

Mathematical Model for Improving Cloud Load Balancing Using Scheduling Algorithms



Prathamesh Vijay Lahande and Parag Ravikant Kaveri

1 Introduction

Cloud is a network that provides on-demand services to the end user over the Internet [1, 2]. Among several services, computing is an essential service provided by the cloud, so users can opt to process and execute their tasks on the cloud computing environment rather than on their local machines [1, 3]. By using limited resources, the cloud uses resource scheduling algorithms and executes the tasks on the cloud Virtual Machines (VMs) by trying to maintain a smooth flow of high task execution and maintain its on-demand availability characteristics [4, 5]. However, the cloud faces major issues and challenges with respect to improper load balancing when the requests are high, and the cloud resources are not available [2]. With accurate load balancing, the cloud can provide better cost and time for processing and executing tasks, otherwise will output limited results. Hence, it becomes essential to study the resource scheduling algorithms with respect to load balancing. The resource scheduling algorithms considered for this study are Max–Min (MX–MN), Minimum Completion Time (MCT), and Min–Min (MN–MN). The tasks are executed in a simulation environment and results are compared with respect to the load-balancing mechanism. The reinforcement learning (RL) [5] mechanism is proposed at the end to improve the load balancing and resource scheduling mechanisms. The research paper is organized as follows: Sect. 2 provides the literature review. Section 3 includes the experimental setup. Section 4 includes the mathematical model for load balancing. Section 5 includes the empirical analysis, followed by the conclusion in Sect. 6.

P. V. Lahande · P. R. Kaveri (✉)
Symbiosis Institute of Computer Studies and Research, Symbiosis International (Deemed University), Pune, India
e-mail: parag.kaveri@sicsr.ac.in

P. V. Lahande
e-mail: prathamesh.lahande@sicsr.ac.in

2 Literature Review

The load-balancing problem is an NP-hard problem. Several researchers have designed and presented their work to address and solve this load-balancing issue. The researchers have proposed a hybrid task scheduling algorithm made from the MX–MN, MN–MN, and genetic algorithm to reduce the makespan and enhance the load balances between the resources [6]. The major use of this hybrid algorithm is the genetic algorithm which decides which tasks should use MX–MN and which ones should use MN–MN. This paper proposed a load-balancing virtual network functions deployment scheme to balance the physical machine loads to save the deployment and migration costs [3]. To improve the poor resource scheduling performance, the researchers have proposed a fuzzy iterative algorithm for the cloud computing environment [7]. The experimental results of this algorithm depict that it can improve the load-balancing mechanism and management efficiency of the cloud platform. This paper presents a load-balancing mechanism applied to a software-defining network in a cloud environment [8]. To balance the load in the host machines, a strategy is presented based on machine learning for Virtual Machine replacement [9].

The researchers have provided a load-balancing scheme to the proposed three-layer mobile hybrid hierarchical peer-to-peer model to balance the load with increasing mobile edge computing servers and query loads [10]. A load-balanced service scheduling approach is presented in this paper, which considers load balancing when scheduling requests to resources using classifications such as important, real time, or time tolerant [11]. This approach also considers the rate of failure of resources to provide better reliability to all the requests. A scheduling method named dynamic resource allocation for the purpose of load balancing is proposed to improve the load imbalance, which affects the scheduling efficiency and resource utilization [4]. The researchers have proposed a load-balancing algorithm to dynamically balance the cloud load in the present work [12]. A novel load-balancing task scheduling algorithm is proposed by combining the dragonfly and firefly algorithms [13]. The researchers have proposed scheduling algorithms for the heterogeneous cloud environment to balance the load and time of resources [14].

