014) 4211–4223.
[22] Y. Huo, Y. Zhuang, J. Gu, S. Ni, Y. Xue, Discrete gbest-guided artificial bee colony algorithm for cloud service composition, Appl. Intell. 42 (4) (2015) 661–678.
[23] F. Seghir, A. Khababa, A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition, J. Intell. Manuf. (2016) 1–20.
[24] M.B. Karimi, A. Isazadeh, A.M. Rahmani, QoS-aware service composition in cloud computing using data mining techniques and genetic algorithm, J. Supercomput. 73 (4) (2017) 1387–1415.
[25] Z.-Z. Liu, D.-H. Chu, C. Song, X. Xue, B.-Y. Lu, Social Learning Optimization, SLO algorithm paradigm and its application in QoS-aware cloud service composition, Inform. Sci. 326 (2016) 315–333.
[26] A. Younes, M. Essaaidi, A. El Moussaoui, SFL algorithm for QoS-based cloud service composition, Int. J. Comput. Appl. 97 (17) (2014) 42–49.
[27] F. Seghir, A. Khababa, J. Gaber, A. Chariete, P. Lorenz, A new discrete imperialist competitive algorithm for QoS-aware service composition in cloud computing, in: Proceedings of the International Symposium on Intelligent Systems Technologies and Applications, Springer, 2016, pp. 339–353.
[28] M. Zhang, L. Liu, S. Liu, Genetic algorithm based QoS-aware service composition in multi-cloud, in: Proceedings of the 2015 IEEE Conference on Collaboration and Internet Computing, IEEE, 2015, pp. 113–118.
[29] Q. Yu, L. Chen, B. Li, Ant colony optimization applied to web service compositions in cloud computing, Comput. Electr. Eng. 41 (2015) 18–27.
[30] H. Kurdi, A. Al -Anazi, C. Campbell, A. Al Faries, A combinatorial optimization algorithm for multiple cloud service composition, Comput. Electr. Eng. 42 (2015) 107–113.
[31] A. Jula, Z. Othman, E. Sundararajan, Imperialist competitive algorithm with Proclus classifier for service time optimization in cloud computing service composition, Expert Syst. Appl. 42 (1) (2015) 135–145.

[32] A. Jula, Z. Othman, E. Sundararajan, A hybrid imperialist competitive-gravitational attraction search algorithm to optimize cloud service composition, in: Proceedings of the 2013 IEEE Workshop on Memetic Computing, IEEE, 2013, pp. 37–43.
[33] D. Wang, Y. Yang, Z. Mi, A genetic-based approach to web service composition in geo-distributed cloud environment, Comput. Electr. Eng. 43 (2015) 129–141.
[34] G.-G. Wang, A.H. Gandomi, A.H. Alavi, D. Gong, A comprehensive review of krill herd algorithm: Variants hybrids and applications, Artif. Intell. Rev. (2017) 1–30.
[35] A.H. Gandomi, A.H. Alavi, Krill herd: A new bio-inspired optimization algorithm, Commun. Nonlinear Sci. Numer. Simul. 17 (12) (2012) 4831–4845.
[36] G.-G. Wang, A.H. Gandomi, A.H. Alavi, A chaotic particle-swarm krill herd algorithm for global numerical optimization, Kybernetes 42 (6) (2013) 962–978.
[37] X.-S. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, 2010.
[38] G.-G. Wang, L. Guo, A.H. Gandomi, G.-S. Hao, H. Wang, Chaotic krill herd algorithm, Inform. Sci. 274 (2014) 17–34.
[39] G.-G. Wang, S. Deb, A.H. Gandomi, Z. Zhang, A.H. Alavi, Chaotic cuckoo search, Soft Comput. 20 (9) (2016) 3349–3362.
[40] L. Guo, G.-G. Wang, A.H. Gandomi, A.H. Alavi, H. Duan, A new improved krill herd algorithm for global numerical optimization, Neurocomputing 138 (2014) 392–402.
[41] G.-G. Wang, L. Guo, H. Duan, H. Wang, A new improved firefly algorithm for global numerical optimization, J. Comput. Theor. Nanosci. 11 (2) (2014) 477–485.
[42] G.-G. Wang, A.H. Gandomi, A.H. Alavi, S. Deb, A hybrid method based on krill herd and quantum-behaved particle swarm optimization, Neural Comput. Appl. 27 (4) (2016) 989–1006.
