IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 17, 2021, accepted February 18, 2021, date of publication February 24, 2021, date of current version March 4, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3061738

An Enhanced Ant Colony Optimization Based Algorithm to Solve QoS-Aware Web Service Composition

FADL DAHAN[1,2], **KHALIL EL HINDI**[3], **AHMED GHONEIM**[4], **(Member, IEEE), AND HUSSAIN ALSALMAN**[3]

[1]Department of Information System, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia
[2]Department of Computer Sciences, Faculty of Computing and Information Technology Alturbah, Taiz University, Taiz 9674, Yemen
[3]Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia
[4]Department of Software Engineering, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia

Corresponding author: Khalil El Hindi (khindi@ksu.edu.sa)

This work was supported by the Deanship of Scientific Research at King Saud University under Grant RG-1439-035.

ABSTRACT Web Service Composition (WSC) can be defined as the problem of consolidating the services regarding the complex user requirements. These requirements can be represented as a workflow. This workflow consists of a set of abstract task sequence where each sub-task represents a definition of some user requirements. In this work, we propose a more efficient neighboring selection process and multi-pheromone distribution method named Enhanced Flying Ant Colony Optimization (EFACO) to solve this problem. The WSC problem has a challenging issue, where the optimization algorithms search the best combination of web services to achieve the functionality of the workflow's tasks. We aim to improve the computation complexity of the Flying Ant Colony Optimization (FACO) algorithm by introducing three different enhancements. We analyze the performance of EFACO against six of existing algorithms and present a summary of our conclusions.

INDEX TERMS Service-oriented computing (SOC), nature-inspired algorithms, discrete optimization, meta-heuristic algorithms, ant colony optimization (ACO), enhanced flying ant colony optimization (EFACO).

I. INTRODUCTION

Service-Oriented Computing (SOC) introduced a new style for distributed applications, where the software components act as services that communicate through the Internet. The problem of Web Service Composition (WSC) integrates the services to provide more sophisticated functionality. In the case of SOC, clients specify their requirements and the WSC searches for services, and then provides clients with best services available to achieve their requirements.

The rapid growth of providers leads to the availability of a large number of WSs that can do the same tasks. This also made the WSC problem more difficult to achieve in polynomial time. The selection process is responsible for selecting the best WS for each task, which generates a good combination of web services to fulfill the workflow. The user will receive a concert executable plan that contains the tasks in the workflow and WSs that satisfy them.

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

The selection process needs to take into account a large number of candidate WSs for each task because all these services perform similar functionality. The QoS constraints represent the non-functional features associated with these WSs [1]. The process of selecting the most suitable atomic WS to the QoS constraints is called QoS-aware WSC problem. QoS constraints may be related to some or all of the following response time, cost, availability, reliability,…etc.

WSC is very challenging because (1) It is very large in scale where the WSs are retrieved from the registry. The challenge is how to choose good combination of best WSs that meet the QoS constraints for each task. (2) It has too many optimization issues. This include QoS criteria and transaction criteria.…etc To select the best WS the algorithm must consider the value of each QoS of each solution taken into account the precedence dependency between different tasks. (3) The QoS attributes have many conflicting objectives that must be handled. For example, the WS that has minimum response time is probably more expensive which increases cost (minimization objective) and also will decrease the availability (maximization objective) of this WS and also it will be less reliable (maximization objective).

WSC aims to satisfy user requirements by creating a workflow that contains abstract representations (tasks) for these requirements. The workflow then searches the web services repositories for Web Services (WSs) that execute these representations [1]. The number of retrieved WSs may be significantly high because of the large number of providers of WSs with similar functionalities, making the selection of the best combination a challenging problem [2], [3]. To evaluate the quality of this combination, different criteria were used such as Quality of Service (QoS) properties such as Cost (C), Response Time (RT), Availability (A), and Reliability (R), etc. [4]. These properties represent the non-functional criteria attached to WSs. QoS criteria are divided into two categories, deterministic and nondeterministic [5]. For deterministic criteria, values such as cost or security must be known in advance before using the WSs. For nondeterministic criteria uncertain values associated with using WSs must be identified first e.g., response time, availability, and reliability. Therefore, providers must test WSs to measure the values of these criteria in advance. Another issue is that the target value of each criterion may differ depending on user needs. For example, users may require cost and response time values to be minimized, whilst simultaneously expecting availability and reliability values to be maximized.

The best solution represents a complete combination of WSs for different tasks in a workflow. The number of available combinations (solutions) for this problem is exponential which makes the challenge of finding the best solution an NP-hard problem [6].

In [7], a Flying Ant Colony Optimization (FACO algorithm has been proposed, based on Ant Colony Optimization (ACO) [8]. However, it incorporates imaginary flying ants which inject pheromones from a distance. In this case, the best ants share the amount of pheromones with their neighboring nodes. Although FACO enhances ACO in terms of solution quality, but it increases the execution time [7] because it must determine neighboring nodes.

Multi-objective optimization problems are difficult to solve directly because of the discrepancies in the objectives and the lack of efficient tools for solving such problems [2]. There are two main approaches for handling multi-objective optimization problems: preference and Pareto optimality. The preference-based approach (also called the scalarization) aims to transform a multi-objective optimization problem into a single-objective optimization problem. The new single-objective problem is a composition function that contains the weight assigned to each objective based on a preference vector [3]. With Pareto optimality approach, the objective vectors are compared to find an optimal objective vector. Each vector contains the values as solutions for the problem objectives. This approach works based on the dominance relation between the solutions [3]. Most of the recent literature that introduced to solve the WSC problem using the preference-based approach where they deal with the multi-objective of the QoS criteria by converting these criteria into a single objective criterion [1], [2].

The literature shows that the representation of the WSC can be using two main models which are workflow based model (WM) [5], [6] and input-output dependency-based model [5], more details about the features of each model can be found in [8].

In this paper, we address the problem of WSC using preference-based approach and the WM for the problem representation. In addition, we introduce efficient solutions for the FACO problem that aim to reduce the execution time and enhance the solution quality by:

(1) Restricting the flying process so that it occurs only if there is an improvement on solutions to decrease the execution time while neighboring selection.

(2) Introducing an efficient neighboring selection process to effectively balance between exploration and exploitation mechanisms.

(3) Applying a more efficient multi-pheromone distribution method where the multi-pheromone distribution allows ants to follow the weight of the QoS attributes to construct solutions.

The proposed algorithm is named Enhanced Flying Ant Colony Optimization (EFACO). We conducted multiple experiments to evaluate the effectiveness of the proposed algorithm. The experiments show that the restrictive flying ant process with the new neighboring selection process are decreased the time required to find better solutions, while the multi-pheromone improves solution quality.

This paper is organized as follows. Section 2 reviews the problem and related work involving ACO. Section 3 then describes the proposed algorithm. In section 4, we present an analysis of experimental results. Finally, we conclude our research in section 5.

II. PROBLEM DEFINITION AND RELATED WORK

Typically, a workflow contains many tasks that represent user requirements and each task is an abstract representation of a user requirements. Let n equal the number of tasks and m equal the number of WSs that can be used to achieve a given task. In fact, the value of m varies but for simplicity in this study the variable is fixed for each dataset i.e., in the experiment each dataset contains a special workflow with a fixed value for m. ***Figure 1*** shows a hypothetical workflow of tasks with corresponding WSs [7], [9], [10]. The challenge of finding the best combination of WSs is a combinatorial optimization problem. A solution represents a complete path that contains the best combination of WSs for the tasks in the workflow. The available combinations (potential solutions) for this problem are m^n, thus, the problem is an NP-hard problem [6].

WSs are selected based on their QoS attributes. These attributes may include cost, response time, availability, and reliability, etc. [4]. In this study, we consider four QoS attributes which are the non-functional constraints for the WSs [4]. These four constraints are the cost that represents

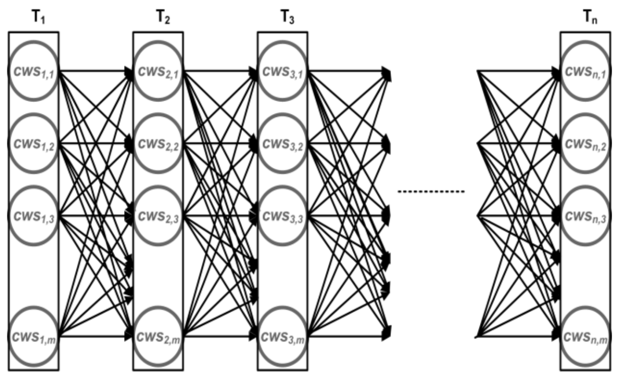

FIGURE 1. Workflow simulation [7].

the fees of the services utilization, response time which represents the delay of using the services, availability which represents the invocation rate, and reliability which represents the message error rate. Therefore, the values of these QoS attributes are calculated to evaluate the fitness of a potential solution. Table 1 lists the formulas used to calculate these attribute values [11]. In Table 1, n denotes the task number and j represent the web service selected for task i. The selection process adds an additional challenge for the problem that can be represented as a multi-objective combinatorial optimization problem.

