Multimedia Tools and Applications
https://doi.org/10.1007/s11042-021-10616-6

Energy, network, and application-aware virtual machine placement model in SDN-enabled large scale cloud data centers

Soha Rawas[1]

Received: 12 March 2020 / Revised: 12 January 2021 / Accepted: 25 January 2021
Published online: 04 February 2021

Abstract
Cloud computing has been considered a core model of elastic on-demand resource allocation using a pay-as-you go model. One of the big challenges of this environment is to provide high quality service (QoS) through efficient and stringent management of cloud data center resources. With the increasing demand for cloud based services, the traffic volume inside cloud data centers (DC) has been increased exponentially. Accordingly, and to provide high QoS, a proper scheduling mechanism has to be followed by the cloud service provider. Furthermore, accurate scheduling is necessary for advancing the problem of energy consumption and resource utilization. In this paper, we propose an optimal resource allocation and consolidation virtual machine (VM) placement model for multi-tier applications in modern large cloud DCs. The proposed model targets to optimize the DCs' energy and communication cost that influence the overall cloud performance through Software Defined Networking (SDN) control features. To solve the formulated multi-objective optimization problem, a novel adaptive genetic algorithm is proposed. The experimental results validate the efficacy of the proposed model through extensive simulations using synthetic and real workload traces. These results show that the proposed model jointly optimizes cloud QoS as well as energy consumption.

Keywords Virtualization · Cloud computing · Multi-tier application · Green computing · Knapsack problem · Genetic algorithm

1 Introduction

Cloud computing, which is a model of delivering computing resources on-demand using a pay-as-you-go model, shapes the way that information technology (IT) resources are designed

✉ Soha Rawas
Soha.rawas2@bau.edu.lb

[1] Department of Mathematics and Computer Science, Beirut Arab University, Beirut, Lebanon

and purchased [14]. Therefore, it has become very popular in recent years so that organizations and cloud customers can rent their computing resources instead of buying them. However, the rapid evolution of cloud computing services has led to the adoption of a large-scale cloud data centers (DCs) to meet the user's requirements. Accordingly, DCs with thousands of computing and storage nodes, contribute to a vast amount of energy leading to high-cost rates and carbon dioxide (CO2) emissions to the environment [12]. Thus, the efficiency and managing of such DCs have become a prominent problem to insure cloud sustainability.

Studies show that [5, 14] servers and storage contribute to 28% of the total DC energy consumption, while network devices contribute to 20% (as shown in Fig. 1). Although the cooling system is the most causative of energy consumption, however, this problem needs mechanical and thermos technical solutions which are out of the scope of this paper. Therefore, our focus in this paper is on servers, storage, and network hardware that are the main causative of energy consumption. So how can we drag these factors to be energy efficient? It is obviously through energy management and resource utilization. Studies show that an activated server consumes 70% of its power even if idle and 80% of DCs activated servers have less than 50% of their CPU utilization. Therefore, the solution is to drag the active DC devices to be power proportional and to set the unused devices (servers, switches) to sleep mode (only 300 milliseconds to start-up) through energy and network-aware virtual machine (VM) placement models that enhance entire DC resource utilization.

Moreover, high energy consumption means high carbon emission, which drastically leads to the greenhouse effect [18]. Other recent studies indicate that cloud DCs contribute to 3.5% of the world's CO2 emission by 2018 and predicted to be 14% by 2040 [22]. Therefore, energy-efficient solutions are needed to ensure cloud sustainability. Greenpeace's recent study shows that online people are currently around 2.5 billion [3]. However, they expect this number to increase by 60% in the next 5 years. Thus, managing Internet server systems is an essential concern.

Although several technologies are applied for the sake of green cloud computing [14, 18], virtualization and consolidation techniques remain the key contributor's concept for power improvement. Virtualization is a technology that abstracts physical resources for utilization improvement [14]. Since cloud computing has a concept of renting resources instead of owing it, virtualization has become a crucial demand for this environment. VM scheduling plays a vital role in cloud resource utilization. However, two crucial facts make this concept not an easy task: 1- heterogeneity in DCs computing resources, and 2- VM resource requirements variations (such as CPU utilization, storage, bandwidth and memory). Moreover, cloud-hosted

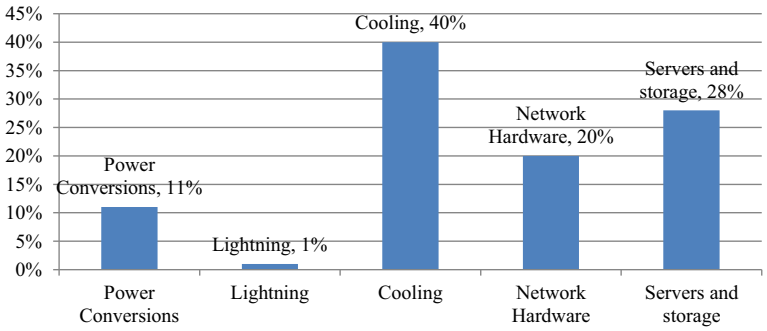

Fig. 1 DC energy consumption

application type and dependencies among VMs are important factors that should be considered in the VM scheduling criterion [11].

Moreover, the evolution of cloud computing has given rise to new technologies known as Software Defined Networking (SDN). This new technology that controls the internal cloud data centers network elements plays an important role in optimizing the network resource management and communication cost that influence the overall cloud performance [21]. Nowadays, the joint optimization of the system, as well as network status to perform efficient VM placement, is an attractive problem in SDN-based cloud DCs. SDN technology can identify the network topology in real-time. Therefore, this technology eases the process of route configuration and delivers network load to less-used nodes and less busy paths which lead to efficient resource utilization (servers and network devices) as well as DC management.

To overcome the resource management challenges, this paper proposes and evaluates a novel approach in the field of resource management in SDN based cloud DCs named **E**nergy, **N**etwork, and **A**pplication-aware **V**M placement model (ENAV). ENAV employs a cost-aware adaptive genetic algorithm for multi-tier applications to minimize the cloud provider cost and optimize intra-DC network performance behavior. The main contributions of this paper are as follows:

1. Develop a novel VM placement model that considers energy cost, VMs cost and the communication cost through considering communication dependencies between VMs of multi-tier application and the energy consumption of the hosted servers as well as the network devices.
2. Design and implement a novel adaptive genetic algorithm ENAV-G for cost-aware VM placement to solve the proposed multiple knapsack optimization problem.
3. Conducting extensive simulations to validate the proposed model using both synthetic and real DC workload data.

The rest of the paper is organized as follows: Section 2 describes the related work; Section 3 illustrates the model problem and its formulation; Section 4 proposes and develops the solution for the proposed ENAV model; Section 5 evaluates the proposed model; and Section 6 concludes the paper and introduces future work.

2 Related work

VM scheduling in large scale cloud DCs have received more attention in the recent years. According to the literature [4, 6, 12, 13, 23], there are a number of approaches proposed to solve this problem.

