[1]Saravanakumar. K
[2]Anand Viswanathan
[3]Ravikumar. K
[4]Reka. M

Reinforcement Learning Based Metaheuristic Algorithm For Optimized Load Balancing In Cloud Environment

Abstract:- In cloud computing, workload balancing remains a difficult issue, as overburdened or underburdened servers are capable of causing issues with computation speed or even cause an entire network to collapse, subsequently this is a scenario that is never supposed to occur when using the cloud. Therefore, in order to avoid these issues, the system should be load balanced—that is, allocate job across each of the accessible computing assets by taking into account an acceptable schedule of utilization. The Virtual Machines are supposed to be used effectively, in response to the distributed load mechanism. In this work, a multifaceted load balancing optimization based on the Enhanced Firefly algorithm (EFA) with a Partial Order Markov Decision (POMD) process algorithm is proposed as an autonomous job allocation strategy in cloud computing. In order to overcome the limits of concurrent considerations, the suggested solution seeks to maximise VM productivity, streamline task allocation and the use of resources, and provide load balancing amongst virtual machines based on Time to Finish, expenditure, and consumption of resources. Using CloudSim, the performance study of the suggested approach was contrasted with the load balancing algorithms now in use in datasets like Google Cloud Jobs (GoCJ). According to the trial results, the suggested POMD-EFA strategy performed better than the other algorithms in terms of decreasing the Time to Finish, minimizing expenses, and increasing the efficiency in the application of resources.

Keywords: Reinforcement learning, Markov decision process, Firefly algorithm, load balancing, cloud computing

I. INTRODUCTION

With cloud computing, service firms can instantaneously make a variety of configurable computer assets, such as storage, networks, software applications, and products and services, available to clients over an easily accessible wireless network connectivity at any point in time, with minimal intervention on the part of the administrator [1]. The four choices for implementation for a cloud computing paradigm are community, private, public, and hybrid, which correspond to its fundamental architecture. According to the name indicates, a public cloud is an environment on a remote server that is open to numerous users refrained from any limitations imposed by the company providing the cloud service [2]. Evidently are two contexts in which one can implement a private cloud. A private cloud that is only used by a private organization may be administered and governed by the company that supplies the cloud service.

In the other instance, the private cloud is implemented solely inside one company, and the company is in charge of overseeing and supervising it. In any case, an individual enterprise uses a private cloud in an administrative setting for reasons of privacy. Community clouds are instances of cloud computing where connectivity has been restricted to end-users and companies with comparable purposes [3]. This kind of cloud can be collaboratively administered and supervised by members of the community, or it can be served by an alternative service provider that provides cloud services. Numerous cloud methods of implementation have been integrated to establish hybrid systems that utilise the cloud. In this way, public and private clouds can be connected [4]. While information of lower importance may be kept in a public cloud environment, enterprises as well as consumers frequently prefer hybrid cloud environment because of considerations about security issues.

Many essential features are combined in cloud computing, such as streaming independent service, collaborative use of resources, quantifiable assistance, instantaneous adaptability, and widespread network connectivity. Essentially, measurements like speed, information, and delay can be used to characterise services provided by the cloud [5]. When compared to conventional technological alternatives, services offered by the cloud can result in substantial expenses diminutions, with the associated expenses mostly being dictated by the amount of assets consumed [6]. When we describe cloud computing adaptability, this refers to about the ability of the architecture to respond to demand fluctuations by automatically allocating and releasing assets. The asset aggregating architecture uses a distributed design to provide support for numerous clients by sharing both real and digital assets, which are allocated in real time and transferred depending upon requests made by the user. Discovering services on a computer system and making use of them through a variety of computing environments and equipment such as mobile devices, handheld devices, notebooks, and desktop computers—is known as

[1]Associate Professor, Department of Computer Science and Engineering, Sri Eshwar College of Engineering ,Coimbatore, Tamilnadu, India
[2]Professor, Department of Information Technology, Ponjesly College of Engineering , Nagercoil, Tamilnadu, India
[3]Associate Professor, Department of Computer Science and Engineering, RRASE College of Engineering , Chennai, Tamilnadu, India
[4]Assistant Professor, Department of Computer Applications, Sona College of Arts and Science, Salem
saravanakumar.phdauc@gmail.com[1], itsanandmtech@gmail.com[2], ravikumarcsephd@gmail.com[3], rekamca2010@gmail.com[4]
Correspondence Mail: saravanakumar.phdauc@gmail.com

comprehensive accessibility to the network. Instantaneous independent service refers to the idea of enabling prospects to make use of data and resources stored in the cloud whenever they require access to, without requiring guidance from an individual [7].

The volume of load placed on the servers needs to be distributed evenly in order to enhance the efficiency of the cloud. Certain servers may have high consumption because of complex computational activities or large number of procedures running at once. Some other services might be idle at precisely the same moment. In this case, the effectiveness of the network would decline and its power would be squandered by the dormant machines. Additionally, before transmitting queries to a service that is overwhelmed, it is imperative to permit time for the congested service to finish its currently being processed operations [8]. This interruption is caused by the constraint that the system is able to execute a specific quantity of queries concurrently. As a consequence, there is a reduction in accessibility to services, which exacerbates consumers. The likelihood of the consumer to acquire the intended resource as frequently they prefer it is referred to as the service's accessibility. Load balancing can be implemented in this manner in order to improve productivity and minimise the utilization of power. Through seamless migrations of virtual machines that are available in different hosts that is congested to a host that is unproductive, the computational load is evenly divided among the computer systems, leading to improvements in network productivity [9].

The potential of the Internet of Things, neural networks, bio-inspired optimization algorithms, clustering and deep learning, to transform the distribution of resources, productivity maximization, and responsiveness in cloud computing settings makes them crucial to solve the problem of cloud load balancing [10]. Instantaneous data gathering and cloud resource surveillance are made possible by integrating a variety of connected devices, which allows for adaptive assignment of load based on request. Load balancing solutions may now proactively gain insight into past information as well as channel trends owing to the advancements in deep learning approaches. These techniques improve their capacity to forecast requirements for resources and make wise allocation of workload recommendations. For intricate and constantly changing cloud workloads, bio-inspired optimization methods are essential for enhancing use of resources and tweaking load distribution characteristics [11].

Deep learning improves the effectiveness of the management of loads and makes it possible to detect issues with performance and irregularities because of its capacity effortlessly identify complicated structures from data stored in the cloud [12]. The significance of deep learning lies in its ability to determine multifaceted connections and reliance between different resources in the cloud. This facilitates the establishment of more effective strategies for distributing workloads and utilisation of resources, which in turn improves the total efficiency and usage of the infrastructure used by the cloud [13]. By determining the most pertinent and exclusive cloud properties, optimization techniques-based feature selection strategies facilitate much faster load shifting procedures and lower processing complexity. Cloud load balancing maximises the possibility of these state-of-the-art innovations to achieve improved adaptability, resource conservation, and speed of reaction. This guarantees effective use of cloud assets, flawless user interactions, and the capacity to efficiently tweak to fluctuating computing settings [14].

