Computer & Robotics

A MAPE-K Loop Based Model for Virtual Machine Consolidation in Cloud Data Centers

Negin Najafizadegan[a], Eslam Nazemi[b], Vahid Khajehvand[a]

[a] *Faculty of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*
[b] *Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran*

Received 20 October 2020; Accepted 23 December 2020

Abstract

Today, with the rise of cloud data centers, power consumption has increased and cloud infrastructure management has become more complex. On the other hand, meeting the needs of cloud users is an important goal in the cloud infrastructure. To solve such problems, an autonomous model with predictive capability is needed to do virtual machine consolidation at runtime effectively. In fact, using the feedback system of autonomous systems can make this process simpler and more optimized. The goal of this research is to propose a cloud resource management model that makes the virtual machine consolidation process autonomous, and by using a prediction method compromises between service level agreement violations and energy consumption reduction. In this research, an autonomous model is presented which detects overloaded servers in the analysis phase by a prediction algorithm. Also, at the planning phase, a multi heuristic algorithm based on learning automata is proposed to find proper servers for virtual machine placement. Cloudsim version 3.0.3 was used to evaluate the proposed model. The results show that the proposed model has reduced averagely the service level agreement violations, energy and migration counts by 67.08%, 11.61% and 70.64% respectively, compared to other methods.

Keywords: Autonomous, Cloud Environment, Virtual machine, Prediction, Learning automata, Service Level Agreement

1.Introduction

Cloud computing is a popular computational pattern and an internet-based environment that data and applications can be shared on it. In this environment customers can use their required applications for a specified time by payment and thousands of networked systems can share their resources and access to the others [1]. Cloud computing proposes the resources as the services using virtualization technology and provides a software environment in the Virtual Machine (VM) format. Virtualization is one of the most important technologies that recently has affected computations and can execute several operating systems on a computer simultaneously [2, 3].

When a server does not have sufficient resources to meet its requirements, it is named hot spot meaning that it is overloaded, and it needs to migrate some of its VMs to another server. In addition to service level agreement (SLA) violation, hot spot results in an increment of the server temperature due to high processor consumption. The resulted high temperature leads to the increment of power consumption of cooling systems. Moreover, the busy processor makes the longer task execution time that leads to longer activation of the servers [4, 5].

* Corresponding Author Email: nazemi@sbu.ac.ir

The movement process of a VM to another server is called migration. Live migration of VMs without disruption of their services or end users disruption is live migration that results in a minimum time of service not working and moves all the VM to the server [5]. Virtual machine migration is one of the prominent features of virtualization that allows to compensate for the lack of resources if the workload increases [6]. Three basic questions that should be answered for VM migration are as the following:

1. Which VMs should be immigrated?
2. Where the VMs should be immigrated?
3. When the VMs should be migrated? [5]

This paper focuses on the questions 2 and 3. The second question is expressed in the form of virtual machine placement and the third question is explained in the form of overloaded server identification.

In cloud data centers, overloaded servers result in load imbalance, and the imbalanced load reduces the quality of service. To remove such an overloaded situation, VMs should be moved from the overloaded servers to underloaded ones. Thus, VMs migration to eliminate the hot spots to guarantee the SLA of applications in cloud data centers is essential [7].

The placement scheme of the VMs is classified to static and dynamic groups. In static VM placement, the mapping of virtual machine to physical machine is constant throughout the life of a virtual machine, but in dynamic mode it is possible to change the initial placement of virtual machine on physical machine for reasons such as system load change. Moreover, the dynamic VM placement algorithms are categorized into reactive and proactive groups. In the reactive type after the system reaches a specified undesired state, the change are applied but in the preventive mode, before the system reaches a specified undesired state, the VM's server is changed [8].

In virtual cloud environments, the main issue is that which server is selected for VMs placement firstly or where the VMs should migrate. These decisions should make autonomously by the management tool of cloud, based on the users' behavior, the users that no determine their VM location. Although VMs migration is essential for resource management of the cloud environment, usually, they have a high overload and reduce the performance of physical machines and network switches [9]. Such migrations, if done at too much and in inappropriate times, not only the performance of virtual machine services but also because of the cloud sharing environment, compromises the performance of other cloud virtual machine services [5, 10, 11]. Therefore reducing waste migration is an important research challenge.

In the cloud computing environment, autonomous computing has attracted particular attention in recent years, which dynamically adapting cloud resources and services to changes. Indeed, the aim of autonomous management is to manage the cloud resources with minimum humans' intervention. It uses a MAPE-K loop commonly known as autonomous administrator. An autonomous administrator uses a knowledge base to collect monitored data from cloud resources, analyzes them, and generates a series of planned changes to run on managed cloud resources [12-15].

To manage energy consumption and resources, the virtual machine consolidation problem must be solved. This problem itself consists of four sub-problems: (1) Identifying overloaded hosts; (2) identifying under-loaded hosts; (3) migrant virtual machine selection from overloaded hosts; (4) virtual machine placement on suitable destination host [16]. This paper focuses on sub-problems (1) and (4).

This paper focuses on the problem of dynamic and predictive virtual machines placement in the cloud. To solve the problem, the autonomous administrator is used which apply the prediction algorithm in the analysis phase and learning automata algorithm in the planning phase to finally achieve the best possible trade-off between consumed energy and SLA violations reduction with minimum human intervention while decreasing the virtual machine migration counts. The most important innovations of this paper are:

- Customizing an autonomic VM consolidation model based on the MAPE-K loop.
- Designing a novel multi heuristics method for virtual machine placement in cloud environment
- Improving the performance of analysis phase by an ensemble prediction algorithm
- Enhancing the flexibility of planning phase by learning automata algorithm

The paper is segmented as follows: Section 2 briefly describes the related past works, section 3 explains the background of the concepts used in the research, in section 4, the problem is formulated and the assumptions are stated, section 5 presents the proposed model along with algorithms described in each phase, in section 6 the proposed method is evaluated by CloudSim simulator and compared with other methods, and ultimately section 7 explains about conclusions and future work.

2.Related Works

The on-time process of overloaded servers identification and virtual machine placement can reduce consumed energy and the number of SLA violations. Using IBM self-adaptive loop can make the process autonomous. This section reviews past works on the servers load prediction, virtual machine placement, and the use of MAPE-K loop in cloud resource management. Related works is divided into three categories:

1. Physical machines load prediction methods
2. Virtual machine placement methods
3. MAPE-K loop models for cloud resource management

2.1. Physical Machines Load Prediction Methods

A predictive virtual machine placement requires a prediction algorithm to detect overloaded servers. Using a proper prediction algorithm, overloaded servers can be identified before they cause SLA violations. This subsection addresses research about server load prediction.

Beloglazov et al. [16] has used linear (LR) to predict the CPU utilization of servers in a time series method. The proposed method identified overloaded physical machines at each time interval. Xiao et al. [17] provided a load forecasting algorithm that predicted the amount of resource consumption by applications in the future without dealing with virtual machines. This paper has used exponentially weighted moving average method. Shaw et al. [4] presented a time series-based forecasting method that used simple exponential smoothing and double exponential smoothing to predict the load of physical machines. In this method, the current and future loads of the physical machine are required to identify overloaded physical machine.

Dinda et al. [18] designed a prediction software that predicts the future load of servers based on their CPU utilization history. They investigated the performance of some linear prediction models. Autoregressive (AR) has been selected as the best prediction model, because of low overload. Liang et al [19] presented a multi-source prediction model for server load prediction. This model predicts a server's load by calculating its relationship to other sources. So a source behavior is almost like a source with which it is related. Arianyan et al. [20] has proposed a window moving average policy for identifying overloaded servers that takes into account all input criteria in the decision making process and decreases virtual machine migration event caused by sudden load increases.

The paper [21] proposed a hybrid prediction model for energy forecasting of virtual machines that has high accuracy due to the use of several forecasting models. Because the amount of cloud resources use varies dramatically, so determining a model for predicting cloud resource usage not only depends on time but also depends on trend of resource usage change. As a result, using hybrid prediction algorithms that combine multiple prediction models can be useful for achieving the desired goal [22]. Table 1 shows the comparison of server load prediction methods. Since the loading of physical machines fluctuates, a forecasting method should be used that can take into

account changes in the workload and predict workload fluctuations. Also, the accuracy of the forecasting method must be independent of the type of workload. By using only one predictive model, the probability of predictive error is high. The proposed forecasting method uses a combination of linear and nonlinear prediction models to predict the workload of physical machines and also considers the workload fluctuations in the forecast.

Table 1
Server load prediction methods comparison

Method	Advantages	Disadvantages
[17]	Reduces the number of migrations and overloaded physical machines	Using only one prediction model
[16]	Has little computational overhead	Using only one prediction model
[4]	Before migrating, considered the current and future state of the physical machine's load	Due to the dynamic workload, it is difficult to determine the method parameter
[18]	Has little computational overhead	Does not take into account workload changes
[19]	High prediction accuracy in both workstations and grid environment	Inability to predict workload fluctuations
[20]	Reduces virtual machine migration	Using only one prediction model in each time interval

2.2. Virtual Machine Placement Methods

In dynamic VM placement techniques, the server of the VMs can be changed during execution. In this section, some of the researches about VM placement has been reviewed.

Basu et al. [1] used an improved genetic algorithm for reducing energy consumption and performing VMs scheduling. Hence, the servers do not become overloaded or underloaded. Each chromosome of the population considered as a server, and each VM assigned to a server. The VMs on the server is like the genes of the chromosome. Crossover and mutation operations were performed after the optimization operation to obtain task allocation results. This improves load balancing and resource utilization. In fact, this paper presents a modified genetic algorithm with local search optimization that reduces memory and power consumption, however in the evaluation of the algorithm, the number of VM migrations that results in SLA violation increment, is not investigated. Alharbi et al. [23] considered virtual machine placement in a data center as a finite hybrid optimization problem and used virtual and physical machine profile information to minimize energy consumption of the entire active physical machines. They also combined an ant colony system with new heuristics to present a solution for energy optimization problem. This paper didn't consider SLA violation in

Shawa et al. [24] focused on the improvement of the VM placement problem by considering the relation between the migrating VMs prior to placement. Based on the prediction results, this method combined the VMs which need each other to complete the execution, as well as improved energy consumption and efficiency. Furthermore, the processor resource and required network bandwidth of the migrated VMs were gained and used in VM placement strategy. However, this method does not take into account the number of virtual machines and amount of energy of the servers in selecting destination physical machine. Ghobaei et al. [22] presented an algorithm for VMs allocation that reduces energy consumption and SLA violations. The proposed method is based on the best fit decreasing algorithm which uses learning automata theory, correlation coefficient and group prediction algorithm in virtual machine allocation. This paper does not consider the number of VMs of servers in physical machine selection process. .

Farahnakian et al. [25] introduced a distributed system architecture that dynamically consolidate virtual machines to reduce data center energy consumption while maintaining optimum service quality. Also here an online heuristic ant colony optimization algorithm has been used. Hallawi et al. [26] proposed a new approach using genetic algorithms for virtual machine consolidation. It tries to find the best solutions for bin packing problem. In this paper SLA violation was not considered. Ferdaus et al. [27] provided a VM consolidation scheme that focused on balancing utilization of the servers' resources

among different computation resources. This paper uses ant colony optimization and vector algebra. This paper did not consider the SLA violations in the proposed method.

Teng et al. [28] considered a batch-oriented consolidation and online placement for reserved VMs and on-demand VMs, respectively. They determined the most appropriate voltage frequencies that are only based on the type of processor to reduce energy and provided an upper bound of energy saving through DVFS techniques. Moreover for online states, an online time balancing heuristic method was designed for VM placement which is on-demand and can reduces transitions between modes by time balancing and server utilization. The proposed method saved energy and provided efficient SLAs but did not investigate the number of VMs migrations. In [16] Beloglazov et al. proposed a new adaptive heuristic to consolidate virtual machines dynamically based on past data analysis of VMs resource consumption. The proposed method reduced energy and met SLA requirements. This paper does not consider the number of VMs of servers in physical machine selection process.

Beloglazov et al. in [29] defined an architectural framework and a set of rules for energy-efficient cloud computing. They also provided algorithms to provision and allocate resources for cloud computing management in a way that saved energy. The presented method provided data center resources for customer applications in a way that reduced energy consumption while maintaining agreed service quality but the weights of their selection metrics are equal in different time intervals.

Horri et al. [30] presented a quality-aware virtual machine consolidation method that operated based on the history of VM resource utilization. This paper improved service quality and energy consumption, but does not consider the number of VMs of servers in physical machine selection process. Arianyan at al. [31] presented a cloud resource management procedure as well as a multi-criteria decision-making method for both determination of underloaded physical machines and placement of migrating VMs. The proposed method reduced energy, SLA violations and migration counts but they considered equal weights for all considered criteria. Table 2 compares the related works about VM placement and Table 3 shows advantages and disadvantage of compared methods.