TABLE 1. The formulas for calculating the QoS attributes.

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)	$\prod_{i=1}^{n} R(cws_{ij})$

The workflow can be divided based on the representation patterns of the user requirements into complex and simplex workflows. Representation patterns show if the requirements can be executed concurrently or must be executed sequentially. Loops indicate that some WSs must be executed several times [11], [12]. ***Figure 2*** shows each of the four workflow patterns. A complex workflow contains all four representation patterns, while a simplex contains only the sequential pattern. ***Figure 3***(a) shows a complex workflow, while ***Figure 3***(b) shows a decomposition of the complex workflow into simplex workflows [11], [12]. These simplex workflows are perceived as individual problems; therefore, their decomposition has been neglected. Potential solutions are then compared based on the fitness values of the simplex workflows.

A. ANT COLONY OPTIMIZATION (ACO)

ACO is a meta-heuristic algorithm that inspires the behavior of the ants while searching for food. It aims to find the shortest possible path to a food source within the least amount of time using a swarm of ants that share information. In fact, real ants forage randomly and while moving they deposit a chemical with an evaporation feature called a pheromone. This pheromone helps other ants to track and follow each other. The best ants that find better food resources will return to the nest. In this case the amount of pheromones will be stronger for the best ants than for others, and they can recruit more ants to follow them. While searching for food, this mechanism is a tradeoff between two techniques called exploration and exploitation. Exploration represents the random search for food resources while exploitation represents information sharing [8], [13], [14].

As shown in ***Figure 4***, [15], [16] the ACO algorithm simulates the behavior of real ants. At the beginning, all ants have a consistent amount of pheromones and will search randomly for better solutions and return to their nest. At this stage, the ants have knowledge of the search space and can share information with other ants. Next, the ACO uses a tradeoff formula for ants to construct new solutions based on the distance τ representing exploration, and last knowledge η (pheromone) represents exploitation. To achieve this, the ants use the following transition formula:

$$P_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}\,[\eta_{ij}]^{\beta}}{\sum_{l\in\mathcal{N}_i^k}[\tau_{il}]^{\alpha}\,[\eta_{il}]^{\beta}}\, if j \in \mathcal{N}_i^k \tag{1}$$

where α and β represent the coefficient parameters that determine the importance of local distance and pheromone. $\mathcal{N}_i^k$ denotes the unvisited food sources for ant k. The pheromone deposition is also simulated. The ants deposit pheromones while moving to invite other ants to follow them, and this pheromone evaporates. The process of deposition and evaporation is called the local pheromone update, and is represented by Eq. (2):

$$\tau_{\mathrm{ij}} = (1-\rho)\tau_{\mathrm{ij}} + \rho\tau_0 \tag{2}$$

where τ_0 represents the pheromone value at the initialization stage and ρ denotes the evaporation rate.

During the last stage, the ACO algorithm compares the solution of all ants and memorizes the best ant solution. This ant then deposits pheromone on its path (global update) to attract other ants in future iterations as represented below.

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

In which:

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

where L^{gb} represents the length of the best solution found so far.

The ACO algorithm has many applications. However, the ants accumulate pheromones as more paths are visited

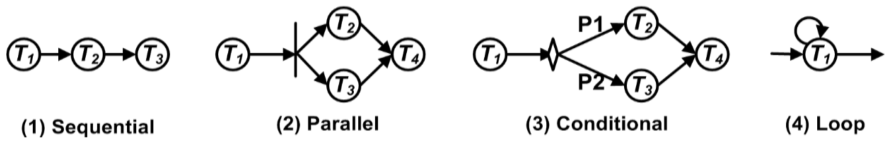

FIGURE 2. The available workflow patterns [7].

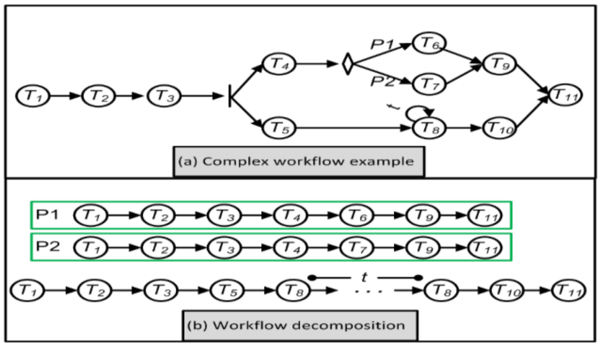

FIGURE 3. Complex workflow composition/decomposition example.

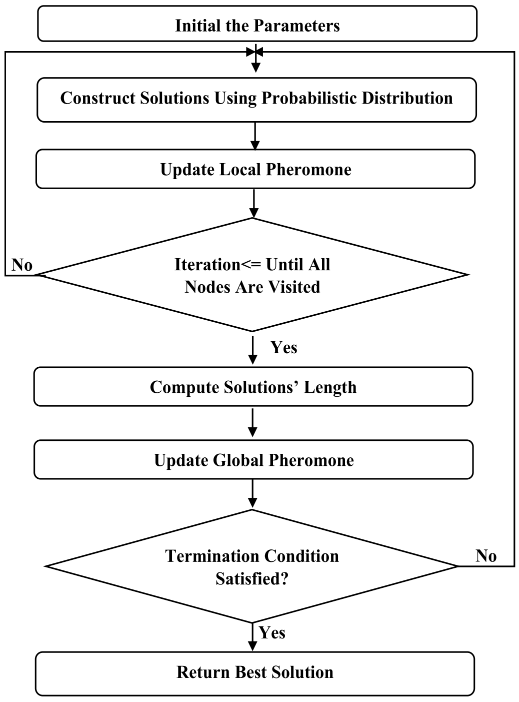
Initial the Parameters
Construct Solutions Using Probabilistic Distribution
Update Local Pheromone
Iteration<= Until All Nodes Are Visited
No
Yes
Compute Solutions' Length
Update Global Pheromone
Termination Condition Satisfied?
No
Yes
Return Best Solution

FIGURE 4. The ACO flowchart.

while searching for solutions, and this causes the stagnation problem [17]. Therefore, the ACO can become stuck in local minima.

The FACO algorithm is proposed as a balancing mechanism to avoid the stagnation problem, however, this mechanism is time consuming where the ants in each iteration have to search each service' neighboring in the solution path. The flying process is repeated for each ant in each iteration and for each service in the solution path which is time-consuming. Therefore, the process time complicity is *O(Ants*$*$*Max*$_{iteration}$$*$*n*$*$*m)* where *Ants* represents the population number, *Max*$_{iteration}$ represents the iterations number, *n* represents the number of tasks, and *m* represents the number of WSs that can be used to achieve a given task. In this study, we continue improving FACO to overcome the time problem and to ensure that the solution remains based upon the restriction, neighboring selection, and multi-pheromone mechanisms.

B. FLYING ANT COLONY OPTIMIZATION (FACO)

FACO [7] modifies the ACO algorithm to better balance exploration and exploitation operations based on neighborhood relations. This creates a tradeoff between exploration and exploitation that is necessary to find a better food source and to avoid the stagnation problem. Also, the FACO algorithm was generalized to solve the traveling salesperson problem [18].

The ants use transition formula (Eq. (5) to seek superior service combination of WSC problem, where they select the next WSs non-deterministically. The equation calculates the probability of moving to task *t+1* and web service s' from task *t*. Then, it uses the roulette wheel selection algorithm to select one web service based on the probability of each.

