Optimizing the load balancing efficiency using enhanced genetic algorithm in cloud computing

MSc Research Project
Msc Cloud Computing

Rohit Rajesh Salvi
Student ID: 21127336

School of Computing
National College of Ireland

Supervisor: Rashid Mijumbi

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Rohit Rajesh Salvi…………………………………………
Student ID:	X21127336……………………………………………………
Programme:	Msc Cloud Computing………… **Year:** Jan 2022
Module:	Msc Research Project………………………………………
Supervisor:	Rashid Mijumbi……………………………………………
Submission Due Date:	15/12/2022………………………………………………
Project Title:	Optimizing the load balancing efficiency using enhanced genetic algorithm in cloud computing
Word Count:	5562…………………… **Page Count:** 17…………………

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Rohit Rajesh Salvi…………………………………………
Date:	14/12/22……………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Optimizing the load balancing efficiency using enhanced genetic algorithm in cloud computing

Rohit Salvi
21127336

Abstract

Cloud computing (CC) technology has received a great deal of interest in recent years from both academics and businesses. To boost the scalability and flexibility of cloud Data Centers (DC), load-balancing solutions are essential. One of the most important concerns in a distributed computing system is the Load Balancing (LB) technique. Because a cloud provider must service several customers in a cloud setting, task scheduling with efficient LB is a key issue in CC. Many strategies, algorithms, and methodologies have been developed over the years to enhance the LB approach in CC. These strategies are primarily concerned with minimizing execution time, cutting energy consumption and overall resource utilization, and rapid task scheduling by swiftly distributing the tasks in a cluster of Virtual Machines (VM). Because no consideration is given to the present load of the VM, the VM in the cluster may begin to experience overloading concerns. As a result, there is a need for a technique that considers not only the load of the VM but also resource, time, and energy metrics. To apply flexible and effective LB in a cloud system, this research suggests an Enhanced Genetic Algorithm (EGA) based on the Genetic Algorithm (GA). The goal of this method is to analyze the load of the VMs and allocate jobs to VMs that will not get overloaded. Based on performance parameters such as resource usage, energy, and time consumption, the algorithm's results will be compared to Particle Swarm Optimization (PSO), a popular LB technique. The results demonstrate that, as the number of cloudlets increases, EGA's execution time, resource utilization, and energy consumption are much lower than those of PSO.

1 Introduction

CC as a distributed computing paradigm, delivers scalable and virtualized computing infrastructure to its users over the internet to provide computing services as a utility. With recent technological breakthroughs, CC has grown in recent years and will continue to be a comprehensive computing service in the future. As the use of CC expands, so do the issues, with new ones emerging daily. LB is one such common challenge that occurs in CC. The goal of LB is to distribute the local workload evenly and dynamically among all the nodes that exist in the existing cluster of VMs. In a CC environment, intake of random tasks with an unpredictable CPU service time needs can overload a certain VM while the rest of the VMs are occupying less load or kept idle. As a result, the primary usage of LB techniques is to balance the current and incoming load on the existing set of VMs or deploy new VM and ensure that the system is constantly stable. Most LB techniques exploits the use of various optimization algorithms to balance the load between the VMs. However, these optimization strategies are ideally suited for task scheduling for recently deployed tasks. As a result, these optimization strategies are restricted since they are largely concerned with distributing the most recent task and do not consider the overall balance of the existing cluster of VMs. Simultaneously, the efficiency of Load Balance Techniques is diminished, causing consumers to wait for an unusually lengthy period. This necessitates the use of a Load Balancing mechanism that considers the existing load on a cluster of VMs.

1.1 Background

Various meta-heuristic optimization algorithms were developed by researchers such as Hybrid Firefly-Genetic Algorithm, which was developed by Rajagopalan A et al. (2019) to accelerate the convergence of solutions to near-optimal levels and to schedule activities with the goal of minimizing execution time across the board. To evaluate the performance, execution time was used as the evaluation metric. Although the suggested method accomplished its primary goal of shortening execution time, it employs a huge solution space in the algorithm, which makes it not load efficient. Alameen A et al. (2020) proposed Fitness Rate-Based Rider Optimization Algorithm (FR-ROA), a meta-heuristic algorithm based on Rider Optimization Algorithm (ROA). To evaluate the performance of this method, the completion time as well as the aggregate of all task completion times were employed. The findings of this method revealed that it worked well under smaller jobs and for shorter periods of time, but it was sophisticated and encountered a complexity issue. Abualigah L et al. (2020) introduced a unique hybrid antlion optimization technique based on elite-based differential evolution to address multi-objective job scheduling challenges in cloud systems. Execution time, response time, and imbalance degree were utilized to evaluate performance. The outcomes of this method indicated that it had a faster execution time than other algorithms, but it used a significant amount of memory and did not take the overload factor into account. These studies showed disadvantages in terms of high cost, poor execution time, and no consideration of existing load on the VMs and overloading issues. Disadvantages in terms of high cost, short execution time, and no consideration of existing load on VMs and overloading concerns were observed in these studies.