This research recommends an approach to address the load balancing problems in the context of cloud computing, which are essential for maximising resource efficiency and boosting system functionality [15]. Cloud computing provides a adaptable and extensible platform, but effective distribution of workload among server components is necessary to prevent inadequate utilization as well as excessive utilization difficulties. This current investigation is motivated by the necessity to address shortcomings triggered by uneven distribution of loads in cloud environments, which can result in lower reliability of services and wasteful consumption of energy. The main contributions of this present research is to propose a reinforcement learning based metaheuristic algorithm to solve the problem of load balancing in cloud. The remainder of the paper is organized as follows. Section 2 presents the existing works on load distribution in clouds. Section 3 explains the proposed methodology using Partial Order Markov Decision process and Enhanced Firefly algorithm. Section 4 discusses the results obtained on experimental evaluation of proposed technique on GoCJ dataset. Section 5 concludes the present research.

II. RELATED WORKS

This section elaborates the recent researches on the application of meta heuristic algorithms to solve the problem of load balancing in cloud computing environments. A bioinspired method called Hive Optimization was employed in [16] to schedule tasks in a way that improves the proper distribution of loads on virtual machines. Based on the assessment of assigned workload and processing pace of the virtual machine, the proposed approach determines the optimal job to be allotted to a virtual machine in the data centre. The meta heuristic searching mechanism suggested in this work enables tasks to proactively deliver queries to virtual machines as needed. The reapplication of previously assigned tasks is repeated. It is said that the object of exploration area is a n-dimensional cube. Each weight shifts a number of elements of its weight placement vector as it crosses over a n-dimensional cube node. The placement matrix is coded in two-dimensional form as the outcome.

A novel method based on genetic algorithm was proposed in [17] for effective distribution of the cloud load. The load distribution process was found to be capable of exhibiting more equitable balance due to the

suggested approach. Both the aggregate sustaining expense and the time required for transfer were decreased by the approach employed. The authors suggested a workload levelling strategy that relied on the swarm optimization technique in [18]. Originally, the purpose of this technique was to address optimization problems that are non-deterministic using the controlled sampling technique. In addition, this approach is essential to the elimination method associated with heat recurrence which is significant in the field of entropy. This approach was put up for determining the perfect outcomes using the stochastic variables occurrence.

The researchers in [19] suggested an Artificial Bee colony-based load-sharing method to effectively exchange requests that are received in the environment of cloud computing. Its main goal was to increase productivity by keeping the virtual machines in an equilibrium of maximized harmony. It can adjust the importance of the jobs on the virtual machines so that the total amount of duration required to wait is primarily trimmed down to the lowest possible amount. Along with a reduced duration of waiting for tasks in the waiting list, it also demonstrated an apparent decrease in the overall average time taken to complete the tasks. Subsequently, in order to guarantee an elevated level of load distribution and allocation in the clouds, a load balancing technique based on Enhanced Ant colony optimization was proposed in [20]. The goal of this modified meta heuristic load balancing strategy was to shorten task necessary to complete the tasks while also eliminating the number of transfers required in the virtual machines for task execution. In the implemented technique, it characterized the total quantity of tasks segregated from the saturated virtual machines and the tasks that are under-utilized [21]. In order to successfully distribute the load among the readily accessible virtual machines in the cloud environment, the hunting practices of the artificial bees are incorporated into the load management operation.

The challenges involved in the application of metaheuristic optimization method use in cloud load distribution was investigated [22]. The authors employed a hybrid method to schedule the jobs submitted by the users efficiently in a short span while also reducing the expenditure. Further, the statistical analysis of the proposed method was also executed. There were two steps in the procedure itself. The initial step involved determining the virtual machine's largest capacity with the greatest processing efficiency. Deploying all of the jobs left to the virtual machine with the highest speed was the second step. The outcomes demonstrated that the suggested approach could cut down the overall time to execute. However, there were some issues with diminishing usage of computing assets and workload disequilibrium. In another work [23], the research scientists have suggested tweaking the workload management technique by employing randomized slope descents algorithm.

The researchers in [24] presented about using the Grey Wolf Optimization (GWO) technique to shorten processing times in cloud computing settings. The proposed method provided evidence of the effectiveness of GWO use. Compared to other traditional techniques like the First Come First Serve algorithms, GWO performed better. An enhanced Particle Swarm Optimization algorithm has been created in [25] to minimize length of processing and enhance accessibility to resources. The approach included adding some undetermined weights into the last stages of the algorithm and transforming the element weights as the number of trials progressed. The goal was to prevent the production of regional ideal outcomes during the latter stages of the algorithm.

An integrated finite whale optimization algorithm was presented in [26] for addressing adaptable load management challenges in cloud systems. This method's goal was to reduce the highest possible finishing time, the total amount of work across every item, and the highest load for every item. The authors in [27] presented the Behavioural Job Allocation with Ant Colony Optimization (BACO) Algorithm for organizing and handling cloud assets with the goal of regulating load and lowering finish duration in the cloud computing environment. The findings of the research demonstrated that the greatest successful approach for organizing and controlling cloud services was BACO using the Greatest Task Early adaptive algorithm.

Scheduling tasks and balanced load distribution with Harris Hawks optimization approach was suggested in [28]. Minimizing the time taken to process the tasks that were submitted to the data centre was ultimately the objective of the proposed technique. An innovative meta-heuristic method which combines Cuckoo search algorithm with genetic algorithm, was demonstrated [29] to address the load distribution of Internet of Things devices in a cloud setting with several processors. When evaluated with multiple tasks, network topologies and chip measurements, the technique ensured adequate resolution in terms of throughput and completion time. In a diverse multi-core condition, researchers discovered that the hybrid algorithm surpassed the outcomes produced by these algorithms individually.

A Simulated Annealing (SA) approach for minimizing energy consumption, duration of processing, expenditure, and formulating consumption of resources in cloud computing environments was put forth in [30]. The centralized repository was given tasks via a multifaceted SA algorithm. The comparison of the various approaches in the literature with respect to the load distribution in cloud computing environments.

Table. 1: Comparison of existing methods on cloud load balancing

References	Category of load balancing	Algorithm used	Inference	Limitations
[13]	Task based load balancing	Ant Colony Optimization	The proposed technique has proven to handle the task loads efficiently	The technique used in the experimentation employs only a policy based on space sharing while disregarding the time-based constraints
[15]	Task based load balancing	Genetic Algorithm	Less time taken for task completion	Only one objective is solved by the suggested method
[16]	Resource based load balancing	Simulated annealing	Resource utilization is improved	This technique does not handle a greater number of tasks
[18]	Virtual Machine based load balancing	Particle Swarm Optimization	Balances VM load with more reliability	Response time is lesser
[19]	Server based load balancing	Whale Optimization	Computational overhead is lessened	Does not take into consideration the service level constraints
[20]	Task based load balancing	Grey Wolf Optimization	Productivity is improved	Scalability issues exists
[21]	Server based load balancing	Dragonfly Optimization	This technique makes the system flexible in heterogeneous environments	More time required for allocation of tasks and reduced execution pace
[23]	Resource based load balancing	Harris Hawks Optimization	Minimum energy and cost for task completion	Prolonged time taken to respond to the user requests

III. PROPOSED METHODOLOGY

Based on the positive aspects of combining reinforcement learning methods with metaheuristic optimization algorithms, the proposed method is presented as a sustainable endeavour to enable significant distribution of load between virtual machines that are designated in the cloud environments. The goal of this suggested system is to facilitate the best possible method for load balancing across virtual machines in clouds by improving individual activation and space for exploration. This approach includes an efficient balanced job planning procedure that was developed using the quality-of-service parameters such as power, execution time, turnaround time, server expenditure, and level of disequilibrium.

