International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 15, No.1 (Feb-24)

http://dx.doi.org/10.12785/ijcds/150142

Multi-Faceted Job Scheduling Optimization Using Q-learning With ABC In Cloud Environment

Sanjeev Sharma[1] **and Neeraj Kumar Pandey**[2]

[1]*School of Computing, DIT University, Dehradun, Uttarakhand, India*
[2]*Department of Computer Science & Engineering, Graphic Era Deemed to be University, Dehradun, Uttarakhand, India*

Received 28 Apr. 2023, Revised 8 Jan. 2024, Accepted 21 Jan. 2024, Published 1 Feb. 2024

Abstract: Resource allocation is the utmost challenging and common problem, particularly in the cloud service model in Infrastructure as a Service (IaaS). The issue of load balancing is so harmful that irregular load balancing may result in a structure smash. Adopting a suitable access plan and allowing the system to spread work among all existing resources leads to utilizing Virtual Machines (VMs) appropriately. To get enhanced results from the Artificial Bee Colony Algorithm (ABC), a reinforcement learning technique Q-learning is combined using multifaceted job scheduling optimization based on ABC(QMFOABC) has been proposed. The proposed approach improves resource utilization and scheduling created on resource, cost, and makespan. The efficiency of the suggested strategy was evaluated in Datasets Synthetic Workload, Google Cloud Jobs (GoCJ), and Random by using CloudSim to the remaining scheduling strategies for load matching like Max-min, Multifaceted Cuckoo Search (MFCS), Multifaceted Particle Swarm Optimization (MFPSO), Q-learning, Heuristic job scheduling with Artificial Bee Colony approach with Largest Job First algorithm (HABC_LJF), First come first serve (FCFS). According to the findings of the experiments, the algorithm that employed the QMFOABC method has better results in resource utilization, throughput, cost, and makespan. Compared to Max-Min (82.31%), MOPSO (35.62%), HABC_LJF (21.65%), Q-Learning (11.72%), VTO-QABC_FCFS(5.87%), and VTO-ABC_LJF (5.86%) shorter time than MOCS, a considerable improvement is found.

Keywords: Cloud computing, multifaceted job, scheduling, Qlearning, virtual machines, ABC

1. Introduction

The provision of shared resources by the different users as per their demand through the internet is an important facility provided by cloud computing. Cloud computing is becoming popular in IT as an adaptable, elastic, and simple approach to developing service platforms using outsourced resources [1]. Furthermore, a parallel and distributed computing hybrid allows us to use collective resources on a pay-as-you-go basis, and the only requirement is an internet connection. IaaS is essential in managing Virtual Machines (VM), data centers, and servers among the different service models. By providing a cloud server, IaaS helps its users store and process data in a virtual cloud environment and helps them by giving maintenance-free hardware, applications, or different operating systems.

Furthermore, depending on the rental agreement between the cloud service provider (CSP) and the cloud service user, the user can fulfill requirements like service level agreements (SLA) or quality of service (QoS). Not only this, but IaaS also helps its users remotely access servers and execute various functions on Virtual machines using cloud computing [2]. Cloud computing empowers service providers to deliver high-level functioning resources to data centers to assist cloud service consumers.

There is an unbalanced resource allocation with the uneven change in the numerous cloud users and their frequent requirement for virtual resources for servers, CPU memory, or RAM. Furthermore, many users (VMs) don't have access to the resources, so they need preemptive and non-preemptive associates between virtual machines [3]. Therefore, the VM should be capable of running rapidly whenever a job is performed in a cloud environment to minimize waiting periods. Accordingly, a work schedule must be allocated and spread across available resources. Numerous tasks are assigned to different virtual machines to compile work better, and they will operate in parallel to finish the assignment. All VMs should be assigned work in parallel so that each one can fully utilize its resources. The work must ensure that all jobs are spread equally across all VMs to prevent system instability or unavailability. The scheduling must consider additional elements such as resource usage, price, and makespan to prevent this. Several academics have developed approaches to load balancing in mixed and similar systems [4]. Passing on jobs in a load-

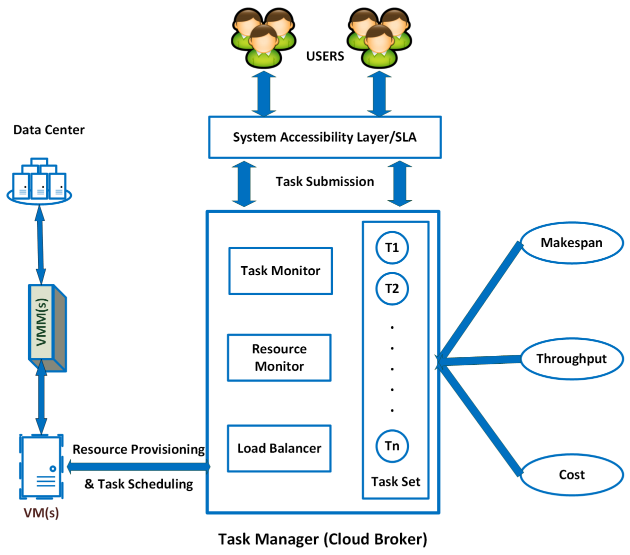

Figure 1. Process for Q-learning strategy

balancing form maximizes job circulation between existing resources and minimizes system execution time.

The framework for the managing of cloud resources is shown in figure 1. The primary goal for researchers to strive to develop an efficient algorithm is to pass the request to the cloud broker as soon as the user requests the system. The suggested approach should allow users to effectively send tasks to the appropriate VM based on specified parameters, such as the deadline or other criteria. In addition, it must guarantee that the requests given by the user are completed in compliance with the SLA document's specifications. Various requests generated by the user are maintained on VMs. The Cloud computing services subsequently distribute QoS-based inquiries to confirm that the user's bids are correctly completed. This procedure depends on the scheduling policy's performance (Data Broker). As a result, the workload balance between the machine and the server must be modified. Virtual Machine Monitors (VMM) or hypervisors play a significant part in cloud computing. Multiple VMs can be run on the same hardware layer with hypervisors [5]. VMware is one of the utmost well-known software facilities for handling company server resources. Although virtualization plays a significant part in cloud technology, poor scheduling or delegating jobs to VMs that cannot satisfy requirements still exist. Cloud computing should provide scheduling and load balancing across nodes to maximize resource consumption to tackle this challenge. Due to load balancing and scheduling across nodes, processing times were slashed, but the system became unbalanced.

In the virtual environment, task assignments must consider success rate, cost, and time. Therefore, much research has examined QoS parameters for optimizing cloud resources and their scheduling in single or bi-objective optimization [6], [7], [8] while some papers have discussed multi-objective optimization [9]. However, a few cases of approaches that have been utilized to improve the competence of the on-demand job are the Strength Pareto evolutionary algorithm (SPEA) [10], the Niched Pareto genetic algorithm (NPGA) [11], the Genetic Algorithm (GA) [12] and Particle Swarm Optimization (PSO) [13], [14].

Karaboga devised the Artificial Bee Colony (ABC) algorithm, a meta-heuristic approach for solving issues and finding optimized results near the suitable worth [15], [16]. The ABC algorithm is modeled after bee colonies' foraging activity, which requires adaptability to changing habitats and food sources. The ABC algorithm has been shown to help solve job scheduling challenges [17] and traveling salesman problems [18]. The ABC algorithm uses an environment-based learning technique akin to Reinforcement Learning (RL), which uses observation to anticipate or choose an appropriate answer. Agents learn how to act in a specific situation by doing and watching the outcomes. The Q-learning method is used in this study to assist the system in making predictions and judgments regarding the best schedule to follow. To resolve the scheduling problem in a virtual environment, we offer a multifaceted optimization scheduling model in combination with the Q-learning technique and ABC algorithm (QMFOABC). The study aims to design a multifaceted scheduling strategy to optimize work schedules to decrease simultaneous makespan, cost, and resource utilization while also considering the system's load balance.

The following are the research objectives along with the proposed method:

- Suggested a scheduling method for multifaceted optimization. This method involves various constraints like utilization of resources, cost, and execution time for load balancing and scheduling.
- Construct mathematical representations for finding cost, makespan, and resource utilization for scheduling problems.
- The suggested approach combines Q-learning and ABC algorithms for solving scheduling problems with multi-objective optimization. The presented system is a combination of two heuristic scheduling systems, LJF and FCFS.
- The suggested approach tries to gain stability through fitness function by finding the individual loads of VM. Then, the simulation process was carried out with the help of CloudSim using Q-learning.

The remaining part of the paper consists of a literature review, problem definition and formulation, and a hybrid algorithm evaluation of the experimental work. The last part is the conclusion and future scope.

2. Literature Review

Finding a solution for the NP-hard problem is a difficult task in cloud computing, and load balancing and task scheduling optimization are among them. Therefore, resource scheduling and allocation research have been presented to optimize the load, power, and execution time. We must evaluate a concept in scheduling numerous targets or cloud settings to use current resources or rapid processing efficiently. Several ideas have been offered to handle scheduling problems and load balancing difficulties in the virtualized environment, employing heuristic, meta-heuristic, hybrid meta-heuristic algorithms, or even machine learning approaches [19], [20].

