IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 29 September 2023, accepted 26 October 2023, date of publication 2 November 2023,
date of current version 17 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3329557

APPLIED RESEARCH

Reinforcement Learning Approach for Optimizing Cloud Resource Utilization With Load Balancing

PRATHAMESH VIJAY LAHANDE[1], PARAG RAVIKANT KAVERI[1], JATINDERKUMAR R. SAINI[1], KETAN KOTECHA[2], AND SULTAN ALFARHOOD[3]

[1]Symbiosis Institute of Computer Studies and Research, Symbiosis International (Deemed University), Pune 411016, India
[2]Symbiosis Centre for Applied Artificial Intelligence, Symbiosis Institute of Technology, Symbiosis International (Deemed University), Pune 411016, India
[3]Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia

Corresponding author: Jatinderkumar R. Saini (saini_expert@yahoo.com)

This work was supported by King Saud University, Riyadh, Saudi Arabia, through the Researchers Supporting Project under Grant RSPD2023R890.

ABSTRACT Cloud computing is a technology that enables the delivery of various computing services over the Internet. The Resource Scheduling (RS) and Load Balancing (LB) mechanisms are essential for the cloud to provide consistent results. The submitted tasks by the users are computed on the cloud platform using its Virtual Machines (VMs). The cloud ensures an ideal LB mechanism, where no VMs will be overloaded or idle. This research paper focuses on this LB mechanism by experimenting in the WorkflowSim environment and computing tasks using the Sipht task dataset. The RS algorithms First Come First Serve (FCFS), Maximum – Minimum (Max – Min), Minimum Completion Time (MCT), Minimum – Minimum (Min – Min), and Round-Robin (RR) are utilized to balance the computational load of VMs. The experiment was conducted in four phases, where the Sipht task dataset varied in task length in each phase. Each phase included sixteen scenarios, where each scenario differed from another by the number of VMs used. The final results of this experiment convey that the load balanced by the algorithms FCFS, Max – Min, MCT, Min – Min, and RR were 51.98 %, 41.71 %, 51.98 %, 59.43 %, and 52.17 %, respectively, across all four phases. Lastly, the Reinforcement Learning (RL) model is suggested to add an intelligence mechanism to LB and optimize the cloud resource utilization using these RS algorithms to provide the best Quality of Service (QoS).

INDEX TERMS Cloud computing, load balancing, performance, reinforcement learning, resource scheduling.

I. INTRODUCTION

The cloud computing platform provides an interface enabling users to process and compute their tasks smoothly [37]. Due to the simplicity, ease of learning, and pay-as-you-go mode, the users opt for the cloud computing platform to compute their tasks rather than on their local machines [37], [38]. Among several reasons, one of the primary reasons why users choose the cloud computing environment is its high availability and high reliability. The Load Balancing (LB) and Resource Scheduling (RS) mechanisms of the cloud play a vital role in ensuring the cloud stays highly available and reliable at all times [7], [37]. The cloud manages this load using the RS algorithms available at its end [15]. Here, the load constitutes the tasks that are submitted for processing and computing, which are ultimately computed on the cloud's Virtual Machines (VMs). However, due to several challenges in task processing and computing, the LB techniques used by the cloud need optimizations, thereby providing a gap for researchers to study the same and suggest various solutions to improve it [38]. Hence, the principal objective of this research paper is to study these RS algorithms from the LB perspective and suggest intelligent solutions to enhance and optimize the LB process. The RS algorithms considered for the study of LB include First Come, First Serve (FCFS), Maximum – Minimum (Max – Min), Minimum Completion Time (MCT),

The associate editor coordinating the review of this manuscript and approving it for publication was Jiachen Yang.

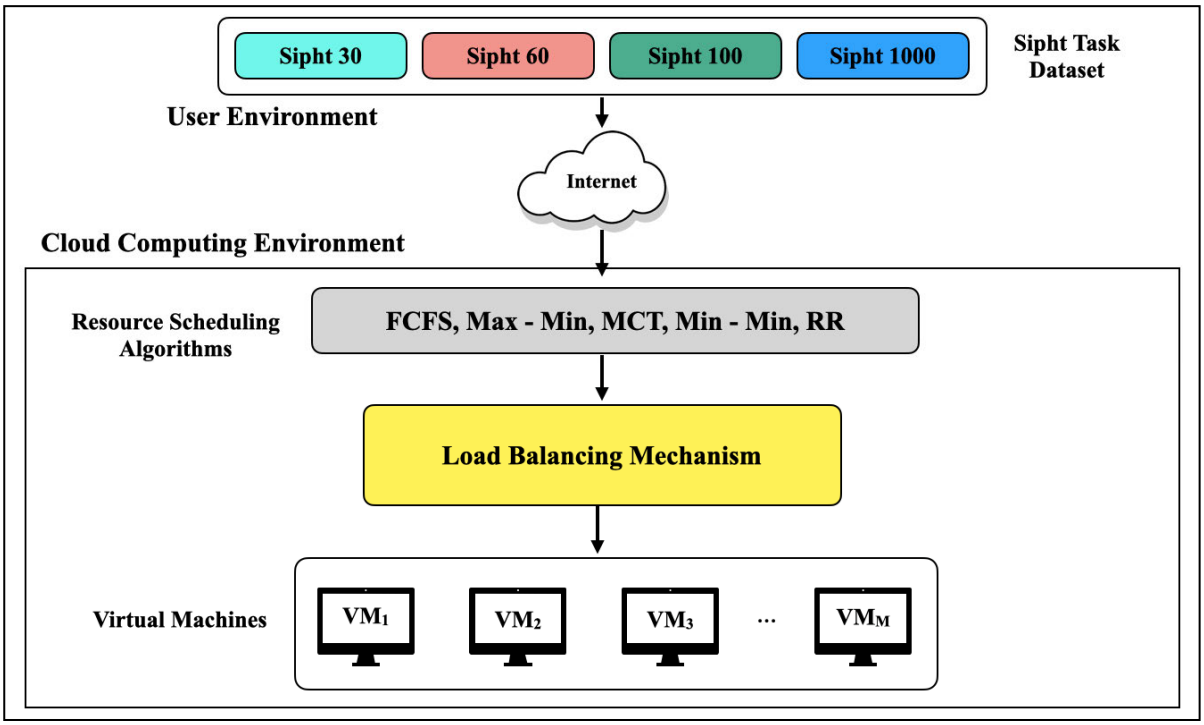

FIGURE 1. Flowchart of the conducted experiment.

Minimum – Minimum (Min – Min), and Round – Robin (RR) [6], [31], [40]. This paper includes an experimental study in a simulation environment where Sipht dataset tasks were processed and computed in several phases. Results are presented in each phase of the experiment. Lastly, the Machine Learning's Reinforcement Learning (RL) [1], [4], [5], [39] method has been suggested to optimize the cloud resource utilization by enhancing the entire LB process and ensure ideal Quality of Service (QoS). Figure 1 depicts the flowchart of the experiment conducted.

The rest of the research paper is organized as follows:

- Related Works presented in section II;
- Experimental Design presented in section III;
- Experimental Results and their Implications presented in section IV;
- Cumulative Results and Issues with the Existing System presented in section V;
- Reinforcement Learning (RL) approach for optimizing cloud resource utilization with Load Balancing (LB) presented in section VI.
- Mathematical Model provided in section VII.
- Conclusion provided in section VIII.

II. RELATED WORKS

LB is not only limited to the cloud computing environment but has been used in several other domains. With an ideal LB mechanism, any system becomes balanced regarding all performance parameters. This LB mechanism, along with the amalgamation of the RL method, is being studied and implemented across several cloud application domains and is an essential metric for any researcher working in the cloud computing domain. The LB problem is an NP-hard problem. Several researchers have designed and presented their work to address and solve this LB issue. To shorten the make-span and improve resource load balances, the researchers have demonstrated a hybrid task scheduling algorithm based on the Max-Min, Min-Min, and genetic algorithms.

