EEBA: Energy-Efficient and Bandwidth-Aware Workload Allocation Method for Data-intensive Applications in Cloud Data Centers

Soha Rawas and Ahmed Zekri

Abstract— **Cloud computing is a promising technology for providing efficient virtualized compute and storage resources to users on a pay-per-usage model. Large-scale geographically distributed data centers have been established to support the increasing demand for cloud services. Execution of data-intensive workloads is a challenging problem especially when objectives such as load balancing and energy reduction are essential to reduce cloud providers operational cost while providing high quality-of-service to users. Meantime, the high rates of data transfers result in network congestion that increases the makespan of workloads. This paper presents a novel Energy-Efficient and Bandwidth-Aware workload allocation method to run data-intensive applications on geo-distributed cloud DCs. We formulated the workload allocation problem as a multi-objective optimization problem to minimize the workload makespan, data centers energy consumption, and communication network congestion overhead. We designed a meta-heuristic genetic algorithm to find a near-optimal workload allocation. Extensive simulations using synthetic and real traces showed a 32% average reduction of workload makespan and 35% average reduction in network traffic compared to benchmark allocation methods.**

Index Terms— **Green Computing, Energy Efficiency, Geo-Distributed Data Centers, Task Scheduling, Deadline, Data-intensive.**

I. Introduction

Cloud computing provides virtualized compute, storage, and network resources with cheaper prices in a pay-as-you-go model [1, 2, 27]. The growing demand for cloud services has led to the establishment of Data Centers (DCs) worldwide to cover users' requests. Currently, large providers, such as Google, Amazon, and Microsoft, operate tens of DCs around the globe to ensure higher availability and disaster recovery [3, 26]. The Wide Area Network (WAN) that connects the users' and the geo-distributed DCs plays an important role in satisfying user's satisfaction and providers' profit. Therefore, the network delay incurred in transferring data back and forth between the initial residence location and the virtual machines hosting the users' applications must be optimized for delivering excellent Quality of Service (QoS).

It is worth noting that fully replicating data across all DCs exhaust the storage space and the network bandwidth due to the overhead of updating all distributed copies. Besides, the Service-Level-Agreements (SLA) may enforce restrictions of replication due to security and privacy constraints. Therefore, managing data movement between the storage nodes and computing nodes have recently attracted researchers' attention.

Data-intensive applications manipulate and process large volumes of data resulting in higher network transfer cost than computational cost. Consequently, the allocation of users' tasks to virtual machines should optimize the WAN network delay and hence avoid link congestion. Therefore, workload allocation methods should exploit the available bandwidths of the network links to efficiently distribute the workload tasks among the available virtual machines.

Currently, cloud providers established geo-distributed DCs with thousands of computing, network, and storage resources leading to massive consumption of energy that pollute the environment with CO2 produce higher carbon emission. (3% of the global electricity production is due to server clusters in DCs [2, 25] with3.5% of the global carbon emission in 2018, to become 14% by 2040 [4].) Hence, cloud providers need to employ efficient workload allocation methods that lead to minimizing energy consumption for the sake of reducing operational cost and keeping the environment clean by reducing carbon dioxide emissions.

This paper proposes an Energy-Efficient and Bandwidth-Aware (EEBA) workload allocation method to help providers reduce operational cost while offering good QoS to users. The contributions of this paper are summarized as follows:

We formulate the problem of allocating data-intensive workload tasks to running VMs in geo-distributed cloud DCs as a multi-objective optimization problem that targets the minimization of energy consumption, network links congestion and workload makespan.

Manuscript received January 22, 2021; revised May 14, 2021

Soha Rawas, is a lecturer, supervisor and program coordinator at the Center for Continuing and Professional Education (CCPE), Beirut Arab University, Lebanon, e-mail: soha.rawas2@bau.edu.lb.

Ahmed Zekri, is an Assistant Professor in: the department of Mathematics & Computer Science, Faculty of Science, Beirut Arab University, Lebanon, e-mail: a.zekri@bau.edu.lb; the department of Mathematics & Computer Science, Faculty of Science, Alexandria University, Egypt, e-mail: ahmed.zekri@alexu.edu.eg

We design a meta-heuristic genetic algorithm that solved the formulated optimization problem efficiently.

We conduct extensive simulations using CloudSim toolkit to validate the efficiency of the proposed EEBA method using both synthetic and real workload traces. The results are compared with benchmark workload allocation algorithms showed the efficiency of our method in reducing power consumption and data transfer cost, and hence providing better QoS and fulfilling user satisfaction.

The rest of the paper is organized as follows. Section II presents the works related to the workload allocation problem in cloud systems. Section III presents a description of our proposed cloud model. Section IV shows our formulation to the workload allocation problem as a multi-objective optimization problem. Section V presents our proposed meta-heuristic genetic algorithm to solve the optimization problem. Section VI presents the evaluation of the proposed EEBA method using CloudSim simulation toolkit. Section VII concludes the paper.

II. Related Work

Banerjee et al. targeted the minimization of load balancing in solving the workload allocation problem [5]. They proposed a greedy task allocation method that minimized the makespan of the VMs and hosts as well as the tasks/cloudlets completion times, which lead to improving the hosts load balancing. Dong et al studied the minimization of DC energy consumption during tasks allocation [6]. They proposed the Most-Efficient-Server-First (MESF) method that schedules tasks to most energy-efficient servers. Chatterjee et al. proposed the Conductance cloudlets allocation policy calculated through the ratio of each VM processing speed to the sum of all available VMs processing speed [7]. Although the authors used the conductance of each VM to calculate the VMs capacity, however, the proposed algorithm missed the importance of the cloudlets length as well as the cloudlets distribution, which lead to load imbalance. Huai et al addressed the problem of energy reduction in heterogeneous cloud environments [8]. They proposed the Benefit-driven Scheduling (BS) method that maps tasks to the most energy-efficient server. Also, they proposed two different heuristic algorithms, Power Best First (PBF) and Load Balancing (LB), for tasks scheduling on homogenous servers. However, they didn't consider the network delay in selecting the computing resources for workload execution.

Klizavoivh et al. considered the network delay and bandwidth congestion on resource allocation [9]. The authors proposed an energy-efficient task scheduler with traffic load balancing, e-STAB, which consolidates jobs to a minimum number of activating servers to minimize congestion and network delay. However, the method targets environments with a single DC. Alizadeh et al. studied the problem of inter-DC network traffic generated by MapReduce jobs when allocating them to geo-distributed DCs [10]. Their proposed optimization problem jointly optimizes input data movement and task placement. Although their experiments showed promising results in reducing the inter-DC network traffic, however, the applied method does not address the problem of energy consumption.

Toosi et al. [11] proposed a provisioning algorithm for scheduling deadline-constrained data-intensive applications while taking into account aspects such as data transfer time, the location of data, and the network bandwidth. However, the proposed algorithm does not target the minimization of DCs' energy. Abdi et al. proposed a model for deadline-constrained bag-of-tasks applications in federated hybrid clouds that minimized providers cost. However, they ignored the cost incurred due to energy consumption and data transmission [12].

Most of the proposed task allocation methods discussed above tackle the problem of minimizing the workload makespan while trying to reduce the energy consumption of the cloud resources. However, none studied the collective optimization among the makespan, load balancing, energy consumption, network links congestion, which is the goal of the current paper.

III. Model Description

The placement of users' data on storage nodes, such as Amazon S3 cloud, could raise the overhead of moving data to/from the compute nodes if the user's task is not allocated to a proper virtual machine, and hence physical machine [13]. Insufficient bandwidths of network links lead to network congestion and longer transfer delays resulting in data packet loss and connection blocking [14]. Therefore, efficient workload allocation has to take into consideration the network delay cost, which significantly could degrade performance severely and hence reduce QoS as well as providers' profit. Besides, incorporating energy cost in the mapping process will further reduce the operational cost in addition to obtaining sustainable environment due to reducing the Carbon dioxide footprint.

This section introduces the cloud system model components and the target application model.

