Improving Load Balancing in Cloud Computing to Minimize Response time and enhancing resource utilization by using Hybrid meta-heuristic Ant Colony Optimization-Simulated Annealing Algorithm

MSc Research Project
Cloud Computing

Anoop Kumar
Student ID: x22249401

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Anoop Kumr
Student ID:	x22249401
Programme:	MSc. Cloud Computing
Year:	2023-2024
Module:	Research Project
Supervisor:	Sean Heeney
Submission Due Date:	12/08/2024
Project Title:	Improving Load Balancing in Cloud Computing to Minimize Response time and enhancing resource utilization by using Hybrid meta-heuristic Ant Colony Optimization-Simulated Annealing Algorithm.
Word Count:	7971
Page Count:	21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	12th August 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

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

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

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

Improving Load Balancing in Cloud Computing to Minimize Response time and enhance resource utilization by using Hybrid meta-heuristic Ant Colony Optimization-Simulated Annealing Algorithm

Anoop Kumar
Student ID: x22249401

Abstract

In cloud computing load balancing and task scheduling is critical and is used for the efficient distribution of workloads and computing resources among virtual machines. Load balancing is an NP-hard optimization problem. Unequal distribution of tasks among can lead to underloaded or overloaded VMs leading to poor resource utilization This research explores a hybrid meta-heuristic algorithm for task distribution and load balancing named ACO-SA, which combines the Ant colony optimization for exploring the solution search space and the Simulated annealing algorithm for Exploiting the search space and refining the solution. ACO is a bio-inspired meta-heuristic algorithm and mimics the foraging behavior of ants to find the shortest past to food representing the best way to distribute tasks among VMs. On other hand Simulated annealing algorithm is used to refine the search space for solutions and converge towards a near-optimal solution. The objective function(fitness function) is defined to minimize response time in ACO and SA. Response time and resource utilization is considered as evaluation parameters. The simulation was performed using the Cloudsim toolkit. The simulation results of the proposed ACO-SA algorithm showed improved performance in minimizing response time by 7% compared to traditional meta-heuristic algorithms such as ant colony optimization(ACO) and Particle swarm optimization(PSO). The hybrid algorithms combines the strength of ACO with SA to optimize load balancing in cloud computing.

Keywords: Cloud Computing, Taks scheduling, Load Balancing, Ant Colony Optimization, Simulated annealing algorithm, Objective function, Response time, Resource Utilization

1 Introduction

Load balancing in cloud computing is a critical challenge and useful for distribution of workload among virtual machines. Cloud computing technology is the on-demand sharing of computing resource using the internet. The main goal of load balancing is to distribute tasks efficiently among VMs such that no VMs is overloaded or underutilized. Static load balancing algorithms often fail to handle varying workload patterns. Dynamic load balancing algorithms can handle varying workloads, Various heuristic and metaheuristic algorithm approaches were studied to balance the load among virtual machines for optimizing resource utilization and improving performance (Gamal et al. 2019). With the growing complexity and diversity of cloud workloads, the current load-balancing algorithms have some limitations in achieving the efficient distribution of the load, response time, and energy consumption. the load on cloud computing resources entails solving the so-called "NP-hard" problems characterized by their complexity to the extent that no optimization method can come up with an ideal solution in polynomial time(Afzal & Kavitha 2019). Static load balancing methods does not consider the current state of the server when distributing load and it is suitable where the load on the system is fixed and constant to overcome the above problem we have Dynamic Load balancing which can predict in real-time the amount of load to be distributed to the servers. Dynamic load balancing is also used in the scaling of the application depending on the users' demand ,making it highly fault tolerant and optimized for performance. Heuristic and metaheuristic methods are now frequently used in cloud computing to attain load balancing. These methods have a number of main characteristics, for example an extended search space and random search capabilities, that aid in the identification of optimal solutions for scheduling problems within the required time frame. Cloud computing is a service on the Internet resource base that follows the concept of "everything can be a service". It consists of changing computing hardware and software resources into web services that can be accessed over the Internet. The three primary cloud computing models are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) by offering "virtual machine" (VM) technology. Virtualization has made it possible for ordinary users to run system and application software on VMs much more cheaply than the traditional method of acquiring and owning physical computers. A VMH is a term used to refer to a host machine that can be powered by one or more virtual machines (VMs) depending on its capacity. A critical challenge in cloud is managing large amounts of VMHs and task by dynamically reallocating resources to VMs in order to increase cost-performance efficiency and at the same time ensuring SLAs and load balancing among VMs. One of the methods of load balancing is to migrate VMs from overcrowded VMHs to other VMHs with the aim of distributing the workload evenly(Sekaran et al. 2019).Many algorithms are proposed for the load balancing in cloud computing which are based on the optimization algorithms. The existing meta-heruistic optimization algorithms like ACO, PSO, firefly etc. are unable to balance load in minimum amount of time due to complex optimization function. The load balancing scheme needs to be designed which can balance network load in minimum amount of time and also improve resource utilization. In this research work hybrid optimization algorithm is developed which is the combination of ACO (Ant colony optimization) and SA (Simulated Annealing). This hybrid algorithm aims to optimize response time and resource utilization.

1.1 Research Motivation and Background

Traditional load-balancing techniques often struggle with changing workloads and dynamic resource allocation leading to increase in response time and inefficient resource utilization. Metaheuristic algorithms offer promising solutions for solving complex optimization problems. Combining one or more of these algorithms can exploit their complementary strengths. The motivation behind developing the hybrid ACO-SA for load balancing comes from the need to achieve improved response time and enhance resource utilization.

1.2 Research Objectives

1. To develop and implement a hybrid meta-heuristic algorithm combining the Ant colony optimization and simulated annealing algorithm for load balancing using task scheduling.
2. Evaluate the Performance of the Hybrid Algorithm with other existing algorithms.

1.3 Research Question

The above research problem motivates the following research question:

1.) To what extent does the Hybrid meta-heuristic Ant Colony Optimization Algorithm in Load balancing minimize response time and enhance resource utilization to enhance cloud load balancing?

It is necessary to develop load-balancing algorithms that is efficienct and has less computational overhead to ensure that the optimization process itself does not become a problem. This research aims to fill these gaps by developing and optimizing the hybrid meta-heuristic ACO-SA task scheduling algorithm to optimize load balancing in the cloud. It will also ensure a comprehensive comparison of existing strategies with an emphasis on enhancing QoS factors such as resource consumption and response time. By addressing these aspects, the proposed research will not only advance the theoretical understanding of hybrid metaheuristic algorithms in cloud computing but also offer a practical approach to enhance load-balancing.

1.4 Document Structure

This research paper is further divided into 6 sections. Section 2 outlines the related work previously carried out in the area of task scheduling, load balancing, and hybridization of algorithms. Section 3 covers the research Methodology, including the techniques and algorithms used for research. Section 4 covers the Design specifications of the proposed algorithm. Section 5 contains the implementation of the proposed load-balancing algorithm. Finally, sections 6 and 7 consist of the experiment, results, and conclusion respectively.

2 Related Work

2.1 Load balancing techniques

(Pradhan et.al 2020) addressed the issue of poor workload distribution among virtual machines (VMs). They suggested using a modified particle swarm optimization technique (LBMPSO) to enhance load balancing and task scheduling. The primary objective was to decrease makespan and enhance resource usage with the incorporation of a fitness function that assesses virtual machine execution times for the best possible task distribution. Based on simulation findings, their approach minimized resource usage by 10% and decreased makespan by 15%. (Devaraj et.al 2020) identified the issue of optimizing resource utilization arising due to the inefficient workload distribution.To solve this issue they proposed Improved Multi-Objective Particle Swarm Optimization (FIMPSO) which combines firefly to minimize the search space and IMPSO algorithm to find the best solution for allocating task among VMs using the minimum distance criterion. The simulation result performed using MATLAB showed that FIMPSO performed better than the tradition algorithms such as Round Robin, FCFS, and Genetic algorithm. The IMPSO algorithm finds the minimum distance between the global best (gbest) particle and line by a point. The FIMPSO algorithm achieved the lowest average response time of 13.58 ms, highest CPU utilization of 98%, memory utilization at 93%, throughput of 72% and reduced makespan by 148s.

(Mrhari & Hadi 2019) proposed SASPSOLB load-balancing algorithm which uses game theory and a modified version of particle swarm optimization to balances workloads among Virtual machines. In order to identify the best or nearly best solution and minimize the expected response time, the load balancing problem has been represented as a constrained optimization problem. The game was taken into consideration as a noncooperative user game. The implemented algorithm described in this paper was compared with another genetic optimization-based technique. As per the simulation findings, this approach worked better in terms of Makespan and predicted reaction time. (Menaka and Kumar 2024) created a structure for load balancing and mixed support that was strategy-oriented to make maximum use of virtual machines with a comparable weight distribution. Time-Conscious Scheduling and Supportive Particle Swarm Optimization were combined in the SPSO-TCS technique to achieve initial load balancing and minimize make-span time. The goal of this step was to determine the best make span time minimization for every virtual environment. The research goal was to find the order of tasks that required the least amount of calculation time and to reduce the time needed to complete each task. Utilizing the hybrid idea led to the lowest energy usage and a shorter makespan. SPSO-TCS reduced makespan by 70-80% and improved resource utilization by 10-20% compared to traditional methods.

