International Journal of Applied Mathematics
Volume 38 No. 9s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

A METHOD BASED ON MODIFIED SIMULATED ANNEALING AND OPTIMAL VECTOR FOR LOAD BALANCING IN FOG COMPUTING

Ali Malik Hammood[1], Ehsan Shoja[2,*], Parviz Rashidi khazaee[3]

[1, 2] Department of electrical and computer engineering, Urmia University, Urmia, Iran.

[3] Department of Information Technology and Computer Engineering, Urmia University of Technology, Urmia, Iran

a.malikhammood@urmia.ac.ir, e.shoja@urmia.ac.ir, p.rashidi@uut.ac.ir

*Corresponding author: Ehsan Shoja, e.shoja@urmia.ac.ir

Abstract

Fog computing is a distributed and hierarchical framework designed to deliver services such as computing, storage, and network resources. It reduces latency and communication frequency between users and edge nodes, offering a solution that minimizes delays and traffic congestion, while also lowering power consumption and bandwidth usage. This helps alleviate the burden on cloud data centers. While cloud computing remains a practical solution for sustainable development, particularly in the IoT sector, it still faces several unresolved challenges. One of the key issues in fog computing is load balancing, which plays a crucial role in system management. A major challenge is selecting the most suitable source for each task. Load balancing in cloud computing involves distributing workloads across multiple nodes to ensure high resource utilization, equitable resource allocation, and user satisfaction. In this work, simulated annealing and optimal vectors are used to achieve load balancing in fog computing. The simulated annealing algorithm helps identify the best values for the request allocation vector, ensuring tasks are efficiently assigned to servers. The method was implemented in MATLAB, and the results demonstrate notable improvements in average response time, total execution time, and load balancing performance.

Keywords: *Fog computing, load balancing, simulated annealing algorithm, optimal vector, cloud computing, IoT.*

1- Introduction

Load balancing in fog computing is a critical challenge due to the dynamic, decentralized, and resource-constrained nature of the environment. Efficient load distribution is essential for ensuring that computational resources are optimally utilized while minimizing latency and maximizing service reliability. Various approaches, including heuristic algorithms and optimization techniques, have been proposed to address this challenge. In [1], the authors discuss the foundational aspects of fog computing, including its role in extending cloud capabilities to edge devices. The focus is on how fog computing provides real-time data processing and resource management to support IoT applications, where load balancing becomes a vital aspect for ensuring efficient resource utilization. The study in [2] introduces a

Received: August 12, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

dynamic resource allocation technique for load balancing in fog environments, proposing an algorithm that adapts to changing system conditions. The method adjusts resource allocation based on real-time network traffic and load variations, helping to optimize performance across fog nodes. In [3], the authors explore strategies for minimizing delays in IoT systems through fog computing. This study highlights the importance of reducing computational load at the edge to prevent bottlenecks and improve task execution times, thus emphasizing the relevance of effective load balancing techniques. The work in [4] focuses on risk estimation techniques used in fog computing environments, which indirectly impact load balancing. By understanding risk and resource availability, load distribution decisions can be made more strategically to ensure that no node is overburdened, thus maintaining system stability. [5] presents an adaptive risk-based access control model in IoT systems, which can influence load balancing strategies by controlling resource access in a way that aligns with system security and performance needs. In [6], the authors offer a detailed review of cloud computing architectures, which are closely related to fog computing. Their findings on resource scheduling and cloud-based optimization techniques provide a broader context for understanding the challenges and solutions applicable to fog environments. The paper in [7] surveys various cloud computing models, touching on resource management and load balancing as central issues. Although focused on cloud environments, it lays the groundwork for understanding how similar approaches can be adapted to fog computing. In [8], the concept of fog computing is extended as a platform for IoT and analytics. The authors emphasize the need for efficient data processing and management techniques, which include load balancing algorithms that ensure optimal resource allocation across fog nodes. [9] discusses the democratization of access to IoT infrastructure and presents a model for managing fog resources in a way that optimizes load balancing and system scalability. This study is relevant for understanding how to efficiently allocate resources in a large-scale fog network. The work in [10] investigates security and trust issues in fog computing. Since security concerns often intersect with load balancing, the authors propose strategies that consider both resource distribution and secure management of fog nodes, ensuring that the system remains robust against attacks and failures. In [11], the authors provide an early exploration of fog computing, focusing on its role in enhancing IoT applications. The load balancing techniques discussed here are foundational, offering insights into how fog nodes can be managed to ensure efficient task allocation across a distributed network. [12] offers a comprehensive framework for fog computing that integrates various challenges, including load balancing. This framework provides a high-level perspective on how multiple factors such as node heterogeneity, task priority, and resource constraints must be considered in load balancing algorithms. The study in [13] assesses the suitability of fog computing for IoT environments, emphasizing the importance of minimizing latency through load balancing techniques. By distributing workloads effectively, fog computing can offer significant performance improvements over traditional cloud-based systems. In [14], the authors discuss security and privacy challenges in fog computing, which can affect load balancing decisions. Ensuring that sensitive data is handled securely during load balancing operations is an important consideration for maintaining both performance and privacy in fog environments. [15] reviews the middleware paradigm for fog computing, highlighting its role in bridging the gap between IoT devices and

