Journal of Ambient Intelligence and Humanized Computing (2020) 11:3975–3987
https://doi.org/10.1007/s12652-019-01631-5

ORIGINAL RESEARCH

FACO: a hybrid fuzzy ant colony optimization algorithm for virtual machine scheduling in high-performance cloud computing

Awatif Ragmani[1] · Amina Elomri[1] · Noreddine Abghour[1] · Khalid Moussaid[1] · Mohammed Rida[1]

Received: 24 July 2019 / Accepted: 30 October 2019 / Published online: 13 December 2019

Abstract
High-performance cloud computing has recently become the focus of much interest. Extensive research has shown that scheduling and load balancing are among the key aspects of performance optimization. The allocation of a set of requests into a set of computing resources, which is considered as an NP-hard problem, aims to distribute efficiently the load within the cloud architecture. To resolve this problem, the last decade has seen a growing trend towards using hybrid approaches to combine the advantages of different algorithms. In this paper, we propose a hybrid fuzzy ant colony optimization algorithm (FACO) for virtual machine scheduling to guarantee high-efficiency in a cloud environment. The proposed fuzzy module evaluates historical information to calculate the pheromone value and select a suitable server while keeping an optimal computing time. The experimental work presented in this study provides one of the first investigations into how to choose the optimal parameters of ant colony optimization algorithms using the Taguchi experimental design. We have simulated the proposed algorithm through the Cloud Analyst and CloudSim simulators by applying different cloud configurations to evaluate the performance of the proposed algorithm. Our findings highlight how response time and processing time are improved compared to the Round Robin algorithm, Throttled algorithm and Equally Spread Current Execution Load algorithm, especially in the case of a high number of nodes. FACO algorithm could be applied to define efficient cloud architecture adapted to high-performance applications.

Keywords Ant colony optimization · Fuzzy logic · Cloud computing · Load balancing · Scheduling · Taguchi DOE

1 Introduction

Cloud computing corresponds to programs and services that run on a distributed network based on virtualized infrastructure and accessed using common Internet protocols and networking standards. It is an efficient and economical model for provisioning different types of services. Cloud computing is based on two main concepts including abstraction that is based on the idea of pooling physical resources and virtualization (Sosinsky 2011). The virtualization technique is based on the concept of sharing and abstraction of material resources. Thus, a physical machine can host multiple virtual machines (VMs) and can be used by several users at the same time. Virtualization relies on a central operating system called a host system. This technology has different benefits such as energy efficiency, optimization of infrastructure costs, and the possibility to migrate a virtual machine from one server to another almost instantly (Mijumbi et al. 2016). The last decade has seen a rapid development of Cloud applications in many industries because of their economic and technical advantages (Arunarani et al. 2019; Routaib et al. 2014). Particularly, this paradigm is characterized by its elasticity, which allows suppliers to immediately adjust storage capacity and computing resources to users' requirements (Tamilvizhi and Parvathavarthini 2019).

However, networks and the Internet's exponential growth has made classical administration techniques inadequate. In particular, Cloud services providers have to deal with new challenges in the areas of scheduling and load balancing of a strongly complex and widely extended network. Firstly, the scheduling policy plays a critical role in Cloud architecture. In the case of inappropriate scheduling strategy, the performance decreases (Arunarani et al. 2019). Secondly, load balancing is a technique of dispatching the workload

✉ Awatif Ragmani
ragmaniawatif@gmail.com

[1] LIMSAD Laboratory, Faculty of Sciences Ain Chock, University Hassan II of Casablanca, 20100 Casablanca, Morocco

on servers to avoid overloaded nodes. Regarding the high number of virtual machines of a Cloud architecture, the main challenge faced by many researchers is the definition of an efficient load balancing strategy (Shetty and Shetty 2019).

Commonly, load balancing algorithms are classified into two groups: static algorithms and dynamic algorithms. Static algorithms are characterized by their simple operating concept, which reduces their turnaround time. However, their applications are limited to a few specific cases. Dynamic algorithms are more suitable for networks with many nodes and queries. Several attempts have been made to develop efficient load balancing algorithms adapted to Cloud architecture. However, the millions of virtual machines in the Cloud require intelligent and autonomous management techniques to ensure load balancing in data centers (Mikaeeli Mamaghani and Jabraeil Jamali 2019).

Recently, many research studies have shown an increased interest in the applications of meta-heuristic algorithms to improve load balancing policies (Xu et al. 2017). Since its introduction by Dorigo et al. (2006), the ant colony algorithm was applied to solve several NP-hard problems. This algorithm is an approach inspired by the behavior of ants in finding the optimal paths from the nest to the food. The ants work as a group based on an indirect method of communication facilitated by pheromone. The premature convergence probability of the system is minor.

In this paper, we implement a hybrid fuzzy ACO algorithm to optimize three objectives, including response time, processing time, and load balancing. We propose a hybrid FACO algorithm which includes two novel contributions. The first contribution is the definition of a fuzzy module dedicated to pheromone evaluation, which aims to improve the general performance of the ACO algorithm. Our results show that the proposed fuzzy controller presents better processing time compared to the classical pheromone probability calculation. The second contribution is the identification of the optimal ACO parameters using the Taguchi experimental design (Taguchi et al. 2005). To the best of our knowledge, few studies have investigated the application of Taguchi experience design to evaluate the interaction between key performance indicators (KPI) and the values of ACO parameters. The different simulations conducted in the Cloud Analyst simulator confirmed the effectiveness of the proposed FACO algorithm compared to previous algorithms such as Round Robin.

This study is an extended version of Ragmani et al. (2019) with a further method and results discussion. The remainder of the paper is organized as follows. Section 2 and section 3 review the related works and the main concepts of this study. Section 4 describes our approach to modeling the studied system. In Section 5 the FACO load-balancing algorithm is presented, and Section 6 highlights the simulations and key findings. Section 7 concludes the paper.

2 Related works

Load balancing and scheduling in cloud computing have been widely examined in the literature, and several policies have been proposed to improve the efficiency and performance of applications and services. This section introduces various points of view adopted by researchers. Shetty and Shetty (2019) investigated the administration of the millions of simultaneous requests from users and introduced a modified central load balancer (MCLB) algorithm where the load is balanced among all the available virtual machines. This algorithm aims to avoid the overloading and under loading of virtual machines.

Seghir and Khababa (2018) focus on the QoS-aware Cloud service design question and came up with a hybrid genetic algorithm (HGA) to deal with it. The suggested algorithm merges two phases to achieve the evolutionary process search, which combines the genetic algorithm stage and the fruitfly optimization stage. The authors have applied the Taguchi DOE to describe the parameter settings of the introduced HGA. The experimental results establish that the introduced algorithm exceeds the simple genetic algorithm and the simple fruit fly optimization algorithm. In Zahoor et al. (2018), the authors indicate that Cloud and Fog computing offer on-demand computing resources that contribute to an appropriate solution to resolve smart grid hurdles. They proposed a cloud-fog-based model for resource administration in the smart grid. This strategy presents various good features including flexibility, cost, and energy-saving, scalability, and agility.

Yu et al. (2016) introduced a two-stage policy to improve the efficiency of task scheduling and limit needless task allocation. The proposed stochastic load balancing strategy tries to avoid resource overloading with virtual machine migration and reducing the total migration overhead. Gao et al. (2013) proposed an effective strategy for optimizing the resources used as well as improving the quality of service within the Cloud computing environment. The proposed methodology is based on the definition of a dynamic and efficient workload balancing policy inspired by ACO algorithms.