(Kruekaew & Kimpan 2022) proposed MOABCQ algorithm, a multi-objective task scheduling optimization to address the problem of inefficient task scheduling and load balancing. The approach combined the explorative nature of the ABC Algorithm with the exploitative nature of the Q-learning algorithm, for reinforcement learning to enhance the speed and efficiency of the ABC algorithm over independent task scheduling. The performance of the proposed approach is tested against the scheduling and load balancing techniques in use currently on three datasets: Random, Google Cloud Jobs, and Synthetic workload using CloudSim. Results demonstrate that the MOABCQ algorithms improved task distribution efficiency , reduced makespan and lowered cost. (Muneeswari et.al 2024) devised a new technique of cloud load balancing for virtual machines. A number of the input jobs from various users were collected into one task collector and then forwarded to the load balancer, which was empowered with the Bi-LSTM deep learning network . The task details will be sent to the load balancer to initiate the virtual machine migration when the load is unbalanced. Further, the refined Bi-LSTM, optimized by a GEP optimizer in the previous phase, was used to attain load balancing in virtual machines. Then, the effectiveness of the suggested LBVM with regards to the evaluation criteria like configuration delay, detection rate, accuracy, etc., has been evaluated against established methods such as MVM, PLBVM, and VMIS. The experimental results showed that the proposed solution decreased migration time by 49%, 41.7%, 17.8%, compared to MVM, PLBVM, and VMIS respectively.

2.2 Meta-heuristic algorithm based load balancing techniques

(Kumari et.al (2024) introduced the FHO (Fire Hawk Optimization) method for cloud-based dynamic load balancing. In order to improve system performance, their study took into account multi-objective functions as Makespan, use of resources, reaction time (RT), Level of Imbalance, and throughput. According to the experimental results, the FHO-based load balancing produced the following results for the number of tasks: Makespan, RT, RU, DOI, and throughput were 2246, 6.7, 134.23, 141, and 5.5, respectively. Comparing these results to other techniques such as PSO, QMPSO, and the Binary JAYA algorithm, showed better performance. (Abualigah et.al 2024) developed an optimization method for scheduling tasks in cloud computing using the Levy flying mechanism along with the Jaya algorithm and SSO (Synergistic Swarm Optimization), to achieve maximum trade-offs between exploration and exploitation and increasing the rate of convergence. A synergistic optimization paradigm is formed by combining the cooperative search of the SSO with the Jaya algorithm's capability to find and use the best solutions available. Levy flights added a random element to the search process, making it easier for the algorithm to traverse through complex solution spaces and escape local optima. The modified technique performed better than the original by 10% and with an overall accuracy of 88%.

(Singhal, et. al 2024) put forward a metaheuristic-based technique in order to provide an optimal load distribution with the help of Rock hyrax for efficient energy consumption. The algorithm solves the issues with energy consumption and local optima with the help of QoS parameters. The algorithm results were analyzed both qualitatively and quantitatively with static and dynamic work modes. The solution minimizes energy consumption in data center by 8%–13% and makespan by 10%–15% from that of trad scheduling algorithms. These results demonstrated the efficacy of the Rock Hyrax-based load balancing algorithm in improving data center productivity and energy conservation. (Khaleel 2024) presented a three-phase method for the RASA (Regional Awareness dynamic Scheduling Algorithm). Using a task categorization model, tasks were first grouped according to important factors such as CPU usage, memory utilization, and I/O operations. Estimating each node's CPU, memory, and I/O capacity was the second step. The nodes were then grouped into clusters using a special coalitional game-theoretic merge-and-split procedure. To address the sluggish convergence and local optimization problems with the conventional Sparrow Search Algorithm (SSA), they introduced an upgraded version of the algorithm in the final phase, which improved job placement. The suggested algorithm reduced processing time by 14%, workload imbalance by 15%, latency overhead by 9%, energy consumption by 19%, idle times by 26%, and workload imbalance by 15%. In addition, it increased resource efficiency and availability by 27% and 22%, respectively, and increased service throughput by 32%.

(P. K. K R et.al 2023) provided a novel load balancing technique called FFBSO, combining Bird Swarm Optimization (BSO) with the Firefly algorithm (FF), which reduced the search space. BSO models tasks as birds and VMs as destination food patches, drawing inspiration from the flock behavior of birds. Tasks were seen as independent and non-preemptive in the cloud environment. However, by determining the potential optimal locations, the BSO algorithm mapped jobs onto appropriate virtual machines. The results of the simulation showed that the FFBSO algorithm outperformed the other methods, achieving a makespan of 35s, the lowest average reaction time of 13ms, and maximum resource use of 99%. (Jena et.al 2020) introduced a new approach to balance the load among the VMs (virtual machines) for which a hybrid technique called QMPSO (modified Particle swarm optimization-improved Q-learning) was implemented. Both the algorithms were integrated for adjusting the velocity of the MPSO depending on the best action obtained from the improved Q-learning. The waiting time of tasks was optimized such that the load was balanced among the VMs, the throughput of VMs was increased and the balance was maintained among priorities of tasks. The findings revealed that, in comparison to the conventional methods, the new strategy was reliable and the use of adaptive strategies can help in dealing the unequal distribution of workload.

2.3 Hybrid Load Balancing algorithms

(Kaur & Kaur 2022) identified the challenge of efficient load balancing and resource utilization among VM's and proposed a hybrid heuristic-metaheuristic approach combining predict earliest finish time(PEFT) and Heterogeneous Earliest Finish Time (HEFT) heuristics with Ant Colony Optimization (ACO), termed as HPA and HHA respectively. They considered makespan, cost, and resource utilization as key factors. The two approaches were inspected and assessed to determine which solution to balance load was superior for the suggested HDD-PLB structure. Result showed that the average makespan was reduced by up to 23.41% improved resource utilization and ensuring a more balanced distribution of tasks across virtual machines. (Hodžić & Mrdović 2023) identified the problem of inefficient load balancing and poor resource usage, and increased response time. The proposed genetic algorithm technique for load balancing dynamically distributes incoming requests amongst cloud resources. The program was designed to process each request as soon as it arrived. The suggested method outperformed throttled load balancing, ESCE, and round-robin algorithms in terms of response and processing time, according to the test simulations that were run by reducing overall response time by 10.05% and data center processing time by 10.87%.

(Haris & Zubair 2022) developed the MMHHO (Mantaray modified multi-objective Harris hawk optimization to solve the problem of underloaded and overloaded VM's leading to increased power consumption and machine failure. The hybridization procedure optimized the search space of Harris Hawk Optimization (HHO) by considering variables including the use of resources, expense, and reaction time using the MRFO (Manta Ray Forging Optimization) method. The hybrid strategy improved the system performance by improving the virtual machine throughput, distributing the load among the VMs, and preserving the balance of task priorities by altering the waiting period. The recommended MMHHO-algorithm technique was implemented in the Cloud Sim tool. The results of the simulation indicated that the recommended MMHHO approach performed better than standard algorithms. Their work shows that hybridization can result in enhancement in throughput, makespan and energy consumption.

2.4 Summary of Literature Review and Proposed method

The literature review addresses various efficient optimization techniques for load balancing techniques in cloud computing. Swarm intelligence methods, such as LBMPSO and FIMPSO, enhance task scheduling and resource utilization. SASPSOLB and SPSO-TCS are evolutionary algorithms that show low makespan and energy consumption according to the game theory.Bio-inspired algorithms like MOABCQ enhance the issue of task distribution. Literature reviews on comparative studies demonstrate the efficiency of multi-objective optimization approaches, FHO and other hybrid techniques, combining Levy Flight, Jaya Algorithm, and SSO. HEFT (Heterogeneous Earliest Finish Time) and PEFT (Predict Earliest Finish Time) heuristics with ACO are two examples of a hybrid strategy that combines heuristic approaches and metaheuristic algorithms. The MMHHO (Mantaray modified multi-objective Harris hawk optimization), which enhanced dynamic load balancing by updating the search space based on resource consumption, cost, and response time. A genetic algorithm to balance cloud load efficiently assigned cloudlets to suitable virtual machines and validated faster execution and improved distribution of load as opposed to the baseline algorithms. In general, all of these different approaches contribute towards the improvement of load balancing and the conservation of resources in cloud computing. These studies highlight diverse hybrid and heuristic-metaheuristic approaches that can be used to optimize load balancing.

3 Methodology

The goal of this work is to provide a novel hybrid solution to tackle the cloud load balancing challenge. This requirement results from the constantly changing nature of the demand for cloud computing, which includes the growing complexity of jobs, the variety of services available, and the dynamic allocation of workloads. Investigating cutting-edge load-balancing options that can effectively manage resource variability and dynamically adjust to the evolving needs of cloud computing.