1.2 Motivation

To maximize resource productivity and equally distribute tasks to VMs to keep a constant rate of load among the cluster of VMs, an efficient load balancing technique is needed. The efficiency of a cloud system is affected when certain VMs in a cluster are overloading due high number of tasks allocated to them. As the time taken to execute these tasks will be increased and hence the overall cloud system will be constrained. Simultaneously, when the cloud system is overloaded with high volume of tasks, the cloud system must provision the incoming tasks and must divide its tasks equally among the existing VMs in a cluster or deploy new VM if required. The performance of the overall cloud system will enhance and increase efficiently if the cloud system can allocate the incoming task among the VMs while also considering the existing load on them. As a result, the aim of the research project is to provide an effective and efficient load balancing algorithm which is based on Genetic Algorithm (GA) to avoid scheduling tasks to VMs that are already under massive load due to high volume of tasks and subsequently overloading them. The goal of the proposed algorithm is to find the best possible VM among a cluster of VM. The incoming task that needs to be allocated to a VM, will be allocated to the best possible VM which is found out from the Enhanced Genetic Algorithm (EGA) which will not overload the VM. The cloud system's effectiveness will be affected in proportion to the prolonged execution time of EGA will be. As a result, the overall cost and the number of migrants is lowered in order to shorten the time necessary to execute the algorithm. This will be achieved by modifying the GA to improve the reliability and efficiency of the load balancing process.

1.3 Problem Statement

The major issue of CC that affects network performance is load balancing. The VM's load is unbalanced because of the machine's over-utilization of its resources. The techniques that have already been proposed transfer the load of one machine to another, but it increases the likelihood that the machine on which the load is transferred will become overloaded. An optimization algorithm for load balancing in CC will be proposed in this research work.

1.4 Research Question

How can an improved genetic algorithm that assigns optimal resource allocation in cloud computing improve load balancing efficiency?

2 Related Work

The cloud provides scalable and efficient services to its users without the need for provisioning and managing physical infrastructure. As the cloud sector expands daily, new challenges emerge. One such challenge is cloud load balancing. A significant amount of recent research has been conducted to overcome these challenges, which proposes various methodologies, algorithms, frameworks, and so on. This section will provide an in-depth overview of research in Task Scheduling, VM Allocation, and Resource Allocation Using Optimization Methodologies.

2.1 Task Scheduling in Cloud Computing

In the cloud, task scheduling is critical because inefficient task scheduling can lead to overall cloud performance degradation. As a result, compared to the traditional methods, task scheduling in cloud environment is different. In Task Scheduling, priority is assigned to the tasks and subsequently these tasks are assigned to the available resources in order of their priority. Various recent algorithms have been proposed to improve the efficiency of task scheduling. This section elaborates such recent proposed algorithms.

Hongji Liu et al. (2022) created an adaptive task scheduling method based on the ant colony algorithm (ACO). The pheromones in the polymorphic ACO are significantly altered in response to changes in the environment, avoiding the creation of a local optimal solution and enhancing the algorithm's convergence speed. The new algorithm produced a distribution plan that had a balanced load rate, a quicker execution time, and a cheaper cost. Xuan Chen et al. (2020) employed the meta-heuristic whale optimization algorithm (WOA) to increase the performance of a cloud system with restricted computational resources for cloud job scheduling. Based on WOA, an enhanced WOA was constructed, which also enhances the optimum search optimal solution capacity in task scheduling. The method increased the efficiency with which small and large task system resources were utilized. For efficient task scheduling makespan minimization and energy consumption, Gokuldhev Mony et al. (2020) introduced Local Pollination-based Gray Wolf Optimizer (LPGWO) algorithm. The metrics used for performance evaluation were the makespan and energy usage. To distribute the data to the next packet of the candidate solution, the optimal searching factor was used to maximize the convergence speed. Kumar KP et al. (2020) designed a crow search algorithm (CSA) for task scheduling in the CC environment to shorten the work schedule's duration. This CSA finds the appropriate VM (VM) and shortens the makespan. CSA was used to schedule tasks with permanently set flight duration values. The main disadvantage of this strategy was that the flight length was not changed and an alternate VM was not recognized by the local search algorithm in place of the random selection.

The main technological gaps discovered in recent research are that the efficiency of contemporary algorithms has reduced and customers must wait a long time before their activity is scheduled in a cloud system. This occurs because of a strong emphasis on reducing algorithm execution time, resource consumption, and cost while ignoring the load on the VMs in issue.

2.2 Virtual Machine Allocation in Cloud Computing

One of the key challenges faced when allocating or Migrating VMs is to recognize whether a server is being overloaded. Many recent studies have been undertaken in this regard to efficiently and appropriately assign or migrate VMs in a server. This section discusses some of the most recent studies.

To handle the VM migration in the cloud, Mohd Sha Alam Khan et al. (2022) presented a hybrid optimization algorithm. This hybrid optimization technique is based on the cuckoo search and particle swarm optimization algorithms. This approach successfully reduced computation time, migration costs, and energy consumption. Additionally, this method achieved maximum resource allocation. Haiying Shen et al. (2020) presented an approach for balancing loads based on resource intensity (RIAL). RIAL considers communication interdependence between VMs while picking destination PMs to limit communication between VMs after migration. The algorithm enables quicker and lower-cost load balance convergence and minimizes the likelihood of future load imbalance by considering weights while picking VMs to migrate out and destination Physical Machines. Minoo Soltanshahi et al. (2019) created the Krill Herd technique for distributing VMs to physical hosts in cloud DCs. According to the simulation results, an efficient integration, and the selection of simple VM migration helped to improve energy efficiency. The algorithm for VM selection reduces energy consumption by 35% and 17%, respectively, by showing random selection and lowest migration time. In a heterogeneous cloud environment, a multiobjective Emperor Penguin Optimization (EPO) technique is suggested by Jitendra Kumar Samriya et al. (2021) to distribute VMs with power consumption. The preliminary findings show that the proposed EPO-based system is extremely successful in limiting energy usage, SLA violations (SLAV), and expanding QoS requirements for providing adequate cloud service.