Zhang et al. (2017) implemented a solution, called CBMinDia, that relies on grouping methods for the placement of VMs within the different data centers. The proposed algorithm is based on the characteristic of two approximations and is more suitable in the case of large data centers (DC). The algorithm is based on the exploitation of the information relating to the density and the current capacity of the network. The experiments carried out by the authors demonstrate that the proposed algorithm is more efficient in the case of clustered data center distribution. Concerning the virtual machine partition, the

authors defined a more efficient algorithm by introducing the concept of Half Communication Model based on the definition of two parameters which are approximate outer traffic (AOT) and approximate inner traffic (AIT). Those parameters are used for selecting a suitable node in order to optimize intra-DC and inter-DC traffic. The approach adopted by the authors made simplified the VMs allocation which induces an improvement in efficiency by a factor of three.

Bui et al. (2017) applied the concept of the non-cooperative game for the definition of a policy of virtual machine scheduling that guarantees an optimal load balancing state. The model proposed by the authors makes it possible to calculate the gain of the game by relying on load balancing parameters and waste resources. The strategy defined by the authors relies on a distribution indexing of virtual machines within physical machines according to the load state of the physical devices.

Boveiri et al. (2019) introduced a high-performance method based on the Max-Min Ant System (MMAS). This approach is a variation in the family of ACO algorithms. The authors aimed to manipulate the priority values of requests to improve the robustness and efficiency in the multiprocessor task-graph scheduling problem. Gao and Wu (2015) proposed an effective strategy for optimizing the resources used as well as improving the quality of service within the Cloud computing environment. The contribution of this study lies in the description of a load balancing algorithm inspired by an ACO algorithm. The operating parameters of the proposed algorithm were strongly adapted to the case of Cloud computing by defining improved pheromone functions.

3 Background

3.1 Ant colony optimization algorithms

The ant colony optimization approach is a section of metaheuristic algorithms based on the performance of ants colonies during their mutual quest for food. Initially, ants move aimlessly in all directions searching the finest sources of food. After each journey, ants deposit a chemical substance called pheromone, detectable by other ants. After several round-trips between the feed sources and the nest, the accumulation of pheromones on the straightest routes is enlarged while the pheromones previously deposed on the longer paths are evaporated (Gonzalez-Pardo et al. 2017). During trips, ants choose their new path according to a probability p based on the concentration rate of pheromone τ. Finally, only the shortest paths will remain visible to the other ants. These conclusions inspired Dorigo et al. (2006) when setting up the ant colony optimization algorithm, which allowed the identification of optimal solutions to several combinatorial problems. Also, ACO algorithms are increasingly applied to continuous and multi-objective optimization issues. ACO algorithm includes two key stages: building local solutions and updating the pheromone trail. The ultimate solution is defined, step by step, based on probabilities that are calculated by exploiting the pheromones τ and a heuristic value η (Arunarani et al. 2019). In the case of having several matrices of pheromones, it is essential to use an aggregation of the matrix of pheromone and heuristic function. The most commonly used functions are the weighted sum, the weighted product method, and random selection. The value of the weight coefficient may be static or dynamic (Gao et al. 2013). The standard structure of the ACO algorithm is outlined in algorithm 1.

Algorithm 1 Main steps of ACO Algorithm

```
Initialization
repeat
    The ants are placed on the starting nodes
    Each ant achieves a state transition procedure to gradually build a solution
    Each ant updates the local pheromone until all the ants have built a solution
    Update of the overall pheromone under rule preset until the achievement of the ending condition
until End condition
Stop further iterations
```

3.2 Fuzzy logic controller

Fuzzy logic is currently attracting a great deal of interest from researchers, engineers and industry who need to automate decision making in their field and build artificial systems capable of performing the tasks usually supported by humans (Van Broekhoven and De Baets 2008). The fuzzy logic controller, as introduced by Mamdani and Assilian Mamdani and Assilian (1999) is presently considered as one of the most significant applications of the fuzzy set theory initiated by Zadeh (Li and Tong 2017; Masulli et al. 2013). This approach is based on the concept of the fuzzy set, which is a generalization of the ordinary set. The main feature of this theory is the definition of a membership function f^a that takes the value of a degree from the interval [0, 1]. In other words, the membership function characterizes the level of truth in fuzzy logic. The membership function is defined by various graphical forms. In brief, a fuzzy set A is described as pairs:

$$((x, u_A(x)), A = (x, u_A(x)|x \in A, u_A(x) \in [0, 1]) \quad (1)$$

Definition 1 Let X be a set. A fuzzy subset A of X is characterized by a membership function $f^a : X \rightarrow [0, 1]$.

Fuzzy logic controllers classically describe a non-linear mapping from the system's state space to the control space. Therefore, the output of the fuzzy logic controller could be considered a non-linear control surface. The substantial advantage of fuzzy controllers is the establishment of the control system for problems that cannot be defined by accurate mathematical formulas. As depicted in Fig. 1, a fuzzy logic control includes:

- A Knowledge Base (KB) that consists of the information exploited by the expert administrator in the form of linguistic control rules;
- A Fuzzification Interface that translates the crisp values of the input variables into fuzzy sets, which will be used in the fuzzy inference procedure;
- An Inference System that applies the fuzzy values produced by the Fuzzification Interface and the information included in Knowledge Base to accomplish the reasoning procedure;
- The Defuzzification Interface that receives the fuzzy action from the inference procedure and transforms it into crisp values to be used by the control variables (Cingolani and Alcalá-Fdez 2013).

The concept of fuzzy logic presents various features including the most straightforward aspect of its mathematical model because fuzzy reasoning is considerably intuitive.

Definition 2 According to the usual definitions of fuzzy operators, we have the properties of commutativity, distributivity, and associativity of classical operators. However, there are two notable exceptions:

- In fuzzy logic, the law of excluded middle is contradicted : $A \cup \overline{A} \neq X$;in other words $u_{A \cup \overline{A}}(x) \neq 1$;
- In fuzzy logic, an element can belong to A and non A simultaneously, in other words $A \cap \overline{A} \neq \oslash$.

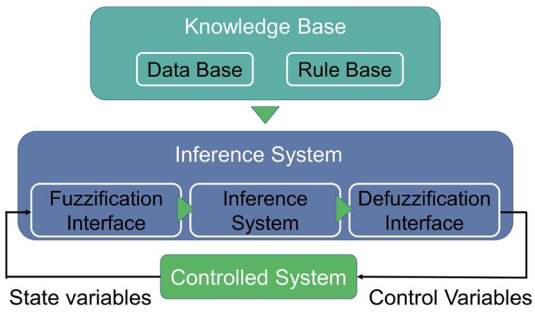

Fig. 1 The basic structure of fuzzy logic controller (Cingolani and Alcalá-Fdez 2013)

4 Problem statement

The proposed algorithm tries to define the optimal configuration for virtual machine placement in such a way to optimize resource utilization. The Table1 presents a comprehensive list of parameters and terminologies used throughout the paper.

Definition 3 Let A be a set of virtual machines created at a time t and B all the servers hosting virtual machines from set A. We note $|A|$ and $|B|$ in each case the number of virtual machines and the number of physical machines available at instant t.

As illustrated in Fig. 2, each physical machine b_j is defined by a CPU_j, a memory capacity M_j, a storage capacity S_j and bandwidth Bd_j.

