RESEARCH PAPER

Optimizing Cloud Resource Scheduling: A Comparative Study of ACO, PSO, GA, and Hybrid Approaches in CloudSim

Ms. Roopali Gupta[1]*, Dr. Om Prakash Sharma[2]

1*Research Scholar, Faculty of Engineering and Technology, Jagan Nath University, Jaipur, Rajasthan, India
roopaliakshaygupta@gmail.com
2 Professor, Faculty of Engineering and Technology, Jagan Nath University, Jaipur, Rajasthan, India
opsmnit@gmail.com

Abstract:
In this paper different scheduling patterns for the cloud computing environment are introduced and implemented in the Java programming language using the CloudSim simulation framework. The presented work objectively analyses seven scheduling approaches that include Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Round robin (RR), Shortest Job First (SJF), the Simulated Annealing – Particle Swarm Optimization (SAPSO) and Strength Pareto Adaptive Weighted PSO (SAWPSO). Each algorithm was implemented to schedule cloudlets (tasks) onto virtual machines (VMs) whilst optimizing the critical performance metrics which include execution time, load balancing, and overall makespan. The process involved initializing datacenters, brokers, VMs, and cloudlets to configure the simulation framework and generating communication and execution matrices to mimic realistic cloud interactions. Results showed that ACO, PSO, GA, SAPSO, and SAWPSO provided efficient task scheduling with a low makespan and balanced resource utilization, while RR achieved very high execution times, due to the nature of their established scheduling policies. Through this thorough examination, the trade-off is established between simplicity and optimality and sheds light on the extent to which these advanced optimization techniques can improve scheduling in cloud settings in an important and potentially highly dynamic context-based manner.

Keywords: Virtual Machine, Algorithm, Round robin, shortest job first, Simulated Annealing – Particle Swarm Optimization (SAPSO), and Strength Pareto Adaptive Weighted PSO (SAWPSO)
How to cite this article: Gupta R, Sharma OP. Optimizing Cloud Resource Scheduling: A Comparative Study of ACO, PSO, GA, and Hybrid Approaches in CloudSim. Int J Drug Deliv Technol. 2026;16(52s): 42-48. DOI: 10.25258/ijddt.16.52s.6

I. Introduction

In contemporary cloud computing architecture, effectively provisioning tasks to coalesce resources is vital [1]. Cloud service providers need to handle highly dynamic workloads and heterogeneous resources [2], so the scheduling algorithms are crucial in providing optimal performance, high resource utilization and load balancing [3]. Recently, simulation frameworks like CloudSim have provided powerful tools for modelling and evaluating multiple scheduling strategies prior to actual implementation in real systems [4]. One of these strategies is the Ant Colony Optimization (ACO) Scheduler, solving the problem of mapping cloudlets (tasks) to virtual machines (VMs) by using the natural behaviour of ants searching for food [3].

1.1 Role of Scheduling in Cloud Computing

The problem of scheduling tasks in any cloud is a fundamentally complex one [1]. A cloud workload could consist of a combination of compute-intensive, memory-bound, or I/O-bound workloads that need to run on a pool of abstracted resources [5]. It is not only the task allocation for the VM, but also the order of execution will establish the execution time (i.e.makespan) and the overall wastage of resources [6]. In many projects' performance metrics like response time, throughput, and energy consumption have also been taken into consideration in the design and evaluation of scheduling algorithms [3,7]. In order to tackle such multidimensional challenges, several heuristic and metaheuristic methodologies attracted the attention of the scientific community in the past [8]. One of these heuristics that has gained attention is Ant Colony Optimization (ACO), which due to its distributed behaviour, flexibility and efficiency, has been found useful for many optimization problems [3].

1.2 Overview of Ant Colony Optimization (ACO)

The Ant Colony Optimization (ACO) operates through metaheuristics over populations; it mimics the behaviour of ants searching for food [3]. In nature, ants release pheromones along the paths they take to food, and over time those pheromones help others follow the most efficient routes [8]. In the scheduling domain, "artificial ants" traverse the space of potential task-to-VM assignments [3]. Each ant generates a candidate solution by selecting the routes (i.e. scheduling decisions) probabilistically based on the quality of previous solutions and pheromone intensities [8]. In the process of iterations in ACO algorithm, pheromone values are updated in accordance with the quality of solutions found, reinforcing better mappings for future iterations toward their solutions [3]. The strength of this algorithm is found in its parallelism and ability to get out of local optimal by diversifying search space [8]. Thus, for the balance between multiple objectives can be considered using ACO if it is scaled up in some kind of simulation environment like CloudSim where it will

*Author for Correspondent; roopaliakshaygupta@gmail.com

minimize the makespan of the workload and maximize the resource utilization to be there and at the same time minimizing the communication overhead between the link of the cloudlet and the data centers [1].

