Novel energy-aware approach to resource allocation in cloud computing

Article *in* Multiagent and Grid Systems · December 2021

DOI: 10.3233/MGS-210350

CITATIONS
2

READS
125

3 authors:

Saidi Karima
Batna2 University
5 PUBLICATIONS **15** CITATIONS
SEE PROFILE

Hioual Ouassila
Abbes Laghrour - Khenchela University
35 PUBLICATIONS **62** CITATIONS
SEE PROFILE

Siam Abderrahim
Abbes Laghrour - Khenchela University
15 PUBLICATIONS **40** CITATIONS
SEE PROFILE

Multiagent and Grid Systems – An International Journal 17 (2021) 197–218
DOI 10.3233/MGS-210350
IOS Press

Novel energy-aware approach to resource allocation in cloud computing

Karima Saidi[a,*], Ouassila Hioual[b] and Abderrahim Siam[a]
[a]*ICOSI Laboratory, Abbes Laghror University, Khenchela, Algeria*
[b]*Abbes Laghror University, Khenchela, LIRE Laboratory, Constantine2, Algeria*

Received 24 March 2021
Accepted 15 October 2021

Abstract. In this paper, we address the issue of resource allocation in a Cloud Computing environment. Since the need for cloud resources has led to the rapid growth of data centers and the waste of idle resources, high-power consumption has emerged. Therefore, we develop an approach that reduces energy consumption. Decision-making for adequate tasks and virtual machines (VMs) with their consolidation minimizes this latter. The aim of the proposed approach is energy efficiency. It consists of two processes; the first one allows the mapping of user tasks to VMs. Whereas, the second process consists of mapping virtual machines to the best location (physical machines). This paper focuses on this latter to develop a model by using a deep neural network and the ELECTRE methods supported by the K-nearest neighbor classifier. The experiments show that our model can produce promising results compared to other works of literature. This model also presents good scalability to improve the learning, allowing, thus, to achieve our objectives.

Keywords: Cloud computing, resource allocation, energy-efficient, ELECTRE methods, K-nearest neighbor, deep neural network

1. Introduction

Cloud computing is a distributed computing model that uses resources on the Internet. It is designed to run large-scale applications. This is a cost-effective way to achieve the Service Level Agreement (SLA) and Quality of Service (QoS) according to application demand.

There are many similarities between Cloud, Grid, and Cluster Computing. However, the special feature and, at the same time, the main advantage of this technology is the use of virtualization. This latter is a technology that separates computational and physical resources [1].

One of the biggest issues of Cloud computing is how to allocate resources to user requests. These resources can be for example CPU, memory capacity, storage space, or network bandwidth. Resource allocation can be described as an efficient allocation of resources among several users according to their demands over a given period [2]. This problem presents several challenges, the two main ones are task scheduling [3,4] and energy-efficient [5,6].

Recently, energy-efficient has become a major issue in large data centers. This is due to financial and environmental impacts. Therefore, the increasing use of Cloud computing resources is expected to quickly increase the energy consumption used in the data center.

*Corresponding author: Karima Saidi, ICOSI Laboratory, Abbes Laghror University, Khenchela, 40004, Algeria. E-mail: kariming2008@gmail.com or saidi.karima@univ-khenchela.dz.

ISSN 1574-1702/$35.00

Several research works have been proposed to reduce energy consumption in Cloud computing [7]. These latter shared the idea of turning off unused physical machines (PMs) to increase their use. This idea is to allocate virtual machines (VMs) to the smallest possible number of PMs (VM consolidation) [8].

There are some difficulties in the issue of online consolidation [9]. These difficulties are how to identify PMs' status in a data center, overloaded and underloaded PMs; thus, how to identify a load of overloaded PMs by choosing a subset of their VMs for migration.

Among the effective solutions that have been proposed, we can cite those based on the theory of prediction. These allowed the allocation of resources to be faster [10].

With more computational resources, we need more depth in computation. Therefore, deep learning is a significant way to increase the amount of available data to calculate. A Deep Neural Network (DNN) can develop a leading solution to the engineering shortage. It produces more layers in a simple neural network with supporting some techniques that help us to reach the best result.

In the present work, we focus on proposing a proactive approach for resource allocation. For that, we were based on the work presented in [11], which used a metaheuristic method to determine the best placement of tasks and VMs. Our contribution is based on the combination of deep learning, K-Nearest Neighbor (KNN), and Multicriteria Decision Analysis (MCDA) methods. The main contributions of this paper are:

- The presentation of an approach to integrate task scheduling and VM Placement as one problem. We investigate effective parameters with more objectives in the cloud environment because earlier works in this field focused on task scheduling or VM placement as a separate problem.
- The proposition of a new proactive approach for distributing resources among different users in Cloud computing. That allows us to maximize resource utilization at each PM and to minimize the overall energy consumption in the data center,
- The proposition of a new algorithm that combines the KNN classifier and ELECTRE method, which decides the best placement of each selected VM (for migration or selection),
- The use of a modified nonlinear DNN model, which enables us to improve our learning by proposing new algorithms and increasing the impact of our model's data. Additionally, we use transfer learning to make learning easier.

The rest of the paper is organized as follows. Section 2 investigates related work in recent literature. Section 3 presents the proposed work and addresses the architecture of our approach. Then, Section 4 details the proposed approach by describing the functioning of algorithms. Some experiment results are presented in Section 5. Finally, a conclusion and some future works are presented in Section 6.

2. Literature review

In these recent years, several ideas have been suggested. A review in [10] discussed several research papers which addressed the problem of resource allocation in different ways. Currently, there are various studies in Cloud data centers that focus on the development of methods and mechanisms to solve this problem by using heuristic and meta-heuristic algorithms. Like [12], the authors used Teaching Learning-Based Optimization algorithm (TLBO) and Grey Wolves (GW) to solve the problem. The contribution in [13] was mainly the use of the Shuffled Frog Leaping Algorithm (SFLA) with the Genetic Algorithm (GA). Furthermore, the authors in [5] employed Multi-objective Ant Colony Optimization (MACO).

There are two main strategies for energy efficiency in cloud data centers: Dynamic Voltage and Frequency Scaling (DVFS) and consolidation of the execution of multiple VMs on the same physical

host [14]. Besides, VM Placement is another aspect to solve the energy-efficient resource allocation. VM placement with consolidation can be considered as a bin packing issue with bins of various sizes, it is NP-hard [8]. The known algorithms that are used to solve this latter are First Fit Decreasing (FFD) [15] and Best Fit Decreasing (BFD) [16]. However, they were not focused on energy consumption, the VM may be placed on a server that has low computational capacity but consumes high power. On the other hand, Modified Best Fit Decreasing (MBFD) algorithm by Beloglazov and al. [2] is used to minimize the energy consumption by VM placement. This latter is based on the BFD algorithm that is used to sort the VMs in descending order based on their CPU requirements. After sorting, all VMs are assigned to the hosts based on the power model. Additionally, [17] proposed Resource-aware Virtual Machine Placement (RVMP) algorithm to minimize energy consumption and reduce the active PMs. Thus, the number of needed active PMs relies on the placement of VMs on PMs, as mentioned [18]. The latter focused on two significant ways to reduce energy consumption: VM placement and consolidation. Concerning VMs consolidation, it allowed predicting the number and the amount of VM requests to be received in the future [19,20]. Authors in [21] presented another formulation of a VM consolidation by predicting overutilized and under-utilized host detection techniques using the Markov chain model. The VM workload prediction helps decision-makers in mapping VMs to PMs (VM placement). Based on this principle, authors in [22] had demonstrated that the use of the deep learning model had become a critical factor in resolving these complicated issues due to the depth of layers in the neural network.

Based on this comprehensive literature, current solutions can be classified, depending on the used areas, into two key groups: AI-based solutions and MCDA-based solutions.

2.1. AI-based solutions

These last years, there is a lot of enthusiasm around artificial intelligence especially machine learning and deep learning.