In [4], the author proposed a task scheduling model to improve cloud-based web applications cost through controlling the cloud application feature set based on the Service Level Agreement (SLA). The proposed model achieved its targets by improving the developed cloud application availability and minimizing its running cost. Dias et al. [6] proposed a bin-packing scheduling algorithm to reduce traffic communication and traffic cost. The proposed algorithm targets to group the highly communicated VMs into clusters then mapping these clusters to several servers. Wei et al. [23], proposed an energy-efficient VM placement method to reduce both the communication cost as well as the DC power consumption. The author used an improved ant colony mechanism to solve the proposed optimization problem using an adaptive

parameter setting to introduce a robust and fast convergence algorithm. Rawas et al. [15] proposed ENAGS, energy, and network-aware genetic scheduling algorithm that takes into account the servers' energy consumption and communication dependency cost. In [2], Cao et al. proposed a consolidation VM placement approach that is a communication-aware using a modified SCAN algorithm to lower the computation cost and minimize inter-rack traffic though they do not take into account the energy cost in placing VMs. However, in his latest work [1], Cao et al. proposed a new VM dynamic consolidation approach to optimize both the energy consumption and communication traffic in cloud DC. The proposed approach used a multi-objective genetic algorithm (GA) to attain the model objectives. Zhao et al. [25] deployed the energy-aware genetic and tabu (GATA) algorithm. GATA was able to balance the DC load among the available resources through finding the best placement machine that leads to energy minimization. Son et al. [8], developed a dynamic overbooking strategy which jointly combines compute and network optimization for VM and traffic consolidation. The proposed strategy dynamically adapts overbooking ratio depending on real-time workload for both VM and network flows. In [26], a workload-aware revenue maximization (WARM) approach has been proposed to maximize the cloud provider net revenue in the context of SDN-enabled DCs. This approach combined chaotic search, simulated annealing (SA), and particle swarm optimization (PSO) algorithms, to determine the optimal combination of a VM and routing path for each application.

In contrast to the aforementioned, the proposed ENAV model formulates the VM placement problem in terms of DC energy consumption cost (including servers, storage, as well as switches nodes), VMs communication cost (considering the intra-DC network bandwidth), as well as the VMs cost. Moreover, in this paper, we designed and implemented an adaptive novel ENAV-G genetic algorithm that uses an adaptive genotype to robust the whole genetic algorithm and achieve the ENAV model objectives. Moreover, the proposed ENAV-G algorithm helps in solving the multi-objective optimization problem without contradiction between each sub-objective.

3 Models and problem formulation

3.1 Problem statement

Cloud computing services increased in popularity due to the wide variety of services and application types that have been developed. However, deploying a composite application in cloud DCs such as multi-tier applications needs efficient assignments of VMs to some hosts. This type of application is divided into two or more components and the number of tiers varies by application requirements. Cloud customers may request such an application type. On the other hand, they may request a VM cluster of multiple computing virtual devices to deploy a composite application and form their virtual private cloud. The main problem in such scenarios is that these application types (multi-tier applications) or requested VM clusters have specific traffic flow and need to exchange data in between tiers/VMs. Therefore, the performance of these applications is influenced by the communication latency generated by the communicated data and computing components. Consequently, appropriate placement of VMs into available hosts is an important and well-known NP-complete problem. However, the main aim of this paper is to optimize the total provider cost in such an environment without scarifying cloud quality of service (QoS) as its primary objective. This paper targets to answer the following questions:

- How to design and implement an online VM consolidation model on a large scale and highly dynamic cloud DC.
- How to successfully coordinate communication, energy, and VMs cost without scarifying SLA constraints and cloud customers' satisfaction.
- How to optimize the VM placement and consolidation problem that includes multiple objectives for all type of DC's network topology.

3.2 System model and architecture

Figure 2 generally presents the ENAV system model architecture. Cloud users send their application deployment requests to the cloud provider. The ENAV controller analyzes the cloud user request to find the best VM selection type in terms of computing specifications and cost. Since the deployed applications are of multi-tier or workflow type, the ENAV controller cluster VMs that perform a cloud users request to form a virtual cloud for each. Then, the ENAV controller directs the clustered virtual clouds to the best placement method through the ENAV Host Selection Manager module. A physical cloud environment is made up of several nodes. These nodes are of two forms: 1- Computing nodes servers denoted as Se, 2- and Storage nodes denoted as St. Accordingly, each cluster of the user request (VMs) deployed over the cloud physical nodes. A physical node may host multiple VMs from different clusters. Noting that VM cluster for certain users' requests may be served on different nodes. However, the ultimate objective of the proposed ENAV model is to provide cloud users with ideal deployed application performance while saving cloud energy consumption and provider operational cost.

3.3 VM placement in cloud DC

The VM placement problem is defined as the number of possible mapping of VMs to available physical machines (PM). Our goal is to minimize the total cost that includes energy cost,

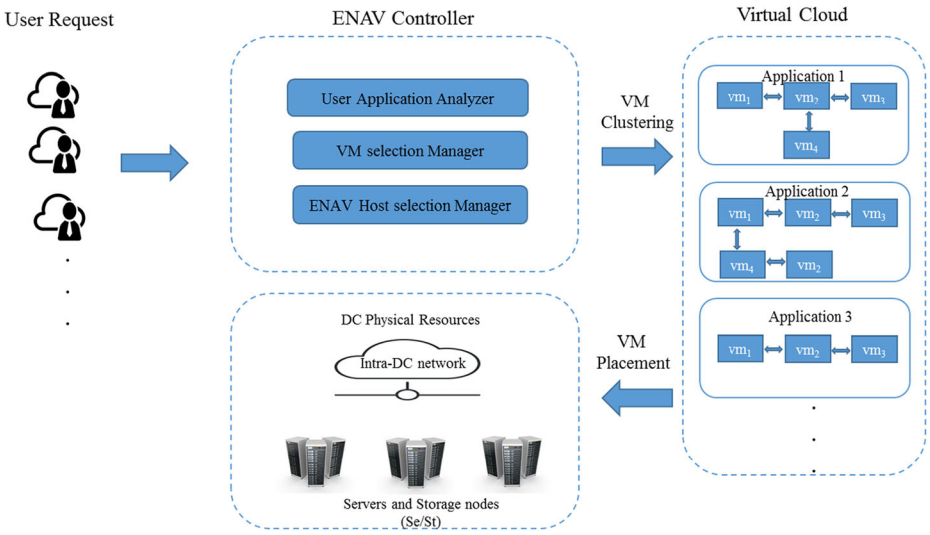

Fig. 2 Architecture design of ENAV model

communication cost, and VM cluster cost. Noting that minimizing the communication cost aims to minimize the volume of traffic between the communicated VMs that lead to intra-DC network optimization. This could be attained through an optimal mapping of VMs to PMs such as the following are achieved:

1- Minimize servers' energy by reducing the number of active servers and consolidating VMs to the minimum number of PMs.
2- Minimize switches' energy by reducing the number of active switches
3- Minimize intra-DC network traffic cost by placing communicated VMs near each other and by considering the amount of available bandwidth
4- Minimize the VMs cost by considering the appropriate VMs to allocate the requested applications

Table 1 summarizes the various notations used in the VM scheduling problem.

3.4 Formal definition

Define an application as A = {V, D}, where V and D is the set of requested VMs and used data respectively to process the application A: $V = \{vm_i : 1 \le i \le n_v\}$, $D = \{d_i : 1 \le i \le n_d\}$. Define a DC as DC = {Se, St, N}, where Se is the set of servers that forms the computing nodes: $Se = \{se_i : 1 \le i \le n_{se}\}$, St is the set of storage nodes: $St = \{st_i : 1 \le i \le n_{st}\}$, and N is the physical network topology. Let N defined as N = {Sw, L}, where Sw is the set of network devices in a DC: $Sw = \{sw_i : 1 \le i \le n_{sw}\}$, and L is the set of links in an L: $L = \{l_i : 1 \le i \le n_l\}$.

