Multi-objective cuckoo optimizer for task scheduling to balance workload in cloud computing

Special Issue Article Published: 17 August 2024

Volume 106, pages 3447–3478, (2024) Cite this article

Brototi Mondal & Avishek Choudhury

Abstract

A cloud load balancer should be proficient to modify it's approach to handle the various task kinds and the dynamic environment. In order to prevent situations where computing resources are excess or underutilized, an efficient task scheduling system is always necessary for optimum or efficient utilization of resources in cloud computing. Task Scheduling can be thought of as an optimization problem. As task scheduling in the cloud is an NP-Complete problem, the best solution cannot be found using gradient-based methods that look for optimal solutions to NP-Complete problems in a reasonable amount of time.

Therefore, the task scheduling problem should be solved using evolutionary and meta-heuristic techniques. This study proposes a novel approach to task scheduling using the Cuckoo Optimization algorithm. With this approach, the load is effectively distributed among the virtual machines that are available, all the while keeping the total response time and average task processing time(PT) low. The comparative simulation results show that the proposed strategy performs better than state-of-the-art techniques such as Particle Swarm optimization, Ant Colony optimization, Genetic Algorithm and Stochastic Hill Climbing.

Data availability

No datasets were generated or analysed during the current study.

References

1. Weiss A (2007) Computing in the clouds. Networker 11(4):16–25

   Article Google Scholar

2. Somula R, Sasikala R (2019) A honey bee inspired cloudlet selection for resource allocation. In: Smart intelligent computing and applications: proceedings of the second international conference on SCI 2018, vol 2. Springer, pp 335–343

3. Al Nuaimi K, Mohamed N, Al Nuaimi M, Al-Jaroodi J (2012) A survey of load balancing in cloud computing: challenges and algorithms. In: 2012 second symposium on network cloud computing and applications, IEEE, pp 137–142

4. LD B, Krishna V (2013) Honey bee behavior inspired load balancing of tasks in cloud computing environments. Appl Soft Comput 13(5):2292–2303

   Article Google Scholar

5. Nakai A, Madeira E, Buzato LE (2015) On the use of resource reservation for web services load balancing. J Netw Syst Manage 23:502–538

   Article Google Scholar

6. Dasgupta K, Mandal B, Dutta P, Mandal JK, Dam S (2013) A genetic algorithm (ga) based load balancing strategy for cloud computing. Procedia Technol 10:340–347

   Article Google Scholar

7. Wang Q, Fu X-L, Dong G-F, Li T (2019) Research on cloud computing task scheduling algorithm based on particle swarm optimization. J Comput Methods Sci Eng 19(2):327–

335

Google Scholar

8. Kruekaew B, Kimpan W (2014) Virtual machine scheduling management on cloud computing using artificial bee colony. Proc Int MultiConference Eng Comput Sci 1:12–14

Google Scholar

9. Azad P, Navimipour NJ (2017) An energy-aware task scheduling in the cloud computing using a hybrid cultural and ant colony optimization algorithm. Int J Cloud Appl Comput (IJCAC) 7(4):20–40

Google Scholar

10. Farrag AAS, Mohamad SA, El Sayed M (2019) Swarm intelligent algorithms for solving load balancing in cloud computing. Egypt Comput Sci J 43(1):45–57

Google Scholar

11. Madni SHH, Latiff MSA, Coulibaly Y, Abdulhamid SM (2017) Recent advancements in resource allocation techniques for cloud computing environment: a systematic review. Clust Comput 20:2489–2533

Article Google Scholar

12. Azad P, Navimipour NJ, Hosseinzadeh M (2019) A fuzzy-based method for task scheduling in the cloud environments using inverted ant colony optimisation algorithm. Int J Bio-Inspir Comput 14(2):125–137

Article Google Scholar

13. Zaman SK, Maqsood T, Ali M, Bilal K, Madani SA, Khan A (2019) A load balanced task scheduling heuristic for large-scale computing systems. Comput Syst Sci Eng 34:4

Google Scholar

14. Mishra SK, Sahoo B, Parida PP (2020) Load balancing in cloud computing: a big picture. J King Saud Univ-Comput Inf Sci 32(2):149–158

Google Scholar

15. Balaji K, Sai Kiran P (2017) Efficient resource allocation algorithm with optimal throughput in cloud computing. J Adv Res Dyn Control Syst 9:1902–1910

Google Scholar

16. Annie Poornima Princess G, Radhamani A (2021) A hybrid meta-heuristic for optimal load balancing in cloud computing. J Grid Comput 19(2):21

Article Google Scholar

17. Kumar C, Marston S, Sen R, Narisetty A (2022) Greening the cloud: a load balancing mechanism to optimize cloud computing networks. J Manag Inf Syst 39(2):513–541

Article Google Scholar

18. Kamila NK, Frnda J, Pani SK, Das R, Islam SM, Bharti P, Muduli K (2022) Machine learning model design for high performance cloud computing & load balancing resiliency: an innovative approach. J King Saud Univ-Comput Inf Sci 34(10):9991–10009

Google Scholar

19. Kumar KV, Rajesh A (2023) Multi-objective load balancing in cloud computing: a meta-heuristic approach. Cybern Syst 54(8):1466–1493

Article Google Scholar

20. Kumar KP, Ragunathan T, Vasumathi D, Prasad PK et al (2020) An efficient load balancing technique based on cuckoo search and firefly algorithm in cloud. Algorithms 423:422–432

Google Scholar

21. Wickremasinghe B, Calheiros RN, Buyya R (2010) Cloudanalyst: a cloudsim-based visual modeller for analysing cloud computing environments and applications. In: 2010 24th IEEE international conference on advanced information networking and applications, IEEE, pp. 446–452

Funding

No funding is involved.

Author information

Avishek Choudhury has contributed equally to this work.

Authors and Affiliations

Department of Computer Science, Sammilani Mahavidyalaya, University of Calcutta, E.M.Bypass, Kolkata, West Bengal, 700094, India
Brototi Mondal

Department of Computer Science, New Alipore College, L Block, New Alipore, Kolkata, 700053, West Bengal, India
Avishek Choudhury

Contributions

The first author wrote most of the manuscript, where few sub-sections like Sects. 3.3.1 (partly), 3.3.3, 3.3.4 have been written by the second author, along with preparing the Figs. 8 and 9. All authors reviewed the manuscript.

Corresponding author

Correspondence to Brototi Mondal.

Ethics declarations

Conflict of interest

The authors declare no conflict of interest.

Additional information

Publisher's Note

Rights and permissions

About this article

Cite this article

Mondal, B., Choudhury, A. Multi-objective cuckoo optimizer for task scheduling to balance workload in cloud computing. *Computing* 106, 3447–3478 (2024). https://doi.org/10.1007/s00607-024-01332-8

Received	Accepted	Published
04 February 2024	24 July 2024	17 August 2024

Issue Date

November 2024

DOI

https://doi.org/10.1007/s00607-024-01332-8

Keywords

Cloud computing Task scheduling Cuckoo optimization algorithm Response time

Processing time

Mathematics Subject Classification

68 49

Multi-objective Cuckoo Optimizer for Task Scheduling to Balance Workload in Cloud Computing

Brototi Mondal[1*] and Avishek Choudhury[2†]

[1*]Department of Computer Science, Sammilani Mahavidyalaya, University of Calcutta, E.M.Bypass, Kolkata, 700094, West Bengal, India.
[2]Department of Computer Science, New Alipore College, L Block, New Alipore, Kolkata, 700053, West Bengal, India.

*Corresponding author(s). E-mail(s): brototi.snp@gmail.com;
Contributing author: avishek.nac.cs@gmail.com;
†These authors contributed equally to this work.

Abstract

A cloud load balancer should be proficient to modify it's approach to handle the various task kinds and the dynamic environment. In order to prevent situations where computing resources are excess or underutilized, an efficient task scheduling system is always necessary for optimum or efficient utilization of resources in cloud computing. Task Scheduling can be thought of as an optimization problem. As task scheduling in the cloud is an NP-Complete problem, the best solution cannot be found using gradient-based methods that look for optimal solutions to NP-Complete problems in a reasonable amount of time. Therefore, the task scheduling problem should be solved using evolutionary and meta-heuristic techniques. This study proposes a novel approach to task scheduling using the Cuckoo Optimization (CO) algorithm. With this approach, the load is effectively distributed among the virtual machines (VMs) that are available, all the while keeping the total response time (RT) and average task processing time(PT) low. The comparative simulation results show that the proposed strategy performs better than state-of-the-art techniques such as Particle Swarm optimization (PSO), Ant Colony optimization (ACO), Genetic Algorithm (GA) and Stochastic Hill Climbing (SHC).