2.1.1. Research work based on Machine Learning (ML)

The work presented in [23] reduces the workload (with minimal response time and cost). The authors used machine learning to create an efficient knowledge model and applied software self-adaptation technology. Moreover, these authors applied the genetic algorithm to solve the problem of allocating computing resources, which ensures that the solution can optimize the configuration of resources. In this work, the VM pool is responsible to manage the number of VMs, create and close VMs at the right time.

The work presented in [24] explains the role of the K-means method in the context of the use of a proactive approach to consolidate VM. K-means is one of the most commonly used clustering methods. The issue of resource allocation is solved by using prediction techniques for machine learning. The latter combined with the gray forecast can be used to predict the behavior of nonlinear time series. Besides, another usage of this method mentioned [20,25,26], is to design an approach to find VMs of the appropriate size to optimize resource utilization. Energy efficiency was a major challenge in the proposed approach; it allowed fewer VM instances to be used and proved to be able to reduce the number of rejected tasks, which had reduced energy waste in data centers.

2.1.2. Research work based on Deep Learning (DL)

Several papers addressed the problem of how Cloud computing includes company-owned deep learning because this system offers databases, storage, and networking tools that require deep computational learning with a very broad dataset and different platforms that are provided by multiple providers. Based on [10], we can cite some of these works. In [27], researchers proposed a hierarchical framework to solve

the resource allocation problem at a global level on VMs and power management, and at a local level on local servers. This work was based on deep learning and more precisely on Deep Reinforcement Learning (DRL). Decision-making was done automatically from the reduction of the state and action spaces for the allocation of global resources. Otherwise, the workload predictor was responsible for providing future workload forecasts to facilitate the local operation of the Long Short-Term Memory (LSTM) power management algorithm.

Authors in [28] considered load balancing and the Service Broker strategy as two main areas to solve the problem of resource allocation. First, a Multi-Agent Deep Reinforcement Learning model (MADRL-DRA) was used, wherein the Local User Agent (LUA) was used to predict the environmental activities of the user task and to allocate the task to the VM based on priority. Then, a Load Balancing (LB) was performed in the VM, which increases the flow rate and reduces the response time of the resource allocation task. Secondly, Dynamic Optimal Service Load-Aware Service Broker (DOLASB) was used in the Global User-Agent (GUA) to plan the task and to provide services to users based on the cloud brokers available to minimize the costs of cloud customers, and to generate a profit for Cloud Service Broker (CSB) at the same time. Finally, the authors proposed the Benders Decomposition method with the Mixed Integer Programming algorithm (BD-MIP) algorithm that provides an optimal solution to the problems of optimizing multiservice configuration, VM allocation, and CSB.

2.2. MCDA-based solutions

MCDA Methods were designed to assist decision-making in either the choosing or ranking of actions [29]. Furthermore, according to the research in [10], we noticed that the use of these methods was remarkable in some papers published in the literature in recent years to solve this problem. For example, authors in [3] proposed a heuristic algorithm that efficiently schedules tasks and allocates resources in the cloud. The authors have combined the Modified Analytical Hierarchy Process (MAHP), Bandwidth Aware Divisible Scheduling (BATS), BAR optimization, Longest Expected Processing Time preemption (LEPT), and division and conquest methods to perform the task and resource planning. The MAHP process allows scientific tasks to be prioritized, and the combination of BATS and BAR optimization methods allows resources to be allocated according to bandwidth constraints and cloud resource load. Besides, LEPT preemption was used to give the status of the VM, and a modified divided methodology to conquer was proposed to aggregate the results after task preemption.

Authors in [30] proposed an energy-efficient task scheduling algorithm based on Best-Worst Methodology (BWM) and a Technique for Order Choice based on Optimal Solution Similarity (TOPSIS). The ranking process using BWM-TOPSIS methodology allowed, according to the authors, to determine which more important cloud scheduling solution is more comprehensive and principled. Additionally, an energy-efficient dispatcher was responsible for two important processes, the first is to dispatch different tasks among dynamic priority queues into four levels (very high, high, medium, and low); and the second is to send the next task from dynamic priority queues to the proper VM for processing; all of which takes place after having ranked the tasks and VMs.

None of the previous works use multi-classification by deep supervised learning to solve the resource allocation energy consumption issue. In addition, few of the above works rely on deep learning and MCDA methods to look at their advantages to solve this issue. Thus, this paper proposes the improvement of learning based on multicriteria for achieving our objectives. A review of some previous works based on AI (DL & ML) and MCDA methods in the environment of Cloud computing is illustrated in Table 1.

Table 1
A comprehensive review of previous literature based on AI (DL & ML) and MCDA methods in Cloud computing

Type of classification	Author name	Methods/ technique	Contributions	Great challenge	Performance metric			
					Cost	Time	Load balancing	Energy consumption
Based on AI (ML&DL)	Dabbagh and al. [20]	K-means & stochastic theory	Predicting the number and the amount of VM requests to be received in the future Use real Google data traces	Energy efficient	×	×	×	√
	Lin, and al. [23]	ML & GA	Software self-adaptation technology e.g., MAPE-K and control loop	Cost Response time	√	√	×	×
	Zhou and al. [25]	ATEA	The dynamic thresholds are more energy-efficient than the fixed threshold	Energy efficient	×	×	×	√
	Alsadie, and al. [26]	K-means	Determine the appropriate VM type based on Google Cloud traces	Energy efficiency	×	×	√	√
	Liu, and al. [27]	DRL	Used a Global Resource Allocation Framework based on Deep Neural Networks (DNN) and A deep Q-learning framework	Energy efficiency Elasticity	×	×	√	√
	Jyoti, A. and Shrimali, M. [28]	MADRL-DRA & DOLASB	Based on load balancing and Service Broker strategy Use the CloudSim simulator	Elasticity Task scheduling Energy efficiency	√	√	√	√
Based on MCDA methods	Khorsand and Ramezanpour [30]	BWM-TOPSIS	Determine the cloud scheduling solution is more important to select	Energy-efficient Task scheduling	×	×	√	√
	Gawali, and al. [3]	MAHP & BATS & LEPT	Cloud Simulator in the experimental phase Uses real Cyber-shake and Epigenomics scientific work-flows as core tasks for the system	Task scheduling	√	√	×	×
AI (ML&DL) + MCDA	Our proposal	DNN & ELECTRE methods & KNN	New proactive approach for the placement of tasks and VMs with their consolidation	Energy efficient	×	×	√	√

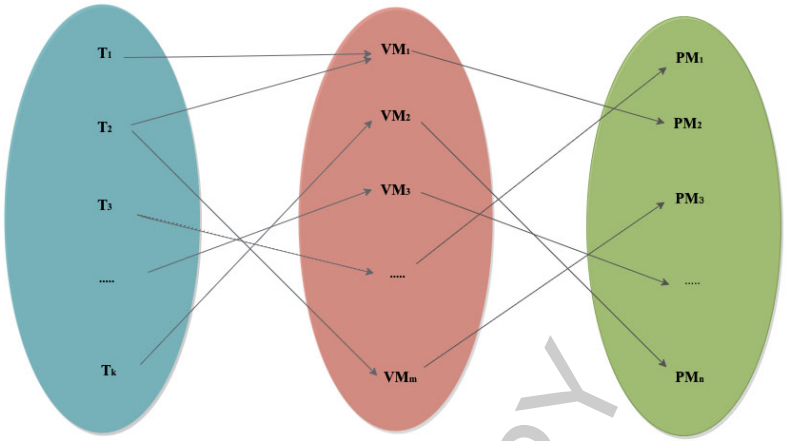


Fig. 1. Multipartite graph of the proposal processes.

3. Proposed approach

In this section, we will define the general architecture and the model of the proposed approach. Initially, we present, in the first subsection, the general proposal processes to clarify this approach. Then, in the next ones, we will detail our approach.