Table 1 Notations and their semantics

Notation	Description
A	Application
V	Set of requested VMs
D	Set of requested data
N	The network topology in a DC
Se	Set of servers
St	Set of storage nodes
Sw	Set of network devices
L	Set of links
vm_i	A single VM
n_v	Total number of VMs in an A
d_i	A single data
n_d	Total number of data in an A
se_i	A single server
n_{se}	Total number of servers in a DC
st_i	A single storage node
n_{st}	Total number of storage nodes in a DC
sw_i	A single switch
n_{sw}	Total number of switches in an N
l_i	A single network link between two devices
n_l	Total number of links in an N
n_{No}	Total number of computing and storage nodes for in an A
n_e	Total number of virtual edges
t_i	vm_i active time slot [0, T]

Consider that the application A deployed and hosted in a DC that has a well-designed physical network N (as shown in Fig. 3).

Each computing node se_i has four different resource capacities: $se_i = \left(se_i^{core}, se_i^{ram}, se_i^{storage}, se_i^{bandwidth}\right)$, where the four-vector components represent the number of available cores, the amount of available random access memory (RAM), the available storage capacity, and the amount of available bandwidth. While each storage node st_i has storage and bandwidth capacity denoted as $st_i^{storage}$, $st_i^{bandwidth}$ respectively.

Each VM vm_i is to be mapped into a computing node se_j request four resource capacities represented by the vector $vm_{i,j} = \left(vm_{(i,j)}^{core}, vm_{(i,j)}^{ram}, vm_{(i,j)}^{storage}, vm_{(i,j)}^{bandwidth}\right)$, where the four-vector components represent the number of cores, RAM, storage capacity, and bandwidth respectively.

The DC Network N connects the network devices, computing, and storage nodes using dedicated links L (as shown in Fig. 2). Let the communication dependencies among a set of hosted VMs running multi-tier application denoted as LG = {No, E}, where *No* is the set of nodes of computing or storage type: $No = \{no_i : 1 \le i \le n_{No}\}$, while E is the set of logical or virtual edges among the communicated nodes: $E = \{e_i : 1 \le i \le n_e\}$. Figure 4 illustrates the VM nodes communication environment in a cloud DC. Noting that e_i indicates a logical/virtual edge with available bandwidth (BA) and network distance (DS) between two communicated nodes.

3.4.1 VM/server placement relationship

Each server se_j can host more than one VM and each VM is executed at only one se_j. Let A be m x n matrix showing the mapping status of the m VMs to the n servers as follows:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

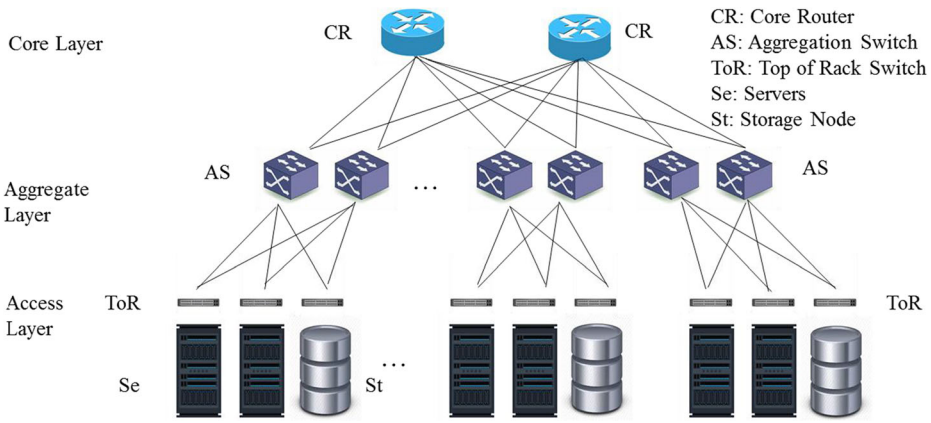

Fig. 3 DC 3-tier network architecture

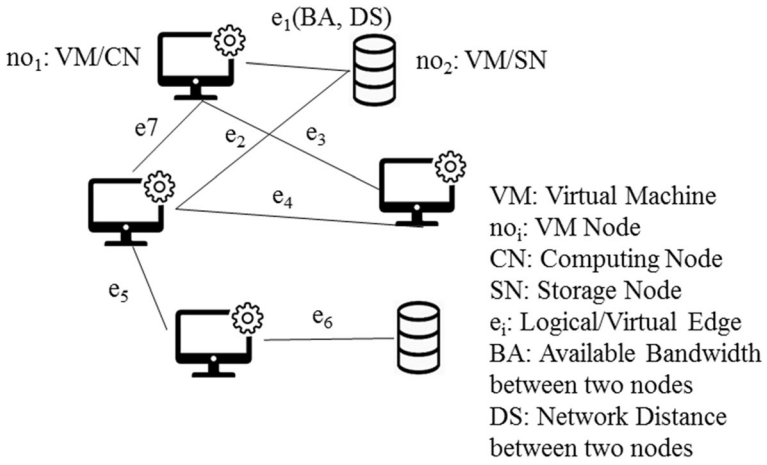

Fig. 4 VM nodes communication environment in cloud DC

where a_{ij} is a binary variable (0/1) such that:

$$a_{ij} = \begin{cases} 1, & if\ vm_i\ is\ allocated\ to\ se_j \\ 0, & otherwise \end{cases}$$

3.4.2 VM/VM dependency relationship

Let the communication dependency among a set of communicated VMs in an application A represented by m x m matrix B showing the dependencies between m VMs as follows:

$$B = \begin{bmatrix} b_{11} & \cdots & b_{1m} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mm} \end{bmatrix}$$

where b_{ij} is a binary variable (0/1) such that:

$$b_{ij} = \begin{cases} 1, & if\ vm_{i,k}\ and\ \ vm_{j,l}\ are\ dependent \\ 0, & otherwise \end{cases}$$

3.4.3 Intra-DC traffic cost (ITC)

Cloud providers can host different applications inside one DC. Therefore, different traffic types can be generated such as:

- Guest traffic: traffic that generated due to communication between multiple VMs belongs to the same tenant.
- Public traffic: traffic that generated due to and from the internet bound for VMs in DC.
- Storage traffic: traffic generated due to moving large chunks of data for running a specific type of heavy communicated applications such as Hadoop.

This traffic should be isolated from each other to prevent intra-DC network congestion and cloud provider service degradation. Therefore, cloud providers employ network virtualization techniques to improve intra-DC network performance. When customers send their application to a cloud environment, the cloud provider offers them a virtual topology that consists of VM types and virtual links between VMs. These virtual links are mapped to a set of physical links, while the VMs are mapped to Ses. Nevertheless, and due to security reasons, cloud providers never give any information about these virtual topologies [1].

Therefore, the intra-DC communication traffic cost is directly proportional to 1- the amount of transferred data between two dependent VMs (TD), 2- the available bandwidth and capacity between two dependent VMs (BA), 3- the network distance (DS) between dependent VMs that is measured using the number of hops (switches).