Keywords: Cloud Computing,Task Scheduling, Cuckoo Optimization Algorithm, Response Time, Processing Time

1 Introduction

The primary goal of cloud computing is to provide end users with distributed, virtualized, and elastic resources, treating them as utilities [1].

The number of data centers in the cloud is increasing exponentially to meet the demand for computing power[2].

The majority of cloud providers adhere to Service Level Agreement (SLA) criteria while providing support for their services. The many Quality of Service (QoS) guidelines that the supplier promises make up the SLAs. Task scheduling is essential for preserving SLA and quality of service in cloud computing. One of the key elements of fully using cloud computing's potential is to schedule tasks efficiently.

The scheduling of tasks in a cloud environment has become a significant challenge in recent times and has gained significant attention in the field of research. Task scheduling involves mapping tasks to available resources based on the characteristics and requirements of the tasks. Once the users input their assigned tasks, the cloud broker considers the qualities of both the resources and the tasks to map them to accessible resources. These resources should be used appropriately and efficiently in order to allocate the optimal resource to the task, taking into account the requirements of an optimum resource allocation and attaining acceptable Quality of Service.

In cloud environments, scheduling is classified as an NP-complete problem.As the number of users and the size of their associated computational tasks grow, the complexity of scheduling these tasks increases proportionally.Existing task scheduling strategies often fall short of meeting these demands. Therefore, improved algorithms for task scheduling are necessary to decrease computation time and associated costs. Task scheduling not only assists in managing the makespan of VMs but also plays a crucial role in upholding the conditions outlined in the Service Level Agreement (SLA), thereby ensuring better service quality for consumers through improved Quality of Service (QoS).

Task scheduling is one crucial step in raising cloud computing's overall performance.An efficient task scheduling algorithm has a direct impact on overall system performance.The process of assigning user tasks to VMs for processing is known as task scheduling. Customers should be able to get their specified tasks completed on the VM in the least period of time by using an effective scheduling strategy. However, the service provider needs a certain kind of scheduler that can optimize resource use and boost the satisfaction of customers. This makes it more important for the service provider to have an appropriate task scheduling strategy.

Numerous solutions have been put up to address the task scheduling issue, and each one seeks to meet one or more of the restrictions that service providers and consumers have taken into consideration [3].This study differs from the previous ones in that the proposed approach aims to present a dynamic multi-objective solution for task scheduling on VMs, while also taking other Quality of Service aspects into consideration and boosting the degree of load balancing. The following goals are intended to be achieved by the suggested strategy:

- Decreasing makespan
- Maximizing resource utilization

- Reducing the degree of imbalance on VMs
- Lowering of response and processing times

In summary, the key contributions of this paper are as follows:

1. The robust search characteristic of Cuckoo search is used to identify the over-utilized hosts and move one or more Virtual Machines (VMs) to the other hosts from them. The over-utilized hosts' use may decrease as a result of this approach.
2. To migrate tasks from overloaded VMs and optimize task processing time as well as overall response time, a multi-objective task scheduling optimization model was designed and to solve this suggested optimization model, a multi-objective Cuckoo Search Optimizer was also developed.
3. Conducting comprehensive experiments by extending CloudAnalyst, we have successfully validated that the proposed algorithm is viable in a heterogeneous environment.

The following sections make up the remaining part of this paper: The review of the literature is presented in the next part along with a quick explanation of task scheduling in cloud computing. The cuckoo search algorithm is presented in Sect. 3, and our suggested method for cloud scheduling is shown in Sect. 4. The presentation of the experimental findings and assessment work is aided by Section 5. Our conclusion is given in Section 6.

2 Literature Review

Over the past decade, numerous studies have explored load balancing in the cloud environment, providing researchers with a robust framework to comprehend the various aspects of this issue. This section provides a summary of earlier survey studies, consolidating the findings and insights gained from prior research in this domain.

In [4], a model of honeybee behavior for seeking and gathering food has been created and the amount of time that tasks in the queue must wait is minimized by taking into consideration their priority. When there are many tasks to be accomplished and little resources available, this can be especially troublesome since lower-priority tasks might never receive the chance to be completed.The stability and resilience of the system are seriously jeopardized by a single point of failure brought about by the production of bees from a single source. Our proposed technique overcomes this drawback.

In the work by Nakai et al.[5], have offered a technique based on the reserve policy to divide requests across repeating servers . This enables overloaded servers to hold onto some of the distant servers' capacity before processing a new request; requests that exceed the normal limits of the remote servers will have some of their capacity removed. The outcomes of the simulation demonstrated that their suggested approach shortens response times. Even though this strategy speeds up response times, certain queries are still turned down. Consequently, this approach was inappropriate for our job.

Dasgupta et al. [6] introduced a load balancing method using one of the most popular artificial intelligence techniques—the Genetic Algorithm.In the context of a cloud environment, the authors utilized GA-based load balancing techniques to identify a globally optimal processor for a task. They treated the arrival of tasks as linear, and rescheduling was not considered as a universally optimal solution. This approach highlights the use of GA in addressing load balancing challenges within cloud computing.

Various optimization techniques, such as Particle Swarm Optimization [7], Artificial Bee Colony [8], and Ant Colony Optimization [9], have been employed to tackle load balancing issues. According to this study, these evaluated algorithms outperform conventional ones in terms of metrics like makespan and response time. In dynamic cloud environments where workloads change often, there is a risk of performing less well than ideal. The algorithm might not be able to adjust to these changes fast enough, which would lead to an ineffective load distribution. Ultimately, the reliability and overall efficiency of cloud services may be impacted by these constraints.

In [10], the authors introduced three variants of the Ant-Lion Optimizer (ALO) and Grey Wolf Optimizer (GWO) as task schedulers aimed at minimizing makespan. While ALO and GWO outperformed FFA in terms of makespan reduction, they demonstrated comparable results to PSO, with instances where ALO even surpassed PSO in performance.But this technique could find it difficult to adjust quickly in extremely dynamic cloud environments when workload and resource availability change regularly, which could cause delays and poor system performance. These approaches may also be computationally demanding, which raises the overhead for cloud systems and can offset the advantages of optimal scheduling. Consequently, the general effectiveness and dependability of cloud services may be impacted by the usage of ALO and GWO as task schedulers.

In the study conducted by Syed Hamid and Hussain Madni [11], resource allocation strategies in cloud computing were thoroughly examined and evaluated. The paper identified factors that could enhance the functions and features of cloud systems. Additionally, the article delves into the discussion of the requirements for resource distribution in the cloud, covering policies, strategies, and algorithms for efficient resource distribution and migration to best support both providers and users in the cloud computing environment.

Azad et al. [12] proposed a fuzzy-based inverted ant colony optimization technique for scheduling tasks and balancing the load in a cloud environment. This approach likely involves leveraging fuzzy logic and ant colony optimization principles to address the challenges of load balancing in the context of cloud computing. Fuzzy logic allows for handling uncertainty and imprecision, while ant colony optimization is inspired by the foraging behavior of ants and is used to optimize paths and allocations in a network. The combination of these techniques aims to achieve effective task scheduling and load balancing in the cloud.

In the computational model considered by Zaman et al. [13], each machine in a cloud environment has a bounded capacity to execute a predefined number of tasks simultaneously. In this context, the authors proposed a task scheduling heuristic called Extended High to Low Load (ExH2LL). This heuristic aims to balance the workload across the available computing resources, improving resource utilization and reducing the makespan. ExH2LL dynamically identifies task-to-machine assignments by considering the existing load on all machines.

In the work by Mishra et al.[14], the authors explored heuristic-based techniques and investigated the use of various loads, including network, CPU, memory, etc., to enhance performance in the cloud environment. This suggests that the study focused on developing heuristic methods to address resource allocation and optimization challenges in cloud computing, particularly considering diverse types of loads.

On the other hand, Balaji and Saikiran [15] discussed various resource allocation problems and provided optimal capital allocation strategies for demanding task requests. This work likely delves into resource management and allocation strategies, aiming to optimize the allocation of resources in response to specific task requirements in the cloud environment.

