sensors

Article

Improving Quality-of-Service in Cloud/Fog Computing through Efficient Resource Allocation †

Samson Busuyi Akintoye * and Antoine Bagula

ISAT Laboratory, Department of Computer Science, University of the Western Cape, Bellville 7535, South Africa; abagula@uwc.ac.za

* Correspondence: 3515640@myuwc.ac.za

† This paper is an extension version of the conference paper: Akintoye, S.B.; Bagula, A. Optimization of virtual resources allocation in cloud computing environment. In Proceedings of the IEEE AFRICON, Cape Town, South Africa, 18–20 September 2017.

Received: 11 January 2019; Accepted: 9 February 2019; Published: 13 March 2019

Abstract: Recently, a massive migration of enterprise applications to the cloud has been recorded in the IT world. One of the challenges of cloud computing is Quality-of-Service management, which includes the adoption of appropriate methods for allocating cloud-user applications to virtual resources, and virtual resources to the physical resources. The effective allocation of resources in cloud data centers is also one of the vital optimization problems in cloud computing, particularly when the cloud service infrastructures are built by lightweight computing devices. In this paper, we formulate and present the task allocation and virtual machine placement problems in a single cloud/fog computing environment, and propose a task allocation algorithmic solution and a Genetic Algorithm Based Virtual Machine Placement as solutions for the task allocation and virtual machine placement problem models. Finally, the experiments are carried out and the results show that the proposed solutions improve Quality-of-Service in the cloud/fog computing environment in terms of the allocation cost.

Keywords: CloudSim; virtual machine; greedy heuristics; cloud computing; fog computing; genetic algorithm; data center; CloudSim and hungarian algorithm

1. Introduction

Cloud computing is a computing model to provide on-demand network access to a large pool of networking, storage and computing resources over the internet [1]. This type of computing provides cost reduction because customers do not need to procure hardware for their operations, rather they subscribe for computing resources from the Cloud Service Provider (CSP) only when the cloud services are needed and also only pay for services they consume. Basically, cloud computing is grouped into four deployment models: Private cloud, community cloud, hybrid cloud and public cloud [1,2]. A private cloud is solely owned and managed by an individual organization. In the community cloud model, the cloud infrastructures are owned and shared by various organizations and supports a specific community that has similar operations. The hybrid cloud is a combination of two or more clouds such as public, private and community that remain distinctive entities but are joined together by uniform technology that enables data and applications to be moved easily. In the public cloud model, the cloud infrastructures are made available to the public on a pay-as-you-use basis by the CSP. Broadly speaking, cloud computing is divided into three service models as follows: Platform-as-a-Service (PaaS), Software-as-a-Service (SaaS) and Infrastructure-as-a-Service (IaaS). PaaS allows users to rent virtualized platforms on which to run their own applications or services [3]. In the SaaS model, the cloud customers use the provider's applications running on a cloud infrastructure.

In IaaS, customers are provided with CPU, network bandwidth, storage and other computing resources, all of which they are subsequently able to reconfigure as needed. Invariably, the cloud services are offered to cloud customers and users through the concept of virtualization and distributed computing. Virtualization is an abstraction of computing resources such as storage and processing power to provide virtualized resources called virtual machines (VMs) for customer's applications [4]. Virtualization technology can be classfied into four groups, namely: Full virtualization, para-virtualization, native virtualization and operating system-level virtualization. Full virtualization provides a full image of the essential hardware. Para-virtualization modifies a guest operating system and is able to communicate directly with the hypervisor [5]. In native virtualization, the guest operating system and host use the same hardware. Operating system-level virtualization does not depend on a hypervisor, rather it modifies the operating system to isolate securely many instances of an operating system within a single host machine. In cloud computing, virtualization technologies such as network, storage and compute offer users an abstraction layer (i.e., VM) that provides a uniform computing platform by concealing the underlying hardware heterogeneity, internal management difficulty and geographic boundaries [4]. When the cloud receives a task request from a user, either a new VM is initialized or an existing VM of the same user is assigned to the request [2]. After the task is processed successfully, every assigned resource is released to the pool of free resources. Sometimes, the number of the available physical resources may not be commensurate with the number of task requests from the cloud user, then resource allocation becomes a critical problem which needs to be solved by CSPs. In such cases, the CSP has the responsibility to determine the number of VMs to be initiated, based on the number of requests from cloud users. The problem of resource allocation in cloud computing involves assigning tasks to VMs and the placement of VMs on Physical Machines (PMs). The computing resource provided is normally allocated based on the Service Level Agreement (SLA) contract between the cloud customers and service providers. The SLA spells out the details regarding the quality of service (QoS) to be offered by the CSP in terms of a range of performance parameters such as reliability, response time, and throughput. The SLA may also specify the payment process and the breach of SLA contract penalty [6]. Thus, the ultimate goal of the CSP is to optimize resource utilization and maximize profit while that of the cloud user is to ensure the cost of leasing the resources is minimized.

1.1. Cloud/Fog Computing Resource Management Framework

The resource management framework in Figure 1 summarizes the work described in this paper. The multi-layer framework includes:

1. A physical resource layer which is composed of data centres that host PMs in the form of host machines. The PMs are interconnected by switches (SWs).
2. A virtual resource layer which lies above the physical resource layer to virtualize the physical resources as VMs for better resource management.
3. An application layer which lies above the virtual resource layer to provide a variety of services to users. These include SaaS, PaaS and IaaS.

When considered from a service perspective, the framework in Figure 1 can be presented as a two-layer architecture including:

1. A virtual resource scheduling module, where a mapping between physical machines and the virtual machines is made. We assume in this paper that each physical machine (host machine) provides at least one virtual machine.
2. A task allocation management module enabling the virtual resources to be allocated to the users in a cost effective way.

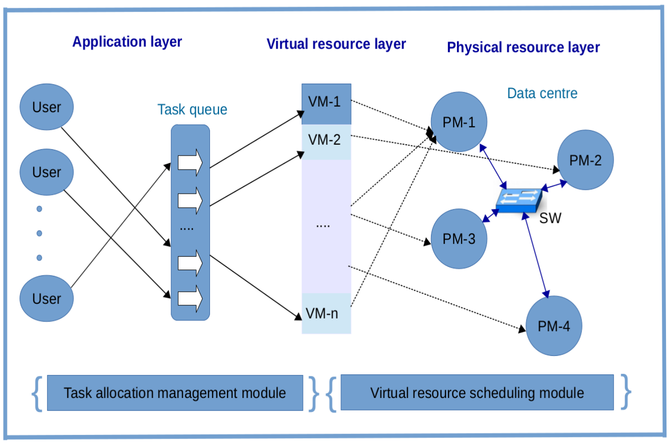


Figure 1. cloud/fog computing resource management framework.

1.2. Contributions

This paper proposes an implementation of the framework by assuming that: (i) The tasks and virtual resources are varied and the physical resources are uniform, and (ii) the number of on-demand requests initiated by cloud users is higher than the number of available resources. In this work, we propose a model for assigning the tasks (cloudlets) to VMs, and VM placement with the aim of improving quality of service in the cloud/fog computing environment. The main contributions of this paper are outlined as follows:

- **Problem formulation**: The task allocation and VM placement problem models in the cloud computing environment are formulated and presented. These models aim to minimize the resource allocation cost in a setting where multiple cloud user requests have to be processed on a limited number of physical resources.
- **A task assignment strategy**: We propose the Hungarian Algorithm Based Binding Policy (HABBP) as a heuristic solution to the linear programming problem, and use the algorithm to implement a novel assignment strategy for the famous CloudSim simulator. We also propose the assignment strategy module as a contributed module to CloudSim which includes: (i) A graphical user interface as a front-end component which enables cloud users to interact and communicate with CloudSim and to configure the tasks, VM and PM parameters from the interface, rather than embedding parameter values in the CloudSim source code and (ii) a novel assignment strategy as a back-end component.
- **VMs placement solution**: We propose a Genetic Algorithm Based Virtual Machine Placement (GABVMP) to solve and optimize the VM placement problem in the cloud computing environment.
- **Analysis of experimental results**: We evaluate and compare the performance of the proposed binding policy with the conventional binding policy implemented by the CloudSim simulator and benchmark both solutions against the Simplex algorithm commonly used as a linear programming solver. The proposed GABVMP solution is also compared with the greedy heuristics: Random Placement and First Fit Placement.

1.3. Paper Organization

The rest of the paper is arranged as follows: Section 2 presents existing works related to the resource allocation problem in cloud computing. The linear programming model for task allocation is proposed in Section 3. Section 4 presents a task allocation algorithmic solution as a solution for the optimization of the task allocation problem model. Section 5 describes the VM placement problem. In Section 6, the VM placement problem is solved using GABVMP. Section 7 describes the implementation of HABBP and GABVMP, for task allocation and VM placement, respectively. Lastly, we conclude our paper with Section 8.

2. Related Work

In cloud/fog computing, resource allocation is the process of assigning available resources to the needed cloud applications over the internet. These resources are allocated based on cloud user request and pay-per-use method. Resources in cloud computing could be either virtual resources or physical resources. Cloud service providers must effectively manage, provide, and allocate these resources to provide services to cloud consumers based on service level agreements (SLAs). Therefore, the appropriate allocation of resources in cloud data centers is also one of the important optimization problems in cloud computing especially when the cloud infrastructure is made of lightweight computing devices.

The quality of service in cloud/fog computing is based on its resource allocation process, and the cloud service provider should assign the resource to the cloud users in an optimal way. The result of any optimal resource allocation strategies must consider certain parameters such as latency, throughput, reduction of energy consumption, minimization of allocation cost and response time. There are many existing works relating to resource allocation in cloud/fog computing. Maguluri et al. [7] propose a stochastic model for resource allocation in cloud computing in which jobs arrive according to a stochastic process and request a variety of virtual machines. The authors use a non-pre-emptive for load balance among the cloud servers and to schedule VM configurations. In order to minimize the communication complexity, the authors consider a distributed system such that each server maintains its own queues. The experimental evaluations reveal that there is only a small difference in delay performance between distributed and centralized queueing systems. Furthermore, the evaluations show that the non-pre-emptive algorithm adopted in this work outperforms the best-fit scheduling algorithm in terms of throughput. Baker et al. [8] present a requirements model for the runtime execution and control of an intention-oriented Cloud-Based Application. The requirements modelling process known as Provision, Assurance and Auditing, and an associated framework are defined and developed where a given system's functional and non-functional requirements are modelled in terms of intentions and encoded in a standard open mark-up language. An autonomic intention-oriented programming model, using the Neptune language, then handles its deployment and execution. Al-khafajiy et al. [9] propose a fog computing architecture and framework to improve QoS through the request offloading method. The proposed method uses a collaboration strategy among fog nodes in order to permit data processing in a shared mode which satisfies QoS and serves the largest number of IoT requests. The experimental result shows that the performance of fogs layer is significantly increased when the overload is distributed over several fog nodes.

In [10] the author investigates existing resource scheduling algorithms, and classfies them according to some determining factors, such as cost, energy and time. The advantage of the study is that it helps CSPs in the adoption of appropriate scheduling algorithms based on their ultimate goals. Liu et al. [11] propose an earliest finish time duplication algorithm to schedule multiple tasks in heterogeneous data centres. The algorithm can also be referred to as a directed acyclic graph based scheduling algorithm. The performance evaluation of the study reveals that the combination of pre-processing the cloud resources before scheduling and the proposed algorithm, performs better than the heterogeneous earliest finish time algorithms, in terms of task scheduling time. In [12] the authors propose a virtual cloud resource allocation model based on constraint programming to improve the Quality-of-Service (QoS) in cloud computing and decrease the cost of resource utilization. Moreover, the authors [13] propose a VM Repacking Scheduling Problem (VRSP) to minimise the energy consumption while placing VM in the data centres. The benefit of the study is that it is flexible, it enables users to generate automatically the SLA constraints, and it reduces energy utilization.