Definition 4 We note L the set of requests to be processed by the system and $|L|$ the number of queries. Each query l_k is represented by the location of the user who initiates the query, the size of the request and the frequency of the query.

Each server b_j is identified by the location within data centers, memory, storage, CPU, and bandwidth. Each VM a_i has a cost depending on the data center, which hosts the server. Each request l_k is supported by a virtual machine a_i.

Because this study is conducted via the simulation platform Cloud Analyst (Wickremasinghe et al. 2010), the description of the studied problem is partially influenced by the architecture of the simulator. The allocation of virtual machines to servers is provided by two algorithms. The first algorithm concerns a dynamic scheduling policy and the second algorithm is a load balancing strategy that applies the ACO algorithm and fuzzy logic concept. The scheduling algorithm is adjusted dynamically according to the constraints of the load balancing target to avoid any imbalance within the cloud. We define two variables x_{ij} which will be equal to 1 if the machine i is assigned to the server j and zero otherwise, and y_j which specifies if the server j is working or not.

$$\sum_{j=1}^{m} \sum_{i=1}^{n_j} x_{ij} \times CPU_i \leq \sum_{j=1}^{m} y_j \times CPU_j \tag{2}$$

$$\sum_{j=1}^{m} \sum_{i=1}^{n_j} x_{ij} \times storage_i \leq \sum_{j=1}^{m} y_j \times storage_j \tag{3}$$

$$\sum_{j=1}^{m} \sum_{i=1}^{n_j} x_{ij} \times memory_i \leq \sum_{j=1}^{m} y_j \times memory_j \tag{4}$$

where

Table 1 Summary of notation

Parameter	Definition
α	The influence factor of the pheromone
β	The influence factor of cost
γ	The influence factor of the queue of user's requests
ρ	Evaporation ratio
σ	Weight of the CPU
ω	Weight of the memory
φ	Weight of the storage
δ	Weight of the bandwidth
Num_{ants}	Number of ants generated by the algorithm
$Max_{Iterations}$	Maximum number of iterations
Total cloudlets	The waiting queue request of a given virtual machine
Total cost	Includes transfer cost and virtual machine cost charged by virtual machine i to handle a user request r
SNR	Signal to noise ratio used by Taguchi concept to evaluate each input factor
DOE	Design of experiments which is an approach to problem solving
b_j	A physical machine
l_k	A user's request
a_i	A virtual machine
$\|A\|$	The number of VMs
$\|B\|$	The number of computing nodes
σ, ω, φ, and δ	The weight of the CPU, the memory, the storage, and the bandwidth

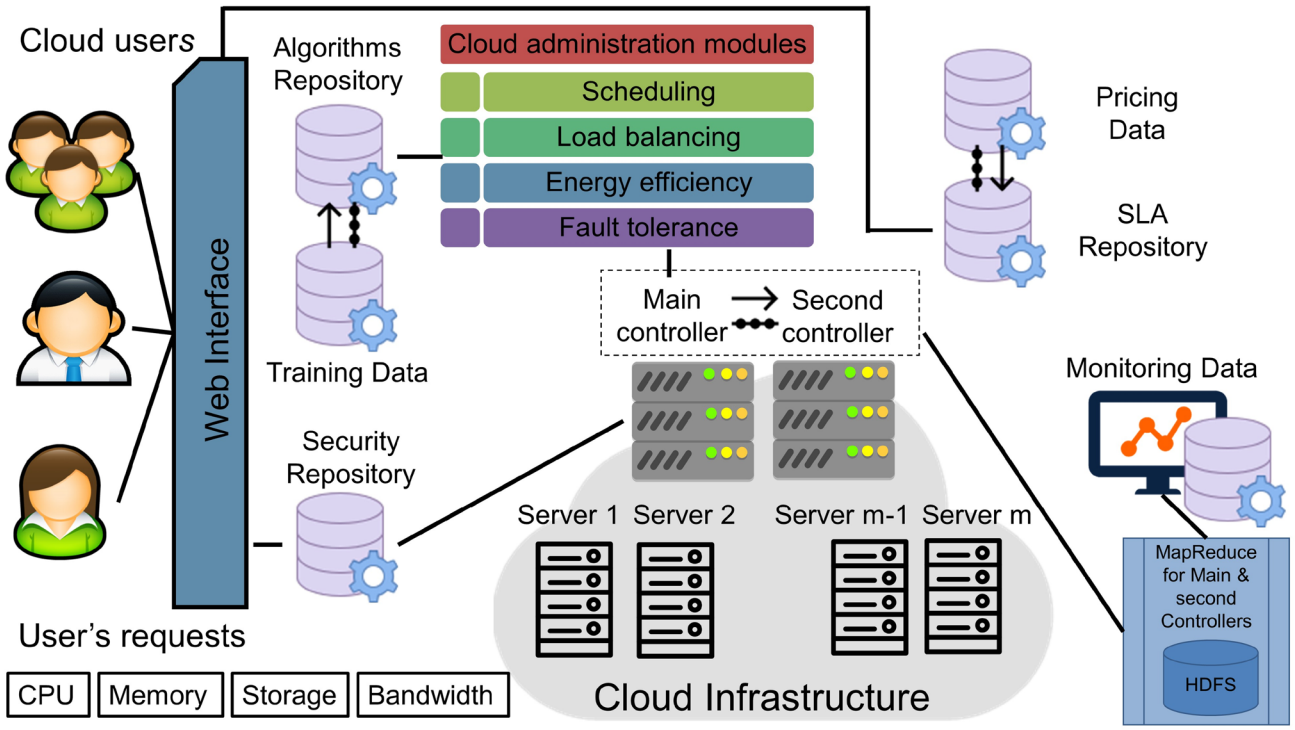

Fig. 2 The proposed cloud computing architecture (Ragmani et al. 2017)

$$\sum_{i=1}^{m} n_j = |A|, m = |B| \quad (5) \qquad \sum_{i} x_{ij} = 1 \quad (6)$$

Constraints (2), (3), and (4) ensure that the capacity of VMs placed on a server b_j is less than the capacity of the server. The constraint (6) guarantees the fact that the virtual machine a_i is assigned to a unique server of b_j.

The algorithm of virtual machine scheduling and load balancing attempts to assign each user's request to a suitable server while improving the balance of the load within the cloud architecture. In this paper, we use four features of resources which are CPU, memory, storage, and bandwidth. We defined the load balancing indicator (LBI) to measure the ratio of resource utilization as shown in equation (7). The parameters σ, ω, φ, and δ are used to set respectively the weight of the CPU, the memory, the storage, and the bandwidth. Thus the system's administrator could adapt the different weights regarding the technical characteristics of the user's requests.

$$LBI = \sum_{j=1}^{m} \left(\sigma \times \left(\frac{CPU_j}{CPU_{maxj}} \right) + \omega \times \left(\frac{Memory_j}{Memory_{maxj}} \right) + \varphi \times \left(\frac{Storage_j}{Storage_{maxj}} \right) + \delta \times \left(\frac{Bandwidth_j}{Bandwidth_{maxj}} \right) \right) \tag{7}$$

Subject to

$$\sigma + \omega + \varphi + \delta = 1 \tag{8}$$

The total cost includes the cost of the virtual machine and the cost of data transferred from one location to another.

$$Total_{Cost} = Transfer_{Cost} + VirtualMachine_{Cost} \tag{9}$$

As described in equations (2) to (7), we can observe that the studied problem has to respect multiple constraints. This case study investigates the optimal virtual machine allocation. In other words, the solution is a set of pairs (a_i, b_j) where a_i is a virtual machine in charge of processing the user's request l_k and b_j is the server which hosts the virtual machine a_i. The combinatorial optimization problem is defined by a set of variables $x_1, x_2, x_3, \cdots, x_n$ and a set of constraints $y_1, y_2, y_3, \cdots, y_m$ attached to different variables and the objective function $f(x)$. In our case study, the problem is to identify all the possible combinations represented by:

$$C = \{c = (a_1, b_1), (a_2, b_3), \cdots, (a_i, b_j), \cdots, (a_n, b_m)\} \tag{10}$$

where c is any possible solution (a_i, b_j) answering y_i constraints that meet memory, CPU, storage and bandwidth requirements.

