International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V72I7P102

Volume 72 Issue 7, 11-23, July 2024

Original Article

A Comparative Analysis and Evaluation of Swarm-Adaptive Scheduling Methods in Cloud Environment

Anu Kadian[1], Kamna Solanki[2], Amita Dhankhar[3]

[1,2,3]*Department of Computer Science and Engineering, UIET, Maharshi Dayanand University, Rohtak, India.*

[1]*Corresponding Author : anukadian182315@gmail.com*

Received: 25 February 2024 Revised: 18 June 2024 Accepted: 01 July 2024 Published: 26 July 2024

Abstract *- Cloud computing provides a distributed environment to share resources and optimise task processing. Load balancing and makespan are the primary challenges of task scheduling that affect performance and user satisfaction in the cloud environment. An effective task scheduling method in a cloud environment can optimise resource utilisation and generate an effective sequence of task execution. The optimisation algorithms can be integrated within the task scheduler to map and execute the tasks effectively. In this paper, four swarm-based scheduling algorithms are implemented and compared in the cloud environment. The algorithms included in this work are Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC) algorithm, Ant Colony Optimization (ACO), and Grey Wolf Optimization (GWO) algorithms. These algorithms are simulated in different scenarios with 10 to 100 tasks. The comparative evaluation is conducted against Response Time, Makespan, Resource Utilization, and migration count parameters. The analysis results identified that the GWO achieved moreeffective results than ACO, PSO, and ABC algorithms.*

Keywords *- Cloud computing, Resource scheduling, Task scheduling, Resource allocation, Cloud environment.*

1. Introduction

In recent times, the cloud network has been scaled into almost every field and application. It removes the dependability of a customer on hardware, data management, and data security. Cloud architecture provides a customised business model that can be implemented for any firm or institute based on their requirement. High communication speed, instant, and real-time integration, and scalability are the factors that contributed to the phenomenal growth of cloud systems. It provides a global platform to share hardware, software, storage space, integrated libraries, security, etc[6]. The standard architecture of a cloud system with virtual and physical elements. This architecture is dynamic and scalable under economic and demand control. The virtualisation is integrated into the cloud environment to optimise the distribution. Above the hardware or physical layer, the virtual environment is set up to handle customer requests. In this virtual layer or environment, the Virtual Machines(VM) are defined with partial hardware support and sharing. Each cloud system contains M number of virtual machines, and each machine is defined with specific capacity, bandwidth, memory, and other processing components. The application requests are processed and executed by these virtual machines. VM is responsible for load balancing and scheduling. A task scheduler integrated cloud architecture is provided in Figure 1. In Figure 1, the internal cloud computing environment is defined as the top layer of this scheduler architecture. This top layer has a centralised control, and the physical devices are divided into M virtual machines. Each machine is defined with certain responsibilities and capabilities. End users exist at the lowest layer of this scheduler architecture. The task requests are generated at this layer with the specification of task requirements in terms of execution time, deadline, and memory requirement. The scheduler exists as the middle layer to maintainthe user requests and to allocate them to specific virtual machines.

This task of virtual machine mapping and ordering is the fundamental task of this middle layer. The centralised processing and control of this scheduling layer is done by two integrated components called VM Manager and Task Scheduler. VM manager contains complete information about the available virtual machine, the capacity of each virtual machine, and the free resources with each machine. The VM manager also maintains the statistical information related to processing and reliability.

This information is derived from the history of each virtual machine. The VM manager cancreate a new virtual machine based on the requirement and activate the VM on request of the scheduler. The tasksgenerated by a user when entered into this layer, these tasks are maintained in the form of a queue.

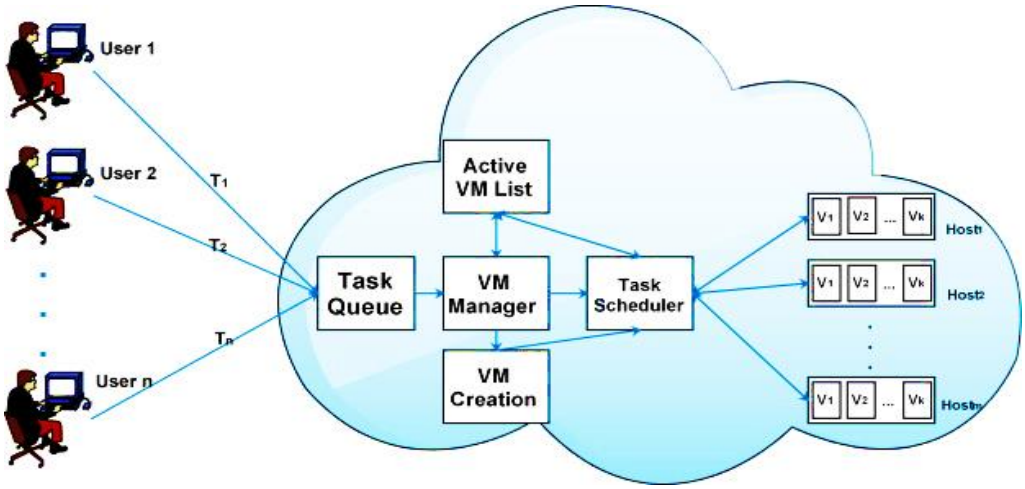

Fig. 1 Task scheduling architecture for cloud computing[6]