Recent study has mostly concentrated on time or cost-efficient approaches to handle the problem of assigning or relocating VMs and increasing the resource allocation of the VMs but has not focused on services and load on the VMs, which can be regarded major shortcomings of these studies.

2.3 Resource Allocation using Optimization Methodologies in Cloud Computing

Due to limited available resources and growing consumer demands, the work of resource distribution becomes increasingly difficult. As a result, several novel models, and strategies for allocating resources have been developed. This section further elaborates some recent research conducted to develop optimization methods for enhancing resource allocation in CC.

A load balancing-based CC resource node allocation technique (NA-LB) was presented by Ye Bo et al. (2022). The deviations in the VM process parameter were detected by the algorithm as vector values of geographical vectors. Additionally, NA-LB algorithm is provided and employed as a load balancing index to evaluate the allocation of resource nodes. The algorithm performs well in terms of load balancing and significantly enhances the data processing efficiency of CC clusters. An optimum resource allocation based on hybrid differential parallel scheduling is proposed by Jing Wei et al. (2018) enhance resource allocation and task scheduling in CC. The algorithm optimizes resource allocation and boosts the efficiency of CC. The use of the algorithm shows enhanced clustering performance, strong ability to manage convergence computing resources. Ali Belgacem et al. (2020) proposed a dynamic resource allocation methodology that more efficiently and quickly fulfils the demands for resource allocation. The methodology is based on Spacing Multi-Objective Antlion Method (S-MOAL), a multi-objective search algorithm, to reduce both the time and cost of employing VMs. Results showed that it had a great influence on energy usage and fault tolerance and decreased the overall makespan. Zhao Tong et al. (2020) designed an algorithm based on Deep reinforcement learning (DRL) which determines if the work should be offloaded and assigns computer resources to the task. The best computing node is selected using DRL approach based on optimization objective and the best strategy for the objective problem is addressed. The DRL approach shows enhance performance in minimizing overall energy consumption and average task reaction time which boasts systems utility.
The recent studies have revealed that although the priority and cost of task scheduling has been greatly enhanced, there’s a need for methodologies that optimize the existing resource allocation in load balancing by undertaking the current and maximum load of the subsequent load of a VM.

2.4 Summary of Literature Review

The Table 1 shows the overall summary of the Literature Review.

Table 1: Summary of Literature Review

Article	Optimization method	Concept	Description	Findings
Hongji Liu et al	Ant Colony Optimization	Actual ant colonies' foraging behavior.	Created an adaptive task scheduling method based on the ant colony algorithm (ACO).	balanced load rate, a quicker execution time, and a cheaper cost.
Xuan Chen et al	Whale Optimization Algorithm	The social hunting behaviors of humpback whales.	Employed the meta-heuristic WOA to increase the performance of cloud job scheduling.	Increased efficiency of large and small tasks.
Gokuldhev Mony et al	Local Pollination-based Gray Wolf Optimizer Algorithm	Non-linear model of grey wolf position.	Introduced LPGWO algorithm to optimal searching factor was used to maximize the convergence speed.	efficient task scheduling makespan minimization and energy consumption.
Kumar KP et al	Crow search algorithm	Crows' intelligent behavior of hiding and retrieving food.	Designed a crow search algorithm (CSA) for task scheduling in the CC environment to shorten the work schedule's duration.	Permanently set flight duration values to shorten the makespan.
Mohd Sha Alam Khan et al	Hybrid cuckoo search and particle swarm optimization algorithms	CSA is based on reproduction of cuckoo birds and PSO is based on the study of the predation behavior of birds.	To handle the VM migration in the cloud, presented a hybrid optimization algorithm based on the cuckoo search and particle swarm optimization algorithms.	Reduced computation time, migration costs, and energy consumption.
Minoo Soltanshahi et al	Krill Herd technique	Based on the least distance between the food location and position of a krill.	Created the Krill Herd technique for distributing VMs to physical hosts in cloud DCs.	Efficient integration and the selection of simple VM migration helped to improve energy efficiency.
Jitendra Kumar Samriya et al.	Emperor Penguin Optimization	Inspired by social huddling behavior of emperor penguins	A multiobjective (EPO) technique is suggested to distribute VMs with power consumption.	Extremely successful in limiting energy usage, SLA violations (SLAV), and expanding QoS requirements.
Ali Belgacem et al	Spacing Multi-Objective Antlion Method	Based on Nature-inspired metaheuristics of the swarm intelligence field.	A dynamic resource allocation methodology that more efficiently and quickly fulfils the demands for resource allocation.	Great influence on energy usage and fault tolerance and decreased the overall makespan.

3 Research Methodology

The focus of this research is to improve the traditional GA's mutation process in order to improve the efficiency of load balancing in CC. The typical GA passes through three stages: initial population, crossover, and mutation. The EGA will aid in reducing the time it takes the DC to handle all the work performed by its VMs.

3.1 Genetic Algorithm

GAs are meta-heuristic algorithms that are based on natural processes. It is a well-known optimization procedure that is used to identify the best potential solutions. The GA contains three phases: Initial Population, Crossover, and Mutation.

3.1.1 Initial Population

The initial population is generated during the first step of the GA. The initial population consists of a group of individuals who are potential solutions to an issue. The most popular way of initializing the population based on a set of rules is heuristic initialization. The initial population is made up of chromosomes that function as individuals. Genes that contain information about the chromosomes live on the chromosomes. Figure 1 depicts the initial population.

