Received December 26, 2018, accepted January 5, 2019, date of publication January 24, 2019, date of current version February 27, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2894683

Two-Step Artificial Bee Colony Algorithm Enhancement for QoS-Aware Web Service Selection Problem

FADL DAHAN, HASSAN MATHKOUR, AND MOHAMMED ARAFAH
Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh 11543-51178, Saudi Arabia

Corresponding author: Hassan Mathkour (mathkour@ksu.edu.sa)

This work was supported by the Deanship of Scientific Research at King Saud University through the Research Group under Grant RGP-1436-002.

ABSTRACT This paper presents an enhanced artificial bee colony (ABC) algorithm for solving the web service selection problem. The proposed algorithm searches the best possible combination of web services to satisfy user requirements. An adapted neighborhood selection and replacement process and a swapping process are used to improve the ABC behavior. Neighboring nodes are employed to enhance ABC performance by encouraging exploration in early iterations, where bees have no knowledge regarding the search space, and by encouraging exploitation in later iterations to exploit bee knowledge of the search space. The swapping process is used to enhance ABC performance by randomly swapping portions among the best two solutions randomly. The idea behind this swap is to exploit the characteristics of the best solutions to generate new solutions. We compared the proposed algorithm with other algorithms in terms of quality and execution time using 60 different datasets. These datasets have different numbers of tasks and web services. The results indicate that the proposed algorithm finds better solutions compared to other algorithms. In addition, the results' summarization on 60 datasets with 30 different executions shows that the proposed algorithm outperforms the threshold-based algorithm by 6% and the enhanced ABC by 3% in terms of solutions' quality.

INDEX TERMS Artificial bee colony (ABC), service-oriented computing (SOC), web service (WS), web service selection (WSS).

I. INTRODUCTION

In current practice, enterprises aim to integrate different applications on heterogeneous platforms and take advantage of the Internet's infrastructure. Consequently, these enterprises define their business processes to support applications comprehensively and automatically [1]–[3].

Service oriented computing which is concerned with building, calling, maintaining, and enhancing distributed applications [4], [5] has been introduced as a solution. Service-oriented architecture (SOA) introduces an architecture for new solutions for planning, designing and delivering the information technology functionality to achieve business processes [4].

In SOA, enterprises have the freedom to use any autonomous and loosely coupled technology platform to apply SOA architecture [6], [7]. Web service (WS) platforms are most often associated with SOA [8].

An individual WS is not sufficient to create a compound business process (workflow) [9]. Web service selection (WSS) problem aims at creating complex business processes from a set of web services. This is because web services are interoperable, loosely coupled, well defined, and just-in-time integration. WSS problem is a challenging problem because there are many providers that provide similar web services [10]. The WSS problem aims at selecting a best combination of web services to fulfill the tasks in a given workflow. A workflow is a translation of customer requirements into a set of tasks.

The users' requirements can be presented in different forms called patterns. There are four different patterns based on different execution requirements, as shown in Figure 1 [10]. In the sequential pattern, requirements are fulfilled in order. In the parallel pattern, requirements are fulfilled concurrently. In the conditional pattern, certain requirements have specific execution conditions. In a loop pattern, some requirements may need to be fulfilled many times.

Workflows can be either simple or complex [10], [11]. A simple workflow has tasks in sequential patterns.

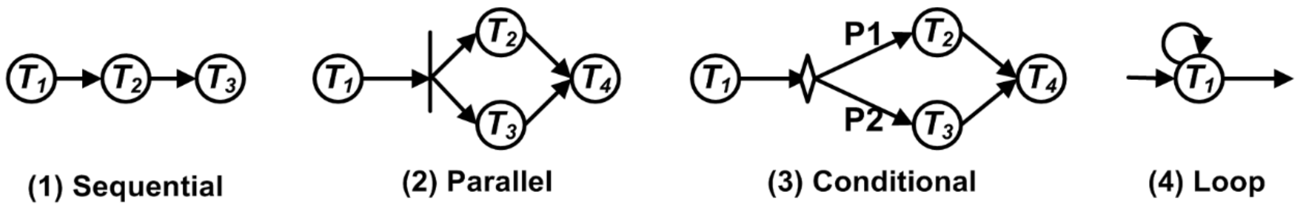


FIGURE 1. Workflow patterns.

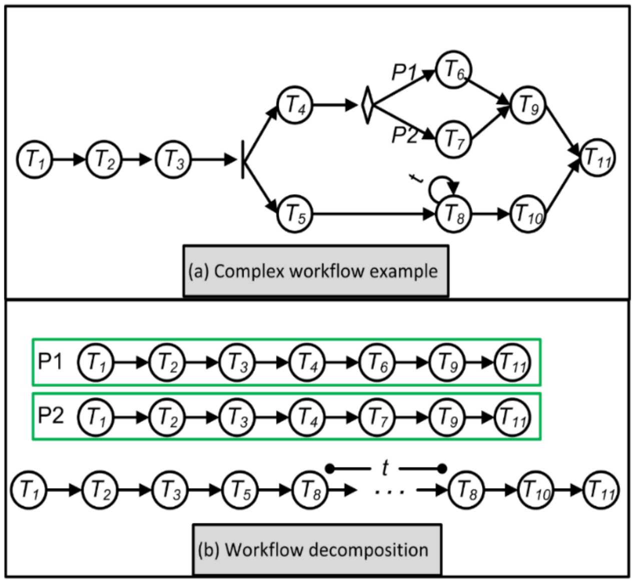


FIGURE 2. Workflow types and decomposition.