3.1. Overview of the overall approach

There are three key layers of Cloud Computing: Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS). This paper focuses on how to address the resource allocation issue in the Infrastructure as a Service layer. Thus, our approach determines the appropriate placement prediction. Moreover, the resource allocation problem can be treated as a two-processes vector bin packing problem. The first process allocates tasks to VMs, and the second one allocates VMs to PMs (Since we focus on the CPU as the main resource in our work). The multipartite graph below (cf. Fig. 1) illustrates the different processes of our proposal which are defined as:

- A set of tasks T $= (T_1, T_2, \ldots, T_k)$, where $p \in [1, k]$ and k is the total number of tasks,
- A set of VMs VM $= (VM_1, VM_2, \ldots, VM_m)$, where $j \in [1, m]$ and m is the total number of VMs and,
- A set of PMs PM $= (PM_1, PM_2, \ldots, PM_n)$, where $i \in [1, n]$ and n is the total number of PMs.

The model that we use is the result of data training. Thus, the data are trained to create an accurate model that correctly responds to a request. To build our model, we take into consideration the following points:

(1) We define a learning problem as a process of studying the weight values of the layers in the network. The input can be correctly mapped to its corresponding target. Thus, it is a supervised collection of learning data.
(2) The used method for supervised learning is classification since the classes are the labels on which we classify new data.
(3) Multiple classifications are a basic classification extension. They are used to help in the prediction of the new data category among multiple classes (labels) depending on several parameters or requirements.

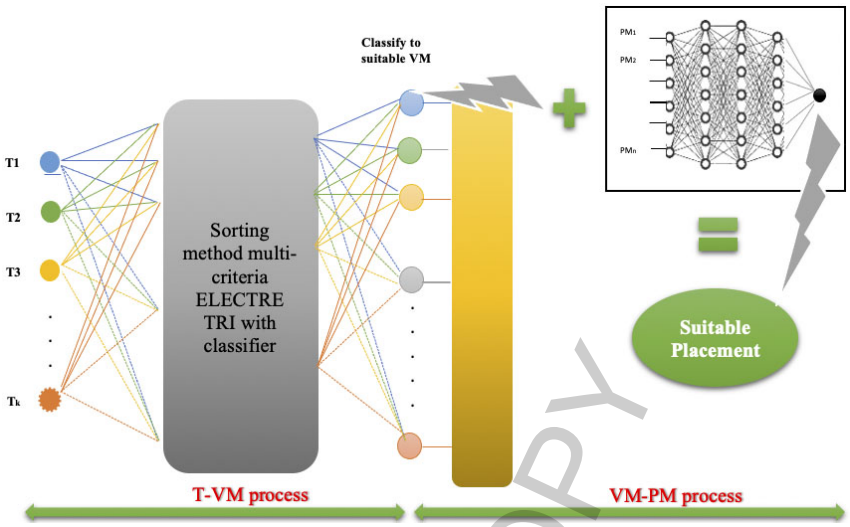


Fig. 2. The proposed DNN-based architecture.

(4) We have chosen the deep learning algorithm to build a model since there is a vast amount of request resources to carry out various tasks in the cloud system. Thereby, deep learning works better with massive amounts of data and complex problems to solve.
(5) We have chosen the MCDA methods (multicriteria classification) because among the relevant characteristics to use these methods is that the decision-maker (DM) must include in the model at least three criteria [29].

3.2. *Description of the proposed approach*

The architecture of our system is a DNN-based one, it is composed of two processes. The first one allows a new task to be allocated to the correct VMs (T-VM) when the cloud receives a user's task request. Whereas, the second process is responsible for choosing the best PM based on a set of parameters. We assume that each task has a set of characteristics according to user preferences (user needs), such as memory capacity, CPU speed, storage space, and network bandwidth (computing resource requirement).

The proposed model in the first process (cf. Fig. 2), uses the method of evaluating multicriteria. Since our model considers heterogeneous VMs, we classify a collection of user tasks into three categories, according to the size of VMs, as small, medium, and large. Then, we use the adequate classifier to assign each task to acceptable VMs.

Afterward, we need another model to choose the most appropriate physical device for each VM (VM-PM). In the second process (cf. Fig. 2), we use a combination of DNN with the ELECTRE method. We use also the KNN classifier to classify each VM in the appropriate PM class. This process aims to improve energy efficiency by reducing energy consumption, resource use, load balancing between active PMs, and VM consolidation. It is a multi-objective approach.

In our work, we consider, for each PM, the following characteristics: its use of resources, the energy consumption, its current status (Active, Idle, or Inactive), and the cost to assign each VM to an adequate PM.

Our approach consists of three components. The first one is the Hidden Layer Controller (HLC), whose role is to execute the proposed algorithms in collaboration with hidden layers. The second

Table 2
List of notations used in this paper

Notation	Description
IDs	Identifications of VMs
$\mathbf{UPM}_{\mathbf{CPU}}^{\mathbf{idle}}$	CPU resource utilization percentage of PM in idle mode
A	An action
Cl	A class
Ucl, Mcl, Ocl	The threshold of 3 classes (Underloaded, Moderate loaded, and Overloaded energy consumption)
Ua	The performance of an action
$\mathbf{UPM}_{\mathbf{CPU}}^{\mathbf{underloadEC}}$	CPU resource utilization percentage in each Underloaded energy consumption PM
$\mathbf{AU}_{\mathbf{cpu}}$	Average CPU utilization of a PM
$\mathbf{NPM}_{\mathbf{active}}$	Number of active PMs
$\mathbf{EC}_{\mathbf{PM}}$	Energy consumption of PM
$\mathbf{EC}_{\mathbf{total}}$	The total energy consumption in a data center
$\mathbf{EC}_{\mathbf{PM}}^{\mathbf{idle}}$	Energy consumption of EC_{PM} in idle mode
$\mathbf{EC}_{\mathbf{PM}}^{\mathbf{max}}$	Energy consumption ofEC_{PM} in the maximum loaded mode
MinU	The minimum threshold of the utilization percentage of Underloaded energy consumption PM

component (Physical Machine Controller (PMC)) allows us to collect information such as the resource utilization percentage of each PM, as well as other requested data such as the current state of PMs, energy consumption, the number of VMs assigned to each PM, and the number of VMs migration, which must be updated after any modification. The objective of the third component (Central Controller (CC)) is how to make a final decision.

3.3. Assumptions and restrictions

For the proper functioning of our approach, we assume that:

- A task must be assigned to multiple VMs,
- Each VM can be provisioned to more than one task,
- A VM must be assigned to one PM,
- VMs and PMs are heterogeneous in the Cloud data center.

4. Proposed algorithms functioning

In this section, we present the proposed algorithms and describe how they operate. The notations used in the remainder of this paper are listed and described in Table 2.

4.1. T-VM process

The main idea of the below algorithm is to classify all tasks based on their criteria. We need to analyze our input data and formulate them as a vector or a tensor of rank 1. In our context, the input data represents the characteristics of the user tasks that reflect a set of requirements such as the CPU, memory capacity, storage space, and network bandwidth. The tasks are distributed within three categories; each one has similar criteria (CPU, memory capacity, etc.). After assigning tasks to the suitable category, the mapping of these tasks to adequate VMs is done in two steps. In the first step, each category will be mapped separately to another category. For this, the ELECTRE TRI method (ELimination Et Choix TRadusiant la REalite in English meaning Elimination and Choice Expressing Reality TRI) is used to sort alternatives. In the second step, depending on the class of the task, classifier 1, Classifier 2, or Classifier 3 will be

executed to assign each task to suitable VMs. The responsible for this process is HLC1; Algorithm 1 below represents its operation.

P.S. The operation of Classifier 1, Classifier 2, and Classifier 3 is not the subject of this paper.