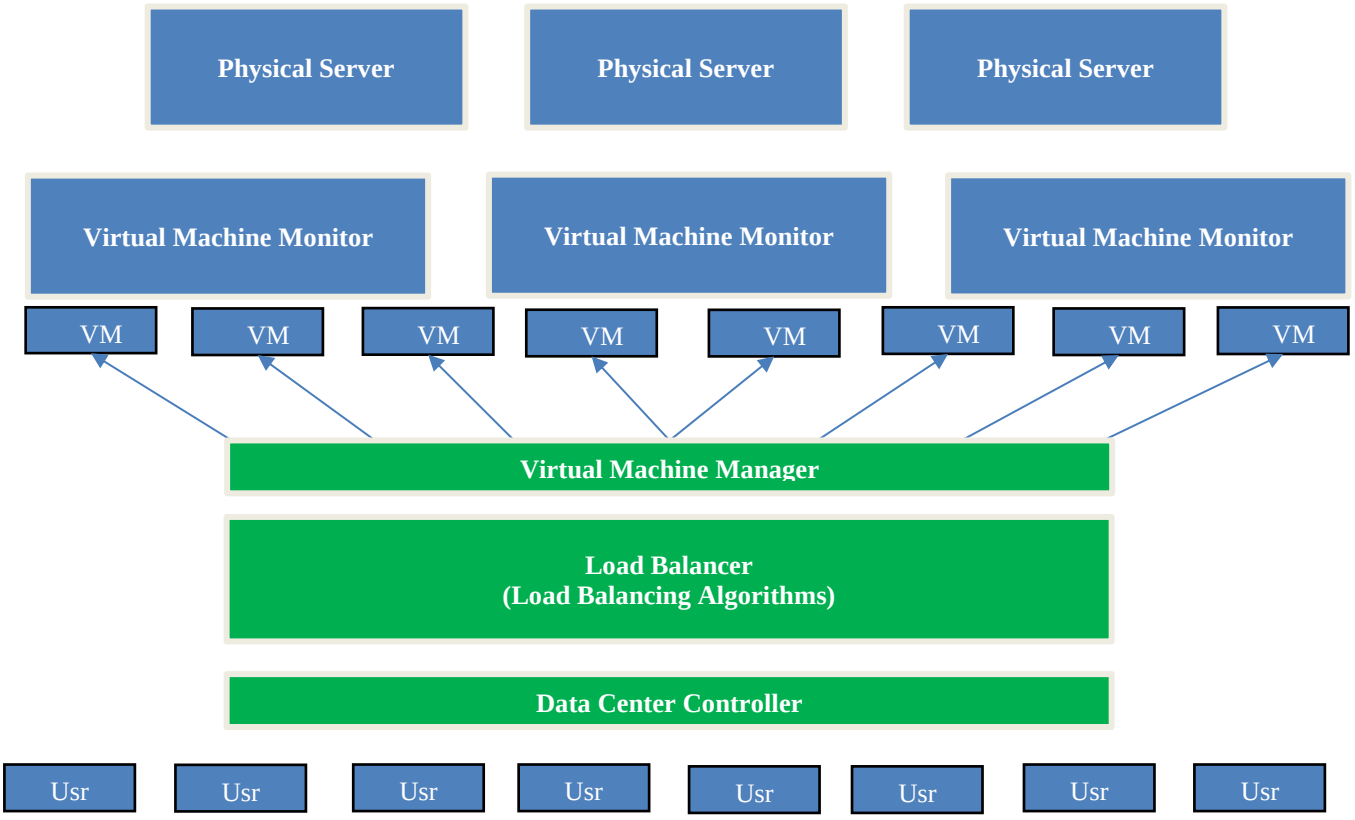

Fig. 2 Load balancing architecture in cloud environment

This task queue and available virtual machines are analysed and processed by the task scheduler for allocating the VMs and settingthe order of task execution. Any optimisation algorithm can be integrated within the task scheduler to optimise the behaviour of cloud computing architecture, load balancing, and task failure problems[6]. A cloud computing environment is a public environment that has limited resources, but the requests depend on the application. Even these limited resources are defined with restricted capacity. The capacity is defined in termsof memory, processing power, job queue, etc. In the case of heavy load situations, the performance and reliability of request execution degrades. Load balancing[8][17] is the primary constraint and objective of any scheduling algorithm. A standard architecture of load-balancing integrated cloud computing architecture is provided in Figure 2. In this architecture, the Load balancer algorithm lies between the virtual machine manager and the Data center controller. It works as an intermediate layer, and as the requests are generated and approved by the data center, theload balancer analyses the virtual machines and allocates the resource under

the load-balancing algorithm. The load balancing algorithm can be part of a resource allocator or scheduler to optimise the performance in real time.A load balancing algorithm includes some approaches and constraints related to transfer policy, location policy, information, and selection policy. The load balance algorithm[12][20] must adapt the information policy to retrieve the resource information and regular updates on it. The situation and location-specific information are required, and updation in it is a must because the network is dynamic in nature. The location policy identifies the source and processing request component for a request. The identification of ideal resources is done by location policy. Once the resource is identified, the request is allocated to it. But if the load is heavier on that few requests can be transferred to other resources. Another associated policy with load balancing is the selection policy.

This policy identifies and categorises the available requests. It identifies the jobs that are facing starvation, or that can be migrated to other resources. The constraint-specific rules are defined to categorise the tasks under priority, delay, or process time parameters. Various distributed, centralised, and hierarchical load balancing algorithms were integrated within the scheduling algorithm to handle the overload and under situations[5][21]. The objectiveof these algorithms was to avoid task failure, task delay, and starvation situations in the environment.

This paper identified an adaptive heuristic swarm optimisation method to handle various issues of scheduling algorithms. The comparative study is provided against 4 existing swarm optimisation algorithms. The algorithmic process and functional behaviour of each algorithm are defined. Each of the algorithms is experimented with in an identical cloud environment, and evaluation is conducted. In this section, a brief introduction to cloud computing is provided respectively to the requirement of scheduling and resource allocation algorithms. The issues and challenges are identified in these algorithms, and associated features and constraints are defined. The load balancing and scheduling algorithms and their significance are also described. In section 2, the existing studies and investigations of resource allocation, scheduling, and load balancing problems are discussed respectively to the problem handled and solution discussed.

Various heuristic and optimisation algorithms for task scheduling in the cloud environment are discussed with their capabilities and outcomes. In section 3, a study on four popular swarm optimisation algorithms is provided. The algorithmic process, features, and behaviour of these algorithms are discussed using flow charts. In section 4, the experimental environment and comparative results are provided to analyse the performance of swarm-based scheduling algorithms. In section 5, the conclusion andfuture scope of this presented work are discussed.

2. Literature Review

Cloud computing faces various challenges heavy load, execution delay, and scheduling issues. The researchers investigated various heuristic, rue-based, mathematical, and optimisation adaptive models to handle variousissues of cloud computing. These issues consist of high makespan, higher failure rate, low throughput, and heavy load. Resource utilisation and resource adaptation is also the challenge for this environment. In this section, various studies that are targeting these issues and methods are explored. Ebadifard et al.[4] dynamic scheduling methods for handling the overloading problem in a cloud computing environment. An effective task scheduling and resource allocation algorithm was defined to minimise the makespan and to improve the reliability of the environment. This method ensured the fair distribution of workloadamong virtual machines. The author defined a honeybee adaptive load balancing and scheduling method to handleload issues in the cloud environment. Shafiq et al.[9] used the SLA parameters and deadline within the load-balancing algorithm.

The algorithm was designed to optimise the resource allocation and load balancing under QoS consideration. The author used the VM priority and QoS measures for optimising the dynamic behaviour of LBA. The results identified a significant reduction in makespan and execution time. Neelima et al.[11] used an adaptive dragonfly algorithm to handle the heavy load situation in a cloud environment. This algorithm providedthe solution to the NP-hard problem for effective load balance. The author combined the dragonfly algorithm withthe firefly algorithm to optimise the behaviour of the scheduling algorithm.

This multiobjective algorithm was based on processing cost, completion time, and cost. The comparative results identified that the proposed model ensured effective load balancing in comparison with state-of-art methods. Negi et al.[13] combined the supervised and unsupervised learning approaches within the CMODLB (Clustering-based Multiobjective Dynamic Load balancing) technique. The neural network model is integrated within the model to identify the load situation in the network.

The Bayesian optimisation with KMeans clustering was applied to optimise the scheduling with effective resource utilisation. The swarm adaptive solution was generated in this method to achieve robustness against different cloud criteria. This model achieved better resource utilisation and scheduling against Max Min and round-robin methods. Tong et al.[18] provided a deep reinforcement learning-based model to handle issues of task scheduling. The service level agreement with load balancing was targeted to avoid task rejection rate. ThisDRL method predicts the chances of VM violation, and accordingly, a request is assigned. This model reduced thetask rejection rate and improved the performance of cloud computing.