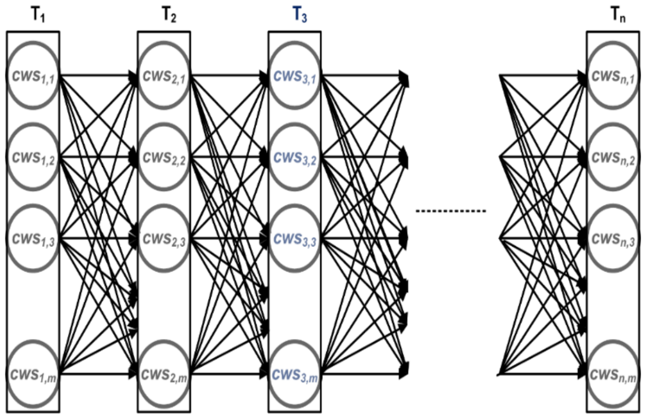


FIGURE 3. WSS problem representation.

TABLE 1. QoS attributes aggregation models.

QoS Criteria	Calculation Formula
Cost (C)	$\sum_{i=1}^{n} C(cws_{ij})$
Response Time (RT)	$\sum_{i=1}^{n} RT(cws_{ij})$
Throughput (T)	$\prod_{i=1}^{n} T(cws_{ij})$
Reliability (R)	$\prod_{i=1}^{n} R(cws_{ij})$

A complex workflow can have tasks in sequential, parallel, loop, or conditional patterns. However, complex workflows can be decomposed into many simple workflows. Figure 2(a) presents an example of a complex workflow and Figure 2(b) presents the decomposition process for converting a complex workflow into multiple simple workflows [12], [13].

A representation of the WSS problem has n tasks. Each task has m web services from different providers that can fulfill that task. Therefore, the number of possible solutions is equal to m^n. This representation is illustrated in Figure 3.

The m web services for each task have similar functionality. Therefore, the selection process requires additional criteria to perform the selection among them. Further, the quality of service (QoS) attributes are used to aid the selection and to describe non-functional properties of the web services. For each generated solution, the attributes are calculated based on the formal definitions provided in TABLE 1. In TABLE 1, cws_{ij} is the j^{th} WS associated with the i^{th} task. n is the number of tasks [13].

The WSS is described as an NP-hard problem because it is a multi-objective optimization problem [14] . To solve such a complex problem, we adopted a preference-based method [15]. In preference-based method, the objective vectors are aggregated into a single-objective value based on an aggregation function based on the formulas listed in TABLE 1.

The paper is organized as follows. Section II discusses the related work. Section III presents the concept of artificial bee colony (ABC) algorithm. Section IV describes our proposed two-step enhancement for the ABC algorithm. In Section V, we present and analyze simulation results. The last section discusses our conclusions.

II. RELATED WORK

Using the ABC algorithm to deal with the WSS problem has been a topic of research. Kousalya *et al.* [16] introduced an enhancement for the ABC algorithm where the random search is combined with local search to select better web services for a given workflow. Zhenwu and Wan [17] introduced a modified algorithm called the discrete ABC algorithm. This algorithm contains a discretization mechanism to optimize the ABC algorithm for this problem. He *et al.* [18] introduced a mathematical model for this problem. They enhanced the ABC algorithm using the taboo strategy with a chaos factor. They called their algorithm the improved ABC algorithm.

Wang *et al.* [19] introduced a definition for the optimal continuity concept and proposed two enhancements for the ABC algorithm based on neighborhood selection. These enhancements are called the threshold-based algorithm (TBA) and distance-based algorithm. Wang *et al.* [19] introduced a method to represent the QoS-aware WSS problem as a combinatorial optimization problem. Additionally, they introduced an enhancement for the ABC algorithm using the similarity and priority of web services. Dahan *et al.* [12] proposed an enhancement for the ABC algorithm. They used a dynamic selection and replacement of neighbors. The goal of this selection and replacement process is to balance the exploration and exploitation mechanisms of the algorithm.

Zhang and Zhang [20] introduced a hybrid algorithm. The new algorithm is a hybridization of the ant colony optimization algorithm and ABC algorithm called the hybrid ABC algorithm. They used a skyline query to shrink the task's web services. A composition graph is first created using a clustering process. Next, a swarm of bees executes a foraging step and a swarm of ants works to enhance solutions quality. Chifu *et al.* [21] proposed two enhancements for the QoS-aware WSS problem. First enhancement was for the ABC. The second enhancement was for cuckoo search algorithm. Liu *et al.* [22] studied the effect of parameters tuning on the ABC using the code of Wang *et al.* [19].

III. THE ABC ALGORITHM

The ABC algorithm (shown in Figure 4 [23]) was first introduced by Karaboga [24]. In nature, honeybees send a group of bees to explore the surrounding area for food. These bees then return to the hive and recruit other bees to follow them based on quantity of food they discovered. The ABC algorithm simulates this behavior while searching for better solutions to the problem.

We have selected ABC algorithm because it is relatively simple to use with three control parameters. These parameters are the food sources (S), iterations numbers (Z), and threshold for converting the employed/onlooker bees into a scout (*limit*). In addition, the adaptation of ABC for new problems is easier than other swarm algorithms and has revealed better performance results for different types of problems [23].

