Computer Systems Science & Engineering
DOI: 10.32604/csse.2023.037692
Article

Hybridizing Artificial Bee Colony with Bat Algorithm for Web Service Composition

Tariq Ahamed Ahanger[1,*], Fadl Dahan[2,3] and Usman Tariq[1]

[1]Department of Management Information Systems, College of Business Administration, Prince Sattam Bin Abdulaziz University, Al-Kharj, 11942, Saudi Arabia
[2]Department of Management Information Systems, College of Business Administration-Hawtat Bani Tamim, Prince Sattam Bin Abdulaziz University, Al-Kharj, 11942, Saudi Arabia
[3]Department of Computer Sciences, Faculty of Computing and Information Technology Al-Turbah, Taiz University, Taiz, 9674, Yemen
*Corresponding Author: Tariq Ahamed Ahanger. Email: t.ahanger@psau.edu.sa
Received: 14 November 2022; Accepted: 29 December 2022

Abstract: In the Internet of Things (IoT), the users have complex needs, and the Web Service Composition (WSC) was introduced to address these needs. The WSC's main objective is to search for the optimal combination of web services in response to the user needs and the level of Quality of Services (QoS) constraints. The challenge of this problem is the huge number of web services that achieve similar functionality with different levels of QoS constraints. In this paper, we introduce an extension of our previous works on the Artificial Bee Colony (ABC) and Bat Algorithm (BA). A new hybrid algorithm was proposed between the ABC and BA to achieve a better tradeoff between local exploitation and global search. The bat agent is used to improve the solution of exhausted bees after a threshold (limits), and also an Elitist Strategy (ES) is added to BA to increase the convergence rate. The performance and convergence behavior of the proposed hybrid algorithm was tested using extensive comparative experiments with current state-of-the-art nature-inspired algorithms on 12 benchmark datasets using three evaluation criteria (average fitness values, best fitness values, and execution time) that were measured for 30 different runs. These datasets are created from real-world datasets and artificially to form different scale sizes of WSC datasets. The results show that the proposed algorithm enhances the search performance and convergence rate on finding the near-optimal web services combination compared to competitors. The Wilcoxon signed-rank significant test is used where the proposed algorithm results significantly differ from other algorithms on 100% of datasets.

Keywords: Internet of things; artificial bee colony; bat algorithm; elitist strategy; web service composition

1 Introduction

The service-oriented architecture represents the user needs as business processes and uses a composition of reusable Web Services (WSs) based on the standard protocols [1]. In case of these needs are complex, a set of WSs are integrated to fulfill them. The task of integrating the WSs and optimizing their selection based on the user needs with the different QoS constraints is called web service composition (WSC). To this end, different WSs are retrieved from different providers, producing WSs achieve similar functional and non-functional constraints, such as quality of services (QoS).

In WSC, the clients submit their needs, represented as workflow (business processes) containing a set of tasks (n); each task represents a specific task of these needs. Then, a set of WSs (m) are retrieved according to the functionality of the task with different QoS features. As a result, The WSC has m^n possible solutions, so it is represented as an NP-hard problem [2]. Fig. 1 shows the model of the WSC problem.

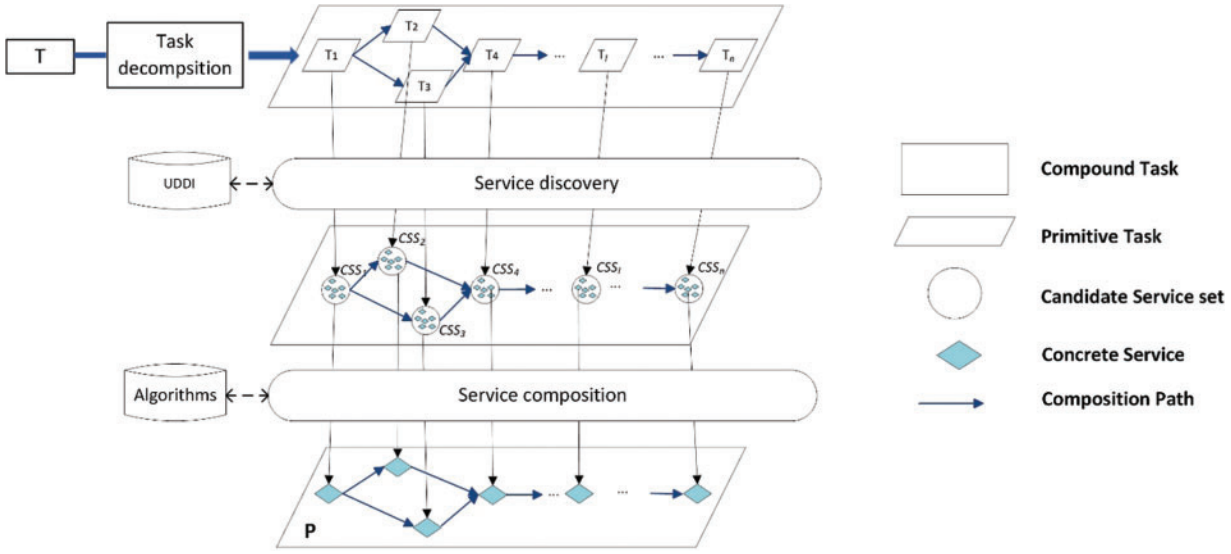

Figure 1: The WSC problem representative model

To solve the WSC problem, different nature-inspired improvements were proposed, such as Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), artificial bee colony (ABC), Whale Optimization Algorithm (WOA), bat algorithm (BA), Cuckoo Search (CS),.. etc. The main concerns on these improvements are convergence rate and execution time to search for the best solutions while the available WSs/task increases.

In the Internet of things (IoT), the users have complex needs, and the WSC problem was introduced to address these needs, especially since these needs increased while the disease outbreak [3,4]. The search for optimal WSs combination to satisfy these needs requires high computation time that increases while the WSC scale increases. In this work, we introduce an extension of our previous works on improving the ABC [5–7] and BA [8]. A new hybrid algorithm was proposed between the ABC and BA to achieve a better tradeoff between local exploitation and global search. The bat agent is used to improve the solution of exhausted bees after a threshold (limits), and also an elitist strategy (ES) is added to BA to increase the convergence rate.

The paper is organized as follows: Related work is explained in Section 2. The basic algorithms background and the proposed hybrid algorithm are presented in Section 3. The experiment and analysis are represented in Section 4. Section 5 describes the paper's conclusion.

2 Related Work

Different nature-inspired approaches have been used to address the WSC problem and obtain the near-optimal web service combination. In the following sub-section, a review of the state-of-the-art research was introduced.

2.1 ABC Related Work

Seghir [9] proposed an enhancement to the ABC algorithm based on the fuzzy distance and ranking methods. This enhancement aims to enhance the search diversity of the bees. The neighborhood search is introduced in [10] to enhance the ABC search, and the opposition learning method is applied to initialize the population. The proposed algorithm is named Optimized Artificial Bee Colony (OABC). Arunachalam et al. [11] proposed an enhancement to the ABC algorithm based on the integrated probability with rule-based acceptance. This enhancement aims to maintain the exploration and exploitation of ABC.

