Computer Communications 214 (2024) 136–148

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

A Utilization Based Genetic Algorithm for virtual machine placement in cloud systems

Mustafa Can Çavdar *, Ibrahim Korpeoglu, Özgür Ulusoy

Department of Computer Engineering, Bilkent University, Ankara, Turkey

ARTICLE INFO

Keywords:
Cloud computing
Virtualization
Genetic algorithm
Virtual machine placement

ABSTRACT

Due to the increasing demand for cloud computing and related services, cloud providers need to come up with methods and mechanisms that increase the performance, availability and reliability of data centers and cloud systems. Server virtualization is a key component to achieve this, which enables sharing of resources of a single physical machine among multiple virtual machines in a totally isolated manner. Optimizing virtualization has a very significant effect on the overall performance of a cloud computing system. This requires efficient and effective placement of virtual machines into physical machines. Since this is an optimization problem that involves multiple constraints and objectives, we propose a method based on genetic algorithms to place virtual machines into physical servers of a data center. By considering the utilization of machines and node distances, our method, called Utilization Based Genetic Algorithm (UBGA), aims at reducing resource waste, network load, and energy consumption at the same time. We compared our method against several other placement methods in terms of utilization achieved, networking bandwidth consumed, and energy costs incurred, using an open-source, publicly available CloudSim simulator. The results show that our method provides better performance compared to other placement approaches.

1. Introduction

Cloud computing has become a pervasive technology for providing computing, storage, and software services on the Internet. This is reasoned by the fact that cloud systems can provide nearly any type of service customers may need and relieve customers from building their own physical infrastructures. It can also offer services with the pay-as-you-go charging model where customers pay according to the amount of the services they use, which makes it even more attractive. Those services can be at different levels, such as Infrastructure-as-a-Service (IaaS), Software-as-a-Service (SaaS), and Platform-as-a-Service (PaaS). There are various commercial cloud computing platforms such as Amazon Web Services, Google Cloud Platform, and Microsoft Azure. There are also open-source cloud platforms such as Openstack, OpenShift, CloudStack, and Cloudify, which can be used to offer various cloud services to users.

Due to the increasing demand for cloud services, optimization of resource consumption becomes even more essential to increase the performance, availability and reliability of cloud systems. Virtualization technologies have been proven to be very useful to meet these requirements. Virtualization enables users and applications to share physical cloud resources in an efficient and effective manner. With server virtualization, multiple virtual machines can run on a single physical machine in a totally isolated manner. In this way, cloud providers can serve more customers in a flexible and efficient manner.

Different virtual machines requested by users may have different processing, memory, I/O and networking requirements. Physical servers can also have different capacities. This leads to an optimization problem known as *virtual machine placement*. Solutions provided to this problem aim to increase the utilization of physical machines to reduce costs and energy consumption. Due to the increasing demand for cloud computing services mentioned earlier, optimization of resource usage becomes a very essential issue to save more energy, reduce costs, and meet customer service level agreements (SLAs).

The virtual machine placement problem is a multi-objective constrained NP-hard problem as shown by [1]. If the number of virtual machines is n and the number of physical machines is m, then the number of possible mappings of virtual machines to physical ones is m^n. Despite this complexity, choosing the optimal or near optimal physical machine to host a virtual machine is highly critical [2]. Choosing the appropriate physical host has many benefits for cloud owners such as reducing power consumption, increasing resource utilization, and providing better QoS.

The combinatorial nature of the virtual machine placement problem and the number of possible solutions can make this problem

* Corresponding author.
E-mail address: mustafa.cavdar@bilkent.edu.tr (M.C. Çavdar).

https://doi.org/10.1016/j.comcom.2023.11.028
Received 31 August 2021; Received in revised form 21 July 2023; Accepted 27 November 2023
Available online 29 November 2023

computationally infeasible to find an exact optimal solution within a reasonable amount of time. Finding an exact optimal solution for complex optimization problems typically requires exhaustive search techniques, such as integer programming or constraint satisfaction algorithms. These approaches can be computationally expensive and time-consuming. Genetic algorithms, on the other hand, provide a trade-off between time and solution quality. They can converge to reasonably good solutions in a relatively short amount of time, making them suitable for real-time or near-real-time decision-making scenarios.

In this paper, we propose a genetic algorithm based solution to the virtual machine placement problem, which uses a novel fitness function and chromosome structure, and considers resource utilization, network bandwidth usage, and energy costs at the same time. Our method is called Utilization Based Genetic Algorithm (UBGA) and aims to find a close-to-optimum solution. We designed our chromosome structure as a tree representing a data center network topology. The leaves of the tree represent the physical machines, and each leaf node has a pointer to a list of virtual machines. The fitness function of our genetic algorithm is designed uniquely to combine multiple objectives together.

We integrated our algorithm into the publicly available CloudSim [3] cloud computing simulator and conducted extensive simulation experiments to evaluate it. We compared our algorithm against a number of other approaches, including a random strategy, the default allocation method of CloudSim simulator, the First Fit Decreasing (FFD) method, and AFED-EF method from the literature [4]. Our experiments consider homogeneous, heterogeneously deterministic and uniform random distribution of resource capacities and demands. The results show that our method is completely superior to random and default CloudSim allocation methods, achieves better performance compared to FFD, and shows similar or better performances compared to AFED-EF for various evaluation metrics.

The organization of this paper is as follows. In Section 2, we provide a summary of the related work. Section 3 provides the problem formulation. In Section 4, we present the details of our virtual machine placement algorithm. In Section 5, we present the results of our extensive simulation experiments and compare our algorithm against other methods. Finally, in Section 6, we conclude the paper.

2. Related work

Evolutionary algorithms such as genetic algorithms can be used in solving resource allocation and assignment problems in cloud computing as in [5], which proposes an algorithm for data replica placement both within a data center and among a number of data centers. There are evolutionary approaches to solve the virtual machine placement problem as well. Various methods have been proposed to find an optimal solution considering a number of metrics. [6] propose a family genetic algorithm approach that is integrated into CloudSim. The algorithm divides the process among different families running in parallel to assign virtual machines to physical machines. They made the mutation probability dynamic depending on some parameters and they compare their algorithm with existing allocation policies of CloudSim. [1] describe a multi-purpose genetic algorithm, which is called Group GA, that considers virtual machine placement as a bin-packing problem. In [7], the authors propose a variant of the Binary Swarm Particle Optimization algorithm for virtual machine placement to reduce energy consumption and increase resource utilization. They consider the memory and CPU utilization of physical machines. [8] present an evolutionary algorithm to deal with the virtual machine placement problem. They experimented under both simulations and real environments and show that their method reduces energy consumption and improves overall profit. Zhang et al. [9] deal with VM migration problem in self-driving domain. They model the problem as a bin packing problem and solve that with a cluster-based genetic algorithm. Ghetas [10] proposes MBO-VM, a Monarch Butterfly Optimization to solve the VM placement problem in cloud data centers. The proposed method aims to maximize packaging efficiency and decrease the number of active physical hosts. Zhou et al. [11] describe Multi-objective Task Scheduling Strategy (MTSS) which is a differential evolution algorithm based on clustering. MTSS aims to reduce load balancing, task execution cost and task completion time.

The work in [12] is a hybrid approach of two genetic algorithms to minimize resource wastage for each server. [13] describe an evolutionary based game theory approach for the dynamic virtual machine placement problem. They consider the initial mapping of virtual machines to physical ones and create a list of live migrations of virtual machines. [14] propose a two-phase scheme that considers the optimization of power consumption, economical revenue, resource utilization, and reconfiguration time. The scheme works under the uncertainty of several relevant parameters. IADE [15] is a differential evolution algorithm that reduces the energy cost of communication and computations realized in networked data centers. In [16], authors present an Ant Colony based method for online VM placement in cloud systems that minimizes the total energy cost of physical machines. [17] design a VM placement algorithm based on an improved genetic algorithm. They aim to minimize the number of active physical machines to reduce power consumption and improve the balance of used resources of physical hosts. [18] propose a krill herd based algorithm to solve VM placement problem by considering resource utilization and power use and compare their work against First Fit Decreasing and an existing Ant Colony based model.

Masoudi et al. [19] propose a Particle Swarm Optimization based algorithm to reduce power consumption and maximize load balancing in cloud data centers. Li et al. [20] model VM placement as a multi-dimensional bin packing problem. They describe a Particle Swarm Optimization algorithm that is based on sine and cosine perturbation and reverse learning to reduce resource consumption. Kiani and Khayyambashi [21] describe an abstraction model to estimate data center power consumption and propose a Chemical Reaction Optimization algorithm to reduce power consumption in clouds. Nabavi et al. [22] present TRACTOR, an ant colony based algorithm that minimizes the network traffic between communication virtual machines and power dissipation at switches and hosts in the data center. Balaji et al. [23] propose a Firefly Algorithm to minimize the number of physical hosts in use. Therefore, they reduce the overall power consumption in the cloud data center.

Xing et al. [24] propose ETA-ACO which is an Ant Colony Optimization algorithm to minimize jointly power consumption at physical hosts and switches and overall network bandwidth traffic. They improve the performance of their algorithm with three schemes. Saxena et al. [25] describe a secure VM placement framework and propose a Whale Optimization Genetic Algorithm to reduce resource wastage and energy consumption. Salami et al. [26] propose a cuckoo search algorithm with newly developed cost and perturbation functions to minimize the number of non-idle physical servers in a cloud data center.