G. Annie Poornima and Radhaman [16] synergized the strengths of Harries Hawks Optimization and the Pigeon-Inspired Optimization Algorithm to create an efficient load balancing system. The primary goal of this scheme is to enhance resource utilization while minimizing task response time. According to simulation results, the load balancing approach using the combined Hawks Optimization and Pigeon-Inspired Optimization algorithm successfully achieved optimal load distribution among VMs in a shorter duration when compared to existing algorithms. Diverse performance metrics, such as computational time, cost, throughput analysis, makespan, latency, and execution time, underwent evaluation and were compared against Harries Hawks Optimization, Spider Monkey Algorithm, Ant Colony Optimization, and Honey Bee Optimization.

Kumar et al. [17] conducted a study concentrating on harnessing cloud computing technologies to enhance resource allocation for sustainability in an organization's cloud infrastructure. They introduced a specialized pricing and allocation mechanism designed for a private cloud computing service, facilitating efficient load balancing of computing resources. The proposed approach implemented an optimal pricing strategy within a dynamic pricing model, focusing on maximizing the net value of users within the private cloud. To achieve load balancing in the cloud, they developed a task allocation algorithm based on this dynamic pricing model. The research findings emphasized the importance of evenly distributing the number of tasks across individual resource servers for optimal task allocation.

In a research study, Kamila et al. [18] introduced and merged artificial intelligence and machine learning methodologies on a cloud platform with the concept of high-performance computing (HPC). The objective was to ensure system performance and continuous traffic flow resilience by utilizing networking and computing performance data for validation, prediction, and classification of traffic and performance patterns. The proposed integrated design method has been evaluated on various real-time instances, employing machine learning regression and classification models to automatically correct the system's performance.

Kumar and Rajesh [19] introduced an innovative multi-objective load balancing architecture aimed at achieving optimal load balancing in the cloud. The proposed paradigm considers several crucial factors, including memory utilization, migration costs, power consumption, bandwidth consumption, and load balancing parameters such as response time, turnaround time, and server load. To achieve optimal load balancing, they devised a unique hybrid optimization technique termed the Mouse Customised Golden Eagle optimization (MCGEO) model. This model represents a conceptual amalgamation of the classic Golden Eagle Optimizer (GEO) and the Cat and Mouse-Based Optimizer (CMBO).

For additional illustration, Table 1 compares several load balancing strategies used in a cloud setting and lists the primary contributions of each study as well as the characteristics that were taken into consideration.

3 Problem description and proposed solution

Task scheduling issues are crucial for the efficiency of all cloud computing infrastructure. In distributed systems, scheduling algorithms aim to distribute the workload across processors to optimize their usage and minimize the overall task processing time.

3.1 Problem Definition

An appropriate VM is assigned in cloud, based on the user's request. Load balancing strategies are employed to carry out this allocation. The scheduling issue becomes a major problem in the cloud environment [12] as the count of cloud service providers grows daily and the load on the cloud servers also increases. Some VMs may be overused while others are underused when tasks are scheduled on them [13]. When tasks are scheduled on VMs, it is possible for some VMs to be overused while others stay underused.

As a result, an effective LB technique is required to balance the workload on the servers by dividing the system's entire load among the connected VMs [14]. By using this technique, each VM is guaranteed to finish approximately the same number of tasks. Response time, throughput, reliability and resource optimization are all improved, and system bottlenecks that might result from an unbalanced load are also avoided. In unbalanced clouds, a two level load balancing architecture model is presented for achieving the best load balancing. Physical Machine (PM) level load balancing is used for the first level, and VM level load balancing is used for the second level. This model abstracts the VM manager and VM monitor. Intra-VM task migration and migration of tasks between VMs are required to carry out the load distribution. Load balancing consists of scheduling and assigning tasks to VMs in accordance with their needs.

3.1.1 Task Scheduling

Task scheduling involves distributing tasks among computing resources for computation. Additionally, the task scheduler must address scenarios where no VM is available to execute a particular task; in such cases, the task should be migrated to another VM for execution.

• **Determining the user task requirements:** The resource requirements of the user tasks that will be scheduled to run on a VM are determined during this phase.

Table 1 Comparative study of load balancing strategies in cloud computing

		Considered Parameters							
Reference	Main contribution	Response time	Overhead	Throughput	Resource Utilization	Makespan	Availability	Scalability	Reliability
[4]	Suggesting a load balancing approach inspired by the foraging behavior of honey bees which considers the priorities of tasks transferred from overloaded VMs	✓	✓	×	×	✓	×	×	×
[5]	Offering a technique based on the resource reservation policy	✓	×	✓	✓	×	×	×	✓
[6]	Reviewing contemporary research in load balancing, proposed a load balancing approach employing a genetic algorithm that aims to determine the globally optimal processor for tasks	✓	✓	✓	×	×	✓	×	×
[7], [8], [9]	Utilizing diverse optimization techniques, including PSO, ABC, and ACO load balancing is resolved	✓	×	✓	×	×	✓	×	×
[10]	Presenting three adaptations of the Ant-Lion Optimizer and Grey Wolf Optimizer as task schedulers with the objective of minimizing makespan	✓	✓	✓	×	✓	✓	×	×
[12]	Presenting a fuzzy-based inverted ant colony optimization scheme for load balancing	✓	✓	×	✓	✓	×	✓	×
[13]	Suggesting an Extended High to Low Load task scheduling heuristic approach to distribute the workload across the available resources	×	✓	×	✓	✓	✓	✓	×
[15]	Proposing a resource management and allocation plan with the goal of optimizing resource allocation in the cloud environment	✓	✓	✓	✓	×	×	×	×
[16]	Highlighting the benefits of the Pigeon-Inspired optimization Algorithm and Harries Hawks optimization to build a load balancing system	✓	✓	✓	✓	✓	✓	×	×
[19]	Presenting a task allocation algorithm based on dynamic pricing model to accomplish load balancing in the cloud	✓	×	✓	✓	✓	✓	×	×
[18]	Providing an integrated design approach that uses machine learning regression and classification models to make performance corrections to the system automatically	✓	✓	✓	✓	✓	×	×	×

- **VM resource details recognition:** This verifies a VM's resource information status. It provides the unallocated resources and the VM's current resource usage. Based on this stage, the status of VM with respect to a threshold can be identified as balanced, overloaded, or under-loaded.
- **Scheduling of tasks:** Once a VM's resource information has been determined, a scheduling algorithm schedules the tasks to the proper resources on the proper VMs.
- **Allocation of resources:** The resources are allotted to run the scheduled tasks.To do this, a resource allocation strategy is being used. Although scheduling is necessary to expedite execution, allocation policy is used to manage resources effectively and boost resource performance.
- **Migration of VMs:** The load balancing process in the cloud is incomplete without migration. To solve the overloading issue, VM migration involves moving a VM from one physical host to another.
- **Task Migration:** Task Migration is a crucial load balancing metric that involves moving active tasks from one physical machine to another with likely different design. Therefore, it is necessary to save the task's current state and convalesce it on the remote host.

3.2 Task Deployment Policy

To reallocate the task that is removed from the overloaded VM, the proposed technique must identify suitable under loaded VMs. In the data centers (DC), there might be more than one VM that can complete the task. Because of this, it is essential to consider the VM that can handle the task. Let's consider the VM's capacity, load, degree of imbalance and load fairness which are essential elements in identifying underloaded VMs. The primary objective is to provide users with the services they require in the shortest response time which is achieved by the suggested CO algorithm by taking into account a number of factors when estimating the VM capacity, including processing speed, bandwidth, memory etc.

3.3 Computational Model

There are some computing resources available at each data center to carry out user tasks. The amount of time it takes for each task to complete can be used to calculate the VM's load. As a result, the processing times for each task vary, changing the load on the VM. For ease of reference, Table 2 provides definitions and descriptions of the notations used.