C is commonly called a solution space where each pair is considered a potential solution. In other words, solving the problem corresponds to the identification of the best solution $c_o ptimal \in C$ which minimizes the objective function f. Given the complexity of solving combinatorial problems, several algorithms have been developed to reduce the response time and the identification of the optimal solution process. Let n be the number of VMs to be assigned to the m servers placed in different data centers in the Cloud environment. The problem to be solved is to minimize the total VMs cost while optimizing the load balancing state as shown in equation (11).

$$\min f = \min \left(d \times LBI + (1 - d) \times \sum_{i=1}^{n} (x_{ij}) \times \frac{Total_{cost_i}}{maxTotal_{cost}} \right) \tag{11}$$

where d could take values between 0 and 1. The case of $d = 0$ induces the minimization of the cost and $d = 1$ represents the case of optimizing the system load without any impact on the VMs cost.

5 Proposed FACO algorithm

We chose to improve an ACO algorithm to solve the problem of scheduling and load balancing in the cloud system. This decision is motivated by the capabilities of the ACO algorithm to find high-quality solutions while maintaining a low computing time. The implementation of an ACO algorithm requires the identification of the pheromone formulation and the data that will be used to calculate the pheromone values. In our case, the pheromone allows the evaluation of the adaptability of a selected server b_j to receive a VM a_i based on its technical capacity, cost, and heuristic information. The heuristic part is updated after each successful assignment of a VM a_i to a server b_j. The implemented Ant Class describes the process of the proposed FACO algorithm including the update of the pheromone and the fuzzy module dedicated to the evaluation of the quality of the destination node(see Algorithm 2).

The FACO makes it possible to reduce the computing time by replacing the calculation of the pheromone value by a fuzzy evaluation. The calculation of the pheromone is assigned to a fuzzy controller that receives as inputs the values of memory, storage, CPU and bandwidth. This module delivers as output a pheromone value that can be high which will attract the next ants or medium or low.

For each iteration, some ants look for the best configuration, which guarantees an optimal response time and the best load balancing level. Each ant has to visit all available nodes. At the end of each iteration, the identified solution is compared to the previous one to keep the optimal path and routes passed by the ants. Then, the pheromone values are updated by the evaporation ratio. The algorithm repeats this process until the end condition is reached or the optimal solution is found.

In brief, the proposed FACO algorithm includes two significant steps. The first step is to identify the local solution that corresponds to an optimal allocation of the virtual machine and the second step corresponds to the update of

the pheromones matrix by emphasizing servers that are still available and avoid unavailable servers. After each iteration, the ants update the matrix of pheromones and the probability values. At the beginning of the algorithm, the ants are distributed randomly (Gendreau and Potvin 2010). Then after each step, the ants apply each a probabilistic function to select the next location to move on it. The calculation of the probability for an ant k, placed in the node s to move to node d is done according to the following equation:

$$P^k(s,d) = \frac{[\tau(s,d)]^{\alpha}[\eta(s,d)]^{\beta}[\mu(s,d)]^{\gamma}}{\sum_k[\tau(s,d)]^{\alpha}[\eta(s,d)]^{\beta}[\mu(s,d)]^{\gamma}} \quad (12)$$

where $\tau(s,d)$ is the pheromone value calculated using fuzzy module, $\eta(s,d) = 1/totalcost$, $\mu(s,d) = 1/totalCloudlets$, α is the influence factor of the pheromone, β is the influence factor of cost, and γ is the influence factor of the queue of user's requests.

Algorithm 2 FACO algorithm

Require: VM List, Cloudlets List, Hosts, Datacenters
return List of pairs VMs and Hosts (a_i, b_j)
Initialize pheromones Table $(VMlist.size + 1)(VMlist.size + 1)$
Initialize ants
Set algorithms parameters
while $Time < T$ **do**
  while $nextVM \neq CurrentVM$ and $itteration < Max_{iteration}$ **do**
    $CurrentVM = nextVM$
    SendAnt()
    for $i = 0$ to $i = probability.length$ **do**
      if VM(i) is Visited **then**
        probability (i) = 0
      else
        $probability(i) = \frac{[\tau(s,d)]^{\alpha}[\eta(s,d)]^{\beta}[\mu(s,d)]^{\gamma}}{\sum_k[\tau(s,d)]^{\alpha}[\eta(s,d)]^{\beta}[\mu(s,d)]^{\gamma}}$
      end if
      $sum = sum + probability(i)$
      i++
    end for
    for $i = 0$ to $i = probability.length$ **do**
      $probability(i) = \frac{probability(i)}{sum}$
      i++
    end for
    for $i = 0$ to $i = ants_{number}$ **do**
      calculate randomization
      if randomization <= probability(i) **then**
        $VM = i$
      else
        $VM = random(VMList.size)$
      end if
    end for
    VM = MemorizeBestSolution()
    AffectVM(VM)
    UpdatePheromone(CurrentVM, nextVM) based on fuzzy module
    itteration++
  end while
  Update GlobalPheromone ()
end while

5.1 Pheromone trials

In the beginning, the entire edges have an identical amount of pheromone. Then, the value of the pheromone is updated after each passage of the ants according to the quality of the solution. The novelty of our algorithm is to calculate the value of the pheromone of each solution using the fuzzy logic concept (Cingolani and Alcalá-Fdez 2013). The quality of each solution is evaluated via a fuzzy logic module. In general, the pheromone can increase or decrease according to the quality of the current solution compared to the previous solution. In other words, if the technical characteristics that include the available storage, memory, bandwidth and the CPU of the new solution are better, the pheromone increases and in the opposite case the pheromone decreases.

As described in Table 2, the proposed fuzzy logic module applies ten (If-Then) rules to evaluate the quality of the pheromone to be dropped. To define the membership function for the memory, bandwidth, CPU, and storage we apply triangular functions due to its practical aspect (see Fig. 3). The fuzzy set for inputs is {low, medium, high}. The applied output function is Gaussian and the fuzzy set for the output pheromone is {low, medium, good} (see Fig. 4).

In brief, the ant that constructs the best solution will get a higher quantity of pheromone. Furthermore, the proposed algorithm applies a procedure for the calculation of the rate of evaporation of the pheromone trail as depicted in equation 13. This rate is used to encourage the ants to explore new paths and to avoid early convergence. Indeed, an early convergence could impact negatively the workload within the cloud system.

$$\tau_{s,d}(t+1) = (1-\rho) \times \tau_{s,d}(t) \quad (13)$$

where ρ is the pheromone evaporation ratio and $0 < \rho \leq 1$.

