Cloud Computing Load Balancing using Genetic and Throttled Hybrid Algorithm

Shymaa G. Eladl [#1], Nesreen I. Ziedan[*2], Tamer S. Gaafar[#3]
Computer and Systems Engineering Department, Faculty of Engineering Zagazig University, Egypt.
[1] eng.sheladl@gmail.com
[2] ziedan@outlook.com
[3] tamer_samy_gaafar@yahoo.com

***Abstract*— The demand for cloud computing resources is increasing due to its accessibility everywhere at any time. When the number of clients for cloud services increases, the load on the cloud nodes becomes high. This status requires load balancing to evenly distribute client requests among the available Virtual Machines (VMs) in a Data Center (DC). There are different standard dynamic load balancing techniques, such as Throttled and Active Monitoring. In this paper, a Genetic Algorithm (GA) is incorporated with a throttled to improve load balancing. The improvement is achieved by enhancing the overall response time, the data center processing time, and the maximum resource utilization. Simulation results show the improved performance of the proposed method compared to the ESCE and Throttled.**

Keyword - Cloud Computing, Load Balancing, Genetic Algorithm, Cloud Analyst

I. INTRODUCTION

Cloud computing technology has seen a speedy growth in recent years. It has affected the growth in several sub-technologies, like storage, distributed networks, virtualization, participation, and connectivity. In [1, 2], a cloud is considered as a distributed system that can handle diverse resource requirements by users. The rule system of a cloud-user relationship is planned by the Service Level Agreement (SLA), which is an agreement between a user and a service provider. As indicated in[3], the physical structure and repairing system can be achieved by the cloud, not the user. This automatically decreases the total cost and increases system efficiency. As indicated in [4], cloud computing gives an easy way to hold data and files, and it includes the following features: virtualization, distributed computing, and web services. Any complex task that calls for enormous computational resources can be serviced by cloud computing using distributed resources in a decentralized style [5].

Although cloud computing has many countenances, there are obstacles as load balancing over the resources and task scheduling [6]. Task scheduling in a cloud environment is a problem of specifying tasks to an appropriate machine to finish their work. A task should be done within a given period. The cloud task scheduler restores the information from the cloud service manager about the case of available resources [7]. Therefore, the scheduling of task problem can be qualified as the method of finding out a model mapping for execution of user tasks with the aim of reaching the desired goals [8]. Algorithms of task scheduling in cloud computing can be done depending on diverse objectives such as balancing the load, minimizing waiting time, and maximizing the utilization of resources and the throughput of the full system. Therefore, an efficient task scheduling algorithm aims to balance diverse and conflicting parameters together at the same time[9]. Moreover, resources are not utilized efficiently due to the rise in the load so for that reason load balancing is required [10].

Load balancing is the approach of redistributing the whole load into separate nodes to guarantee that no node is overloaded, doing very little work, or idle [11]. Load balancing qualifies network resources for best response and performance and provides high gratifications to consumer [12]. As a result, providing effective load balancing techniques is key to the success of cloud computing [6]. Balancing became one of the necessary interest in cloud computing since it is not possible to predict the requests number that is rolled at each second. The inability to predict is due to changing the behavior of the cloud. Therefore, load balancing algorithms can be classified depending on the system state as static and dynamic [13, 14].

Dynamic load balancing works depending on the dynamic changes in the status of the node, i.e., it gathers, keeps and analyzes data about the entire system status. If some node has missed, it will not halt the work of the entire network, but it will affect the system performance. Some of the dynamic algorithms are Ant Colony Optimization (ACO), Honey Bee Foraging, Genetic Algorithm (GA) Active Monitoring, and Throttled.

This paper introduces an improvement to load balancing. A Load Balancing algorithm is proposed to improve Load Balancing in a heterogeneous cloud computing environment through improving the overall response time, DC Processing Time, and the maximum resource utilization. The proposed algorithm incorporates a GA with a throttled, and it is called Dynamic Throttled Genetic Algorithm (DTG).

II. RELATEDWORK

This section provides an overview of some load balancing techniques in cloud computing.

The performance of a load balancing algorithm in cloud computing is measured by the following metrics:

Execution Time (*ExT*): It is the time taken to execute the given set of tasks on VM, which is defined by [15] as shown in Eq. (1):

$$ExT = LT/VMP \quad (1)$$

Where *VMP* is the power processing of VM, *LT* is the length of the task.

Makespan: It is defined as the total execution time of all the tasks. This metric must be reduced to reduce the cost and execution time. It is the total amount of time required to complete a group of tasks which indicates the maximum completion time. It can be calculated by [15-17]

$$\text{Makespan} = \sum_{i=0}^{n} Executiontime(Tasks_i) \quad (2)$$

$$\text{Makespan} = \text{Max (Comp\_Time [i, j])} \quad (3)$$

$\{1 \leq I \leq N, 1 \leq j \leq M\}$

Where, Comp_Time [i, j] is the time at which task i ends on VM j.

Completion Time (*CT*): It is the sum of the Execution Time (ExT) of all the previous tasks and execution time (ExTm) of the present task allocated in the same VM, which can be calculated by [15] as shown in Eq. (4):

$$CT = ExTm + \sum_{i=1}^{m-1} ExT \quad (4)$$

Response Time (*RT*): It represents the total amount of time taken by the load balancing algorithms to respond to a user. This metric must be reduced and can be calculated by [15] as shown in Eq. (5):

$$RT = \sum_{i=1}^{n} (CT_i + SB_i) \quad (5)$$

Where, *CT* is the completion time of a task, *SB* is the submission time of task. The average of response time for each VM is calculated by[15] as shown in Eq. (6):

$$\text{Avg.}RT = RT/N \quad (6)$$

Garg, Gupta, and Dwivedi proposed an Enhanced Active Monitoring Load Balancing (EAMLB) method to reduce the response time in cloud environments. EAMLB got a better response time than active monitoring and Round Robin (RR). Cloud analyst is used in the comparison between active monitoring, RR and proposed algorithm (EAMLB). Results showed the benefit that one VM will not be assigned in a continuous manner if it is the least loaded [18]. Kulkarni and Annappa suggested an algorithm that spreads the load equally throughout all the VMs even when the frequency of requests is high during peak hours. This approach aims at guaranteeing faster response times to consumers. It is observed that the active load balancer algorithm (packaged in cloud analyst) produced load imbalance. The suggested algorithm repaired the problem of the active balancer algorithm by using a reservation table among the phase of the chosen and assignment of VMs[19].

Shakir and Razzaque discussed some load balancing algorithms and the simulation was performed using the cloud analyst tool to test the performance. The result showed that RR is the best compared to the others [20].Nishad, Kumar, Bola, Beniwal, and Pareek suggested RR policy. The RR policy was worked effectively when it concerned with resource utilization. The total cost was the same in experimentation when compared with a traditional algorithm such as Compare and Balance, VectorDot, and Throttled. The choice of a suitable DC for carrying out a task is an important advantage to develop the performance of the cloud [21].

Das, George, and Jaya introduced A new algorithm by merging Weighted Round Robin (WRR) in Honeybee inspired algorithm with a view to obtain the least processing and response time. Where tasks with a priority are processed first. The honeybee algorithm allocated weights to each VM, and the VM is chosen according to the task requirement from resources. Tasks with no priority are processed later using WRR. Experimental results showed that the proposed algorithm gives better response and processing time [22].Makasarwala and Hazari suggested a GA based technique for load balancing. For population initialization, the priority request is taken into account according to their time. The idea from the priority is to obtain actual world visualization. The proposed technique is done using cloud analyst. Results showed that the suggested technique performs better than RR and throttled. The suggested technique also gives better average response time compared with previously available techniques[23].

Rjoub and Bentahar proposed a machine learning algorithm by using a multi-standard decision to get better performance. The main goal of their work is to reduce the Makespan of a given task. Their technique was simulated by the cloud sim toolkit package. Experiment results showed that the proposed algorithm reduced the execution time and got better performance of the load balancing [24].Sadia, Jahan, Rawshan, Jeba, and Bhuiyan proposed a new strategy that carried out the division of loads through VM that relies on priority. Their aim was to maximize the throughput with minimum execution time. To achieve that, the VMs were arranged based on their processing powers, and tasks are allocated to the VMs that rely on their instruction numbers and priorities. The proposed strategy was tested using cloud Sim, and the results proved that the performance of their strategy is better than other conventional algorithms [25].

Aruna, Bhanu, and Karthik proposed a load balancing technique using a Joint Firefly algorithm (FA) and Particle Swarm Optimization (PSO). The main goal was to balance the load of the entire system while at the same time minimize the makespan of tasks. This strategy has been simulated with cloud sim toolkit package. The results proved that the proposed algorithm gave better performance than the Min–Min, PSO, and FCFS methods [26]. Jena proposed a method that focused on task scheduling using a Multi-Objective nested Particle Swarm Optimization (MOPSO) to make the best of energy and processing time. The result was obtained from cloud sim and was compared with BRS and Random Scheduling Algorithm (RSA). MOPSO gave an ideal balance results for multiple objectives and helped in reducing the number of failed tasks. MOPSO reduced 30% of energy consumption and 25% of makespan compared with other approaches [27].

Garg, Dwivedi, and Chauhan proposed a method that focuses on load balancing to decrease the status of overload or underload that leads to getting better performance of cloud on VMs. Comparative analysis was done using cloud analyst. The STVMLB is prepared by making an alteration in the concept of throttled algorithm. STVMLB has raised the utilization of VM better than throttled and active monitoring. It achieved a better result as to efficiently assign the coming request and increase the response time in a cloud environment, although the response time was not better than a throttled algorithm [28].Domanaland Reddy suggested a hybrid algorithm gathering the methodology of divide-and-conquer and throttled algorithms known as DCBT. The goal was to minimize the total execution time and maximize resource utilization. The result proved that DCBT made use of the VMs more effectively while bringing down the execution time of the tasks by 9.972% compared to the modified throttled technique[29].