The roulette-wheel is a widely adopted algorithm used alongside meta-heuristic algorithms to select solutions based on their fitness probability [19]. This algorithm assumes that the probability of each solution is proportional to its fitness for that solution.

$$P^{k}_{(t,s)(t+1,s')} = \frac{\left[\tau_{(t,s)(t+1,s')}\right]^{\alpha}\left[\eta_{(t,s)(t+1,s')}\right]^{\beta}}{\sum_{l\in\mathcal{N}^{k}_{t}}\left[\tau_{(t,s)(t+1,l)}\right]^{\alpha}\left[\eta_{(t,s)(t+1,l)}\right]^{\beta}} \quad \text{if } x' \in \mathcal{N}^{k}_{t+1} \tag{5}$$

In Eq. (5). α and β represent the parameters that determine the weight of the local heuristic η and pheromone τ. $\mathcal{N}^{k}_{t+1}$ denotes the list of unvisited WSs from task *t+1* for ant *k*.

The ants deposit pheromones on their path while moving to invite other ants to follow these paths in future iterations using the following local pheromone update formula:

$$\tau_{(t,s)(t+1,s')} = (1-\rho)\,\tau_{(t,s)(t+1,s')} + \rho\tau_0, \tag{6}$$

where $\tau_{(t,s)(t+1,s')}$ represents the edge's pheromone between web service *s* (task *t*) and web service s' (task *t+1*). τ_0 denotes the initialization value of the pheromone, while ρ represents the evaporation rate.

Finally, the ant that finds the best solution deposits pheromones along its path to increase the chances that the

WSs will also determine the solution to be selected by other ants. This is achieved using the following global pheromone update formulas:

$$\tau_{(t,s)(t+1,s')} = (1-\rho)\tau_{(t,s)(t+1,s')} + \rho\Delta\tau_{(t,s)(t+1,s')} \quad (7)$$

in which the quantity of pheromone laid on path $\Delta\tau_{(t,s)(t+1,s')}$ is:

$$\Delta\tau_{(t,s)(t+1,s')} = \begin{cases} {}^{1}/_{L^{gb}} & \text{if arc (t, s) } \left(t+1, s'\right) \in \text{the best tour} \\ 0 & \text{otherwise,} \end{cases} \quad (8)$$

where L^{gb} represents the length of the global best solutions. After generating the global pheromone update, the best ant behaves as a flying ant and injects pheromones on neighboring nodes. It determines the neighbors of each web service and increases their pheromone values where the number of increments each neighbor gets, is inversely proportional to the distance between itself and the actual node (WS), in determining the best solution. The distance between WSs is calculated using Eq. (9)

$$d_{(s,y)} = \sqrt{(C_s-C_y)^2+(RT_s-RT_y)^2+(A_s-A_y)^2+(R_s-R_y)^2}, \quad (9)$$

where C, RT, A, and R represent QoS attributes. s is the web service from best path in task $t+1$. y is a web service for the task $t+1$, and $s \neq y$.

A flying ant selects Ns nearest WSs to increase their pheromone values, where the number of increments is inversely proportional to the distance. Eq. (10) shows the formula for modifying the pheromone value of a neighboring web services.

$$\tau_{(t+1,s')(t+1,l)} = \tau_{(t+1,s')(t,l)} + \left(\tau_{(t+1,s)(t,s')} / \left(1+\sqrt{d'\left(\eta_{(t+1,s')(t+1,l)}\right)}\right)\right), \quad (10)$$

where $\tau_{(t,s)(t+1,s')}$ represents the pheromone trails from the global pheromone update. $d'\left(\eta_{(t,s')(t,l)}\right)$ denotes the normalization for distance between a web service in task $t+1$ and its neighboring web service l in the same task, and $l \in N_s$. $d'\left(\eta_{(t+1,s')(t+1,l)}\right)$ is calculated using the following equation:

$$d'\left(\eta_{(t+1,s')(t+1,l)}\right) = \frac{\eta_{(t+1,s')(t+1,l)}}{\sum_{q=1}^{N_s} \eta_{(t+1,s')(t+1,q)}} \quad (11)$$

Recall that the FACO algorithm is proposed to enhance the performance of the ACO algorithm and avoid the stagnation problem using the flying process. However, the flying process is time consuming. In this work, we propose three different enhancements to avoid the FACO time problem.

C. RELATED ACO-BASED METHODS

The QoS-aware WSC problem has gained much attention in recent years, thus, many studies have investigated solving it using different ACO-based methods.

The ACO algorithm is used for WSC to find the best possible solution that satisfies user requirements. This solution is created using ACO by moving through each task in a workflow and selecting one web service with the best QoS attributes for the solution.

Xia *et al.* [20] added multi-pheromones to the standard ACO to deal with the problem. This algorithm searches for near optimal solutions on a composition graph. Qiqing *et al.* [21] introduced a Multi-Objective Ant Colony Optimization (MOACO). MOACO deals with the problem as a multi-objective optimization problem, based on the modeling of QoS attributes. Li *et al.* [22] introduced a Multi-Objective Chaos Ant Colony Optimization (MOCACO), which used a chaos operator to enhance the convergence speed of MOACO. Wang *et al.* [23], [24] analyzed the effect of using different parameters on the performance of the ACO algorithm. Shanshan *et al.* [25] introduced an enhancement for ACO. The new enhancement aims to get a Pareto optimality based on avoidance of the stagnation problem. It includes a random variable added to the ACO algorithm. Zhang *et al.* [26] introduced a workflow decomposition method and used a multi-objective ACO for searching for solutions. Zheng *et al.* [27] introduced a graph based method to represent the problem workflow. Then, they used ACO to find the solution path in parallel execution. Alayed *et al.* [16] introduced an enhancement for ACO based on the swapping method. The swapping method aimed to maintain a good balancing between exploration and exploitation. An inverted ACO is proposed in [28] where the repulsion effect is introduced as instead of the regular pheromone effect. The WSC has been used to facilitate the manufacturing service composition problem. A grey wolf (GW) and genetic algorithms are proposed in [29] where the genetic strategies is used to help GW to avoid the stagnation problem. The GW also used in [30] where three different enhancements were introduced.

Yang *et al.* [31] introduced a hybrid of a genetic algorithm and ACO. Dhore and Kharat [32] used mobile agents to enhance the performance of ACO. Wang *et al.* [33] introduced an enhancement for ACO called Adaptive Ant Colony Optimization (AACO). This new algorithm selects the WSs for a workflow based on the degree of trust and QoS parameters. Zhang *et al.* [34] introduced a new algorithm called Clustering Ant Colony Selection (CACS). This algorithm uses a skyline query to decrease the candidates' services for each task and CACS to search for solutions efficiently. Le *et al.* [35] introduced a new algorithm based on the Max-Min Ant System (MMAS) and heuristic information to search for an efficient solution. Shen and Yuan [36] introduced an ACO algorithm that searches for a solution based on peers. These peers collect QoS attribute information. Other researchers [34]–[40] have proposed algorithms to search for solutions efficiently. An adaptive genotype is proposed by Jatoth *et al.* [44] where it evaluated the solution quality using discrete uniform service rank distribution. Chattopadhyay and Banerjee [45] proposed a search space pruning method using clustering. The biogeography-based optimization is introduced in [46]. Asghari *et al.* [47] introduced a hybrid shuffled frog leaping and genetic algorithm. Wakrime *et al.* [48] proposed a workflow composition method using the minimally unsatisfiable algorithm and proposed genetic based algorithm for searching better solutions.

The aim of the majority of the researches mentioned in the literature is to improve ACO's exploitation behavior [49], with the exception of the FACO algorithm. FACO aims to introduce an enhancement for ACO's exploration and exploitation behaviors. In ACO, the aim of the exploration behavior is to help ants to find local solutions while the aim of exploitation is to help the ants to determine the global optima [50] This research also aims to enhance the mechanism of exploration and exploitation behaviors of ACO. This enhancement improves ACO by avoiding the premature stagnation problem whilst simultaneously reducing the execution time of FACO.

III. ENHANCED FLYING ANT COLONY OPTIMIZATION (EFACO)

FACO is primarily proposed for solving the QoS-aware WSC problem [7]. This algorithm outperforms the standard ACO in terms of the quality of the solution. However, it is slower than ACO due to the flying process. The flying process in FACO requires searching for neighboring WSs for each web service in determining the best solution which is time consuming [7].