6 Simulation and results analysis

The implementation of the FACO algorithm is achieved using the JAVA toolkit and the IDE applied for simulation is Eclipse luna. We use also Cloud Analyst to evaluate the performance of the FACO algorithm (Wickremasinghe et al. 2010). This platform is a simulator developed based on the CloudSim framework by extending its main functionalities. This framework allows researchers to analyze the operation of large-scale Internet applications in the cloud and to adjust the optimal configuration for each use case. Cloud Analyst has several features such as flexibility, graphical outputs, repeatability, and low cost. The main concepts of the Cloud Analyst architecture are the cloud application services broker that controls the routing of traffic between user locations and data centers. In addition to the scheduling and

load balancing strategies already configured, Cloud Analyst provides the ability to add a customized scheduling or load balancing policies. The proposed Ant Class and Fuzzy module of the FACO algorithm are developed in JAVA on a Windows platform.

6.1 Optimization of FACO's parameters

One of the key aspects of the efficiency of the ACO algorithm is the definition of the parameters (α, β,...) adapted to the case study. In our case, we applied the Taguchi orthogonal array to identify the optimal combination of parameters among all possible combinations (Seghir and Khababa 2018). This experimental approach has been introduced by Taguchi et al. (2005) as part of quality improvement. The strength of this approach is to evaluate the relationship between inputs as the algorithm's parameters (see Table 3) and outputs which are response time, processing time and total cost. The simulations are achieved in two stages. The Taguchi approach relies on the use of predefined tables of experience which allow us to realize the optimal trials configurations. Among the most well-known Taguchi array, we note the tables $L_8(2^7)$, $L_{16}(2^{15})$, $L_{32}(2^{31})$, $L_{27}(3^{13})$, $L_{36}(2^{11} \times 3^{12})$ and $L_{81}(9^6 \times 3^{16})$. For example, the $L_{81}(9^6 \times 3^{16})$ table handles up to six factors at nine levels and sixteen factors at three levels. As depicted in Table 3,

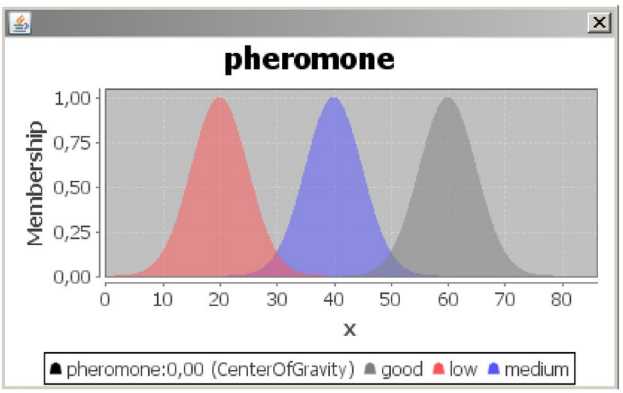

Fig. 4 The membership function of output variable

we evaluate the influence of the five inputs factors (α, β, ρ, Num_{ants}, $Max_{iterations}$) on the output indicators (response time, processing time, total cost). we apply a L_{25} Taguchi array because we have five input factors and for each factor we choose five levels. The identification of the optimal parameters of the FACO algorithm is a complex process due to the infinite number of possible combinations and interactions between parameters (see Fig. 5).

To resolve this problem, we conducted several experiments via the Cloud Analyst simulator by applying the

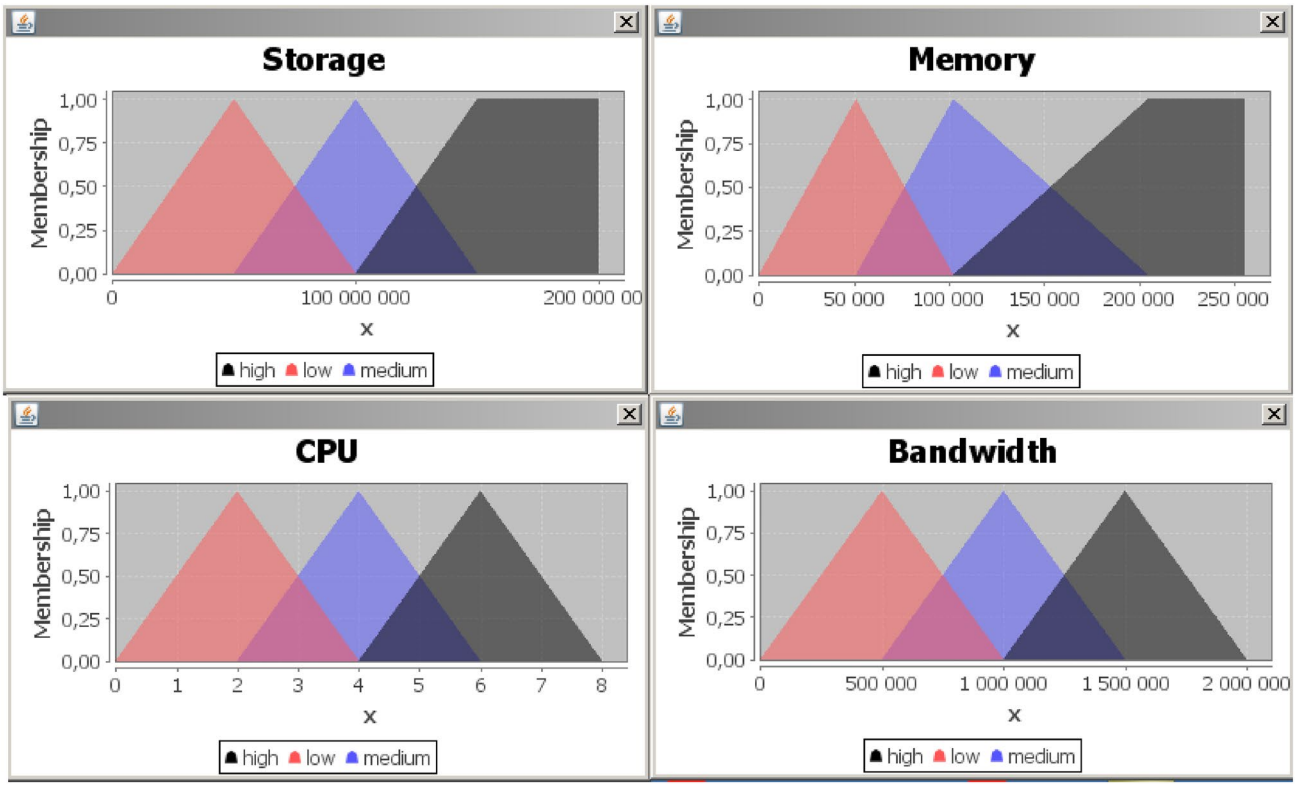

Fig. 3 The membership functions of inputs variables

Table 2 Fuzzy inference rules base for pheromone evaluation

Rule number	Description
Rule 1	IF (Memory IS low) OR (Bandwidth IS low) THEN pheromone IS low
Rule 2	IF (Memory IS low) AND (Storage IS low) THEN pheromone IS low
Rule 3	IF (Memory IS low) AND (CPU IS low) THEN pheromone IS low
Rule 4	IF (Bandwidth IS high) AND (CPU IS high) THEN pheromone IS good
Rule 5	IF (Bandwidth IS high) AND (Memory IS high) THEN pheromone IS good
Rule 6	IF (Bandwidth IS high) AND (Storage IS high) THEN pheromone IS good
Rule 7	IF ((Bandwidth IS low) OR (Storage IS low)) OR (Memory IS low) THEN pheromone IS low
Rule 8	IF (Bandwidth IS high) AND (Memory IS medium) THEN pheromone IS medium
Rule 9	IF ((Memory IS medium) OR (Storage IS medium)) AND (CPU IS medium) THEN pheromone IS medium
Rule 10	IF ((Memory IS high) OR (Storage IS high)) AND (CPU IS high) THEN pheromone IS good