The crucial advantages of the proposed research include the categorization of virtual machines into classes that are inadequately and adversely assigned with tasks during the load allocation procedure. Secondly, the focus is towards reducing the expenditure endured on the whole by lowering the expenses caused due to the power expenses in the servers. In addition to this, the identification of the utilization of entire collection of resources available in the hosts for achieving load distribution in a productive way. By visualizing the number of tasks entering the cloud environment, it is possible to determine boundaries that indicate the highest as well as the lowest levels of usage of virtual machines.

A Cloud Model

The fundamental structure of a cloud model consists of two levels which comprises of the jobs submitted by the users and the hosts that receive the submitted requests. These two levels work together in order to attain the primary goals of the cloud environment. Whenever the jobs are received, the associated information such as the time required to finish the particular job, total dimension of the job received along with the dimension of the job file is collected in order to facilitate appropriate provisioning of the virtual machines. The information pertaining to the virtual machines and the linked servers are reserved in the hosts. The reserved information includes the details regarding the number of items available for containers that accumulate data, process data, rate at which data can be transmitted. The information available in the host level and job level are correlated in order to identify the optimal resources to be allocated and balance the load on the virtual machines for job execution. The structure of the cloud setup is depicted in Figure 1.

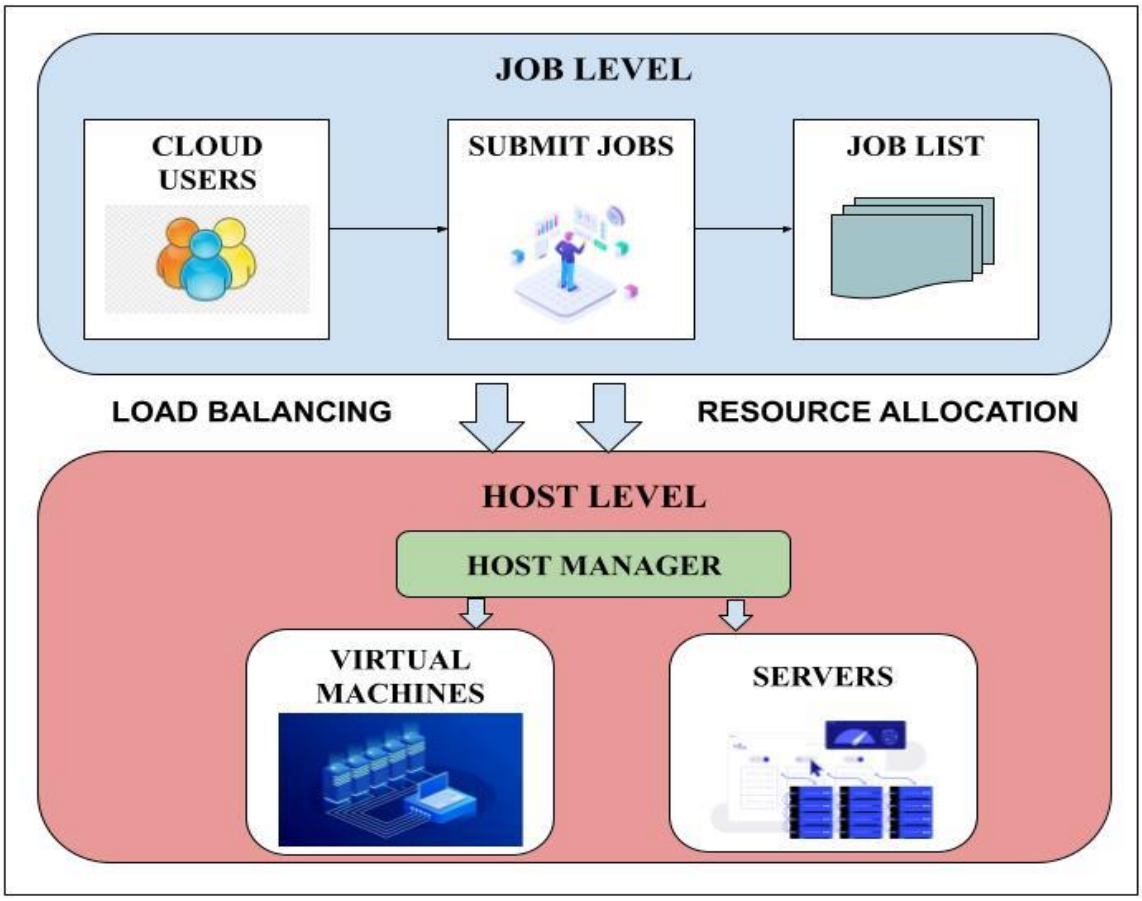

Fig. 1: Cloud Model

B Problem Definition

As the number of consumers of cloud environment increases, the need for efficient allocation of the available resources is also becoming more vital. Scheduling of cloud assets according to the requirements of the tasks is essential to avoid the assets being inadequately or adversely utilized. The load balancing in cloud settings involve the appropriate selection of virtual machines for the execution of the jobs. Consider there are K number of virtual machines represented as $VRM = \{vrm_1, vrm_2, vrm_3, \dots, vrm_k\}$, here $vrm_1\ to\ vrm_k$ represents the virtual machines that are available in the hosts. Every virtual machine in the cloud environment is associated with a certain set of resources including memory, processing power and expenses for the utilization. The tasks that are submitted to the hosts for execution can be represented as $TS = \{ts_1, ts_2, ts_3, \dots, ts_k\}$. In this case, an aggregate of k number of tasks are allocated to the virtual machines for processing. Every task submitted to the host will contain the details about the length of the specification, processing power and storage constraints needed. The core objective of the load balancing problem is to allocate virtual machines for appropriate jobs in such a way that the time taken to finish the jobs is reduced while also ensuring efficient utilization of cloud resources.

A task allocation method for cloud computing is suggested that makes use of multifaceted allocation conditions in order to maximize the consumption of resources and boost the productivity of the virtual machine, and establish the distribution of load across multiple virtual machines that are present in the host. Expenditure, the use of power, job finish time, job delay period, transit period, error level, job finish period, and trustworthiness are the most often used assessment variables. The suggested approach uses job finish period, Expenditure, and computing asset usage as elements that aid in optimal load balancing to identify acceptable characteristics. The development of an overall objective function is necessary for asset allocation as well as managing load in cloud computing environments in order to give the most feasible alternatives that take into account the multifaceted objectives.