Geethu, Vasudevan, nd International Symposium on Big, and Cloud Computing Challenges focused on Min-Min and Max-Min load balancing algorithms. Comparison between Min-Min and Max-Min algorithms showed that makespan is reduced for Max-Min compared with Min-Min, so the Max-Min outperformed the Min-Min [30]. Babu, Joy, and Samuel suggested a bee colony for effective load balancing, which relies on the foraging style of honey bees. Tasks taken away from overloaded VMs were considered as honey bees, and underloaded VMs were considered as the food sources. The suggested method considered the priorities of tasks in the waiting queues and attempted to obtain less response time and decrease the number of task migrations. The experimental result showed a reduction in the makespan and gave better performance to the consumers [31].

Table1. Summary of the reviewed state-of-the-art

References	Algorithm	Main objective	Metrics	Compared Algorithms	Advantages
[32](2017)	Enhanced Honeybee Inspired Load Balancing Algorithm	Allocate the task according to their priority, resource requirement and by calculating the computing power.	Minimize the response time, minimize processing time and balance the load.	Honeybee Inspired Load Balancing Algorithm using cloud analyst.	Enhanced honeybee inspired load balancing algorithm gives less response time and processing time.
[33](2016)	Round Robin, Throttled and Equally Spread Current Execution	Calculating the performance of load balancing techniques.	Minimum response time and minimize VM Cost.	RR, Throttled and Equally Spread Current Execution using cloud analyst.	Throttled give a better performance than the others, as it uses a threshold and accessible VM list for forbidding serve the load by overloaded VMs.
[34](2016)	Modified Active Monitoring Load Balancer	Allocate the incoming request to the obtainable VMs wisely rely on their priority, state and memory utilization.	Minimize response time	RR, Active Monitoring and Throttled using cloud analyst.	A suggested algorithm does better than the other three algorithms on the base of response time.
[35](2016)	Ant Colony Optimization	To develop the entire response times of services in the cloud.	Minimize response time.	RR, Throttled using cloud analyst.	This algorithm is highly appropriate with the manner of a distributed network and raises the algorithm robustness.
[36](2015)	A combination of a GA and gravitational emulation local search (GEL)	The suggested algorithm attempted to reduce the makespan as well as possible decrease the number of VMs who are going to lose their deadlines.	Minimizing the makespan and improve the system load.	First Come First Serve (FCFS), Stochastic Hill Climbing (SHC), GA and ACO using cloud analyst.	Suggested algorithm better than the others, improved the response time and ensure the quality of service (QoS) requirement to the user.
[37](2015)	Dynamic Load Balancing Algorithms	Comparison of three dynamic load balancing algorithms relies on their performance.	Improve response time, processing time, performance and Cost.	Modified throttled, FCFS and PSO using cloud analyst.	PSO is more effective as it has lower response time and cost.
[28](2015)	Synchronized Throttled Load Balancing Algorithm	STVMLB is prepared by working alteration in the notion of throttled to develop the performance of cloud.	Maximize the resources utilization and minimize average response time.	Throttled, RR and Active Monitoring using cloud analyst.	STVMLB get better utilization of VMs rather than active monitoring and throttled algorithms.
[38](2015)	Dynamic Load Management Algorithm	An algorithm has been suggested for allocation of the whole coming request efficiently between the VMs.	Minimize processing time, response time and total cost.	VM Assign Algorithm using cloud analyst.	It allocates the load between VMs effectively with amended time in comparison with the other algorithms. Also, it has a minimum response time and suitable resource utilization.

III. METHODOLOGY

A. Problem Statement and Definition

When requests come from users randomly, some servers could undergo a heavy workload while others are sleeping or have a light load. Due to the unfair distribution of load in the cloud, the cloud DC may be influenced by overloading and underloading of VMs. When servers are equally loaded, the performance will improve. The proposed algorithm aims to avoid these unfair distributions of the load through the VMs by distributing the load among the VMs in an appropriate manner. Therefore, a new efficient scheduling algorithm is suggested and then implemented in cloud computing using cloud analyst, in Java language. The proposed algorithm is a hybrid approach that combines GA and throttled strategies, and it is called Dynamic Throttled Genetic (DTG).

The efficient allocation of resources and scheduling is a vital task in cloud computing based on which the performance of the system is rated. In general, algorithms of load balancing are multi-objective to guarantee the maximum use of all resources along with the enhancement in response time, throughput, makespan and, cost. Optimization is picking out the best solution. Any optimization case is either maximization or minimization, which depends on the nature of the problem [39].

In existing throttled, when a request comes from a client to the Data Center Controller (DCC) and wants to be allocated to a VM, then the accessibility of VM is examined starting from the first VM. However, it is better that the accessibility checking starts from the next to be assigned VM for load balancing. Moreover, existing throttled does not take into account the processing times for each individual requests [12, 13, 28].To overcome this problem, a GA is incorporated with a Throttled to improve load balancing.

GA can have a key role in load balancing. It can handle the scheduling mechanism in which the requests are allocated to resources. This determines which resource will be appropriate for which task. Moreover, GA can be used to decrease the scheduling time. The concept of GA is that the new generation of a solution should be better compared to the previous one. A solution is represented by a chromosome [40].

GA is a nature-inspired algorithm which relies on the ‘existence of the fittest’ idea. GA is used as an optimization process in diverse applications due to its ability to locate the global maximum in several different modes. It converges progressively in respect of a global optimum solution according to the target function, which is one of the most significant countenances of any optimization algorithm. According to its many features such as robustness and adaptability, it has acquired popularity to resolve the optimization issues, particularly in mysterious environments. GA can be used to find the most suitable processors to carry out the specified set of tasks for improving response time, resource utilization, etc. Generally, GA consists of a four-steps, which are selection, crossover, mutation, and termination [23].

The basic process for GA is as follows [41]:

I. Initialization: Creates an initial population. This population is usually randomly generated and can be any desired size.
II. Evaluation: Calculates 'fitness' for each chromosome. The fitness value is calculated by how well it fits with the desired requirements.
III. Selection: Selection works by throwing the weak and preserving the best chromosome in the population (good solutions are chosen).
IV. Crossover: Creates new offspring, which will inherit the best feature from its parents (crossover the parent’s to build new offspring).
V. Mutation: Makes very little alterations at random to chromosomes. Every combination of solutions one can inspire would be in the initial population (changing the gene estimate in the chromosome).
VI. Termination: The GA can be restarted until it reaches a stop condition.

Some concepts used with GA are as follows[36]:

I. Fitness Function: It is a type of an objective function, which is used to represent how close the solution is getting to the set target. The fitness function of the chromosome in DTG is the min cost.
II. Population: It is a set of possible solutions for the proposed problem (in DTG a collection of chromosomes represented in all available VMs).
III. Chromosome: It represents the individuals in the population (one solution which consists of genes).
IV. Gene: It represents a variable in a chromosome.

B. Architecture of the Dynamic Throttled Genetic (DTG) Methodology

The proposed DTG algorithm concentrates mostly on how incoming cloudlet requests are assigned to the appropriate VMs intelligently. The purpose of this proposed load balancer algorithm is to design efficient scheduling that uniformly divides the workload among the available VMs, decreases the overall response time and DC processing time to improve the resource utilization. This can be considered as an optimization problem.

An index schedule is used to keep the VM-id and its condition either AVAILABLE or BUSY. At the start, all VMs are available. At any moment, when a new request comes from the client to the DCC, the DCC forwards the request to the proposed load balancer and asks for request assigning. The proposed DTG load balancer checks all the available VM in the index schedule. Throttled returns all the available VMs-id to the GA. GA considers all the available VMs as an initial population and begins to evaluate every chromosome fitness cost.

Decimal numbers are used for genes representation. The chromosome presentation used in the proposed method is as follows:

A = {1 2 3 4 5 6 7 8}

The digit represents the (VM_ID) and the place at which it is located is Cloudlet ID (C_ID).

The fitness function is the base part of the evolution of the algorithm. The chromosomes having good fitness is selected for generating a child. The fitness function used in the proposed algorithm is based on choosing the chromosome with minimum cost. The less the value of fitness function, the more chromosomes are fit.

The initial population is created randomly so the size of the available resources should be known. If there are two available resources and a new request coming, there are two solutions. In that case, the proposed DTG will compute the expected cost of the two available resources. It will take the resource that has a minimum cost, where the min cost on VM_id1 and min cost on VM_id2 are arranged in list index. The number of chromosomes is a population size (the number of chromosomes is set =512). The fitness function will be computed on each chromosome. This is considered the evaluation for all chromosomes. Following that, the good population is selected for the generation of new children, and the crossover operations are done on the selected chromosomes. The next step mutates the new children with specific mutation probability, and then updates the population with the new children to form the new population and remove the bad ones. The last step ends by returning the best chromosomes of the final population after a number of iterations that are specified in the setting (128 here).

Initial population (The set of the available resources (VMs), and the population is randomly generated. Therefore, the representation of solutions for each gene or (chromosome) consists of VM_ID and ID for each task to be executed on these VM). As shown in Fig.1.