Algorithm 1: Assigning each task to suitable VMs

Input: data-task (Required CPU, Required memory capacity, Required storage space, Required bandwidth)
Three classes of tasks (small, medium, and large)
Output: Placement of each task in acceptable VMs (The output must be IDs of VMs)
1 initial data-task (Required CPU, Required memory capacity, Required storage space, Required bandwidth)
2 **For** each task **do**
3 Apply ELECTRE Tri to classify each task in one of the three classes (small, medium, large)
4 End for
5 **Switch** classes
6 Case1: small
7 apply Classifier 1 break;
8 case 2: medium
9 apply Classifier 2 break;
10 case 3: large
11 apply Classifier 3break;
12 end Switch

4.2. VM-PM process

This subsection presents the prediction model that is proposed in [31] and improved, in this paper, by supporting the KNN classification algorithm. To improve how to predict the position of each VM, we will use another DNN classifier that combines ELECTRE methods and a KNN classifier, which we call Modified DNN (MDNN). To generate the process of VM-PM in our model, we need to migrate VMs to other PMs. This step allows us to improve energy consumption.

4.2.1. MDNN model

In the MDDN model, HLCs execute four HLC algorithms (Algorithm 3–6) in collaboration with its hidden layers. We introduce these algorithms, in our MDNN, due to the constant changes in the data center. Moreover, using deep learning enables the extraction of high-level features to predict the best placement of new data and the update of weights according to the output of each algorithm.

In the MDNN prediction model, we can improve accuracy by adding HLC algorithms. These latter integrate the algorithm of ELECTRE-KNN (Algorithm 2) to make the process of VM-PM collaborate with the number of hidden layers. Therefore, we need the outputs of the HLC algorithms as the inputs of the following layer to improve our learning by taking into consideration the Cloud computing environment needs that change constantly. Moreover, since we change the weights progressively, we need an algorithm to reduce errors in the MDNN. For this, we use the backpropagation learning algorithm (cf. Fig. 3).

The proposed model is composed of three components: the PMC, HLC, and CC. The PMC and HLC5 (cf. subsection 4.2.3. (4)) are used to generate decisions to be taken by the CC later. After applying the algorithm HLC5, the controller of the PM (PMC) sends a message to the CC to verify the placement of each VM and to assign VMs or confirm their placement.

4.2.2. KNN-ELECTRE algorithm

In our context, we need to use a classifier (KNN) combined with the ELECTRE method. The large capacity of KNN allows it to achieve high accuracy with a large drive assembly. However, when the data set is limited, the training result becomes unreliable [32]. To solve this problem, we have combined KNN

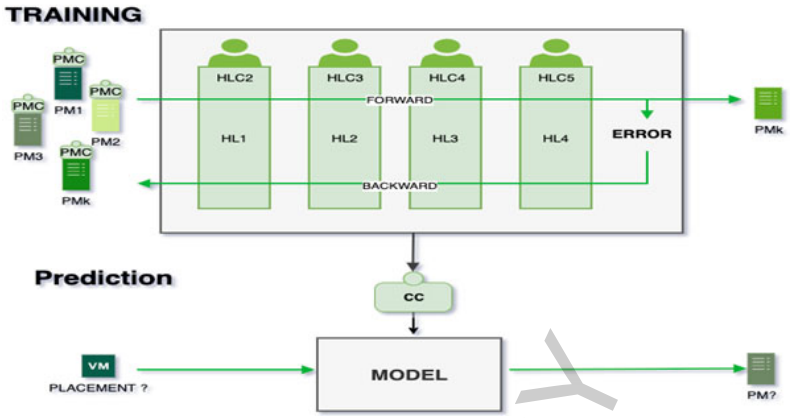


Fig. 3. Training and prediction of MDNN model.

with the ELECTRE III method because this latter use the discriminating (indifference and preference) threshold [29]. This combination allows finding the best placement (PM) of each selected VM (for migration or selection). The operation of this component is illustrated through Algorithm 2 below. However, the application of KNN with the ELECTRE III method in complex systems (with high computational data) involves a large amount of data that take more time and memory. For that, we suggest calling Algorithm 2 by HLC algorithms to make the process of VM-PM, whenever selecting or migrating the VMs.

Algorithm 2: (Test_data): KNN-ELECTRE

Input: Test_data, data_VM (training data)
Output: Return the class (PM) of Test_data
1 Initialize K by assuming the number of neighbors which is the number of PMs
2 **For** K numbers of PMs **do**
3 Calculate the Euclidean distance between the data_VM and the current Test_data using Eq. (1)
4 Save the distance and the ID of each VM
5 End for
6 Calculate the number of VMs assigned to each PM.
7 Save the shortest distance between VMs that compose each PM.
8 Calculate the energy consumption and the cost of each PM.
9 Sort PMs in ascending order by using ELECTREIII with the following criteria: The number of VMs, distance, energy consumption, and cost of each PM.
10 Assign Test_data class (PM) to the first element of the sorted collection (see line 9).

As mentioned above, the problem of heterogeneous characteristics with multidimensional parameters between PMs and VMs is solved by combining KNN-ELECTRE. When increasing the value of K, our prediction becomes more stable due to the majority voting. Thus, it is more likely to make accurate predictions.

We have chosen the shortest distance between VMs and PMs because it reduces the required time to complete the process of migration to other PMs. Therefore, time is an essential factor for energy dissipation [33].

In our work, the Euclidean Distance (ED) is between Test_data and training data. It is calculated as follows [34]:

Let D1 and D2 which are represented by feature vectors D1 $= (x_1, x_2, \ldots, x_m)$ and D2 $= (y_1, y_2, \ldots, y_m)$, where m is the dimensionality of the feature space.

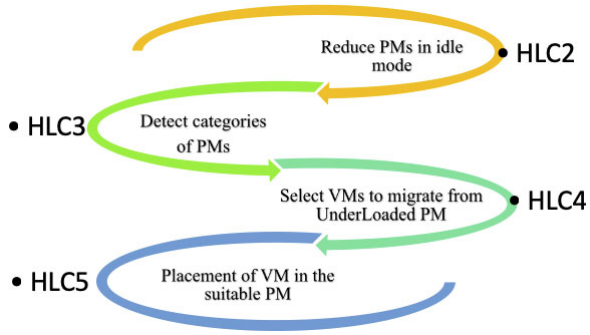


Fig. 4. The functioning of different HLC algorithms.

$$ED(D1, D2) = \sqrt{\frac{\sum_{i=1}^{m}(x_i - y_i)^2}{m}} \tag{1}$$

The feature vectors represent the parameters of each VM: The Capacity of CPU, Use of VM, and Cost.

We consider only one type of resource requirement (CPU Resource) because it is the most energy-consuming resource compared to other resources [35]. PM energy consumption is the sum of the energy that all VMs consume on this active PM [36], it is calculated by Eq. (2).

$$EC_{PM} = EC_{PM}^{idle} + (EC_{PM}^{max} - EC_{PM}^{idle}) \times AU_{cpu} \tag{2}$$

This algorithm not only considers the capacity of available resources of the PM (Since it does not necessarily have less energy than the others), but it also takes into account the number of VMs on each PM.

4.2.3. Proposed HLC algorithms

There are four HLCs that execute Algorithms 3 to 6 in cooperation with the DNN hidden layers to save energy. In particular, the output of each HLC serves as the input to the following HL. In other words, the connections between layers are the outgoing connections from all neurons in the previous layer with the outputs of the HLC algorithm to all the neurons in the next layer.

Algorithm 3 reduces the energy in the data center by reducing the number of PMs in the idle state as it consumes a lot of energy. Algorithm 4 categorizes three classes of PM; each class has a specific objective. Then, Algorithm 5 allows some VMs to be moved to another server. While Algorithm 6 takes care of the latest underloadCE class to select the best placement for VMs (See Fig. 4).

(1) HLC2-Algorithm

The role of the HLC2-Algorithm (cf. Algorithm 3) is to switch some PMs from idle to sleep mode. This algorithm shows that an idle PM with a few affected VMs has a greater chance of transit to sleep mode compared to a PM with a significant number of VMs.

The advantages of the above algorithm are:

- Finding the CPU resource usage percentage of each PM in the idle mode,
- When UPM_{CPU}^{idle} is less than 20%, we can migrate all VMs in this PM and switch it to the sleep mode. The PM in this state consumes a lot of energy which can reach 70%,
- Finding the best placement to migrate VMs from PMs in idle mode.

According to [37], we can assume that the CPU resource utilization percentage thresholds of PM classes are illustrated in Table 3.