Consider a task $ts_a \epsilon TS$ is allocated to virtual machine denoted by vrm_b, in this case the tasks allocated to a particular virtual machine $vrm_b = \{ts_{a1}, ts_{a2}, \dots, ts_{ak}\}$. The aggregate time taken to finish the processing of the tasks on vrm_b can be represented as in equation (1),

$$TF_{vrm_b} = \sum_{ts_{ab} \epsilon\, vrm_b} TE(ts_{ab}) \;\; = \frac{\sum_{ts_{ab} \epsilon\, vrm_b} len_{ts_{ab}}}{processor(vrm_b)} \tag{1}$$

In equation (1), TF_{vrm_b} denotes the time to finish task processing on virtual machine vrm_b, $TE(ts_{ab})$ denotes the time to execute the task ts_{ab} on vrm_b, $len_{ts_{ab}}$ denotes the total task size and $processor(vrm_b)$ denotes the computing power of the virtual machine. The value of $TE(ts_{ab})$ in the above equation can be determined by using the formulation in equation (2) as,

$$TE(ts_{ab}) = \frac{len_{ts_a}}{processor(vrm_b)} \tag{2}$$

The size of the task in equation (2) is generally represented as the count of the instructions and computing power of the virtual machine is the pace at which the requests are processed. The aggregate time to complete task execution $CompTime$ is considered as the highest value of the time taken to complete the tasks individually on a particular virtual machine and it is denoted as shown in (3),

$$CompTime = Max\big(TE(vrm_b)\big) \tag{3}$$

Similarly, the lowest value of the time to complete job execution on a virtual machine is represented as in equation (4),

$$MinCompTime = Min\big(TE(vrm_b)\big) \tag{4}$$

Thus, the objective function for the time to finish tasks is represented as in equation (5),

$$OF_1 = \frac{MinCompTime}{CompTime} \tag{5}$$

The expenditure incurred for the execution of the tasks on the virtual machine is computed as shown in equation (6) based on the cost taken for processing (ct_1), storage (ct_2) and transmission capacity (ct_3).

$$CT(ts_{ab}) = \big(ct_1 * TE(ts_{ab})\big) + \big(ct_2 * TE(ts_{ab})\big) + \big(ct_3 * TE(ts_{ab})\big) \tag{6}$$

The aggregate cost $ACT(vrm_b)$ to finish all the tasks on a specific virtual machine can be represented as it is shown in equation (7),

$$ACT = \sum_{b=1}^{y} \sum_{a=1}^{x} CT(ts_{ab}) \tag{7}$$

The lowest value of the expenditure incurred for the execution of a set of jobs that are assigned to one particular virtual machine is represented as given in equation (8),

$$MinACT = \sum_{ts_a \in TS} MinCT(ts_a) \tag{8}$$

Thus, the objective function for the expenditure to finish tasks on a specific virtual machine is represented as in equation (9),

$$OF_2 = \frac{MinACT}{ACT} \tag{9}$$

The load distribution on the virtual machines in the cloud for completing the jobs is determined based on the computing load and the storage load as in equation (10).

$$ST_{load} = STB_{load} + \frac{STA_{load}}{STT_{load}} \tag{10}$$

In the above equation, STB_{load} denotes the storage before job processing, STA_{load} denotes the storage after job processing and STT_{load} denotes the total storage available to perform job processing. The other factor that should be considered for load distribution is the computing load which is represented in equation (11) as,

$$CP_{load} = CPB_{load} + \frac{CPA_{load}}{CPT_{load}} \tag{11}$$

In the above equation, CPB_{load} denotes the computing power before job processing, CPA_{load} denotes the computing power after job processing and CPT_{load} denotes the total computing power available to perform job processing.

The total usage on a virtual machine is computed as shown in equation (12),

$$VRM_{usg} = \frac{\delta_1}{1 - ST_{load}} * \frac{\delta_2}{1 - CP_{load}} \tag{12}$$

In the above equation δ_1 and δ_2 are the random gradients assigned to the storage and computing units. The cumulative load that is placed on a server is represented as shown in equation (13),

$$CM_{load} = \sum_{a=0}^{k} VRM_{usg}(a) \quad (13)$$

The mean value of the load on the cloud setting is based on the cumulative load and the aggregate count of the servers (N) that are available as shown in equation (14),

$$Mean_{load} = \frac{\sum_{a=0}^{k} CM_{load}(a)}{N} \quad (14)$$

Thus, the objective function for the load distribution to assign tasks to appropriate virtual machines is represented as in equation (15),

$$OF_3 = |CM_{load} - Mean_{load}| \quad (15)$$

The final objective function comprising of all the individual objective functions is represented in equation (16) in which β is the coefficient of the sum of the objective functions,

$$FOF = \beta(OF_1 + OF_2 + OF_3) \quad (16)$$

C Partial Order Markov Decision Process

Significant assumptions regarding the level of adjustment likelihood is not required by the Partial Order-Markov Decision (POMD) process algorithm proposed in this research. The issue of the most effective way to distribute load among the resources that are readily available can be resolved by using the POMD process based on the reinforcement learning method. The approach utilized in this research establishes an agile flexible asset selection model for balancing load in the clouds and thus facilitate to maximally utilize offered cloud computing assets.

The inquiry administrator, behaviour, external state S, and external reward R are the four components that make up the proposed POMD system. A learning network and an intelligent agent are components of the inquiry administrator. Intelligent agents are employed to connect with the external world in order to monitor its condition and reap its benefits. The first step in training the learning network is to obtain the behaviour metrics for particular state using an estimation method. The learning network is then equipped, its configurations modified once more, and the decision method is continually optimized through reinforcement learning to make it better. Ultimately, the learning network returns the behaviour metrics of the most recent state to the intelligent agent for managing decisions, allowing the most efficient cloud load management scheme to be established.

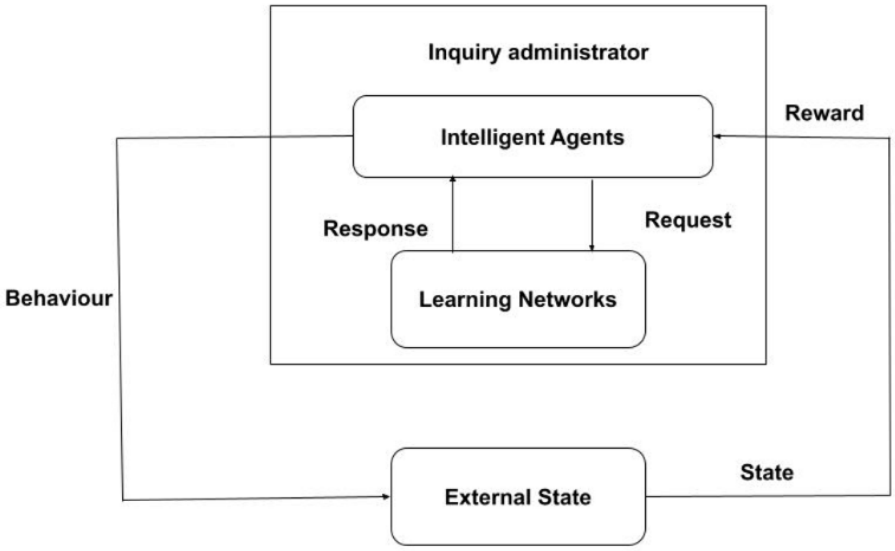

Fig. 2: Partial Order Markov Decision Process

D Enhanced Firefly Algorithm

The renowned metaheuristic algorithm, Firefly algorithm, was influenced by fireflies, specifically their community interactions and luminosity. Given the intricate nature and intelligence of the genuine firefly habitat, this metaheuristic serves as a closest match for estimation. One major issue that needs to be resolved in the FA's implementation is the real definition of desirability. Generally speaking, the objective function determines the firefly's luminosity, which determines how enticing it is. The luminosity of the firefly is computed using the equation (17),

$$DS(i) = \begin{cases} \frac{1}{h(i)}, & if\ h(i) > 0 \\ 1 + |h(i)|, & otherwise \end{cases} \quad (17)$$

In above equation, DS(i) represents the desirability of the location i and h(i) represents the function that characterizes the objectives. As the luminosity and desirability factors will decrease with the increase in the space of the light source, it can be mathematically formulated as denoted in equation (18),

$$DS(m) = \frac{DS_0}{1 + \alpha m^2} \quad (18)$$

Here m is used to denote the space factor and the impact of the same can be generalized and represented as shown in equation (19),

$$DS(m) = DS_0 \,.\, e^{-\alpha m^2} \quad (19)$$

The generalized form of the desirability is further represented for a every single firefly as given in equation (20),

$$\mu(m) = \mu_0 \,.\, e^{-\alpha m^2} \quad (20)$$

In the above equation μ_0 represents the desirability when the space value is found to be zero.