Mapetu et al.[3] identified the functional problem of the cloud computing environment. Scheduling in this algorithm with a heavy load can become an NP-hard problem because of various constraints, including high resource utilisation rate, low scheduling time, low make space, and execution cost. The author proposed an effective binary version of Particle Swarm Optimization (PSO) to reduce the complexity and to improve the reliability and completion ratio. The proposed optimised algorithm achieved better load-balancing results than state-of-art methods. Ragmani et al.[7] considered the response time and heavy load problems of cloud computing. The author integrated the fuzzy rules within Ant Colony optimisation to optimise the functioning of the scheduling algorithm. The results identified a significant improvement in response time for different scenarios. Hung et al.[10] proposed an improved Max-min scheduling algorithm to reduce the request execution time.

The learning-based clustering method was incorporated to improve resource utilisation. This method was compared against theMin-min, Max-Min, and round-robin methods and claimed the lesser completion time. Priya et al.[14] presented a fuzzy-based multidimensional resource scheduling algorithm for optimising the utilisation of cloud infrastructure. The proposed model achieved fair and effective load balancing by engaging a multidimensional queuing system. This model reduces the latency and improves the utilisation of resources in real-time. The simulation results verified a significant improvement of 7% in resource scheduling and 35.5% in response time in comparison with state-of-the-art methods.

Gamal et al.[16] proposed a hybrid metaheuristic technique by combining osmotic behaviour with the bio-inspired algorithm. This osmotic behaviour is adaptive to dynamic virtual machines and requests migration based on load. The author used ant colony and artificial bee colony optimisation algorithms for optimising the dynamic behaviour of request processing. This method achieved better results against heavy load situations. The results identified a reduction in energy consumption and an improvement in quality of service. Patel et al.[19] handled the load-balancing problem by combining the honeybee algorithm with a weighted round-robin approach. This model improved the system performance and reduced the task completion time.

3. Optimisation Methods Adaptive Task Schedulers

3.1. Artificial Bee Colony Algorithm

Artificial Bee Colony (ABC)[1] is a swarm-based optimisation technique that adapts the functionality of foragingof honeybees. In this algorithm, honeybees are the swarm particles that identify the local and global optimum values. This algorithm defines a food source for honeybees with different constraints like food amount, distance and the way to get that food easily from the nectar. The algorithm is performed for different bee types, where different constraints and problems are handled at different levels by different bees. In this algorithm, the employed bee, unemployed bee, outlooker bee and scout bee are defined as the functional unit with different roles. In thistask scheduling algorithm of cloud computing, these bees are applied to handle different constraints and are featured to optimise the resource. In this algorithm, the employed bee contains constraint information about the cloud network, including the number of tasks, capacity of resources, load on resources, etc. The unemployed bees are responsible for processing this information and making some decisions about allocation and ordering. Onlooker is the type of unemployedbees that collect information from employed bees and perform analysis over it. Scout is another bee form that identifies the possible solutions based on the available constraints. The algorithmic process of the bee colony-based scheduling algorithm is shown in Figure 3.

Figure 3 shows the detailed functioning of the scheduling process defined within a cloud environment using the ABC algorithm. This algorithm is implemented over the environment where the network is defined with N number of resources and M number of requests generated by the users. These requests are processed by the ABC-integrated scheduler for effective resource allocation and scheduling. In this algorithm, the bees are distributed as controller and functional devices. The employed bee collects are the features and constraints of every user requestand available resources. The request information includes the request time, deadline, process time, type of process,etc. The resource information includes load capacity, processing capability, memory size, etc. The algorithm is defined for a maximum number of iterations, or the objective does not meet. The scout bee identifies the possible solution of allocation of requests on different resources. The onlooker bee processes this information and analyses the possible solution against the fitness rule and critical constraints. The onlooker bees perform the load, deadline and failure rate analysis to validate the obtained solution against criticality constraints and fitness rule. If it satisfies all rules and constraints, then it is compared against the existing solution. If the new solution is better than the existing solution, then it replaces that global solution. The process is performed till themaximum iterations, and after the iterations, the final solution is returned as output. The comparative evaluation and significance of this algorithm are provided in Section 4.

3.2. Grey Wolf Optimization Algorithm

Grey Wolf Optimization[2] algorithm adapts the capability of wolf hunting to generate the optimised solution. Thealgorithm acquired the social behaviour of wolves and their living and hunting in groups and families. This optimisation algorithm is controlled and processed by the alpha (α) wolf, which is identified as the controller or leader wolf.

The complete decision-making, instructions, and constraint mapping are performed by this leader. All other elements or wolves follow the same rules of sleeping, awaking, hunting, group making etc. Another kind of wolf included in this algorithm is the beta wolf, which represents the second level in this wolf hierarchy. It is defined as an intermediator and controller between the alpha wolf and lower-level wolves. It's setup responsibilities for low-rank wolves. Generate a conclusive analysis to replace or change the position of lower-level wolves. The decisions are taken bythe alpha wolves based on the information collected by beta wolves. The third rank wolves are delta wolves, which are functional wolves with different wolves, including hunting, guards, spies and supporters. These are distributed over the region and border to handle different situations. The constraints and limits for each wolf are defined to handle real-time situations. These wolves also take care of weak and sick wolves. The responsibility sharing, changing position and promotion of the wolf can be done based on the capabilities of the wolf.

