IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 30, 2022, accepted April 21, 2022, date of publication April 25, 2022, date of current version May 2, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3170103

An Improved Ant Colony Algorithm for Solving a Virtual Machine Placement Problem in a Cloud Computing Environment

NAWAF ALHARBE[1], MOHAMED ALI RAKROUKI[1,2,3], AND ABEER ALJOHANI[1]

[1]Applied College, Taibah University, Medina 42353, Saudi Arabia
[2]Ecole Supérieure des Sciences Économiques et Commerciales de Tunis, University of Tunis, Tunis 1089, Tunisia
[3]Business Analytics and Decision Making Laboratory (BADEM), Tunis Business School, University of Tunis, Tunis 2059, Tunisia

Corresponding author: Nawaf Alharbe (nrharbe@taibahu.edu.sa)

ABSTRACT The world is currently witnessing many successive changes in several fields, especially in the field of healthcare which forced countries to impose health policies to provide good services, which drew health sector strategies focusing on the digital transformation aspect. E-government healthcare applications are used by dozens of millions of users. This high usage of applications generates a huge network traffic and needs reliable cloud computing platforms and efficient virtual machine placement models. In this context, we proposed an improved Ant Colony algorithm to solve a virtual machine placement problem in a cloud computing environment in order to minimize the total network traffic and the maximum link utilization. The experimental results demonstrate the effectiveness and efficiency of our proposed algorithm.

INDEX TERMS Ant colony optimization, cloud computing, placement, scheduling, virtualization.

I. INTRODUCTION

Cloud computing is an emerging technology in the information field in recent years, and the key technology virtual machine placement has attracted much attention. The research focuses on improving application performance, resource load balancing, and saving data center energy consumption. Some achievements have also been made, but there are still some problems or defects waiting to be resolved. At present, another impact of the COVID-19 pandemic, the scale of cloud computing data centers is expanding rapidly. This is due on the one hand to distance working/learning/entertainment, and on the other hand to the implementation of healthcare applications imposed by the countries [1]–[3]. These latter applications are required to provide many healthcare related services (mainly they are used to provide digital proof that a person has been vaccinated against COVID-19, has recovered from COVID-19 or has a test result). Therefore, this paper focuses on the problem of virtual machine placement, and alleviates this problem in a cloud computing environment through efficient placement algorithms in order to minimize the total network traffic and the maximum link utilization.

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

The remainder of this paper is organized as follows. The Section II is an overview of cloud computing, virtualization technology, and the basic model of virtual machine placement. In Section III, some important related works to the virtual machine placement problem are presented. The description of the problem under consideration is presented in Section IV. In Section V, we propose an improved Ant Colony Optimization algorithm to our problem. The experimental results are presented in Section VI. Finally, Section VII summarizes this research work.

II. CLOUD COMPUTING AND VIRTUALIZATION

Cloud computing is the development of distributed computing, parallel computing, and grid computing. Through virtualization technology, the infrastructure is logically formed into a virtual resource pool. Computing resources, storage resources, and network resources are distributed on a large number of distributed computers. Users no longer need to deploy server clusters locally, and can rent cloud resources on demand through the Internet.

Technically, virtualization is a resource management technology, located on the upper layer of hardware resources. Through this technology, the underlying hardware resources such as memory, CPU, network, etc. can be abstracted and

separated, so that users can use it in a better way. These resources, the virtualized resources are not restricted by the deployment method, geographic area or physical configuration of the existing resources. The essence of this technology is to view heterogeneous or homogeneous physical resources from a logical perspective, and to utilize resources from a logical perspective rather than a physical perspective. The original intention of the development of virtualization technology is to solve some of the problems faced by traditional IT infrastructure, such as low infrastructure utilization, rising costs, failover and insufficient disaster protection, etc. Currently, this technology is applied to servers, networks, and storage.

The mapping of virtual machines to physical machines is related to cloud data center system performance, resource utilization, electricity consumption, etc. Effective mapping not only brings business benefits to cloud providers, but also improves applications performance on it and improves service quality. The existing virtual machine placement strategies mainly focuses on server resources, such as CPU, memory, etc. Dong *et al.* [4] proposed an effective placement of virtual machines in order to reduce the number of active physical servers in the data center. Li *et al.* [5] considered the balanced utilization of multi-dimensional resources (CPU, memory) on the physical server to avoid resource waste caused by the bucket effect and improve resource utilization.

Researchers often ignore the impact of cloud data center network performance on the system. When the network delay is large, the number of tasks processed by the virtual machine per unit time is reduced, and the quality of service is reduced. Large-scale cloud data centers generally have thousands of servers, which are connected layer by layer through switches to form a huge network system. Al-Fares *et al.* [6] presented the Tree network topology (see Figure 1) as an example and pointed out that the servers connected to the same switch have the largest network bandwidth, such as server A and server B; if the communication between the servers has to go through for switches at the aggregation layer, the network bandwidth will generally be reduced by 1/4 to 1/8 of that of the rack switches, such as server A and server D; if it passes through the core layer switch, the network bandwidth will drop more and the delay will increase, such as server A with server E. It can be seen that the more switches that pass through the network communication process, the lower the communication performance.

In a cloud data center, users' web applications or other applications such as high-performance distributed tasks will be deployed on multiple virtual machines. Due to the inherent dependencies between multiple components in the application, frequent network communication occurs. Through the above analysis, it can be seen that this group of associated virtual machines should be deployed on the same physical machine as much as possible or the communication between them should pass through fewer switches to reduce the communication delay between virtual machines and improve service quality.

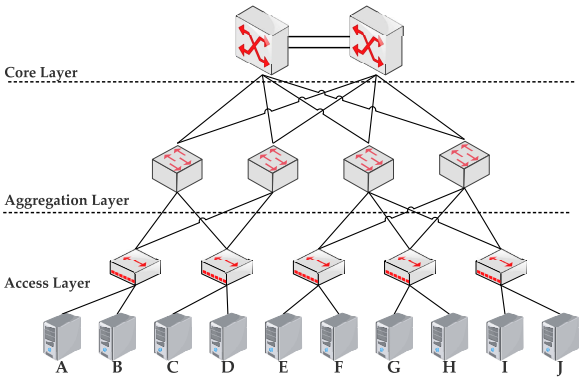


FIGURE 1. Tree network topology.

The influence of network factors on system performance has begun to be widely considered in the literature. Pires *et al.* [7] have proposed a multi-objective optimized virtual machine placement strategy. While considering energy saving and the economic benefits of cloud providers, he proposed minimizing data center network traffic by placing large-traffic virtual machines in a centralized manner. Wen *et al.* [8] proposed an efficient online virtual machine placement algorithm to solve the problem of network congestion, and optimized the total data center network traffic through virtual machine placement to localize the distribution of traffic. Generally speaking, reducing the total network traffic will reduce the link congestion rate. However, the traffic localization reduces the risk of link congestion at the core layer, but it increases the risk of congestion at access layer and aggregation layer.

III. RELATED WORK

Regarding the literature on virtual machine placement, the most common studied performance criteria is the optimization of energy consumption and network performance. Pires and Barán [9] have considered SLA (Service-Level Agreement) as constraint to minimize energy consumption, minimize network traffic, and maximize economic benefits. Fang *et al.* [10] proposed a power-efficient solution by aggregating virtual machines on to fewer physical machines to make the communication distance between virtual machines closer. They considered the Fat-Tree network topology and the OpenFlow network protocol. In this way, the network traffic will pass through the core switch as little as possible. However, minimizing electricity consumption costs and work delays, generate a significant increase in local link utilization and cause network congestion.

Luo *et al.* [11] studied the problem of reduced data center resource utilization and increased network latency due to some virtual machines being shut down or stopped. They proposed a network-aware virtual machine placement strategy to improve the overall network communication performance and user service quality through periodic virtual machine migration. However, the virtual machine migration process requires a large network bandwidth, and the

resources overheads on the physical machine where it is located increase, which will inevitably reduce the performance of other virtual machines on it.