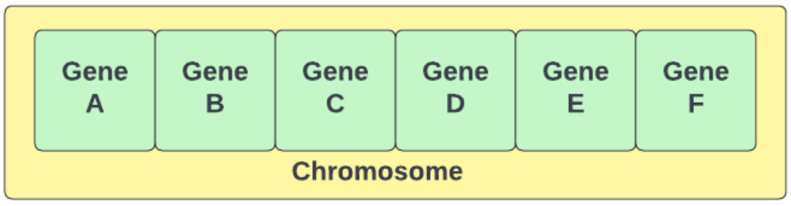

Figure 1:Initial Population.

3.1.2 Fitness Function

This phase is regarded to be part of the initial population phase. This phase assesses how well the chromosome in the original population fits the needed parameters. The fitness score identifies which individual is the best viable solution to replicate to address a set of problems.

3.1.3 Crossover

Using the fitness function, two parent genes are chosen as the fittest individuals in this phase. A crossover operator is applied to the chromosome using these two genes as the fittest parents to build a new fittest chromosome. The most frequent crossover procedure is single-point crossover, in which two parent genes are switched directly. The crossover phase is depicted in Figure 2.

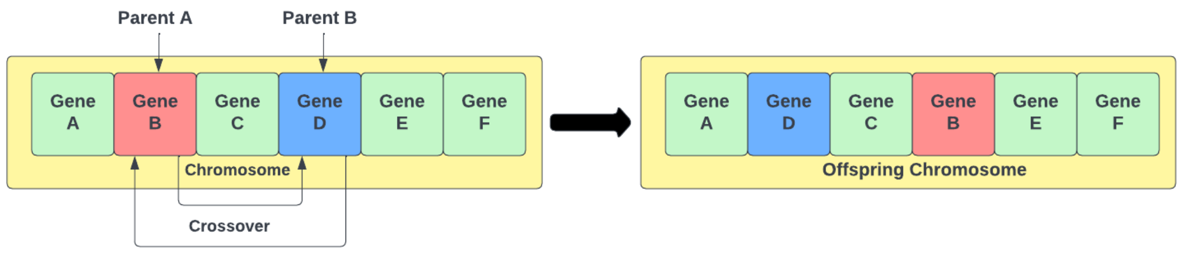

Figure 2: Crossover.

3.1.4 Mutation

The offspring chromosome is mutated with a low random frequency, where the value of the gene is altered to a new random best suited value for the chromosome. This procedure is mostly used to increase variety among the best-fitting genes on the chromosomes. The Mutation Phase is depicted in Figure 3.

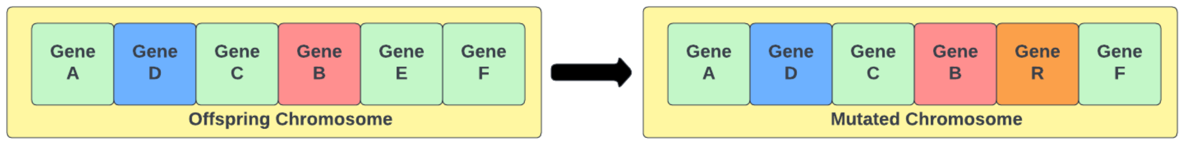

Figure 3: Mutation.

This mutated chromosome is used to provide the best potential solution for a set of problems using a GA.

3.1.5 Process Flowchart

The GA's flowchart is shown in Figure 4. Six different steps make up the GA method. The algorithms' parameters are initialized during the initialization step. The parameters that were passed in the previous stage are used to produce a population in the following phase. The population's best fitness function value and the two fittest parents

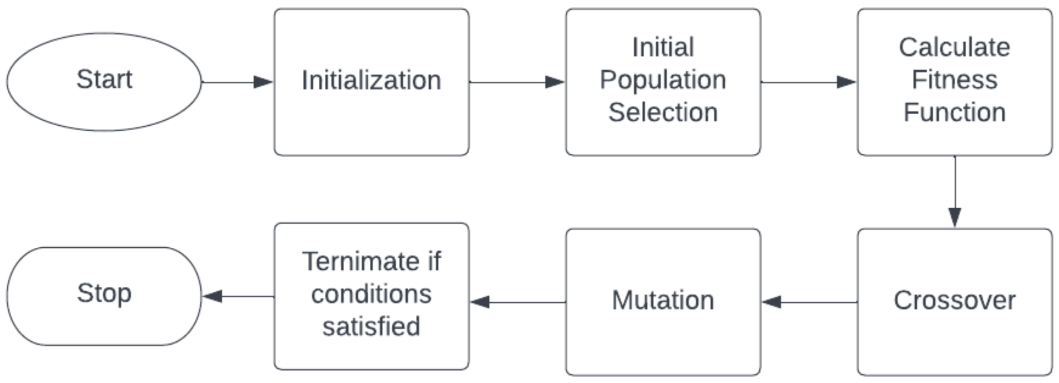
Start
Initialization
Initial Population Selection
Calculate Fitness Function
Crossover
Mutation
Ternimate if conditions satisfied
Stop

Figure 4: Genetic algorithm flowchart.

are computed in the subsequent step. In the crossover procedure, these two parents are immediately switched, and a random gene from the population is modified by altering its parameters. If the termination criteria are satisfied as a new unique population is produced utilizing the preceding stages, the process is terminated.