Alternatively, it can also be written as the representation in equation (21) assuming that the desirability of the firefly changes predominantly.

$$\mu(m) = \frac{\mu_0}{1 + \alpha m^2} \quad (21)$$

Any firefly x chosen in random fashion will make a transit towards another firefly y which is more luminous and desirable in nature and it is mathematically formulated as shown in equation (22),

$$i_x^{d+1} = i_x^d + \mu_0 . e^{-\alpha m_{x,y}^2} \left(i_y^d - i_x^d\right) + r^d (G - 0.5) \quad (22)$$

In above equation μ_0 denotes the desirability, r and G denotes arbitrary factors. The space between the two fireflies x and y is determined using equation (23),

$$m_{x,y} = \left|\left|i_x - i_y\right|\right| = \sqrt{\sum_{z=1}^{T} (i_{x,z} - i_{y,z})^2} \quad (23)$$

Here T represents the total count of the parameters involved in the process. In this system, the set of values for μ_0 is one and r takes the arbitrary value between zero and one. The proposed Enhanced Firefly algorithm is modified from the conventional firefly algorithm by introducing the concept of desirability of a firefly x towards three other fireflies such as y1, y2 and y3 and thus equation (23) is also modified as given in (24).

$$i_x^{current} = i_x + \mu_1 . e^{-\alpha m_{x,y1}^2} \left(i_{y1}^d - i_x^d\right) + \mu_2 . e^{-\alpha m_{x,y2}^2} \left(i_{y2}^d - i_x^d\right) + \mu_3 . e^{-\alpha m_{x,y3}^2} \left(i_{y3}^d - i_x^d\right) \quad (24)$$

The algorithmic steps in Enhanced Firefly algorithm are presented in Figure 3.

Algorithm: Enhanced Firefly Algorithm (EFA)
Input: Initial set of population of fireflies
Output: Most optimal solution
Step 1: Set initial values for the population with possible values
Step 2: Calculate the fitness of each population to find the local optimum and least optimum solutions
Step 3: for y = 1 to 3 in initial population
Step 4: Update value of $i_x^{current}$ using equation (24)
Step 5: if h ($i_x^{current}$) < h (i_x)
Step 6: Update old solution with current solution
Step 7: else
Step 8: retain old solution
Step 9: end if
Step 10: if h ($i_x^{current}$) <= h ($i_{optimal}$)
Step 11: then set $i_{optimal} = i_x^{current}$
Step 12: end if
Step 13: end for
Step 14: return most optimal solution, $i_{optimal}$

Fig. 3: Enhanced Firefly algorithm

E Working methodology

A multifaceted function is used by the proposed POMD-EFA technique to determine which virtual machines that are most suited for assigning upcoming or existing jobs. The assignment as well as reassignment are contingent upon the fundamental restrictions, which underscore the need for the load on virtual machines to surpass the highest possible level subsequent to task distribution. Then, when there is an extensive degree of

capacity in the virtual machines, the time frame limitation is taken into account. Furthermore, depending on the jobs' needed time to finish, moving them from an extremely burdened virtual machine to a minimally packed virtual machine is implemented.

On the other hand, regardless of the length of time to finish the upcoming job or redistributing job is moderate, virtual machines with larger and intermediate finish time tasks are considered. Furthermore, the virtual machines are organized entirely on the basis of their current load. Based on the evaluation of the target function, two types of groups such as inadequately burdened and adversely burdened virtual machines are created and assigned in this suggested system. The virtual machines in the adversely burdened virtual machines are forced to withdraw the tasks and hold off on assigning assets until they are able to locate a suitable virtual machine for the upcoming cycle. The virtual machines in the inadequately burdened categories are assigned to tasks that are either pending or require redistribution.

Here, the resulting outcomes denote the optimal solutions obtained by the firefly agents, which are produced according to the arbitrary sampling principle. The partial order ranking of the Markov decision process is used in this proposed POMD-EFA scheme to address the multifaceted optimization problems related to job assignment to virtual machines based on requirements, which is measured in terms of inadequately burdened and adversely burdened constraints. Additionally, it keeps track of the marginalized responses that were previously generated using the firefly agent's past record of optimal solutions. As a result, this helped with the possible job assignment into the virtual machines (VMs) based on their inadequate and adverse assignment restrictions, which are essential to the load balancing approach.

IV. RESULTS AND DISCUSSION

This section discusses the performance of the proposed system through evaluation and comparison with other existing approaches using GoCJ dataset. CloudSim was used as a simulation tool to execute the experimental evaluation of the present research. This simulation environment mimics the setup of a real cloud environment with all the required resources. A dataset simulated based on the loads received by the Google cloud is utilized in this work. It consists of a varied number of tasks ranging from small to large in sizes.

The performance of the proposed hybrid algorithm POMD-EFA is compared against the existing techniques such as Decision Trees with Particle Swarm Optimization, Randon Forest with Binary Particle Swarm Optimization, Support Vector Machine with Artificial Bee Colony Optimization and Naïve Bayes with Cuckoo Search Optimization. Three important metrics such as Task Finish Time, cost as well as resource utilization are considered to assess the performance of conventional techniques and proposed method.

Table 2 shows the performance obtained on varying the task sizes from 200 to 1000. It can be noticed that the time to finish the tasks increases gradually with the increase in the number of tasks to be executed. Among the conventional techniques, SVM-ABC exhibits lower finish time for tasks compared to other methods.

Table. 2: Performance Comparison based on Task Finish time varying number of tasks

Number of tasks	DT-PSO	RF-BPSO	SVM-ABC	NB-CSO	Proposed POMD-EFA
N=200	95	105	85	120	65
N=400	115	120	95	135	75
N=600	135	150	105	150	90
N=800	145	170	125	160	110
N=1000	175	195	145	180	125

However, it is higher compared to the proposed method which completes the tasks at 65s for 200 tasks, 75s for 400 tasks, 90s for 600 tasks, 110s for 800 tasks and 125s for 1000 tasks. The results are depicted graphically in Figure 4.