The evolutionary algorithm, which determines which tasks should utilize Max - Min and which ones should use Min - Min, is the main application of this hybrid method. The researchers have introduced a brand-new taxonomy for LB and RS in the edge computing environment while taking into account its applications, algorithms, and goals [1]. The architecture of edge computing for information and task processing is also briefly outlined in the second section. Future directions have also been provided in the end for the researchers.

The intelligent LB models, including those based on machine learning (ML) technology, that have been created in HetNets are surveyed in this work [8]. Finally, the existing difficulties in implementing these models and the potential operational implications of LB in the future are discussed. This work presents a load-balanced service scheduling strategy that considers LB when scheduling requests to resources utilizing categories like critical, real-time, or time-tolerant [17]. This method also finds resource failure rates to improve the reliability of all submissions. The researchers developed a system to guarantee that each

cloud node is balanced correctly [9]. A novel algorithm having multiple objectives based on the Artificial Bee Colony Algorithm (ABC) with a Q-learning algorithm - a RL technique that makes the ABC algorithm run more quickly—is proposed as an independent task scheduling approach for cloud computing [10]. According to the experimental findings, MOABCQ-based algorithms beat other algorithms to lower makespan, cost, and degree of imbalance, boost throughput, and utilize resources on average. The researchers have added three algorithms to the LB mechanism [25]. In this study, the authors have introduced a powerful scheduling technique using the RL with parallel optimization of particles to quickly and accurately solve the LB problem and its many parameters [14].

To improve LB, reduce make-span, and maximize makespan, researchers have suggested combining the swarm intelligence algorithm of an artificial bee colony with the heuristic algorithm [29]. The researchers have proposed a fuzzy iterative technique for the cloud computing environment to improve the subpar RS performance [18]. The experimental findings of this algorithm show that it can enhance the cloud platform's management effectiveness and LB mechanism. A machine learning-based technique for virtual machine replacement is provided to balance the load on the host machines [27]. To reduce the load imbalance, which has an impact on scheduling effectiveness and resource usage, a scheduling technique called dynamic resource allocation is proposed [19]. The authors have amalgamated two existing methods and presented an approach focusing on the LB factor in the cloud computing environment [21]. This study suggests Swarm Intelligence (SI) as a LB method for cloud computing [3].

The research's findings are encouraging compared to more established methods, attaining optimized quick convergence and shortening response times. A hybrid soft computing-inspired strategy is presented to obtain an ideal load on the virtual machines by instructing the cloud environment [22]. When compared to other existing algorithms, this method produces better LB outcomes. The researchers have suggested a LB algorithm to balance the cloud load dynamically [23]. This work proposes a decision tree-based multi-objective job scheduling algorithm for use in heterogeneous environments [11]. Compared to the current algorithms, the one being presented produces better outcomes. The researchers have suggested two techniques to achieve LB consolidated systems with the most minor processing, memory, and power [35]. A software-defining network is subject to LB in a cloud setting, as presented in this paper [24]. The researchers have reviewed the literature on LB methods and fault tolerance in cloud computing platforms [12].

To handle LB issues in edge computing, the authors have introduced LB for optimizing resources in a collaborative cloudlet platform [13]. The proposed approach significantly outperforms the traditional edge-based strategy regarding several performance parameters. The dragonfly and firefly algorithms are combined to provide a novel LB work scheduling method [30]. To balance the loads on physical machines and reduce deployment and migration costs, this study developed a LB virtual network function deployment scheme [16]. The researchers have developed a LB strategy to likely assure against resource overloading with VM migration to reduce the LB and overall migration overhead [33]. To balance the burden on the growing number of mobile edge computing servers and query loads, the researchers have introduced a LB technique to the proposed three-layer mobile hybrid hierarchical peer-to-peer architecture [7].

Researchers have devised scheduling algorithms to balance the load and time of resources in the diverse cloud environment [34]. A novel hybrid model that classifies the amount of files in the cloud using file type formatting is being presented to give a better LB solution for cloud computing with extensive data [28]. They are outperformed by the suggested approach, which also offers good performance, scalability, and robustness. This article contains research that considers the workload on edge servers and the travel time needed to offload jobs to these servers [20]. According to numerical simulations, the proposed approach can produce better results than traditional heuristics. This work provides an algorithm to better schedule scientific workflows in cloud-fog environments. In light of the Quality of Service (QoS) job parameters, the priority of VMs, and resource allocation, the proposed approach aims to optimize resources and enhance LB [26]. The authors have reviewed the existing literature based on the cloud's LB feature with a novel approach for fault tolerance [32].

III. EXPERIMENTAL DESIGN

This section represents the detailed design of the experiment conducted. To test and study the mechanism of LB process, an experiment was conducted in the WorkflowSim [36] java-based simulation framework where the dataset of Sipht tasks were submitted for computing on the cloud VMs. This experiment was conducted in four phases. Table 1 depicts the four phases of the experiment.

TABLE 1. Four phases of the conducted experiment.

Phase No.	Dataset	No. of Tasks	Tasks
1	Sipht Thirty	30	$T_1, T_2, T_3, \ldots, T_{30}$
2	Sipht Sixty	60	$T_1, T_2, T_3, \ldots, T_{60}$
3	Sipht Hundred	100	$T_1, T_2, T_3, \ldots, T_{100}$
4	Sipht Thousand	1000	$T_1, T_2, T_3, \ldots, T_{1000}$

Table 1 shows that the first, second, third, and fourth phase includes processing and computing thirty, sixty, hundred, and one thousand tasks of the Sipht dataset, respectively. Each phase of the experiment includes sixteen scenarios, where the tasks are computed on varying numbers of VMs in each scenario. Table 2 depicts the sixteen scenarios of each phase.

TABLE 2. Sixteen scenarios of each phase of the conducted experiment.

Scenario No.	Dataset	Tasks
1	5	VM_1, VM_2, VM_3, …, VM_5
2	10	VM_1, VM_2, VM_3, …, VM_{10}
3	15	VM_1, VM_2, VM_3, …, VM_{15}
4	20	VM_1, VM_2, VM_3, …, VM_{20}
5	25	VM_1, VM_2, VM_3, …, VM_{25}
6	30	VM_1, VM_2, VM_3, …, VM_{30}
7	35	VM_1, VM_2, VM_3, …, VM_{35}
8	40	VM_1, VM_2, VM_3, …, VM_{40}
9	45	VM_1, VM_2, VM_3, …, VM_{45}
10	50	VM_1, VM_2, VM_3, …, VM_{50}
11	100	VM_1, VM_2, VM_3, …, VM_{100}
12	200	VM_1, VM_2, VM_3, …, VM_{200}
13	300	VM_1, VM_2, VM_3, …, VM_{300}
14	400	VM_1, VM_2, VM_3, …, VM_{400}
15	500	VM_1, VM_2, VM_3, …, VM_{500}
16	1000	VM_1, VM_2, VM_3, …, VM_{1000}

Table 2 shows that the number of VMs increases with each scenario, starting from five in the first to one thousand in the sixteenth scenario. The main reason for having multiple scenarios is to thoroughly study the LB mechanism across different circumstances.

IV. EXPERIMENTAL RESULTS AND THEIR IMPLICATIONS

This section includes the experimental results along with their implications, represented into four sub-sections: sub-sections A, B, C, and D includes the experimental results concerning Sipht Thirty, Sipht Sixty, Sipht Hundred, and Sipht Thousand datasets respectively.

A. EXPERIMENTAL RESULTS WITH RESPECT TO SIPHT THIRTY

This sub-section represents the detailed experimental results concerning the Sipht Thirty dataset, where thirty tasks were submitted for processing and computations. Figure 2 represents the graph of LB percentage of the RS algorithms across all the scenarios for the Sipht Thirty task-dataset.

