J Grid Computing (2021) 19: 7
https://doi.org/10.1007/s10723-021-09547-1

Energy Refining Balance with Ant Colony System for Cloud Placement Machines

Hamed Tabrizchi · **Marjan Kuchaki Rafsanjani**

Received: 2 November 2019 / Accepted: 26 December 2020 / Published online: 20 February 2021

Abstract Cloud computing has been one of significant domains of processing service in social networks like the internet and local networks in recent years. One of the main problems in cloud computing is placing a virtual server onto physical servers. This problem will have a remarkable effect on energy consumption, because if a suitable placement is not chosen for it, a great amount of energy will be used to keep the physical servers on. This paper aims to optimize the use of energy in physical servers and in order to achieve it, the last placement in Virtual Machines (VMs) and Physical Machines (PMs) is considered. The proposed approach for allocating resources to VMs is the use of ant colony algorithm. This approach solves virtual machine placement problem and attempts to have the least effects on the environment and energy consumption.

Keywords Ant Colony optimization (ACO) · Cloud computing · Virtual machine placement · Energy consumption

H. Tabrizchi
Department of Computer Science, Faculty of Mathematical Sciences, University of Tabriz, Tabriz, Iran
e-mail: hamed.tabrizchi@gmail.com

M. Kuchaki Rafsanjani (✉)
Department of Computer Science, Faculty of Mathematics and Computer, Shahid Bahonar University of Kerman, Kerman, Iran
e-mail: kuchaki@uk.ac.ir

1 Introduction

In recent year, cloud computing has become a popular computing paradigm for hosting and delivering services over the Internet [1]. In addition, it is generally considered a new field in internet computing, which processes and saves the information and offers services to the users based on their demands [2–5]. Further, this computing paradigm mainly aims to provide access to a lot of virtual computing resources. In a nutshell, all the cloud services are virtually presented and the cloud offers three types of services to the user through the internet as follows [6–8]:

- Infrastructure as a Service (IaaS);
- Platform as a Service (PaaS);
- Software as a Service (SaaS).

Virtual Machine (VM) was first innovated in 1970, and the idea was formed so that to make it possible to use some computing environment onto the physical environment resulting in sharing the physical infrastructure for a large number of users instead of using separate physical systems for every single user. Furthermore, according to [6], VM placement is a process of mapping VMs to Physical Machines (PMs). VM starts to operate after processing the capacity of the physical server based on the type of the operating system and the amount of the required resources such as CPU, RAM, storage, and bandwidth which are selected by the users.

VMs and PMs should be placed based on the available resources and demands. In fact, VMs use the available PM resources together. Placing the resources together only requires dovetailing the resources with the demands. However, keeping and using physical servers require spending money and energy, and thus attempts are made to find the best placement in order to save the energy. It is undeniable that the optimization of energy consumption relies on network traffic, performance and resource utilization optimization [9]. For this reason, an optimal placement is the one in which the most possible number of VMs are placed onto the last number of PMs and the one which keeps the VMs off and wastes less energy. Moreover, based on the most economical PMs, giving priority to resource allocation is considered another viewpoint which can optimize the energy consumption when it varies in PMs.

The problem of optimizing the use of energy for placing the VMs in cloud computing is called "NP-hard problem" which is remarkably similar to the Bin packing problem [10]. Heuristic algorithm including First Fit, as well as metaheuristic algorithms like Simulated Annealing (SA) [11], Genetic Algorithm (GA) [12], and Ant Colony Optimization (ACO) [13] can solve the problem, though they are unable to find the best solution. Ant Colony Optimization is a meta-heuristic algorithm that aims to find near-optimal solutions based on a probabilistic technique. In the present paper, ACO has been used due to the fact that this algorithm is efficient for solving graph represented problems and also be adaptable in dynamic applications. Moreover, ACO is able to use to solve problems belonging to the NP class and also there are numerous methods have been introduced among which ACO method has succeeded in finding better solutions compared to heuristic methods [14]. For this reason, the Ant Colony Optimization method has been used as a reference method in this work.

Based on Ant Colony System (ACS) [15], the present study introduces a method for optimizing VM placement. This method aims to reduce the energy consumption by minimizing the number of servers while working and balancing the VMs. Achieving this goal involves considerable computing expanses which nearly every way considers for VM placement based on a single resource while the present study considers the VMs based on the multiple resources such as CPU, RAM, and I/O. This further challenge the computation and makes it closer to the reality. Additionally, this paper proposes and elucidates Energy Refining Balance (ERB) approach for ACS in order to refine the energy consumption, which leads to the creation of ERBACS algorithm.

In summary, the main contributions of this paper can be considered as follows:

(1) The main purpose of this research is to present a comprehensive method of placing virtual machines on physical machines with respect to multiple resources such as CPU, RAM, and I/O, with the least amount of energy loss.
(2) This research presents a new ant colony based approach with dual fitness for solving the problem of placement in the cloud computing environments. Moreover, both novel functions evaluate the solutions during the computational process of the ant colony algorithm, taking into account the energy loss and satisfying the virtual machine placement constraints.
(3) To achieve the best possible solution after all iterations, an innovative approach refined current best solution of each iteration based on the all amounts of used energy in active physical servers.

The remaining parts of the current study are organized as follows. The preliminaries for explaining the VM placement and defining the ACO are presented in Section 2. In addition, Section 3 addresses current achievements in this domain, and each achievement discusses similar results with different advantages. Further, Section 4 expounds on phases of ERBACS approach. Section 5 describes ERBACS algorithm. In Section 6, the proposed approach is implemented and compared to the other heuristic and metaheuristic methods, and finally, the conclusion is drawn in Section 7.

2 Preliminaries

This section focuses on the preliminaries of the online VM placement problem and the definition of ant colony system (ACS).

2.1 Virtual Machine Placement

The users impose their demands on every data center based on the resources they need. Bandwidth is taken into account for sending and receiving the information between the resources required by the users. Therefore, the bandwidth is regarded as part of the constraints related to VM placement and the placement is considered such that to take into account the bandwidth while the users have great access to the resources they demand.

For example, suppose the situation depicted in Fig. 1. There are three servers, each having a processor capable of performing VMs. A simple process for VM placement is as follows [16]:

(1) For each server, compute the resource requirements of the application using the resource usage statistics of the server over a period of time;
(2) Select a target server with compatible virtualization software and CPU types, similar network connectivity, and usage of the shared storage;
(3) Place the first VM on the first server in step 2. Then, place the second VM on the same server if it can satisfy the resource requirements. Otherwise, add a new PM and place the VM on this new machine. Continue this step until each of the VMs is placed on a PM and add a new PM when required;
(4) The set of the resulting hosts at the end of step 3 comprises the consolidated server cluster.

There are N number of VMs and M number of PMs which are regarded as a set. The sets of PMs and VMs include all the PMs and VMs, respectively. The set used for PMs is called R and it represents the resources. Furthermore, the size of the set is the same as the number of resources and is demonstrated by d, therefore $d = 3$ and $R = \{\text{CPU, RAM, I/O}\}$.