Montgomery and Randall [12] proposed using virtual machine migration technology to reduce the total network traffic. They showed that the localization of network traffic can reduce the maximum link utilization rate and reduce network congestion. Therefore, they believe that optimizing the total network traffic is consistent with the goal of minimizing the maximum link utilization. Jiang *et al.* [13] reviewed and evaluated two methods of joint virtual machine placement and routing selection to optimize data center network performance and network resource utilization. The network traffic is localized through the virtual machine placement strategy, and the network traffic of all virtual machines does not spread across the entire link of the data center. However, their methods are not suitable for a large number of virtual machines requested by users of large data centers.

Kalra and Singh [14] provided an extensive survey of several metaheuristic algorithms, in cloud and grid environments. A comparative analysis of these algorithms has been presented under different aspects. The studied metaheuristics have been mainly compared in term of encoding scheme, initial population generation, optimization criteria, and nature of scheduled tasks.

Liu *et al.* [15] proposed an ant colony-based algorithm combined with some local search approaches for minimizing the number of active physical servers. They showed that the proposed algorithm can group candidate virtual machines together, and then efficiently decreases the number of active utilized servers. Alharbi *et al.* [16] considered the problem of virtual machine placement problem with the objective of minimizing energy consumption. They proposed an ant colony-based approach by combining a First-Fit-Decreasing (FFD) algorithm with two known ant colony systems, and showed that the obtained method provide better solutions. Supreeth and Patil [17] have presented an extensive literature review on virtual machine scheduling strategies. They discussed literature findings and indicated some research concerns in scheduling tasks and virtual machine placement techniques. They also discussed virtual machine placement methods, as well as their advantages and specifications. Wei *et al.* [18] used an adaptive parameter setting technique for improving an ant colony optimization algorithm, with the objective of minimizing energy consumption and communication costs. They showed that the proposed algorithm outperforms some existing algorithms and the runtime has been significantly reduced under various traffic patterns and setups.

Recently, Abohamama and Hamouda [19] proposed a hybrid virtual machine placement algorithm in order to improve the energy consumption rate of cloud data centers through minimizing the number of active servers that host virtual machines. They showed that the proposed genetic algorithm provides promising results in term of energy saving and resource wastage compared to other approaches. A parallel ant colony algorithm, namely PACO-VMP, has been proposed by Peake *et al.* [20]. The proposed method has taken advantages of parallelization and modern processor technologies in order to generate high quality solutions, and the effectiveness of the proposed algorithm has been showed by comparison to well-known nature-inspired algorithms. Raghmani *et al.* [21] considered the problem of load balancing in a high-performance cloud computing environment. They proposed a hybrid fuzzy ant colony algorithm in order to distribute the load efficiently on servers, and showed the effectiveness of the proposed hybrid algorithm, especially in the case of a high number of nodes. Ajmal *et al.* [22] proposed a combined an ant colony algorithm with a genetic algorithm in order to efficiently scheduling tasks in a cloud data center. By splitting tasks into groups and identifying loaded virtual machines, they showed that the proposed approach considerably reduces solution space. The experiments showed that the computation time and the total data center costs have been reduced by 64% and 11%, respectively.

From the literature review, we can see that the objectives considered in the virtual machine placement problem are diverse and complicated, and various approaches have been proposed to solve this challenging problem. In the following is a summary of our paper's main contribution:

1) Optimizing the total network traffic and the maximum link utilization based on multi-constraints.
2) An improved Ant Colony Optimization (ACO) algorithm by integrating a Simulated Annealing (SA) procedure in order to provide more satisfactory, especially in term of quality of solutions.
3) Analyze the efficiency of the proposed algorithm on various network topologies.

IV. PROBLEM DESCRIPTION

In this paper, we investigate the problem of virtual machine placement. We have to assign a group of virtual machines to physical nodes with the objective of optimizing network performance under the premise of meeting the resource constraints of the physical nodes. More precisely, the problem can be stated as follows.

We are given a cloud data center sharing M physical machines, denoted as $P = \{P_1, P_2, \ldots, P_i, \ldots, P_M\}$. Each physical machine P_i has d available resources (such as CPU, memory, bandwidth, storage, etc.). Each resource j has a capacity $H_{i,j}$, where $j \leq d$. The users applied for a group of N virtual machines $V = \{V_1, V_2, \ldots, V_i, \ldots, V_N\}$. The required resource j of any virtual machine V_i is denoted as $R_{i,j}$.

The network traffic of each group of virtual machine pairs is estimated according to user needs, and represented by an undirected graph $G = (V, E)$ where V is the set of virtual machines requested by the user, $E = \{(V_i, V_j) : V_i, V_i \in V\}$ is the set of link (Communication traffic) between the virtual machines V_i and V_j. The weight of edges is denoted by T_{ij}. Figure 2 shows a schematic diagram of a group of associated virtual machines.

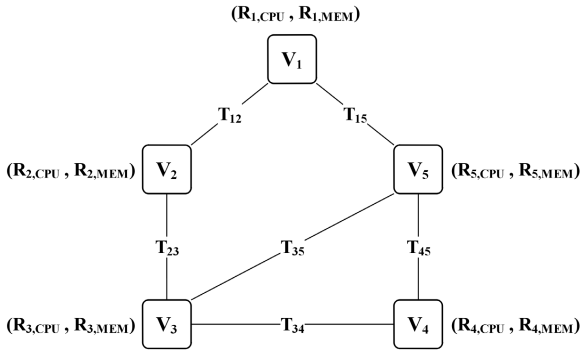


FIGURE 2. Group of associated virtual machines.

TABLE 1. Symbols and their meanings.

Symbol	Meaning
M	Number of physical machines
N	Number of virtual machines requested by the user
P	Physical machine set $P = \{P_1, P_2, ..., P_i, ..., P_M\}$
$H_{i,j}$	The capacity of resources of type j on physical machine P_i
V	A set of virtual machines to be deployed $V = \{V_1, V_2, ..., V_i, ..., V_N\}$
$R_{i,j}$	The number of resources of type j required by virtual machine V_i
π_j^i	A binary variable, 1 means V_j is deployed on P_i, 0 otherwise
T_{ij}	Communication traffic between virtual machines V_i and V_j
C_{ij}	Network bandwidth between physical machines P_i and P_j
Y_i	A binary variable, 1 means that P_i is in running state, 0 otherwise
A	Network traffic matrix of virtual machines
B	Number of hops between physical machines in the data center

The data center network topology is represented by an undirected graph $D = (P, E)$, where P is the set of all physical machines, $E = \{(P_i, P_j) : P_i, P_i \in P\}$ is a link (communication traffic) between the physical machines P_i and P_J. The weight of edges C_{ij} indicates the bandwidth of the link.

The main goal is to deploy a group of associated virtual machines on a physical machine in order to minimize the total network traffic of the data center while minimizing the maximum link utilization under the premise of meeting the resource constraints of the physical machines.

All symbols and their meaning are provided in Table 1.

A. RESOURCE CONSTRAINTS

The basis of the virtual machine placement problem is to select the corresponding physical machine to carry the virtual machine requested by the user. Therefore, the first condition is that the user's most basic resource request, that is, the physical server resource request, must be satisfied. Let π_{ij} a binary variable, $\pi_{ij} = 1$ if a virtual machine V_i is deployed on a physical machine P_j, $\pi_{ij} = 0$ on the contrary. Y_i represents the state of a physical machine P_i, $Y_i = 1$ indicates that the physical machine is running, and $Y_i = 0$ indicates that the physical machine is in the shutdown state. Hence, the feasible decision space for the virtual machine placement problem is expressed as follows:

$$\pi = \{\pi_j^i; \pi_m^i \in \{0, 1\}, \sum_{m=1}^{M} \pi_m^i = 1\}; \quad \forall i \in \{1, 2, \ldots, N\}$$
$$\sum_{i=1}^{N} \pi_j^i R_{i,x} \le Y_j H_{j,x}; \quad \forall x \in \{1, 2, \ldots, d\} \quad (1)$$

The first equation means that for any request, the virtual machine must be deployed on the corresponding physical machine, and the latter equation is the resources constraints of the physical machine. All resources requirements of all virtual machines deployed on the same physical machine must be less than the resources capacities of the physical machine.