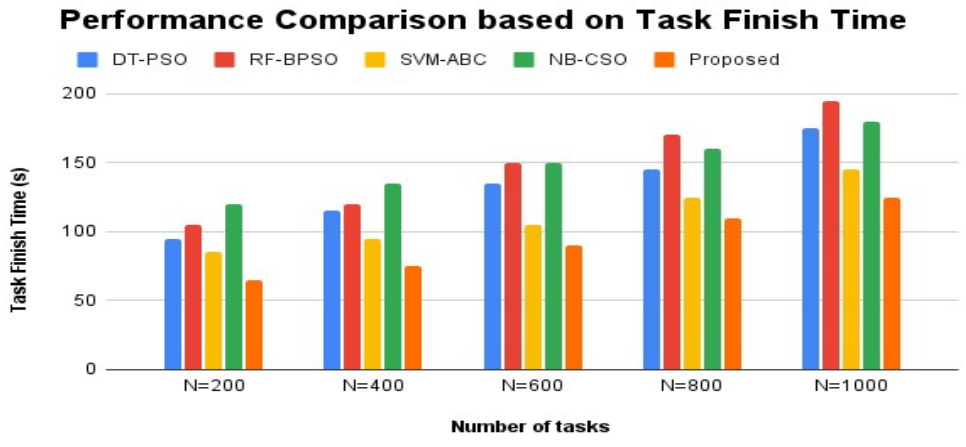

Fig. 4: Performance Comparison on Task Finish time for varying tasks

The results obtained by varying the number of virtual machines as 50,100,150,200 and 250 is presented in Table 3 and depicted in Figure 5. It can be observed that the task finish time is more when more number of VMs are involved in the execution.

Table.3: Performance Comparison based on Task Finish time varying number of VMs

Number of VMs	DT-PSO	RF-BPSO	SVM-ABC	NB-CSO	Proposed POMD-EFA
VM=50	105	125	95	135	75
VM=100	118	136	106	146	85
VM=150	124	145	114	158	97
VM=200	136	156	128	178	113
VM=250	148	175	150	194	128

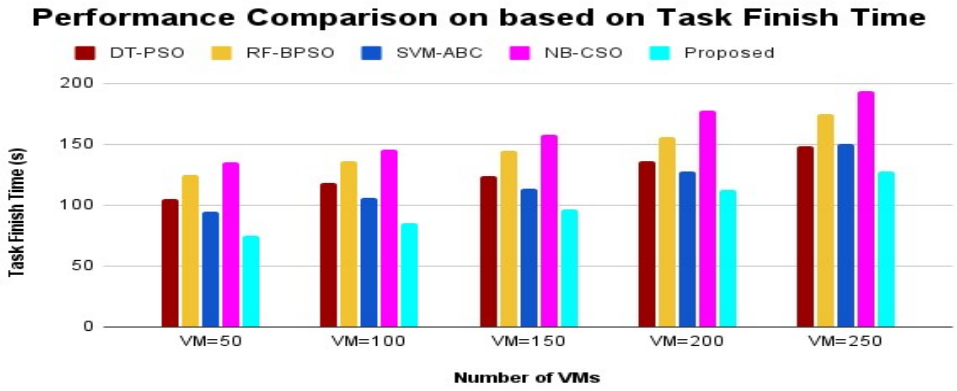

Fig. 4: Performance Comparison on Task Finish time for varying VMs

Similarly, the number of tasks is varied for the conventional and proposed technique and the cost incurred to complete the tasks in the cloud environment is computed and presented in Table 4 and Figure 5. It can be seen that DT-PSO takes 1550 G$ for 200 tasks whereas 1950 G$ for 1000 tasks. RF-BPSO incurs 1950 G$ for 400 tasks and 2250 G$ for 600 tasks.

Table. 4: Performance Comparison based on Cost varying number of tasks

Number of tasks	DT-PSO	RF-BPSO	SVM-ABC	NB-CSO	Proposed POMD-EFA
T=200	1550	1850	1445	1650	1150
T =400	1650	1950	1560	1750	1190
T =600	1780	2250	1680	1890	1200
T=800	1890	2350	1790	1960	1260
T=1000	1950	2445	1900	2150	1350

SVM-ABC incurs quite less cost compared to other conventional methods with 1650 for 200 and 1750 for 400 tasks. NB-CSO produces a cost of 1890 G$ for 600 tasks and 2150 G$ for 1000 tasks. The proposed POMD-EFA incurs an expenditure of 1150 G$ for 200 tasks and 1350 G$ for 1000 tasks which is the least expenditure among the models under comparison.

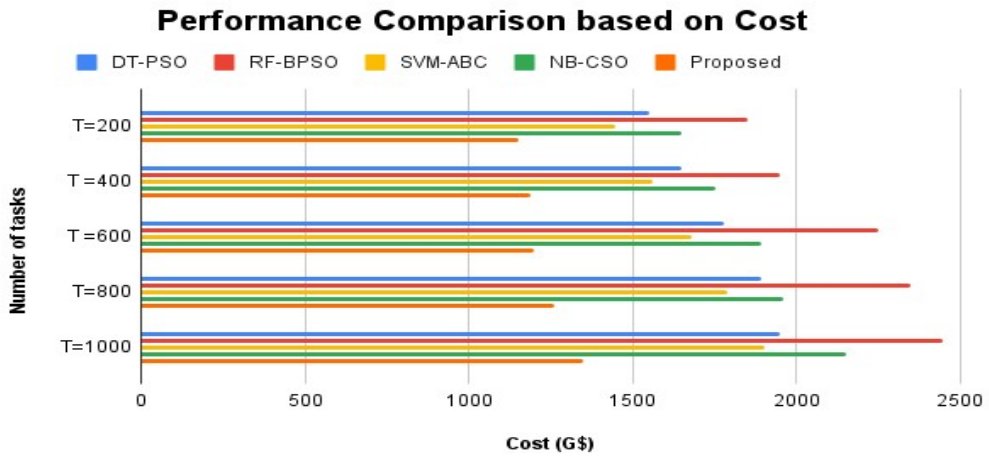

Fig. 5: Performance Comparison on Cost for varying tasks

The cost analysis is also carried out by differing the number of virtual machines and the cost values are recorded in Table 5 and presented pictorially in Figure 6. It can be noted that the cost values are the highest for RF-BPSO combination compared to the other methods.

Table. 5: Performance Comparison based on Cost varying number of VMs

Number of VMs	DT-PSO	RF-BPSO	SVM-ABC	NB-CSO	Proposed POMD-EFA
VM=50	1350	1500	1160	1400	990
VM=100	1420	1650	1250	1590	1090
VM=150	1570	1720	1370	1640	1150
VM=200	1640	1800	1420	1730	1240
VM=250	1780	1900	1500	1800	1390