3.1 Traditional Ant Colony Optimization and its drawbacks

ACO (Ant Colony Optimization) is a bio-inspired community-based meta-heuristic algorithm for resolving complex optimization issues. The method draws inspiration from the foraging behavior of ants, who use pheromone trails to determine the shortest path from their colony to a food source, the stronger the pheromone trail more likely it is that more ants would follow. Traditional ACO possess many limitations that can reduce their effectiveness due to their slow convergence time which is due to the probabilistic nature of search and exploration processes involved in the learning mechanism. ACO also has the tendency to quickly fall into local optima, which limits the solution space's exploration capabilities. To find a balance balance between exploration and exploitation pheromone level needs to be controlled, If the rate of pheromone evaporation is high the algorithm may not converge at all, and If it is very low the algorithm may converge quickly towards a less optimal solution (Liu. (2022).

3.2 Hybrid approach toward load balancing

To overcome above discussed limitation, the ACO algorithm it must be hybridized with other meta-heuristic algorithms which can speed up convergence and avoid getting trapped in local optimum solution. ACO method shows the decentralized nature of cloud computing where multiple agents (ants) work in parallel and operate independently to find solutions which leads to efficient search as algorithms adapts to change in problem space. ACO updates pheromone trails are based on current solutions to find the best path(solution) through complex solution spaces(Gao. 2024). Simulated annealing algorithm as the ability to mitigate the problem of Local optima in ACO and converge toward a global optimum solution. It is a probabilistic optimization technique inspired by the annealing process in metallurgy and imitates the process of cooling. It avoids local optima and searches for a larger solution space to approximate global optima by accepting less optimal solutions with a certain probability that decreases over time. The temperature factor in SA offers a balance between exploitation (refining current solutions) and exploration (searching new areas of the solution space) (Javadi & Gandhi 2022). This flexibility is crucial in field of cloud computing for effective load balancing. ACO start with certain number of ants working together to assign cloudlets(tasks) to virtual machines (VMs) based on current workload and pheromone levels in order to find the best possible solutions. The hybrid technique uses SA to refine and adjust the initial solutions generated by ACO by exploring neighboring solution and changing cloudlet(task) distribution based on a temperature element that controls the acceptance of possible improvement. Pheromone levels are changed after each iteration, with more better solutions having higher pheromone levels, which direct other ants toward more optimal solution. The combination of SA's local optimization capabilities with ACO's global search, the hybrid ACO-SA algorithm can offer a powerful approach toward efficient load balancing. Each ant starts by searching the path, probability of choosing the node i node j-th individual in the population is determined by the probability expression (Shi & Z 2021):

$$P_{ij}^{k} = \begin{cases} \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{s\in a_k}\tau_{is}^{\alpha}\eta_{is}^{\beta}}, & \textbf{if } j \in a_k \\ 0, & \textbf{otherwise} \end{cases}$$

Here, α (alpha) and β (beta) are parameters that control the influence of pheromone trail and heuristic information respectively in decision-making process of ants. Higher value of α (alpha) and β (beta) influence the ant's path toward a optimal solution. τij is the pheromone density of the path. The pheromone density update is described by the below expression(Shi & Z 2021) :

$$\tau_{ij}^{t+1} = (1-\rho)\cdot\tau_{ij}^{t} + \Delta\tau_{ij}$$

τ_{ij}^{t+1} represents the pheromone density between nodes. Pheromone evaporation(ρ) prevents the algorithm from converging too quickly.The pheromone values on all paths are reduced by a factor of (1−ρ). For each ant's solution, pheromone is deposited to the paths which increases the pheromone value making those paths more suitable for future iterations and solution, in proportion to the inverse of the solution's cost. After the ACO stage is completed, pheromone evaporation is applied to further explore the solution space to avoid premature convergence. After this the SA phase starts to refine the best solution found during the ACO stage. During the refinement process, neighboring solutions are generated through perturbation and evaluated against the current best solution. If the new solution shows better fitness value it is accepted, if not it can still be accepted probabilistically, depending on the fitness value and the current temperature, which decreases gradually over each iteration. The cooling function makes the search more focused and narrow in order to refine the solution space. After all, iterations are complete, the best solution returned has the best way to distribute tasks and balance load. This show the effectiveness of combining the exploration abilities of ACO with the refinement capability of SA to enhance the distribution of task in cloud systems for better load balancing.

3.3 Proposed Research Architecture

The proposed architecture discusses about the flow of workload generation to task allocation and its execution, utilizing a hybrid algorithm for task distribution. See figure 1. For archiecture diagram

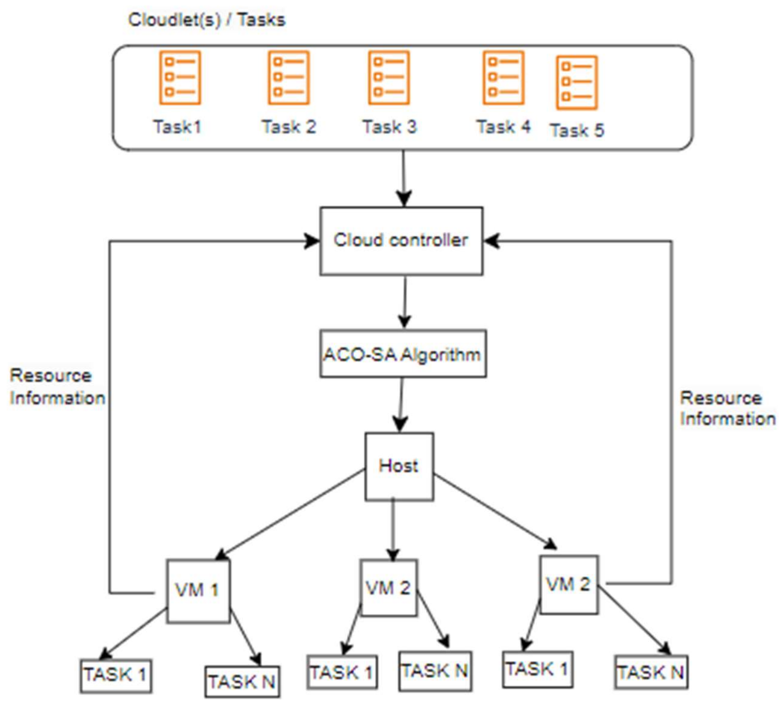
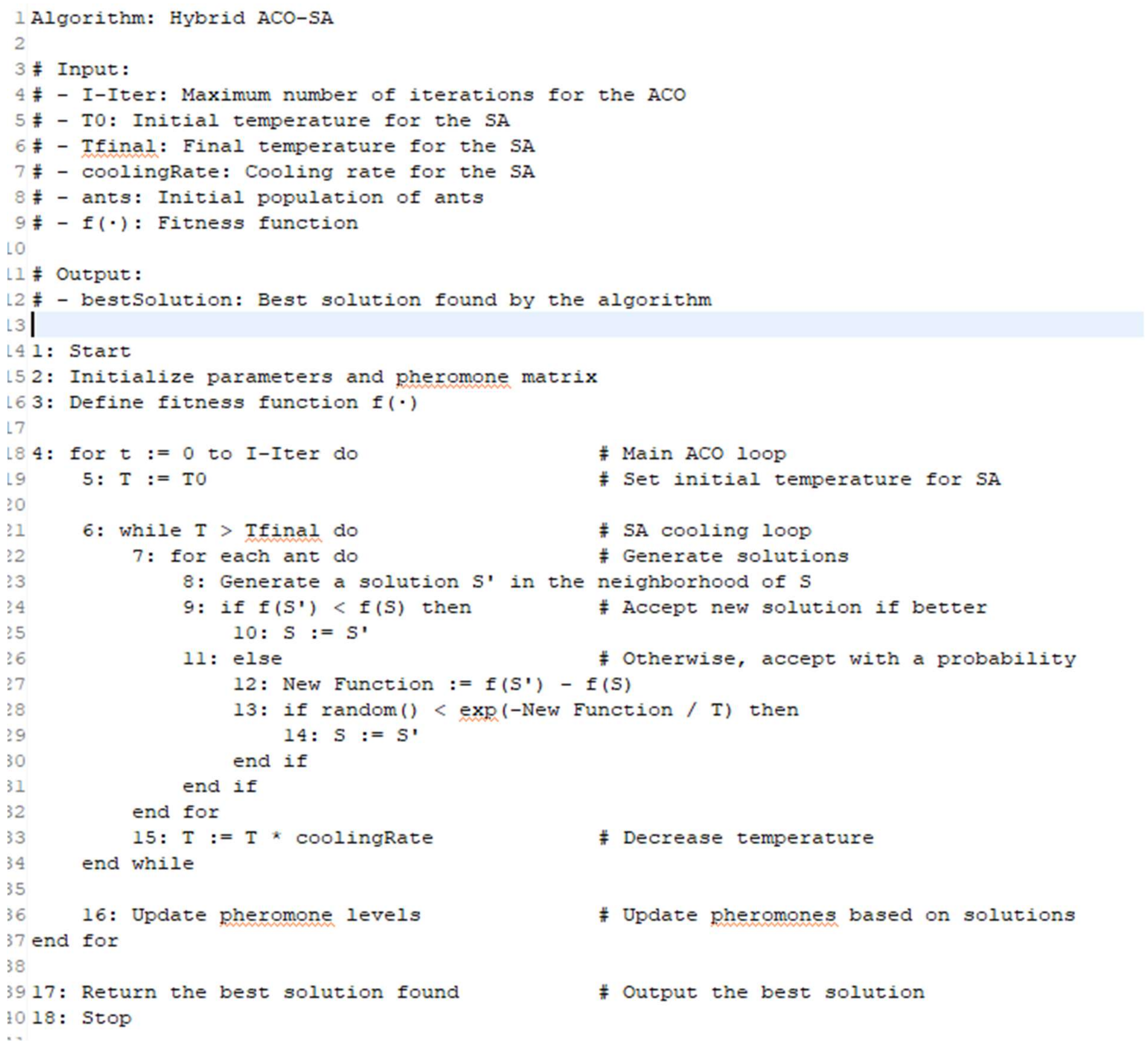