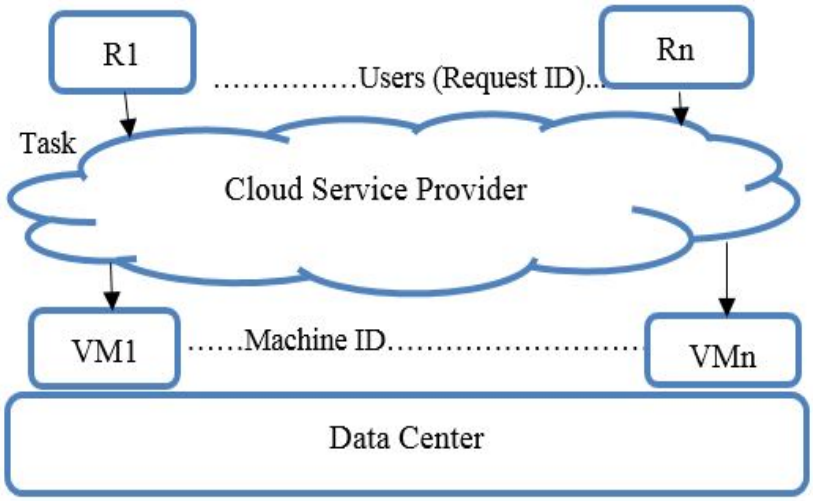


Fig.1. Representation tasks and VMs.

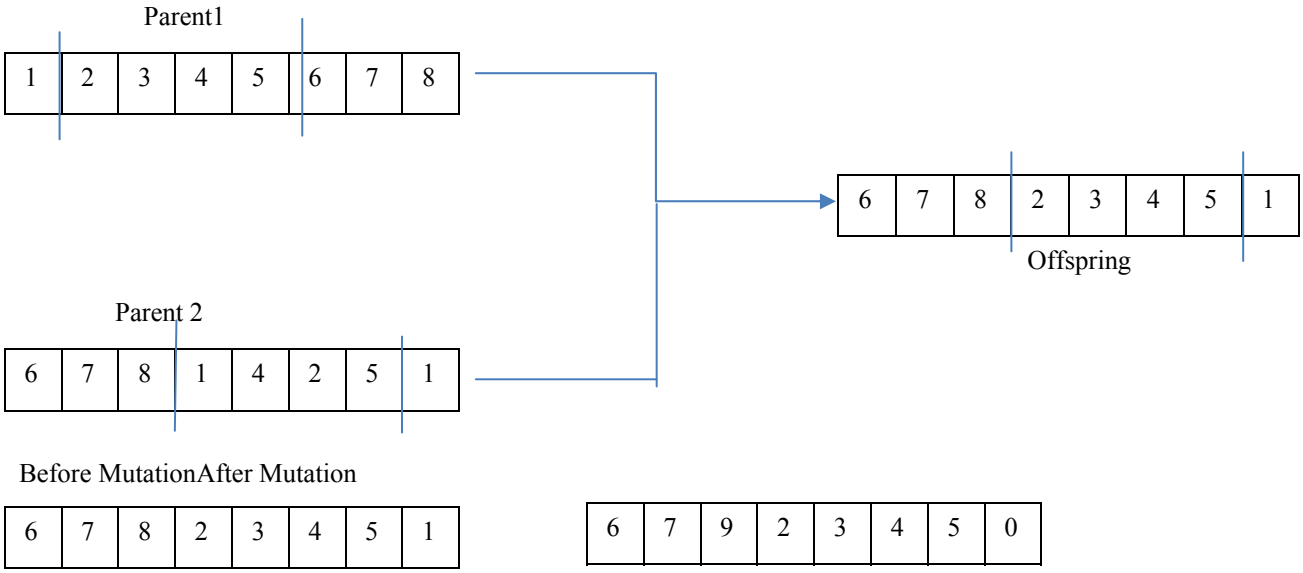
Fitness function evaluation: The chromosomes having good fitness is selected for generating a child. The fitness function used in DTG method is based on choosing the min cost.

Parent's selection: The fitter of the two individuals is selected to be a parent.

Crossover: It can be achieved by selecting two parent individuals and then creating a new individual by alternating and reforming the parts of those parents. Therefore, two chromosomes which are selected for the crossover process to generate two offspring will be considered as offspring also.

Table2. Crossover process

Parent1	VM1 VM2 VM1 ‖ VM3 VM2 VM1
Parent2	VM2 VM1 VM3 ‖ VM1 VM3 VM3
Offspring1	VM1 VM2 VM1 ‖ VM1 VM3 VM3
Offspring2	VM2 VM1 VM3 ‖ VM3 VM2 VM1



Mutation: The mutation is done for an abrupt change in population. In DTG algorithm, swapping of Digits is done.

Original Offspring	VM1 VM 2 VM1 …VM 1 VM 3 VM3
Mutated Offspring	VM1 VM4 VM1 … VM2 VM2 VM3

GA iterates the process until it finds the best available VM for the request according to its min cost, and then it returns VM-id to DCC.

The DCC tells the proposed DTG load balancer to alter the values in the allocation schedule and keep the state BUSY of an allotted machine in VM State schedule. If the proposed DTG algorithm does not find an available VM in the index schedule, then it returns -1. When VMs finish the process and DCC receives the response, it notifies the proposed DTG algorithm to de-allocate the VM. The DTG algorithm then alters the state of the VM as AVAILABLE. If more requests are waiting, then the process of assigning is started again. The basic methodology of the proposed algorithm is shown in Fig. 2 and the detailed work of GA in the proposed DTG as illustrated in Fig. 3.

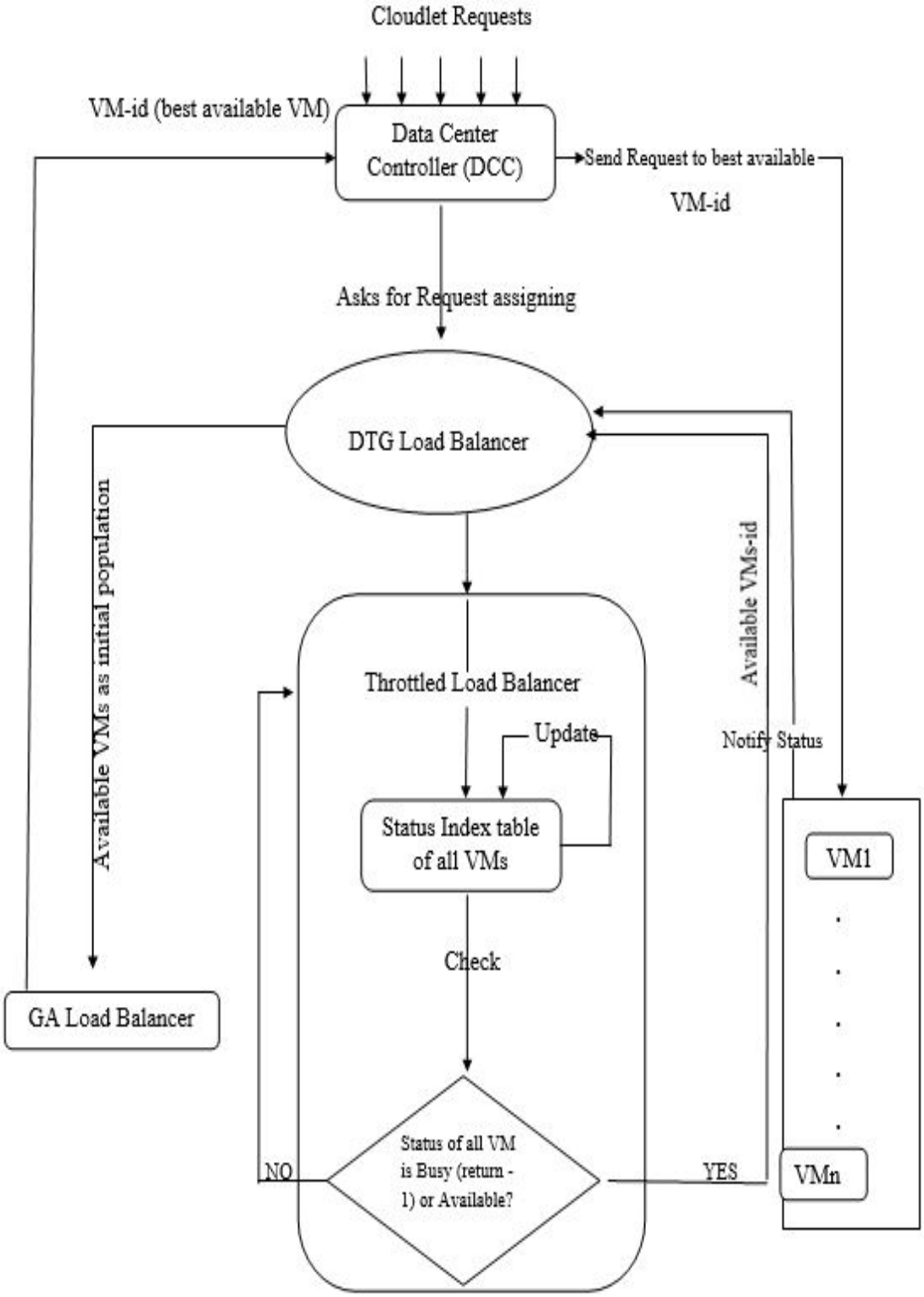


Fig.2.The sequence of Dynamic Throttled Genetic Algorithm (DTG).