Heuristic approaches are much more approachable in providing solutions to problems in cloud computing, such as predicting the execution time and load balancing in the virtual environment, a successful solution based on the max-min approach given by Mao [21]. The practice shows that the execution time decreases with the increase in resources by VM. Another method that utilizes the idle resources by rescheduling the jobs based on their completion time was based on Enhanced load balances static scheduling proposed by Patel [22].

A task scheduling approach in the grid cloud to diminish execution time and cost was proposed by Zhang [23]. The strategy also uses computational models to solve the challenge of grid-cloud work scheduling. A combination of non-preemptive and independent jobs in a dynamic scheduling manner is another approach for load balancing using resource-aware has been proposed by Hussain et al. [24]. The initial step was to choose the VM with the largest processing share size. The second stage included assigning the jobs left to the shortest VM. The outcomes revealed that the algorithm could shorten the makespan with leftover issues with resource consumption and load imbalance. [25] recommended that the scheduler technique be changed by applying soft computation built stochastic hill-climbing on load management.

Adhikari and Koley [26] introduced the competent, dynamic multi-workflow scheduling (CDMWS) method, an emotional multi-workflow scheduling technique. This strategy was designed to enhance the ratio of makespan, decrease makespan and increase CPU utilization to meet the deadline. There are two aspects to this strategy. The initial section calculated the computation for every job based on the task dependencies and deadline. The other part was to decrease power consumption by arranging virtual machines to maximize resource use. CDMWS beat the Round Robin (RR) and Enriched Workflow Scheduling Algorithm (EWSA) algorithms in the tests.

Shrimali and Patel [27] studied optimal resource allocation, emphasizing the multi-objective optimization (MOO) technique to tackle energy efficiency challenges. MOO has also been presented as a VM allocation method. According to the findings of the experiments, MOO resulted in efficiency improvements owing to optimal resource allocation without impacting data center performance.

Meta-heuristic techniques are trendy in solving the issues related to cloud computing. Ant colony optimization(ACO) reduces job completion time, as shown in the study by Song et al. [28]. The authors proved the effectiveness of ACO. Other approaches, such as the RR and FCFS algorithms, performed worse than ACO. An enhanced PSO technique was developed to enhance resource access and decrease the makespan time. The approach involved changing particle weights as the number of iterations increased and introducing additional arbitrary values in the PSO's final phases. The goal was to avoid finding locally optimal solutions in the final stage.

The approach presented by Chen et al. [29] emphasizes improving the completion time per the deadline restriction to decrease the cost of work and complete tasks on time, known as dynamic objective genetic algorithm (DOGA). The resource allocation mechanism for VM in cloud computing was suggested by Amini et al. [30] using the dragonfly optimization algorithm. The dragonfly optimization algorithm outperformed the Hybrid Algorithm Based on Particle Swarm Optimization and Ant Colony Optimization algorithms regarding resource provisioning and load balancing (PSO-ACO).

To address problems with elastic task allocation in a virtual environment, Li and Han [17] proposed a fusion of different ABC techniques. The goal was to reduce all devices' workload and execution time. Finally, Kruekaew and Kimpan [31] presented the combination of an Artificial Bee Colony in the Heuristic Task Scheduling (HABC) Algorithm to plan and manage cloud resources to decrease the load balance and finish time of the virtualized computing. The experiments revealed that HABC effectively handled cloud resources with the Largest Job First heuristic algorithm (HABC_LJF).

To reduce the completion time of the data center process, a genetically simulated annealing approach was presented by Gan et al. [32]. Basu et al. [12] presented GAACO, a hybrid meta-heuristic method that combined the GA and ACO algorithms to handle the Internet of Things job scheduling in multiprocessing virtualization. The technique ensured suitable convergence when evaluated with various task graph sizes and processor counts regarding makespan and effectiveness. Furthermore, it was discovered that the GAACO approach outperformed the GA and ACO algorithms in a varied multi-processor situation.

Alsadie [33] suggested a meta-heuristic structure for dynamic VM allocation with enhanced job setting up in a virtualized environment for multifaceted job arrangement situations in a virtual condition that is overcome with a meta-heuristic technique and a fusion meta-heuristic procedure (MDVMA). MDVMA was a multifaceted scheduling system that used a non-dominated arrangement of genetic algorithms for energy consumption, time, and cost. The CloudSim Simulator was utilized in the studies using datasets as Heterogeneous Computing Scheduling Problems (HCSP). The outcomes of MDVMA revealed that the approach could enhance job arrangement well than the PSO, Whale Optimization Algorithm (WOA), an ABC approach to lower the cloud data center's energy consumption, makespan, and cost.

Guo [34] introduced a fuzzy stability method for multifaceted job scheduling, outperforming the competition regarding VM resource usage, deadline violation rate, and lowest finishing time. PBACO [35] and the Task Allocation approach based on RL are two multifaceted optimization scheduling methods based on the Ant Colony Algorithm in Cloud Computing. Zuo et al. [35] suggested a multifaceted optimization scheduling technique for cost and performance. PBACO was utilized to find the best result. The experiment was compared to the traditional heuristic method and was inferior to PBACO.

To enhance job completion, energy consumption, cost, and resource utilization, He et al. [36] introduced Adaptive Multifaceted Task scheduling (AMTS). AMTS employed a PSO-based strategy for multifaceted development that measured activity and transmission delay. In addition, the adaptive acceleration coefficient was used in the suggested approach to account for particle variety. As a result, AMTS discovered the optimum solution for the cloud-based scheduling problem after enhancing the PSO algorithm.

An ABC-based strategy was developed by Jena [37] for cloud computing energy efficiency, cost, processing time, and resource usage. A multifaceted ABC method was used to assign tasks to the data center. Kumar and Venkatesan [38] employed an Ant-based heritable process to handle multifaceted scheduling issues in the cloud to improve execution time and latency while optimizing throughput. Finally, Alsadie [39] presented the optimization WOA to progress development in the virtual environment. The attempt was to make a fitness-based plan created on three criteria: resource, energy consumption, and quality of service. The completion time and charge of the VMs were lowered after considering the stated criteria.

Madni et al. [40] introduced a Multi-objective Cuckoo Search Optimization (MOCSO) method to lower cloud user fees with improving performance by shortening makespan time for scheduling in the IaaS virtualized environment. The study shows that MOSCO has better results than other Multi-objective methods using PSO, Min-max, GA, and ACO strategies.

With the constraints of scheduling performance and time for multifaceted job scheduling, a hybrid algorithm was proposed by Pang et al. [41] built on the valuation of the distribution algorithm (EDA) and genetic algorithm (GA). This method's rapid convergence speed and excellent search capabilities were its benefits. Furthermore, considering the test findings, the EDA-GA hybrid algorithm substantially lowered task finish time and increased load balancing capabilities while tested using CloudSim simulation against EDA and GA.

A hybrid method of firefly(FA) and dragonfly procedure (DA) with the name Adaptive Dragonfly Algorithm (ADA) for load balancing and job scheduling method in the cloud was introduced by Neelima and Reddy [42]. The proposed method was focused on load, execution cost, and completion time. Considering execution time and cost as performance criteria, the suggested way provided better load-balancing outcomes than DA and FA.

Machine learning techniques have been employed in cloud computing to handle complex issues [43]. For example, Reinforcement Learning in combination with Dynamic Consolidation (RL-DC) technique helps in improving resource utilization and minimizing energy utilization in virtual data hubs (Farahnakian et al.) [19].

Deep Reinforcement Learning (DRL) helps the VMs get the optimal position in the various data centers of a virtual environment [44]. A hybrid algorithm of Q-learning and modified PSO (QMPSO) helps balance loads of VMs in virtual networks. The said algorithms reduce the waiting time and power consumption. QMPSO also improves the throughput and makespan [45].

A methodology in green computing for resource allocation was built using fuzzy logic and a reinforcement learning mechanism to deploy resources [46]. The CPU usage at the data center determines the efficiency, hence calculating the Data Centre Infrastructure Efficiency (DCiE) and Power Center Infrastructure efficiency (PUE). CloudSim simulations show that this substructure may deliver operative execution for high data center energy proficiency while avoiding SLA violations. The hybrid deep reinforcement learning with ACO resource allocation and task scheduling aims to optimize idle resources by employing a binary in-order traversal tree with weighted values and reducing its completion time. Rugwiro et al. used a DRL method to discover idle assets and ACO to determine the best VM for every work while job scheduling [47].

As previously stated, establishing effective strategies for job scheduling and resource selection in virtual environment settings were discovered to be critical. Numerous revisions spend a lot of work arranging activities or assigning appropriate resources for respective exercises while seeing single, bi, and multi-objective scheduling. As a result, this work presented multifaceted optimization of the task scheduling issue. In contrast, earlier research mainly concentrated on the purposes of time (makespan or execution) and cost, as both purposes suit the users' expectations. However, different circumstances or goals must be considered to operate with cloud computing. Also, cloud load balancing must be addressed. To enhance cloud performance, a hybrid meta-heuristic method has been devised. Nonetheless, some strategies perform poorly in global search, while others do poorly in local search optimization. Due to the exploratory behavior, the ABC algorithm can help solve the multifaceted task scheduling issue.