In order to address the VM placement problem in a data centre, the authors [14] propose a greedy-based algorithm to reduce resource usage, the network traffic and the number of cloud servers. The work divides traffic flows and routes them through two link-disjoint paths to decrease congestion, at the same time meeting the requirements for protection grade as well as bandwidth. Furthermore, the authors [15] propose an online heuristic-based VM placement algorithm which is based on a multi-dimensional space partition model. The objective of the work is to make a trade-off between balancing multi-dimensional resource usage and reducing the number of the PMs used for VM placement. The advantage of the algorithm is that it reduces the number of running PMs as well as the total energy consumption. In [16], authors propose an ant-colony based optimization model with the aim to optimize resource utilization and total power consumption concurrently. The model performs better than the previous multi-objective VM placement algorithm. Pascual et al. [17] propose multi-objective evolutionary algorithms to solve the placement problem. The objectives of work are: (i) The consolidation of VMs on a small set of processors, and (ii) the minimization of associated energy costs for servers and network equipment. The algorithms were implemented using a Flat Tree topology and tiered applications, such as a web server with an associated database. The major advantage of the algorithms is that they enhance the application performance and energy consumption. The work in [18] proposes algorithms for the placement of precedence-constrained parallel virtual machines. The aim of the work is to reduce energy consumption by consolidating virtual machines on the available physical machines yet not degrading the makespan. The algorithms were evaluated using benchmarks of real-world distributed applications and they achieved efficient results.

Georgiou et al. [19] propose VM placement algorithms for the Portland network architecture with the aim to allocate communicating virtual machines in physical proximity to avoid the creation of network bottlenecks. The authors propose two algorithms: the first algorithm is proposed for rapid placement of closely located virtual machines, while the second algorithm is designed to identify network regions that can best host the virtual machines and then, using the first algorithm, maps these virtual machines on the servers. The benefit of the approach is that it has the capability to reduce the intensity of traffic in the links of top-level switches.

Meng et al. [20] propose a Cluster-and-Cut algorithm to improve the scalability of data center networks with traffic-aware VM placement. The goal of the algorithm is to reduce network traffic among VMs and related communication cost by placing inter-communicating VMs in the same PM. The VM placement problem is formulated as a quadratic assignment problem (QAP) to find a suboptimal placement which minimizes network traffic, considering the associated communication cost and a static-single path routing. The allocation cost is defined as the number of switches between two inter-communicating VMs and each PM is divided by slots with the capacity to accommodate a single VM with the assumption of an equal number of VMs and slots. If the number of VMs is lower than the number of slots, dummy VMs are introduced with zero traffic which has no significant effect on the solution of the problem. The performance evaluations of the algorithm show a significant performance improvement compared to existing genetic algorithmic methods.

Breitgand et al. [21] investigate the problem of placing images and VM instances on the servers with the aim to increase the affinity between them to mitigate communication overhead and latency. The problem is modelled as an extension of the Class Constrained Multiple Knapsack problems (CCMK) and present a polynomial time local search algorithm for the same size images. Specifically, this model focuses on an off-line placement problem, where there are a given set of demands and available servers. In order to solve this problem, the local search algorithm was applied as a basis for ongoing optimization which periodically improves the VM placement and greedy placement of a new set of VM instances by allowing migrations of the VMs.

Vakilinia et al. [22] propose a platform for virtual machine (VM) placement/migration to minimize the total power consumption of cloud data centers (DCs). The platform is divided into two parts. Firstly, an estimation module is introduced to predict the incoming load of the DC. Secondly, two schedulers are designed to determine the optimal assignment of VMs to the PMs. The proposed schedulers apply a column generation method to solve the large-scale optimization problem in conjunction with the cut-and-solve-based algorithm and the call back method to decrease the complexity and the time to obtain the optimal solution. The trade-off between optimality and time is investigated. The numerical results show that the proposed platform produces the optimal solution for a limited time-frame. Selmy et al. [23] present virtual machines migration and selection policies to reduce the power consumption of servers in the cloud computing environment. The authors propose neural networks for classification and prediction, Self Organizing Map (SOM) and K-Means Clustering algorithms for the policies. The results of implementation of the proposed policies show significant reduction of energy consumption of the servers in the data center.

All the works mentioned above have been able to solve one or two problems of VM placement in the cloud computing environment. There is still much to be done, however, to mitigate the effect of these problems.

3. Task Allocation Problem Model

In this section, we present a linear programming problem model for assigning task requests (cloudlets) from cloud users to VMs. To express the model mathematically, we consider a set of VMs represented by vm_i for $i \leq n$ and $n > 1$. Similarly, a set of tasks (cloudlets) corresponding with each on-demand user request (job) represented by τ_j for $j \leq m$ and $m > 1$. In this work, we assume a one-to-one allocation model where each VM executes only one cloudlet and each cloudlet needs to be assigned to only one VM. However, a many-to-one allocation model may also be considered where several tasks are allocated to a single virtual machine. The many-to-one allocation model is not considered in this paper.

Furthermore, we set $n = m$ and $C = [\omega_{ij}]$ to be an $n \times n$ matrix in which ω_{ij} is the cost of assigning vm_i to $cloudlet_j$, i.e.,

$$\omega_{ij} = \frac{f_j}{vm_i}. \tag{1}$$

We also set $\chi = [\alpha_{ij}]$ to be the $n \times n$ matrix where

$$\alpha_{ij} = \begin{cases} 1, & \text{if } vm_i \text{ is assigned to } cloudlet_j, \\ 0, & \text{if } vm_i \text{ is not assigned to } cloudlet_j. \end{cases} \tag{2}$$

Our goal is to minimize the total cost $\rho(\chi)$, defined as the sum of the cost of assigning cloudlets to the available VMs. Thus, we present an optimization problem as a linear programming model in terms of a function ρ as follows:

$$\text{minimize } \rho(\chi) = \sum_{j=1}^{m}\sum_{i=1}^{n} \omega_{ij}\alpha_{ij} \tag{3}$$

subject to the following constraints

$$\sum_{i=1}^{n} \alpha_{ij} \quad = \quad 1, \text{ for j = 1, 2, \ldots, m,} \tag{4}$$

and

$$\sum_{j=1}^{m} \alpha_{ij} \quad = \quad 1, \text{ for i = 1, 2, \ldots, n,} \tag{5}$$

such that

$$\alpha_{ij} \quad = \quad 0 \text{ or } 1. \tag{6}$$

Thus, any matrix satisfying the Equations (4) and (5) is a solution and conforms to a permutation σ of a set $N = \{1, 2, \ldots, n\}$ generated by setting $\sigma(i) = j$ if and only if $\alpha_{ij} = 1$. In addition, if χ is a solution relating to σ, then

$$\sum_{j=1}^{n} \omega_{ij}\alpha_{ij} = \omega_{i\sigma(i)}. \tag{7}$$

summation over i from 1 to n, we obtain

$$\sum_{i=1}^{n} \omega_{i\sigma(i)} = \sum_{i=1}^{n}\sum_{j=1}^{n} \omega_{ij}\alpha_{ij}. \tag{8}$$

Hence, any solution χ on which $\rho(\chi)$ is minimal is known as an optimal solution. We can reform a given allocation problem specified by C into another one specified by a matrix $\overline{C} = [\overline{\omega}_{ij}]$, in which $\overline{\omega}_{ij} \geq 0, \forall$ pairs i, j, where the two problems have the equal set of optimal solutions. If χ^* is an optimal solution to the problem given by $\overline{C}$, then it is important to know that χ^* is also an optimal solution to the one given by C.

Theorem 1 illustrates the steps to reform a matrix into another with the same set of optimal solutions.

Theorem 1. *A solution X is an optimal solution for $p(X) = \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n} c_{ij}x_{ij}$ if and only if it is an optimal solution for $\overline{p}(X) = \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n} \overline{c}_{ij}x_{ij}$ where $\overline{c}_{ij} = c_{ij} - u_i - v_j$ for any of $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ and u_i and v_j are real numbers for all i and j.*

Proof. We establish that the difference between the functions $p(X)$ and $\overline{p}(X)$ is constant $\sum\limits_{i=1}^{n} u_i + \sum\limits_{j=1}^{n} v_j$.

$$\begin{aligned}
\overline{p}(X) &= \sum_{i=1}^{n}\sum_{j=1}^{n} \overline{c}_{ij}x_{ij}, \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} (c_{ij} - u_i - v_j)x_{ij}, \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n} u_i x_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n} v_j x_{ij}, \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n} u_i x_{ij} - \sum_{j=1}^{n}\sum_{i=1}^{n} v_j x_{ij}, \\
&= p(X) - \sum_{i=1}^{n} u_i \sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} v_j \sum_{i=1}^{n} x_{ij}.
\end{aligned}$$

From Equations (4) and (5),

$$= p(X) - \sum_{i=1}^{n} u_i - \sum_{j=1}^{n} v_j.$$

This shows that, $p(X) - \overline{p}(X) = \sum\limits_{i=1}^{n} u_i + \sum\limits_{j=1}^{n} v_j$. Therefore, a solution X minimizes $p(X)$ if and only if it minimizes $\overline{p}(X)$. □

4. Task Allocation Algorithmic Solution

In this section, we present a task allocation algorithmic solution that is based on Hungarian algorithm [24,25] known as Hungarian Algorithm Based Binding Policy (HABBP) to solve the task allocation problem in the cloud computing environment.

4.1. Notation and Preliminaries

- Given a cost-matrix ωm of size $n \times m$,
- n is the number of VMs,
- m is the number of cloudlets,

- ωm_{ij} represents the time required to complete $cloudlet_i$ by vm_j.

4.2. Procedures of the Algorithm

Algorithm 1 represents the pseudo-code of the HABBP for tasks-to-VMs allocation in a cloud computing environment. The first four lines (1–4) in the algorithm represent the initialization of different variables. Subsequently, we initialize *cost-matrix* by dividing the cloudlet length by the MIPS of VM. In the case of many-to-one allocations where the number of cloudlets and the number of VMs are not equal, we then add the dummy cloudlets/VMs to turn the *cost-matrix* into a square matrix.

Algorithm 1: Computation of the total assignment cost C.

```
input  : $n$: denotes the no. of VMs
         $m$: denotes the no. of cloudlets
         $\tau_i$ : a set of cloudlets $i \in [1, 2, \ldots, m]$
         $vm_j$ : a set of VMs $j \in [1, 2, \ldots, n]$
output: $C$: total assignment cost
1  /* Initialize and generate cost-matrix */
2  for $i \in [1, 2, \ldots, n]$ do
3      for $j \in [1, 2, \ldots, m]$ do
4          $\omega m_{ij} = \tau_i \ / \ vm_j$
5      end
6  end
7  if ($n$ not equal to $m$) then
8      add dummy cloudlets to make cost-matrix square matrix
9  end
10 $\eta r \leftarrow$ smallest Element in cost-matrix row
11 $\eta c \leftarrow$ smallest Element in cost-matrix column
12 /* calculates the reduced-cost-matrix */
13 for $j \in [1, 2, \ldots, n]$ do
14     $\omega m_{nj} = cm_{nj} - \eta r$
15 end
16 for $i \in [1, 2, \ldots, n]$ do
17     $\omega m_{ni} = \omega m_{ni} - \eta c$
18 end
19 /* calculate the line-cost-matrix */
20 $l_n \leftarrow$ minimum-number-line()
21 if $l_n < m$ then
22     for $j \in [1, 2, \ldots, n]$ do
23         for $j \in [1, 2, \ldots, n]$ do
24             if (element are uncovered) then
25                 $\omega m_{ij} \leftarrow (\omega m_{ij} - \min(uncoveredElement))$ elseif (element are covered by two
                   line) $\omega m_{ij} \leftarrow (\omega m_{ij} + \min(uncoveredElement))$
26             end
27         end
28     end
29 end
30 /* find the mapping */
31 apply the matching to the original matrix, discarding dummy rows.
32 the addition of the costs will give the total minimum cost
33 return total assignment cost $C$
```

Thereafter, we compute the *reduced-cost-matrix* from the *cost-matrix* by subtracting the minimum value of each row from the elements of its row, turning each minimum value into zero, and by subtracting the minimum value from the elements of each column, turning the minima into zeros. From that, we compute the *line-cost-matrix*. If the number of lines is not equal to the number of VMs, we then subtract the minimum uncovered element from every covered element. If an element is covered twice, we then add the minimum element to it.