Moreover, according to [17], a vector called the "capacity of resource" is considered for every P_i, which has d number of dimensions and is referred to as "Resource Capacity Vector" (RCV).

$$\textit{For each } P_i \in PM, \text{ there exists } C_i^k, \quad \textit{where } C_i^k \textit{ is total capacity of resource } R_i^k, \textit{ where } 1 \le k \le d \tag{1}$$

Additionally, another vector called the "demand of resource" exists for every V_j, which includes d number of dimensions and indicates the amount of resource

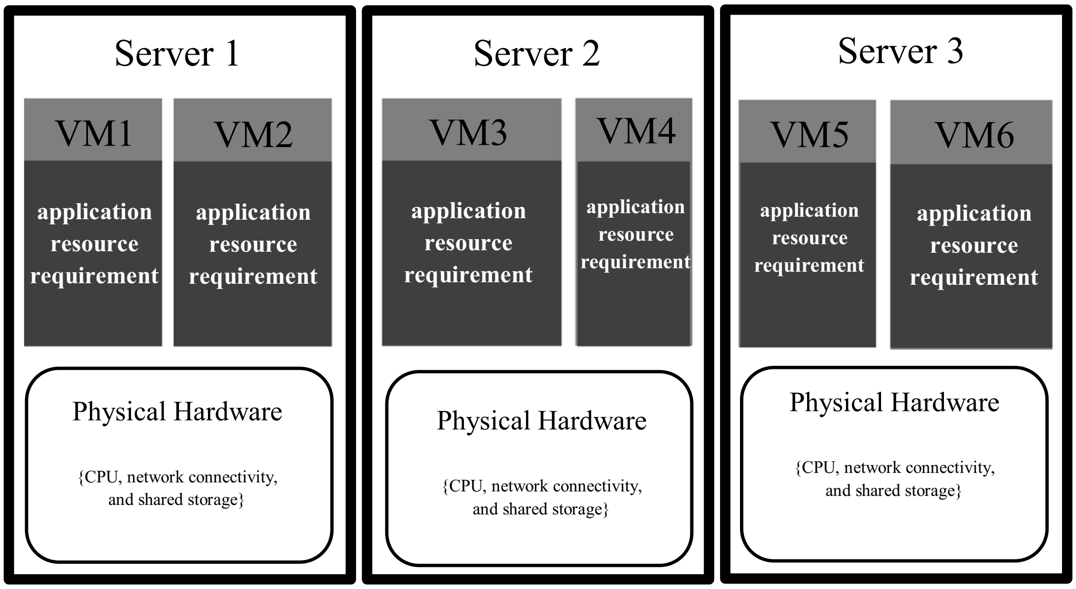

Server 1
Server 2
Server 3
VM1
VM2
VM3
VM4
VM5
VM6
application resource requirement
Physical Hardware
{CPU, network connectivity, and shared storage}

Fig. 1 An illustration of VM placement in a virtualized environment

demand for VM. According to [17], this vector is called "Resource Demand Vector" (RDV).

$$\begin{gathered} For\ each\ V_j \in VM, there\ exists\ D_j^k, \\ where\ \ D_j^k\ is\ demand\ of\ resource\ R_i^k, where\ 1 \leq k \leq d \end{gathered} \tag{2}$$

In addition, another vector is presented as "Resource Utilized Vector" (RUV) which is used to compute the amount of the resource allocation to VMs by PMs and keeps all the allocated resources.

$$U_i^k \ = \sum D_j^k \left(\text{for} \forall x_{ij} = 1 \right) \tag{3}$$

$$x_{ij} = \begin{cases} 1, if\ v_j\ is\ placed\ on\ p_i \\ 0\ , \qquad\qquad otherwise \end{cases} \tag{4}$$

In the above-mentioned equation, the element of x_{ij} equals to 1 if P_i includes V_j, otherwise, it is equal to 0. In fact, x_{ij} determines the placement in physical and virtual servers. However, some constraints should be considered while placing these servers [18].

$$\sum_{j=1}^{N} VC_j \cdot x_{ij} \leq PC_i \cdot y_i \quad \forall i \tag{5}$$

$$\sum_{j=1}^{N} VM_j \cdot x_{ij} \leq PM_i \cdot y_i \quad \forall i \tag{6}$$

$$\sum_{j=1}^{N} VI_j \cdot x_{ij} \leq PI_i \cdot y_i \quad \forall i \tag{7}$$

The elements *VC*, *VM*, and *VI* refer to the amounts of the required VM [6] in the above-mentioned equations. Further, *PC, PM, PI* determine the amount of available resources in PM for CPU, memory, and I/O.

Furthermore, another vector called "Energy Consumption Vector" (ECV) is presented for data center energy consumption, which represents the amount of every P_i which is proportional to the allocated resources.

$$\begin{gathered} For\ each\ P_i \in P \\ \text{there exists } E_i^k, where\ E\ is\ Energy\ consumption\ of\ resource\ R_i^k \end{gathered} \tag{8}$$

As previously mentioned, the present study seeks to minimize the amount of energy consumption in PM and to compute the energy use. Moreover, it is possible to use ECV vector for a suitable placement which has the least number of active PM and consumes the least amount of energy. In other words, the energy consumption vector holds the consumed energy per unit time by considering the power consumption of PMs.

2.2 Ant Colony System (ACS)

The ACS computing approach is inspired by special behavior of the ant. This method was first used to find a solution for the Traveling Salesman Problem (TSP) by metaheuristic computation of the ant from the nest to the food. This system differs from the previous ant colony algorithm due to three main aspects [15]:

- The state transition rule provides a direct way to balance between exploring the new edges and exploiting a priori and accumulated knowledge about the problem;
- The global updating rule is only applied to edges which belong to the best ant tour;
- A local pheromone update rule is applied while the ants construct a solution.

In this method, pheromone can typically be a help to achieve a solution for the graph problems.

2.2.1 Pheromone

The ant can go through the pathway by a substance called pheromone when leaving the nest to find food. In fact, ants use heuristic information in the pheromone to find the nearest path. The components of the problem and its constraints should be taken into account. Additionally, with considering the information in the problem, the probability of choosing the path by artificial ants which is actually the probability of placing V_j to P_i is found, based on the constraints of the problem, resource and demands, as is shown in the Fig. 2.

Heuristic quantities play a vital role in the process of achieving a solution. In addition, the parameters of the problem constraints should be considered for initializing the quantities.