In contrast, the spectator bee's exploitative conduct was lacking. Because Q-learning may enhance solution quality, reinforcement learning can help tackle this problem. As a result, we offer a solution for solving the multifaceted job scheduling issue that employs Q-learning with the ABC algorithm (QMFOABC). QMFOABC assists in determining the sequence of tasks for acceptable available resource settings to discover the best job-scheduling strategy. QMFOABC also has a load-balancing mechanism.

The strengths and weaknesses of prior work are shown in Table I

3. Identification And Optimization Of The Problem

The scheduling procedure is critical for load balancing, resource time, usage, and latency in the cloud. Job scheduling can be defined as:

- Let VM represent the set of virtual machines. Therefore $VM = \{vm_1, vm_2, vm_3, \ldots, vm_i, \ldots, vm_n\}$, where vm_i is the i^{th} virtual machine, and i ranges from 1 to n. Each virtual machine contains resources like bandwidth, RAM, CPU, etc.
- Let T represent the set of allocated tasks. Hence $T = \{t_1, t_2, t_3, \ldots, t_i, \ldots, t_m\}$, where t_i is the i^{th} task and m are the number of autonomous jobs accomplished on the virtual machines. Every task must contain the required resources, like CPU, memory, instruction sets, etc.

The above input is optimized scheduling, i.e., mapping of m tasks and n virtual resources, which helps increase resource utilization, maximum load balance, minimize cost, and decrease makespan.

Multifaceted optimization challenges require different objective functions. For example, if $OF_i(x)$ is the i^{th} objective function and TOF is the total number of objectives, then a set of objective functions can be defined as $OF(x) = \{OF_1(x), OF_2(x), OF_3(x), \ldots, OF_i(x), \ldots, OF_{TOF}(x)\}$. These challenges are under non-deterministic problems. Hence no single solution exists in these cases.

By using the multifaceted scheduling strategies, the proposed scheduling method helps improve load balancing between different virtual machines and enhances the throughput of the VM, resource utilization, and scheduling optimization. The suggested algorithm helps appropriately schedule tasks by finding suitable environments using evaluation factors like reliability, makespan, resource utilization, carbon emission, profit, failure rate, time (flow, waiting, completion), energy consumption, and cost. Furthermore, the fitness function is required to get the optimal scheduling and load-balancing solution. The weighted sum approach [48] was used to solve the multifaceted optimization problem. Considering weights as decision-makers, the weighted sum approach helps change multifaceted problems into single-objective optimization problems.

The following goals are covered in this research:

The Ist goal is defining the state of the task's execution time or makespan, i.e., the time required by the system to execute its task. The makespan helps the task to be performed prematurely by decreasing the execution time and hence plays an essential role in multifaceted scheduling. Makespan determined the time of execution for completing the job through the VM. The time of execution is directly proportioned to the makespan, i.e., if the task distribution is not done correctly, then the execution time will be high and result in a higher value of makespan. So, to get better performance, the tasks must be distributed appropriately, resulting in a lower value of execution and makespan.

Let vm_i ($vm_i \in VM$) the virtual machine be assigned with each task t_j ($t_j \in T$). Each task assigned to vm_i is denoted as t_{ji}. Therefore task on vm_i is represented as $vm_i = \{t_{pi}, t_{qi}, \ldots, t_{ri}\}$.

$$TotalExecutionTime(TET) = totaltask * effectiveCPI * \frac{1}{f} \tag{1}$$

where effective CPI (ECPI)

$$ECPI = \frac{totalclockcyclesneededtoexecutealltask}{totalnumberoftask} \tag{2}$$

and f is the CPU rate

$$ECPI_{ji} = \frac{t_{pi} * CPI(t_{pi}) + t_{qi} * CPI(t_{qi}) + +t_{ri} * CPI(t_{ri})}{t_{pi} + t_{qi} + \ldots . + t_{ri}} \tag{3}$$

$$TET(vm_i) = t_{pi} + t_{qi} + \ldots + t_{ri} * ECPI_{ji} * \frac{1}{f_i} \tag{4}$$

or

$$TET(vm_i) = length(t_j) * \frac{1}{f_i} \tag{5}$$

where length(task) is the no.of instructions.

The minimum value of $TET(vm_i)$ is the $min\_makespan$ and can be calculated by Equation 6.

$$min\_makespan = minTET(vm_i); \tag{6}$$

where i=1 to n.

The value of an objective function of makespan can be calculated by Equation 7.

$$OF_1 = \frac{min\_makespan}{makespan} \tag{7}$$

TABLE I. COMPARISON OVERVIEW OF EXISTING LITERATURE

Reference work	Methodology	Performance Metric	Strength and Weakness
[20]	Gray Forecasting Model	Energy consumption, QoS	Use of prediction-based algorithms in a heterogeneous environment.Limitation: Only heuristic algorithms were used, which discussed only the limited parameters like response time and availability of resources.
[23]	MCTE	Response time, cost	The proposed strategy helps to reduce the task completion and cost. Limitation: Priority is only given to response time and other parameters are missing
[24]	RALBA	Makespan, throughput, resource utilization.	The results of the research work help in the utilization of the resources and the QoS parameters. Limitation: research work lacks discussion of the parameters like load balancing and energy consumption.
[26]	CDMWS	Execution, response time.	Research focuses on the reduction of execution time and fails to justify the other QoS parameters.
[27]	MOO	Resource allocation and utilization	The research helps utilize energy with efficient resource allocation but lacks QoS parameters like makespan and throughput.
[28]	IMOACO	Makespan	Research helps to minimize makespan but ignores the QoS parameters like total cost and load balancing.
[30]	Dragonfly	Execution – response time, load balancing.	Research helps in finding improved resource allocation strategy by considering execution time but lacks in QoS parameters like throughput, makespan, etc.
[31]	HABC	Makespan and load balancing.	Research helps improve resource scheduling. Limitation: lack of analysis of other deadline constraints.
[33]	MDVMA	Makespan, energy consumption.	Research helps in optimizing task scheduling. Limitation: Limited parameters related to energy consumption are discussed.
[38]	HGA-ACO	Throughput, Response time, completion time.	The research helps in task allocation and discusses the QoS parameters like throughput and execution time but also ignores the makespan, energy consumption, etc.
[33]	TSMGWO	Makespan	The results parameters are compared with different meta-heuristic techniques but lack many QoS parameters like energy consumption, compilation time, cost, etc.
[40]	Cuckoo search optimization	Cost, makespan	Research helps understand the concept of resource scheduling by considering the parameters of cost and makespan. The limitation of the research is exposed QoS parameters like energy, throughput processing time, etc.
[41]	GA	Load balancing, task completion time	The research shows good load-balancing ability and minimal task completion time. Limitation: the research fails to consider the uncertainty and dynamics of cloud computing. Hence, research is very far away from focusing on task scheduling issues of real cloud computing.
[42]	Dragon Fly	Load, completion time, cost	In this, the author discussed the parameters of load and processing time cost. Limitation: Research lacks discussion on energy consumption and power demand.
[43]	Machine learning with cloud computing	Cloud security threats	In this author discussed the use of ML to overcome the different cloud security issues with the help of reinforcement learning. Limitation: Other QoS parameters are not considered.
[45]	MPSO	Load balancing	In this paper, the author discussed dynamic load balancing by using the Q learning technique and hence improves results in overall enhancement throughput of the system. Limitation: The research work can be improved by using other reinforcement techniques in cloud computing.

After obtaining the first goal, the IInd goal is to calculate the total cost. The total cost is the cost of processing the task and is expressed in terms of bandwidth usage, memory usage, and CPU cost. When a task t_j is processing on the virtual machine vm_i the estimated cost Ecost can be calculated by Equation 8.

$$Ecost(t_{ji}) = (p_1 * ET(t_{ji})) + (p_2 * ET(t_{ji})) + (p_3 * ET(t_{ji})) \tag{8}$$

where $ET(t_{ji})$ is the execution time of (t_j) processing on (vm_i) and p_1 is bandwidth usage per unit,p_2 is the memory usage per unit and p_3 is the CPU cost per unit, respectively.

So the addition of all costs of task execution on VMs can be calculated by Equation 9.

$$TEcost = \sum_{j=1,i=1}^{m,n} Ecost(t_{ji}) \tag{9}$$

The minimum cost of the set $TEcost$ is only for t_j and is the minimum value in the set of allocated task T is executed on VM. Therefore min_TEcost can be calculated by Equation 10

$$min\_TEcost = \sum_{t_j \in T} minEcost(t_j) = \sum_{t_{ji} \in T} minEcost(t_{ji}) \tag{10}$$

The value of the objective function in terms of cost is

$$OF_2 = \frac{min\_TEcost}{TEcost} \tag{11}$$

The third goal is to find the consumption of memory and CPU sent to various processing units. For a particular task sent to vmi, a load of memory (M_L) is calculated as

$$M\_L_i = \frac{M\_R_j}{M\_T_i} + M\_RB_i \tag{12}$$

Where M_L_i is the load of memory on the i^{th} machine, M_R_j is the memory required for the demand of the j^{th} job, M_T_i is the maximum memory, and M_RB_i is the memory required before execution of the j^{th} task.