A. Cloud system components

The target cloud computing system employed in this paper consists of four components as depicted in Fig. 1:

1. Provider Data Centers: The cloud provider has a set of geographically distributed DCs. Each DC has a set of heterogeneous physical machines (compute nodes) that host a set of running virtual machines ready for running workload tasks.
2. Cloud Broker: A cloud broker is an intermediary between the compute resources on DCs and the users. It accepts the users' workload tasks and allocates them to a set of preconfigured and running virtual machines distributed among the providers' DCs. The users' requests require the movement of data files to the virtual machines where their tasks run. It is assumed the data files associated with the workload tasks are in one Storage Node (SN) that could be on the S3 cloud server. To move data faster between the SN and the physical machines hosting the workload VMs, we assume network links are configured and set prior to the workload allocation, and the bandwidth of the available links are known.
3. Cloud Users: Cloud users send their services' requests to a cloud provider broker that assigns them to the currently running VMs at different provider DCs.

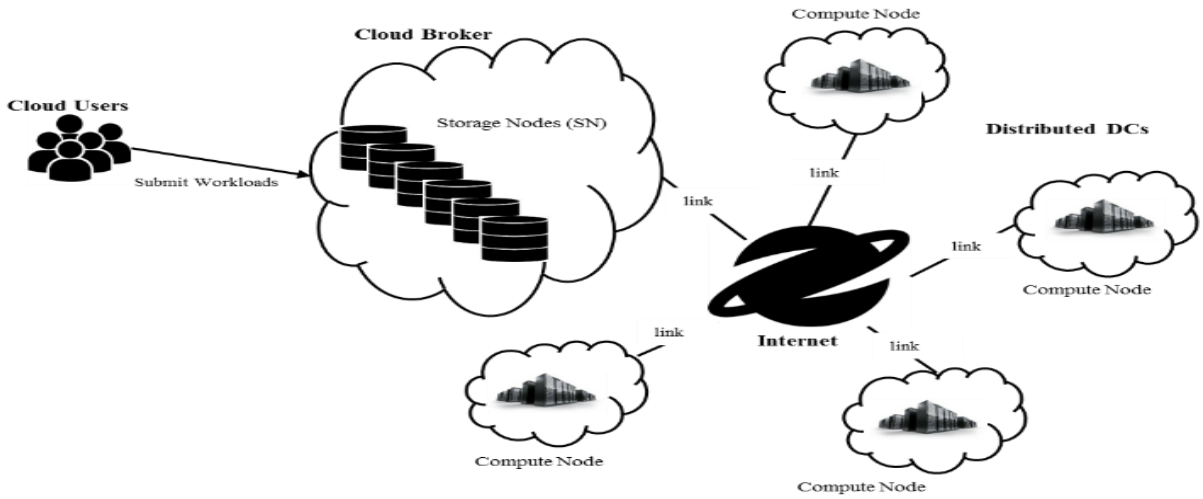

Fig. 1. The target cloud computing system model for running data intensive workloads.

B. Workload Model

There are a variety of cloud service provisioning models depending on the users' requirements. Cloud providers and their brokers need to employ efficient allocation strategies to provide enhanced QoS while increasing their profits. One provisioning situation arises from offering a reservation model where independent users have their own tasks and associated data that are required to execute during a specific period of time (such as virtual classrooms.). In such situations, the workload is static and require a fixed number of virtual machines. We target workloads consisting of tasks characterized by long data transfer times compared to their shorter processing times. For example, a collection of users need to process a set of independent large files, such as video clips, in a short period of time. Therefore, cloud brokers should employ allocation algorithms that schedule a collection of independent tasks (Bag-of-Tasks BoTs) so that the overall makespan of the BoT is minimal while employing a number of preconfigured VMs placed on geographically distributed DCs owned by a provider.

Our proposed EEBA allocation method considers heterogeneous VMs which are pre-configured and ready to use during a specific period of time. The pre-configuration as in Azure cloud environment decreases the overhead of setting up and creating new VMs on the spot leading to the reduced computational cost of the software stack and the operating system configuration [14].

Our objective is to efficiently allocate tasks and their data so that the overall completion time of the workload execution is minimum while reducing the power consumption of computing resources to decrease DCs operational costs.

IV. Problem Formulation

In this section, we present a mathematical formulation of the proposed EEBA workload allocation method as a multi-objective optimization problem.

A. Model Assumptions

The EEBA allocation method assumes the following:

- The target data-intensive workload consists of a set of independent tasks, i.e. it is a bag-of-tasks (BoT) workload.
- There is a set of available VMs pre-configured and ready for execution on a set of PMs hosted in a set of distributed DCs.
- Each VM executes one task at a time, and the task can use all the processing cores allocated for the VM.
- The data files associated with the tasks are initially located in a storage node (SN), where they are moved back and forth to the DCs where the tasks reside.
- The network links between the DCs are preconfigured before starting the workload allocation. Therefore, the network links delays and available bandwidths are known a priori.

TABLE 1 DESCRIPTION OF PARAMETERS USED IN THE PROBLEM FORMULATION.

Notation	Description
T	set of independent tasks (BoT) to be scheduled
D	Set of geographically distributed DCs
V	Set of VMs hosted on D
H	Set of physical machines hosting V
F	Set of data files associated with tasks T
m	Total number of hosts
d	Total number of DCs
l	Total number of VMs
vj	Total number of VMs allocated to host hj
n	Total number of tasks
taski	A single task (cloudlet) submitted by a user, taski ∈ T
fi	A data file associated with a task ti, fi ∈ F
dci	DC at location i s. t. dci ∈ D
vm_i^j	virtual machine i allocated to host hj s. t. vmj ∈ V and hj ∈ H
Tt	Subset of worload tasks , Tt ⊂ T to be run on resource vm_i^j
t	Total number of tasks assigned to be run on resource vm_i^j
DT_T	A deadline time to execute workload T; set in the SLA between the broker and users.
$et(t_k, vm_i^j)$	Execution time of task tk on vmi allocated to host hj
$tt(t_k, vm_i^j)$	Transfer time of task tk to be processed on compute resource vmi allocated to host hj
$ll(task_i)$	Length of a task taski in million instructions
$mips(vm_i^j)$	The available MIPS for virtual machine vmi allocated to host hj
$pe(vm_i^j)$	The number of processing elements assigned to vmi allocated to host hj
$LinkBw(dc_i)$	The available network link bandwidth between the SN

	and dc_i
dt(dc_i)	The delay time before sending a data file between the SN and the virtual machine vm_i^j at DC dc_d
P$ower(vm_i^j)$	power consumption of the vmi allocated to host hj
CTT	The completion time of the workload T
$tc(LinkBw(dc_i)$	Congested time span of the network link between the SN and dci
$ta(LinkBw(dc_i))$	Available time span of the network link between the SN and dci

B. Mathematical Formulation

Given a workload consisting of n tasks T={t1, t2, …,tn}, a set of f files F={f1, f2, …,ff} associated with tasks T, a set of d DCs D={dc1, dc2, …, dcd} distributed in different geographical regions, a set of m hosts H={h1, h2, …,hm} distributed among d DCs, and a set of l VMs V={vm1, vm2, …,vml} allocated to m number of hosts, The goal is to allocate the set of tasks T to the set of virtual machines V such that a Deadline time set by the SLA is met. Our proposed EEBA method target the minimization of three sub objectives: workload makespan, network links congestion, and DCs energy consumption. Our proposed method allocates all tasks of workload as one BoT. It also treats all VMs as one aggregated list, even they are located on different DCs. The hosts are also aggregated in one list. Therefore, we introduce three matrices to depicts the relationships between these entities.