Table 2
Objective metrics for previous VM placement methods

Ref	Objective Metrics					Approach
	SLAV	ESM	ESV	Energy	Migrations	
[1]				✓		Genetic Algorithm
[23]				✓		Ant Colony
[24]	✓			✓	✓	Anti-correlated VMP
[22]	✓			✓	✓	Learning Automata
[25]	✓			✓	✓	Ant Colony
[26]				✓		Genetic Algorithm
[27]				✓		ACO Metaheuristic
[28]	✓			✓		TRP and OTP heuristics
[16]	✓			✓	✓	Adaptive heuristics
[29]	✓			✓	✓	Multi heuristic
[30]	✓		✓	✓	✓	Correlation based
[31]	✓	✓	✓	✓	✓	Multi heuristic
Proposed	✓	✓	✓	✓	✓	Multi heuristic and LA

Table 3
Advantages and disadvantages of previous VM placement methods

Ref.	Advantages	Disadvantages
[1]	Reduces memory and power consumption parameters	Does not take into account the number of virtual machine migrations
[23]	Reduces energy consumption	Does not consider SLA violation in evaluation metrics
[24]	Improves energy consumption	does not take into account the amount of energy in server selection
[22]	Reduces energy consumption and SLA violations	does not consider the number of VMs of servers in physical machine selection process
[25]	Reduces energy consumption	Does not examine the properties of physical machines for selecting the appropriate physical machine.
[26]	Reduces energy consumption	Does not consider service level agreement violations
[27]	Reduces power consumption and wastage of resources	Does not consider service level agreement violations

[28]	Reduces energy consumption and ensures efficient service level agreement	Does not consider the number of virtual machine migrations
[16]	Reduces energy consumption and service level agreement violations	Only considers one criterion for selecting the destination physical machine
[29]	Reduces energy consumption while maintaining agreed service quality.	Considers equal weight for different selection metrics.
[30]	Improves service quality and energy consumption criteria.	Considers equal weight for different selection metrics in different time intervals.
[31]	Reduces energy consumption, SLA violations and the number of virtual machines	Does not detect overloaded physical machines

2.3. MAPE-K Loop Models For Cloud Resource Management

Cloud resource management process can be autonomous by using MAPE-K loop. The following works each provides a model based on IBM self-adaptive loop in the cloud computing environment.

Singh et al. [32] proposed an autonomous energy-aware cloud system for scheduling cloud resources in data centers. This paper focuses only on the energy and does not address the SLA violation criterion. Singh et al. [33] introduced an autonomous resource management approach that is aware of the SLA. The proposed method reduces SLA violations and optimizes quality of service parameters that are effective in delivering efficient cloud services but energy consumption reduction is not investigated. Singh et al. [34] presented an autonomous energy-efficient resource scheduling framework for scheduling cloud computing resources in data centers. The proposed framework is based on fuzzy logic and schedules cloud computing resources to be energy efficient. This paper does not consider the SLA violations in resource scheduling. Singh et al. [35] introduced an autonomous resource management approach that is aware of quality of service, which configures applications and has self-optimizing feature to maximize resource utilization. The proposed method reduces cost, energy, runtime, SLA violations, and resource conflict. Ghobaei et al. [36] proposed a hybrid approach for cloud service provisioning based on the combination of autonomous computing and reinforcement learning. The paper also presents a framework for autonomous resource provisioning based on the cloud layered model. The proposed method has reduced costs and increased resource utilization but does not address energy savings and does not investigate VM placement. These challenges will be addressed in our research.

Outin et al. [37] proposed a system that uses genetic algorithm to optimize energy consumption. This paper also used machine learning technologies to improve the fitness function of a real distributed cluster of servers. The proposed method presents an energy model but does not address the virtual machines migration issue. Hadded et al. [12] proposed a new method to optimize autonomous management of service-based applications, which minimized the cost of autonomous administrators to prevent bottlenecks in management, as well as the placement cost (virtual machine-to-machine communication cost). This paper presents two algorithms. One algorithm determines the number of optimal autonomous managers to manage service-based applications, and the other, approximates the optimal placement of autonomous managers in the cloud. But this paper does not consider virtual machine placement in the cloud. Maurer et al. [38] discussed first steps towards revealing the current MAPE-K loops for the application of cloud infrastructures. It also proposes techniques for cloud monitoring and discusses the knowledge management approach, and finally provides solutions for managing SLA. This paper does not do anything about VM placement.
All the works presented in this section were about using MAPE-K loop for energy management or resource provisioning in the cloud, and none of them has addressed the problem of VM consolidation. Table 4 shows the comparison of the previous MAPE-K loop models for cloud resource management.

The three criteria of SLA violation, consumed energy and virtual machine migration counts are very important in the issue of VM consolidation. To reduce SLA violations, overloaded physical machines should be identified by predictive methods. Some of past

work on VM placement didn't consider both SLA violations and virtual machine migration counts metrics, or not to provide a way to determine overloaded physical machines. In this paper, we propose an autonomous VM consolidation model based on MAPE-K loop which in the analysis phase applies a weighted ensemble prediction algorithm consists of moving average, weighted moving average and polynomial regression model for overloaded physical machines detection. For VM placement, in the planning phase, physical machines will be investigated in the terms of energy, load, SLA violations and number of VMs by self-adaptive weights. The proposed model will be able to simultaneously reduce energy, SLA violations and number of migration counts.

Table 4
MAPE-K loop-based models for cloud resource management

Methods	Advantages	Disadvantages
Singh [32]	Automatically optimizes the efficiency of cloud resources by reducing energy consumption.	Does not address SLA violations
Singh [33]	Reduce SLA violation and optimize QoS parameters	Has done nothing to reduce energy consumption.
Singh [34]	Works efficiently in energy consumption. .	Does not consider SLA violation
Singh [35]	Reduces cost, energy, runtime, SLA violations, and resource conflict	It has not explored VM placement in reducing energy consumption
Ghobaei [36]	It has reduced costs and increased resource utilization	Does not do anything to reduce energy
Outin [37]	It has presented an energy model.	Does not address the VM migration issue
Hadded [12]	Minimized the cost of autonomous administrators and communication between virtual machines	Does not do anything in VM placement
Maurer [38]	Provides a self-adaptive loop for cloud applications and offers solutions for managing service level agreement	Has not done anything about virtual machine placement

3.Background

This section provides a brief explanation about autonomous computing and learning automata method.

3.1. Autonomous Computing

The term autonomous computing was first used by IBM in 2001 to describe self-managed computing systems [39, 40]. An autonomous model is based on MAPE-K loop which includes four phases of monitoring, analysis, planning and execution and a knowledge base. Data is collected in the environment by sensors and provided to the monitoring phase. In the monitoring phase it is aggregated and given to the next phase for analysis. If an undesirable situation is identified in the analysis phase, the planning phase begins. In this phase, appropriate decisions are made autonomously to solve the problem. Finally, the decisions are executed in the last phase. Also all of the necessary data are shared in knowledge base.

For effective management of cloud resources, an autonomous model is needed to timely identify and respond to adverse conditions. One of the undesirable conditions in the cloud resource management system is overloading of physical machines. An autonomous model for virtual machine consolidation can anticipate, analyze, and resolve this situation by selecting the appropriate plans. In the monitoring phase, data such as CPU utilization, energy, physical machine features, etc. can be collected and provided to the analysis phase. Execution phase operations can also be the migration of virtual machines or shutting down servers.

In the proposed model of this paper, the data collected by sensors are CPU utilization, energy consumption and number of VMs of servers. Also, changes made by effectors is to migrate virtual machines and shut down servers.

3.2. Theory of Learning Automata

The learning automata algorithm [41] performs an action on the environment and receives the feedback. It then updates its experiences based on the received feedback. Finally, it chooses the best action in the next time period. When learning automata selects a_i in step i and receives a proper response from the environment, $P_i(n)$ will increase, and other probabilities will decrease. But if it takes an inappropriate response from the environment, $P_i(n)$ will decrease and other probabilities will increase. After each change, the sum of all probabilities is constant and equal to one. An appropriate answer is created by Eq. (1) and Eq. (2) where *a* is the reward parameter.

$$P_i(n+1) = P_i(n) + a[1 - P_i(n)] \quad (1)$$

$$P_j(n+1) = (1-a)P_j(n) \; \forall \; \forall j \; j \neq i \quad (2)$$

An inappropriate answer is created by Eq. (3) and Eq. (4) where *b* is the penalty parameter and *r* is number of probability.

$$P_i(n+1) = (1-b)P_i(n) \quad (3)$$

$$P_j(n+1) = \frac{b}{r-1} + (1-b)P_j(n) \; \forall j \; j \neq i \quad (4)$$

3.3.Problem Statement and Assumptions

In this problem there is *N* server that is shown as $S=\{S_1, S_2, \ldots, S_n\}$. Each server has *K* number of virtual machines that are represented as $V = \{v_1, v_2, \ldots, v_k\}$. User requests are assigned to virtual machines by the service provider. Service level agreement is set between users and service providers, which determines the level of service quality required by users. By changing the user request, the workload of virtual and physical machines changes. Because the virtual machine consolidation process takes place at the IaaS layer, the investigated system is independent of user applications, and we do not consider the penalty in the current study. Instead, we calculate the degree of service level agreement violations which is the reflection of it.

When the amount of resources requested from a physical machine exceeds the amount of resources available, that physical machine will become overloaded. If a physical machine becomes overloaded, the virtual machine's SLA is violated. Using the virtual machine consolidation process, a number of VMs of overloaded server can be migrated to other servers. Each physical machine has features such as CPU utilization, power consumption and number of VMs. Predicting and timely identification of overloaded servers and selecting suitable physical machines for VM placement is the problem investigated in this research. In other words because of workload changes, a dynamic VM placement is needed to apply suitable selection scenarios in different time intervals. So an autonomous VM consolidation model with feedback capability can achieve these goal.

The system model is based on the MAPE-K loop to identify overloaded servers and choose appropriate servers for VM placement with the aim of reducing SLA violations and energy consumption. In the autonomous model of VM consolidation, the sensors first collect the workload amount from each server. Monitor phase monitors physical and virtual machines status and gives the results into analysis phase. In the analysis phase, the load rate of each server is analyzed and the future rate will be predicted by using a hybrid prediction algorithm which uses several prediction models. This prediction method combines the results of several prediction models together and returns the final prediction result. If the server load exceeds a threshold limit, that server is designated as an overloaded server and the planning phase will be triggered. At the planning phase, using several heuristics and learning automata algorithm, the proper servers are selected for VM placement and underloaded servers are determined. Finally, in the execution phase, the immigrant VMs are migrated to the target destination and underloaded servers go off. This reduces the total energy consumption, SLA

violations and migration counts. The system model is shown in Fig. 1 As the figure shows, the self-adaptive feedback loop is located at the PaaS layer which manages physical and virtual machines at IaaS layer.

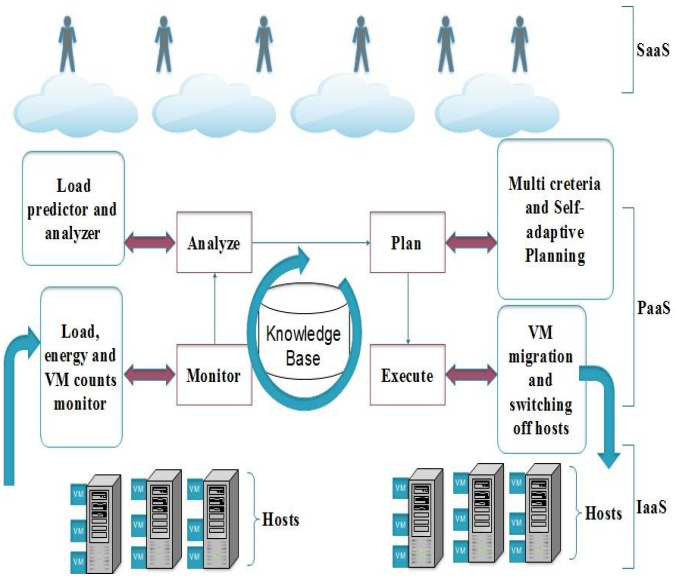

Fig. 1. The system model

In general, there are four situations for every server: normal, overloaded, underloaded and switched off. If the load of a server exceeds its allocated resources, this machine is in overloaded state. If the amount of resources of a server exceeds that required, this machine is in underloaded state. By migrating VMs from an overloaded server it goes to normal and from an underloaded goes switched off status. Fig. 2 shows transitions between server's states.