II. Related Reviews

Table 1 shows a comparison of research studies from 2020 to 2025. These studies are about cloud computing. They focus on task scheduling, load balancing, service selection and managing resources in an energy-efficient way.

Most studies used a combination of optimization techniques and Multi-Criteria Decision-Making (MCDM) methods. Examples include TOPSIS, BWM, FAHP, FTOPSIS and ELECTRE. Some studies also used machine learning. The goal was to improve Quality of Service (QoS) throughput and resource utilization.

The studies found that smart and hybrid scheduling algorithms work better. They reduce the time it takes to complete tasks (energy consumption, latency and operational cost). At the same time, they improve availability, scalability and resource utilization. Many studies noted that machine learning, fuzzy logic and adaptive optimization are often used. These techniques help manage cloud resources.

Current research trends focus on scheduling that saves energy, handling faults, balancing loads across geographic locations and using AI for optimization. The goal is to create cloud environments for the future. Cloud computing and fog computing are areas of research. Task scheduling and load balancing are important in cloud computing.

Table 1: Literature Review of Load Balancing and Task Scheduling Approaches in Cloud Computing

Author	Research Area	Keywords	Methodology	Tools Used	Findings
Khorsand, R. & Ramezanpour, M. (2020) [3]	Cloud Task Scheduling	Energy-efficient scheduling, BWM, TOPSIS	BWM, TOPSIS, ANOVA	CloudSim	Reduced makespan, energy use; improved QoS compliance
Hosseinzadeh et al. (2020) [9]	Cloud Service Selection	MCDM, service evaluation	Taxonomy, literature survey	QoS datasets	Outlined MCDM usage in service ranking
Rafieyan et al. (2020) [10]	Adaptive Scheduling	QoS-based dynamic scheduling	BWM + VIKOR	Simulation tests	Improved throughput, makespan, cost efficiency
Shahid et al. (2020) [2]	Fault-Tolerant Load Balancing	Review + fault-tolerant LB proposal	Review + LPPA algorithm	Various algorithms	Emphasized FT inclusion in LB models
Negi et al. (2021) [5]	Hybrid Load Balancing	ML + soft computing scheduling	ANN, BOEK-means, TOPSIS-PSO, IT2FS	Benchmarks	31-71% improved completion & resource utilization
Alruwaili et al. (2021) [11]	Scalable Load Balancing	Cloud LB optimization	Algorithmic design	Simulation	Load balancing strategy proposal
Kumar et al. (2021) [12]	Service Selection Framework	CCS-OSSR (BWM + TOPSIS)	Hybrid MCDM	Sensitivity analysis	Stable, reduced-comparison selection model
Shekhar & Sharvani (2021) [13]	Multi-tenant LB	MTLBP model	Behavioral analysis	Traffic simulation	Enhanced resource consumption and throughput
Moori et al. (2022) [6]	Task Prioritization	AHP-TOPSIS + OPSO	Hybrid MCDM	CloudSim	Improved resource use and queue allocation
Nazeri & Khorsand (2022) [7]	Energy-aware Scheduling	FAHP + FTOPSIS	Hybrid fuzzy logic	QoS modeling	Optimized energy and performance
Kashani &Mahdipour (2022) [14]	Fog Load Balancing	Fog LB taxonomy	Systematic review	Fog parameters	Grouped algorithms; pros & cons analyzed
Ebrahim & Hafid (2023) [15]	Fog Resilience	MCDA + ELECTRE	ELECTRE for LB	Fog-IoT simulation	Balanced Fog load and low latency
Saha et al. (2023) [16]	Cloud Service Matchmaking	CREM + SWARA	Subjective-objective MCDM	User review dataset	Composite model improved selection