Lastly, we apply the mapping to the original matrix, discarding dummy rows, and we add the cost of assigning cloudlets to VMs to give the total minimum cost C.

4.3. Illustration

We illustrate, through an example, the concept of the HABBP and the steps that need to be followed in order to optimize the assignment of tasks to virtual resources in the cloud computing environment. Table 1 depicts three cloudlets in the queue with a broker and Table 2 depicts VMs initiated in the data centre.

Table 1. Cloudlet details.

	$cloudlet_1$	$cloudlet_2$	$cloudlet_3$
id	0	1	2
file-size	500	1000	1000
length	40,000	80,000	120,000
output-size	500	2048	2048

Table 2. VM specifications.

	vm_1	vm_2	vm_3
id	0	1	2
size	1000	1000	1000
mips	400	1000	500
ram	2048	2048	2048
pes-number	1	2	2
bandwidth	500	500	500

The algorithm works as follows: We initialize the *cost-matrix* by dividing the *length* of the cloudlet by the *mips* of the VM as depicted in Table 3.

Table 3. Initialize *cost-matrix*.

	$cloudlet_1$	$cloudlet_2$	$cloudlet_3$
vm_1	100	200	300
vm_2	40	80	120
vm_3	80	160	240

In this case, the number of cloudlets is equal to the number of VMs. Thus, we do not need to add the dummy cloudlet/VM values to turn the *cost-matrix* into a square matrix. We compute the *reduced-cost-matrix* by subtracting the minimum value of each row and column from the row and column of the *cost-matrix* to yield Tables 4 and 5 respectively.

Table 4. Row *reduced-cost-matrix*.

	$cloudlet_1$	$cloudlet_2$	$cloudlet_3$
vm_1	0	100	200
vm_2	0	40	80
vm_3	0	80	160

Table 5. Column *reduced-cost-matrix*.

	$cloudlet_1$	$cloudlet_2$	$cloudlet_3$
vm_1	0	60	120
vm_2	0	0	0
vm_3	0	40	80

Then, we calculate the *line-cost-matrix*; this represents the lines that cover all zeros in the *reduced-cost-matrix*. In this case, there are two lines. The lines are on column 1 and row 2 of the *reduced-cost-matrix*. Since the number of lines is not the same as the number of VMs, we remove the lowest of all uncovered elements from all uncovered elements as indicated in Table 6.

Table 6. *reduced-cost-matrix*.

	$cloudlet_1$	$cloudlet_2$	$cloudlet_3$
vm_1	0	20	80
vm_2	0	0	0
vm_3	0	0	40

Again, we calculate the minimum number of lines required to cover all zeros in the matrix. The lines are on column 1, row 2 and row 3 of the *reduced-cost-matrix*. Since the number of lines is the same as the number of VMs, an optimal assignment exists among the zeros in the *reduced-cost-matrix*. Therefore, $cloudlet_1$ is assigned to vm_1, $cloudlet_3$ is assigned to vm_2, and $cloudlet_2$ is allocated to vm_3 as indicated in Table 7.

Table 7. Optimal Assignment.

Cloudlets	Virtual Machines
$cloudlet_1$	vm_1
$cloudlet_2$	vm_3
$cloudlet_3$	vm_2

The total cost of assigning cloudlets to virtual machines optimally is: 100 s + 120 s + 160 s = 380 s.

Alternatively, we solve the example mentioned above using Simplex method [26,27] and compare the result with the one that is already generated using HABBP. Using the Equations (3)–(6), the optimization objective function can be formulated as:

$$\text{minimize } \rho(\chi) = 100\alpha_{11} + 200\alpha_{12} + 300\alpha_{13} + 40\alpha_{21} + 80\alpha_{22} + 120\alpha_{23} + 80\alpha_{31} + 160\alpha_{32} + 40\alpha_{33} \quad (9)$$

subject to the following constraints

$$\begin{aligned} \alpha_{11} + \alpha_{12} + \alpha_{13} &= 1, \\ \alpha_{21} + \alpha_{22} + \alpha_{23} &= 1, \\ \alpha_{31} + \alpha_{32} + \alpha_{33} &= 1, \\ \alpha_{11} + \alpha_{21} + \alpha_{31} &= 1, \\ \alpha_{12} + \alpha_{22} + \alpha_{32} &= 1, \\ \alpha_{13} + \alpha_{23} + \alpha_{33} &= 1 \end{aligned} \quad (10)$$

and

$$\alpha_{11}, \alpha_{12}, \alpha_{13}, \alpha_{21}, \alpha_{22}, \alpha_{23}, \alpha_{31}, \alpha_{32}, \alpha_{33} \geqslant 0 \quad (11)$$

Since the objective function is in minimization form, then we convert it into maximization form and add the artificial variables as:

$$\text{maximize}\, \rho(\chi) = -100\alpha_{11} - 200\alpha_{12} - 300\alpha_{13} - 40\alpha_{21} - 80\alpha_{22} - 120\alpha_{23} - 80\alpha_{31} - 160\alpha_{32} - 40\alpha_{33} \quad (12)$$

$$\begin{aligned} \alpha_{11} + \alpha_{12} + \alpha_{13} + S_6 &= 1, \\ \alpha_{21} + \alpha_{22} + \alpha_{23} + S_5 &= 1, \\ \alpha_{31} + \alpha_{32} + \alpha_{33} + S_4 &= 1, \\ \alpha_{11} + \alpha_{21} + \alpha_{31} + S_3 &= 1, \\ \alpha_{12} + \alpha_{22} + \alpha_{32} + S_2 &= 1, \\ \alpha_{13} + \alpha_{23} + \alpha_{33} + S_1 &= 1 \end{aligned} \quad (13)$$

and

$$\alpha_{11}, \alpha_{12}, \alpha_{13}, \alpha_{21}, \alpha_{22}, \alpha_{23}, \alpha_{31}, \alpha_{32}, \alpha_{33}, S_1, S_2, S_3, S_4, S_5, S_6 \geqslant 0 \quad (14)$$

In Phase 1 of the two-phase simplex method, we remove the artificial variables and find an initial feasible solution of the original problem which gives the final Tableau in the Table 8.

Table 8. Phase 1 final Tableau.

Tableau			**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**−1**	**−1**	**−1**	**−1**	**−1**	**−1**
Base	C_b	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}
P_2	0	0	1	1	0	0	0	−1	0	0	−1	−1	0	0	0	0	1
P_3	0	1	0	0	1	0	0	1	0	0	1	1	0	0	0	0	0
P_{13}	−1	0	0	0	0	0	0	0	0	0	0	−1	−1	−1	1	1	1
P_7	0	1	0	0	0	0	0	0	1	1	1	1	1	1	0	−1	−1
P_4	0	0	1	0	0	1	0	0	0	−1	−1	−1	−1	0	0	1	1
P_5	0	1	−1	0	0	0	1	1	0	1	1	1	1	0	0	0	−1
$\rho(\chi)$		**0**	0	0	0	0	0	0	0	0	0	2	2	2	0	0	0

The basic feasible solution at the end of Phase 1 computation is used as the initial basic feasible solution of the problem. The original objective function is introduced in Phase 2 computation and the usual simplex procedure is used to solve the problem. The Phase 2 gives final optimal value in Table 9.

Table 9. Final optimal Tableau.

Tableau			**−100**	**−200**	**−300**	**−40**	**−80**	**−120**	**−80**	**−160**	**−240**
Base	C_b	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
P_2	−200	0	0	1	1	−1	0	0	−1	0	0
P_9	−240	0	0	0	1	−1	−1	0	0	0	1
P_{13}	−1	0	0	0	0	0	0	0	0	0	0
P_8	−160	1	0	0	−1	1	1	0	1	1	0
P_1	−100	1	1	0	0	1	0	0	1	0	0
P_6	−120	1	0	0	0	1	1	1	0	0	0
$\rho(\chi)$		**−380**	0	0	20	100	40	0	20	0	0

Similar to the HABBP, the simplex method gives the total optimal cost of assigning cloudlets to virtual machines as 100 s + 120 s + 160 s = 380 s. That is, $cloudlet_1$ is assigned to vm_1, $cloudlet_3$ is assigned to vm_2, and $cloudlet_2$ is allocated to vm_3.

5. Virtual Machine Placement Problem

In this section, the problem of optimally placing a set of VMs into a set of PMs in the single cloud environment is formulated. As depicted in Figure 2, the tree network topology consists of five PMs and connection points called switches (SWs). The placement of any VM in a PM will be determined by at least a switch node in the figure. In the light of that, there will be huge end-to-end traffic between a given VM and the switch which the VM is dependent on.

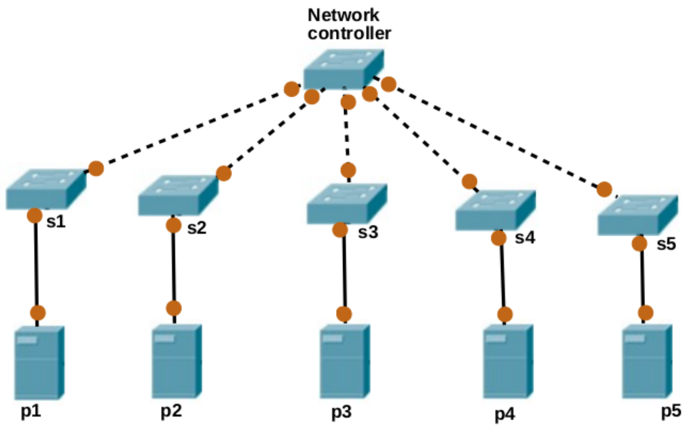


Figure 2. Physical machines and switches in a tree network topology

It is assumed that the intensity of communication between PMs is negligible compared to the intensity of communication between PMs and SWs. Placing the VMs in PMs that offer an optimal placement cost according to the demands of the VMs will be a major determinant factor in this work. Each PM-SW pair is associated with a cost. Thus, it will not be a good idea to place a VM with intensive demand for a switch in a PM that has a high cost associated with that switch.

The VM placement problem in the data center network can be represented mathematically as a graph $G(P, S, E)$, where P is a set of PMs, S is a set of SWs, and E is a set of links between the PMs and SWs. The links are weighted and represent the cost between any PM-SW pair. In addition, it is also assumed that there is no congestion in the links between the PMs and SWs. The links have enough capacity to handle the switch flow demands of VMs appropriately. More information about the network is as follows.

5.1. Parameters

- $P = \{p_1, p_2, \ldots, p_n\}$ is a set of PMs.
- $V = \{v_1, v_2, \ldots, v_m\}$ is a set of VM requests.
- $S = \{s_1, s_2, \ldots, s_k\}$ is a set of switches.
- $l_{s_h p_i}$ is the latency between p_i and s_h.
- $b_{s_h p_i}$ is the bandwidth for $p_i - s_h$ link.
- δ_j represents MIPS of each $v_j \in V$.
- μ_i represents MIPS of each $p_i \in P$.
- U_i represents utilization of p_i.
- E_i^{idle} is the power consumed by p_i when it is doing nothing but powered on.
- E_i^{peak} is the power consumed when the p_i is fully loaded/utilized or at the peak load.