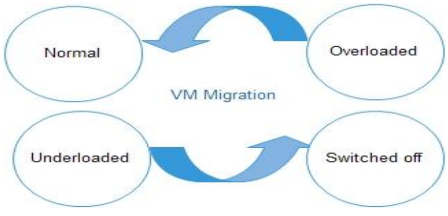

Fig. 2. Server's status

The metrics used for proposed model formulation are number of SLA violations, virtual machine migrations, energy consumption, ESV and ESM. A violation of service level agreement occurs when the requests volume for a server exceeds the amount of its available resources and it is overloaded. These metric also are used for proposed model evaluation. Evaluation criteria are presented in Eq. (5), (6), (7), (8) and (9) based on the paper [20, 25]. *SLAV* describes the performance decline that has occurred due to overloaded state of servers or migration counts.

$$SLAV = SLAVO \times SLAVM \tag{5}$$

SLAVO shows the percentage of times during which active servers had CPU utilization rates above 100%, or the number of requests they received exceeded their capacity.

$$SLAVO = \frac{1}{M} \sum_{i=1}^{M} \frac{T_{s_i}}{T_{a_i}} \tag{6}$$

M describes the server counts; T_{si} presents the total time the server i had CPU utilization rates above 100% or were overloaded. T_{ai} shows the total time that the physical machine i was active. *SLAVM* shows the total decrease in performance due to the migration of virtual machines.

$$SLAVM = \frac{1}{N} \sum_{j=1}^{N} \frac{C_{d_j}}{C_{r_j}} \tag{7}$$

N is the number of VMs; C_{d_j} shows an estimate of performance loss for the virtual machine j that occurred due to its migrations. C_{r_j} is the total amount of CPU requested by virtual machine j during its lifetime.

The combined *ESV* criterion is used to indicate a trade-off between energy consumption and service level agreement violation. Also *ESM* criterion in relation is used to indicate a trade-off between energy, SLA violations and migration counts.

$$ESV = Energy\ Consumption \times SLAV \quad (8)$$

$$ESM = ESV \times Number\ of\ VM\ Migrations \quad (9)$$

According to the virtual machine consolidation process, in this study, we have divided the ESM rate into four parts. ESM due to the detection of overloaded physical machines, ESM_{ohd}, ESM due to the immigrant virtual machine selection, ESM_{vs}, ESM due to the virtual machine placement, ESM_{vp} and ESM due to the detection of underloaded physical machines, ESM_{udh} .Our goal in this study is to minimize the ESM resulting from the overloaded servers detection and virtual machine placement. For ESM_{vs} and ESM_{uhd} we have considered minimum migration time (MMT) method and simple method (SM) [16] respectively. In future research, our goal is to provide solutions to minimize the ESM due to virtual machine selection and the ESM due to the detection of underloaded physical machines. Eq. (10) shows the goal function where ESM_t is the total amount of ESM.

$$ESM_t = ESM_{ohd} + ESM_{vs} + ESM_{vp} + ESM_{uhd} \quad (10)$$

In the analysis phase of the proposed autonomous model, the ESM due to the detection of the overloaded server and in the planning phase, the ESM due to the choosing appropriate physical machines for VM placement is minimized.

4.Proposed Model

In this section, the proposed model will be first described in general and then in details by the algorithms used in each steps.

The proposed model is based on the MAPE-K loop that has automated the process of overloaded servers identification and virtual machine placement in order to reduce power consumption and SLA violations using the self-adaptive feedback loop. The proposed model comprises a self-adaptive feedback control loop, which is located in PaaS layer, and manages physical and virtual machines that are in bottom layer of the platform as a service layer. This model consists of four main phases and a knowledge base, each includes operations that are described in the following.

Fig.3 shows the proposed model with the components of feedback loop phases and the relationship between them. As shown in the figure, in the monitor phase; the amount of load, number of VMs, and the amount of power consumption of each server is monitored at specified intervals and the monitored information is stored in the knowledge base. In the analysis phase, the server load prediction component predicts the server load in the next time interval using the past workload information of the servers stored in the knowledge base. It then investigates whether the server will become overloaded or not in the next time period. If the answer is yes, then the overloaded state number of this server will be updated and stored in the knowledge base and the planning phase will be initiated.

At the planning phase, each server takes a score by using information in the knowledge base, including the number of times the server has become overloaded until current time intervals, the number of virtual machines available on each server, the CPU utilization of server and the amount of energy consumed by the server. To score each server, heuristics have weights which are updated by learning automata algorithm. Ultimately the best server with minimum score will be selected for VM placement and underloaded servers will be identified. During the execution phase, some of VMs on the overloaded server are transferred to the one selected at the planning phase using live migration operations and underloaded servers go switched off mode.

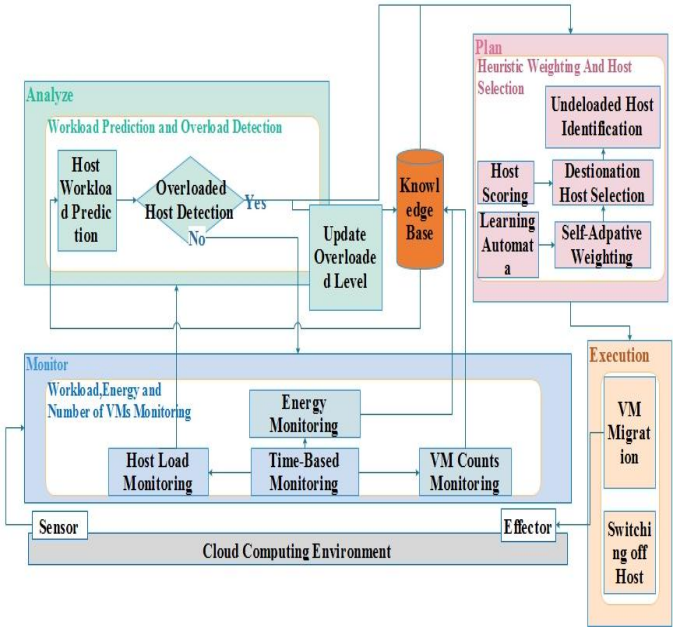

Fig. 3. Level one schema of proposed model

In the following, algorithms of each phase is presented. Table 5 also shows the abbreviations along with their definitions used in algorithms.

Table 5
Abbreviations and definitions used in algorithms

Abbreviations	Definitions
sl	Total number of servers
vl	Total number of VMs
Δt	Current time interval
mvl	immigrant virtual machine list
os	List of overload servers
S_i	Server i
v_j	Virtual Machine j
$Upper\_threshold$	Server load threshold
$Service\ Usage_{\Delta t+1}(S_i)$	The amount of load of server i in the next time interval
Map (S_i,v_j)	Mapping virtual machine j to server i
Map_List	List of all virtual machine mappings to servers
nh	Number of heuristic methods
ch	Chosen heuristic method
$P(h^k)$	Probability of choosing heuristic k
$W(h^k)$	Weight of heuristic k

Algorithm 1 shows whole of the proposed model. At first, in every time interval, all physical and virtual machines are monitored. Using the monitored data, the migrant virtual machines list is returned as output of the analysis phase. In fact in time interval Δt the migrant virtual machines list is extracted from the overloaded servers list. Then at planning phase, based on the state of servers, migrant virtual machines are placed on the servers which are not overloaded. In line 8, the mapping of virtual machines on servers is executed at execution phase. In fact, at the execution phase, control signals are sent to migrate virtual machines to appropriate servers. Also any decision made at the planning phase is executed at the execution phase like switching off underloaded hosts. In the following, proposed model is described into two parts:

1. Monitor and analysis phase
2. Planning and execution phase

Algorithm 1: *MAPE-based VM consolidation*

1: **Input:** Server_List *sl*
2: **Output:** Mapping list of VMs to servers.
3: **Begin**
4: **for each** time slot Δt **do**
5: *status* ← **Monitor** the whole servers and VMs status at time interval Δt
6: *mvl* ← **Analysis** (*status*, *sl*, Δt) /*Algorithm 2*/
7: *mappings* ← **Plan** (*mvl*, *sl*, *ch*, *status*, Δt) /*Algorithm 4*/
8: *status* ← **Execute** (*mappings*, *sl*, Δt) /*Algorithm 6*/
9: **end for**
10: *results* ← **getFinalResults**(*sl*, Δt)
11: **return** *results*
12:**end**

4.1. Monitor And Analysis Phase

In the monitor phase, the amount of the servers' load, power and number of VMs is monitored at different time intervals and sent to the analysis phase.

Our goal in the analysis phase is to minimize ESM_{ohd} by providing a suitable solution for timely detection of overloaded physical machines. ESM_{ohd} is shown in Eq. (11).

$$ESM_{odh} = Minimize\left(\sum_{i=1}^{N}\left|U^{p}_{\Delta t+1}(S_i) - U^{a}_{\Delta t+1}(S_i)\right|\right) \quad (11)$$

In relation 11 $U^{p}_{\Delta t+1}(S_i)$ shows the amount of predicted CPU utilization for S_i in the next time interval. Also $U^{a}_{\Delta t+1}(S_i)$ is the actual CPU utilization in the next time interval for S_i. The value of $U^{p}_{\Delta t+1}(S_i)$ is calculated using Eq. (12).

$$U^{p}_{\Delta t+1}(S_i) = (U^{m}_{\Delta t+1}(S_i) + U^{en}_{\Delta t+1}(S_i))/2 \quad (12)$$

In this regard, $U^{m}_{\Delta t+1}(S_i)$ shows the median CPU utilization of the physical machine S_i for previous time periods. Also $U^{en}_{\Delta t+1}(S_i)$ shows predicted CPU utilization of S_i for the next time period using the combined forecasting method. By Eq. (13) and Eq. (14), the value of is $U^{m}_{\Delta t+1}(S_i)$ calculated.

$$U^{m}_{\Delta t+1}(S_i) = \{(w+1)/2\}^{th}\ value\ of\ ts_i \quad (13)$$

$$ts_i = \{U^{a}_{\Delta t}(S_i), U^{a}_{\Delta t-1}(S_i), U^{a}_{\Delta t-2}(S_i), \dots, U^{a}_{\Delta t-w+1}(S_i)\} \quad (14)$$

In Eq. (14), ts_i shows the set of CPU utilization samples for S_i in previous time periods and w is number of samples or window size. Eq. (15) shows $U^{en}_{\Delta t+1}(S_i)$ calculation.

$$U^{en}_{\Delta t+1}(S_i) = (W^{i}_{MA} \times pm^{MA}_{ts_i[w]}) + (W^{i}_{WMA} \times pm^{WMA}_{ts_i[w]}) + (W^{i}_{PR} \times pm^{PR}_{ts_i[w]}) \quad (15)$$

A hybrid predictor is used to predict the load of S_i in the next time period. This predictor uses moving average (MA), weighted moving average (WMA) and polynomial regression (PR) prediction models to predict the physical machines load in the next time period. Eq. (16) shows the set of predictors used. In the above relations, $pm^{MA}_{ts_i[w]}$, $pm^{WMA}_{ts_i[w]}$, $pm^{PR}_{ts_i[w]}$ show moving average (MA), weighted moving average (WMA) and polynomial regression (PR) prediction models respectively for w CPU utilization samples of server i and W^{MA}_{i}, W^{WMA}_{i} and W^{PR}_{i} show the weight of the MA, WMA and PR forecasting models, respectively.

$$PM = \{MA, WMA, PR\} \quad (16)$$

Since the forecasting models have different accuracy in forecasting, the Sum of Absolute Errors (SAE) criterion has been used to evaluate these predicting models. So the predictive model that has less SAE has more weight. Eq. (17) shows the SAE calculation equation and Eq. (18) shows the SAE value of the prediction model l for CPU utilization history of server i by assuming that window size is w.

$$SAE = \sum_{j=1}^{w}\left|U^{p}_{ts[j]} - U^{a}_{ts[j]}\right| \quad (17)$$

$$SAE^{l}_{i} = \sum_{j=1}^{w}\left|pm^{l}_{ts_i[j]} - U^{a}_{ts_i[j]}\right| \quad (18)$$

In the Eq. (17), $U^{p}_{ts[j]}$ shows the jth prediction value of CPU utilization and in relation 18 SAE^{l}_{i} shows SAE of prediction model l for server i. In Eq. (19), Eq. (20) and Eq. (21), the weight of each prediction model was calculated using their SAE.

$$W^{MA}_{i} = \frac{Max\{SAE^{l}_{i}\}_{l\in PM} - SAE^{MA}_{i}}{\sum_{l=1}^{nm}\left(Max\{SAE^{l}_{i}\}_{l\in pm} - SAE^{l}_{i}\right)} \quad (19)$$

$$W^{WMA}_{i} = \frac{Max\{SAE^{l}_{i}\}_{l\in PM} - SAE^{WMA}_{i}}{\sum_{l=1}^{nm}\left(Max\{SAE^{l}_{i}\}_{l\in pm} - SAE^{l}_{i}\right)} \quad (20)$$

$$W^{PR}_{i} = \frac{Max\{SAE^{l}_{i}\}_{l\in PM} - SAE^{PR}_{i}}{\sum_{l=1}^{nm}\left(Max\{SAE^{l}_{i}\}_{l\in pm} - SAE^{l}_{i}\right)} \quad (21)$$

In the above relations SAE^{MA}_{i}, SAE^{WMA}_{i} and SAE^{PR}_{i} show the SAE values of the MA, WMA and PR prediction models respectively for a single server S_i. Also as shown in Eq. (22), the sum of W^{MA}_{i}, W^{WMA}_{i} and W^{PR}_{i} is equal to one.

$$W^{MA}_{i} + W^{WMA}_{i} + W^{PR}_{i} = 1 \quad (22)$$

Analysis phase with the past data about the amount of servers load in the previous time intervals, stored in the knowledge base, and by using the prediction method, predict the future servers load. Monitor and analysis process is shown in Algorithm 2 and Algorithm 3 shows prediction algorithm.