Gad-Elrab et al. (2024) [17]	Fog-Cloud Load Balancing	FAHP + FTOPSIS	Adaptive scheduling	Fog simulation	Reduced load variation, boosted utilization
Darius et al. (2024) [18]	Geo-distributed LB	MCRBgLB algorithm	Cloud Analyst	Simulation	27% cost cut, 63% time gain
Prity & Hossain (2024) [19]	LB Taxonomy Review	Algorithm review	Classification, simulation	Comparative charts	Comprehensive QoS-based taxonomy
Devi et al. (2024) [4]	Job Scheduling Review	SLR on LB & job scheduling	Systematic review	63 articles	ML optimization review, trends identified
Kotteswari et al. (2025) [20]	Energy-Efficient LB	MDP for task allocation	Markov Decision Process	Comparative simulation	52% energy saved, 96% availability
Hayyolalam& Ã–zkasap (2025) [21]	CBWO Scheduling	Chaos + BWO	CBWO algorithm	CloudSim	67% less makespan, 29% less energy used
Sonia & Nath (2025) [22]	ML in Load Balancing	ML/DL load balancing review	Review + insight	Various LB tools	ML trends for LB and edge/cloud contexts

III. Implementation Using CloudSim

CloudSim is a simulation framework specifically designed to model and analyse cloud computing systems, it is used to satisfy ACO Scheduler [4]. CloudSim provides a flexible platform for creating VMs and cloudlets, allowing researchers to experiment with new scheduling policies in an idealized environment [6].

3.1 Key Components of the ACO Scheduler Implementation:

Main Class (ACO_Scheduler): This class is as the entry for the simulation [3]. Its main method is used to initialize the CloudSim library, create one or more data center(s), and create broker which will handle task execution [4]. The first step of the code reads the cloudlet tasks from an input file [9]. This file includes information of the tasks and also gives input to generate the communication and execution matrices, which are important values during the scheduling decision-making process [3].

3.2 Creation of VMs and Cloudlets

VM Creation (createVM method): A predefined number of VMs are created with parameters (moment size, RAM, MIPS, bandwidth, and number of processing elements/PEs) [4]. These VMs are a pool of resources to run the scheduled tasks [5].

Cloudlet Creation (createTasks method): Cloudlets are generate by reading the input file [6]. The input file specifies the length or computation cost of several cloudlets in the file, which is parsed and added to the simulation [4]. These tasks are called cloudlets, which are based on real-world tasks that require processing in a cloud environment [3].

Broker and Pheromone Trail Initialization: A custom datacenter broker, normally an extension of the broker class of CloudSim, is generated to incorporate the ACO algorithm [3]. This broker is used to keep the pheromons trails and manages the process of interacting the ants (i.e. candidate solutions) with the real scheduling process [8]. The ACO algorithm signals trigger using the core method of the broker, commonly called RunACO, which applies an iterative ACO algorithm to find the optimal or near-optimal mapping of cloudlets to VMs.

Matrix Generation (GenerateMatrices Class): The GenerateMatrices class is responsible for computing two important matrices, the communication matrix, and the execution matrix The communication matrix model reflects the requirements for data exchange over cloudlets, while the execution matrix shows the estimated execution time on the different VMs[3]. These matrices are input into the ACO algorithm, where each is used to guide the probability of choosing certain mappings based on execution efficiency and communication overhead[8].

Simulation Flow and Result Processing: After the VMs, cloudlets, and matrices have been set up, they are handed off to the broker [4]. Then the simulation begins by invoking CloudSim. startSimulation() which performs the scheduled tasks based on ACO algorithm[3]. After running the simulation, that is, model execution, CloudSim provides detailed execution data, such as tasks start and finish times, and VMs used for execution. Finally, PrintResults is a dedicated method that takes this data and prints it in a tabular format, allowing researchers to study the performance of the scheduler [4].

3.3 Advantages of the ACO Scheduler

The ACO Scheduler offers several advantages over traditional scheduling methods [3]:

Adaptive Optimization: The dynamic changes in the cloud data center environment are dynamically adapted by the Ant Colony Optimization Algorithm by employing the pheromone-based communication of real ants. It iteratively refines its mapping strategy by evaluating the quality of solutions in prior iterations [3].

Envirotex is vs based on figure eg report and way the scenario, it offers robust solution even highly heterogeneous [2].