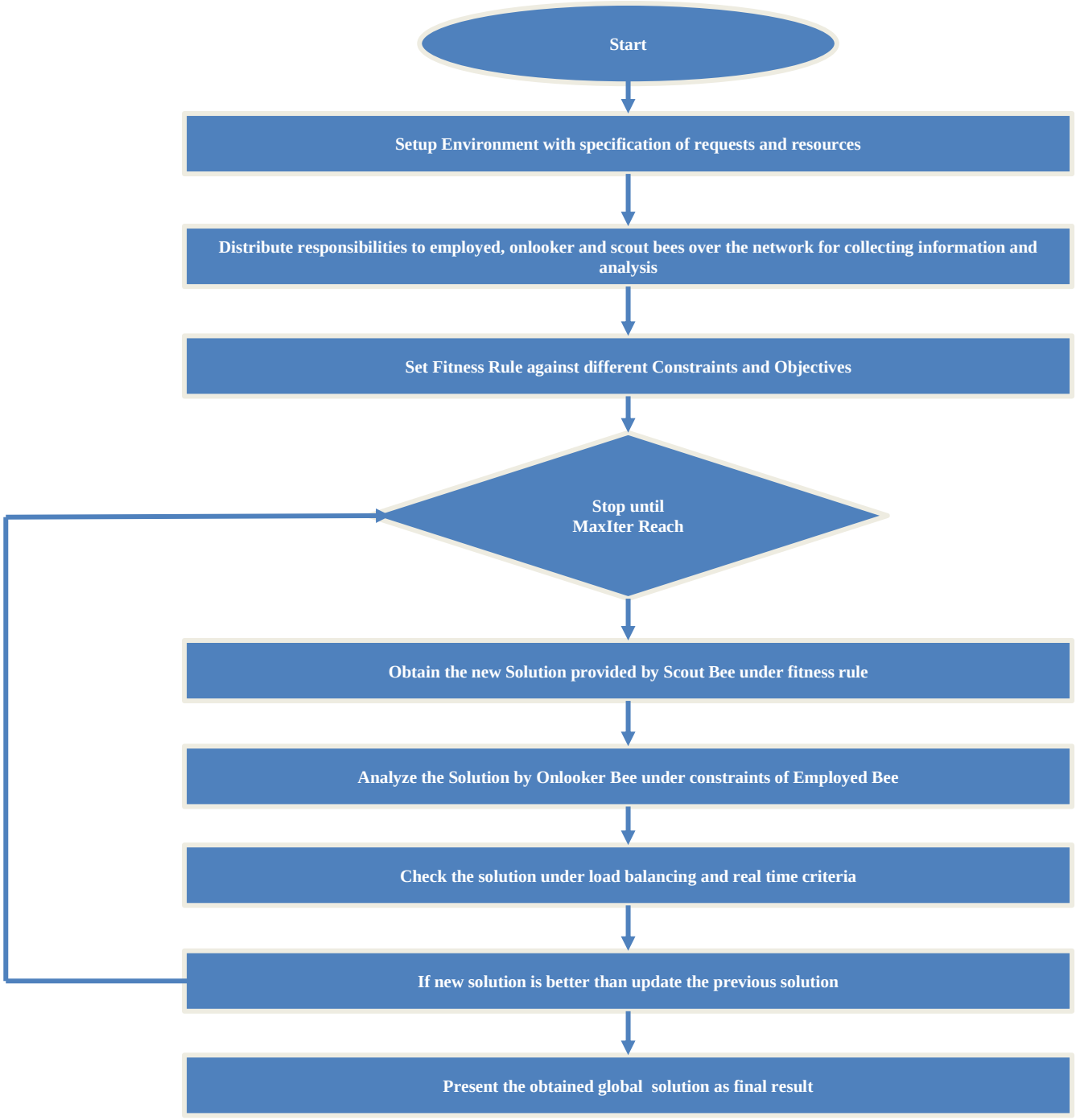

Fig. 3 Flow chart of ABC integrated task scheduling

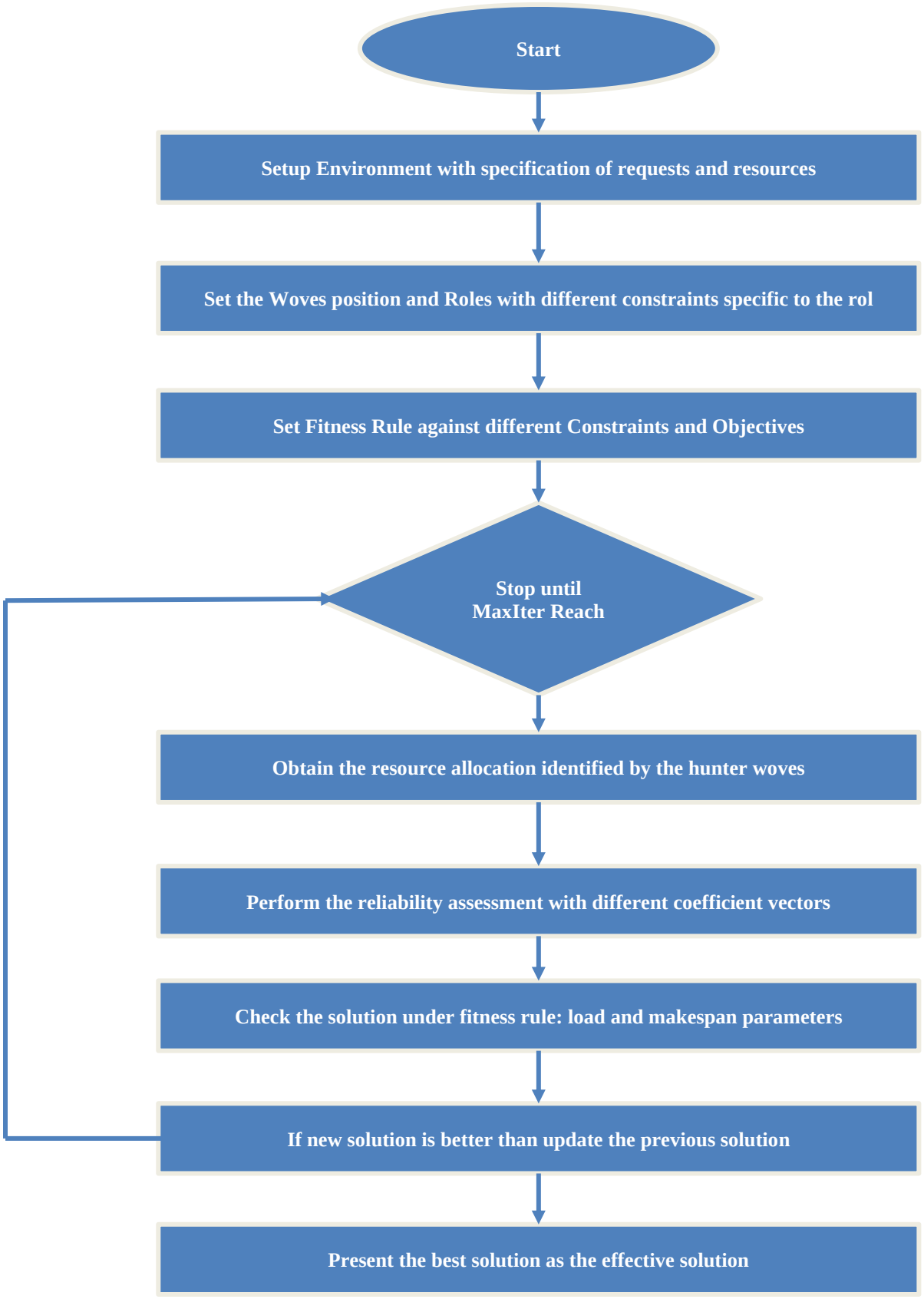

Fig. 4 Flow chart of GWO integrated task scheduling

In this research work, traditional GWO is taken and implemented in a cloud computing environment for optimising the resource allocation and scheduling process. The fitness rule and the constraints for the algorithms are defined by the single alpha wolf in this algorithm.

The coverage, energy and capabilities of different elements are analysed to categorise them as beta, delta and omega wolf. The beta defines the responsibilities of delta wolves or nodes to fulfil a specific objective. The delta wolves are dedicated to that particular objective and analyse the resource usage and allocation respective to that objective only.

The objectives are defined in terms of load balancing, delay time restriction, and optimising the functional behaviour of the cloud computing schedular. When the hunting processing in scheduling architecture begins, the grey wolf keeps himself at the center to control thefunctioning. With each iteration it validates the positioning and responsibilities of beta wolves.