3.3.1 The Scheduling Optimization Model based on Constraints and Performance

The proposed model consists of a number of data centers(DC), physical machines(PM), virtual machines(VM), etc. Let us assume that CC is a cloud computing system, and P stands for the number of Physical Machines, and that V stands for the number of VMs. In Eq.1, the n number of PMs is shown.

$$CC = \{P_1, P_2, P_k..., P_n\} \qquad 1 < k < n \tag{1}$$

Table 2 Algorithm notations and descriptions

Notations	**Explanation of the notations**
CC	Cloud configuration
DC	Data center
PM	Physical machine
VM	Virtual machine
P	Number of Physical Machines
V	Number of Virtual Machines
P_n	n^{th} number of Physical Machine
P_k	k^{th} number of Physical Machine
m	Total number of virtual machines in the k^{th} PM
T	Set of tasks
t	Total number of tasks
V_x	Set of load balancing parameters of VMs
R_x	Number of processors
$MIPS_x$	Million Instructions Per Second
B_x	Bandwidth
C_x	Cost of migration
M_x	Amount of memory
$VMcap_x$	Virtual Machine capacity
$VMLoad_j$	Load on the j^th Virtual machine
$VMload_{avg}$	Average load of Virtual Machine
RC_{ij}	Resource consumption
$VMload_{avg}$	Average load across all VMs
$VMload_{max}$	Maximum load among all VMs
$VMload_{min}$	Minimum load among all VMs
DI	Degree of imbalance
CU	CPU utilization
CPU_{used}	CPU time used by the VMs
CPU_{total}	Total CPU time available
LF	Load fairness index
PUV	Processing Unit Vector
α	Cost associated with executing a single instruction
L	Delay cost
TUV	Task Unit Vector
T	Type of service
NIC	Number of instructions of task
TAT	Task Arrival Time
ζ	Fitness function
ω_1, ω_2	Weights
φ	Fitness Function
Υ	Multi-objective Fitness Function

where P_n is denoted as the n^{th} number of Physical Machine and P_k is denoted as the k^{th} number of Physical Machine. Each PM is made up of a number of VMs, as demonstrated mathematically in Eq. 2.

$$V = \{V_1, V_2, ..., V_j, .., V_m\} \qquad 1 < j < m \tag{2}$$

here m is the total number of VMs in the k^{th} PM. Additionally, the cloud system supports numerous users, as stated mathematically in Eq.3.

$$T = \{T_1, T_2, ..., T_i, .., T_t\} \tag{3}$$

while T is a set of tasks, t denotes the total number of tasks. The VM is given each task. Tasks are processed in the cloud system without any issues when the VM workload status is normal. When a VM's workload is overloaded, a load balancing technique (LB) is needed to move tasks from overloaded to underloaded VMs. The load balancing parameters that are described in Eq. 4. are what the VMs are dependent on.

$$V_x = \{R_x, MIPS_x, B_x, M_x, C_x\} \tag{4}$$

where the number of processors R_x, $MIPS_x$ stands for Million Instructions Per Second , the variable bandwidth B_x, the amount of memory used M_x and the cost of task migration C_x, are all given. Each task has a different execution duration and priority value. Additionally, tasks are distributed to the VMs based on two important criteria, including (i) tasks with a higher priority and (ii) tasks with the shortest execution times.

- **VM capacity:** The capacity of VM depends on the number of processors, $MIPS$, memory and bandwidth. It is mathematically expressed as VM_{cap} and is indicated in Eq.5. [20] ,

$$VMcap_x = \left[\left(\frac{R_x \times MIPS_x \times Mx}{1000}\right) + B_x\right] \times \frac{1}{2} \tag{5}$$

By adding up the resource usage of all the tasks that have been given to a VM and dividing the result by the VM's capacity (Eq.5), Eq.6 determines the load on the VM.This yields a normalized load value, indicating the extent to which the VM's capacity (Eq.6) is being utilized.

$$VMload_j = \frac{\sum_{i=1}^{t} RC_{ij}}{VM_{cap}} \tag{6}$$

Here, the load on the j^{th} VM is denoted by $VMload_j$ and the resource use of the i^{th} task on the j^{th} VM is indicated by the RC_{ij}.The total number of tasks allocated to the j^{th} VM is t and the j^{th} VM's capacity is denoted by $VMcap_x$.

- **Task processing execution time of a VM:** It is the total of all the tasks' execution times that are currently executing on the virtual machine. Eq.7 can be used to determine the task processing execution time $TaskProTime(t_{ij})$ on v_j of task t_i.

$$TaskProTime(t_{ij}) = \frac{length(t_i)}{ProcessorRate(v_j)} \tag{7}$$

  where the length of the task is expressed in terms of the number of instructions ($millioninstructions$), and $length(t_i)$, is the length of the i_{th} task. The processing speed of the j_{th} VM in the cloud is denoted by $ProcessorRate(v_j)$.

- **Degree of imbalance:** It measures how workload is distributed among VMs based on their execution capabilities. A lower DI value indicates a more balanced load distribution. The formula 8 is used to determine DI:

$$DI = \frac{TaskProTime_{max} - TaskProTime_{min}}{TaskProTime_{avg}} \tag{8}$$

$TaskProTime_{max}$, $TaskProTime_{min}$, and $TaskProTime_{avg}$ refer to the highest, lowest, and average completion times, respectively, for the different virtual machines. Let, Eq.9 denotes as a set of task processing execution time of m VMs.

$$TaskProTime = \{TaskProTime_1, TaskProTime_2, ..., TaskProTime_m\} \tag{9}$$

The numerator of Eq.8 calculates the maximum completion time difference between m heterogeneous VMs which can be expressed using Eq.10,

$$TaskProTime(dif)_{max} = TaskProTime_{max} - TaskProTime_{min} \tag{10}$$

where $TaskProTime_{max} = \max\{TaskProTime_1, TaskProTime_2, ..., TaskProTime_m\}$ and $TaskProTime_{min} = \min\{TaskProTime_1, TaskProTime_2, ..., TaskProTime_m\}$ denote maximum completion time, minimum completion time, respectively. Thus, we can deduce that:

– If $TaskProTime(dif)_{max}$ is extremely high, then $TaskProTime(dif)_{max}$ is likewise extremely high, and some virtual machines are overloaded and underloaded.

- **Load fairness :** Determining load fairness usually entails determining how equitably the workloads or computational tasks are allocated among various node in cloud. We have used the formula to calculate load fairness based on coefficient of variation (CV).To estimate load fairness the following steps are performed:
  Determine the Average Load : Eq.11 finds out the average load across all VMs where m is the total number of VMs.

$$VMload_{avg} = \frac{\sum_{j=1}^{m} VMload_j}{m} \tag{11}$$

Compute Standard Deviation(σ): To calculate the dispersion, the loads' standard deviation is estimated using Eq.12,

$$\sigma = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(VMload_j - VMload_{avg})^2} \tag{12}$$

Determine the Coefficient of Variation (CV): Using Eq.13 is determined,

$$CV = \frac{\sigma}{VMload_{avg}} \tag{13}$$

Load fairness index has a range of 0 to 1, where 1 denotes maximum fairness (distribution of loads equally) and a value of 0 denotes total unfairness (one resource is overutilized).For example, a high degree of load fairness among the resources is indicated by a load fairness index of 0.8 which is close to 1.

3.3.2 Calculation of Fitness Function

The Proposed algorithm for task scheduling in cloud computing leverages the principles of multi-objective scheduling methods.The suggested approach uses two objective functions— Eq.16 and Eq.19 —to calculate the appropriateness requirements.Using this method, a multi-objective optimization problem is reduced to a single objective with weights that represent the decision maker's preferences among the objectives.