Results of load balancing simulation shows, well-balanced distribution of the cloudlets on the available VM. A balance ensures that there is less idle time, which allows resources to be more effectively utilized, and, in turn, leads to a reduced overall execution time [5].

Scalability: As ACO algorithm is inherently parallel so when number of tasks and VMs are increased performance is good [3]. The typical response from the simulation results indicates that the scheduler can remain effective and efficient at mapping with a large number of cloudlets without considerable performance degradation [8].

3.4 Comparison with Other Scheduling Techniques

The ACO Scheduler is one such cloud task scheduling advanced technique, but it is by no means the only one [19]. Multiple algorithms such as Particle swarm optimization, Genetic algorithms, Round Robin, Shortest Job first also intervene in the larger simulation framework [4].

Particle Swarm Optimization (PSO): PSO is inspired by social behaviour in birds flocking or fish schooling [21]. An optimization involving a group of potential solutions (particles) that moves within the problem space in accordance with both personal and group memories to optimize a task allocation problem [22]. PSO are then very fast convergent algorithms with the risk of falling into local optima [21].

Genetic Algorithm (GA): These algorithms mimic the natural process of evolution by using selection, crossover, and mutation to evolve a set of candidate solutions [4]. The GA, being a derivative free global optimizer, it is perfectly suited to student class problems with large, complex search space and also excels at the multi-objective optimization problem since it keeps a pool of diverge candidate solutions [8].

Round Robin (RR) and Shortest Job First (SJF): More classical algorithms [19]. While Round Robin distributes tasks in a round-robin manner to prevent starvation, SJF prioritizes tasks with the least expected execution time [2]. Such algorithms tend to be simplistic, but they can work reasonably well if tasks exhibit uniform or predictable behaviour in their environment. However, they might not be as successful in addressing heterogeneous workloads as metaheuristic approaches do [4].

Hybrid Approaches (SAPSO, SAWPSO, etc.): More sophisticated hybrid algorithms integrate components from different optimization methods [21]. A case in this category is SAPSO, which applies a Simulated Annealing (SA) operator on PSO, using SA for faster exploration and PSO for faster convergence [22]. Likewise, SAWPSO realizes Pareto-based selection and adaptively updates the weight for tackling multi-objective optimization problems [16]. These hybrids are designed to address and reduce the weaknesses of their individual components; they should be able to find better trade-offs between exploration and exploitation [21].

3.5 Simulation Results and Insights

When analysing the performance of the ACO Scheduler through CloudSim simulations, it is important to consider the following performance metrics [6]:

Execution Time and Make span: The make span is the overall time taken to finish all the cloudlets [3] It is proved that effective mappings can exploit aspects such as parallelism and thus lower communication overhead, making ACO-based scheduling effective in reducing the makespan substantially[8].

Minimized idle time — The ACO Scheduler reduces idle time and increases throughput of data center by balancing workloads across VMs available [5]. Distributed sprinkling improves overall system performance without being burdened by a single resource [20].

Scalability and Robustness Simulation results show that the ACO Scheduler can manage on-demand VMs and cloudlets at scale without all of its performance degrading significantly [3]. Its iterative, pheromone-driven approach enables it to achieve a high degree of robustness even when workload conditions change [8].

Comparison with Other Algorithms: Instead of specific algorithm alternatives, the ACO Scheduler outperforms PSO, GA, etc. in more complicated tasks and VM environments that would rival real-life implementations [21]. ACO algorithm is an umbrella term for numerous algorithms that integrates its own component, Out of which these are famous and finding applications in the multidimensional optimization problem which is proving its performance without execution time and facilitates the load balancing [3].

IV. Methodology

Java was used to implement various scheduling algorithms in the CloudSim simulation framework in this research [4]. CloudSim was initialized with a number of datacenters, brokers, virtual machines, and cloudlets, and communication and execution matrices were generated and configured in CloudSim to reflect realistic cloud environments [6]. Custom brokers that mapped cloudlets to VMs were implemented for each scheduling technique ACO, PSO, Genetic Algorithm, Round Robin, SJF, SAPSO and SAWPSO [3]. Simulation was carried out to measure execution times, load balancing, and makespan respectively, a detailed performance admixture of each algorithm under varying workloads and resource conditions [4].