An enhancement-based neighborhood search was proposed by Dahan et al. [5]. The proposed algorithm used an adaptive neighborhood search based on the distance of the neighbors. Dahan et al. [6] use an additional step to improve the exploitation of the search space knowledge of the best bees using the swapped method. Chandra et al. [12] introduce the enhancement of ABC exploration and exploitation mechanism based on the search procedure and differential evolution. Seghir et al. [13] also enhanced the neighborhood selection using the interval-based method. Arunachalam et al. [14] used the combinatorial strategical method and cosine similarity to improve the ABC search. This cosine similarity supports ABC exploitation combinatorial strategical search for exploration. Li et al. [15] added the genetic algorithm to the ABC to improve the ABC searching. Karthikeyan et al. [16] introduced an enactment to ABC based on the genetic algorithm. Dahan et al. [7] introduced an enactment to ABC based on the cuckoo search. The ABC with Pareto dominance is used in Zhou et al. [17,18]. The proposed algorithm used the CS Lévy flight in the employed phase to enhance the search in [17], the onlookers phase in [18], and an adaptive parameter strategy is also introduced.

2.2 BA Related Work

Boussalia et al. [19] proposed an extension for the BA algorithm. The proposed algorithm had three different features. The first feature was an adaptation for BA to work with the WSC problem. Next, a constructive heuristic method was used to initialize the population. Finally, a transformation for the real solutions to binary vectors was proposed, and heuristic reparation was to verify and correct the transformation of real solutions. Podili et al. [1] introduced an evaluation of BA and hybrid BA against particle swarm optimization and genetic algorithm. Boussalia et al. [20] introduced two multi-objective algorithms based on CS and BA. For the CS-inspired, quantum computing integrates with CS where the measurement operator, the quantum mutation, and the qubit representation from quantum computing were added to the core of the CS to improve the CS performance. An adaptation for BA was adapted to work with the WSC problem for the BA-inspired. Xu et al. [21] introduced a bat algorithm with a Fuzzy operator. The Fuzzy operator is used for coding and encoding the obtained solution to redefine the velocity. Whereas virtual bats are created based on each bat, a crossover

operator is between them. Xu et al. [22] introduced a bat algorithm with three flight behaviors. Two behaviors are introduced based on the local and global resetting mechanisms to improve exploration and exploitation. The third one is the investigation of the differences between the flying and resetting mechanisms. Dahan [8] proposed an enhancement to BA based on three strategies: improved the initialization and searching of BA agents. The self-adaptive Doppler with local search is introduced in [23] to enhance the BA search mechanism.

2.3 Nature-Inspired Algorithms Related Work

Recently, various researchers applied other nature-inspired algorithms in the face of finding the best possible web services path. Allali et al. [24] proposed enhancing the ACO-based algorithm based on mobile agents. Wang [25] added the genetic algorithm to enhance the search process of the ACO. The multi-pheromone with a neighboring selection process was added to ACO [26]. The multi-agent ACO is introduced by Dahan [27]. The improved eagle algorithm is introduced in [28]. Dogani et al. [29] enhanced the PSO based on the genetic algorithm.

Subbulakshmi et al. [30] introduced an enhancement to the CS based on the genetic algorithm. Ghobaei-Arani et al. [31] introduced an enhancement to the CS based on the distributed network. The e Lévy flight enhancement was added to the CS Kouchi et al. [32]. Wang et al. [33] enhanced the CS using local optimization and the credibility of service. Zhao et al. [34] enhanced the CS using linear population reduction and parameter adaption. Thangaraj et al. [35] used CS to address the WSC. Kurdi et al. [36] proposed a multi-cuckoo to address the WSC. The eagle strategy uniform mutation was added to WOA by Jin et al. [37]. The potential energy convergence with aggregation logarithmic was introduced to enhance the WOA [38]. Dahan [39] proposed an enhancement to WOA based on three strategies: improved the initialization and searching of WOA agents.

The nature-inspired algorithms proposed above have a stochastic behavior which leads to no guarantees of finding the best solution. Based on the No-Lunch-Free theorem (NLF) [40], the different optimization algorithms cannot obtain enough good solutions for optimization problems.

3 Basic Algorithms Background and Proposed Algorithm

3.1 Artificial Bee Colony (ABC)

ABC is a nature-inspired algorithm introduced by Karaboga [41] in 2005. It has the advantages of a few numbers of parameters and easy-to-use. The ABC parameters are maximum iterations (T), searching limitations (limits), and population size (N). The bees in ABC are divided into three searching procedures: employed procedure, onlooker procedure, and scout procedure. These procedures started after the initialization of all bees randomly. Then, the solution quality is measured based on the problem-based fitness function, and the ABC algorithm memorizes the best solution. This step is common between the initialization and these procedures, where the ABC memorizes the best solution after comparing the best solution memorized with the current solution.

In the first procedure, the bees start exploring and recording the nectar of their food sources based on the neighborhood search using Eq. (1). Afterward, they return to the beehive and share the food source information with the onlooker bees using the swing dance. Then, the food source value is calculated.

$$v_{ij} = x_{ij} + \varphi(x_{ij} - x_{kj}) \tag{1}$$

where v_{ij} denotes the candidate food position, x_{ij} is the current solution, x_{kj} represents the solution selected from neighborhood search, and $\varphi \in [-1, 1]$.

In the second procedure, each bee is recruited by an employed bee based on the probability of the employed bee's solution quality calculated using Eq. (2). Similar to employed bees, the onlooker bees search for new solutions based on the neighborhood search using Eq. (1). Then, the food source value is calculated.

$$p_i = \frac{fit(x_i)}{\sum_{l=1}^{N} fit(x_l)} \tag{2}$$

where *fit* represents the solution quality of i^{th} food source. N is the source number.

In the third procedure, the bees monitor the searching process of employed and onlooker bees, and the exhausted solution (poorer bees) is abandoned. Then the scout bees re-initialize the poorer bees into a new food source using Eq. (3).

$$x_{ij} = x_{min,j} + rand(0,1)(x_{max,j} - x_{min,j}) \tag{3}$$

where x_{ij} represents the new j^{th} solution for i^{th} bees. $x_{max,j}, x_{min,j}$ are the maximum and minimum fitness values of the j^{th} solution during the search in the iteration.

3.2 Bat Algorithm (BA)

BA [42] simulates the bats' echolocation behavior, where it builds a scenario of three-dimensional surroundings based on the echo emission and detection time delay, two ears time difference, and echo loudness variations. The BA process is started by initializing the loudness (A_i^t) and pulse emission rate (r_i^t) of the bats using Eqs. (4) and (5), respectively. Then, the solution quality is measured based on the problem-based fitness function, and the BA memorizes the best solution Eqs. (5) and (6)

$$A_i^{t+1} = \propto (A_i^t) \tag{4}$$

$$r_i^{t+1} = [1 - \exp(-\alpha)] \tag{5}$$

where t is a timestamp, α represents a user-specified variable between [0–1].