Beta wolves assign and collect the performance of delta and omega wolves. These are functional wolves defined with specific responsibilities of resource mapping, resource allocation, resource validation, ordering setup etc. They check for the high-level constraints and update the current positions to the beta wolf.

Beta wolves collect and analyse this information and update it to the alpha wolf. Finally, the decision about the positional change responsibility change is taken by the alpha wolf. With each iteration, a lot of positional changes and responsibility changes are performed, and with each configuration, a new solution is obtained.

This obtained solution of allocated resources and ordering is compared with the global solution. If the new solution is better than the achieved best solution, then the best solution will be replaced by the new solution.

3.3. Particle Swarm Optimisation (PSO)

PSO[15] is an adaptive swarm-based metaheuristic algorithmic approach introduced by Kennedy and Dberheart. PSO can be integrated within the scheduling algorithm to optimise resource utilisation and to improve the performance of the scheduling algorithm. This algorithm distributes the swarm particles over the resources to obtain the VM features.

The particle analyses each of the resources in terms of their capacity, usage history, limitations and constraints. It identifies the availability of resources and the total time for that a resource can be utilised. Each of the particles is defined with some mobility and dynamic behaviour to perform the computation. Once the users generate the requests, these requests are processed by the PSO integrated VM manager.

This manager takes the decision about resource allocation and setting up the order of request processing. The PSO is integrated into the scheduling and resource allocation algorithm. After collecting the information fromall swarm particles, it performs a mapping of requests to different resources. This algorithm also decides the order of request processing. Now, it checks the cost delay and makes span based multiparameter based analysis to identify the effective sequence of request processing.

The obtained solution is considered the local best. Now, this solution is compared with other possible solutions that are obtained with each iteration and provided by the computation of each swarm particle. If the solution is better than the previous solution, then update this global best solution with this local best solution. This process is repeated till the maximum iterations are not reached or thedesired objective is not achieved. The flowchart of this algorithm is provided in Figure 5.

3.4. Ant Colony Optimization (ACO)

ACO[22] algorithm is based on ant behaviour and provides a powerful technique to solve NP-complete problems.This algorithm is inspired by the food searching and collecting method of ants. In this method, ants perform the food search using a pheromone trail that some other ant leaves behind.

In the scheduling algorithm, ACO can be integrated to identify the most effective resource under different parameters. These parameters can be made span, response time or resource scheduling. The multiobjective fitness function can be defined within ant colony optimisation to control resource hunt. In this optimisation approach, ants are distributed randomly, and they search for the resource randomly.

As the ant identifies a resource, the fitness rule is performed to perform the required validation. After satisfying the fitness rule, the task is allocated to the resource. With the identification of resources, the pheromone trail is split on the path. This pheromone trail of previous resources is tracked by the ants. In this way, the majority of ants move towards the best solution obtained in terms of resource allocation and scheduling.

Different ants search the multiple scheduling options, but the best solution is followed by the ant. The best solution depends on the amount of pheromones and the adaptation of the fitness rule.

This process is interaction-based with each iteration; the ant movement is performed on a virtual machine. The process is repeated till all the tasks are not mapped to a particular resource. The functional process of ACO based task scheduler is provided in Figure 6.