1. Despite the fact that cloud computing is dynamic, the load balancing issue mentioned earlier can be solved by allocating N tasks submitted by cloud users to M cloud processing units at any given instance. For each processing unit, a processing unit vector (PUV) indicating the processing unit utilisation status will be present. The components of this vector are $MIPS$,α indicates expense associated with executing a single instruction and L, which stands for Delay Cost. The delay cost is an estimation of the fine that a cloud service provider (CSP) must impose on a client if the task takes longer than the CSP's allotted timeframe. PUV is referred to as Eq. 14,
$$PUV = \Psi(MIPS, \alpha, L) \tag{14}$$
Similar to above, each task that a cloud user submits can be depicted by a task unit vector (TUV). The worst-case completion time (ωc) represents the minimal duration necessary for a processing unit to finish the task. Therefore, Eq. 15 can be used to represent the attribute of various tasks.
$$TUV = \Phi(T, NIC, TAT, \omega c) \tag{15}$$
where T is the type of service, such as Platform-as-a-Service ($PAAS$), Infrastructure as a Service ($IAAS$), or Software as a Service ($SAAS$), that is required to complete the task ($PAAS$). The NIC stands for the task's number of instructions. The processor decides how many instructions are needed to complete the task. As measured by the wall clock, the task's arrival time (TAT) is the time it first entered the system. The fitness function ζ must be minimised by the cloud service provider when distributing these N tasks among M processors, as shown in Eq. 16.
$$\zeta = \omega_1 \times \alpha(NIC \div MIPS) + \omega_2 \times L \tag{16}$$
where weights ω_1 and ω_2 are predetermined. Choosing or maximising the weights can be challenging; one criterion might be that the weight should increase the more general the factor is. The preference or weight users give to one factor over another is another example of logic. The latter strategy has been used in this instance, and the optimization process is then carried out using the provided set of weights. When the weights are considered, $\omega_1 = 0.8$ and $\omega_2 = 0.2$, the sum equals 1.

2. The second objective is specified in terms of makespan, the longest time a virtual machine can take to complete all of its input tasks.The makespan determines

the task's execution time, which varies for each VM. When using multi-objective scheduling techniques, the makespan is a helpful factor that can shorten task execution times and enable early task completion. In the event that both the maximum execution time value and the makespan value are large, the system is deemed to have inadequately dispersed tasks among the VMs. On the other hand, the makespan value is also low if the maximum execution time value is low. MakespanMax Eq. 17, which can be computed using Eq.7, is the maximum value of all VM's execution duration.

$$MakespanMax = Max(TaskProTime(v_j)), \qquad 1 \leq i \leq m \tag{17}$$

The lower bound of makespan, or the least amount of time the system needs to finish all tasks, is called MakespanMin, Eq. 18 is used to compute MakespanMin.

$$MakespanMin = Min(TaskProTime(v_j)), \qquad 1 \leq i \leq m \tag{18}$$

Fitness function in terms of makespan (φ) can be calculated by Eq.19.

$$\varphi = \frac{MakespanMin}{MakespanMax} \tag{19}$$

The weighted average of each individual fitness function is used to produce the fitness function. Eq. 20 displays the suggested fitness function Υ.

$$\Upsilon = (\phi_1 \times \zeta) + (\phi_2 \times \varphi) \tag{20}$$

where ϕ_1, ϕ_2 and $\phi \in [0, 1]$ are the balance coefficients. A better solution is obtained by maximising the fitness function Υ.

3.3.3 Task Migration in proposed model

The task scheduler should trigger load balancing when it determines that balancing the load is necessary. Load balancing involves identifying overloaded VMs, understanding the demand (load requirements), locating underloaded VMs, and assessing the available supply (offered load). To queue removed tasks from overloaded VMs effectively, it is important to first determine their priority and then choose the best VM for them. Tasks that were previously removed from overloaded VMs (host's eggs) also assist in choosing the optimal low-loaded VM (cuckoo's eggs), with Lévy flights proving to be more effective for search tasks than a straightforward random walk.After completing the load balancing task, the cuckoo searches for the best nest with high-quality eggs, which are similar to the host bird's eggs, for the next task. Consequently, high-priority tasks will be distributed to the machine with the least number of existing high-priority tasks.When a high-priority task is submitted to an underloaded machine, it must be evaluated along with all other high-priority tasks that have already been allocated to the VMs. Table 3 illustrates how the cloud environment was created in imitation of cuckoo birds, which used to deposit their eggs in other birds' nests. The proposed

Table 3 Cuckoo behavior for task migration in cloud environment

Cuckoo Search	**Cloud Computing System**
Host Nests	Cloudlets(Data centers)
Host's egg	Virtual Machine
Cuckoo searches Host Nests with more similar host bird eggs(High quality eggs)	Mapping Cloudlets to VM
Cuckoo getting worn out by searching host nest with high quality eggs	VM is overloaded
Cuckoo searches for a new host nest	Transferring the task from a heavily loaded VM to a lightly loaded VM

Cuckoo Optimizer is used to examine the virtual machines (VMs) and determine which ones would be optimal for taking on more work. To guarantee optimal performance, this entails assessing the capacity and present load of each virtual machine using Eq.5 and Eq.6 respectively.

3.3.4 Task Scheduling using proposed technique in a Dynamic Cloud Environment

Task scheduling algorithms need to be extremely flexible in order to adjust to changing conditions in a dynamic cloud environment. Adaptive methods and real-time modifications allow metaheuristic algorithms, such as Cuckoo Optimization, to be customized to tackle dynamic workloads.The proposed algorithm includes a task prioritization mechanism that ranks tasks based on urgency, deadlines, and resource requirements. Cuckoo Optimization is utilized for its ability to efficiently explore and exploit solutions, making it well-suited for dynamic environments by adjusting parameters like step size and discovery rates. Resource availability is continuously monitored, and scheduling decisions are made in real-time according to the current state of the environment. To manage task scheduling effectively in a dynamic cloud environment, the task schedule is periodically reevaluated to account for newly arrived tasks and updated resource information.

3.4 Cuckoo Optimization Algorithm

The CO algorithm employs Lévy flights, enabling extensive, long-range movements which enhances the exploration of the search space and aids in avoiding local minima.By combining global search capability with local refinement, the CO algorithm typically converges more quickly to optimal solutions than Genetic Algorithms, Particle Swarm Optimization etc. This rapid convergence is especially advantageous in dynamic cloud environments where swift decision-making is crucial.The CO algorithm can adeptly manage both hard and soft constraints in optimization problems, which is essential in cloud computing to meet resource constraints and quality of service requirements. For a variety of cloud computing optimization issues, the proposed method outperforms other metaheuristic algorithms due to its strengths in balancing exploration and exploitation, quick convergence, simplicity, resilience, and effective management of constraints. The aggressive reproductive strategy cuckoo birds used to lay their eggs in other birds' nests is the foundation of this metaheuristic. The creation of the Cuckoo Optimization algorithm, which has a population of eggs or nests, was motivated by the unique way of life and reproduction strategy. The new solution describes the cuckoo eggs, whereas the solutions represent each egg in the nest. If the host egg is very similar to the cuckoo egg, the fitness function for the host egg needs to be calculated in the solutions. The inferior solution is swapped out for a superior one in the nest (cuckoo).

The following rules are part of the Cuckoo Optimization algorithm:

- A cuckoo can only lay one egg at a time, and it can be dropped into any nest.
- The best nest, which will be passed down to the following generation, is one with high-quality eggs.
- Since there are a fixed number of host nests, the probability of the host species is used to predict the egg that a cuckoo would lay. The host species will either destroy the egg in this situation or leave the nest.
- Take into account $P[0, 1]$ as the likelihood that an alien egg will be found in a host bird's nest.

For generating a new solution $X(\tau + 1)$, a levy flight is performed (Eq.21),

$$X(\tau + 1) = X(\tau) + \rho \times \tau^{-\gamma} \tag{21}$$

Where $\rho > 0$ represents the step size and is related to the size of the problem under consideration and $(1 < \gamma \leq 3)$.

The proposed algorithm 1 is as given below:

Algorithm 1 Cuckoo Optimization Algorithm

```
Require: n represents the population size
Require: Input: List of VMs, List of Tasks
    Step 1: Randomly initialize a set of population of n hosts (processing units). [Start].
    Step 2: Using Eq.20 [Fitness], determine the fitness value of each solution.
    Step 3: Take action while either the maximum number of iterations is reached or the best solution is identified.
    Step 3(a): Choose a cuckoo at random, let's say x, and create a new solution using levy flights.
    Step 3(b): Evaluate fitness using $\Upsilon$.
    Step 3(c): Pick one nest (let's say y) at random from n nest.
  if ($\Upsilon$(x) $\geq$ $\Upsilon$(y)) then Replace y by the new solution x
  end if
    Step 3(e): Discard or abandon a fraction($P_m$) of worse nests.
    Step 3(f): Create fresh solutions at random using cuckoos(Build a new ones at new locations via Levy flight (Eq.21) .
    Step 3(g): Evaluate the fitness.
    Step 3(h): Keep the better solutions.
    Step 3(i): Test the end condition [Test].
    Step 4: Compare the solutions to find the one that is currently the best (optimal VM).
    Step 5: End.
```

3.5 Complexity of the Algorithm

Any algorithm's complexity study comprises both spatial and computational complexity analyses (time and space complexity analyses).The algorithm's complexity is determined by taking into account the search dimension factor d and a population of size n.Each population member requires $O(d)$ operations, leading to a complexity of $O(n \times d)$ per iteration.The complexity shown above is for a single iteration; nevertheless, CO often executes over a number of iterations. Hence, the maximum number of iterations m_{max}, determines the overall complexity. Therefore, the total time complexity of the algorithm G is Eq.22,

$$G = O(n \times d \times m_{max}) \tag{22}$$

The population size n and search dimension d have the greatest effects on space complexity, so the algorithm's space complexity is $O(n \times d)$. The table 4 shows the

4 Simulation Results and Analysis

In CloudAnalyst [21], the proposed algorithm is simulated using the case of a social media network like Facebook.