V. Results and Discussion

5.1Algorithm used and their outcomes

Ant Colony Optimization (ACO) Scheduler: The ACO scheduler starts with the initialization of the CloudSim environment in order to create the brooks and datacentres and reads the cloudlet tasks from a file [3]. Communication and execution matrices are built based on cloudlet tasks [6]. Then the algorithm simulates the foraging behaviour of ants, where many artificial “ants” search different mappings between the cloudlets and the virtual machines (VMs). Therefore, in

one simulation run, we assigned cloudlets onto VMs, which were identified by numbers from 0 to 9, with the execution times were between 1.11 to 3.91-time units [3]. Cloudlets initiated at approximately 0.1-time units and terminated between 1.21- and 6.24-time units, reflecting highly efficient operation between processing in both parallel and sequential with relatively less idle time, producing an optimized makespan [8].

PSO Scheduler: In the PSO scheduler, the same process is performed on CloudSim, including the creation of datacenters and a specific PSODatacenterBroker and the generation of communication and execution matrices [21]. The PSO algorithm uses a "particle" to represent each mapping solution, updating its position through the best experience of itself and its neighbours [22]. For example, it correctly allocated cloudlets among VMs (IDs 0-9) with execution time between 1.23 and 3.61-time units in one trace [21]. The experimental results show that the PSO scheduler is able to balance the workload on available resources and reduce the overall time being spent between tasks (GPIO) if 0.1-time units were allocated for almost all tasks [22].

GA scheduler: Cloudlets are sorted according to their lengths and VMs are sorted by their MIPS to create an initial population of potential solutions [4]. Using operations such as selection, crossover, and mutation, the genetic algorithm iteratively evolves the solution population to minimize the makespan. Simulation results exhibited that cloudlet execution times varied from approximately 1.22–3.69-time units that demonstrated the load distribution across VMs was balanced by GA successfully. The mapping from cloudlets to VMs further enhanced after each evolutionary generation, leading to strong scheduling and resource usage at the end [8].

Round Robin (RR) Scheduler: In round robin scheduler, Cloudlets are distributed to virtual machines sequentially in a cyclic way [19]. Using a RoundRobinDatacenterBroker provides a good approximation as tasks will be dispatched from one VM to another in a sequential manner (e.g., on some of the test cases all tasks processed on VM ID 2)[2]. Due to such characteristics of this scenario, every cloudlet will finish successfully, but execution times may be larger in some cases, which resulted in cloudlet 0 running up to 4364.25-time units [19]. The challenge in Round Robin scheduling is that it guarantees fairness and an even distribution of work, but can lead to a longer overall makespan than more dynamic optimization techniques [2].

SAPSO (Simulated Annealing–Particle Swarm Optimization) Scheduler: The SAPSO scheduler is a combination of two optimization techniques that utilize the exploration characteristic of the Simulated Annealing and Particle Swarm Optimization that exploit the nodes [21]. The communication and execution matrices are generated and the SAPSO algorithm is used to find an optimal mapping of tasks to VMs after the CloudSim environment is initialized and after VMs and cloudlets are created [22]. For one such set of results: Times for the run ranged from about 456 to around 4998 units of time; average execution time was around 3097.08; average start time was around 4.36 units; average finish time was 4.43 units [21]. The observed variations are due to the differences in the workloads of the cloudlets while the SAPSO approach provides a more robust mechanism in order to escape local optima and balances the resources effectively [22].

5.2 Proposed Modified Algorithm

SAWPSO (Strength Pareto Adaptive Weighted PSO) Scheduler: SAWPSO is an extension of traditional PSO, by incorporating the concepts of Pareto dominance and adaptive weighting to tackle multiple objectives at once. In these solutions, evaluation of each particle's potential solutions are done based on execution time and resource utilization trade-offs. For instance, in a single simulation run execution in a single data centre using VMs with 1000 MIPS, the times for completion ranged from 0.69-time units (Cloudlet ID 1) to 2.71-time units (Cloudlet ID 13). The compromise between competing goals between the objectives surfaced a well-tuned system performance that optimized the trade-off objectives in a Mult objective scheduling approach.

5.3Comparative outcome

Table 2 compares different cloud task scheduling algorithms based on parameters such as execution time range, average start time, makespan (finish time), load balancing capability, and resource allocation efficiency. The algorithms analysed include ACO, PSO, GA, RR, SABPSO, and SAWPSO.