Algorithm 2 shows the analysis phase that part of it is associated with the prediction algorithm. In Algorithm 2, at the beginning, S_i from *sl* is investigated. If the prediction algorithm identifies it overloaded, then it is added to the overloaded servers list. Indeed, *os* shows the list of all overloaded servers in the next time interval. In the line 9 and 10, for each server in the *os* list, the migrant virtual machines are selected to prevent them from being overloaded. Because based on the prediction models, it has been speculated that these servers will be overloaded in the next time interval. Then the selected virtual machines are added to the list of migrant virtual machines. Minimum migration time (MMT) method [16] is used for VM selection. Finally, this algorithm returns the list of migrant virtual machines as output.

Algorithm 2: *Host and VM analysis (**Analysis phase**)*

1: **Input:** Server List *sl*
2: **Output:** *mvl* /*Migration VM List*/
3: **Begin**
/* Host over load detection*/
4: **for each** server S_i **in** *sl* **do**
5: **if**(server S_i is overloaded ==**true**) **then**
/*According to Algorithm 3*/
6: *os*.Add(S_i)
7: **end if**
8: **end for**
/* Select immigrant VMs from overloaded hosts*/
9: **for each** server S_i **in** *os* **do**
10: *mvl* ← *mvl* +Select migrating VMs from overloaded host S_i /*VM selection by **MMT** algorithm [16] */
11: **end for each**
12: **return** *mvl*
13: **end**

In Algorithm 3, we investigate all of the servers to determine whether they will become overloaded in future or not. This section uses a time series to predict the future. There is also a defined window size that takes into account recent times (the assumption for window size is 10 and each interval is 5 minutes). In line 11 and 12, the weight of each prediction model is calculated, the smaller the SAE of the prediction model, the greater the weight is. In line 13, the value of each prediction model for the next time period is multiplied by its weight and the sum of the results is calculated. Finally, the prediction result of the hybrid forecasting model are averaged with the median of load data in *w* past time intervals. If the result shows that the CPU utilization of S_i exceeds a threshold, the state of that server is identified overloaded.

Algorithm 3: *Host overload detection with ensemble prediction algorithm*

1: **Input:** Server S_i
2: **Output:** *Is_Overloaded*
3: **Begin**
4: *ts*=**new** timeseries()
5: **for** *w* = 0 to Window_Size **do**
6: ts.add(Resource_Usage_History($S_i, \Delta t - w$)
7: **end for**
8: **for each** constituent prediction model pm^l **do**
9: $SAE_i^l = \sum_{j=1}^{w} \left| pm_{ts_i[j]}^l - U_{ts_i[j]}^a \right|$

10: **end for each**
11: **for each** constituent prediction model pm^l **do**
12: $W_i^l \leftarrow \frac{Max\{SAE_i^l\} - SAE_i^l}{\sum_{l=1}^{nm}\left(Max\{SAE_i^l\} - SAE_i^l\right)}$ (over $l \in PM$ and $l \in pm$)

13: $Service\ Usage_{\Delta t+1}(S_i) = Avg(\sum_{l=1}^{nm}\left(W_i^l \times pm_{ts_i[w]}^l\right), U_{\Delta t+1}^m(S_i))$

14: end for each
$eshold)$ **then**
16: Increase $Overload\ Num(S_i, \Delta t)$
17:**end if**
$eshold$
19:**end**

4.2. Planning And Execution Phase

If at the analysis phase, the server is detected overloaded for the next time interval, then the planning phase is triggered. This phase finds appropriate servers for virtual machine placement. The decision to choose the proper server is made at the planning phase. This phase uses some heuristics by dynamic weights to select the appropriate servers. Finally, with different servers, diverse decisions are made to choose the suitable destination server. Selecting the appropriate server process is shown in Algorithms 4 and 5. Algorithm 4 is a multi-heuristic virtual machine allocation method. It is a virtual machine placement algorithm that uses several heuristic methods to find the appropriate server as a destination for migrant virtual machines.

In Algorithm 4, the weight of heuristic methods is updated by Algorithm 5. Heuristics are algorithms that try to solve a problem in a semi-optimal way with a very simple and specific idea of their own, and their main advantage is their high speed because they don't examine a too large state space. In this algorithm it is assumed that there is *nh* heuristics and here *nh* is considered 4. It is also assumed that each heuristic has a weight between zero and one (based on Algorithm 5). At the beginning of the algorithm, the weight of all these heuristics is equal. So, the first step is to update the weight of the heuristics using the algorithm 5. In this phase all of the virtual machines in *mvl* assign to appropriate servers. *mvl* is the list of migrant virtual machines that have been selected in the previous phase using the minimum migration time method. Four heuristic methods are used to find a proper server, which are outlined in lines 8 to 12. In fact, here are some ideas that are applicable to assign virtual machines to servers to minimize energy consumption.

The first heuristic uses classification method as it divides servers into several classes based on their number of VMs. It gives a score to every server based on what class it is located and finally decides whether this server is a suitable server for virtual machine placement or not. If a server is in the first or last class, compared to other servers, it means that it has a lot of or a few virtual machines, but if it is located in the middle classes, this server is a good destination for migrant virtual machines. Because a server with a large number of virtual machines is likely to suffer from overloading, on the other hand, a server with a very small number of virtual machines can be considered as an underloaded server and be shut down which saves energy. Therefore, a server that has an average number of virtual machines is suitable for VM placement. In fact $h_{S_i}^{VML}(\Delta t)$ shows the VM level and $h_{S_i}^{VMC}(\Delta t)$ the VM class of server *i* in current time interval. A server in first or last class, gets a high score, which means it is not a good server to choose, and in fact a server is better suited for virtual machine placement, that has zero score. Eq. (23) and Eq. (24) show the first heuristic. In this relation $S_i^{VMN}(\Delta t)$ shows the number of VMs of server *i* in Δt time and ∂ shows the number of classes which is assumed 10 here.

$$\partial = 10$$

$$h_{S_i}^{VML}(\Delta t) = \partial \times \frac{S_i^{VMN}(\Delta t) - Min\{S_i^{VMN}(\Delta t)\}}{\underset{S_i \epsilon\, sl}{Max\{S_i^{VMN}(\Delta t)\} - Min\{S_i^{VMN}(\Delta t)\}}} \quad (23)$$

$$h_{S_i}^{VMC}(\Delta t) = \begin{cases} \frac{\partial - h_{S_i}^{VML}(\Delta t)}{\partial}, & h_{S_i}^{VML}(\Delta t) \le \frac{\partial}{5} \\ \frac{h_{S_i}^{VML}(\Delta t)}{\partial}, & h_{S_i}^{VML}(\Delta t) \ge \partial - \frac{\partial}{5} \\ 0 & otherwise \end{cases} \quad (24)$$

The second heuristic method has been used in [29]. This heuristic chooses a physical machine that consumes less energy than other physical machines after virtual machine placement. The lower the energy consumption, the more suitable this physical machine is. Eq. (25) shows the second heuristic. In this relation $S_i^P(v, \Delta t)$ shows the power of server *i* after allocation of VM *v* in Δt time.

$$h_{S_i}^P(\Delta t) = \frac{S_i^P(v, \Delta t)}{\underset{S_i \epsilon sl}{Max\{S_i^P(v, \Delta t)\}}} \quad (25)$$

The third heuristic investigates physical machine's CPU utilization in the current and previous time interval. If a physical machine has a low average CPU utilization, it is a better choice for placing immigrant virtual machines. Eq. (26) shows the third heuristic. In this relation $S_i^{CPU}(\Delta t)$ is the CPU utilization of server i in Δt time.

$$h_{S_i}^{CPU}(\Delta t) = \frac{S_i^{CPU}(\Delta t)+S_i^{CPU}(\Delta t-1)}{\underset{S_i \in sl}{Max}\{S_i^{CPU}(\Delta t)+S_i^{CPU}(\Delta t-1)\}} \quad (26)$$

Last heuristic method investigates the number of times that the server has become overloaded until Δt time, by using information stored in knowledge base. The overloaded state for a server indicates that the requests volume for this server exceeded the resources that it had, and ultimately the server failed to deliver the application QOS of its virtual machines which caused SLA violations. So the higher the number of times a server goes overloaded, the greater the likelihood of a violation of service level agreement is, and thus the server is not proper for deploying migrant virtual machines. Eq. (27) shows the last heuristic. In this relation $S_i^{ON}(v,\Delta t)$ shows the number of times that server i has experienced overloaded state until Δt time. The heuristics set is brought in Eq. (28).

$$h_{S_i}^{ON}(\Delta t) = \frac{S_i^{ON}(\Delta t)}{\underset{S_i \in sl}{Max}\{S_i^{ON}(\Delta t)\}} \quad (27)$$

$$H = \{VMC, P, CPU, ON\} \quad (28)$$

In line 13 $costscore_S$ shows the sum of multiplication of each heuristic method weight to its value. This parameter determines how well server i is suitable for VM v. The server with the lowest rank among all the servers is selected as a destination for migrant virtual machine. Ultimately each migrant virtual machine owns a server and is added to map-list. This map-list is returned as the output of the fourth algorithm.

After identifying the appropriate destination servers, it is time to determine the underloaded servers. To do this, all servers are sorted by their CPU utilization. Then servers that are not overloaded or destination, will be identified as underloaded servers based on their CPU utilization. The underloaded servers identification is based on Simple Method (SM) [16]. Finally the goal of this phase is to reduce ESM_{vp}. The ESM_{vp} relationship is given in Eq. (29).

$$ESM_{vp} = Minimize\left(\sum_{i=1}^{N}\left(W\left(h_{S_i}^{VMC}(\Delta t)\right)\times h_{S_i}^{VMC}(\Delta t) + W\left(h_{S_i}^{P}(\Delta t)\right)\times \frac{S_i^{P}(v,\Delta t)}{\underset{S_i \in sl}{Max}\{S_i^{P}(v,\Delta t)\}} + W\left(h_{S_i}^{CPU}(\Delta t)\right)\times \frac{S_i^{CPU}(\Delta t)+S_i^{CPU}(\Delta t-1)}{\underset{S_i \in sl}{Max}\{S_i^{CPU}(\Delta t)+S_i^{CPU}(\Delta t-1)\}} + W\left(h_{S_i}^{ON}(\Delta t)\right)\times \frac{S_i^{ON}(\Delta t)}{\underset{S_i \in sl}{Max}\{S_i^{ON}(\Delta t)\}}\right)\right) \quad (29)$$

Eq. (30) shows that the total weight of the heuristics is equal to one.

$$W\left(h_{S_i}^{VMC}(\Delta t)\right) + W\left(h_{S_i}^{P}(\Delta t)\right) + W\left(h_{S_i}^{CPU}(\Delta t)\right) + W\left(h_{S_i}^{ON}(\Delta t)\right) = 1 \quad (30)$$

The weights used in Eq. (28) are generated autonomously using the feedback capability of the self-adaptive model and the learning automata algorithm. To calculate the weights autonomously, first the probabilities of selecting the heuristics and then their weights are updating. Eq. (31) shows the heuristic probability updates.

$$P(h^k) = \begin{cases} \begin{cases} P(h^k) + a \times \left(1 - P(h^k)\right), & k = ch \\ (1-a) \times P(h^k), & k <> ch \end{cases} & Pleasing \\ \begin{cases} (1-b) \times P(h^k), & k = ch \\ \frac{b}{nh-1} + (1-b) \times P(h^k), & k <> ch \end{cases} & Unpleasing \end{cases} \quad (31)$$

In Eq. (31), $P(h^k)$shows the probability of choosing heuristic *k*, where *nh* is the number of heuristics and α and β are the reward and penalty parameters, respectively, which are obtained through parameter tuning. After updating the probabilities using the roulette wheel algorithm and generating a random number *x*, one of the heuristics is selected. The roulette wheel algorithm is given in Eq. (32).

$$ch = \begin{cases} VMC\ ,\ x > P\left(h_{S_i}^{VMC}(\Delta t)\right) \\ \text{P}\ ,\ x > P\left(h_{S_i}^{VMC}(\Delta t)\right) + P\left(h_{S_i}^{P}(\Delta t)\right) \\ \text{CPU},\ x > P\left(h_{S_i}^{VMC}(\Delta t)\right) + P\left(h_{S}^{P}(\Delta t)\right) + P\left(h_{S_i}^{CPU}(\Delta t)\right) \\ \text{ON}\ ,\ x > P\left(h_{S_i}^{VMC}(\Delta t)\right) + P\left(h_{S}^{P}(\Delta t)\right) + P\left(h_{S_i}^{CPU}(\Delta t)\right) + P\left(h_{S_i}^{ON}(\Delta t)\right) \end{cases} \quad (32)$$

In the above relation $P\left(h_{S_i}^{VMC}(\Delta t)\right)$ for example shows the probability of chosen heuristic VMC for server *i* in time Δt. Finally, the heuristic weights is updated using Eq. (33).

$$W(h^k) = \begin{cases} W(h^k) + a \times \left(1 - W(h^k)\right), & k = ch \\ (1-a) \times W(h^k), & k <> ch \end{cases} \quad (33)$$

In Eq. (33) $W(h^k)$is the weight of heuristic *k* and *α* is learning parameter of the algorithm.

