IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 21, 2019, accepted June 14, 2019, date of publication July 9, 2019, date of current version August 5, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2927769

Enhancement of Ant Colony Optimization for QoS-Aware Web Service Selection

HASHEM ALAYED[1,2], FADL DAHAN[3], TAHA ALFAKIH[4], HASSAN MATHKOUR[1], AND MOHAMMED ARAFAH[5]

[1]Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia
[2]Vice Deanship of Chairs (VCH), King Saud University, Riyadh 11543, Saudi Arabia
[3]Department of Information System, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, AlKharj 11942, Saudi Arabia
[4]Department of Information System, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia
[5]Department of Computer Engineering, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia

Corresponding author: Fadl Dahan (f.naji@psau.edu.sa)

The authors are grateful to King Saud University for funding this research project through Vice Deanship of Scientific Research Chairs (VCH).

ABSTRACT In service-oriented computing, web services composition is the process of translating user requirements into a workflow. This workflow comprises many tasks, each of which includes an abstract definition for some of the user requirements. Web services can be aggregated to handle the workflow. Many of these services are available from various providers for each task; they are referred to, in aggregate, as the candidate list. The web service selection (WSS) problem centers on selecting the best service from these candidates based on the quality of service (QoS) features. In this paper, we propose an enhancement to the ant colony optimization (ACO) algorithm based on a swap concept for the QoS-aware WSS problem. The aim of the enhancement to the ACO is to avoid the trap of local optima and reduce the search duration. We believe that the integration of many potent solutions will help the ACO algorithm yield a better solution and avoid stagnation. Several experiments were conducted to compare the proposed algorithm with the ACO and flying ACO (FACO) algorithms. Two different types of experiments using 22 datasets were done with 30 independent repetitions. The first type of experiment's results shows that the proposed algorithm is better than ACO by 12% and FACO by 11% in terms of quality of solutions. The results in the second type of experiment show that the proposed algorithm continuously outperforms both algorithms in terms of quality of solutions.

INDEX TERMS Service-oriented computing (SOC), web services composition (WSC), web service (WS), web service selection (WSS), ant colony optimization (ACO).

I. INTRODUCTION

Service-oriented computing (SOC) introduces a new paradigm for distributed applications. This paradigm changes the way applications are integrated, designed and delivered. Web services provide autonomous, low-cost, reusable, and platform independent applications with straightforward maintenance for SOC [1]. These services use the web infrastructure to interact [2]–[4].

Providers design web services and store the description into a registry for the end users. Meanwhile, users have requirements, which translate into a workflow. This workflow in turn contains many tasks, each of which containing an abstract definition for some of these requirements. The workflow can be handled, and the requirements satisfied, by a single or several web services (WS). If a web service is not sufficient, others can be integrated [5].

The web service selection (WSS) aims to select the best combination of web services to complete the tasks tied to the workflow. This combination is ultimately the plan that the user should consider. The WSS problem is presented in various areas such as cloud service composition [6], smart city services [7]–[9].

Researchers introduced several algorithms to solve the WSS problem with a high-quality solution and a lower execution time. Ant colony optimization (ACO) algorithms have been used successfully to solve many problems such as the travelling salesman person [10], the multidimensional Knapsack problem [11], and the semantic web [12].

The associate editor coordinating the review of this manuscript and approving it for publication was Miltiadis Lytras.

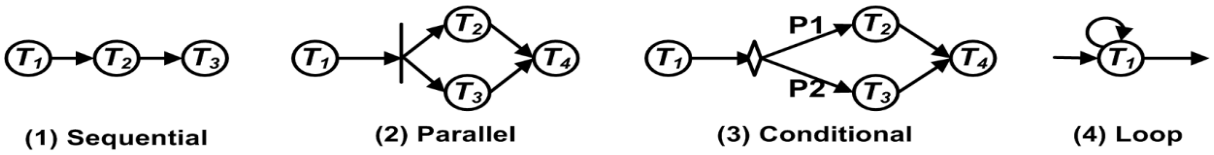

FIGURE 1. Workflow patterns.

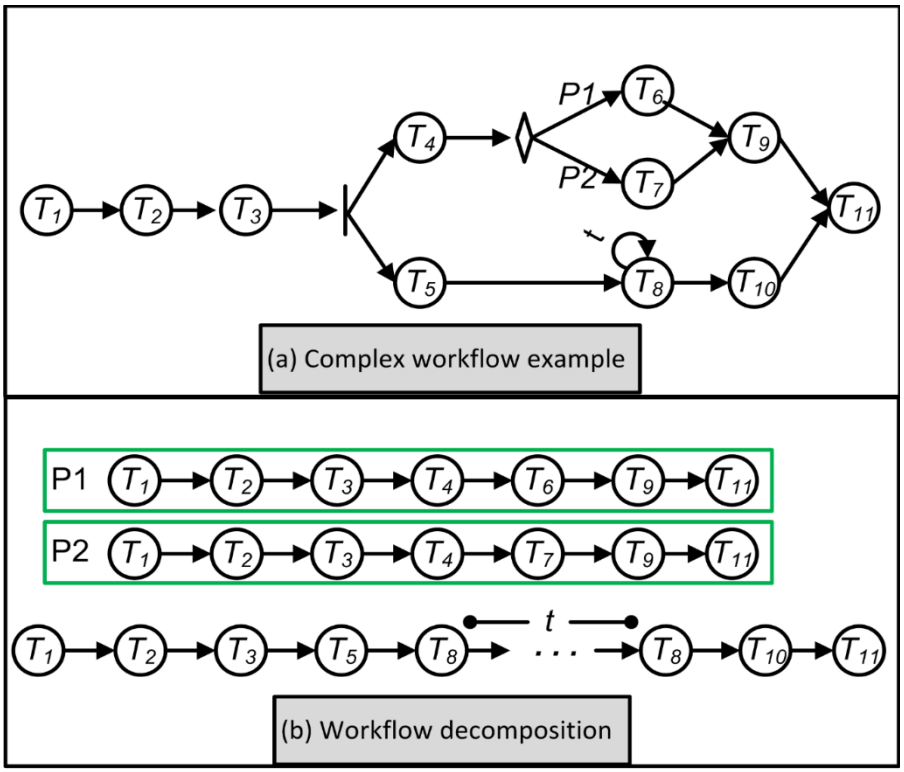

FIGURE 2. Workflow types and decomposition.

The aim of this paper is to contribute to ACO by 1) avoiding local minima traps using a swap process 2) searching for high quality solutions by increasing diversity 3) reducing the computational time to find these solutions. Section II introduces the problem definition. Section III discusses related work, involving the use of ACO in the context of the quality of service-aware (QoS-aware) WSS problem. Section IV presents the concept behind ACO. In Section V, we introduce the proposed algorithm. Section VI reports the results of the experiments. Finally, section VII concludes this paper.

II. THE WSS PROBLEM

The process of processing user requirements into a workflow and identifying web services to handle this workflow is called web service composition (WSC). With the rapid growth in providers, several web services with similar functionality can be used for each task. The QoS constrains represent the non-functional features associated with these web services [13].

User requirements comprise different patterns; hence they may be executed sequentially, concurrently, conditionally, or repeatedly [14], [15]. **Figure 1** [15] represents these patterns.