The researchers have provided a mechanism to ensure that each node in the cloud is appropriately balanced [15]. A hybrid soft computing-inspired technique is introduced to achieve an optimal load of the VMs by tutoring the cloud environment [16]. This technique gives better load-balancing results when compared to other existing algorithms. To minimize the load balancing and overall migration overhead, the researchers have proposed a load-balancing method to provide a probable assurance against resource overloading with VM migration [17]. The researchers have presented two algorithms to distribute the cloud physical resources to obtain load-balancing consolidated systems with minimum power, memory, and time or processing [18]. The researchers have contributed three algorithms to improve the load-balancing mechanism [19]. The researchers have proposed a combination of the swarm intelligence algorithm of an artificial bee colony with the heuristic algorithm to improve the load balancing and makespan and minimize the makespan [20].

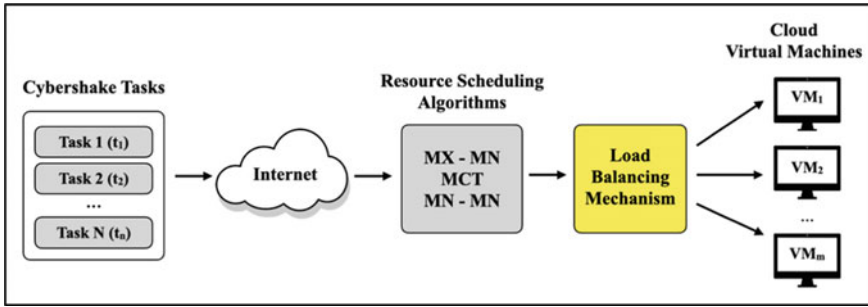


Fig. 1 Architecture of the conducted experiment

3 Experimental Setup

An experiment was conducted in the WorkflowSim environment where the Cyber-shake tasks are processed and executed on the cloud VMs in four scenarios. The Cybershake 30, 50, 100, and 1000 tasks are processed and executed in each scenario on the cloud VMs in the series: 5, 10, 15, ..., and 50. All these tasks are submitted to the cloud for processing and execution. The cloud accepts these tasks and places them in the ready queue, which contains a list of all the tasks waiting to be assigned a VM. The respective resource scheduling algorithm selects the tasks from this ready queue, performs the load-balancing mechanism, and appropriately schedules it to the target VM. Figure 1 depicts the architecture of the conducted experiment.

4 Mathematical Model for Load Balancing

This section includes the mathematical model for load balancing using the resource scheduling algorithms MX–MN, MCT, and MN–MN.

The task set of the Cybershake dataset used for the experiment is as follows:

$$\text{Task set} = T_1 + T_2 + T_3 + \dots + T_n.$$

The Virtual Machine (VM) set can be depicted as follows:

$$\text{VM set} = \text{VM}_1 + \text{VM}_2 + \text{VM}_3 + \dots + \text{VM}_m.$$

Every VM processes and executes tasks, and the count of tasks that the VM has processed and executed is as follows:

$$\text{VM}(x) = \sum_{i=0}^m [\text{VM}_i = x].$$

The ideal expected load for each VM ‘ m ’ processing ‘ n ’ number of tasks in a certain scenario is as follows:

$$\text{Average}(A) = \frac{n}{m}.$$

To measure load balance of a VM ‘ VM_i ’, the amount of deviation caused by the VM ‘ VM_i ’ from the ideal expected average load ‘ A ’ can be calculated as

$$\text{Deviation}(VM_i) = \text{Absolute} \left(\sum_{i=0}^m [VM_i = x] - A \right).$$

Load balancing using MX–MN scheduling algorithm: The MX–MN algorithm first finds the minimum execution time (ET) from all the tasks. Later, it chooses the task ‘ T ’ with maximum ET from them, depicted as follows:

$$\text{MX--MN}(T) = \text{Max} \left[\text{Min} \left[\sum_{i=0}^n \text{E.T.}(T_i) \right] \right].$$