5.2. Assumptions

Consider a VM to be placed into PM through the SW in a data center network, the following assumptions are made.

- Each PM has different latency to all SWs in the network.
- Each PM has one and only one link to the SW in the network.
- Each PM can accommodate more than one VM depending on the capacity of the PM.
- Each link between PMs and SWs has enough capacity and there is no congestion on the links.
- The number of VMs, PMs and SWs are equal i.e., $n = m = k$.

5.3. The Mathematical Model

The cost in terms of the time taken to use the $p_i - s_h$ link is defined as:

$$c_{s_h p_i} = \frac{vm_{size(j)}}{b_{s_h p_i}}. \tag{15}$$

where $vm_{size(j)}$ denotes size (MB) of the v_j routed through the $p_i - s_h$ link.

The placement of a v_j into a p_i depends on the latency between v_j and s_h, and the cost associated with the $p_i - s_h$ link. Thus, the total cost to place v_j into p_i through s_h is computed as,

$$t_{p_i v_j} = \beta c_{p_i s_h} + \alpha l_{s_h v_j} \tag{16}$$

where β and $\alpha \in \{0, 1\}$ is the weighting for the link and latency. The goal is to place VMs into PMs such that the total placement cost for the PM-SW links consumption and latency between VM and SW is minimized. Thus, an optimization model is defined as follows:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} t_{p_i v_j} x_{v_j p_i} = \sum_{i=1}^{n} \sum_{j=1}^{n} (\beta c_{p_i s_j} + \alpha l_{s_i v_j}) x_{v_j p_i} \tag{17}$$

where

$$x_{v_j p_i} = \begin{cases} 1, & \text{if } v_j \text{ is placed into } p_i \text{ ,} \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

subject to

$$\sum_{i=1}^{n} x_{v_j p_i} = 1, \forall j = 1, 2, \ldots, n, \tag{19}$$

$$x_{v_j p_i} \in \{0, 1\}, \text{ for } i = 1, 2, \ldots, n, \text{ and } j = 1, 2, \ldots, n. \tag{20}$$

$$\sum_{j=1}^{u} \delta_j \leq \mu_i, \text{ for } i = 1, 2, \ldots, n, \text{ and } u < n. \tag{21}$$

$$\beta + \alpha = 1 \tag{22}$$

$$c_{s_j p_i} \geq 0 \tag{23}$$

$$l_{s_i v_j} \geq 0 \tag{24}$$

Equation (19) ensures that each VM is mapped to one PM and all VMs are placed. Also, Equation (21) ensures that the total MIPS of VMs placed on a PM should not exceed its capacity. For a given PM, the sum of the MIPS requirements of all VMs placed on it should be less than or equal to the total available capacity of the PM.

Furthermore, it is assumed that there is a linear relationship between the power consumption and utilization of a physical machine in a data center. The energy consumed, E_i, by a PM $p_i \in P$ can be calculated as shown in [28]:

$$E_i = E_i^{idle} + (E_i^{peak} - E_i^{idle})U_i \tag{25}$$

where

$$U_i = \frac{\sum\limits_{j \in \gamma_i} \delta_j}{\mu_i} \tag{26}$$

where γ_i is a set of virtual machines placed on the p_i.

Thus, the total energy consumed by the PMs after VMs placement can be calculated as

$$\sum_{i=1}^{n} E_i = \sum_{i=1}^{n} E_i^{idle} + (E_i^{peak} - E_i^{idle})U_i \tag{27}$$

6. Virtual Machine Placement Algorithmic Solution

The section presents the Genetic Algorithm Based Virtual Machine Placement (GABVMP) for solving the Virtual Machine Placement problem in the cloud computing environment.

6.1. Genetic Algorithm Based Virtual Machine Placement

Genetic Algorithm (GA) is a computerized search and optimization algorithm based on the mechanics of natural genetics and natural selection. The GA is proposed by John Holland [29] where each potential solution is encoded in the form of a string and a population of strings is created which is further processed by three operators: Reproduction, crossover, and mutation. Reproduction is a process in which individual strings are copied according to their fitness function. Crossover is the process of swapping the content of two strings at some point(s) with a probability. Lastly, mutation is the process of flipping the value at a particular location in a string with a very low probability.

Figure 3 describes the GABVMP. The algorithm consists of four parts: Input, initialization, looping and output. In the initialization part, the set of physical machine chromosomes which are also known as population, is generated randomly. The looping part contains fitness evaluation and checks if the optimal solution condition is met according to the optimization objectives. If not, the looping continues, the selection, crossover, mutation and replace functions are applied sequentially. At the end of the loop, the optimal solution will be produced as the output.

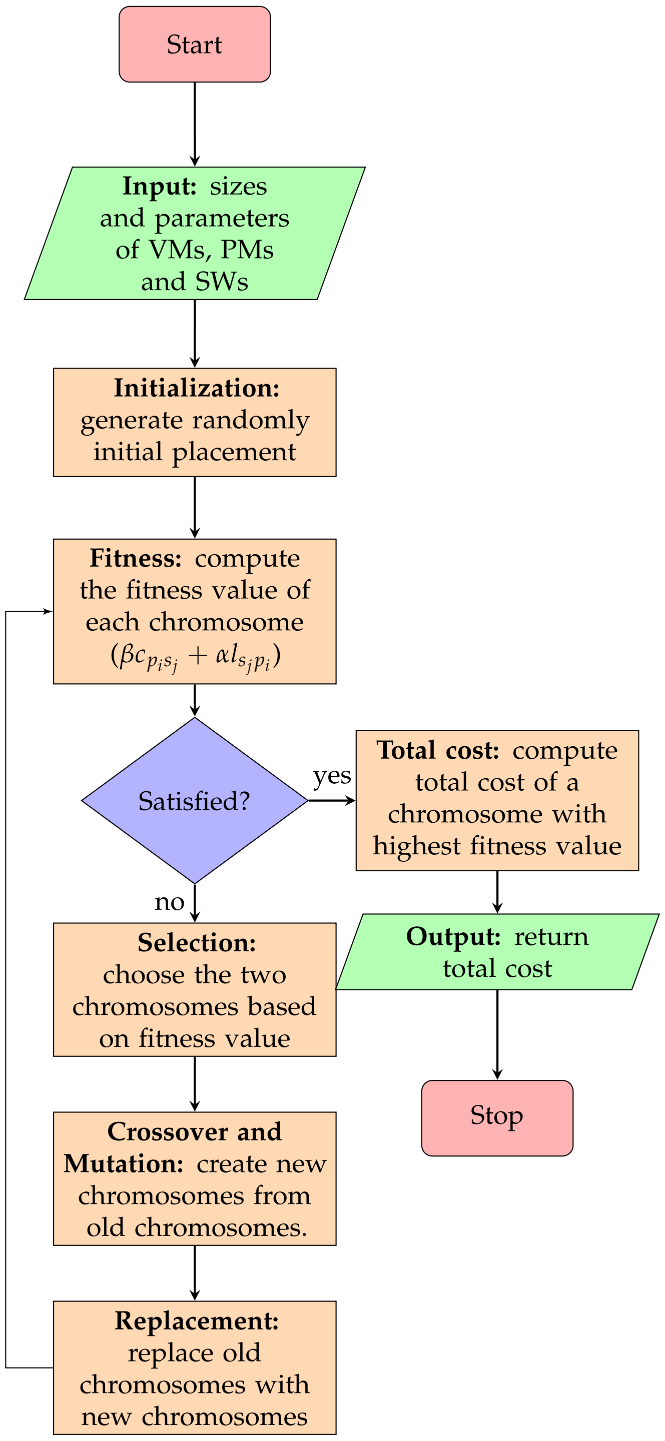

Start
Input: sizes and parameters of VMs, PMs and SWs
Initialization: generate randomly initial placement
Fitness: compute the fitness value of each chromosome $(\beta c_{p_i s_j} + \alpha l_{s_j p_i})$
Satisfied?
yes
no
Total cost: compute total cost of a chromosome with highest fitness value
Output: return total cost
Stop
Selection: choose the two chromosomes based on fitness value
Crossover and Mutation: create new chromosomes from old chromosomes.
Replacement: replace old chromosomes with new chromosomes

Figure 3. Genetic Algorithm Based Virtual Machine Placement

6.2. Initialization

Each chromosome in the GABVMP contains genes which represent the allocated physical resources and switches to the virtual resources. The value of a gene positive integer representing the identity of the VM being placed in the PM through SW. For Example, let $v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8$ be a set of VM to be placed in the $p_1, p_2, p_3, p_4, p_5, p_6$ a set of PM through s_1, s_2, s_3 a set of SW in a data center network. Let's assume that $p_1s_1, p_2s_1, p_3s_2, p_4s_3, p_5s_3, p_6s_3, p_7s_2, p_8s_1$ are links between PMs and SWs which is one of the factors to be considered while placing VMs on the PMs. The initial population contains a set of PM-SW link chromosomes where the genes represent the identity of VMs. The initial population is generated randomly by using Algorithm 2.

Algorithm 2: Initial population algorithm.

```
   input  : a set of links between the PMs and SWs
            a set of VMs with corresponding latency between the VM and SW
   output : initial population
1  counter = 0;
2  while (counter $\leq$ 5) do
3      for (all VMs) do
4          randomize the set of VM identities into number of the PM-SW links according to
             network topology
5      end
6      counter = counter + 1;
7  end
8  return initial population
```

6.3. Fitness Evaluation

The objective is to minimise the total cost of placing VMs on the PMs through SWs. As defined in Section 5, The VM placement cost consists of the cost of PM-SW link usage and the latency between the VM and SW. The objective function used by the GABVMP is the same objective function as that of the mathematical model. Thus, the fitness value of each chromosome is calculated as,

$$Fitness(chromosome) = \beta c_{p_i s_j} + \alpha l_{s_j p_i} \tag{28}$$

6.4. Generating the Next Population

A new population is generated from an initial population of solutions using their fitness values and genetic operators: Selection, crossover, mutation and reproduction. In order to generate a new population, individuals are selected for participation and the genetic operators are applied as follows.

6.4.1. Selection Process

To select the best chromosomes that would pass their genes into the next generation, the fitness proportionate selection approach is implemented using roulette wheel selection. The fitness function is the total cost of the VM placement represented by each chromosome. The lower the total cost, the fitter the VM placement represented by that chromosome [30]. Thus, the chromosomes with lower values are selected for the generation of the next population.

6.4.2. Crossover Operator

The crossover operator works on two parent chromosomes and produces a new individual. In GABVMP, a midpoint crossover with crossover probability 0.8 is adopted and crossover operator process is described in Algorithm 3. Figure 4 shows two parent and offspring chromosomes before and after mid crossover respectively.