Hosts/DCs relationship. Each DC contains more than one host and each host is located at only one DC. Let A be m x d matrix showing the mapping status of the m hosts to the d DCs as follows:

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

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

$$a_{ij} = \begin{cases} 1, & if\ h_i\ is\ located\ in\ DC\ dcj \\ 0, & otherwise \end{cases}$$

VMs/Hosts relationship. Each host can hold more than one VM and each VM is executed at only one host. Let B be l x m matrix showing the mapping status of the l VMs to the m hosts as follows:

$$B = \begin{bmatrix} b_{11} & \cdots & b_{1m} \\ \vdots & \ddots & \vdots \\ b_{l1} & \cdots & b_{lm} \end{bmatrix}$$

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

$$b_{ij} = \begin{cases} 1, & if\ vm_i\ is\ allocated\ to\ host\ hj \\ 0, & otherwise \end{cases}$$

Tasks/VMs relationship. Each VM can execute more than one task and each task is executed at only one VM. Let C be n x l matrix showing the mapping status of the n tasks to the l VMs as follows:

$$C = \begin{bmatrix} c_{11} & \cdots & c_{1l} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nl} \end{bmatrix}$$

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

$$c_{ij} = \begin{cases} 1, & if\ t_i\ is\ mapped\ to\ vmj \\ 0, & otherwise \end{cases}$$

Now, we show how the workload makespan, the network link overhead, and the total energy consumption are calculated using the above three matrices A, B, and C.

Makespan calculation. Let the completion time of task taskk on vm_i^j is the time instance where it finishes execution. This time includes processing/execution time of the task and the data transfer time.

The execution time of task taskk on vm_i^j will be defined as:

$$et(task_k, vm_i^j) = \frac{ll(task_k)}{mips(vm_i^j) * pe(vm_i^j)} \quad (1)$$

In Equation (1), we assume a task taskk is processed in parallel using all the available PEs/cores of the virtual machine vm_i^j.

The transfer time to send a data file fa, associated with task taska, from the SN where it initially resides to a virtual machine vm_i^j located in a DC dc_x, and then return the file back to its initial location can be computed as follows:

$$tt(task_a, vm_i^j) = \frac{inSize(f_a) + outSize(f_a)}{LinkBw(dc_x)} + 2 * dt(dc_x) \quad (2)$$

We assume that the available link bandwidth between the storage node (SN) and the DC, where a virtual machine vm_i^j resides, are the same in both directions

From Equations (1) and (2), the completion time for a task taskk to be processed on VM vm_i^j is calculated as follows:

$$ct(task_k, vm_i^j) = \begin{cases} tt(task, vm_i^j) + et(task, vm_i^j), \\ if there\ are\ available\ resources\ in\ vm_i^j \\ ct(task_{k-1}, vm_i^j) + tt(task, vm_i^j) + et(task, vm_i^j), \\ otherwise \end{cases} \quad (3)$$

In Equation (3), we assume the subset of tasks allocated to a VM is queued and processed one by one as revealed in Fig. 2.

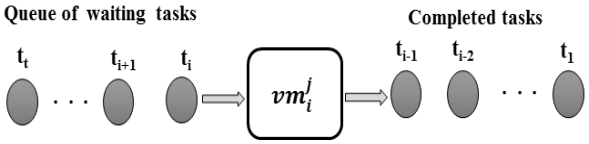

Fig. 2. Scheduling t tasks to $\boldsymbol{vm_i^j}$, the waiting tasks are inserted into a FIFO queue

The makespan schedule to execute a set of t tasks Tt={taska, taskb,…, taskt} allocated to a resource vm_i^j s. t. Tt ⊂T, will be simply the completion time of the last queued task, taskt, as given in Equation (4) below.

$$MS_{vm_i^j} = ct\left(task_t, vm_i^j\right) \tag{4}$$

Consider a host hk and a set of v VMs running on this host such that Vv={vm_1^k, vm_2^k, ..., vm_v^k} ⊂ V. Therefore, the makespan schedule to execute all tasks on host hk is defined as the maximum among all makespan schedules on all VMs running on host hk, as given in Equation (5).

$$MS_{h_k} = max_{i=1}^{v} b_{ik} * MS_{vm_i^k} \tag{5}$$

Given a set of m hosts, H = {h1, h2, ..., hm}, distributed on a set of d DCs. Using Equations (4)-(5) above, we can calculate the makespan to execute all tasks in T on all active hosts as:

$$MS_T = max_{i=1}^{d} max_{i=1}^{m} a_{il} * MS_{h_i} \tag{6}$$

Network Overhead calculation. Data-intensive applications are highly sensitive to WAN communication network (between SN and specific DC) and its available bandwidth that leads to communication delay and consequently networks congestion [11]. In this paper, we introduce the network overhead (NOV) as a metric to guide us in effectively allocating a workload's tasks to the VMs hosts on different hosts in different DCs such that the accompanied data transfers are minimum. The overhead occurs when the available bandwidth of the network links connecting the storage nodes, where data resides, and the compute nodes/hosts on DCs is insufficient leading to excessive network traffic due to link congestions.

Let $tc(LinkBw(dc_i))$ be the time span of the network link between the SN and the VMs placed on dci being congested, and $ta(LinkBw(dc_i))$ be the time span of the network link between the SN and the VMs placed on dci being active. Hence, we define the network overhead NOV_T as the average ratio of the time period when the network links experience bandwidth utilization of 100% as given in Equation (7) below.

$$NOV_T = \frac{1}{d}\sum_{i=1}^{d} \frac{tc(LinkBw(dc_i))}{ta(LinkBw(dc_i))} \tag{7}$$

Energy Consumption calculation. Developing an energy-aware task allocation algorithm requires measuring the dynamic power consumption to run the tasks on the compute resources. To derive a new power consumption model, real-time server power consumption monitoring is needed. However, this is out of the scope of this paper. We used the linear power model done in [13]. In this model, the power consumption of a server scales linearly with its CPU utilization. The total power consumed by a host/server is given by:

$$P(u) = P_{idle} + \left(P_{full} - P_{idle}\right) * u \tag{8}$$

where Pidle is the server power consumption without running any load on the server, Pfull is the amount of consumed power when the server is fully utilized, and u is the percentage of CPU utilization. Therefore, the power consumption of a host hj,d holding v number of VMs (sketched in Fig. 3 below) on DC dcd equals to the total power consumed by v VMs allocated on host j. Assume $Power(vm_i^j)$ is the amount of power consumed by a virtual machine vm_i^j at an instant of time, then the total power consumption of host hj,d at a given instant of time can be calculated as in Equation (9). Hence, the total energy consumption EC_T of all the m hosts H={ h1, h2, …,hm} for processing all the workload tasks T for time interval [0, ht] can be calculated as in Equation (10) below.

$$Power\left(h_{j,d}\right) = \sum_{i=1}^{v} b_{ij} * Power(vm_i^j) \tag{9}$$

$$EC_T = \sum_{k=1}^{d}\sum_{j=1}^{m} a_{jk} * Power(h_{j,k}) * ht \tag{10}$$

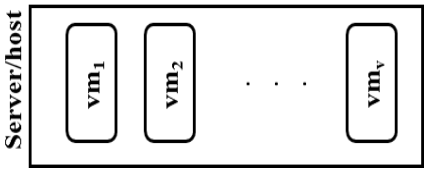

Fig. 3. A physical server/host with v VMs

C. The Optimization Problem

The main goal of the proposed method is to map a set of tasks T to a set V of VMs placed on geo-distributed DCs without violation of the workload deadline constraint DTT given by the SLA agreement between the provider and cloud user. In additions, the mapping should result in a reduction in the total execution cost. To fulfill this goal, we target three objectives. The first objective is to minimize the makespan to execute the set of T tasks on the set of available VMs. The second objective is to minimize the network's links overhead incurred due to data files transmission from the SN to the computing nodes. The third objective is to minimize the energy consumption of the computing servers because of running the VMs.