There is also a considerable amount of work on resource allocation in cloud computing that do not involve evolutionary algorithms. Deng et al. [27] deal with the VM placement in distributed clouds. They propose an M/M/1 queuing model to minimize both total and maximum latency. In [28], it is aimed to increase the revenue of the cloud owners by considering virtual machine migrations among physical machines. [29] describes a virtual machine placement algorithm that considers computational resources, Quality of Service metrics, and I/O data by using a priority based queuing model to minimize overall job completion time and maximize the throughput of cloud links. [30] aim to minimize the number of active hosts by assigning ranks to virtual machines and placing the machines based on their ranks. They introduce a new metric, resource usage factor, and use this metric to maximize resource utilization of physical machines. Their model improves the utilization in a balanced manner and reduces the number of migrating virtual machines. In [31], authors present a method for both flow management and VM placement in hybrid

clouds. They introduce a threshold-based flow placement algorithm, then they suggest a set of VM placement algorithms to improve traffic locality. Zhou et al. [32] propose MCEC that considers virtual machine reconfiguration and energy consumption in virtualized networked data centers.

The algorithms introduced in [33] for placement of virtual machines in cloud data centers have the objective of maximizing a new metric named satisfaction, which reflects the relative suitability of a physical machine for any virtual machine to assign to it. [34] propose a network-aware placement algorithm based on empirical estimation of required bandwidth for communication among virtual machines. [35] present an algorithm that uses a model predictive control to devise optimal maps between physical and virtual machines. In [36], the authors consider peak workload characteristics of virtual machines and model them using a mathematical method. [37] aims to deal with the virtual machine placement problem in environments that have psychical machines with multiple processor cores. Omer et al. [38] provide a traffic and power aware approach to solve the VM placement problem in cloud data centers. Their method aims to minimize power consumption and resource wastage.

[39] describe a method for online VM placement problem. Their algorithm requires the network topology as input and calculates the migration cost of VMs and decides reserved resources of physical machines to prevent SLA violations. As a result, they effectively reduce time spent for VM migrations and energy consumption. [40] propose an energy efficient VM placement method. They aim to minimize the energy cost of physical machines by using four algorithms based on best fit decreasing. The work by [41] considers load balancing in cloud data centers in terms of both inter-PM and intra-PM. They describe a live VM migration algorithm to reduce I/O complexity and they realize experiments with both synthetic and real world data. [42] design a method for VM placement by predicting required processing and bandwidth resources and improving energy efficiency in a single data center.

Azizi et al. [43] propose a randomized greedy algorithm to solve virtual machine placement in large cloud data centers. Their algorithm considers multi-dimensional heterogeneous resources. Feng et al. [44] describe a VM placement strategy that considers heat recirculation in a cloud data center to reduce the total energy consumption by reducing the cooling cost. They propose a simulated annealing based method to address the VM placement problem. Bheda et al. [45] describe a VM placement algorithm that searches for the best possible physical machine to reduce the number of VM migrations. Kim et al. [46] consider the disk bandwidth usage as the critical factor and propose MMEVMP algorithm that aims to minimize SLAV rate and energy consumption. Feng et al. [47] propose a two-step SAG algorithm to decrease the energy consumption in data centers. Their energy consumption model considers both the usages of IT resources such as network use and non-IT resources such as cooling systems. Sadegh et al. [48] model VM placement problem as finding the minimum weight K-vertex-connected induced sub-graph. The algorithm they propose is a two-phase one. In the first phase, they select the best possible rack to improve load balance and in the second phase, they use a greedy algorithm to place VMs to hosts.

Our proposed work differs from the related work explained above by the following contributions:

- We propose a novel genetic algorithm, called UBGA, with a unique chromosome structure and fitness function that can find close-to-optimal solutions for the virtual machine placement problem.
- While placing virtual machines, we target not only high CPU utilization of physical machines, but also high utilization of memory and network connections of physical machines. In this way, we achieve a well-balanced load on the components of a physical machine. Considering the utilization of all these components at the same time enables minimization of the number of physical machines that need to be turned on, which can reduce energy consumption in the data center.
- To reduce the overall communication cost in the data center network, our approach tries to place virtual machines that are communicating with each other as close as possible in the network.

3. Problem description

In cloud systems, the computing and storage resources of service providers are totally virtualized. Optimization of resource utilization and virtual machine placement is essential for cloud owners for the satisfaction of their customers since more optimized cloud environments provide better performance, availability and reliability.

As mentioned in the related work section, the virtual machine placement problem can be considered as a bin packing problem [1]. Virtual machines can be considered as objects and physical machines as bins. However, the virtual machine placement problem has multiple constraints rather than only one constraint as in the bin packing problem.

The set of parameters presented in Table 1 is used for a formal description of our problem.

Among a wide range of network topologies, the network architecture we consider in this work is a three-level hierarchical (tree) topology. Leaf nodes represent the physical hosts and non-leaf nodes represent the network switches of a cloud data-center. Our method is aware of the network topology that consists of the connections among switches and the connections between switches and physical hosts. Our genetic algorithm creates *chromosomes* with exactly the same tree structure, as we explain in the next section.

We aim to minimize the wasted resources (CPU, memory and bandwidth) of physical machines, minimize the energy consumed by physical machines, and minimize the forwarding load on network switches due to communication among virtual machines. Wasted resources are the unused resources of non-idle physical machines which cannot be turned off. We can formally state the virtual machine placement problem as follows:

$$\sum_{p_j \in P} \boldsymbol{E_j} \tag{1}$$

$$\sum_{p_j \in P} (\boldsymbol{w_j^{cpu}} + \boldsymbol{w_j^{mem}} + \boldsymbol{w_j^{bw}}) \tag{2}$$

$$\sum_{v_i \in V} \sum_{v_j \in V} \boldsymbol{N_{ij}},\ where\ i < j \tag{3}$$

are minimized subject to

$$\boldsymbol{V} = \bigcup_{p_j \in P} V_j \tag{4}$$

$$\boldsymbol{p_j^{bw}} \geq \sum_{v_i \in V_j} v_i^{bw} \tag{5}$$

$$\boldsymbol{p_j^{mem}} \geq \sum_{v_i \in V_j} v_i^{mem} \tag{6}$$

$$\boldsymbol{p_j^{cpu}} \geq \sum_{v_i \in V_j} v_i^{cpu} \tag{7}$$

$$\boldsymbol{V_i} \cap \boldsymbol{V_j} = \emptyset,\ if\ \ i \neq j \tag{8}$$

$$\boldsymbol{U_j} = 1 - \boldsymbol{w_j^{cpu}} \tag{9}$$

$$\boldsymbol{E_j} = \boldsymbol{E_j^{idle}} + (\boldsymbol{E_j^{max}} - \boldsymbol{E_j^{idle}}) * \boldsymbol{U_j} \tag{10}$$

$$\boldsymbol{N_{ij}} = \boldsymbol{S_{ij}} * \boldsymbol{F_{ij}} \tag{11}$$

Table 1
Parameters for problem description

V	A set of virtual machines
P	A set of physical machines
P_{all}	Set of allocated physical machines
v_i	Virtual machine i
v_i^{cpu}	CPU requirement of v_i
v_i^{mem}	Memory requirement of v_i
v_i^{bw}	Network bandwidth requirement of v_i
p_j	Physical machine j
p_j^{cpu}	CPU capacity of p_j
p_j^{mem}	Memory capacity of p_j
p_j^{bw}	Network bandwidth capacity of p_j
V_j	List of virtual machines assigned to p_j
w_j^{cpu}	Wasted CPU percentage of p_j
w_j^{mem}	Wasted memory percentage of p_j
w_j^{bw}	Wasted bandwidth percentage of p_j
S_{ij}	Number of switches between v_i and v_j
F_{ij}	Data flow demand between v_i and v_j
N_{ij}	Communication cost of the connection between v_i and v_j
E_j	Total energy consumed by p_j
E_j^{idle}	Energy consumed by p_j in idle state
E_j^{max}	Energy consumed by p_j when it is fully utilized (100% CPU utilization)
U_j	Utilization of p_j

Eq. (4) states that every virtual machine is assigned to one physical machine, i.e. no virtual machine stays unplaced while Eq. (8) specifies that each virtual machine is assigned to only one physical machine.

Eqs. (5), (6), and (7) state that the total demand by virtual machines for each resource (CPU, memory and bandwidth) does not exceed the amount of the corresponding resource of the physical machine to which those virtual machines are assigned. As stated in Eq. (9), we define the utilization of a physical machine as percentage of CPU usage, as in [49]. The energy consumption of a machine is calculated based on the utilization of the machine, as shown in Eq. (10). An idle machine also consumes energy. Eq. (11) shows the calculation of communication cost. A virtual machine may communicate with another virtual machine and the cost of this communication is defined to be the number of switches between the physical machines hosting the virtual machines *times* the data flow demand among the virtual machines. We assume a traffic demand matrix of size $|V|$x$|V|$ that shows the desired data flow demands between virtual machines is given. Both energy consumption and communication cost calculations are inspired from [49].