Algorithm 4: *Proposed multi-heuristics VM allocation algorithm (**Plan phase**)*

1: **Input:** *ch* /*chosenHeuristic*/ , *sl* /*Server List*/, *mvl* /*Migrating VM list*/
2: **Output:** *Map_List*
3: **Begin**
4:Update heuristics weights by **Algorithm 5**
*/** Allocate migrating VMs from overutilized hosts**/*
5: **for each** VM v_j **in** *mvl* **do**
6: **for each** Server S_i **in** *sl* **do**
7: $\partial = 10$
8:
$$h_{S_i}^{VML}(\Delta t) = \partial \times \frac{S_i^{VMN}(\Delta t) - Min\{S_i^{VMN}(\Delta t)\}}{\underset{S_i \in sl}{Max\{S_i^{VMN}(\Delta t)\} - Min\{S_i^{VMN}(\Delta t)\}}}$$
9:
$$h_{S_i}^{VMC}(\Delta t) = \begin{cases} \frac{\partial - h_{S_i}^{VML}(\Delta t)}{\partial}, & h_{S_i}^{VML}(\Delta t) \le \frac{\partial}{5} \\ \frac{h_{S_i}^{VML}(\Delta t)}{\partial}, & h_{S_i}^{VML}(\Delta t) \ge \partial - \frac{\partial}{5} \\ 0 & otherwise \end{cases}$$ /* VM level*/
10: $h_{S_i}^{P}(\Delta t) = \frac{S_i^P(v,\Delta t)}{\underset{S_i \in sl}{Max\{S_i^P(v,\Delta t)\}}}$ /*Power level*/
11: $h_{S_i}^{CPU}(\Delta t) = \frac{S_i^{CPU}(\Delta t) + S_i^{CPU}(\Delta t-1)}{\underset{S_i \in sl}{Max\{S_i^{CPU}(\Delta t) + S_i^{CPU}(\Delta t-1)\}}}$
/*Load level
12: $h_{S_i}^{ON}(\Delta t) = \frac{S_i^{ON}(\Delta t)}{\underset{S_i \in sl}{Max\{S_i^{ON}(\Delta t)\}}}$
/*SLAV risk level*/
13: $S_i^{Cost\ Score}(\Delta t) = \Big(W\left(h_{S_i}^{VMC}(\Delta t)\right) \times h_{S_i}^{VMC}(\Delta t) + W\left(h_{S_i}^{P}(\Delta t)\right) \times \frac{S_i^P(v,\Delta t)}{\underset{S_i \in sl}{Max\{S_i^P(v,\Delta t)\}}} + W\left(h_{S_i}^{CPU}(\Delta t)\right) \times \frac{S_i^{CPU}(\Delta t) + S_i^{CPU}(\Delta t-1)}{\underset{S_i \in sl}{Max\{S_i^{CPU}(\Delta t) + S_i^{CPU}(\Delta t-1)\}}} + W\left(h_{S_i}^{ON}(\Delta t)\right) \times \frac{S_i^{ON}(\Delta t)}{\underset{S_i \in sl}{Max\{S_i^{ON}(\Delta t)\}}} \Big)$
14: **end for each**
15: $S_i = Server\ with\ minimum\ Cost\ Score$
16: *Map_List* ←*Map_List* + *Map* (S_i,v_j)
17: **end for each**
*/**Turn off as much underutilized host as possible and add migrate their VMs**/*
18: **for each** host S_i **in** *sl* ***order by*** *CPU utilization* **do**
19: **if** S_i was not overloaded **and** S_i **is not in** *Map_List* **then**
20: **update** *Map_List* by migrating VMs **in** S_i
21: **sign** S_i **as** underloaded host
22: **end if**
23: **end for each**
24: **return** *Map_List*
25: **end**

Fifth algorithm shows how the learning automata is applied to weight each heuristic method used in the fourth algorithm. The learning automata implements a series of operations on the environment and receiving feedback from it. Then rewarding and penalizing its operations based on the feedback it receives, also updating the likelihood of any operations. Based on the probabilities, it selects new operations and applies them to the environment. In the fifth algorithm, every heuristic method is an action and takes a weight, based on its probability. Different weights can be assigned to different heuristics based on what happens to each server. It makes the algorithm more flexible because it is not completely dependent on one heuristic and can autonomously behave.

At the beginning of the algorithm 5, if the time interval is initial and no heuristic has been selected yet, it means that there is no past and no experience, so the probability of selecting every heuristic, while there are four heuristic methods, is 0.25. Also the initial weights of all heuristic methods are equal. Otherwise at first, the feedback from the learning automata is assumed to be pleasing. In this algorithm, every physical machine that has been selected for VM placement in the last time interval, is examined and *dhList* shows the list of these physical machines. If a server has been selected for VM placement in the last time interval, and then in the current time interval it has been identified overloaded, then the feedback is unpleasing. Because it indicates that the algorithm's decision to choose this server was not right that caused the server to experience overloaded state.

If the feedback is desirable, the heuristic method chosen in the previous step takes reward. Therefore, the algorithm based on the standard learning automata formula adds α percentage of the probability of other heuristic methods to the probability of chosen heuristic method. Also reduces α percentage of other heuristic's probabilities. But if the feedback is not desirable, it means that the chosen heuristic was not appropriate, so β percentage of the chosen heuristic probability is subtracted from it and β percentage of probability is added to the other's probabilities.

After updating the probability of heuristic methods, based on the Roulette-Wheel-Selection algorithm, every heuristic method takes an area of the circle proportional to its probability. So a heuristic method is selected based on its probability. After selection, α percent of its weight is added to its weight and for other heuristic methods α percent of their weight is subtracted from their weight. Finally at first probability and then the weights of the heuristic methods are updated based on the feedback received from the environment.

Algorithm 5: *LA-based Weight assignment for heuristics*

1: **Input:** *ch* /*Chosen Heuristic*/, *nh* /* number of Heuristics */ , α /*reward rate*/, β /*penalty rate/ *dhList* /*Destination hosts of migrating VMs in the last time interval
2: **Output:** Null
3: **Begin**
4:**if** Δt=1 **and** *ch*==null **then** /* If there is no past experience*/
5: $P(h^k) = {}^1/_{nh} \quad \forall\, 1 \le k \le nh$
6: $W(h^k) = {}^1/_{nh} \quad \forall\, 1 \le k \le nh$
7:**else**
8: feedback= "pleasing"
9: **for each** host *h* in *dhList* **do**
10: **if** S_i(Δt-1) **is** *overloaded* **then** /*If host load exceeds host capacity*/
11: feedback= "unpleasing"
12: **end if**
13: **if** feedback is "pleasing" **then** /*If host load doesn't exceed host capacity*/
14: $P(h^k) = P(h^k) + a \times \left(1 - P(h^k)\right)$
15: **for each** $P(h^k)$ in *heuristics* **do**
16: **if** *k* <> *ch* **then**
17: $P(h^k) = (1 - a) \times P(h^k)$
18: **end if**
19: **end for each**
20: **else**
21: $P(h^k) = (1 - b) \times P(h^k)$
22: **for each** $P(h^k)$ in *heuristics* **do**
23: **if** *k* <> *ch* **then**
24: $P(h^k) = \frac{b}{nh-1} + (1 - b) \times P(h^k)$
25: **end if**

```
26:        end for each
27:     end if
28:  end for each
29:     Choose S_i with Roulette-Wheel-Selection based on its probability
30:     W(h^k)= W(h^k) + a × (1 − W(h^k))
31:     for each w(k) in heuristics do
32:        if k <> i then
33:              W(h^k)= (1 − a) × W(h^k)
34:        end if
35:     end for each
36:end if
37:end
```

In the execution phase, migration operation will be done for all of the virtual machines that have been mapped to the appropriate server during the planning phase. Also, the servers that were previously found underloaded, go switched off. The execution phase is shown in Algorithm 6.

Algorithm 6: *Execution phase*

```
1: Input:  Map_List /*Allocation map*/ , sl /*Server list*/
2: Output: server status, VM status
3: Begin
4:   for each v_j in Map_List do
5:     Live migrate v_j to S_i in Map_List
6:   end for each
7:    for each server S_i in sl order by CPU utilization do
8:      if (sign(S_i) = underloaded) then
9:        Switch  S_i to off
10:    end if
11: end for each
12:end
```

In our proposed model, we use self-adaptive MAPE K loop to identify overloaded physical machines and place their virtual machines in appropriate hosts. In this model, in the analysis phase, the overloaded servers are identified using ensemble prediction algorithm and in the planning phase, the appropriate hosts are determined based on various heuristics and flexible weights. Using the ensemble prediction algorithm, the prediction accuracy for overloaded server identification will be improved. Also, considering different criteria with self-adaptive weights to select the proper hosts will reduce the SLA violations, energy consumption and number of VM migrations, compared to previous methods. In the following, the proposed solution is evaluated.

5. Evaluation

In this section cloudsim [42] is employed to simulate the proposed approach and ten days of real workload from CoMon [43] project is used for simulation. In this workload, the CPU utilization of more than thousands of virtual machines has been collected every 5 minutes. These virtual machines are located on more than 500 servers scattered around the world. Each algorithm used in this paper has been run ten times by the simulator and the average of the results has been used for evaluation. The evaluations of this paper are divided into 4 categories:

1. The analysis phase evaluation
2. The plan phase evaluation
3. Overall evaluation of the proposed autonomous model
4. Comparison of the proposed autonomous model with other scenarios

Table 6 shows the configuration of the servers used in this simulation. This table is based on the standard papers. The standard is used in AmazonEC2.

Table 6
Configuration of the servers used in the simulation

Server	CPU Model	Cores	Frequency (MHz)	RAM (GB)
HP ProLiant G4	Intel Xeon 3040	2	1860	4
HP ProLiant G5	Intel Xeon 3075	2	2660	4

Table 7 shows the amount of energy consumed by all servers defined in Table 6. As can be seen, the power consumption of servers increases with increasing processor utilization. Also, different servers with different CPU frequencies show different power consumption at the same CPU utilization efficiency.

Table 7
Power consumption of servers in watts

HP ProLiant G5	*HP ProLiant G4*	*Server*
93.7	86	Idle
97	89.4	10%
101	92.6	20%
105	96	30%
110	99.5	40%
116	102	50%
121	106	60%
125	108	70%
129	112	80%
133	114	90%
135	117	Full

Tables 8 and Table 9 show the types of virtual machines and the workload for different working days, respectively. Workloads are for 10 different days. For each working day the dataset shows, the number of VMs, the average and standard deviation of the load variations.

Table 8
Types of Virtual Machines Used in Evaluation

VM Type	*High-CPU Medium Instance*	*Extra Large Instance*	*Small Instance*	*Micro Instance*
CPU (MIPS)	2500	2000	1000	500
RAM (MB)	870	1740	1740	613

Table 9
Workload information

Date	Number of VMs	Means (%)	SD (%)
03/03/2011	1052	12.31	17.09
06/03/2011	898	11.44	12.83
09/03/2011	1061	10.70	15.57
22/03/2011	1516	9.26	12.78
25/03/2011	1078	10.56	14.14
03/04/2011	1463	12.39	16.55
09/04/2011	1358	11.12	15.09
11/04/2011	1233	11.56	15.07
12/04/2011	1054	11.54	15.15
20/04/2011	1033	10.43	15.21

5.1. The Analysis Phase Evaluation

In this section, we compare the efficiency of the proposed prediction method with two other forecasting methods. One of these methods is the simple prediction method LR [16], and the other is a combined prediction method of Polynomial Regression (PR) and Double Exponential Smoothing (DES). This predictive method considers the maximum value of two prediction models to have SLA assurance. All three predictive models are examined in the baseline scenario MMT / PABFD / SM of [16]. The simulation results in Fig. 4, Fig. 5 and Fig. 6 show that the proposed forecasting method reduced the SLA violations, consumed energy and virtual machine migration counts respectively compared to the other two methods.

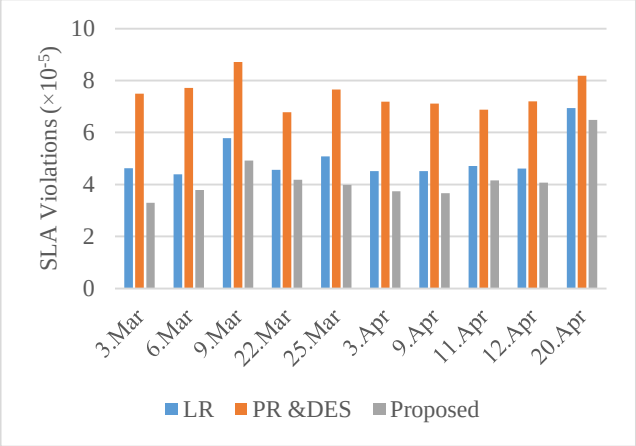

Fig. 4. SLAV comparison of different prediction methods

As Fig. 4 shows, the number of SLA violations of the proposed method has decreased by 14.96% due to the correct prediction of overloaded physical machines compared to the LR method and by 43.54% compared to the PR & DES.