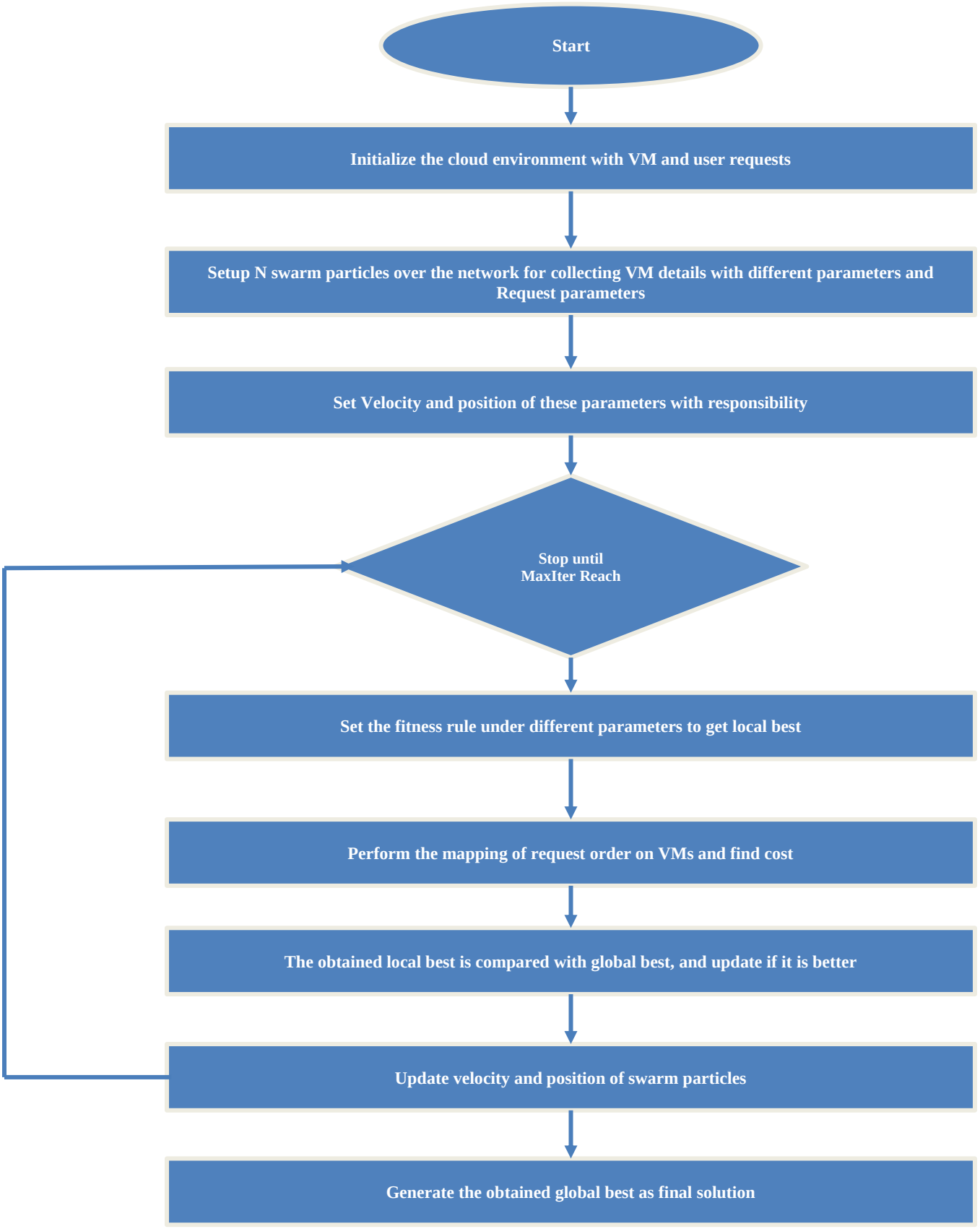

Fig. 5 Flow chart of PSO for task scheduler in cloud computing

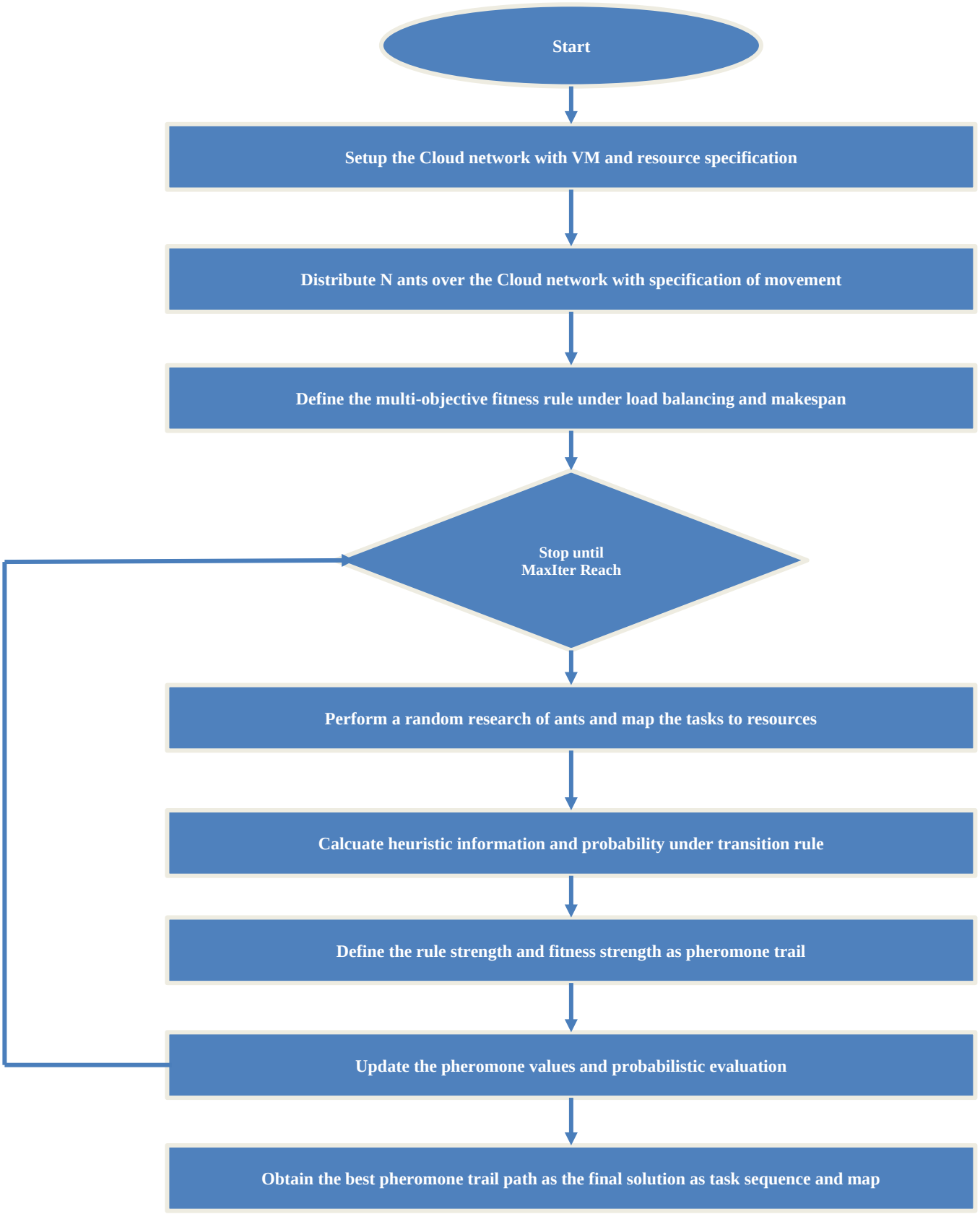

Fig. 6 Flow chart of ACO for task scheduler in cloud computing

4. Results and Discussions

In this paper, four swarm-optimised scheduling models are explored and compared respectively to the objective and real time situations. These algorithms are simulated under different scenarios. The virtual environment is built with fixed specifications of resources and a varied number of requests. The simulation environment is setup with 512 MB RAM and 2500 Mips processing speed. The hard disk capacity is 1 GB, and the bandwidth is 400 Mbps. The number of virtual machines is 4, and number of requests lies between 10 and 100.

Each swarm algorithm is applied for a maximum of 100 iterations and with multiobjective evaluation, as described in the previous section. Each of the requests is generated with the specification of request generation time, request size, memory requirement, and task criticality.

The analysis of the work is done in terms of makespan, response time, resource utilisation, and migration count parameters. Makespan is the efficiency parameter to evaluate the performance of an algorithm. Makespan is the time taken by the cloud computing model to execute the task. The lesser makespan confirms the efficiency of an algorithm.

Figure 7 provides a comparative evaluation against the makespan parameter. In this line graph, the number of tasks is provided at the x-axis and the y-axis shows the makespan. The line graph shows that the makespan result of ACO is the worst, and the GWO algorithm provides the best results. As the number of tasks is increased in a scenario, the makespan will be increased. In heavy load situations, the migrations can occur in the system. Task switching and substitution can delay the execution process, and the overall makespan will be increased. The average makespan of all scenarios is 17.3 sec for PSO, 14.41 for ABC, 20.5 for ACO and 11.8 for GWO. The results show that the GWO achieved better efficiency with the least average make span.

Another efficiency parameter considered in this evaluation is Response time. Response time is the difference between the actual execution time and request time. A higher response time is defined as a higher delay in the request processing. Figure 8 presents a comparative assessment of the response time measure. The result shows that the ACO and PSO algorithms have the least performance with higher response time. The results obtained by ABC and GWO algorithms are effective with lesser response time and delay.

Resources are the cloud computing component including virtual machine and the physical structure. Resource utilisation is the measure that confirms the allocation and usage of resources without keeping them in an ideal state. The utilisation of resources is evaluated in terms of ratio. The higher utilisation of resources confirms the continuous execution of tasks in the cloud environment.

It confirms the effective resource allocation without wastage of resources. Figure 9 provides the comparative analysis of the presented algorithms against this parameter. The resource utilisation obtained by the GWO algorithm is 0.53 for 10 tasks, 0.79 for 50 tasks, 0.93 for 80 tasks and 0.96 for 100 tasks. It shows that as the number of tasks are increasing, the better utilisation of resources is achieved. The line graph shows that the resource utilisation in GWO is effectively better than ABC, PSO and ACO algorithms.