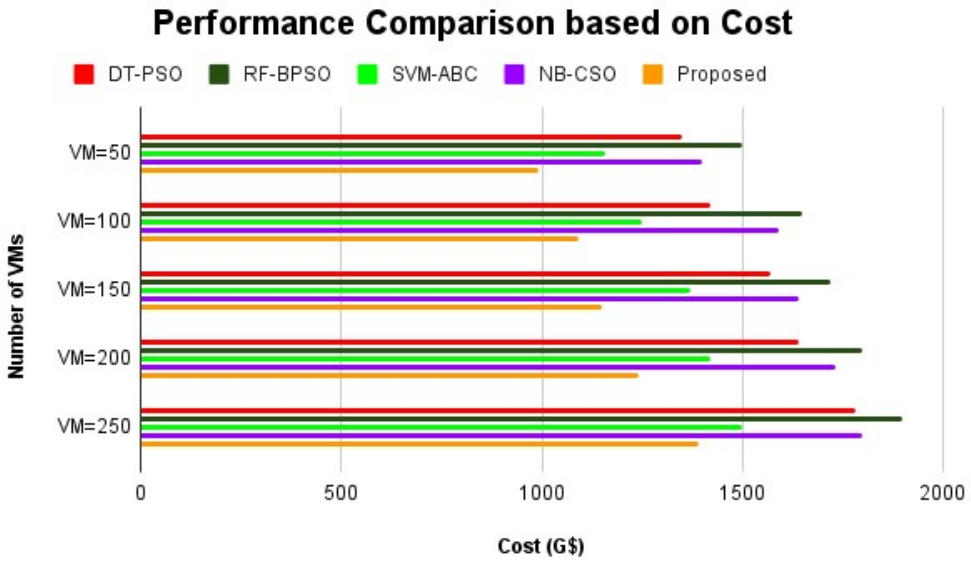

Fig. 6: Performance Comparison on Cost for varying VMs

The third important parameter to be noted for the performance evaluation is the resource utilization made by the models. More resource utilization defines that the load on the cloud is distributed in an even manner. The results obtained are presented clearly in Table 6 and Figure 7. Poor resource utilization is exhibited by DT-PSO model starting from 15% for 200 tasks to only 40% for 1000 tasks.

Table. 6: Performance Comparison based on Resource utilization varying number of tasks

Number of tasks	DT-PSO	RF-BPSO	SVM-ABC	NB-CSO	Proposed
T=200	15	28	35	42	45
T =400	25	35	39	46	55
T =600	35	39	46	53	68
T=800	38	43	51	58	75
T=1000	40	52	55	62	90

The resource utilization varies between 28% to 52% for RF-BPSO, 35% to 55% for SVM-ABC and 42% to 62% for NB-CSO. The proposed model makes the maximum possible usage of the available resources for efficient allocation of resources to the tasks ranging from 45% for 200 tasks, 55% for 400 tasks, 68% for 600 tasks, 75% for 800 tasks and maximum of 90% for 1000 tasks.

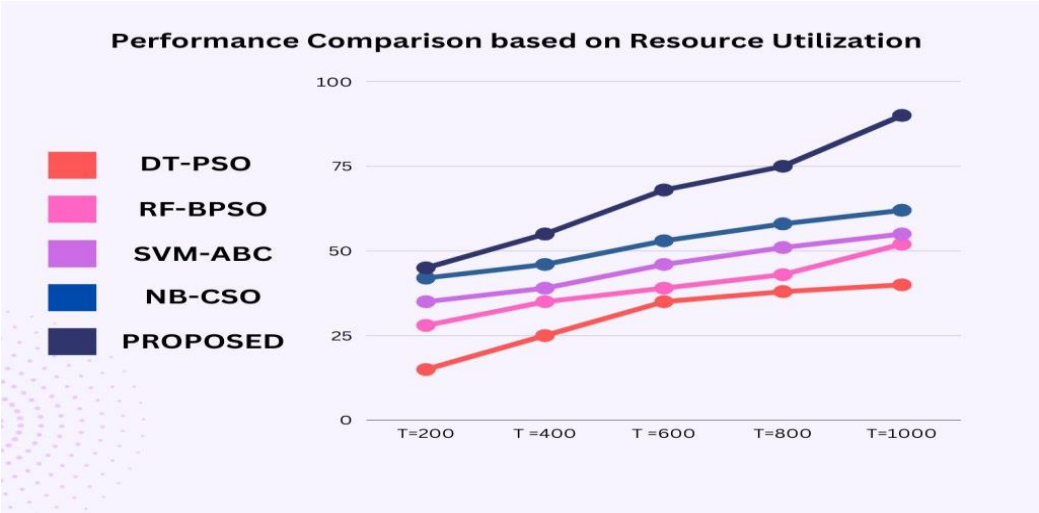

Fig. 7: Performance Comparison on Resource utilization for varying tasks

Performance of the proposed model on resource utilization is also compared against the other conventional models by differing the number of virtual machines involved. It can be seen that the resource utilization is achieved to a maximum of 95% by the proposed method for 250 virtual machines. The experimental outcomes are presented in Table 7 and depicted in Figure 8.

Table. 7: Performance Comparison based on Resource utilization varying number of VMs

Number of VMs	DT-PSO	RF-BPSO	SVM-ABC	NB-CSO	Proposed POMD-EFA
VM=50	25	33	35	45	50
VM=100	32	38	40	50	58
VM=150	36	41	45	55	67
VM=200	42	46	55	60	82
VM=250	48	53	60	65	95

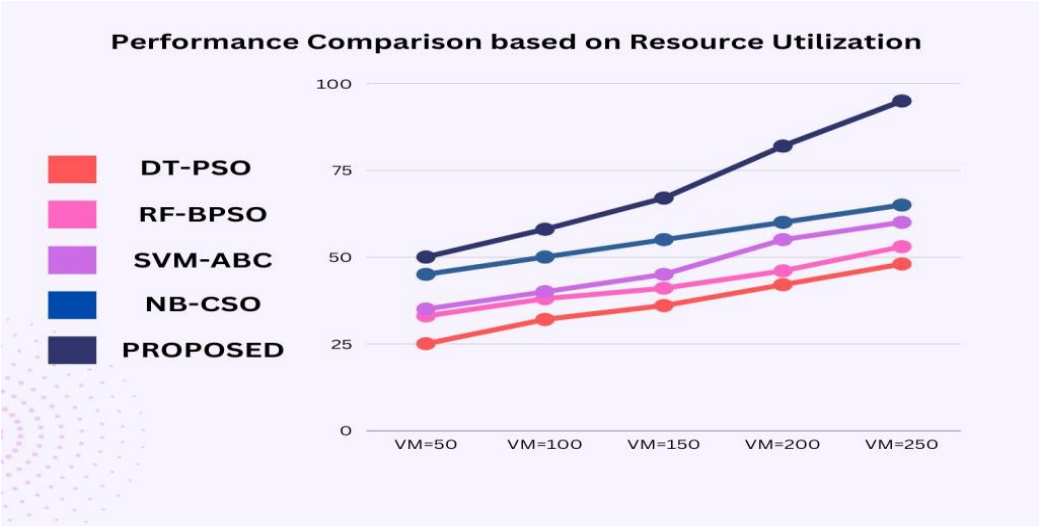

Fig. 8: Performance Comparison on Resource utilization for varying VMs

V. CONCLUSION

In this work, POMD-EFA method is proposed as a multifaceted optimization load balancing approach in diverse cloud computing scenarios. Using this method, the suitability of each virtual machine was identified in order to determine the most appropriate ones. To evaluate the performance of the suggested methods, tests were carried out using GoCJ dataset. In comparison to the DT-PSO, RF-BPSO, SVM-ABC, NB-CSO algorithms, the suggested approach promotes lowering of time to finish, minimizing expenditures incurred and spikes the use of computing resources, in addition to aiding in load balancing activities. The suggested approach performed better than the others, according to the findings from the study. As a future enhancement to the present research, efficient job allocation and load distribution in fog or edge environment may prove to be a stimulating and demanding task.

It is possible to apply alternative algorithms for machine learning in future versions. Furthermore, the performance of the POMD-EFA approach can be assessed by testing it in a real-world context.