In each iteration, the bats' positions are updated using Eqs. (6)–(8).

$$f_i^t = f_{min}^t + \left(f_{max}^t - f_{min}^t\right) * rand() \tag{6}$$

where f_{min}^t, f_{max}^t are the minimum and maximum frequencies. *rand ( )* is a random variable between [0–1].

$$v_i^t = v_i^{t-1} + \left(x_i^t - x_*\right) f_i^t \tag{7}$$

where v_i^{t-1} is the previous velocity. x_i^t, x_* are the current and best bat positions, respectively.

$$x_i^{t+1} = x_{new} + v_i^t \tag{8}$$

where x_{new} denotes the new bats' positions that are computed using Eq. (9)

$$x_{new} = x_i^t + randi[-1,1]A_i^t \tag{9}$$

Then, the food source value is calculated, the best solution will be compared with the best solution memorized, and the BA algorithm will memorize the best one.

3.3 The Proposed Hybrid Algorithm (ABC-BA)

The proposed algorithm combines the ABC and BA algorithms with an elitist strategy. We selected ABC because it has the advantage of few control parameters and easy implementation [6]. The idea behind this combination is to help the ABC algorithm to overcome its slow convergence disadvantages. The scout procedure is replaced by the BA with the ES algorithm in the proposed algorithm to avoid

ABC's slow convergence limitation. Fig. 2 presents the process of the proposed algorithm, and in the following subsections, we will describe the main process in detail.

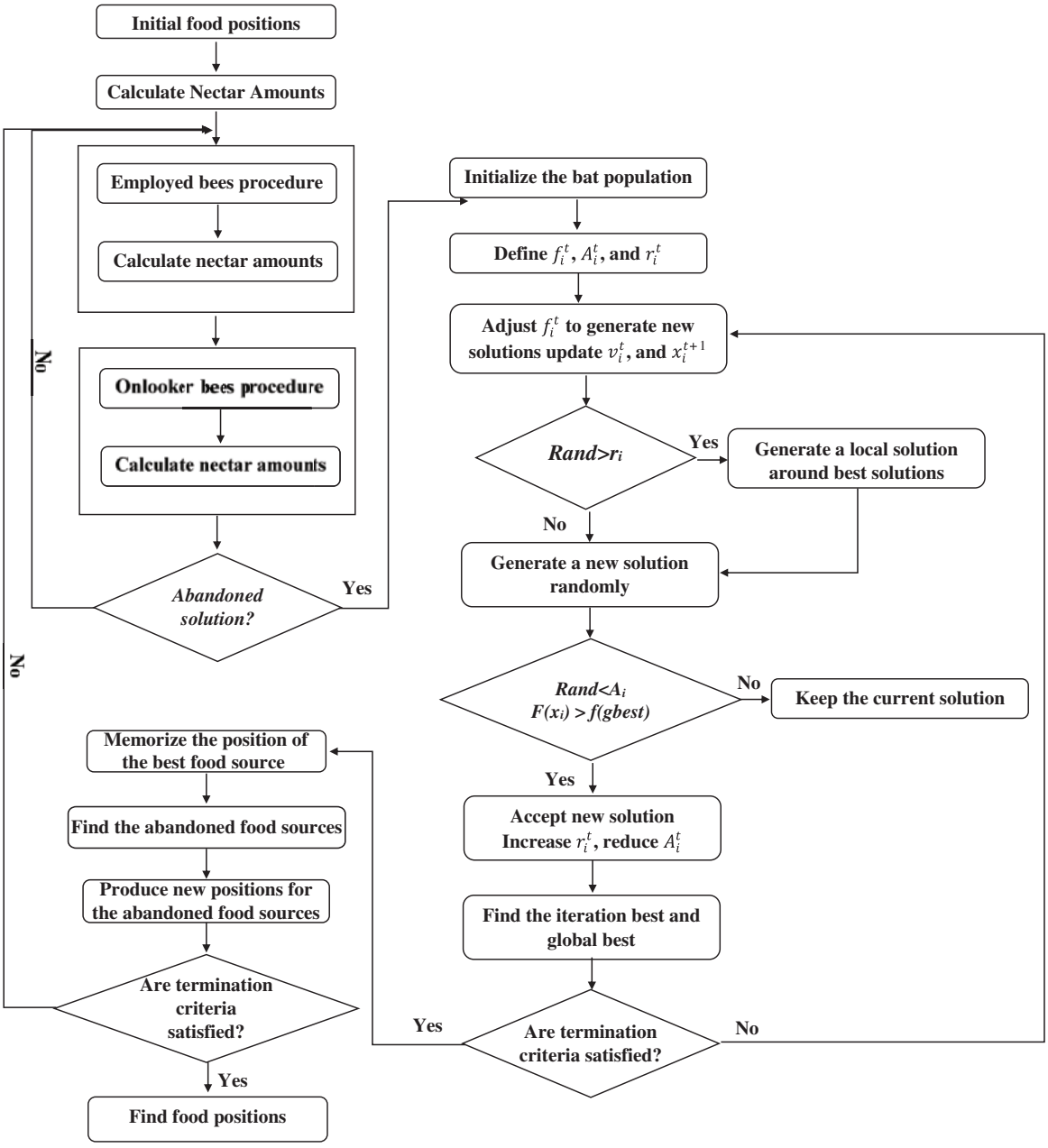

Figure 2: ABC-BA Flowchart

3.3.1 Step1: Initialization

The initialization step of the proposed algorithm is to generate a solution for each bee randomly. This solution contains a web service for each task in the total of m web services. The random selection is a number between 1 and *m*, where *m* is the number of WS retrieved for each task. To calculate the fitness

of each solution, the WSC is represented as a multi-bjective optimization problem where the QoS constraint values either need to be maximized or minimized. In this work, four QoS constraints have adopted the Cost (C), Response Time (RT) (Their objective values are to be minimized), Reliability (R), and Throughput (T) (Their objective values are to be maximized). The reference-based method [37] is adopted in this work. In the reference-based method, the multi-objective optimization vectors are converted into single-objective based on the QoS constraints aggregation function shown in Table 1. The fitness formula of each solution calculation is shown in Eq. (10). For clarity, the user preferences regarding the QoS constraints are unified, so they aren't reflected in Eq. (10).

Table 1: The QoS aggregation formulas

QoS criteria	Aggregation formula
Cost (C)	$\sum_{i=1}^{n} C\left(ws_{ij}\right)$
Response time (RT)	$\sum_{i=1}^{n} RT\left(ws_{ij}\right)$
Throughput (T)	$\prod_{i=1}^{n} T\left(ws_{ij}\right)$
Reliability (R)	$\prod_{i=1}^{n} R\left(ws_{ij}\right)$

$$F_i = \left(\prod_{j=1}^{n} T_{jb} + \prod_{j=1}^{n} R_{jb} - \sum_{j=1}^{n} C_{jb} - \sum_{j=1}^{n} RT_{jb}\right) \tag{10}$$

where n is the number of tasks, F_i is the solution's fitness, and i represents the i^{th} bee/bat.

3.3.2 Step2: Employed Procedure

In the employed procedure, each bee searches for a new solution based on the neighborhood search. The employed bees are initialized in step 1 and use Eq. (1) to select the new solutions. In WSC, the solutions represent a complete path, which contains a web service for each task as $WS^{b}_{1,x'}, WS^{b}_{2,x'}, WS^{b}_{3,x''}, \ldots WS^{b}_{n,x}$ where b the i^{th} bee, *1,2,3,..n* are the task sequence, and x′, x″, x″, .., x are web services between 1 and m.