These can be further categorized into two forms: simple or complex. The simple workflow merely consists of sequentially ordered tasks while the complex workflow comprises one or more of these patterns. **Figure 2** (a) shows a complex workflow example. Complex workflow can be decomposed into many simple ones as shown in **Figure 2** (b). Following [14], [15], each simple workflow represents an individual problem, which indicates that these workflows do not need to be recomposed. In addition, a fitness value is assigned to each one and the best fitness represents the optimal solution to be introduced to clients.

The workflow representation consists of a set of tasks (n) containing the abstract definitions of the requirements in a specific order. The retrieval process conducts a search in

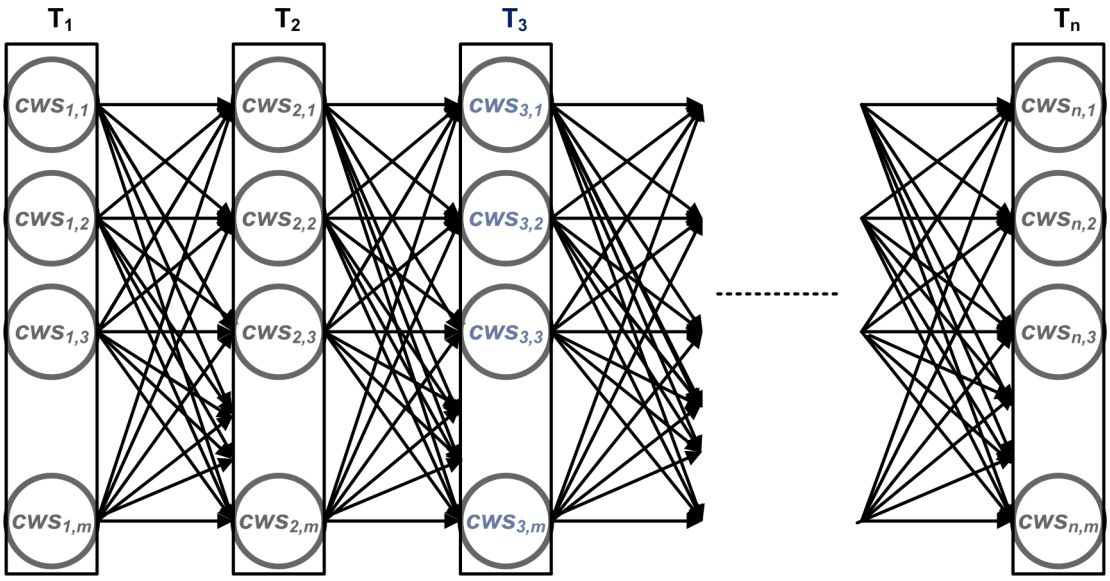

FIGURE 3. WSS problem representation.

a service repository for the web services that are consistent with these definitions. As a result, several web services (m) from various providers can be retrieved for each task, and these services are referred to as the candidate list. **Figure 3** shows the directed graph representation of this problem where the number of available solutions equals m^n [16], [17]. In **Figure 3**, T_i refers to the task i and $i = \{1, 2, 3, \ldots, n\}$ *i*n the workflow , n is the number of tasks, $cws_{i,j}$ refers to the candidates web services (CWS) for task T_i, $j = \{1, 2, 3, \ldots, m\}$, and m is the number of web services.

Figure 3 shows that the solution must contain a combination of web services for all tasks, which forms an executable plan. The ultimate plan selection is output to the users to meet their requirements. Any algorithm used should compare the QoS values for each executable plan and select the best one. There are many QoS for the WSS problem such as cost, response time, availability, and reliability. The calculation of these attributes is differentiated as shown in TABLE 1. In this Table, n is the number of tasks and j is the WS selected for task i.

In this work, we use four QoS attributes with different objective values similarly to [18]. The objective for the cost and response time is to find a minimum, conversely a maximum will be sought concerning availability and reliability.

The WSS problem must now consider the four QoS attributes to select the best execution plan for users, formulating the problem as a multi-objective multi-dimensional combinatorial optimization. In addition, it is proven that it is an NP-hard problem [19], [20].

III. RELATED WORK

QoS-aware WSS is a multi-objective multi-dimensional problem which consists of selecting WS between a set of candidates based on QoS attributes. The search space of the problem increases exponentially because there are several web services available from various providers for each task. Many researchers proposed solutions to this problem using various algorithms. In this section, we review these studies, as directly related to this work.

TABLE 1. Some aggregation models for QoS values calculation.

QoS Criteria	Expression
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 dynamic ant colony optimization algorithm (DACO) is introduced in [21]. The problem is handled by creating a composition graph, and then DACO looks for the shortest path. Qiqing *et al.* [22] generated a QoS model for the problem, and then handled it as the optimization of a multi-objective by using a multi-objective ant colony optimization (MOACO).

Chaos factor is used to solve the problem with the ant-based algorithm introduced in [23]. The new algorithm is called a multi-objective chaos ant colony optimization (MOCACO). The justification for the use of ACO with such problems was introduced in [24]. The impact of the parameters on the improvement in ACO performance for the web services composition problem was introduced in [25]. The Pareto optimality is used to address the problem in [26]. Zhang *et al.* [27] proposed a method to decompose complex

composition workflows, and then used the ACO algorithm. Zheng *et al.* [28] proposed a graph representation for web services composition where the ants search for parallel execution paths. Dahan *et al.* [16] introduced an enhancement to ACO where the best ant in each iteration becomes a flying ant. The enhancement was called the flying ant colony optimization (FACO). The flying ant shared its amount of pheromone with its neighbors to increase the chance of being visited in feature iterations. Qi *et al.* [29] proposed a new method that uses the skyline to delete redundant web services, For the remaining web services, the ACO is used to select the best web services to meet user requirements.

A genetic algorithm with an ant-inspired algorithm is used in [30]. Yang *et al.* [31] proposed a dynamic ant-colony genetic hybrid algorithm (DAAGA) for cloud service composition optimization. Mobile agents used to enhance the search mechanism of ACS are proposed in [32]. The trust degree is used in [33], leading to an adaptive ant colony optimization (AACO). A clustering ant colony selection (CACS) is introduced in [34]. Le and Nguyen [35] proposed a novel max-min ant system (MMAS) to search for a solution. Shen and Yuan [36] proposed a gathering mechanism for QoS information using peers from the candidate web services, and then used ACS.

Many researchers addressed the semantic WS composition problem, which centers on discovering and retrieving web services using analogy and then ant-inspired algorithms to find a satisfactory solution [37]–[43].

Most of the works reviewed above aim to solely enhance the exploitation process of the ACO [44], except the FACO algorithm, which focuses on enhancing exploration and exploitation. In fact, exploration aims at helping ants to reach the best local solutions while exploitation aims at helping the algorithm to reach global optima [45]. In this work we aim to improve both exploration and exploitation strategies to help ACO to avoid premature stagnation using swapping. The swapping consists of producing two new solutions (exploration), but with these solutions being exploited through the features of the two non-randomly generated best solutions (exploitation)

IV. ACO ALGORITHM

It is a nature-inspired meta-heuristic algorithm similar to bee colony optimization, particle swarm optimization…etc. Meta-heuristic algorithms consist of finding the best possible solution within the lowest execution time for a given problem. It simulates real ants' behavior while foraging. In practice, ants search for food in a manner that guarantees finding the nearest food source. They leave from the nest in a random manner, and they deposit a chemical called pheromone in their paths. The pheromone represents the communication mechanism between them as it incites other ants to follow them. The nearest food source would logically be tied to more pheromones compared with the farthest one. The ACO algorithm flowchart [10], [46], [47] is shown in **Figure 4** [48].