[43] G.-G. Wang, A.H. Gandomi, A.H. Alavi, G.-S. Hao, Hybrid krill herd algorithm with differential evolution for global numerical optimization, Neural Comput. Appl. 25 (2) (2014) 297–308.
[44] G.-G. Wang, Moth search algorithm: A bio-inspired metaheuristic algorithm for global optimization problems, Memet. Comput. (2016) 1–14.
[45] G.-G. Wang, A.H. Gandomi, A.H. Alavi, S. Deb, A multi-stage krill herd algorithm for global numerical optimization, Int. J. Artif. Intell. Tools 25 (02) (2016) 1550030.
[46] G. Wang, L. Guo, A.H. Gandomi, L. Cao, A.H. Alavi, H. Duan, J. Li, Lévy-flight krill herd algorithm, Math. Probl. Eng. 2013 (2013).
[47] G.-G. Wang, L. Guo, A.H. Gandomi, A.H. Alavi, H. Duan, Simulated annealing-based krill herd algorithm for global optimization, in: Abstract and Applied Analysis, vol. 2013, Hindawi, 2013.
[48] G.-G. Wang, A.H. Gandomi, A.H. Alavi, Stud krill herd algorithm, Neurocomputing 128 (2014) 363–370.
[49] G.-G. Wang, A.H. Gandomi, A.H. Alavi, Study of Lagrangian and evolutionary parameters in krill herd algorithm, in: Adaptation and Hybridization in Computational Intelligence, Springer, 2015, pp. 111–128.
[50] G. Wang, L. Guo, H. Wang, H. Duan, L. Liu, J. Li, Incorporating mutation scheme into krill herd algorithm for global numerical optimization, Neural Comput. Appl. 24 (3–4) (2014) 853–871.
[51] G.-G. Wang, A.H. Gandomi, A.H. Alavi, An effective krill herd algorithm with migration operator in biogeography-based optimization, Appl. Math. Model. 38 (9–10) (2014) 2454–2462.
[52] H. Wang, J.-H. Yi, An improved optimization method based on krill herd and artificial bee colony with information exchange, Memet. Comput. (2017) 1–22.
[53] G.-G. Wang, S. Deb, Z. Cui, Monarch butterfly optimization, Neural Comput. Appl. (2015).
[54] G.-G. Wang, S. Deb, X. Zhao, Z. Cui, A new monarch butterfly optimization with an improved crossover operator, Oper. Res. (2016) 1–25.
[55] Y. Feng, G.-G. Wang, S. Deb, M. Lu, X.-J. Zhao, Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization, Neural Comput. Appl. 28 (7) (2017) 1619–1634.
[56] G.-G. Wang, A. Hossein Gandomi, X.-S. Yang, A. Hossein Alavi, A novel improved accelerated particle swarm optimization algorithm for global numerical optimization, Eng. Comput. 31 (7) (2014) 1198–1220.
[57] G.-G. Wang, A.H. Gandomi, X. Zhao, H.C.E. Chu, Hybridizing harmony search algorithm with cuckoo search for global numerical optimization, Soft Comput. 20 (1) (2016) 273–285.
[58] Z. Cui, B. Sun, G. Wang, Y. Xue, J. Chen, A novel oriented cuckoo search algorithm to improve DV-Hop performance for cyber–physical systems, J. Parallel Distrib. Comput. 103 (2017) 42–52.
[59] G. Wang, L. Guo, H. Duan, H. Wang, L. Liu, M. Shao, Hybridizing harmony search with biogeography based optimization for global numerical optimization, J. Comput. Theor. Nanosci. 10 (10) (2013) 2312–2322.
[60] G.-G. Wang, H.E. Chu, S. Mirjalili, Three-dimensional path planning for UCAV using an improved bat algorithm, Aerosp. Sci. Technol. 49 (2016) 231–238.
[61] W. Guohua, P. Witold, P.N. Suganthan, L. Haifeng, Using variable reduction strategy to accelerate evolutionary optimization, Appl. Soft Comput. 61 (2017) 283–293.
[62] R.M. Rizk -Allah, R.A. El -Sehiemy, S. Deb, G.-G. Wang, A novel fruit fly framework for multi-objective shape design of tubular linear synchronous motor, J. Supercomput. 73 (3) (2017) 1235–1256.
[63] R.M. Rizk -Allah, R.A. El -Sehiemy, G.-G. Wang, A novel parallel hurricane optimization algorithm for secure emission/economic load dispatch solution, Appl. Soft Comput. 63 (2018) 206–222.