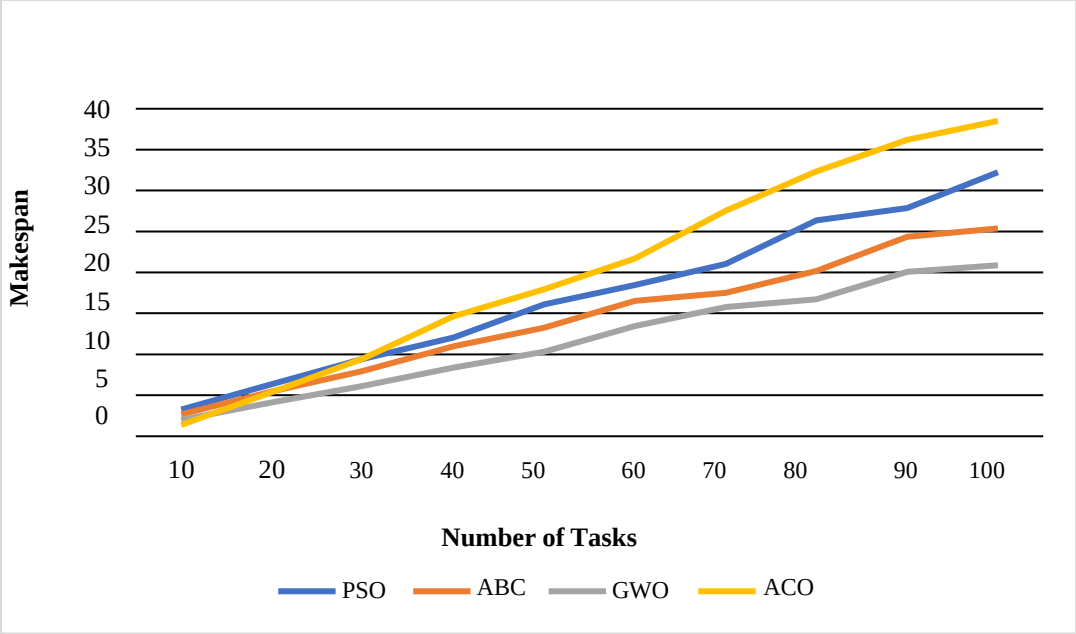

Fig. 7 Makespan analysis

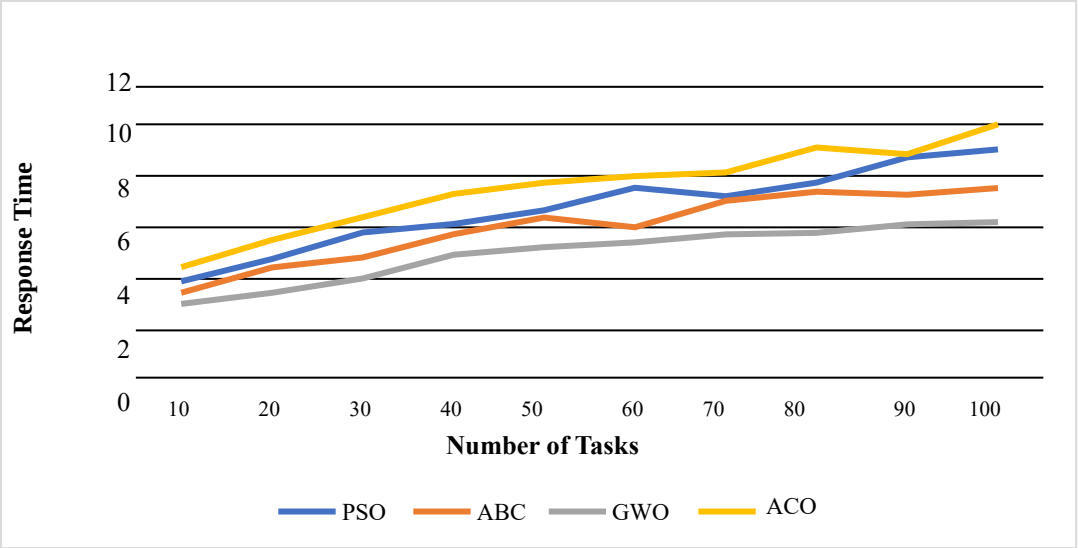

Fig. 8 Response time analysis

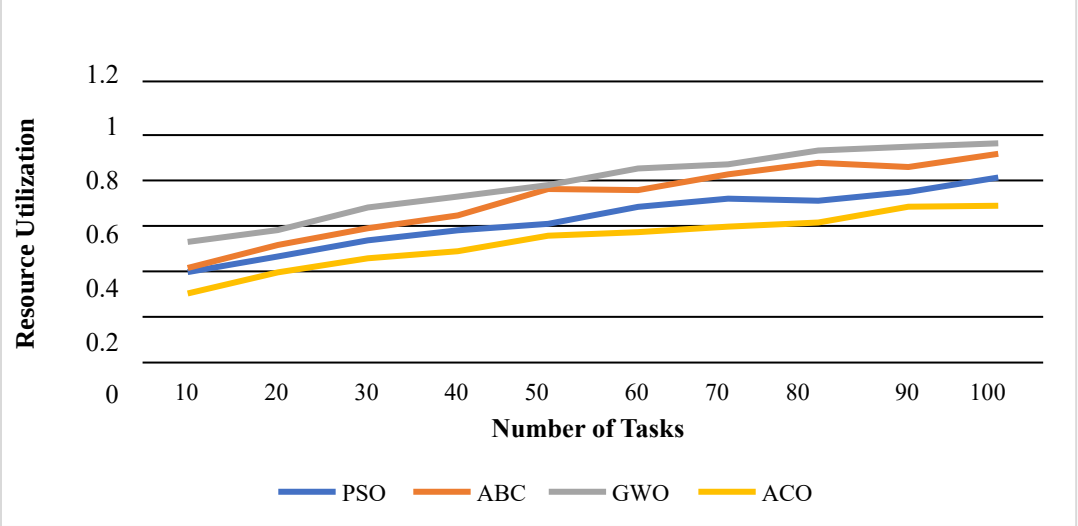

Fig. 9 Resource utilisation analysis

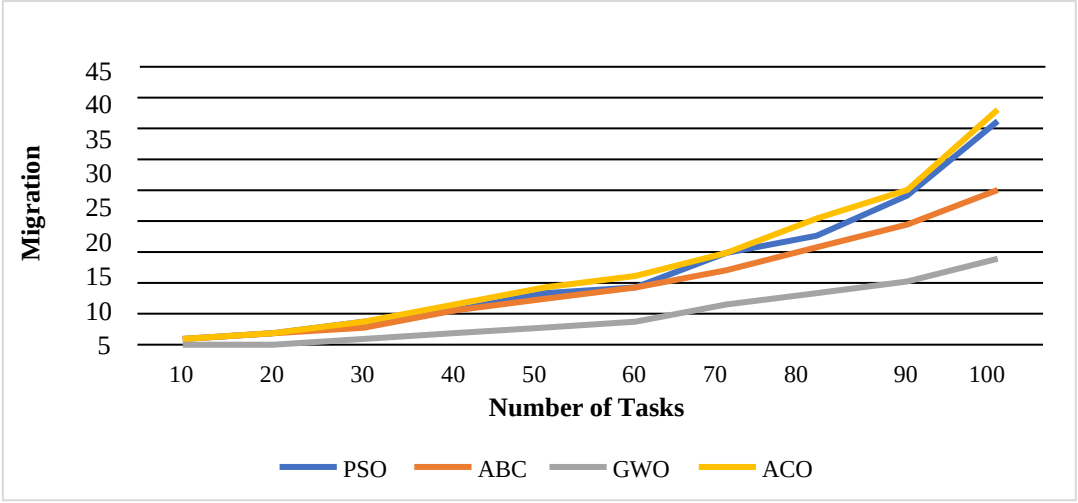

Fig. 10 Migration count analysis

Another evaluation parameter considered to compare the performance of projected algorithms is migration count analysis. In a scheduling algorithm, the VM is allocated to a requested task. But if the VM is not capable of executing it within the deadline, then the task can be switched to another virtual machine. This migration reduces the chances of task failure and improves the reliability of the system. However, as the migrations increase in a cloud environment, it can affect the performance and increase the processing delay. The efficiency and reliability can be affected by higher migration. In heavy load situations, the migrations can be increased. Figure 10 shows that the number of migrations is maximum for ACO and PSO algorithms. The numer of migrations is the least for the GWO algorithm thatachieved the most promising results.