The ABC algorithm performs the search process in three different stages [25]. First, in the "employed" stage, employed bees explore the search space to find better solutions. In the "onlooker" stage, onlooker bees wait for employed bees and follow them by using the solutions

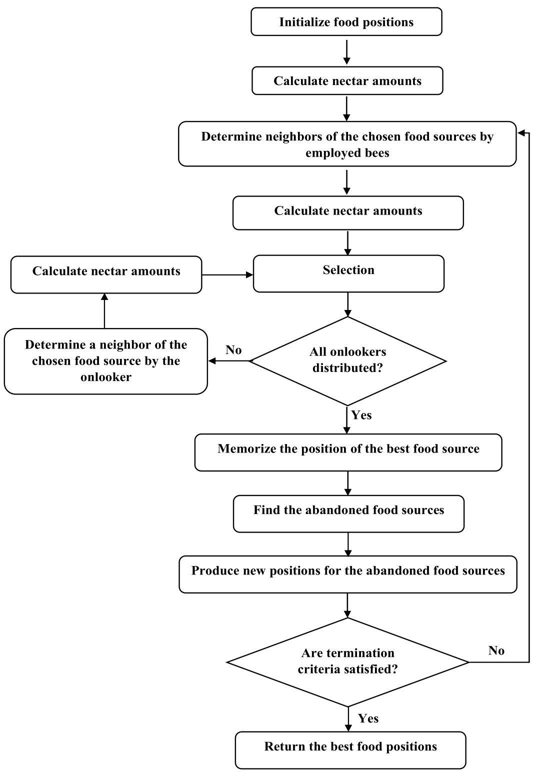

Initialize food positions
Calculate nectar amounts
Determine neighbors of the chosen food sources by employed bees
Calculate nectar amounts
Calculate nectar amounts
Selection
Determine a neighbor of the chosen food source by the onlooker
No
All onlookers distributed?
Yes
Memorize the position of the best food source
Find the abandoned food sources
Produce new positions for the abandoned food sources
Are termination criteria satisfied?
No
Yes
Return the best food positions

FIGURE 4. Flowchart of the ABC algorithm.

fitness values. In the employed and onlooker phases, bees remember only the best solution in each iteration. The final stage is the "scout" stage, where scout bees search for new solutions if all previous solutions are exhausted.

Initially, the employed bees are initialized randomly on the food sources. Then they search for new solutions by using the initial solutions. New solution v_{ij} can be found using

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

where x_{ij} is the previous solution, $j \in \{1, 2, 3, \ldots, D\}$, and D is the dimension, x_{kj} is a neighboring solution, $i, k \in \{1, 2, 3, \ldots, S\}$, S is the employed bees number $i \neq k$, and r is a random number $\in [-1, 1]$.

The employed bees compute the solutions quality of their solutions and remember the best solution that is better than the initial solutions.

The onlooker bees follow the employed bees. They wait for the employed bees to begin a new search. Each onlooker uses the employed bees' fitness value to choose the one to follow as shown bellow

$$P_i = \frac{F_i}{\sum_{n=1}^{S} F_n} \tag{2}$$

where F_i is the employed bee's solution fitness. S is the population size.

The onlooker bees compute the fitness values of their solutions and also remember the best solution. The employed and onlooker bees compare their best solutions and keep the best one of the two solutions.

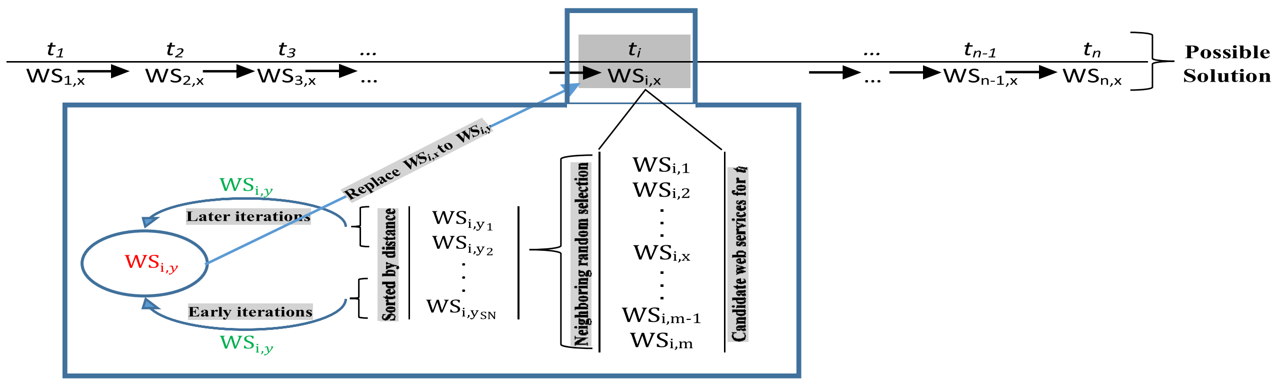

t1 t2 t3 ... ti ... tn-1 tn
WS1,x WS2,x WS3,x ... WSi,x ... WSn-1,x WSn,x
Possible Solution
Replace WSi,x to WSi,y
WSi,y
Later iterations
Early iterations
Sorted by distance
WSi,y1 WSi,y2 WSi,ySN
Neighboring random selection
WSi,1 WSi,2 WSi,x WSi,m-1 WSi,m
Candidate web services for ti

FIGURE 5. Selection and replacement process.