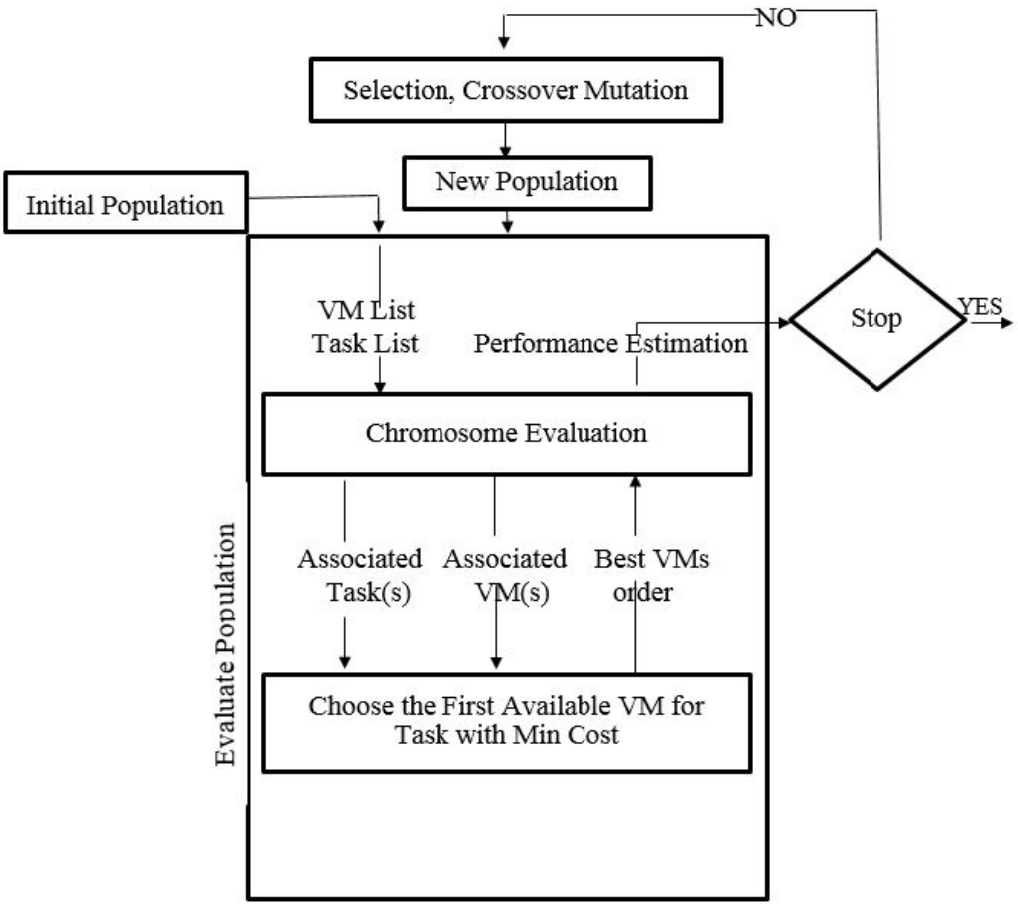


Fig.3. Genetic algorithm operation diagram.

Algorithm 1: Dynamic Throttled Genetic Algorithm (DTG)

Input: DC Cloudlet Requests R1, R2, R3, R4,..., Rn
Available VMs (Population of VMs) VM1, VM2, VM3, VM4… VMn.
Output: Scheduled all incoming DC Cloudlet Requests to appropriate first available VMs based on min cost.
1: Throttled Load Balancer algorithm Create an index schedule of VM-id and status.
All VM's are available at the beginning.
2: DCC receives a new Cloudlet Request from the client.
3: DCC asks the DTG Load Balancer for Cloudlet Request allocation.
4: DTG Load Balancer starts checking for the availability of all the VMs-id.
Case 1: if found

i. Throttled Load Balancer returns all the available VMs-id to the GA Load Balancer.
ii. GA Load Balancer initializes a population (a collection of a chromosome) using all available VMs.
iii. GA Load Balancer evaluates every chromosome which consists of genes (A solution) due to its cost.
iv. Select the best chromosome based on fitness cost.
v. Do Crossover operations on selected chromosome (Perform Crossover).
vi. Mutate the new children with defined Mutation probability (Perform Mutation).
vii. A new population can be produced by repeating the three operations: Selection, Crossover, and Mutation.
viii. GA Load Balancer finished iterating (Check for termination condition) and return the appropriate VM-id to DCC.
ix. The DCC sends the Cloudlet Request to the VM specified by GA Load Balancer.
x. DCC notifies the DTG Load Balancer of the new allotment.

xi. DTG Load Balancer updates the allotment index schedule accordingly

Case 2: if not found

i. DTG Load Balancer returns -1.

5: When the VM finishes processing the Cloudlet Request, and the DCC receives the response Cloudlet, it notifies the DTG Load Balancer of the VM de-allocation.

6: If there are more Cloudlet Requests, DCC repeats step 4.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

A. Cloud Analyst

In this section, the performance of the proposed DTG algorithm is analyzed in a highly effective dynamic environment. It is hard to carry out the test iteratively with reliability, and so cloud sim tool has been selected to test algorithms before displaying them in a real cloud. The tests are done on the cloud analyst. It is an open source toolkit, and it is built on top of cloud sim toolkit as shown in Fig. 4. It also, provides analysis of algorithms performance, and simple graphical user interface that displays the result of simulation in graphical form as shown in Fig. 5. It lets researchers test their suggested algorithms free, and find a solution to a performance deadlock before posting in the real cloud. It can also execute tests with several factors repeatedly. Cloud analyst emulates the real-time with six user bases acting as the six continents of the world. The number of users is set out of a specific application such as twitter users from North America etc. All user bases are assigned to a single time zone. Random samples online through peak hours, and offline through off hours, have been examined. This tool is also elastic[18, 35, 40, 42].

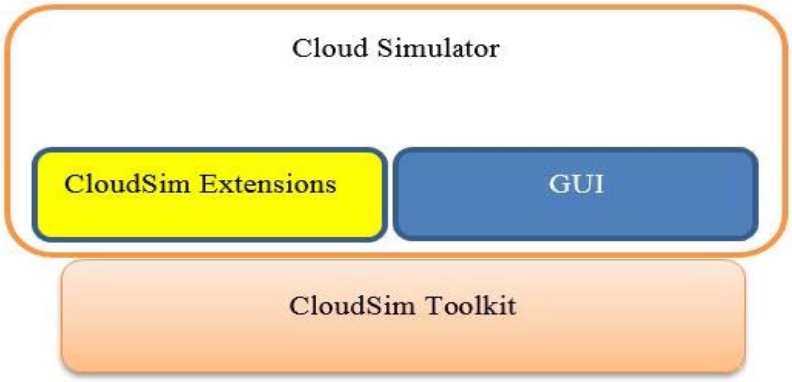


Fig.4. Cloud analyst design built on top of cloud sim toolkit.

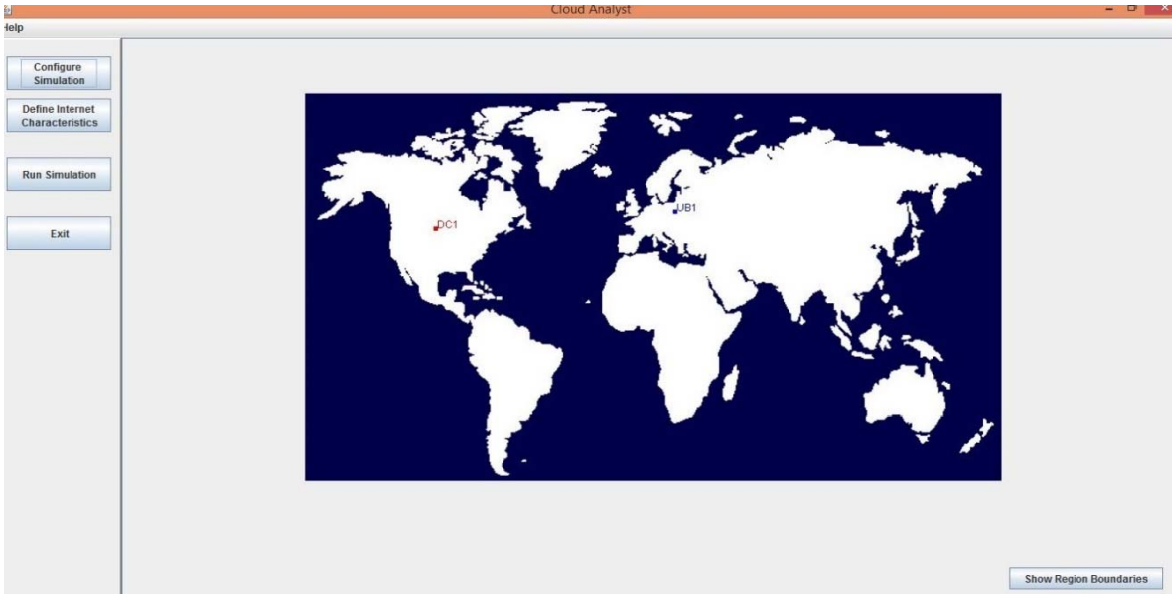


Fig.5. Cloud analyst main screen.

B. EXPERIMENTAL TESTS

In order to test the proposed DTG load balancer algorithm, the optimization process is simulated by NetBeans IDE 8.0.2 using advanced java for coding [43]. In this test, six User Bases (UB) are fixed in six different location/regions of the world. The four DCs involving the following characteristics: a number of resources, cache size, and DC bandwidth. The entire ordering is depicted in Table 3 under the same scenario which consists of four DCs and six UBs in six different geographical points as shown in Fig.6. The four DCs are considered to serve the requests of users. A first DC is located in region 0 which consists of 32 VM, the second one is in region 1 which consists of 17 VM, the third one is in region 2 which consists of 50 VM and the fourth one is in region 3 which consists of 81 VM. The simulation duration is about 60 minutes. The proposed method will be taken on two diverse levels. There are three popular routing protocols that are available in the simulator, which are: "Closest Data Center", "Optimize Response Time" and "Reconfigure Dynamic with Load". In the first level, the closest data center is selected as a broker policy and set the others as default. The second level implements the proposed DTG load balancer, which executes the load balance policy when serving and assigning requests.

