Hindawi
Scientific Programming
Volume 2022, Article ID 3042173, 13 pages
https://doi.org/10.1155/2022/3042173

Research Article

Performance Evaluation of Novel Random Biased-Genetic Algorithm (NRB-GA): A Hybrid Load Balancing Algorithm in a Cloud Computing Environment

Karpaga Selvi Subramanian [1] **and Gemmachis Teshite** [2]

[1]*School of Electrical and Computer Engineering, Ethiopian Institute of Technology-Mekelle, Mekelle, Ethiopia*
[2]*Software Engineering Department, College of Computing and Informatics, Haramaya University, POB 138, Dire Dawa, Ethiopia*

Correspondence should be addressed to Karpaga Selvi Subramanian; karpaga.selvi@mu.edu.et

Received 25 May 2022; Revised 30 November 2022; Accepted 8 December 2022; Published 21 December 2022

Academic Editor: Sadiq Hussain

A novel random biased-genetic algorithm (NRB-GA) load-balancing algorithm that exhibits the characteristics of both genetic algorithms and biased random algorithms is designed and developed to improve the processing time and response time metrics of the cloud computing environment. The NRB-GA is designed to discover a virtual machine with fewer loads by applying a genetic algorithm with a fitness function that is inversely proportional to the average load over a period of time for each virtual machine and with biased parent selection to maximize the fitness values of offspring. The developed NRB-GA load-balancing algorithm is evaluated by analysing its performance for various simulated scenarios in a cloud computing environment with different user bases and data center configurations. The analysis of the experimental results of NRB-GA indicates that the average response time is reduced by 27.22%, 21.15%, and 22.34%, and the processing time is reduced by 25.73%, 16.14%, and 18.82% for one, two, and three data centers, respectively. It is evident that the proposed NRB-GA algorithm for load balancing outperforms other existing algorithms significantly.

1. Introduction

The cloud computing environment enables convenient on-demand access for sharing a pool of configurable computing resources (e.g., servers, network storage, applications, and services). With minimal service provider interaction and less management effort, shared resources can be rapidly provisioned and released [1, 2]. Before the emergence of modern cloud computing, all software applications, data, and controls resided on a centralized server. The clients request a service from the servers, which is called as client-server computing. Following client-server computing, distributed computing evolves, where all computers are networked and resources are shared as per need. Afterward, different types of distributed systems emerged, including grid computing, utility computing, and cluster computing. The concept of cloud computing emerged later, on top of the foundation of grid computing, utility computing, and virtualization technology [3, 4]. The evolution of cloud computing opens up challenges in privacy, security, performance, availability, scalability, portability, interoperability, and load balancing. The primary challenge in cloud computing is load balancing, which ensures that no single node is overwhelmed by distributing the dynamic workload across multiple nodes in a balanced way [5]. Various load-balancing algorithms, such as the round-robin algorithm, the equally spread current execution (ESCE) algorithm, the ant colony algorithm, the honey bee behavior (HBB) algorithm, the grey wolf technique, the genetic algorithm, and the biased random algorithm, have been proposed for cloud computing environments. The related works section of this article elaborates on some recent and related research in load balancing in cloud computing environments.

On studying various state-of-the-art load-balancing algorithms for cloud computing environments, we found that the performance of the cloud with respect to processing time

and response time metrics could be improved by hybridising the existing load-balancing algorithms by adopting the advantages of two or more algorithms that are combined together. Hybridisation with more algorithms is computationally ineffective, and there is a chance of degrading the performance of the cloud. This trade-off could be neutralised by a careful selection of algorithms that can be hybridised and by limiting the number of algorithms that are combined. We strongly believe that the combination of a genetic algorithm and a random-biased algorithm could be used to develop a hybrid load-balancing algorithm that improves the processing time and response time in a cloud computing environment. The genetic algorithm is used as an optimized scheduler, and a biased random method is used to select parents with high fitness for crossover and mutation operations of the genetic algorithm. In our proposed solution, a biased random selection of parents accelerates the convergence of the genetic algorithm, reducing the processing time and response time. The genetic algorithm identifies the virtual machine with less load over a period of time (not on the instant); hence, reducing the migration between virtual machines and, in turn, improving the response time and processing time metrics. In the following section, we discuss the basics of genetic algorithms and biased random algorithms.

1.1. Background Study

1.1.1. Biased Random. A random sampling of the system domain is being used to achieve the self-organization that balances the load among all nodes in the system. A virtual directed graph is constructed with nodes representing the load on the server and in-degree representing the free resources of that node [6]. Each process is associated with a threshold value representing the maximum walk length. The definition of a walk is traversing from one node to another until the destination. Walk length is defined as the number of nodes traversed in the walk. On receiving a request, the load balancer selected a node randomly and compared the current walk length with the threshold. If the current walk length is the same or greater than the threshold, then the request is processed in that respective node, and its in-degree value is reduced by one; if not, the current walk length is incremented, and another neighbour node with a minimum of one in-degree is selected randomly. Upon completion of the process, the node's in-degree value is incremented by one. The drawback of this algorithm is that its overall performance is inversely proportional to the number of servers and population diversity.

1.1.2. Genetic Algorithm. The genetic algorithm (GA) is inspired by the evolution of law, which searches for the optimized solutions from a population that eventually evolved in consecutive generations. The search direction adjusts toward the optimized solution by selecting the more probable candidates with a high fitness value to generate the next population [7]. The selected individuals were crossed over and mutated to generate a new population, representing a new solution set. The disadvantage associated with GA is that the individuals are chosen completely randomly from the population, which leads to the possibility of choosing both bad chromosomes and having poor-quality offspring in the next generation. In our proposed method, this drawback is overcome by applying a biased random selection of parents with high fitness values. State-of-the-art load-balancing algorithms and their advantages and limitations are discussed in the following section.

2. Related Works

Ragmani et al. proposed a method to enhance load balancing by introducing a fuzzy logic module to calculate the pheromone value in ant colony optimization. It is proved by experimental results that this method is more appropriate to handle complex networks [8]. Akawee et al. present some resource allocation problems and issues that can be solved with the help of load-balancing techniques and algorithms. This paper focused on highlighting the importance of resource allocation and its relationship with load balancing, and it also discussed the limitations of resource allocation in cloud computing [9].

Sethi et al. proposed a load-balancing algorithm that used fuzzy logic and round-robin scheduling. Workload assignment was based on various parameters, like processor speed and the number of currently allocated requests to each virtual machine (VM). Whenever a new request arrived, the scheduler looked for a minimum loaded VM for assigning the new workload. In the case where more than one virtual machine was identified, then the workload assignment was carried out depending on the processor speed and current load in the VM [10] by a fuzzy logic algorithm.