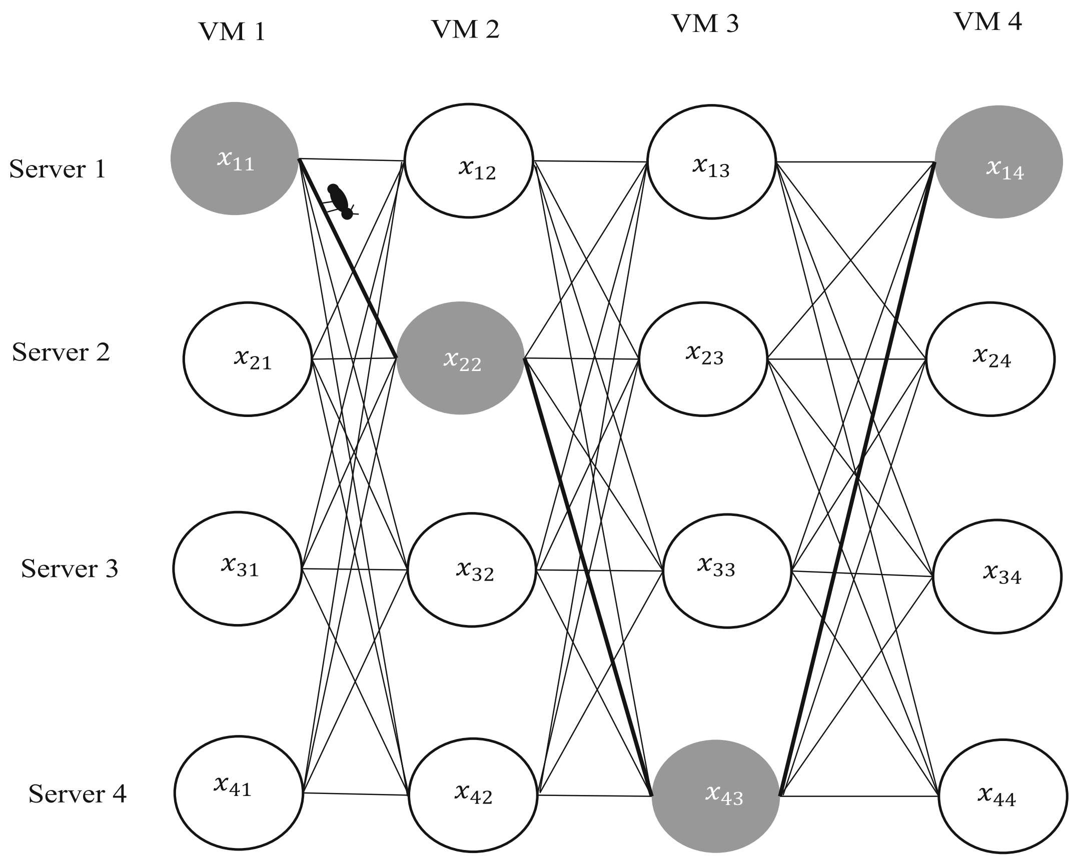


Fig. 2 An example of a construction scheme

So far, RUV and RCV vectors have been presented. The RUV vector is equal to $C_x + M_y + I_z$ in which *I, M*, and *C* represent I/O, memory, and CPU, respectively.

Another vector named "Resource Imbalanced Vector" (RIV) equals $(C-H)_x + (M-H)_y + (I-H)_z$, where H is $(C+M+I)$. Based on the placement of the resource dimensions, the balance is preserved when VMs are placed on PMs in this vector [17].

$$|RIV| = \sqrt{(C-H)^2 + (M-H)^2 + (I-H)^2} \tag{9}$$

3 Related Work

Placing virtual machines (VMs) for decreasing the energy consumption while increasing the efficiency in data centers is one of the most considerable and interesting issues in the cloud computing world. Hence, numerous researches have been carried out in this field and various approaches were introduced to solve the problem accordingly. The majority of these studies attempted to find an approximate solution for VM placement problem and the obtained algorithms were categorized into three general groups including heuristic, metaheuristic, and compound.

3.1 Heuristic Algorithms

In heuristic methods, a group of constraints are highlighted to find a feasible solution for a specific problem. Offering an effectively acceptable solution, especially in a limited time is the advantage of heuristic algorithms. Further, specific heuristic algorithms are implemented more simply compared to the metaheuristic algorithms. Heuristic algorithms are performed quickly and thus are suitable for online schedules. In a literal sense, most of the heuristic algorithms employed to solve the placement problem are better than other types of greedy algorithms offered for Bin Packing problem. However, based on NP-hard placement problem, there is no guaranty that greedy algorithms can find an optimal solution. Furthermore, these algorithms are unable to solve the problem based on the multiple resources [19]. Constraint Programing (CP) is a type of logical programing and uses a group of constraints which can easily expand to have more dimensions. However, the search space is limited to the domain of PMs and PMs and thus data center is constrained by using the CP, which leads to the limitation of the search space [20].

3.2 Metaheuristic Algorithms

During recent years, many researchers have presented techniques to increase resource utilization and reduce energy consumption in cloud data centers. However, the heuristic bin packing methods suffer from local-optimal solutions. For this reason, many researchers have focused on the metaheuristic algorithms that are inspired from nature or biologically computing in order to manage the large scale problems in an efficient way [21]. The difference between heuristic and metaheuristic algorithms is that the first one is a problem-specific method while the latter one can be applied to a wide range of problems. In general, exploration and exploitation are considered two major components of the metaheuristic algorithm. The exploration component generates a solution to explore the search space on a global scale, whereas the exploitation component tends to focus on a current suitable achieved solution in the local region. In fact, the metaheuristic algorithms are mainly inspired by the nature, such as the ant colony algorithm [13], genetic algorithm [22], and particle swarm optimization [23]. Considering metaheuristic algorithms, the obtained results achieve a better solution including better energy consumption and placement compared to heuristic algorithms. However, metaheuristic algorithms need more time for finding the final solution in the large search space. Moreover, metaheuristic is generally a random process, which convergence time and solution finding depend on the nature of the problem, initialization of the experimental parameters, and method of the solution searching [24, 25]. The reordering grouping genetic algorithm (RGGA) [22] is one of the proposed methods for the placement problem. Using this algorithm leads to a lower number of physical servers. Additionally, RGGA works well for appropriate resource allocation while it imposes a lot of overheads on the search space due to lack of considering some of the solutions in every generation and thus it fails to provide a suitable running time. Simulated Annealing (SA), which is inspired by the annealing process in metallurgy, is a probabilistic single-solution-based search method where a solid is slowly cooled until its structure is eventually frozen at a minimum energy configuration. There are numerous SA variants. Experimental results demonstrated that Simulated Annealing Virtual Machine Placement (SAVMP) [26] can generate better VM placement compared to First-Fit Decreasing (FFD) [27]; however, it takes a very long time to find the best solution when the search space of the problem expands. In addition, another method presents a workload-optimized approach which is based on a Discrete Particle Swarm Optimization (DPSO) [28] in order to minimize the number of active PMs in VM placement. The main contribution of this method optimizes the host PM workload to achieve an efficiency. Similar to many studies, placement process seeks to minimize the number of active PMs. Further, each PM should be efficiently used when working under the maximum load. In [25], a Power-Aware method is proposed to determine the appropriate placement for the VMs. This Power-Aware method uses a discrete version of PSO to optimize power consumption by reducing the number of active servers along with the number of overloaded hosts.

3.3 Compound Algorithms

In the compound algorithm, heuristic algorithms are applied for initial VMs placement and VMs optimization during the computation. Thus, metaheuristic algorithms can be employed to generate a set of solutions for the first time and then heuristic algorithms are used for achieving an optimal solution based on these solutions. As a result, both running time and solutions space decrease while the complexity of implementation increases [18, 28].

In this article, the proposed method is the compound one. In various studies, cloud environment and the schedule were emphasized in cloud computing. However, by comparing the present research with the other studies, the method for placement of the virtual servers

Table 1 Comparison of VM Placement Algorithms

Criteria Algorithms	Based on	Resources considered	Strengths	Weaknesses	Performance better than
VMPACS [6]	Ant colony system algorithm	CPU, network & storage	Near-optimal solution, Energy efficient, Minimum resource wastage	High computation time	FFD, RGGA, SA
SAVMP [25]	Simulation Annealing	CPU, memory & network	Within reasonable time limit, VM placement that costs the energy very close to the low boundary	Problem size grows larger, the time needed to find the solution can be very long	First Fit Decreasing
Enhanced FFD [26]	Bin Packing	CPU	More energy efficient System throughput	Service-level agreement violation	Greedy, Round Robin & FFD algorithms
RGGA [22]	Genetic Algorithm	CPU	Reduced number of active PMs, Improved CPU utilization	The overhead of number of large searching spaces	Genetic Algorithm (genetic algorithm never solves it)
DPSO [27]	Discrete particle swarm optimization-based workload optimization	CPU, memory	Minimizing the number of active physical machines	Not having the ability to explore for better solutions. (Early convergence)	FFD, BFD, RGGA
PAPSO [28]	Discrete particle swarm optimization	CPU	Reduced the power consumption in data centers without violating SLA	Needing more resources into consideration such as memory, bandwidth, and network factors.	Power-Aware Best Fit Decreasing algorithm (PABFD)

on the physical servers is clearly highlighted in the cloud data center. The comparison of several VM placement algorithms is represented in Table 1.