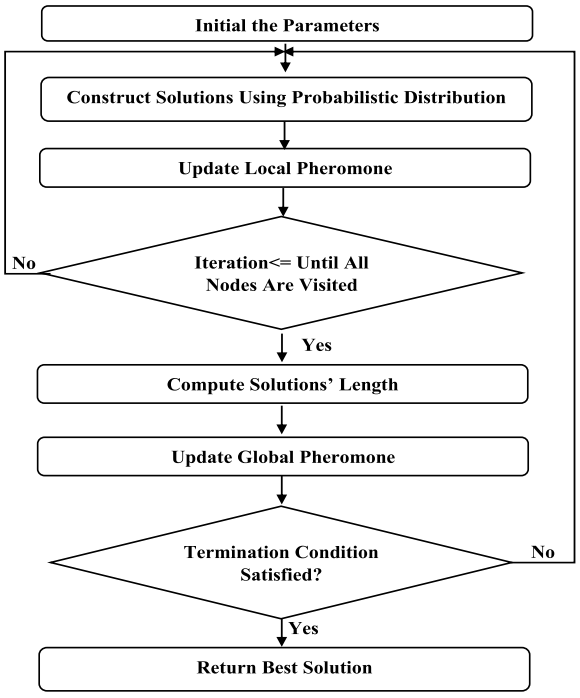

FIGURE 4. Flowchart of ACO algorithm.

In ACO algorithms, all ants are initialized randomly and then search for a potential solution. In addition, the amount of pheromone is initialized to a constant amount. In each iteration, each ant constructs its solution by moving on unvisited nodes until reaching the goal based on nodes distance τ and last experience η (pheromone). To simulate this process in the ACO algorithm, ants use the following equation.

$$P_{ij}^{k}(t) = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}} if\ j \in N_i^k \quad (1)$$

where α and β are the coefficient parameters used to determine the pheromone and local heuristic importance. $\mathcal{N}_i^k$ is the list of unvisited nodes from node i for ant k.

Equation 1 uses α and β to balance last experience (exploitation) based on pheromone and random search (exploration) based on distance, respectively. The amount of pheromone is updated using the evaporation rate ρ while ants move from one node to another using equation 2. This update is called the local pheromone update.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\tau_0 \quad (2)$$

where τ_0 is the initial pheromone value. ρ is the evaporation rate.

At the end of each iteration, the ants assess the quality of the solution they just constructed. Thus, the ACO uses a greedy selection method to retain only the best ant, the one that obtained the best solution. This ant then has the ability to update the pheromone trails on its path. This process is called a global pheromone update and is described in equation 3.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}(t) \quad (3)$$

In which:

$$\Delta\tau_{ij}(t) = \begin{cases} {}^{1}/_{L^{gb}}(t) & \text{if arc } (i,j) \in \text{the best tour} \\ \tau_{ij} & \text{otherwise} \end{cases} \quad (4)$$

where $L^{gb}(t)$ is the tour length for the globally best tour.

The ACO algorithm suffers from the stagnation problem [49] because pheromones accumulate on the most visited paths and the probability of unvisited paths forming decreases constantly.

In this work, we improve the ACO algorithm to enhance diversity and avoid stagnation using the swap concept for the solutions selected.

V. PROPOSED ALGORITHM

ACO is an ant-inspired algorithm introduced for combinatorial problems. The ants in a QoS-aware selection problem must start from any WS in the first task and then move on to the next task, until they get any WS in the last task. The selection is based on the QoS attributes and the amount of pheromone.

In this work, we introduce two different enhancements to the ACO algorithm focusing on performance and on overcoming the stagnation problem. First, we introduce a swapping process that lets ACO remember two solutions temporarily and then, in a final step, return the best one only. This process aims to swap a specific number of web services in the two best solutions found so far to generate two new solutions. The selection of the web services to be swapped is performed randomly. Then, the fitness values of these new solutions are calculated and compared to the best solution remembered by the algorithm. The incorporation of the two best solutions will help the ACO algorithm to get a better solution and avoid the stagnation problem as each one of these solutions can contribute parts that can produce excellent solutions if they are merged.

As a second enhancement, we adopted a multi-pheromone version of the ACO algorithm where each QoS constraint (cost, response time, availability, and reliability) is tied to a pheromone value of its own. This improves the chances for ants to consider more options in the solution space (i.e., increases the chance for exploration).

In standard ACO, the pheromone is represented by a single value. In this work, we represent pheromones using multiple values, according to the number of QoS attributes as $[\tau^{c}_{(i,x)(i+1,x')}, \tau^{RT}_{(i,x)(i+1,x')}, \tau^{A}_{(i,x)(i+1,x')}, \tau^{R}_{(i,x)(i+1,x')}]$ where $\tau^{c}_{(i,x)(i+1,x')}$ is the amount of pheromone to move from WS x at task i to WS x' at task $i+1$ for the cost (C) feature, and RT is for response time, A is for availability and R for reliability. The idea here is to consider the importance of each QoS feature and to become an effective guide while searching for a better solution.

The transition process is the process of selecting the next node to move on while building the tour. For basic ACO this process is shown in equation 1. In the proposed algorithm, this equation changes based on the new pheromone representation for ant k at time t, as follows:

$$P^{k}_{(i,x)(i+1,x')}(t) = \frac{\left[\tau_{(i,x)(i+1,x')}\right]^{\alpha}\left[\eta_{(i,x)(i+1,x')}\right]^{\beta}}{\sum_{l\in m}\left[\tau_{(i,x)(i+1,l)}\right]^{\alpha}\left[\eta_{(i,x)(i+1,l)}\right]^{\beta}} \quad (5)$$

where

$$\tau_{(i,x)(i+1,x')} = \tau^{A}_{(i,x)(i+1,x')} + \tau^{R}_{(i,x)(i+1,x')} \mp \tau^{c}_{(i,x)(i+1,x')} + \tau^{RT}_{(i,x)(i+1,x')}$$

$$\eta_{(i,x)(i+1,x')} = (\eta^{A}_{(i,x)(i+1,x')} + \eta^{R}_{(i,x)(i+1,x')}) - (\eta^{c}_{(i,x)(i+1,x')} + \eta^{RT}_{(i,x)(i+1,x')}).$$

The new formula helps ants to consider the value of the QoS features individually. This technique allows ants to explore the search space efficiently compared to the single-pheromone aggregation of these features.

When an ant k moves from each task to the next it updates the pheromone locally on its path. In the proposed algorithm, the updating changes to consider the new pheromone distribution based on the QoS features using equations 6, 7, 8 and 9.