Hwang and Wood enhanced the user experience by proposing a soft real-time scheduler that employed flexible priority designations and automated scheduler class detection. The implemented scheduler was deployed in the Xen virtualization platform and established that the overhead could be reduced by less than 2% from 66% with existing schedulers. The effect of a smaller scheduling time quantum in a virtual desktop infrastructure (VDI) setting was calculated and proved that the order of average overhead time per scheduler call remained unchanged [11].

Begam et al. proposed a method for the selection of routing path and server selection based on a prediction of traffic type and response time of the server under current load, response time, bandwidth, and server utilization by multiple regression-based searching [12].

A load-balancing algorithm based on ant colony optimization was proposed by Mishra and Anant [13]. The selection of the optimal path to the target depends on the strength of the ant's pheromone. Similarly, each node held a pheromone. Every row in the pheromone table represented a preferred path to the destination node, and every column represented the probability of choosing a neighbour as the next hop. At the choice point, the next hop node with the highest probability was chosen, and a random node was chosen if no pheromone was present. The table was updated

by increasing the probability of the chosen node and decreasing the probability of all other nodes.

Ouhame and Hadi proposed a modification in the local search section and fitness function value in the grey wolf optimization algorithm to improve throughput, energy consumption, and average network execution time in VM for cloud computing [14]. Load-balancing algorithms that distributed the load among a number of heterogeneous servers were developed by Kaur and Jain [15]. VMs from different data centers were created with a host specification that detailed the core processor, processing speed, memory, storage, etc. Each VM has been characterised by a weighted count proportional to the amount of RAM allocated to it and the current allocation count. The selection of VMs for load assignment is carried out by identifying an available VM with higher RAM. Once an assignment is made, the virtual machine ID is returned to the data center controller for updating the allocation count of VMs and the busy list. On finishing the request, the algorithm deallocates the VM and updates the busy list by removing the VM ID.

Honey bee behavior (HBB) in search of a food source was modeled as a scheduling algorithm in cloud computing by Miriam et al. [16]. The virtual machine workload was calculated by the HBB algorithm and classified as overloaded, balanced, and light-weighed. The task with high priority was removed from overloaded VMs and assigned to lightweight VMs. These tasks were playing the role of scout bees in the next step.

Ziyath and Subramaniyan consider the current status of the virtual machine (VM) in a cluster for calculating the placement value of the task in the queue and reshuffling. The performance of the system is validated by the time taken to allocate 1000 jobs but not by the response time or processing time of the jobs [17]. Kofani et al. have measured the performance of their priority-based optimized data center selection method with processing time and response time [18]. Jena and Mohanty proposed a two-phase load balancing technique in which the first phase is genetic algorithm-based resource allocation and the second phase is shortest task-first scheduling [19]. Hafiz proposed a load-balancing algorithm for heterogeneous cloud computing environments that improves efficiency and performance based on randomization and a greedy algorithm [20]. A short summary of important literature is tabulated in Table 1.

After a deep study of load-balancing algorithms discussed in various scholarly articles [21–24], genetic algorithms could be used for optimization, and biased parent selection will help the genetic algorithm to converge faster. Hence, the combination of a genetic algorithm and a biased random strategy are the probable candidates to improve the performance of load balancing in a cloud computing environment.

3. Problem Statement

Load balancing is a major concern and leads to performance degradation in resource allocation in cloud computing. Presently, in a cloud computing environment, the scheduling of virtual machines is carried out by considering only the current system state and ignoring the previous state of the system, which leads to load imbalance. While balancing loads the associated cost increases with a number of virtual machine migrations due to a granularity of VM resources and suspension of VM service during an enormous amount of data transfer in the migration process. A better solution, NRB-GA, is proposed for VM resource scheduling in cloud computing environments to improve performance. NRB-GA is a hybrid approach based on biased random and genetic algorithms. NRB-GA predicts the impact on the system of assigning a new task to VMs by utilizing historical data and the current state of the system. The solution that has the least effect on the system is picked. This solution ensures better load balancing by reducing the number of dynamic VM migrations during load balancing. The detailed design and implementation of NRB-GA for cloud computing environments that exhibit meritorious characteristics of both biased random algorithms and genetic algorithms are discussed in the following section.

4. NRB-GA Load-Balancing Algorithm

In the NRB-GA load-balancing algorithm, the population of virtual machines is divided into two groups based on their fitness. The top group is the best group because it contains the best individuals, and the bottom group is the nonbest group with the remaining candidates. The GA is restricted to choosing one individual randomly from each group for future genetic operations like crossover and mutation. Each member of a group has an equal probability of being selected. This biased the selection of at least one individual from the best group, which propagated to the next generation. Two important steps of the proposed NRB-GA load-balancing algorithm are explained here.

The first step is the distribution of virtual machines over physical machines (hosts) according to physical machine CPU capacity and processor speed. The largest number of virtual machines was housed in a host with highly qualified CPU capacity. For illustration, let us assume we need 6 VMs and have 3 hosts. The first host has one CPU with a processing speed of 10000 MIPS. The second one has two CPUs, each with a processing speed of 10000 MIPS. The third host has three CPUs, each with a processing speed of 10000 MIPS. Based on the host capacity, the first host takes one VM, the second host takes two VMs, and the third host takes three VMs.

The second step is the construction of an index table that records the loads of each virtual machine. The load balancer would update this index table upon assigning a load to a VM and that VM completing the request. Whenever the data center receives a request from a user, the NRB-GA load balancer algorithm initializes a population of nodes (VMs) randomly and evaluates the fitness of each populated VMs. After sorting the initialized VM in decreasing order of fitness value, the algorithm groups the VMs into two groups, with the top individuals as the best group and the remaining individuals as the nonbest group.

Table 1: A short summary of important literatures.