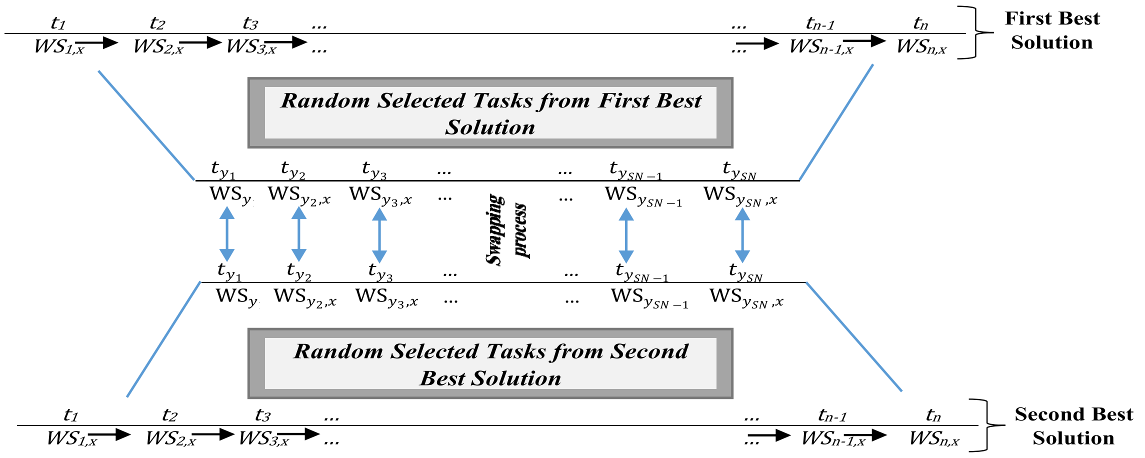

First Best Solution
Random Selected Tasks from First Best Solution
Swapping process
Random Selected Tasks from Second Best Solution
Second Best Solution

FIGURE 6. Swapping process.

The scout bees monitor the search processes of the employed and onlooker bees. If no further improvements in the solutions quality can be achieved, the employed bees are converted into scout bees. The scout bees then reinitialize the search randomly.

The ABC algorithm has two main mechanisms for searching called exploration and exploitation [26]. The purpose of the exploration mechanism is to guide the bees to reach the best local solutions. On the other hand, the exploitation focuses on guiding the algorithm to reach global optima [27]. In this study, we introduced an enhancement to balance these two mechanisms.

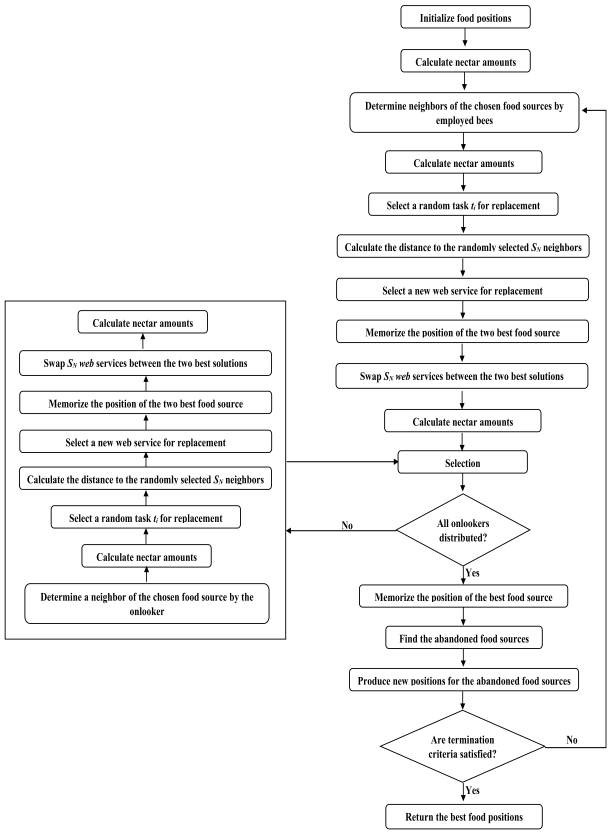
IV. THE PROPOSED TWO-STEP ABC

The ABC algorithm was designed for continuous problems. However, the QoS-aware WSS is a discrete optimization problem. Therefore, any proposed method that uses the ABC has to discretize the ABC algorithm. Initially, the bees in the QoS-aware WSS problem begin from the first task and jump until they reach task n. At each jump, they add one web service to the solution path. In subsequent iterations, they attempt to improve solutions quality and calculate solutions' limits if there are no improvements.

In this study, we introduce an additional enhancement to the enhanced ABC (EABC) algorithm [12]. This enhancement introduces a two-step search process. In the first step, the proposed algorithm uses the EABC selection and replacement strategies. The search process using employed and onlooker bees was modified to adopt selection and replacement strategies. The selection strategy picks a task from the best solution to replace one web service by another web service based on the distance between the neighboring web services S_N from this task. The value of S_N is experimentally tested based on [12] to be equal to 5. To do this, the selection strategy selects a random task t_i from the generated solutions. One of this task's web service is replaced by a neighboring web service of this task. Euclidean distance is used to calculate the distance to the randomly selected S_N neighbors, as shown in equation (3). Figure 5 presents the selection and replacement strategies in details [12].

$$d_{xj} = \sqrt{(C_x - C_j)^2 + (RT_x - RT_j)^2 + (T_x - T_j)^2 + (R_x - R_j)^2} \quad j \in [1, S_N] \tag{3}$$

where C_x and C_j are the web services' (x and j) cost; RT_x and RT_j are the web services' (x and j) response time; T_x and T_j are the web services' (x and j) throughput; and R_x and R_j are the web services' (x and j) reliability. S_N is the number of web services' neighboring nodes.