B. TOTAL NETWORK TRAFFIC

The more network links the network traffic flows through, the longer the network delay, and delay is one of the most important factors to influence the quality of service. Therefore, by restricting the selection of unnecessary long paths, the total network traffic and total transmission time can be reduced. The total network traffic is calculated based on the estimated network traffic and communication delay between virtual machine pairs. The network traffic matrix $A = (a_{ij})_{N\times N}$ for a given virtual machine group can be represented by (2), where a_{ij} represents a network traffic between two virtual machines V_i and V_j. The two elements of communication delay are propagation delay and queue delay. Queue delay is the delay experienced by packets while waiting for transmission on the link, which can be reflected by the link utilization rate, so we only consider here the propagation time. The propagation delay can be measured by the number of route hops passed in the propagation process. The propagation delay matrix $B = (b_{ln})_{M\times M}$ can be represented by (3), where b_{ln} represents the number of switches or routers passing through between two physical machines Pl and Pn, the greater the number, the greater the communication delay. The total network traffic objective function is expressed by (4), where π_{ij} represents a virtual machine V_i, deployed on a physical machine V_j.

$$A = \begin{bmatrix} a_{11} & \ldots & a_{1N} \\ \vdots & \vdots & \vdots \\ a_{N1} & \ldots & a_{NN} \end{bmatrix} \quad (2)$$

$$B = \begin{bmatrix} b_{11} & \ldots & b_{1M} \\ \vdots & \vdots & \vdots \\ b_{M1} & \ldots & b_{MM} \end{bmatrix} \quad (3)$$

The optimization goal for this criterion is,

$$\text{minimize } W_{traffic} = \sum_{i,j=1}^{N} a_{ij} b_{\pi_j^i} \quad (4)$$

C. NETWORK LINK UTILIZATION

The maximum link utilization rate is used to characterize the degree of network congestion. The network is regarded as the integration of all links, and network traffic distribution is balanced to reduce network congestion. The link utilization rate can be expressed as follows:

$$L_{utilize} = \frac{\sum_{i,j=1}^{N} x_{s,t}^{i,j}}{C_{s,t}} \quad (5)$$

where $C_{s,t}$ is the maximum bandwidth that the link (s, t) can handle, $x_{s,t}^{i,j}$ is the time of the network traffic allocated to the communication between virtual machine V_i, V_j via link (s, t). The optimization goal for this criterion is:

$$\text{minimize } \max(L_{utilize}) \tag{6}$$

D. GLOBAL OBJECTIVE FUNCTION

Our aim is to optimize the network performance of virtual machine groups through virtual machine placement. First, we minimize the total network traffic of the data center, and minimize the maximum link utilization under the condition that it has little effect on the total network traffic. The objective function is as follows:

$$\begin{aligned} &\text{minimize } f = aW_{traffic} + L_{utilize} \\ &\text{s.t.} \begin{cases} \sum_{i,j=1}^{N} x_{s,t}^{i,j} - x_{t,s}^{i,j} \\ \sum_{i,j=1}^{N} x_{s,t}^{i,j} \le C_{s,t} \\ x_{s,t}^{i,j} \ge 0 \end{cases} \end{aligned} \tag{7}$$

where a is a constant.

V. ANT COLONY OPTIMIZATION ALGORITHM

Metaheuristic algorithms are generally used to solve combinatorial optimization problems, such as Ant Colony Optimization algorithm (ACO), Genetic Algorithm (GA), Simulated Annealing (SA), Tabu Search (TS), etc. Compared with traditional methods, these algorithms can still search for good solutions and even optimal solutions even when the instance scale is large. These metaheuristics has been successfully used to solve different combinatorial optimization problems (Traveling Salesman Problem (TSP) [23], Job-shop scheduling problem [24], assignment problem [25], etc.).

A. ACO ALGORITHM DESCRIPTION

ACO algorithm is a probabilistic algorithm, proposed by Dorigo [26] for finding an optimized path in a graph. According to a long-term study by imitation biologists, although ant colonies have no vision, they can trigger from the source to find the shortest path to a food source. The reason is that the ants themselves release a secretion-pheromone. When an ant arrives at a certain intersection, it will determine which path to choose based on the pheromone concentration in each direction. The higher the pheromone concentration, the greater the probability of being selected. At the same time, the ant will leave the pheromone on the path passed by. The pheromone concentration is inversely proportional to the path length. Also, some ants will randomly choose a path to move forward at intersections without pheromone, thus forming a positive feedback mechanism. After the ants find the food source, they will return with the food and go back and forth between the source point and the food source. This way, the ants on a short path will go back and forth more times, leaving a naturally high concentration of pheromone, which makes more and more ants choose this path. In ACO algorithm, each ant is an independent individual (solution), looking for a path independently, and all ants exchange information through pheromone, so the ant colony is a highly self-organizing system.

The following will analyze the advantages of the basic ant colony algorithm ACO, clarify the reasons for choosing this algorithm in this paper, analyze its defects as a basis for subsequent improvements [27]–[29].

(1) Advantages of ACO algorithm:

(a) Strong robustness: Compared to other metaheuristic algorithms, it has strong robustness and good performance when it is well set, and its basic algorithm model can be applied to problems with easy modifications.

(b) Parallelism: Using a positive feedback mechanism, all ants search for the solution of the problem independently and in parallel, which is consistent with the characteristics of the cloud computing environment.

(2) Defects of ACO algorithm:

(a) The convergence speed is slow: After all the ants in the algorithm complete a round of searching, they update the pheromone of all the paths passed by, resulting in the pheromone difference of each path is not obvious.

(b) Easy to stagnate: After the search has progressed to a certain level, all individual solutions are consistent, and there is no way to proceed with the next trial search to jump out of the local optimum.

B. MATHEMATICAL MODEL

In order to clearly explain the mathematical model of the ACO algorithm, we will explain it based on the classical symmetry traveling salesman problem (TSP). The general formulation of the TSP problem is: a traveling salesman departure from the city C_1, needs to go to the cities $\{C_2, C_3, \ldots, C_n\}$ to sell goods and finally return to the city C_1, where the distance between any two cities is known, how should the traveling salesman choose the route to make the total route the shortest. It has been proved that the TSP problem is an $\mathcal{NP}$-hard problem. The road map composed of n cities is abstracted as a graph $G = (V, E)$, where V is the vertex set, which represents the set of all cities; $E = (V_i, V_j) : V_i, V_j \in V$ is the edge set, the weight of the edge d_{ij} represents the distance between the cities $\{C_i$ and $C_j\}$. The objective of the TSP problem is to find a Hamiltonian loop with the shortest length from the graph G. Let x_{ij} a binary variable such as:

$$x_{ij} \begin{cases} 1; & \text{if } (V_i, V_j) \text{ is on the Hamiltonian loop} \\ 0; & \text{Otherwise} \end{cases} \tag{8}$$

Then the classical symmetric TSP problem can be expressed by the following mathematical model:

$$\text{minimize } Z = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \tag{9}$$

$$\text{s.t.} \begin{cases} \sum_{j=1}^{n} x_{ij} = 1; \forall i \in V \\ \sum_{i=1}^{n} x_{ij} = 1; \forall j \in V \\ \sum_{i\in S}\sum_{j\in S} x_{ij} \le |S| - 1; \forall S \subset V; 2 \le |S| \le n-1 \\ x_{i,j} \in \{0, 1\} \end{cases} \tag{10}$$

Constraints 1 and 2 indicate that for each vertex in the graph, only one edge is allowed in and one edge is out; Constraint 3 indicates that no sub-loop solutions are allowed to be generated. The ACO algorithm solves the TSP problem as follows. Set the number of ants in the system as K, and assign the same initial value to each path at the beginning as the initial pheromone τ_0. Then, all ants independently look for a Hamiltonian loop with the shortest length in graph G.