Load balancing using MCT scheduling algorithm: The MCT algorithm schedules the task ‘ T ’ based on the expected Minimum Completion Time (CT) among all the tasks, which can be depicted as follows:

$$\text{MCT}(T) = \text{Min} \left[\sum_{i=0}^n \text{C.T.}(T_i) \right].$$

Load balancing using MN–MN scheduling algorithm: The MN–MN algorithm first finds the minimum execution time (ET) from all the tasks. Later, it chooses the task ‘ T ’ with minimum ET from them. The below equation depicts the same:

$$\text{MN--MN}(T) = \text{Min} \left[\text{Min} \left[\sum_{i=0}^n \text{E.T.}(T_i) \right] \right].$$

5 Empirical Analysis of the Results and Their Implications

This section includes the detailed empirical analysis of the results of the experiment conducted with respect to the load-balancing mechanism along with their implications, which is further divided into four sub-sections: Sections 5.1, 5.2, 5.3, and 5.4

include the empirical analysis with respect to load balancing for Cybershake 30, 50, 100, and 1000 tasks, respectively.

5.1 Scenario 1: Empirical Analysis with Respect to Cybershake 30 Tasks

Table 1 depicts the deviation percentage of algorithms for Cybershake 30 tasks.

Figure 2 depicts the deviation graph of algorithms for Cybershake 30.

Table 1 Deviation percentage of load balancing for Cybershake 30

VMs	MX–MN	MCT	MN–MN
5	56.7742	42.5806	61.9355
10	37.276	56.6308	91.0394
15	37.276	56.6308	91.0394
20	37.276	56.6308	91.0394
25	37.276	56.6308	91.0394
30	37.276	56.6308	91.0394
35	37.276	56.6308	91.0394
40	37.276	56.6308	91.0394
45	37.276	56.6308	91.0394
50	37.276	56.6308	91.0394

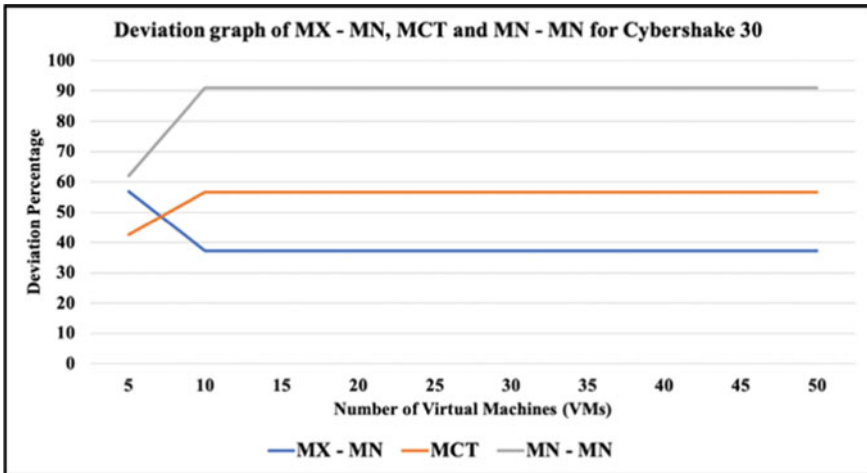


Fig. 2 Deviation graph for resource scheduling algorithms for Cybershake 30

Table 2 Empirical analysis of load balancing with respect to Cybershake 30 tasks

Parameter	MX–MN	MCT	MN–MN
Linear regression equation	$y = -1.60x + 45.0$	$y = 0.76x + 51.01$	$y = 1.58x + 79.39$
Regression line slope	- 1.6035	0.7664	1.5875
Y-intercept	45.07	51.011	79.398
Relationship	Negative	Positive	Positive
R^2 value	0.2727	0.2727	0.2727
VM–LB analysis	\uparrow VM = \uparrow LB	\uparrow VM = \downarrow LB	\uparrow VM = \downarrow LB
Performance	MX–MN \approx MCT \approx MN–MN		

Table 2 depicts the empirical analysis of load balancing for the resource scheduling algorithms with respect to Cybershake 30 tasks.

