



EPBLA: energy-efficient consolidation of virtual machines using learning automata in cloud data centers

Nayereh Rasouli¹ · Ramin Razavi² · Hamid Reza Faragardi³

Received: 11 May 2019 / Revised: 24 January 2020 / Accepted: 1 February 2020 / Published online: 28 March 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

High demand for computational power by business, science, and applications has led to the creation of large-scale data centers that consume enormous amounts of energy. This high energy consumption not only imposes a significant operating cost but also has a negative impact on the environment (greenhouse gas emissions). A promising solution to reduce the amount of energy used by data centers is the consolidation of virtual machines (VMs) that allows some hosts to enter low consuming sleep modes. Dynamic migration (replacement) of VMs between physical hosts is an effective strategy to achieve VM consolidation. Dynamic migration not only saves energy by migrating the VMs hosted by idle hosts but can also avoid hotspots by migrating VMs from over-utilized hosts. In this paper, we presented a new approach, called extended-placement by learning automata (EPBLA), based on learning automata for dynamic replacement of VMs in data centers to reduce power consumption. EPBLA consists of two parts (i) a linear reward penalty scheme which is a finite action-set learning automata that runs on each host to make a fully distributed VM placement considering CPU utilization as a metric to categorize the hosts, and (ii) a continuous action-set learning automata as a policy for selecting an underload host initiating the migration process. A real-world workload is used to evaluate the proposed method. Simulation results showed the efficiency of EPBLA in terms of reduction of energy consumption by 20% and 30% compared with PBLA and Firefly, respectively.

Keywords Energy consumption · Learning automata · Placement of virtual machines · Cloud computing · VM migration

1 Introduction

Cloud computing is a new computing paradigm emerged in recent years. It is a model that enables ubiquitous, convenient and on-demand network access to a shared pool of

configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. In recent years, the IT infrastructure is rapidly growing due to the demand for computing power used by applications. In addition, modern data centers in cloud computing are hosting a variety of advanced applications. The high energy cost and greenhouse gas emissions are significant problems arising due to high energy consumption in large data centers. Nowadays, given the importance of energy in the world, reduction of energy consumption is regarded as a key challenge in both academia and industry. In addition to the operating costs, high energy consumption leads to an increase in temperature which in turn reduces the reliability and longevity of hardware resources. Low-utilized servers are the major factors contributing to low energy-efficiency of data centers. For example, reports showed that in a Google data

✉ Nayereh Rasouli
n.rasouli@hiau.ac.ir

Ramin Razavi
Ramin_razavi@ut.ac.ir

Hamid Reza Faragardi
hrfa@kth.se

¹ Department of Computer Engineering, Hashtgerd Branch, Islamic Azad University, Hashtgerd, Iran

² School of Electrical and Computer Engineer, University of Tehran, Tehran, Iran

³ Department of Computer Science and Communication, KTH Royal Institute of Technology, Stockholm, Sweden

center, the average utilization of physical machines is less than 30% [1].

The IT infrastructures provided by data centers must meet various Service Level Agreements (SLAs) established between the provider and the clients. SLAs could be related with either the resources such as amount of computing power (e.g., number of CPUs), memory, storage capacity, and network bandwidth or performance metrics (e.g., availability). Virtual machines (VMs) consolidation not only focuses on minimizing energy consumption but also should consider SLA as the important constraints. Using energy-aware scheduling mechanisms could result in considerable savings in energy consumption. According to the saved energy, these mechanisms should make a compromise between the performance and reduced service cost. According to the performance requirements, energy-aware data centers support the dynamic migration of VMs between the physical nodes. When VMs do not use all of the provided resources, they can be modified to a reasonable size and deployed to the minimum number of physical nodes. To reduce the energy consumption by the idle node and also data center, total energy consumption changes those nodes to sleep mode or turns them off [2]. Virtualization capabilities can be also exploited to reduce the energy consumption. In a general definition, virtualization creates an abstract level of computing resources that can be applied at different levels of a machine. It is a key enabler for emerging cyber-physical production systems [3]. Before the virtualization, the resource categories in each granularity level and the virtualization models of different resources should be defined [4]. The functionality of a resource at various granularity levels should be modeled [5]. Virtualization improves productivity by making efficient use of their resources. It also can reduce the amount of hardware resources needed. As VMs sit on a hardware resource, they are independent of each other, so the finish time of their activity is different from one another and this makes the part of capacity on hardware machines remain unused. A new technique which has been developed to address this problem is migration. Through using this technique along with moving and finding a better place for each VM, it is possible to have data centers with higher performances on the other hand, it is applicable to reduce the lateral cost including electricity costs.

In this paper we extend our previous work (PBLA) [6], called as Extended-PBLA (EPBLA), a new approach based on Learning Automata for dynamic replacement of virtual machines in data centers to reduce power consumption. EPBLA unlike the past algorithms does not use previous solutions but tries to use the ability of learning automata to do virtual machine placement. The proposed approach can effectively handle heterogeneous infrastructures and heterogeneous VMs and does not require any knowledge

about applications running on VMs. Live migration and forcing idle nodes to sleep are the main policies used in EPBLA.

Compared to our previous work (PBLA) our contributions are:

1. Testing the impact of number of VMs on the algorithm performance.
2. Investigating the impact of an energy-aware method on the energy consumption against the none-power-aware.
3. Adopting the MMT policy among the two policies used in our previous study which were originally proposed by Beloglazov and Buyya [2]. The reason to adopt MMT is that the number of migrations to the number of shutdown hosts is optimal in MMT.
4. Considering two new factors that significantly affect the overall energy consumption. The factors list below:
 - (A) Meantime before a host shutdown.
 - (B) Meantime before a VM migrate.

The rest of the paper is structured as follows: previous works are discussed in Sect. 2. In Sect. 3, Learning Automata is introduced. VMs energy-aware Extended-Placement by Learning Automata (EPBLA) is discussed in Sect. 4. Evaluation of the proposed method is represented in Sect. 5. The conclusions are given in Sect. 6.

2 Related work

A lot of works have been done to deal with VM placement in data centers.

Beloglazov et al. [7] proposed an energy-aware resources allocation algorithm. Their approach was based on the idea of “mixed utilization thresholds”. Tsai et al. [8] studied a parallel-machine scheduling including both task processing and resource allocation using an Improved Differential Evolution Algorithm (IDEA). Their method works based on their proposed cost and time models in cloud computing environment. Compared to our LA approach, Learning Automata has more accurate decisions and it is not very expensive.

Wei et al. [9] introduced an energy-efficient VM placement scheme, which tried to descend communication cost and power consumption over traffic-aware data center networks by using an improved Ant Colony Optimization with an adaptive parameter setting. The problem of VM placement and energy efficiency in the cloud using an Ant Colony System (ACS) is targeted in [10]. The authors also use a local search technique for order exchange and migration (OEM). Furthermore, a combination of First Fit and Ant Colony is employed by authors in [11]. Their goal is the reduction of energy consumption in data centers.

Barlaskara et al. [12] addressed VM placement issues by two meta-heuristic algorithms namely, the enhanced modified Firefly algorithm (MFF) and the Hierarchical Cluster-based Modified Firefly algorithm (HCMFF), presenting the comparative analysis of energy optimization. EPBLA, unlike the mentioned algorithms, uses the ability of Learning Automata to do VM placement. Combining the Genetic Algorithm (GA) with the Tabu Search algorithm, the authors in [13] proposed an energy-aware algorithm named GATA. Their goal was energy efficiency while maximizing load balance among varieties of resources. In [14], the authors presented an alternative multi-objective optimization (MOP) approach to solve the VM placement problem. Their method is based on a combination of the Slap Swarm and Sine–Cosine algorithms.

The authors in [15] provided a VM consolidation algorithm with multiple usage prediction (VMCUP-M; referred to resource types and the horizon) to improve the energy efficiency of cloud data centers. In this context, multiple usage refers to both employed to predict future utilization. The joint use of resources allowed for a reliable characterization of overloaded and under loaded servers, thereby decreasing the load and the power consumption after consolidation.