Algorithm 3: How to switch some PMs from idle to sleep mode

Input: PMs list
Output: Some PMs transferred from idle to sleep mode
1. **For** each PM **Do**
2. **If** PM mode = idle **then**
3. Calculate the current UPM_{CPU}^{idle}
4. **If** $UPM_{CPU}^{idle} < 20\%$ **Then**
5. **For** each VM assigned to this PM **do**
6. Migrate VM to another PM by using Algorithm 2 (VM) to the best-selected placement.
7. Put PM in sleep mode
8. End

Table 3
The CPU resource utilization percentage thresholds of PM classes

Classes of PM	Utilization percentage threshold
UnderloadEC	1 to 20%
Moderate loadEC	21 to 70%
OverloadEC	71 to 99%

(2) HLC3-Algorithm

The HLC3-Algorithm (cf. Algorithm 4) enables the identification of underloadEC, overloadEC, and moderate loadEC classes. In this work, we use MCDA specific sorting methods that allow us to assign a PM to one of the categories specified by a set of criteria. These categories have to be defined a priori.

The definition of a category is based on the idea that all possible actions attributed to it will be treated in the same way. To determine the category to which it seems appropriate to allocate PMs, each action is assumed to be independent of the others [29]. We can assume that each controller of PM is a Decision Maker (DM), for this several DMs and components are involved in the decision-making process.

The research work carried out in [38] revealed, in the case of sorting problems, that neural networks can provide an efficient preference modeling mechanism for sorting problems. However, to be more efficient, we use the DNN with the ELECTRE TRI sorting algorithm, which is the most commonly used method based on the outranking relation approach. Building an outranking relation is based on two main concepts [29]:

(1) Concordance: a concordance index c (a, b) between two alternatives a and b is defined as the weight of the sum of the criteria for which "a is at least as strong as b".
(2) Non-discordance: Discordance indices d_j (a, b) are calculated on the criterion for which there is the most "opposite effect", that is to say for which a is better than b with a maximum relative deviation.

Thus, ELECTRE incorporates the relation of concordance and non-discordance to define the outranking relation.

The main advantages of this algorithm are:

– The proposal of a certain number of criteria that correspond to the constraints related to the purpose of this algorithm (the main criteria of PMs),
– Finding overloadEC PMs to prevent the SLA violation,
– Finding underloadEC PMs for possible placement for assigning or migrating VMs,
– Finding moderate PMs to avoid SLA violations.

(3) HLC4-Algorithm

This algorithm proposes to choose VMs that can be migrated and where to migrate them to save energy.

Algorithm 4: Detects the categories of each PM

Input: A set of criteria (C1, C2, C3, C4) where: C1 is the Usage of PM, C2 is the Consumption of energy, C3 is the Cost, C4 is the mode
We assume: The weights, Actions, Performances
Threshold = (U, M, O) for (UnderloadEC, Moderate loadEC, OverloadEC)
Output: assign each PM to the appropriate class = ['UnderloadEC', 'Moderate loadEC', 'OverloadEC']

```
Function Concordance (a, cl)
conc = 0
For each criterion in Criteria do
weights = weights[criteria]
Ua = Performances[a][criteria]
Ucl, Mcl, Ocl = threshold[cl][criteria]
if Ucl – Ua >= Ocl then
conc ← conc + 1* weights
else if Ucl – Ua <= Mcl then
conc ← conc + 0* weights
else conc ← conc + (((Ocl + Ua) − Ucl)/(Ocl − Mcl)) * weights
return conc
Function Discordance (a, cl, criteria)
Ua = Performances[a][criteria]
Ucl, Mcl, Ocl, vcl = threshold[cl][criteria]
If Ua <= Ucl – vcl then
return 1
else if Ua >= Ucl + Ocl then
return 0
else
return ((Ucl + Ocl) − Ua)/(Ocl + vcl)
```

VMs migration has great benefits, such as load balancing and server consolidation [39]. Nevertheless, in Cloud Computing environments, the cost of VMs migration needs careful study. However, we do not address this issue in this paper.

After finding the underloadEC PMs, this algorithm will select the VMs for migration. Furthermore, dynamic VM consolidation is related to the percentage of utilization of this class (cf. Algorithm 5).

Algorithm 5: VMs selection for migration for underloadEC class

Input: The underloadEC class.
Output: Adequate placement for VMs migration

```
MinU = 7% ((we assume MinU equal 7%)
For each PM in UnderLoadEC class Do
    If UPM_CPU^underloadEC <= MinU Then
        vmToMigrate ← Choose VM with Random Choice of this PM
For each vmToMigrate assigned to PM Do
NewPlacement ← Algorithm 2 (vmToMigrate)
    If utilization of (NewPlacement) > MinU Then
    Assign vmToMigrate to the NewPlacement
    Else Choose the next ranking "next placement" of Algorithm 2 (vmToMigrate)
end for
Put this PM in sleep mode
end for
```

The main advantage of this algorithm is finding the best PMs for VMs migration. This latter can reduce the energy consumption in the active PMs since it migrates VMs from PMs having resource utilization less than 7% to PMs having resource utilization greater than 7% in the underloadEC PMs class.

(4) HLC5-Algorithm

HLC5-Algorithm (cf. Algorithm 6) is the process of assigning each VM to a suitable PM. For this purpose, we propose a dynamic placement method that we call DPlacementKNN-ELECTRE.

Algorithm 6: DPlacementKNN-ELECTRE

Input: VM, list of PMs in UnderloadEC
Output: Best Placement of VM
1. BestPlacement ← null
2. Placement ← Algorithm 2 (VM)
3. **If** (Placement has enough resources for VM) **then**
4. BestPlacement ← Placement
5. **Else** BestPlacement ← Use the SoftMax function in the output layer to get the highest score of all classes of PMs from UnderloadEC class
6. **If** BestPlacement<>NULL **then**
7. DPlacementKNN-ELECTRE.add (VM, BestPlacement)
8. **Return** DPlacementKNN-ELECTRE

Algorithm 2 chooses the best placement if the selected PM has enough resources. Else, the HLC5 solicits the output layer, which applies the SoftMax function to represent the probability distribution over different classes (PMs in underloadEC).

5. Experiments and evaluations

In this section, we present an evaluation of the proposed work. First, we present some performance metrics, then we describe experimental settings. Finally, we present and discuss the experimental results.

5.1. Performance metrics

To evaluate the proposed model performance, we use multiple metrics which, are defined below:

- The number of active PMs (NPM_{active}): This metric represents the total number of needed PMs to allocate VMs.
- The number of VMs migration is estimated by executing Algorithm 2.
- Total energy consumption: The total energy consumption in the data center. We calculate it through Eq. (3).

$$EC_{total} = \sum_{i1}^{n} \quad EC_{PM_i} \tag{3}$$

5.2. Experimental setups

The difficulty of a Cloud computing environment becomes a challenge when carrying out a study on real infrastructure. Python language is used to overcome this obstacle. We use, also, Anaconda [40] which is a free Python distribution that provides necessary libraries for analyzing and comparing the performance of the proposed model.

Experimentation of the proposed model is performed in a personal computer with Intel Core i5 CPU, 3.33 GHz, 4 GB of RAM, and macOS. The simulation programs are written in Python3 to test the proposed model. Additionally, we use Scikit-Learn, Keras, TensorFlow libraries to implement the classification model. Besides these libraries, we also use some complementary Python libraries like NumPy, Pandas, and matplotlib.

Table 4
The confusion matrix

	P' (Predicted)	N' (Predicted)
P (Actual)	True Positive (TP)	False Negative (FN)
N (Actual)	False Positive (FP)	True Negative (TN)

5.3. Datasets and training

5.3.1. Datasets

In this research area, the amount of training data is limited for a variety of reasons: firstly, it is difficult to find data in a Cloud computing environment because data changes in a fast and dynamic way. Secondly, most papers in this area are based on simulation according to the nature of the environment and rarely use a large dataset.