From Fig. 2, Tables 1 and 2, following points can be observed for Cybershake 30 tasks:

- \uparrow VM = \uparrow LB: As VMs increase, load balancing improves for MX–MN.
- \uparrow VM = \downarrow LB: As VMs increase, load balancing degrades for MCT and MN–MN.
- Performance (MX–MN) \approx Performance (MCT) \approx Performance (MN–MN).

5.2 Empirical Analysis with Respect to Cybershake 50 Tasks

Table 3 depicts the deviation percentage of algorithms for Cybershake 50 tasks.

Figure 3 depicts the deviation graph of algorithms for Cybershake 50.

Table 4 depicts the empirical analysis of load balancing for the resource scheduling algorithms with respect to Cybershake 50 tasks.

From Fig. 3, Tables 3 and 4, following points can be observed for Cybershake 50 tasks:

Table 3 Deviation percentage of load balancing for Cybershake 50

VMs	MX–MN	MCT	MN–MN
5	29.8039	49.4118	33.7255
10	58.0392	69.8039	61.1765
15	61.1765	80.0000	81.5686
20	66.1765	87.2549	87.9902
25	66.1765	87.2549	87.9902
30	66.1765	87.2549	87.9902
35	66.1765	87.2549	87.9902
40	66.1765	87.2549	87.9902
45	66.1765	87.2549	87.9902
50	66.1765	87.2549	87.9902

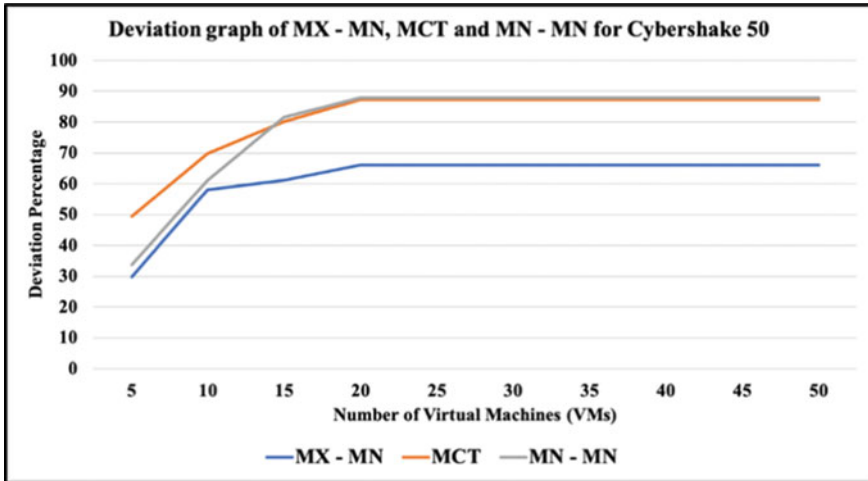


Fig. 3 Deviation graph for resource scheduling algorithms for Cybershake 50

Table 4 Empirical analysis of load balancing with respect to Cybershake 50 tasks

Parameter	MX-MN	MCT	MN-MN
Linear regression equation	$y = 2.48x + 47.58$	$y = 3.02x + 64.36$	$y = 4.29x + 55.63$
Regression line slope	2.4807	3.0244	4.292
Y-intercept	47.582	64.366	55.634
Relationship	Positive	Positive	Positive
R^2 value	0.4343	0.5398	0.5171
VM-LB analysis	↑ VM = ↓ LB	↑ VM = ↓ LB	↑ VM = ↓ LB
Performance	MX-MN > MN-MN > MCT		

- ↑ VM = ↓ LB: As VMs increases, load balancing degrades for all algorithms.
- Performance (MX-MN) > Performance (MN-MN) > Performance (MCT).

5.3 Empirical Analysis with Respect to Cybershake 100 Tasks

Table 5 depicts the deviation percentage of algorithms for Cybershake 100 tasks.