Similar to our method, some articles analyzed the performance of their approaches by the CloudSim toolkit. Shu et al. [16] proposed an improved clonal selection algorithm based on time cost and energy consumption models in cloud computing environment. This approach had immense potential as it offered significant improvement in energy efficiency of the data center and could effectively meet the service level agreement requested by the users. In [17], the authors proposed a predictive anti-correlated placement algorithm (PACPA), which assumed both CPU and bandwidth resource consumption. Ranjbari et al. [18] proposed a new algorithm according to Learning Automata, which improves resource usage and reduces energy consumption. This algorithm reduces the number of migrations, and shuts down idle servers to reduce the energy consumption of the data center.

Esfandiarpour et al. [19] proposed several VM consolidation algorithms based on network infrastructure, shelves, and connectivity to decrease energy consumption. Furthermore, migration performance was determined in two stages that comprise migrating the selected VM on the overloaded server and all VMs on the servers in the unused shelf. In [20], authors introduced Fit Decreasing Algorithm that use the Learning Automata theory to improve energy consumption and SLA violation. By this aim, Mosa et al. [21] proposed the method based on Utility-based approach.

Addis et al. [22] proposed a unifying framework which satisfy performance and availability guarantees and minimize energy costs in very large cloud centers. They proposed a scalable distributed hierarchical framework according to a mixed-integer nonlinear optimization data. Arianyan et al. [23] proposed a new holistic cloud resource management procedure and a heuristic according to multi-criteria decision making method for determination of under loaded hosts and placement of the migrating VMs. Kessaci et al. [24] provided an energy-aware Multi-start Local Search algorithm (EMLS-ONC) that optimizes the energy usage of an OpenNebula-based cloud. Dai and Li [25] used immune algorithm for energy optimization in cloud computing by designing an experimental platform for resource allocation, energy optimization and performance analysis. The authors in [26] introduced an adaptive Genetic Algorithm (GA) to diminish energy consumption while considering the response time of the tasks in a cloud data center.

In [27], different types of tasks are considered which are ignored in our paper and could be a future work. In [28], minimization of completion time of tasks considered. Similarly, a multi-objective service placement using Fuzzy algorithm presented in [29].

Compared to the aforementioned related works listed in Table 1, our contributions are:

1. Each host concerning its CPU consumption independently which is able of learning to enhance its performance. As a result of this autonomy, there is no need for a global manager or doing a local search.
2. A few articles use Learning Automata, however their method only focus on designing the special policy or using categorization of VMs, not for physical machines.
3. Authors in the previous works use only one kind of Learning Automata whereas we use two types of LA.
4. In some articles, authors only consider only the overloaded status and ignore other status of the hosts in term of their CPU's usage.
5. Only in a few works two important factors are taken into account including: (i) meantime before a host shutdown and (ii) meantime before a VM migrates. Simulation results show that these factors are important for QoS and energy consumption.
6. Methods that use meta-heuristics like, GA and Ant Colony find the optimal solution after many iterations which imposes a lot of overload and make them inappropriate for dynamic systems.

Table 1 Taxonomy of VM placement approaches in cloud system

Placement approaches	VM or task consolidation	Need to know task execution time	Optimization algorithm method	Objective(s)
EPBLA	VMs	No	Meta heuristics	Energy consumption, meantime before a host shutdown, meantime before a VM migrates
An improved ant colony optimization [9]	VMs	No	Meta heuristics	Energy consumption, communication cost
OEMACS [10]	VMs	No	Local search techniques	Energy consumption, multiple resources
MF and HCMFF [11]	VMs	No	Meta heuristics	Energy consumption, different workload
GATA [12]	VMs	No	Meta heuristics	energy consumption, load balancing
MOSSASCA [13]	VMs	No	Meta heuristics	Energy consumption, meantime before a host shutdown, improving SLA
THE MBFD TECHNIQUE [18]	VMs	No	Modified best fit decreasing	Energy consumption, idle servers and racks
EMLS-ONC-MO [21]	Task	Yes	Multi-start local search algorithm	Energy consumption, VMs efficiency
Adaptive GA [23]	Task	Yes	Integer linear programming, Meta heuristics	Energy consumption, consider the response time
Utility-based approach [25]	VMs	No	Meta heuristics	Energy consumption, improving SLA
Fuzzy-AHP [26]	VMs	Yes	Heuristics	Energy consumption, improvement of performance
Ant colony [27]	VMs	No	First fit, meta heuristics	Energy consumption
UMC [28]	VMs	No	Best fit decreasing, meta heuristics	Energy consumption, improving SLA
Optimal VM placement algorithm [29]	Task	Yes	Queuing Approach	Processing cost

3 Learning automata

3.1 Finite action-set learning automata

Learning Automata [30] is an adaptive decision-making unit that improves its performance by learning how to choose the optimal action from a finite set of allowed actions through repeated interactions with a random environment. The action was chosen randomly based on a probability distribution kept over the action-set and at each instant, the given action was served as the input to the random environment. The environment responded to the taken action in turn with a reinforcement signal. The action probability vector was updated based on the reinforcement feedback from the environment.

The objective of a Learning Automata was to find the optimal action from the action-set so that the average penalty received from the environment is minimized. Learning Automata was found to be useful in systems where incomplete information about the environment exists. Learning Automata were also proved to perform

well in complex, dynamic and random environments with a large amount of uncertainties.

Every environment was represented by $E = \{\alpha, \beta, c\}$, where $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ was a set of inputs, $\beta = \{\beta_1, \beta_2, \dots, \beta_r\}$ is a set of outputs, and $c = \{c_1, c_2, \dots, c_r\}$ was a set of penalty probabilities. Whenever set β has just two members, the environment was a kind of P. In this environment $\beta_1 = 1$, $\beta_2 = 0$ were considered as penalty and reward respectively. Similarly, environment in a kind of Q contained a finite set of members. Also, environment in a kind of S had an infinite number of members. C_i was the penalty probability of taken action α_i .

Learning Automata can be classified into two main families: fixed structure Learning Automata and variable structure Learning Automata. Learning Automata with variable structure was introduced as follows; Learning Automata with variable structure is represented by $\{\alpha, \beta, p, T\}$, where $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ is a set of actions, $\beta = \{\beta_1, \beta_2, \dots, \beta_r\}$ was a set of inputs, $p = \{p_1, p_2, \dots, p_r\}$ is the action probability vector, and $p(n+1) = T[\alpha(n), \beta(n), p(n)]$ was the learning algorithm. Learning Automata operated as follows; Learning Automata chooses an action

from its probability vector randomly (P_i) and took it. Suppose that the chosen action is α_i . Learning Automata after receiving reinforcement signal from environment updated its action probability vector according to formulas 1 and 2 in case of desirable and undesirable received signals respectively. In formulas 1 and 2, a and b are reward and penalty parameters respectively. If $a = b$, then, the algorithm was named *LR-P*. Also, if $b < a$ then the algorithm was named *LR ϵ P*. Similarly, if $b = 0$ then the algorithm was called *LR-I* [31].

$$\begin{aligned} p_i(n+1) &= p_i(n) + a \cdot (1 - p_i(n)) \\ p_j(n+1) &= p_j(n) + a \cdot p_j(n) \end{aligned} \quad (1)$$

$$\begin{aligned} p_i(n) &= (1 - b) \cdot (p_i(n)) \\ p_j(n+1) &= \frac{b}{r-1} + (1 - b)p_j(n) \end{aligned} \quad (2)$$

3.2 Continuous action-set learning automata (CALA)

So far, the LA model was considered where the set of actions was finite. Hence, while finding the optimal parameter values to maximize a performance index, it is needed to discretize the parameter space so that the actions of LA could be the possible values of parameters. A more satisfying solution would be to employ an LA model where the action-set could be continuous. Such a model is called *continuous action-set learning automaton or CALA* [30]. The action-set of CALA was the real line. The action probability distribution at k was $\mathcal{N}(\mu(k), \sigma(k))$, the normal distribution with mean $\mu(k)$ and standard deviation $\sigma(k)$. At each instant, the CALA updated its action probability distribution by updating $\mu(k)$ and $\sigma(k)$. Let $\alpha(k) \in \mathfrak{R}$ be the action chosen at k and let $\beta(k)$ be the reinforcement at k . Here, instead of reward probabilities for various actions, we now have a reward function $\mathcal{F} : \mathfrak{R} \rightarrow \mathfrak{R}$ defined by $f(x) = E[\beta(k) | \alpha(k) = x]$. We shall denote the reinforcement in response to action as $f(x) = E\beta_x$. The objective for CALA was to learn the value of x at which attains a maximum f . That is, we want the action probability distribution $\mathcal{N}(\mu(k), \sigma(k))$ to converge to $N(x_0, 0)$ where x_0 was a maximum of f . However, it was not allowed $\sigma(k)$ converge to zero to ensure that the algorithm did not get stuck at a no optimal point. Therefore, another parameter

was used, $0 < \sigma_\ell$ (with σ_ℓ sufficiently small) and kept the objective of learning as $\sigma(k)$ converging to σ_ℓ and $\mu(k)$ converging to a maximum of f .