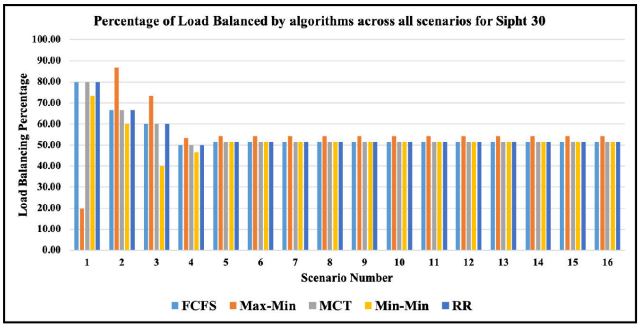

FIGURE 2. Graph of LB by algorithms across all scenarios for Sipht Thirty task-dataset.

Table 3 represents the experimental results with respect to the Deviation Percentage (DP) for Sipht Thirty task-dataset across all the scenarios.

TABLE 3. Deviation Percentage (DP) from expected load for Sipht Thirty.

VMs	FCFS	**Max - Min**	MCT	Min-Min	RR
5	20.00	**80.00**	20.00	26.67	20.00
10	33.33	**13.33**	33.33	40.00	33.33
15	40.00	**26.67**	40.00	60.00	40.00
20	50.00	**46.67**	50.00	53.33	50.00
25	48.57	**45.71**	48.57	48.57	48.57
30	48.57	**45.71**	48.57	48.57	48.57
35	48.57	**45.71**	48.57	48.57	48.57
40	48.57	**45.71**	48.57	48.57	48.57
45	48.57	**45.71**	48.57	48.57	48.57
50	48.57	**45.71**	48.57	48.57	48.57
100	48.57	**45.71**	48.57	48.57	48.57
200	48.57	**45.71**	48.57	48.57	48.57
300	48.57	**45.71**	48.57	48.57	48.57
400	48.57	**45.71**	48.57	48.57	48.57
500	48.57	**45.71**	48.57	48.57	48.57
1000	48.57	**45.71**	48.57	48.57	48.57

From Table 3, we can observe that the average (Avg.) DP for all the algorithms is as follows:

- Avg. DP (FCFS) = 45.39 %
- Avg. DP (Max – Min) = 44.70 %
- Avg. DP (MCT) = 45.39 %
- Avg. DP (Min – Min) = 47.68 %
- Avg. DP (RR) = 45.39 %

Table 4 represents the experimental results with respect to the LB percentage of the RS algorithms for Sipht Thirty task-dataset across all the scenarios.

TABLE 4. Load Balancing (LB) percentage for Sipht Thirty.

VMs	FCFS	**Max - Min**	MCT	Min-Min	RR
5	80.00	**20.00**	80.00	73.33	80.00
10	66.67	**86.67**	66.67	60.00	66.67
15	60.00	**73.33**	60.00	40.00	60.00
20	50.00	**53.33**	50.00	46.67	50.00
25	51.43	**54.29**	51.43	51.43	51.43
30	51.43	**54.29**	51.43	51.43	51.43
35	51.43	**54.29**	51.43	51.43	51.43
40	51.43	**54.29**	51.43	51.43	51.43
45	51.43	**54.29**	51.43	51.43	51.43
50	51.43	**54.29**	51.43	51.43	51.43
100	51.43	**54.29**	51.43	51.43	51.43
200	51.43	**54.29**	51.43	51.43	51.43
300	51.43	**54.29**	51.43	51.43	51.43
400	51.43	**54.29**	51.43	51.43	51.43
500	51.43	**54.29**	51.43	51.43	51.43
1000	51.43	**54.29**	51.43	51.43	51.43

From Table 4, we can observe that the average LB percentage for all the algorithms is as follows:

- Avg. DP (FCFS) = 54.61 %
- Avg. DP (Max – Min) = 55.30 %
- Avg. DP (MCT) = 54.61 %
- Avg. DP (Min – Min) = 52.32 %
- Avg. DP (RR) = 54.61 %

From Table 3 and Table 4, the following implications can be made with respect to Sipht Thirty tasks:

- DP (Max – Min) < [DP (FCFS) ≈ DP (MCT) ≈ DP (RR)] < DP (Min – Min)
- LB (Max – Min) > [LB (FCFS) ≈ LB (MCT) ≈ LB (RR)] > LB (Min – Min)

B. EXPERIMENTAL RESULTS WITH RESPECT TO SIPHT SIXTY

This sub-section represents the detailed experimental results concerning the Sipht Sixty dataset, where sixty tasks were submitted for processing and computations. Figure 3 represents the graph of LB percentage of the RS algorithms across all the scenarios for the Sipht Sixty task-dataset.

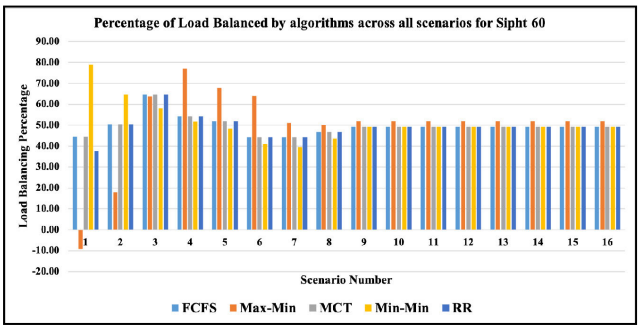

FIGURE 3. Graph of LB by algorithms across all scenarios for Sipht Sixty task-dataset.

Table 5 represents the experimental results with respect to the Deviation Percentage (DP) for Sipht Sixty task-dataset across all the scenarios.

TABLE 5. Deviation Percentage (DP) from expected load for Sipht Sixty.

VMs	FCFS	Max - Min	MCT	**Min-Min**	RR
5	55.59	109.15	55.59	**21.02**	62.37
10	49.49	82.03	49.49	**35.25**	49.49
15	35.25	36.16	35.25	**41.81**	35.25
20	45.76	23.05	45.76	**48.31**	45.76
25	48.14	32.14	48.14	**51.53**	48.14
30	55.71	36.05	55.71	**58.98**	55.71
35	55.79	48.81	55.79	**60.44**	55.79
40	53.14	49.92	53.14	**56.36**	53.14
45	50.77	48.02	50.77	**50.77**	50.77
50	50.77	48.02	50.77	**50.77**	50.77
100	50.77	48.02	50.77	**50.77**	50.77
200	50.77	48.02	50.77	**50.77**	50.77
300	50.77	48.02	50.77	**50.77**	50.77
400	50.77	48.02	50.77	**50.77**	50.77
500	50.77	48.02	50.77	**50.77**	50.77
1000	50.77	48.02	50.77	**50.77**	50.77

From Table 5, we can observe that the Avg. DP for all the algorithms is as follows:

- Avg. DP (FCFS) = 50.31 %
- Avg. DP (Max – Min) = 50.09 %
- Avg. DP (MCT) = 50.31 %
- Avg. DP (Min – Min) = 48.74 %
- Avg. DP (RR) = 50.74 %

TABLE 6. Load Balancing (LB) percentage for Sipht Sixty.