Table3. Experimental Parameter

Parameters	**Value Used**
Service Broker Policy	Closest Data Center
VM Image Size	10000
VM Memory	512
VM Bandwidth	1000
No. of VMs	DC1-32/DC2-17/DC3-50/DC4-81
Data Center Architecture	X86
Data Center OS	Linux
Data Center VMM	Xen
Data Center No. of Machines (Physical h/W units)	10
Data Center Memory per Machine	204800
Data Center Storage per Machine	100000000
Data Center Available BW per Machine	1000000
Data Center No. of Processors per Machine	4
Data Center Processors Speed	10000
Data Center VM Policy	TIME_SHARED
User Grouping Factor	10000
Request Grouping Factor	1000
Executable Instruction Length	100000
Load Balancing Policy	Dynamic Throttled Genetic (DTG)

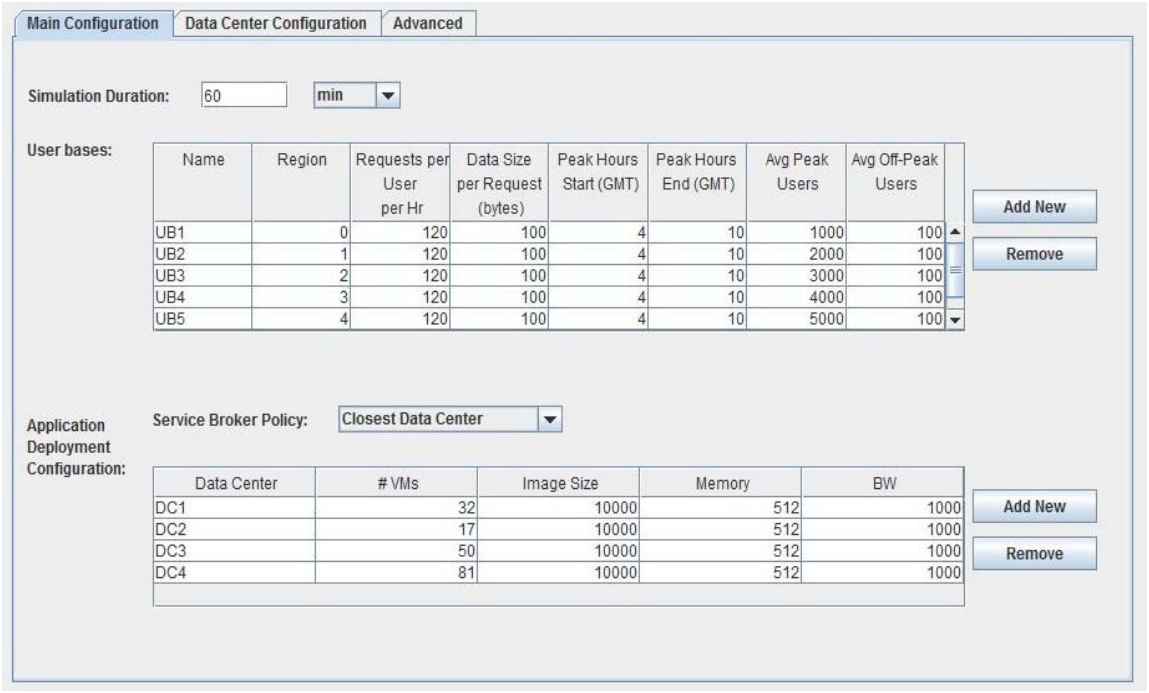


Fig. 6. Cloud Analyst Main Configuration.

C. Results

Each DC host has a particular amount of VMs. In this test, four DCs are considered with 32, 17, 50 and 81 VMs. The simulation is repeated for ESCE, Throttled and the proposed DTG.

The results of the overall response time and DC processing time of ESCE algorithm obtained based on the above considerations. The overall response time is 1384.80 ms. The DC processing time is 1267.42 ms. The response time by region for all six UBs is shown in Fig.7.

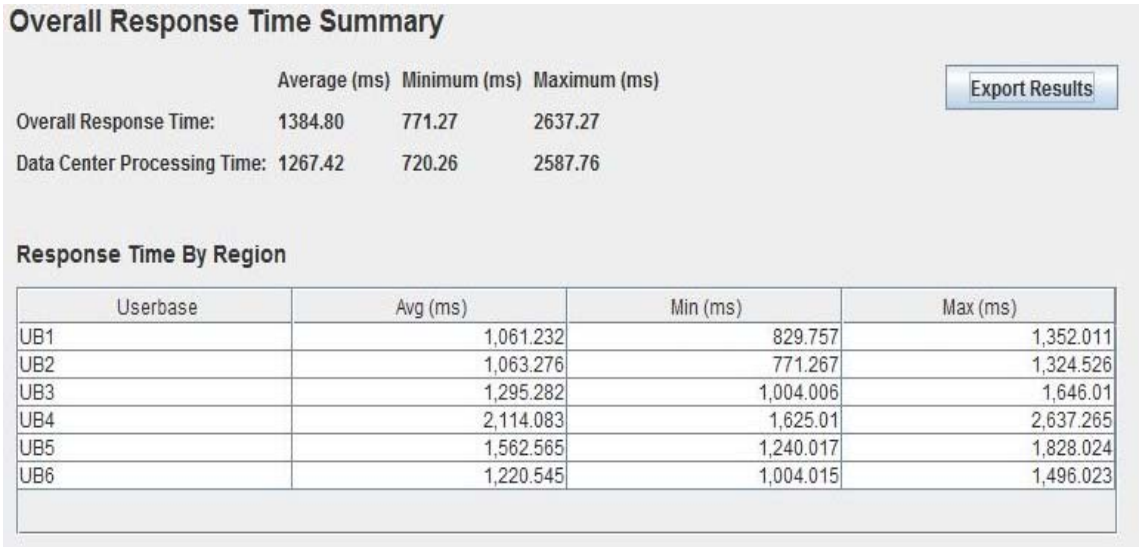


Fig.7. Overall response time and DC processing time of ESCE.

The results for the simulation done based on the specification provided in the above TABLE 3. The geographical location of each DC with their average, minimum and maximum execution time for the ESCE algorithm is shown in Fig.8, where the figure shows the overall response time for the DCs for the VMs to serve the requests if the service broker policy is set to the closest DC.

Fig.8. Data Center locations and user bases requests for ESCE.

The result of overall response time and DC processing time of throttled algorithm are as follows. The overall response time is 1408.74 ms. The DC processing time is 1291.37 ms. The response time by region for all six user bases is shown in Fig.9.

Overall Response Time Summary

Export Results

	Average (ms)	Minimum (ms)	Maximum (ms)
Overall Response Time:	1408.74	771.27	2796.77
Data Center Processing Time:	1291.37	720.26	2745.01

Response Time By Region

Userbase	Avg (ms)	Min (ms)	Max (ms)
UB1	1,061.256	829.757	1,352.011
UB2	1,063.17	771.267	1,324.526
UB3	1,295.258	1,004.006	1,646.01
UB4	2,260.277	1,822.51	2,796.768
UB5	1,562.749	1,240.017	1,828.024
UB6	1,220.282	1,004.015	1,496.023

Fig.9. Overall response time and DC processing time of throttled.

Fig.10 gives the simulation complete screen which shows the results graphically for each DC with their average, the minimum and maximum execution time for a throttled algorithm. It displays the results which indicate the response time by region whether this request is made from a single user or multiple users.

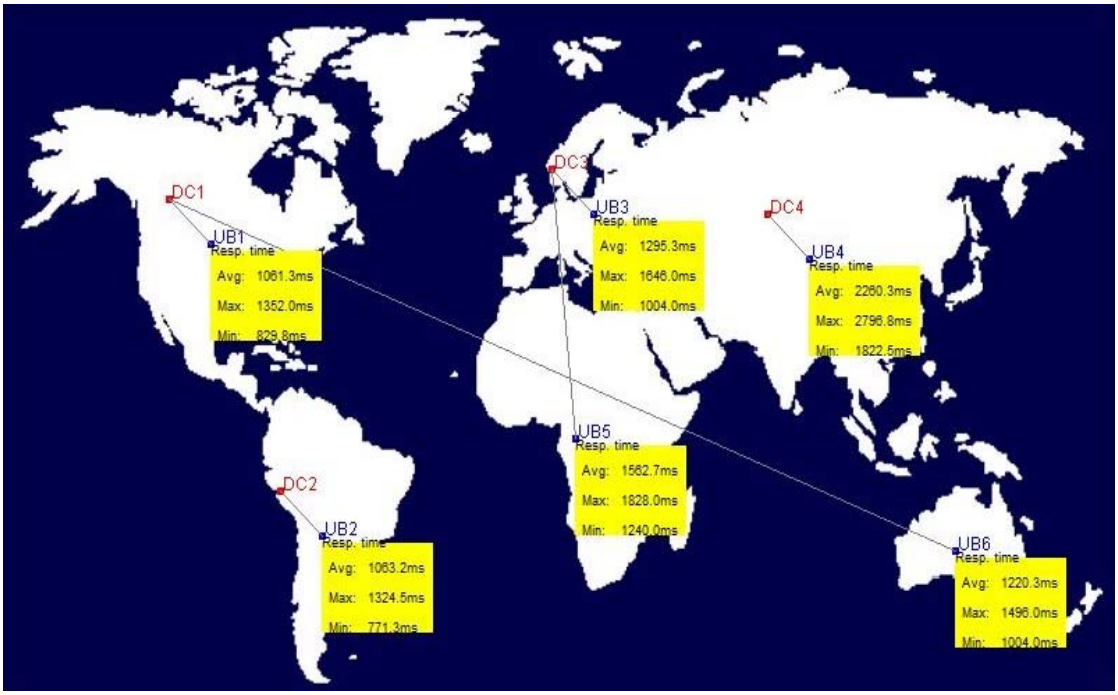


Fig.10. DC locations and user bases requests for Throttled.