Initialize food positions
Calculate nectar amounts
Determine neighbors of the chosen food sources by employed bees
Calculate nectar amounts
Select a random task t_i for replacement
Calculate the distance to the randomly selected S_N neighbors
Select a new web service for replacement
Memorize the position of the two best food source
Swap S_N web services between the two best solutions
Calculate nectar amounts
Selection
All onlookers distributed?
No
Yes
Memorize the position of the best food source
Find the abandoned food sources
Produce new positions for the abandoned food sources
Are termination criteria satisfied?
No
Yes
Return the best food positions
Determine a neighbor of the chosen food source by the onlooker
Calculate nectar amounts
Select a random task t_i for replacement
Calculate the distance to the randomly selected S_N neighbors
Select a new web service for replacement
Memorize the position of the two best food source
Swap S_N web services between the two best solutions
Calculate nectar amounts

FIGURE 7. **The proposed algorithm flowchart.**

The distances of selected neighbors and the best web service are sorted. To select a new web service for replacement, we use

$$index = \left(\frac{rand}{i}\right) * (S_N) \tag{4}$$

where i is the iteration number, $rand$ is uniform random number $\in]0, 1]$. and S_N is the number of neighboring web services from same task t_i.

Equation (4) aids the proposed algorithm to maintain a balance between exploration and exploitation. In the early iterations, the algorithm selects a web service randomly from the farthest neighbors for replacement. In later iterations, it selects web service randomly from the nearest neighbors. This process encourages exploration in early iterations and exploitation in later iterations.

In the second step, we introduce a swapping process for the two best solutions. The aim of this process is to swap between the web services of two solutions to generate two new solutions. As illustrated in Figure 6, the proposed algorithm remembers the two best solutions for swapping. These two solutions are the two best solutions discovered globally. The number of web services to be swapped is similar to the number of neighbors in the first process (S_N). The selection of web services to be swapped is performed randomly. The result of this process is two new solutions. We then calculate the fitness values of these solutions and compare them to the best solution remembered by the algorithm. The algorithm retains only the best solution for further iteration.

Algorithm 1: Proposed Algorithm

```
Input: Algorithm parameters, $T=[t_1,t_2,...,t_n]$ set of tasks,
       each $t_i$ contains a set of web services=$[ws_1, ws_2,.., ws_m]$
Output: Best solution
Initialize: swarm size $S$, the $limit$s, maximum iterations $Z$,
       and neighboring web services size $S_N$
Begin
    Initialize $limit$ of each bee to 0
    $eb=ob= S/2$
    $sb=S$
    Initialize $eb$
    Compute the fitness
    for $iter$=0 → $Z$-1
        for $i$=1→ $eb$
            Produce new solution
            Compute the fitness of the new solution
            Execute the selection and replacement procedure
            Compute the fitness of the new solution $\acute{X}_i$ using $eq.5$
            if $fitness(\acute{X}_i)>fitness(X_i)$ then
                $X_i = \acute{X}_i$
            else
                $limit[i]=limit[i]+1$
            end
            Calculate the $prob[X_i]$ using $eq.2$
        end
        for $j$=1→ $ob$
            Select a solution according to $prob[]$
            Execute the selection and replacement procedure
            Compute the fitness of the new solution $\acute{X}_j$ using $eq.5$
            if $fitness(\acute{X}_j)>fitness(X_j)$ then
                $X_j = \acute{X}_j$
            else
                $limit[i]=limit[i]+1$
            end
        end
        for $i=1$→ $sb$
            if $limit[i]==limits$ then
                Initialize $sb_i$ randomly
                $limit[i]=0$
            end
        end
        Select the best two solutions from S$1$, $S2$
        for $r=1$→ $S_N$
            Generate a random number $R1 \in [1, n]$
            Swap $ws_{R1}^{s1}$ and $ws_{R1}^{s2}$
        end
        Keep best solution
    end
    return best solution
end
```

FIGURE 8. **Proposed algorithm.**

Similar to [12], solutions' fitness values are computed using preference-based method. In preference-based method, a multi-objective problem is converted into a single-objective problem using an aggregation function. The aggregation function used in the proposed method is shown below [12]

$$F = \left(\prod_{i=1}^{n} T_{ib} + \prod_{i=1}^{n} R_{ib} - \sum_{i=1}^{n} C_{ib} - \sum_{i=1}^{n} RT_{ib}\right) \tag{5}$$

where C refers to cost, RT refers to response time, T refers to throughput, and R refers to reliability. n refers to the number of tasks; and b refers to the employed or onlooker bees' selection for a web service to task i in each iteration.

The two steps above are utilized by both employed and onlooker bees. However, employed bees should recruit the onlooker bees to follow them based on the fitness values of their solutions using equation (2). Any employed bees that find no improvements for their solutions are reinitialized as scout bees. The flowchart of the proposed algorithm is shown in Figure 7. Figure 8 presents the proposed algorithm in details.