VMs	FCFS	Max - Min	MCT	**Min-Min**	RR
5	44.41	-9.15	44.41	**78.98**	37.63
10	50.51	17.97	50.51	**64.75**	50.51
15	64.75	63.84	64.75	**58.19**	64.75
20	54.24	76.95	54.24	**51.69**	54.24
25	51.86	67.86	51.86	**48.47**	51.86
30	44.29	63.95	44.29	**41.02**	44.29
35	44.21	51.19	44.21	**39.56**	44.21
40	46.86	50.08	46.86	**43.64**	46.86
45	49.23	51.98	49.23	**49.23**	49.23
50	49.23	51.98	49.23	**49.23**	49.23
100	49.23	51.98	49.23	**49.23**	49.23
200	49.23	51.98	49.23	**49.23**	49.23
300	49.23	51.98	49.23	**49.23**	49.23
400	49.23	51.98	49.23	**49.23**	49.23
500	49.23	51.98	49.23	**49.23**	49.23
1000	49.23	51.98	49.23	**49.23**	49.23

Table 6 represents the experimental results with respect to the LB percentage of the RS algorithms for Sipht Sixty task-dataset across all the scenarios.

From Table 6, we can observe that the average LB percentage for all the algorithms is as follows:

- Avg. DP (FCFS) = 49.69 %
- Avg. DP (Max – Min) = 49.91 %
- Avg. DP (MCT) = 49.69 %
- Avg. DP (Min – Min) = 51.26 %
- Avg. DP (RR) = 49.26 %

From Table 5 and Table 6, the following implications can be made with respect to Sipht Sixty tasks:

- DP (Min-Min) < DP (Max-Min) < [DP (FCFS) ≈ DP (MCT)] < DP (RR)
- LB (Min-Min) > LB (Max-Min) > [LB (FCFS) ≈ LB (MCT)] > LB (RR)

C. EXPERIMENTAL RESULTS WITH RESPECT TO SIPHT HUNDRED

This sub-section represents the detailed experimental results concerning the Sipht Hundred dataset, where hundred tasks were submitted for processing and computations. Figure 4 represents the graph of LB percentage of the RS algorithms across all the scenarios for the Sipht Hundred task-dataset.

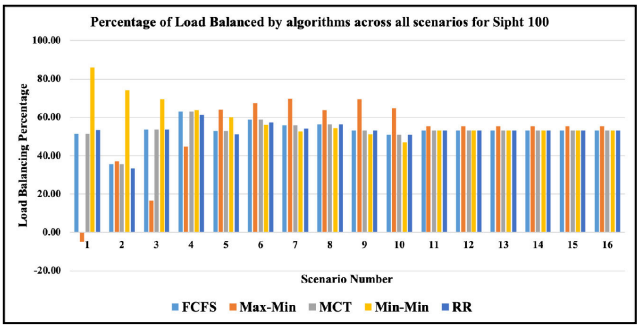

FIGURE 4. Graph of LB by algorithms across all scenarios for Sipht Hundred task-dataset.

TABLE 7. Deviation Percentage (DP) from expected load for Sipht hundred.

VMs	FCFS	Max - Min	MCT	**Min-Min**	RR
5	48.57	104.90	48.57	**13.88**	46.53
10	64.49	62.86	64.49	**25.71**	66.53
15	46.39	83.40	46.39	**30.34**	46.39
20	36.94	55.10	36.94	**36.12**	38.78
25	47.02	35.76	47.02	**39.76**	48.90
30	41.09	32.65	41.09	**43.67**	42.59
35	44.08	30.20	44.08	**47.35**	45.71
40	43.47	36.12	43.47	**45.51**	43.47
45	46.80	30.48	46.80	**48.84**	46.80
50	48.98	35.27	48.98	**52.90**	48.98
100	46.83	44.73	46.83	**46.83**	46.83
200	46.83	44.73	46.83	**46.83**	46.83
300	46.83	44.73	46.83	**46.83**	46.83
400	46.83	44.73	46.83	**46.83**	46.83
500	46.83	44.73	46.83	**46.83**	46.83
1000	46.83	44.73	46.83	**46.83**	46.83

Table 7 represents the experimental results with respect to the Deviation Percentage (DP) for Sipht Hundred task-dataset across all the scenarios.

From Table 7, we can observe that the average DP for all the algorithms is as follows:

- Avg. DP (FCFS) = 46.80 %
- Avg. DP (Max – Min) = 48.45 %
- Avg. DP (MCT) = 46.80 %
- Avg. DP (Min – Min) = 41.57 %
- Avg. DP (RR) = 47.23 %

Table 8 represents the experimental results with respect to the LB percentage of the RS algorithms for Sipht Hundred task-dataset across all the scenarios.

TABLE 8. Load Balancing (LB) percentage for Sipht Hundred.

VMs	FCFS	Max - Min	MCT	**Min-Min**	RR
5	51.43	-4.90	51.43	**86.12**	53.47
10	35.51	37.14	35.51	**74.29**	33.47
15	53.61	16.60	53.61	**69.66**	53.61
20	63.06	44.90	63.06	**63.88**	61.22
25	52.98	64.24	52.98	**60.24**	51.10
30	58.91	67.35	58.91	**56.33**	57.41
35	55.92	69.80	55.92	**52.65**	54.29
40	56.53	63.88	56.53	**54.49**	56.53
45	53.20	69.52	53.20	**51.16**	53.20
50	51.02	64.73	51.02	**47.10**	51.02
100	53.17	55.27	53.17	**53.17**	53.17
200	53.17	55.27	53.17	**53.17**	53.17
300	53.17	55.27	53.17	**53.17**	53.17
400	53.17	55.27	53.17	**53.17**	53.17
500	53.17	55.27	53.17	**53.17**	53.17
1000	53.17	55.27	53.17	**53.17**	53.17

From Table 8, we can observe that the average LB percentage for all the algorithms is as follows:

- Avg. DP (FCFS) = 53.20 %
- Avg. DP (Max – Min) = 51.56 %
- Avg. DP (MCT) = 53.20 %
- Avg. DP (Min – Min) = 58.43 %
- Avg. DP (RR) = 52.77 %

From Table 7 and Table 8, the following implications can be made with respect to Sipht Hundred tasks:

- DP (Min-Min) < [DP (FCFS) ≈ DP (MCT)] < DP (RR) < DP (Max-Min)
- LB (Min-Min) > [LB (FCFS) ≈ LB (MCT)] > LB (RR) > LB (Max-Min)

D. EXPERIMENTAL RESULTS WITH RESPECT TO SIPHT THOUSAND

This sub-section represents the detailed experimental results concerning the Sipht Thousand dataset, where thousand tasks were submitted for processing and computations. Figure 5 represents the graph of LB percentage of the RS algorithms across all the scenarios for the Sipht Thousand task-dataset.

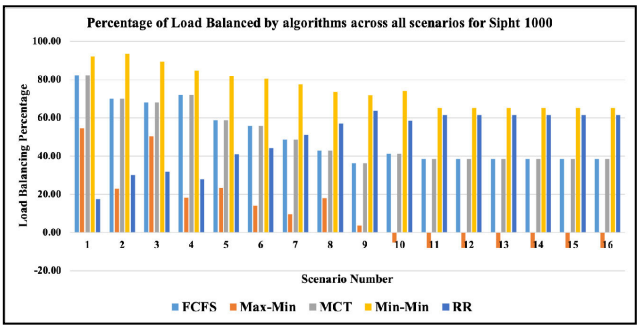

FIGURE 5. Graph of LB by algorithms across all scenarios for Sipht Thousand task-dataset.

Table 9 represents the experimental results with respect to the Deviation Percentage (DP) for Sipht Thousand task-dataset across all the scenarios.

TABLE 9. Deviation Percentage (DP) from expected load for Sipht Thousand.