Taguchi DOE L_{25}. This approach contributes (i) to the improvement of the robustness of the product or process and (ii) to the optimization of the number of achieved experiments. For example, we achieve 25 trials instead of $5^5 = 3215$ trials. The functioning of the Taguchi concept is based on the notion of identifying influential factors among all the elements studied, which in our case is to determine the parameter that has the most influence on the performance of the FACO algorithm and to predict the best combination of appropriate values among all possible arrangements. The main phases of the Taguchi method are the determination of the influential factors and the choice of the experimental array which determines the number of experiments to be carried out as well as the value of each influential factor per trial. The Taguchi concept applies the signal-to-noise ratio (SNR) to examine the measurements recorded. The improvements of outputs could be obtained by reducing the function defined in the formulation hereafter which corresponds to the smaller is the best:

$$SNR_i = -10 \log \left(\sum_{u=1}^{N_i} \frac{y_u^2}{N_i} \right) \quad (14)$$

where u: trial number; i: experiment number and N_i: Number of trials for the experiment, y: trials' result.

During the trials defined by the Taguchi array L_{25}, we kept the same Cloud configuration and we modify the FACO parameter for each experiment. The applied scenario is based on the use of one user UB1 who initiates an average of 60 requests per hour of a size of 100 bytes per request, and three areas DC1, DC2 and DC3 regrouping respectively 12 virtual machines. The achieved results are summarized in (see Table 4).

The graphical analysis achieved via Minitab 16 allows us to find out various conclusions such as the optimal configuration per key performance indicator (KPI) to enhance one KPI such as response time or the three KPI which includes response time, processing time and total cost. As shown in

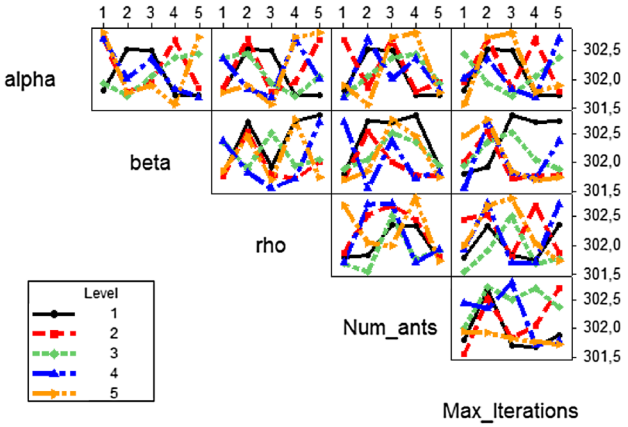

Fig. 5 Interaction Plot for Response time based on Taguchi analysis of the trials results

Table 3 Factors values per level

Level	α	β	ρ	Num_{ants}	$Max_{iterations}$
1	0.5	0.5	0.1	10	20
2	0.6	0.6	0.2	20	30
3	0.7	0.7	0.3	30	50
4	0.8	0.8	0.4	40	60
5	0.9	0.9	0.5	50	100

Fig. 6, 7, 8, the best combination of parameters could be achieved by keeping the highest level of each parameter. In other words, the proposed method allows the identification of the values of the proposed FACO parameters according to the expected objectives. Moreover, Taguchi analysis identifies the factor that has the greatest impact on the key performance indicator (see Tables 5, 6, 7). Thus, the number of ants' parameters is the one that has the most influence on the response time and processing time while the α factor is the one that has the most influence on the value of the total

cost. The results obtained from these analyses allow us to predict the optimal configuration to apply for the optimization of our FACO algorithm. This configuration represents the combination applied during trial number 24 (see Table 4) which corresponds to $\alpha = 0.9$, $\beta = 0.8$, $\rho = 0.3$, $ant_{num} = 20$, and $Max_{iterations} = 20$.

6.2 Performance evaluation of FACO algorithm

As illustrated in Table 8, the second part of the simulation embraces 5 trials that aim to validate the performance of the proposed FACO algorithm. We conducted experiments on CloudSim and CloudAnalyst to evaluate the performance of our proposed algorithm. We examine the performance in terms of response time, processing time and total cost. The results obtained via the FACO algorithm were compared to the results obtained by other algorithms, such as Round Robin, which highlights the performance of the proposed algorithm. Referring to Figures 9, 10, 11, we note that the response time and processing time achieved by the proposed FACO algorithm outperform the other algorithms in the case of a high number of nodes and requests. The values of the cost indicator are relatively close because the setting of FACO parameters ($\alpha, \beta, \rho, Num_{ants}, Max_{Iterations}$) were done in such a way to optimize the response time and the processing time. It remains possible to give an advantage to the cost if this is the objective of the system's administrator. Thus, it has been demonstrated that response time and processing time could be relatively improved by applying a more efficient

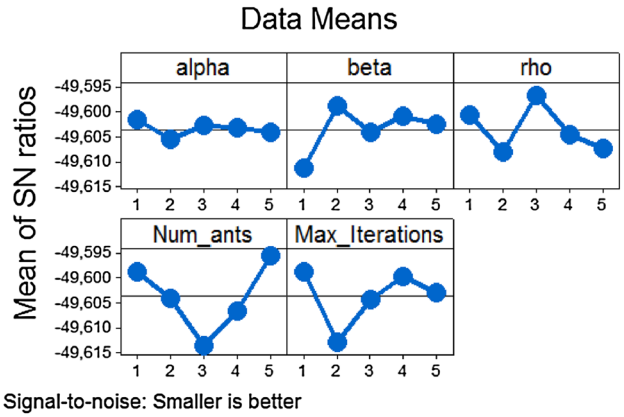

Fig. 6 Main Effects Plot for SN ratios for response time indicator

Table 4 Taguchi L_{25} experiment array and achieved simulation results

N° Trial	α	β	ρ	Num_{ants}	$Max_{Iterations}$	Response time (ms)	Processing time (ms)	Total cost ($)
1	1	1	1	1	1	301.80	1.81	7 712.29
2	1	2	2	2	2	302.53	2.04	7 640.14
3	1	3	3	3	3	302.51	2.03	7 696.48
4	1	4	4	4	4	301.71	1.76	7 679.62
5	1	5	5	5	5	301.72	1.73	7 640.12
6	2	1	2	3	4	302.70	2.30	7 576.19
7	2	2	3	4	5	301.77	1.77	7 604.96
8	2	3	4	5	1	301.94	1.85	7 671.46
9	2	4	5	1	2	302.69	2.23	7 671.83
10	2	5	1	2	3	301.84	1.78	7 602.46
11	3	1	3	5	2	301.92	1.83	7 680.63
12	3	2	4	1	3	301.70	1.75	7 647.95
13	3	3	5	2	4	302.04	1.87	7 656.63
14	3	4	1	3	5	302.37	1.97	7 704.47
15	3	5	2	4	1	302.45	1.95	7 623.14
16	4	1	4	2	5	302.71	2.07	7 549.65
17	4	2	5	3	1	302.00	1.82	7 597.46
18	4	3	1	4	2	302.35	1.89	7 576.47
19	4	4	2	5	3	301.82	1.78	7 597.96
20	4	5	3	1	4	301.68	1.69	7 603.95
21	5	1	5	4	3	302.82	2.06	7 601.66
22	5	2	1	5	4	301.76	1.80	7 712.28
23	5	3	2	1	5	301.88	1.82	7 592.29
24	5	4	3	2	1	301.54	1.66	7 536.95
25	5	5	4	3	2	302.73	2.32	7 552.36