In this study, we propose EFACO (***Figure 5*** shows the EFACO flowchart) to overcome the execution time problem and to enhance the quality of solution. EFACO is different from FACO in different ways. First, the flying ant process is restricted, such that it occurs only if there is an improvement in the solution obtained thus far. This improves the execution time but may reduce the quality of solution. Second, a new neighboring selection method is proposed, such that the proposed method only selects randomly S_N neighboring WSs to inject pheromone on them and increase their chance to be selected in future iterations and adaptively select one web service from these neighboring WSs to replace the web services in the best path. This also improves the execution time but may reduce the quality of solution. Third, to compensate for this loss, we use a multi-pheromone method where each QoS attribute has its own pheromone value. This creates an opportunity for each ant to visit more solutions (i.e., the exploration process is increased) [16]. The following subsections explain these differences in details.

A. RESTRICTION ON FLYING ANTS

The flying process in the FACO is repeated continuously in each iteration to search nearest neighbors. This process is time consuming, so in this work we introduced an adaptive flag that manages the execution of the flying process. This flag tracks the solutions' enhancement and operates the flying process if the new solution is better than all pervious solution found so far.

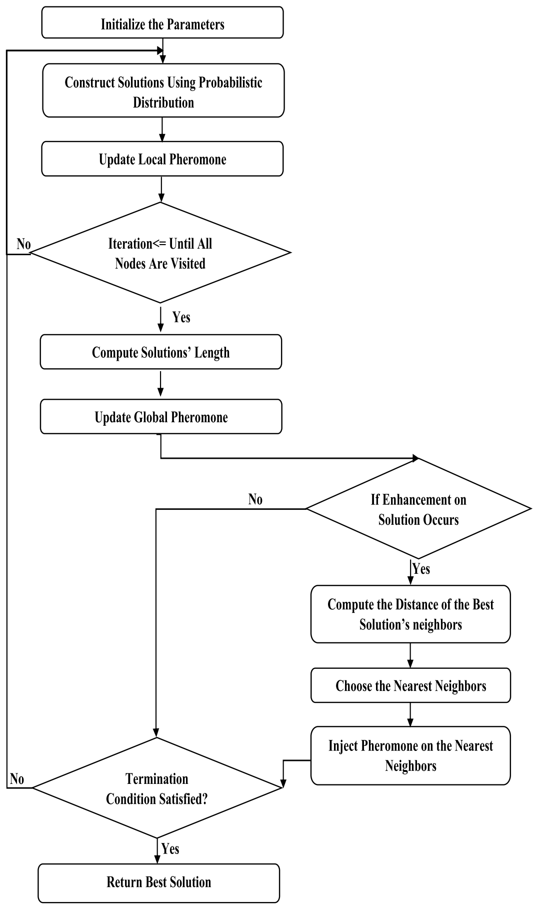
Initialize the Parameters
Construct Solutions Using Probabilistic Distribution
Update Local Pheromone
Iteration<= Until All Nodes Are Visited
No
Yes
Compute Solutions' Length
Update Global Pheromone
If Enhancement on Solution Occurs
No
Yes
Compute the Distance of the Best Solution's neighbors
Choose the Nearest Neighbors
Inject Pheromone on the Nearest Neighbors
Termination Condition Satisfied?
No
Yes
Return Best Solution

FIGURE 5. Flowchart of EFACO algorithm.

B. NEIGHBORING SELECTION

The flying process in the FACO scans all neighboring WSs of the best solution to select a set of nearest neighboring to inject pheromones on them and increase their chance to be selected in future iterations.

In this work we replaced the flying process with a new neighboring selection method (shown in **Figure*6***) that inspired from [9] and [16]. The aim of this new method is to seek only S_N random numbers of neighboring WSs. These WSs are sorted based on the distance to a random selected web service $WS_{i,x}$ that belongs to the best solution found so far. Meanwhile, all these WSs should be in the same task t_i of $WS_{i,x}$. The pheromone injection on S_N neighboring WSs is similar to injection formula in FACO using Eq. (11). The new neighboring selection method proposed an adaptive balancing method between exploration and exploitation. It replaces the $WS_{i,x}$ with one of the S_N neighboring WSs in a way that guarantees exploration in early iterations by selecting randomly from farthest neighbors and exploitation in later iterations by selecting randomly from nearest neighbors.

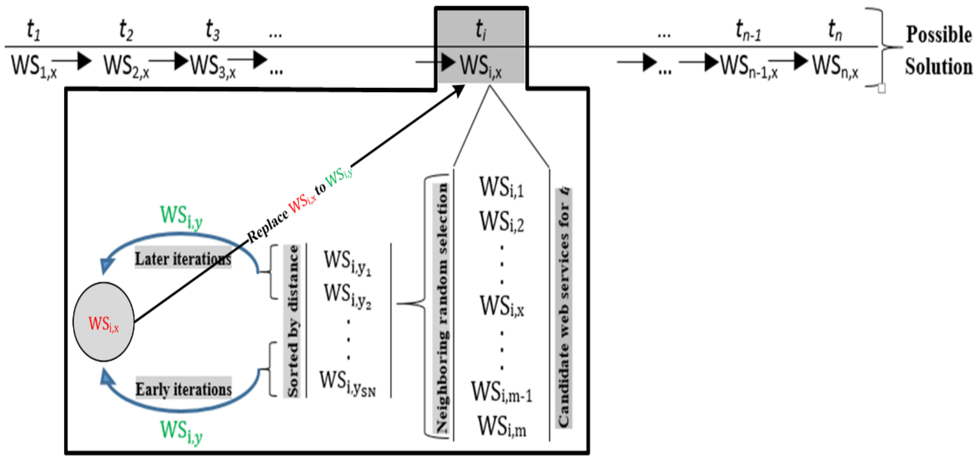

FIGURE 6. Neighboring selection process.

The adaptive balancing method is as follow:

$$y = \left(\frac{R}{j}\right) * S_N \tag{12}$$

where R is a random number $\in]0, 1]$, j is the iteration number, S_N is the numbers of neighboring WSs.

C. MULTI-PHEROMONE MECHANISM

In the FACO algorithm, the pheromone value was represented as a single value while in EFACO we represent it as multi-value based on the number of QoS attributes. Therefore, all equations, which are used in the standard ACO and FACO, were modified in EFACO accordingly.

The transition process (Eq. (5)) is responsible for selecting the next web service for an ant to move to and to create a solution based on the local heuristic (distance) η parameter of the problem and the amount of pheromone value τ parameter. These two parameters are calculated based on EFACO multi-pheromone mechanism as follows:

$$\begin{aligned}\tau_{(t,s)(t+1,s')} &= (\tau^{A}_{(t,s)(t+1,s')} + \tau^{R}_{(t,s)(t+1,s')}) \\ &\quad + (\tau^{c}_{(t,s)(t+1,s')} + \tau^{RT}_{(t,s)(t+1,s')}). \end{aligned} \tag{13}$$

$$\begin{aligned}\eta_{(t,s)(t+1,s')} &= (\eta^{A}_{(t,s)(t+1,s')} + \eta^{R}_{(t,s)(t+1,s')}) \\ &\quad - (\eta^{c}_{(t,s)(t+1,s')} + \eta^{RT}_{(t,s)(t+1,s')}). \end{aligned} \tag{14}$$

where C, RT, A, and R are the QoS attributes values.

The calculation formulas of the pheromone and the local heuristic are modified from FACO to carry out the difference between the QoS attributes.

The local (Eq. (6)) and global (Eq. (7) and Eq. (8)) pheromone update formulas are updated the pheromone trail for each QoS attributes because in EFACO the ants should consider the value of each attribute to find a solution.

In the EFACO, the preference-based method [51] manages the multi-objective problem as a single-objective maximization/minimization problem. Recall that the preference-based method aims to aggregate all objectives into one fitness objective function. Eq. (15) [7] shows the fitness calculation formula for the EFACO algorithm.

$$\mathrm{argmax}_{k\in \mathrm{Ants}}(f) = \frac{\left(W_a \prod_{t=1}^{n-1} A^{k}_{(t,s)(t+1,s')} + W_r \prod_{t=1}^{n-1} R^{k}_{(t,s)(t+1,s')}\right)}{\left(W_c \sum_{t=1}^{n-1} C^{k}_{(t,s)(t+1,s')} + W_{rt} \sum_{t=1}^{n-1} RT^{k}_{(t,s)(t+1,s')}\right)}, \tag{15}$$

where W_c, W_{rt}, W_a, and W_r denote the user selection weight the QoS attributes. n represents the task number. S_N represents to the k^{th} ant, and $1 \leq\ k\ \leq Ants$ where $Ants$ is the population size.