1) TRANSITION PROBABILITY CRITERION

When the ant reaches a certain node, it decides the next search node according to the size of the pheromone on the path. The transition probability is calculated according to the size of the pheromone and heuristic information. In order to facilitate the memory of the nodes searched by ants, a tabu list $tabu_k$ is added for each ant k and the nodes searched by each ant k are added to its tabu list $tabu_k$. The subsequent search should be for the nodes that are not in $tabu_k$. The transition probability is calculated as follows.

$$P_{ij}^{k}(t) = \begin{cases} \dfrac{(\tau_{ij}(t))^{\alpha}(\eta_{ij}(t))^{\beta}}{\sum_{s\subset allow_k}(\tau_{is}(t))^{\alpha}(\eta_{is}(t))^{\beta}}; & \text{if } j \in allow_k \\ 0; & \text{Otherwise} \end{cases} \tag{11}$$

$$\eta_{ij}(t) = \frac{1}{d_{ij}} \tag{12}$$

where the set $allow_k = V - tabu_k$; $\tau_{ij}(t)$ is the pheromone size between V_i and V_j; $\eta_{ij}(t)$ is the heuristic information that reflects the ants at the node V_i. The degree of expectation is calculated according to (11); d_{ij} the smaller the ants arrive to V_i from V_j, the higher the expectation level; α reflects the importance of pheromone in the process of ant optimization. The larger the value, the more likely the ant is to choose the path taken by other ants, and the stronger the cooperation; β is a heuristic information factor, which reflects the degree of influence the ants have on the heuristic information in the search process [12].

2) LOCAL UPDATE RULE

In order to reduce the repetitive selection of the path that has been previously selected and the algorithm falls into a local optimum, a negative feedback mechanism is introduced, and the pheromone on the path will be correspondingly reduced over time.

$$\tau_{ij}(t+1) = (1-\xi)\tau_{ij}(t) + \xi\tau_0; \tag{13}$$

$$\tau_0 = \frac{1}{nd_{ij}^{min}} \tag{14}$$

where τ_0 is the initial pheromone, $\xi \in [0, 1]$ is a decay coefficient, and d_{ij}^{min} is the value with the smallest weight among all edges.

3) GLOBAL UPDATE RULE

The ant that has obtained the global best solution performs a global pheromone update to the path of the solution, and the update rule is as follows.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}(t) \tag{15}$$

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{K}\Delta\tau_{ij}^{k}(t) \tag{16}$$

where $\rho \in [0, 1]$ is a volatile factor, and $\Delta\tau_{ij}(t)$ is the total pheromone increase through (V_i, V_j).

C. THE PROPOSED HACOS ALGORITHM

In this section, we describe our hybrid ACO algorithm (denoted HACOS). Figure 3 is the overall flow chart of our proposed algorithm. The hybridization and improvement points are reflected in the dashed rectangle in Figure 3. The pheromone update rule is improved according to the simulated annealing mechanism. Instead of updating the pheromone of all the paths searched by the ants, it updates part of the paths to speed up the convergence of the algorithm, and locally optimize the current round of best solutions generated after each iteration, so that the algorithm has the ability to jump out of the local optimum. The pheromone in our algorithm is defined as the support rate for deploying virtual machines on physical nodes, and in order to avoid premature convergence, the pheromone is volatile. The heuristic information is defined as the degree of expectation of deploying virtual machines on physical nodes. Larger, the greater the probability of successful deployment.

Algorithm quality mainly depends on the definition of pheromone and the determination of pheromone update rules. Pheromones for the virtual machine placement problem are generally divided into two categories: one is established between the virtual machine and the physical node, known as the VM-Host, and the other VM-VM [30] is established between virtual machines. In our algorithm, we have chosen the first method. Assume that the number of physical nodes is M, the number of virtual machines is N, and the number of ants is K. All ants work independently. Initially, they are randomly assigned to each virtual machine node, and each

has a queue of virtual machines to be deployed. The specific process of our proposed algorithm is presented in Figure 3.

1) PHEROMONE DEFINITION

Our optimization goal is to minimize the total traffic of the data center network and the maximum link utilization. Therefore, the pheromone will consider these two factors at the same time, and the degree of influence of the two factors on the optimization problem is uncertain, so the ant is proceeding. A random number is generated when the path is transferred, which indicates the importance of the total network traffic and the maximum link utilization. Since there was no pheromone generated during the initial test, all the ants randomly choose a deployment route.

2) HEURISTIC INFORMATION DEFINITION

Heuristic information η_{ij} indicates the degree of expectation of a virtual machine V_i can be assigned to a physical machine P_j. We minimize the data center network traffic, and minimize the maximum link utilization when it has little impact on the network traffic. Therefore, when an ant is preparing to deploy a virtual machine V_i, it should give priority to selecting a physical node P_i with the minimum network traffic. Therefore, heuristic information is defined as the product of network traffic and communication distance as follows:

$$\eta_{ij} = \sum_{n=1}^{N} a_{in} b_{j\pi(n)} \tag{17}$$

where N is the number of virtual machines, a_{in} is the network traffic between a virtual machine V_i and a virtual machine V_n, and $b_{j\pi(n)}$ is the number of route hops between two physical machines P_j and $P_{\pi(n)}$.

3) STATE TRANSITION RULE

In ACO algorithm, each ant k has a corresponding tabu list $tabu_k$. When an ant k deploys a virtual machine V_i to a physical machine P_j, the virtual machine V_i is placed on the tabu list $tabu_k$. The subsequent search process can only be for the node that are not in the tabu list. The ants deploy virtual machines to physical machines one by one according to the transition probability. The transition probability is expressed as follows:

$$P_{ij}^{k}(t) = \begin{cases} \dfrac{(\tau_{ij}(t))^{\alpha}(\eta_{ij}(t))^{\beta}}{\sum\limits_{i \notin tabu_k} (\tau_{ij}(t))^{\alpha}(\eta_{ij}(t))^{\beta}}; & \text{if } j \in allow_k \\ 0; & \text{Otherwise} \end{cases} \tag{18}$$

where $P_{ij}^{k}(t)$ is the probability of the ant k deploys the virtual machine V_i to the physical node P_j; $allow_k$ is the set of physical nodes that the ant k can select; α is the pheromone factor (indicates the degree of influence of the pheromone in the process of choosing a path by the ant); β is the heuristic factor (which indicates the degree of influence of heuristic information in the process of ants for choosing a path).

4) SIMULATED ANNEALING MECHANISM

Another improvement in our proposed algorithm is to include to it a Simulated Annealing (SA) mechanism. As the temperature gradually decreases with the number of iterations, according to the current temperature. According to the SA mechanism, an updated solution set is generated from the solution space searched by all ants. Then, we update the corresponding pheromone in the updated solution set, and use the global update rule to strengthen the pheromone. At the end of each iteration, the solution in the candidate set is selected with a certain probability, and the pheromone update is performed on the path it has passed. Initially, the solution in the candidate set with high temperature is selected with a large probability, so that the global pheromone is widely distributed and will not fall into a local optimum. As the number of iterations increases, the temperature gradually decreases, and the probability that a worst solution in the candidate set is selected gradually decreases. Thereby, the pheromone update is concentrated on the better path, which speeds up the algorithm convergence speed. The specific definition and process are as follows:

Definition 1: All ants will generate a deployment plan after completing an iteration, and all the plans are formed into a candidate solution set, denoted as $C = ant_k(S_k, f_k); 1 < k \leq K$, where S_k represents the deployment plan generated by the ant k, f_k indicates the objective function value of the plan, calculated by (7).