In Eq. (1), a loop from 1 to the n tasks is used to create a new solution path for each bee. For each path, the current web service is replaced based on the current web service and another web service from another selected solution represented by k with a random value represented by φ.

The solution fitness is obtained using Eq. (10), and the best solution of the employed bees will be compared with the best solution memorized, and the proposed algorithm will memorize the best one.

3.3.3 Step3: Onlooker Procedure

In the onlooker procedure, the bees monitor the employed bees after their tour. The employed bees share their solution quality and recruit the onlooker bees based on the fitness probability. Eq. (2) calculates the employed bees' fitness probability. After the recruitment, the onlooker bees repeat the same process as an employed procedure for searching for a new solution.

The solution fitness is obtained using Eq. (10). The best solution for the onlooker bees will be compared with the best solution memorized, and the proposed algorithm will memorize the best one.

3.3.4 Step4: BA Optimization

In the proposed algorithm, the process of the scout procedure is replaced by the BA with ES. The process of the BA is described in Section 3.2 with modification based on the ES [43] applied. The idea behind ES is to enhance the convergence rate of the proposed algorithm. In ES, the two best solutions are allowed to be survived in the next iterations; these are the best local solution in each iteration (S_{Ibest}) and the best global solution among all bats (S_{Gbest}). The S_{Ibest} and S_{Gbest} are calculated based on the solutions' fitness, and the selection formulas are shown in Eqs. (11) and (12).

$$S_{Ibest} = \max(\text{iteration} - \text{based fitness value}) \tag{11}$$

$$S_{Gbest} = \max(\text{distance fitness value}) \tag{12}$$

The S_{Ibest} and S_{Gbest} values are obtained using Eq. (10).

After selecting S_{Ibest} and S_{Gbest}, their values will be compared with new values based on the iteration or bats. Then, S_{Ibest} and S_{Gbest} will be updated using Eqs. (13) and (14), respectively.

$$S_{Ibest} = \begin{cases} S^{t-1}_{Ibest} = S^{t}_{Ibest} & if\ f(t) \geq f(t-1) \\ S^{t}_{Ibest} = S^{t-1}_{Ibest} & if\ f(t) \leq f(t-1) \end{cases} \tag{13}$$

$$S_{Gbest} = \begin{cases} S^{t-1}_{Gbest} = S^{t}_{Gbest} & if\ D(t) \geq D(t-1) \\ S^{t}_{Gbest} = S^{t-1}_{Gbest} & if\ D(t) \leq D(t-1) \end{cases} \tag{14}$$

Based on the ES, a modification of bats' basic formulas search, velocity, and frequency equations are required to attain a better tradeoff among searching mechanisms using Eqs. (15)–(18) [43].

$$f^{t}_{i} = \frac{\min(S^{t}_{Gbest}) + \left(\max(S^{t}_{Gbest}) - \min(S^{t}_{Gbest})\right) rand()}{\max(S^{t}_{Ibest})} \tag{15}$$

$$v^{t}_{i} = v^{t-1}_{i} + \left(S^{t}_{Gbest} - S^{t}_{Ibest}\right) f^{t}_{i} \tag{16}$$

$$x_{new} = S^{t}_{Gbest} + randi[-1, 1] \tag{17}$$

$$x^{t+1}_{i} = \begin{cases} S^{t}_{Gbest} & if\ rand > r_i \\ x_{new} + v^{t}_{i} & otherwise \end{cases} \tag{18}$$

where i refer to i^{th} bat at iteration t and rand is a random value between 0 and 1.

4 Results and Discussion

Several experimental aspects are used to evaluate the performance competitiveness and effectiveness of the ABC-BA algorithm compared to the different nature-inspired algorithms.

4.1 Experimental Settings

Different datasets resources were used to compare the proposed algorithm's performance with other real-world datasets called QWS 2.0 [44] and generated randomly [45].

From the QWS 2.0, we created the small-size datasets (first dataset group) where the number of tasks size is kept constant, and the WSs in each task are different in a total of 23000 WSs. The experiments on small-size datasets aim to show the proposed algorithm's capabilities compared to the competitors on the local exploitation [46] because these datasets' dimensionality (search space) is small.

From the generated random datasets, we created the medium-size and large-size datasets (second dataset group) where the number of WSs in each task is kept constant, and the tasks are different in a total of 52000 WSs. The experiments using these datasets aim to show the proposed algorithm's capabilities compared to the competitors on the global search [46] because the dimensionality of these datasets is large compared to the first dataset group. A description of the datasets used in this work is shown in Table 2.

Table 2: Datasets description

Dataset	Size	No. Tasks	No. WSs/task
DS1	Small	10	100
DS2		10	400
DS3		10	800
DS4		10	1000
DS5	Medium	30	100
DS6		40	100
DS7		50	100
DS8		60	100
DS9	Large	70	100
DS10		80	100
DS11		90	100
DS12		100	100

To ensure a level of unbiased and fair experiments, we implemented all algorithms in Java using Windows 10 with Intel(R) Core(TM) i7-7500U CPU @ 2.70 GHz and 8.0 GB RAM. In addition, four algorithms were used to compare the ABC-BA performance, which are ABC_CS [7], MOHABC [17], SABC [6], and OABC [10]. The parameters setting of these algorithms is set based on their works preferences. Table 3 presents the initial values of the ABC-BA parameters. There are some common parameters between the ABC-BA and other algorithms: the iteration numbers (Z) set to 500 and the population (P) set to 100.

Table 3: Parameters' settings of ABC-BA

Parameter	Value
Limits	100
th	0.5
P_a	0.5
r	0.5
F_{min}	0

(Continued)

Table 3: Continued

Parameter	Value
F_{max}	2
A	0.25

4.2 Comparative Results Discussion

According to the dataset groups, two types of experiments are conducted to validate the performance competitiveness of ABC-BA. The first experiment is conducted on the first group of the dataset to validate the local exploitation capabilities of the ABC-BA, while the second experiment is conducted on the second group of the dataset to validate the global search capabilities of the ABC-BA. In the 30 different runs, the Average Fitness Value (AFV) of each algorithm, best fitness values (BFV), and execution time were measured. These three criteria were the comparison evaluation criteria for ABC-BA compared to the competitors.

The notched boxplot is used to visualize the empirical distribution of results obtained by all competitors in terms of the AFV, as shown in Figs. 3–5. More detailed results can be observed using notched boxplots, which show the numerical data distribution. Figs. 3–5 show the average fitness value distributions over all competitors for small, medium, and large datasets, respectively. In addition, Table 4 presents the empirical results of all competitors in terms of best fitness value and average execution time over all datasets.