4. Proposed method

4.1. Genetic algorithms

A genetic algorithm is a search and optimization technique inspired by the process of natural selection and genetics [50]. It is a meta-heuristic algorithm invented by John Henry Holland at the University of Michigan in the early 1970s, that aims to find good solutions to complex optimization problems [51].

An overview of how a genetic algorithm works is as follows:

- Initialization: The algorithm begins by initializing a population of candidate solutions, often referred to as individuals. Each individual represents a potential solution to the problem.
- Evaluation: Each individual in the population is evaluated and assigned a fitness value, which measures its quality with respect to the problem's objectives. The fitness function guides the search by providing a quantitative measure of how well an individual solves the problem.
- Selection: Individuals are paired up with other individuals to create the offspring for the next population. The process is generally based on the principle of *survival of the fittest* where individuals with higher fitness have a higher chance of being selected.
- Reproduction: The selected individuals undergo genetic operators, such as crossover and mutation, to create offspring. Crossover involves exchanging genetic information between two parent individuals to create new offspring, while mutation introduces random changes to the genetic information of individuals to avoid local maxima.
- Termination Condition: The algorithm continues iterating through the steps of selection and reproduction until a termination condition is met. Termination conditions can include reaching a maximum number of iterations (generations) or achieving a satisfactory solution.

By applying the steps of selection and reproduction iteratively, genetic algorithms explore the search space and evolve towards better solutions. The idea is that through the processes of crossover and mutation, the algorithm can discover and combine beneficial characteristics from different individuals, which leads to improved solutions over generations.

Genetic algorithms are known for their ability to handle complex, high-dimensional, and multi-objective optimization problems. They are widely used in various fields, such as engineering, computer science, and finance, to solve real-world problems where finding an exact optimal solution is computationally infeasible.

It is important to note that the performance of a genetic algorithm depends on several factors, including the way genetic operators are designed, population size, fitness function choice, and the termination condition. Careful tuning of these parameters is necessary to achieve good results for a specific problem.

4.2. Utilization Based Genetic Algorithm (UBGA)

We propose a genetic algorithm based approach, called Utilization Based Genetic Algorithm (UBGA), to optimize virtual machine placement considering the utilization of physical machines. Our approach aims to reduce the total wasted resources of physical machines in the cloud. We define the unused amount of resources of a physical machine as wasted if that physical machine has at least one virtual machine placed on it so that it cannot be shut down. We do not consider the resources of idle machines as wasted since they can be shut down. We integrated our algorithm into CloudSim toolkit which is an open-source cloud computing simulator allowing the creation and simulation of data centers, physical and virtual machines in an easy way.

Since genetic algorithms mimic the natural selection process of real life, where better individuals have more chances to survive in a

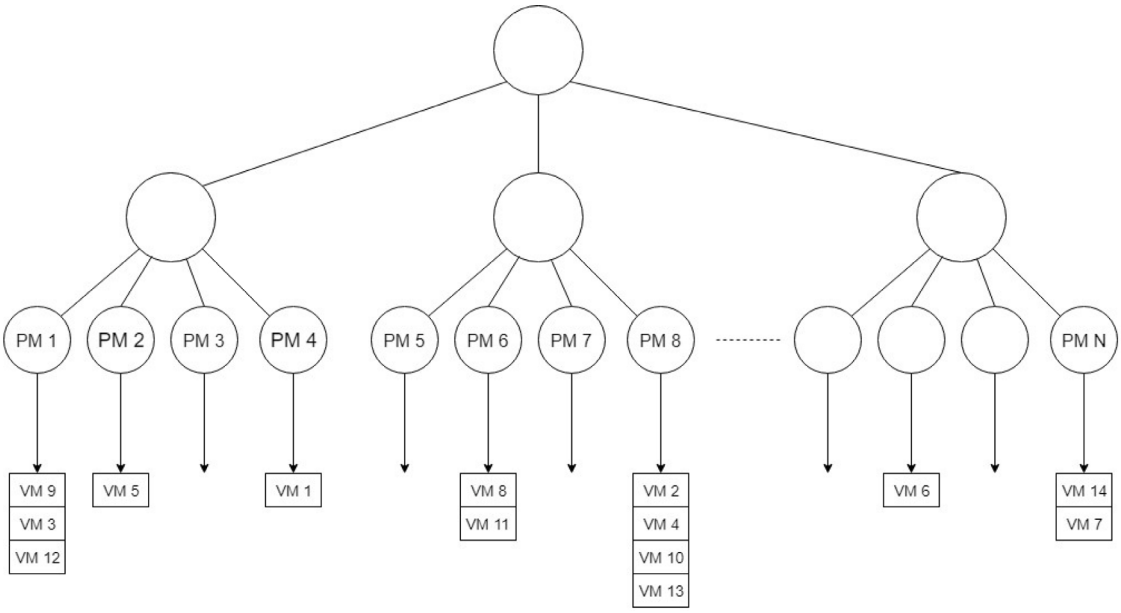

Fig. 1. The chromosome structure of our genetic algorithm.

population according to evolution theory, they can be used to find close-to optimum solution for a lot of complex problems [50]. As evolution in real life progresses, species have populations with better individuals after some number of generations. Similarly, a genetic algorithm starts with a population containing random individuals each of which represents a solution to the given problem, and improves these individuals at each generation to have better solutions. Therefore, a genetic algorithm produces a better solution to the given problem from one generation to another according to a given criterion. The criterion we use in this paper is a novel *fitness* function that we define. At the end, the algorithm reaches a solution that is good and that can approximates the optimal solution to the problem.

The only setback of genetic algorithms can be their running time complexity since a genetic algorithm may take too much time depending on the specified evolution termination condition and the number of individuals in the population in each generation. However, since the scenario that we focus on in this paper is not a highly dynamic one, the run-time required for finding a good solution is not the most prior concern, as long as the run-time is in an acceptable range, and therefore does not prevent us from using a genetic algorithm.

We next describe the components of our genetic algorithm for virtual machine placement.

4.3. Chromosome structure

A chromosome in our proposed genetic algorithm is a tree data structure whose leaves represent the physical machines and non-leaf nodes represent the network switches that connect these physical machines. Each leaf node has a pointer to a list, which keeps the virtual machines that are assigned to the physical machine represented by the leaf node. Hence, we have $|P|$ number of lists. Obviously, some of these lists may be empty when no virtual machine is assigned to the corresponding physical machine. The proposed chromosome structure has $|V|$ genes each one representing a single virtual machine. An example chromosome can be seen in Fig. 1.

With this tree structure, it is easier and faster to determine what percentage of resources of a physical machine is wasted. We also make use of this structure to calculate the communication cost between two virtual machines.

A *particular placement* of virtual machines to physical machines (feasible or unfeasible) is represented by such a tree and its lists. This is called an *individual*.

4.4. Fitness function

We define the *fitness* value of an individual who has k number of over-demanded (over-loaded) physical machines as in Eq. (12). Here, we have $0 \leq k \leq n$, where n is the number of physical machines in the data center.

$$\begin{aligned} fitness(i) = (k+1) * \Big(&\frac{1}{T_{cpu}} * \sum_{p_j \in P_{all}} (-\boldsymbol{w_j^{cpu}}) * \boldsymbol{p_j^{cpu}} \\ &+ \frac{1}{T_{mem}} * \sum_{p_j \in P_{all}} (-\boldsymbol{w_j^{mem}} * \boldsymbol{p_j^{mem}}) + \\ &\frac{1}{T_{bw}} * \sum_{p_j \in P_{all}} (-\boldsymbol{w_j^{bw}} * \boldsymbol{p_j^{bw}}) - \frac{\sum_{v_i \in V} \sum_{v_j \in V} \boldsymbol{N_{ij}}}{\boldsymbol{N_{max}}}\Big) \end{aligned} \quad (12)$$

where;

$$\boldsymbol{T_{cpu}} = \sum_{p_j \in P_{all}} \boldsymbol{p_j^{cpu}} \quad (13)$$

$$\boldsymbol{T_{mem}} = \sum_{p_j \in P_{all}} \boldsymbol{p_j^{mem}} \quad (14)$$

$$\boldsymbol{T_{bw}} = \sum_{p_j \in P_{all}} \boldsymbol{p_j^{bw}} \quad (15)$$

$$\boldsymbol{N_{max}} = H * \sum_{v_i \in V} \sum_{v_j \in V} \boldsymbol{F_{ij}},\ where\ i < j \quad (16)$$

The fitness function, whose value is negative, indicates how well a placement is. A bad placement will have a very low fitness value (a very large negative value) and this is obtained by punishing such bad placements as much as possible. The fitness function punishes over-demand by virtual machines the most, since this is the most undesired case where capacity constraints of physical machines are violated. That is to say, for one of the resources, if the total demand by virtual machines exceeds the capacity of the assigned physical machine, we reduce that individual's fitness score by multiplying the inside sum (which calculates the sum of wasted resources' percentage and normalized communication cost) in Eq. (12) with the number of over-demanded physical machines (k). Since the inside sum is negative, we severely penalize over-demand. With this big penalty, we try to prevent the existence of over-demanded physical machines in the final solution.