Equation 6 shows the local pheromone update for the cost feature.

$$\tau^{C}_{(i,x)(i+1,x')}(t+1) = (1-\rho)\tau^{C}_{(i,x)(i+1,x')}(t) + \rho\tau^{C}_{0} \quad (6)$$

The local pheromone update for the response time feature is shown in Equation 7.

$$\tau^{RT}_{(i,x)(i+1,x')}(t+1) = (1-\rho)\tau^{RT}_{(i,x)(i+1,x')}(t) + \rho\tau^{RT}_{0} \quad (7)$$

Equation 8 shows the local pheromone update for the availability feature.

$$\tau^{A}_{(i,x)(i+1,x')}(t+1) = (1-\rho)\tau^{A}_{(i,x)(i+1,x')}(t) + \rho\tau^{A}_{0} \quad (8)$$

Equation 9 shows the local pheromone update for the reliability feature.

$$\tau^{R}_{(i,x)(i+1,x')}(t+1) = (1-\rho)\tau^{R}_{(i,x)(i+1,x')}(t) + \rho\tau^{R}_{0} \quad (9)$$

At the end of each iteration, all ants complete the construction of the possible solutions and then calculate the fitness of their solutions. Each possible solution contains n concrete web services adhering to the abstract definition of the workflow. The calculation formula [16] for the fitness of the ants' solution is:

$$\mathrm{F} = \Big(\prod_{i=1}^{n} \mathrm{A}^{k}_{(i,x)(i+1,x')} + \prod_{i=1}^{n} \mathrm{R}^{k}_{(i,x)(i+1,x')} - \sum_{i=1}^{n} \mathrm{C}^{k}_{(i,x)(i+1,x')} - \sum\nolimits_{i=1}^{n} \mathrm{RT}^{k}_{(i,x)(i+1,x')}\Big) \quad (10)$$

where n is the number of tasks. K refers to the ants' number with $1 \leq k \leq S$ and S is the swarm size.

The ACO memorizes the best solution at each iteration and allows the best ant to update the amount of pheromone on its path. Therefore, the global pheromone update formula also changes to consider QoS features, as does the local pheromone update. Equations 11, 12, 13 and 14 are

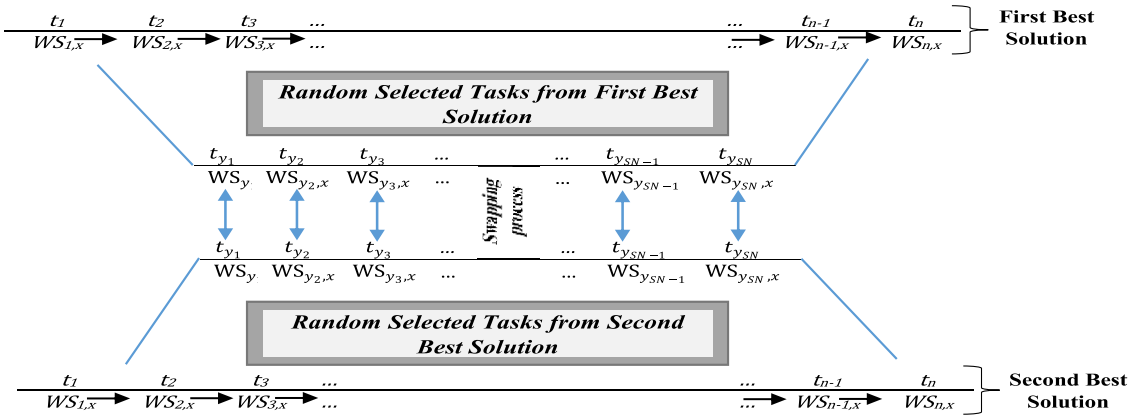

FIGURE 5. Swapping process.

used to update the amount of pheromone globally for each QoS feature.

Equation 11 shows the global pheromone update for the cost feature.

$$\tau^{C}_{(i,x)(i+1,x')}(t+1) = (1-\rho)\tau^{C}_{(i,x)(i+1,x')}(t) + \rho\Delta\tau^{C}_{(i,x)(i+1,x')}(t) \quad (11)$$

where:

$$\Delta\tau^{C}_{(i,x)(i+1,x')} = \begin{cases} \dfrac{1}{\sum_{i=1}^{n} C^{best}_{(i,x)(i+1,x')}} & \text{if } WS_x \text{ and } WS_{x'} \in \text{best path} \\ 0 & \text{otherwise} \end{cases}$$

Equation 12 shows the global pheromone update for the response time feature.

$$\tau^{RT}_{(i,x)(i+1,x')}(t+1) = (1-\rho)\tau^{RT}_{(i,x)(i+1,x')}(t) + \rho\Delta\tau^{RT}_{(i,x)(i+1,x')}(t) \quad (12)$$

where:

$$\Delta\tau^{RT}_{(i,x)(i+1,x')} = \begin{cases} \dfrac{1}{\sum_{i=1}^{n} RT^{best}_{(i,x)(i+1,x')}} & \text{if } WS_x \text{ and } WS_{x'} \in \text{best path} \\ 0 & \text{otherwise} \end{cases}$$

Equation 13 shows the global pheromone update for the availability feature.

$$\tau^{A}_{(i,x)(i+1,x')}(t+1) = (1-\rho)\tau^{A}_{(i,x)(i+1,x')}(t) + \rho\Delta\tau^{A}_{(i,x)(i+1,x')}(t) \quad (13)$$

where:

$$\Delta\tau^{A}_{(i,x)(i+1,x')} = \begin{cases} \prod_{i=1}^{n} A^{best}_{(i,x)(i+1,x')} & \text{if } WS_x \text{ and } WS_{x'} \in \text{best path} \\ 0 & \text{otherwise} \end{cases}$$

Equation 14 shows the global pheromone update for the reliability feature.

$$\tau^{R}_{(i,x)(i+1,x')}(t+1) = (1-\rho)\tau^{R}_{(i,x)(i+1,x')}(t) + \rho\Delta\tau^{R}_{(i,x)(i+1,x')}(t) \quad (14)$$

where:

$$\Delta\tau^{R}_{(i,x)(i+1,x')} = \begin{cases} \prod_{i=1}^{n} R^{best}_{(i,x)(i+1,x')} & \text{if } WS_x \text{ and } WS_{x'} \in \text{best path} \\ 0 & \text{otherwise} \end{cases}$$

In the proposed algorithm, we postpone the global pheromone update to the end of the swapping process and select the two best ants for the swapping process; this process aims to merge the current two solutions to produce two new solutions. **Figure 5** presents the swapping process used in the proposed algorithm. In **Figure 5**, the proposed algorithm remembers the two best solutions for swapping. These two solutions are the two best solutions discovered globally. The number R of web services to be swapped is generated randomly within $[1,n]$ where n is the number of tasks. The selection of web services to be swapped is performed randomly. The swapping involves the web services in the first solution and the web services in the second solution.

The fitness of the solutions is calculated and compared with that of the prevailing best solution. The best of the two is kept and then the related global pheromone update is started. The algorithm is described in detail in **Figure 6**. The proposed algorithm only needs one pass for a selected task in the swapping process.

A. COMPLEXITY ANALYSIS

This sub-section presents the complexity analysis of the proposed algorithm. There are five processes in the proposed algorithm that should be considered for the time complexity analysis. These processes are the initialization, the ants search, the local pheromone update, the fitness calculation, and the swapping process.