Given the above three mini-objectives, we can formulate our multi-objective optimization problem as follows:

$$minimize\begin{pmatrix} MS_T \\ NOV_T \\ EC_T \end{pmatrix} \tag{11}$$

Subject to:

1- The completion time of executing the workload T on a set of available VMs V should not exceed the deadline

time constraint DT(T) given by the SLA between the provider and user,

$$MS_T \leq DT_T \quad (12)$$

2- At any instance, a task is executed on only one VM,

$$\sum_{j=1}^{l} c_{ij} = 1, \qquad \forall\, i, i = 1,2,\dots,n \quad (13)$$

3- At any instance, each VM is executed at only one host:

$$\sum_{j=1}^{m} b_{ij} = 1, \qquad \forall\, i, i = 1,2,\dots,l \quad (14)$$

4- At any instance, each host is assigned to only one DC

$$\sum_{j=1}^{d} a_{ij} = 1, \qquad \forall\, i, i = 1,2,\dots,m \quad (15)$$

5- The task requirements do not exceed the capacity of the VM allocated to it.

$$\sum_{k=1}^{t} c_{ij} * task_k \leq Capacity(vm_j^h) \quad (16)$$
$$\forall\, i,j,h\ such\ that: i = 1,2,\dots,n\ ;\ j = 1,2,\dots,l;\ h = 1,2,\dots,m$$

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

V. The EEBA Methodology

In this section, we propose a genetic-based EEBA workload allocation method to solve the formulated multi-objective optimization problem in Equations (11) – (16), which is an NP-hard problem [15].

Given a set T={t1, t2, …, tn} of n tasks , a set F={f1, f2, ..., ff} of m files (input/output files of tasks T), a set D={dc1, dc2, …, dcd} of DCs distributed in different regions, and a set of distributed and running VMs V={vm1, vm2, …, vml} with different computing specification, the EEBA genetic algorithm tries to optimally allocate the set of tasks T to the set of running VMs V so that the total power consumption of the cloud DCs is minimized and the QoS is satisfied.

A. Proposed EEBA Genetic-Based Algorithm

In this section, we adopt Genetic Algorithms (GA) to solve the formulated multi-objective EEBA model given in Equations (11) - (16). We denote our adapted genetic algorithm as EEBA. First, we describe the genetic operators used in the genetic algorithm and relate them to the task allocation problem under investigation. Then, we present algorithms to calculate the three components of the multi-objective function (Equation 11) that we use as the algorithm's fitness function.

A.1. Genetic operators

Encoding: A chromosome represents a solution to the problem. That is the allocation of the n workload tasks to the running VMs. A chromosome consists of n genes each of which is responsible for mapping a task to a specific VM. The value of a gene is a positive integer between 1 and l, the total number of VMs. Table 2 shows an example of mapping a workload consisting of 10 tasks to 3 VMs. Therefore, a chromosome consists of 10 genes.

TABLE 2 A ONE CHROMOSOME CONSISTING OF TEN GENES (TASKS)

Task #	10	9	8	7	6	5	4	3	2	1
Binary representon of a gene	00 01	00 10	00 01	01 00	00 01	00 10	00 01	01 00	00 10	00 01
VM #	1	2	1	3	1	2	1	3	2	1

Initial Population: This is a set of chromosomes that are randomly created to initialize a way to an optimal solution. Each chromosome in the population represents a solution to the problem (T tasks allocated to V VMs) and it is called a population individual. From this initial population, the fittest individual(s) are selected to mate and produce the next generation.

Selection: There are various strategies to select the best individuals that produce fittest new generation such as Boltzmann strategy, rank-based selection, roulette wheel, and tournament selection. We used the roulette wheel where rank is given to each individual according to its fitness value.

Crossover: This is an important operator of GA that improves the quality of the newly generated population. Pairs of chromosomes (parents/individuals) are selected to produce next-generation individuals. We used random point crossover to exchange VMs assignment between corresponding tasks.

Mutation: To avoid generating uniform populations, a mutation operator is used to maintain genetic diversity in the subsequent generations. Moreover, mutation recovers the good characteristics lost during the crossover. It is used to modify the genes of a randomly selected chromosome using a mutation probability.

Fitness function: A successful GA algorithm depends on selecting a suitable fitness function to guide the chromosomes selection and hence the final solution to the original problem. We selected the objective function of the formulated multi-objective optimization problem given in Equation (11) as the fitness function. The three components of the objective function, namely the workload makespan, the network overhead, and the DCs energy consumption, are merged into a single function by a weighted sum of the three components.

B. Fitness function calculation

The following three algorithms show how the three components of the multi-objective function (Equation 11) are calculated.

Network Overhead Calculation: Algorithm 1 shows pseudocode for calculating the network overhead (discussed in Section III). "D" and "LinkBw" are the sets of a geo-distributed DCs and network link bandwidths between the SN and distributed computing nodes, respectively. "d" refers to one of "c" DCs in "dc" list. The key idea of this algorithm is to estimate the average ratio of the time when the network links experience bandwidth utilization of 100% and the active times of the links. It tracks the available and the requested bandwidth of each network link (line 3 and 4). If the available bandwidth is less than the requested one, it

calculates the elapsed time of the link being congested (lines 5-8). After computing the total time of network links being congested, it is divided by the time span of the links being active (line 10). The algorithm runs until the end of workload processing (line 11). Finally, the algorithm retrieves the NOV value after dividing the total congested time of inter-DCs links by the number of DCs "c" (line 12-13).

Algorithm 1 Network Overhead Calculation NOV(T)

Input: D={dc1, dc2, …, dcd}, a set of d DCs
$LinkBw = \{LinkBw(dc_1), LinkBw(dc_2), \dots, LinkBw(dc_d)\}$, a set of configured network link bandwidths between the SN and the distributed DCs D.
Output: NOV value
Processing:
1: Do
2: For each DC i in D list
3: AvailableBw(dci) = LinkBw(dci).getUtilization() // get utilization of LinkBw(dci)
4: RequestedBw(dci)=LinkBw(dci).getRequested() // get requested of LinkBw(dci)
5: If AvailableBw(dci) < RequestedBw(dci)
6: timeDifference=LinkBw(dci).getTime()-LinkBw(dci).previousActiveTime()
//get the elapsed time of the LinkBw(dci) being congested
7: OverheadTimeDifferenceLinkBw(dci)+=timeDifference // compute the total congested
time of LinkBw(dci)
8: End If
9: End For
10: NOVDC+= OverheadTimeDifferenceLinkBw(dci)/activeTime // activeTime: is the time
span of the network link between the SN and dci being active.
11: Until the workload end of processing
12: NOV= NOVDC/d
13: Return NOV

Makespan Calculation: Algorithm 2 below calculates the makespan of a workload T. For each vmi of v number of VMs allocated to host hj (line 4) the vmi makespan is calculated using Equation 4 (line 5). Consequently, the host hj makespan will be the maximum makespan of v VMs scheduled on host hj (as shown in Equation 5 and lines 6-8). Hence, the makespan of workload T is the maximum makespan of m hosts (lines 10-12).

Algorithm 2 Makespan Calculation MS(T)

Input: H={h1, h2, …,hm}, a set of m hosts
V={vm1, vm2, …,vml}, a set of l VMs
T = $\{t_1, t_2 \dots, t_n\}$, a set n tasks
Output: Makespan value of workload T
Processing:
1: MS(T)=-1
2: for each host hj in H
3: MS(hj)=-1
4: for each VM vmi in v VMs allocated to host hj
5: MS(vm_i^j) = CompletionTime(t_t, vm_i^j) // MS(vm_i^j) is the completion time of the last queued task
tt allocated to vm_i^j as shown in Equation (4.4), such that CompletionTime(t_t, vm_i^j) calculated using Equation 4.3
6: if MS(vm_i^j)>MS(hj)
7: MS(hj) = MS(vm_i^j)
8: End if
9: End for
10: if MS(hj) > MS(T)
11: MS(T)= MS(hj)
12: End if
13:End for
14: Return MS(T)

Energy Consumption Calculation: Algorithm 3 below calculates the expected total energy consumption of DCs to execute the workload T. The GetPowerConsumption() function (line 4) returns the power consumption of each hi using the real consumption data provided in SPECpower benchmark [23] according to the level of utilization in Watts -Table 9 (implementation details are given in Section VI).

Algorithm 3 Energy Consumption Calculation EC(T)

Input: D={dc1, dc2, …, dcd}, a set of d DCs
H={h1, h2, …,hm}, a set of m hosts
Output: EC value
Processing:
1:EC=0
2: For each DC dci in DC list D
3: For each host hj in dci
4: EC+= hi.GetPowerConsumption()*ActiveTime(hi)// compute the energy consumption of hi
as per its utilization as shown in Table 9-Section VI
5: End For
6: End For
7: Return EC

C. The EEBA genetic-based algorithm

The proposed EEBA adaptive genetic algorithm is given in Algorithm 4 below. It describes the solution to the multi-objective optimization problem in Equations (11-16).