On the other hand, wasted resources and communication cost are more acceptable than over-demand. Therefore, they have less penalty,

which is the result of the inside sum. The inside sum is allowed to increase at most to −4, which is the sum of the total wasted percentage of CPU, memory and bandwidth of all used physical machines and normalized communication cost of the virtual machines in the cloud. Therefore, we aim directly to reduce the total wasted percentage of resources and communication cost in the whole data center.

Communication cost is normalized in the fitness function by dividing it by the maximum possible communication cost. Therefore, it has a value between 0 and 1. As a result, resource wastage and communication cost have an equal impact on the fitness score of individuals. Maximum possible communication cost is formulated in Eq. (16). We multiply the flow demand between two virtual machines with H, where H is the maximum number of network switches on the path between two virtual machines in the data center.

Moreover, we do not penalize individuals for having totally idle physical machines (machines with no VM assigned to them), since idle machines can be shut down. We want to utilize physical machines as much as possible, and to realize that we try to increase the number of idle machines, that can be shut down, as much as possible. Hence, more idle machines in a placement means less energy consumption, as can be seen in Eq. (10), which we aimed in Eq. (1).

4.5. Crossover operation

The crossover operation is realized to create the individuals of the next generation. It determines which genes are inherited from which parents, as described in Algorithm 1. For each virtual machine, we determine the physical host it is placed. The $uniformRate$ in the algorithm is to determine which parent is selected for inheritance. In our experiments, we decided to set $uniformRate$ to 0.5 to have an unbiased gene selection from either parent to improve the variety of newly created individuals. For each virtual machine, we generate a random value and according to the value, a parent is selected, and in the offspring's chromosome the virtual machine is assigned to the same physical machine to which it is assigned in the selected parent's chromosome. We do not favor the parent with the higher fitness score to increase variety in the offspring.

Algorithm 1 Crossover Operation

Require: Two parent chromosomes: C_1 and C_2
Ensure: One offspring chromosome: C_{new}
procedure CROSSOVER
  for $i = 1$ *to* $|V|$ **do**
    randomly create a value between 0 and 1, r;
    if $r \leq uniformRate$ **then**
      In C_{new}, assign VM i to the physical host that it is assigned to in C_1
    else
      In C_{new}, assign VM i to the physical host that it is assigned to in C_2
    end if
  end for
  return New individual with chromosome C_{new}
end procedure

4.6. Selection operation

We use the Tournament Selection operation to pair up individuals and then perform crossover operations for pairs. For each individual in the general population, we invoke the selection operation to pair up the individual with another for the crossover operation. Our selection operation, which is described in Algorithm 2, creates a sub-population called tournament population with randomly selected individuals from the overall population. Then, it returns the individual with the best fitness value in the tournament population. In the experiments, we set the size of this sub-population to be 5% of the overall population. We decided to use this value because it is large enough to have better pairs and it is small enough not to always bring the best individual, which is required for increasing variety among individuals, as we see in the experiments.

Algorithm 2 Selection Operation

Require: The tournament population, Pop
Ensure: an individual chromosome
procedure SELECTION
  $bestScore \leftarrow -\infty$
  $bestInd \leftarrow Null$
  for each Individual x in Pop **do**
    calculate its fitness score, $newScore$
    if $bestScore \leq newScore$ **then**
      $bestScore \leftarrow newScore$
      $bestInd \leftarrow x$
    end if
  end for
  return $bestInd$
end procedure

4.7. Mutation operation

The mutation operation randomly selects a virtual machine, and removes that virtual machine from its current physical machine, and places it to a newly selected physical machine. If this migration would cause over-demand on the newly selected physical machine, then the mutation operation is cancelled. Algorithm 3 describes in detail how mutation works. Mutation operation is essential for the methods that use genetic algorithms as it increases variety among the individuals in the offspring population. Moreover, mutation operation is useful to avoid being trapped at a local maximum.

Algorithm 3 Mutation Operation

Require: An individual chromosome: C_{old}
Ensure: A mutated individual chromosome: C_{mut}
procedure MUTATION
  $C_{mut} \leftarrow C_{old}$
  In C_{mut}, randomly select a virtual machine v; where $1 \leq v \leq |V|$
  In C_{mut}, randomly select a physical machine p; where $1 \leq p \leq |P|$
  In C_{mut}, remove v from its current physical machine
  In C_{mut}, add v to p
  return C_{mut}
end procedure

4.8. The genetic algorithm

The proposed genetic algorithm works very similarly to the traditional genetic algorithm. The flowchart of the algorithm can be seen in Fig. 2. We start with an initial population. Then, inside the generation loop, we start with calculating the fitness value of each individual followed by selecting another individual for each one to pair up to produce a new offspring. After that, we apply mutation operation if the randomly produced mutation value is less than the desired mutation rate. Moreover, we find the best individual among the new population and compare it with the current best one. If the new population's best individual is better than the current best individual, we set the new one as the current best. For the next iteration of the generation loop, we continue with the new generation and discard the old one, since the new generation consists of better individuals than the old population. The pseudo-code of our main algorithm is given in Algorithm 4.

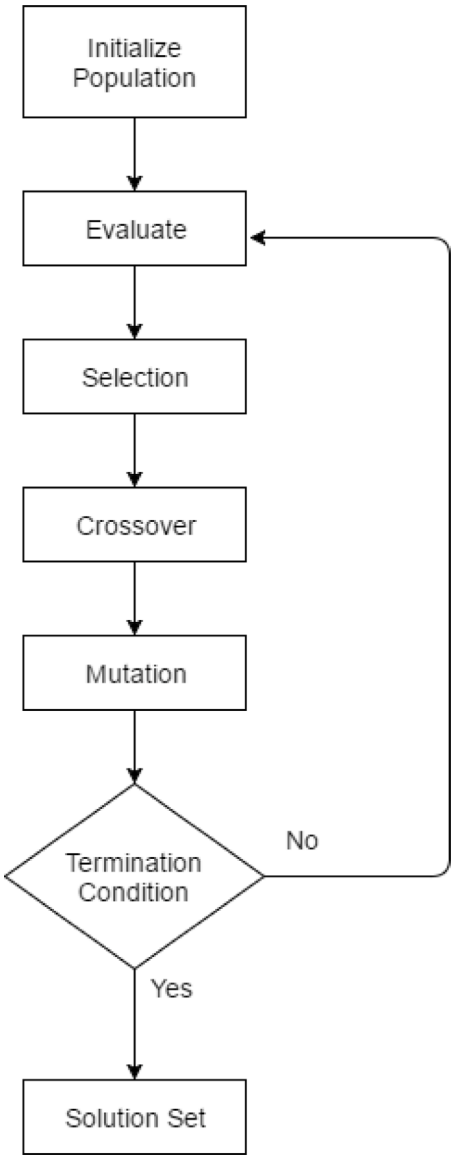

Fig. 2. Flowchart of genetic algorithm.

Algorithm 4 The Genetic Algorithm

procedure THE GA
  generate a population of $POPSIZE$ number of random individuals, POP;
  while THE TERMINATION CONDITION is not *true* **do**
    for each Individual i in POP **do**
      calculate its fitness value $f(i)$
    end for
    for each Individual i in POP **do**
      invoke the Selection Operation, that is using tournament selection technique to select another individual j to pair
    end for
    for each pair of parents **do**
      use Uniform Crossover Operation to produce an offspring
    end for
    for each offspring **do**
      apply Mutation Operation according to mutation rate
    end for
    find the best individual among offsprings, $newBest$
    if $newBest$ is better than current best individual **then**
      replace current best individual with $newBest$
    end if
  end while
  return best individual
end procedure

Table 2
Resource demand values of each virtual machine.

Resource	Values
CPU	250 Mips–1500 Mips
Memory	500 MB–5000 MB
Bandwidth	500 Mbps–1500 Mbps

Table 3
Resources provided by each physical machine.

Resource	Values
CPU	1000 Mips–3000 Mips
Memory	1 GB–10 GB
Bandwidth	1 Gbps–5 Gbps

Table 4
Resources and demands for each VM and PM in homogeneous distribution case.

Resource	VM demand	PM resource
CPU	250 Mips	2500 Mips
Memory	500 MB	5 GB
Bandwidth	100 Mbps	1 Gbps

5. Experimental results

To evaluate the performance of our proposed algorithm, we conducted extensive simulation experiments by using CloudSim v.3.0.3, which is an open-source cloud environment simulator framework implemented in Java. CloudSim has its own default virtual machine allocation policy. We extended this default policy to use the genetic algorithm library we created. Our library implements our proposed method.

We compared our method, Utilization Based Genetic Algorithm (UBGA), with CloudSim's default virtual machine placement policy (which we call CloudSim in the figures and in the rest of the paper), Random Allocation method (RA), First Fit Decreasing method (FFD), and AFED-EF method proposed by [4]. We did comparisons by using the following metrics:

1. Wasted CPU (percentage),
2. Wasted memory (percentage),
3. Wasted bandwidth (percentage),
4. Number of physical machines used,
5. Energy consumed by physical machines,
6. Communication cost.

We created test cases to compare our UBGA algorithm with the methods mentioned above. Information about the physical machine resource capacity values and virtual machine demand values for those test cases can be seen in Tables 1–3.

In all our test cases, the number of virtual machines to be allocated is varied between 200 and 1000 with an increment of 200. The number of physical machines is fixed at 200. All physical machines reside in a single data center.