The learning algorithm for CALA was described next. Since the updating given for $\sigma(k)$ did not automatically guarantee $\sigma(k) > \sigma_\ell$, a projected version of $\sigma(k)$ is always used, denoted by $\mathcal{Q}(\sigma(k))$, while choosing actions. Furthermore, unlike FALA, CALA interacted with the environment through a choice of two actions at each instant.

At each instant k , CALA chose a $x(k) \in \mathfrak{R}$ at random from its current action probability distribution $\mathcal{N}(\mu(k), \mathcal{Q}(\sigma(k)))$, where \mathcal{Q} was the function specified below. Then, it got the reinforcement from the environment for the two actions: $\mu(k)$ and $x(k)$. Let these reinforcements be β_μ and β_x .

4 Extended placement by using learning automata (EPBLA)

VMs are allocated into two categories: First part is the acceptance of new applications for securing VM and placing VMs on the hosts. The second part is to optimize the current allocation of VMs. The current allocation of the VM optimization is performed in two steps: First, select the VM that requires migration. Secondly, the VM selected in the previous step, is put on a suitable host.

The use of distributed reinforcement learning model might be helpful according to the parallel, non-deterministic, stochastic and distributed nature of cloud systems. One of the reinforcement learning tools is stochastic Learning Automata which are used in this paper to access to distribution feature and the local Community. As described in the previous section, the random automaton without any information about the optimal action (means considering equal probability for all its actions at the beginning) was trying to answer to the problem. (A) As a component of a network of Learning Automata with a finite number of actions in which each automata's decision was independent of the others. The proposed system schema shows in Fig. 1. (B) For learning the optimal value of a continuous parameter, a certain type of Learning Automata is applied to, and have set its action on continuous interval exists. Furthermore, each of these areas are explained completely.

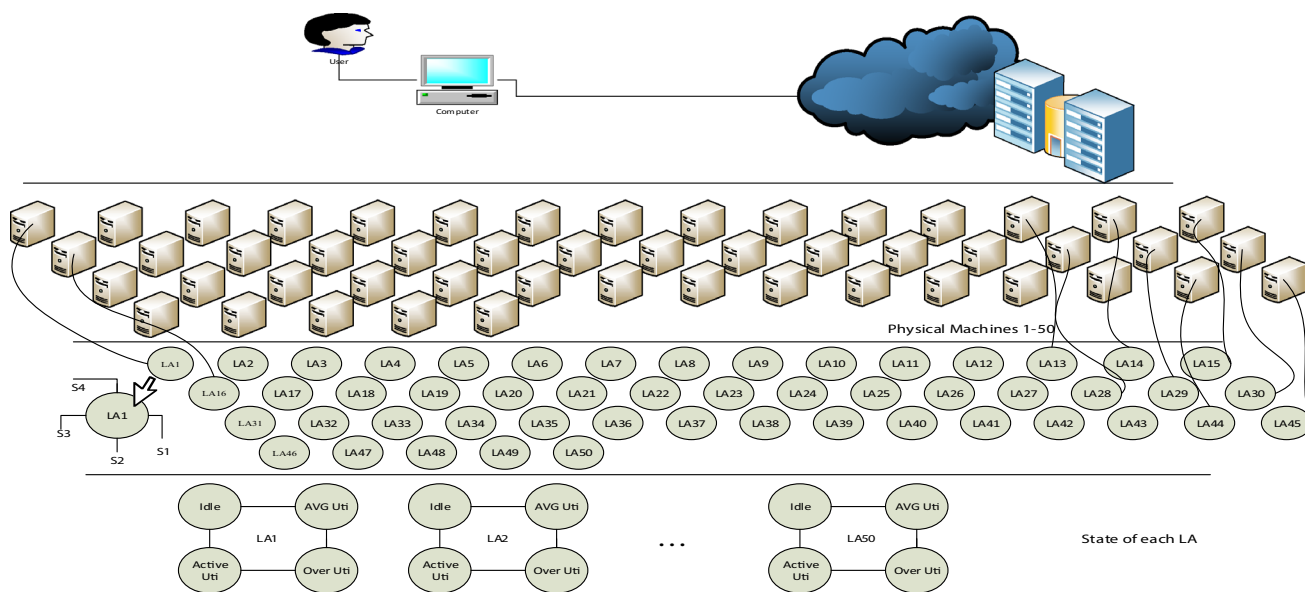


Fig. 1 Proposed system schema

The proposed system in the cloud’s data center has N heterogeneous physical nodes. Any physical nodes are considered as automata which made the decision independently from the others. The categories behavior of the physical nodes as the sender, receiver, and neutral are provided. Their features included: Virtualization, supporting multiple system resources like CPU, disk storage, and network interface. Their Elements are heterogeneous. The techniques to save energy is “turn off the physical machine”. Their goal was to reduce the energy consumption.

VMs: Multiple VMs can start dynamically and they can stop on a physical machine based on the respond to the accepted requests.

Physical machines: Physical servers offered the context of hardware infrastructures physical computing to create virtual resources to meet the demands of the service.

The proposed approach considers in the two-stage optimization phase which is sub-divided into four steps as follows:

Step 1: Select the sender host. To select a set of sender hosts that needs to migrate some of their VMs.

Step 2: Select the VM to migrate. Migration of VMs from the current place to the other physical machines is performed.

Step 3: Select the receiving host. At this stage, a set of receiver hosts can be specified.

Step 4: Allocation of VMs. VM assigned to the appropriate receiver host.

In the following, each of these steps will be explained further. Pseudo-code for EPBLA is presented in Algorithm 1.