The result of overall response time and DC processing time of the proposed DTG algorithm are as follows. The overall response time it is 1372.02 ms. The DC processing time is 1254.49 ms. The response time by region for all six UBs is shown in Fig.11. The performance of the DTG algorithm is enhanced as follows. DTG gives less response time and less processing time with better resource utilization compared with ESCE and throttled algorithms.

Overall Response Time Summary

Export Results

	Average (ms)	Minimum (ms)	Maximum (ms)
Overall Response Time:	1372.02	771.27	2796.77
Data Center Processing Time:	1254.49	720.26	2745.01

Response Time By Region

Userbase	Avg (ms)	Min (ms)	Max (ms)
UB1	1,061.31	829.757	1,352.011
UB2	1,063.131	771.267	1,324.526
UB3	1,295.212	1,004.006	1,646.01
UB4	2,035.097	1,625.01	2,796.768
UB5	1,563.02	1,240.017	1,828.024
UB6	1,221.299	1,004.015	1,496.023

Fig.11. Overall response time and DC processing time of DTG.

Fig.12 shows the geographical location of each DC with their average, the minimum and maximum execution time for DTG algorithm

Fig.12. DC locations and user bases requests for DTG.

D. Comparative Results

To verify the improvement of the proposed DTG algorithm, it is compared with the standard ESCE and throttled as shown in Table 4. The enhancement achieved by the proposed DTG algorithm is verified by comparative analysis. The proposed DTG algorithm tries to minimize the response time as well as reduce the DC processing time of VMs that are going to miss their deadlines. It can be seen that the proposed DTG algorithm guarantees a fair allocation of the requests to each DC. This fair allocation safely gets a better response, unlike traditional policies. With the proposed DTG algorithm, the overall response time and DC processing time for a request has been improved compared to the other two algorithms. The result proved that DTG makes use of VMs more effectively while bringing down the response time of the tasks by 36.72% in comparison with others. Experimental results show that the DTG load balancing algorithm gives less response time and less processing time. The results also suggest that the DTG VM load balancer allocated the requests to VMs evenly by overcoming the limitation of throttled VM load balancer.

Table4. Comparison result between Existing and DTG

Parameters	**Overall Response time**			**Data Center Processing Time**		
	ESCE	Throttled	**DTG**	**ESCE**	**Throttled**	**DTG**
Avg(ms)	1384.8	1408.74	1372.02	1267.42	1291.37	1254.49
Min(ms)	771.27	771.27	771.27	720.26	720.26	720.26
Max(ms)	2637.27	2796.77	2796.77	2587.76	2745.01	2745.01

The performance of the DTG algorithm is evaluated by objective 1: if the response time is low, then the method is said to be more efficient. It is measured in terms of milliseconds (ms). Fig. 13 shows the average response time, which illustrates that the response time of the proposed DTG algorithm is better than the other investigated algorithms. From the results of all the three algorithms, it can be concluded that there is no difference in the cost, but there is a difference between the overall response time of all the UBs. The DTG algorithm is found to be the best with Avg response time of 1372.02 ms. While Avg response time is 1408.74 ms for throttled and 1384.80 ms for ESCE.

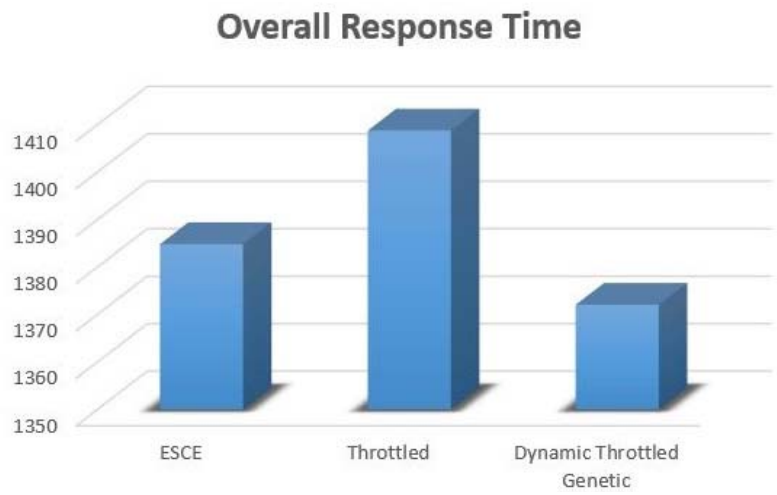


Fig.13. Comparison overall response time between Existing and DTG.

The performance of the DTG algorithm is evaluated by objective 2: if the DC processing time is low, then the method is said to be more efficient. It is measured in terms of ms. Fig. 14 shows the results based on the DC processing time. As can be seen, the DC processing time of the proposed DTG algorithm is better than the others. The results of all the three algorithms indicate that there is no difference in the cost, but there is a difference between the DC processing time of all the UBs. The DTG algorithm is found to be the best with Avg of 1254.49 ms. While Avg is 1291.37 ms for throttled, and 1267.42 ms for ESCE.

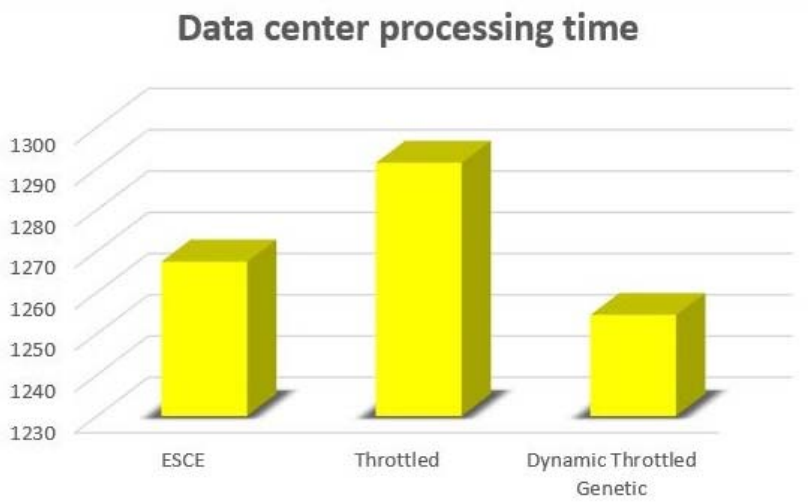


Fig.14. Comparison DC processing time between Existing and DTG.

The DC request servicing time is analyzed by comparing the three loading policy ESCE, Throttled and DTG, and the results are shown in Table 5. Fig. 15 shows the comparison results. The X-axis indicates the DCs and Y-axis indicates the average request servicing time at each DC in ms. A load balancing is considered more efficient when the processing of requests at each DC takes less time. The result shows that there is a slight difference between the four DC request servicing time, and DTG is found to be the best.

Table5. Results indicating the Request Servicing Times at each Data Center.

Data	ESCE	Throttled	DTG
DC1	1015.67	1015.67	1015.68
DC2	1013.41	1013.42	1013.41
DC3	1253.95	1253.95	1253.95
DC4	2064.54	2210.6	1985.67

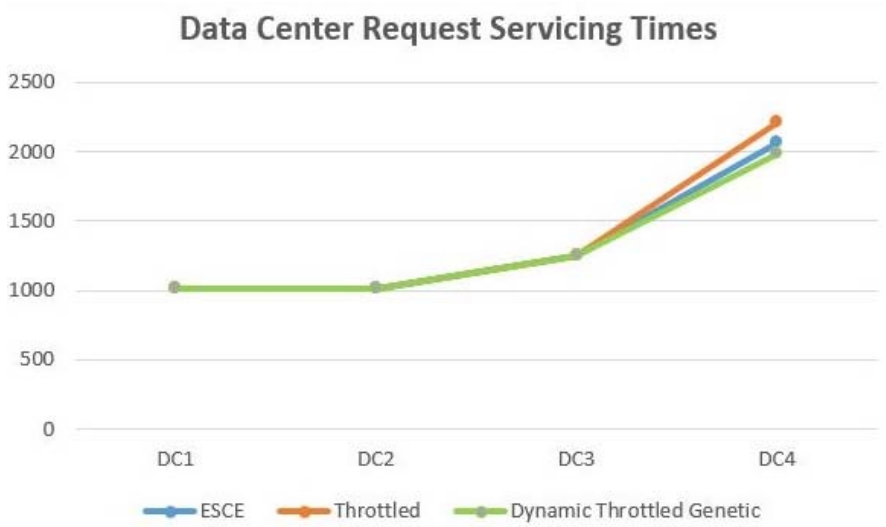


Fig.15. DC request servicing times between DTG, Throttled and ESCE.

The response time of all the UB (UB1-UB6) in various regions is analyzed, and the result is shown in Table 6 and Fig. 16. As can be seen, there is a slight difference between the response time of all the UB, and DTG is found to be the best.

Table6. Average response time by Region where User bases are created

User Base	ESCE	Throttled	DTG
UB1	1061.23	1061.26	1061.31
UB2	1063.28	1063.17	1063.13
UB3	1295.28	1295.26	1295.21
UB4	2114.08	2260.28	2035.1
UB5	1562.56	1562.75	1563.02
UB6	1220.54	1220.28	1221.3

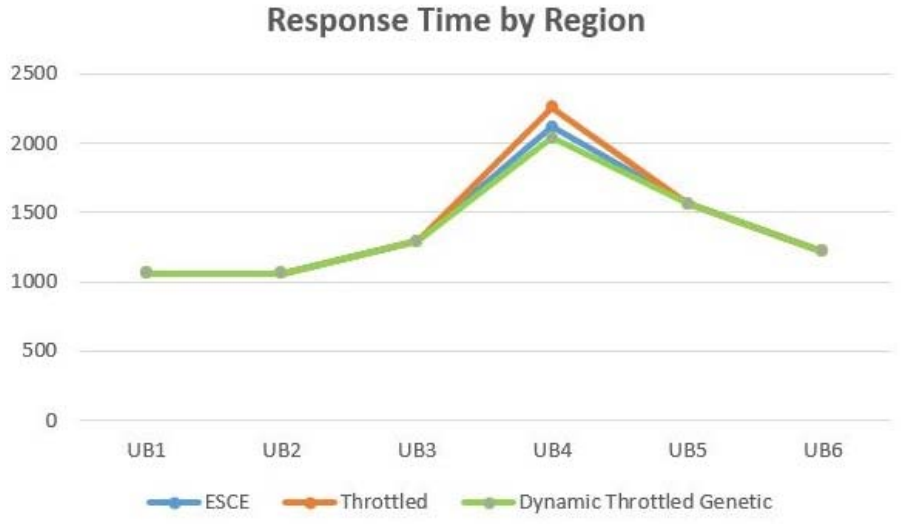


Fig.16. Response time by region chart among ESCE, Throttled and DTG.