Transferred data (TD) Let the amount of transferred data between two dependent VMs $vm_{i,k}$ *and* $vm_{j,l}$ denoted as $TD(vm_{i,k}, vm_{j,l})$ such that vm_i *and* vm_j are either:

- two computing nodes hosted on se_k and se_l
- one computing node hosted on se_k and one storage node hosted on st_l

noting that $TD(vm_{i,k}, vm_{j,l}) = 0$

- if there is no dependency between $vm_{i,k}$ and $vm_{j,l}$
- if $vm_{i,k}$ and $vm_{j,l}$ hosted on the same rack as formulated using [15].

Available bandwidth (BA) the available bandwidth between two communicated VMs affects the total cost to transfer *TD* data between two computing nodes or computing and storage nodes. Let $BA(vm_{i,k}, vm_{j,l})$ be the available bandwidth between two dependent $vm_{i,k}$ *and* $vm_{j,l}$ such that vm_i *and* vm_j are either (as shown in Fig. 4):

- two computing nodes hosted on se_k and se_l
- one computing node hosted on se_k and one storage node hosted on st_l

Network distance (DS) the network distance between two dependent $vm_{i,k}$ and $vm_{j,l}$ is denoted by $DS(vm_{i,k}, vm_{j,l})$. *DS* designates the number of hops between $vm_{i,k}$ and $vm_{j,l}$ such that vm_i *and* vm_j are again either (as shown in Fig. 3):

- two computing nodes hosted on se_k and se_l
- one computing node hosted on se_k and one storage node hosted on st_l

noting that $DS(vm_{i,k}, vm_{j,l}) = 0$

- if there is no dependency between $vm_{i,k}$ and $vm_{j,l}$
- if $vm_{i,k}$ and $vm_{j,l}$ hosted on the same rack as formulated using [15].

Accordingly, the ITC for deploying and running an application A on a data center DC can be defined using the following equation:

$$ITC(A, DC) = \sum_{i,j=0}^{n_v} a_{i,j} * b_{i,j} * TD\left(vm_{i,k}, vm_{j,l}\right) * BA\left(vm_{i,k}, vm_{j,l}\right) * DS\left(vm_{i,k}, vm_{j,l}\right), \forall (k,l), \quad k, l \in n_{se/st} \quad (1)$$

3.4.4 DC energy consumption cost (DECC)

One of the important objectives of the proposed ENAV model is to optimize the energy consumption of the whole DC when deploying application A inside a cloud DC. This could be achieved through optimizing the number of active servers that holding the VMs processing application A on the minimum and optimum type of energy-efficient servers and storage devices. Consequently, DC IT devices (i.e. computing servers, storage, and network devices) contribute to a large amount of energy consumption as shown in Fig. 1. Thus, this paper aims to minimize the amount of energy consumption consumed by the aforementioned DC IT devices.

3.5 Power models

Computing servers Accountable for the largest amount of DC energy consumption. However, Dynamic voltage frequency scaling (DVFS) is an effective technique to control and adjust the se CPU power [16]. According to studies [14], an idle server releases up to 70% of its power consumption once it is turn on to keep I/O, memory, and disk resources active. However, the 30% remaining changed according to CPU utilization. Therefore, one of the ENAV model goals is to minimize the number of active servers so that it can drag the unused ones to sleep mode and consequently save up to 90% of their power consumption. Moreover, and to optimize the power consumption of the used servers, the ENAV model uses the quadratic relation between the frequency adjustment and the CPU dynamic power consumption as proposed by Huai et al. [5]. Consequently, the following equation is used to find the total power consumption of a server se:

$$P_{se} = P_{se-static} + P_{se-dynamic} \quad (2)$$

where

- $P_{Se-static}$ is the static/idle power consumption known as leakage and denoted as γ.
- $P_{se-dynamic}$ is the CPU dynamic power consumption determined using the quadratic relation between the CPU power consumption and the frequency adjustment as follow

$$P_{se-dynamic} = SA * C * SV^2 * \quad (3)$$

where *SA*, *C*, *SV*, *and f* are the switching activity, the physical capacitance, the supply voltage, and the clock frequency of the processor respectively.

Storage node Is a physical server with multiple storage nodes for the sake of data availability in case of any hardware failure. In such devices, disk storage device energy consumption dominates other main storage node (SN) devices such as CPU, memory, I/O, etc. However,

most of the current storage nodes platform does not allow a separate measuring of power consumption for each SN device [5] and consider the power model as follow:

$$P_{st} = P_{CPU} + P_{Memory} + P_{disk} + P_{IO} \tag{4}$$

such that P_{st} which is the power consumed by a storage node *st,* expressed as the total power consumption of *st* CPU, memory, disk and, I/O devices.

Network devices Are the second largest power consumer in IT devices that includes the installed switches that form the key enabling component of the network inside cloud DC. Based on the benchmarking suite of network devices [5], the power consumption of a switch in a DC network is defined as:

$$P_{sw} = P_{Chassis} + n_{line\ cards} * P_{line\ cards} + \sum_{i=0}^{configs} \left(P_{configs\ i} * Nports_{configs\ i} * Uf_i \right) \tag{5}$$

where $P_{Chassis}$ is the power consumed by the chassis-based hardware, $n_{line\ cards}$ is the number of line cards plugged into the switch, $P_{line\ cards}$ is the power consumed by the switch line cards with no ports turned on, *configs* is the number of configuration for the port line rate, $P_{configs\ i}$ is the power for a port running at line rate i (such that i = 1 (10Mbps (megabits per seconds)), i = 2 (100Mbps), i = 3 (1Gbps)), $Nports_{configs\ i}$ is the number of ports using configuration type i, and Uf_i is the scaling factor to account for the utilization of each port.

3.6 Modeling the DC energy consumption cost

Using the above-defined Power models (P_{se}, P_{st}, *and* P_{sw}) we can model the DECC as a demand of used servers, storages, and switches such that

$$DECC = EC_{se} + EC_{st} + EC_{sw} \tag{6}$$

such that:

– EC_{se} is the server energy consumption that can be modeled using the P_{se} quadratic power model described in the above section. Consequently, EC_{se} for deploying and running application A inside cloud DC is denoted as:

$$EC_{se}(A, DC) = \sum_{i=1}^{n_v} P_{se}(j) * t_i \quad \forall j, j \in n_{se} \tag{7}$$

where $P_{se}(j)$ is the power consumption of a server j holding number of VMs during the slot time [0, T].

– EC_{st} is the storage node energy consumption that can be modeled using the P_{st} power model (as shown in the above section). Consequently, EC_{st} for holding application A data inside cloud DC is denoted as:

$$EC_{st}(A, DC) = \sum_{i=1}^{n_v} P_{st}(j) * t_i \quad \forall j, j \in n_{st} \tag{8}$$

where $P_{st}(j)$ is the power consumption of a storage node j holding number of VMs during the slot time [0, T].

– EC_{sw} is the switches energy consumption that can be modeled using the P_{sw} power model (as shown in the above section). Consequently, EC_{sw} for s number of active switches to deploy and run application A inside cloud DC is denoted as:

$$EC_{sw}(A, DC) = \sum_{i=1}^{s} P_{sw}(i) * t_i \tag{9}$$

where $P_{sw}(i)$ is the power consumption of i active switch during the time slot [0, T].