VMs	FCFS	Max - Min	MCT	**Min-Min**	RR
5	17.63	45.28	17.63	**7.72**	12.71
10	30.03	77.09	30.03	**6.52**	29.41
15	31.99	49.78	31.99	**10.53**	29.14
20	27.95	81.70	27.95	**15.34**	39.80
25	41.16	76.71	41.16	**18.04**	41.93
30	44.25	85.80	44.25	**19.34**	45.92
35	51.13	90.16	51.13	**22.53**	44.18
40	57.21	82.00	57.21	**26.37**	51.68
45	63.75	96.48	63.75	**28.18**	57.87
50	58.70	105.25	58.70	**25.81**	61.18
100	61.57	108.05	61.57	**34.76**	58.98
200	61.57	108.05	61.57	**34.76**	58.98
300	61.57	108.05	61.57	**34.76**	58.98
400	61.57	108.05	61.57	**34.76**	58.98
500	61.57	108.05	61.57	**34.76**	58.98
1000	61.57	108.05	61.57	**34.76**	58.98

From Table 9, we can observe that the average DP for all the algorithms is as follows:

- Avg. DP (FCFS) = 49.58 %
- Avg. DP (Max – Min) = 89.91 %
- Avg. DP (MCT) = 49.58 %
- Avg. DP (Min – Min) = 24.31 %
- Avg. DP (RR) = 47.98 %

TABLE 10. Load Balancing (LB) percentage for Sipht Thousand.

VMs	FCFS	Max - Min	MCT	**Min-Min**	RR
5	82.37	54.72	82.37	**92.28**	17.63
10	69.97	22.91	69.97	**93.48**	30.03
15	68.01	50.22	68.01	**89.47**	31.99
20	72.05	18.30	72.05	**84.66**	27.95
25	58.84	23.29	58.84	**81.96**	41.16
30	55.75	14.20	55.75	**80.66**	44.25
35	48.87	9.84	48.87	**77.47**	51.13
40	42.79	18.00	42.79	**73.63**	57.21
45	36.25	3.52	36.25	**71.82**	63.75
50	41.30	-5.25	41.30	**74.19**	58.70
100	38.43	-8.05	38.43	**65.24**	61.57
200	38.43	-8.05	38.43	**65.24**	61.57
300	38.43	-8.05	38.43	**65.24**	61.57
400	38.43	-8.05	38.43	**65.24**	61.57
500	38.43	-8.05	38.43	**65.24**	61.57
1000	38.43	-8.05	38.43	**65.24**	61.57

Table 10 represents the experimental results with respect to the LB percentage of the RS algorithms for Sipht Thousand task-dataset across all the scenarios.

From Table 10, we can observe that the average LB percentage for all the algorithms is as follows:

- Avg. DP (FCFS) = 50.42 %
- Avg. DP (Max – Min) = 10.09 %
- Avg. DP (MCT) = 50.42 %
- Avg. DP (Min – Min) = 75.69 %
- Avg. DP (RR) = 52.02 %

From Table 9 and Table 10, the following implications can be made with respect to Sipht Thousand tasks:

- DP (Min-Min) < DP (RR) < DP (FCFS) ≈ DP (MCT) < DP (Max-Min)
- LB (Min-Min) > LB (RR) > [LB (FCFS) ≈ LB (MCT)] > LB (Max-Min)

V. CUMULATIVE RESULTS AND ISSUES WITH THE EXISTING SYSTEM

This section represents the cumulative results of the experiment, along with the discussions of issues with the existing system concerning the cloud's LB process. The average deviation percentage (DP) from the expected load and the load balanced by all the algorithms across all sixteen scenarios of every phase is calculated. Table 11 represents the average DP from the expected load for all the RS algorithms across all experiment phases.

TABLE 11. Average Deviation Percentage (DP) of the resource scheduling algorithms for all the phases of experiment.

Sipht Dataset	FCFS	Max - Min	MCT	**Min-Min**	RR
60	45.39	44.7	45.39	**47.68**	45.39
100	50.31	50.09	50.31	**48.74**	50.74
500	46.8	48.45	46.8	**41.57**	47.23
1000	49.58	89.91	49.58	**24.31**	47.98

From Table 11, the cumulative average of DP is calculated for each algorithm across all the four phases, which is represented as follows:

- Cumulative Avg. of DP (FCFS) = 48.02 %
- Cumulative Avg. of DP (Max – Min) = 58.29 %
- Cumulative Avg. of DP (MCT) = 48.02 %
- Cumulative Avg. of DP (Min – Min) = 40.58 %
- Cumulative Avg. of DP (RR) = 47.84 %

Table 12 represents the average LB by the RS algorithms for all phases of the experiment.

TABLE 12. Average load balancing results of the resource scheduling algorithms for all the phases of the experiment.

Sipht Dataset	FCFS	Max - Min	MCT	**Min-Min**	RR
60	45.39	44.7	45.39	**47.68**	45.39
100	50.31	50.09	50.31	**48.74**	50.74
500	46.8	48.45	46.8	**41.57**	47.23
1000	49.58	89.91	49.58	**24.31**	47.98

From Table 12, the cumulative average of LB is calculated for each algorithm across all the four phases, which is represented as follows:

- Cumulative Average of LB (FCFS) = 51.98 %
- Cumulative Average of LB (Max – Min) = 41.71 %
- Cumulative Average of LB (MCT) = 51.98 %
- Cumulative Average of LB (Min – Min) = 59.43 %
- Cumulative Average of LB (RR) = 52.17 %

Therefore, the overall performance comparison with respect to DP and LB is as follows:

- DP (Min – Min) < DP (RR) < [DP (FCFS) ≈ DP (MCT)] < DP (Max – Min)
- LB (Min – Min) > LB (RR) > [LB (FCFS) ≈ LB (MCT)] > LB (Max – Min)

According to the experiment conducted, the Min – Min algorithm is the best LB algorithm, which has the least DP against the expected load and manages to balance 59.43 % of the task load across all four experimental phases. The performance of Min – Min is followed by the performance of the RR algorithm, which is the second-best algorithm, balancing 52.17 % of the task load. The FCFS and MCT algorithms are the third-best, whose behavior is similar and manages to balance 51.98 % of the task load. The Max – Min algorithm manages 41.71 % of the task load and delivers the least load-balanced results. If individual experimental results are considered, no certain RS algorithm provides the ideal LB results across all the experiment phases. With these algorithms in use, cloud performance is hampered due to their lack of ability to adapt to the dynamics of the cloud's environment.

The RS algorithms individually provide certain positives but suffer from several pitfalls. The FCFS RS algorithm is simple and effective to implement in the cloud computing environment for the LB mechanism but lacks preemption and suffers from the convoy effect. These issues of starvation, preemptiveness, and convoy effect from FCFS are solved in Max – Min, MCT, and Min – Min algorithms, which provide more stability for the LB process but have limited applicability due to their overhead of time complexities concerning

sorting tasks and later choosing from them using various strategies, thereby increasing the computational time. RR RS algorithm provides a fair time-sharing environment for tasks, preventing starvation, but the overhead of context switching is always present, causing additional workload to the cloud computing environment. Additionally, all these algorithms lack consideration for task deadlines, tasks' faults, preemption complexities, etc., to name a few. Hence, an intelligent model is required for the cloud computing environment, which can not only solve these issues dynamically by adapting its LB strategies using these RS algorithms, but also to optimize the cloud performance.

VI. REINFORCEMENT LEARNING (RL) APPROACH FOR OPTIMIZING CLOUD RESOURCE UTILIZATION WITH LOAD BALANCING (LB)

This section represents using the Reinforcement Learning (RL) approach for optimizing cloud resource utilization with Load Balancing (LB). The working mechanism of RL is based entirely on feedback and past experiences [2]. With RL, the cloud can improve its decision-making process, which can be used to distribute the incoming network traffic computational task load across multiple VMs to ensure optimal resource utilization [2], [33]. With RL, the cloud computing platform minimizes its response time and enhances performance. This section represents a model where the RL technique is embedded into the cloud computing environment to enhance its LB process, ultimately optimizing it and providing ideal Quality of Service (QoS). The main reason for opting for the RL technique is because the cloud system need not have any past data to start learning [39]. The specialty of using RL is that it can handle dynamic, unknown, erroneous situations and manage them, even in challenging scenarios [39]. This phenomenon can be used to handle the faults caused by the tasks while being computing on the cloud VMs. Additional challenges, such as consideration of task deadlines, preemption of tasks, etc., can be handled by this RL method.