D. COMPLEXITY ANALYSIS

The EFACO algorithm has five processes. These processes should be used for analyzing the time complexity of EFACO. The processes consist of initialization of ants at the beginning of the algorithm, ants' transmission, local pheromone update, solution quality calculation, and the proposed enhancement process.

During initialization, all ants search randomly which means that each ant will select one web service from each task to construct its solution. Thus, the initialization time complexity is proportional to the population number (A) and the task number (N) which equal $A*N$.

Next, each ant will search for a new solution by calculating the distance between its WSs in task 1 with all WSs in task 2 based on the QoS attributes. This process repeats until it reaches the exit condition. Therefore, the ant's transmission is proportional to the population number, the task number, service number (M) in each task, repetition number (I), and QoS attributes (Q) which equal $A*N*M*I*Q$.

The ants move from task to task while searching and depositing pheromones on the fly. Thus, the deposition time complexity is $N*Q$.

Additionally, the ants calculate the quality of the solution based on the fitness function, Therefore, the solution quality calculation for time complexity is $N*Q$.

The best ant searches for its neighbors to share pheromones with them. Therefore, the worst case time complexity of the proposed enhancement is proportional to $I*S_N*N*M$ where S_N represents the number of neighbors.

Finally, we can calculate the EFACO time complexity based on the last process as $A*N + I(A*N*M*Q+N*Q+N*Q) + I*S_N*N*M$ which generates the algorithm $O(I*A*M*N*Q)$.

IV. EXPERIMENTS AND RESULTS

A. EXPERIMENTS SETTINGS

To evaluate the performance and effectiveness of the EFACO we conducted several experiments. This was achieved by comparing it with 6 different algorithms which are the standard ACO [8], FACO [7], Alayed *et al.* algorithm [16] (SACO for short), Xia *et al.* algorithm [20] (MACO for short), Qiqing *et al.* algorithm [21] (MOACO for short), and Li *et al.* algorithm [22] (MOCACO for short). All experiments were carried out on an Intel®i7-3770 (CPU 3.44 GHz, 8 GB RAM) running on Windows 8. All algorithms were implemented in MATLAB R2016a.

The control parameters for all algorithms are the same as those used in [7], to standardize experiences among all algorithms, which were determined empirically. Table 2 lists these parameter values. For each dataset, 30 experiments were performed.

TABLE 2. All algorithms' control parameters.

Parameters	Value
α	2
β	1
P	0.9
τ_0	0. 1
Ants	30
$Max_{iteration}$	100
K (number of neighbors)	5
FR (flying ratio)	0.9

B. EXPERIMENTS DATASETS

Following [7], all algorithms were compared using two datasets (*Dataset1* and *Dataset2*) with four QoS attribute values (cost, response time, availability, and reliability) that had been used in FACO as listed in Table 3.

TABLE 3. Workflow representation on each dataset.

Dataset1	Tasks	50
	WSs per task	30
Dataset2	Tasks	20
	WSs per task	15

For clarity, we conducted more extensive experiments using real dataset named Quality of Web Service (QWS) version 2.0 [52]. The QWS [52] includes real measurements of QoS properties for 2,507 web services using a web service broker in 2008. Each service was measured using nine QoS properties (response time, availability, throughput, successability, reliability, compliance, best practices, latency, and documentation).

These new datasets are grouped into two scenarios: the first scenario comprises of a fixed number of tasks (10 tasks) and a different number of WSs for each task, in the range [100, 200, 300,.., 1000]. This scenario includes 10 datasets and these datasets are sequentially named as *Dataset3, Dataset4,.., Dataset12*. The second scenario comprises of a fixed number of WSs (100 WSs) and a different number of tasks in the range [10, 20, 30,..., 100] for each dataset. This scenario includes 10 datasets more and these datasets are sequentially named as *Dataset13, Dataset14,.., Dataset22*. Therefore, the total number of datasets is 22 datasets.

Since we considered the QoS attributes as in FACO experiments, we used the same attributes that used in FACO and we expanded the QWS dataset with extra attribute that represents the cost attribute in the interval [45], [55].

C. EXPERIMENTS DISCUSSION

Several sets of experiments were conducted to evaluate the performance and effectiveness of the EFACO with competitors as described below.

In the first set of experiments, we compared EFACO including the restriction on flying ants and neighboring selection without multi-pheromones with ACO, FACO, SACO, MACO, MOACO and MOCACO. This experiment aims to show the effectiveness of the restriction technique and neighboring selection on FACO in terms of execution time. ***Figure 7*** and ***Figure 8*** show the results of this experiment. ***Figure 8*** shows that EFACO outperforms FACO, SACO, MACO and MOCACO in terms of execution time by 1.15s, 0.68s, 0.18s, and 0.48s respectively, while **Figure *7*** shows that SACO is better in terms of solution quality compared to other algorithms while FACO exhibits better performance than EFACO. ***Figure 8*** also shows that ACO and MOACO is better than EFACO in terms of execution time by 0.26s and 0.2s respectively.

As shown in the figures, EFACO with the restriction technique and neighboring selection enhance the execution time of FACO, but the solution quality decreases. In Thus, we conducted the second experiment.

The second experiment is shown in **Figure *9*** and **Figure *10***, this contains the comparison between EFACO, including the restriction technique, neighboring selection, and multi-pheromone with ACO, FACO, SACO, MACO, MOACO and MOCACO using the same datasets in Table 3. This experiment studies the effectiveness and performance of adding the multi-pheromone to the proposed algorithm. The experiments show that the solution quality improved for EFACO when compared with ACO, FACO, SACO, MACO,

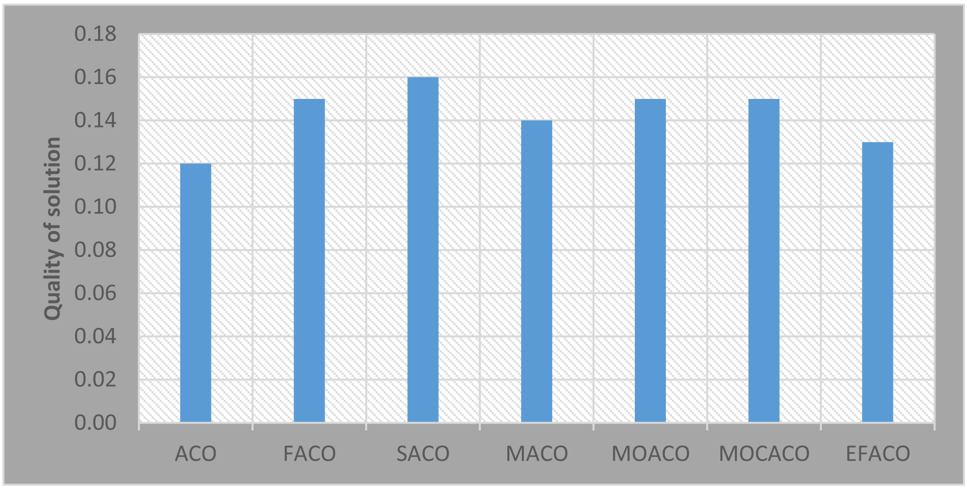

FIGURE 7. Results of ACO, FACO, SACO, MACO, MOACO, MOCACO and EFACO without multi-pheromone in terms of quality of solution.

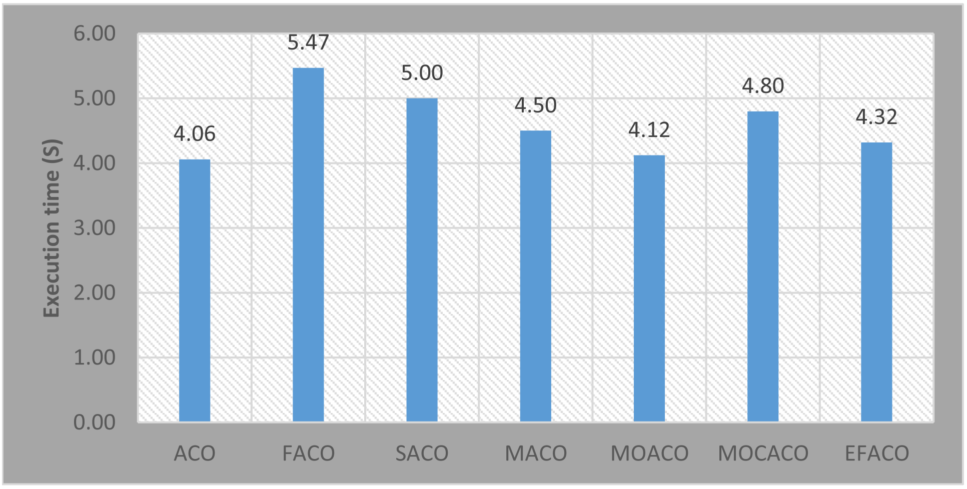

FIGURE 8. Results of ACO, FACO, SACO, MACO, MOACO, MOCACO, and EFACO without multi-pheromone in terms of execution time.