3.6.1 VM placement cost (VMPC)

One of the core objectives of the proposed ENAV model is to optimize the VMPC through proper selection type of available VMs to host an application A in cloud DC. Accordingly, the cost of assigning application A in cloud DC to the number of v VMs is formulated as follows:

$$VMPC(A, DC) = \sum_{i=1}^{n_v} a_{i,j} * Cost\left(vm_{i,j}\right) \quad \forall j, j \in n_{se}/n_{st} \tag{10}$$

where $Cost(vm_i, se_j)$ is the cost of assigning one of the applications A tasks to VM vm_i hosted on server/storage node se_j/st_j.

3.7 Modeling the optimization problem

The objective of the ENAV model is to optimize the total placement cost of the deployed application A inside of a cloud DC which is a demand for Intra-DC traffic cost (ITC), DC energy cost (DECC), and VM placement cost (VMPC). Consequently, ENAV objective function f is to minimize the overall cost as the following:

$$f(A, DC) = minimize \begin{pmatrix} ITC \\ DECC \\ VMPC \end{pmatrix} \tag{11}$$

subject to

1- VM Placement constraint: denotes the VM/Server relationship. It mandates that each VM executed and assigned to only one server/storage node such that:

$$\sum_{i=1}^{n_v} a_{ij} = 1 \quad \forall j, j \in n_{se}/n_{st} \tag{12}$$

2- Capacity constraints: signifies that the VMs capacity requirements cannot exceed the hosted physical server/storage nodes total capacity (i.e. Core, RAM, storage, and bandwidth) as follow:

$$\sum_{j=1}^{n_{se}} \sum_{i=1}^{n_v} a_{ij} * vm_{(i,j)}^{core} \leq se_{(i,j)}^{core} \tag{13}$$

$$\sum_{j=1}^{n_{se}} \sum_{i=1}^{n_v} a_{ij} * vm_{(i,j)}^{RAM} \leq se_{(i,j)}^{RAM} \tag{14}$$

$$\sum_{j=1}^{n_{se}/n_{st}} \sum_{i=1}^{n_v} a_{ij} * vm_{(i,j)}^{storage} \leq se_{(i,j)}^{storage} / st_{(i,j)}^{storage} \tag{15}$$

$$\sum_{j=1}^{n_{se}/n_{st}} \sum_{i=1}^{n_v} a_{ij} * vm_{(i,j)}^{bandwidth} \leq se_{(i,j)}^{bandwidth} / st_{(i,j)}^{bandwidth} \tag{16}$$

$$a_{ij}, b_{ij} \in \{0, 1\}$$

4 The ENAV methodology

ENAV model formulated as an optimization problem is a variant of multiple knapsacks, which is an NP-hard problem [7]. Accordingly, this paper proposes the ENAV-G algorithm after several strategic methods to find a near-optimum solution to the proposed problem. The main idea behind the proposed approach is to decompose the ENAV problem into two main sub problems. More specifically the ENAV model uses the following methodology:

1- Select a proper VM from a given set of available VM types at cloud DC through the VM selection manager module as shown in Fig. 2. The appropriate selection based on analyzing the application A requirements to identify the best VMs instance types (in terms of Core, RAM, storage, bandwidth…) that fit to deploy and run such an application. The main goal of this step is to satisfy one of the ENAV model objectives, i.e. minimize VMPC.
2- Apply a novel Energy, Network, and Application-aware VM genetic-based (ENAV-G) algorithm that aims to minimize both of the ENAV model objectives, i.e. ITC and DECC. For more explanation, the section below illustrates the pseudocode of the proposed algorithm.

4.1 ENAV-G algorithm

ENAV-G proposed approach is an adaptive genetic algorithm that aims to consolidate the set of selected VMs to deploy an application A in a minimum number of servers and storage nodes such that ENAV model objectives are minimized (i.e. ITC, DECC, VMPC) without violating the specified model constraints.

A genetic algorithm (GA) is a heuristic random searching mechanism [9, 10]. GA works via the method of fittest survival. It adjusts and optimizes its searching space using a probability optimization method. GA complexity depends on the number of generations (g), population size (n), and the size of individuals (m). Accordingly, GA complexity is O(gnm).

Figure 5 represents the adaptive genetic operators used to form the ENAV-G algorithm:

VM	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Gene	0001	0010	1000	0100	0001	1000	0010	0010	1000	0001	0100	1000	0010	0100	1000
Rack	4	3	1	2	4	1	3	3	1	4	2	1	3	2	1

Fig. 5 VMs as chromosomes

Encoding This is the process of representing the solution in the form of a string of bits. Consequently, the solution for the ENAV proposed VM mapping problem represented as a chromosome of n genes. Each gene is responsible for mapping a VM to a specific server/storage node. Figure 5 shows an example of VMs placement including 4 racks and 15 VMs and its corresponding chromosome.

Adaptive encoding The ENAV-G algorithm follows an adaptive encoding method to satisfy one of the ENAV model objectives (minimize ITC). More precisely, the ENAV-G algorithm encodes the chromosomes such that each gene represents one of the cloud customer requests, i.e. each gene represent a set of selected VMs that are suitable to deploy and run an application A**.** Accordingly, Fig. 6 shows an example of VMs placement including 4 racks and 15 VMs and its corresponding chromosome. Noting that each of the 15 VMs used to deploy 3 different application types (for three different cloud customers). Consequently, Fig. 6 illustrates adaptive encoding.

Selection This is the process of finding the best individuals/chromosomes that form the new fittest generation. Different selection methods are used such as Boltzmann strategy, rank-based selection, roulette wheel, and tournament selection [9]. In this paper, the roulette wheel strategy is used where rank is given to each individual according to its fitness value.

Crossover This is one of the important GA operators that leads to a new fittest generation. Using a crossover operator, a pair of chromosomes are selected to produce next-generation individuals. In this paper, a random point crossover is used to exchange VMs assignment between corresponding racks. The example shown in Fig. 7 illustrates a simple example of the used randomly selected crossover point and the newly generated individuals.

Mutation This operator encourages GA diversity and prevents generating uniform populations that lead to GA convergence to its local minima. By applying the mutation, GA can recover the good characteristics lost during the crossover. As a crossover, in this paper, a random point mutation is used to modify the selected gene characteristics.

Fitness function This is an effective GA algorithm determined by its fitness function. A successful fitness function evaluates each chromosome to guide the selection operator to find the best individuals and consequently find the final solution to the original problem. In this paper, the ENAV-G fitness function was chosen to be one of the ENAV model objective functions (the DECC function).

DECC pseudocode To calculate the DC energy consumption for deploying application A on cloud DC using Eq. 6, Algorithm 1 is used. The DECC is an important objective for the proposed ENAV-G algorithm since it represents the ENAV-G algorithm fitness function. Algorithm 1 shows a high-level pseudo-code to calculate the expected total energy consumption in cloud DC such that the requested customers' application is deployed and runs correctly.

Application Name	**A**					**B**					**C**				
VM	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Gene	0001	0010	1000	0100	0001	1000	0010	0010	1000	0001	0100	1000	0010	0100	1000
Rack			4					3					1		

Fig. 6 Application as chromosomes

The GetPowerConsumption() function (line 4) returns the power consumption of each $se_i/st_i/s_{wi}$ used to deploy and run application A in cloud DC (using Eq. 6).