Figure 4 depicts the deviation graph of algorithms for Cybershake 100.

Table 6 depicts the empirical analysis of load balancing for the resource scheduling algorithms with respect to Cybershake 100 tasks.

From Fig. 4, Tables 5 and 6, following points can be observed for Cybershake 100 tasks:

- ↑ VM = ↓ LB: As VMs increase, load balancing degrades for all algorithms.

Table 5 Deviation percentage of load balancing for Cybershake 100

VMs	MX-MN	MCT	MN-MN
5	18.2178	41.1881	11.0891
10	40.5941	38.8119	39.0099
15	20.5941	50.165	48.1848
20	44.9505	54.7525	58.9109
25	56.7129	72.7129	76.7525
30	64.7808	77.5106	81.8953
35	64.7808	77.5106	81.8953
40	64.7808	77.5106	81.8953
45	64.7808	77.5106	81.8953
50	64.7808	77.5106	81.8953

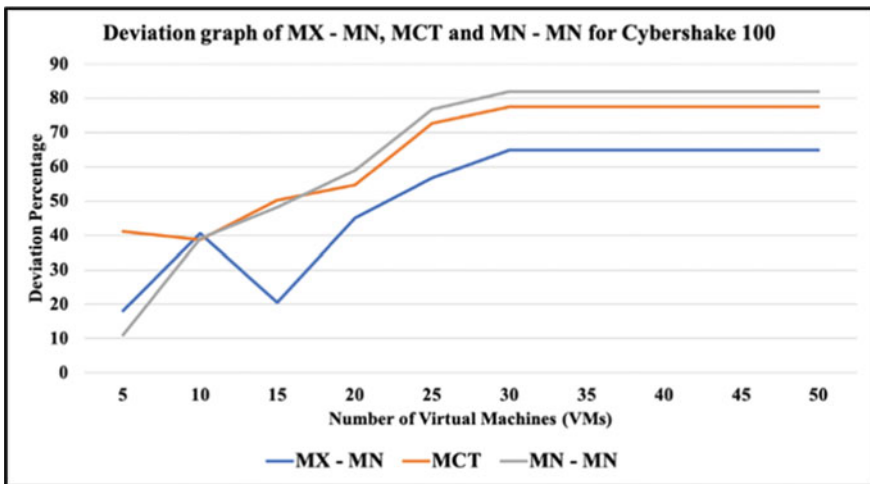


Fig. 4 Deviation graph for resource scheduling algorithms for Cybershake 100

Table 6 Empirical analysis of load balancing with respect to Cybershake 100 tasks

Parameter	MX-MN	MCT	MN-MN
Linear regression equation	$y = 5.31x + 21.26$	$y = 4.89x + 37.59$	$y = 7.15x + 25.00$
Regression line slope	5.3143	4.8945	7.1521
Y-intercept	21.268	37.599	25.006
Relationship	Positive	Positive	Positive
R^2 value	0.7459	0.8175	0.7725
VM-LB analysis	↑ VM = ↓ LB	↑ VM = ↓ LB	↑ VM = ↓ LB
Performance	MX-MN > MN-MN > MCT		

Table 7 Deviation percentage of load balancing for Cybershake 1000

VMs	MX–MN	MCT	MN–MN
5	2.5175	3.3567	1.9581
10	2.9571	11.6684	7.9321
15	7.1795	21.2055	8.9644
20	3.5665	20.2698	10.9091
25	4.9791	22.6654	13.8982
30	19.374	28.2918	13.4333
35	21.3387	38.2818	17.8622
40	34.6554	37.2828	15.3247
45	6.9842	41.7317	17.0541
50	4.5795	43.2808	20.4956

- Performance (MX–MN) > Performance (MN–MN) > Performance (MCT).

5.4 Empirical Analysis with Respect to Cybershake 1000 Tasks

Table 7 depicts the deviation percentage of algorithms for Cybershake 1000 tasks.