Authors [ref]	Title/year	Methods	Findings/remarks
Ragmani et al. [8]	An improved hybrid fuzzy-ant colony algorithm applied to load balancing in cloud computing environment/2019	Fuzzy logic module to calculate pheromone values for any colony optimization	Hybrid load-balancing algorithm Empirical results proved appropriate for handling complex networks
Akawee et al. [9]	Using resource allocation for seamless service provisioning in cloud computing/2022	Survey of recourses allocation	Highlights the importance of resource allocation and its relationship with load balancing, and discusses the limitations of resource allocation in cloud computing
Sethi et al. [10]	Efficient load balancing in cloud computing using fuzzy logic/2012	Fuzzy logic and round-robin scheduling	Hybrid method Fuzzy logic is used to identify the virtual machine with the minimum load. Round-robin scheduling for virtual machine assignment for requests
Hwang and Wood [11]	Adaptive dynamic priority scheduling for virtual desktop infrastructures/2012	Soft real-time scheduler	The effect of a smaller scheduling time quantum in a virtual desktop infrastructure (VDI) setting was calculated and proved that the order of average overhead time per scheduler call remained unchanged
Begam et al. [12]	Load balancing in DCN servers through SDN machine learning algorithm/2022	Multiple regression	Server selection is based on a prediction of the response time of the server as a function of current load, response time, bandwidth, and server utilization
Ouhame and Hadi [14]	Enhancement in resource allocation system for cloud environment using modified grey wolf technique/2020	Grey wolf optimization algorithm	Metrics studied are throughput, energy consumption, and average network execution time in VM for cloud computing
Miriam et al. [16]	An efficient job scheduling in isometric HPCLOUD using ZBLA optimization/ 2015	Honey bee behavior	The virtual machine workload was calculated by the HBB algorithm and classified as overloaded, balanced, and light-weighed. The task with high priority was removed from overloaded VMs and assigned to lightweight VMs. These tasks were playing the role of scout bees in the next step
Jena and Mohanty [19]	GA-based customer-conscious resource allocation and task scheduling in multicloud computing/2018	Genetic algorithm and shortest task first scheduling	Hybrid load-balancing algorithm Two-phase approach
Hafiz [20]	Efficient load balancing algorithm in cloud computing/2015	Randomization and greedy algorithm	Hybrid load-balancing algorithm A load-balancing algorithm for heterogeneous cloud computing environments that improves the efficiency

A biased random strategy is applied to generate a new population from both groups for crossover and mutation. The best VM found from the operation is chosen for the job, and the NRB-GA algorithm returns the VM ID number to the data center. The data center assigns the load to the selected VM and updates the index table.

4.1. Mathematical Model. Let P be the set of host machines in the whole system and represented as $P = \{P_1, P_2, P_3, \ldots, P_N\}$ where N is the total number of physical host systems and P_i is the individual host machine with identification number of i.

Each physical machine P_i is having a set of virtual machines $V_i = \{V_{i1}, V_{i2}, V_{i3}, \ldots, V_{im}\}$, where m is the number of VMs on the physical server P_i.

Let S_i is the distribution structure of the VM V arranged in physical machine P_i. Let $S = \{S_1, S_2, S_3, \ldots, S_N\}$, represents the distribution solution set. The sum of all running VMs on a physical machine P_i represents the load on a physical machine.

Let T be the duration to monitor historical data. At any point in time, the last T minutes are called the historical data zone, which would be used in solving the load balancing problem. The physical machine load of the duration T could be divided into n subsequent time intervals as $[(t_1 - t_0), (t_2 - t_1), (t_3 - t_2), \ldots, (t_n - t_{n-1})]$ by applying variation law.

Let $\mathrm{VL}(i,k)$ be the load of VM i in the interval k only when the load of VM V is stable in all periods of the interval $(t_k - t_{k-1})$. The average load of VM on physical server Pi during the period T is defined by the following equation:

$$\overline{\mathrm{VL}(i,T)} = \frac{1}{\mathrm{T}} \sum_{k=1}^{n} \mathrm{VL}(i,k) * (t_k - t_{k-1}). \quad (1)$$

Load of a physical machine P_i, during the duration T, is defined as the summation of the average load of all VMs

present in the physical machine P_i and is expressed as the following equation:

$$\mathrm{PL}(i,T) = \sum_{j=1}^{\mathrm{mi}} \overline{\mathrm{VL}(j,T)}. \tag{2}$$

By knowing the resource information of the virtual machine V, the load on the virtual machine V could be estimated as V' while distributing the VM to the system. Whenever the VM V is assigned to a physical server, the load of each Pi is calculated by the following given equation:

$$\mathrm{PL}(i,T)' = \begin{cases} \mathrm{PL}(i,T) + V', & \text{after distribute } V', \\ \mathrm{PL}(i,T), & \text{otherwise.} \end{cases} \tag{3}$$

After assigning VM V with Pi, system load is altered and the load adjustment factor should be calculated for load balancing. The load deviation $\sigma_i(T)$ in system due to solution Si is obtained by the following equation:

$$\sigma i(\mathrm{Si},T) = \sqrt{\frac{1}{\mathrm{N}} \sum_{i=1}^{N} \overline{\left(\mathrm{PL}(T)' - \mathrm{PL}(i,T)'\right)^2}}, \tag{4}$$

where $\mathrm{PL}(T)' = 1/N \sum_{i=1}^{N} \mathrm{PL}(i,T)'$.

The migration cost or cost advisor $\rho(S_\mathrm{i})$ for achieving load balancing with a solution is defined as the ratio of the number of virtual machines that need to be migrated (M') over the total number of virtual machines (M). The equation is given as follows:

$$\rho(\mathrm{Si}) = \frac{M'}{M}. \tag{5}$$

4.2. System Model. The proposed NRB-GA algorithm follows the same structure as the genetic algorithm with respect to population initialization but also introduces the following two methods to improve the processing time and response time metrics:

(1) A biased random technique to select individuals from the population for crossover and mutation.

(2) The fitness function used in our proposed method is considering the load on each virtual machine over a period of time (not on the instance) for reducing the number of migrations.

In GA, binary codes are used to mark the chromosome structure of genes [25]. We choose to represent every distribution solution as a tree structure that forms the chromosomes of genes. The root node of the tree represents the scheduling and managing node of the system, and the children of the root nodes (N-nodes) are the physical machines. The leaf nodes (M-nodes) of the tree are virtual machines. The virtual machines housed in one physical machine are represented as the children of that physical machine node in the tree.

4.2.1. Population Initialization. A spanning tree is used to initialize the population. The spanning tree is constructed with a physical machine set and a virtual machine set:

(i) The root node is the predefined management source node.

(ii) All physical machines are children of the root node.

(iii) All leaf nodes are VM nodes. VMs belong to the physical machines, and Pi are children of the node Pi.

4.2.2. Fitness Function. The genetic fitness of any individual is proportional to the number of descendants produced. In the NRB-GA load-balancing algorithm, the fitness function should be related to the quality of the solutions in the population. Solutions with a higher fitness value perform better, and vice versa. In the consecutive generations of the GA, the solutions with higher fitness values grew while solutions with fewer fitness values extinct. The fitness function chosen for the NRB-GA load-balancing algorithm should reduce the average load of the virtual machines and is defined by the following equation:

$$\mathrm{fi}(\mathrm{Si},T) = \frac{1}{\overline{\mathrm{VL}(i,T)}}, \tag{6}$$

where $\mathrm{VL}(i,T)$ is the average load of VM Vi.