In our genetic algorithm, the number of generations is set to 20 and the population size for each generation is constant and is set to 100. We chose these values because we observed in our experiments that any value greater than these does not significantly increase the fitness score of the best individual. That means increasing generation count and population size does not improve the solution quality much, and

therefore we did not set the values of these parameters too large to reduce the running time of our proposed method.

The parameter values we used for the components of our algorithm are set as follows:

- In the crossover operation, we set $uniformRate$ to be 0.5. This means an offspring has an equal chance of inheriting from its parents. As stated above, this value is selected for unbiased gene selection from parents to improve variety.
- In the selection operation, we set tournament population size to 5, which is 5% of the population size.
- In the genetic algorithm, the termination condition is reaching generation 20. We have seen in the experiments that this value is large enough, since the use of a larger value for the number of generations does not significantly change the fitness score of the best individual.
- In the genetic algorithm, the mutation rate is set to be 0.02. In our experiments, we observe that this is the optimal value for our algorithm. Typical values used for mutation rate in the literature are below 0.05, as can be seen, in [1,49].

5.1. CloudSim integration

The integration of our method to CloudSim involves extending CloudSim's own virtual machine allocation method with our algorithm. We implemented our genetic algorithm in a library. Our algorithm uses a tree data structure to represent a chromosome that has one-to-one mapping with the network and server interconnection topology of a data center where virtual machines are placed.

5.2. Homogeneous distribution

In homogeneous distribution case, we set the capacities of physical machines and demands of virtual machines as in Table 4. Each physical machine has the same initial amount of resources and each virtual machine has the same amount of demand for each type of resource.

When compared in terms of wasted resource amount, UBGA, AFED-EF, and FFD algorithms provide similar results and are the best ones. Random allocation and CloudSim's default allocation policy are far behind. The reason for the superior performance of FFD is that this case is the optimum scenario for that method. Both UBGA and FFD achieve placing virtual machines to the least number of physical machines. Of the m physical machines that might be used for the given set of VMs, the first $m-1$ machines are fully utilized and the last machine is partially utilized and is the only one that contributes to the waste of a particular resource type.

5.3. Heterogeneous deterministic distribution

In heterogeneous deterministic distribution case, resources demands of virtual machines are determined between the values shown in Table 2 and resource capacities provided by physical machines are determined between within the values shown in Table 3 as follows. The set of virtual machines is divided into 10 groups. The virtual machines in Group 0 have the minimum demands of CPU, memory, and bandwidth shown in Table 2. The virtual machines in Group i have $(10\times i)$% more demands than the ones in Group 0. For instance, the virtual machines in Group 1 have 10% more resource demands than the ones in Group 0, and the demands of virtual machines in Group 7 are 70% more than the demands of virtual machines in Group 0. The same logic applies to physical machines in terms of resource capacities.

The results of the heterogeneous deterministic distribution case are shown in Figs. 3 through 8. When we consider resource waste, we can say that UBGA and AFED-EF are clearly superior to CloudSim's method, random allocation method, and FFD method, as they cause far more memory, CPU, and bandwidth waste for five different virtual machine count cases, as shown in Figs. 3, 4, and 5, respectively. UBGA and AFED-EF have similar results in terms of wasted resources. AFED-EF is slightly better with the CPU waste while UBGA is clearly better with memory waste.

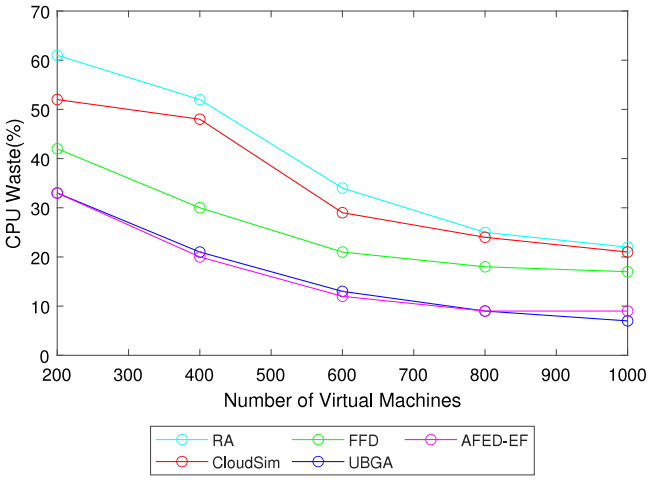

Fig. 3. The result of CPU wastage with heterogeneous deterministic distribution.

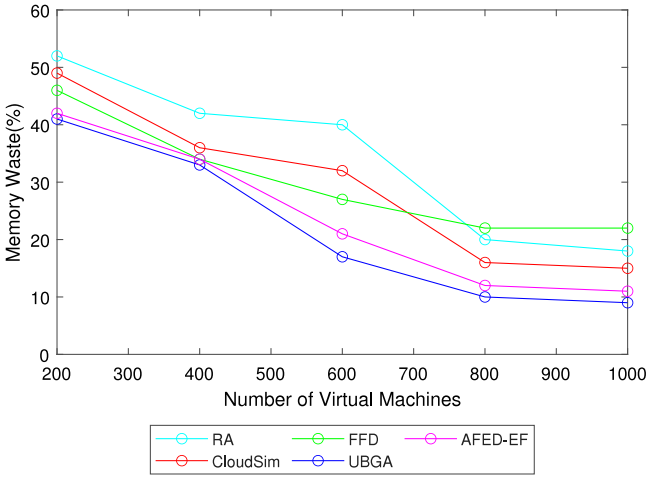

Fig. 4. The result of memory wastage with heterogeneous deterministic distribution.

AFED-EF uses fewer physical machines compared to other algorithms. Our UBGA method closely follows it, as shown in Fig. 6. UBGA tries to put virtual machines as close as possible in the network and since we do not penalize individuals for idle machines in the penalty calculation method, UBGA tends to use the least possible number of physical machines as well. However, FFD, RA, and CloudSim do not always try to produce solutions with fewer physical machines. They use all physical hosts in the data center for 800 and 1000 virtual machine cases.

For energy consumption of physical machines, we use the formula given in Eq. (10), where both utilization and number of idle machines are important. In this equation we use the values $E_j^{max} = 100$ and $E_j^{idle} = 10$, as they are CloudSim's default parameter values for energy calculation. Since AFED-EF directly targets energy consumption, it has the best results among compared algorithms. Our UGBA has the next-best performance. With fewer virtual machines, FFD has the worst performance, while RA and CloudSim are slightly better. The performance gap among the three algorithms closes with more virtual machines being requested. Fig. 7 shows the energy consumption results of the methods.

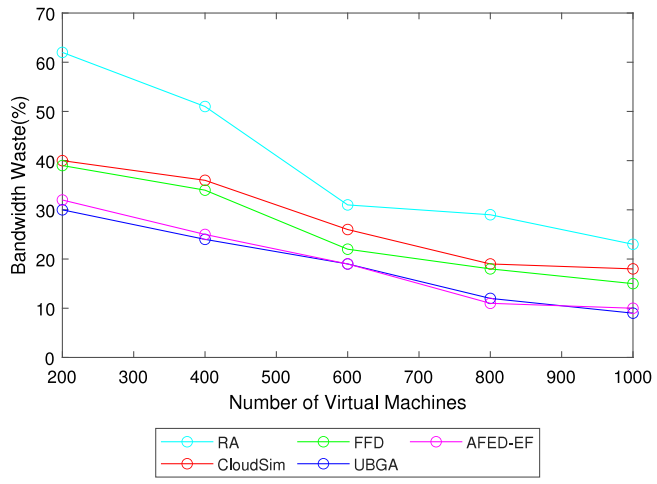

Fig. 5. The result of bandwidth wastage with heterogeneous deterministic distribution.

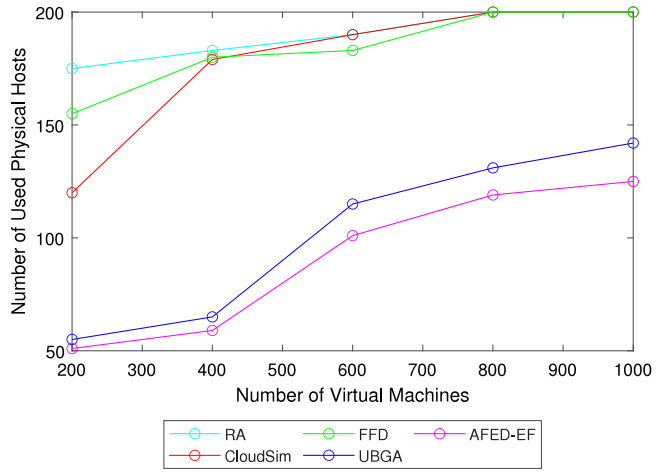

Fig. 6. Number of PMs used with heterogeneous deterministic distribution.