Fig. 5 and Fig. 6 show consumed energy and VM migration counts of proposed method and other methods respectively. With the timely identification of overloaded physical machines, the consumed energy of physical machines in the proposed method has decreased by 12.16% compared to the LR method and 45.4% compared to the PR & DES method. Also, since the proposed forecasting method has largely prevented the physical machines to become overloaded, the number of migrations of its virtual machines has decreased by 20.18% compared to the

LR method and 44.33% compared to the PR & DES method.

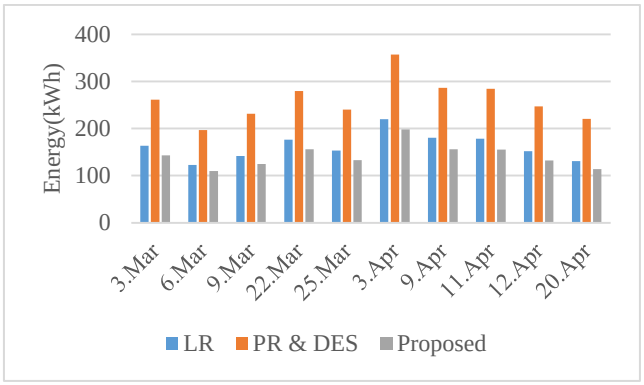

Fig. 5. Consumed energy comparison of different prediction methods

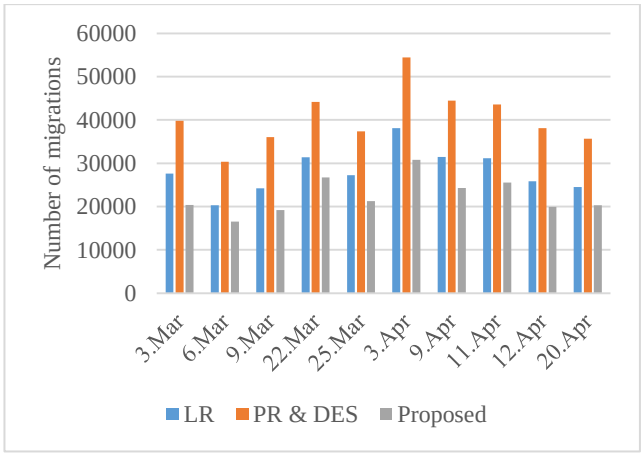

Fig. 6. Migration counts comparison of different prediction methods

Fig. 7 compares energy and SLAV trade-off of the proposed prediction method with the other two methods, which shows that the ESV of the proposed method is 25.59% lower than the LR method and 69.44% lower than the PR & DES combination method. Fig. 8 also compares the trade-off of energy, SLAV, and number of VM migrations of the proposed prediction method with the other two methods. In this comparison, the proposed method ESM has decreased by 40.43% compared to the LR method and 82.95% compared to the PR & DES.

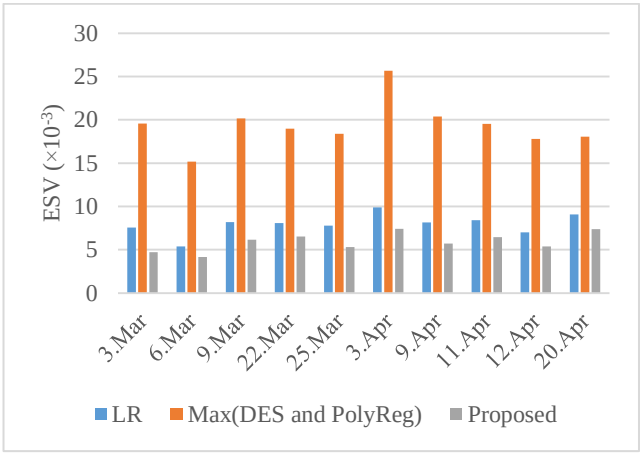

Fig. 7. ESV comparison of different prediction methods

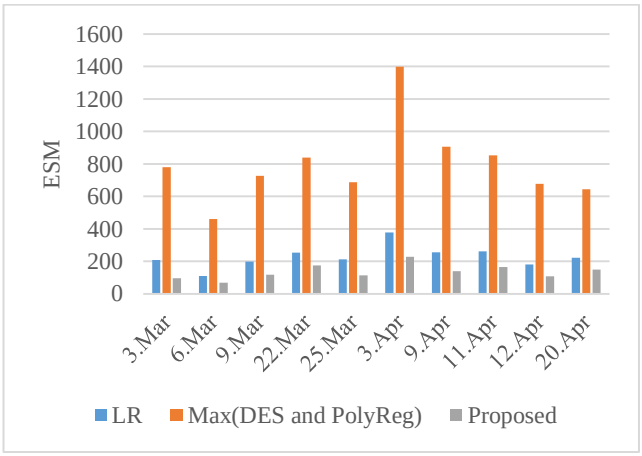

Fig. 8. ESM comparison of different prediction methods

Table 10 shows the average results of comparing the proposed prediction method with other methods in 10 working days in terms of SLA violations, energy, migration counts, and ESV and ESM metrics.

Table 10
Average results on the prediction proposed method and other methods for all workload days

Method/ Metrics	**Energy**	**SLA Violation ×10^{-5}**	**Migration Counts**	**ESV ×10^{-3}**	**ESM**
LR	161.87	4.974	28174.7	7.953997	228.0389
Max(DES and PR)	260.415	7.492	40401.8	19.36564	796.6999
Proposed	**142.184**	**4.23**	**22490.1**	**5.**	**135.8405**

5.2. The Plan Phase Evaluation

In this section, the efficiency of the proposed virtual machine placement method is compared with PABFD [16] and TOPSIS [31] methods. PABFD uses power metric for VM placement and TOPSIS is multi heuristics method for VM placemen without flexible weights. Fig. 9, Fig 10, and Fig. 11 show the results of the proposed method comparison with the other two methods in terms of SLA violations, energy and number of migrations respectively.

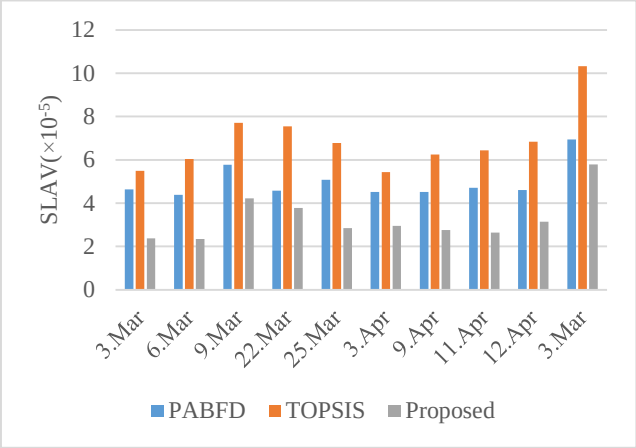

Fig. 9. SLAV comparison of different VM placement methods

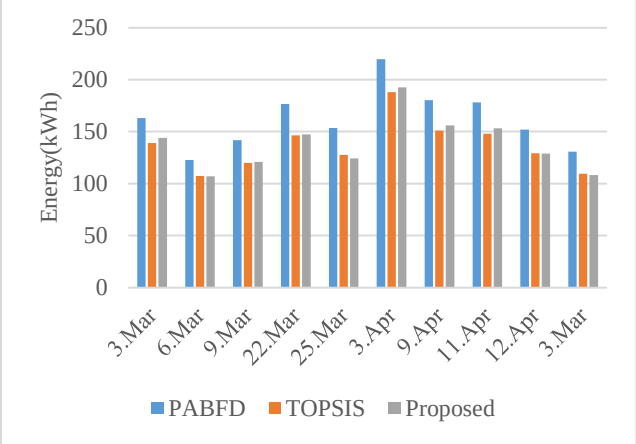

Fig. 10. Energy comparison of different VM placement methods

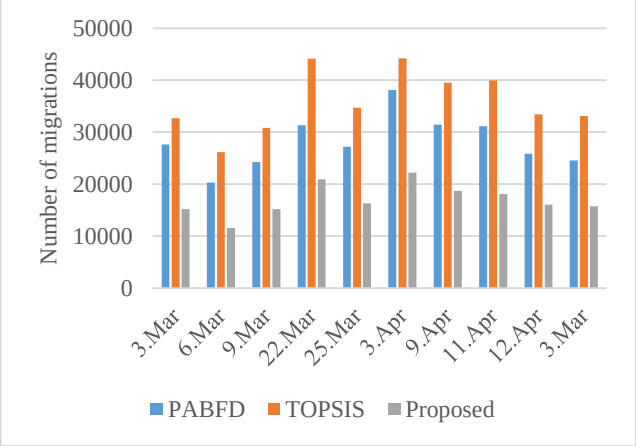

Fig. 11. Migration counts comparison of different VM placement methods

The simulation results show that the proposed virtual machine placement method using flexible weights was able to reduce SLA violations, energy consumption and the number of migrations compared to PABFD by 33.96%, 14.57% and 39.67% respectively. Compared to the second method, the proposed method has significantly reduced the number of violations of the service level agreement by 52.28% and number of migrations by 52.6%, but the energy consumption rate compared to this method has increased by 1.17%.

Fig. 12 shows the amount of compromise between energy and SLA violations in all three methods. Also, the amount of compromise between the three criteria of energy, SLA violations and number of migrations is shown in Fig. 13.

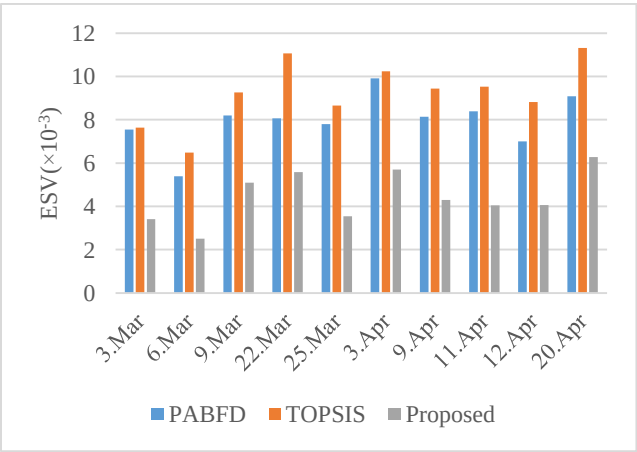

Fig. 12. ESV comparison of different VM placement methods

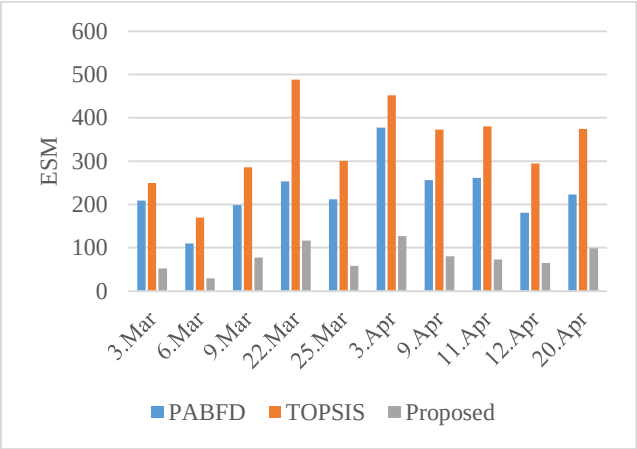

Fig. 13. ESM comparison of different VM placement methods

The results of the comparison show that the proposed virtual machine placement method was able to establish a better trade-off between the criteria of SLA violations and energy, as well as SLA violations, energy and number of migrations. So that the ESV of the proposed method has decreased by 44.05% compared to the PABFD and 51.85% compared to the TOPSIS. Also, the ESM of the proposed method was 65.94% lower than the PABFD and 76.93% lower than the TOPSIS method.

Table 11 shows the average results of comparing the proposed VM placement method with other methods in 10 working days in terms of SLA violations, energy, migration counts, and ESV and ESM metrics.

Table 11
Average results on the proposed VM placement method and other methods for all workload days

Method/Metrics	SLA Violation ×10-5	Energy	Migration Counts	ESV ×10-3	ESM
PABFD	4.974	161.87	28174.7	7.95	228.04
TOPSIS	6.88	136.69	35859.51	9.24	336.75
Proposed	**3.28**	**138.29**	**16998.43**	**4.45**	**77.67**

5.3. Overall Evaluation Of The Autonomous Model

In this section, the values of parameters *a* and *b* are set for the learning automata algorithm used in Algorithm 5 and a three-dimensional diagram is used to illustrate it. In this chart the ESM value is inverted to find the maximum value as the best value. As shown in Fig. 14, the highest point in this graph, is the lowest ESM value. The value of both parameters at this point is 0.03. These values are used to evaluate the proposed method.