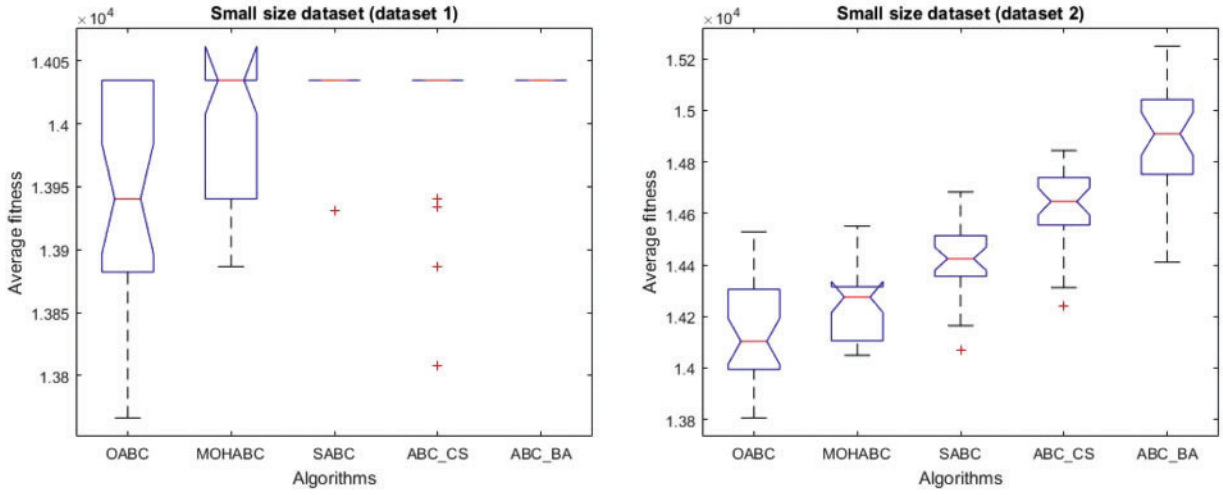

Figure 3: (Continued)

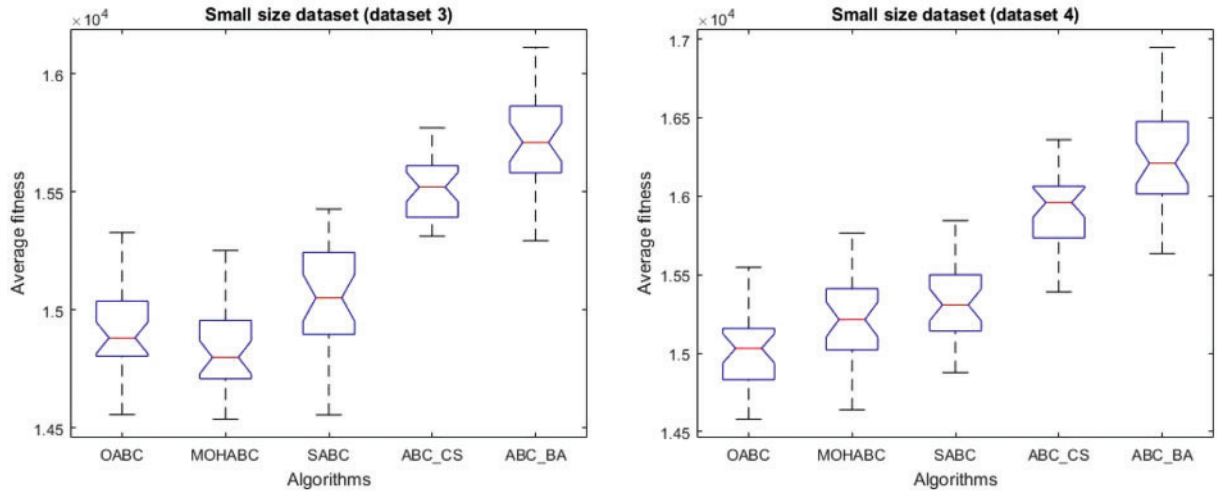

Figure 3: The AFV of compared algorithms for the small-size datasets

4.2.1 Local Exploitation Validation Experiments

In this experiment, the local exploitation capabilities of the proposed algorithm compared to other algorithms are verified on small-size datasets. Fig. 3 presents the AFV of the proposed algorithm and competitors in 30 independent runs. As seen from the figure, the ABC-BA has a concentrated AFV, which means that the AFV is better because the median value is larger than others. In addition, the ABC-BA hasn't an outlier value and better stability.

Fig. 3 also shows that the ABC_BA results significantly differ because the AFV of ABC_BA has no notches overlapping over all datasets. In addition, the boxplots of ABC_BA are significantly higher than others.

As seen in Table 4, the proposed algorithm has the best BFV values of competitors' on all small datasets, which indicates the competitive performance of ABC_BA in convergence accuracy.

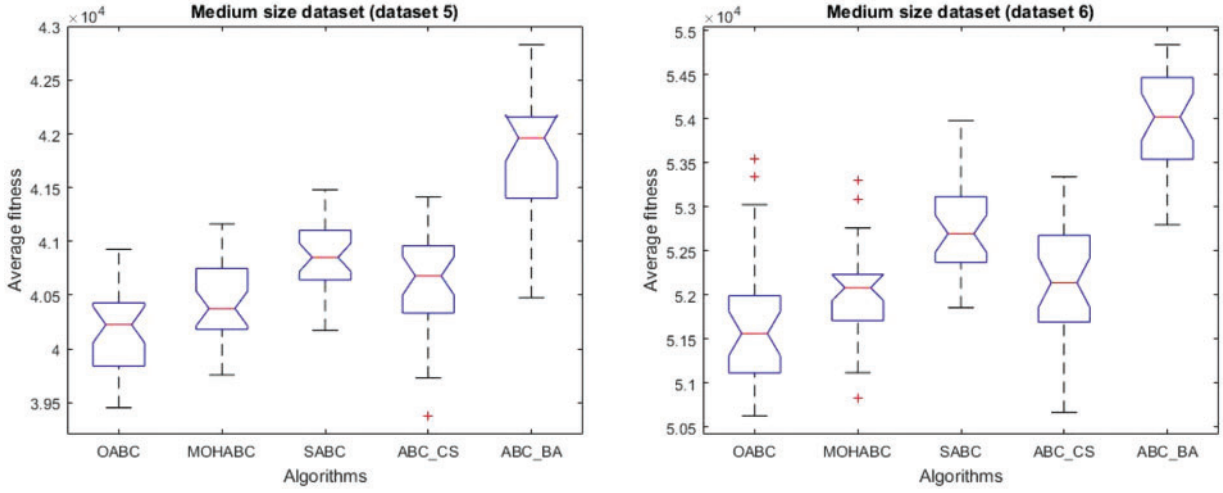

Figure 4: (Continued)