The next task is to find the CPU load for the j^{th} task on *vmi*. Let's assume that the CPU load on the i^{th} machine is CPU_L_i, CPU_R_j is the CPU requirement for the request of the j^{th} task, CPU_T_i is the total CPU availability, and CPU_RB_i is the CPU usage required before execution of the j^{th} task.

$$CPU\_L_i = \frac{CPU\_R_j}{CPU\_T_i} + CPU\_RB_i \tag{13}$$

Since the memory and CPU are equally important, we assume the equal weightage of memory and CPU. So $(w_1 + w_2) = 1$, where w_1 and w_2 are the weightage of memory and CPU. Therefore, the utilization of a virtual machine (VU_i) can be calculated as

$$VU_i = \frac{w_1}{(1 - M\_L_i)} * \frac{w_2}{(1 - CPU\_L_i)} \tag{14}$$

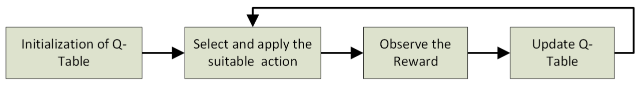
Initialization of Q-Table
Select and apply the suitable action
Observe the Reward
Update Q-Table

Figure 2. Process for Q-learning strategy

The above equation gives the load on the single machine, so if there are n number of virtual machines, then a total load of n machines (TLM_n) will be

$$TLM_n = \sum_{i=1}^{n} VU_i \tag{15}$$

If x is the total no. of hosts in the virtual environment, then the average value of load (AVL) of all virtual machines in the cloud is

$$AVL = \frac{\sum_{i=1}^{x} TLM_n}{x} \tag{16}$$

The difference in each machine's average value and load will result in the objective function in terms of utilization.

$$OF_3 = \sum_{n=1}^{x} \mid AVL - TLM_n \mid \tag{17}$$

The value of the final proposed objective function is

$$OF = (BF_1 OF_1) + (BF_2 OF_2) + (BF_3 OF_3) \tag{18}$$

$BF_1, BF_2, and BF_3$ are the balancing coefficient of makespan, cost, and utilization of resources. For a better solution, the value of the utility function OF must be high.

4. Load Balancing With Hybrid ABC Algorithm

A. *Q-Learning Algorithm*

One of the different machine learning strategies that learn from the environment is Reinforcement Learning (RL). RL allows agents to learn from environmental feedback and helps to react by changing the state regarding penalties or rewards. Q-learning is one of the different types of Reinforcement Learning (RL). The primary goal of RL is to teach the agent via environmental errors and trials. The agent can receive the status of the environment and take a behavior that impacts the environment to achieve a better benefit and know from mistakes. Because of the uncertain environment, the Markov decision procedure (MDP) provides a structure aimed at agent supervisory, and the outcome is occasionally random. The agent selects a similar act given the identical state or condition, but the effect is not necessarily equivalent. The process of Q-learning can be viewed in figure 2.

Let $A = \{a_1, a_2, \ldots, a_n\}$ be the set of actions of the agent and $S = \{s_1, s_2, \ldots, s_m\}$ be the set of states. At time t, the agent chooses an action $a_t \in A$ in a state $s_t \in S$, and a reward of r_{t+1} is awarded if the agent moves to the next state $s_{t+1} \in S$. To complete the task, choosing a proper deed that increases the Q-value of a discrete instance is required.

By picking a suitable action a_t in a state s_t, the Q-value is likely to gain the maximum rewards and is in its best state and can be evaluated as

$$Q(s_t, a_t) = (1-\beta)Q(s_t, a_t) + \beta(r_t + \omega \max_{a+1} Q(s_{t+1}, a_{t+1})) \quad (19)$$

Where β is their learning rate calculated as β=1/(1+total number of visits to state s_t), ω is the deduction aspect ($0 < \omega < 1$) effect on the consecutive instance by the preceding act, and r_t is the reward or penalty corresponding to the action in the instance s_t. Using the greedy approach, the Q-learning strategy will generate an optimal solution from the knowledge recorded in the Q-table. Equation 20 helps in generating the Q-value.

$$Q_{t+1}(s_t, a_t) = Q(s_t, a_t) + \beta(r_t + \omega \max_{a'_t} Q_t(\tau(s_t, a_t), a'_t) - Q_t(s_t, a_t)) \quad (20)$$

Where τ is the conversion factor and $max_a Q(s_{t+1}, a)$ approximates the finest upcoming value. Balancing of load and allocating resources in the cloud are described in Algorithm 1.

Algorithm 1 Q_value_update()

```
1:  Create a Q table by initializing the value of reward r, discount factor $\omega$, and learning rate $\beta$.
2:  Let n be the number of total states, and p are the total actions in each state
3:  for $x = 1$ to $n$ do
4:      for $y = 1$ to $p$ do
5:          $Q(s_x, a_y) = 0$
6:      end for
7:  end for
8:  Move to the next state $s_{t+1}$ by selecting an action $a_x$ from the action set A, $a_x \in A$.
9:  Calculate the new value of the learning rate, reward
10: Update $Q_{t+1}$ using the equation
11: $Q_{t+1}(s_t, a_t)=Q(s_t, a_t) + \beta[r_t + \omega \max_{a'_t}(Q_t\tau(s_t,a_t),a'_t)-Q_t(s_x, a_y)]$
12: Repeat this step for all other states
```

B. Improved ABC Strategy

The ABC strategy [18] is a swarm-intelligence-based meta-heuristic technique. The swarm scheme comprises agents communicating with one another and their surroundings. The agent's purpose within that algorithm is to discover the optimal food origin. The food origin reflects a bee that symbolizes a collection of different possibilities in the search space and every agent.

Based on the roles of bees, the agents in ABC are classified into three bee categories: employed, onlooker, and scout bees. At first, each employed bee identifies an arbitrary food source. After finding the food resource, the employed bees try to discover a new source (food) nearby their existing food position. Subsequently gathering nectar, the employed bee identifies the optimal food origin. The bee will switch to the next food origin only if the new food source is superior to the prior one and shares this information with the onlooker bees. Based on knowledge gained from the employed bees, the onlooker bees decide on new food sources. Onlooker bees are likelier to choose a food source with a lot of food or excellent quality. Therefore, every onlooker bee will seek a different food origin. Finally, they choose a food origin and then shift to another one. The number of iterations is likewise fixed, so if an improved food origin is unavailable, the employed bees who possess the chosen food origin develop scout bees and are in charge of researching the unused food origin in an unexplored region of exploration space.

Individual bee set exhibits diverse exploratory and exploitative activities [49]. For example, a search agent's experimental activity entails looking for an unused food origin in the exploration region. In contrast, exploitative action, on the other hand, entails looking for an improved food origin nearby the existing food supply. We employ Q-learning in this paper to improve our approach and deliver more relevant solutions to challenges. Because the ABC algorithm exhibits strong exploratory and weak exploitative performance, we strive to enhance exploitative actions. The procedure can be summarized as: The ABC approach depicts the position of the food supply with various results to the situation during the startup phase. Primarily, the food supply is produced arbitrarily, as given in Equation (21). The employed bees are then linked to food origin. The Q-starting table's value is 0.

$$y^0_{p,q} = (y^{max}_q - y^{min}_q) * rand(0, 1) + y^{min}_q \quad (21)$$

Where y^{max}_q and y^{min}_q are the upper bound and lower bound of the q^{th} optimization parameter.

Also, The ABC method divides the procedure into three categories: the employed, the onlooker, and the scout bee phase. The method cycles through all three steps till the extreme figure of values is achieved.

During the first phase, employed bees establish the adjacent food source $a\_fs^t_{p,q}$ of the current food source $c\_fs^{-t}_p$ by Equation 22.

$$a\_fs^t_{p,q} = (c\_fs^t_{p,q} - c\_fs^t_{i,q}) * rand[-1, 1] + c\_fs^t_{p,q} \quad (22)$$

Where $a\_fs^t_{p,q}$ is the q_{th} optimization parameter of $a\_fs_p{}^{-t}$ and the index of the food source is denoted by i.

If the fitness function value of the new source is larger than the present food source i.e $fitness(a\_fs_p{}^{-t}) > fitness(c\_fs_p{}^{-t})$ then move to the new location and forget the current site. After moving to a new location, the Q-table value is updated using

Equation (21). In this case, the new food source will receive the reward and the current food source with receiving the penalty. Whereas if the fitness value of the new source is smaller than the present, the current food source will get the reward, and the new source will receive the penalty. Whenever an employee discovers an appropriate food source, the Q-table is updated. As a result, if bee_Emp is the no. of bees employed, Q-table will be updated bee_Emp times. Therefore, the Q-table will be updated bee_Emp times if bee_Emp is the number of bees engaged. The onlooker's bee chooses the employed bee's food sources during the employed bee phase from the Q-table value. The Q-table will be revised by using Equation 23 after the replacement of the old food origin with the fresh food origin, and the onlooker bees carry out this process.

$$a\_fs^t_{p,q} = (c\_fs^t_{p,i} - c\_fs^t_{RFS,i}) * \omega^t_{p,q} * r^t_q + c\_fs^t_{p,i} \quad (23)$$

$\omega \in [-1, 1]$, and $c\_fs^t_{RFS,i}$ is the optimization limit of the ideal food origin produced by arbitrary collection. Onlooker bees exploit available food and upgrade to all extents by altering weight standards instead of providing information on dimensions. Q-values (penalty and reward) are altered at this stage. If a scout bee makes numerous unsuccessful attempts to locate an improved neighbor, it will reject a food source and begin searching for a new one at random. The QMFOABC algorithm is discussed in Algorithm 2.