Algorithm 1: EPBLA**Result: Return the re-assignment of VMs with low energy consumption**

```

1 Updated at each Time frame
2 Randomly initialize host and assign VMs to them
3 for each  $LA_i \in P_j$  do
4 Initialize  $P_i$  (the actions' probability) equally
5 Calculate the  $p_{cpu}^j$  and determine the state of each  $LA_i$  (Idle, Low Uti, Active, Over)
6 If state of  $LA_i == Idle$  OR Low Uti OR Over then
7 Put  $P_j$  on Sender list candidate
8 end if
9 end for
10 for each  $P_j \in$  Sender list do
11 Use MM policy to put  $V_i$  on candidate migration list (to avoid unnecessary migration use CALA only for underload state  $LA_i$ )
12 end for
13 for each  $LA_i \in P_j$  do
14 if conditions (Eq1. To Eq5.) are fulfill
15 Put  $P_j$  on Receiver list candidate
16 end if
17 end for
18 for  $LA_i \in$  Receiver list do
19 if action of  $LA_i \in$  Receiver list is disabled then
20 do nothing
21 else
22 Choice randomly an action (Accept or Reject) for requested VM
23 end if
24 while Capacity of  $p_{mem}^j \in$  Receiver list available do
25 if action of  $LA_i \in$  Receiver list == accept then
26 Place  $VM_j$  on  $PM_i$ 
27 Calculate and Update amount of  $p_{mem}^j$  &  $p_{cpu}^j$ 
28 Update the state of each (source and destination)  $LA_i$ 
29 Get response of environment  $\beta$  according to the formula:  $\frac{1}{1 + \frac{\text{source state}}{\text{destination state}}}$ 
30 if  $PM_j \in$  active state then
31 Reward or penalty action of  $LA_i$  according to

$$p_i(n+1) = p_i(n) + a \cdot (1 - \beta_i(n)) \cdot (1 - p_j(n)) - a \cdot \beta_i(n) \cdot p_i(n)$$


$$p_j(n+1) = p_j(n) - a \cdot (1 - \beta_i(n)) \cdot p_j(n) + a \cdot \beta_i(n) \cdot \left[ \frac{1}{r-1} - p_j(n) \right] - a \cdot (1 - \beta_i(n)) \cdot p_j(n) \quad \forall j \neq i$$

32 end if
33 end if
34 else if action of  $LA_i \in$  Receiver list == reject then
35 do nothing
36 Repeat lines 27 & 28 again
37 if  $PM_j \in$  idle state then
38 Reward or penalty action of  $LA_i$  according to

$$p_i(n+1) = p_i(n) + a \cdot (1 - \beta_i(n)) \cdot (1 - p_j(n)) - a \cdot \beta_i(n) \cdot p_i(n)$$


$$p_j(n+1) = p_j(n) - a \cdot (1 - \beta_i(n)) \cdot p_j(n) + a \cdot \beta_i(n) \cdot \left[ \frac{1}{r-1} - p_j(n) \right] - a \cdot (1 - \beta_i(n)) \cdot p_j(n) \quad \forall j \neq i$$

39 end if
40 end else if
41 end while
42 end for

```

The first step is the selection of a set of sender hosts that need to migrate some of their VMs. Hosts are classified according to their CPU usage; this classification is based on the fixed parameters in the ranges of 0–1. This classification is updated in each time frame, meaning that the amount of CPU usage of each host is calculated, and the hosts categories are determined. These statuses are mentioned as Overloading host, Underload host, Average utilization host, Active host Selection of the sender host which is done according to which status (The use of the host processor) host is in.

(a) If it is in Overloading host status, then its utilization of CPU is more than a specific value and its VMs is chosen to migrate according to avoid hotSpots and ensuring the QoS. (b) If it is in Active host status then the node is active and fulfilling service level agreements.

And if the VM is taken from it, there is still a sufficient number of VMs on that node, so it cannot be switched off, for these reasons the algorithm will avoid migrating from the active host, so that not to impose the cost of vain migration. (c) If it is in Average utilization host, algorithm tries to migrate its VM and change its status to an underload host so it can be shut down to save more energy in the data center. (d) If the CPU usage is very low (0–0/1) in order to reduce energy consumption, it is better to move VMs on the host and make the idle host turn off.

When a host is selected as the sender, the second step is choosing which VMs have to migrate from the sender host. The Minimization of Migrations (MM) policy migrate the least number of VMs to minimize the migration overhead. This policy is used for choosing which VMs have to migrate in all state of Learning Automata except underload

status. When a host is in the underload status as explained before, its utilization of CPU is very low, so all of its VMs are chosen to migrate according to the amount of saved energy so that all of them must be migrated in a way that the host can be turned off.

Since migration of VM is expensive, it is better to choose less VMs to migrate and avoid unnecessary migrations. When an underload volunteer host is added to its VMs to move to another node, it is essential that all VMs on that nodes find a suitable receiver node to migrate to it and that idle node be turned off. It should be considered that if all VMs are not moved, the underload node should not be turned off, due to the imposed cost of migration systems.

To find out if there is receiver host for all VMs located on an underload node and avoiding unnecessary migration, Learning Automata with continuous action-set is used. With the help of this automata, the probability of finding a receiver host for all VMs settled on the underload host is estimated. If the obtained probability of a certain amount which has been gained from the initial assumption with the use/help of statistical tools is higher than the mentioned probability, the host's VMs will be selected for migration, otherwise, the migration of VMs on that underload host will be regardless. Environment response (β) is considered as a binary value, where one is assumed as a favorable result and zero is considered as the unfavorable response. In underload status all VMs should be migrated as explained, before using CALA for this situation. Algorithm 2 shows pseudo-code of the proposed algorithm. The effect of changing the initial values on the accuracy and precision of the Learning Automata parameters are investigated. The simulations are carried up to 3000 iterations with step size $\lambda = 0.1$, $\sigma_L = 0.01$ and $k = 5$. The results of simulation for various initial conditions are indicated in Table 2.

After the selection of VMs to migrate, the set of receiver hosts must be specified. At this step, a suitable destination host for the selected VM should be found which chooses to migrate from sender host at the previous steps. So the third step is finding a suitable destination host for VMs which were chosen by VM selection policy. The notations used in the paper are defined in Table 3.

To select the receiver host, the host having the following characteristics will be selected.

1. In order to save more energy, the number of turned-off hosts must be raised up, so the receiver host should not be one of the turned-off hosts. To check this condition, Eq. 3 examines whether at least one VM is assigned to the considered host.
2. The receiver host must have sufficient resources to accept the VM.
3. The sender host cannot be a volunteer to receive a VM because by removing a VM from the sender host its status may be changed to idle which can be turned-off. We divide the hosts into two separate lists (sender and receiver). A host is not allowed to participate in both lists simultaneously.
4. Only after the migration process, the state of the host can be specified. So, the receiver host could not accept any other VMs until the migration process is completed because after accepting the VM, it may violate some of the conditions (Eqs. 3 to 7) for the admission.
5. After the adoption of the VM, the condition that the status of the receiver host did not change to overloading status must be established. Because the receiver host must have sufficient space for accepting a VM, on the other hand, according to providing QoS and SLA the overloaded host should be in a sender host list not in a receiver one.

Algorithm 2: Continuous Action-Set Learning Automata (CALA)

Result: the list of VMs to migrate from underload state host.

1 **Receive the underload state host list.**

2 **for** each LA_i that state of $LA_i = \text{Underload}$ **do**

3 Using the mean (μ) and standard deviation (σ) values to count X value randomly.

4 Apply two actions **if** (this value > specified value) **then**

5 Put VMs on that host on the migration list.

6

7 **If** (the host had chosen its guests (VMs) were completely transferred) **then**

8 Give it reward (the bonus: 1).

9 **Else If** (the host had chosen not transferred all its guests (VMs) and residents remained on the source host)

10 **then**

11 Give it penalty (the punishment: 0)

 Then, the distribution was updated as follows:

$$\mu(k+1) = \mu(k) + \lambda \frac{(\beta_x - \beta_\mu)(x(k) - \mu(k))}{\phi(\sigma(k)) \phi(\sigma(k))}$$

$$\sigma(k+1) = \sigma(k) + \lambda \frac{(\beta_x - \beta_\mu)}{\phi(\sigma(k))} \left[\left(\frac{(x(k) - \mu(k))^2}{\phi(\sigma(k))} - 1 \right) + \lambda \{ \mathcal{C}[\sigma_\ell - \sigma(k)] \}$$

12

end for

Table 2 Simulation result of CALA

Initial values		Step(λ)	After 3000 repetition		Final function
$\mu(0)$	$\sigma(0)$		$\mu(8000)$	$\sigma(8000)$	
0.5	0.3	0.1	0.9309	0.2053	0.7409
0.2	0.6	0.1	0.4444	0.1987	0.2204
0.8	0.3	0.1	2.1696	0.195003	2.60029
0.8	0.6	0.1	5.09417	0.205607	5.01161
0.7	0.3	0.1	1.16558	0.120736	304,241
0.4	0.6	0.1	0.07944	0.205096	0.0768
0.4	0.1	0.1	0.0304	0.20316	0.0175

Table 3 Notations used in constrains modeling

Symbol	Definition
M	Number of physical machine
N	Number of VM
V_i	The i th VM
P_j	The j th physical machine
VP_{ij}	VM V_i assign to P_j
V	Set of N VM v_1, v_2, \dots, v_n
p_{cpu}^j	%CPU utilization of P_j
p_{mem}^j	%RAM utilization of P_j
v_{cpu}^i	The CPU demand of V_i
v_{mem}^i	The RAM demand of V_i
O	The host is shutdown
I	At least one VM assign to the host
S	Set of Sender list candidate
R	Set of Reciever list candidate

6. Once the host accepts the VM and before the VM starts to run, the status of the host should be checked to ensure that the host is not overloaded. If so, the migration must be canceled and the VM is removed.