4 ERBACS: Proposed Approach

4.1 Initialization

Based on the ant colony algorithm, a method is used to conclude that the consumption minimizes in the cloud computing centers. In fact, ERBACS is employed to achieve a solution with the least number of host servers for the VMs. At initialization, the quantities are first determined for the layers of the pheromone based on every VM-PM. Furthermore, a matrix called $\tau_{n \times m}$is considered for the ants set of the PM and the shuffled VM is taken into account for the quantities. Then, a possible method is employed to achieve a different solution for the ants based on a different order of VMs [18].

4.2 Finding a Solution

The feasible solution is the right placement which is proportional to the constraints of the resources and the energy use in physical servers. As shown in Fig. 2, the matrix with the dimension of $M_t \times N$ is sleeked to provide the best possible placement based on the constraints and M_t represents the number of active physical servers attempting to reduce the number of them as much as possible. In fact, M_t is different at the running time for every ant. For instance, it can be declared that if there are four servers on $P = \{P_1, P_2, P_3, P_4\}$ and every P_i has its how number of VMs which is based on the path of the ants; Fig. 2 is drawn which illustrates the moving ants as $(X_{11}, \ldots, X_{14})$. When the ants move on the node modelled as the graph, every VM is placed on the PMs one by one. Then, VMs are placed based on the movement of the ant. As N number of the steps are taken to find a solution as N number of the VMs are placed, and every step changes the accessibility of the physical servers in this placement. Therefore, based on the available resources, the eqs. (5), (6), and (7) are considered during the placing process.

The ants moved on the graphs by the impact of the pheromones and heuristic quantities. In ERBACS algorithm, pheromones are considered in VMs and VM-PMs. Pheromones had an impact on placing VM_j and VM_k or placing VM_j on server S_i. The following equation is used to initialize the amount of pheromone for placing VM_j on S_i.

$$\tau(i,j)=\left\{\frac{1}{|S_i|}\sum_{k\in Si}\tau\alpha(k,j),\quad if\ S_i\neq 0\ \tau_0\ ,\qquad otherwise\right. \tag{10}$$

where, $S_i$denotes the existing VMs set on the server i and $|S_i|$ represents the number of VMs and is dependent on the server i [9].

Ant colony optimization algorithm requires heuristic information by the pheromones, along with other heuristic information in order to find a better option by the local searching strategies. Such information is supposed to decrease the number of active servers and to make the resource allocation reach the highest level of allocation in a server. Thus, heuristic information is considered such that it could balance the distribution of the resources. The scale for this information is based on the balance in allocated resources between the VM placement and the servers.

$$\eta_{ij}=b\times|\mathrm{RIV(v)}_p|+(1-\mathrm{b})\times\left(\sum_{r\in R}RUV_p^r+RDV_p^r\right) \tag{11}$$

where, β represents a non-negative integer parameter and the importance of the pheromones compared to the heuristic information.

$$F_s=\{S_i|S_i\ true\ in\ Eqs.(5),(6)\ and\ (7)\} \tag{12}$$

F_s denotes a group of physical servers based on the conditions in eqs. (5), (6), and (7).

Two methods were used for selecting server i for VM_j in the F_s set. Therefore, parameter q is first initialized in the range of 0 and 1. Then, a random number called q_0 is generated in the range of 0 and 1.

The server is selected if the random number is less than parameter q based on the pheromone scale and heuristic information. Otherwise, the server is selected based on the roulette wheel selection which is based on probability of distribution of the ants which choose the path [18].

$$i=\begin{cases}\mathrm{arg}max_{l\in Fs_j}\tau(l,j)\times\eta(l,j), & if\ q_0\leq q\\ roulette\ wheel\ selection\ from\ Fs_j\ , & otherwise\end{cases} \tag{13}$$

4.3 Fitness Function

After finding the solution, a measure had to be provided for evaluating the solution. The main aim of placing VMs on the physical servers in ERBACS is based on two constrains since, based on all the dimensional resources required by VMs, the most possible capacity of the resource is allocated based on the form of the placement in PMs, and the amount of energy use decreased as low as possible.

$$f_1(ant.solution)=\begin{cases}\sum_{i=1}^{M_t}y_i & ,if\quad x_{ij}\geq 1\\ M_t+1\ , & otherwise\end{cases} \tag{14}$$

$$f_2(ant.solution)=\sum_{i\in y_i}ECV(i) \tag{15}$$

In fitness function, M_t represents the number of servers required for placing at the time of t. Moreover, y_i equals to one when the server i is active while it is equal to zero when it is inactive. Therefore, f_1 indicates all the active servers for a solution, the number of which should not exceed that of the M_t. Additionally, the number of f_1 equals to M_t+1 if no placement can be found for the solution since no solution is found for this placement [18].

Fitness function f_2 displays the amount of energy consumption for a solution and it computes all amounts of energy use in active physical servers based on ECV vector. Thus, f_1 is used in ERBACS computation for evaluating the solutions. However, f_2 is employed as the measure of elevation if f_1 is the same in both different solutions. In fact, using the second evaluation function improved the process of exploring the search space when the search process for the best assignment is faced with the convergence of solutions. These two functions of evaluation aim to determine the number of active servers and their energy consumption in order to achieve a more accurate difference between the solutions.

4.4 Updating Pheromones

The pheromones on the path indicated the information in the history and behavior of the ant. The amount of the pheromone on the path altered after the ant moved on the graph modelled and built the solutions. Therefore, the new amount had to be computed due to the change of this amount. The amount of pheromone can be locally or globally updated with different impacts on searching behavior of the ant.

Local updating made the other ants go through the new paths for resource allocation while global updating forced the ants go through a harbinger path in order to find better solutions. The equation below is used for the local update.

$$\tau(k,j) = (1-\rho)\times\tau(k,j) + \rho\times\tau_0 \tag{16}$$

After providing a solution for every ant, local update is given for VMs [15]. Equation (17) is applied to the global update:

$$\tau\alpha(i,j) = (1-\rho)\times\tau\alpha(i,j) + \rho\times\Delta\tau\alpha_{ij} \tag{17}$$

The parameters ρ and Δτ represent the pheromone evaporated from the path and the pheromone added to the path, respectively [18].