4 Design Specification

The conventional GA's method will be performed by the EGA, although the mutation process will be improved. The specifications for the DC, VMs, and cloudlets are set during the initialization process. These specifications are used to establish the DC, VMs, and cloudlets, which will serve as the initial population for the EGA. After successfully creating the initial population, the best fitness value function is applied to it in order to determine the best fitness value and the two best fit VMs, parent A and parent B. Then, using a single-point crossover procedure, the two VMs that fit the best are immediately switched, producing an offspring. The following phase will include mutating these offspring. In a typical GA, one parameter of a VM that is chosen at random is modified, which causes the VM to mutate. But as a result, the resultant VM might not work since the parameters weren't allocated properly. To prevent this, the VM will be modified by switching it out completely with a different VM that has the appropriate specifications, rather than merely altering one parameter. The resulting mutated population will be used to generate a cloud infrastructure with DCs, VMs and cloudlets and a simulation will be run which will process the cloudlets in the VMs created in the DC and output the overall task completion time of the cloudlets.

4.1 Enhanced Genetic Algorithm

The EGA is based on the GA. This algorithm will be used to generate the best possible VM deployment in a CC infrastructure while also taking the load of the VMs into consideration. The EGA will also have three phases like the GA, but the mutation phase will be enhanced to deploy a better VM in the architecture.

4.1.1 Initial Population

During the initial process of the EGA, the initial population is generated. The initial population will consist of the individuals that are important to solve a problem. In our case, the problem is improving the efficiency of load balancing and the individuals that contribute to finding a solution to this problem is the DC, the VMs and the Cloudlets. As we need to find the ideal placement of VMs, the chromosomes will be the group of VMs where each VM represents an individual gene. Figure 5 depicts the initial population of the proposed architecture.

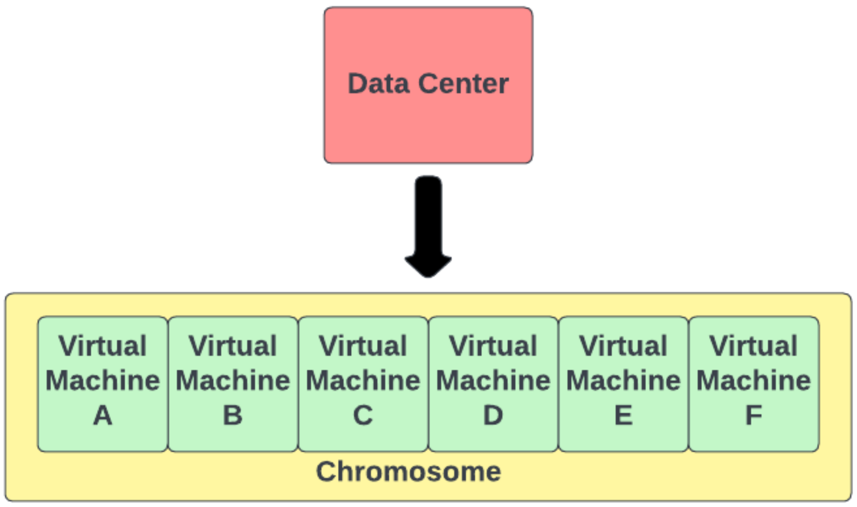

Figure 5: Initial Population EGA.

4.1.2 Fitness Function

This phase will be the part of the Initial Population process where the fitness function of the VMs will be calculated. The fitness function will be calculated using a threshold value, that being the maximum threshold load of the VMs. Using this, the two best fit parents will also be determined will be the input for the next process of EGA. The two best fit parents will be the two best VMs.

4.1.3 Crossover

Taking the input from the previous process, the two determined best fittest parents are swapped using a crossover operator. This process is called crossover. The two fittest parents will be the two best fittest VMs determined by the fitness function. The crossover process is depicted in Figure 6.

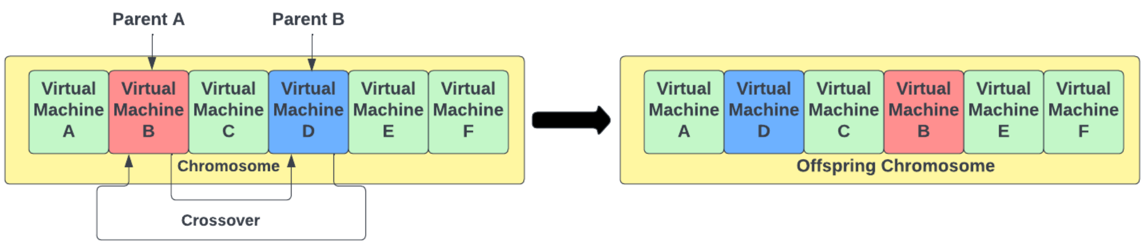

Figure 6: Crossover.

4.1.4 Mutation

The generated offspring VM will be mutated in this process. In a tradition mutation process, a random VM will be selected from the pool of VMs and a parameter of the VM would have been changed, like the number of CPUs, RAM, Storage, etc. However, this process will be enhanced by changing the whole VM with better parameters. This process is depicted in the Figure 7.

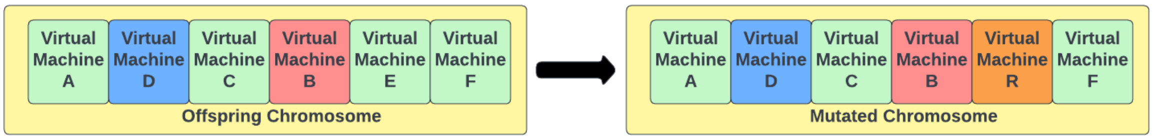

Figure 7: Mutation EGA.