Algorithm 3: Crossover function.

```
    input  : $Q_1$, $Q_2$ : two parent chromosomes
    output : $Q_{\lambda 1}$, $Q_{\lambda 2}$ : two offspring chromosomes
1   $\Phi = length(Q_1)$;
2   $cp = \frac{Q_1}{2}$;
3   mid cross point;
4   $Q_{\lambda 1} = Q_1(1 : cp)UP_2(cp : \Phi)$;
5   $P_{\lambda 2} = Q_1(cp : \Phi)UQ_2(1 : cp)$;
6   return $Q_{\lambda 1}$, $Q_{\lambda 2}$.
```

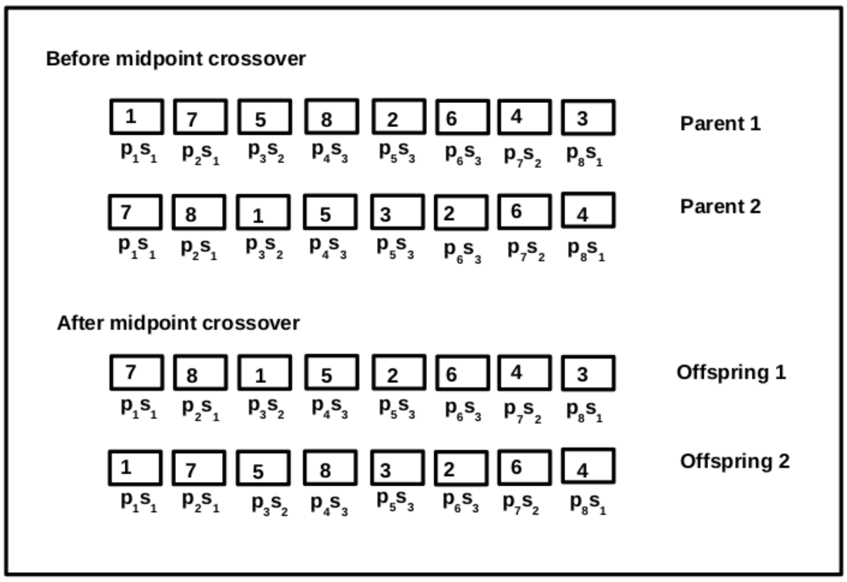


Figure 4. Midpoint crossover.

6.4.3. Mutation Operator

In the GABVMP, the next operation is mutation of the offspring. Mutation helps to prevent premature convergence and promotes diversity in the population. In other words, it helps to avoid getting trapped in local solutions. In this work, inversion mutation is adopted where a subset of genes in a chromosome is selected and inverted to form mutated offspring. Figure 5 illustrates the inversion mutation operation on the offspring 1. In offspring 1, a subset of genes (1, 5, 2, 6) in chromosome (7, 8, 1, 5, 2, 6, 4, 3) are selected and inverted to give a new chromosome (7, 8, 6, 2, 5, 1, 4, 3).

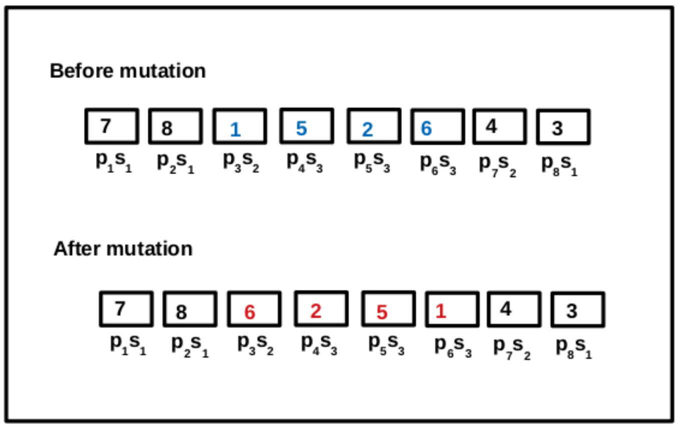

Before mutation
7 8 1 5 2 6 4 3
After mutation
7 8 6 2 5 1 4 3
p_1s_1 p_2s_1 p_3s_2 p_4s_3 p_5s_3 p_6s_3 p_7s_2 p_8s_1

Figure 5. Inversion mutation operation.

6.4.4. Replacement

The replacement operator replaces old chromosomes in the current population with the new chromosomes to form a new population.

6.4.5. Stopping Criterion

GABVMP stops either when the maximum number of generations is reached or the optimal total placement cost is obtained.

The comparison of the GABVMP and the existing related VM placement approaches as mentioned above is presented in Table 10. The comparison parameters include: Latency awareness, energy awareness, network awareness, Internal traffics, flow path allocation and the method adopted to solve the VM placement problem.

Table 10. Comparison of related VM placement problems.

Paper	Latency-Aware	Energy-Aware	Network-Aware	Internal Traffic	Flow Path Allocation	Method Adopted
[18]	No	Yes	No	Yes	No	Scheduling algorithms
[14]	No	No	Yes	Yes	Yes	Greedy method
[15]	No	Yes	No	No	No	EAGLE algorithm
[16]	No	Yes	No	No	No	Ant-colony based algorithm
[20]	No	No	Yes	Yes	Yes	Cluster-and-Cut algorithm
[17]	No	Yes	Yes	Yes	No	Multi-objective evolutionary algorithms
[19]	No	No	Yes	Yes	Yes	Virtual Infrastructure Opportunistic fit (VIO) and VIcinity-BasEd Search (VIBES)
[21]	Yes	No	Yes	Yes	No	Local search algorithm
[22]	No	Yes	No	No	No	Column generation method, cut-and-solve-based algorithm and the call back method
[23]	No	Yes	No	No	No	Neural networks, Self Organizing Map (SOM) and K-Mean Clustering algorithms
GABVMP	Yes	Yes	Yes	Yes	Yes	Genetic algorithm

7. Experimental Results

In this section, we evaluate the performance of our proposed models. We carried out experiments on a desktop computer with specifications: Intel Core i7 CPU @ 2.80 GHz CPU and 4 GB RAM.

7.1. Implementation of the Proposed HABBP

We implemented the proposed HABBP on the famous open source cloud simulator known as CloudSim [31] Netbeans IDE 8.2. CloudSim is developed by the Cloud Computing and Distributed Systems (CLOUDS) Laboratory as an extensible Java-based open source framework for modelling and simulation of cloud computing infrastructures and services. Garg et al. [32] refer to CloudSim as an advanced simulator for cloud computing infrastructures and services with great properties such as scaling as well as a low simulation overhead. It provides classes for data centres, VMs, applications, users, computational resources, and scheduling policies. As depicted in Figure 6, there are different stages of the CloudSim life cycle ranging from initialization of cloud infrastructures to the simulation results. Our goal is to validate the performance gains derived from the proposed HABBP compared to the default task assignment strategy as implemented in the CloudSim.

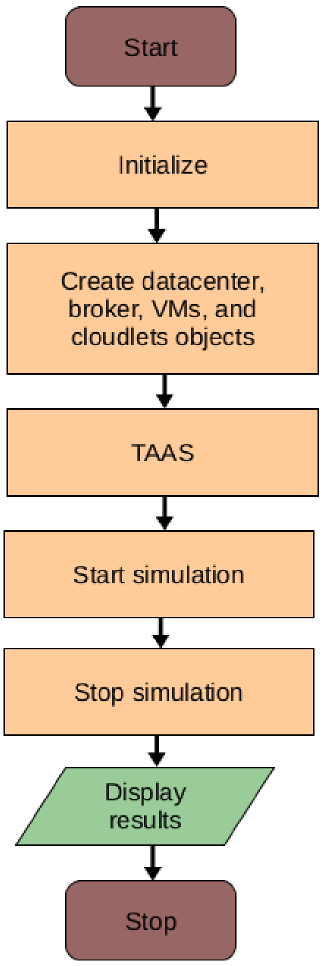
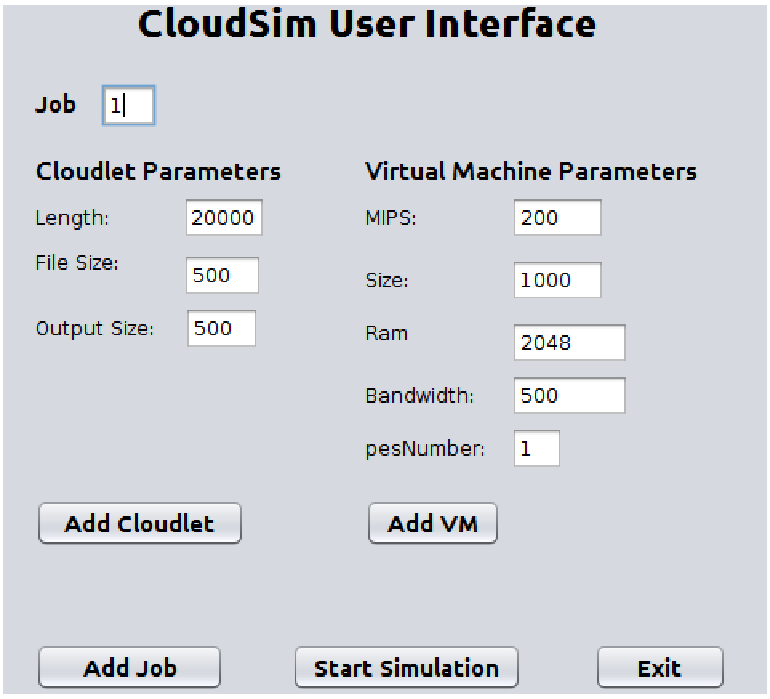


Figure 6. CloudSim Life cycle with HABBP.

The main disadvantage of the current CloudSim, however, is the lack of a graphical user interface (GUI) that allows cloud users to configure cloudlets and the cloud infrastructure parameters, and the lack of optimal cloudlets-to-VMs assignment strategy. In the current work, we extended CloudSim to (i) implement and integrate a graphical user interface using a java *Jframe* class as shown in Figure 7 and (ii) introduced a new cloudlets-to-VMs assignment strategy by creating a new method called *TaskAllocationAlgorithmicSolution()* in the *DatacenterBroker* class of CloudSim.


CloudSim User Interface

Job: 1

Cloudlet Parameters

Length: 20000

File Size: 500

Output Size: 500

Virtual Machine Parameters

MIPS: 200

Size: 1000

Ram 2048

Bandwidth: 500

pesNumber: 1

Figure 7. CloudSim User Interface.

The model was implemented by assigning each VM to different PMs of the same capacity and all PMs were located in the same data centre. Subsequently, we simulated five jobs using HABBP, default assignment strategy and the Simplex algorithm to assign cloudlets to VMs in CloudSim. Each job had 20 cloudlets and they assigned them to heterogeneous VMs. Each cloudlet and VM had different lengths and MIPS values respectively. We assumed that other parameters, such as file size, output size values of all cloudlets, size, ram, bandwidth and pesNumber of all VMs, were constant, as shown in Tables 11 and 12.

Table 11. A set of jobs.

$cloudlet_{id}$	job1	job2	job3	job4	job5
	$(length)$	$(length)$	$(length)$	$(length)$	$(length)$
0	20,000	30,000	150,000	40,000	200,000
1	60,000	50,000	80,000	20,000	80,000
2	90,000	110,000	130,000	80,000	160,000
3	40,000	10,000	90,000	30,000	20,000
4	120,000	70,000	10,000	10,000	40,000
5	200,000	20,000	40,000	90,000	90,000
6	70,000	80,000	120,000	110,000	120,000
7	80,000	40,000	60,000	60,000	10,000
8	50,000	160,000	20,000	150,000	130,000
9	10,000	90,000	70,000	200,000	100,000
10	150,000	350,000	45,000	75,000	5000
11	250,000	250,000	250,000	300,000	25,000
12	45,000	100,000	85,000	450,000	155,000
13	25,000	95,000	140,000	87,000	95,000
14	75,000	25,000	100,000	250,000	4000
15	30,000	85,000	35,000	50,000	110,000
16	300,000	180,000	130,000	100,000	210,000
17	55,000	35,000	75,000	130,000	55,000
18	65,000	15,000	95,000	95,000	85,000
19	85,000	9000	200,000	400,000	350,000

Table 12. A set of VMs.

vm_{id}	$mips\ value$
0	1000
1	500
2	200
3	2000
4	250
5	100
6	50
7	125
8	150
9	400
10	1500
11	350
12	450
13	600
14	700
15	850
16	900
17	550
18	1200
19	300