Definition 2: After each iteration, a local best solution will be generated, which is the best solution among all ants in this iteration, denoted as $ant_{localbest}(S_{localbest}, f_{localbest})$. The current global best solution is denoted as $ant_{best}(S_{best}, f_{best})$.

Definition 3: According to the SA mechanism, it's determined whether the solutions in the candidate set need to be updated one by one. The selected solutions form an update set are denoted as $U = ant_k(S_k, f_k)$.

The candidate set is generated after completing an iteration according to the above-defined algorithm, and the solutions in the candidate set will be accepted to be added to the update set according to the Metropolis–Hastings [30], [31] criterion (Eq. (21)). Let ε a randomly generated number drawn on [0, 1], if the acceptance probability P is more than ε, then add the solution to the update set as in (21), otherwise the solution is rejected. The initial acceptance probability of annealing is close to 1, and the pheromone update is performed on the deployment plan corresponding to the updated centralized solution to make the pheromone distribution more widely and avoid falling into a local optimum. After all the ants complete one iteration, the algorithm performs cooling processing, setting a reasonable cooling processing mechanism, such as in (22). As the number of iterations increases, the temperature gradually decreases, the acceptance probability decreases, and the probability of a poor solution being accepted decreases, which makes the pheromone distribution on the path more concentrated and speeds up the algorithm convergence speed. In order to strengthen the solution quality,

the current global best solution $ant_{best}(S_{best}, f_{best})$ is required to update its pheromone after each iteration t.

$$\Delta f = f_k - f_{localbest} \tag{19}$$

$$P = \begin{cases} \exp^{-\theta \Delta f / T}, & \text{if } \Delta f > 0 \\ 1, & \text{if } \Delta f = 0 \end{cases} \tag{20}$$

$$\begin{cases} P \geq \varepsilon, & \text{Update the pheromone} \\ P < \varepsilon, & \text{Don't update the pheromone} \end{cases} \tag{21}$$

$$T(t+1) = \varphi T(t) \tag{22}$$

where φ is the cooling factor $\varphi \in [0, 1]$.

As the temperature T gradually decreases, the pheromone update is more and more concentrated on the better solution. In order to avoid the algorithm from falling into a local optimum prematurely, we perform the following partial optimization function $ant_{localbest}(S_{localbest}, f_{localbest})$ after each iteration: Randomly exchange virtual machine pairs on the physical machine to generate a new solution $ant_{new}(S_{new}, f_{new})$, such as $f_{new} - f_{localbest} < 0$ then accept the solution and add it to the update set. The random exchange is repeated h times. During the exchange process, all constraints must be met.

5) PHEROMONE UPDATE

According to the update set obtained previously, the search path corresponding to the solution in the set is updated locally, and the update rule is as in (23).

$$\tau_{ij}(t) = (1-\rho)\tau_{ij}(t-1) + \sum_{k=1}^{K} \Delta\tau_{ij}^{k} + \Delta\tau_{ij}^{best} \tag{23}$$

$$\tau_{ij}^{k} = \frac{f_{localbest}}{f_k} \tag{24}$$

$$\tau_{ij}^{best} = \frac{f_{localbest}}{f_{best}} \tag{25}$$

where $\Delta\tau_{ij}^{k}$ is the pheromone left behind when the ant k deploys the virtual machine V_i to the physical machine P_j, τ_{ij}^{best} is the pheromone on the current global best solution path, and ρ is the volatile factor.

6) HACOS ALGORITHM IMPLEMENTATION

Our proposed HACOS algorithm is shown in Figure 3 and Algorithm 1. The algorithm process can be stated as follows.

Input:

Virtual machine required resources R, Physical machine resources H; Network topology G, link capacity, estimated traffic between virtual machines; Algorithm parameters K, α, β, ρ; Simulated annealing mechanism parameters $T_{max}, T_{min}, \theta, \varphi$.

Output:

Mapping relationship between virtual machine and physical machine π; The final solution objective function value f.

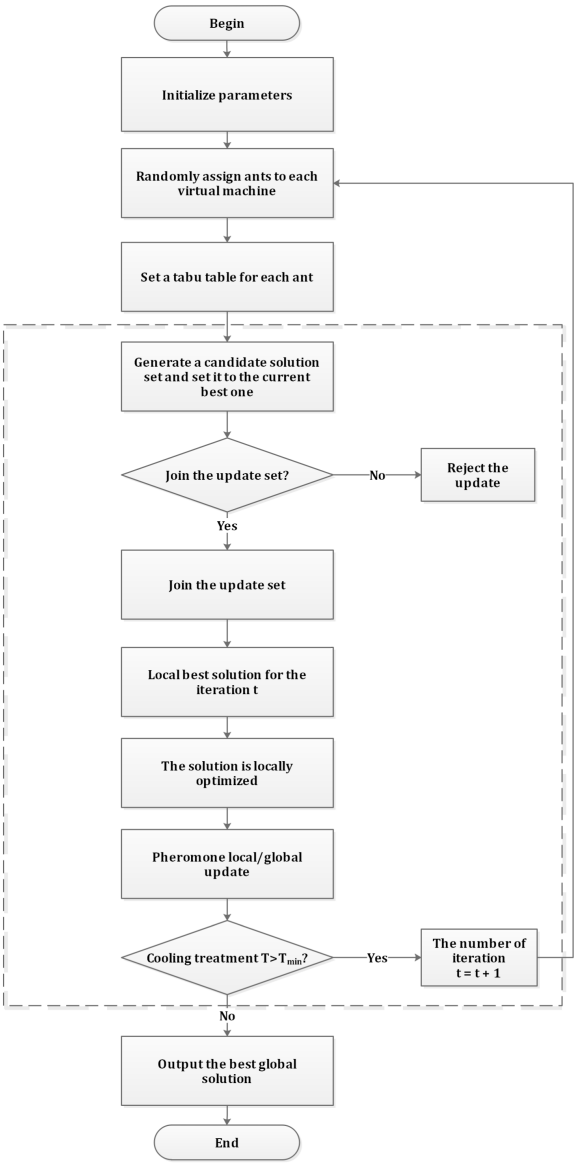


FIGURE 3. Flow chart of HACOS algorithm.

Algorithm flow:

1) Initialize parameters and randomly generate a virtual machine deployment plan;

2) Randomly place the ant k put on the virtual machine i; select a physical node according to (18); add the deployed virtual machine to the tabu table $tabu_k$;

3) According to the method of step 2, ants deploy virtual machines others than in $tabu_k$ until all virtual machines are deployed, and generate a deployment plan π_k;

4) Repeat steps 2 and 3 for all ants to generate a candidate set C, update the global best solution f, and generate a local best solution $f_{localbest}$;

5) Locally optimize the local best solution generated in this iteration, and randomly exchange virtual machine pairs on the physical machine to generate a new solution $ant_{new}(S_{new}, f_{new})$, such as $f_{new} - f_{localbest} < 0$. Then accept the solution, add it to the update set U, and repeat random

exchange h times (all constraints must be satisfied during the exchange process);

6) Select the solution in the candidate set to the update set according to (20)(21), and update pheromone according to (23)(24)(25);

7) If $T < T_{min}$, the algorithm ends, and outputs π, f; Otherwise update T according to (22) and return to step 2;