Algorithm 2 QMFOABC()

```
1: Create the population and compute discrete fitness standards.
2: Configure the following parameters:
       1) Optimal result.
       2) No. of iterations (max).
       3) Population size.
3: Determine the Optimal result.
4: While the halting requirements are met.
   The phase of Employed Bees:
5: Do for all locations, Update employed bee position by equation (22).
6: Estimate the new position.
7: if fitness($a\_fs^{-t}_p$) > fitness($c\_fs^{-t}_p$) i.e if the new position's fitness value is higher.
8: Substitute the new position for the present position and END if of step 7.
9: Compute the possibility, through the onlooker phase revise the Q-table for a certain place and END for of step 5.
   The phase of Onlooker Bees:
10: for all onlookers, do
11: Choose a location depending on the possibility and Q-value.
12: Modify the location of the onlooker bee by Equation (23)
13: Determine the new location's value.
14: If the new location's fitness value is higher.
15: Substitute the new location for the present location and END if of step 14.
16: Revise the Q-table (20) and END for of step 10.
17: Determine the best result (Q-table).
   The phase of Scouting Bees:
18: For each location, do
19: Abandon the efficient results and produce new ones at arbitrary.
20: Update the Q-table (20) and END for of step 18.
21: END while.
```

5. Investigational Work And Its Discussion

The input parameters and the outcomes of the experimentation are defined in this segment. Next, we distinguished the proposed approach (QMFOABC) with LJF, FCFS, and the Max-Min task scheduling algorithm to evaluate its performance [21]. We also determined the QMFOABC method with well-known meta-heuristic job scheduling approaches as the CS, PSO, and HABC_LJF algorithms. Finally, a test scenario, standard datasets, variable scenarios for the suggested methodology and the comparative approaches, experimentational outcomes, and the temporal complexity of the QMFOABC technique were used to assess the performance of the recommended approach.

A. Setup For Simulation

This section describes an experiment that was carried out to compare the performance of the suggested approach (MOABCQ) with existing methods for task scheduling in a heterogeneous virtual situation. A test case was constructed and established via CloudSim 3.0.3 [50]. CloudSim is a tool for simulating virtual resources, simulation, modeling and experimenting with virtualized cloud data. This experimentation used a PC equipped with RAM 16 GB and i7 Intel 8750H CPU (clocked at 2.20 GHz). A simulated situation was used in this experiment to illustrate the suggested strategy's efficacy in scheduling and load balancing. The simulation setup of this experimentation is given in Table II

B. Datasets Scale

Different datasets were used to assess the scheduling performance of the suggested method: 1) Synthetic workload dataset, 2) GoCJ – Goole Cloud Jobs dataset [51], and 3) Random dataset. The parameters and dataset are shown in Table III.

C. Parameter Set For The Recommended Technique And Their Evaluation Approach

As previously stated, parameter setting influences algorithm efficiency, which relies on the size or type of the problem. As a result, we must tune the parameters to confirm we utilize the proper parameters for the issue type and dataset. The table below shows how the parameters are defined. The settings for the experiment's application of the Kruekaew and Kimpan's ABC algorithm [31] are listed in Table IV. Finally, the MOCS, Multi-objective particle

TABLE II. SIMULATION SETUP

Type	Parameter	Value
Host	No. of Host	40
	Bandwidth	10GB/s
	RAM Requirement	16 GB
	Virtual Machine Monitor	Xen
	MIPS	180650
	Storage	2TB
Data Centre	No. of Data Centre	1
	VM Scheduler	Time Shared
	VM Monitor	Xen
	Cost per storage	0.1-1.0
	Cost per Memory	0.1-1.0
Task Cloudlet	Task length	1k-900k
	No. of Task	200-1000
Virtual Machine	No. of VM required	10-100
	Processor Speed	3500-100000 MIPS
	Memory Required	1-4GB
	Bandwidth required	1000-10000
	Cost per memory	0.1-1.0
	Cost per Storage	0.1-1.0
	Cloudlet Scheduler	Time Shared
	No. of Processing Elements	1
	VM monitor	Xen

TABLE III. THE DATASET AND ITS PARAMETERS

Data Set No.	Data set Name		Task Size in MIs		No. of Task
1.	Synthetic Workload Dataset	1 - 45000	Tiny jobs	1 - 250	1k
			Small size jobs	800 - 1200	
			Medium size jobs	1800 - 2500	
			Large size jobs	7000 - 10000	
			Extra-large size jobs	30000 - 45000	
2.	GoCJ Dataset	15000 – 900000	Small size jobs	15000 – 55000	
			Medium size jobs	59000 – 99000	
			Large size jobs	101000 – 135000	
			Extra-large size jobs	150000 – 337500	
			Huge size jobs	525000 - 900000	
3.	Random Dataset	1000- 70000			

swarm optimization (MOPSO), and HABC approaches were contrasted using the suggested technique to assess scheduling performance.

D. Experimentation And Outcomes

The standard dataset is used in this segment to appraise investigational results in views of DI, makespan, throughput, cost, and ARUR [24] for a recommended scheduling performance. In this experimentation, we evaluated the concert of the suggested approach (QMFOABC). First, we joined the QMFOABC strategy with the LJF and FCFS to generate ”QMFOABC_LJF”and QMFOABC_FCFS,” respectively. Then, we compared them with the MOCS, MOPSO, HABC_LJF [31], Max-Min task scheduling algorithm [21], and FCFS scheduling. The average of the outcomes for each dataset after 20 iterations is in the following sections.

A assessment of the recommended method’s concert in relationships of makespan is shown in the first part. 100 VMs were given in this trial, and the system was provided 200 to 1000 jobs. The experiment outcomes are displayed in Figure 2,3,4. According to the experimental findings in Figure 3, the QMFOABC technique was exposed to lesser the average makespan improved than the overhead conversed techniques when the random dataset was studied. However, MOCS providing the least average makespan of the methods revised on 400 datasets. Max-Min, MOPSO, Q-Learning,

TABLE IV. PARAMETER SET FOR THE RECOMMENDED TECHNIQUE AND THEIR EVALUATION APPROACH

Algo Name	Parameter used	Values
QMFOABC	No. of Population(n)	950
	No. of locations visited among n explored locations (m)	94
	No. of best sites (c)	1
	No. of Emp bees	195
	No. of Onlooker bees	755
	Total repetitions	1000
HABC [31]	No. of Population(n)	950
	No. of locations visited among n explored locations (m)	94
	No. of best sites (c)	1
	No. of Emp bees	195
	No. of Onlooker bees	755
	Iterations maximum	1000
MOCS [40]	Population size	20
	Abandon probability	0.25
	Step size	0.01
	Maximum iterations	1000
MOPSO [52]	Particle size	100
	Weight min	0.1
	Weight Max	0.9
	Self-recognition coefficient	1.49
	Maximum iterations	1000

HABC_LJF, QMFOABC_FCFS, QMFOABC_LJF, and FCFS finished at 82.39%, 23.41%, 20.77%, 13.85%, 8.82%, 3.77%, 3.69% less time than MOCS respectively.

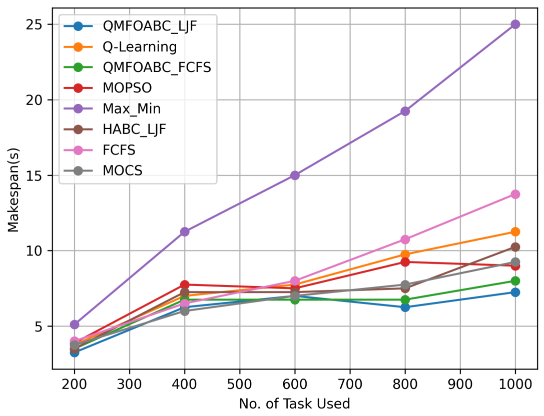

Figure 3. Performance comparison of Makespan on random dataset

If we consider the insight of QMFOABC, it can be found that for 0 to 1000 jobs (with an interval difference of 200), QMFOABC_LJF provides a lesser average makespan than QMFOABC_FCFS. On average, QMFOABC_FCFS had a makespan 0.51% shorter than QMFOABC_LJF for 600 jobs.