4.2.3. Biased Random Strategy. In biased random sampling, elements in the sample space are associated with some biased probability, so that the accuracy of a few elements is higher than that of others. This biased random sampling technique is used in GA for the selection of parents with good fitness to generate the consecutive population. In the NRB-GA load-balancing algorithm, the fitness value of each solution in the current population is calculated. After sorting the solution candidates in decreasing order of their fitness value, they are grouped as the best group and nonbest group. The selection of a solution to form the next generation of the population is biased such that one solution from the best group is selected and another solution from either the nonbest group or population is used for crossover and mutation operations.

4.2.4. Crossover Operation. Since tree coding is used in the solution set, the normal crossover operation (exchanging some parts of the genes of the parents) does not work. Instead, replication is performed so that the child takes the same gene from the parents and assures the legitimacy of the leaf nodes. The crossover mechanism with a crossover probability value of Pc = 0.9 is explained as follows:

(i) Two parental solutions T_1 and T_2 are selected by applying the biased random strategy.

(ii) Two selected individuals are crossed over to get a new individual tree T_0. If the new individual maintains the the child-parent relation of the leaf nodes, then keep it, else discard it.

(iii) For different leaf nodes in the two parental individuals, first compute their selection probability according to the load of every VM, then based on selection probability distribute them as leaf nodes to

the least loaded nodes in the physical machine set until the distribution is completed. Selection probability based on fitness is defined in the following equation:

$$\mathbf{pi}(\mathbf{S}) = \frac{\mathbf{fi}(\mathbf{Si}, \mathbf{T})}{\sum_{\mathbf{i=1}}^{\mathbf{D}} \mathbf{fi}(\mathbf{Si}, \mathbf{T})}, \quad (7)$$

where fi(Si, T) is fitness of VM Vi in population and "D" is scale of the population

(iv) Repeat the crossover as mentioned above until the required next-generation population is produced T_0.

4.2.5. Mutation Operation. Mutation operations are essential in GA to preserve the diversity of the population and to avoid prematurity. Self-adaptive mutation probability (P_m) defined in the following equation, is used in the NRB-GA load-balancing algorithm:

$$\mathbf{P_m} = \frac{\mathbf{exp}\ (-\mathbf{1.5} * \mathbf{0.5t})}{\mathbf{D} * \sqrt{\mathbf{M}}}, \quad (8)$$

where t is the number of generations; D the scale factor of population; and M is the number of VMs.

4.3. Steps in NRB-GA Algorithm. The proposed NRB-GA algorithm goes through different steps to attain the finest system-level load balancing to ensure high performance in the cloud system. The flowchart of the NRB-GA load-balancing algorithm is shown in Figure 1.

The NRB-GA algorithm steps are:

(1) (Start) Initialize random population V of VMs.

(2) (Fitness) Calculate the fitness value fi(Si, T) of every VM in the given population using equation (6).

(3) (Group) Sort the population in decreasing order of fitness and group it into two: say the top V_b individuals, the best group, and the bottom V_{nb} individuals, the nonbest group.

(4) Copy the best group V_b to the next generation; say Y.

(5) (Biased Random) Repeat the following steps until a new population is created.

5.1 (Selection) Select one individual from the best group V_b and another individual from either the nonbest group V_{nb} or population V

5.2 (Crossover) Perform crossover operation among the selected individuals by using the crossover probability and generating the new offspring (V_o)

5.3 (Mutation) Perform mutation operation on (V_o) with the probability of mutation given in equation (9).

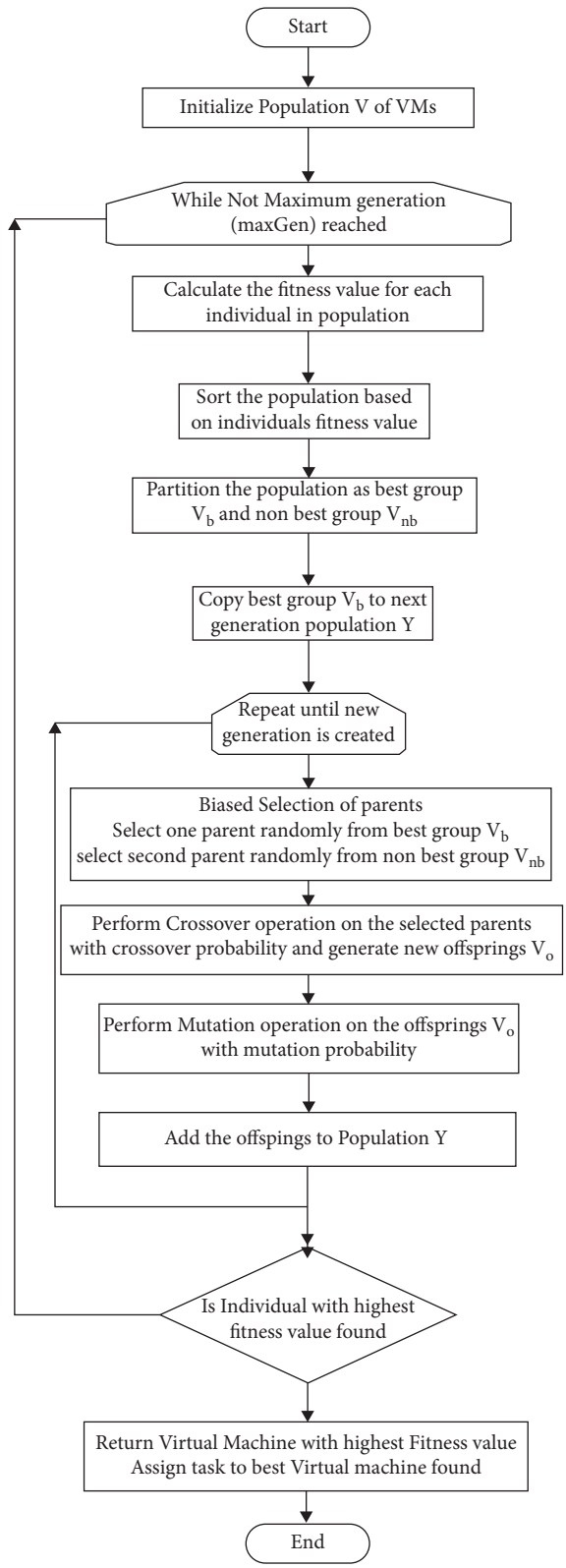

Figure 1: Flowchart for the proposed NRB-GA algorithm.

5.4 (Accepting) Add new offspring to the next generation of population ($Y = Y + V_o$)

(6) (Replace) Assign the current generation to the new generation ($V \leftarrow Y$).

(7) (Test) Check if the individual with the highest fitness value is found, and then

(a) Assign the task to the identified VM with the highest fitness value
(b) End algorithm

(8) (Loop) go to step 2.