Algorithm 4: EEBA-G algorithm

Input: T= {t1, t2, …,tn}, a set of workload tasks. V= {vm1, vm2, …, vml}, a set of distributed VMs.
D={dc1, dc2, …, dcd}, a set of geo-distributed DCs, workload Deadline time DT(T).
Population size (pop_size; the number of solutions in each generation), Maximum number of generations (max_gen) (the used parameters are given in Table 3 –Section VI)
Output: allocating set of task T to the set of VMs V.
Processing:
1: Begin
2: Generate the initial population randomly with pop_size individuals
3: Evaluate each candidate solution (individual) by calculating the fitness value for each
individual in the initial population using Algorithms 1, 2 and 3.
4: Find the top two fittest individuals and consider them elite; pass them to next generation without
any changes.
5: While gen_size < pop_size
6: Use random Roulette Wheel method to select two chromosomes as parents
7: Perform crossover between the selected chromosomes
8: Pass the new individuals to the next generation
9: End while
10: Replace the current generation with the newly created generation.
11: Apply mutation operator with probability Pm=0.15
12: Go to step 3 until the max_gen or if the achieved makespan is less than or equal to DT(T) (Equation 12)
13: End

As shown in Algorithm 4, GA goes through the following phases:

1- Creation of the initial random population (line 2), where each population individual is considered a possible solution to the problem.
2- Calculating the Fitness evaluation (line 3) using the above Algorithms 1, 2 and 3. The fitness function is a weighted sum of three objectives.
3- Parents' selection to generate the new fittest populations by applying crossover and mutation GA operators to the selected parents (line 4-11). After a number of iterations, the algorithm retrieves the individual with the highest fitness from the last population as an optimal solution to the problem (line 12).

VI. Performance Evaluation

This section validates the effectiveness of the proposed EEBA method through extensive simulations using CloudSim 3.0.3 simulator. We start by presenting a brief description of the CloudSim environment as well as the modules that are used in our implementations. Then we show the performance of our method using both synthetic and real data sets.

A. CloudSim Toolkit

CloudSim is a toolkit for the simulation of cloud computing systems. CloudSim is an open-source development toolkit that supports the development of new resource management, application scheduling, VM allocations, migrations methods, and much more new implementation policies to improve the cloud environment from its various levels [24]. To model the adaptive EEBA genetic algorithm, we utilized CloudSim 3.0.3 by modifying the DC broker algorithm that plays the role of mediator between the cloud user and service provider.

B. Simulation Setup

The extensive simulations were conducted on Intel(R) core(TM) i7 Processor 3.4GHz, Windows 7 platform using NetBeans IDE 8.0.2 and JDK 1.8. Different scenarios were conducted through varying the number of distributed DCs and their specifications as well as the workload characteristics to validate the effectiveness of the proposed EEBA method. The network links characteristics are represented by the following two matrices:

1- Delay Matrix (DM) that stores and resembles the average value for communication delay between the SN and geographically distributed DCs hosting the compute nodes (VMs). The communication delay was approximated using the geographical distance since there is no general analytical model for the delay in the network [13]. However, in our experiment, we used the WAN Latency Estimator [16] to estimate the network latency in milliseconds.

2- Bandwidth Matrix (BM) that represents the bandwidth link capacity between the SN and the compute nodes at the DCs. The bandwidth between the SN and the geo-distributed DCs is randomly generated between [1Gb/s, 10Gb/s].

We conducted two different simulation scenarios. The first one uses a synthetic workload trace. This scenario randomly models the distributed cloud environment and measures the effectiveness of the proposed EEBA method using time-based metrics including the network overhead incurred. However, the second one uses the benchmark Planetlab workload traces [17] with real data about hosts' energy consumption to show an accurate estimate of the energy-saving due to using our proposed EEBA model.

For comparison purposes, the proposed adaptive EEBA genetic algorithm was compared and analyzed according to three benchmark task allocation algorithms: Shortest Job First (SJF), Round Robin (RR), an energy-efficient genetic-based task allocation algorithm [18], which we call Green Genetic algorithm (GGA), and the Location-aware Energy-Efficient (LAEE) genetic-based algorithm, that proposed in [13]. LAEE is a designed heuristic takes into account the cost of the data transfer time as well as the workload makespan without considering the network overhead between the SN and computing nodes.

Table 3 shows the experimental setup of the different genetic operators used in the proposed EEBA adaptive genetic algorithm based on a benchmark used parameters [13, 19].

Table 3 Genetics parameter settings

Parameter	Value
Population size	100
Number of generations	100
Crossover rate	0.8
Mutation rate	0.15

C. Experimental Results

C.1. The first Scenario

In this experiment, the cloud provider owns 4 different DCs distributed among 4 different regions: US, Asia, Australia, and Brazil. The SN of the cloud provider (as shown in Section III) is located in Australia.

Table 4 shows the average distance and latency between the cloud provider SN and geo-distributed DCs. To measure effectively the makespan metric, two different cloud environments are tested. One considered hosts are homogeneous of Type 1 (as shown in Table 5), and use small VM instance type (as shown in Table 6). The others are considered heterogeneous hosts of types: Type 1-7 (as shown in Table 5) and four different VM types (as shown in Table 6). The number of hosts for each DC varies within the range [100:300]. We assume that hosts will consume the full system power when the server is on. Moreover, we consider that SN and DCs are fully connected, and the capacity of different links varies within the range [0.5 Gb/s: 10 Gb/s].

To measure the efficacy of the proposed EEBA method, we focus on measuring the three time-based performance metrics, namely, workload makespan, VM makespan, and host makespan. Moreover, the newly proposed network overhead (NOV) parameter (see Section III) is used to measure the degree of the WAN communication network congestion. Consequently, nine BoT clusters are created (as shown in Table 7 and 8). Each cluster workload is built synthetically using a random uniform distribution generator to consider cloud user applications. The generated clusters contain different tasks types varying in the range of [1000:6000] as a task length (MI), and in the range of [0.05:1000] as a task data file size (MB). To study the importance of EEBA model and its effect on data transmission delay between the SN and the geo-distributed DCs, the data sizes of each generated workload (BoT) is considered as the following three types:

1- Simple BoT: Bag-of-tasks with small size data
2- Mixed BoT: Bag-of-tasks for a mixed size data
3- Heavy BoT: Bag-of-tasks for data-intensive

Tasks with large data size traces are expected to show the importance of the EEBA method in data placement and DCs selection that achieves higher improvement in makespan. For the plotted results of the genetic-based algorithms (i.e., LAEE, GGA, and EEBA) we plot the average of 25 independent executions in order to avoid the effect of the initial random population.