Figure 1: Proposed Architecture

The architecture starts with Workload(Cloudlet/task) generation which are produced based on user requests or simulation scenarios within CloudSim. Tasks are stored in a task queue managed by the cloud controller. The controller is responsible for managing the task scheduling and execution process, The Cloud controller is an important component as it maintains information on available resources(Virtual machines), including the information about current load on each VM, their processing capabilities, and their availability status. Once task are queued in task queue Cloud controller invokes the ACO-SA algorithm for optimal allocation of these task to the available VMs. The Ant colony optimization(ACO) algorithm explores potential solutions (best VM to allocate each task based on pheromone trails and heuristic information). Then SA algorithm is applied to further refine current solution and explore the neighboring solutions and compare them, using a probabilistic approach to avoid getting stuck in local optima and converge towards a global optimal distribution of task. After the hybrid ACO-SA algorithm has processed task allocation details, it sends this information back to the cloud controller, which then assigns each cloudlet to a specific VM as per the algorithm. Virtual machines(VMs) execute tasks and provide report metrics such as start time, finish time, CPU usage, and response time. Because the entire process is iterative, the Cloud controller uses this information to modify the distribution of tasks in the future, minimizing response time and resource utilization.

3.4 Pseudo-code of the Algorithm



```
Algorithm: Hybrid ACO-SA

# Input:
# - I-Iter: Maximum number of iterations for the ACO
# - T0: Initial temperature for the SA
# - Tfinal: Final temperature for the SA
# - coolingRate: Cooling rate for the SA
# - ants: Initial population of ants
# - f(·): Fitness function

# Output:
# - bestSolution: Best solution found by the algorithm

1: Start
2: Initialize parameters and pheromone matrix
3: Define fitness function f(·)

4: for t := 0 to I-Iter do                    # Main ACO loop
    5: T := T0                                # Set initial temperature for SA

    6: while T > Tfinal do                    # SA cooling loop
        7: for each ant do                    # Generate solutions
            8: Generate a solution S' in the neighborhood of S
            9: if f(S') < f(S) then           # Accept new solution if better
                10: S := S'
            11: else                          # Otherwise, accept with a probability
                12: New Function := f(S') - f(S)
                13: if random() < exp(-New Function / T) then
                    14: S := S'
                end if
            end if
        end for
        15: T := T * coolingRate              # Decrease temperature
    end while

    16: Update pheromone levels               # Update pheromones based on solutions
end for

17: Return the best solution found            # Output the best solution
18: Stop
```

Figure 2 : Pseudo-code of the ACO-SA Hybrid Algorithm

The algorithm starts by initializing initial parameters such as the number of ants, maximum number of repetitions, pheromone levels, beginning temperatures, and the rate of cooling including the pheromone matrix which guides ACO ants in their search for solutions and fitness function that will evaluate the quanlity of solutions that meet the optimization criteria(minimum response time). The fitness function guides the search process towards optimal solutions. The main ACO loop runs for a certain number of iterations which drives the ACO process, The initial temperature (T) is reset to initial value(T0). Within each ACO iteratin SA phase begins runs till the temperature T is crosses threshold value of (Tfinal).Each ant generate a potential solution(S') which is compared to the current solution (S) using the fitness function. if S' is better than the Current solution(S), it is accepted as the new current solution and if not it may still be accepted based on probability that allows it to escape local optima by accepting the worst solution occasionally. After the solution is generated and evaluated temperature is decreased gradually by multiplying it with the cooling rate and converging toward a better solution After the SA phase is completed with an ACO iteration, the pheromone level are updated and solution that had better results have stronger pheromone trail , guiding other ants toward that path. After all the iteration are completed the best solution is returned representing optimal or near optimal task distribution among VMs. The algorithm stop after finding the best solution.

3.5 Evaluation Parameters

Relevant parameters were established in order to evaluate the algorithms performance and conduct a comparison with an already existing meta-heuristic scheduling algorithm such as ACO and PSO. The hybrid algorithm's main goals are to effectively schedule tasks, reduce response time, and improve resource utilization. The evaluation parameters were chosen in a way that made it simple to assess the algorithm's effectiveness. The following are parameters were employed in the assessment.

- **Response time -** This represents the total time spent on task execution following scheduling of task. The goal of the hybrid algorithm is to minimize execution time by assigning tasks to the best possible virtual machine (VM) based on heuristic factors like current load and availibility of resource.
- **Resource Utilization -** This metric evaluates how efficiently the available resources, like CPU and memory, are being used across all virtual machines. Because the fitness function of the hybrid algorithms considers the execution time and load on each virtual machine and assigns the task to the least loaded VM in the network which in return maximizes resource utilization.

4 Design Specification

This work presents a hybrid meta-heuristic ACO-SA (Ant Colony optimization-simulated annealing) technique to balance cloud load. The main objective is to improve the response time and resource utilization while achieving load balance in the cloud by merging the exploratory powers of ACO with the exploitation powers of SA. The hybrid Ant Colony Optimization (ACO) and Simulated Annealing (SA) algorithm design features offer a comprehensive solution for cloud load balancing. Here, the goal is to distribute the cloudlets(tasks) among virtual machines (VMs). This Hybrid approach primary components include an ACO and SA algorithms component that is distinct but complimentary. Below is the flow chart of the proposed algorithm(Figure 3).

The first step is to create a datacenter ,with defined number of cloudlets(numTasks) and virtual machines(numVMs). The next steps is setting the algorithm's parameters for the number of ants(numAnts), maximum number of iterations, rate of pheromone evaporation(ρ), initial temperature(T0), and the cooling rate for SA algorithm. The ACO-SA algorithm's objective function (fitness function) is defined for minimizing response time which is the sum of the total time taken to perform all the tasks.

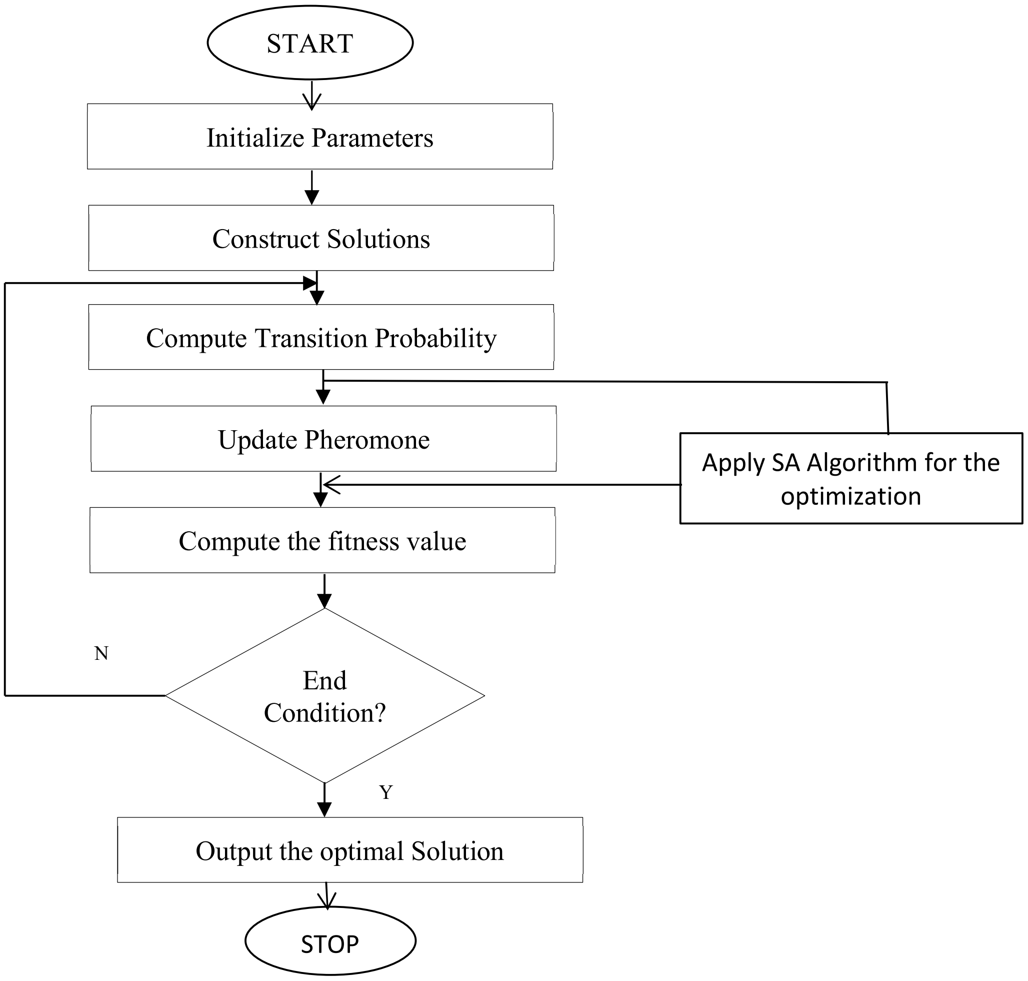

START
Initialize Parameters
Construct Solutions
Compute Transition Probability
Update Pheromone
Apply SA Algorithm for the optimization
Compute the fitness value
N
End Condition?
Y
Output the optimal Solution
STOP