Table 2:Comparative Performance Analysis of Task Scheduling Algorithms in Cloud Environment

Algorithm	Execution Time Range (units)	Average Start Time (units)	Average Finish Time / Makespan (units)	Observations
ACO	~1.11 – 3.91	~0.1	Up to ~6.24	Balanced distribution across VMs; minimized idle time and reduced makespan.
PSO	~1.23 – 3.61	~0.1	Finishing around ~5.79 (max observed)	Efficient load balancing with prompt task initiation.
GA	~1.22 – 3.69	~0.1	Optimized order, lower	Robust evolutionary

			overall time	scheduling with effective resource utilization.
RR	~67.84 – 4364.25	Varies (sequential assignment)	Very high makespan	Fair cyclic allocation but overall processing time is significantly higher.
SAPSO	Average ≈ 3097.08 (varies)	~4.36	~4.43	High variation reflecting different workloads; robust resource allocation.
SAWPSO	~0.69 – 2.71	~0.1	Up to ~2.71	Achieves the best performance with an optimal balance of multiple objectives.

The comparison clearly shows that optimization-based scheduling algorithms outperform scheduling methods in cloud computing. SAWPSO delivers the best performance because it executes very quickly, reduces overall task completion time, and efficiently optimizes multiple objectives. The algorithm, such as SAWPSO is good for cloud task scheduling systems which helps in reducing the time to complete tasks and using resources efficiently which helps in minimizing makespan and maximizing resource utilization.

5.4 Final Findings

Results from this simulation showed that complex optimization algorithms such as ACO, PSO, GA, SAPSO and SAWPSO had considerably less execution times and also better load balancing with respect to simpler methodologies. In particular, SAWPSO achieved the best performance, finishing in as low as 2.71 unit fit as it performed best to multi-objective by adaptively weighting and employing pareto selection. On the other hand, classical strategies like Round Robin led to significant makespans of 4364.25 units respectively owing to their inflexible scheduling policies that caused poor resource utilization. That indicated the significance of dynamic and hybrid scheduling methods, as they are effectively capable of distributing jobs to the available VMs and reducing idle times in cloud environments. In general, results confirm that while simplicity ensures fairness in execution, using sophisticated algorithm approaches will provide better execution time, less execution time and overall efficiency in complex nature of cloud computing.

VI. Conclusion

In order to analyse the performance gain of different scheduling algorithms in the cloud environment which used CloudSim tool, the comparative study was done. The simpler methods (non-complex) would have a poor performance compared to the advanced optimization techniques (complex) such ACO, PSO, GA, SAPSO, and SAWPSO which performed better due to their capability in minimizing the overall makespan and in its load balancing in the time within a range resource of all the available VMs. SAWPSO in particular provided the most powerful results with the lowest overall finish times through its combination of a multi-objective optimization procedure with adaptive weighting and a Pareto-based selection process. The study further suggests that the performance of the system can be significantly improved by just selecting an appropriate scheduling algorithm, and this result encourages more research in the area of hybrid techniques for cloud resource management. In general, the research offers useful guidance for system designers and practitioners interested in leveraging task scheduling across complex, multi-resource and multi-host systems.

References

1. R. Gupta and O. P. Sharma, “A review exploration of load balancing techniques in cloud computing,” *Educational Administration: Theory and Practice*, vol. 30, no. 2, pp. 580–590, 2024.
2. M. A. Shahid, N. Islam, M. M. Alam, M. M. Su’ud, and S. Musa, “A comprehensive study of load balancing approaches in the cloud computing environment and a novel fault tolerance approach,” *IEEE Access*, vol. 8, pp. 130500–130526, 2020.
3. R. Khorsand and M. Ramezanpour, “An energy-efficient task-scheduling algorithm based on a multi-criteria decision-making method in cloud computing,” *International Journal of Communication Systems*, vol. 33, no. 9, p. e4379, 2020.
4. N. Devi, S. Dalal, K. Solanki, S. Dalal, U. K. Lilhore, S. Simaiya, and N. Nuristani, “A systematic literature review for load balancing and task scheduling techniques in cloud computing,” *Artificial Intelligence Review*, vol. 57, no. 10, p. 276, 2024.
5. S. Negi, M. M. S. Rauthan, K. S. Vaisla, and N. Panwar, “CMODLB: an efficient load balancing approach in cloud computing environment,” *The Journal of Supercomputing*, vol. 77, no. 8, pp. 8787–8839, 2021.