A. COMPLEXITY ANALYSIS

We present in this sub-section the complexity analysis of the proposed algorithm. To analysis the time complexity, we have six calculations: initialization, employed and onlooker search, fitness calculation, scouts search, selection and replacement process, and swapping process.

The initialization cost is $O(S^*N)$ where S is the populations size and N is the number of tasks. The employed and onlooker search cost complexity is $O(Z^*S^*N^*D)$ where Z is the iterations number, and D is the dimensionality. The fitness calculation is $O(Z^*S^*N^*D)$. The scouts search cost is $O(Z^*S)$. The time complexity of the selection and replacement process is $O(Z^*S^*S_N{}^*D)$ where S_N is the neighbors' number Finally, the swapping process time complexity is $O(Z^*S_N)$. Thus, the overall time complexity of the proposed method is $O(S^*N) + O(Z^*S^*N^*D) + O(Z^*S^*N^*D) + O(Z^*S) + O(Z^*S^*S_N{}^*D) + O(Z^*S_N) = O(Z^*S^*N^*D)$.

V. SIMULATION RESULTS

Simulations were conducted in this study. The experiments focused on comparing the proposed method to the EABC [12] and TBA [19] methods. The experiments were run on a PC with an Intel i7 processor and Windows 8 OS environment. The implementation was carried out in Java, where we directly used the Java code provided in [12] and [19].

A. SETTINGS

Similar to the empirical experiments in [12] and [19], we used two types of datasets based on the distribution of the tasks number and each task's web services. The first type of datasets has 10 tasks with different numbers of web services of 100, 200, 300, 400, 500, 600, 700, 800, 900, and 1000. The second type of datasets has 100 web services with different numbers of tasks of 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100. There were 30 datasets of each type. The datasets were artificially generated, similar to [19], and the values of the four QoS attributes were defined in the range [0, 1000].

TABLE 2. Experimental parameter settings.

Parameters	Value
S	100
Z	500
limits	100
β *(for TBA)*	0. 7
S_N	5

TABLE 2 lists the control parameters used by the EABC and TBA, which are similar to those used by the proposed method.

For each dataset, the experiments were repeated 30 times to guarantee the robustness of the proposed algorithm against random factors, which may affect the algorithm behavior. The evaluation metrics for the compared algorithms were the average quality of the best solutions and average execution time.

B. RESULTS

This section presents our simulation results. Figure 9 presents the results when the web services were changed (first type of datasets). Figure 10 presents the results when the tasks were changed (second type of datasets).

Figure 9(a) shows that the proposed algorithm consistently outperforms both the EABC algorithm and TBA in terms of solutions quality. However, in Figure 9(b), the results reveal that the proposed algorithm is closer to the EABC algorithm in terms of execution time.

Figure 10 (a) shows that the proposed algorithm also consistently outperforms the EABC algorithm and TBA in terms of solutions quality. In Figure 10 (b), the results reveal that the execution time of the proposed algorithm is comparable to that of the EABC algorithm and TBA.

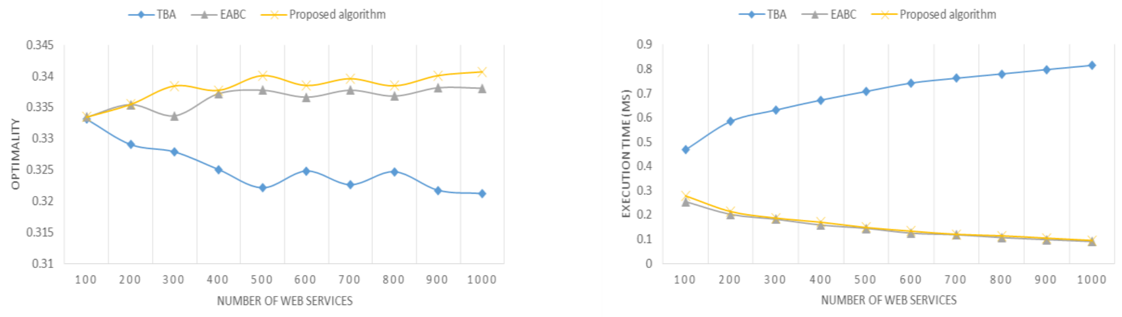
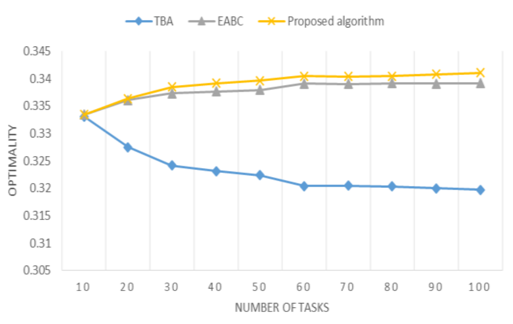
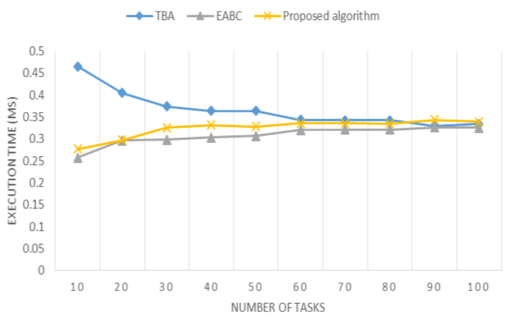


FIGURE 9. Algorithms' performance for different numbers of web services.