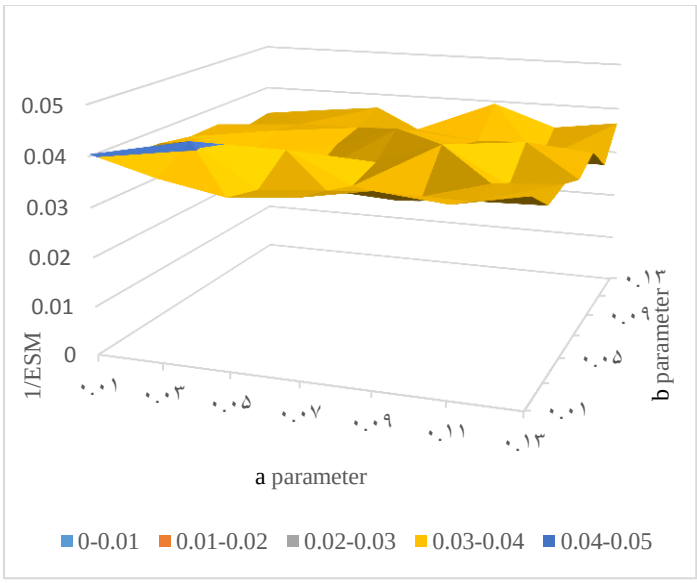

Fig. 14. 1/ESM results for tuning parameters a and b in the applied LA algorithm

Fig. 15 shows the use of different heuristic methods for different working days. The horizontal axis shows the working days and the vertical axis indicate the rate of heuristic methods use, and for each working day the total rate for using different heuristic methods is 100%. As the graph shows, for example, in the working day 9.Apr, the heuristic based on power level allocated itself about 30% of the total use of heuristic methods, while on this working day importance of the heuristic based on load level was less than 20%, but the same heuristic in the 6.Mar working day allocated itself over 30% of the total heuristic methods. This difference in the rate of use of different heuristics justifies the idea of using several different heuristics and weighing them. Because if all of the heuristics had been used equally throughout the working days, the use of the autonomous weight adjustment algorithm would not have been logical. So

different heuristics can have different functions and the proposed model using learning automata algorithm can apply heuristics with proper proportions at different time intervals.

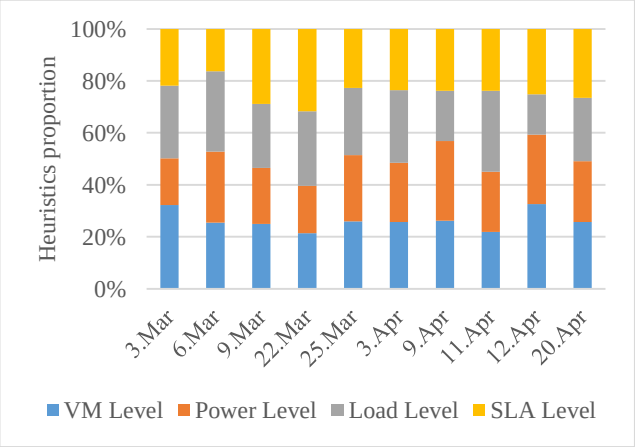

Fig. 15. Different rates of heuristic methods used for different workload days

Fig. 16 shows the violations of the service level agreement ($\times 10^{-5}$) for all virtual machines and for the 3.Mar workload day with the dotted graph. This chart shows the distribution of overall service level agreement violations for one working day. The reason for showing SLA violations is to show whether the violations in the proposed method has improved uniformly or a lot of virtual machines may have a very high level of service level agreement violations. As shown in Fig. 16 and Table 9, there are 1052 virtual machines for that day of workload. Each dot in this chart represents a virtual machine in data-set. The horizontal axis is the number of virtual machines and the vertical axis is overall SLAV value for every VM. Fig. 16 shows that SLA violations are below 2% for most of virtual machines, indicating that the proposed method was successful to place most virtual machines in the appropriate destination servers to avoid high-level SLA violation, and there are only a few virtual machines whose level of SLA violations are up.

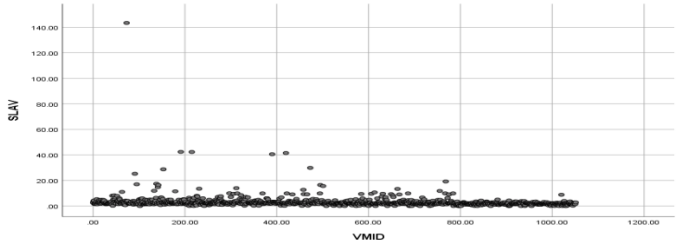

Fig. 16. Distribution of Virtual Machines on Service Level Agreement Violation for Workload day 20110303 in Proposed Method

5.4. Comparison Of The Proposed Autonomous Model With Other Scenarios

In this section, the proposed method, which includes all the steps of monitoring, analysis, planning and execution in the MAPE-K control loop, is compared with four basic methods. We consider 4 scenarios to do an efficient evaluation of the proposed method. The descriptions of scenarios are brought in Table 12.

Table 12
Details of basic algorithms for virtual machine consolidation

Algorithm names	Description
LR/MMT/PABFD/SM	LR policy for detecting overloaded hosts; MMT policy for VM selection; PABFD for VM placement; SM policy for detecting underloaded hosts [16]
LR/MMT/TOPSIS/SM	LR policy for detecting overloaded hosts [16]; MMT policy for VM selection [16]; TOPSIS for VM placement [31]; SM policy for detecting underloaded hosts [16]
MAD/MMT/PABFD/SM	MAD policy for detecting overloaded hosts [16]; MMT policy for VM selection; PABFD for VM placement; SM to detect underloaded hosts [16]
LR/MU/TOPSIS/MDL	LR policy for detecting overloaded hosts [16]; MU policy for VM selection [16]; TOPSIS for VM placement [31]; MDL policy for detecting underloaded hosts [31]

Fig. 17 compares the number of SLA violations of the proposed method with the other four scenarios. The results show that due to the timely forecasting of overloaded physical machines and taking into account criteria such as the number of virtual machines, CPU utilization and the number of overload status times of hosts in the proposed method, the number of SLA

violations compared to LR / MMT / PABFD /SM, LR / MMT / TOPSIS / SM, MAD / MMT / PABFD / SM and LR / MU / TOPSIS / MDL decreased by 64.14%, 74.11%, 46.76% and 83.31%, respectively.

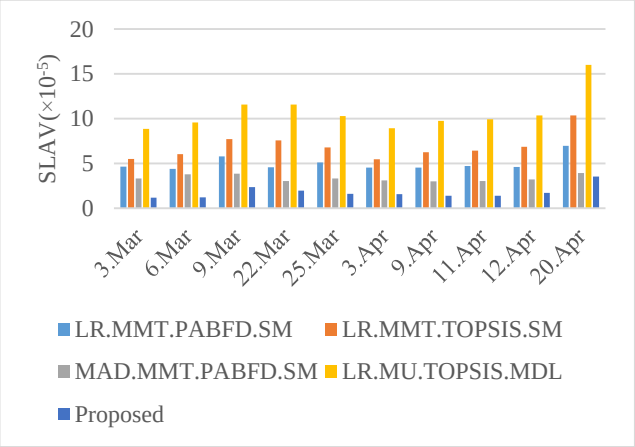

Fig. 17. SLA violation comparison of different VM consolidation methods

Fig. 18 compares the amount of energy consumed in the proposed method with the other four scenarios. Since the proposed method has been able to identify overloaded physical machines well and also place virtual machines on hosts with lower power consumption, the power consumption in the proposed method compared to LR / MMT / PABFD / SM, LR / MMT / TOPSIS / SM, MAD / MMT / PABFD / SM and LR / MU / TOPSIS / MDL scenarios decreased by 15.86%, 0.37%, 25.78% and 4.42%, respectively.

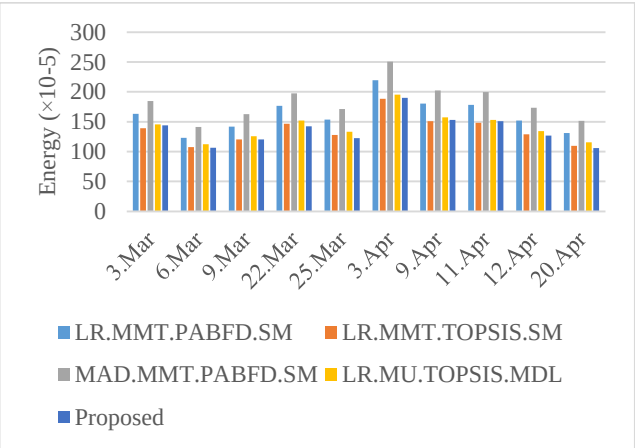

Fig. 18. Energy Consumption comparison of different VM consolidation methods

Fig. 19 compares the number of VM migrations of the proposed method with the other four scenarios. Because the proposed method largely prevents physical machines to become overloaded and correctly identifies overloaded physical machines, it has been able to reduce the futile migration of virtual machines. The results show that the number of virtual machine migrations in the proposed method compared to LR / MMT / PABFD / SM, LR / MMT / TOPSIS / SM, MAD / MMT / PABFD / SM and LR / MU / TOPSIS / MDL scenarios decreased by 67.05%, 74.11%, 64.7% and 76.7%, respectively.

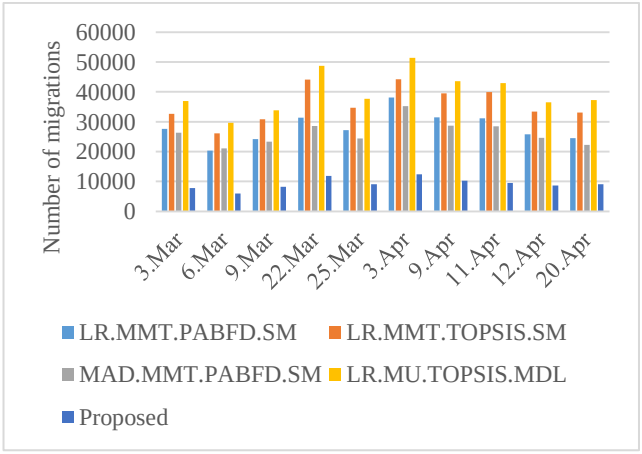

Fig. 19. Number of VM Migrations comparison of different VM consolidation methods

Fig. 20 and Fig. 21 compare the ESV and ESM rates of the proposed method with other four scenarios. The comparison results show that the proposed method was able to make a very good trade-off between energy consumption, the number of SLA violations and the number of virtual machine migrations.

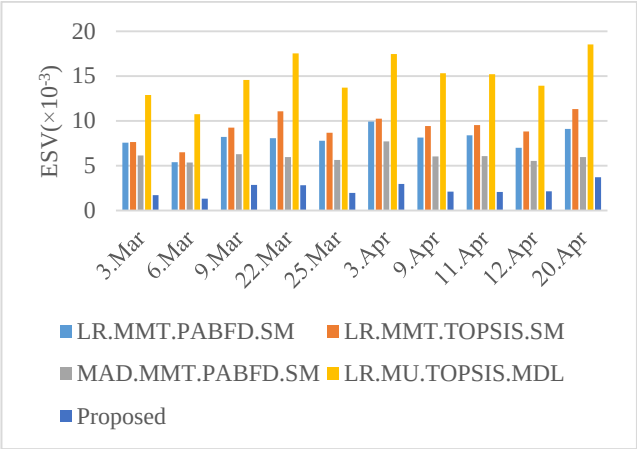

Fig. 20. ESV comparison of different VM consolidation methods

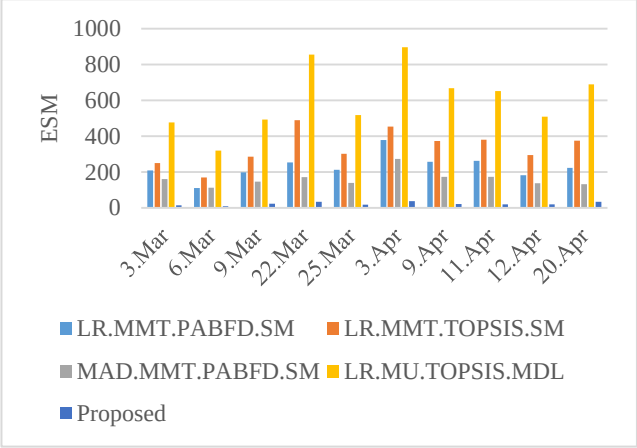

Fig. 21. ESM comparison of different VM consolidation methods

As shown in Fig. 17, Fig. 18 and Fig. 19, due to the reduction of all three criteria, number of SLA violations, energy and the number of VM migrations, the ESV rate of the proposed method compared to LR / MMT / PABFD / SM, LR / MMT / TOPSIS / SM, MAD / MMT / PABFD / SM and LR / MU / TOPSIS / MDL decreased by 70.35%, 74.49%, 61.11% and 84.26%, respectively. Also, the ESM rate in the proposed method compared to LR / MMT / PABFD / SM, LR / MMT / TOPSIS / MDL, MAD / MMT / PABFD / SM and LR / MU / TOPSIS / MDL decreased by 90.11%, 93.3%, 86.03% and 96.28%, respectively.