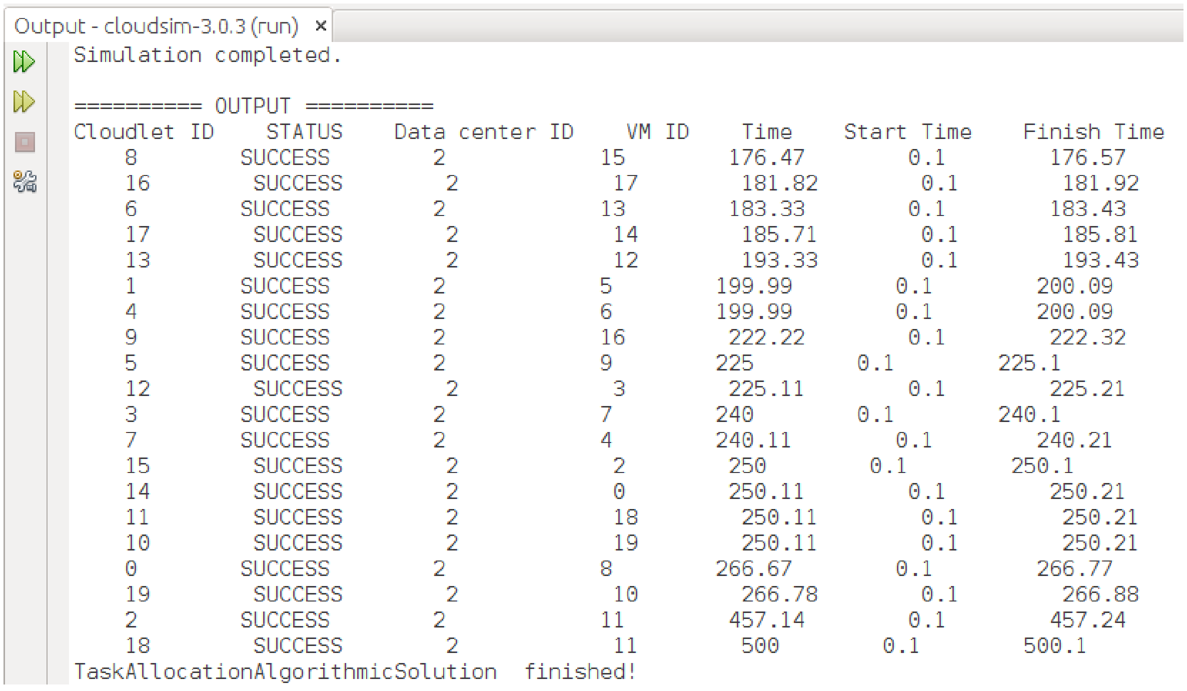
The simulation results are presented in Tables 13 and 14, Figures 8 and 9, and plotted in Figures 10 and 11. In job 1, under the default assignment strategy, cloudlets were assigned to the VMs sequentially, that is $cloudlet_0$ to vm_0, $cloudlet_1$ to vm_1, $cloudlet_2$ to vm_2, $cloudlet_3$ to vm_3, $cloudlet_4$ to vm_4, etc. On the other hand, HABBP assigned cloudlets to VMs based on the operations in HABBP. For instance, $cloudlet_0$ is assigned to vm_8, $cloudlet_1$ to vm_5, $cloudlet_2$ to vm_{11}, $cloudlet_3$ to vm_7, $cloudlet_4$ to vm_6, etc. in job 4; see Figure 8.

Table 13. Default assignment strategy in CloudSim.

$cloudlet_{id}$	**job1**		**job2**		**job3**		**job4**		**job5**	
	vm_{id}	**proc. time**	vm_{id}	**proc. time**	vm_{id}	**proc. time**	vm_{id}	**proc. time**	vm_{id}	**proc. time**
0	0	20	0	30	0	150	0	40	0	200
1	1	120	1	100	1	160	1	40	1	160
2	2	450	2	550	2	650	2	400	2	800
3	3	20	3	5	3	45	3	15	3	10
4	4	480	4	280	4	40	4	40	4	160
5	5	2000	5	200	5	400	5	900	5	900
6	6	1400	6	1600	6	2400	6	2200	6	2400
7	7	640	7	320	7	480	7	480	7	80
8	8	333	8	1066	8	133	8	1000	8	866
9	9	25	9	225	9	175	9	500	9	250
10	10	100	10	233	10	30	10	50	10	3
11	11	714	11	714	11	714	11	857	11	71
12	12	100	12	222	12	189	12	1000	12	344
13	13	42	13	158	13	233	13	145	13	158
14	14	107	14	36	14	143	14	357	14	6
15	15	35	15	100	15	41	15	59	15	129
16	16	333	16	200	16	144	16	111	16	233
17	17	100	17	64	17	136	17	236	17	100
18	18	54	18	13	18	79	18	79	18	71
19	19	283	19	30	19	667	19	1333	19	1167
total processing time	7356		6147		7009		9842		8109	

Table 14. HABBP in CloudSim.

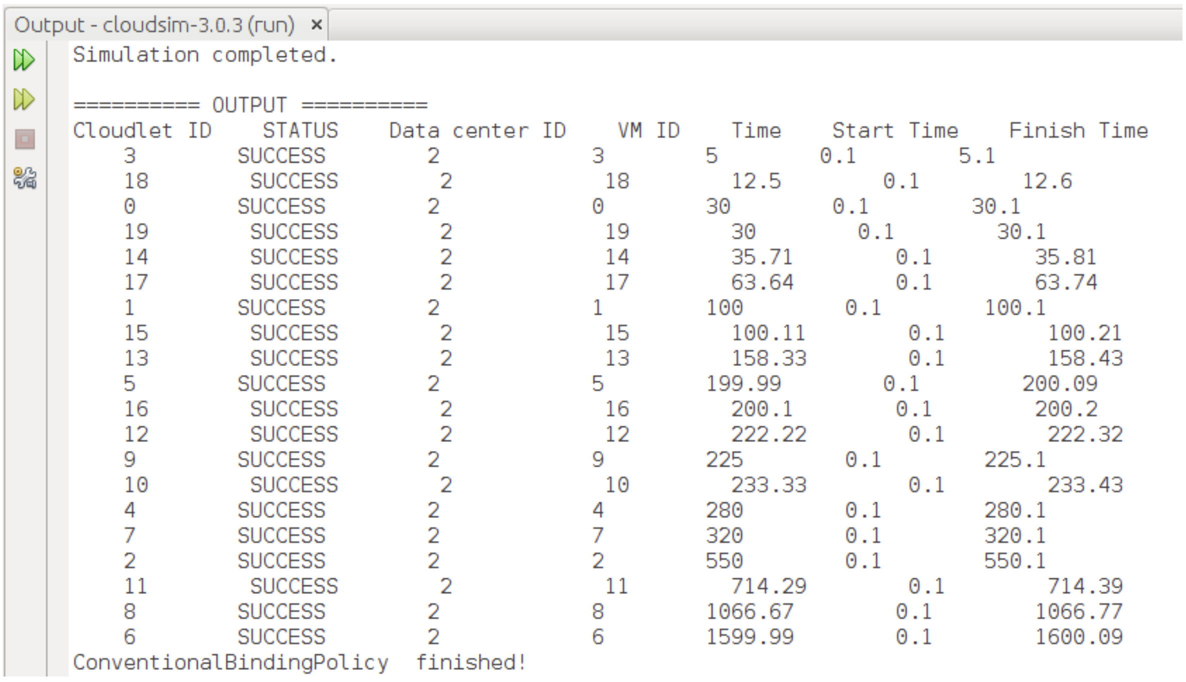
$cloudlet_{id}$	job1		job2		job3		job4		job5	
	vm_{id}	**proc. time**	vm_{id}	**proc. time**	vm_{id}	**proc. time**	vm_{id}	**proc. time**	vm_{id}	**proc. time**
0	5	200	4	120	18	125	8	267	18	167
1	9	150	9	125	9	200	5	200	11	229
2	15	106	15	129	15	153	11	229	0	160
3	2	200	5	100	1	180	7	240	8	133
4	16	133	12	156	6	200	6	200	4	160
5	18	167	8	133	8	267	9	225	12	200
6	1	140	1	160	14	171	13	183	14	171
7	13	133	11	114	4	240	4	240	7	80
8	19	167	0	160	5	200	15	176	15	153
9	6	200	13	150	19	233	16	222	17	182
10	0	150	3	175	2	255	19	250	5	50
11	10	167	10	167	3	125	18	250	2	125
12	4	180	16	111	12	189	3	225	16	172
13	7	200	14	136	0	110	12	193	1	190
14	17	136	2	125	17	182	0	250	6	80
15	8	200	17	155	7	280	2	250	13	183
16	3	150	18	150	16	144	17	182	10	140
17	11	157	19	117	11	214	14	186	19	183
18	12	144	7	120	13	158	11	190	9	213
19	14	121	6	180	10	133	10	267	3	175
total processing time	3201		2783		3759		4425		3146	


Output - cloudsim-3.0.3 (run)

```
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    8        SUCCESS        2             15      176.47      0.1        176.57
    16        SUCCESS        2             17      181.82      0.1        181.92
    6        SUCCESS        2             13      183.33      0.1        183.43
    17        SUCCESS        2             14      185.71      0.1        185.81
    13        SUCCESS        2             12      193.33      0.1        193.43
    1        SUCCESS        2             5      199.99      0.1        200.09
    4        SUCCESS        2             6      199.99      0.1        200.09
    9        SUCCESS        2             16      222.22      0.1        222.32
    5        SUCCESS        2             9      225      0.1      225.1
    12        SUCCESS        2             3      225.11      0.1        225.21
    3        SUCCESS        2             7      240      0.1      240.1
    7        SUCCESS        2             4      240.11      0.1        240.21
    15        SUCCESS        2             2      250      0.1      250.1
    14        SUCCESS        2             0      250.11      0.1        250.21
    11        SUCCESS        2             18      250.11      0.1        250.21
    10        SUCCESS        2             19      250.11      0.1        250.21
    0        SUCCESS        2             8      266.67      0.1        266.77
    19        SUCCESS        2             10      266.78      0.1        266.88
    2        SUCCESS        2             11      457.14      0.1        457.24
    18        SUCCESS        2             11      500      0.1      500.1
TaskAllocationAlgorithmicSolution  finished!
```