MOACO and MOCACO. Meanwhile, the execution time for EFACO is faster than FACO, SACO, MACO and MOCACO by 1.32s, 0.85s, 0.35s, and 0.65s respectively but it is slower than ACO and MOACO by only 0.09s and 0.03s respectively.

Figure 11 and ***Figure 12*** show the results in terms of quality of solution and execution time, respectively. ***Figure 11*** shows that EFACO ACO, FACO, SACO, MACO, MOACO and MOCACO algorithms consistently in terms of solution quality. ***Figure 12*** shows that EFACO needs less execution time compared to FACO, SACO, MACO, MOACO and MOCACO but it is very close to ACO.

Figure 13 and ***Figure 14*** show the experimental results for the second scenario of datasets in terms of quality of the performance for all algorithms, and execution time, respectively. ***Figure 13*** shows that EFACO also outperforms ACO, FACO, SACO, MACO, MOACO and MOCACO consistently in terms of solution quality. Furthermore, the performance gap between EFACO and other algorithms grows in favor of EFACO as the number of tasks increase. The figure also shows that the solution improves as the number of tasks increase in EFACO. This is because the number of tasks does not affect the performance of EFACO. ***Figure 14*** shows that EFACO also requires less execution time compared to FACO, SACO, MACO, MOACO, MOCACO, and is close to the execution time of ACO.

The experiments show that EFACO is more robust when compared to other algorithms, with different dataset distributions. The multi-pheromone helps EFACO to discover better solutions regardless of dataset variants. For execution time, the proposed enhancements bring EFACO closer to ACO, making it faster than FACO, SACO, MACO, MOACO and MOCACO because the restriction, and neighboring selection mechanisms prevent time wasting if there are no better solutions found thus far.

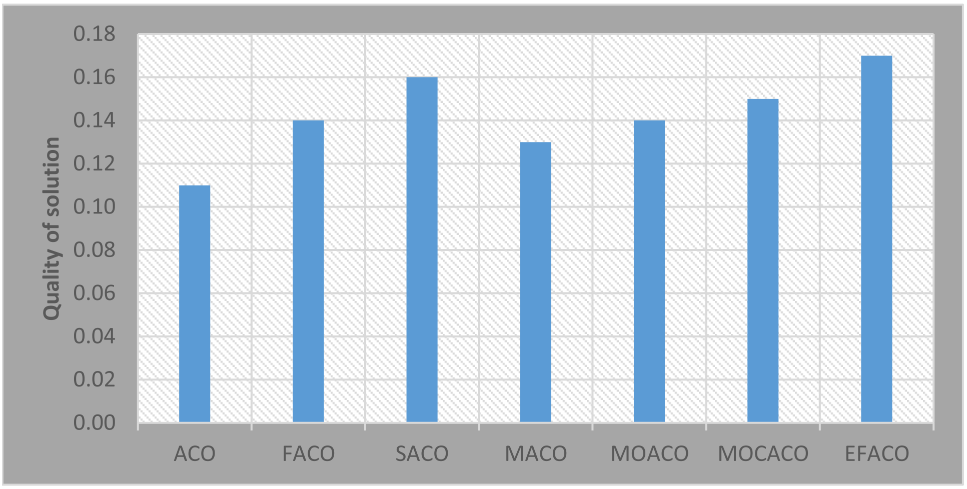

FIGURE 9. Results of ACO, FACO, SACO, MACO, MOACO, MOCACO, and EFACO with multi-pheromone in terms of quality of solution.

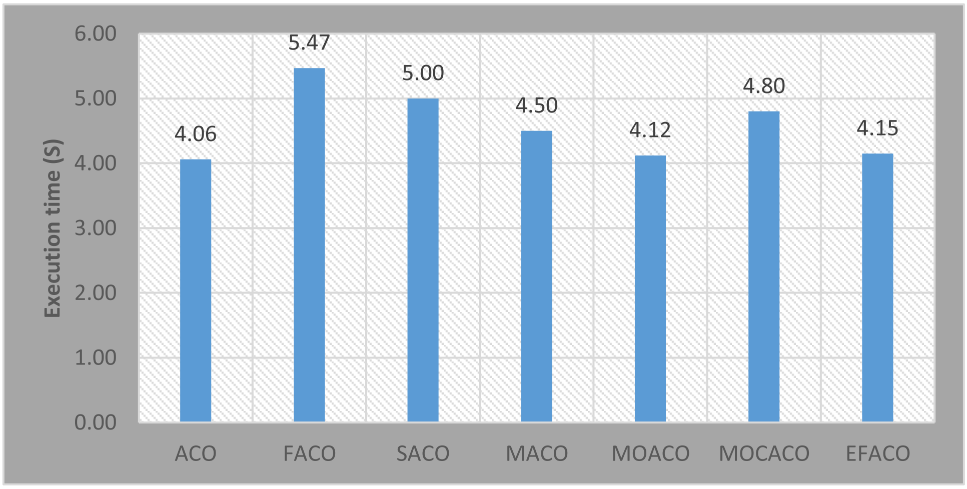

FIGURE 10. Results of ACO, FACO, SACO, MACO, MOACO, MOCACO, and EFACO with multi-pheromone terms of execution time.

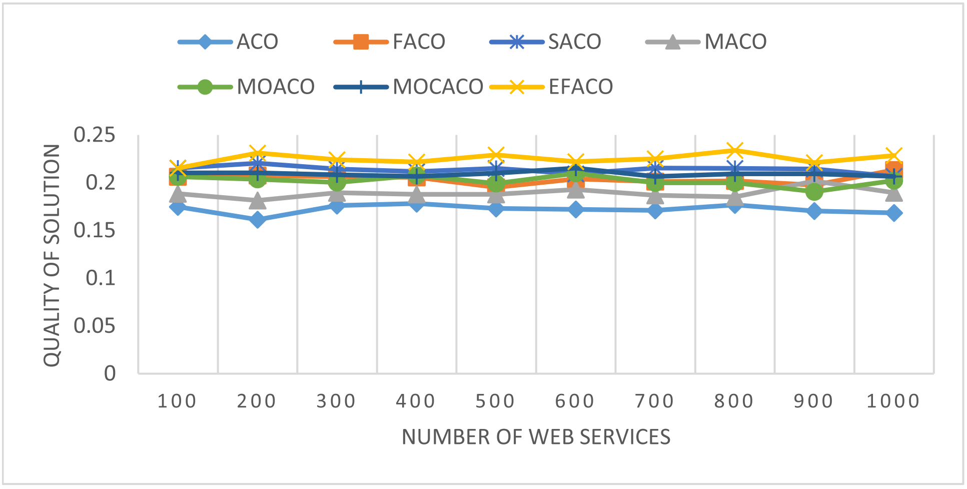

FIGURE 11. The solutions quality of all algorithms on first scenario.