Table 13 shows the average results of comparing the proposed total method with other methods in 10 working days in terms of SLA violations, energy, migration counts, and ESV and ESM metrics. The results show that the proposed method has shown better results than the compared scenarios in all metrics.

Table 13
Average results on the proposed method and other methods for all workload days

Method/ Metrics	SLA Violation ×10-5	Energy	Migration Counts	ESV ×10-3	ESM
LR/MMT/ PABFD/SM	4.97	161.87	28174.7	7.95	228.04
LR/MMT/ TOPSIS/SM	6.88	136.69	35859.51	9.24	336.75
MAD/MMT/PABFD/SM	3.35	183.49	26305.1	6.06	161.46
LR/MU/TOPSIS/MDL	10.68	142.49	39855.27	14.98	607.27
Proposed	**1.78**	**136.19**	**9284.23**	**2.36**	**22.55**

Table 14 and Table 15 present the results of paired t-test for ESV and ESM. The Mean, standard deviation and p-value differences between the proposed and compared methods are calculated. Because the results of test shows that *p* is less than 0.05 in all cases, so the difference between proposed method and other methods is meaningful. It means that more than 95% the proposed method behaves better than others.

Table 14
Statistical analysis for ESV

Baseline Algorithm	Mean	Std. Deviation	p-value
LR/MMT/PABFD/SM	5.595925	0.790410	3.3502E-9
LR/MMT/TOPSIS/MDL	6.884898	0.891624	1.5517E-9
MAD/MMT/PABFD/SM	3.705292	0.702254	4.4604E-8
LR/MU/TOPSIS/SM	12.623966	1.748975	2.8231E-9

Table 15
Statistical analysis for ESM

Baseline Algorithm	Mean	Std. Deviation	p-value
LR/MMT/PABFD/SM	205.483406	62.040292	0.000002
LR/MMT/TOPSIS/MDL	314.200172	87.454274	0.000001
MAD/MMT/PABFD/SM	138.903456	38.691180	0.000001
LR/MU/TOPSIS/SM	584.713442	170.471140	0.000002

6.Conclusions and Future Works

With the increasing growth of cloud data centers, the energy consumption of these data centers is also increasing. Analyzing the load status of servers and examining different scenarios for selecting the appropriate server to be deployed by the virtual machine is a difficult process for the human factor. To solve such problems, an autonomous model with predictive capability is needed to effectively deploy virtual machines to the suitable servers at runtime. In fact, using the feedback system of autonomous systems can make this process simpler and more optimized. This paper presents a model based on the MAPE-K control loop for autonomous virtual machine consolidation. The proposed model uses an ensemble prediction algorithm for identifying overloaded servers. Also learning automata algorithm and a combination of several heuristic methods are used to select the appropriate destination servers for deploying virtual machines. The simulation results show that the proposed method has reduced averagely the service level agreement violations, energy and VM migration counts by 51.54%, 16.64 and 50.24 respectively, compared to other methods. In the future, reinforcement learning will be used for VM placement and autonomous selection of immigrant virtual machines will also be presented.

References

[1] S. Basu, G. Kannayaram, S. Ramasubbareddy, C. Venkatasubbaiah, Improved genetic algorithm for monitoring of virtual machines in cloud environment, in: Smart Intelligent Computing and Applications, Springer, 2019 319-326.https://doi.org/10.1007/978-981-13-1927-3_34105

[2] D. Agarwal, S. Jain, Efficient optimal algorithm of task scheduling in cloud computing environment, arXiv preprint arXiv:1404.2076, (2014),

[3] A. Ponraj, Optimistic virtual machine placement in cloud data centers using queuing approach, Future Generation Computer Systems, 93 (2019) 338-344.https://doi.org/10.1016/j.future.2018.10.022

[4] S.B. Shaw, A.K. Singh, Use of proactive and reactive hotspot detection technique to reduce the number of virtual machine migration and energy consumption in cloud data center, Computers & Electrical Engineering, 47 (2015) 241-254,

[5] M.C. Silva Filho, C.C. Monteiro, P.R. Inácio, M.M. Freire, Approaches for optimizing virtual machine placement and migration in cloud environments: A survey, Journal of Parallel and Distributed Computing, 111 (2018) 222-250.https://doi.org/10.1016/j.jpdc.2017.08.010

[6] R.W. Ahmad, A. Gani, S.H.A. Hamid, M. Shiraz, A. Yousafzai, F. Xia, A survey on virtual machine migration and server consolidation frameworks for cloud data centers, Journal of network and computer applications, 52 (2015) 11-25,

[7] Z. Li, An adaptive overload threshold selection process using Markov decision processes of virtual machine in cloud data center, Cluster Computing, 22 (2019) 3821-3833.https://doi.org/10.1007/s10586-018-2408-4

[8] M. Masdari, S.S. Nabavi, V. Ahmadi, An overview of virtual machine placement schemes in cloud computing, Journal of Network and Computer Applications, 66 (2016) 106-127.https://doi.org/10.1016/j.jnca.2016.01.011

[9] H.-P. Jiang, W.-M. Chen, Self-adaptive resource allocation for energy-aware virtual machine placement in dynamic computing cloud, Journal of Network and Computer Applications, 120 (2018) 119-129,

[10] Z. Luo, Z. Qian, Burstiness-aware server consolidation via queuing theory approach in a computing cloud, 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, (2013) 332-341,

[11] W. Voorsluys, J. Broberg, S. Venugopal, R. Buyya, Cost of virtual machine live migration in clouds: A performance evaluation, IEEE International Conference on Cloud Computing, (2009) 254-265,

[12] L. Hadded, F.B. Charrada, S. Tata, Optimization and approximate placement of autonomic resources for the management of service-based applications in the cloud, OTM Confederated International

Conferences" On the Move to Meaningful Internet Systems", (2016) 175-192.https://doi.org/10.1007/978-3-319-48472-3_10

[13] P. Jamshidi, A. Ahmad, C. Pahl, Autonomic resource provisioning for cloud-based software, Proceedings of the 9th international symposium on software engineering for adaptive and self-managing systems, (2014) 95-104,

[14] M. Mohamed, M. Amziani, D. Belaïd, S. Tata, T. Melliti, An autonomic approach to manage elasticity of business processes in the cloud, Future Generation Computer Systems, 50 (2015) 49-61,

[15] M. Mohamed, D. Belaïd, S. Tata, Extending OCCI for autonomic management in the cloud, Journal of Systems and Software, 122 (2016) 416-429,

[16] A. Beloglazov, R. Buyya, Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers, Concurrency and Computation: Practice and Experience, 24 (2012) 1397-1420. https://doi.org/10.1002/cpe.1867

[17] Z. Xiao, W. Song, Q. Chen, Dynamic resource allocation using virtual machines for cloud computing environment, IEEE transactions on parallel and distributed systems, 24 (2012) 1107-1117,

[18] P.A. Dinda, Design, implementation, and performance of an extensible toolkit for resource prediction in distributed systems, IEEE Transactions on Parallel and Distributed Systems, 17 (2006) 160-173.https://doi.org/10.1109/TPDS.2006.24

[19] J. Liang, K. Nahrstedt, Y. Zhou, Adaptive multi-resource prediction in distributed resource sharing environment, IEEE International Symposium on Cluster Computing and the Grid, 2004. CCGrid 2004., (2004) 293-300.https://doi.org/10.1109/CCGrid.2004.1336580

[20] E. Arianyan, H. Taheri, S. Sharifian, Novel heuristics for consolidation of virtual machines in cloud data centers using multi-criteria resource management solutions, The Journal of Supercomputing, 72 (2016) 688-717.https://doi.org/10.1007/s11227-015-1603-9

[21] J. Subirats, J. Guitart, Assessing and forecasting energy efficiency on Cloud computing platforms, Future Generation Computer Systems, 45 (2015) 70-94.https://doi.org/10.1016/j.future.2014.11.008

[22] M. Ghobaei-Arani, A.A. Rahmanian, M. Shamsi, A. Rasouli-Kenari, A learning-based approach for virtual machine placement in cloud data centers, International Journal of Communication Systems, 31 (2018) e3537. https://doi.org/10.1002/dac.3537

[23] 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 Systems with Applications, 120 (2019) 228-238.https://doi.org/10.1016/j.eswa.2018.11.029

[24] R. Shaw, E. Howley, E. Barrett, An energy efficient anti-correlated virtual machine placement algorithm using resource usage predictions, Simulation Modelling Practice and Theory, 93 (2019) 322-342.https://doi.org/10.1016/j.simpat.2018.09.019

[25] F. Farahnakian, A. Ashraf, T. Pahikkala, P. Liljeberg, J. Plosila, I. Porres, H. Tenhunen, Using ant colony system to consolidate VMs for green cloud computing, IEEE Transactions on Services Computing, 8 (2014) 187-198.https://doi.org/10.1109/TSC.2014.2382555

[26] H. Hallawi, J. Mehnen, H. He, Multi-Capacity Combinatorial Ordering GA in Application to Cloud resources allocation and efficient virtual machines consolidation, Future Generation Computer Systems, 69 (2017) 1-10.https://doi.org/10.1016/j.future.2016.10.025

[27] M.H. Ferdaus, M. Murshed, R.N. Calheiros, R. Buyya, Virtual machine consolidation in cloud data centers using ACO metaheuristic, European conference on parallel processing, (2014) 306-317.https://doi.org/10.1007/978-3-319-09873-9_26

[28] F. Teng, L. Yu, T. Li, D. Deng, F. Magoulès, Energy efficiency of VM consolidation in IaaS clouds, The Journal of Supercomputing, 73 (2017) 782-809.https://doi.org/10.1007/s11227-016-1797-5

[29] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future generation computer systems, 28 (2012) 755-768.https://doi.org/10.1016/j.future.2011.04.017

[30] A. Horri, M.S. Mozafari, G. Dastghaibyfard, Novel resource allocation algorithms to performance and energy efficiency in cloud computing, The Journal of Supercomputing, 69 (2014) 1445-1461.https://doi.org/10.1007/s11227-014-1224-8

[31] E. Arianyan, H. Taheri, S. Sharifian, Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers, Computers & Electrical Engineering, 47 (2015) 222-240.https://doi.org/10.1016/j.compeleceng.2015.05.006

[32] S. Singh, I. Chana, M. Singh, R. Buyya, SOCCER: self-optimization of energy-efficient cloud

resources, Cluster Computing, 19 (2016) 1787-1800.https://doi.org/10.1007/s10586-016-0623-4

[33] S. Singh, I. Chana, R. Buyya, STAR: SLA-aware autonomic management of cloud resources, IEEE Transactions on Cloud Computing, (2017).https://doi.org/10.1109/TCC.2017.2648788

[34] S. Singh, I. Chana, EARTH: Energy-aware autonomic resource scheduling in cloud computing, Journal of Intelligent & Fuzzy Systems, 30 (2016) 1581-1600.10.3233/IFS-151866

[35] S.S. Gill, I. Chana, M. Singh, R. Buyya, CHOPPER: an intelligent QoS-aware autonomic resource management approach for cloud computing, Cluster Computing, 21 (2018) 1203-1241.https://doi.org/10.1007/s10586-017-1040-z

[36] M. Ghobaei-Arani, S. Jabbehdari, M.A. Pourmina, An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach, Future Generation Computer Systems, 78 (2018) 191-210.https://doi.org/10.1016/j.future.2017.02.022

[37] E. Outin, J.-E. Dartois, O. Barais, J.-L. Pazat, Enhancing cloud energy models for optimizing datacenters efficiency, 2015 International Conference on Cloud and Autonomic Computing, (2015) 93-100.https://doi.org/10.1109/ICCAC.2015.10

[38] M. Maurer, I. Breskovic, V.C. Emeakaroha, I. Brandic, Revealing the MAPE loop for the autonomic management of cloud infrastructures, 2011 IEEE symposium on computers and communications (ISCC), (2011) 147-152.https://doi.org/10.1109/ISCC.2011.5984008

[39] J.O. Kephart, D.M. Chess, The vision of autonomic computing, Computer, 36 (2003) 41-50.https://doi.org/10.1109/MC.2003.1160055

[40] B. Jacob, R. Lanyon-Hogg, D.K. Nadgir, A.F. Yassin, A practical guide to the IBM autonomic computing toolkit, IBM Redbooks, 4 (2004),

[41] S. Younesszadeh, M.R. Meybodi, A link prediction method based on learning automata in social networks, Journal of Computer & Robotics, 11 (2018) 43-55,

[42] 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, Software: Practice and experience, 41 (2011) 23-50. https://doi.org/10.1002/spe.995

[43] K. Park, V.S. Pai, CoMon: a mostly-scalable monitoring system for PlanetLab, ACM SIGOPS Operating Systems Review, 40 (2006) 65-74.https://doi.org/10.1145/1113361.1113374