Cloud analyst is used to regenerate the results of throttled, ESCE and RR algorithms, described in previous papers, according to the settings used in each paper. After that, changes are made to the settings to evaluate the performance of the previous algorithms and the proposed DTG algorithm. This is done to test the algorithms on different scenarios and to prove that the proposed DTG algorithm is the best, as shown in Fig. 17. Each scenario represents a different configuration. These scenarios are shown in Table 7. It is clear that the proposed DTG algorithm is more efficient for the cloud load balancing. The proposed algorithm performs better than the rest of the algorithms on the basis of response time.

Table7. Simulation Scenarios

Reference	Alteration	Algorithm	Response Time (ms)			Data Center Processing Time (ms)		
			Avg (ms)	Min (ms)	Max (ms)	Avg (ms)	Min (ms)	Max (ms)
[44](2018)		**Throttled**	640.1	47.71	2190.4	576.26	1.48	2087.26
		ESCE	1209.79	47.71	2215.77	1141.39	1.48	2125.94
		RR	1209.8	47.71	2215.77	1141.39	1.48	2125.94
Scenario1	Change Peak Users and Executable Instruction Length	**(DTG)**	389.96	48.51	684.03	340.2	6.26	625.77
		Throttled	389.96	48.51	684.03	340.2	6.26	625.77
		ESCE	389.96	48.51	684.03	340.2	6.26	625.77
		RR	389.96	48.51	684.03	340.2	6.26	625.77
[45](2016)		**Throttled**	57.09	34.45	87.01	6.64	0	24.34
		ESCE	63.19	34.45	9818.03	12.84	0	9768.77
		RR	62.85	34.45	9818.03	12.5	0	9768.77
Scenario2	Change Memory, Duration 5 min and Decrease NO.DC	**(DTG)**	157.64	39.85	352.94	0.61	0.05	1.39
		Throttled	157.66	39.85	352.94	0.61	0.05	1.39
		ESCE	157.68	39.85	352.94	0.61	0.05	1.39
		RR	157.65	39.85	352.94	0.61	0.05	1.39
[46](2015)		**Throttled**	177.66	39.21	388.64	0.37	0.01	0.96
		ESCE	177.66	39.21	388.64	0.37	0.01	0.96
		RR	177.64	39.18	388.64	0.36	0.01	0.97
Scenario3	Change Services policy to Closest Data Center and Decrease UB	**(DTG)**	170.51	38.71	393.14	0.38	0.02	0.96
		Throttled	170.52	38.71	393.14	0.38	0.02	0.96
		ESCE	170.54	38.71	393.14	0.38	0.02	0.96
		RR	170.52	38.71	393.14	0.38	0.02	0.96
[47](2014)		**Throttled**	150.1	39.95	386.33	10.13	0.31	32.08
		ESCE	150.1	39.95	386.33	10.13	0.31	32.08
		RR	150.07	39.76	386.33	10.1	0.31	32.08
Scenario4	Change Duration 10 min and Service policy to Closest Data Center	**(DTG)**	124.46	47.66	403.83	8.48	2.83	14.38
		Throttled	125.35	47.93	403.83	8.52	2.83	14.38
		ESCE	125.09	47.66	403.83	8.49	2.83	14.38
		RR	124.93	47.93	403.83	8.51	2.83	14.38
[48](2014)		**Throttled**	117.89	36.76	401.88	1.33	0.07	2.17
		ESCE	117.89	36.76	401.88	1.33	0.07	2.17
		RR	117.89	36.76	401.88	1.33	0.07	2.56
Scenario5	Change Duration 10 min	**(DTG)**	118.21	38.13	365.77	1.28	0.13	2.01
		Throttled	118.27	38.13	365.77	1.28	0.13	2.01
		ESCE	118.31	38.13	365.77	1.28	0.13	2.01
		RR	118.28	38.13	365.77	1.28	0.13	2.01

Experimental results show that the DTG load balancing algorithm gives less response time according to the different five scenarios in different five papers. The results of all the four algorithms indicate that there is no difference between the DC processing time, but there is a slight difference between the overall response time of all the UBs. The DTG provides the best Avg response time. The results prove that combining GA and Throttled improves the response time of VMs as compared to ESCE, throttled, and RR. The results also suggest that the DTG VM load balancer allocated the requests to VMs evenly, and thus it overcome the limitation of throttled VM load balancer.

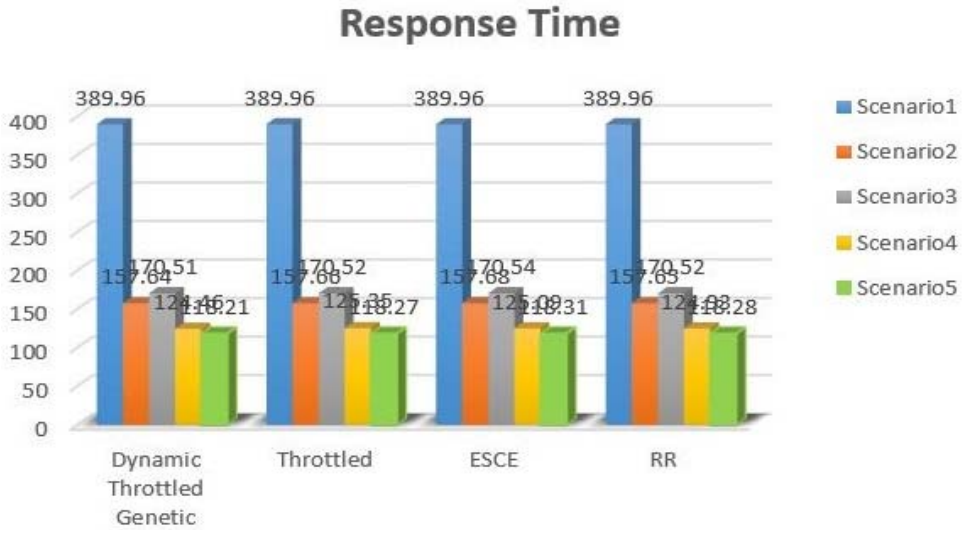


Fig.17. Avg response time in different scenarios among Throttled, ESCE, RR, and DTG.

V. CONCLUSION

This paper proposes a dynamic load balancing algorithm that employs a Genetic Algorithm. The proposed algorithm is called DTG. A load balancing algorithm aims at solving the problem of the cloud data centers being affected by some servers having to serve a heavy load, while other servers are asleep or have a little amount of load. When servers are equally loaded, the performance will improve. This is done by reassigning from a heavily loaded server to a lightly loaded server. One important aspect of cloud computing is the minimization of response time so as to balance the workload and raise business rendering with client satisfaction. The proposed DTG aims at avoiding unfair distribution of the load through the virtual machines.

The performance of the proposed DTG algorithm is investigated with the help of a graphical user interface based Cloud Analyst tool. Java language is used to develop the class file for implementation in the tool. From the results of the simulation, it can be concluded that the proposed DTG algorithm works efficiently when it comes to resource utilization, the processing time of the data center, and response time of the user base. The simulation results show that the overall response time and data center processing time of DTG is improved, and the cost is reduced as compared with the Throttled and ESCE algorithms.

REFERENCES

[1] S. Basu, A. Bardhan, K. Gupta, P. Saha, M. Pal, M. Bose, et al., "Cloud computing security challenges & solutions-A survey," in Computing and Communication Workshop and Conference (CCWC), 2018 IEEE 8th Annual, 2018, pp. 347-356.
[2] F. F. Moghaddam, M. Ahmadi, S. Sarvari, M. Eslami, and A. Golkar, "Cloud computing challenges and opportunities: A survey," in Telematics and Future Generation Networks (TAFGEN), 2015 1st International Conference on, 2015, pp. 34-38.
[3] M. H. Ghahramani, M. Zhou, and C. T. Hon, "Toward cloud computing QoS architecture: Analysis of cloud systems and cloud services," IEEE/CAA Journal of Automatica Sinica, vol. 4, pp. 6-18, 2017.
[4] J. Moura and D. Hutchison, "Review and analysis of networking challenges in cloud computing," Journal of Network and Computer Applications Journal of Network and Computer Applications, vol. 60, pp. 113-129, 2016.
[5] V. R. Kanakala, V. K. Reddy, and K. Karthik, "Performance analysis of load balancing techniques in cloud computing environment," in Electrical, Computer and Communication Technologies (ICECCT), 2015 IEEE International Conference on, 2015, pp. 1-6.
[6] E. Jafarnejad Ghomi, A. Masoud Rahmani, and N. Nasih Qader, "Load-balancing algorithms in cloud computing: A survey," YJNCA Journal of Network and Computer Applications, vol. 88, pp. 50-71, 2017.
[7] N. Patil and D. Aeloor, "A review-different scheduling algorithms in cloud computing environment," in Intelligent Systems and Control (ISCO), 2017 11th International Conference on, 2017, pp. 182-185.
[8] N. Panwar and M. S. Rauthan, "Analysis of various task scheduling algorithms in cloud environment," in Cloud Computing, Data Science & Engineering-Confluence, 2017 7th International Conference on, 2017, pp. 255-261.
[9] E. S. Alkayal, N. R. Jennings, and M. F. Abulkhair, "Survey of task scheduling in cloud computing based on particle swarm optimization," in Electrical and Computing Technologies and Applications (ICECTA), 2017 International Conference on, 2017, pp. 1-6.
[10] J. M. Shah, K. Kotecha, S. Pandya, D. Choksi, and N. Joshi, "Load balancing in cloud computing: Methodological survey on different types of algorithm," in Trends in Electronics and Informatics (ICEI), 2017 International Conference on, 2017, pp. 100-107.
[11] A. Dave, B. Patel, and G. Bhatt, "Load balancing in cloud computing using optimization techniques: A study," in Communication and