[64] K. Liu, D. Gong, F. Meng, H. Chen, G.-G. Wang, Gesture segmentation based on a two-phase estimation of distribution algorithm, Inform. Sci. 394 (2017) 88–105.
[65] G. Wu, X. Shen, H. Li, H. Chen, A. Lin, P. Suganthan, Ensemble of differential evolution variants, Inform. Sci. 423 (2018) 172–186.
[66] G.-G. Wang, X. Cai, Z. Cui, G. Min, J. Chen, High performance computing for cyber physical social systems by using evolutionary multi-objective optimization algorithm, IEEE Trans. Emerg. Top. Comput. (2017).
[67] G. Wu, W. Pedrycz, H. Li, M. Ma, J. Liu, Coordinated planning of heterogeneous earth observation resources, IEEE Trans. Syst., Man, Cybern.: Syst. 46 (1) (2016) 109–125.
[68] J.W. Zhang, G.G. Wang, Image matching using a bat algorithm with mutation, in: Applied Mechanics and Materials, vol. 203, Trans Tech Publ, 2012, pp. 88–93.
[69] R. Wang, R.C. Purshouse, P.J. Fleming, Preference-inspired coevolutionary algorithms for many-objective optimization, IEEE Trans. Evol. Comput. 17 (4) (2013) 474–494.
[70] G.-G. Wang, M. Lu, Y.-Q. Dong, X.-J. Zhao, Self-adaptive extreme learning machine, Neural Comput. Appl. 27 (2) (2016) 291–303.
[71] G.-G. Wang, Y. Tan, Improving metaheuristic algorithms with information feedback models, IEEE Trans. Cybern. (2017).
[72] G.-G. Wang, A.H. Gandomi, X.-S. Yang, A.H. Alavi, A new hybrid method based on krill herd and cuckoo search for global optimisation tasks, Int. J. Bio-Inspired Comput. 8 (5) (2016) 286–299.
[73] Y. Feng, G.-G. Wang, Binary moth search algorithm for discounted 0-1 knapsack problem, IEEE Access 6 (2018) 10708–10719.
[74] G.-G. Wang, S. Deb, X.-Z. Gao, L.D.S. Coelho, A new metaheuristic optimisation algorithm motivated by elephant herding behaviour, Int. J. Bio-Inspired Comput. 8 (6) (2016) 394–409.
[75] G.-G. Wang, S. Deb, L. Coelho, Earthworm optimization algorithm: A bio-inspired metaheuristic algorithm for global optimization problems, Int. J. Bio-Inspired Comput. (2015), http://dx.doi.org/10.1504/IJBIC.2015.10004283.
[76] Y. Feng, J. Yang, C. Wu, M. Lu, X.-J. Zhao, Solving 0–1 knapsack problems by chaotic monarch butterfly optimization algorithm with Gaussian mutation, Memet. Comput. (2016) 1–16.
[77] J.-H. Yi, J. Wang, G.-G. Wang, Improved probabilistic neural networks with self-adaptive strategies for transformer fault diagnosis problem, Adv. Mech. Eng. 8 (1) (2016) 1–3.
[78] R. Wang, Q. Zhang, T. Zhang, Decomposition-based algorithms using Pareto adaptive scalarizing methods, IEEE Trans. Evol. Comput. 20 (6) (2016) 821–837.
[79] R. Wang, Z. Zhou, H. Ishibuchi, T. Liao, T. Zhang, Localized weighted sum method for many-objective optimization, IEEE Trans. Evol. Comput. 22 (1) (2016) 3–18.
[80] G. Wu, Across neighborhood search for numerical optimization, Inform. Sci. 329 (2016) 597–618.
[81] G.-G. Wang, A.H. Gandomi, A.H. Alavi, Y.-Q. Dong, A hybrid meta-heuristic method based on firefly algorithm and krill herd, in: Handbook of Research on Advanced Computational Techniques for Simulation-Based Engineering, IGI Global, 2016, pp. 505–524.
[82] J. Chandrashekar, G.R. Gangadharan, QoS-aware web service composition using quantum inspired particle swarm optimization, in: Proceedings of the 7th International KES Conference on Intelligent Decision Technologies, Springer, 2015, pp. 255–265.
[83] C. Jatoth, G.R. Gangadharan, U. Fiore, R. Buyya, QoS-aware big service composition using mapreduce based evolutionary algorithm with guided mutation, Future Gener. Comput. Syst. (2017), http://dx.doi.org/10.1016/j.future.2017.07.042.