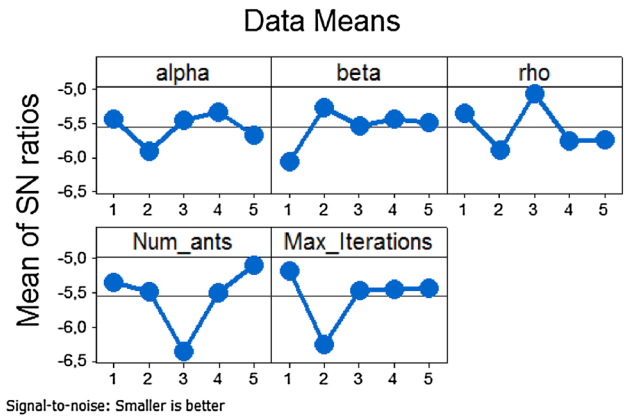

Fig. 7 Main Effects Plot for SN ratios for processing time indicator

Fig. 8 Main Effects Plot for SN ratios for total cost indicator

Table 5 Factors effects ranking for response time indicator

Level	α	β	ρ	Num_{ants}	$Max_{Iterations}$
1	−49.60	−49.61	−49.60	−49.60	−49.60
2	−49.61	−49.60	−49.61	−49.60	−49.61
3	−49.60	−49.60	−49.60	−49.61	−49.60
4	−49.60	−49.60	−49.60	−49.61	−49.60
5	−49.60	−49.60	−49.61	−49.60	−49.60
Delta	0	0.01	0.01	0.02	0.01
Rank	5	3	4	1	2

load balancing strategy without a negative impact on the cost. To date, several studies confirmed the effectiveness of the meta-heuristic algorithm in improving the load balancing in the cloud environment. Table 9 shows the comparative view of the different load balancing algorithms.

Table 6 Factors effects ranking for processing time indicator

Level	α	β	ρ	Num_{ants}	$Max_{Iterations}$
1	−5.433	−6.047	−5.337	−5.348	−5.18
2	−5.902	−5.264	−5.888	−5.472	−6.249
3	−5.447	−5.532	−5.064	−6.357	−5.461
4	−5.323	−5.435	−5.749	−5.495	−5.449
5	−5.659	−5.488	−5.729	−5.093	−5.426
Delta	0.579	0.783	0.824	1.264	1.069
Rank	5	4	3	1	2

Table 7 Factors effects ranking for total cost indicator

Level	α	β	ρ	Num_{ants}	$Max_{Iterations}$
1	−77.70	−77.64	−77.69	−77.67	−77.65
2	−77.65	−77.66	−77.62	−77.61	−77.64
3	−77.69	−77.66	−77.64	−77.64	−77.65
4	−77.60	−77.66	−77.64	−77.64	−77.67
5	−77.61	−77.62	−77.65	−77.69	−77.64
Delta	0.1	0.04	0.06	0.07	0.03
Rank	1	4	3	2	5

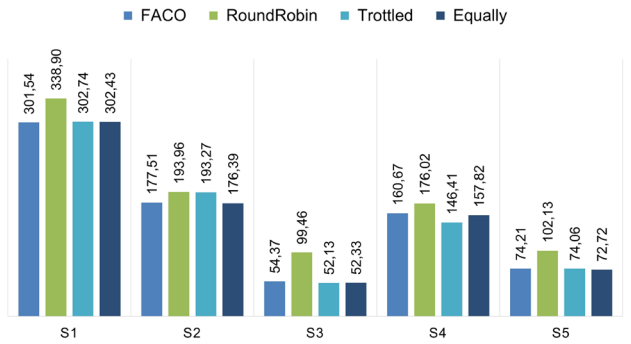

Fig. 9 FACO Response time (ms) compared to others load balancing algorithms

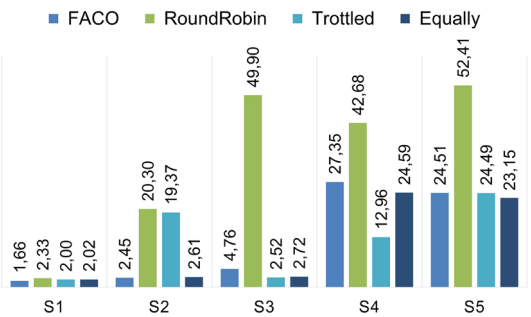

Fig. 10 FACO Processing time (ms) compared to others load balancing algorithms

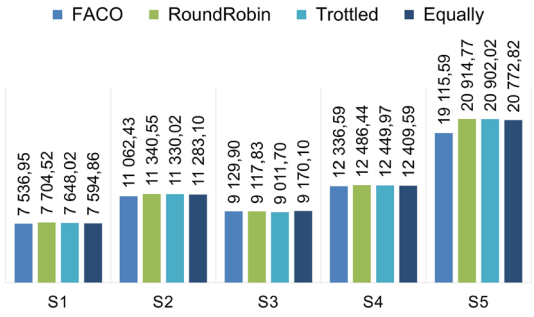

Fig. 11 FACO Total cost ($) compared to others load balancing algorithms

7 Conclusion

This paper proposes a FACO algorithm for virtual machine scheduling with load balancing adapted to cloud computing architecture. This study has identified that the ant colony optimization approach has been used to resolve the load balancing issue due to its capacity to handle a high number of nodes. This paper introduces a hybrid fuzzy ant colony algorithm by including a fuzzy logic module to calculate the pheromone value and the Taguchi concept to optimize the ACO's parameters. The experimental results achieved by the Cloud Analyst simulator confirmed that the FACO algorithm is more appropriate for handling large and distributed networks. The main contributions of the proposed solution are (i) the application of fuzzy logic to calculate the pheromones probability to minimize the computing time (ii) the use of Taguchi experience design for selecting the best ACO's parameters to find the optimal parameters' combination. The applied FACO algorithm includes a procedure of evaporation from the trial of pheromones to avoid earlier convergence towards non-optimal solutions. The achieved simulations within the Cloud Analyst and CloudSim platforms showed that the proposed approach allows improving load balancing in the Cloud architecture while reducing response time by up to 82%, the processing time by up to 90% and total cost by up to 9% depending on the applied scenario. Although the FACO algorithm outperforms the other algorithms, we aim to generalize the pheromone definition to encompasses other distributed systems. Our future work includes an evaluation of the FACO algorithm within a real and multi-cloud computing architecture.