5. Experimental Design

The implemented NRB-GA load-balancing algorithm is tested with the simulation toolkit CloudAnalyst by creating simulated scenarios for social network applications such as Facebook. On June 30, 2017, Facebook user base distribution with respect to continents was as follows: North America: 263 million users; South America: 370 million users; Europe: 343 million users; Asia: 736 million users; Africa: 160 million users; and Oceania: 19 million users [26]. This data, reduced by a scale factor of 10, is chosen to evaluate the performance of the implemented NRB-GA load-balancing algorithm and to correlate the use of the cloud in similar social networking applications. The cloud analyst simulator is used to create a hypothetical configuration that divides the world into six regions to correspond to the six continents in the world. Six user bases are modeled according to the collected data to represent users from six major continents. The experiments are limited by assuming all regions are in the same time zone. Another important assumption is 5% of registered users are online during peak hours simultaneously and only 1/10 of these peak-time users are online during off-peak hours. The NRB-GA load-balancing algorithm is tested by simulating a new request from each online user every five minutes.

A cloud environment is simulated with 5 hosts in each data center, dedicated to applications. The host machines are having X86 architecture, VM monitor, Xen, Linux operating system, two GB RAM, and 100 GB storage. Physical host machines in data centers are different in terms of the number of CPU cores and processing speed. It is assumed that the first host has 4 cores with a processing speed of 2000 MIPS, the second host has 5 cores with a processing speed of 5000 MIPS, the third host has 3 cores with a processing speed of 9000 MIPS, the fourth host has dual cores with a processing speed of 10000 MIPS, and the fifth host has a single core with a processing speed of 15000 MIPS. The application image size that occupies the VM is 100 MB. Each virtual machine is configured to have 1 GB of RAM and 10 MB of available bandwidth. Resources are scheduled to VMs by time sharing policy. Users are grouped by a factor of 1000, and requests are grouped by a factor of 100. It is assumed that 250 instructions need to be executed to process each user request.

6. Results and Discussion

Experiments were conducted on single, two, and three data centers with 6 different cloud configurations to evaluate the performance in terms of processing time and response time by using the CloudAnalyst simulator.

6.1. Experiment 1: Single Data Center. In the first experiment, the performance of the proposed NRB-GA load-balancing algorithm is studied without considering the effect of network delay by assuming there is only one centralized data center (DC) to process all user requests around the world. All six user bases are in the same region and use the same data centers (Table 2). Each machine has a different number of CPU cores and processing speed, as shown in Table 3. The configuration data used in the experiment to configure a single data center having 50, 75, and 100 VMs, and its specifications, are shown in Table 4.

The response time and processing time obtained by the simulation are tabulated in Tables 5 and 6, respectively. From Figures 2 and 3, it is easily comparable and observed that the proposed NRB-GA algorithm outperforms the existing algorithms in both response time and processing time metrics.

6.2. Experiment 2: Two Data Center. In the second experiment, the performance of the implemented NRB-GA load-balancing algorithm is studied by considering the effect of network delay using two data centers. In this setup scenario, all user base configurations are in two different data centers and in different regions. Each machine within a DC is heterogeneous, having a different number of CPUs and speeds.

In this experiment, two data centers are defined in such a way that each has 50, 75, or 100 VMs, with a combination of each VM (i.e., 50 and 75, 50 and 100, 75 and 100) of the cloud configuration allocated to the application. The other DC specifications are the same as in experiment 1, (Table 3). The effect of network delay is tested by distributing the user base among six regions: UB1 in NorthAmerica, UB2 in South America, UB3 in Europe, UB4 in Asia, UB5 in Africa, and UB6 in Oceania.

The configuration files used for simulation are shown in Tables 7 and 8. The simulated response time and processing time obtained for experiment 2 are tabulated in Tables 9 and 10, respectively. Figures 4 and 5 show a comparison that the proposed NRB-GA algorithm outperfoms the existing algorithms in both response time and processing time metrics when a network delay is introduced by having two data centers and a user base from different regions of the globe.

6.3. Experiment 3: Three Data Centers. The objective of this experiment is to study the effect of the NRB-GA load-balancing algorithm on network delay in a heterogeneous host environment using three data centers. Six different user base locations are raising requests to all three data centers. Each machine within DCs is also heterogeneous, having a different number of CPUs and speeds, like in experiment 2 with two DCs.

In this configuration scenario, three DCs are defined, each having 50, 75, and 100 VMs, and a combination of each

Table 2: User base configuration for one data center.

User base name	Region	Single user request per hour	Data size per request (bytes)	Start of peak hours (local)	End of peak hours (local)	Avg. peak-hour users	Avg. off-peak hour users
UB1	0	12	100	07	09	1315000	131500
UB2	0	12	100	07	09	1850000	185000
UB3	0	12	100	07	09	1715000	171500
UB4	0	12	100	07	09	3680000	368000
UB5	0	12	100	07	09	800000	80000
UB6	0	12	100	07	09	95000	9500

Table 3: Datacenter configuration.

Id	RAM memory (Mb)	Secondary storage (Mb)	Available BW (Mb)	Number of processor cores	Processor speed	VM policy
0	2048	102400	1024	4	2000	TIME_SHARED
1	2048	102400	1024	5	5000	TIME_SHARED
2	2048	102400	1024	3	9000	TIME_SHARED
3	2048	102400	1024	2	10000	TIME_SHARED
4	2048	102400	1024	1	15000	TIME_SHARED

Table 4: Application deployment configuration for one data center.

No. of data centers	Cloud configuration	No. of VMs	Image size (MB)	Memory (MB)	BW (MB)
One	CC1	50	100	1024	10
	CC2	75	100	1024	10
	CC3	100	100	1024	10

Table 5: Response time—one data center.

Cloud configuration	NRB-GA	BRS	RR	ESCE	GA
CC1	435.10	561.46	576.04	561.50	559.14
CC2	435.06	561.45	575.46	560.81	558.41
CC3	350.13	561.79	576.29	561.52	559.18
Average response time	**406.7633**	561.5667	575.93	561.2767	**558.91**
Average response time reduction percentage ((min (BRS, RR, ESCE, GA)-NRB-GA)/min (BRS, RR, ESCE, GA))					**27.22%**

Proposed NRB-GA algorithms response time is compared with least response time of Existing algorithms and percentage of reduction in response time is measured with this two quantities.

Table 6: Processing time—one data center.