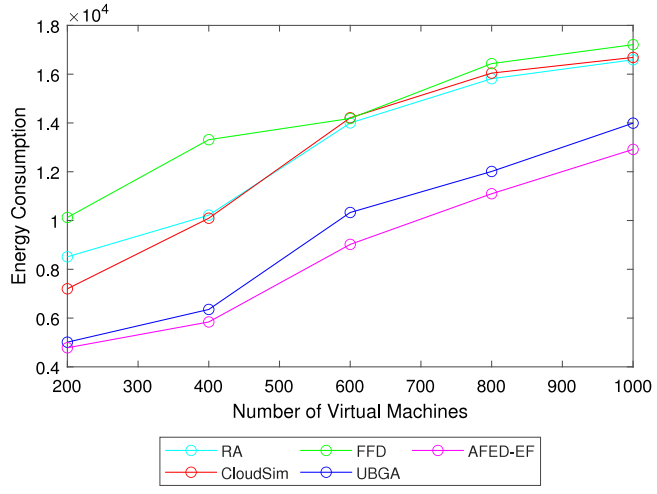

Fig. 7. Energy consumption by PMs used with heterogeneous deterministic distribution.

The communication cost of a VM placement is considered as the total load incurred on the switches of the data center network due to communication happening among VMs. This total load is computed as follows. For each pair of VMs that communicate, the flow demand between these VMs is multiplied by the number of switches between the physical machines where these VMs are placed. In this way, the load incurred by that pair of VMs is found. Then we sum the loads by all VM pairs and find out the total load incurred on the network, which we consider as the communication cost of the placement. In our experiments, we assigned a random data flow demand between 0 and 4 traffic units to each virtual machine pair. We have $\binom{|V|}{2}$ values in total. In this scenario, the methods that allocate virtual machines having larger flow demand between them as closer as possible in the cloud network topology produce better results. As can be seen in Fig. 8, UBGA, AFED-EF, and FFD are clearly superior to the other two methods. There is a very small difference between the performance results of these top three methods. For fewer virtual machines to be placed, the three methods have very close results. For larger number of virtual machines, UBGA performs slightly better than the others (Fig. 8). The reason why UBGA produces the best result in terms of communication cost is that it also incorporates communication cost in the fitness function, while other methods do not.

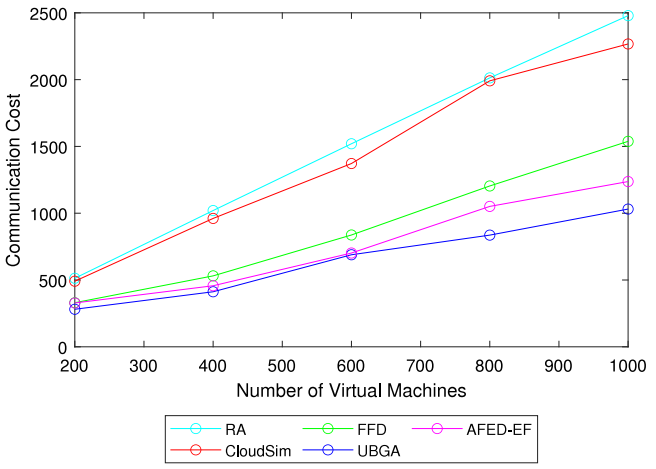

Fig. 8. Communication cost with heterogeneous deterministic distribution.

5.4. Uniform random distribution

In the last case considered in our experiments, resource demands of virtual machines are randomly determined within the value range shown in Table 2, and the resource capacities provided by physical machines are randomly determined within the value range shown in Table 3.

The results for the uniform random distribution of resource capacities and demands are shown in Figs. 9 through 14. For resource waste metrics, the random allocation has the worst performance as in the previous case. CloudSim default policy and FFD are slightly better than the random one, but still, they do not provide the best solution. AFED-EF and UBGA have the best performance. In terms of, CPU and bandwidth waste, they have very similar performance while the performance gap in memory waste is clearer with UBGA being the better one (Figs. 9, 10, 11).

In terms of the number of physical machines used, the energy consumption of physical machines, and communication cost, we have similar results as in the heterogeneous deterministic distribution case. The methods having optimization concern provide better performance than the ones which do not have such a concern (Figs. 12, 13, and 14). In terms of energy cost and the number of non-idle physical machines, AFED-EF performs better compared to UBGA while UBGA has the best communication cost performance among all five compared methods.

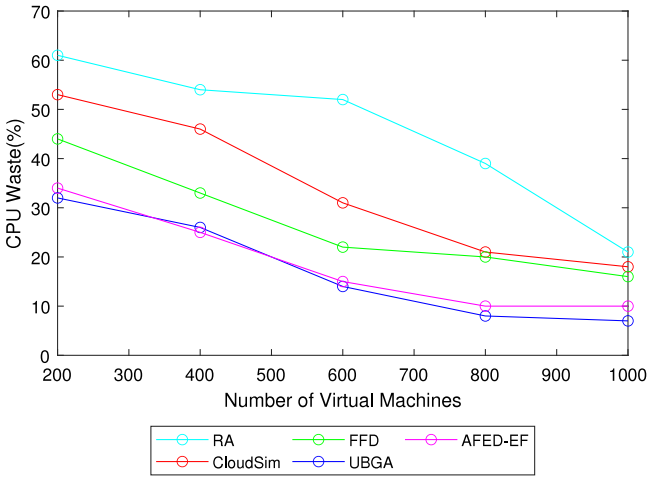

Fig. 9. The result of CPU wastage with uniform random distribution.

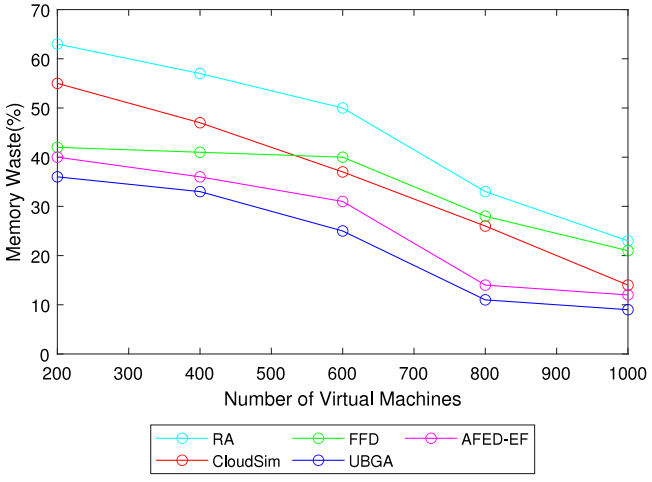

Fig. 10. The result of memory wastage with uniform random distribution.

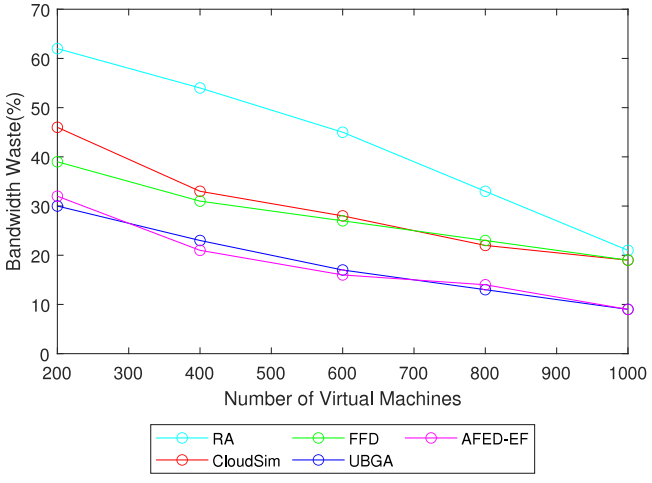

Fig. 11. The result of bandwidth wastage with uniform random distribution.

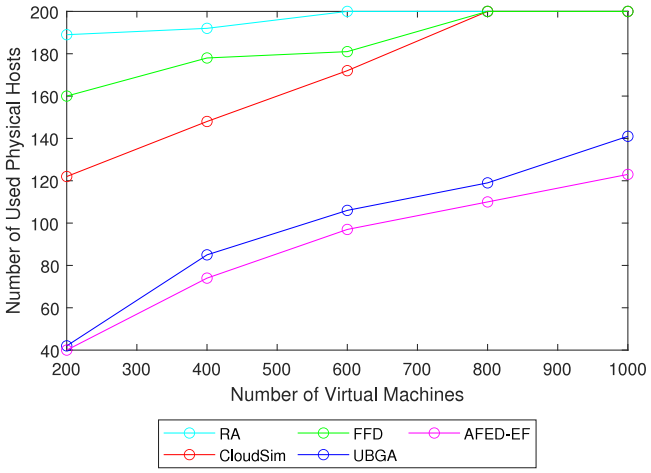

Fig. 12. Number of PMs used with uniform random distribution.

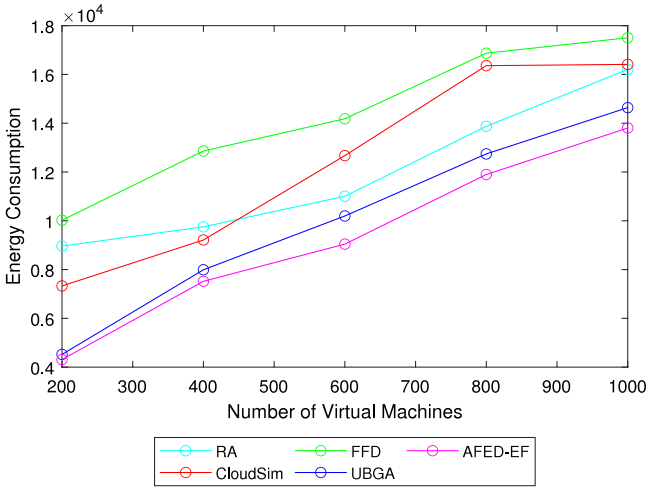

Fig. 13. Energy consumption by PMs used with uniform random distribution.