Algorithm 1 HACOS Algorithm

Initialize: $\alpha, \beta, \theta, \phi, T_{\min}, T_{\max}$
  $T = T_{\max}$;
  while $T > T_{\min}$ **do**
    for ant $k = 1$ to K **do**
      for VM $i = 1$ to N **do**
        //Deploy all unassigned V_i
        for host $j = 1$ to $allow_k$ **do**
          if $i \notin tabu_k$ **then**
            Calculate η_{ij} according to (17)
            Calculate P_{ij}^k according to (18)
            if $P_{ij}^k > P_{\max}$ **then**
              //Record $P_{\max}$ with the highest P_{ij}^k
              $P_{\max} = P_{ij}^k$; $h = j$;
            end if
            $i \rightarrow h$ //Ant k deploys V_i to P_h
          end if
        end for
      end for
      //Add the deployment plan of ant k to the candidate set
      $C = \pi_k$;
      Calculate f_k according to (7);
      if $P > \varepsilon$ **then**
        //Add the deployment plan of ant k to the update set
        $U = \pi_k$;
      end if
    end for
    Update pheromone according to (23)(24)(25);
    $T = \varphi T$;
    $U = \emptyset$; $C = \emptyset$;
  end while
  return f, π; //Return the best deployment plan

VI. EXPERIMENTAL RESULTS

In order to verify the effectiveness of our proposed algorithm, we conducted simulation experiments based on the well-known CloudSim platform. The experiments have been coded in JAVA language, and the hardware configuration of the experiment environment is: CPU 3.2GHz, RAM 8GB.

The test problems as generated as follows. The number of virtual machines configured in the experiments is 100, the number of physical machines is 50, the computing capacity of the physical machines is 1500MIPS, the memory is 8GB, the computing resource requests of the virtual machines is $C \in$ {200MIPS, 500MIPS, 700MIPS}, and the memory resource requests is $M \in$ {512MB, 1024MB, 2560MB}. According to the sets C and M, 150 virtual machines are randomly generated. The network traffic between virtual machines is similar to the measurement data in Meng *et al.* [32]. Our experiments are divided into two parts: one is a single-objective optimization of total network traffic, which is carried out in three different network topology environments, namely Tree, Fat-Tree, and VL2 (Virtual Layer 2); the other is a bi-objective optimization of total network traffic and link utilization rate under the Fat-Tree network topology. In the experiments, the tasks are set as full-load tasks, that is, the task demand is constant, and the data of the data center is simulated for 3 hours.

TABLE 2. ACO algorithm parameters.

Number of ants K	45
Pheromone factor α	0.5
Heuristic factor β	1.5
The maximum number of iterations G	500

TABLE 3. Simulated annealing mechanism parameters.

Maximum temperature $T_{\max}$	200
Minimum temperature $T_{\min}$	0.1
Cooling factor φ	0.9

The selection of different parameters of an algorithm will have different effects on the algorithm. According to Kumar and Raza [33], in the analysis of the ant colony algorithm parameter selection, combined with our proposed algorithm, the parameters are constantly adjusted during the experiments, in order the stronger global search performance. The parameters of our HACOS algorithm are shown in Tables 2 and 3.

A. PERFORMANCE OF THE PROPOSED ALGORITHM FOR MINIMIZING THE TOTAL NETWORK TRAFFIC

The purpose of this paper is to optimize the total network traffic of the data center, and to minimize the maximum link utilization when the total network traffic does not change much. First, we optimize the total network traffic with a single objective, and test the total network traffic and link utilization under the three mentioned network topologies. Our proposed algorithm is compared to the basic ACO and BFD (Best-fit decreasing) algorithms. The BFD algorithm abstracts the virtual machine group into a power undirected graph, sorts all the nodes in the graph in descending order of degree, and then selects the physical machine based on the principle of placing the virtual machines with high traffic on the same physical machine node.

When optimizing the total network traffic only, it can be seen from the comparison chart of total network traffic in Figure 4, the comparison chart of maximum link utilization in Figure 5, and the comparison chart of minimum link utilization in Figure 6:

(a) Our proposed algorithm HACOS is very effective and outperforms both ACO and BFD. Indeed, HACOS present

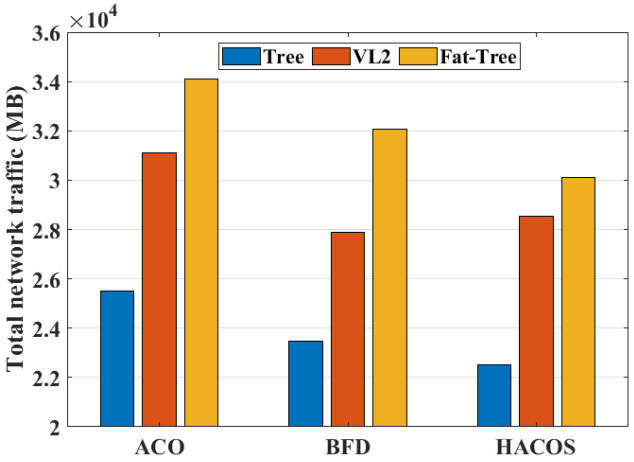


FIGURE 4. Comparison of total network traffic (single objective optimization).

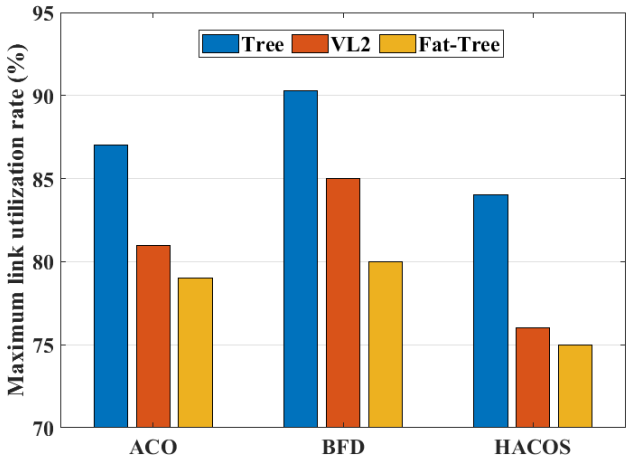


FIGURE 5. Comparison of maximum link utilization rate (single objective optimization).

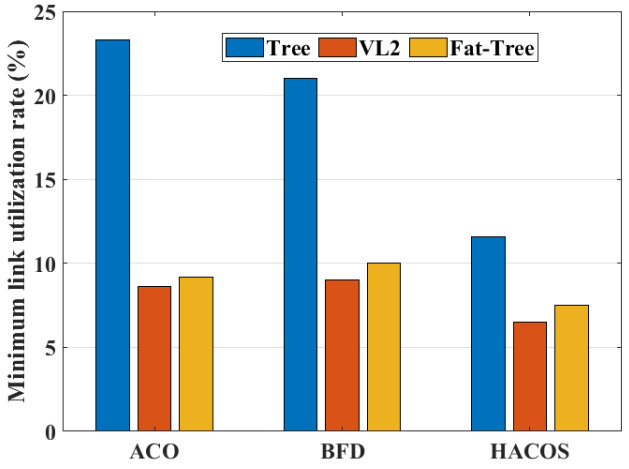


FIGURE 6. Comparison of minimum link utilization rate (single objective optimization).

almost the lowest values, except for the total network traffic under VL2 topology;

(b) The total network traffic generated by the same optimization algorithm under different network topologies is different. Under the Tree network topology, the total network traffic is the smallest whereas it's the largest under the Fat-Tree topology;

(c) Tree: The network traffic under the Tree network topology is relatively small, but the maximum link utilization rate is relatively high, and the minimum link utilization rate is also higher than that under the other two network topologies. We can conclude that a placement strategy that only optimizes the total network traffic will generate network congestion caused by this network topology;

(d) Fat-Tree: The maximum link utilization rate is the smallest, and the total network traffic is relatively large, indicating that the traffic distribution under this network topology is relatively balanced;

(d) VL2: The large difference between the maximum link utilization rate and the minimum link utilization rate under this network topology indicates that the network traffic is unevenly distributed.