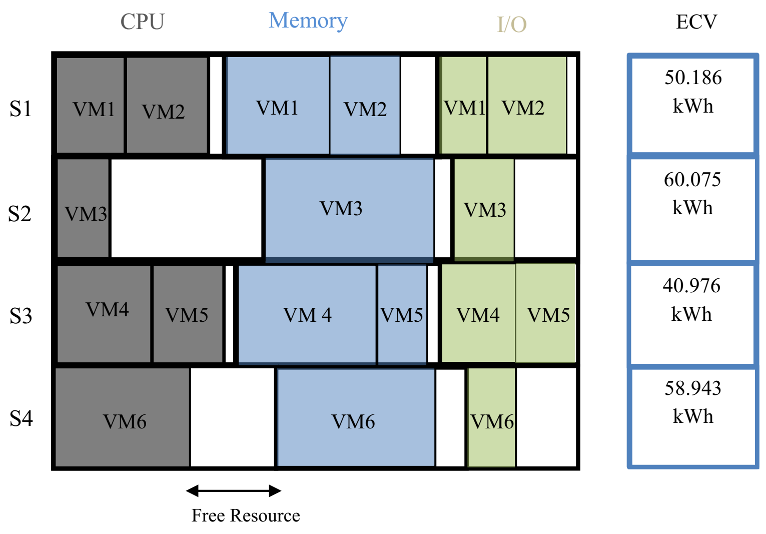
$$\Delta\tau\alpha_{ij} = \frac{nVM_j}{nActivePM_i} + \frac{1}{NRC_i + NRM_i + NRI_i + 1} \tag{18}$$

In the above equation, *nVM* is the number of placed VMs and *nActicePM* denotes the number of active physical servers. In addition, NRI, NRM, NRC represent normalized remaining I/O, memory, and CPU on the server *i*, respectively. Finally, global updating was performed when all the ants reached their own solution and after each iteration based on the best solution [15].

4.5 Local Energy Refining Balance

It is a vivid fact that cloud load balancing deals with a great amount of distributing workloads across multiple computing resources. For this reason, a suitable balancing method is able to reduce energy consumption and maximizes the availability of resources [29, 30]. In ERB approach, the server is actively placed based on the amount of energy consumption. This approach is used at the end of every iteration in the ERBACS algorithms. Further, it is employed if there is no change in the amount of fitness for the best solution *ch* times. Furthermore, ERB approach worked such that if *h* number of ants had a solution with the same amount of f_1 fitness function, then one ant is selected and the energy

Fig. 3 Local refining energy balance scheme



consumption of that solution is refined based on the amount of the second fitness for each of them by using the roulette wheel method.

Moreover, the physical server with the greatest amount of energy use is selected based on ECV vector in order to refine the energy consumption in the selected solution. Then, the physical server with the least amount of allocated resources is selected based on RUV vector. Next, VMs are placed on the servers with considerable use from the least to the greatest, respectively, when the two physical servers are selected. Additionally, VMs were swapped by the servers which had the least amount of allocated resources based on the amount of the used CPU. This was repeated to break the conditions in eqs. (5), (6), and (7) otherwise, VMs failed to be swapped (Fig. 3).

This method is only applied on one solution since it is undesirable to be run on all the solutions by considering the time. Therefore, it is attempted to use this method in special situations.

5 ERBACS Algorithm

ERBACS algorithm is a method on ant colony aiming at decreasing the energy consumption in the cloud centers. Figure 4 displays the stages of the algorithm in details.

The initialization, which includes the initial pheromone on the path, is considered as τ_0 and the number of ants in every iteration, which is regarded as *m* is computed; in addition, *g* represents each ant. Finally, parameter *t* denotes that the number of iterations in the algorithm equals to 1 at this stage. Further, m_t=m_{min}−1 and m_{min} represents the least number of active physical servers and $m_t$represents *t* number of active physical servers. Then, feasible solutions are produced. To this end, VMs are randomly shuffled and then, respectively selected to be placed.

VMs are placed based on the available physical servers and their capacity, as well as the required resources. Furthermore, the constraints of the problems, along with placing one VM is considered on only one physical server.

Moreover, placement on the server is performed when the capacity of the physical servers was large enough to be filled. Then, the next VMs are placed on the next physical servers after the server is filled. Next, the servers are evaluated after building *m* number of paths. Additionally, the local pheromone is updated in the next stage. Afterward, the best solution is considered as Global Best Solution (GBS) after initializing and evaluating all the ants. In addition, m_t in the fitness function is updated. Then, the condition of using the refining energy consumption method is evaluated. Further, this method is applied for placement refining if GBS failed to change the *ch* time. Furthermore, general pheromones are updated. Moreover, GBS is evaluated to find whether the updated GBS is used for updating the general pheromone if ERB approach is applied before this stage. Afterward, the final condition is processed; the algorithm stopped working when the maximum number of iterations is computed, otherwise, $t = t + 1$, and it goes back to the second stage for the last iteration.

Algorithm 1 indicates the high-level architecture of our presented algorithm in more details. The detailed pseudocode of our energy refine balancing method is shown in Algorithm 2.

5.1 Time Complexity Analysis of ERBACS Algorithm

In general, the computational complexity of the ERBACS reliant on the computational complexity of ACO and the local energy refining balance method. In this subsection it is assumed that: n is the total number of PMs; m is the number of VMs; A is the number of ants; T is the total number of iterations. The time complexity of each part in the algorithm is defined as follows:

- Initializing ERBACS all k parameters: O (k).
- Producing the list of available PMs: O (n).
- Initializing the iteration number: O (1).
- Ensuring that placemet is in the valid range: O (n × m).
- Calculating the fitness function for each ant: O (m × n).
- Energy refining balance: roulette wheel selection: O (A^2) + swapping the VMs: O (*m*)
- Incrementing the iteration number: O (T).
- Producing the final placemet: O (m)

In each iteration, quicksort algorithm (complexity of O(m × log m)) was used to rank the ants. As a consequence, the total time complexity for the presented algorithm is O (T × A × m).

Algorithm 1. Pseudocode of ERBACS

```
Input:
        Set of VMs with resource demand, set of PMs with capacity constraint and set of the
        parameters
Output:
        Placement Graph (represented as a binary matrix)

1.  Initializing parameters: $q_0$, $\rho$, $\varepsilon$, $\beta$, $h$, $ch$
2.  Initialize all pheromone values
3.  Producing a list of the available hosts, Descending sorting for the VMs according to CPU
    utilization
4.  Make feasible solutions:
5.      for each ants do
6.          for each PM do
7.              Select a new individual VM from set of VMs
8.          end for
9.  Evaluate both fitness function for each ants
10.     end for
11. Repeat
12. Local pheromone Update (Eq. (16))
13. Find current iteration global best ant
14. Check the improvement of the global best solution
15. Refined global best solution ←Energy Refine Balancing Method ()
16. Global pheromone Update (Eq. (17))
17. until the maximal number of Iterations is reached
18. return Placement Graph;
```

6 Implementation and Comparisons

In this section, implementations are taken into consideration in order to evaluate the implementation of ERBACS algorithm and to compare it with the other methods. Additionally, every algorithm is implemented in MATLAB software (2017b) and is

Algorithm 2. Pseudocode of Energy Refine Balancing Method

```
Input:
        Set of ants
Output:
        Refined global best solution

1. Select one ant by using roulette wheel method
2. Repeat
3.     Refine resource allocation for selected ant with regard to ECV and RUV vectors
4.     Swap VMs
5. until break one of conditions in (Eq. (5,6,7))
6. return Refined global best solution
```