Input: Algorithm parameters, $T=[t_1, t_2, ..., t_n]$ a set of tasks,
each t_i contains a set of candidates$=[ws_1, ws_2, .., ws_m]$
Output: Concrete executable plan
begin
  Initialize swarm size S, maximum iterations Z
  Repeat
    Initialize each ant randomly on t_1
    for $i=1..\ S$ ***do***
      for $j=2..\ n$ ***do***
        Select probabilistically the next ws from t_j using *eq.5*
        Apply a local pheromone update using *eq.6, eq.7, eq.8* and *eq.9*
      end
      Compute the solution fitness using *eq.10*
    end
    Generate a random number $R \in [1, n]$
    Select the best two solutions from best ants k_1, k_2
    for $r=1..\ R$ ***do***
      Generate a random number $R_1 \in [1, n]$
      Swap $\boldsymbol{ws}_{R_1}^{k_1}$ and $\boldsymbol{ws}_{R_1}^{k_2}$
    end
    Compute the fitness for new solutions using *eq.10*
    Compare the last best solution with new once and keep the best
    Apply a global pheromone update for best ant path using *eq.11, eq.12, eq.13* and *eq.14*
  end
end

FIGURE 6. Proposed method.

The initialization time is O($S*n$) where S is the number of ants and n is the number of tasks. The ants' searching time complexity is O($Z*S*n*D$) where Z is the number of iterations and D is the number of QoS attributes. The local pheromone update time complexity is O($n*D$). The fitness calculation time complexity is O($n*D$). The time complexity of the swapping process is O(R) where R is the number of neighbors. The overall time complexity of the proposed algorithm is O($S*n$) + O($Z*S*n*D$) (O($n*D$) + O($n*D$) + O(R)) = O($Z*S*n*D$).O($n*D$).

VI. EXPERIMENTAL RESULTS

We conducted an experiment to compare the proposed algorithm with ACO and FACO. We proceeded to the implementation using MATLAB 8.4 on an Intel®i7 and Windows 7 with two different datasets as in FACO. As far as we know, there are no datasets for the QoS-aware WSC problem that have the four QoS attributes that were used in these experiments. These datasets were generated artificially with different tasks number. For each task, a set of random QoS values were assigned to its candidates. The ranges of these values were chosen as similar to the ranges used in [16] as follows: C and RT were between *1* and *50* while A and R were between *0* and *1*. TABLE 2 shows the number of tasks for each dataset with the number of candidates.

TABLE 2. Number of tasks and CWS of each dataset.

Dataset 1	Tasks	50
	Candidates per task	30
Dataset 2	Tasks	20
	Candidates per task	15

We conducted additional experiments using various artificial datasets. These were generated similarly to the datasets in [16]. The total number of new datasets was 20, with 10 of them having a fixed number of tasks and different numbers of candidate web services for each dataset in the range [100–1000]. The remaining 10 datasets had a fixed number

TABLE 3. Number of tasks and WSs for the new artificial datasets.

Dataset	*Tasks*	*WSs/Task*	*Dataset*	*Tasks*	*WSs/Task*
Dataset 3	10	100	*Dataset 13*	10	100
Dataset 4	10	200	*Dataset 14*	20	100
Dataset 5	10	300	*Dataset 15*	30	100
Dataset 6	10	400	*Dataset 16*	40	100
Dataset 7	10	500	*Dataset 17*	50	100
Dataset 8	10	600	*Dataset 18*	60	100
Dataset 9	10	700	*Dataset 19*	70	100
Dataset 10	10	800	*Dataset 20*	80	100
Dataset 11	10	900	*Dataset 21*	90	100
Dataset 12	10	1000	*Dataset 22*	10	100

TABLE 4. Experiments' parameter set.

Parameters	Value
α	2
β	1
ρ	0.9
τ_0	0. 1
S	30
Z	150

of candidate web services and variable numbers of tasks for each dataset in the range [10–100]. TABLE 3 shows the name of each dataset and the distribution of the tasks and candidates web services among them.

For clarity, we used the same control parameters as in FACO. These values are selected experimentally for the best performance of FACO [16]. There are two methods for parameters selection: first, based on the experiments, and second, based on automatic parameters tuning. In this work, we used the same parameters as in FACO, which were selected based on experiments. TABLE 4 shows the values of common control parameters used in standard ACO, FACO and proposed method such as $\alpha, \beta, \rho, \tau_0$, the population size S, and the iteration number Z.

For both algorithms, we conducted *30* independent repetitions on each dataset. Then, the performance of the algorithm was evaluated in terms of average of best solution quality (*AVG*), standard deviation (StD), and average of running time (*Time*). Solution quality means the quality of the best solution in each execution while running time is the time taken to find the best solution in seconds.

The values of AVG and StD are normalized and shown in percentage. The normalized formula is [50]

$$norm\left(alg_g\right) = \frac{Results(alg_g)}{\sum Results\_all\_algorithms} \times 100\% \quad (15)$$

where g is ACO, FACO, MACO, MFACO, or the proposed algorithm. The formula works as follows: numerator $Results(alg_g)$ contains the results of specific algorithm (i.e. ACO, FACO, MACO, MFACO, or the proposed algorithm) in terms of AVG or StD divided by the denominator $\sum Results\_all\_algorithms$ which contains the sum of all results in terms of AVG or StD getting by all algorithms. The division result is multiplied by 100% to get the percentage of results.

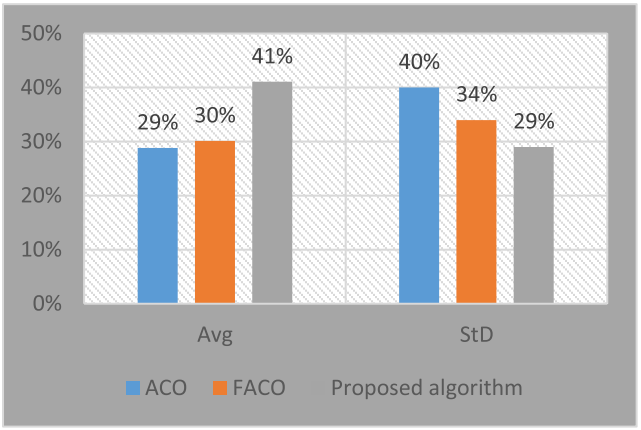

FIGURE 7. Experimental results for the proposed algorithm compared to ACO and FACO in terms of solutions quality and standard deviation on dataset1 and dataset2.

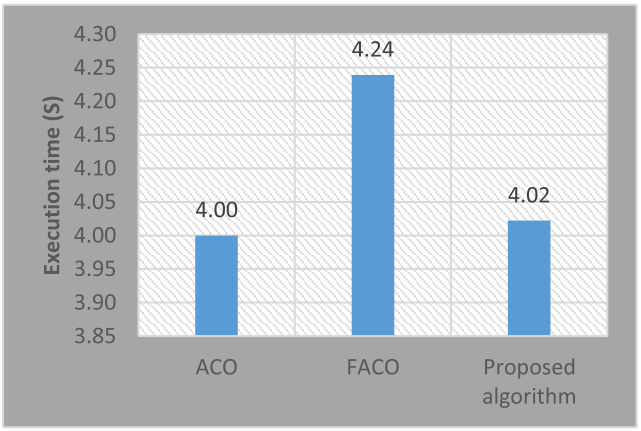

FIGURE 8. Results of the execution time for the proposed algorithm compared to ACO and FACO on dataset1 and dataset2.