Cloud configuration	NRB-GA	BRS	RR	ESCE	GA
CC1	367.65	492.23	511.96	492.27	490.41
CC2	367.61	492.22	511.38	491.58	489.68
CC3	282.68	492.56	512.21	492.29	390.45
Average processing time	**339.3133**	492.3367	511.85	492.0467	**456.8467**
Average processing time reduction percentage ((min (BRS, RR, ESCE, GA)-NRB-GA)/min (BRS, RR, ESCE, GA))					**25.73%**

Proposed NRB-GA algorithms response time is compared with least response time of Existing algorithms and percentage of reduction in response time is measured with this two quantities.

VM's (i.e., 50, 75, and 100) cloud configuration is assigned to the application. Other DC configurations are shown in experiment 1, (Table 3) and the user base configuration is shown in experiment 2 (Table 7),. The effects of network delay with 3 data centers are studied by distributing the user-based, as shown in experiment 2 and in Table 8. The cloud configuration of the three DCs is shown in Table 11. The simulated response time and processing time obtained for experiment 3 are tabulated in Tables 12 and 13, respectively. Figures 6 and 7 show that the proposed NRB-GA algorithm outperfoms the existing algorithms in both response time and processing time metrics when a network delay is introduced by having three data centers and a user base from different regions of the globe.

6.4. Discussion. In the first experiment using a single data center, the experiment showed that the NRB-GA algorithm achieved a better result than the other existing algorithms. The NRB-GA algorithm using a single DC recorded the best average response time, comprising 456.41 (ms) using 50 VMs, 456.28 (ms) using 75 VMs, and 452.14 (ms) using 100

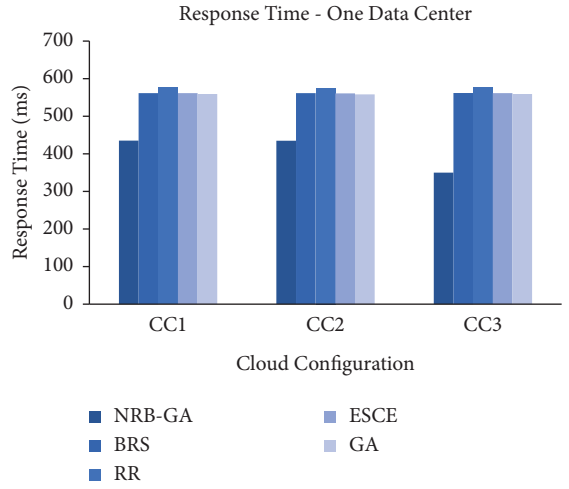

Figure 2: Response time comparison for one data center.

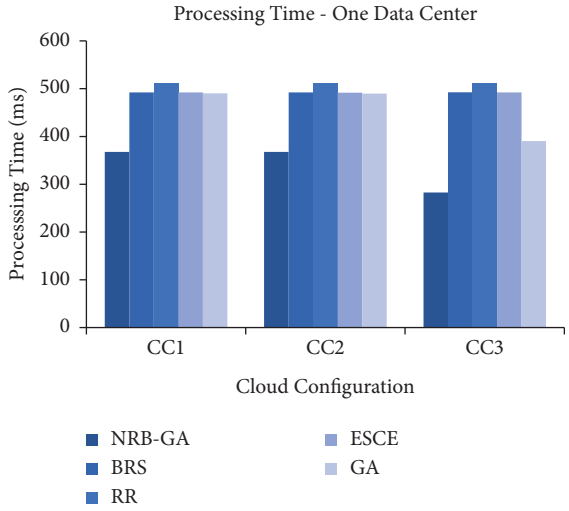

Figure 3: Processing time comparison for one data center.

Table 7: Application deployment configuration for two data centers.

Number of data centers	Cloud configuration	Number of VMs	Image size (Mb)	Memory (Mb)	BW (Mb)
Two	CC1	50	100	1024	10
	CC2	75	100	1024	10
	CC3	100	100	1024	10
	CC4	50, 75	100	1024	10
	CC5	50, 100	100	1024	10
	CC6	75, 100	100	1024	10

Table 8: User base configuration for two data centers.

User base name	Region	Single user request per hour	Data size per request (bytes)	Start of peak hours (local)	End of peak hours (local)	Avg. peak-hour users	Avg. off-peak hour users
UB1	N-America	12	100	13:00	15:00	1315000	131500
UB2	S-America	12	100	15:00	17:00	1850000	185000
UB3	Europe	12	100	20:00	22:00	1715000	171500
UB4	Asia	12	100	01:00	03:00	3680000	368000
UB5	Africa	12	100	21:00	23:00	800000	80000
UB6	Oceania	12	100	09:00	11:00	95000	9500

Table 9: Response time—two data centers.

Cloud configuration	NRB-GA	BRS	RR	ESCE	GA
CC1	466.86	597.88	617.31	607.73	610.49
CC2	461.81	592.18	613.53	603.91	606.01
CC3	461.45	592.18	611.4	602.64	605.91
CC4	456.71	589.17	609.53	600.95	604.88
CC5	457.69	589.76	609.53	599.31	603.31
CC6	458.14	542.61	606.69	593.09	594.54
Average response time	**460.44333**	**583.96333**	611.33167	601.27167	604.19
Average response time reduction percentage ((min (BRS, RR, ESCE, GA)-NRB-GA)/min (BRS, RR, ESCE, GA))					**21.15%**

Proposed NRB-GA algorithms response time is compared with least response time of Existing algorithms and percentage of reduction in response time is measured with this two quantities.

Table 10: Processing time—two data centers.

Cloud configuration	NRB-GA	BRS	RR	ESCE	GA
CC1	399.41	528.65	553.23	538.5	541.76
CC2	394.36	522.95	549.45	534.68	536.28
CC3	394	522.95	547.32	533.41	437.18
CC4	389.26	519.94	545.45	531.72	436.15
CC5	390.24	520.53	545.45	530.08	434.58
CC6	390.69	520.34	542.61	523.86	425.81
Average processing time	**392.993333**	522.56	547.25167	532.04167	**468.62667**
Average processing time reduction percentage ((min (BRS, RR, ESCE, GA)-NRB-GA)/min (BRS, RR, ESCE, GA))					**16.14%**

Proposed NRB-GA algorithms response time is compared with least response time of Existing algorithms and percentage of reduction in response time is measured with this two quantities.

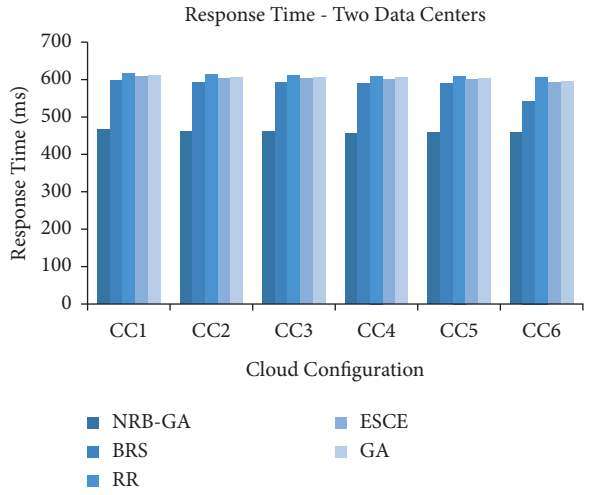

Figure 4: Response time comparison for two data centers.