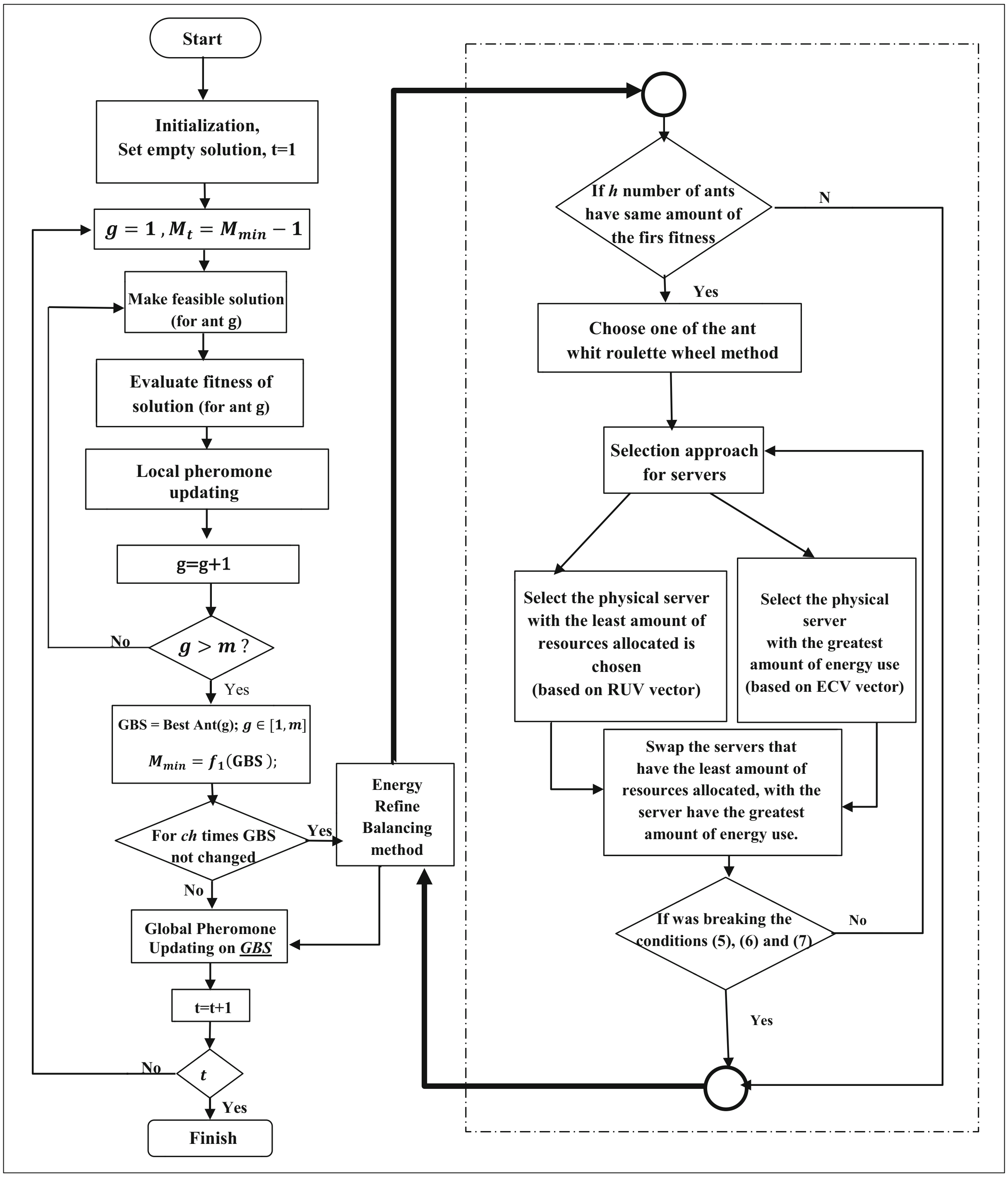

Start
Initialization,
Set empty solution, t=1
$g = 1, M_t = M_{min} - 1$
Make feasible solution (for ant g)
Evaluate fitness of solution (for ant g)
Local pheromone updating
g=g+1
$g > m$?
No
Yes
GBS = Best Ant(g); $g \in [1, m]$
$M_{min} = f_1(\text{GBS})$;
For ch times GBS not changed
Yes
No
Energy Refine Balancing method
Global Pheromone Updating on GBS
t=t+1
t
No
Yes
Finish
If h number of ants have same amount of the firs fitness
N
Yes
Choose one of the ant whit roulette wheel method
Selection approach for servers
Select the physical server with the least amount of resources allocated is chosen (based on RUV vector)
Select the physical server with the greatest amount of energy use (based on ECV vector)
Swap the servers that have the least amount of resources allocated, with the server have the greatest amount of energy use.
If was breaking the conditions (5), (6) and (7)
No
Yes

Fig. 4 Flowchart of the ERBACS algorithm

run in a system with processing details of RAM 16GB, Intel Core i5.

Two experimental sets are considered from one dataset to compare the methods of VM placement problem (http://gwa.ewi.tudelft.nl/fileadmin/pds/trace-archives/grid-workloads-archive/datasets/gwa-t-12/fastStorage.zip). The server environment is taken into account as the real heterogeneous world in both sets. In addition, the number

Table 2 The initial parameters employed in scenarios A and B

Algorithms	Parameters	Values
ERBACS	q_0	0.2
	ρ	0.1
	ε	0.1
	β	2.0
	h	5
	ch	7
	Maximum iteration	60
SAVMP	Initial temperature	0.99
	Minimum difference between solutions	0.01
	Initial acceptance probability	1
	Maximum iteration	60
RGGA	Crossover operator probability	0.8
	Mutation operator probability	0.1
	Weights	Between 20,000 and 35,000
	Maximum iteration	60
DPSO	c1 = c2	1.49445
	ω	0.7
	Vmax	0.8
	Maximum iteration	60
VMPACS	ρ	0.1
	ε	0.1
	β	2.0
	Maximum iteration	60

of VMs to be placed is regarded as the difference between the two sets. Therefore, five samples from the first set with less number of VMs to be placed are displayed as S1 to S5 while two samples from the second set with more VMs for placement are represented as B1, and B2. Furthermore, the setting of parameters in the proposed approach had a direct effect on the algorithm performance. Preliminary experiments which determined the parameter values and the final parameters are considered as q_0=0.2, $\rho = 0.1$, $\varepsilon = 0.1$, $\beta = 2.0$, $h = 5$, $ch = 7$, and maximum iteration $T = 60$.

Table 2 lists the initial parameters employed in scenarios A and B. In both scenarios, the number of individual solutions is the same in all considered algorithms (50).

6.1 Scenario a: Small Scale Environments

As shown in Table 3, ERBACS is compared with SAVMP [26], RGGA [22], FFD [31], BFD [31], DPSO [28], and VMPACS which is the basic method considered for the proposed algorithm. Three dimensions are considered for CPU, memory, and I/O resources. The same running time is set for each of above-mentioned resources in order to fairly compare all the metaheuristic methods. The comparison criterion in Scenario A is based on the number of Active Physical Machines (APMs) with respect to the placement. Moreover, Fig. 5 presents experimental results comparisons in this scenario. Furthermore, it is useless to consider a fixed time period for BFD and FFD heuristic algorithms since they quickly converged. Therefore, the running time is taken until they are implemented. The number of active physical servers is considered in order to evaluate the proposed solution by every compared method. As shown in Table 3, 100–800 VMs are used in five speared experiments to compare the results. A total of 800 heterogeneous PMs are used in all the experiments. In samples S1 and S2, the results are not considerably different from each other; however, they are useful when the environment of the experiment expands.