The RL model components for optimizing the utilization of cloud resources for LB can be expressed as follows:

. **Learning Agent 'LA':** The Learning Agent 'LA' is the enhanced LB technique using the RS algorithms. The learning agent 'LA' interacts and acts like a bridge between the user and the cloud's VMs and balances the incoming task load across the available VMs for computations. The LA also, with time, can decide to opt for the best RS algorithm for LB, considering the dynamics of the cloud computing environment at a given instance.

. **Environment 'E':** Environment 'E' constitutes the cloud computing environment where the submitted tasks are computed timely on the available cloud VMs, and services are provided to the end user to gain the ideal Quality of Service (QoS).

. **State Space 'S':** State Space 'S' represents the set of places a specific task needs to undergo from the time of submission until completion.

. **Action Space 'A':** Action Space 'A' represents the set of choices the learning agent 'LA' takes to ensure the ongoing computational task transits from one state to another.

. **Reward 'R':** Reward 'R' is the backbone of this suggested approach, representing the scalar values the environment 'E' provides to the learning agent 'LA' for every action. For every ideal LB decision, the environment offers a higher reward to the learning agent 'LA'; for every incorrect LB, a smaller reward is offered. Next time, for similar actions, the learning agent 'LA' refers to the earlier received reward and tries to make a better decision to maximize the cumulative reward.

. **Q-Table:** Q-Table is a special memory that stores all the rewards the Learning Agent 'LA' earned over time. The Learning Agent 'LA' refers to this Q-Table to improve decision-making and choose the ideal RS algorithm for LB.

In the proposed system, the tasks will be submitted to the cloud for computations and the cloud computing environment 'E' will add these tasks in its queue. The RS algorithms will select the tasks according to their respective policies and forward them to the learning agent 'LA'. LA will keep track of the current loads handled by the cloud VMs and handle all the rewards in the Q-Table. Using these rewards as a feedback for LB, the LA will redirect the future incoming tasks to those VMs who are not overloaded, or will redirect to the under-utilized VMs, thereby improving the LB mechanism. Initially, the LA will get lower rewards, but with time, it will enhance its rewards and improve the overall LB mechanism. Figure 6 represents the flowchart of how RL can be used to optimize the cloud resource utilization with LB.

VII. MATHEMATICAL MODEL

This section represents the mathematical model for the LB algorithms and the RL mechanism which can be used to improve the LB and RS at the cloud's platform.

. The Sipht task dataset, which contains 'N' number of tasks can be expressed as follows:

$$\textbf{Sipht Task Set} = \mathbf{S_1} + \mathbf{S_2} + \mathbf{S_3} + \ldots + \mathbf{S_N} \tag{1}$$

. The VMs set, which contains 'M' number of VMs, used to compute these tasks can be expressed as follows:

$$\textbf{VMs} = \mathbf{VM_1} + \mathbf{VM_2} + \mathbf{VM_3} + \ldots + \mathbf{VM_M} \tag{2}$$

. The optimal expected load to be balanced for a VM which has to compute 'N' tasks can be expressed as follows:

$$\textbf{Average Tasks to Compute (ATtC)} = \mathbf{N/M} \tag{3}$$

. The DP caused by a certain VM 'VMi', where 'i' ranges from '1' to 'M' VMs, against the optimal expected LB can be expressed as:

$$\textbf{DP}[\mathbf{VM_i}] = \textbf{Absolute}\left(\sum[\mathbf{VM_i} = \mathbf{x}] - \textbf{ATtC}\right) \tag{4}$$

. The mathematical equation for RS FCFS algorithm selecting task 'T' can be expressed as follows:

$$\textbf{FCFS}(\mathbf{T}) = \textbf{Minimum}\,[\textbf{Arrival Time}\,(\mathbf{T_i})] \tag{5}$$

Here in equation (5), 'i' ranges from '1' to 'N' tasks.

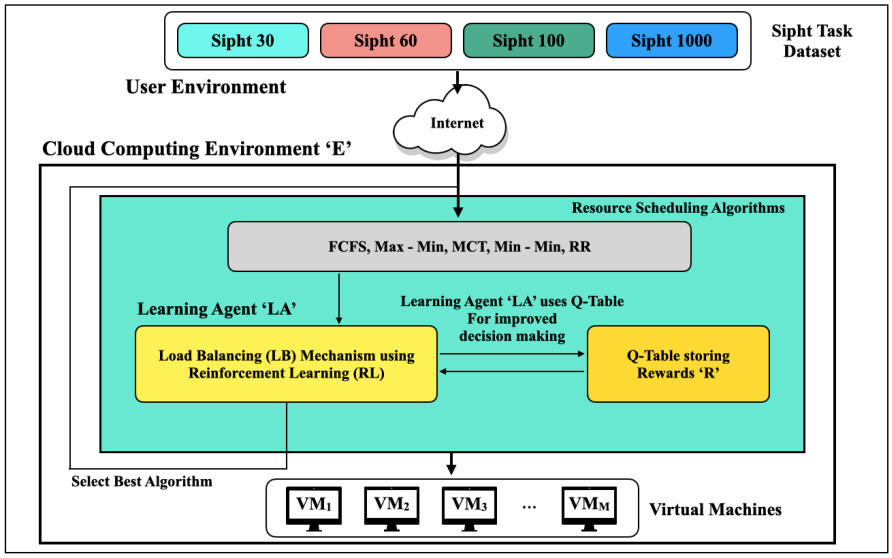

FIGURE 6. LB using RL to optimize the cloud resource utilization.

. The mathematical equation for the RS Max – Min algorithm selecting task 'T' can be expressed as follows:

$$\mathbf{Max-Min(T)} = \mathbf{Maximum}\left[\mathbf{Minimum}\left[\sum \mathbf{Computing\ Time(Ti)}\right]\right] \quad (6)$$

Here in equation (6), 'i' ranges from '1' to 'N' tasks.

. The mathematical equation for the RS MCT algorithm selecting task 'T' can be expressed as follows:

$$\mathbf{MCT(T)} = \mathbf{Minimum}\left[\sum \mathbf{Completion\ Time(Ti)}\right] \quad (7)$$

Here in equation (7), 'i' ranges from '1' to 'N' tasks.

.The mathematical equation for the RS Min – Min algorithm selecting task 'T' can be expressed as follows:

$$\mathbf{Min-Min(T)} = \mathbf{Minimum}\left[\mathbf{Minimum}\left[\sum \mathbf{Computing\ Time(Ti)}\right]\right] \quad (8)$$

Here in equation (8), 'i' ranges from '1' to 'N' tasks.

. The mathematical equation for the RS RR algorithm selecting task 'T' can be expressed as follows:

$$\mathbf{RR(T)} = \mathbf{Time\ Quantum}\,[\mathbf{Computing\ Time(Ti)}] \quad (9)$$

Here in equation (9), 'i' ranges from '1' to 'N' tasks.

. The equation for the RL model which can be used to improve the LB process and RS algorithms can be expressed as follows:

$$\mathbf{RL-LB} = \mathbf{R\ (RS\ Algorithm)} + \gamma^* \sum \mathbf{P(S, A, S')}^* \mathbf{V(S')} \quad (10)$$

Here, **R** represents the reward offered by the RS algorithm; γ represents the discount factor describing significance of future rewards; **P** represents the transition probability from state **S** to **S'** after opting the action **A**; **V(S')** represents the value of the state **S'** to measure how well is the **LA** performing.