Using this mutated list of VMs, the load will be balanced across the CC architecture.

4.2 Pseudocode

The Pseudocode of the algorithm is given as follows:

```
Step 1: Generate Specifications for DCs, VMs and Cloudlets.
Step 2: Initialize Initail Population by creating DCs, VMs and Cloudlets using the specifications.
Step 3: Find the fitness value, best_fit_parent_and best_fit_parent_b using the fitness
        function:
        Set threshold = 10000
        Set temp = 0
        Set parent_a = 0
        Set parent_b = 0
        For i = 0 to number of VMs
        For j = 0 to number of cloudlets
        temp = length of cloudlet / mips of VMs
        Set sum = sum +temp
        If sum < threshold
        threshold = sum
        parent_b = parent_a
        parent_a = i
        End If
        End For
        End for
Step 4: Crossover Machines by swapping VMs parent_a with VM parent_b.
Step 5: Select a random VM from the list of VMs and mutate it by changing the VM to a new and better VM.
```

Step 6: Submit the DCs, Mutated VMs and Cloudlets to the simulation tool and run the simulation.

5 Implementation

CloudSim simulations are utilized to carry out the proposed study. CloudSim is a platform for CC that simulates a virtual CC environment utilizing necessary provisioned resources. Figure 8 shows the Cloud architecture produced in CloudSim.

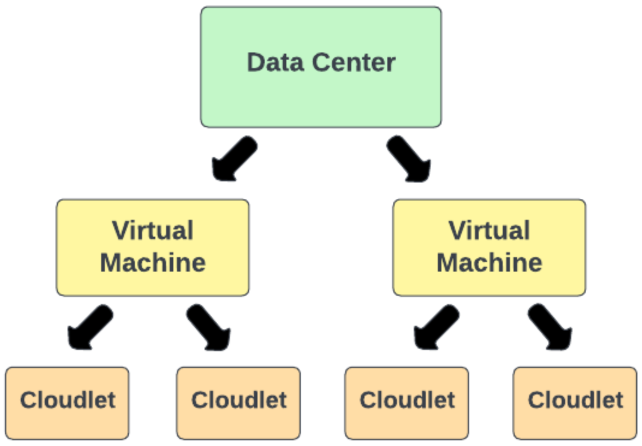

Figure 8: Proposed Architecture.

5.1 Data center

It performs the functions of a physical machine. It serves as a server for the creation of VMs. It is one of a CC infrastructure's crucial parts. Low level processing duties will be handled by the VMs that were established in the DC. The parameters listed in Table 2 are used to establish one DC to implement EGA.

Table 2: Data Center Parameters

Parameter	Value
RAM	16384
Storage	100000
Bandwidth	10000
Architecture	x86
OS	Linux
VMM	Xen
Cost	5
Cost per Memory	0.1
Cost per Storage	0.2
Cost per Bandwidth	0.2
Timezone	10

5.2 Virtual Machine

A DC builds VMs, which are not physical machines. Being a separate computer inside the DC allows it to replicate the architecture of physical machines while operating on resources provided by the DC. These VMs are hosted by the DCs and carry out the execution of tasks allocated to them. Table 3 shows the parameters used to created VMs in CloudSim.

Table 3: Virtual Machine Parameters

Parameters	Value
Size	8000
Ram	512
Mips	4000
Bandwidth	20
Pesnumber	2
Vmm	Xen

5.3 Cloudlets

A cloudlet in CloudSim is an equivalent of a task that is assigned to a VM to execute. The length of the cloudlet is the total number of instructions taken by a cloudlet to complete. Table 4 shows the parameters of the cloudlet.

Table 4: Cloudlets Parameters

Parameters	Value
Length	4000
File size	200
Output size	200
Pesnumber	1

The EGA and the well-known particle swarm optimization method will be applied in CloudSim using this CC scenario to produce the optimal VMs for the DCs to conduct a variety of experiments. The overall number of experiments will be Ten, with Five in EGA and Five in PSO, each with various numbers of cloudlets. The parameters for the experiments are given in Table 5:

Table 5: Number of VMs and Cloudlets

Iteration	VMs	Cloudlets
1	10	50
2	10	75
3	10	100
4	10	125
5	10	150

6 Evaluation

In CloudSim, the Cloud Architecture is built, and two different simulations are run. In the first set of simulations, different numbers of cloudlets are used to balance the load throughout the proposed architecture using the EGA. The second set use the PSO method to evenly distribute the load across the suggested architecture's various cloudlet sizes. The simulation's outcomes are as follows:

6.1 Execution Time

Figure 9 depicts the time it takes the VMs to execute all the cloudlets. Execution time is measured by the overall time taken by the DC to execute all the cloudlets. As shown in Figure 9, the EGA algorithm outperforms the PSO algorithm. This is because the EGA-allocated mutated VMs execute the cloudlets faster than the PSO-allocated VMs.