Algorithm 1 DECC Energy Consumption DECC(A)

Input: set of applications A= {A_1, A_2, …, A_n}, DC
Output: DECC(A) value
Processing:
1:DECC=0
2: For each application A_i in A
3: For each hosted server/storage/switch se_j /st_j/ sw_j node
4: DECC+= se_j /st_j/ sw_j.GetPowerConsumption()*ActiveTime(se_j /st_j/ sw_j)
5: End For
6: End For
7: Return DECC

The high-level pseudocode of the adaptive ENAV-G algorithm is presented in Algorithm 2. As shown in Algorithm 2, the adaptive ENAV-G algorithm receives the cloud customers' applications hosted on a suitable VMs type and the DC components (available servers/storage nodes) to return the best VM/server mapping for each application such that the ENAV model objectives are satisfied. As the natural GA algorithm, the proposed adaptive ENAV-G algorithm goes through the following phases: 1- Generate the initial random population (line 2). 2- Evaluate the fitness function (line 3) (calculated using Eq. 6 as shown in Algorithm 1). 3- Select the top fittest chromosomes (line 6) and the other number of parents using the Roulette Wheel used selection method (line 8). 4- Generate a new population by applying crossover and mutation operators to the selected parents (lines 7–11). After several iterations (line 14), the proposed adaptive ENAV-G algorithm retrieves the fittest chromosome with the highest fitness value that forms the ENAV model solution. Moreover, the algorithm takes into account the ENAV model constraint by checking the servers/nodes/network capacity (lines 4–5).

Algorithm 2: ENAV-G

Input: set of applications A= {A_1, A_2, …, A_n}, DC
Output: VMs-to-servers/storage nodes mapping.
Processing:

1. Start
2. Generate initial population of chromosomes
3. Evaluate the fitness function for each chromosome/individual
4. Calculate the servers/storage nodes load after application A_i VMs allocation
5. Drop individuals that violate ENAV capacity constraints
6. Find the top two fittest chromosomes and consider them elite;
7. While !(initial population size)
8. Using random Roulette Wheel method select two parents (X & Y) from the fittest generated population
9. Perform crossover between X & Y
10. Move the newly formed individuals to the next generation
11. End While
12. Replace the current population.
13. Mutate using probability Pm=0.01- 0.02
14. go to step 3 until a specified termination condition (i.e. fitness threshold or end of iterations)
15. End

Parent 1:	1110 0010 1000 … 1000	0100 0011 0100
Parent 2:	1000 0100 0110 … 0010	0001 1001 0101
Child 1:	1110 0010 1000 … 1000 0001 1001 0101	
Child 2:	1000 0100 0110 … 0010 0100 0011 0100	

Fig. 7 Chromosome crossover

5 Performance evaluation

This section simulates the effectiveness of the proposed ENAV model and answers the following questions through extensive simulation using CloudSimSDN [20] simulator.

1- Does the ENAV model guarantee deadline constraint and the workload of data-intensive applications completed within the scheduled time given by SLA?
2- Does the ENAV model optimize the energy consumption of the cloud environment?
3- How effective is the ENAV model in reducing the intra-DC traffic?
4- Does the proposed adaptive ENAV-G provide a good solution to solve the ENAV model?
5- Does the ENAV-G algorithm provide a scalable and effective solution?
6- How do the objective factors (Eqs. 1, 6, 10, and 11) that form the ENAV model contribute to cloud performance and QoS?
7- How does the ENAV model compare with other VM placement model?

5.1 CloudSimSDN toolkit

CloudSimSDN toolkit is an add-on library to CloudSim that has been used for the simulation of the SDN cloud environment. It supports the development of a new SDN operating system to control the routing protocols and new controller development besides VM scheduling, migration, and consolidation methods. It provides comprehensive modeling of the energy consumed by the DC resources such as servers and storage nodes and different network levels (edge, aggregation, and core network level). To model our proposed ENAV-G algorithm, we adapted the CloudSimSDN toolkit to handle our proposed adaptive genetic algorithm.

5.2 Simulation setup

The simulation experiments conducted on Intel(R) Core(TM) i7 Processor 3.4GHz, Windows 10 platform using NetBeans IDE 8.2, and JDK 8u111. To address the efficacy of the proposed model in handling a large scale cloud DCs and the increasing complexity of the proposed ENAV-G algorithm, different scenarios were generated that include synthetic and real workload traces, varying the DC topology, the number of servers and storage nodes, the number of VMs requested by each application type and the dependencies among them, as well as varying the load characteristics of the servers/storage nodes and VMs.

To deploy and run the requested cloud customers' applications, we emulate Amazon VMs specifications to set the property of RAM, MIPS (million instructions per second), storage, and bandwidth (BW) as shown in Table 2. To measure effectively the response time metric, two different cloud environments are tested. One considered servers are homogeneous of Type 1 while the other is considered heterogeneous hosts of types Type 1–4 (as shown in Table 3). We assume that servers will consume the full system power when the server is on. Each host's power consumption is based on a power model, which is based on a benchmark result provided by SPEC (https://aws.amazon.com/about-aws/global-infrastructure/). Table 4 presents the server's power consumption at different load levels. Table 5 shows the parameters setting of the genetic algorithm based on a benchmark used parameters [19].

The emulated DC network topology was generated using a three-tier network topology (as shown in Fig. 3) that made up of 4 cores, 16 aggregation, and 32 rack switches. Thus, each rack holds 32 servers connected using physical links of 100 Mbps. While, 1 Gbps links were used for the interconnecting edge, aggregation, and core switches.

5.2.1 Synthetic scenario

Using synthetic workload traces, this scenario is used to practice the effect of the proposed ENAV-G in optimizing the response time and the energy consumption using 3-tier web applications in different environments. To imitate a real loaded DC, the number of generated applications varies between 100 and 300 consisting of 300–900 VMs. The generated workload is based on a web service model [20] to ensure the validity of active switches during the VMs running time.

The effectiveness of the proposed ENAV-G algorithm shall be tested and compared to a number of benchmark VM placement heuristic algorithms that are used usually for such types of tested applications. Accordingly, Best Fit (BF), Worst Fit (WF), Combined Least Full First (CLFF), and Combined Most Full First (CMFF) are used. BF and WF are VM placement algorithms that choose the most and least full host in terms of computing power respectively. On the other hand, CLFF and CMFF are well-known VM placement methods that used a joint network-energy efficient policy to choose the least and full host in terms of both compute and network bandwidth respectively.

5.2.2 Wikipedia scenario

This scenario aims to find the effect of the proposed ENAV model using a more real cloud environment. The internal cloud configuration was the same as indicated in the previous

Table 2 Amazon VMs specifications

VM instance type	VM model	Cores	MIPS	RAM(GB)	Bandwidth(Mbps)
Web Server	Standard	1	2000	0.5	100 Mbps
	Memory optimized	2	1500	2	100 Mbps
Application Server	Large	1	1500	2	100 Mbps
	XLarge	2	2400	3	100 Mbps
Database Server	Medium	2	2000	8	450 Mbps
	Large	4	2400	16	750 Mbps

Table 3 Server type and specifications

Server type	Specifications
Type 1	IBM server ×3250 (1 x [Xeon ×3470 2933 MHz, 4 cores], 32GB)
Type 2	IBM server ×3250 (1 x [Xeon ×3480 3067 MHz, 4 cores], 64GB)
Type 3	IBM server ×3550 (2 x [Xeon ×5670 2933 MHz, 6 cores], 48GB)
Type 4	IBM server ×3550 (2 x [Xeon ×5675 3067 MHz, 6 cores], 64GB)

synthetic scenario. However, the generated workloads based on Wikipedia traces are obtainable for the three-tier application model.