VIII. CONCLUSION

This research paper focuses on the experimental study of the cloud's Load Balancing (LB) mechanism using its Resource Scheduling (RS) algorithms. The WorkflowSim cloud simulation platform was used to experiment, and tasks were processed and computed in four phases. Each phase differed in task size, and the dataset utilized was the Sipht task event dataset. RS algorithms First Come First Serve (FCFS), Maximum – Minimum (Max – Min), Minimum Completion Time (MCT), Minimum – Minimum (Min – Min), and Round-Robin (RR) were used to compute these tasks. For a rigorous comparison of LB, each phase had sixteen scenarios where the number of Virtual Machines (VMs) used differed in each scenario, starting with five VMs in the first scenario to one thousand VMs in the last one. From the experimental conducted and results obtained, the LB by FCFS, Max – Min, MCT, Min – Min, and RR were 51.98 %, 41.71 %, 51.98 %, 59.43 %, and 52.17 %, respectively, across all the phases and scenarios. Lastly, the intelligence mechanism of the Reinforcement Learning (RL) method is suggested to make the LB process dynamic. With RL better optimization of cloud resources will occur with enhanced LB process through RS algorithms, thereby providing the cloud ideal Quality of Service (QoS). The presented model can be implemented practically to improve the real-world applications used today such as Amazon Web Services (AWS), Google Cloud Computing (GCP), etc.

ACKNOWLEDGMENT

The authors extend their appreciation to King Saud University for funding this research through Researchers Supporting Project Number (RSPD2023R890), King Saud University, Riyadh, Saudi Arabia.

REFERENCES

[1] T. Cui, R. Yang, C. Fang, and S. Yu, "Deep reinforcement learning-based resource allocation for content distribution in IoT-edge-cloud computing environments," *Symmetry*, vol. 15, no. 1, p. 217, Jan. 2023.

[2] M. Raeisi-Varzaneh, O. Dakkak, A. Habbal, and B.-S. Kim, "Resource scheduling in edge computing: Architecture, taxonomy, open issues and future research directions," *IEEE Access*, vol. 11, pp. 25329–25350, 2023.
[3] M. S. Al Reshan, D. Syed, N. Islam, A. Shaikh, M. Hamdi, M. A. Elmagzoub, G. Muhammad, and K. H. Talpur, "A fast converging and globally optimized approach for load balancing in cloud computing," *IEEE Access*, vol. 11, pp. 11390–11404, 2023.
[4] D. Vivekanandan, S. Wirth, P. Karlbauer, and N. Klarmann, "A reinforcement learning approach for scheduling problems with improved generalization through order swapping," *Mach. Learn. Knowl. Extraction*, vol. 5, no. 2, pp. 418–430, Apr. 2023.
[5] H. Yang, W. Ding, Q. Min, Z. Dai, Q. Jiang, and C. Gu, "A meta reinforcement learning-based task offloading strategy for IoT devices in an edge cloud computing environment," *Appl. Sci.*, vol. 13, no. 9, p. 5412, Apr. 2023.
[6] I. S. Aref, J. Kadum, and A. Kadum, "Optimization of max-min and min-min task scheduling algorithms using G.A in cloud computing," in *Proc. 5th Int. Conf. Eng. Technol. Appl. (IICETA)*, May 2022, pp. 238–242.
[7] Z. Duan, C. Tian, N. Zhang, M. Zhou, B. Yu, X. Wang, J. Guo, and Y. Wu, "A novel load balancing scheme for mobile edge computing," *J. Syst. Softw.*, vol. 186, Apr. 2022, Art. no. 111195.
[8] E. Gures, I. Shayea, M. Ergen, M. H. Azmi, and A. A. El-Saleh, "Machine learning-based load balancing algorithms in future heterogeneous networks: A survey," *IEEE Access*, vol. 10, pp. 37689–37717, 2022.
[9] R. Kaviarasan, P. Harikrishna, and A. Arulmurugan, "Load balancing in cloud environment using enhanced migration and adjustment operator based monarch butterfly optimization," *Adv. Eng. Softw.*, vol. 169, Jul. 2022, Art. no. 103128.
[10] 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.
[11] H. Mahmoud, M. Thabet, M. H. Khafagy, and F. A. Omara, "Multiobjective task scheduling in cloud environment using decision tree algorithm," *IEEE Access*, vol. 10, pp. 36140–36151, 2022.
[12] V. Mohammadian, N. J. Navimipour, M. Hosseinzadeh, and A. Darwesh, "Fault-tolerant load balancing in cloud computing: A systematic literature review," *IEEE Access*, vol. 10, pp. 12714–12731, 2022.
[13] M. Z. Nayyer, I. Raza, S. A. Hussain, M. H. Jamal, Z. Gillani, S. Hur, and I. Ashraf, "LBRO: Load balancing for resource optimization in edge computing," *IEEE Access*, vol. 10, pp. 97439–97449, 2022.
[14] A. Pradhan, S. K. Bisoy, S. Kautish, M. B. Jasser, and A. W. Mohamed, "Intelligent decision-making of load balancing using deep reinforcement learning and parallel PSO in cloud environment," *IEEE Access*, vol. 10, pp. 76939–76952, 2022.
[15] D. Subramoney and C. N. Nyirenda, "Multi-swarm PSO algorithm for static workflow scheduling in cloud-fog environments," *IEEE Access*, vol. 10, pp. 117199–117214, 2022.
[16] Y.-C. Wang and S.-H. Wu, "Efficient deployment of virtual network functions to achieve load balance in cloud networks," in *Proc. 23rd Asia–Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Sep. 2022, pp. 1–6.
[17] F. Alqahtani, M. Amoon, and A. A. Nasr, "Reliable scheduling and load balancing for requests in cloud-fog computing," *Peer Peer Netw. Appl.*, vol. 14, no. 4, pp. 1905–1916, Jul. 2021.
[18] D. Bin, T. Yu, and X. Li, "Research on load balancing dispatching method of power network based on cloud computing," *J. Phys., Conf. Ser.*, vol. 1852, no. 2, Apr. 2021, Art. no. 022047.
[19] S. Chhabra and A. K. Singh, "Dynamic resource allocation method for load balance scheduling over cloud data center networks," *J. Web Eng.*, vol. 20, no. 8, pp. 2269–2284, Nov. 2021.
[20] P.-C. Huang, T.-L. Chin, and T.-Y. Chuang, "Server placement and task allocation for load balancing in edge-computing networks," *IEEE Access*, vol. 9, pp. 138200–138208, 2021.
[21] L.-H. Hung, C.-H. Wu, C.-H. Tsai, and H.-C. Huang, "Migration-based load balance of virtual machine servers in cloud computing by load prediction using genetic-based methods," *IEEE Access*, vol. 9, pp. 49760–49773, 2021.
[22] S. Negi, N. Panwar, M. M. S. Rauthan, and K. S. Vaisla, "Novel hybrid ANN and clustering inspired load balancing algorithm in cloud environment," *Appl. Soft Comput.*, vol. 113, Dec. 2021, Art. no. 107963.
[23] S. Negi, M. M. S. Rauthan, K. S. Vaisla, and N. Panwar, "CMODLB: An efficient load balancing approach in cloud computing environment," *J. Supercomput.*, vol. 77, no. 8, pp. 8787–8839, Aug. 2021.
[24] Y. A. H. Omer, M. A. Mohammedel-Amin, and A. B. A. Mustafa, "Load balance in cloud computing using software defined networking," in *Proc. Int. Conf. Comput., Control, Electr., Electron. Eng. (ICCCEEE)*, Feb. 2021, pp. 1–6.
[25] W. Saber, W. Moussa, A. M. Ghuniem, and R. Rizk, "Hybrid load balance based on genetic algorithm in cloud environment," *Int. J. Electr. Comput. Eng. (IJECE)*, vol. 11, no. 3, p. 2477, Jun. 2021.
[26] D. A. Shafiq, N. Z. Jhanjhi, A. Abdullah, and M. A. Alzain, "A load balancing algorithm for the data centres to optimize cloud computing applications," *IEEE Access*, vol. 9, pp. 41731–41744, 2021.
[27] A. Ghasemi and A. T. Haghighat, "A multi-objective load balancing algorithm for virtual machine placement in cloud data centers based on machine learning," *Computing*, vol. 102, no. 9, pp. 2049–2072, Sep. 2020.
[28] M. Junaid, A. Sohail, A. Ahmed, A. Baz, I. A. Khan, and H. Alhakami, "A hybrid model for load balancing in cloud using file type formatting," *IEEE Access*, vol. 8, pp. 118135–118155, 2020.
[29] 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, p. 496, 2020.
[30] P. Neelima and A. R. M. Reddy, "An efficient load balancing system using adaptive dragonfly algorithm in cloud computing," *Cluster Comput.*, vol. 23, no. 4, pp. 2891–2899, Dec. 2020.
[31] M. S. Sanaj and P. M. J. Prathap, "An enhanced round Robin (ERR) algorithm for effective and efficient task scheduling in cloud environment," in *Proc. Adv. Comput. Commun. Technol. High Perform. Appl. (ACCTHPA)*, Jul. 2020, pp. 107–110.
[32] M. A. Shahid, N. Islam, M. M. Alam, M. M. Su'ud, and S. Musa, "A comprehensive study of load balancing approaches in the cloud computing environment and a novel fault tolerance approach," *IEEE Access*, vol. 8, pp. 130500–130526, 2020.
[33] L. Yu, L. Chen, Z. Cai, H. Shen, Y. Liang, and Y. Pan, "Stochastic load balancing for virtual resource management in datacenters," *IEEE Trans. Cloud Comput.*, vol. 8, no. 2, pp. 459–472, Apr. 2020.
[34] W. Lin, G. Peng, X. Bian, S. Xu, V. Chang, and Y. Li, "Scheduling algorithms for heterogeneous cloud environment: Main resource load balancing algorithm and time balancing algorithm," *J. Grid Comput.*, vol. 17, no. 4, pp. 699–726, Dec. 2019.
[35] H. Nashaat, N. Ashry, and R. Rizk, "Smart elastic scheduling algorithm for virtual machine migration in cloud computing," *J. Supercomput.*, vol. 75, no. 7, pp. 3842–3865, Jul. 2019.
[36] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," in *Proc. IEEE 8th Int. Conf. E-Sci.*, Chicago, IL, USA, Oct. 2012, pp. 1–8.
[37] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.
[38] T. Dillon, C. Wu, and E. Chang, "Cloud computing: Issues and challenges," in *Proc. 24th IEEE Int. Conf. Adv. Inf. Netw. Appl.*, Apr. 2010, pp. 27–33.
[39] D. Vengerov, "A reinforcement learning approach to dynamic resource allocation," *Eng. Appl. Artif. Intell.*, vol. 20, no. 3, pp. 383–390, Apr. 2007.
[40] S. Uwe and Y. Ramin, "Analysis of first-come-first-serve parallel job scheduling," in *Proc. Annu. CM-SIAM Symp. Discrete Algorithms*, 1998, pp. 629–638.