According to the above-mentioned viewpoints, the maximum link utilization rate is significantly higher when the total traffic of the data center network is optimized by a single objective. Under different network topologies, the network traffic distribution is different. The Tree network topology has lower network traffic, but the maximum link utilization rate is relatively high. The main reason is that there is generally a bandwidth convergence ratio between the access layer and the aggregation layer of the network. The higher the convergence ratio, the easier the link will be congested. In the Fat-Tree network topology, there are multiple parallel links between servers, which facilitates the balanced distribution of network traffic. Therefore, in the experiments, the total traffic under this network structure is relatively high, but the maximum link utilization rate is low.

B. PERFORMANCE OF THE PROPOSED ALGORITHM FOR MINIMIZING THE TOTAL NETWORK TRAFFIC AND THE MAXIMUM LINK UTILIZATION

In this section, we consider the optimization of the total network traffic and the maximum link utilization under the Fat-Tree network topology, and our proposed algorithm is compared to an ACO based virtual machine placement algorithm named VMPACS [34], and to a Particle Swarm Optimization algorithm (PSO) [13]. We select three groups of different numbers of virtual machines, and the number and configuration of physical machines are the same as the above experiments. The experimental results are shown in Figure 7 (total network traffic) and Figure 8 (maximum link utilization).

The results reported in Figures 7 and 8 reveal that our proposed algorithm has achieved very good performance in optimizing the total network traffic. Indeed, HACOS algorithm is obviously better in optimizing the total network traffic and the maximum link utilization rate. The obtained results in term of maximum link utilization are nearly close when compared to VMPACS. Also, we remarked that the

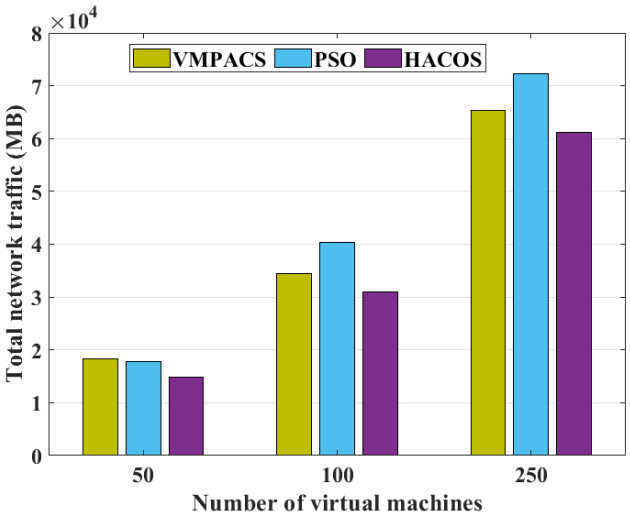


FIGURE 7. Comparison of total network traffic (bi-objective optimization).

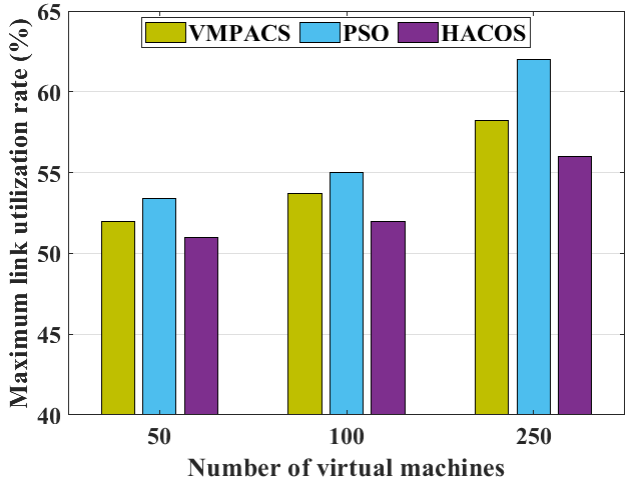


FIGURE 8. Comparison of maximum link utilization rate (bi-objective optimization).

total network traffic and the maximum link utilization rate have decreased significantly compared to the single-objective optimization on the above experiments.

From Figures 7 and 8, we remark that the PSO algorithm have given the worst results and the gap compared to the remaining algorithms increases when the number of virtual machines increase.

Another interpretation of the reason of performance difference between the algorithms is the heuristic information setting method: The VMPACS algorithm considers two optimization goals when determining the heuristic information; In the PSO algorithm the fitness function that determines the particle moving direction is jointly determined by the network traffic and the maximum link utilization rate; Our proposed algorithm uses a two-stage optimization to minimize the network traffic first, and reduce the link utilization rate without much impact on the network traffic.

As a conclusion, we notice that the results presented in Section VI-A support the results provided by Figures 7 and 8.

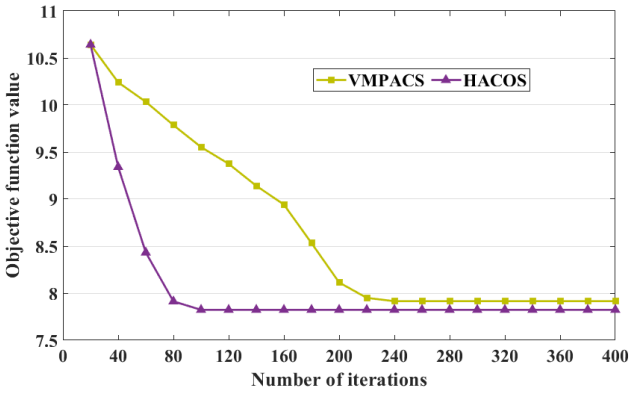


FIGURE 9. Convergence performance analysis.

C. CONVERGENCE ANALYSIS

In order to get more accurate idea about the performance of our proposed algorithm, we compared it to VMPACS in term of algorithm convergence. We run both algorithms 10 times each. The number of virtual machines is set to 250 and we calculate the objective function value according to (7). It can be seen from Figure 9 that both HACOS and VMPACS converge within 400 iterations, and the solution of HACOS is smaller. The convergence speed of HACOS is significantly faster than VMPACS algorithm. It has begun to converge after the algorithm is iterated to 100 times, and the VMPACS algorithm has a convergence trend after 240 iterations. It can be seen that HACOS has been improved in terms of algorithm convergence performance.

VII. CONCLUSION

This paper investigates the very challenging virtual machine placement problem on cloud data centers. We proposed an improved Ant Colony Optimization algorithm to solve this problem. The computational experiments shows firstly that the network congestion has been caused by optimizing the total network traffic alone, and secondly provides evidence that our proposed algorithm performs consistently well.

Several various adjustments, testing, and experiments have been left to be completed in the future. Future work concerns exploring more virtual machine placement objectives related to cloud users. QoS-related problems, in particular latency, can be considered in the future.

REFERENCES

[1] Z. R. Alashhab, M. Anbar, M. M. Singh, Y. B. Leau, Z. A. Al-Sai, and S. A. Alhayja'a, "Impact of coronavirus pandemic crisis on technologies and cloud computing applications," *J. Electron. Sci. Technol.*, vol. 19, Mar. 2021, Art. no. 100059.

[2] R. Singh, "Cloud computing and COVID-19," in *Proc. 3rd Int. Conf. Signal Process. Commun. (ICPSC)*, May 2021, pp. 552–557.

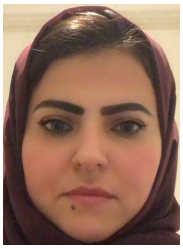
[3] S. Alhomdy, F. Thabit, F. H. Abdulrazzak, A. Haldorai, and S. Jagtap, "The role of cloud computing technology: A savior to fight the lockdown in COVID 19 crisis, the benefits, characteristics and applications," *Int. J. Intell. Netw.*, vol. 2, pp. 166–174, 2021.