Table 3 Results Comparison in Small Scale Environments

Experiment instances	#VM	ERBACS		SAVMP [25]		RGGA [22]		FFD [32]		BFD [32]		DPSO [27]		VMPACS [6]	
		Time(s)	APMs	Time (s)	APMs	Time (s)	APMs	Time (s)	APMs	Time (s)	APMs	Time (s)	APMs	Time (s)	APMs
S1	100	**30**	36.0	**30**	48.0	**30**	85.0	2.41	54.0	2.98	59.0	49	**30**	**30**	36.0
S2	200	**30**	71.0	**30**	81	**30**	167	6.88	96.0	7.04	101	80	**30**	**30**	69.0
S3	300	**60**	106	**60**	255	**60**	289	7.26	145	7.09	164	123	**60**	**60**	110
S4	600	**60**	141	**60**	501	**60**	567	10.72	179	11.34	182	161	**60**	**60**	150
S5	800	**60**	281	**60**	779	**60**	800	18.35	350	14.9	387	324	**60**	**60**	320

Moreover, both simulated annealing and genetic algorithms require more time to achieve better solutions when the environment of the experiment expands. Additionally, they find unsuitable answers compared to FFD algorithm which has easier implementation. The ant colony algorithm archives better results for the placement problem by this expansion and therefore, it is better to find the probabilities for a better amount of pheromone and better evaluation of the ants. Although ACO is a suitable method for achieving good results, ERBACS algorithm and the refining method of the global best solution find better results.

6.2 Scenario B: Large Scale Environments

Scenario A, as the comparison, was compared on less number of heterogeneous servers while scenario B, as the experimental samples, had a large number of VMs and the servers are heterogeneous. By this comparison, the amount of energy waste is taken into account, along with considering the number of active PMs. Given the comparison, the Table 4 indicates that the ERBACS algorithm is considerably successful in decreasing the amount of energy waste in the data center. In addition, running time of the ERBACS algorithm is quite slow compared to that of the ACO algorithm due to adding some computational functions and thus the result is acceptable. Figure 6 illustrates the amount of decreased energy waste based on the passing of generations. As shown, at first, ERBACS algorithm has the same answers as ACO algorithm while energy refining balance makes the solutions better and decrease the amount of energy waste after passing of the generations and the iteration of the global best solutions in several specific generations. The convergence curves on experiment instances B1 and B2 are depicted in the Fig. 7.

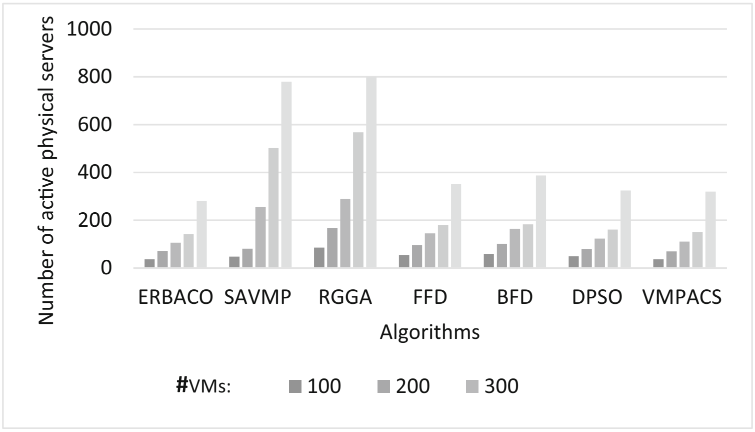


Fig. 5 Experimental results comparisons in scenario A

Table 4 Results Comparison in Large Scale Environments

Experiment instances	**# VM**	**ERBACS**			**VMPACS [6]**			**DPSO [27]**		
		Time (Min)	APMs	Energy wastage (kWh)	Time (Min)	APMs	Energy Wastage (kWh)	Time (Min)	APMs	Energy Wastage (kWh)
B1	3000	7.39	712	532.002	6.18	720	588.747	6.84	1389	745.263
B2	4000	11.02	952	542.335	10.66	961	592.418	10.89	1635	894.364

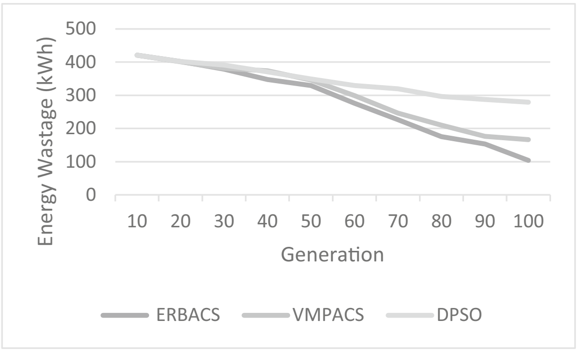


Fig. 6 Comparison energy wastage between ERBACS, VMPACS [6] and DPSO [28]

7 Conclusion and Future Work

In general, energy consumption has the greatest cost in cloud computing. Further, placing VMs on the physical servers has a significant effect on the energy use. Thus, it was attempted to optimize the energy use by considering the best placement. With the increasing number of users who need computing resources, a large number of real-life optimization problems have raised. Using Metaheuristic algorithms has shown an increasing interest in the scientific community, which provides robust and efficient ways to address optimization problems in the actual application. Accordingly, based on the ACO system, ERBACS algorithm was presented in this paper to solve the NP-hard problem. This algorithm aimed to place VMs by considering the least number of active physical servers with the least amount of energy consumption and the placement was performed by artificial ants. Furthermore, based on the available pheromones on the path and other heuristic quantities related to the Virtual Machines placement problem, suitable solutions were gradually achieved by considering the probability distribution function which may be used to define a particular probability distribution.

The best solutions found by the ants in every generation were selected to refine the energy consumption due to the energy refining balance approach considered in ERBACS, and therefore this approach provided a better placement with less amount of the consumed

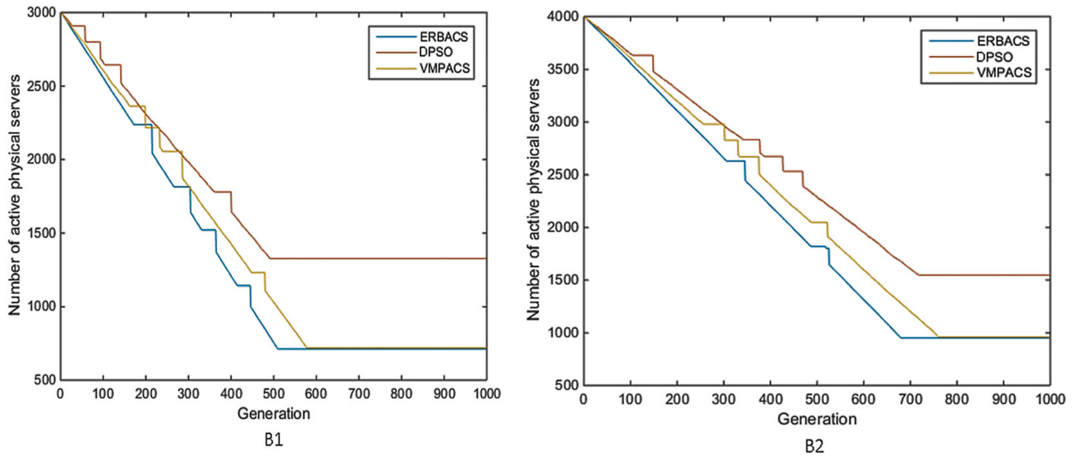


Fig. 7 The convergence curves of ERBACS approach on B1 and B2

energy. Another strength point regarding applying this approach is that, it can be used to swap on the new requirements after providing the placements. In most cloud centers, new VMs are added to the centers in every time period and accordingly, it is possible to use energy refining approach only based on the last placement and the amount of the required resources. As it is clearly illustrated in the compresence, based on the quality of the placement, the running time in this algorithm is less complicated compared to the other algorithms.