Electronics Systems (ICCES), International Conference on, 2016, pp. 1-6.
[12] S. Aslam and M. A. Shah, "Load balancing algorithms in cloud computing: A survey of modern techniques," in Software Engineering Conference (NSEC), 2015 National, 2015, pp. 30-35.
[13] S. Patel, R. Patel, H. Patel, and S. Vahora, "CloudAnalyst: A Survey of Load Balancing Policies," International Journal of Computer Applications, vol. 117, 2015.
[14] I. N. Ivanisenko and T. A. Radivilova, "Survey of major load balancing algorithms in distributed system," in Information Technologies in Innovation Business Conference (ITIB), 2015, 2015, pp. 89-92.
[15] M. A. Alworafi, A. Dhari, A. A. Al-Hashmi, and A. B. Darem, "An improved SJF scheduling algorithm in cloud computing environment," in Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT), 016 International Conference on, 2016, pp. 208-212.
[16] A. A. Alexander and D. L. Joseph, "An Efficient Resource Management for Prioritized Users in Cloud Environment Using Cuckoo Search Algorithm," Procedia Technology, vol. 25, pp. 341-348, 2016.
[17] A. Gupta and R. Garg, "Load Balancing Based Task Scheduling with ACO in Cloud Computing," in Computer and Applications (ICCA), 2017 International Conference on, 2017, pp. 174-179.
[18] S. Garg, D. Gupta, and R. K. Dwivedi, "Enhanced Active Monitoring Load Balancing algorithm for Virtual Machines in cloud computing," in System Modeling & Advancement in Research Trends (SMART), International Conference, 2016, pp. 339-344.
[19] A. K. Kulkarni and B. Annappa, "Load balancing strategy for optimal peak hour performance in cloud datacenters," in Signal Processing, Informatics, Communication and Energy Systems (SPICES), 2015 IEEE International Conference on, 2015, pp. 1-5.
[20] M. S. Shakir and A. Razzaque, "Performance comparison of load balancing algorithms using cloud analyst in cloud computing," in Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), 2017 IEEE 8th Annual, 2017, pp. 509-513.
[21] L. S. Nishad, S. Kumar, S. K. Bola, S. Beniwal, and A. Pareek, "Round robin selection of datacenter simulation technique cloudsim and cloud analsyt architecture and making it efficient by using load balancing technique," in Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on, 2016, pp. 2901-2905.
[22] N. K. Das, M. S. George, and P. Jaya, "Incorporating weighted round robin in honeybee algorithm for enhanced load balancing in cloud environment," in Communication and Signal Processing (ICCSP), 2017 International Conference on, 2017, pp. 0384-0389.
[23] H. A. Makasarwala and P. Hazari, "Using genetic algorithm for load balancing in cloud computing," in Electronics, Computers and Artificial Intelligence (ECAI), 2016 8th International Conference on, 2016, pp. 1-6.
[24] G. Rjoub and J. Bentahar, "Cloud Task Scheduling Based on Swarm Intelligence and Machine Learning," in Future Internet of Things and Cloud (FiCloud), 2017 IEEE 5th International Conference on, 2017, pp. 272-279.
[25] F. Sadia, N. Jahan, L. Rawshan, M. T. Jeba, and T. Bhuiyan, "A priority based dynamic resource mapping algorithm for load balancing in cloud," in Advances in Electrical Engineering (ICAEE), 2017 4th International Conference on, 2017, pp. 176-180.
[26] M. Aruna, D. Bhanu, and S. Karthik, "An improved load balanced metaheuristic scheduling in cloud," Cluster Comput. Cluster Computing, pp. 1-9, 2017.
[27] R. K. Jena, "Multi Objective Task Scheduling in Cloud Environment Using Nested PSO Framework," Procedia Computer Science Procedia Computer Science, vol. 57, pp. 1219-1227, 2015.
[28] S. Garg, R. K. Dwivedi, and H. Chauhan, "Efficient utilization of virtual machines in cloud computing using Synchronized Throttled Load Balancing," in Next Generation Computing Technologies (NGCT), 2015 1st International Conference on, 2015, pp. 77-80.
[29] S. G. Domanal and G. R. M. Reddy, "Load balancing in cloud environment using a novel hybrid scheduling algorithm," in Cloud Computing in Emerging Markets (CCEM), 2015 IEEE International Conference on, 2015, pp. 37-42.
[30] G. P. P. Geethu, S. K. Vasudevan, D. nd International Symposium on Big, and I. Cloud Computing Challenges, "An in-depth analysis and study of Load balancing techniques in the cloud computing environment," Procedia Comput. Sci. Procedia Computer Science, vol. 50, pp. 427-432, 2015.
[31] K. R. Babu, A. A. Joy, and P. Samuel, "Load Balancing of Tasks in Cloud Computing Environment Based on Bee Colony Algorithm," in Advances in Computing and Communications (ICACC), 2015 Fifth International Conference on, 2015, pp. 89-93.
[32] M. S. George, K. N. Das, and B. Pushpa, "Enhanced honeybee inspired load balancing algorithm for cloud environment," in Communication and Signal Processing (ICCSP), 2017 International Conference on, 2017, pp. 1649-1653.
[33] M. R. Mesbahi, M. Hashemi, and A. M. Rahmani, "Performance evaluation and analysis of load balancing algorithms in cloud computing environments," in Web Research (ICWR), 2016 Second International Conference on, 2016, pp. 145-151.
[34] A. Kumar and M. Kalra, "Load balancing in cloud data center using modified active monitoring load balancer," in Advances in Computing, Communication, & Automation (ICACCA)(Spring), International Conference on, 2016, pp. 1-5.
[35] A. Ragmani, A. El Omri, N. Abghour, K. Moussaid, and M. Rida, "A performed load balancing algorithm for public Cloud computing using ant colony optimization," in Cloud Computing Technologies and Applications (CloudTech), 2016 2nd International Conference on, 2016, pp. 221-228.
[36] S. Dam, G. Mandal, K. Dasgupta, and P. Dutta, "Genetic algorithm and gravitational emulation based hybrid load balancing strategy in cloud computing," in Computer, Communication, Control and Information Technology (C3IT), 2015 Third International Conference on, 2015, pp. 1-7.
[37] P. A. Pattanaik, S. Roy, and P. K. Pattnaik, "Performance study of some dynamic load balancing algorithms in cloud computing environment," in Signal processing and integrated networks (SPIN), 2015 2nd International Conference on, 2015, pp. 619-624.
[38] R. Panwar and B. Mallick, "Load balancing in cloud computing using dynamic load management algorithm," in Green Computing and Internet of Things (ICGCIoT), 2015 International Conference on, 2015, pp. 773-778.
[39] J. Nayak, B. Naik, A. Jena, R. K. Barik, and H. Das, "Nature Inspired Optimizations in Cloud Computing: Applications and Challenges," in Cloud Computing for Optimization: Foundations, Applications, and Challenges, ed: Springer, 2018, pp. 1-26.
[40] M. Gupta and A. Jain, "A survey on cost aware task allocation algorithm for cloud environment," in Signal Processing, Computing and Control (ISPCC), 2017 4th International Conference on, 2017, pp. 642-646.
[41] M. Lagwal and N. Bhardwaj, "Load balancing in cloud computing using genetic algorithm," in Intelligent Computing and Control Systems (ICICCS), 2017 International Conference on, 2017, pp. 560-565.
[42] A. N. Singh and S. Prakash, "WAMLB: Weighted Active Monitoring Load Balancing in Cloud Computing," in Big Data Analytics, ed: Springer, 2018, pp. 677-685.
[43] R. Sachdeva and S. Kakkar, "A Novel Approach in Cloud Computing for Load Balancing Using Composite Algorithms," IJARCSSE International Journal of Advanced Research in Computer Science and Software Engineering, vol. 7, pp. 51-56, 2017.
[44] A. Meftah, A. E. Youssef, and M. Zakariah, "Effect of Service Broker Policies and Load Balancing Algorithms on the Performance of Large Scale Internet Applications in Cloud Datacenters," INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS, vol. 9, pp. 219-227, 2018.
[45] S. P. Singh, A. Sharma, and R. Kumar, "Analysis of Load Balancing Algorithms using Cloud Analyst," International Journal of Grid and Distributed Computing, vol. 9, pp. 11-24, 2016.

[46] D. Patel and A. S. Rajawat, "Efficient throttled load balancing algorithm in cloud environment," International Journal of Mordern Trends in Engineering and Research, vol. 2, 2015.
[47] V. Behal and A. Kumar, "Cloud computing: Performance analysis of load balancing algorithms in cloud heterogeneous environment," in Confluence The Next Generation Information Technology Summit (Confluence), 2014 5th International Conference-, 2014, pp. 200-205.
[48] S. Lamba and D. Kumar, "A Comparative Study on Load Balancing Algorithms with Different Service Broker Policies in Cloud Computing."