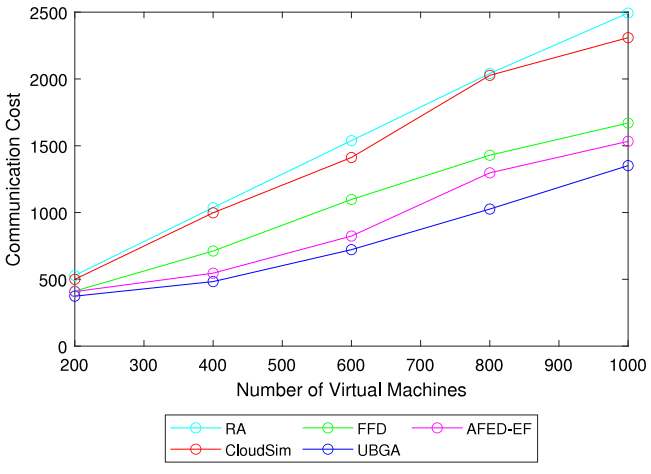

Fig. 14. Communication cost with uniform random distribution.

5.5. Statistical analysis of the results

A statistical significance test is conducted at a 5% significance level to check whether the average results obtained by our UBGA algorithm compared to other algorithms are statistically significant. For all metrics, the p-values provided by the t-tests that compare the average results produced by UBGA with Random Allocation, CloudSim and FFD are less than 0.001. However, the t-tests that compare results of UBGA and AFED-EF are not always less than 0.001, therefore the

Table 5
p-values of comparison with AFED-EF.

	|VM| = 200	|VM| = 400	|VM| = 600	|VM| = 800	|VM| = 1000
CPU Waste	<0.001	0.037	0.024	<0.001	<0.001
Memory Waste	<0.001	<0.001	<0.001	<0.001	<0.001
BW Waste	0.007	0.002	0.015	0.029	0.041
Used PM Count	<0.001	<0.001	<0.001	<0.001	<0.001
Energy Cost	<0.001	<0.001	<0.001	<0.001	<0.001
Communication Cost	<0.001	<0.001	<0.001	<0.001	<0.001

Table 6
Comparison of bandwidth waste results between CloudSim and CloudStack experiments.

VM/PM	Uniform Random Dist.		Heterogeneous Det. Dist.	
	CloudSim	CloudStack	CloudSim	CloudStack
1	30%	32%	31%	34%
2	24%	25%	22%	21%
3	18%	17%	18%	16%
4	14%	14%	13%	15%
5	11%	12%	11%	13%

Table 7
Comparison of CPU waste results between CloudSim and CloudStack experiments.

VM/PM	Uniform Random Dist.		Heterogeneous Det. Dist.	
	CloudSim	CloudStack	CloudSim	CloudStack
1	32%	35%	33%	31%
2	27%	26%	21%	19%
3	14%	14%	12%	13%
4	9%	11%	10%	11%
5	9%	10%	9%	10%

p-values are presented in Table 5. For wasted memory, the number of non-idle physical machines, energy cost and communication cost, all p-values are still less than 0.001. Some p-values obtained by the t-tests that compare wasted CPU and bandwidth are larger than 0.001, however, they are still less than 0.05 which is our significance level. All experimental results are statistically significant with a significance value of 0.05.

5.6. Validation of the algorithm and simulations

In order to validate the effectiveness of our algorithm, we modified the placement algorithm of Apache CloudStack to embed our algorithm into this open-source cloud platform, and realized simulations on CloudStack as well. Moreover, for the experiments conducted with CloudStack, we also increased the number of physical and virtual machines to real cloud systems scale. The results obtained in these experiments show that our method provides similar performance results compared to the aforementioned experiments with smaller parameter values.

We performed experiments with 50 000 physical machines and 50 000–250 000 virtual machines to have the same $\frac{\text{number of VM}}{\text{number of PM}}$ ratios as in our previous experiments. The demands of virtual machines and the resources of physical machines are first distributed in a heterogeneous deterministic way as in Section 5.3, and then randomly using the parameter values that are provided in Tables 2 and 3.

The comparative performance results of our algorithm obtained with CloudSim and CloudStack are provided in Tables 6, 7, and 8. The values given in the tables are averages of 100 runs. The number of machines employed in the experiments conducted through CloudStack is at real cloud-scale with a much higher number of machines compared to the parameter values used in the CloudSim simulations. Validation of simulation results is ensured with both random and uniform distribution of resources and demands as shown in the corresponding tables. Our proposed algorithm can scale to real cloud systems with the same effectiveness while aiming to minimize wasted resources of physical hosts.

6. Conclusion & future work

How to place virtual machines in a data center considering multiple objectives and constraints is one of the most important and challenging problems in virtualized cloud systems. In this paper, we proposed the so-called Utilization Based Genetic Algorithm (UBGA) that makes use of an innovative fitness function and chromosome structure with the aim to increase server utilization, decrease resource waste, and reduce network overhead, all at the same time.

Our unique fitness function and related chromosome structure enable combining multiple objectives together in a compact form. Our method considers the utilization of multiple machine components in making placement decisions: CPU, memory, and network links. It balances the load on these different types of resources in a server. Balancing the load is very important to get the maximum utility from a server, i.e., place and run as many virtual machines and applications as possible.

Our approach also considers network topology to make effective placement decisions. By also considering the physical network topology interconnecting the physical servers and the distances between them, the cost of communication among virtual machines is reduced. This prevents the network from being a bottleneck while making effective placements.

We evaluated the performance of our algorithm via extensive simulation experiments and compared it with four others: Random allocation (RA), CloudSim's default allocator (CloudSim), First Fit Decreasing method (FFD), and AFED-EF [4] algorithm from the literature. Our results show that Random and CloudSim allocators perform worse than the other approaches. FFD provides an average performance for heterogeneous deterministic and uniform random capacity and demand distributions, and excellent performance for homogeneous distribution, which is its optimum case. AFED-EF performs better than our UBGA in terms of energy consumption and the number of physical machines in use. In terms of wasted resources, the two algorithms have similar performances. In terms of communication cost, the performance of UBGA is the best.

There are a few directions to extend our work further. Since virtual machine requests in cloud systems can be made dynamically, we want

Table 8
Comparison of memory waste results between CloudSim and CloudStack experiments.

VM/PM	Uniform Random Dist.		Heterogeneous Det. Dist.	
	CloudSim	CloudStack	CloudSim	CloudStack
1	38%	36%	42%	45%
2	32%	30%	34%	34%
3	22%	23%	18%	20%
4	10%	12%	9%	9%
5	9%	12%	8%	8%

to extend our algorithm to handle online VM placement. In this case, we also want to consider VM migrations to maximize utilization of physical hosts and customer satisfaction at the same time. Moreover, since in dynamic cases, VM requests can be considered for a finite amount of time, we want to investigate the resource scheduling problem as well.

CRediT authorship contribution statement

Mustafa Can Çavdar: Conceptualization, Methodology, Software, Investigation, Validation, Writing – original draft, Writing – review & editing, Visualization. **Ibrahim Korpeoglu:** Conceptualization, Writing – review & editing, Supervision, Project administration. **Özgür Ulusoy:** Conceptualization, Writing – review & editing, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] R. Sookhtsaraei, M. Madani, A. Kavian, A multi objective virtual machine placement method for reduce operational costs in cloud computing by genetic, Int. J. Comput. Networks Commun. Secur. 2 (8) (2014) 250–256.

[2] M. Masdari, S. Nabavi, V. Ahmadi, An overview of virtual machine placement schemes in cloud computing, J. Netw. Comput. Appl. 66 (2016) http://dx.doi.org/10.1016/j.jnca.2016.01.011.

[3] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A. De Rose, R. Buyya, CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Softw. - Pract. Exp. 41 (1) (2011) 23–50.

[4] Z. Zhou, M. Shojafar, M. Alazab, J. Abawajy, F. Li, AFED-EF: An energy-efficient VM allocation algorithm for IoT applications in a Cloud Data Center, IEEE Trans. Green Commun. Netw. 5 (2) (2021) 658–669.

[5] L. Cui, J. Zhang, L. Yue, Y. Shi, H. Li, D. Yuan, A genetic algorithm based data replica placement strategy for scientific applications in clouds, IEEE Trans. Serv. Comput. 11 (4) (2018) 727–739.

[6] C.T. Joseph, K. Chandrasekaran, R. Cyriac, A novel family genetic approach for virtual machine allocation, Procedia Comput. Sci. 46 (2015) 558–565.

[7] A. Tripathi, I. Pathak, D.P. Vidyarthi, Energy efficient VM placement for effective resource utilization using modified binary PSO, Comput. J. 61 (6) (2018) 832–846.

[8] X. Zhang, T. Wu, M. Chen, T. Wei, J. Zhou, S. Hu, R. Buyya, Energy-aware virtual machine allocation for cloud with resource reservation, J. Syst. Softw. 147 (2019) 147–161.