TABLE 4 AVERAGE DISTANCE AND LATENCY BETWEEN THE SN AND THE SELECTED DCS

DC	dc1	dc2	dc3	dc4
Average Distance (miles)	10500	6500	2200	8400
Average Latency (milliseconds)	194	122	46	150

TABLE 5 HOST'S TYPE AND SPECIFICATIONS

Host's type	Specifications
Type 1	HP ProLiant ML110 G4 (1 x [Xeon 3040 1860 MHz, 2 cores], 16GB)
Type 2	HP ProLiant ML110 G5 (1 x [Xeon 3075 2660 MHz, 2 cores], 16GB)
Type 3	HP ProLiant ML110 G3 (1 x [Pentium D930 3000 MHz, 2 cores], 16GB)
Type 4	IBM server x3250 (1 x [Xeon X3470 2933 MHz, 4 cores], 32GB)
Type 5	IBM server x3250 (1 x [Xeon X3480 3067 MHz, 4 cores], 32GB)
Type 6	IBM server x3550 (2 x [Xeon X5670 2933 MHz, 6 cores], 48GB)
Type 7	IBM server x3550 (2 x [Xeon X5675 3067 MHz, 6 cores], 64GB)

TABLE 6 AMAZON EC2 VM(S) SPECIFICATIONS

VM instance Type	Cores	MIPS	RAM (MB)	Bandwidth (Mbps)	Storage (GB)
Extra Small	1	500	613	100	0.633
Small	1	1000	1740	100	1.7
Medium	1	1500	1740	100	0.85
Large	1	2000	870	100	3.75

TABLE 7 VM CLUSTER SPECIFICATIONS

Cluster Type	Number of VMs	Number of Cloudlets
Small	500	1000
Medium	1000	2000
Large	1500	3000

TABLE 8 THREE TYPES OF WORKLOAD SPECIFICATIONS WITH VARIABLE DATA SIZES

Simple BoT	Mixed BoT	Heavy BoT
[0.05:1] MB	33% [0.05:1] MB	[100:1000] MB
	33% [2:500] MB	
	33% [501:1000] MB	

Workload Makespan. The main objective of this experiment is to study the importance of combining data transfer time, network delay (Equations 2 and 3), and NOV (Equation 7) as factors when executing different types of BoT application. Figures 4a and 4b show the corresponding workload makespan results for the three different workload clusters under RR, SJF, GGA, LAEE, and EEBA using three different BoT workload type (Simple, Mixed, Heavy).

As shown in the figures (Figures 4a and 4b) EEBA algorithm significantly outperforms other competing algorithms in achieving high cloud QoS with approximate 11%, 38%, and 15% rate of makespan enhancement using Simple, Mixed, and Heavy BoT respectively.

The makespan improvement as shown in Figures 4a and 4b varies in its rates according to workload cluster type (see Table 8). It is shown that the EEBA genetic algorithm achieves significant improvements in makespan in case of the Mixed BoT (up to 38%). On the other hand, the improvement is limited to up 11% and 15% compared to other competing algorithms Heavy BoT respectively. The following illustrates precisely the importance of the EEBA model using different workload specifications (as shown in Table 8) carried out in homogeneous and heterogeneous cloud environment of Type 1 and types: Type 1-7 respectively as shown in Table 5:

1- In the case of Simple BoT that reflects the small size data applications, we observe that the data transfer time and the NOV incurred due to WAN communication network congestion is negligible compared to the total BoT makespan time. That is why the improvement rate is limited to 11% compared to other competing algorithms.
2- In the case of Mixed BoT that reflects the mixed size data application, we observe the main efficacy of EEBA algorithm and its high importance in achieving high improvement in makespan compared to other competing ones. The importance of considering the WAN communication network link BW is clearly shown in this experimental part and how the EEBA model to direct the data-intensive tasks to high available bandwidth DCs without ignoring the importance of data transfer time and links delay. Moreover, although LAEE algorithm takes into its consideration the data transfer time and the links delay [13], however, and as shown in Fig. 4, EEBA outperforms the LAEE with an average rate of 20% due to the importance of considering the WAN communication network link bandwidth to minimize the NOV (using Equation 7). The figure also shows the unstableness of the SJF algorithm using this type of applications.
3- In the case of Heavy BoT that reflects the data-intensive application type, the EEBA algorithm outperform the competing non-location aware (GGA, SJF, and RR) with an average makespan improvement rate of 18%. However, the improvement is limited to 10% compared to LAEE that is data transfer and link delay aware algorithm.

As a conclusion, EEBA model guarantees a perfect makespan of an application in different cases and contributes with an approximate 21% rate of makespan improvement (using different scenarios and environments).

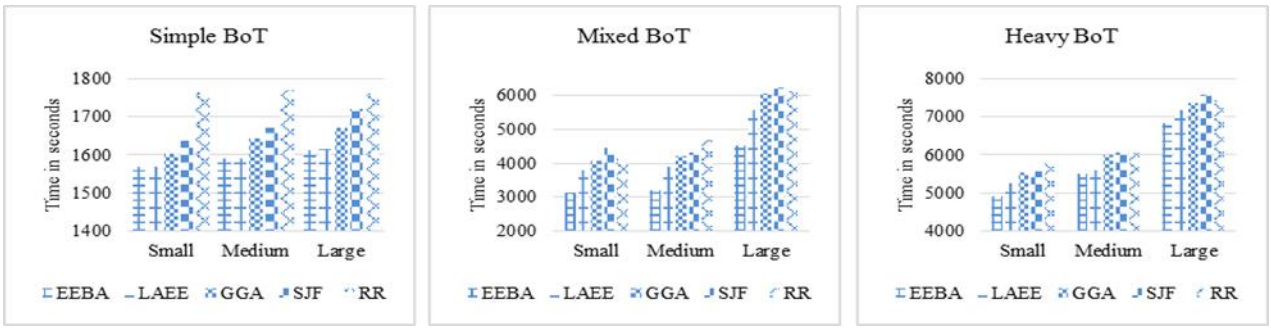

Fig. 4a. Workload Makespan in different number of cloudlets and VMs –Homogenous environment (Type 1 as shown in Table 5)

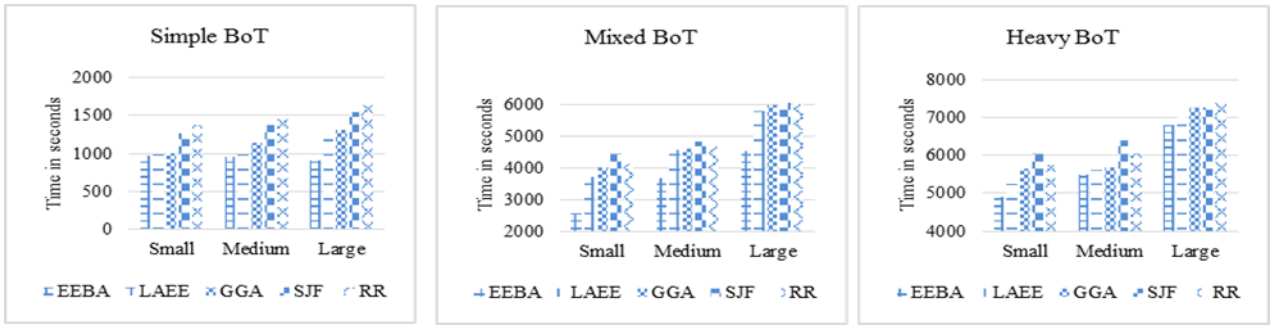

Fig. 4b. Workload Makespan in different number of cloudlets and VMs –Heterogeneous environment (types: Type 1-7 as shown in Table 5)

VM Makespan. Fig. 5 shows the comparison results of the average of VMs makespan where 10 VMs are selected randomly using the average of three different cluster types (as shown in Table 8) conducted on homogeneous as well as a heterogeneous cloud environment. This metric is an indicator that reflects the importance of the EEBA allocation algorithm to improve the utilization of the available VMs. Fig. 5 shows that the EEBA algorithm has approximately a uniform VM makespan among all available VMs compared to RR, SJF, GGA, and LAEE using the average of different workload type as shown in Table 8. This reflects the success of the EEBA algorithm to balance the workload on the available VMs, which lead to the workload makespan improvement seen in Fig. 4 above.

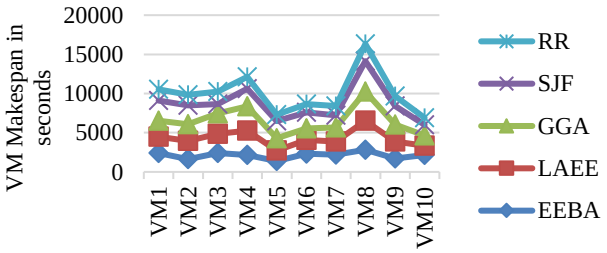

Fig. 5. VM Makespan: Comparing our proposed EEBA algorithm with RR, SJF, GGA and LAEE algorithms.

Host makespan. The Host makespan (calculated using Equation 5) is a good indication of the degree of loading balancing of the allocation algorithm. Fig. 6 shows the rate of improvement in host makespan using the average of the three workload types given in Table 8 conducted on homogeneous as well as heterogeneous cloud environment using 10 hosts selected at random from all available geo-distributed DCs.