QMFOABC_LJF provided the shortest average makespan when since the investigational findings in Figure 4 and Figure 5 utilized the GoCJ and Synthetic workload datasets, respectively. The makespan average of MOABCQ LJF was 8.21%, 30.26%, 34.94%, 42.25%, 46.17%, 92.15%, and 117.80% less than that of QMFOABC FCFS, MOCS, HABC LJF, MOPSO, Q learning, FCFS, and MaxMin by means of the dataset GoCJ respectively. The makespan average of MOABCQ LJF was 2.97%, 10.76%, 15.35%, 22.87%, 25.53%, 98.61%, and 150.75% less than that of QMFOABC_FCFS, MOCS, HABC_LJF, Q-Learning, MOPSO, FCFS, and Max_Min when using the Synthetic Workload dataset, respectively. Therefore, except for the ratio of the makespan average decrease, it can be shown that QMFOABC_LJF beat the other techniques when both datasets were run.

Overall analyses revealed that the suggested approach might offer the shortest value of makespan as QMFOABC_LJF can assign the work to the most suitable resource. These findings lead us to conclude that the suggested approach can effectively distribute resources within the system.

The efficiency of the approach is compared in the second part for throughput, i.e., the no. of jobs completed in a given amount of time. Figure 5,6,7 shows the experimental results of assessing the effectiveness of a random dataset, which reveal that QMFOABC outperformed other algorithms in terms of throughput. However, during comparison, the MOCS had gained the max value while the others at max-min (45.17%), MOPSO (18.97%), HABC_LJF (17.20%), Q-learning (12.16%), QMFOABC_FCFS (8.10%), QM-

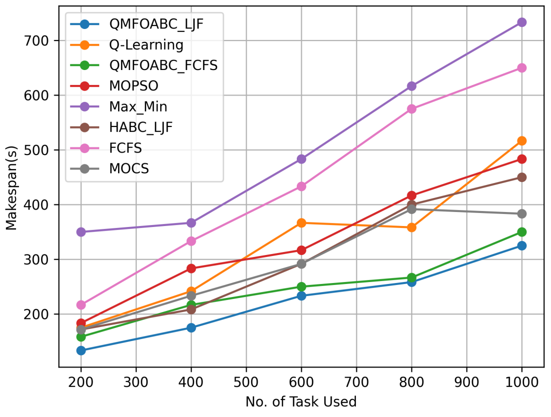

Figure 4. Performance comparison of Makespan on GoCJ dataset

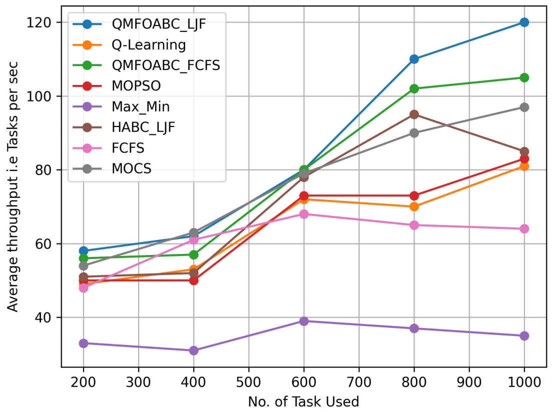

Figure 6. Performance comparison of throughput on random dataset

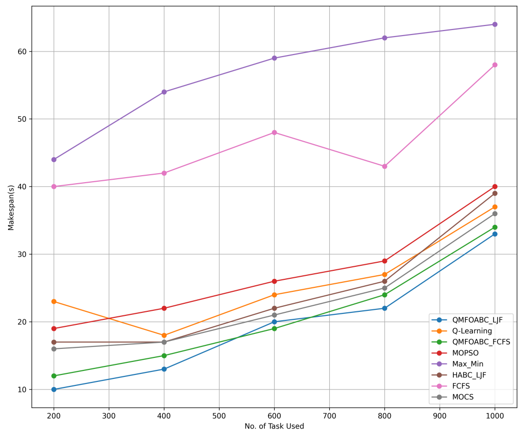

Figure 5. Performance comparison of Makespan on Synthetic workload dataset

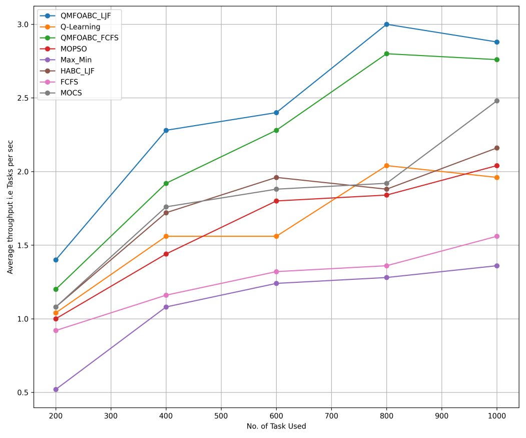

Figure 7. Performance comparison of Throughput on GoCJ dataset

FOABC_LJF (3.63%) and FCFS (3.56%) respectively. Considering the QMFOABC, QMFOABC_LJF has a larger throughput as compared to QMFOABC_FCFS in 200, 800, and 1000 jobs. QMFOABC_FCFS offered 0.50% more throughput than QMFOABC_LJF for 600 processes.

GoCJ and Synthetic workload conducted throughput tests, and the outcomes presented that QMFOABC achieved improved efficiency. The outcomes are presented in Figures.7 and 8. The throughput offered by QMFOABC LJF was 6.14% higher than QMFOABC_FCFS during comparison. The total throughput test trials lead us to conclude that QMFOABC_LJF can assign tasks to appropriate resources consistent with the testing outcome in the first section.

In the third part, the suggested method's effectiveness is compared to the ARUR, and other crucial needs for job scheduling in the structure. This experimentation used 3 datasets with one hundred virtual machines and one thousand jobs. Table V presents the experimentation outcomes.

We discovered that QMFOABC provided greater ARUR values than the other approaches. The algorithms utilized in this experiment that produced results similar to those of the ARUR were MOCS, MOPS, HABC_LJF, Q-learning, QMFOABC LJF, and QMFOABC FCFS, in contrast to FCFS and Max-Min. QMFOABC LJF produced higher ARUR values in tests using the random dataset than MOPSO, HABC_LJF, Q-Learning, MOCS, and QMFOABC_FCFS, with values of 21.53%, 20.94%, 18.72%, 14.01%, and 6.15%, respectively.

Compared to HABC_LJF, MOCS, Q-learning, MOPSO, and QMFOABC_FCFS, QMFOABC_LJF

TABLE V. PERFORMANCE EVALUATION IN VIEW OF ARUR ON DIFFERENT DATASETS

	Task Scheduling Methods							
Dataset	QMFOABC_LJF	QMFOABC_FCFS	MOCS	MOPSO	Q-learning	Max-Min	HABC_LJF	FCFS
Random	0.822	0.772	0.708	0.647	0.67	0.21	0.652	0.342
GoCJ	0.772	0.722	0.682	0.664	0.59	0.17	0.672	0.232
Synthetic work-load	0.811	0.806	0.771	0.732	0.765	0.188	0.733	0.515

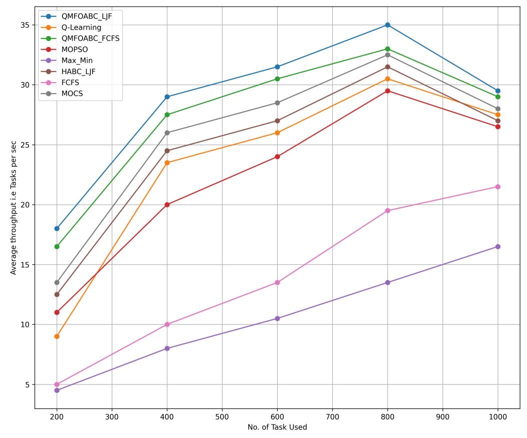

Figure 8. Performance comparison of Throughput on Synthetic workload dataset

provided higher ARUR values using the GoCJ dataset at 6.31%, 11.34%, 12.63%, 13.61%, and 22.99%. Using the Synthetic Workload dataset for testing, the QMFOABC_LJF approach produced ARUR values that were greater than those of the MOPSO, HABC_LJF, Q-Learning, MOCS, and QMFOABC_FCFS methods, at 9.86%, 9.74%, 5.74%, 4.99% and 0.62% respectively.

The three datasets were tested, and it was discovered that QMFOABC_LJF offers the best ARUR value for further approaches. Consequently, we can infer that the MOABCQ_LJF technique may effectively plan system jobs and support fairly allocating work among the accessible resources, which can aid in the structure's maintenance of stability approach.

In the fourth part, the suggested strategy's performance is compared in relation to the degree of imbalance (DI), which is used to evaluate the system's balancing load. The same datasets used in the earlier experiment sections were also used for these trials. One hundred virtual machines and 200, 400, 600, 800, and 1000 jobs were used to test these studies.

Compared to MOCS, MOPSO, HABC_LJF, Q-Learning, FCFS, and Max-Min, the suggested method (QM-FOABC) was evaluated, and the outcomes are shown in Table 4. The DI values of QMFOABC_LJF were the least while tested on a dataset of GoCJ and random, demonstrating that QMFOABC_LJF can distribute jobs more evenly among available resources in the system than the other approaches. However, we discovered that QM-FOABC_LJF provided a lesser DI value as compared to other techniques while utilizing the Synthetic workload dataset and setting jobs in the scheme equal to 200 to 1000 jobs. QMFOABC_FCFS has the lowest DI value except tested with 400 jobs. With a task distribution rate of 4.37%, QMFOABC_FCFS is superior to QMFOABC_LJF.