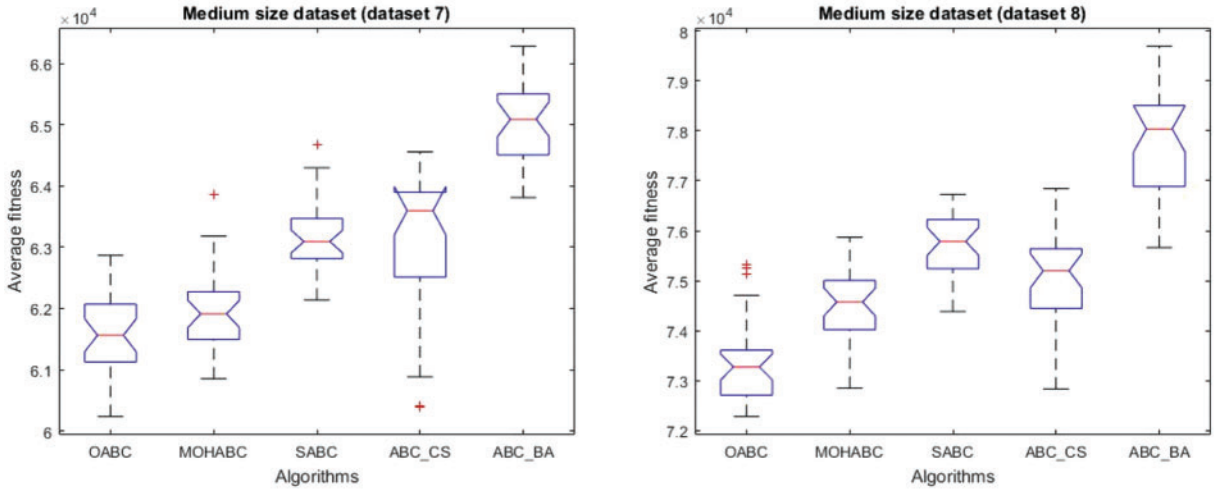

Figure 4: The AFV of compared algorithms for the medium-size dataset

Table 4 shows that the proposed algorithm is not the best in terms of execution time because the proposed enhancement needs more time. However, the convergence accuracy of these algorithms' average and best fitness values are worse than that of the ABC_BA.

4.2.2 Global Search Validation Experiments

In this experiment, the global search capabilities of the proposed algorithm compared to other algorithms are verified on medium-size and large-size datasets, as shown in Figs. 4 and 5, respectively. Figs. 4 and 5 present the AFV of the proposed algorithm and competitors in 30 independent runs. As seen from the figure, the ABC-BA has a concentrated AFV, which means that the AFV is better because the median value is larger than others. In addition, the ABC-BA hasn't an outlier value and better stability.

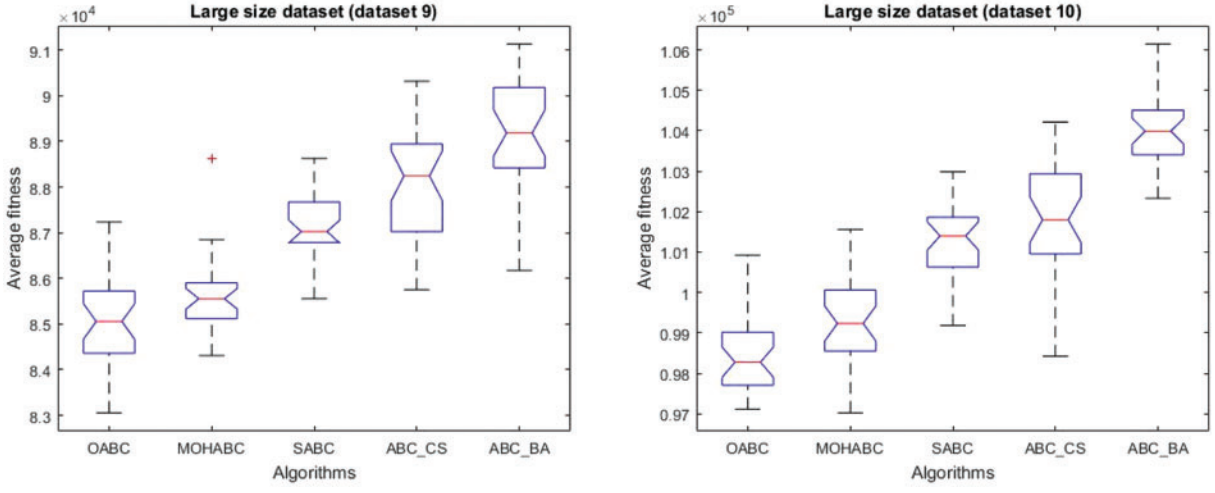

Figure 5: (Continued)

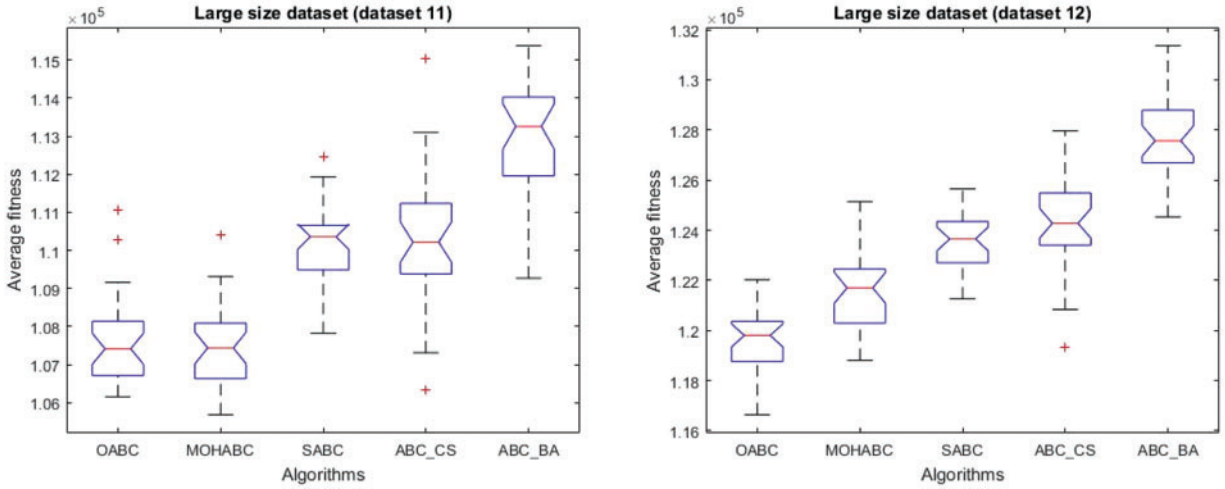

Figure 5: The AFV of compared algorithms for the large-size dataset

Figs. 4 and 5 also show that the ABC_BA results significantly differ because the AFV of ABC_BA has no notches overlapping over all datasets. In addition, the boxplots of ABC_BA are significantly higher than others except for DS 9.