(a)



(b)

FIGURE 10. Algorithms' performance for different numbers of tasks. (a) Quality of solutions versus the number of tasks. (b) Execution time versus the number of tasks.

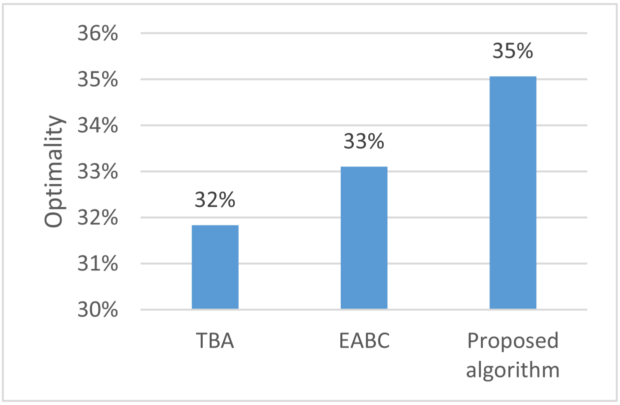


FIGURE 11. The solutions quality for different numbers of web services.

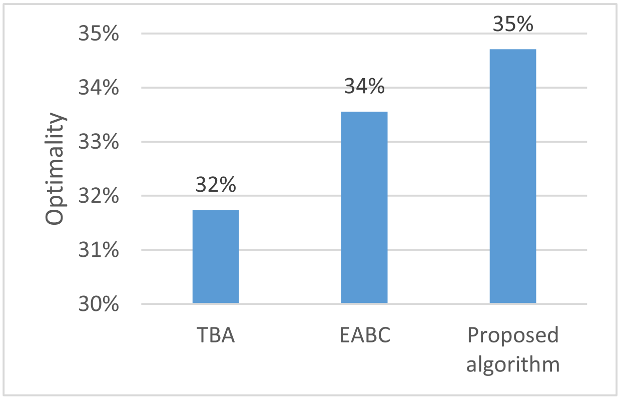


FIGURE 12. The solutions quality for different numbers of tasks.

In general, the results reveal that the proposed method provides better solutions quality compared to the EABC algorithm and TBA for both types of datasets. Recall that we tested the performance of all algorithms on two different types of datasets. The first type has different numbers of web services and the second type has different numbers of tasks. Furthermore, the results revealed that the proposed method is close to the EABC algorithm and better than the TBA in terms of execution time. This is because the proposed method simply adds an extra process to the EABC algorithm to enhance its performance.

In Figure 11 and Figure 12, we summarize the results on all datasets with 30 different executions. Recall that we test all algorithms using 60 datasets, which are divided into 2 different types (each one has 30 datasets). To show the results clearly we normalized the results using

$$Norm\left(alg_g\right) = \frac{Results(alg_g)}{\sum Results\_all\_algorithms} \times 100\% \qquad (6)$$

where g is TBA, EABC, or proposed algorithm.

Figure 11 summarizes the normalized experimental results for first type of datasets. The results indicate that the proposed algorithm achieved 35% which is better than EABC 33% and TBA 32% out of 100%.

Figure 12 summarizes the normalized experimental results for 30 executions for second type of datasets. The results also indicate that the proposed algorithm achieved 35% which is better than EABC 34% and TBA 32% out of 100%.

VI. CONCLUSIONS

The web service selection problem is concerned with finding the best web services combination for a given set of user requirements. It is a challenging problem as it is an NP-hard problem. This paper introduced a novel enhancement for the ABC algorithm. The enhancement is based on a two-step process. First, we use neighboring nodes to enhance the performance of the ABC algorithm. This encourages exploration in early iterations, where bees have no knowledge regarding the search space, and exploitation in later iterations to exploit bees' knowledge regarding the search space. Second, we proposed a swapping method for ABC algorithm performance enhancement by swapping portions of the two best solutions randomly. Our experiments revealed that the proposed method successfully enhanced the ABC algorithm performance and found better solutions compared to other algorithms. We tested all algorithms using 60 different datasets. The datasets were divided into two groups based on the numbers of tasks and numbers of web services. Our experiments using 30 different executions showed that the proposed algorithm outperformed TBA by 6% and EABC by 3% in terms of solutions quality.

REFERENCES

[1] T. Andrews *et al. Business Process Execution Language for Web Services v1.1. IBM DeveloperWorks*. Accessed: May 2003. [Online]. Available: ww.ibm.com/developerworks/library/specificatio/ws-bpel

[2] J. Li, X. Luo, Y. Xia, Y. Han, and Q. Zhu, "A time series and reduction-based model for modeling and QoS prediction of service compositions," *Concurrency Comput., Pract. Exper.*, vol. 27, no. 1, pp. 146–163, 2015.

[3] W. Li, Y. Xia, M. Zhou, X. Sun, and Q. Zhu, "Fluctuation-aware and predictive workflow scheduling in cost-effective Infrastructure-as-a-Service clouds," *IEEE Access*, vol. 6, pp. 61488–61502, 2018.

[4] E. A. Marks and M. Bell, *Service-Oriented Architecture (SOA): A Planning and Implementation Guide for Business and Technology*. Hoboken, NJ, USA: Wiley, 2006.

[5] A. Bouguettaya, Q. Z. Sheng, and F. Daniel, Eds., *Web Services Foundations*. New York, NY, USA: Springer, 2014.