[84] X.-S. Yang, S. Deb, Eagle strategy using Lévy walk and firefly algorithms for stochastic optimization, in: Nature Inspired Cooperative Strategies for Optimization, NICSO 2010, Springer, 2010, pp. 101–111.
[85] J.A. Nelder, R. Mead, A simplex method for function minimization, Comput. J. 7 (4) (1965) 308–313.
[86] X.-S. Yang, Firefly algorithms for multimodal optimization, in: Proceedings of the International Symposium on Stochastic Algorithms, Springer, 2009, pp. 169–178.
[87] R. Jia, D. He, Artificial bee colony algorithm with two-stage eagle strategy, in: Proceedings of the 9th International Conference on Computational Intelligence and Security, CIS, IEEE, 2013, pp. 16–20.
[88] X.-S. Yang, S. Deb, S. Fong, Metaheuristic algorithms: Optimal balance of intensification and diversification, Appl. Math. 8 (3) (2014) 1–7.
[89] X.-S. Yang, S. Deb, X. He, Eagle strategy with flower algorithm, in: Proceedings of the International Conference on Advances in Computing, Communications and Informatics, IEEE, 2013, pp. 1213–1217.

[90] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67.
[91] E. Al -Masri, Q.H. Mahmoud, Qos-based discovery and ranking of web services, in: Proceedings of 16th International Conference on Computer Communications and Networks, IEEE, 2007, pp. 529–534.
[92] H. Yapıcı, N. Çetinkaya, An improved particle swarm optimization algorithm using eagle strategy for power loss minimization, Math. Probl. Eng. 2017 (2017).
[93] M.N. Ab Wahab, S. Nefti -Meziani, A. Atyabi, A comprehensive review of swarm optimization algorithms, PLoS One 10 (5) (2015) e0122827.
[94] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evolutionary Comput. 1 (1) (2011) 3–18.
[95] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Amer. Statist. Assoc. 32 (200) (1937) 675–701.
[96] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Ann. Math. Stat. 11 (1) (1940) 86–92.
[97] D. Quade, Using weighted rankings in the analysis of complete blocks with additive block effects, J. Amer. Statist. Assoc. 74 (367) (1979) 680–683.
[98] O.J. Dunn, Multiple comparisons among means, J. Amer. Statist. Assoc. 56 (293) (1961) 52–64.
[99] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. (1979) 65–70.
[100] B.S. Holland, M.D. Copenhaver, An improved sequentially rejective bonferroni test procedure, Biometrics (1987) 417–423.

Siva Kumar Gavvala received his B.Tech. in Information Technology from Sastra University and M.Tech. in Information Technology from University of Hyderabad, Hyderabad, India in 2014 and 2017 respectively. Currently, he is currently working as Assistant Manager at National Payments Corporation of India (NPCI), Hyderabad, India. His research interests focus on QoS, service composition, and computational intelligence techniques.

Chandrashekar Jatoth received his B.E. in Information Technology from Osmania University and M.Tech. in Artificial Intelligence from University of Hyderabad, Hyderabad, India in 2008 and 2010 respectively. Currently, he is working towards the Ph.D. degree in University of Hyderabad and Institute for Development and Research in Banking Technology (IDRBT), Hyderabad, India. His research interests focus on QoS, service composition, and computational intelligence techniques.

G.R. Gangadharan is an Associate professor at the Institute for Development and Research in Banking Technology, Hyderabad, India. His research interests focus on the interface between technological and business perspectives. Gangadharan received his Ph.D. in information and communication technology from the University of Trento, Italy, and the European University Association. He is a senior member of IEEE and ACM. Contact him at geeyaar@gmail.com.

Rajkumar Buyya is a Fellow of IEEE, Professor of Computer Science and Software Engineering, Future Fellow of the Australian Research Council, and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft Pty Ltd., a spin-off company of the University, commercializing its innovations in Grid and Cloud Computing. Dr. Buyya has authored/co-authored over 450 publications. He is one of the highly cited authors in computer science and software engineering worldwide. Microsoft Academic Search Index ranked Dr. Buyya as one of the Top 5 Authors during the last 10 years (2001–2012) and #1 in the world during the last 5 years (2007–2012) in the area of Distributed and Parallel Computing. For further information on Dr. Buyya, please visit: http://www.buyya.com.