Figure 3: Flowchart of the proposed ACO-SA Hybrid algorithm

In the ACO phase, each ant starts with constructing a solution by iterative assignment of tasks among virtual machines (VMs) based on pheromone levels and heuristic information, the quality of each solution is evaluated and pheromone levels are adjusted accordingly. The pheromone evaporation is applied to further explore the solution space. In the refinement phase of SA, the temperature decreased gradually, while neighboring solutions are generated and their fitness function is evaluated , if the neighboring solution is better and has better fitness, it replaces the current solution. By iteratively updating and re-evaluating solutions based on the pheromone levels and the probabilistic acceptance criteria that are dependent on temperature, the hybrid ACO-SA algorithm provides a balance between exploration and exploitation to achieve better response time and resource utilization.

5 Implementation

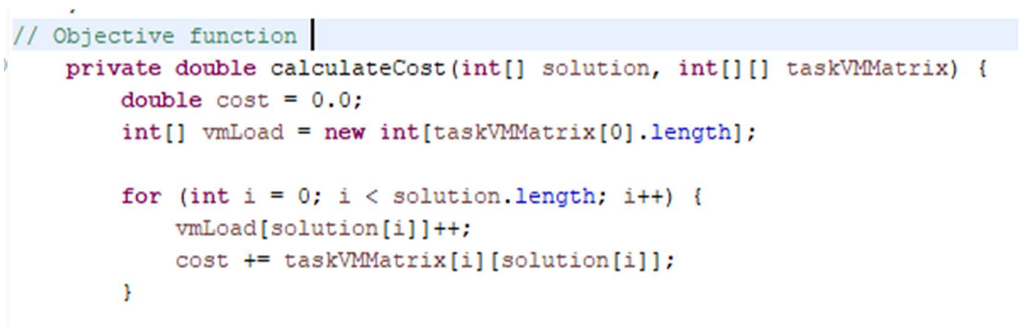
The research work is based on load balancing using hybrid optimization algorithm. The cloud sim is the simulator which is used with eclipse for the simulation of proposed model. The Window 11 system is used with 8 GB RAM and 256 GB Hard disk. The various simulation parameters are considered to evaluate performance of proposed model.

5.1 Tool used for Evaluation

The Cloud Sim is the open-source framework which is used to model and simulate the cloud computing environment. The cloud sim is written in java and it is designed by the cloud Labs. The cloud sim is used to evaluate the designed algorithm based on the simulation and it can produce the test results. CloudSim framework supports the simulation of large-scale data center. The cloud Sim is a package which is open source and available free of cost.

5.2 Formulating the objective function

The objective function or fitness function, is the main component in the algorithm and optimization process. The objective function is defined to reduce the response time, which is the total time required to complete the tasks. This function is the main criterion for optimizing the task distribution for better load balancing which is implemented in files “SA.java” and “ACO.java” and is defined with the “calculateCost” method in both ACO and SA phase to calculate response time(Figure 4). This function takes into consideration the solution array that defines the task-VM assignment in order to determine the total response time. The taskVMMatrix has the execution time of each task on every virtual machine. The below objective function calculates total response time for all cloudlets based on their VM allocation. In hybrid algorithm ACO generates the initial solutions by assigning cloudles to VMs and SA optimizes the solution by minimizing the objective function by exploring the neighboring solution.



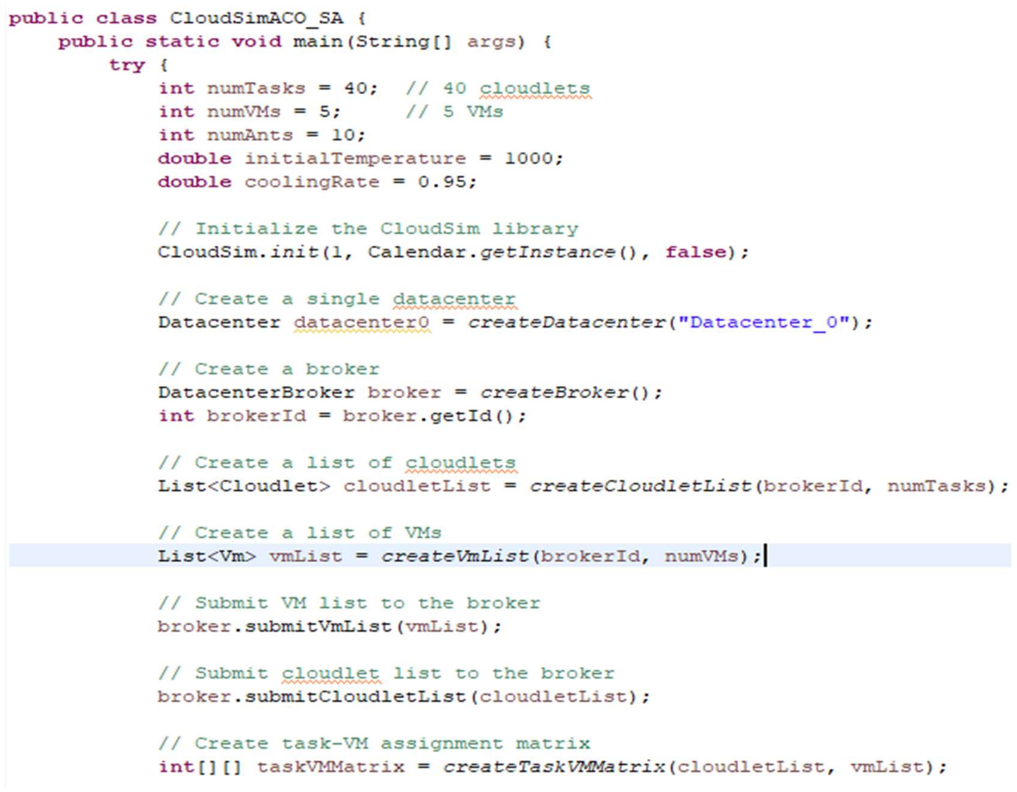
```
// Objective function
    private double calculateCost(int[] solution, int[][] taskVMMatrix) {
        double cost = 0.0;
        int[] vmLoad = new int[taskVMMatrix[0].length];

        for (int i = 0; i < solution.length; i++) {
            vmLoad[solution[i]]++;
            cost += taskVMMatrix[i][solution[i]];
        }
```

Figure 4: Objective function to calculate fitness of solution

5.3 Checking the initial simulation parameters

The initial simulation settings involves creating the datacenter, cloudlets (tasks), and virtual machines (VMs), and algorithm parameters (Figure 5).



```
public class CloudSimACO_SA {
    public static void main(String[] args) {
        try {
            int numTasks = 40;  // 40 cloudlets
            int numVMs = 5;     // 5 VMs
            int numAnts = 10;
            double initialTemperature = 1000;
            double coolingRate = 0.95;

            // Initialize the CloudSim library
            CloudSim.init(1, Calendar.getInstance(), false);

            // Create a single datacenter
            Datacenter datacenter0 = createDatacenter("Datacenter_0");

            // Create a broker
            DatacenterBroker broker = createBroker();
            int brokerId = broker.getId();

            // Create a list of cloudlets
            List<Cloudlet> cloudletList = createCloudletList(brokerId, numTasks);

            // Create a list of VMs
            List<Vm> vmList = createVmList(brokerId, numVMs);

            // Submit VM list to the broker
            broker.submitVmList(vmList);

            // Submit cloudlet list to the broker
            broker.submitCloudletList(cloudletList);

            // Create task-VM assignment matrix
            int[][] taskVMMatrix = createTaskVMMatrix(cloudletList, vmList);
```

Figure 5: Initial simulation Parameter function

Several important settings must be defined while configuring a virtual machine (VM) in order to guarantee optimal performance and allocation of resources. To uniquely identify the virtual machine (VM) within the system, for example, the virtual machine identification (vmid) is set to 0. The MIPS (Million Instructions Per Second) rating, which in this instance is set to 1000, indicates the computing capacity of the VM and determines the processing power allotted to it. The storage allocated to each VM is a size of 10,000 MB. 512 MB of RAM is allotted to the virtual machine (VM), which is sufficient to handle the workload effectively. Furthermore, 1000 Mbps bandwidth (bw) have been assigned to each virtual machine (VM), ensuring high-speed data transfer between VM and the data center. The virtual machine is set up with a single processing element (pesNumber), meaning that one CPU core will be used for operation. The virtual machine (VM) will run over the Xen Virtual Machine Monitor (vmm), which oversees the VM's execution and offers virtualization features. The creation and configuration of VMs are implemented under the “createVmList” method in the “CloudsimACO_SA.java” file which generates list of VMs with the above-specified configuration parameters and assigns them to datacenter broker for management. With this configuration, a balanced cloud environment is simulated capable of handling the dynamic workload generated by cloudlets.