The relations which showed these constraints were listed in Table 4.

The host is evaluated for admitting the selected virtual. After accepting all stipulated conditions, this host is placed in the receiver host list.

At the final step, the selected VMs have to be allocated to the appropriate receiver hosts. This is solved through a network of Learning Automata having a finite actions and independent of other decision making automata. Each automaton is a physical machine in the cloud data center that its learning was done through the acceptance or rejection of the VM from the source node to the destination node.

Each automaton has two actions which only one of them is active at a time. As we explained before, according to the

Table 4 Constraints for choosing a host as a reciever

$\sum_{i=1}^N vp_{ij} \neq 0$	(3)
$(v_{cpu}^i \leq p_{cpu}^j) \text{ and } (v_{mem}^i \leq p_{mem}^j)$	(4)
$P_i \neq P_j \forall P_j \in S$	(5)
$\forall P_j : \text{If } P_j \text{ accept } V_i \text{ then } R - \{P_j\}$	(6)
$\text{If } vp_{ij} = 1 \text{ and } p_{cpu}^j > 0.8 \text{ then } R - \{P_j\}$	(7)

utilization of processors, we define four statuses. From a mathematical point of view, each of these statuses are equivalent to the state of a Markov chain [32]. The transition from one state to another state was identical to the change state of a physical machine to another state after VM admittance. Each automaton has its own Markov chain and at each of these states of the chain are doing their learning separately from other states. By reason of the different benefit of each action in the individual case, without being a particular case, automata would not be capable of learning. According to the source and destination nodes reward and penalty of automata would be determined. In the next iteration, destination host uses this environment response, until the new assignment is created to be closer to the optimal schedule.

In fact, the VM moved from the source host to the destination host by using migration. After receiving the list of selected VMs to migrate and suitable receiver hosts list for the selected VMs, if the host satisfied all conditions mentioned before, the selected action of Learning Automata is checked according to its current status. If the selected action of automata is the acceptance of VM, the VM on the source host that we assume it as a sender host in EPBLA, migrated to that host. We also consider the destination host for the selected VMs as a receiver. After the allocation, the amount of energy uses by both nodes are calculated in watt. But if the selected action of automata is a rejection of the VM, host do not accept the VM although the host satisfies all conditions. The rejected VM tries on the chance of admission another eligible recipient host which would be able to migrate from the source host to that host. The code of the proposed method is uploaded to Github for public access [33].

We assume M as the number of physical machines and N as the number of VMs. If we consider I as the set of idle state node $I \in o(S)$. M_{host} is the capacity of each host in the receiver host candidate list and M_{VM} is the least amount of memory needed by the VM which is selected to migrate on the available receiver host. S is the set of sender list candidate and R is the set of receiver list candidate. Thus, the complexity of the algorithm is $\theta(M) + \theta(S * I) + \theta(M) + (O(R * M_{host}/M_{VM})) \cong o(M^2)$.

5 Evaluation

We improve EPBLA against PBLA by 20% in terms of energy consumption. The results in [1] proved the efficiency of PBLA. For appropriate simulation experiments, the workload of a real system is used. The details of the workload are taken from the CoMon [34] project, a monitoring infrastructure for PlanetLab. We simulate a data center by using CloudSim [35] simulator. We repeat the experiments by different numbers of heterogeneous VMs submitted by the user, and the features of heterogeneous physical nodes are set as the following: EXP1 is a small data center with only 50 hosts and 100 VMs and EXP2 has a large number of hosts 500 and only 100 VMs. The number of VMs is the same in both EXP1 and EXP2 to show the performance of the method for different number of hosts. The various experiments have been conducted to further investigate the effectiveness of the proposed method in a data center with a large number of hosts and VMs, EXP3 has 600 hosts and 1052 VMs which are submitted by the user.

The results from EXP1 is shown in Fig. 2. In this experiment, the number of hosts is half of the number of VMs thus raising migration is natural for proper placement. Due to the number of turned off nodes, the results indicate a good performance of the proposed approach. Due to the reduced size of the data center (low number of hosts), energy consumption has declined but remained high in terms of the rate of violation of the service agreement. Therefore, it can be used in systems that the service level agreement is not critical there. The results of EXP2 are shown in Fig. 3. In EXP1 and EXP2, the number of VMs are identical, however, in EXP2 the number of hosts rises to show the flexibility of the algorithm. The reducing number of migrations beside the increasing number of switch off nodes indicate proper functioning of the proposed approach. Due to the high number of hosts and the

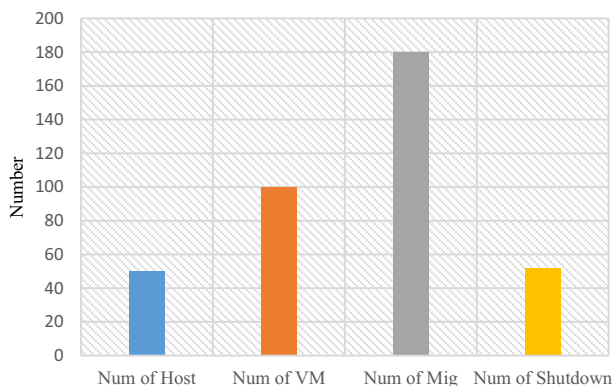


Fig. 2 The result of EXP1

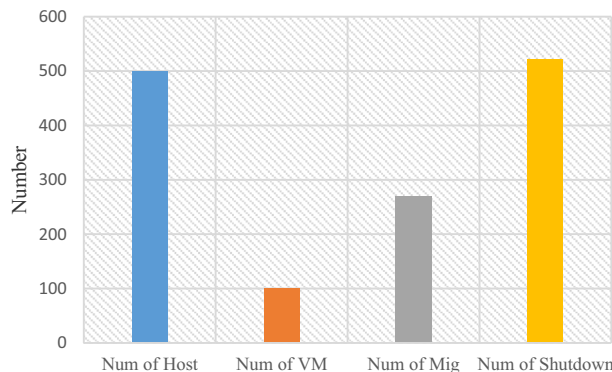


Fig. 3 The result of EXP2

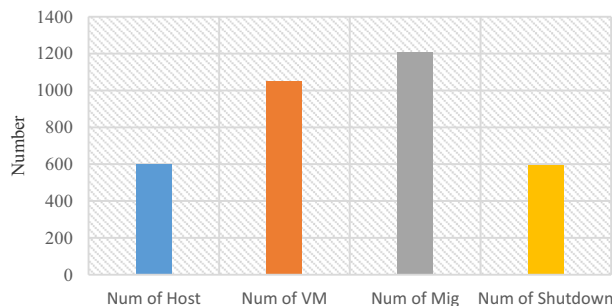


Fig. 4 The result of EXP3

service level agreement violations mean, energy consumption of the system is suitable.