Figure 5 depicts the deviation graph of algorithms for Cybershake 1000.

Table 8 depicts the empirical analysis of load balancing for the resource scheduling algorithms with respect to Cybershake 100 tasks.

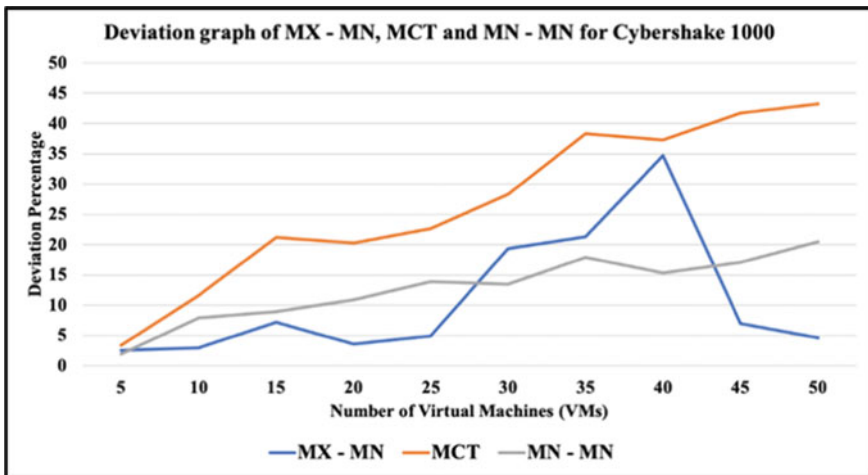


Fig. 5 Deviation graph for resource scheduling algorithms for Cybershake 1000

Table 8 Empirical analysis of load balancing with respect to Cybershake 1000 tasks

Parameter	MX–MN	MCT	MN–MN
Linear regression equation	$y = 1.52x + 2.41$	$y = 4.30x + 3.14$	$y = 1.71x + 3.35$
Regression line slope	1.5263	4.3019	1.7145
Y-intercept	2.4185	3.1431	3.3536
Relationship	Positive	Positive	Positive
R^2 value	0.1855	0.9494	0.8966
VM–LB analysis	\uparrow VM = \downarrow LB	\uparrow VM = \downarrow LB	\uparrow VM = \downarrow LB
Performance	MX–MN > MN–MN > MCT		

From Fig. 5, Tables 7 and 8, following points can be observed for Cybershake 1000 tasks:

- \uparrow VM = \downarrow LB: As VMs increase, load balancing degrades for all algorithms.
- Performance (MX–MN) > Performance (MN–MN) > Performance (MCT).

6 Conclusion

Load balancing plays a critical role in improving cloud performance. With an enhanced load-balancing technique, the performance of the cloud will be enhanced; otherwise, the cloud faces increased downtime with improper load balancing. Therefore, it becomes vital to study the resource scheduling algorithms with respect to the load-balancing mechanism. In this research paper, the Cybershake 30, 50, 100, and 1000 tasks were processed and executed in the WorkflowSim environment utilizing the resource scheduling algorithms MX–MN, MCT, and MN–MN under four different scenarios. Based on the experiment, results, and empirical analysis, it can be concluded that the load balancing degrades as the number of tasks increases in each scenario. The performance of all these algorithms is similar when the task size is comparatively smaller. With a gradual increase in the tasks in every scenario, the MX–MN algorithm gives the best performance, followed by performances of MN–MN and MCT algorithms, respectively. To improve the load-balancing mechanism, the cloud system should be provided with an intelligence mechanism. The machine learning (ML) technique of reinforcement learning (RL) is well known for enhancing the performance of any system when applied to it. RL's mechanism is similar to how humans learn, i.e., using feedback, trial and error, and past experiences. The significant advantage of applying RL to the cloud is that no past data is required for the cloud to learn. The cloud will first be in a learning phase with RL. Over time, the cloud system will understand and adapt load balancing, improve its resource scheduling, and ultimately improve the overall cloud performance.