To implement our proposal and due to the lack of datasets that are related to our context, we have collected our data to achieve our objectives and to improve the learning algorithms. We need two datasets to assess our experiment results.

(1) Dataset 1

Dataset 1 is divided into a training set and a testing set. There is a total of 347 instances (VMs), 70% of them are used for training and 30% are used for testing.

(2) Dataset 2

To have experimental results, we will adapt an existing dataset to our context. The latter is an individual household electric power consumption dataset [41]. This dataset includes 2075259 measurements collected between December 2006 and November 2010 (47 months) in a house located in Sceaux (7 km from Paris, France). We assume that the house is the data center and the measurements are taken of electric power consumption in this house are the energy consumption of the data center.

5.3.2. Training

For training, we use transfer learning as a type of machine learning technique. We use this technique because it saves time and avoids starting the process from zero [42]. Hence, it is very useful in deep learning.

5.4. Results and discussion

The problem of resource allocation is a multi-objective optimization problem. It aims to improve energy efficiency, resource utilization, load balancing, and server consolidation, etc.

(1) Scenario 1: Efficiency of the Algorithm 2

Since Algorithm 2 is used at several levels in our work, we need to test its efficiency using Scikit learn library of Python. This library is very fast to implement by calculating the accuracy, precision, F1, and recall same sensitivity measures. The results of this algorithm are shown in Fig. 5.

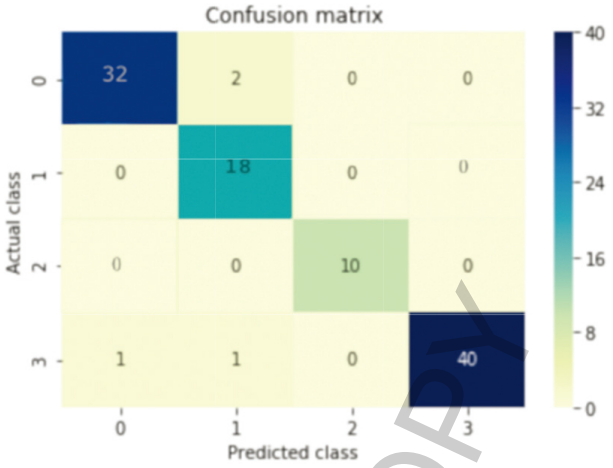
(a) Algorithm 2 evaluation by using the confusion matrix

The confusion matrix is the basic tool to evaluate models and understand how they work in a good way (cf. Table 4). The confusion matrix in Table 4 is used to illustrate Algorithm 2 in Table 5.

The measures to evaluate the algorithm were proposed in [43]:

$$\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{FP} + \text{FN} + \text{TN})$$

Table 5
The confusion matrix of Algorithm 2



k = 4 begin

	precision	recall	f1-score	support
0	0.97	0.94	0.95	34
1	0.86	1	0.92	18
2	1	1	1	10
3	1	0.95	0.98	42
accuracy			0.96	104
macro avg	0.96	0.97	0.96	104
weighted avg	0.96	0.96	0.96	104

Fig. 5. Accuracy, precision, recall same Sensitivity and F1 of Algorithm 2.

Fig. 6. Effect of the value of K on the accuracy of Algorithm 2.

Precision = TP/(TP + FP)
Recall same Sensitivity = TP/(TP + FN)
F1 = 2TP/(2TP + FP + FN)

(b) Relation between K and Algorithm 2 accuracy

To test the relation between the value of K and the accuracy of Algorithm 2, we use Python and dataset1. As it is illustrated in Fig. 6, we note that if the value of K is equal or close to the number of classes or PM labels, the accuracy will be improved.

Table 6
Measures used in our model

Layers	Number of epochs	Activation function	Loss function	Optimizer	Hidden units
4	100	Relu, SoftMax	MSE	SGD	512,256,128,32
4	500	Relu, SoftMax	MSE	SGD	2024,512,128,32

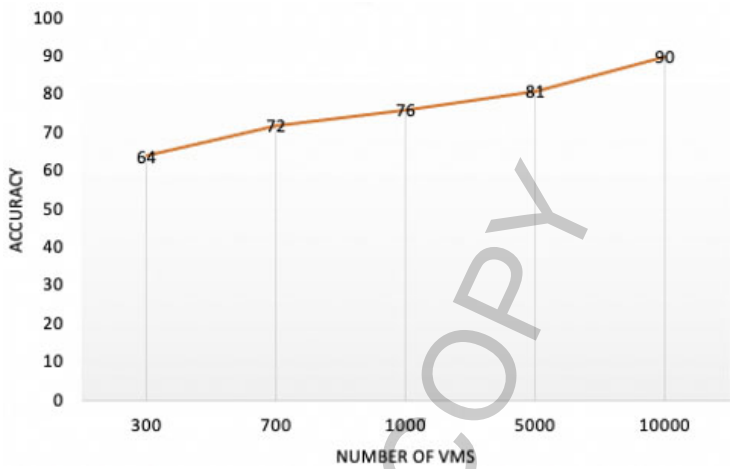


Fig. 7. The accuracy of our model.

(2) Scenario 2: The proposed model efficiency

We have tried to achieve the best accuracy by improving the appropriateness of the allocation. The deep learning model is often related to the amount of obtained training data. However, we can improve learning in the beginning by checking gradually instances from 300 to 10000 taken from dataset 2. We notice that we can reach 90% of accuracy by using the library of TensorFlow and Keras (cf. Fig. 7). Besides, we take into consideration some measures as shown in Table 6. We use SoftMax and Relu as activation functions, the Mean Squared Error (MSE) as loss function. Furthermore, a good choice for the optimization algorithm for a feedforward network is usually Stochastic Gradient Descent (SGD) with momentum.

(3) Scenario 3: Impact of dataset increments to improve the learning and accuracy of the model

To get the impact of an increase in the dataset to improve learning and accuracy, we vary the number of instances with increasing dataset 2 from 5000 to 100000. We note that the accuracy of our model increases from 81% to 97%. In this scenario (cf. Fig. 8), we compare the proposed model MDNN with the model that uses the simple DNN. We conclude that the proposed model improves our learning compared to the simple DNN mod.

(4) Scenario 4: Impact of our model on the number of VMs migration

In this scenario (cf. Fig. 9), the number of VMs migration is estimated by executing Algorithm 2. We note that the number of VMs migration is reduced according to the number of PMs that need to be considered while searching for the best placement for VMs.

5.5. Comparative results

To show the efficiency of our model, we present the comparative summary of the total energy consumption and the number of active PMs of the proposed model MDNN and three techniques: MADRL-DRA

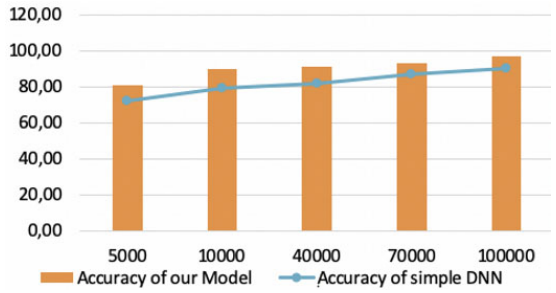


Fig. 8. Comparison between our model and simple DNN model.