5.4 Pheromone update rule for ACO

In the ACO-SA hybrid, the pheromone update rule is the most important factor that determines the trade-off between exploration and exploitation to find optimal or near-optimal solutions. The pheromone levels are updated after the construction of solutions where the quality of the solutions found is taken into consideration.

Pheromone deposit: Once the ant has constructed the solution, the fitness function is used to evaluate the quality of the solution with respec to factor such as response time and resource utilization. Pheromone levels are increased inversely proportional to the path length, due to this pheromone deposition on shorter paths increase. The amount of pheromone deposited on shorter paths(best solution) is more which encourages other ants to follow the path.

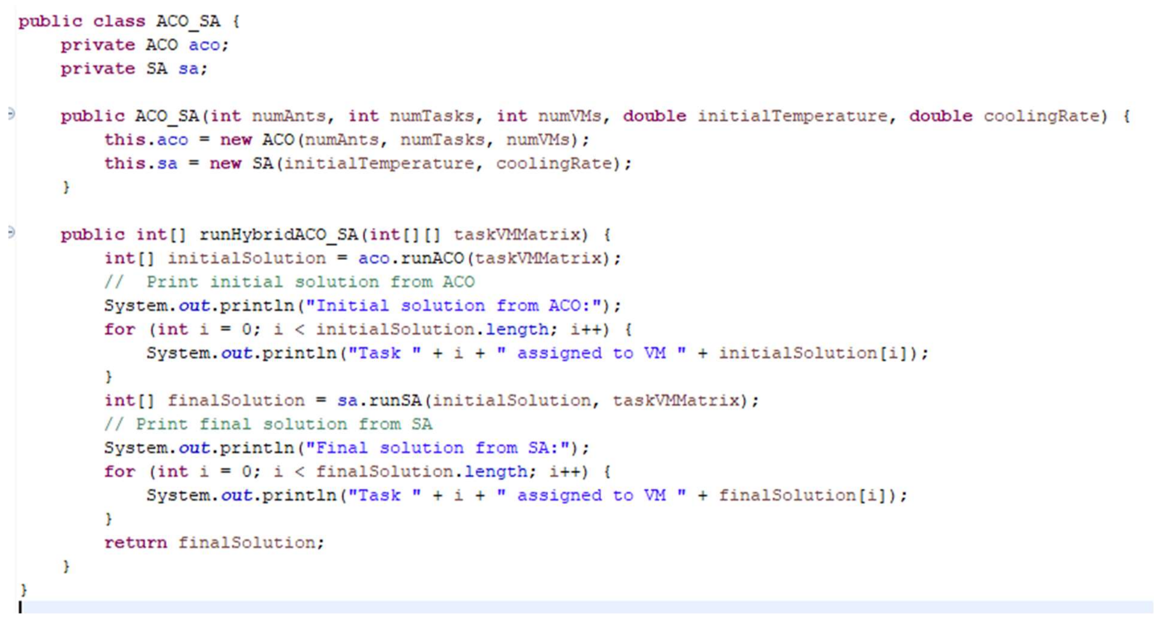
Pheromone Evaporation: With time pheromones on all paths taken by ants start evaporating, controlled by the evaporation rate(rho). The evaporation rate(rho), makes sure that the intensity of the pheromones on suboptimal paths decreases with time while the algorithm converge towards an optimal or near-optimal solution.

In this implementation,"Update_pheromones" method in ACO.java file is responsible for controlling the pheromone level during after each iteration of ACO.

5.5 Hybridization of ACO and SA Algorithm

The hybrid of ACO and SA algorithms is implemented in this research makes it possible to find the best solutions and optimize them, thus achieving better performance and efficiency. The hybridization is implemented in three steps:

1.) ACO phase: This step involves constructing an initial solution using ‘‘runsACO()‘‘ method by assigning tasks to VMs based on Parameters such as the number of ants, pheromone levels, and heuristic information. In next step the quality for each solution is evaluated using the fitness function(response time) and pheromone levels is update based on the quality of solution. Increasing the number of ants increases the ability to explore a larger solution space but increases the computation time. Pheromone levels are updated based on the quality of the solutions. A higher rate of evaporation benefits in exploration since it minimizes the impact of past solutions. A lower rate benefits the exploitation of the best-found solutions. The ACO phase is implemented in the "ACO.java" file.
2.) SA phase: This step involves accepting the best solution generated by the ACO phase via "runSA()" method in "SA" class and refining it by exploring the neighboring solution generated through perturbation and evaluating against the current best solution to avoid getting trapped in local optima by using a probabilistic acceptance criterion based on fitness change(Solutions that improve the fitness are accepted) and temperature, which decrease gradually over iterations thus improving the overall solution quality. SA phase is implemented in the "SA.java" file.
3.) Hybrid Integration : This step integrates the ACO and SA phase in "runHybridACO_SA()" method by running the ACO algorithm phase first and generate initial solution and take the best solution from ACO phase and pass it to the SA phase for refinement. Higher initial temperatures enable SA to accept less-optimal solutions at the beginning of the search process, which facilitates exploration. Lower temperatures however decrease this effect with more emphasis on exploitation. A slower cooling rate (closer to 1) enables a gradual refinement of the solutions, which may result in improved solutions, but at the cost of longer execution time. A faster cooling rate results in faster convergence but the solutions obtained may not be the most optimal. The main aim of integration is to balance exploration(finding a large solution) and exploitation(refining the best solution) This step is implemented in the "ACO_SA.java" file using the ACO_SA class(Figure 6) .



```java
public class ACO_SA {
    private ACO aco;
    private SA sa;

    public ACO_SA(int numAnts, int numTasks, int numVMs, double initialTemperature, double coolingRate) {
        this.aco = new ACO(numAnts, numTasks, numVMs);
        this.sa = new SA(initialTemperature, coolingRate);
    }

    public int[] runHybridACO_SA(int[][] taskVMMatrix) {
        int[] initialSolution = aco.runACO(taskVMMatrix);
        //  Print initial solution from ACO
        System.out.println("Initial solution from ACO:");
        for (int i = 0; i < initialSolution.length; i++) {
            System.out.println("Task " + i + " assigned to VM " + initialSolution[i]);
        }
        int[] finalSolution = sa.runSA(initialSolution, taskVMMatrix);
        // Print final solution from SA
        System.out.println("Final solution from SA:");
        for (int i = 0; i < finalSolution.length; i++) {
            System.out.println("Task " + i + " assigned to VM " + finalSolution[i]);
        }
        return finalSolution;
    }
}
```

Figure 6 : Integration of ACO with SA phase.

By carefully tuning parameters such as pheromone, evaporation rate, initial temperature, cooling rate, the hybrid algorithm can balance exploration and exploitation to optimize response time and resource utilization in cloud load balancing.

6 Evaluation

In this section, the performance of the hybrid meta-heuristic ACO-SA (Ant Colony Optimization-Simulated Annealing) algorithm is evaluated. The Response time and resourc utilization is the parameters used for the Performance Analysis. The results are compared standalone Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) algorithm.