Table 8 FACO performance compared to Round Robin

Scenario	Response time(ms)			Processing time(ms)		
	FACO	Round-Robin	Improvement%	FACO	Round-Robin	Improvement%
S1	301.54	338.90	12.39%	1.66	2.33	28.8%
S2	177.51	193.96	9.27%	2.45	20.30	87.9%
S3	54.37	99.46	82.93%	4.76	49.90	90.5%
S4	160.67	176.02	9.55%	27.35	42.68	35.9%
S5	74.21	102.13	37.62%	24.51	52.41	53.2%

Table 9 Comparative view of load balancing algorithms

Research paper	Applied Algorithms	Strength	Considered feature	General Improvement
(Bui et al. 2017)	ACO, non-cooperative game, and Nash equilibrium	Taking into account customer and service provider	Virtual machines provision solution	A moderate improvement compared to conventional algorithms
(Yang and Zhuang 2010)	ACO	Minor premature convergence probability	Resolving mobile agent routing	Improvement of time to find best solution by up to 32%
(Saffar et al. 2011)	Bayesian networks based on ant colony optimization	A hybrid performed algorithm including Bayesian network models, ACO and simulated annealing	Power loss and load balancing index	Load balancing index improvement up to 41.51% and loss reduction up to 59.84%
FACO	ACO and Fuzzy Logic	Improved ACO's parameters, simpler and faster to implement	Response time, processing time and load balancing index	Improvement of response time by up to 82% and processing time by up to 90%

Acknowledgements The authors would like to thank the reviewers for their valuable reviews and constructive comments, which have contributed to enhancing the quality of this paper.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Arunarani A, Manjula D, Sugumaran V (2019) Task scheduling techniques in cloud computing: a literature survey. Future Gener Comput Syst 91:407–415

Boveiri HR, Khayami R, Elhoseny M, Gunasekaran M (2019) An efficient Swarm-Intelligence approach for task scheduling in cloud-based internet of things applications. J Ambient Intell Human Comput 10(9):3469–3479

Bui KT, Pham TV, Tran HC (2017) A load balancing game approach for VM provision cloud computing based on ant colony optimization. In: Cong Vinh P, Tuan Anh L, Loan NTT, Vongdoiwang Siricharoen W (eds) Context-aware systems and applications, vol 193. Springer International Publishing, Cham, pp 52–63

Cingolani P, Alcalá-Fdez J (2013) jFuzzyLogic: a Java library to design fuzzy logic controllers according to the standard for fuzzy control programming. Int J Comput Intell Syst 6(sup1):61–75

Dorigo M, Birattari M, Stützle T (2006) Ant Colony Optimization Artificial Ants as a Computational Intelligence Technique. IRIDIA—TECHNICAL REPORT SERIES TR/IRIDIA/2006-023

Gabi D, Ismail AS, Zainal A, Zakaria Z, Abraham A (2018) Orthogonal Taguchi-based cat algorithm for solving task scheduling problem in cloud computing. Neural Comput Appl 30(6):1845–1863

Gao R, Wu J (2015) Dynamic load balancing strategy for cloud computing with ant colony optimization. Future Internet 7(4):465–483

Gao Y, Guan H, Qi Z, Hou Y, Liu L (2013) A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. J Comput Syst Sci 79(8):1230–1242

Gendreau M, Potvin J-Y (eds) (2010) Handbook of metaheuristics, volume 146 of International series in operations research & management science. Springer, USA

Gonzalez-Pardo A, Jung JJ, Camacho D (2017) ACO-based clustering for ego network analysis. Future Gener Comput Syst 66:160–170

Kahraman C, Pardalos PM, Du D-Z (eds) (2008) Fuzzy multi-criteria decision making, volume 16 of Springer optimization and its applications. Springer USA

Li Y, Tong S (2017) Adaptive fuzzy output-feedback stabilization control for a class of switched nonstrict-feedback nonlinear systems. IEEE Trans Cybern 47(4):1007–1016

Mamdani M, Assilian S (1999) An experiment in linguistic synthesis with a fuzzy logic controller. Int J Hum Comput Stud 51(2):135–147

Masulli F, Pasi G, Yager R, Hutchison D, Kanade T, Kittler J, Kleinberg JM, Mattern F, Mitchell JC, Naor M, Nierstrasz O, Pandu Rangan C, Steffen B, Sudan M, Terzopoulos D, Tygar D, Vardi MY, Weikum G (eds) (2013) Fuzzy logic and applications, vol 8256. Lecture notes in computer science. Springer International Publishing, Cham

Mijumbi R, Serrat J, Gorricho J-L, Bouten N, De Turck F, Boutaba R (2016) Network function virtualization: state-of-the-art and research challenges. IEEE Commun Surv Tutor 18(1):236–262

Mikaeeli Mamaghani S, Jabraeil Jamali MA (2019) A load-balanced congestion-aware routing algorithm based on time interval in wireless network-on-chip. J Ambient Intell Human Comput 10(7):2869–2882

Ragmani A, Elomri A, Abghour N, Moussaid K, Rida M (2019) An improved hybrid fuzzy-ant colony algorithm applied to load balancing in cloud computing environment. Proc Comput Sci 151:519–526

Ragmani A, Omri AE, Abghour N, Moussaid K, Rida M (2017) An efficient load balancing strategy based on mapreduce for public cloud. In: ICC 2017: Second international conference on internet of things and cloud computing, ACM Press, Cambridge, pp 1–10

Routaib H, Badidi E, Elmachkour M, Sabir E, ElKoutbi M (2014) Modeling and evaluating a cloudlet-based architecture for Mobile Cloud Computing. In 2014 9th international conference on intelligent systems: theories and applications (SITA-14), IEEE, Rabat, Morocco, pp 1–7

Saffar A, Hooshmand R, Khodabakhshian A (2011) A new fuzzy optimal reconfiguration of distribution systems for loss reduction and load balancing using ant colony search-based algorithm. Applied soft computing 11(5):4021–4028

Seghir F, Khababa A (2018) A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition. J Intell Manuf 29(8):1773–1792

Shetty SM, Shetty S (2019) Analysis of load balancing in cloud data centers. J Ambient Intell Human Comput. https://doi.org/10.1007/s12652-018-1106-7

Sosinsky BA, (2011) Cloud computing bible. Wiley, Indianapolis [John Wiley, distributor]

Taguchi, G., Chowdhury, S., Wu, Y., Taguchi, S., and Yano, H. (2005) Taguchi's quality engineering handbook. Wiley/ASI Consulting Group, Hoboken/Livonia

Tamilvizhi T, Parvathavarthini B (2019) A novel method for adaptive fault tolerance during load balancing in cloud computing. Cluster Comput 22(5):10425–10438

Van Broekhoven E, De Baets B (2008) Monotone Mamdani–Assilian models under mean of maxima defuzzification. Fuzzy Sets Syst 159(21):2819–2844

Wickremasinghe B, Calheiros RN, Buyya R (2010) CloudAnalyst: a CloudSim-based visual modeller for analysing cloud computing environments and applications. In: AINA '10 Proceedings of the 2010 24th IEEE international conference on advanced information networking and applications, IEEE, Perth, pp 446–452

Xu M, Tian W, Buyya R (2017) A survey on load balancing algorithms for virtual machines placement in cloud computing: a survey on load balancing algorithms for VM placement in cloud computing. Concurrency Comput Pract Experience 29(12):e4123

Yang J, Zhuang Y (2010) An improved ant colony optimization algorithm for solving a complex combinatorial optimization problem. Appl Soft Comput 10(2):653–660

Yu L, Chen L, Cai Z, Shen H, Liang Y, Pan Y (2016) Stochastic load balancing for virtual resource management in datacenters. IEEE Trans Cloud Comput. https://doi.org/10.1109/TCC.2016.2525984

Zahoor S, Javaid S, Javaid N, Ashraf M, Ishmanov F, Afzal M (2018) Cloud-fog-based smart grid model for efficient resource management. Sustainability 10(6):2079

Zhang J, Wang X, Huang H, Chen S (2017) Clustering based virtual machines placement in distributed cloud computing. Future Gener Comput Syst 66:1–10