The results of EXP3 is shown in Fig. 4. As seen in the Fig. 4, the number of hosts is much less than VMs; hence, the increase in the number of migrations for proper placement is natural (though the number of migration, the migration is expensive, it is not excessive). Due to the number of inactive nodes, the results indicate a good performance of the proposed approach. Due to the large data center, energy consumption has declined in the system considerably. Reducing energy consumption requires paying the cost of violation of service level agreements. The violation of service level agreement is high. Comparing these three experiments in the amount of energy which

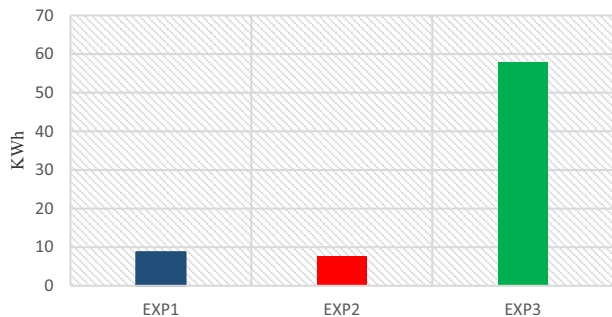


Fig. 5 Comparing 3 experiments, the amount of energy consumption

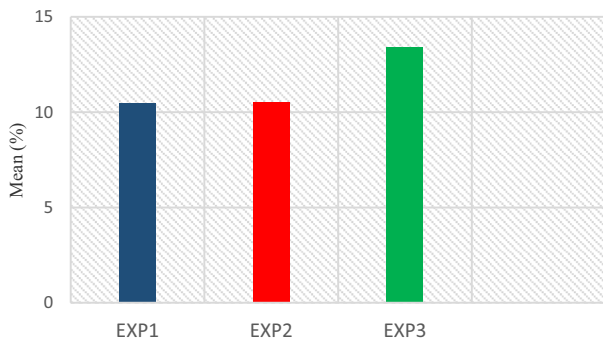


Fig. 6 Comparing 3 experiment, Avg SLA Violation

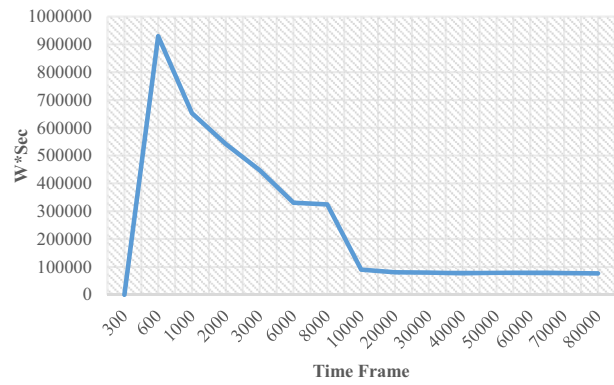


Fig. 8 Energy consumption in the data center for EXP1

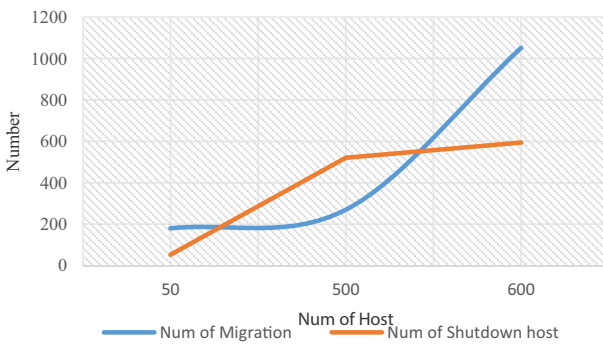


Fig. 7 The impact of data center’s size on number of migration and shutdown hosts

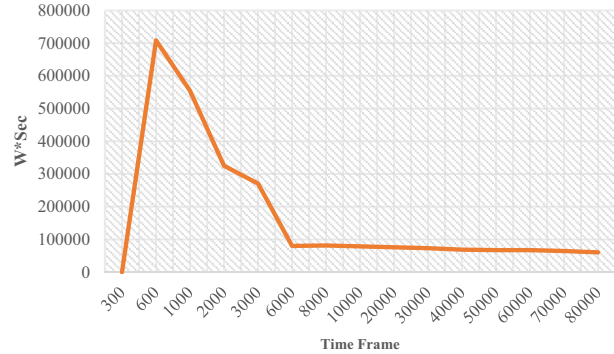


Fig. 9 Energy consumption in the data center for EXP2

each of them used, the result is shown in Fig. 5. In EXP2, which the number of its idle hosts was very high, the energy consumption was certainly less than EXP1. Because, the number of VMs that must allocate to them is very less than number of hosts and the proposed method shows its efficiency. The amount of energy consumption in EXP3, which is a great data center, were more than other experiments and this is perfectly normal. But, the amount of energy in this experience is really low according to its size.

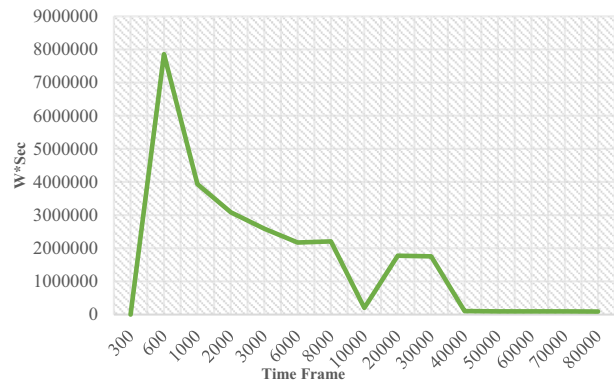


Fig. 10 Energy consumption in the data center for EXP3

Continuing to compare the rates of the three tests as violations of service level agreements can be paid out. The results are shown in Fig. 6. Perhaps, one can understand from the consequences of violating the service level agreement with large data center systems that the value is not change dramatically. This meant that it is not very high; it seems much more in a small data center. The results illustrate that this value depends on the number of physical machines. In EXP 3, average of SLA violation is slightly more than the two other experiments. As the chart shows, there is no significant difference between the amount of the average of SLA violation in Exp1 and Exp2, as a result of their data center’s size that they are identical. The impact of the data center’s size on the number of migration and

switched off hosts according to all three experiments is shown in Fig. 7.

Changes in energy consumption in each time frame for all the experiments are shown in Figs. 8, 9 and 10. As the three charts illustrate when simulations begin, the amount of energy consumption is zero. Then it rises to the highest amount in all experiments, because, VMs are randomly allocated to physical machines based on the non-power-aware method. Then, suddenly it decreases because at each frame the amount of energy that is consumed by each node, takes into account to improve energy consumption. At

6000, the Exp2 has plateau but in Exp1 it happens in the time frame of 10,000. At this time frame, there is a deep reduction in Exp3 then it rises up again until 40,000. In Exp3, after time frame 40,000 as the chart indicates, there is no alternation and the amount of energy consumption is quite reduced. The amount of energy consumed in Exp2 is less than two other experiments and there is less fluctuate in Exp2.

The results obtained from the meantime before a host shutdown in the experiments is shown in Fig. 11. We evaluate the results by using T-Test by 90% accuracy. As the result shows, EXP1 in which the number of hosts is half of the VMs numbers, meantime to shut down a host is much more than the other tests. In EXP2, regarding that many of the hosts are switched off, the required time is much lower than the other two tests and it indicates good performance of the proposed approach. In the Exp3, according to the large data center, this factor plays an important role to reduce more energy consumption.

The results obtained from the meantime before migrating a VM in recent experiments are shown in Fig. 12. We evaluate the results by using T-Test with the 90% confidence all experiments are equal in the time needed before a VM migrates from the host. As the chart illustrates, there is no large difference in the values of these experiments. This factor plays an important role in cloud’s data center, according to the live migration that we use it in our algorithm. Live migration enables the porting of VMs and is carried out in a systematic manner to guarantee minimum operational downtime.