6.1 Experiment / Case Study 1

The objective of this experiment is to evaluate the performance of the proposed hybrid Ant Colony Optimization-Simulated Annealing (ACO-SA) algorithm in terms of response time. The performance of the hybrid algorithm is compared with two other algorithms: Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO).
No of Datacenter : 1
No of VM : 5
Workload : The workload consists of a varying number of cloudlets (tasks), ranging from 10 to 40

Table 1: Response Time Analysis

Number of Cloudlet	Ant Colony Optimization	PSO algorithm	Hybrid Algorithm
10	96 seconds	143 seconds	88 seconds
20	192 seconds	235 seconds	168 seconds
30	261 seconds	383 seconds	243 seconds
40	349 seconds	519 seconds	328 seconds

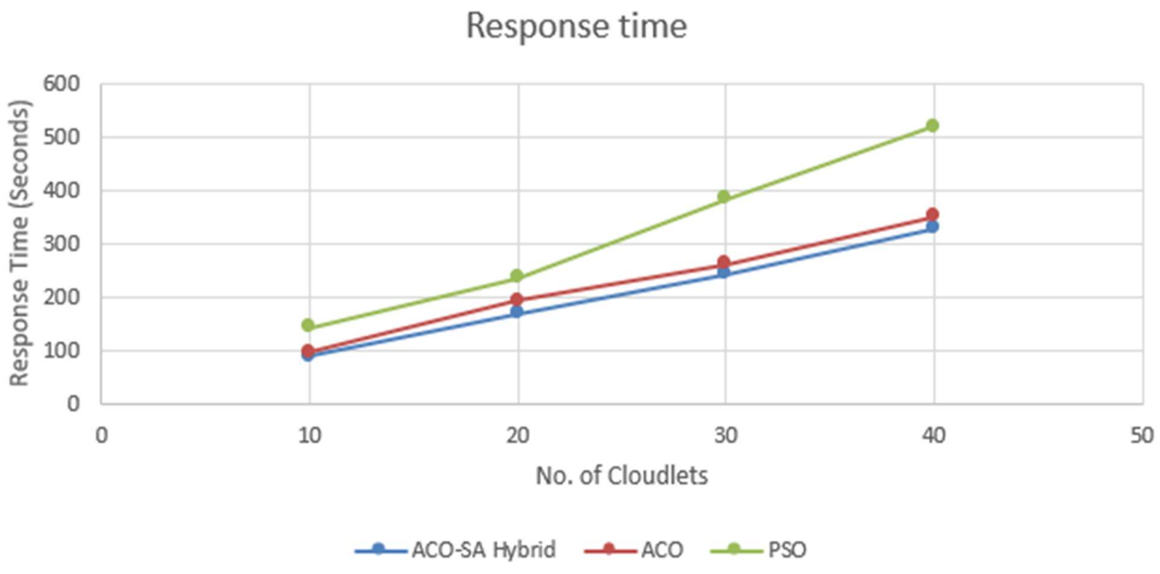


Figure 7. Response Time Analysis with varying cloudlets

Figure 7 shows the comparison of the Hybrid algorithm's response time with that of the ant colony and PSO methods as the cloudlets increase. The x-axis represents the number of cloudlets, and the y-axis represents the response time in seconds. It is analyzed that the proposed algorithm consistently achieves lower response times compared to the ACO and PSO algorithms with the increasing number of cloudlets. The hybrid ACO-SA algorithm performs better than ACO and PSO across all the scenarios. For 40 cloudlets, the response time is approximately 328 seconds, as compared to 349 s of ACO and 519 s of PSO, showcasing better load balancing and resource utilization.

6.2 Experiment / Case Study 2

The objective of this experiment is to evaluate the performance of the proposed hybrid Ant Colony Optimization-Simulated Annealing (ACO-SA) algorithm in terms of response time. The performance of the hybrid algorithm is compared with two other algorithms: Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO).

No of Datacenter: 1

No of VM: No of VMs are varied from 5, 7,10 to 15

Workload : The workload is kept constant , number of cloudlets (tasks) = 50

Table 2: Response Time Analysis

Number of VMs	Hybrid Algorithm	ACO algorithm	PSO Algorithm
5	432 seconds	487 seconds	622 seconds
7	294 seconds	308 seconds	551 seconds
10	218 seconds	229 seconds	468 seconds
15	147 seconds	172 seconds	299 seconds

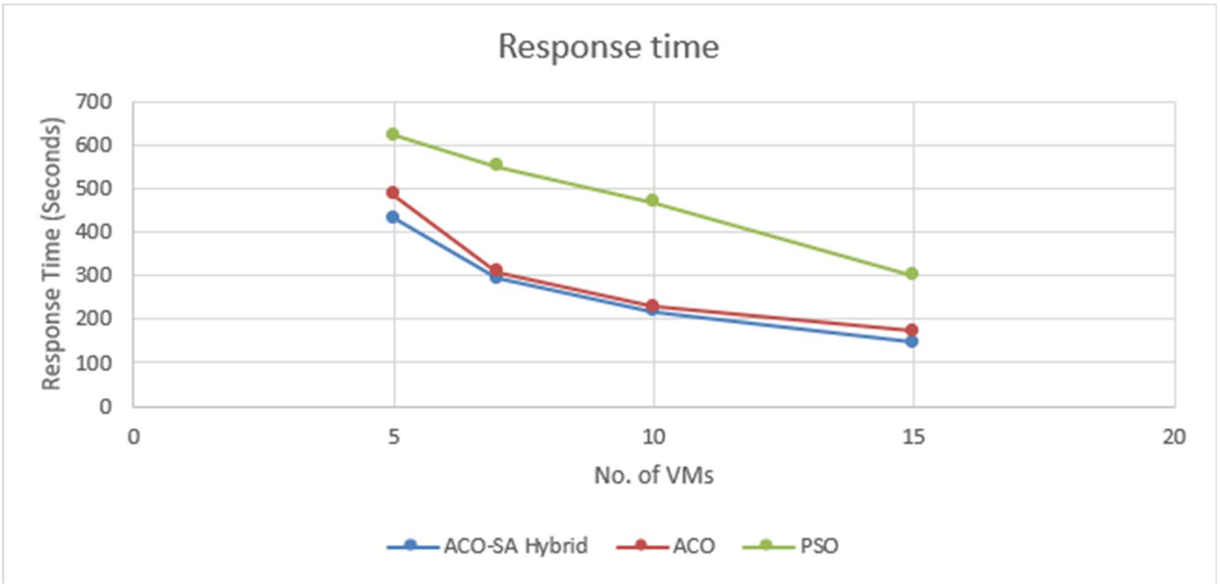


Figure 8 : Response time analysis with varying number of VMs

Figure 8 compares the Hybrid algorithm's response time for with that of the ant colony and PSO algorithms. The x-axis represents the number of VM's, and the y-axis represents the response time in seconds. It is analyzed that the proposed Hybrid algorithm consistently achieves lower response times compared to the ACO and PSO algorithms with the increasing number of Virtual machines. The hybrid ACO-SA algorithm performs better than ACO and PSO across all the scenarios. The hybrid algorithn has the Lowest response times across all configurations, starting from 432 seconds with 5 VMs and decreasing to 147 seconds with 15 VMs. ACO-SA Hybrid provides an overall average reduction in response time of approximately 8.79% compared to ACO, showcasing better load balancing and lower response time.

6.3 Experiment / Case Study 3

The aim of this experiment is to evaluate the resource utilization of the proposed hybrid ACO-SA algorithm in comparison to ACO and PSO. The analysis focuses on how each algorithm manages resource usage as the number of cloudlets(tasks) increases. No of VM varies from 5 to 15
number of cloudlets (tasks) is kept constant at 50, It can be seen that task are distributed evenly across all the virtual machines. As shown in figure 9, the resource utilization of proposed hybrid

algorithm is compared with existing optimization algorithms like ant colony optimization and PSO algorithms. The x-axis represents the number of VMs, and the y-axis represents the percentage of resources utilized.It is analyzed that proposed Hybrid algorithm show the highest overall utilization across all VM configuration as compared to ACO and PSO algorithms

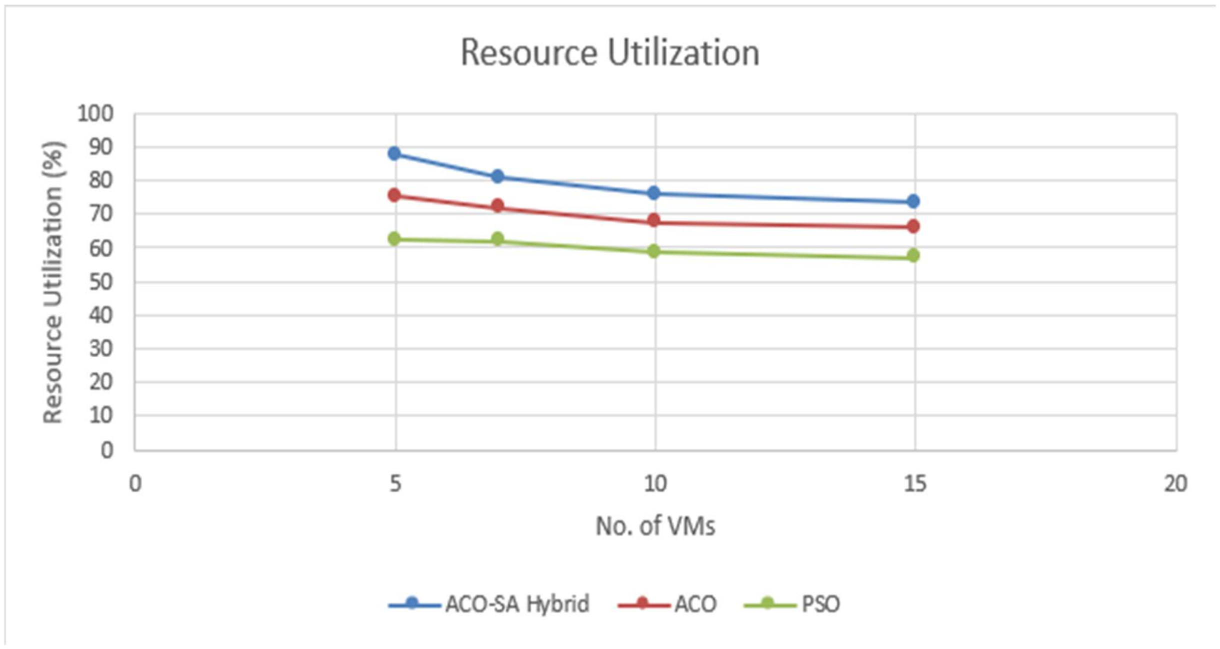


Figure 9. Resource utilization of ACO-SA with 50 cloudlets and 5, 7,10 and 15 VMs

For 5 VM, the resource utilization is approximately 88% compared to ACO‘s 77% and PSO‘s 62%, performs better compared to other algorithms and trend continues with increasing number of VM‘s indicating higher efficiency in managing the workload.