REFERENCES

[1] K. Mishra and S. K. Majhi, "A binary bird swarm optimization based load balancing algorithm for cloud computing environment," Open Computer Science, vol. 11, no. 1, pp. 146–160, 2021.

[2] G. Thejesvi and T. Anuradha, "Selection of a virtual machine within a scheduler (dispatcher) using enhanced join idle queue (EJIQ) in cloud data center," in In Intelligence in Big Data Technologies—Beyond the Hype, J. D. Peter, S. L. Fernandes, and A. H. Alavi, Eds., Springer Singapore, 2021.

[3] V. K. R. Gangireddy, S. Kannan, and K. Subburathinam, "Implementation of enhanced blowfish algorithm in cloud environment,"Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 3, 2021.

[4] S. K. Mishra, B. Sahoo, and P. P. Parida, "Load balancing in cloud computing: a big picture," Journal of King Saud University-Computer and Information Sciences, vol. 32, no. 2, p. 2, 2020.

[5] A. Gupta, H. S. Bhadauria, and A. Singh, "Load balancing based hyper heuristic algorithm for cloud task scheduling," Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 6, pp. 5845–5852, 2021.

[6] Z. Ma, S. Shao, S. Guo, Z. Wang, F. Qi, and A. Xiong, "Container migration mechanism for load balancing in edge network under power internet of things," IEEE Access, vol. 8, pp. 118405–118416, 2020.

[7] I. Alharkan, M. Saleh, M. A. Ghaleb, H. Kaid, A. Farhan, and A. Almar- fadi, ''Tabu search and particle swarm optimization algorithms for two identical parallel machines scheduling problem with a single server,'' J. King Saud Univ.-Eng. Sci., vol. 32, no. 5, pp. 330–338, Jul. 2020, doi:10.1016/j.jksues.2019.03.006.

[8] B. Shrimali and H. Patel, ''Multi-objective optimization oriented policy for performance and energy efficient resource allocation in cloud environment,'' J. King Saud Univ.-Comput. Inf. Sci., vol. 32, no. 7, pp. 860–869, Sep. 2020, doi: 10.1016/j.jksuci.2017.12.001.

[9] M. Haris and S. Zubair, "Mantaray modified multi-objective Harris hawk optimization algorithm expedites optimal load balancing in cloud computing," Journal of King Saud University-Computer and Information Sciences, vol. 34, no. 10, pp. 9696-9709, 2022.

[10] X. Guo, ''Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm,'' Alexandria Eng. J., vol. 60, no. 6, pp. 5603–5609, Dec. 2021.

[11] T. Thein, M. M. Myo, S. Parvin, and A. Gawanmeh, ''Reinforcement learning based methodology for energy-efficient resource allocation in cloud data centers,'' J. King Saud Univ.-Comput. Inf. Sci., vol. 32, no. 10, pp. 1127–1139, Dec. 2020, doi: 10.1016/j.jksuci.2018.11.005.

[12] S. M. Mirmohseni, C. Tang, and A. Javadpour, "FPSO-GA: a fuzzy metaheuristic load balancing algorithm to reduce energy consumption in cloud networks," Wireless Personal Communications, vol. 127, no. 4, pp. 2799-2821, 2022.

[13] G. Annie Poornima Princess and A. S. Radhamani, "A hybrid meta-heuristic for optimal load balancing in cloud computing," Journal of Grid Computing, vol. 19, no. 2, 2021.

[14] K. Mahesh, "Workflow scheduling using improved moth swarm optimization algorithm in cloud computing," Multimedia Research, vol. 3, no. 3, pp. 36–43, 2020.

[15] A. Muteeh, M. Sardaraz, and M. Tahir, "MrLBA: multi-resource load balancing algorithm for cloud computing using ant colony optimization," Cluster Computing, vol. 24, no. 4, pp. 3135–3145, 2021.

[16] S. Sefati, M. Mousavinasab, and R. 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.

[17] M. S. Sanaj and P. M. Joe Prathap, ''An efficient approach to the map-reduce framework and genetic algorithm based whale optimization algorithm for task scheduling in cloud computing environment,'' Mater. Today, Process., vol. 37, pp. 3199–3208, Oct. 2021.

[18] F. Ebadifard and S. M. Babamir, "Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment," Cluster Computing, vol. 24, no. 2, pp. 1075-1101, 2021.

[19] B. Kruekaew and W. Kimpan, "Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning," IEEE Access, vol. 10, pp. 17803-17818, 2022.

[20] K. Mishra and S. K. Majhi, "A binary Bird Swarm Optimization based load balancing algorithm for cloud computing environment," Open Computer Science, vol. 11, no. 1, pp. 146-160, 2021.

[21] B. Pourghebleh, A. A. Anvigh, A. R. Ramtin, and B. Mohammadi, "The importance of nature-inspired meta-heuristic algorithms for solving virtual machine consolidation problem in cloud environments," Cluster Computing, pp. 1-24, 2021.

[22] J.-Q. Li and Y.-Q. Han, ''A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing system,'' Cluster Comput., vol. 23, no. 4, pp. 2483–2499, Dec. 2020, doi:10.1007/s10586-019-03022-z.

[23] L. Caviglione, M. Gaggero, M. Paolucci, and R. Ronco, ‘‘Deep reinforcement learning for multi-objective placement of virtual machines in cloud datacenters,’’ Soft. Comput., vol. 25, pp. 12569–12588, Oct. 2021, doi:10.1007/s00500-020-05462-x.
[24] G. Saravanan, S. Neelakandan, P. Ezhumalai, and S. Maurya, "Improved wild horse optimization with levy flight algorithm for effective task scheduling in cloud computing," Journal of Cloud Computing, vol. 12, no. 1, p. 24, 2023.
[25] A. Muteeh, M. Sardaraz, and M. Tahir, "MrLBA: multi-resource load balancing algorithm for cloud computing using ant colony optimization," Cluster Computing, vol. 24, no. 4, pp. 3135-3145, 2021.
[26] B. Kruekaew and W. Kimpan, ‘‘Enhancing of artificial bee colony algorithm for virtual machine scheduling and load balancing problem in cloud computing,’’ Int. J. Comput. Intell. Syst., vol. 13, no. 1, pp. 496–510, 2020, doi:10.2991/ijcis.d.200410.002.
[27] B. Kruekaew and W. Kimpan, “Enhancing of artificial bee colony algorithm for virtual machine scheduling and load balancing problem in cloud computing,” International Journal of Computational Intelligence Systems, vol.13, no. 1, pp. 496, 2020.
[28] P. Neelima and A. R. M. Reddy, ‘‘An efficient load balancing system using adaptive dragonfly algorithm in cloud computing,’’ Cluster Comput., vol. 23, pp. 2891–2899, 2020, doi: 10.1007/s10586-020-03054-w.
[29] U. K. Jena, P. K. Das, and M. R. Kabat, ‘‘Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment,’’ J. King Saud Univ.-Comput. Inf. Sci., early access. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157819309267,doi:10.1016/j.jksuci.2020.01.012.
[30] D. Alsadie, ‘‘A metaheuristic framework for dynamic virtual machine allocation with optimized task scheduling in cloud data centers,’’ IEEE Access, vol. 9, pp. 74218–74233, 2021, doi: 10.1109/ACCESS.2021.3077901.