We compare EPBLA with a non-power-aware placement approach where all the hosts consume the maximum power all the time. The results are shown in Fig. 13. As shown in Figs. 8, 9 and 10, at the beginning, our algorithm uses the maximum power, similar to the non-power-aware approach, however, the energy consumption then reduces step by step until reaching an equilibrium state. This results indicate a noticeable improvement in the amount of energy

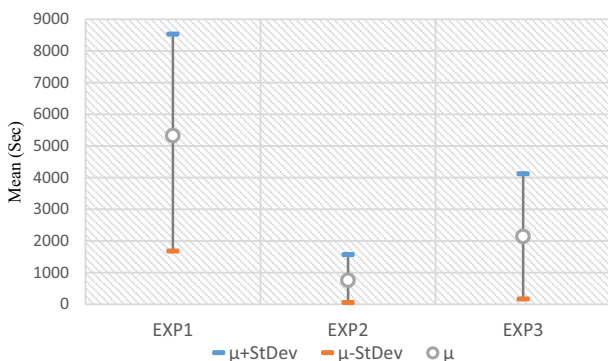


Fig. 11 Comparing three experiments in term of Meantime before a Host Shutdown

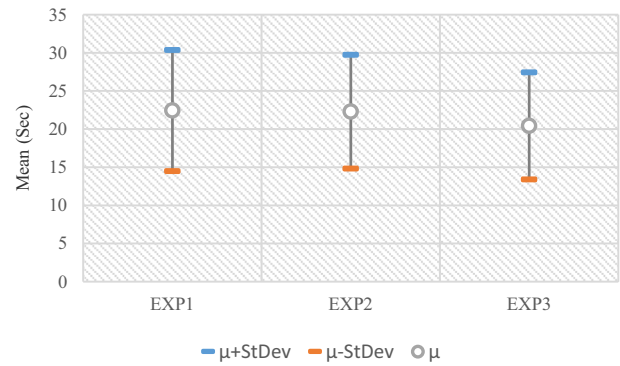


Fig. 12 Comparing three experiments in term of Meantime before a VM Migrate

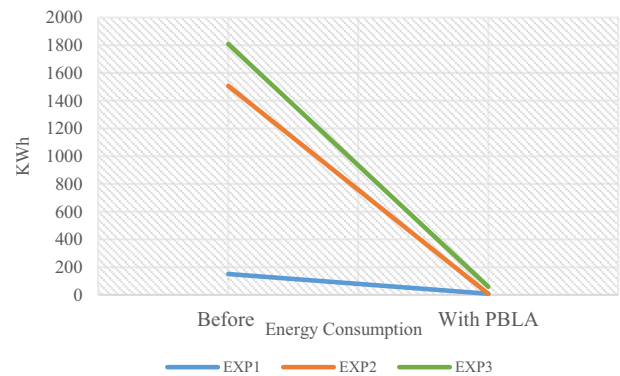


Fig. 13 Compare the amount of energy with and without EPBLA

Table 5 The simulatin results for Workload 1, #Number of VMs:1052, #Number of Hosts:800

Algorithms	Energy (KWh)	SLA	VM migration
<i>IQR-MMT-Firefly</i>	32.21	0.00007	880
<i>LR-MMT-Firefly</i>	32.09	0.00007	874
<i>LRR-MMT-Firefly</i>	30.87	0.00009	971
<i>MAD-MMT-Firefly</i>	32.91	0.00007	824
<i>THR- MMT-Firefly</i>	31.96	0.00009	853
<i>MMT-EPBLA</i>	20.68	0.00006	1334

consumption by 70% compared with the non-power-aware approach.

Tables 5 and 6 give information about the overall performance of all participating parameters with MMT (Selection policy) used in the Firefly method compared with EPBLA for two different workloads. Each of the experiments is run independently and reported in Tables 5 and 6. The results in both workloads illustrate that the Firefly algorithm consumes more energy than EPBLA. Although the number of migrations in EPBLA is slightly

Table 6 The simulation results for Workload 2, #Number of VMs:1516, #Number of Hosts:800

	Energy (KWh)	SLA	VM migration
<i>IQR-MMT-Firefly</i>	33.29	0.00008	898
<i>LR-MMT-Firefly</i>	35.33	0.00010	867
<i>LRR-MMT-Firefly</i>	36.81	0.00010	988
<i>MAD-MMT-Firefly</i>	33.93	0.00011	855
<i>THR- MMT-Firefly</i>	35.56	0.00010	888
<i>MMT-EPBLA</i>	26.17	0.00007	1874

Table 7 Comparing the different methods with EPBLA

	Improvement for Workload 1	Improvement for Workload 2	Average improvement
<i>IQR-MMT-Firefly</i>	36%	21%	29%
<i>LR-MMT-Firefly</i>	36%	26%	31%
<i>LRR-MMT-Firefly</i>	33%	29%	31%
<i>MAD-MMT-Firefly</i>	37%	23%	30%
<i>THR- MMT-Firefly</i>	35%	26%	31%

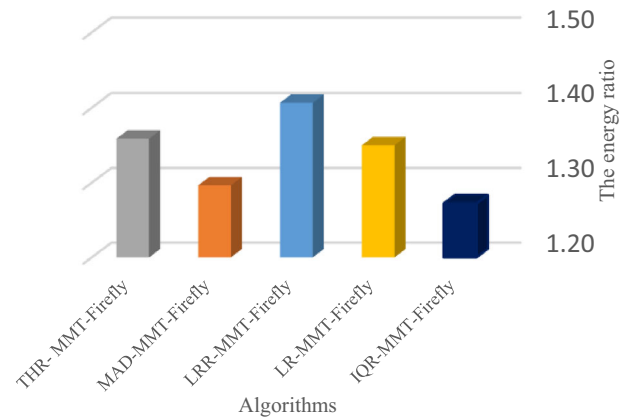


Fig. 15 Energy consumption ratio in Firefly compared to EPBLA for Workload 2

percentage of our method compared to each baseline is reported.

The ratio of energy consumption between the Firefly algorithm and the proposed method is calculated using Eq. 8 which is used in Figs. 14 and 15. These charts show the efficiency of our algorithm in comparison with all the selection policies using the Firefly method.

$$Energy\ ratio = Energy\ of\ method\ X / Energy\ of\ EPBLA \tag{8}$$

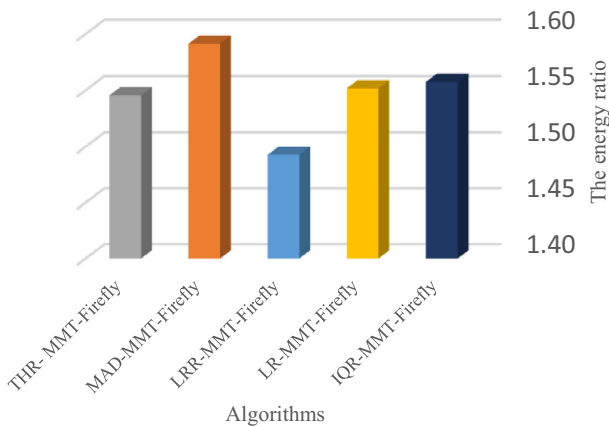


Fig. 14 Energy consumption ratio in Firefly compared to EPBLA for Workload 1

more than Firefly, it outperforms the other baselines in terms of both energy consumption and the percentage of SLA violation. According to Table 6, even when the number of VMs grows the proposed algorithm performs well in reducing the total energy consumption.

Table 7 indicates an overall comparison of our methods with other baselines where the average improvement

6 Conclusion

The VMs placement is a complex NP-hard problem. To solve this problem, in this paper we presented a new approach based on Learning Automata for dynamic replacement of VMs among the physical hosts in a data center. We used dynamic migration to make as much as idle host as possible and forcing idle hosts to shut down. The algorithm could considerably diminish energy consumption while maintaining QoS and thereby reducing the heat and greenhouse gases. Experimental results demonstrated that EPBLA reduced the energy consumed in the considered data center by 20% compared to PBLA (our previous work) while maintaining the quality of services. Compared with a non-power-aware method, the energy consumption decreased by 70%. In comparison with the Firefly algorithm, our approach outperformed to save energy of the data center about 30%. Moreover, the proposed approach could effectively deal with the heterogeneity of cloud data centers. Due to the stochastic nature of availability of physical hosts in the cloud environments, the use of reinforcement learning model is promising. For future work, we aim to consider a new policy for choosing

VMs to migrate. In the migration phase, we take into account the characteristics of tasks running on the VMs. For example, if a real-time task is running on the VM, the VM migration is performed with respect to the real-time constraints of the task. This implies that the VM may not be allowed to migrate or may need to migrate to a certain host respecting the deadline of the task.