Existing Algorithms used in Task scheduling	**Complexity**	**Parameters**
Genetic Algorithm(GA)	$O(g \times d \times n)$	g=Number of generation, d=Dimension of the Problem(No. of task and VMs), n=population size
Ant Colony Optimization(ACO)	$O(g \times d \times n^2)$	g=Number of iteration, d=Dimension of the Problem, n= Number of ants
Particle Swarm Optimization(PSO)	$O(g \times d \times n)$	g=Number of iteration, d=Dimension of the Problem, n= Number of particles
Artificial Bee Colony(ABC)	$O(g \times d \times n)$	g=Number of iteration, d=Dimension of the Problem, n= Number of bees

Table 4 Comparison of Time Complexities of existing Meta-heuristic Task Scheduling Algorithms

4.1 Simulation Setup

Table 5 shows a hypothetical configuration used for experimentation which divides the world into six "Regions," which are just the six continents. Each user base (UB) has a single time zone assigned to it, and for each UB, a representative online user during peak and off-peak hours has been considered. During non-peak hours, only about one-tenth of all online users are available. The application allocates a certain number

Sl. No.	**User Base**	**Region**	**Online users during peak hours**	**Online users during off-peak hours**
1.	UB1	N.America	4,70,000	80,000
2.	UB2	S.America	6,00,000	1,10,000
3.	UB3	Europe	3,50,000	65,000
4.	UB4	Asia	8,00,000	1,25,000
5.	UB5	Africa	1,25,000	12,000
6.	UB6	Oceania	1,50,000	30,500

Table 5 Configuration of Simulation Environment

of VMs to each simulated "data center host". Each device has 4 GB of RAM, 100 GB of storage, 4 CPUs with a combined processing power of 10,000 MIPS, and 100 GB of total storage. VM monitors Xen, Linux, and x86 architecture are all features of simulated hosts. It has been estimated that each user request (task) calls for the execution of 100 instructions. The proposed algorithm is tested based on a number of configurations listed in Table 6. In each Cloud Configuration, one DC is assumed to have initially 25, 50, and 75 VMs (CC). Two, three, four, five, or six DCs are each considered in Tables 7, 8, 9, 10 and 11, with a combination of 25, 50, and 75 VMs for each DC, respectively. The tasks' average response times are calculated and tabulated for the proposed algorithm. The effectiveness of the suggested algorithm is evaluated in comparison to other existing meta-heuristic algorithms used for task scheduling , including Particle Swarm Optimization(PSO), Ant Colony Optimization(ACO), Genetic Algorithm(GA) and Stochastic Hill Climbing(SHC). Figures 1, 2, 3, 4, 5, 6 and 7 show a comparison of the proposed technique with various scenarios and methodologies. The uniqueness of the work is supported by the comparative analysis.

4.2 Experimental Results

In this study the proposed Cuckoo Optimizer for task scheduling takes into account the VMs' bandwidth, processors' processing power, and available memory as the number of VMs changes, the completion time of tasks is reduced.

Sl. No.	CC	DC specification	RT in ms For CO	RT in ms For PSO	RT in ms For ACO	RT in ms For GA	RT in ms for SHC
1.	CC1	One DC with 25 VMs	327.88	328.02	328.43	328.75	329.02
2.	CC2	One DC with 50 VMs	327.30	327.96	328.87	328.42	329
3.	CC3	One DC with 75 VMs	325	325.97	326.84	327.04	329.34

Table 6 Simulation scenario and calculated overall average response time (RT) using One DC

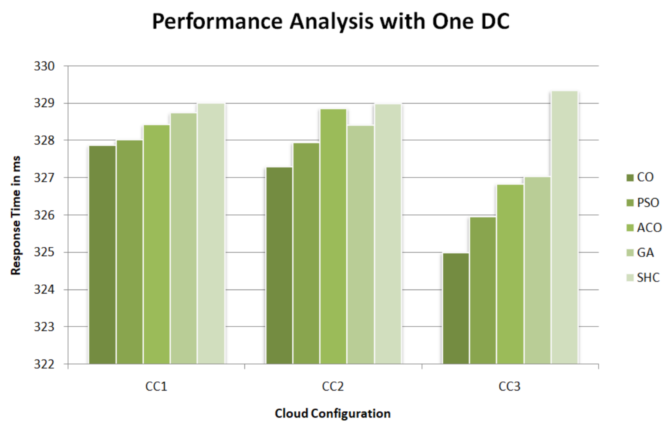

Fig. 1 Performance analysis of proposed CO with PSO, ACO, GA and SHC Result using One DC

4.2.1 Response Time

Shorter response time (RT) is achieved using the suggested technique, as shown in Tables 6, 7, 8, 9, 10 and 11. Additionally, the Cuckoo Optimization algorithm outperforms the existing metaheuristics like PSO, ACO, GA and SHC.

Reduce Response Time with high bandwidth utilization: When the number of VMs grows, the processing load is concentrated on a physical machine in suboptimal systems, increasing the load on it and the likelihood of overload.

Convergence time to the optimal set of solutions: For a method that can reach its optimal solution more quickly, there is no processing overhead incurred by the scheduling machine. In CO, the time required to find the best solution is linear, and as more VMs are added, the growth rate of this time decreases. In GA, the longer it takes to reach the best solution, the larger the problem dimensions are.

Sl. No.	CC	DC specification	RT in ms For CO	RT in ms for PSO	RT in ms For ACO	RT in ms For GA	RT in ms for SHC
1.	CC1	Two DC with 25 VMs each	359.30	359.93	360.12	360.77	365.44
2.	CC2	Two DC with 50 VMs each	354.44	354.97	355.43	355.72	360.15
3.	CC3	Two DC with 75 VMs each	354.10	354.65	354.92	355.32	359.73
4.	CC4	Two DC with 25,50 VMs	349.05	349.85	350.23	350.58	356.72
5.	CC5	Two DC with 25,75 VMs	350.04	350.71	350.86	351.56	357.23
6.	CC6	Two DC with 50,75 VMs	350.06	350.83	351.43	352.01	357.04

Table 7 Simulation scenario and calculated overall average response time(RT) using Two DCs

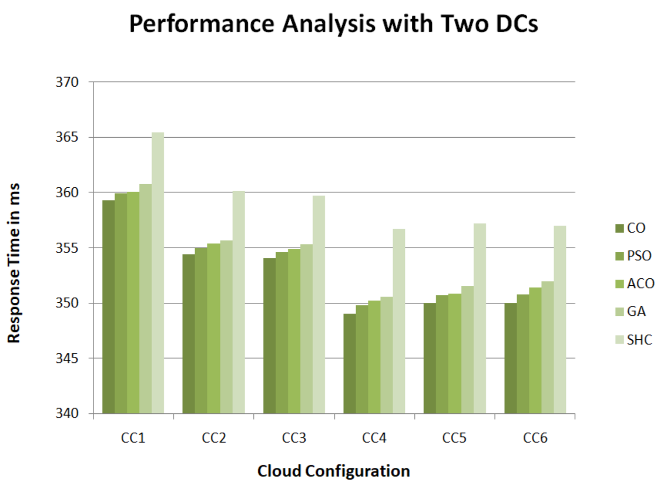

Fig. 2 Performance analysis of proposed CO with PSO, ACO, GA and SHC Result using Two DCs