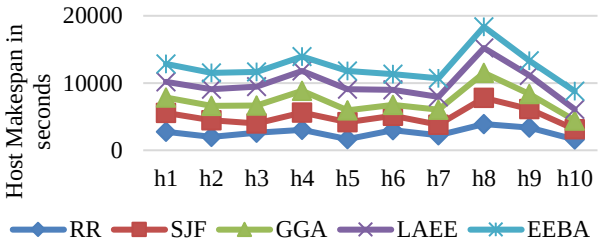

Fig. 6: Host Makespan: Comparing our proposed EEBAG algorithm with RR, SJF, GGA and LAEE algorithms.

The results show the importance of the proposed EEBA in balancing the tasks of the BoT workload among the available hosts that hold a number of VMs, which helps in satisfying the deadline constraint.

Network Overhead (NOV): The NOV metric measures the percentage of network congestion throughout the BoT execution time (as shown in Equation 7), is a crucial indication about the efficacy of EEBA model in achieving high QoS and low SLA violations. Fig. 7 shows that the EEBA model contributes with approximate of 55% of NOV improvement compared to LAEE GGA, SJF and RR using homogeneous and heterogeneous hosts with two different VM cluster type (Mixed and Heavy). On the other hand, using the Simple BoT workload type the NOV is below 1% with respect to all algorithms including EEBA. Fig. 7 reveals that the network links are more congested when executing Heavy BoT workload type, with large data sizes. Accordingly, the makespan achievements of the EEBA model (as shown in Fig. 4), is the result of the model efficacy in reducing the NOV and network congestion as plotted in Fig. 7.

TABLE 9 HP & IBM SERVERS HOST LOAD TO ENERGY (WATT) MAPPING TABLE

Server type	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP G5	93.7	97	101	105	110	116	121	125	129	133	135
HP G3	105	112	118	125	131	137	147	153	157	164	169
IBM x3470	41.6	46.7	52.3	57.9	65.4	73	80.7	89.5	99.6	105	113
IBM x3480	42.3	46.7	49.7	55.4	61.8	69.3	76.1	87	96.1	106	113
IBM x5670	66	107	120	131	143	156	173	191	211	229	247
IBM x5675	58.4	98	109	118	128	140	153	170	189	205	222

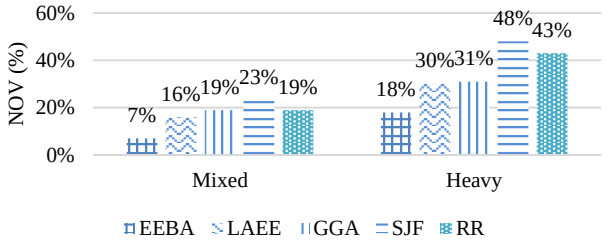

Fig. 7: Network overhead (NOV): Comparison graph among RR, SJF, GGA, and LAEE versus proposed EEBA algorithm

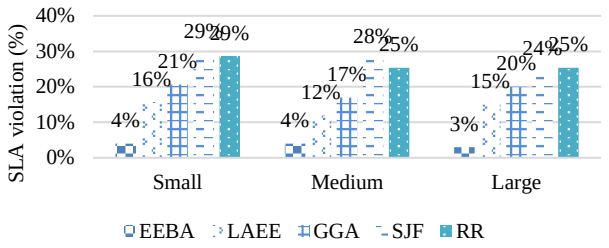

Fig. 8: BoT SLA deadline violation in a different number of cloudlets and VMs

SLA violation: SLA violation due to exceeding the deadline of the workload incurs penalty paid by the cloud provider to compensate users. Moreover, it reduces user satisfaction and degrades cloud providers' QoS. The authors in [20, 21] estimated that a delay for one-second could result in a 16% degradation in customer satisfaction and more than 22% drop in cloud services sales.

Fig. 8 shows the importance of the EEBA algorithm in achieving the users' time constraint with a minimum SLA violation compared to other competing algorithms. It shown that EEBA guarantees the least number of SLA violations with less than 6% compared to actual time constraint set by cloud users'. The total improvement of EEBA algorithm is around 80% over the non-location-aware competing algorithms (GGA, SJF, and RR). However, EEBA model contributes to around 60% improvement in SLA violation over the location and network-aware LAEE algorithm that ignores the network link bandwidths, which lead to more network congestion shown as higher NOV percentage (as shown in Fig. 7).

C.2. The Second Scenario

This scenario measures the effectiveness of the proposed EEBA model in optimizing the power consumption of the provider's DCs, which has a direct impact on leveraging the revenue of the cloud providers and pave the way to green computing. We combined our method with the Dynamic Voltage and Frequency Scaling (DVFS) technique [22] that contributes to the overall reduction of energy by adjusting the processor/core working frequencies whenever the utilization is low.

The experiment was conducted using real Planetlab workload traces [17]. The selected workload is made up of 287,794 cloudlets with different specifications. To emulate the cloud environment, we build a simulation setup made up of 800 servers distributed among 4 DCs to run 800 heterogeneous VMs with Amazon specifications shown in Table 6. However, hosts are considered heterogeneous of type 1-7 as shown in Table 5. According to the linear power model in Equation (1), and the real data from the SPECpower benchmark [23], Table 9 presents the host's power consumption at different load levels. Each WAN link (between SN and distributed DCs) is emulated using one physical link with a bandwidth capacity randomly generated in the interval [1Gb/s, 10Gb/s]. To emulate the WAN delay environment, the delays are generated using the geometric distances between the SN and the 4 distributed DCs as used in [13] and as shown in Table 4. The results shown in Fig. 8 are calculated from one-day simulation time.

Fig. 9 shows the energy consumption improvements when a cloud provider's broker employs the proposed EEBA energy-efficient task allocation method instead of non-power aware algorithms such as RR and SJF. As the figure shows, the average power saving improvement rate of EEBA algorithm is about 14% over the benchmarks algorithms RR and SJF, 9% over the GGA genetic algorithm, and 3% over LAEE algorithm. Moreover, Fig. 9 shows the importance of combining the DVFS technique to task allocation model that leads to high-energy efficiency compared to non-power aware task allocation model (NPA) that does not incorporate the DVFS technique and ignore its importance.

Overall, Fig. 9 prove the importance of load balancing that achieved using EEBA model in reducing the energy consumption considering the minimization of the network links overhead besides the workload makespan over the other competing algorithms as shown in the previous figures.

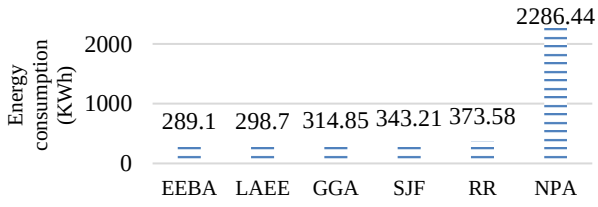

Fig. 9. Comparing the DCs Energy Consumption of the proposed EEBA algorithm with LAEE, GGA, SJF, RR, and NPA algorithms.

Genetic Convergence. Solving an optimization problem using GA requires a critical issue called GA convergence. Crossover and mutation are important operators to produce the diversity needed and avoid suboptimal solutions [19]. Consequently, suboptimal solutions occur due to premature convergence, which means that the GA converges quickly. Therefore, tuning GA parameters is an important concern to get an optimum solution.

Fig. 10 shows that EEBA characterized by good stability and high convergence. The influence of good use and tuning of GA operators reflects the result, which reveals that up to almost 43% of EEBA processing time is saved without affecting the model accuracy.