For both scenarios four metrics are used: 1- DC Power consumption using both servers and network devices, 2- the application response time, 3-the Intra DC network traffic cost, and 4- the total application deployment cost. All of the following experiments started with an empty DC. Though, gradually the DC gets crowded due to the increase in the number of deployed applications that imitate the reality of cloud dynamicity.

DC power consumption To assess the importance of the proposed ENAV model in reducing the total DC power consumption, two environments are tested: homogenous and heterogeneous DC devices as stated in the above section. Figure 8 displays the importance of the proposed ENAV-G algorithm in consolidating the communicated application VMs that lead to an average of 40% energy efficiency using different scenarios, loads, and environments. Through these experiments, the total DC power consumption was measured using the total consumption of the servers and network devices. As shown in Fig. 8, our approach contributes to high energy reduction due to the fact that ENAV-G algorithm always activates the minimum number of servers. It can be observed that, with the increasing number of workloads, the percentage of energy efficiency decreases. This is because more VMs and hosts are used to accommodate the deployed applications. One of the interesting observations in Fig. 8a reveals the power reduction in network devices. This depicts the fact that the ENAV-G algorithm localizes intra-DC traffic more efficiently compared to other benchmark algorithms. Overall, the ENAV-G algorithm has been able to contribute to be a high energy-efficient VM placement algorithm and to be scalable and adaptive in different cloud environments.

Application response time (ART) ART that returns the time difference between cloud user request time and application response time [17], is a key metric to evaluate the significance of the proposed ENAV-G algorithm in achieving high cloud QoS. Moreover, this metric is a significant factor to measure the effect of any placement algorithm on SLA degradation. One of the important parameters to conduct this experiment is scalability. Consequently, 20-time intervals were randomly selected. According to the experimental results, Fig. 9, the ENAV-G algorithm achieved the best response time among the competing algorithms. This is due to the

Table 4 IBM servers host load to energy (Watt) mapping table

Server type	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
IBM ×3470	41.6	46.7	52.3	57.9	65.4	73	80.7	89.5	99.6	105	113
IBM ×3480	42.3	46.7	49.7	55.4	61.8	69.3	76.1	87	96.1	106	113
IBM ×5670	66	107	120	131	143	156	173	191	211	229	247
IBM ×5675	58.4	98	109	118	128	140	153	170	189	205	222

Table 5 Genetics parameter settings

Parameter	Value
Population size	100
Number of generations	50
Crossover rate	0.8
Mutation rate	0.1

significant decrease in intra-DC communication (shown in Fig. 10) that affects and decreases the network transmission time. Thus, it leads to an average of 72% reduction in application response time using synthetic scenario (Fig. 9a) and 60% using Wikipedia scenario (Fig. 9b) that is due to the use of more complex applications and workload. In short, the ENAV-G algorithm improves the performance of the cloud DC deployed applications that could attain high cloud QoS and low SLA violations.

Intra-DC network traffic The consolidation achieved by the ENAV-G algorithm leads to a significant decrease in intra-DC communication traffic. Figure 10 illustrates the impact of the proposed genetic-based consolidation algorithm in achieving a high reduction in Intra-DC communication traffic that reaches up to an average of 90% compared to other baseline VM placement algorithms using both scenarios. One of the most fascinating observations, as shown in Fig. 10, is that the workload consolidation achieved by the ENAV-G algorithm helps the communicated VMs to be connected and communicate using the servers' memory instead of data center network (DCN) devices. Moreover, reducing the intra-DC communication traffic leads obviously to reduce the intra-DC network congestion. Accordingly, the network

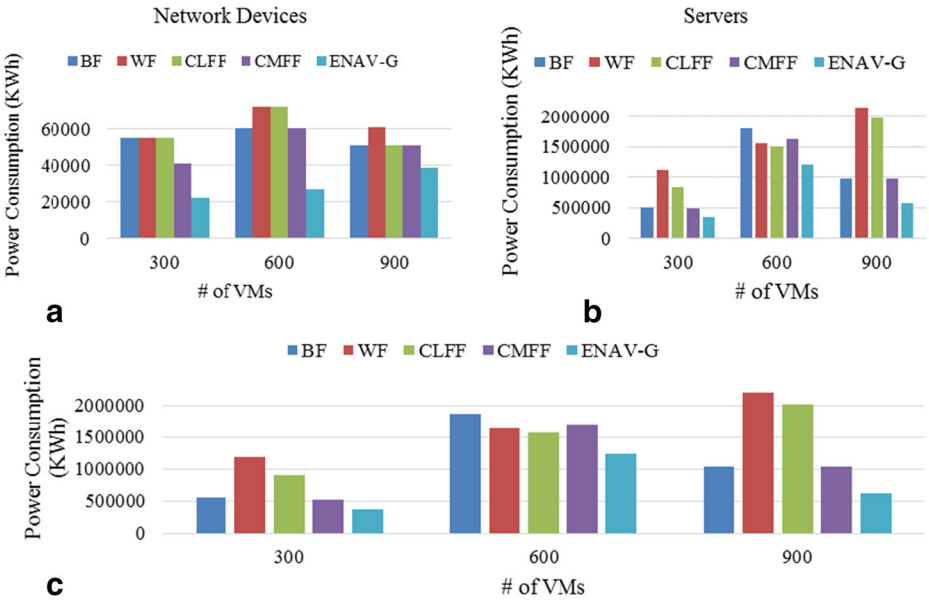

Fig. 8 Comparison of Average DC Power Consumption (**a**) Network devices, (**b**) Servers, (**c**) Including both servers and network devices

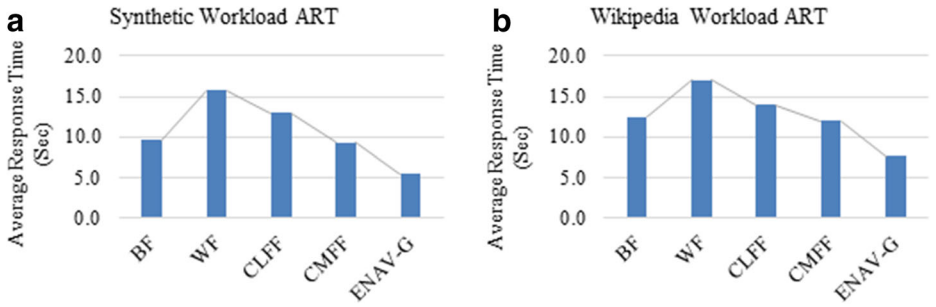

Fig. 9 Applications average response time (ART) using (**a**) Synthetic Workload, (**b**) Wikipedia workload

transmission time and delay decrease harshly and lead to high response time as shown previously in Fig. 10.

Application Deployment Cost (ADC) ADC is an important metric to evaluate the usefulness of the proposed ENAV VM placement model in achieving high revenue for the cloud providers. Noting that to compute the ADC the VM pricing model is taken as designated by amazon [24]. Figure 11 displays the importance of the VMPC function as an important objective in modelling the ENAV VM placement method (Eq. 10). The results display the usefulness of choosing optimum VM selection type through the ENAV Controller module (VM Selection Manager as shown in Fig. 2). According to the experimental results, the ENAV VM placement model achieves up to an average of 50% reduction in ADC and outperforms other benchmark VM placement algorithms. Figure 11 reveals that the ADC cost increases proportionally with the number of users reflected through the applied workload type.