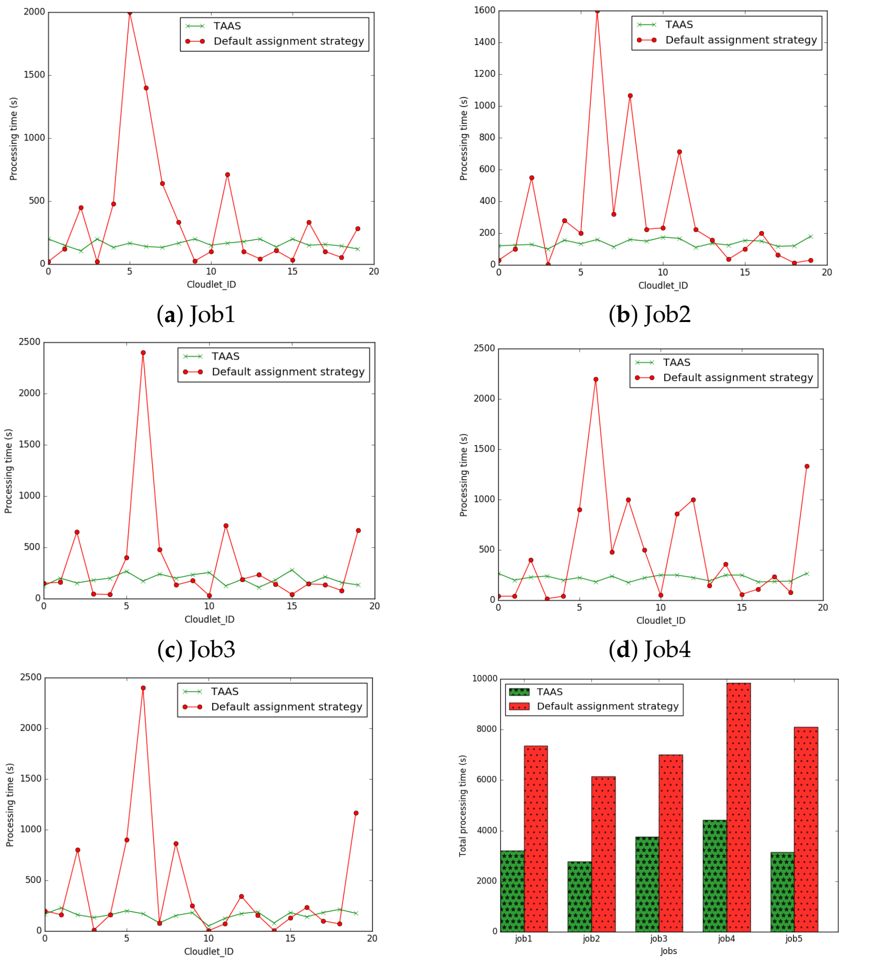
Figure 8. Assigning cloudlets to VMs using HABBP.



```
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    3        SUCCESS        2            3        5        0.1        5.1
    18        SUCCESS        2            18        12.5        0.1        12.6
    0        SUCCESS        2            0        30        0.1        30.1
    19        SUCCESS        2            19        30        0.1        30.1
    14        SUCCESS        2            14        35.71        0.1        35.81
    17        SUCCESS        2            17        63.64        0.1        63.74
    1        SUCCESS        2            1        100        0.1        100.1
    15        SUCCESS        2            15        100.11        0.1        100.21
    13        SUCCESS        2            13        158.33        0.1        158.43
    5        SUCCESS        2            5        199.99        0.1        200.09
    16        SUCCESS        2            16        200.1        0.1        200.2
    12        SUCCESS        2            12        222.22        0.1        222.32
    9        SUCCESS        2            9        225        0.1        225.1
    10        SUCCESS        2            10        233.33        0.1        233.43
    4        SUCCESS        2            4        280        0.1        280.1
    7        SUCCESS        2            7        320        0.1        320.1
    2        SUCCESS        2            2        550        0.1        550.1
    11        SUCCESS        2            11        714.29        0.1        714.39
    8        SUCCESS        2            8        1066.67        0.1        1066.77
    6        SUCCESS        2            6        1599.99        0.1        1600.09
ConventionalBindingPolicy  finished!
```

Figure 9. Assigning cloudlets to VMs using default assignment strategy.



(**a**) Job1 (**b**) Job2

(**c**) Job3 (**d**) Job4

(**e**) Job5 (**f**) HABBP vs. Default assignment strategy

Figure 10. Processing time of the cloudlets in Job 1, Job 2, Job 3, Job 4 and Job 5.

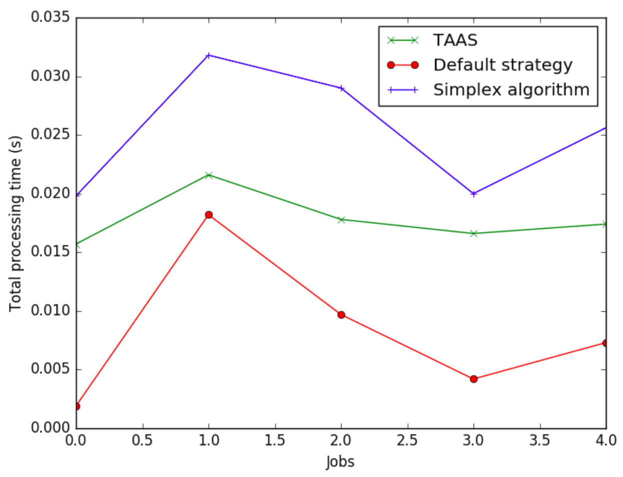


Figure 11. Comparison of computational time.

In addition, we also compared the overall performance of HABBP with the default assignment strategy and benchmarked both solutions against the Simplex algorithm in terms of the computational time of cloudlets in each job and the total processing time of individual jobs. In Figure 10a–e, it can be seen that some cloudlets took a slightly longer time to complete in HABBP than in the default assignment strategy, while some other cloudlets took a significantly longer time to complete under default assignment strategy than under HABBP. Figure 10f, however, where the total processing time performance of different jobs for HABBP and default assignment strategy is presented, shows that HABBP constantly outperformed the default assignment strategy. Take Job 2 as an example, compared to the default assignment strategy, the total processing time for HABBP was reduced by 54.73% compared with that of the default assignment strategy. HABBP and the Simplex algorithm produced the same optimal allocation cost. HABBP, however, outperformed the Simplex algorithm in terms of computational time as shown in Table 15 and Figure 11.

Table 15. Comparison of computational time.

HABBP	Default Assignment Strategy	Simplex Algorithm
0.0157	0.0019	0.0198
0.0216	0.0182	0.0318
0.0178	0.0097	0.0290
0.0166	0.0042	0.0200
0.0174	0.0073	0.0256

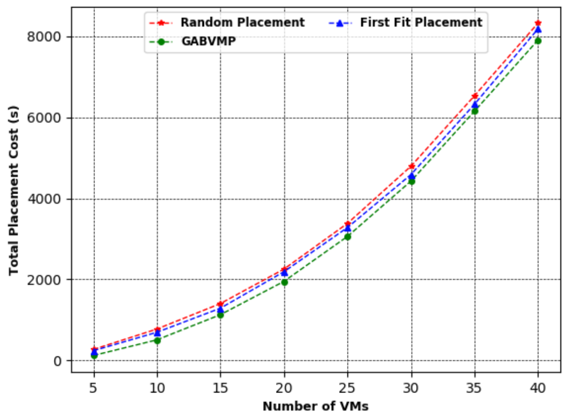
7.2. Implementation of the Proposed GABVMP

In this section, the efficiency of the proposed GABVMP as discussed in Section 6 is evaluated. For the simulation, the mininet module in python3 was used to model the tree topology of the PMs and SWs interconnected in the datacenter. The network topology consisted of equal pairs of PMs and SWs. Each PM-SW link in the network topology had a different capacity in terms of Mbps. The proposed GABVMP and greedy heuristics (Random Placement and First Fit Placement) were implemented and their behaviors were compared on the topology with different numbers of PMs and SWs.

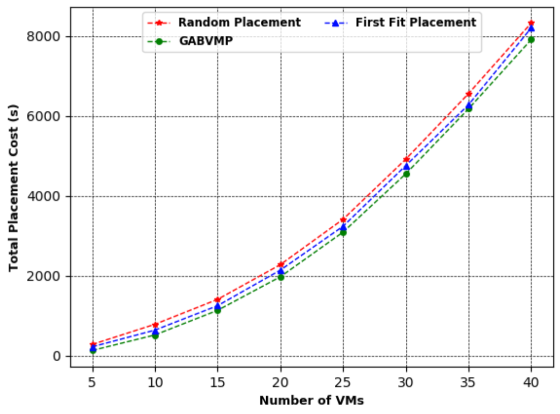
Three experiments were carried out. In the first experiment, 5, 10, 15, 20, 25, 30, 35, 40 VMs were placed on 5, 10, 15, 20, 25, 30, 35, 40 PMs interconnected with the same number of switches using GABVMP with different values of α and β. In the second experiment, 5, 10, 15, 20, 25, 30, 35, 40 VMs were placed on 5, 10, 15, 20, 25, 30, 35, 40 PMs interconnected with the same number of switches using Random Placement with different values of α and β.

In the last experiment, 5, 10, 15, 20, 25, 30, 35, 40 VMs were placed on 5, 10, 15, 20, 25, 30, 35, 40 PMs interconnected with the same number of switches using First Fit Placement with different values of α and β.

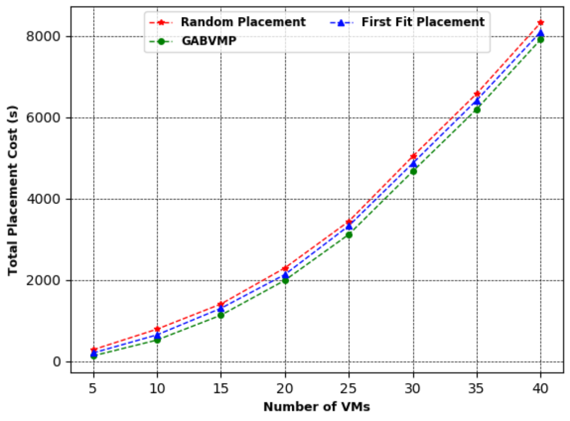
Figure 12 shows the experimental results of total placement cost in terms of time to implement the proposed GABVMP and the other two existing assignment methods in a data center network with tree topology consisting of 5, 10, 15, 20, 25, 30, 35, 40 of PMs and SWs at different values of of α and β. The GABVMP had a lower cost to place VMs on PMs than the Random Placement and First Fit Placement. For instance, at α = 0.2 and β = 0.8, the Random Placement had a total placement cost of 266 s and the First Fit Placement had a total placement cost of 205 s while GABVMP took a total placement cost of 116 s to place five VMs on five PMs interconnected with five SWs. For α = 0.5 and β = 0.5, the Random Placement had a total placement cost of 275 s and the First Fit Placement had a total placement cost of 235 s while GABVMP took a total cost of 125 s to place five VMs on five PMs interconnected with five SWs. For α = 0.8 and β = 0.2, the Random Placement had a total cost of 284 s and the First Fit Placement had a total placement cost of 216 s while GABVMP took a total cost of 134 s to place five VMs on five PMs interconnected with five SWs.



(**a**) α = 0.2, β = 0.8



(**b**) α = 0.5, β = 0.5

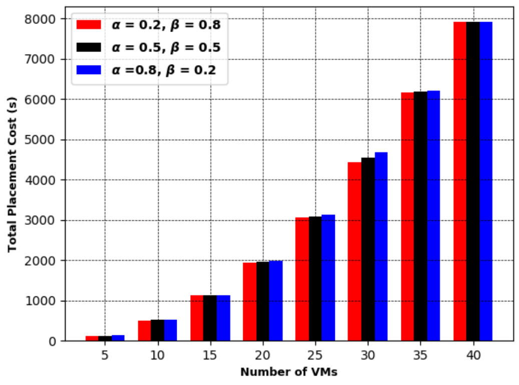


(**c**) α = 0.8, β = 0.2

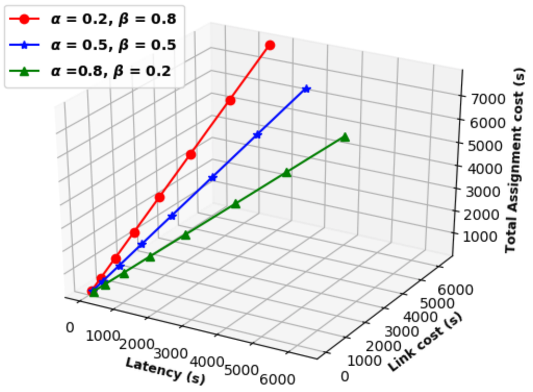
Figure 12. GABVMP vs. Random Placement vs. First Fit Placement.

In addition, Figure 13 illustrates the impact of latency on the total assignment cost of the proposed GABVMP. The higher the value of α which denotes the weight of latency, the higher the total placement cost. For instance, when α = 0.2, the total placement cost was 1947 s, when α = 0.5, the total placement cost was 1970 s and α = 0.8, the total placement cost was 1992 s to place 20 VMs into 20 PMs.

Finally, Figure 14 shows the plot for energy consumption vs. number of VMs. The value of E_i^{peak} and E_i^{idle} was set to 300 J and 200 J respectively [28]. It is observed from the figure that, when the number of VMs was increased, the energy consumed by the used PMs was also increased. Energy consumption in the proposed GABVMP, however, was lower than the Random Placement method. This is because the number of PMs required to place a given number of VMs was less in GABVMP than the Random Placement and First Fit Placement.