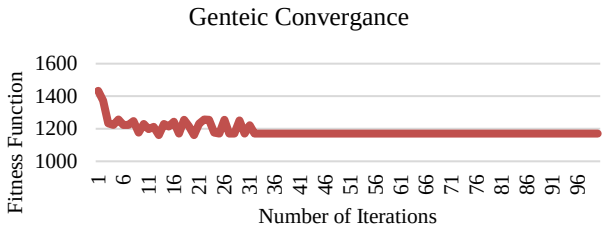

Fig. 10. Genetic Convergence

The above experimental results lead us to the following conclusions:

1- The performance of the EEBA model (as shown in Fig. 4) is highly depends on the generated workload type, i.e. Simple, Mixed, or Heavy. Moreover, incorporating NOV as a factor in data-intensive BoT workload allocation NP problem over geo-distributed DCs has high efficacy in minimizing network link congestion between the SN and geo-distributed DCs as shown in Fig. 7.

2- Task allocation algorithm is essential to achieving load balancing and system efficiency. LAEE and EEBA task allocation models proved their efficacy among the competing algorithms on achieving their objectives in optimizing the cloud QoS through minimizing the workload makespan (as shown in Fig. 4) as well as the DCs energy consumption (Fig. 9). For instance, and since we target the problem of data-intensive workload that specified with high data transfer, NOV should be taken into account to avoid the congested network that leads to QoS degradation. That is why EEBA is more suitable for data-intensive workload type resulting in low SLA violation as shown in Fig. 8.

3- Since we are targeting the data-intensive applications that require high data transfer time compared to its execution time, incorporating the DVFS technique is crucial in achieving high-energy efficiency (as shown in Fig. 9). However, one cannot ignore the importance of high-performance EEBA model that incorporates the energy factor (Equation 9) in achieving high-energy efficiency compared to other competing algorithms (Fig. 9). Moreover, the above experimental results show the efficacy of load balancing achieved using EEBA and LAEE models in extra contribution to energy saving compared to other non-energy aware algorithms.

VII. Conclusion

This paper presented the EEBA method, an Energy-Efficient and Bandwidth-Aware workload allocation method for data-intensive applications in geo-distributed DCs. Our method combined energy consumption, workload makespan and WAN network overhead to fulfill a better QoS for cloud users. We formulated the allocation problem as a multi-objective optimization problem, which is an NP-hard problem. We proposed a meta-heuristic genetic algorithm to find a near-optimal solution to the problem. Extensive simulations are conducted using both real and synthetic workload traces. Our experimental results showed that the proposed EEBA method improved the workload makespan and QoS by respecting the workload deadline constraint. Also, the EEBA adaptive genetic algorithm contributed to reducing the energy consumption of the cloud DCs due to load balancing the workload tasks as well as reducing the communication network links delay due to the proposed network overhead metric (NOV).

References

[1] Welsh, Thomas, and Elhadj Benkhelifa. "On Resilience in Cloud Computing: A survey of techniques across the Cloud Domain." ACM Computing Surveys (CSUR) 53, no. 3 (2020): 1-36.

[2] Rawas, Soha, Wassim Itani, Ali Zaart, and Ahmed Zekri. "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, 2015.

[3] Rawas, Soha, Ahmed Zekri, and Ali El Zaart. "Power and Cost-aware Virtual Machine Placement in Geo-distributed Data Centers." In CLOSER, pp. 112-123. 2018.

[4] Aburukba, Raafat O., Mazin AliKarrar, Taha Landolsi, and Khaled El-Fakih. "Scheduling Internet of Things requests to minimize latency in hybrid Fog–Cloud computing." Future Generation Computer Systems 111 (2020): 539-551.

[5] Banerjee, Sourav, Mainak Adhikari, Sukhendu Kar, and Utpal Biswas. "Development and analysis of a new cloudlet allocation strategy for QoS improvement in cloud." Arabian Journal for Science and Engineering 40, no. 5 (2015): 1409-1425.

[6] Dong, Ziqian, Ning Liu, and Roberto Rojas-Cessa. "Greedy scheduling of tasks with time constraints for energy-efficient cloud-computing data centers." Journal of Cloud Computing 4, no. 1 (2015): 1-14.

[7] Chatterjee, Tamojit, Varun Kumar Ojha, Mainak Adhikari, Sourav Banerjee, Utpal Biswas, and Václav Snášel. "Design and implementation of an improved datacenter broker policy to improve the QoS of a cloud." In Proceedings of the Fifth International Conference on Innovations in Bio-Inspired Computing and Applications IBICA 2014, pp. 281-290. Springer, Cham, 2014.

[8] Huai, Weicheng, Zhuzhong Qian, Xin Li, Gangyi Luo, and Sanglu Lu. "Energy Aware Task Scheduling in Data Centers." J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl. 4, no. 2 (2013): 18-38.

[9] Kliazovich, Dzmitry, Sisay T. Arzo, Fabrizio Granelli, Pascal Bouvry, and Samee Ullah Khan. "e-STAB: Energy-efficient scheduling for cloud computing applications with traffic load balancing." In 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, pp. 7-13. IEEE, 2013.

[10] Alizadeh, Mohammad, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam et al. "CONGA: Distributed congestion-aware load balancing for datacenters." In Proceedings of the 2014 ACM conference on SIGCOMM, pp. 503-514. 2014.

[11] Toosi, Adel Nadjaran, Richard O. Sinnott, and Rajkumar Buyya. "Resource provisioning for data-intensive applications with deadline constraints on hybrid clouds using Aneka." Future Generation Computer Systems 79 (2018): 765-775.

[12] Abdi, Somayeh, Latif PourKarimi, Mahmood Ahmadi, and Farzad Zargari. "Cost minimization for deadline-constrained bag-of-tasks applications in federated hybrid clouds." Future Generation Computer Systems 71 (2017): 113-128.

[13] Rawas, Soha, and Ahmed Zekri. "Location-Aware Energy-Efficient Workload Allocation in Geo Distributed Cloud Environment." J. Comput. Sci. 14, no. 3 (2018): 334-350.
[14] Rawas, Soha. "Energy, network, and application-aware virtual machine placement model in SDN-enabled large scale cloud data centers." Multimedia Tools and Applications 80, no. 10 (2021): 15541-15562.Bienstock, Daniel, and George Nemhauser. Integer programming and combinatorial optimization. Springer International Publishing, 2020.
[15] http://wintelguy.com/wanlat.html. Accessed December 2018.
[16] Planet lab traces, https://www.planet-lab.org, Accessed December 2018.
[17] Kumar, Dilip, Bibhudatta Sahoo, Bhaskar Mondal, and Tarni Mandal. "A genetic algorithmic approach for energy efficient task consolidation in cloud computing." International Journal of Computer Applications 118, no. 2 (2015): 1-6.
[18] Kołodziej, Joanna, Samee Ullah Khan, Lizhe Wang, and Albert Y. Zomaya. "Energy efficient genetic-based schedulers in computational grids." Concurrency and Computation: Practice and Experience 27, no. 4 (2015): 809-829.
[19] Bilal, Kashif, Osman Khalid, Aiman Erbad, and Samee U. Khan. "Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers." Computer Networks 130 (2018): 94-120.
[20] Rawas, Soha, Ahmed Zekri, and Ali El Zaart. "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, 2018.
[21] Garg, Ritu, and Mamta Mittal. "Reliability and energy efficient workflow scheduling in cloud environment." Cluster Computing 22, no. 4 (2019): 1283-1297.
[22] https://www.spec.org/benchmarks.html, Accessed December 2018.
[23] Calheiros, Rodrigo N., Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms." Software: Practice and experience 41, no. 1 (2011): 23-50.
[24] Rawas, Soha, and Ahmed Zekri. "CAEE: Communication-Aware, Energy-Efficient VM placement Model for Multi-Tier Applications in Large Scale Cloud Data Centers." BAU Journal-Science and Technology 2, no. 1 (2020): 11.
[25] Hua-Ping Wu, Hui Li, and Xiao-Lan Sun, "Evolutionary Game for Enterprise Cloud Accounting Resource Sharing Behavior Based on the Cloud Sharing Platform," IAENG International Journal of Applied Mathematics, vol. 51, no.1, pp125-132, 2021
[26] Bindu, GB Hima, K. Ramani, and C. Shoba Bindu, "Optimized resource scheduling using the meta heuristic algorithm in cloud computing," IAENG International Journal of Computer Science, vol. 47, no. 3, pp 360-366, 2020.