6 Conclusion and future work

This paper presents a novel VM placement method for multi-tier applications hosted in large cloud DCs. The proposed ENAV model aims to minimize the cloud DC energy consumption, the VM placement cost, and the intra-DC communication traffic as well as network congestion. The proposed methodology to solve the ENAV model projected an adaptive genetic-based algorithm (ENAV-G) to solve such a problem. Different scenarios are used to imitate a real cloud environment and the proposed model proves its efficacy over a benchmark and

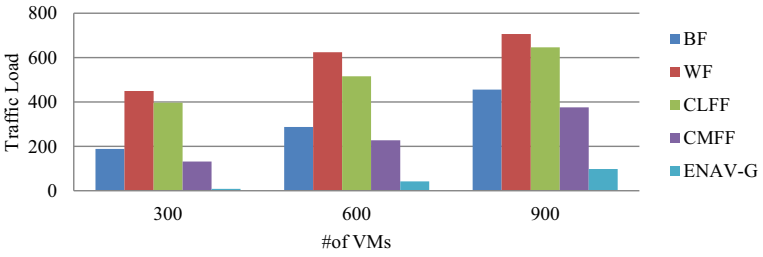

Fig. 10 Average Intra-DC communication traffic of synthetic and Wikipedia workload

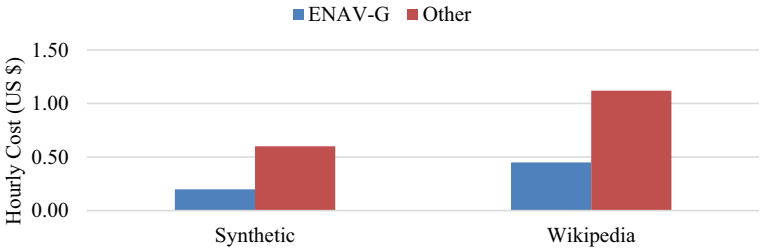

Fig. 11 Average application deployment cost using different workload (i.e. different number users)

baseline approaches in this field. For further studies, application priority could be considered for such a critical application. Moreover, it is also interesting to study the effect of the proposed VM placement model on different parameters that affect cloud QoS such as system failure.

Declarations

Conflict of interest The authors confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

References

1. Cao G (2019) Topology-aware multi-objective virtual machine dynamic consolidation for cloud datacenter. Sust Comput: Inform Syst 21:179–188
2. Cao G, Zhang C, Liu W (2017) Fast communication-aware virtual machine dynamic consolidation for cloud data center. In 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC) (pp. 237-244). IEEE
3. Cook G, Pomerantz D, Rohrbach K, Johnson B, Smyth J (2015) Clicking clean: a guide to building the green internet. Greenpeace Inc., Washington, DC
4. Das MS, Govardhan A, Lakshmi DV (2019) Cost minimization through load balancing and effective resource utilization in cloud-based web services. Int J Nat Comput Res (IJNCR) 8(2):51–74
5. Dayarathna M, Wen Y, Fan R (2015) Data center energy consumption modeling: a survey. IEEE Commun Surv Tutorials 18(1):732–794
6. Dias DS, Costa LHM (2012) Online traffic-aware virtual machine placement in data center networks. In 2012 Global Information Infrastructure and Networking Symposium (GIIS) (pp. 1-8). IEEE
7. El Motaki S, Yahyaouy A, Gualous H, Sabor J (2019) Comparative study between exact and metaheuristic approaches for virtual machine placement process as knapsack problem. J Supercomput 75(10):6239–6259
8. Ghobaei-Arani M, Souri A, Baker T, Hussien A (2019) ControCity: an autonomous approach for controlling elasticity using buffer Management in Cloud Computing Environment. IEEE Access 7:106912–106924
9. Jatoth C, Gangadharan GR, Buyya R (2019) Optimal fitness aware cloud service composition using an adaptive genotypes evolution based genetic algorithm. Futur Gener Comput Syst 94:185–198
10. Kaur K, Kaur N, Kaur K (2018) A novel context and load-aware family genetic algorithm based task scheduling in cloud computing. In Data Engineering and Intelligent Computing (pp. 521–531). Springer, Singapore
11. Kumar P, Kumar R (2019) Issues and challenges of load balancing techniques in cloud computing: a survey. ACM Comput Surv (CSUR) 51(6):1–35
12. Liu P, Bravo G, Guitart J (2019) Energy-aware dynamic pricing model for cloud environments. In International Conference on the Economics of Grids, Clouds, Systems, and Services (pp. 71-80). Springer, Cham
13. Mann ZÁ (2015) Allocation of virtual machines in cloud data centers—a survey of problem models and optimization algorithms. ACM Comput Surv (CSUR) 48(1):1–34

14. Rawas S, Itani W, Zaart A, Zekri A (2015) Towards greener services in cloud computing: research and future directives. In 2015 International Conference on Applied Research in Computer Science and Engineering (ICAR) (pp. 1-8). IEEE
15. Rawas S, Itani W, Zekri A, Zaart AE (2017) ENAGS: energy and network-aware genetic scheduling algorithm on cloud data centers. In Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing (pp. 1-7)
16. Rawas S, Zekri A (2018) Location-aware energy-efficient workload allocation in geo distributed cloud environment. J Comput Sci 14(3):334–350
17. Rawas S, Zekri A, El Zaart A (2018) CELA: cost-efficient, location-aware VM and data placement in geo-distributed DCs. In international conference on cloud computing and services science (pp. 1-23). Springer, Cham
18. Sarkar S, Chatterjee S, Misra S (2015) Assessment of the suitability of fog computing in the context of internet of things. IEEE Trans Cloud Comput 6(1):46–59
19. Son J, Dastjerdi AV, Calheiros RN, Buyya R (2017) SLA-aware and energy-efficient dynamic overbooking in SDN-based cloud data centers. IEEE Trans Sustain Comput 2(2):76–89
20. Son J, Dastjerdi AV, Calheiros RN, Ji X, Yoon Y, Buyya R (2015) Cloudsimsdn: modeling and simulation of software-defined cloud data centers. In 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (pp. 475-484). IEEE
21. Vicentini C, Santin A, Viegas E, Abreu V (2019) SDN-based and multitenant-aware resource provisioning mechanism for cloud-based big data streaming. J Netw Comput Appl 126:133–149
22. Vidal J (2017) http://www.climatechangenews.com/2017/12/11/tsunami-data-consume-one-fifth-global-electricity-2025/
23. Wei W, Gu H, Lu W, Zhou T, Liu X (2019) Energy efficient virtual machine placement with an improved ant colony optimization over data center networks. IEEE Access 7:60617–60625
24. Yuan H, Bi J, Zhou M, Sedraoui K (2017) WARM: workload-aware multi-application task scheduling for revenue maximization in SDN-based cloud data center. IEEE Access 6:645–657
25. Zhao DM, Zhou JT, Li K (2019) An energy-aware algorithm for virtual machine placement in cloud computing. IEEE Access 7:55659–55668