Figure 7 presents the results of ACO, FACO and the proposed algorithm on dataset1 and dataset2. The results were normalized to be presented using percentages. The results reveal that the proposed algorithm outperforms ACO in terms of solutions quality by 12%. In addition, the proposed algorithm outperforms FACO in terms of quality of solutions by 11%. The figure also shows that the proposed algorithm has a lower standard deviation than ACO and FACO.

Figure 8 shows the average execution time for dataset1 and dataset2; the results indicate that the proposed algorithm is slightly slower than ACO by 0.02 seconds, but faster than FACO by 0.22 seconds.

The results reveal that the proposed algorithm increases diversity so that it reaches a better solution within a short time compared with ACO and FACO.

The proposed algorithm uses multi-pheromone distribution to support the searching process whereas ACO and FACO use a single pheromone value. Thus, we conducted other experiments where we added a multi-pheromone distribution for both ACO and FACO, yielding MACO and MFACO respectively. The MACO algorithm is the proposed algorithm

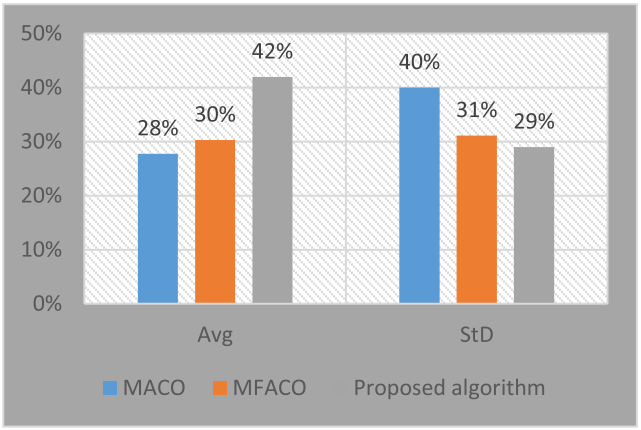

FIGURE 9. Experimental results for proposed algorithm compared to MACO and MFACO in terms of solutions quality and standard deviation on dataset1 and dataset2.

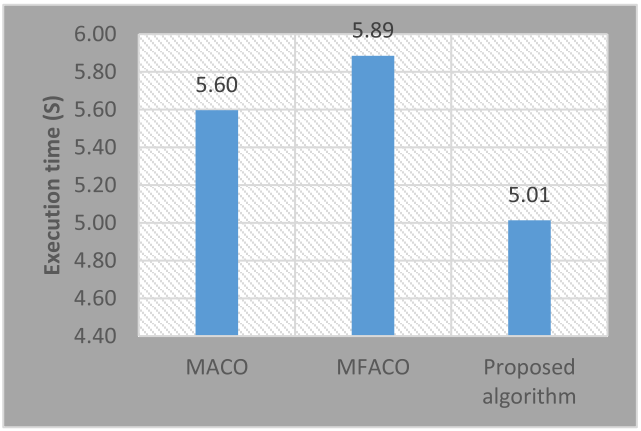

FIGURE 10. Results of execution time for the proposed algorithm compared to MACO and MFACO on dataset1 and dataset2.

with independent pheromone updates and without a swap operation.

Figure 9 shows the results for ACO, FACO with a multi-pheromone distribution and the proposed algorithm on dataset1, dataset2. The results reveal that the proposed algorithm outperforms both MACO and MFACO in terms of solutions quality by 14% and 12%, and in terms of standard deviation by 11% and 2% respectively.

Figure 10 shows the average of execution time for the same experiments in **Figure 9.** The results indicate that the proposed algorithm is finding better results faster than MACO by 0.59 seconds and is faster than MFACO by 0.88 seconds.

The results indicate that the multi-pheromone approach increases the proposed algorithm performance, leading to a better solution and decreases the execution time as well while it decreases the MFACO execution and increases the time for MACO.

For the additional 20 datasets, the proposed algorithm was compared with MACO and MFACO. **Figure 11** and **Figure 12** present the results for the datasets where the number of tasks is fixed in terms of solutions quality and execution time respectively. **Figure 9 Figure 13** and **Figure 14** show the

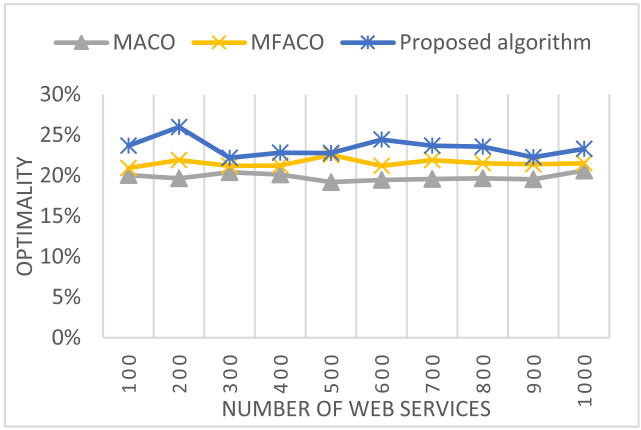

FIGURE 11. The performance of all algorithms when the number of tasks is fixed.

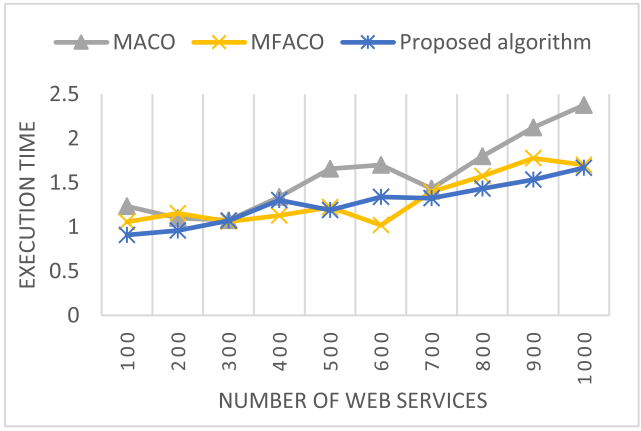

FIGURE 12. The execution time of all algorithms where the number of tasks are fixed.

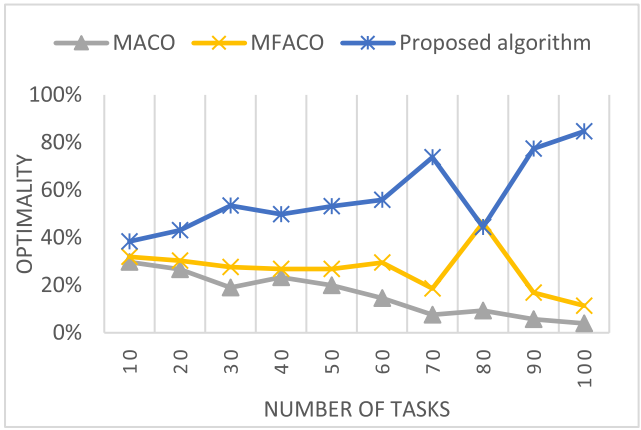

FIGURE 13. The performance of all algorithms where the number of WSs is fixed.

results for the datasets where the number of candidates web services are fixed in terms of solution quality and execution time respectively.

The results in **Figure 11** indicates that the proposed algorithm outperforms the MACO and MFACO algorithms consistently in terms of performance quality.