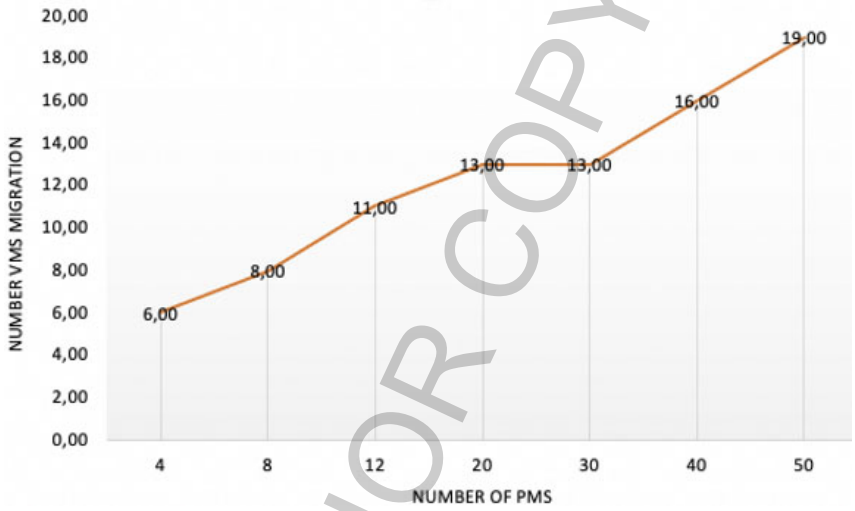


Fig. 9. Impact of our model on the number of VMs migration.

approach, MBFD as a meta-heuristic method, and DVFS as a material technique. These techniques were used to minimize energy consumption. We use these techniques for comparison with the proposed MDDN model by varying the number of VMs from 100 to 1000 (cf. Fig. 10). We also measure the number of active PMs in the data center using the proposed model MDNN and the existing algorithms, by varying the number of VMs from 100 to 1000 as shown in Fig. 11.

As shown in Figs 10 and 11, our model has optimal total energy consumption with minimizing the number of active PMs. This performance is achieved because our model uses a consolidation and migration technique that performs the algorithms of artificial intelligence and ELECTRE methods on the active number of PMs and underloadEC PMs.

In our model, we have demonstrated the impact of using DL by including ML and ELECTRE methods for the prediction of resource allocation in Cloud computing systems, by carefully dealing with the challenges in this environment, such as energy consumption, the number of active PMs, and resource utilization (cf. Figs 10 and 11). In addition, we have described how we can improve learning techniques for making resource allocation decisions for the cloud and the datacenters. In future, we can deal with DL models supporting the KNN and ELECTRE methods enabling better decisions anywhere in computer systems. However, there are several limits that we will investigate in the future, such as migration costs and fault tolerance, as we build the model.

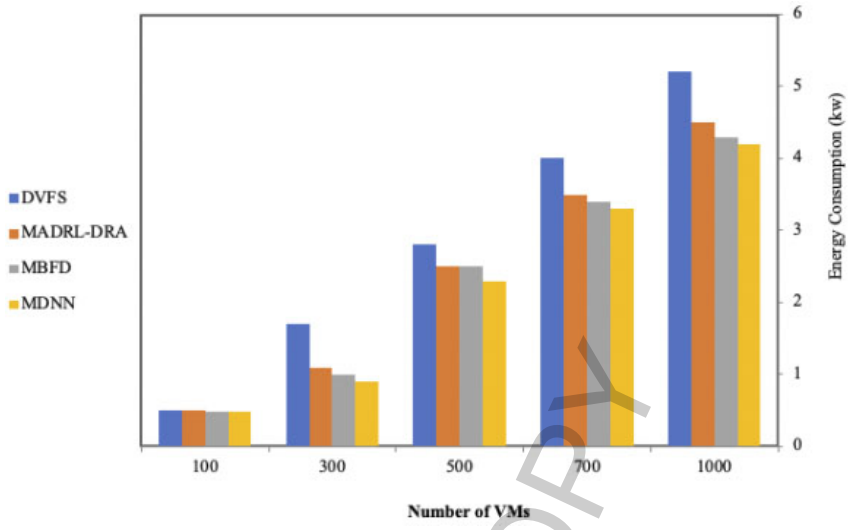


Fig. 10. The energy consumption in the data center.

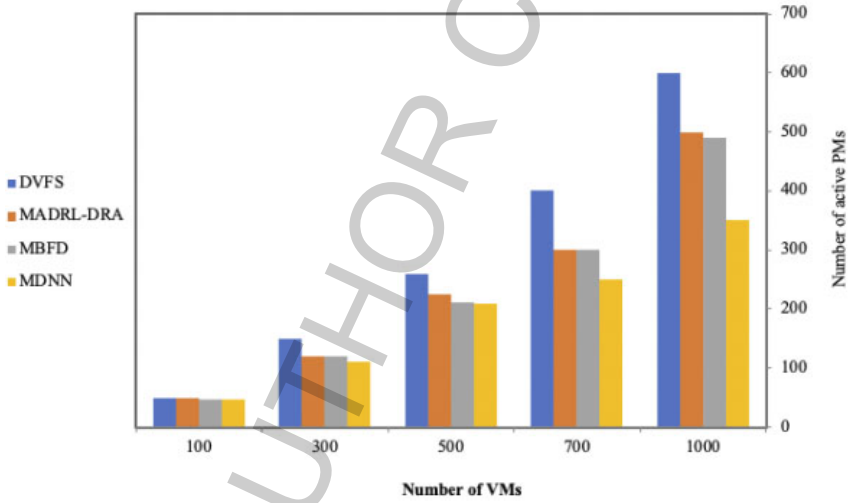


Fig. 11. The number of active PMs in the data center.

6. Conclusion and future works

In this paper, we have proposed an approach to predict the best placement of tasks and VMs using multiple criteria. It combines two DNNs by supporting KNN.

We have detailed our model MDNN in the VM-PM process with the experimental results taking advantage of and defining the weaknesses of our proposal. Thus, our model can effectively reduce energy consumption. Moreover, it can minimize the number of VMs, VMs migration, and allows the exploration of large-scale problems. To evaluate our work, some experiments have been performed using the Python language.

In the future, we aim to improve the proposed model by taking into consideration fault tolerance, migration cost minimization problems, and the association among VMs which can be flatted or hierarchi-

cal. Additionally, we will consider the fact of learning and not only predicting in real-time; this will be another open challenge to consider. Finally, the Task-VM Process generates an intermediate solution, which is simultaneously fed into the VM-PM process, but its training is independent of the first one, as demonstrated in this paper. Therefore, we will focus, in future work, on implementing the Task-VM process separately. Then, we will integrate the two processes (Task-VM and VM-PM) into a two-tier process (Task-VM-PM process) to determine the interdependence among all our work.