Table 4: ABC_BA BFV and execution time values and other algorithms

Dataset	Best fitness value					Average execution time				
	OABC	MOHABC	SABC	ABC_CS	ABC_BA	OABC	MOHABC	SABC	ABC_CS	ABC_BA
DS1	**14034.43**	**14034.43**	**14034.43**	**14034.43**	**14034.43**	95	97	94	**87**	93
DS2	14528.76	14550.74	14683.33	14844.86	**15249.55**	416	**114**	128	120	122
DS3	15327.59	15252.1	15427.51	15771.44	**16112.19**	721	**124**	140	137	134
DS4	15546.62	15764.98	15843.81	16359.06	**16946.5**	857	163	155	**143**	153
DS5	40925.86	41162.14	41480.34	41303.38	**42830.12**	211	256	295	**162**	231
DS6	53542.62	53296.14	53978.77	53339.69	**54839.33**	268	383	430	**224**	345
DS7	62868.06	63866.37	64666.88	64557.34	**66283.27**	**344**	425	507	366	432
DS8	75333.25	75870.73	76726.70	76847.28	**79690.83**	**371**	563	589	441	531
DS9	87238.51	88635.27	88628.77	90320.09	**91132.84**	**466**	767	786	552	643
DS10	100921.44	101562.39	102985.17	104215.63	**106144.3**	**560**	830	953	658	814
DS11	111073.18	110410.59	112471.75	115029.29	**115377**	**632**	807	1027	797	877
DS12	122028.87	125145.18	125650	127963.64	**131362.2**	**641**	1012	1140	929	1027

Table 4 shows that the proposed algorithm is not the best in terms of execution time because the proposed enhancement needs more time. However, the convergence accuracy of these algorithms' average and best fitness values are worse than that of the ABC_BA.

4.3 Result Significance Test

The Wilcoxon signed-rank test with a p-value shown in Table 5 presents the statistically significant results of ABC_BA and competitors over all datasets in terms of average fitness value. The last row in the table shows the test summary, where the s indicates the number of datasets where the results are

significant, and w means the opposite. Table 5 shows that ABC_BA significantly differs from other algorithms on all datasets.

Table 5: Significance test of ABC_AB compared to other algorithms

Datasets	Average fitness values			
	OABC	MOHABC	SABC	ABC_CS
DS1	<.00001	<.00001	.00652	0.0102
DS2	<.00001	<.00001	<.00001	<.00001
DS3	<.00001	<.00001	<.00001	<.00001
DS4	<.00001	<.00001	<.00001	<.00001
DS5	<.00001	<.00001	<.00001	0.03846
DS6	<.00001	<.00001	0.00736	0.04891
DS7	<.00001	<.00001	0.00528	0.037346
DS8	<.00001	<.00001	0.0001	<.00001
DS9	<.00001	<.00001	<.00001	0.03846
DS10	<.00001	<.00001	<.00001	<.00001
DS11	<.00001	<.00001	<.00001	<.00001
DS12	<.00001	<.00001	<.00001	<.00001
s/w	12/0	12/0	12/0	12/0

5 Conclusion

WSC problem aims to find the optimal web services combination to meet user needs. The complexity of this problem is the many services that achieve the functions with different QoS Constraints. The WSC is an NP-hard problem. This work proposes an extension of our previous work for the WSC problem. Previously, we introduced different enhancements for ABC and BA algorithms, and recently, we introduced a hybridization algorithm between these two algorithms and applied the ES to enhance the convergence of the proposed method. In the proposed work, the scout procedure of ABC was replaced by the BA process, and the process of BA was updated to adopt the ES process. The performance and convergence behavior of the proposed hybrid algorithm was tested using extensive comparative experiments with current state-of-the-art nature-inspired algorithms on 12 benchmark datasets using three evaluation criteria (average fitness values, best fitness values, and execution time) that were measured for 30 different runs. The benchmark datasets were based on real measurements of web services and artificial-based datasets. The results show that the proposed algorithm enhances the search performance and increases the convergence rate on finding the near-optimal web services combination compared to competitors. The results also show a limitation of the proposed algorithm, where it is slower in terms of execution time compared to other algorithms. In future work, we plan to investigate other nature-inspired algorithms, such as the whale optimization algorithm, to address such a problem and keep the superiority of the results in terms of the average fitness and best fitness values.

Acknowledgement: The authors would like to express their gratitude to Prince Sattam bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia for providing the support.

Funding Statement: The authors extend their appreciation to the Deputyship for Research and Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number 2022/01/22636.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] P. Podili, K. K. Pattanaik and P. S. Rana, "BAT and hybrid BAT meta-heuristic for quality of service-based web service selection," *Journal of Intelligent Systems*, vol. 26, no. 1, pp. 123–137, 2017.

[2] G. Canfora, M. Di Penta, R. Esposito and M. L. Villani, "A lightweight approach for QoS-aware service composition," in *Proc. 2nd Int. Conf. on Service Oriented Computing (ICSOC'04)*, New York, USA, CM Press, pp. 1–2, 2004.

[3] U. A. Bhatti, Z. Zeeshan, M. M. Nizamani, S. Bazai, Z. Yu *et al.,* "Assessing the change of ambient air quality patterns in Jiangsu Province of China pre-to post-COVID-19," *Chemosphere*, vol. 288, no. 6, pp. 132569, 2022.

[4] U. A. Bhatti, M. M. Nizamani and H. Mengxing, "Climate change threatens Pakistan's snow leopards," *Science*, vol. 377, no. 6606, pp. 585–586, 2022.

[5] F. Dahan, K. El Hindi and A. Ghoneim, "Enhanced artificial bee colony algorithm for QoS-aware web service selection problem," *Computing*, vol. 99, no. 5, pp. 507–517, 2017.

[6] F. Dahan, H. Mathkour and M. Arafah, "Two-step artificial bee colony algorithm enhancement for QoS-aware Web service selection problem," *IEEE Access*, vol. 7, pp. 21787–21794, 2019.

[7] F. Dahan and A. Alwabel, "Artificial bee colony with cuckoo search for solving service composition," *Intelligent Automation & Soft Computing*, vol. 35, no. 3, pp. 3385–3402, 2023. https://doi.org/10.32604/iasc.2023.030651.

[8] F. Dahan, "Neighborhood search based improved bat algorithm for web service composition," *Computer Systems Science and Engineering*, vol. 45, no. 2, pp. 1343–1356, 2023. https://doi.org/10.32604/csse.2023.031142

[9] F. Seghir, "FDMOABC: Fuzzy discrete multi-objective artificial bee colony approach for solving the non-deterministic QoS-driven web service composition problem," *Expert Systems with Applications*, vol. 167, no. 3, pp. 114413, 2021.

[10] S. Zhang, Y. Shao and L. Zhou, "Optimized artificial bee colony algorithm for web service composition problem," *International Journal of Machine Learning and Computing*, vol. 11, no. 5, pp. 327–332, 2021.

[11] N. Arunachalam and A. Amuthan, "Integrated probability multi-search and solution acceptance rule-based artificial bee colony optimization scheme for web service composition," *Natural Computing*, vol. 20, no. 1, pp. 23–38, 2021.

[12] M. Chandra and R. Niyogi, "Web service selection using modified artificial bee colony algorithm," *IEEE Access*, vol. 7, pp. 88673–88684, 2019.

[13] F. Seghir, A. Khababa and F. Semchedine, "An interval-based multi-objective artificial bee colony algorithm for solving the web service composition under uncertain QoS," *The Journal of Supercomputing*, vol. 75, no. 9, pp. 5622–5666, 2019.

[14] N. Arunachalam and A. Amuthan, "Improved cosine similarity-based artificial bee colony optimization scheme for reactive and dynamic service composition," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 2, pp. 270–281, 2022.