6. A. Moori, B. Barekatain, and M. Akbari, "LATOC: an enhanced load balancing algorithm based on hybrid AHP-TOPSIS and OPSO algorithms in cloud computing," *The Journal of Supercomputing*, vol. 78, no. 4, pp. 4882–4910, 2022.
7. M. Nazeri and R. Khorsand, "Energy aware resource provisioning for multi-criteria scheduling in cloud computing," *Cybernetics and Systems*, pp. 1–30, 2022.
8. R. Gupta and O. P. Sharma, "Optimization of load balancing in cloud computing through nature-inspired metaheuristic algorithms," *Journal of Electrical Systems*, vol. 20, no. 11s, pp. 3216–3226, 2024.
9. M. Hosseinzadeh, H. K. Hama, M. Y. Ghafour, M. Masdari, O. H. Ahmed, and H. Khezri, "Service selection using multi-criteria decision making: a comprehensive overview," *Journal of Network and Systems Management*, vol. 28, pp. 1639–1693, 2020.
10. E. Rafieyan, R. Khorsand, and M. Ramezanpour, "An adaptive scheduling approach based on integrated best-worst and VIKOR for cloud computing," *Computers & Industrial Engineering*, vol. 140, p. 106272, 2020.
11. B. A. A. A. M. Alruwaili, M. Humayun, and N. Z. Jhanjhi, "Proposing a load balancing algorithm for cloud computing applications," in *Journal of Physics: Conference Series*, vol. 1979, no. 1, p. 012034, 2021.
12. R. R. Kumar, B. Kumari, and C. Kumar, "CCS-OSSR: a framework based on hybrid MCDM for optimal service selection and ranking of cloud computing services," *Cluster Computing*, vol. 24, no. 2, pp. 867–883, 2021.
13. C. A. Shekhar and G. S. Sharvani, "MTLBP: a novel framework to assess multi-tenant load balance in cloud computing for cost-effective resource allocation," *Wireless Personal Communications*, vol. 120, no. 2, pp. 1873–1893, 2021.
14. M. H. Kashani and E. Mahdipour, "Load balancing algorithms in fog computing," *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1505–1521, 2022.
15. M. Ebrahim and A. Hafid, "Resilience and load balancing in Fog networks: A Multi-Criteria Decision Analysis approach," *Microprocessors and Microsystems*, vol. 101, p. 104893, 2023.
16. M. Saha, S. K. Panda, S. Panigrahi, and D. Taniar, "An efficient composite cloud service model using multi-criteria decision-making techniques," *The Journal of Supercomputing*, vol. 79, no. 8, pp. 8754–8788, 2023.
17. A. A. Gad-Elrab, A. S. Alsharkawy, M. E. Embabi, A. Sobhi, and F. A. Emara, "Adaptive multi-criteria-based load balancing technique for resource allocation in fog-cloud environments," *arXiv preprint arXiv:2402.01326*, 2024.
18. P. S. Darius, P. Majumder, V. N. Manju, S. Saha, J. Suneetha, and P. Mitra, "Multi-criteria rank based geo-distributed load balancing for cloud computing," in *2024 Second International Conference on Emerging Trends in Information Technology and Engineering (ICETITE)*, pp. 1–7, 2024.
19. F. S. Prity and M. M. Hossain, "A comprehensive examination of load balancing algorithms in cloud environments: a systematic literature review, comparative analysis, taxonomy, open challenges, and future trends," *Iran Journal of Computer Science*, vol. 7, no. 3, pp. 663–698, 2024.
20. K. Kotteswari, R. K. Dhanaraj, B. Balusamy, A. Nayyar, and A. K. Sharma, "EELB: an energy-efficient load balancing model for cloud environment using Markov decision process," *Computing*, vol. 107, no. 3, pp. 1–41, 2025.
21. V. Hayyolalam and Ö. Özkasap, "CBWO: A novel multi-objective load balancing technique for cloud computing," *Future Generation Computer Systems*, vol. 164, p. 107561, 2025.
22. Sonia and R. Nath, "A systematic review of various load balancing approaches in cloud computing utilizing machine learning and deep learning," *International Journal of Data Science and Analytics*, pp. 1–23, 2025.