References

- Barroso, L.A., Hölzle, U.: *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan & Claypool Publishers, San Rafael (2009)
- Beloglazov, R. B.: Energy efficient resource management in virtualized cloud data centers. In: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 826–831, Melbourne, Australia, 17–20 May (2010)
- Babiceanu, R.F., Seker, R.: Big data and virtualization for manufacturing cyber-physical systems: a survey of the current status and future outlook. *Comput. Ind.* **81**, 128–137 (2016)
- Lu, Y., Xu, X.: Resource virtualization: a core technology for developing cyber-physical production systems. *J. Manuf. Syst.* **47**, 128–140 (2018)
- Rasouli, N., Meybodi, M.R., and Morshedlou, H.: Virtual machine placement in cloud systems using learning automata. In: 2013 13th Iranian Conference on Fuzzy Systems (IFSC). IEEE (2013)
- Hu, C., Xu, C., Cao, X., Zhang, P.: Study on the multi-granularity virtualization of manufacturing resources. In: ASME 2013 International Manufacturing Science and Engineering Conference collocated with 41st North American Manufacturing Research Conference of the American Society of Mechanical Engineers (2013)
- Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Fut. Gener. Comput. Syst.* **28**(5), 755–768 (2012)
- Tsai, J.-T., Fang, J.-C., Chou, J.-H.: Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. *Comput. Oper. Res.* **40**(12), 3045–3055 (2013)
- Wei, W., Gu, H., Lu, W., Zhou, T., Liu, X.: Energy efficient virtual machine placement with an improved ant colony optimization over data center networks. *IEEE Access* **7**, 60617–60625 (2019)
- 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 (2016)
- Ahmed, A., Ibrahim, M.: Analysis of energy saving approaches in cloud computing using ant colony and first fit algorithms. *Int. J. Adv. Comput. Sci. Appl.* **8**, 248 (2017)
- Barlaskar, E., Singh, Y.J., Issac, B.: Energy-efficient virtual machine placement using enhanced firefly algorithm. *Multiagent Grid Syst.* **12**(3), 167–198 (2016)
- Zhao, D.-M., Zhou, J.-T., Li, K.: An energy-aware algorithm for virtual machine placement in cloud computing. *IEEE Access* **7**, 55659–55668 (2019)
- Alresheedi, S., Lu, S., Elaziz, M.A., Ewees, A.A.: Improved multi objective slap swarm optimization for virtual machine placement in cloud computing. *Human-centric Comput. Inf. Sci.* **9**(1), 15 (2019)
- Nguyen, T.H., Di Francesco, M., and Yla-Jaaski, A.: Virtual machine consolidation with multiple usage prediction for energy-efficient cloud data centers. In: *IEEE Transactions on Services Computing* (2017)
- Shu, W., Wang, W., Wang, Y.: A novel energy-efficient resource allocation algorithm based on immune clonal optimization for green cloud computing. *EURASIP J. Wirel. Commun. Netw.* **2014**(1), 64 (2014)
- Shaw, R., Howley, E., Barrett, E.: An energy efficient anti-correlated virtual machine placement algorithm using resource usage predictions. *Simul. Model. Pract. Theory* **93**, 322–342 (2019)
- Ranjbari, M., Torkestani, J.A.: A Learning Automata-based algorithm for energy and SLA efficient consolidation of virtual machines in cloud data centers. *J. Parallel Distrib. Comput.* **113**, 55–62 (2018)
- Esfandiarpour, S., Pahlavan, A., Goudarzi, M.: Structure-aware online virtual machine consolidation for data center energy improvement in cloud computing. *Comput. Electr. Eng.* **42**, 74–89 (2015)
- Ghobaei-Arani, M., Rahmanian, A., Shamsi, M., Rasouli-Kenari, A.: A learning-based approach for virtual machine placement in cloud data centers. *Int. J. Commun. Syst.* **31**, e3537 (2018)
- Mosa, A., Paton, N.W.: Optimizing virtual machine placement for energy and SLA in clouds using utility functions. *J. Cloud Comp.* **5**, 17 (2016). <https://doi.org/10.1186/s13677-016-0067-7>
- Addis, B., Ardagna, D., Panicucci, B., Squillante, M.S., Zhang, L.: A hierarchical approach for the resource management of very large cloud platforms. *IEEE Trans. Depend. Secure Comput.* **10**(5), 253–272 (2013)
- Arianyan, E., Taheri, H., Sharian, S.: Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers. *Comput Electr Eng* **47**, 222–240 (2015)
- Kessaci, Y., Melab, N., Talbi, E.-G.: A multi-start local search heuristic for an energy efficient VMs assignment on top of the open-nebula cloud manager. *Fut. Gener. Comput. Syst.* **36**, 237–256 (2014)
- Dai, L., Li, J.H.: An optimal resource allocation algorithm in cloud computing environment. *Appl. Mech. Mater.* **733**, 779–783 (2015)
- Ibrahim, H., Aburukba, R.O., El-Fakih, K.: An integer linear programming model and adaptive genetic algorithm approach to minimize energy consumption of cloud computing data centers. *Comput. Electr. Eng.* **67**, 551–565 (2018)
- Baccarelli, E., et al.: Q*: energy and delay-efficient dynamic queue management in TCP/IP virtualized data centers. *Comput. Commun.* **102**, 89–106 (2017)
- Ponraj, A.: Optimistic virtual machine placement in cloud data centers using queuing approach. *Fut. Gener. Comput. Syst.* **93**, 338–344 (2019)
- Son, A., Huh, E.-N.: Multi-objective service placement scheme based on fuzzy-AHP system for distributed cloud computing. *Appl. Sci.* **9**, 3550 (2019)
- Thathachar, M.A.L., Sastry, P.S.: Varieties of learning automata: an overview. *IEEE Trans. Syst. Man Cybern. B (Cybernetics)* **32**, 711–722 (2002)
- Narendra, K.S., Thathachar, M.A.L.: Learning automata—a survey. *IEEE Trans. Syst. Man Cybern.* **4**, 323–334 (1972)
- Harmon, R., Challenor, P.: A Markov Chain Monte Carlo method for estimation and assimilation into models. *Ecol. Model.* **101**(1), 41–59 (1997)
- Rasouli, N. (2019). <https://github.com/TanazR/PBLA-.git>

34. Park, K.S., Vivek, S.P.: CoMon: a mostly-scalable monitoring system for PlanetLab. *ACM SIGOPS Oper. Syst. Rev.* **40**(1), 65–74 (2006)
35. Buyya, R., Ranjan, R., Calheiros, R.N.: Modelling and simulation of scalable cloud computing environments and the CloudSim toolkit: challenges and opportunities. In: *Proceedings of the 7th High Performance Computing and Simulation Conference HPCS2009*, pp. 1–11, IEEE Computer Society (2009)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Nayereh Rasouli graduated from the university of Science and Culture with a bachelor's of the Computer Science. She received her M.S. in Computer Science in 2013 at Gazvin Islamic Azad University. Now, she works as a lecturer at the university. Her research interest includes Cloud and Edge Computing, Network and IoT. She is also involved in some ongoing research works focusing on the optimal placement of SDN-controller nodes in Industrial

IoT systems.



Ramin Razavi is currently a Ph.D. candidate at the University of Tehran. He received a master's degree in computer engineering from the University of Tehran in 2014. He has several years of experience in teaching courses in the computer field such as data structure, algorithm design, and discrete mathematics. His research interest includes Real-time scheduling, Cloud and Edge computing.



Hamid Reza Faragardi has got a Ph.D. in Computer Eng. from MDH University, Sweden. He was then a postdoc researcher at the University of Innsbruck for 1.5 years. He is a lecturer at the department of computer science and communication at KTH Royal Institute of Technology. Now he also works in the research department of Ericsson mainly on Cloud and Edge Computing. He is also involved in some ongoing research works focusing on the optimal placement of SDN-controller nodes in Industrial IoT systems.