[15] T. Li, Y. Yin, B. Yang, J. Hou and K. Zhou, "A self-learning bee colony and genetic algorithm hybrid for cloud manufacturing services," *Computing*, vol. 104, no. 9, pp. 1–27, 2022.

[16] P. Karthikeyan and G. Preethi, "Artificial bee colony and genetic algorithms in selecting and combining web services for enhancing QoS," *Design Engineering*, vol. 2021, pp. 6009–6021, 2021.

[17] J. Zhou and X. Yao, "A hybrid approach combining modified artificial bee colony and cuckoo search algorithms for multi-objective cloud manufacturing service composition," *International Journal of Production Research*, vol. 55, no. 16, pp. 4765–4784, 2017.

[18] J. Zhou and X. Yao, "Multi-objective hybrid artificial bee colony algorithm enhanced with Lévy flight and self-adaption for cloud manufacturing service composition," *Applied Intelligence*, vol. 47, no. 3, pp. 721–742, 2017.

[19] S. R. Boussalia, A. Chaoui and A. Hurault, "Qos-based web services composition optimization with an extended bat inspired algorithm," *Communications in Computer and Information Science*, vol. 538, pp. 306–319, 2015.

[20] S. R. Boussalia, A. Chaoui, A. Hurault, M. Ouederni and P. Queinnec, "Multi-objective quantum inspired Cuckoo search algorithm and multi-objective bat inspired algorithm for the web service composition problem," *International Journal of Intelligent Systems Technologies and Applications*, vol. 15, no. 2, pp. 95–126, 2016.

[21] B. Xu and Z. Sun, "A fuzzy operator based bat algorithm for cloud service composition," *International Journal of Wireless and Mobile Computing*, vol. 11, no. 1, pp. 42–46, 2016.

[22] B. Xu, J. Qi, X. Hu, K. -S. Leung, Y. Sun *et al.,* "Self-adaptive bat algorithm for large scale cloud manufacturing service composition," *Peer-to-Peer Networking and Applications*, vol. 11, no. 5, pp. 1115–1128, 2018.

[23] A. Kouicem, M. E. Khanouche and A. Tari, "Novel bat algorithm for QoS-aware services composition in large scale internet of things," *Cluster Computing*, vol. 25, no. 5, pp. 1–15, 2022.

[24] N. El Allali, M. Fariss, H. Asaidi and M. Bellouki, "A web service composition framework in a heterogeneous environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, no. 2, pp. 1–25, 2022.

[25] Z. Wang, "Optimization of resource service composition in cloud manufacture based on improved genetic and ant colony algorithm," *Smart Innovation, Systems and Technologies*, vol. 268, pp. 183–198, 2022.

[26] F. Dahan, K. El Hindi, A. Ghoneim and H. Alsalman, "An enhanced ant colony optimization based algorithm to solve QoS-aware web service composition," *IEEE Access*, vol. 9, pp. 34098–34111, 2021.

[27] F. Dahan, "An effective multi-agent ant colony optimization algorithm for QoS-aware cloud service composition," *IEEE Access*, vol. 9, pp. 17196–17207, 2021.

[28] V. Rajendran, R. K. Ramasamy and W. -N. Mohd-Isa, "Improved eagle strategy algorithm for dynamic web service composition in the IoT: A conceptual approach," *Future Internet*, vol. 14, no. 2, pp. 56, 2022.

[29] J. Dogani and F. Khunjush, "Cloud service composition using genetic algorithm and particle swarm optimization," in *Proc. 2021 11th Int. Conf. on Computer Engineering and Knowledge (ICCKE)*, Mashhad, Iran, pp. 98–104, 2022.

[30] S. Subbulakshmi, K. Ramar, A. E. Saji and G. Chandran, "Optimized web service composition using evolutionary computation techniques," in *Intelligent Data Communication Technologies and Internet of Things: Proc. of ICICI 2020*, Singapore, pp. 457–470, 2021.

[31] M. Ghobaei-Arani, A. A. Rahmanian, M. S. Aslanpour and S. E. Dashti, "CSA-WSC: Cuckoo search algorithm for web service composition in cloud environments," *Soft Computing*, vol. 22, no. 24, pp. 8353–8378, 2018.

[32] S. Kouchi and H. Nacer, "Service selection in cloud computing environment by using cuckoo search," in *The Int. Conf. on Information, Communication & Cybersecurity*, Khourigba, Morocco, pp. 219–228, 2021.

[33] H. Wang, D. Yang, Q. Yu and Y. Tao, "Integrating modified cuckoo algorithm and creditability evaluation for QoS-aware service composition," *Knowledge-Based Systems*, vol. 140, no. 6, pp. 64–81, 2018.

[34] F. Zhao, J. Bao and F. Ding, "A new integrating adaptive cuckoo search optimization algorithm for management service composition," in *Proc. of the Int. Conf. on Information Technology and Electrical Engineering 2018*, New York, United States, pp. 1–6, 2018.

[35] P. Thangaraj and P. Balasubramanie, "QoS based service composition for service computing using cuckoo search," *Parameters*, vol. 16, pp. 17, 2018.

[36] H. Kurdi, F. Ezzat, L. Altoaimy, S. H. Ahmed and K. Youcef-Toumi, "Multicuckoo: Multi-cloud service composition using a cuckoo-inspired algorithm for the internet of things applications," *IEEE Access*, vol. 6, pp. 56737–56749, 2018.

[37] H. Jin, S. Lv, Z. Yang and Y. Liu, "Eagle strategy using uniform mutation and modified whale optimization algorithm for QoS-aware cloud service composition," *Applied Soft Computing*, vol. 114, no. 5, pp. 108053, 2022.

[38] X. Teng, Y. Luo, T. Zheng and X. Zhang, "An improved whale optimization algorithm based on aggregation potential energy for qos-driven web service composition," *Wireless Communications and Mobile Computing*, vol. 2022, no. 5, pp. 1–13, 2022.

[39] F. Dahan, "An improved whale optimization algorithm for web service composition," *Axioms*, vol. 11, no. 12, pp. 725, 2022.

[40] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[41] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," *Erciyes University, Engineering Faculty, Computer Engineering Department*, vol. 200, no. 1, pp. 1–10, 2005.

[42] X. S. Yang, "A new metaheuristic bat-inspired algorithm," *Studies in Computational Intelligence*, vol. 284, pp. 65–74, 2010.

[43] A. Kaur and Y. Kumar, "Neighborhood search based improved bat algorithm for data clustering," *Applied Intelligence*, vol. 52, no. 9, pp. 1–35, 2022.

[44] E. Al-Masri and Q. H. Mahmoud, "Discovering the best web service," in *Proc. of the 16th Int. Conf. on World Wide Web*, Banff Alberta, Canada, pp. 1257–1258, 2007.

[45] X. Wang, Z. Wang and X. Xu, "An improved artificial bee colony approach to QoS-aware service selection," in *Proc. IEEE 20th Int. Conf. on Web Services, ICWS 2013*, Santa Clara, CA, USA, pp. 395–402, 2013.

[46] J. Li, H. Ren, C. Li and H. Chen, "A novel and efficient salp swarm algorithm for large-scale QoS-aware service composition selection," *Computing*, vol. 104, no. 1, pp. 1–21, 2022.