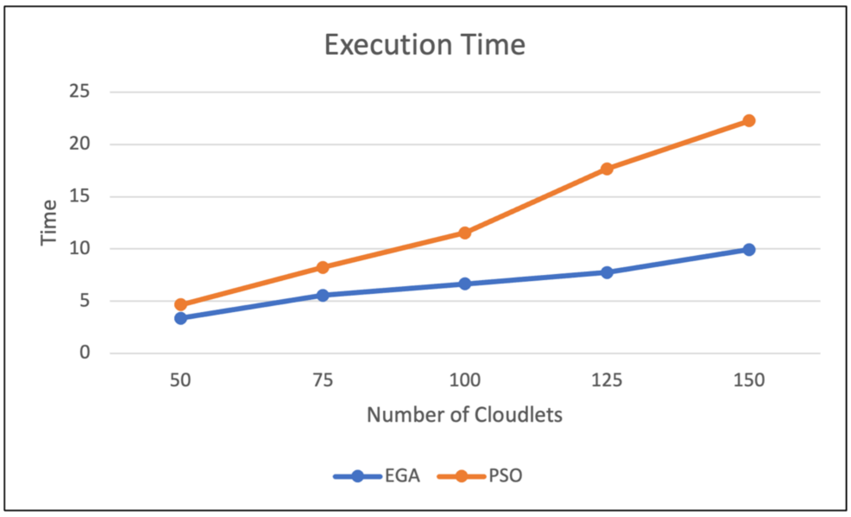

Figure 9: Execution Time Comparison.

6.2 Resource Utilization

The execution time of all cloudlets combined is shorter than that of PSO assigned resources when EGA is used to allocate resources. Using this, the resource utilization of all the VMs can be estimated by multiplying the entire cloudlet execution time into CPU utilization time. Figure 10 depicts the EGA and PSO algorithms' resource utilization. It can be observed that EGA algorithm lends to better results compared to PSO and utilizes less resources as the number of cloudlet increases.

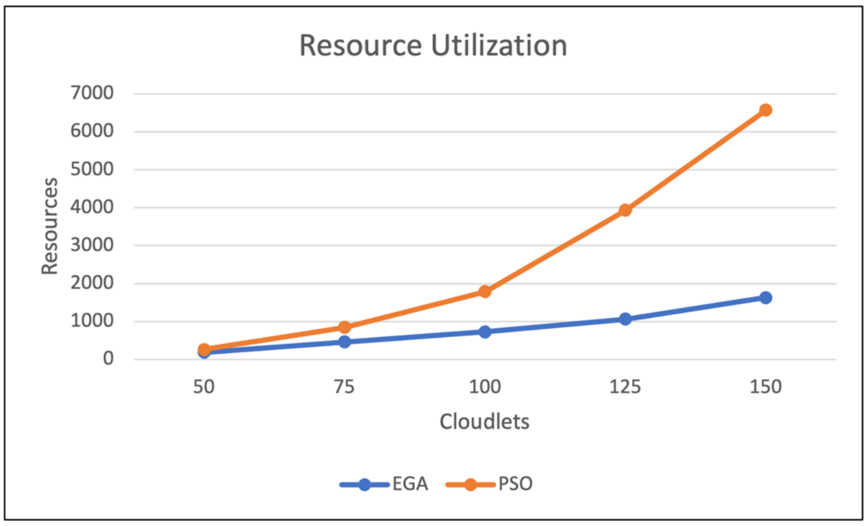

Figure 10: Resource Utilization Comparison.

6.3 Energy Consumption

Figure 11 shows that the simulation results of EGA are more energy-efficient than that of PSO Algorithm. Energy consumption can be calculated by dividing the resource allocation over time. EGA has been able to reduce energy consumptions more significantly than PSO.

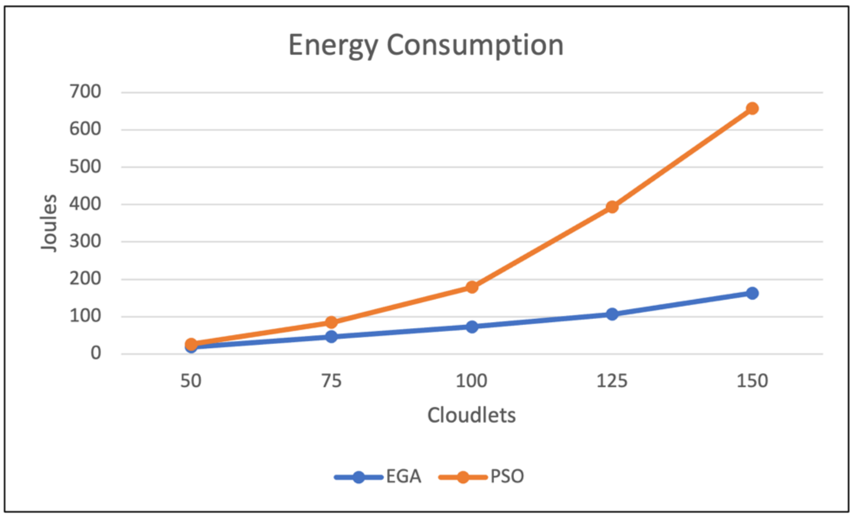

Figure 11: Energy Consumption Comparison.

6.4 Discussion

The results which are obtained, it clearly indicates that performing load balancing in CC using EGA is more efficient than performing with PSO algorithm which is one of the well-known load balancing algorithms. As seen in the results, the increase in time, resource, and energy consumption in PSO algorithm with increasing cloudlets is way more than the increase in time, resource, and energy consumption in EGA algorithm. PSO uses large number of particles and iterates them for large number of times to swamp the best fitted VMs to improve load balancing. This leads to increase in time needed for the algorithm to execute, increases the resource utilization of the VMs and in turn increases the energy consumption of the VMs. On other hand, EGA uses the fitness function to calculate the best two fit VMs, swaps them and then randomly mutates one VM to significantly increase the efficiency of the VMs. As a result, it can be observed that the EGA is better at balancing load throughout different loads of cloudlets. This proposed research fulfils the research objectives of increasing the load balancing efficiency of the CC by decreasing the execution time, resource utilization, and energy consumption by using EGA.