References

1. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M (2010) A view of cloud computing. *Commun ACM* 53(4):50–58
2. Dillon T, Wu C, Chang E (2010) Cloud computing: issues and challenges. In: 2010 24th IEEE international conference on advanced information networking and applications
3. Wang YC, Wu SH (2022) Efficient deployment of virtual network functions to achieve load balance in cloud networks. In: 2022 23rd Asia-Pacific network operations and management symposium (APNOMS)
4. Chhabra S, Singh AK (2021) Dynamic resource allocation method for load balance scheduling over cloud data center networks. *J Web Eng*
5. Vengerov D (2007) A reinforcement learning approach to dynamic resource allocation. *Eng Appl Artif Intell* 20(3):383–390
6. Aref IS, Kadum J, Kadum A (2022) Optimization of max-min and min-min task scheduling algorithms using G.A in cloud computing. In: 2022 5th International conference on engineering technology and its applications (ICETA)
7. Bin D, Yu T, Li X (2021) Research on load balancing dispatching method of power network based on cloud computing. *J Phys Conf Ser* 1852(2):022047
8. Omer YAH, Mohammedel-Amin MA, Mustafa ABA (2021) Load balance in cloud computing using software defined networking. In: 2020 International conference on computer, control, electrical, and electronics engineering (ICCCEEE)
9. Ghasemi A, Toroghi Haghghat A (2020) A multi-objective load balancing algorithm for virtual machine placement in cloud data centers based on machine learning. *Computing* 102(9):2049–2072
10. Duan Z, Tian C, Zhang N, Zhou M, Yu B, Wang X, Guo J, Wu Y (2022) A novel load balancing scheme for mobile edge computing. *J Syst Softw* 186:111195
11. Alqahtani F, Amoon M, Nasr AA (2021) Reliable scheduling and load balancing for requests in cloud-fog computing. *Peer Peer Netw Appl* 14(4):1905–1916
12. Negi S, Rauthan MMS, Vaisla KS, Panwar N (2021) CMODLB: an efficient load balancing approach in cloud computing environment. *J Supercomput* 77(8):8787–8839
13. Neelima P, Reddy ARM (2020) An efficient load balancing system using adaptive dragonfly algorithm in cloud computing. *Clust Comput* 23(4):2891–2899
14. Lin W, Peng G, Bian X, Xu S, Chang V, Li Y (2019) Scheduling algorithms for heterogeneous cloud environment: main resource load balancing algorithm and time balancing algorithm. *J Grid Comput* 17(4):699–726
15. Kaviarasan R, Harikrishna P, Arulmurugan A (2022) Load balancing in cloud environment using enhanced migration and adjustment operator-based monarch butterfly optimization. *Adv Eng Softw* 169:103128
16. Negi S, Panwar N, Rauthan MMS, Vaisla KS (2021) Novel hybrid ANN and clustering inspired load balancing algorithm in cloud environment. *Appl Soft Comput* 113:107963
17. Yu L, Chen L, Cai Z, Shen H, Liang Y, Pan Y (2020) Stochastic load balancing for virtual resource management in datacenters. *IEEE Trans Cloud Comput* 8(2):459–472
18. Nashaat H, Ashry N, Rizk R (2019) Smart elastic scheduling algorithm for virtual machine migration in cloud computing. *J Supercomput* 75(7):3842–3865
19. Saber W, Moussa W, Ghuniem AM, Rizk R (2021) Hybrid load balance based on genetic algorithm in cloud environment. *Int J Electr Comput Eng (IJECE)* 11(3):2477
20. Kruekaew B, Kimpan W (2020) Enhancing of artificial bee colony algorithm for virtual machine scheduling and load balancing problem in cloud computing. *Int J Comput Intell Syst* 13(1):496