Sl. No.	CC	DC specification	RT in ms For CO	RT in ms For PSO	RT in ms For ACO	RT in ms For GA	RT in ms for SHC
1.	CC1	DC with 25 VMs each	348.10	349.11	349.97	350.32	356.82
2.	CC2	DC with 50 VMs each	347.15	349.34	349.92	350.19	355.25
3.	CC3	DC with 75 VMs each	344.05	345.21	345.67	346.01	350.73
4.	CC4	DC with 25,50,75 VMs	341.95	343.82	344.65	345.98	350.01

Table 8 Simulation scenario and calculated overall average response time(RT) using Three DCs

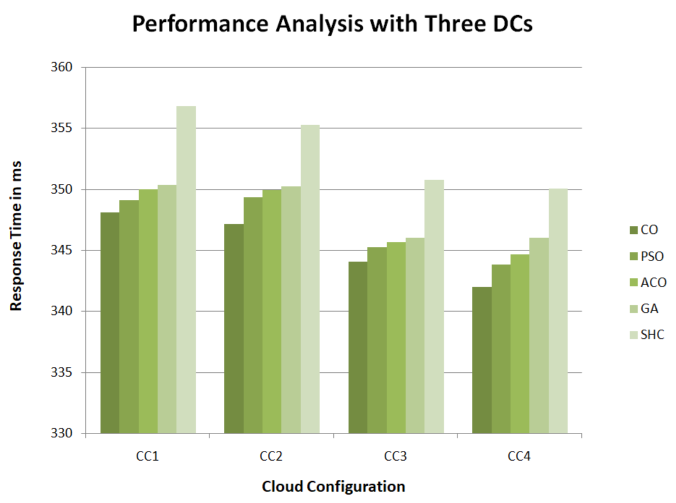

Fig. 3 Performance analysis of proposed CO with PSO, ACO, GA and SHC Result using Three DCs

Sl. No.	CC	DC specification	RT in ms For CO	RT in ms For POS	RT in ms For ACO	RT in ms For GA	RT in ms for SHC
1.	CC1	DC with 25 VMs each	344.23	346.43	347.77	348.85	354.35
2.	CC2	DC with 50 VMs each	341.08	343.56	344.89	345.54	350.71
3.	CC3	DC with 75 VMs each	336.11	338.02	338.93	340.65	346.46
4.	CC4	DC with 25,50,75VMs	332.76	334.61	336.23	337.88	344.31

Table 9 Simulation scenario and calculated overall average response time (RT) in (ms) using Four DC

Sl. No.	CC	DC specification	RT in ms For CO	RT in ms PSO	RT in ms For ACO	RT in ms For GA	RT in ms for SHC
1.	CC1	DC with 25 VMs each	332.45	334.01	334.98	335.64	342.86
2.	CC2	DC with 50 VMs each	324.30	324.93	325.87	326.02	332.84
3.	CC3	DC with 75 VMs each	319.54	320.79	321.86	322.93	329.46
4.	CC4	DC with 25,50,75VMs	316.35	317.59	318.66	319.98	326.64

Table 10 Simulation scenario and calculated overall average response time(RT) using Five DCs

4.2.2 Average Processing Time

Processing time(PT) is the amount of time a DC needs for processing a user-generated request. Comparative analysis reveals that proposed technique responds to requests assigned to it more quickly than existing mechanisms. The average PT taken by DC1,

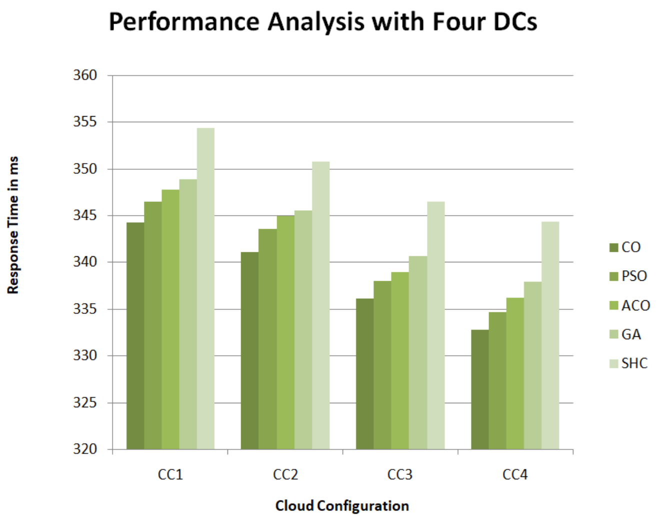

Fig. 4 Performance analysis of proposed CO with PSO, ACO, GA and SHC Result using Four DCs

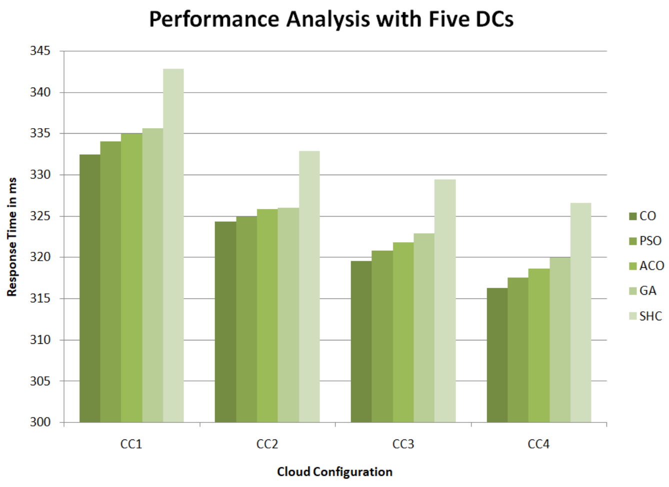

Fig. 5 Performance analysis of proposed CO with PSO, ACO, GA and SHC Result using Five DCs

DC2, DC3 and DC4 using CO is less than PSO, ACO, GA and SHC. Figure 7 compares the average PT for each data center for CO, PSO, ACO, GA and SHC.

4.2.3 Degree of Imbalance

Fig. 8 illustrates the degree of imbalance (DOI) among VMs both prior to and following load balancing using proposed algorithm. In this figure, the X-axis denotes the number of cloudlets, while the Y-axis indicates the degree of imbalance. The results demonstrate that the proposed algorithm effectively distributes the load among VMs, significantly reducing the imbalance.

Sl. No.	CC	DC specification	RT in ms For CO	RT in ms For PSO	RT in ms For ACO	RT in ms For GA	RT in ms for SHC
1.	CC1	DC with 25 VMs each	326.32	328.02	328.78	332.54	336.96
2.	CC2	DC with 50 VMs each	320.24	322.04	322.83	323.01	331.56
3.	CC3	DC with 75 VMs each	315.09	317.32	319.48	321.54	327.78
4.	CC4	DC with 25,50,75VMs	309.35	311.6	314.02	315.33	323.56

Table 11 Simulation scenario and calculated overall average response time (RT) using Six DCs

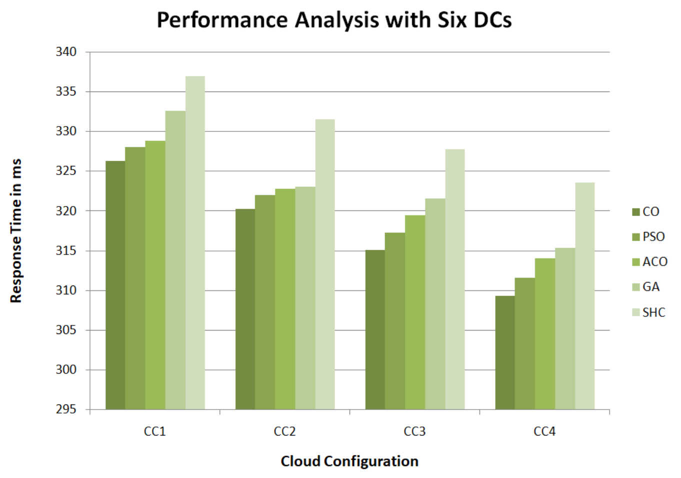

Fig. 6 Performance analysis of proposed CO with PSO, ACO, GA and SHC Result using Six DCs

4.2.4 Load fairness

A significant improvement of load fairness, defined in sub-section 3.3.1, is observed using the proposed technique. The comparison of load fairness is shown in Figure 9, where the load fairness is represented by the Y-axis and the X-axis indicates CO, PSO, ACO, GA, and SHC.