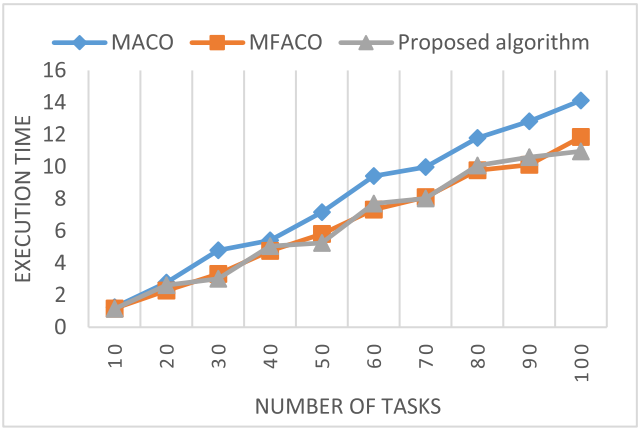

FIGURE 14. The execution time of all algorithms where the number of WSs is fixed.

TABLE 5. Friedman Test results on 22 datasets.

Dataset	*P* value	Dataset	*P* value
Dataset 1	.01087	***Dataset 12***	***<.00001***
Dataset 2	.10323	***Dataset 13***	***.01156***
Dataset 3	***.00066***	***Dataset 14***	***.00003***
Dataset 4	***<.00001***	***Dataset 15***	***.00001***
Dataset 5	.10454	***Dataset 16***	***.00011***
Dataset 6	***.00025***	***Dataset 17***	***<.00001***
Dataset 7	***.00012***	***Dataset 18***	***<.00001***
Dataset 8	***.00089***	***Dataset 19***	***.00003***
Dataset 9	***.00077***	***Dataset 20***	***.00048***
Dataset 10	***.00003***	***Dataset 21***	***<.00001***
Dataset 11	***.00484***	***Dataset 22***	***.00012***

The results in **Figure 12** shows that the proposed algorithm and MFACO are quite similar and both better than MACO. This can be credited to the multi-pheromone distribution, which enhances the MFACO to reduce execution time.

The results in **Figure 13** shows that the proposed algorithm outperforms MACO and MFACO consistently in terms of performance quality as well except when the number of tasks equals 80, in which case the MFACO is better.

The results in **Figure 14** shows that the proposed algorithm and MFACO are close in terms of execution time and both are better than MACO. These experiments also show that the multi-pheromone approach for ACO and FACO is more effective than using a single pheromone value.

The experiments all prove the applicability, robustness, and stability of the new enhancement on various types of datasets.

The Friedman test was used to test whether the results were statistically significant for the 30 independent runs in each one of the 22 datasets between the proposed algorithm and MACO and MFACO. TABLE 5 shows the Friedman test results, which indicate that the proposed algorithm results are statistically significant for the 30 independent runs in 19 datasets out of 22 (Dataset 3, Dataset 4, Dataset 6, Dataset 7, Dataset 8, Dataset 9, Dataset 10, Dataset 11, Dataset 12, Dataset 13, Dataset 14, Dataset 15, Dataset 16, Dataset 17, Dataset 18, Dataset19, Dataset 20, Dataset 21, Dataset 22).

VII. CONCLUSION

In this paper, we propose a new ant-inspired algorithm to improve the performance of ants based on the swap concept for a QoS-aware WSS problem. The stack in local minima is the main problem of a basic ACO. The proposed enhancement aims to leverage the best solutions found in each iteration under the assumption that each one of these solutions has a valuable part. We believe that merging these best solutions helps the ACO algorithm to find a better solution and avoid the stagnation problem.

Several experiments were conducted, demonstrating that the proposed algorithm outperforms standard ACO and FACO in terms of average solution quality and execution time on 22 different datasets.

REFERENCES

[1] T. Erl, *SOA Principles of Service Design*, vol. 1. Upper Saddle River, NJ, USA: Prentice-Hall, 2008.
[2] M. Papazoglou, *Web Services: Principles and Technology*. London, U.K.: Pearson, 2008.
[3] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Services: Concepts, Architectures and Applications*. Berlin, Germany: Springer, 2004.
[4] G. Kang, J. Liu, M. Tang, and Y. Xu, "An effective dynamic Web service selection strategy with global optimal QoS based on particle swarm optimization algorithm," in *Proc. IEEE 26th Int. Parallel Distrib. Process. Symp. Workshops PhD Forum*, May 2012, pp. 2280–2285.
[5] P. Rodriguez-Mier, M. Mucientes, J. C. Vidal, and M. Lama, "An optimal and complete algorithm for automatic Web service composition," *Int. J. Web Services Res.*, vol. 9, no. 2, pp. 1–20, 2012.
[6] S. K. Gavvala, C. Jatoth, G. R. Gangadharan, and R. Buyya, "QoS-aware cloud service composition using eagle strategy," *Future Gener. Comput. Syst.*, vol. 90, pp. 273–290, Jan. 2019.
[7] A. Visvizi, M. D. Lytras, E. Damiani, and H. Mathkour, "Policy making for smart cities: Innovation and social inclusive economic growth for sustainability," *J. Sci. Technol. Policy Manage.*, vol. 9, no. 2, pp. 126–133, 2018.
[8] A. Visvizi and M. D. Lytras, "Rescaling and refocusing smart cities research: From mega cities to smart villages," *J. Sci. Technol. Policy Manage.*, vol. 9, no. 2, pp. 134–145, 2018.
[9] M. D. Lytras and A. Visvizi, "Who uses smart city services and what to make of it: Toward interdisciplinary smart cities research," *Sustainability*, vol. 10, no. 6, p. 1998, 2018.
[10] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.
[11] M. Kong, P. Tian, and Y. Kao, "A new ant colony optimization algorithm for the multidimensional knapsack problem," *Comput. Oper. Res.*, vol. 35, no. 8, pp. 2672–2683, 2008.
[12] V. Viswanathan and I. Krishnamurthi, "Finding relevant semantic association paths through user-specific intermediate entities," *Hum.-Centric Comput. Inf. Sci.*, vol. 2, p. 9, Mar. 2012.
[13] D. A. Menascé, "Composing Web services: A QoS view," *IEEE Trans. Internet Comput.*, vol. 8, no. 6, pp. 88–90, Nov./Dec. 2004.
[14] 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.
[15] 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, p. 6, May 2007.
[16] 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.
[17] 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.
[18] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for Web services composition," *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, May 2004.