[9] B. Zhang, X. Wang, H. Wang, Virtual machine placement strategy using cluster-based genetic algorithm, Neurocomputing 428 (2021) 310–316.

[10] M. Ghetas, A multi-objective Monarch Butterfly Algorithm for virtual machine placement in cloud computing, Neural Comput. Appl. 33 (17) (2021) 11011–11025.

[11] Z. Zhou, F. Li, S. Yang, A novel resource optimization algorithm based on clustering and improved differential evolution strategy under a cloud environment, Trans. Asian Low-Resourc. Lang. Inf. Process. 20 (5) (2021) 1–15.

[12] H. Hallawi, J. Mehnen, H. He, Multi-capacity combinatorial ordering GA in application to cloud resources allocation and efficient virtual machines consolidation, Future Gener. Comput. Syst. 69 (2017) 1–10.

[13] Z. Xiao, J. Jiang, Y. Zhu, Z. Ming, S. Zhong, S. Cai, A solution of dynamic VMs placement problem for energy consumption optimization based on evolutionary game theory, J. Syst. Softw. 101 (2015) 260–272.

[14] F. López-Pires, B. Barán, L. Benítez, S. Zalimben, A. Amarilla, Virtual machine placement for elastic infrastructures in overbooked cloud computing datacenters under uncertainty, Future Gener. Comput. Syst. 79 (2018) 830–848.

[15] Z. Zhou, M. Shojafar, J. Abawajy, A.K. Bashir, IADE: An improved differential evolution algorithm to preserve sustainability in a 6G network, IEEE Trans. Green Commun. Netw. 5 (4) (2021) 1747–1760.

[16] F. Alharbi, Y.-C. Tian, M. Tang, W.-Z. Zhang, C. Peng, M. Fei, An ant colony system for energy-efficient dynamic virtual machine placement in data centers, Expert Syst. Appl. 120 (2019) 228–238.

[17] A. Abohamama, E. Hamouda, A hybrid energy–Aware virtual machine placement algorithm for cloud environments, Expert Syst. Appl. 150 (2020) 113306.

[18] K. Baalamurugan, S.V. Bhanu, A multi-objective krill herd algorithm for virtual machine placement in cloud computing, J. Supercomput. 76 (6) (2020) 4525–4542.

[19] J. Masoudi, B. Barzegar, H. Motameni, Energy-Aware virtual machine allocation in DVFS-enabled cloud data centers, IEEE Access 10 (2021) 3617–3630.

[20] S. Li, L. Li, D. Deng, H. Lin, J. Gao, Y.-s. Wang, A virtual machine placement strategy with low resource consumption, in: 2021 the 13th International Conference on Computer Modeling and Simulation, 2021, pp. 87–94.

[21] M. Kiani, M.R. Khayyambashi, A network-aware and power-efficient virtual machine placement scheme in cloud datacenters based on chemical reaction optimization, Comput. Netw. 196 (2021) 108270.

[22] S.S. Nabavi, S.S. Gill, M. Xu, M. Masdari, P. Garraghan, TRACTOR: Traffic-aware and power-efficient virtual machine placement in edge-cloud data centers using artificial bee colony optimization, Int. J. Commun. Syst. 35 (1) (2022) e4747.

[23] K. Balaji, P. Sai Kiran, M. Sunil Kumar, Power aware virtual machine placement in iaas cloud using discrete firefly algorithm, Appl. Nanosci. (2022) 1–9.

[24] H. Xing, J. Zhu, R. Qu, P. Dai, S. Luo, M.A. Iqbal, An ACO for energy-efficient and traffic-aware virtual machine placement in cloud computing, Swarm Evol. Comput. 68 (2022) 101012.

[25] D. Saxena, I. Gupta, J. Kumar, A.K. Singh, X. Wen, A secure and multiobjective virtual machine placement framework for cloud data center, IEEE Syst. J. (2021).

[26] H.O. Salami, A. Bala, S.M. Sait, I. Ismail, An energy-efficient cuckoo search algorithm for virtual machine placement in cloud computing data centers, J. Supercomput. 77 (11) (2021) 13330–13357.

[27] H. Deng, L. Huang, C. Yang, H. Xu, B. Leng, Optimizing virtual machine placement in distributed clouds with M/M/1 servers, Comput. Commun. 102 (2017) 107–119.

[28] L. Zhao, L. Lu, Z. Jin, C. Yu, Online virtual machine placement for increasing cloud provider's revenue, IEEE Trans. Serv. Comput. 10 (2) (2017) 273–285.

[29] A. Ponraj, Optimistic virtual machine placement in cloud data centers using queuing approach, Future Gener. Comput. Syst. 93 (2019) 338–344.

[30] M.K. Gupta, T. Amgoth, Resource-aware algorithm for virtual machine placement in cloud environment, in: 2016 Ninth International Conference on Contemporary Computing, IC3, IEEE, 2016, pp. 1–6.

[31] H. Roh, C. Jung, K. Kim, S. Pack, W. Lee, Joint flow and virtual machine placement in hybrid cloud data centers, J. Netw. Comput. Appl. 85 (2017) 4–13.

[32] Z. Zhou, K. Li, J. Abawajy, M. Shojafar, C. Morshed, F. Li, K. Li, An adaptive energy-aware stochastic task execution algorithm in virtualized networked datacenters, IEEE Trans. Sustain. Comput. (2021).

[33] A.R. Ilkhechi, I. Korpeoglu, Ö. Ulusoy, Network-aware virtual machine placement in cloud data centers with multiple traffic-intensive components, Comput. Netw. 91 (2015) 508–527.

[34] R. Wang, J.A. Wickboldt, R.P. Esteves, L. Shi, B. Jennings, L.Z. Granville, Using empirical estimates of effective bandwidth in network-aware placement of virtual machines in datacenters, IEEE Trans. Netw. Serv. Manag. 13 (2) (2016) 267–280.

[35] M. Gaggero, L. Caviglione, Model predictive control for energy-efficient, quality-aware, and secure virtual machine placement, IEEE Trans. Autom. Sci. Eng. 16 (1) (2018) 420–432.

[36] W. Lin, S. Xu, J. Li, L. Xu, Z. Peng, Design and theoretical analysis of virtual machine placement algorithm based on peak workload characteristics, Soft Comput. 21 (5) (2017) 1301–1314.

[37] Z.Á. Mann, Multicore-aware virtual machine placement in cloud data centers, IEEE Trans. Comput. 65 (11) (2016) 3357–3369.

[38] S. Omer, S. Azizi, M. Shojafar, R. Tafazolli, A priority, power and traffic-aware virtual machine placement of IoT applications in cloud data centers, J. Syst. Archit. 115 (2021) 101996.
[39] H.-P. Jiang, W.-M. Chen, Self-adaptive resource allocation for energy-aware virtual machine placement in dynamic computing cloud, J. Netw. Comput. Appl. 120 (2018) 119–129.
[40] C. Wei, Z.-H. Hu, Y.-G. Wang, Exact algorithms for energy-efficient virtual machine placement in data centers, Future Gener. Comput. Syst. 106 (2020) 77–91.
[41] R. Li, Q. Zheng, X. Li, Z. Yan, Multi-objective optimization for rebalancing virtual machine placement, Future Gener. Comput. Syst. 105 (2020) 824–842.
[42] R. Shaw, E. Howley, E. Barrett, An energy efficient anti-correlated virtual machine placement algorithm using resource usage predictions, Simul. Model. Pract. Theory 93 (2019) 322–342.
[43] S. Azizi, M. Shojafar, J. Abawajy, R. Buyya, Grvmp: A greedy randomized algorithm for virtual machine placement in cloud data centers, IEEE Syst. J. 15 (2) (2020) 2571–2582.
[44] H. Feng, Y. Deng, Y. Zhou, G. Min, Towards heat-recirculation-aware virtual machine placement in data centers, IEEE Trans. Netw. Serv. Manag. (2021).
[45] H. Bheda, C. Thaker, S. Shah, An optimized VM placement approach to reduce energy consumption in green cloud computing, in: Proceedings of the International Conference on Data Science, Machine Learning and Artificial Intelligence, 2021, pp. 130–135.
[46] M.-H. Kim, J.-Y. Lee, S.A.R. Shah, T.-H. Kim, S.-Y. Noh, Min-max exclusive virtual machine placement in cloud computing for scientific data environment, J. Cloud Comput. 10 (1) (2021) 1–17.
[47] H. Feng, Y. Deng, J. Li, A global-energy-aware virtual machine placement strategy for cloud data centers, J. Syst. Archit. 116 (2021) 102048.
[48] S. Sadegh, K. Zamanifar, P. Kasprzak, R. Yahyapour, A two-phase virtual machine placement policy for data-intensive applications in cloud, J. Netw. Comput. Appl. 180 (2021) 103025.
[49] G. Wu, M. Tang, Y.-C. Tian, W. Li, Energy-efficient virtual machine placement in data centers by genetic algorithm, in: International Conference on Neural Information Processing, Springer, 2012, pp. 315–323.
[50] K.-F. Man, K.-S. Tang, S. Kwong, Genetic Algorithms: Concepts and Designs, Springer Science & Business Media, 2012.
[51] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, 1998.