5. Conclusion

Cloud computing is today's requirement to distribute services without setting up a physical environment. But, as the number of tasks is increased in such an environment, it can result from higher response time and execution failures. Various resource allocation and scheduling algorithms are available to handle these key issues. Various swarm-based and evolutionary optimisation algorithms were also integrated within the scheduling approach to improve the problem of the cloud environment. The researchers used swarm-based schedulers to gain performance and reliability gain. This article has provided a comparative analysis of four popular swarm-based scheduling algorithms. The paper has provided the algorithmic details and functional effectiveness of these algorithms. These algorithms are implemented with different scenarios in a simulation environment. The analysis is done against response time, makespan, resource utilisation and migration count parameters. The simulation results identify that the ACO and PSO are the worst performer scheduling algorithms with higher makespan, response time and migration count. The resource utilisation is also maximum up to .69 for ACO and .81 for PSO.At the same time, the GWO-based scheduler is the best performer that improves the resource utilisation and performance of task processing in the cloud computing environment. The maximum resource utilisation achieved by the algorithm is .96, with effective response time. The algorithm has further scope to improve to gain better resource utilisation and performance. This work can be improved in future to optimise the performance of the scheduling algorithm.

References

[1] Arif Ullah et al., "Artificial Bee Colony Algorithm Used for Load Balancing in Cloud Computing," *IAES International Journal of Artificial Intelligence*, vol. 8, no. 2, pp. 156-167, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[2] Seyed Salar Sefati, Maryamsadat Mousavinasab, and Roya Zareh Farkhady, "Load Balancing in Cloud Computing Environment Using the Grey Wolf Optimization Algorithm Based on the Reliability: Performance Evaluation," *The Journal of Supercomputing*, vol. 78, no. 1, pp. 18-42, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[3] Jean Pepe Buanga Mapetu, Zhen Chen, and Lingfu Kong, "Low-Time Complexity and Low-Cost Binary Particle Swarm Optimization Algorithm for Task Scheduling and Load Balancing in Cloud Computing," *Applied Intelligence*, vol. 49, pp. 3308-3330, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[4] Fatemeh Ebadifard, Seyed Morteza Babamir, and Sedighe Barani, "A Dynamic Task Scheduling Algorithm Improved by Load Balancing in Cloud Computing," *2020 6th International Conference on Web Research*, Tehran, Iran, pp. 177-183, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[5] Abderraziq Semmoud et al., "Load Balancing in Cloud Computing Environments Based on Adaptive Starvation Threshold," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 11, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[6] Sambit Kumar Mishra, Bibhudatta Sahoo, and Priti Paramita Parida, "Load Balancing in Cloud Computing: A Big Picture," *Journal of King Saud University-Computer and Information Sciences*, vol. 32, no. 2, pp. 149-158, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[7] Awatif Ragmani et al., "An Improved Hybrid Fuzzy-Ant Colony Algorithm Applied to Load Balancing in Cloud Computing Environment," *Procedia Computer Science*, vol. 151, pp. 519-526, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[8] Amrita Jyoti, Manish Shrimali, and Rashmi Mishra, "Cloud Computing and Load Balancing in Cloud Computing-Survey," *2019 9th International Conference on Cloud Computing, Data Science & Engineering*, Noida, India, pp. 51-55, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[9] Dalia Abdulkareem Shafiq et al., "A Load Balancing Algorithm for the Data Centres to Optimize Cloud Computing Applications," *IEEE Access*, vol. 9, pp. 41731-41744, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] Tran Cong Hung et al., "MMSIA: Improved Max-Min Scheduling Algorithm for Load Balancing on Cloud Computing," *Proceedings of the 3rd International Conference on Machine Learning and Soft Computing*, Da Lat Viet Nam, pp. 60-64, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[11] P. Neelima, and A. Rama Mohan Reddy, "An Efficient Load Balancing System using Adaptive Dragon Fly Algorithm in Cloud Computing," *Cluster Computing*, vol. 23, pp. 2891-2899, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[12] Dalia Abdulkareem Shafiq, N.Z. Jhanjhi, and Azween Abdullah, "Load Balancing Techniques in Cloud Computing Environment: A Review," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 7, pp. 3910-3933, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] Sarita Negi et al., "CMODLB: An Efficient Load Balancing Approach in Cloud Computing Environment," *The Journal of Supercomputing*, vol. 77, pp. 8787-8839, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[14] V. Priya et al., "Resource Scheduling Algorithm with Load Balancing for Cloud Service Provisioning," *Applied Soft Computing*, vol. 76, pp. 416-424, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[15] Arabinda Pradhan, and Sukant Kishoro Bisoy, "A Novel Load Balancing Technique for Cloud Computing Platform Based on PSO," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 7, pp. 3988-3995, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[16] Marwa Gamal et al., "Osmotic Bio-Inspired Load Balancing Algorithm in Cloud Computing," *IEEE Access*, vol. 7, pp. 42735-42744, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[17] Pawan Kumar, and Rakesh Kumar, "Issues and Challenges of Load Balancing Techniques in Cloud Computing: A Survey," *ACM Computing Surveys*, vol. 51, no. 6, pp. 1-35, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[18] Zhao Tong et al., "DDMTS: A Novel Dynamic Load Balancing Scheduling Scheme Under SLA Constraints in Cloud Computing," *Journal of Parallel and Distributed Computing*, vol. 149, pp. 138-148, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[19] Karan D. Patel, and Tosal M. Bhalodia, "An Efficient Dynamic Load Balancing Algorithm for Virtual Machine in Cloud Computing," *2019 International Conference on Intelligent Computing and Control Systems*, Madurai, India, pp. 145-150, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[20] Mohammed Ala'anzy, and Mohamed Othman, "Load Balancing and Server Consolidation in Cloud Computing Environments: A Meta-Study," *IEEE Access*, vol. 7, pp. 141868-141887, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[21] Muhammad Asim Shahid et al., "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. [CrossRef] [Google Scholar] [Publisher Link]

[22] Ashish Gupta, and Ritu Garg, "Load Balancing Based Task Scheduling with ACO in Cloud Computing," *2017 International Conference on Computer and Applications*, Doha, Qatar, pp.174-179, 2017. [CrossRef] [Google Scholar] [Publisher Link]