***Figure* 15** presents the convergence speed on *Dataset22* of all Algorithms. Recall that *Dataset22* consists of 100 tasks and each sub task has 100 WSs. AS clearly seen from the figure, the EFACO has very fast convergence speed where it converges to the optimal value after 33 iterations. In contrast, the convergence speed of ACO, FACO, SACO, MACO,

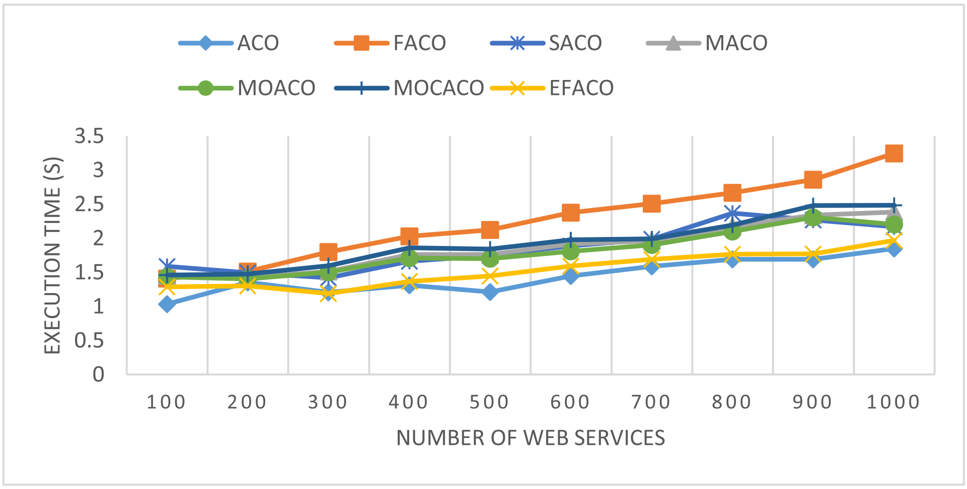

FIGURE 12. **The execution time of all algorithms on first scenario.**

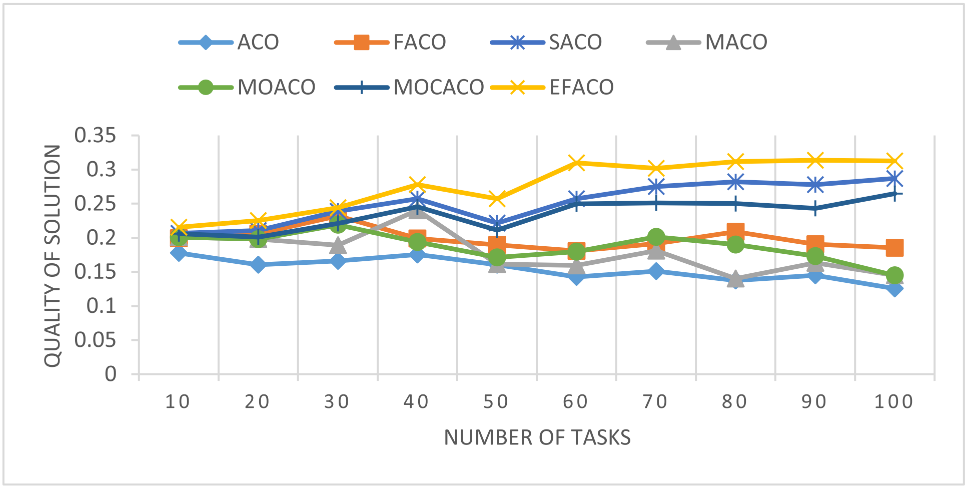

FIGURE 13. **The solutions quality of all algorithms on second scenario.**

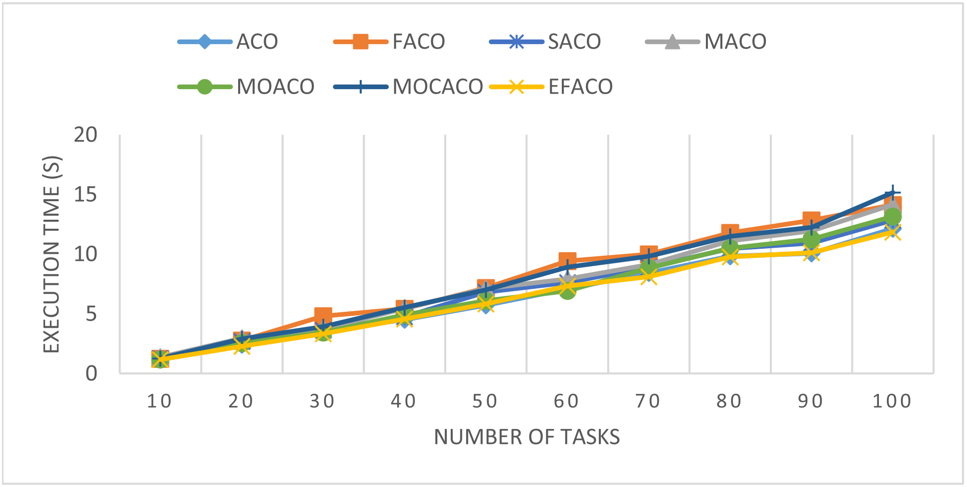

FIGURE 14. **The execution time of all algorithms on second scenario.**

MOACO, and MOCACO are in iterations 41, 67, 57, 54, 51, and 47 respectively. These results reflect the superiority of EFACO where the convergence speed that obtained by EFACO is obviously faster than other algorithms. Meanwhile, the EFACO obtained better value in terms of solution quality which is obviously larger than the values obtained by other

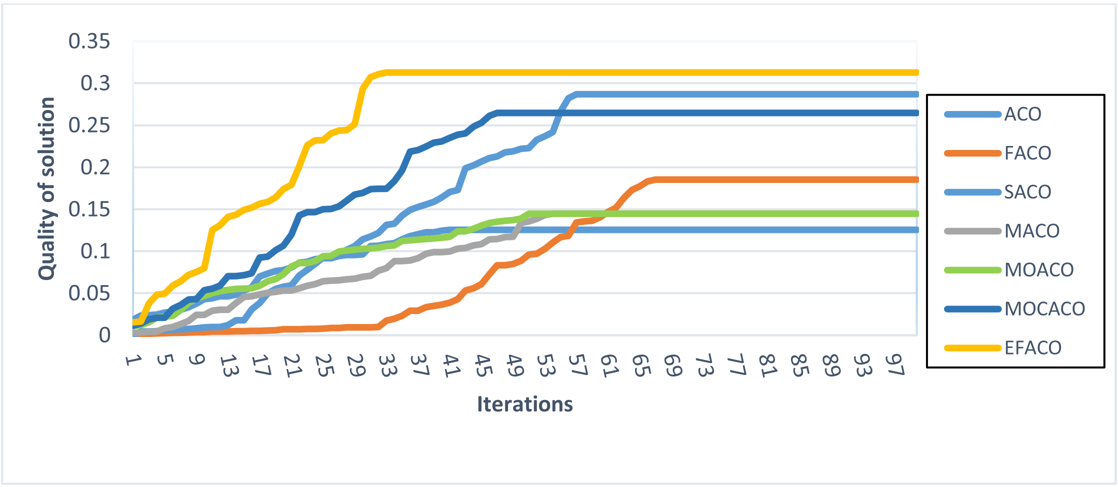

FIGURE 15. Convergence Curve on *Dataset22* of all Algorithms.

algorithms, this is because these algorithms have fallen into the local optimum. The convergence speed of EFACO supports the claim of this work where the proposed improvements adjust the performance of the FACO and ACO to get better solutions and avoiding the time problem.

Now, we have a set of independent measurements for each dataset from the EFCAO and the six other algorithms. These measurements represent the mean of 30 independent runs on each dataset. We noticed that there are differences between the mean of EFACO compared to all other algorithms so we need to measure these differences based on the null hypothesis (P>5%). This hypothesis is true if the mean across all algorithms is equal otherwise, it is false if instead this differs. Statistically significant terms indicate that the false hypothesis has been found. To achieve this, we used the ANOVA test on the 22 datasets, which is a robust statistical method for measuring the null hypothesis.

The results of ANOVA test show that EFACO is statistically significant for 13 datasets out of 22. This demonstrates that the value of the mean for 13 datasets is significant (false hypothesis) and for remaining 9 datasets the value of the mean is not significant.

V. CONCLUSION

FACO modifies ACO to solve the QoS-aware WSC problem. In comparison it produces positive solutions; however, one disadvantage is that it requires more execution time than the original ACO. In this paper, we introduced the EFACO algorithm which differs from FACO in three main ways. First, to avoid the execution time problem, EFACO restricts the flying process to only occur when there are improvements in solution quality. Although this improves on execution time, it decreases the solution quality. Second, we applied a neighboring selection method to avoid scanning all the neighboring nodes which also decreases the solution quality. Thus, we introduced the third modification, which transformed the algorithm into a multi-pheromone algorithm. This enhancement overcomes the drawback of the first, and second developments. Multiple experiments were then conducted to evaluate the effectiveness of the proposed solution. The experiments show that the restrictive flying process and neighboring selection decrease execution time, while the multi-pheromone improves solution quality. The results demonstrate that EFACO outperforms ACO, FACO, SACO, MACO, MOACO and MOCACO in terms of solution quality. Finally, EFACO requires less execution time than other algorithms with comparable execution time to ACO.