The QMFOABC approach was found to evenly allocate the work across the existing resources in the structure, which led to a lower DI, according to the testing with all three datasets. Furthermore, an in-depth analysis of the suggested approach demonstrates that QMFOABC LJF outperformed the other examined methods. Nevertheless, it relies on the test dataset.

The last part compares the effectiveness of the suggested strategy from a financial standpoint to evaluate the expenses or overheads associated with cloud computing using three datasets. The experiment outcomes are displayed in Figures 8,9,10. According to Figure.9 QMFOABC was shown to cut costs more than the above-discussed approaches while utilizing the random dataset. Furthermore, the comparison indicates that QMFOABC_LJF had about 3.38% lower cost than QMFOABC_FCFS. The QMFOABC_LJF method, however, costs 4.888% higher than the QMFOABC_FCFS procedure with 1000 jobs.

The findings in Figure.10 showed that QMFOABC could lower costs more than the other approaches while trying with the dataset of GoCJ, just as it did with the random dataset. For example, we observed that the QM-FOABC_LJF strategy has a low cost compared to the QMFOABC_FCFS method at about 20.9% when associating QMFOABC_FCFS with QMFOABC_LJF and testing on different jobs. Additionally, it was discovered that the QMFOABC_FCFS technique cost 0.48% more than the QMFOABC_LJF method while experimenting with 600 and 1000 jobs.

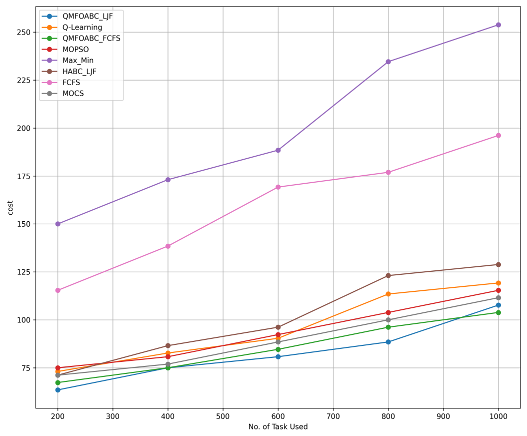

Figure 9. Performance comparison of throughput on random dataset

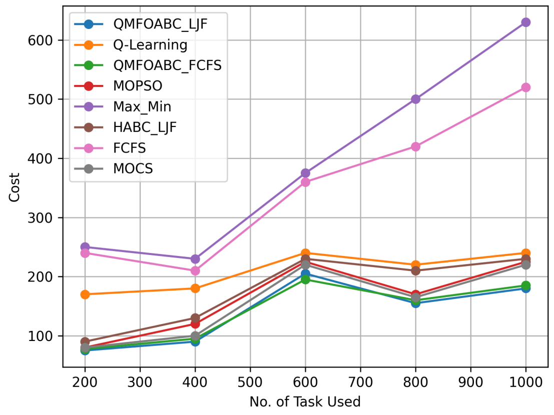

Figure 11. Performance comparison of Throughput on Synthetic workload dataset

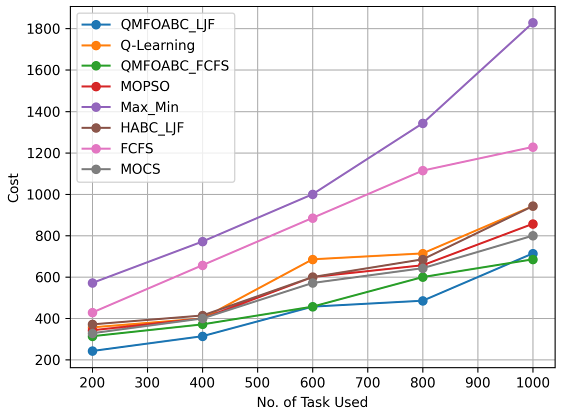

Figure 10. Performance comparison of Throughput on GoCJ dataset

The outcomes in Figure 11 demonstrate that, like testing with the first two datasets, the QMFOABC approach is less expensive compared to approaches when testing with the dataset of Synthetic workload. The QMFOABC_LJF strategy was found to have a low cost as compared to the QMFOABC_FCFS algorithm, excluding 400 jobs, when QMFOABC_FCFS has a low cost than the QMFOABC_LJF technique at 1.90%. The tasks in the system were made to equal 200, 600, 800, and 1000.

The suggested approach of QMFOABC minimizes costs more than the other comparative methods, per the testing with all three datasets. But when we compared QMFOABC LJF with QMFOABC FCFS, we discovered that QMFOABC LJF could be used more effectively for task scheduling given the system's current resources than QMFOABC FCFS. This is because QMFOABC LJF has a lower percentage when compared to the two approaches' different percentages. But it relies on the testable dataset.

6. TIME COMPLEXITY OF QMFOABC

The QMFOABC method's temporal complexity may be determined as follows: an initial population of n bees is supplied in ABC. As a result, it takes n iterations to discover suitable virtual machines in the cloud and n iterations to update the figures in the Q-table. Consequently, the complexity will be the big-oh(n) of QMFOABC. The temporal complexity is equivalent to k * O. Ignoring k, the entire time complexity of QMFOABC is equal to big-Oh.

7. Conclusion

This paper recommends the QMFOABC approach for multi-objective optimization scheduling in a heterogeneous virtual environment. This approach considered selecting suitable VMs after evaluating each VM's fitness. Heuristic methods using FCFS and LJF were also given. The recommended approach signifies improving the scheduling and resource management, hence increasing virtual machine (VM) performance while balancing the load between them. The key goal is to address the constraints of concurrent elements like makespan, cost, and resource utilization. In comparison to various other algorithms such as Max-Min, FCFS, Q-Learning, HABC_LJF, MOPSO, and MOCS, this method helps to balance the workload among the system resources that are available and improves in minimizing makespan, lowering cost, and increasing throughput and ARUR. Finally, tests were run on a variety of datasets to gauge how well the suggested algorithms worked.

In comparison to the previously stated scheduling strategies, the recommended technique aids in improving the drop values of through, cost, DI, makespan drop, and ARUR while also aiding in load-balancing jobs with the system's current resources. The results of the trial show that the suggested technique outperformed the alternatives. Although we observed that the QMFOABC LJF technique is the best we noticed few limitations too, like not all test datasets support the optimization of system performance and some adjustment is needed with a parametric setting.

Job scheduling in edge/fog/multi-virtual situations may present difficult but lucrative challenges in the future. We offer a method for scheduling arrangements that functions well in various contexts. It is also feasible to employ other machine-learning techniques. The performance of the suggested scheme compared to the QMFOABC technique may also be assessed in a practical setting.

References

[1] S. Sharma and N. K. Pandey, "Improved task scheduling strategy using reinforcement learning in cloud environment," in *2022 2nd International Conference on Innovative Sustainable Computational Technologies (CISCT)*. IEEE, 2022, pp. 1–5.

[2] D. Ardagna, G. Casale, M. Ciavotta, J. F. Pérez, and W. Wang, "Quality-of-service in cloud computing: modeling techniques and their applications," *Journal of Internet Services and Applications*, vol. 5, no. 1, pp. 1–17, 2014.

[3] K. Psychas and J. Ghaderi, "On non-preemptive vm scheduling in the cloud," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–29, 2017.

[4] S. Crago, K. Dunn, P. Eads, L. Hochstein, D.-I. Kang, M. Kang, D. Modium, K. Singh, J. Suh, and J. P. Walters, "Heterogeneous cloud computing," in *2011 IEEE International Conference on Cluster Computing*. IEEE, 2011, pp. 378–385.

[5] R. Messier, *Collaboration with cloud computing: Security, social media, and unified communications*. Elsevier, 2014.

[6] O. Alsaryrah, I. Mashal, and T.-Y. Chung, "Bi-objective optimization for energy aware internet of things service composition," *IEEE Access*, vol. 6, pp. 26 809–26 819, 2018.

[7] L. Liu, M. Zhang, R. Buyya, and Q. Fan, "Deadline-constrained coevolutionary genetic algorithm for scientific workflow scheduling in cloud computing," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 5, p. e3942, 2017.

[8] X. L. ZhangjunWu, Z. Ni, D. Yuan, and Y. Yang, "A market-oriented hierarchical scheduling strategyin cloud workflow systems," 2011.

[9] D. Yagyasen, M. Darbari, P. K. Shukla, and V. K. Singh, "Diversity and convergence issues in evolutionary multiobjective optimization: application to agriculture science," *IERI Procedia*, vol. 5, pp. 81–86, 2013.

[10] J. D. Knowles and D. W. Corne, "Approximating the nondominated front using the pareto archived evolution strategy," *Evolutionary computation*, vol. 8, no. 2, pp. 149–172, 2000.

[11] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjective optimization," in *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence*. Ieee, 1994, pp. 82–87.

[12] S. Basu, M. Karuppiah, K. Selvakumar, K.-C. Li, S. H. Islam, M. M. Hassan, and M. Z. A. Bhuiyan, "An intelligent/cognitive model of task scheduling for iot applications in cloud computing environment," *Future Generation Computer Systems*, vol. 88, pp. 254–261, 2018.