5 Conclusion and future work

In particular, task scheduling is an NP-hard problem, making finding an optimal solution either impossible or impractical. Heuristics are crucial to speed up the process of locating a workable solution. In this paper, a metaheuristic based on Cuckoo Optimization (CO) is introduced for task scheduling to balance workload in a cloud computing architecture. According to the findings of a thorough analysis, the recommended task scheduling strategy not only outperforms the current optimization algorithms, but also ensures that the QoS prerequisites of the customer tasks are met. Although in this case fault tolerance issues and lowering energy consumption are not taken into account.

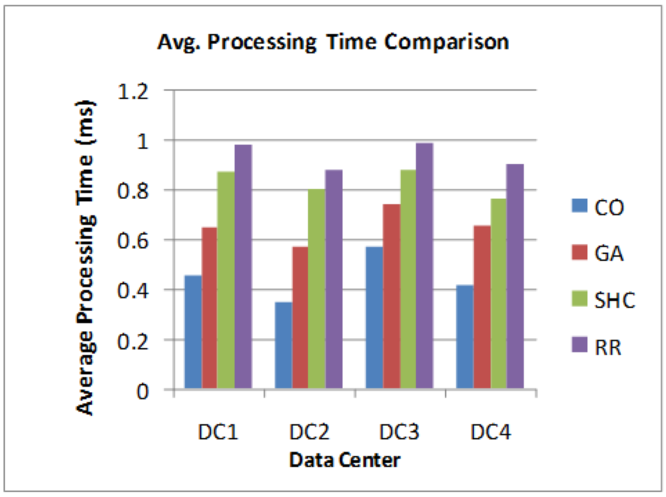

Fig. 7 Average Processing Time Comparison

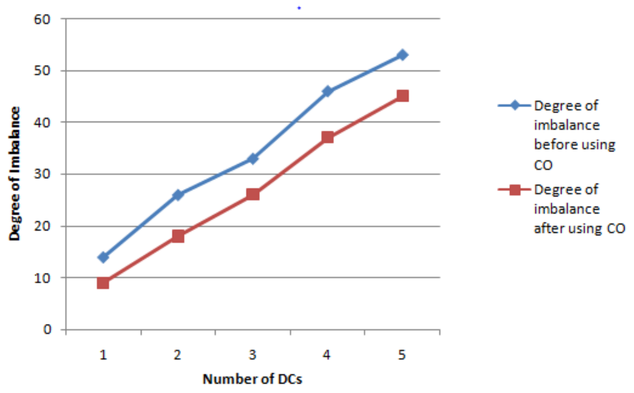

Fig. 8 Measurement of Degree of Imbalance before and after using CO

The pheromone value can still be calculated while taking into account fault tolerance and different function variations. Also as future work, an adaptive CO algorithms can be developed which is capable of learning and adapting to shifting workloads and resource availability over time, thereby enhancing overall scheduling efficiency.

References

[1] Weiss, A.: Computing in the clouds. networker **11**(4), 16–25 (2007)

[2] Somula, R., Sasikala, R.: A honey bee inspired cloudlet selection for resource allocation. In: Smart Intelligent Computing and Applications: Proceedings of the Second International Conference on SCI 2018, Volume 2, pp. 335–343 (2019). Springer

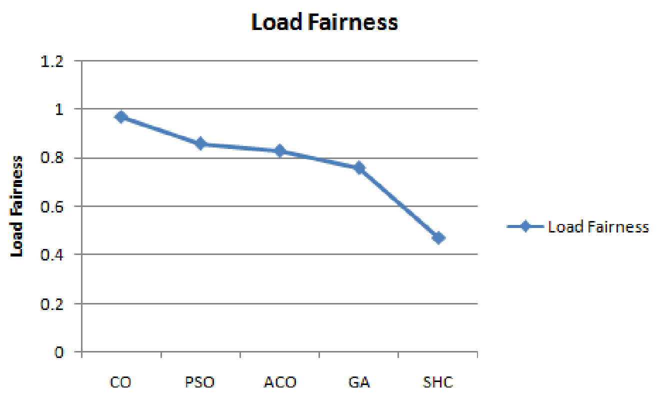

Fig. 9 Comparision of load fairness using CO, PSO, ACO, GA and SHC

[3] Al Nuaimi, K., Mohamed, N., Al Nuaimi, M., Al-Jaroodi, J.: A survey of load balancing in cloud computing: Challenges and algorithms. In: 2012 Second Symposium on Network Cloud Computing and Applications, pp. 137–142 (2012). IEEE

[4] LD, D.B., Krishna, P.V.: Honey bee behavior inspired load balancing of tasks in cloud computing environments. Applied soft computing **13**(5), 2292–2303 (2013)

[5] Nakai, A., Madeira, E., Buzato, L.E.: On the use of resource reservation for web services load balancing. Journal of network and systems management **23**, 502–538 (2015)

[6] Dasgupta, K., Mandal, B., Dutta, P., Mandal, J.K., Dam, S.: A genetic algorithm (ga) based load balancing strategy for cloud computing. Procedia Technology **10**, 340–347 (2013)

[7] Wang, Q., Fu, X.-L., Dong, G.-F., Li, T.: Research on cloud computing task scheduling algorithm based on particle swarm optimization. Journal of Computational Methods in Sciences and Engineering **19**(2), 327–335 (2019)

[8] Kruekaew, B., Kimpan, W.: Virtual machine scheduling management on cloud computing using artificial bee colony. In: Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. 1, pp. 12–14 (2014)

[9] Azad, P., Navimipour, N.J.: An energy-aware task scheduling in the cloud computing using a hybrid cultural and ant colony optimization algorithm. International Journal of Cloud Applications and Computing (IJCAC) **7**(4), 20–40 (2017)

[10] Farrag, A.A.S., Mohamad, S.A., El Sayed, M.: Swarm intelligent algorithms for solving load balancing in cloud computing. Egyptian Computer Science Journal **43**(1), 45–57 (2019)

[11] Madni, S.H.H., Latiff, M.S.A., Coulibaly, Y., Abdulhamid, S.M.: Recent advancements in resource allocation techniques for cloud computing environment: a systematic review. cluster computing **20**, 2489–2533 (2017)

[12] Azad, P., Navimipour, N.J., Hosseinzadeh, M.: A fuzzy-based method for task scheduling in the cloud environments using inverted ant colony optimisation algorithm. International Journal of Bio-Inspired Computation **14**(2), 125–137 (2019)

[13] Zaman, S.K., Maqsood, T., Ali, M., Bilal, K., Madani, S.A., Khan, A.: A load balanced task scheduling heuristic for large-scale computing systems. Comput. Syst. Sci. Eng **34**, 4 (2019)

[14] Mishra, S.K., Sahoo, B., Parida, P.P.: Load balancing in cloud computing: a big picture. Journal of King Saud University-Computer and Information Sciences **32**(2), 149–158 (2020)

[15] Balaji, K., Sai Kiran, P.: Efficient resource allocation algorithm with optimal throughput in cloud computing. Journal of Advanced Research in Dynamical and Control Systems **9**, 1902–1910 (2017)

[16] Annie Poornima Princess, G., Radhamani, A.: A hybrid meta-heuristic for optimal load balancing in cloud computing. Journal of grid computing **19**(2), 21 (2021)

[17] Kumar, C., Marston, S., Sen, R., Narisetty, A.: Greening the cloud: a load balancing mechanism to optimize cloud computing networks. Journal of Management Information Systems **39**(2), 513–541 (2022)

[18] Kamila, N.K., Frnda, J., Pani, S.K., Das, R., Islam, S.M., Bharti, P., Muduli, K.: Machine learning model design for high performance cloud computing & load balancing resiliency: An innovative approach. Journal of King Saud University-Computer and Information Sciences **34**(10), 9991–10009 (2022)

[19] Kumar, K.V., Rajesh, A.: Multi-objective load balancing in cloud computing: A meta-heuristic approach. Cybernetics and Systems **54**(8), 1466–1493 (2023)

[20] Kumar, K.P., Ragunathan, T., Vasumathi, D., Prasad, P.K., *et al.*: An efficient load balancing technique based on cuckoo search and firefly algorithm in cloud. Algorithms **423**, 422–432 (2020)

[21] Wickremasinghe, B., Calheiros, R.N., Buyya, R.: Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. In: 2010 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 446–452 (2010). IEEE