[6] E. Newcomer and G. Lomow, *Understanding SOA With Web Services* (Independent Technology Guides). Reading, MA, USA: Addison-Wesley, 2004.

[7] A. Gustavo, F. Casati, H. Kuno, and V. Machiraju, *Web Services: Concepts, Architectures and Applications*. New York, NY, USA: Springer-Verlag, Sep. 2003.

[8] T. Erl, *SOA: Principles of Service Design*, vol. 1. Upper Saddle River, NJ, USA: Prentice-Hall, 2008.

[9] S. Dustdar and M. P. Papazoglou, "Services and service composition—An introduction," *Inf. Technol.*, vol. 50, no. 2, pp. 86–92, 2008.

[10] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for Web services selection with end-to-end QoS constraints," *ACM Trans. Web*, vol. 1, no. 1, 2007, Art. no. 6.

[11] T. Yu and K.-J. Lin, "Service selection algorithms for Web services with end-to-end QoS constraints," *Inf. Syst. E-Bus. Manage.*, vol. 3, no. 2, pp. 103–126, 2005.

[12] F. Dahan, K. E. 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.

[13] F. Dahan, K. E. Hindi, and A. Ghoneim, "An adapted ant-inspired algorithm for enhancing Web service composition," *Int. J. Semantic Web Inf. Syst.*, vol. 13, no. 4, pp. 181–197, 2017.

[14] N. Ben Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas, and V. Issarny, "QoS-aware service composition in dynamic service oriented environments," in *Proc. Int. Middleware Conf.*, IL, USA, vol. 5896, Dec. 2009, pp. 123–142.

[15] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, vol. 16. Hoboken, NJ, USA: Wiley, 2001.

[16] G. Kousalya, D. Palanikkumar, and P. R. Piriyankaa, "Optimal Web service selection and composition using multi-objective bees algorithm," in *Proc. IEEE 9th Int. Symp. Parallel Distrib. Process. Appl. Workshops (ISPAW)*, May 2011, pp. 193–196.

[17] W. Zhenwu and B. Wan, "Web services selection based on discrete artificial bee colony algorithm," *Int. J. Digit. Content Technol. Appl.*, vol. 6, p. 586, Nov. 2012.

[18] J. He, L. Chen, X. Wang, and Y. Li, "Web service composition optimization based on improved artificial bee colony algorithm," *J. Netw.*, vol. 8, no. 9, pp. 2143–2149, 2013.

[19] X. Wang, Z. Wang, and X. Xu, "An improved artificial bee colony approach to QoS-aware service selection," in *Proc. IEEE 20th Int. Conf. Web Services (ICWS)*, Jun./Jul. 2013, pp. 395–402.

[20] C. Zhang and B. Zhang, "A hybrid artificial bee colony algorithm for the service selection problem," *Discrete Dyn. Nature Soc.*, vol. 2014, Dec. 2014, Art. no. 835071.

[21] V. R. Chifu, C. B. Pop, I. Salomie, M. Dinsoreanu, A. N. Niculici, and D. S. Suia, "Bio-inspired methods for selecting the optimal Web service composition: Bees or cuckoos intelligence?" *Int. J. Bus. Intell. Data Mining*, vol. 6, no. 4, pp. 321–344, 2011.

[22] R. Liu, Z. Wang, and X. Xu, "Parameter tuning for ABC-based service composition with end-to-end QoS constraints," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun./Jul. 2014, pp. 590–597.

[23] B. Akay and D. Karaboga, "Artificial bee colony algorithm for large-scale problems and engineering design optimization," *J. Intell. Manuf.*, vol. 23, no. 4, pp. 1001–1014, 2012.

[24] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Dept. Fac. Comput. Sci. Eng., Erciyes Univ., Kayseri, Turkey, Tech. Rep.-TR06, 2005.

[25] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Global Optim.*, vol. 39, no. 3, pp. 459–471, Apr. 2007.

[26] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: A gravitational search algorithm," *J. Inf. Sci.*, vol. 179, no. 13, pp. 2232–2248, 2009.

[27] P. Civicioglu and E. Besdok, "A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms," *Artif. Intell. Rev.*, vol. 39, no. 4, pp. 315–346, Apr. 2013.

FADL DAHAN received the B.Sc. degree from Thamar University, Yemen, the M.Sc. degree from King Saud University, and the Ph.D. degree from the Department of Computer Science, King Saud University. He is also a Faculty Member of the Department of Computer Science, Faculty of Computer Science in Torba, University of Taiz, Yemen. His research interests include optimization and swarm intelligence.

HASSAN MATHKOUR received the Ph.D. degree from The University of Iowa, USA. He is a currently a Professor with the Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia. He has authored over 100 research articles in journals and conferences. He held several administration posts, including the Dean, the Associate Dean, the Department Chair, the Director of the Research Center, and the Head of the joint Ph.D. Program. He also serves as an IT Consultant. His research interests include intelligent systems, peer-to-peer systems, modeling and analysis, database management systems, data mining, knowledge management, e-learning, and bioinformatics.

MOHAMMED ARAFAH received the Ph.D. degree in computer engineering from the University of Southern California, Los Angeles, CA, USA. He is currently an Associate Professor with the Department of Computer Engineering, King Saud University, Riyadh, Saudi Arabia. He has published in the areas of multistage interconnection networks, MPLS networks, and LTE networks. His current research interests include cooperative communication, 5G mobile communications, software defined radios, and multiple antenna systems.