REFERENCES

[1] T. Erl, *SOA Principles of Service Design* (The Prentice Hall Service-Oriented Computing Series From Thomas Erl). Upper Saddle River, NJ, USA: Prentice-Hall, 2007.

[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] D. A. Menasce, "Composing Web services: A QoS view," *IEEE Internet Comput.*, vol. 8, no. 6, pp. 88–90, Nov. 2004.

[5] Y. Liu, A. H. Ngu, and L. Z. Zeng, "QoS computation and policing in dynamic Web service selection," in *Proc. 13th Int. World Wide Web Conf. Alternate Track Papers Posters (WWW)*, 2004, pp. 66–73.

[6] N. Ben Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas, and V. Issarny, "QoS-aware service composition in dynamic service oriented environments," in *Proc. ACM/IFIP/USENIX Int. Conf. Distrib. Syst. Platforms Open Distrib. Process.*, 2009, pp. 123–142.

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

[8] 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.

[9] 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, May 2017.

[10] 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.

[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, Jul. 2005.

[12] 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.

[13] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *Biosystems*, vol. 43, no. 2, pp. 73–81, Jul. 1997.

[14] 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.

[15] O. Deepa and A. Senthilkumar, "Swarm intelligence from natural to artificial systems: Ant colony optimization," *Netw. (Graph-Hoc)*, vol. 8, no. 1, pp. 9–17, 2016.

[16] H. Alayed, F. Dahan, T. Alfakih, H. Mathkour, and M. Arafah, "Enhancement of ant colony optimization for QoS-aware Web service selection," *IEEE Access*, vol. 7, pp. 97041–97051, 2019.

[17] A. Aljanaby, "An experimental study of the search stagnation in ants algorithms," *Int. J. Comput. Appl.*, vol. 148, no. 14, pp. 1–4, Aug. 2016.

[18] F. Dahan, K. El Hindi, H. Mathkour, H. AlSalman, "Dynamic flying ant colony optimization (DFACO) for solving the traveling salesman problem," *Sensors*, vol. 19, no. 8, p. 1837, Apr. 2019.

[19] A. Lipowski and D. Lipowska, "Roulette-wheel selection via stochastic acceptance," *Phys. A, Stat. Mech. Appl.*, vol. 391, no. 6, pp. 2193–2196, Mar. 2012.

[20] 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.

[21] 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.*, May 2009, pp. 37–42.

[22] W. Li and H. Yan-Xiang, "A Web service composition algorithm based on global QoS optimizing with MOCACO," in *Proc. Int. Conf. Algorithms Archit. Parallel Process.*, 2010, pp. 218–224.

[23] 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.

[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.*, Oct. 2010, pp. 551–555.

[25] 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.*, Aug. 2012, pp. 1998–2001.

[26] W. Zhang, C. K. Chang, T. Feng, and H.-Y. Jiang, "QoS-based dynamic Web service composition with ant colony optimization," in *Proc. IEEE 34th Annu. Comput. Softw. Appl. Conf.*, Jul. 2010, pp. 493–502.

[27] X. Zheng, J.-Z. Luo, and A.-B. Song, "Ant colony system based algorithm for QoS-aware Web service selection," in *Proc. 4th Int. Conf. Grid Service Eng. Manage. (GSEM)*, 2007, pp. 39–50.

[28] S. Asghari and N. J. Navimipour, "Cloud service composition using an inverted ant colony optimisation algorithm," *Int. J. Bio-Inspired Comput.*, vol. 13, no. 4, pp. 257–268, 2019.

[29] H. Bouzary and F. F. Chen, "A hybrid grey wolf optimizer algorithm with evolutionary operators for optimal QoS-aware service composition and optimal selection in cloud manufacturing," *Int. J. Adv. Manuf. Technol.*, vol. 101, nos. 9–12, pp. 2771–2784, Apr. 2019.

[30] Y. Yang, B. Yang, S. Wang, T. Jin, and S. Li, "An enhanced multi-objective grey wolf optimizer for service composition in cloud manufacturing," *Appl. Soft Comput.*, vol. 87, Feb. 2020, Art. no. 106003.

[31] 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.

[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 *Proc. Int. Conf. Algorithms Archit. Parallel Process.*, 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, pp. 1–9, Jul. 2013.

[35] D.-N. Le and G. N. Nguyen, "A new ant-based approach for optimal service selection with E2E QoS constraints," in *Proc. Int. Conf. Soft Comput., Intell. Syst., Inf. Technol.*, 2015, pp. 98–109.

[36] J. Shen and S. Yuan, "Qos-aware peer services selection using ant colony optimisation," in *Proc. Int. Conf. Bus. Inf. Syst.*, 2009, pp. 362–374.

[37] T. Ghafarian and M. Kahani, "Semantic Web service composition based on ant colony optimization method," in *Proc. 1st Int. Conf. Networked Digit. Technol.*, Jul. 2009, pp. 171–176.

[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. 12th Int. Symp. Symbolic Numeric Algorithms Sci. Comput.*, 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. Express*, 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)*, vol. 2, 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] C. Jatoth, G. R. Gangadharan, and R. Buyya, "Optimal fitness aware cloud service composition using an adaptive genotypes evolution based genetic algorithm," *Future Gener. Comput. Syst.*, vol. 94, pp. 185–198, May 2019.

[45] S. Chattopadhyay and A. Banerjee, "QoS-aware automatic Web service composition with multiple objectives," *ACM Trans. Web*, vol. 14, no. 3, pp. 1–38, Jul. 2020.

[46] A. K. Sangaiah, G.-B. Bian, S. M. Bozorgi, M. Y. Suraki, A. A. R. Hosseinabadi, and M. B. Shareh, "A novel quality-of-service-aware Web services composition using biogeography-based optimization algorithm," *Soft Comput.*, vol. 24, pp. 8125–8137, 2020.

[47] P. Asghari, A. M. Rahmani, and H. H. S. Javadi, "Privacy-aware cloud service composition based on QoS optimization in Internet of Things," *J. Ambient Intell. Hum. Comput.*, 2020, doi: 10.1007/s12652-020-01723-7.

[48] A. Ait Wakrime, M. Rekik, and S. Jabbour, "Cloud service composition using minimal unsatisfiability and genetic algorithm," *Concurrency Comput. Pract. Exper.*, vol. 32, no. 15, p. e5282, Aug. 2020.

[49] T. Stützle and H. H. Hoos, "MAX–MIN ant system," *Futur. Gener. Comput. Syst.*, vol. 16, no. 8, pp. 889–914, 2000.

[50] 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, 2013.

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

[52] E. Al-Masri and Q. H. Mahmoud, "QoS-based discovery and ranking of Web services," in *Proc. 16th Int. Conf. Comput. Commun. Netw.*, 2007, pp. 529–534.

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 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, Taiz University, Taiz, Yemen. His research interests include optimization and swarm intelligence.

KHALIL EL HINDI received the B.Sc. degree from Yarmouk University, Jordan, and the M.Sc. and Ph.D. degrees from the University of Exeter, U.K. He is currently a Professor with the Department of Computer Science, King Saud University. His research interests include deep learning methods, Bayesian classifiers, similarity metrics for instance-based learning, and swarm intelligence.

AHMED GHONEIM (Member, IEEE) received the M.Sc. degree in software modeling from the University of Menoufia, Egypt, in 1999, and the Ph.D. degree in software engineering from the University of Magdeburg, Germany, in 2007. He is currently an Associate Professor with the Department of Software Engineering, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia. His research interests include address software evolution, service-oriented engineering, software development methodologies, net-centric computing, and human–computer interaction.

HUSSAIN ALSALMAN received the B.Sc. and M.Sc. degrees in computer science from King Saud University (KSU), Riyadh, Saudi Arabia, and the Ph.D. degree in artificial intelligence from U.K. He worked for several years as a Consultant for a number of companies in private sector and institutes in government sector, Saudi Arabia. From 2009 to 2014, he chaired the Computer Science Department, College of Computer and Information Sciences, KSU. He is currently a Staff Member with the Computer Science Department, KSU. He was a member of review board of *Saudi Computer Journal* from 2004 to 2014.