VMs. It also recorded the best average processing time, containing 388.96 (ms) using 50 VMs, 388.83 (ms) using 75 VMs, and 384.69 (ms) using 100 VMs.

In the second experiment using two DCs, by considering the network delay, the NRB-GA algorithm recorded the best response time, including 466.86 (ms) in CC1, 461.81 (ms) in CC2, 461.45 (ms) in CC3, 456.71 (ms) in CC4, 457.69 (ms) in CC5, and 458.14 (ms) in CC6. The NRB-GA algorithm also recorded average processing times such as 399.41 (ms) in CC1, 394.36 (ms) in CC2, 394.00 (ms) in CC3, 389.26 (ms) in CC4, 390.24 (ms) in CC5, and 390.69 (ms) in CC6.

With increased DCs to three in the third experiment, the NRB-GA algorithm achieved the best response time of 456.41 (ms) in CC1, 456.28 (ms) in CC2, 452.14 (ms) in CC3,

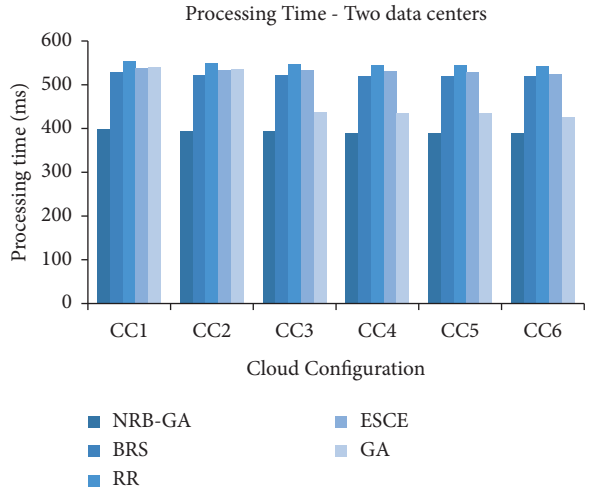

Figure 5: Processing time comparison for two data centers.

Table 11: Application deployment configuration for three data centers.

No. of data centers	Cloud configuration	No of VMs	Image size (Mb)	Memory (Mb)	BW (Mb)
Three	CC1	50	100	2048	10
	CC2	75	100	2048	10
	CC3	100	100	2048	10
	CC4	50, 75, 100	100	2048	10

Table 12: Response time—three data centers.

Cloud configuration	NRB-GA	BRS	RR	ESCE	GA
CC1	456.41	589.26	607.21	594.73	594.73
CC2	456.28	587.28	608.53	594.91	594.01
CC3	452.14	583.18	602.8	593.64	596.51
CC4	452.11	582.46	602.83	592.95	596.38
Average response time	**454.235**	**585.545**	605.3425	594.0575	595.4075
Average response time reduction percentage ((min (BRS, RR, ESCE, GA)-NRB-GA)/min (BRS, RR, ESCE, GA))					**22.43%**

Table 13: Processing time—three data centers.

Cloud configuration	NRB-GA	BRS	RR	ESCE	GA
CC1	388.96	520.03	543.13	525.5	526
CC2	388.83	518.05	544.45	525.68	524.28
CC3	384.69	513.95	538.72	524.41	427.78
CC4	384.66	513.23	538.75	523.72	427.65
Average processing time	**386.785**	516.315	541.2625	524.8275	**476.4275**
Average processing time reduction percentage ((min (BRS, RR, ESCE, GA)-NRB-GA)/min (BRS, RR, ESCE, GA))					**18.81%**

Proposed NRB-GA algorithms response time is compared with least response time of Existing algorithms and percentage of reduction in response time is measured with this two quantities.

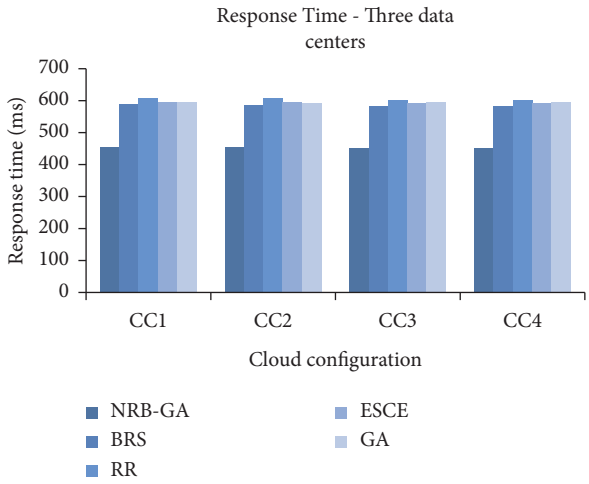

Figure 6: Response time comparison for three data centers.

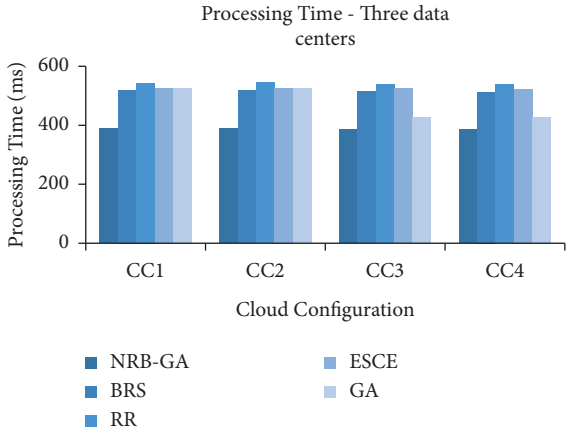

Figure 7: Processing time comparison for three data centers.

and 452.11 (ms) in CC4, and the best processing time of 388.96 (ms) in CC1, 388.83 (ms) in CC2, 384.69 (ms) in CC3, and 384.66 (ms) in CC4.

7. Conclusion

To improve the performance of cloud computing environments, we designed an NRB-GA load-balancing algorithm that inherits the meritorious characteristics of both genetic algorithms and biased random strategies. The NRB-GA load-balancing algorithm considers various factors that include the previous state (historical data), current resource information, the number of processor cores in the CPU, and the processing speed of the CPU to achieve a low response time and processing time.