PRATHAMESH VIJAY LAHANDE received the B.Sc. degree in computer science, in 2013, and the M.C.A. degree from MIT, Kothrud, Pune, India, in 2016. He has been a Faculty Member with the Symbiosis Institute of Computer Studies and Research (SICSR), Symbiosis International (Deemed University), Pune, since May 2019. His research interests include data science, data analytics, machine learning, reinforcement learning, and cloud computing. He received the Gold Medal from MIT. He was qualified for the National Eligibility Test (NET), in 2018, the Graduate Aptitude Test in Engineering (GATE), in 2017 and 2018, and certified as a Microsoft Database Expert.

PARAG RAVIKANT KAVERI received the M.Sc. (Hons.) and Ph.D. degrees in computer science from Sant Gadge Baba Amravati University, Amravati, India. He is currently a Microsoft Certified Technology Specialist. He has published more than 55 research papers in various national and international journals and conferences. He is a member of the International Association of Engineers, Hong Kong. He is also a member of the Screening and Monitoring Committee of NIELIT, Government of India. He is an Editorial Board Member of the *Internet of Things and Cloud Computing* (New York, USA) and *American Journal of Engineering and Technology Management*. He received the Young Scientist Award from Vedant Academics.

JATINDERKUMAR R. SAINI received the Ph.D. degree, in 2009. He is currently a Professor and the Director of SICSR, Symbiosis International (Deemed University), Pune, India. Formerly, he was involved with one of only four licensed Certifying Authorities Ministry of Electronics and Information Technology, Government of India. He has H-index of 21 and i-10 index of 52, with more than 100 Scopus/Web of Science (WoS) indexed publications and more than 1500 citations. Under his supervision, 12 candidates have been awarded the Ph.D. degree. He has reviewed more than 1100 articles, with nearly 150 articles of *ACM Transactions* and IEEE TRANSACTIONS journals alone. He has been included in the Top 1% Computer Science Reviewers in World by WoS. He has completed more than 50 certifications from different organizations, including IBM, Google, the Massachusetts Institute of Technology, Stanford University, The University of Texas at Austin, Johns Hopkins University, the University of Cambridge, The Pennsylvania State University, Rice University, Vanderbilt University, and the University of California, just to name a few. He is a fellow of numerous professional and scientific bodies. He is the Joint Secretary and an Executive Committee Member of ISRS, Pune Chapter. He has been an active Executive Committee Member in various capacities with chapters of different professional bodies, such as CSI, IETE, and ISG. He secured Gold Medals and 1st rank at university level in all years of postgraduation, preceded by Silver Medal in the final year of graduation. He was a recipient of the Prestigious DAAD Fellowship in Germany.

KETAN KOTECHA is currently an Administrator and a Teacher of deep learning. He has expertise and experience in cutting-edge research and projects in AI and deep learning for more than the last 25 years. He has published widely with more than 100 publications in several excellent peer-reviewed journals on various topics ranging from cutting-edge AI, education policies, teaching-learning practices, and A.I. for all. He has published three patents and delivered keynote speeches at various national and international forums, including with Machine Intelligence Laboratories, USA; IIT Bombay under the World Bank Project; the International Indian Science Festival organized by the Department of Science and Technology, Government of India, and many more. His research interests include artificial intelligence, computer algorithms, machine learning, and deep learning. He was a recipient of the two SPARC projects in A.I. worth INR 166 lakhs from MHRD, Government of India, in collaboration with Arizona State University, USA, and The University of Queensland, Australia. He was also a recipient of numerous prestigious awards, such as the Erasmus+ Faculty Mobility Grant to Poland, the DUO-India Professors Fellowship for research in responsible A.I. in collaboration with Brunel University London, U.K., the LEAP Grant from the University of Cambridge, U.K., the UKIERI Grant from Aston University, U.K., and the Grant from the Royal Academy of Engineering, U.K., under the Newton-Bhabha Fund. He is an Associate Editor of IEEE ACCESS.

SULTAN ALFARHOOD received the Ph.D. degree in computer science from the University of Arkansas. He is currently an Assistant Professor with the Department of Computer Science, King Saud University (KSU). Since joining KSU, in 2007, he has made several contributions to the field of computer science through his research and publications. His research spans a variety of domains, including machine learning, recommender systems, linked open data, text mining, and ML-based IoT systems. His work includes proposing innovative approaches and techniques to enhance the accuracy and effectiveness of these systems. His recent publications have focused on using deep learning and machine learning techniques to address challenges in these domains. His research continues to make significant contributions to the field of computer science and machine learning. His work has been published in several high-impact journals and conferences.