6.4 Discussion

One of the most important metrics for assessing the effectiveness of scheduling algorithms is response time. The Hybrid ACO-SA algorithm performs better than ACO and PSO with regard to the response times and resource utilization with varied configurations. With 5 VMs, ACO-SA achieves an average response time of 432s which is lower than that of ACO’s 487s and PSO’s 622s. This continues with the increasing number of VMs and cloudlets. Resource utilization is another important factor, ACO-SA with 5 VMs shows 88% compared to ACO‘s 77% and PSO‘s 62% and this continues with the increased number of VMs. This shows that VMs are effectively utilized, reducing idle time and improving performance. Reduced response times result in quicker task completion and more effective resource utilization, which also reduces energy consumption. The experiment results show that hybrid algorithms can efficiently handle varying workloads and VM configurations. ACO-SA algorithm consistently maintains a lower response time and high resource utilization which is achieved due to the pheromone-based exploration and probabilistic exploitation nature of the algorithm to balance the load across VMs.

7 Conclusion and Future Work

In this research ,a hybrid metaheuristic approach using Ant colony and simulated annealing algorithm is explored, a novel hybrid task scheduling algorithm for load balancing among virtual machines in cloud computing. The ability of ACO to update the pheromone and improve the solution was combined with SA to refine the solution space and find an optimum solution. The simulation of cloud data center and algorithms simulation was done using Cloudsim Toolkit. Response time and resource utilization were the evaluation parameters considered and were compared between the hybrid ACO-SA algorithm, traditional Ant Colony Optimization (ACO), and Particle Swarm Optimization (PSO) algorithms to evaluate the performance. The experimental results show that the proposed algorithm improves cloud computing load balancing by approximately 7-8% compared to the ACO, and PSO algorithms. In the future research integration of Machine learning techniques can be explored to further optimize the algorithm performance to improve load balancing.

References

Afzal, S. and Kavitha, G. (2019) ‘Load balancing in cloud computing – A hierarchical taxonomical classification’, *Journal of Cloud Computing*, 8(1), 22. doi: 10.1186/s13677-019-0146-7.

Gamal, M., Rizk, R., Mahdi, H. and Elnaghi, B.E. (2019) ‘Osmotic bio-inspired load balancing algorithm in cloud computing’, *IEEE Access*, 7, pp. 42735-42744. doi: 10.1109/ACCESS.2019.2907615

Sekaran, K., Khan, M.S., Patan, R., Gandomi, A.H., Krishna, P.V. and Kallam, S. (2019) ‘Improving the response time of M-learning and cloud computing environments using a dominant firefly approach’, *IEEE Access*, 7, pp. 30203-30212. doi: 10.1109/ACCESS.2019.2896253

Pang, S., Li, W., He, H., Shan, Z. and Wang, X. (2019) ‘An EDA-GA hybrid algorithm for multi-objective task scheduling in cloud computing’, *IEEE Access*, 7, pp. 146379-146389.

Javadi, S.A. and Gandhi, A. (2022) ‘User-centric interference-aware load balancing for cloud-deployed applications’, *IEEE Transactions on Cloud Computing*, 10(1), pp. 736-748

Mrhari, A. and Hadi, Y. (2019) ‘A load balancing algorithm in cloud computing based on modified particle swarm optimization and game theory’, in *2019 4th World Conference on Complex Systems (WCCS)*, Ouarzazate, Morocco, pp. 1-6. doi: 10.1109/ICoCS.2019.8930807.

Menaka, M. and Sendhil Kumar, K.S. (2024) ‘Supportive particle swarm optimization with time-conscious scheduling (SPSO-TCS) algorithm in cloud computing for optimized load balancing’, *International Journal of Cognitive Computing in Engineering*, 5, pp. 192-198. doi:

10.1016/j.ijcce.2024.05.002.

Pradhan, A. and Bisoy, S.K. (2020) ‘A novel load balancing technique for cloud computing platform based on PSO’, *Journal of King Saud University - Computer and Information Sciences*, 2(45), pp. 672-680. doi: 10.1016/j.jksuci.2020.10.016

Gao, J. (2024) ‘Simulation design of load balancing optimization for cloud computing data stream storage based on big data algorithms’, in *2024 Asia-Pacific Conference on Software Engineering, Social Network Analysis and Intelligent Computing (SSAIC)*, New Delhi, India, pp. 931-936. doi: 10.1109/SSAIC61213.2024.00188.

Devaraj, A.F.S., Elhoseny, M., Dhanasekaran, S., Lydia, E.L. and Shankar, K. (2020) ‘Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments’, *Journal of Parallel and Distributed Computing*, 142, pp. 36-45. doi: 10.1016/j.jpdc.2020.03.022.

Kruekaew, B. and Kimpan, W. (2022) ‘Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning’, *IEEE Access*, 10, pp. 17803-17818. doi: 10.1109/ACCESS.2022.3149955.

Muneeswari, G., Madavarapu, J.B., Ramani, R., Rajeshkumar, C. and Singh, C.J.C. (2024) ‘GEP optimization for load balancing of virtual machines (LBVM) in cloud computing’, *Measurement: Sensors*, 33, 101076. doi: 10.1016/j.measen.2024.101076.

Kumari, K.A., Soujanya, T., Alsalami, Z., Rohini, I. and Dhandayuthapani, B.V. (2024) ‘Fire Hawk Optimization based multi-objective dynamic load balancing in cloud computing’, in *2024 International Conference on Distributed Computing and Optimization Techniques (ICDCOT)*, Bengaluru, India, pp. 1-4. doi: 10.1109/ICDCOT61034.2024.10516057.

Abualigah, L., Hussein, A.M., Almomani, M.H., Zitar, R.A., Migdady, H., Alzahrani, A.I. and Alwadain, A. (2024) ‘Improved synergistic swarm optimization algorithm to optimize task scheduling problems in cloud computing’, *Sustainable Computing: Informatics and Systems*, 43, 101012. doi: 10.1016/j.suscom.2024.101012.

Singhal, S. et al. (2024) ‘Energy efficient load balancing algorithm for cloud computing using Rock Hyrax optimization’, *IEEE Access*, 12, pp. 48737-48749. doi: 10.1109/ACCESS.2024.3380159.

Khaleel, M.I. (2024) ‘Region-aware dynamic job scheduling and resource efficiency for load balancing based on adaptive chaotic sparrow search optimization and coalitional game in cloud computing environments’, *Journal of Network and Computer Applications*, 221, 103788. doi: 10.1016/j.jnca.2023.103788.

K R, P.K., GM, S., Yamsani, N., T.M., K.K. and Pani, A.K. (2023) ‘A novel energy-efficient hybrid optimization algorithm for load balancing in cloud computing’, in *2023 International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIIE)*, Ballari, India, pp. 1-5. doi: 10.1109/AIKIIE60097.2023.10390196.

Jena, U.K., Das, P.K. and Kabat, M.R. (2020) ‘Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment’, *Journal of King Saud University - Computer and Information Sciences*, 7(3), pp. 1159-1163.

Kaur, A. and Kaur, B. (2022) ‘Load balancing optimization based on hybrid Heuristic-Metaheuristic techniques in cloud environment’, *Journal of King Saud University - Computer and Information Sciences*. doi: 10.1016/j.jksuci.2019.02.010.

Hodžić, L. and Mrdović, S. (2023) ‘Using genetic algorithms for load balancing in cloud computing’, in *2023 XXIX International Conference on Information, Communication and Automation Technologies (ICAT)*, Sarajevo, Bosnia and Herzegovina, pp. 1-6. doi: 10.1109/ICAT57854.2023.10171261.

Haris, M. and Zubair, S. (2022) ‘Mantaray modified multi-objective Harris hawk optimization algorithm expedites optimal load balancing in cloud computing’, *Journal of King Saud University - Computer and Information Sciences*, 34(10, Part B), pp. 9696-9709. doi: 10.1016/j.jksuci.2021.12.003.

Liu, H. (2022) ‘Research on cloud computing adaptive task scheduling based on ant colony algorithm’, *Optik*, 258, 168677. doi: 10.1016/j.ijleo.2022.168677

Shi, Y., Hu, Z. and Lu, Z. (2021) ‘Optimized dynamic load balance method based on ant colony optimization algorithm’, in *2021 IEEE 9th International Conference on Computer Science and Network Technology (ICCSNT)*, pp. 70-73. doi: 10.1109/ICCSNT53786.2021.9615