(**a**) Total cost for α and β values



(**b**) Relationship between latency cost, link cost and total cost

Figure 13. GABVMP for α and β values.

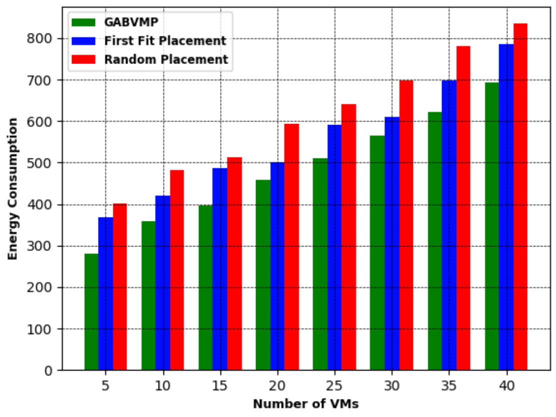


Figure 14. Energy Consumption vs. Number of VMs.

8. Conclusions and Future Work

The level of Quality-of-Service in cloud computing is determined to a large extent by the resource allocation strategy adopted. In this work, the issue of Quality-of-Service in cloud computing environments has been revisited. Two solution models have been proposed. Firstly, the tasks-to-virtual machines allocation problem as a linear-programming problem model was formulated and HABBP was proposed, a load balancing policy for binding cloudlets to virtual machines. The simulation results produced by the contributed code to the CloudSim simulation revealed the relative efficiency of the newly proposed HABBP policy in solving and optimizing the virtual resources allocation problem in the cloud computing environment. Secondly, the virtual machine placement problem was presented and proposed a GABVMP as the solution for optimizing the model. The simulation results show that the GABVMP performed better than the two greedy heuristics, Random Placement and First Fit Placement, in terms of PM-SW links consumption which corresponds to the cost of placing VMs on PMs in the data center.

In the near future, the proposed solutions will be used to optimize resource allocation in federated lightweight cloud computing infrastructures targeting not only drought mitigation [33,34] in the rural areas of Africa but also healthcare, following the framework proposed in [35,36]. For such deployments, the policy will be extended to account for traffic engineering characteristics of the cloud computing network for both local traffic [37] and inter-Africa traffic [38] as these can have a large impact on the access to the cloud nodes and thus influence the QoS provided by the cloud. Using UAVs/drones in the context of 5G as proposed in [39] is another alternative for deployment. The implementation of the newly proposed policy in a real and popular cloud computing management platform, such as OpenStack, is another avenue for future work.

Author Contributions: Conceptualization, S.B.A. and A.B.; Methodology, S.B.A. and A.B.; Software, S.B.A.; Writing—original draft preparation, S.B.A.; Supervision, A.B.; Writing—review and editing, A.B.

Funding: National Research Foundation: NRF FREESTANDING.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mell, P.; Grance, T. *The NIST Definition of Cloud Computing*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2015. Available online: http://www.nist.gov/itl/cloud (accessed on 19 October 2015).
2. Rimal, B.P.; Choi, E.; Lumb, I. A taxonomy and survey of cloud computing systems. In Proceedings of the Fifth International Joint Conference on INC, IMS and IDC, Seoul, Korea, 25–27 August 2009; pp. 44–51.
3. Assante, D.; Castro, M.; Hamburg, I.; Martin, S. The Use of Cloud Computing in SMEs. *J. Inf. Technol. Manag.* **2016**, *83*, 1207–1212. [CrossRef]
4. Zhang, Q.; Cheng, L.; Boutaba, R. Cloud computing: State-of-the-art and research challenges. *J. Internet Serv. Appl.* **2010**, *1*, 7–18. [CrossRef]
5. Chaudhary, V.; Minsuk, C.; Walters, J.P.; Guercio, S.; Gallo, S. A Comparison of Virtualization Technologies for HPC. In Proceedings of the 22nd International Conference on Advanced Information Networking and Applications (AINA 2008), Okinawa, Japan, 25–28 March 2008; pp. 861–868.
6. Buyya, R.; Yeo, C.; Venugopal, S.; Broberg, J.; Brandic, I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.* **2009**, *25*, 599–616. [CrossRef]
7. Maguluri, S.T.; Srikant, R.; Ying, L. Stochastic models of load balancing and scheduling in cloud computing clusters. In Proceedings of the IEEE INFOCOM, Orlando, FL, USA, 25–30 March 2012; pp. 702–710.
8. Baker, T.; MacKay, M.; Randles, M.; Taleb-Bendiab, A. Intention-oriented programming support for runtime adaptive autonomic cloud-based applications. *Comput. Electr. Eng.* **2013**, *39*, 2400–2412. [CrossRef]
9. Al-khafajiy, M.; Baker, T.; Waraich, A.; Al-Jumeily, D.; Hussain, A. IoT-Fog Optimal Workload via Fog Offloading. In Proceedings of the 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), Auckland, New Zealand, 2–5 December 2018.
10. Lin, C.T. Comparative based analysis of scheduling algorithms for RM in cloud computing environment. *Int. J. Comput. Sci. Eng.* **2013**, *1*, 17–23.
11. Liu, Z.; Qu, W.; Liu, W.; Li, Z.; Xu, Y. Resource preprocessing and optimal task scheduling in cloud computing environments. *Concurr. Comput. Pract. Exp.* **2014**, *27*, 3461–3482. [CrossRef]
12. Zhang, L.; Zhuang, Y.; Zhu, W. Constraint Programming Based Virtual Cloud Resources Allocation Model. *Int. J. Hybrid Inf. Technol.* **2013**, *6*, 333–344. [CrossRef]
13. Dupont, C.; Giuliani, G.; Hermenier, F.; Schulze, T.; Somov, A. An energy aware framework for virtual machine placement in cloud federated data centres. In Proceedings of the Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy), Madrid, Spain, 9–11 May 2012; pp. 1–10.
14. Kanagavelu, R.; Bu-Sung, L.; DatLe, N.T.; NgMingjie, L.; MiAung, K.M. Virtual machine placement with two-path traffic routing for reduced congestion in data center networks. *J. Comput. Commun.* **2014**, *53*, 1–12. [CrossRef]
15. Li, X.; Qiana, Z.; Lu, S.; Wub, J. Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center. *J. Math. Comput. Model.* **2013**, *58*, 1222–1235. [CrossRef]
16. Gao, Y.; Guan, H.; Qi, Z.; Hou, Y.; Liu, L. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *J. Comput. Syst. Sci.* **2013**, *79*, 1230–1242. [CrossRef]

17. Pascual, J.; Lorido-Botran, T.; Miguel-Alonso, J.; Lozano, J. Towards a greener cloud infrastructure management using optimized placement policies. *J. Grid Comput.* **2014**, *13*, 375–389. [CrossRef]
18. Ebrahimirad, V.; Goudarzi, M.; Rajabi, A. Energy-aware scheduling for precedence-constrained parallel virtual machines in virtualized data centers. *J. Grid Comput.* **2015**, *13*, 233–253. [CrossRef]
19. Georgiou, S.; Delis, K.T.A. Exploiting network-topology awareness for vm placement in iaas clouds. In Proceedings of the 2013 International Conference on Cloud and Green Computing, Karlsruhe, Germany, 30 September–2 October 2013; pp. 151–158.
20. Meng, X.; Pappas, V.; Zhang, L. Improving the scalability of data center networks with traffic-aware virtual machine placement. In Proceedings of the 2010 IEEE INFOCOM, San Diego, CA, USA, 14–19 March 2010; pp. 1–9.
21. Breitgand, D.; Epstein, A.; Glikson, A.; Israel, A.; Raz, D. Network aware virtual machine and image placement in a cloud. In Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013), Zurich, Switzerland, 14–18 October 2013; pp. 9–17.
22. Vakilinia, S.; Heidarpour, B.; Cheriet, M. Energy Efficient Resource Allocation in Cloud Computing Environments. *IEEE Access* **2016**, *4*, 8544–8557. [CrossRef]
23. Mohamed, H.K.; Alkabani, Y.; Selmy, H. Energy Efficient Resource Management for Cloud Computing Environment. In Proceedings of the 9th International Conference on Computer Engineering and Systems (ICCES), Cairo, Egypt, 22–23 December 2014; Volume 1.
24. Kuhn, H.W. The Hungarian Method for the Assignment Problem. *Nav. Res. Logist. Q.* **1955**, *2*, 83–97. [CrossRef]
25. Konig, D. Grafok es matrixok. matematikai es fizikai lapok. *Matematikai Es Fizikai Lapok* **1931**, *38*, 116–119.
26. Dantzig, G.B. *Programming in a Linear Structure*; USAF: Washington, DC, USA, 1948.
27. Dantzig, G.B. *Linear Programming and Extensions*; Princeton University Press: Princeton, NJ, USA, 2016; Volume 4.
28. Lee, Y.; Zomaya, A. Energy efficient utilization of resources in cloud computing systems. *J. Supercomput.* **2012**, *60*, 268–280. [CrossRef]
29. Holland, J. *The Grid: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*; University of Michigan Press: Ann Arbor, MI, USA, 1975.
30. Su, F.; Zhu, F.; Yin, Z.; Yao, H.; Wang, Q.; Dong, W. New Crossover Operator of Genetic Algorithms for the TSP. In Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization, Sanya, China, 24–26 April 2009; Volume 1, pp. 666–669. [CrossRef]
31. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.F.; Buyya, R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* **2011**, *41*, 23–50. [CrossRef]
32. Garg, S.K.; Buyya, R. NetworkCloudSim: Modelling parallel applications in cloud simulations. In Proceedings of the Fourth IEEE International Conference on Utility and Cloud Computing (UCC), Victoria, Australia, 5–8 December 2011; pp. 105–113.
33. Masinde, M.; Bagula, A. A framework for predicting droughts in developing countries using sensor networks and mobile phones. In Proceedings of the 2010 Conference of the South African Institute of Computer Scientists and Information Technologists, Bela Bela, South Africa, 11–13 October 2010; pp. 390–399.
34. Masinde, M.; Bagula, A.; Muthama, T.N. The role of ICTs in downscaling and upscaling integrated weather forecasts for farmers in sub-Saharan Africa. In Proceedings of the Fifth International Conference on Information and Communication Technologies and Development, Atlanta, GA, USA, 12–15 March 2012; pp. 122–129.
35. Bagula, A.; Mandava, M.; Bagula, H. A Framework for Supporting Healthcare in Rural and Isolated Areas. *Elsevier J. Netw. Commun. Appl.* **2018**. [CrossRef]
36. Bagula, A.; Lubamba, C.; Mandava, M.; Bagula, H.; Zennaro, M.; Pietrosemoli, E. Cloud Based Patient Prioritization as Service in Public Health Care. In Proceedings of the ITU Kaleidoscope 2016, Bangkok, Thailand, 14–16 November 2016.
37. Bagula, A. Hybrid Traffic Engineering: The Least Path Interference Algorithm. In Proceedings of the SAICSIT 2004, Cape Town, South Africa, 4–6 October 2004; pp. 89–96.

38. Chavula, J.; Suleman, H.; Bagula, A. Quantifying the Effects of Circuitous Routes on the Latency of Intra-Africa Internet Traffic: A Study of Research and Education Networks. In *Proceedings of the e-Infrastructure and e-Services for Developing Countries, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*; Springer: Cham, Switzerland, 2014; Volume 147, pp. 64–73.
39. Chiaraviglio, L.; Blefari-Melazzi, N.; Liu, W.; Gutiérrez, J.A.; van de Beek, J.; Birke, R.; Chen, L.; Idzikowski, F.; Kilper, D.; Monti, P.; et al. Bringing 5g into rural and low-income areas: Is it feasible? *IEEE Commun. Stand. Mag.* **2017**, *1*, 50–57. [CrossRef]