[19] 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.*, Springfield, IL, USA, vol. 5896, Dec. 2009, pp. 123–142.
[20] S. Liu, Y. Liu, N. Jing, G. Tang, and Y. Tang, "A dynamic Web service selection strategy with QoS global optimization based on multi-objective genetic algorithm," in *Proc. Int. Conf. Grid Cooperat. Comput.*, 2005, pp. 84–89.
[21] Y.-M. Xia, J.-L. Chen, and X.-W. Meng, "On the dynamic ant colony algorithm optimization based on multi-pheromones," in *Proc. 7th IEEE/ACIS Int. Conf. Comput. Inf. Sci. (ICIS)*, May 2008, pp. 630–635.
[22] F. Qiqing, P. Xiaoming, L. Qinghua, and H. Yahui, "A global QoS optimizing Web services selection algorithm based on MOACO for dynamic Web service composition," in *Proc. Int. Forum Inf. Technol. Appl. (IFITA)*, May 2009, pp. 37–42.
[23] W. Li and H. Yan-Xiang, "A Web service composition algorithm based on global QoS optimizing with MOCACO," in *Algorithms and Architectures for Parallel Processing*. Berlin, Germany: Springer, 2010, pp. 218–224.
[24] R. Wang, L. Ma, and Y. Chen, "The research of Web service selection based on the ant colony algorithm," in *Proc. Int. Conf. Artif. Intell. Comput. Intell. (AICI)*, Oct. 2010, pp. 551–555.
[25] R. Wang, L. Ma, and Y. Chen, "The application of ant colony algorithm in Web service selection," in *Proc. Int. Conf. Comput. Intell. Softw. Eng.*, Dec. 2010, pp. 1–4.
[26] Z. Shanshan, W. Lei, M. Lin, and W. Zepeng, "An improved ant colony optimization algorithm for QoS-aware dynamic Web service composition," in *Proc. Int. Conf. Ind. Control Electron. Eng. (ICICEE)*, Aug. 2012, pp. 1998–2001.
[27] W. Zhang, C. K. Chang, T. Feng, and H.-Y. Jiang, "QoS-based dynamic Web service composition with ant colony optimization," in *Proc. COMPSAC*, Jul. 2010, pp. 493–502.
[28] X. Zheng, J.-Z. Luo, and A.-B. Song, "Ant colony system based algorithm for QoS-aware Web service selection," in *Proc. GSEM*, 2007, pp. 39–50.
[29] L. Qi, W. Yao, and J. Chang, "A large scale transactional service selection approach based on skyline and ant colony optimization algorithm," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Apr. 2018, pp. 1–7.
[30] Z. Yang, C. Shang, Q. Liu, and C. Zhao, "A dynamic Web services composition algorithm based on the combination of ant colony algorithm and genetic algorithm," *J. Comput. Inf. Syst.*, vol. 6, no. 8, pp. 2617–2622, 2010.
[31] Y. Yang, B. Yang, S. Wang, F. Liu, Y. Wang, and X. Shu, "A dynamic ant-colony genetic algorithm for cloud service composition optimization," *Int. J. Adv. Manuf. Technol.*, vol. 102, nos. 1–4, pp. 355–368, 2019.
[32] S. R. Dhore and M. U. Kharat, "QoS based Web services composition using ant colony optimization: Mobile agent approach," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 1, no. 7, pp. 519–527, 2012.
[33] D. Wang, H. Huang, and C. Xie, "A novel adaptive Web service selection algorithm based on ant colony optimization for dynamic Web service composition," in *Algorithms and Architectures for Parallel Processing*. Cham, Switzerland: Springer, 2014, pp. 391–399.
[34] C. Zhang, H. Yin, and B. Zhang, "A novel ant colony optimization algorithm for large scale QoS-based service selection problem," *Discrete Dyn. Nature Soc.*, vol. 2013, Jun. 2013, Art. no. 815193.
[35] D.-N. Le and G. N. Nguyen, "A new ant-based approach for optimal service selection with E2E QoS constraints," in *Intelligence in the Era of Big Data*. Berlin, Germany: Springer, 2015, pp. 98–109.
[36] J. Shen and S. Yuan, "QoS-aware peer services selection using ant colony optimisation," in *Proc. Bus. Inf. Syst. Workshops (BIS)* (Lecture Notes in Business Information Processing), vol. 37, W. Abramowicz and D. Flejter, Eds. Berlin, Germany: Springer, 2009, pp. 362–374.
[37] T. Ghafarian and M. Kahani, "Semantic Web service composition based on ant colony optimization method," *J. Math.*, to be published.
[38] V. R. Chifu, C. B. Pop, I. Salomie, M. Dinsoreanu, T. David, and V. Acretoaie, "Ant-based methods for semantic Web service discovery and composition," *Ubiquitous Comput. Commun. J.*, vol. 6, no. 1, pp. 631–641, 2011.
[39] V. R. Chifu, C. B. Pop, I. Salomie, M. Dinsoreanu, V. Acretoaie, and T. David, "An ant-inspired approach for semantic Web service clustering," in *Proc. 9th RoEduNet IEEE Int. Conf.*, Jun. 2010, pp. 145–150.
[40] C. B. Pop, V. R. Chifu, I. Salomie, M. Dinsoreanu, T. David, and V. Acretoaie, "Ant-inspired technique for automatic Web service composition and selection," in *Proc. SYNASC*, Sep. 2010, pp. 449–455.
[41] C. B. Pop, V. R. Chifu, I. Salomie, M. Dinsoreanu, T. David, and V. Acretoaie, "Ant-inspired framework for automatic Web service composition," *Scalable Comput., Pract. Exper.*, vol. 12, no. 1, pp. 137–152, 2011.
[42] K. Yan, G. Xue, and S.-W. Yao, "An optimization ant colony algorithm for composition of semantic Web services," in *Proc. Asia–Pacific Conf. Comput. Intell. Ind. Appl. (PACIIA)*, Nov. 2009, pp. 262–265.
[43] Y. Xia, C. Liu, Z. Yang, and J. Xiu, "The ant colony optimization algorithm for Web services composition on preference ontology," in *Proc. Int. Conf. Adv. Intell. Awareness Internet (AIAI)*, Oct. 2011, pp. 193–198.
[44] T. Stützle and H. H. Hoos, "MAX-MIN ant system," *Future Generat. Comput. Syst.*, vol. 16, no. 8, pp. 889–914, Jun. 2000.
[45] 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.
[46] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *Biosystems*, vol. 43, no. 2, pp. 73–81, 1997.
[47] L. M. Gambardella and M. Dorigo, "Solving symmetric and asymmetric TSPs by ant colonies," in *Proc. IEEE Int. Conf. Evol. Comput.*, May 1996, pp. 622–627.
[48] O. Deepa and A. Senthilkumar, "Swarm intelligence from natural to artificial systems: Ant colony optimization," in *Proc. Netw. (GRAPH-HOC)*, vol. 8, 2016, pp. 9–17.
[49] A. Aljanaby, "An experimental study of the search stagnation in ants algorithms," *Int. J. Comput. Appl.*, vol. 148, 2016, pp. 1–4.
[50] 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.

HASHEM ALAYED received the Ph.D. degree in computer science from the Department of Computer Science, University of Southern California. He is currently a Faculty Member with the Department of Computer Science, College of Computer Science and Information, King Saud University. His research interests include artificial intelligence, machine learning applications, and behavioral analysis.

FADL DAHAN received the B.Sc. degree from Thamar University, Yemen, and the M.Sc. degree from King Saud University, and the Ph.D. degree from the Department of Computer Science, King Saud University. He is currently an Assistant Professor with the Department of Information System, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Al-Kharj, Saudi Arabia. He is also a Faculty Member with the Department of Computer Science, Faculty of Computer Science in Torba, University of Taiz, Yemen. His research interests include optimization and swarm intelligence.

TAHA ALFAKIH received the M.S degree from the College of Computer and Information Sciences, King Saud University (KSU), where he is currently pursuing the Ph.D. degree. His research interest focuses on mobile edge computing.

HASSAN MATHKOUR received the Ph.D. degree from the University of Iowa, USA. He held several administration posts including the Dean, associate dean, department chair, a director of the research center, and the head of a joint Ph.D. program. He is currently a Professor with the Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia. He also serves as an IT consultant. He has over 100 research articles in journals and conferences. 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, 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.