References

[1] F.A. Omara, S.M. Khattab and R. Sahal, Optimum resource allocation of database in cloud computing, *Egyptian Informatics Journal* **15**(1) (2014), 1–12.
[2] A. Beloglazov, J. Abawajy and R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, *Future Generation Computer Systems* **28**(5) (2012), 755–768.
[3] M.B. Gawali and S.K. Shinde, Task scheduling and resource allocation in cloud computing using a heuristic approach, *J Cloud Comp* **7**(1) (2018), 1–16.
[4] R.G. Shooli and M.M. Javidi, Using gravitational search algorithm enhanced by fuzzy for resource allocation in cloud computing environments, *SN Appl. Sci.* **2**(2) (2020), 1–13.
[5] M.-H. Malekloo, N. Kara and M. El Barachi, An energy efficient and SLA compliant approach for resource allocation and consolidation in cloud computing environments, *Sustainable Computing: Informatics and Systems* **17** (2018), 9–24.
[6] W. Wang, Y. Jiang and W. Wu, Multiagent-based resource allocation for energy minimization in cloud computing systems, *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **47**(2) (2017), 205–220.
[7] M.H. Mohamaddiah, S. Subramaniam, M. Hussin and A. Abdullah, A survey on resource allocation and monitoring in cloud computing, *International Journal of Machine Learning and Computing* **4** (2014), 31–38.
[8] A. Choudhary, S. Rana and K.J. Matahai, A critical analysis of energy efficient virtual machine placement techniques and its optimization in a cloud computing environment, *Procedia Computer Science* **78** (2016), 132–138.
[9] M. Tarahomi, M. Izadi and M. Ghobaei-Arani, An efficient power-aware VM allocation mechanism in cloud data centers: A micro genetic-based approach, *Cluster Computing* **24**(2) (2021), 919–934.
[10] K. Saidi, O. Hioual and A. Siam, Resources Allocation in Cloud Computing: A Survey, in: *Smart Energy Empowerment in Smart and Resilient Cities*, Springer, Bechar Algeria, 2019, pp. 356–364.
[11] D. Alboaneen, H. Tianfield, Y. Zhang and B. Pranggono, A metaheuristic method for joint task scheduling and virtual machine placement in cloud data centers, *Future Generation Computer Systems* **115** (2021), 201–212.
[12] P. Kumar, P.S. Yadav, K. Bhutani, N. Arora, D. Jain and B. Dabas, Allocating resource dynamically in cloud computing, in: *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*, Dec 2017, pp. 249–254.
[13] S. Kayalvili and M. Selvam, Hybrid SFLA-GA algorithm for an optimal resource allocation in cloud, *Cluster Computing* **22**(2) (2019), 3165–3173.
[14] P. Arroba, J.M. Moya, J.L. Ayala and R. Buyya, Dynamic Voltage and Frequency Scaling-aware dynamic consolidation of virtual machines for energy efficient cloud data centers, *Concurrency and Computation: Practice and Experience* **29**(10) (2017), e4067.
[15] G. Keller, M. Tighe, H. Lutfiyya and M. Bauer, An analysis of first fit heuristics for the virtual machine relocation problem, in: *8th International Conference on Network and Service Management (Cnsm) and 2012 Workshop on Systems Virtualiztion Management (Svm)*, Oct 2012, pp. 406–413.
[16] A. Varasteh and M. Goudarzi, Server consolidation techniques in virtualized data centers: A survey, *IEEE Systems Journal* **11**(2) (2017), 772–783.
[17] M.K. Gupta and T. Amgoth, Resource-aware virtual machine placement algorithm for IaaS cloud, *J Supercomput* **74**(1) (2018), 122–140.
[18] S. Azizi, M. Zandsalimi and D. Li, An energy-efficient algorithm for virtual machine placement optimization in cloud data centers, *Cluster Computing* **23**(4) (2020), 3421–3434.
[19] J. Kumar, A.K. Singh and R. Buyya, Ensemble learning based predictive framework for virtual machine resource request prediction, *Neurocomputing* **397** (2020), 20–30.
[20] M. Dabbagh, B. Hamdaoui, M. Guizani and A. Rayes, Energy-efficient resource allocation and provisioning framework for cloud data centers, *IEEE Transactions on Network and Service Management* **12**(3) (2015), 377–391.
[21] A. Tarafdar, M. Debnath, S. Khatua and R.K. Das, Energy and quality of service-aware virtual machine consolidation in a cloud data center, *Journal of Supercomputing* **76**(11) (2020).

[22] Y.S. Patel and R. Misra, Performance Comparison of Deep VM Workload Prediction Approaches for Cloud, in: *Progress in Computing, Analytics and Networking*, Singapore, 2018, pp. 149–160.
[23] J. Lin, Y. Dai, X. Chen and Y. Wu, Resource Allocation of Cloud Application Through Machine Learning: A Case Study, in: *2017 International Conference on Green Informatics (ICGI)*, August 2017, pp. 263–268.
[24] S. Ismaeel, R. Karim and A. Miri, Proactive dynamic virtual-machine consolidation for energy conservation in cloud data centres, *Journal of Cloud Computing* **7**(1) (2018), 1–28.
[25] Z. Zhou, Z. Hu and K. Li, Virtual machine placement algorithm for both energy-awareness and SLA violation reduction in cloud data centers, *Scientific Programming Hindawi* **2016** (2016).
[26] D. Alsadie, Z. Tari, E.J. Alzahrani and A.Y. Zomaya, Dynamic resource allocation for an energy efficient VM architecture for cloud computing, in: *Proceedings of the Australasian Computer Science Week Multiconference on – ACSW '18*, Brisband, Queensland, Australia, 2018, pp. 1–8.
[27] N. Liu et al., A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning, in: *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 372–382.
[28] A. Jyoti and M. Shrimali, Dynamic provisioning of resources based on load balancing and service broker policy in cloud computing, *Cluster Computing* **23**(1) (2019), 377–395.
[29] Salvatore Greco, Matthias Ehrgott, and José Rui Figueira, *Multiple Criteria Decision Analysis*. Springer, 2016.
[30] R. Khorsand and M. Ramezanpour, An energy-efficient task-scheduling algorithm based on a multi-criteria decision-making method in cloud computing, *International Journal of Communication Systems* **33**(9) (2020), e4379.
[31] K. Saidi, O. Hioual and A. Siam, A DSL-MCDA Model for Energy Consumption-Aware in Cloud Computing, in: *International Conference on Internet of Things, Embedded Systems and Communications (IINTEC)*, 2019, pp. 168–173.
[32] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016.
[33] A. Marahatta, S. Pirbhulal, F. Zhang, R.M. Parizi, K.-K.R. Choo and Z. Liu, Classification-based and Energy-Efficient Dynamic Task Scheduling Scheme for Virtualized Cloud Data Center, *IEEE Transactions on Cloud Computing*, 2019.
[34] L.-Y. Hu, M.-W. Huang, S.-W. Ke and C.-F. Tsai, The distance function effect on k-nearest neighbor classification for medical datasets, *Springer Plus* **5**(1) (2016), 1–9.
[35] C. Jin, X. Bai, C. Yang, W. Mao and X. Xu, A review of power consumption models of servers in data centers, *Applied Energy* **265** (2020), 114806.
[36] S.K. Mishra et al., Energy-efficient VM-placement in cloud data center, *Sustainable Computing: Informatics and Systems* **20** (2018), 48–55.
[37] C.-H. Hsu, K.D. Slagter, S.-C. Chen and Y.-C. Chung, Optimizing energy consumption with task consolidation in clouds, *Information Sciences* **258** (2014), 452–462.
[38] C. Zopounidis and M. Doumpos, Multicriteria classification and sorting methods: A literature review, *European Journal of Operational Research* (2002), 229–246.
[39] M. Hosseini Shirvani, A.M. Rahmani and A. Sahafi, A survey study on virtual machine migration and server consolidation techniques in DVFS-enabled cloud datacenter: Taxonomy and challenges, *Journal of King Saud University – Computer and Information Sciences* **32**(3) (2020), 267–286.
[40] https://www.anaconda.com.
[41] Georges Hebrail and Alice Berard, UCI Machine Learning Repository: Individual household electric power consumption Data Set, 2012.
[42] A.S. Qureshi, A. Khan, A. Zameer and A. Usman, Wind power prediction using deep neural network based meta regression and transfer learning, *Applied Soft Computing* **58** (2017), 742–755.
[43] J. Patterson and A. Gibson, Deep Learning: A Practitioner's Approach. O'Reilly Media, Inc., 2017.

Author's Bios

Karima SAIDI received the Engineer, and MS degrees in computer science from Abbes Laghror University, Khenchela, Algeria, in 2009 and 2013, respectively. She is currently a Ph.D. student in Computer science at Abbes Laghror University, Khenchela, Algeria. Her primary research interests include Cloud computing, resource allocation, Multi-Criteria Decision Analysis (MCDA), and artificial intelligence: machine learning and deep learning.

Ouassila HIOUAL received her BS and MS in Computer Science from the Mentouri University of Constantine, Algeria in 2002 and 2005. She has worked as a Lecturer at the Department of Computer Science and Mathematics Science at the Mentouri University of Constantine, Algeria from 2005 to 2008. Currently, she is working as an Associate Professor at the Department of Mathematics and Computer Science at Abbes Laghror University of Khenchela, Algeria since 2008. She supervised many Master and License students. Since September 2006 until October 2011, she has prepared his PhD in Computer Science. She has published a number of articles in international conferences and journals. Her research interests include semantic web, ontology, web services, multi agent system, data management and cloud computing.

Abderrahim SIAM received his BS degree in Computer Science from the University of Batna (Algeria) in 2002, and MS degree in Computer Science from the University of Oum El Bouaghi (Algeria) in 2005, and his Ph.D. in Computer Science from the University of Constantine, Algeria. He is working as Professor at the Department of Mathematics and Computer Science at the University of Khenchela (Algeria). He is currently the vice-rector of the University of Khenchela (Algeria). His research interests include software engineering, fuzzy logic, formal methods, multiagent systems, and complex systems.