7 Conclusion and Future Work

In this research, the concept of GA is used to propose EGA and enhanced to increase the efficiency of load balancing in CC. During the initialization phase of GA, DCs, VMs and cloudlets are created using their parameters and the Initial population is generated. Using the fitness function, the best two fitted parents are in the population of VMs and are swapped. The offspring generated after crossover is further mutated by randomly swapping out VM for a newly created VM. Using the DC, mutated VMs and the Cloulets, a load balancing simulation is run and it is observed that load balancing carried out using EGA takes less execution time, resource utilization and energy consumption when compared to PSO algorithm, one of the well-known load balancing algorithm. In future, other features such as scalability, security, newtwork traffic accessibility, latency, and reliability can be extended by enhancing the EGA Furthermore. Furthermore, EGA has a scope to be further enhanced to also carry out load balancing in CC by allocating tasks that are dependent dynamically.

References

Abualigah, L. and Diabat, A. (2020) "A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in Cloud Computing Environments," *Cluster Computing*, 24(1), pp. 205–223. Available at: https://doi.org/10.1007/s10586-020-03075-5.

Alameen, A. and Gupta, A. (2020) "Fitness rate-based rider optimization enabled for Optimal Task Scheduling in cloud," *Information Security Journal: A Global Perspective*, 29(6), pp. 310–326. Available at: https://doi.org/10.1080/19393555.2020.1769780.

Belgacem, A. *et al.* (2020) "Efficient Dynamic Resource Allocation Method for Cloud Computing Environment," *Cluster Computing*, 23(4), pp. 2871–2889. Available at: https://doi.org/10.1007/s10586-020-03053-x.

Bo, Y. (2022) "Cloud computing resource node allocation algorithm based on load balancing strategy," *2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC)* [Preprint]. Available at: https://doi.org/10.1109/itoec53115.2022.9734399.

Chen, X. *et al.* (2020) "A WOA-based optimization approach for task scheduling in Cloud Computing Systems," *IEEE Systems Journal*, 14(3), pp. 3117–3128. Available at: https://doi.org/10.1109/jsyst.2019.2960088.

Gokuldhev, M., Singaravel, G. and Ram Mohan, N.R. (2019) "Multi-objective local pollination-based Gray Wolf Optimizer for task scheduling heterogeneous cloud environment," *Journal of Circuits, Systems and Computers*, 29(07), p. 2050100. Available at: https://doi.org/10.1142/s0218126620501005.

Khan, M.S. and Santhosh, R. (2022) "Hybrid optimization algorithm for VM Migration in cloud computing," *Computers and Electrical Engineering*, 102, p. 108152. Available at: https://doi.org/10.1016/j.compeleceng.2022.108152.

Kumar, K.P., Ragunathan, T. and Vasumathi, D. (2022) "Virtual machine consolidation using enhanced crow search optimization algorithm in cloud computing environment," *Lecture Notes in Electrical Engineering*, pp. 841–851. Available at: https://doi.org/10.1007/978-981-19-2281-7_77.

Liu, H. (2022) "Research on cloud computing adaptive task scheduling based on Ant Colony algorithm," *Optik*, 258, p. 168677. Available at: https://doi.org/10.1016/j.ijleo.2022.168677.

M. Smale, N. Jamora, and L. Guarino, “Valuing Plant Genetic Resources in genebanks: Past, present and future,” *Plant genetic resources*, pp. 35–53, 2021.

Rajagopalan, A., Modale, D.R. and Senthilkumar, R. (2019) “Optimal scheduling of tasks in cloud computing using hybrid Firefly-genetic algorithm,” *Learning and Analytics in Intelligent Systems*, pp. 678–687. Available at: https://doi.org/10.1007/978-3-030-24318-0_77.

Samriya, J.K. *et al.* (2021) “Intelligent SLA-aware VM allocation and energy minimization approach with EPO algorithm for cloud computing environment,” *Mathematical Problems in Engineering*, 2021, pp. 1–13. Available at: https://doi.org/10.1155/2021/9949995.

Shen, H. and Chen, L. (2020) “A resource usage intensity aware load balancing method for virtual machine migration in cloud datacenters,” *IEEE Transactions on Cloud Computing*, 8(1), pp. 17–31. Available at: https://doi.org/10.1109/tcc.2017.2737628.

Soltanshahi, M., Asemi, R. and Shafiei, N. (2019) “Energy-aware virtual machines allocation by krill herd algorithm in Cloud Data Centers,” *Heliyon*, 5(7). Available at: https://doi.org/10.1016/j.heliyon.2019.e02066.

Wei, J. and Zeng, X.-fa (2018) “Optimal Computing Resource Allocation Algorithm in cloud computing based on hybrid differential parallel scheduling,” *Cluster Computing*, 22(S3), pp. 7577–7583. Available at: https://doi.org/10.1007/s10586-018-2138-7.

Zhang, Y. *et al.* (2022) “Dynamic job shop scheduling based on Deep Reinforcement Learning for multi-agent manufacturing systems,” *Robotics and Computer-Integrated Manufacturing*, 78, p. 102412. Available at: https://doi.org/10.1016/j.rcim.2022.102412.