The results of the comparison indicated that large-scale ERBACS in the environments obtained superior results with the least amount of energy waste compared to the other heuristic and metaheuristic methods. Moreover, based on results, the ERBACS algorithm in a large environment considerably succeeded in decreasing the number of active physical servers and the amount of energy waste. Therefore, ERBACS algorithm is considered a suitable and useful approach for solving VM placement problem and it attempts to have the least effects on the environment and energy consumption by using the computing device. In our upcoming research, we will attempt to provide a temperature and energy-aware VM placement algorithm by using combinatorial metaheuristic algorithms with other objective functions that make the algorithm much green as well.

Acknowledgements The authors would like to express their thanks to the anonymous referees for their valuable comments and suggestions that improved the paper.

Data Availability Data sharing not applicable to this article.

References

1. Zhang, Q., Cheng, L., Boutaba, R.: Cloud computing: state-of-the-art and research challenges. J. Internet Serv. Appl. **1**(1), 7–18 (2010)
2. Stergiou, C., Psannis, K.E., Kim, B.G., Gupta, B.: Secure integration of IoT and cloud computing. Futur. Gener. Comput. Syst. **78**, 964–975 (2018)
3. Manasrah, A.M., Gupta, B.B.: An optimized service broker routing policy based on differential evolution algorithm in fog/cloud environment. Clust. Comput. **22**(1), 1639–1653 (2019)
4. Bhushan, K., Gupta, B.B.: Distributed denial of service (DDoS) attack mitigation in software defined network (SDN)-based cloud computing environment. J. Ambient. Intell. Humaniz. Comput. **10**(5), 1985–1997 (2019)
5. Al-Qerem, A., Alauthman, M., Almomani, A., Gupta, B.B.: IoT transaction processing through cooperative concurrency control on fog–cloud computing environment. Soft. Comput. **24**(8), 5695–5711 (2020)
6. Gao, Y., Guan, H., Qi, Z., Hou, Y., Liu, L.: A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. J. Comput. Syst. Sci. **79**(8), 1230–1242 (2013)
7. Tabrizchi, H., Kuchaki Rafsanjani, M.: A survey on security challenges in cloud computing: issues, threats, and solutions. J. Supercomput. **76**(12), 9493–9532 (2020)
8. Tabrizchi, H., Kuchaki Rafsanjani, M., Emilia Balas, V.: In: Balas, V.E., et al. (eds.) Multi-task scheduling algorithm based on self-adaptive hybrid ICA–PSO algorithm in cloud environment, Part of the Advances in Intelligent Systems and Computing book series, pp. 422–431. AISC 1222 Springer Nature, Switzerland (2021)
9. López-Pires, F., Barán, B.: Many-objective virtual machine placement. J. Grid Comput. **15**(2), 161–176 (2017)
10. Békési, J., Galambos, G., Kellerer, H.: 5/4 linear time bin packing algorithm. J. Comput. Syst. Sci. **60**(1), 145–160 (2000)
11. Van Laarhoven, P.J., Aarts, E.H.: Simulated annealing, in Simulated annealing: Theory and applications, pp. 7–15. Springer, Netherlands (1987)
12. Deb, K.: An introduction to genetic algorithms. Sadhana. **24**(4–5), 293–315 (1999)
13. Dorigo, M., Birattari, M.: Ant colony optimization, in Encyclopedia of machine learning, pp. 36–39. Springer, US (2017)
14. Kansal, N.J., Chana, I.: Energy-aware virtual machine migration for cloud computing - a firefly optimization approach. J. Grid Comput. **14**(2), 327–345 (2016)
15. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. **1**(1), 53–66 (1997)
16. Khanna, G., Beaty, K., Kar, G., Kochut, A.: Application Performance Management in Virtualized Server Environments, pp. 373–338. IEEE/IFIPNOMS 2006, Vancouver (2006)
17. Ferdaus, M.H., Murshed, M., Calheiros, R.N., Buyya, R.: Virtual Machine Consolidation in Cloud Data Centers Using ACO Metaheuristic, pp. 306–317. Euro-Par, Grenoble, France (2014)
18. Liu, X.-F., Zhan, Z.-H., Deng, J.D., Li, Y., Gu, T., Zhang, J.: An energy efficient ant colony system for virtual machine placement in cloud computing. IEEE Trans. Evol. Comput. **22**(1), 113–128 (2018)
19. Zhang, Y., Ansari, N.: Heterogeneity Aware Dominant Resource Assistant Heuristics for Virtual Machine Consolidation, pp. 1297–1302. IEEE GLOBECOM, Atlanta (2013)
20. Dhyani, K., Gualandi, S., Cremonesi, P.: A Constraint Programming Approach for the Service Consolidation Problem, pp. 97–101. CPAIOR, Bologna (2010)
21. Aryania, A., Aghdasi, H.S., Khanli, L.M.: Energy-aware virtual machine consolidation algorithm based on ant Colony system. J. Grid Comput. **16**(3), 477–491 (2018)

22. Wilcox, D., McNabb, A., Seppi, K.: Solving Virtual Machine Packing with a Reordering Grouping Genetic Algorithm, pp. 362–369. IEEE CEC, New Orleans (2011)
23. Kennedy, J.: Particle swarm optimization, in Encyclopedia of machine learning, pp. 760–766. Springer, US (2017)
24. Scarpiniti, M., Baccarelli, E., Naranjo, P.G.V., Uncini, A.: Energy performance of heuristics and meta-heuristics for real-time joint resource scaling and consolidation in virtualized networked data centers. J. Supercomput. **74**(5), 2161–2198 (2018)
25. Ibrahim, A., Noshy, M., Ali, H.A., Badawy, M.: PAPSO: a power-aware VM placement technique based on particle swarm optimization. IEEE Access. **8**, 81747–81764 (2020)
26. Wu, Y., Tang, M., Fraser, W.: A Simulated Annealing Algorithm for Energy Efficient Virtual Machine Placement, pp. 1245–1250. IEEE SMC, Seoul (2012)
27. Alahmadi, A., Alnowiser, A., Zhu, M.M., Che, D., Ghodous, P.: Enhanced first-fit decreasing algorithm for energy-aware job scheduling in cloud, vol. 2, pp. 69–74. CSCI'14, Las Vegas (2014)
28. Yan, J., Zhang, H., Xu, H., Zhang, Z.: Discrete PSO-based workload optimization in virtual machine placement. Pers. Ubiquit. Comput. **22**(3), 589–596 (2018)
29. Ghobaei-Arani, M., Souri, A., Rahmanian, A.A.: Resource management approaches in fog computing: a comprehensive review. J. Grid Comput. **18**(1), 1–42 (2019)
30. Hosseinzadeh, M., Ghafour, M.Y., Hama, H.K., Vo, B., Khoshnevis, A.: Multi-objective task and workflow scheduling approaches in cloud computing: a comprehensive review. J. Grid Comput. **18**(3), 327–356 (2020)
31. Chen, M., Zhang, H., Su, Y.-Y., Wang, X., Jiang, G., Yoshihira, K.: Effective VM Sizing in Virtualized Data Centers, pp. 594–601. IFIP/IEEE IM, Dublin (2011)