From the three conducted experiments, analyses of the results of the NRB-GA indicate that the average response time is reduced by 27.22%, 21.15%, and 22.34%, and the processing time is reduced by 25.73%, 16.14%, and 18.82% for one, two, and three data centers, respectively. This reduction in processing time and response time is achieved in the load balancing algorithm NRB-GA by considering the load of each virtual machine over a period of time T instead of the current load on each virtual machine. It is evident that the proposed NRB-GA algorithm for load balancing outperforms other existing algorithms significantly. In general, performance has improved in a cloud computing environment with heterogeneity in processor processing capacity (processor power).

8. Future Works

Load balancing is one of the major factors in improving the performance of the cloud computing environment. We discussed only improving the performance of three data centers, but there are still other approaches that can be applied to balance the load in a cloud computing environment with more data centers. It is planned to implement a new load-balancing algorithm to improve the service broker policy. NRB-GA is tested and studied to see how response time and processing time are affected by various CPU capacities. As suggested, perform a study on other factors such as memory, bandwidth, and storage. The study can be extended by analysing the other parameters, such as effective utilization of resources, cost, failover, etc. The NRB-GA algorithms' performance is studied by simulating only the normal state, but there are still other states that can be studied, such as the burst load state.

Data Availability

The data used to support the findings of this study are included within the article. The source code used to support the findings of this study are available from the corresponding author upon request.

Disclosure

Haramaya University sponsored the researcher to peruse a postgraduate program only, not specifically for this study and research

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This work was supported by the Haramaya University and Mekele University.

References

[1] M. Kaushik and S. Majhi, "A state-of-art on cloud load balancing algorithms," *International Journal of Computing and Digital Systems*, vol. 9, no. 2, pp. 201–220, 2020.

[2] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[3] "History of Cloud Computing," 2017, https://www.javapoint.com/history-of-cloud-computing.

[4] G. S. Kushwah and V. Ranga, "Distributed denial of service attack detection in cloud computing using hybrid extreme

learning machine," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 29, no. 4, pp. 1852–1870, 2021.

[5] M. Xu, W. Tian, and R. Buyya, "A survey on load balancing algorithms for virtual machines placement in cloud computing," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 12, 2017.

[6] M. Randles, D. Lamb, and A. Taleb-Bendiab, "A comparative study into distributed load balancing algorithms for cloud computing," in *Proceedings of the IEEE International Conference on Advanced Information Networking and Applications Workshops*, pp. 551–556, Perth, Australia, April 2010.

[7] J. Hu, J. Gu, G. Sun, and T. Zhao, "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment," in *Proceedings of the 3rd International Symposium on Parallel Architectures, Algorithms and Programming*, pp. 89–96, IEEE, Liaoning, China, December 2010.

[8] A. Ragmani, A. Elomri, N. Abghour, K. Moussaid, and M. Rida, "An improved hybrid fuzzy-ant colony algorithm applied to load balancing in cloud computing environment," *Procedia Computer Science*, vol. 151, pp. 519–526, 2019.

[9] M. M. Akawee, M. A. Ahmed, and R. A. Hasan, "Using resource allocation for seamless service provisioning in cloud computing," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 26, no. 2, pp. 854–858, 2022.

[10] S. Sethi, S. Anupama, and K. S. Jena, "Efficient load balancing in cloud computing using fuzzy logic," *IOSR Journal of Engineering*, vol. 2, no. 7, pp. 65–71, 2012.

[11] J. Hwang and T. Wood, "Adaptive dynamic priority scheduling for virtual desktop infrastructures," in *Proceedings of the 2012 IEEE 20th International Workshop on Quality of Service*, pp. 1–9, Coimbra, Portugal, June 2012.

[12] G. S. Begam, M. Sangeetha, and N. R. Shanker, "Load balancing in DCN servers through SDN machine learning algorithm," *Arabian Journal for Science and Engineering*, vol. 47, no. 2, pp. 1423–1434, 2022.

[13] R. Mishra and J. Anant, "Ant colony optimization: a solution of load balancing in cloud," *International journal of Web & Semantic Technology*, vol. 3, no. 2, pp. 33–50, 2012.

[14] S. Ouhame and Y. Hadi, "Enhancement in resource allocation system for cloud environment using modified grey wolf technique," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 20, no. 3, pp. 1530–1537, 2020.

[15] A. Kaur and A. Jain, "A modified performance oriented approach for load balancing in cloud computing," *International Journal of Computer Application*, vol. 131, no. 10, pp. 13–16, 2015.

[16] D. D. H. Miriam, B. Prabha, and J. F. Lilian, "An efficient job scheduling in isometric HPCLOUD using ZBLA optimization," *Procedia Computer Science*, vol. 50, pp. 307–315, 2015.

[17] S. P. M. Ziyath and S. Subramaniyan, "An improved Q-learning-based scheduling strategy with load balancing for infrastructure-based cloud services," *Arabian Journal for Science and Engineering*, vol. 47, no. 8, pp. 9547–9555, 2021.

[18] N. A. Kofahi, T. Alsmadi, M. Barhoush, and M. A. Shannaq, "Priority-based and optimized data center selection in cloud computing," *Arabian Journal for Science and Engineering*, vol. 44, no. 11, pp. 9275–9290, 2019.

[19] T. Jena and J. R. Mohanty, "GA-based customer-conscious resource allocation and task scheduling in multi-cloud computing," *Arabian Journal for Science and Engineering*, vol. 43, no. 8, pp. 4115–4130, 2018.

[20] J. Y. Hafiz, "Efficient load balancing algorithm in cloud computing," PG Thesis, Islamic University, Gaza, 2015.

[21] S. K. Mishra, B. Sahoo, and P. P. Parida, "Load balancing in cloud computing: a big picture," *Journal of King Saud University—Computer and Information Sciences*, vol. 32, no. 2, pp. 149–158, 2020.

[22] S. Afzal and G. Kavitha, "Load balancing in cloud computing—a hierarchical taxonomical classification," *Journal of Cloud Computing*, vol. 8, no. 1, 2019.

[23] D. A. Shafiq, N. Jhanjhi, and A. Abdullah, "Load balancing techniques in cloud computing environment: a review," *Journal of King Saud University—Computer and Information Sciences*, vol. 34, no. 7, pp. 3910–3933, 2022.

[24] K. Balaji, P. Sai Kiran, and M. Sunil Kumar, "Load balancing in cloud computing: issues and challenges," *Turkish Journal of Computer and Mathematics Education*, vol. 12, no. 2, pp. 3077–3084, 2021.

[25] "Introduction to genetic algorithm," 2017, http://www.obitko.com/tutorials/genetic-algorithms/crossover-mutation.php.

[26] "Facebook subscriber stats," 2017, https://www.internetworldstats.com/Facebook.htm.