[4] J. Dong, X. Jin, H. Wang, Y. Li, P. Zhang, and S. Cheng, "Energy-saving virtual machine placement in cloud data centers," in *Proc. 13th IEEE/ACM Int. Symp. Cluster, Cloud, Grid Comput.*, May 2013, pp. 618–624.

[5] X. Li, Z. Qian, S. Lu, and J. Wu, "Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center," *Math. Comput. Model.*, vol. 58, no. 5, pp. 1222–1235, 2013.
[6] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, 2008.
[7] F. L. Pires, E. Melgarejo, and B. Baran, "Virtual machine placement. A multi-objective approach," in *Proc. 39th Latin Amer. Comput. Conf. (CLEI)*, Oct. 2013, pp. 1–8.
[8] S. Zou, X. Wen, K. Chen, S. Huang, Y. Chen, Y. Liu, Y. Xia, and C. Hu, "VirtualKnotter: Online virtual machine shuffling for congestion resolving in virtualized datacenter," *Comput. Netw.*, vol. 67, pp. 141–153, Jul. 2014.
[9] F. L. Pires and B. Baran, "Multi-objective virtual machine placement with service level agreement: A memetic algorithm approach," in *Proc. IEEE/ACM 6th Int. Conf. Utility Cloud Comput.*, Dec. 2013, pp. 203–210.
[10] S. Fang, R. Kanagavelu, B.-S. Lee, C. H. Foh, and K. M. M. Aung, "Power-efficient virtual machine placement and migration in data centers," in *Proc. IEEE Int. Conf. Green Comput. Commun. IEEE Internet Things IEEE Cyber, Phys. Social Comput.*, Aug. 2013, pp. 1408–1413.
[11] G. Y. Luo, Z. Z. Qian, and S. L. Lu, "A network-aware VM re-scheduling algorithm," *Chin. J. Comput.*, vol. 38, pp. 932–943, Jan. 2015.
[12] J. Montgomery and M. Randall, "Anti-pheromone as a tool for better exploration of search space," in *Proc. Int. Workshop Ant Algorithms* (Lecture Notes in Computer Science), vol. 2463, Berlin, Germany: Springer 2002, pp. 100–110.
[13] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM placement and routing for data center traffic engineering," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2876–2880.
[14] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," *Egyptian Informat. J.*, vol. 16, no. 3, pp. 275–295, 2015.
[15] X.-F. Liu, Z.-H. Zhan, J. D. Deng, Y. Li, T. L. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 113–128, Feb. 2018.
[16] F. Alharbi, Y.-C. Tian, M. Tang, W.-Z. Zhang, C. Peng, and M. Fei, "An ant colony system for energy-efficient dynamic virtual machine placement in data centers," *Expert Syst. Appl.*, vol. 120, pp. 228–238, Apr. 2019.
[17] S. Supreeth and K. K. Patil, "Virtual machine scheduling strategies in cloud computing—A review," *Int. J. Emerg. Technol. Learn.*, vol. 10, pp. 181–188, Jan. 2019.
[18] W. Wei, H. Gu, W. Lu, T. Zhou, and X. Liu, "Energy efficient virtual machine placement with an improved ant colony optimization over data center networks," *IEEE Access*, vol. 7, pp. 60617–60625, 2019.
[19] A. S. Abohamama and E. Hamouda, "A hybrid energy–aware virtual machine placement algorithm for cloud environments," *Expert Syst. Appl.*, vol. 150, Jul. 2020, Art. no. 113306.
[20] J. Peake, M. Amos, N. Costen, G. Masala, and H. Lloyd, "PACO-VMP: Parallel ant colony optimization for virtual machine placement," *Future Gener. Comput. Syst.*, vol. 129, pp. 174–186, Apr. 2022.
[21] A. Ragmani, A. Elomri, N. Abghour, K. Moussaid, and M. Rida, "FACO: A hybrid fuzzy ant colony optimization algorithm for virtual machine scheduling in high-performance cloud computing," *J. Ambient Intell. Humanized Comput.*, vol. 11, no. 10, pp. 3975–3987, Oct. 2020.
[22] M. S. Ajmal, Z. Iqbal, F. Z. Khan, M. Ahmad, I. Ahmad, and B. B. Gupta, "Hybrid ant genetic algorithm for efficient task scheduling in cloud data centers," *Comput. Electr. Eng.*, vol. 95, Oct. 2021, Art. no. 107419.
[23] 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.
[24] R.-H. Huang and C.-L. Yang, "Ant colony system for job shop scheduling with time Windows," *Int. J. Adv. Manuf. Technol.*, vol. 39, nos. 1–2, pp. 151–157, Oct. 2008.
[25] V. Maniezzo, "Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem," *INFORMS J. Comput.*, vol. 11, no. 4, pp. 358–369, 1999.
[26] M. Dorigo, "Optimization, learning and natural algorithms," Ph.D. dissertation, Dep. Electron., Polytech. Univ. Milan, Milan, Italy, 1992.
[27] V. Selvi and R. Umarani, "Comparative analysis of ant colony and particle swarm optimization techniques," *Int. J. Comput. Appl.*, vol. 5, no. 4, pp. 1–6, 2010.
[28] M. Mulani and V. L. Desai, "Design and implementation issues in ant colony optimization," *Int. J. Appl. Eng. Res.*, vol. 13, no. 16, pp. 12877–12882, 2018.
[29] K. Tewani, "Ant colony optimization algorithm: Advantages, applications and challenges," *Comput. Model. New Technol.*, vol. 21, no. 2 pp. 69–70, 2017.
[30] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, Apr. 1970.
[31] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *J. Chem. Phys.* vol. 21, no. 6, pp. 1087–1092, 1953.
[32] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.
[33] D. Kumar and Z. Raza, "A PSO based VM resource scheduling model for cloud computing," in *Proc. IEEE Int. Conf. Comput. Intell. Commun. Technol.*, Feb. 2015, pp. 213–219.
[34] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *J. Comput. Syst. Sci.*, vol. 79, no. 8, pp. 1230–1242, 2013.

NAWAF ALHARBE received the B.Sc. degree in computer science from the College of Computer Science and Engineering, Taibah University, Medina, Saudi Arabia, in 2007, the M.Sc. degree in advanced computer science from the Department of Computer Science, Huddersfield University, U.K., in 2011, and the Ph.D. degree in computer science from Staffordshire University, in 2015. He is currently working as an Associate Professor with the Applied College, Taibah University. His research interests include knowledge management systems in smart healthcare operations using emerging technology, such as RFID, ZigBee, the Internet of Things (IoT), cloud computing, artificial intelligence, machine learning, big data and data processing, knowledge engineering, knowledge harvesting, and healthcare applications.

MOHAMED ALI RAKROUKI received the B.S., M.S., and Ph.D. degrees in management information systems from the University of Tunis, Tunisia, in 2003, 2005, and 2010, respectively. From 2005 to 2008, he was a Lecturer with the Computer Science Department, University of Tunis. He has been working as an Assistant Professor with the Applied College, Taibah University, Saudi Arabia, since 2010. His research interests include machine scheduling, artificial intelligence, cloud computing, evolutionary computation, data science, and machine learning.



ABEER ALJOHANI received the B.Sc. degree in computer science from the College of Computer Science and Engineering, Taibah University, Medina, Saudi Arabia, in 2008, the M.Sc. degree in information technology for management from the Department of Computer Science, The University of Coventry, U.K., in 2011, and the Ph.D. degree in computer science from The University of Loughborough, in 2019. She is currently working as an Assistant Professor with the Applied College, Taibah University. Her research interests include sensor technology in e-Health and smart health management, smart cities, data science, artificial intelligence, evolutionary computation, machine learning, deep learning, and pattern recognition.