[13] F. Luo, Y. Yuan, W. Ding, and H. Lu, "An improved particle swarm optimization algorithm based on adaptive weight for task scheduling in cloud computing," in *Proceedings of the 2nd International Conference on Computer Science and Application Engineering*, 2018, pp. 1–5.

[14] I. Alharkan, M. Saleh, M. A. Ghaleb, H. Kaid, A. Farhan, and A. Almarfadi, "Tabu search and particle swarm optimization algorithms for two identical parallel machines scheduling problem with a single server," *Journal of King Saud University-Engineering Sciences*, vol. 32, no. 5, pp. 330–338, 2020.

[15] D. Karaboga *et al.*, "An idea based on honey bee swarm for numerical optimization," Technical report-tr06, Erciyes university, engineering faculty, computer ..., Tech. Rep., 2005.

[16] B. Akay and D. Karaboga, "A modified artificial bee colony algorithm for real-parameter optimization," *Information sciences*, vol. 192, pp. 120–142, 2012.

[17] X. Li, Z. Peng, B. Du, J. Guo, W. Xu, and K. Zhuang, "Hybrid artificial bee colony algorithm with a rescheduling strategy for solving flexible job shop scheduling problems," *Computers & Industrial Engineering*, vol. 113, pp. 10–26, 2017.

[18] D. Karaboga and B. Gorkemli, "A combinatorial artificial bee colony algorithm for traveling salesman problem," in *2011 International Symposium on Innovations in Intelligent Systems and Applications*. IEEE, 2011, pp. 50–53.

[19] F. Farahnakian, P. Liljeberg, and J. Plosila, "Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning," in *2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE, 2014, pp. 500–507.

[20] S. Ismaeel, R. Karim, and A. Miri, "Proactive dynamic virtual-machine consolidation for energy conservation in cloud data centres," *Journal of Cloud Computing*, vol. 7, no. 1, pp. 1–28, 2018.

[21] Y. Mao, X. Chen, and X. Li, "Max–min task scheduling algorithm for load balance in cloud computing," in *Proceedings of International Conference on Computer Science and Information Technology: CSAIT 2013, September 21–23, 2013, Kunming, China*. Springer, 2014, pp. 457–465.

[22] G. Patel, R. Mehta, and U. Bhoi, "Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing," *Procedia Computer Science*, vol. 57, pp. 545–553, 2015.

[23] H. Zhang, J. Shi, B. Deng, G. Jia, G. Han, and L. Shu, "Mcte: Minimizes task completion time and execution cost to optimize scheduling performance for smart grid cloud," *IEEE Access*, vol. 7, pp. 134 793–134 803, 2019.

[24] A. Hussain, M. Aleem, A. Khan, M. A. Iqbal, and M. A. Islam, "Ralba: a computation-aware load balancing scheduler for cloud computing," *Cluster Computing*, vol. 21, pp. 1667–1680, 2018.

[25] B. Mondal, K. Dasgupta, and P. Dutta, "Load balancing in cloud computing using stochastic hill climbing-a soft computing approach," *Procedia Technology*, vol. 4, pp. 783–789, 2012.

[26] M. Adhikari and S. Koley, "Cloud computing: a multi-workflow scheduling algorithm with dynamic reusability," *Arabian Journal for Science and Engineering*, vol. 43, pp. 645–660, 2018.

[27] B. Shrimali and H. Patel, "Multi-objective optimization oriented policy for performance and energy efficient resource allocation in cloud environment," *Journal of King Saud University-Computer and Information Sciences*, vol. 32, no. 7, pp. 860–869, 2020.

[28] S. Gao, Y. Li, and H. Huang, "A multi-objective task scheduling method based on aco in cloud environment," in *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*. IEEE, 2020, pp. 1554–1559.

[29] Z.-G. Chen, K.-J. Du, Z.-H. Zhan, and J. Zhang, "Deadline constrained cloud computing resources scheduling for cost optimization based on dynamic objective genetic algorithm," in *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2015, pp. 708–714.

[30] S. R. Branch and S. Rey, "Providing a load balancing method based on dragonfly optimization algorithm for resource allocation in cloud computing," *International Journal of Networked and Distributed Computing*, vol. 6, no. 1, pp. 35–42, 2018.

[31] 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. 496–510, 2020.

[32] G.-n. Gan, T.-l. Huang, and S. Gao, "Genetic simulated annealing algorithm for task scheduling based on cloud computing environment," in *2010 International Conference on Intelligent Computing and Integrated Systems*. IEEE, 2010, pp. 60–63.

[33] D. Alsadie, "A metaheuristic framework for dynamic virtual machine allocation with optimized task scheduling in cloud data centers," *IEEE Access*, vol. 9, pp. 74 218–74 233, 2021.

[34] X. Guo, "Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm," *Alexandria Engineering Journal*, vol. 60, no. 6, pp. 5603–5609, 2021.

[35] L. Zuo, L. Shu, S. Dong, C. Zhu, and T. Hara, "A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing," *Ieee Access*, vol. 3, pp. 2687–2699, 2015.

[36] H. He, G. Xu, S. Pang, and Z. Zhao, "Amts: Adaptive multi-objective task scheduling strategy in cloud computing," *China Communications*, vol. 13, no. 4, pp. 162–171, 2016.

[37] R. Jena, "Task scheduling in cloud environment: A multi-objective abc framework," *Journal of Information and Optimization Sciences*, vol. 38, no. 1, pp. 1–19, 2017.

[38] A. Senthil Kumar and M. Venkatesan, "Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment," *Wireless Personal Communications*, vol. 107, pp. 1835–1848, 2019.

[39] D. Alsadie, "Tsmgwo: Optimizing task schedule using multi-objectives grey wolf optimizer for cloud data centers," *IEEE Access*, vol. 9, pp. 37 707–37 725, 2021.

[40] S. H. H. Madni, M. S. A. Latiff, J. Ali, and S. M. Abdulhamid, "Multi-objective-oriented cuckoo search optimization-based resource scheduling algorithm for clouds," *Arabian Journal for Science and Engineering*, vol. 44, pp. 3585–3602, 2019.

[41] S. Pang, W. Li, H. He, Z. Shan, and X. Wang, "An eda-ga hybrid algorithm for multi-objective task scheduling in cloud computing," *IEEE Access*, vol. 7, pp. 146 379–146 389, 2019.

[42] P. Neelima and A. R. M. Reddy, "An efficient load balancing system using adaptive dragonfly algorithm in cloud computing," *Cluster Computing*, vol. 23, pp. 2891–2899, 2020.

[43] U. A. Butt, M. Mehmood, S. B. H. Shah, R. Amin, M. W. Shaukat, S. M. Raza, D. Y. Suh, and M. J. Piran, "A review of machine learning algorithms for cloud computing security," *Electronics*, vol. 9, no. 9, p. 1379, 2020.

[44] L. Caviglione, M. Gaggero, M. Paolucci, and R. Ronco, "Deep reinforcement learning for multi-objective placement of virtual machines in cloud datacenters," *Soft Computing*, vol. 25, no. 19, pp. 12 569–12 588, 2021.

[45] U. Jena, P. Das, and M. Kabat, "Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 6, pp. 2332–2342, 2022.

[46] T. Thein, M. M. Myo, S. Parvin, and A. Gawanmeh, "Reinforcement learning based methodology for energy-efficient resource allocation in cloud data centers," *Journal of King Saud University-Computer and Information Sciences*, vol. 32, no. 10, pp. 1127–1139, 2020.

[47] U. Rugwiro, C. Gu, and W. Ding, "Task scheduling and resource allocation based on ant-colony optimization and deep reinforcement learning," *Journal of Internet Technology*, vol. 20, no. 5, pp. 1463–1475, 2019.

[48] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: new insights," *Structural and multidisciplinary optimization*, vol. 41, pp. 853–862, 2010.

[49] S. Fairee, S. Prom-On, and B. Sirinaovakul, "Reinforcement learning for solution updating in artificial bee colony," *PloS one*, vol. 13, no. 7, p. e0200738, 2018.

[50] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[51] A. Hussain and M. Aleem, "Gocj: Google cloud jobs dataset for distributed and cloud computing infrastructures," *Data*, vol. 3, no. 4, p. 38, 2018.

[52] H. Saleh, H. Nashaat, W. Saber, and H. M. Harb, "Ipso task scheduling algorithm for large scale data in cloud computing environment," *IEEE Access*, vol. 7, pp. 5412–5420, 2018.

Sanjeev Sharma currently pursuing his Ph.D. from DIT University, Uttarakhand, India. He has completed his M.TECH in CSE from Uttaraknad Technical University, Dehradun, Uttarakhand, India. His research interests include Cloud Computing, Algorithms, Networking, ML, and parallel computing.

Dr. Neeraj K Pandey is an Associate Professor in the Department of CSE, Graphic Era Deemed to be University, Uttarakhand. He has over 13 years of teaching and research experience. He has published more than 45 research papers in international journals and conferences. His research areas include Cloud Computing, Machine Learning and Deep Learning, IoT, Image Processing, etc.