Received: August 12, 2025

International Journal of Applied Mathematics
Volume 38 No. 9s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

cloud infrastructures. Effective load balancing is necessary to manage the varying computational demands of connected devices in real time. [16] provides a review of fog computing and IoT, discussing the various optimization techniques, including load balancing, to improve system efficiency. The study outlines how adaptive algorithms can be used to optimize resource allocation in real-time fog computing environments. In [17], the authors address service allocation in fog-cloud environments, providing insights into how fog nodes can dynamically adjust their workload distribution to balance tasks between fog and cloud layers. This hybrid approach to load balancing enhances overall system performance and scalability. [18] discusses the need for coordinated management between fog and cloud computing systems, focusing on the challenges of balancing loads across both layers. The authors suggest that load balancing techniques must account for both edge and cloud resources to avoid congestion and inefficiency. The paper in [19] explores power consumption and delay tradeoffs in cloud-fog computing systems. It proposes a workload allocation strategy that considers both performance and energy efficiency, an important consideration in load balancing decisions. In [20], a reinforcement learning-based load balancing algorithm for fog networks is proposed. The approach learns optimal task distribution strategies through interaction with the environment, gradually improving performance over time. The study in [21] introduces a hill climbing algorithm for load balancing in fog environments, which iteratively adjusts the distribution of tasks to minimize load imbalances. This method aims to optimize both system throughput and node utilization. [22] presents a cuckoo optimization algorithm for job scheduling in a cloud-fog system, highlighting how intelligent search techniques can be applied to improve load balancing in smart grid applications. In [23], the authors propose a round-robin-inspired load balancing technique that takes historical data into account to adjust task distribution strategies. This technique is particularly useful for ensuring fair and efficient workload distribution across fog nodes. [24] focuses on a fog-cloud platform for resource utilization, employing load balancing to improve efficiency. The study demonstrates the effectiveness of balancing workloads between fog and cloud resources in a hybrid model. [25] introduces an effective load balancing strategy (ELBS) for real-time fog environments, using fuzzy and probabilistic neural networks. This hybrid approach is designed to adapt dynamically to the changing demands of fog computing applications. The paper in [26] addresses the use of cuckoo search optimization for load balancing in cloud-based smart grids. The method optimizes task allocation while minimizing energy consumption and network congestion. [27] discusses energy-aware load balancing in smart factories, proposing a fog computing model that balances loads while considering energy efficiency. This approach ensures that tasks are distributed in a way that reduces overall power consumption. In [28], the authors present a heuristic virtual machine scheduling method for fog-cloud environments. This approach focuses on efficiently scheduling virtual machines to balance the computational load between fog and cloud resources. The study in [29] uses a hybrid genetic algorithm to optimize load balancing in cloud-fog platforms, demonstrating how evolutionary algorithms can improve task allocation and system performance in distributed computing environments. [30] explores sequential randomization methods for load balancing in fog computing, proposing an approach that uses randomization to distribute tasks across nodes while avoiding load imbalances. In [31], the authors discuss secure and sustainable load balancing in edge data

International Journal of Applied Mathematics
Volume 38 No. 9s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

centers. The paper presents techniques for ensuring that load balancing decisions are made while maintaining the security and sustainability of fog networks. The work in [32] introduces an efficient load balancing algorithm for task preprocessing in fog computing. This algorithm optimizes the distribution of tasks across fog nodes, improving both processing speed and system efficiency. [33] presents an optimized resource scheduling algorithm based on genetic algorithms (GA) and ant colony optimization (ACO). The combined approach is aimed at improving load balancing and task scheduling in fog computing systems. [34] discusses a hybrid particle swarm optimization and firefly algorithm for optimal fog node selection in dynamic fog computing services. This hybrid approach helps improve the load balancing process by selecting the most suitable fog nodes based on multiple criteria. In [35], the authors propose a fog-cluster based load-balancing technique that dynamically adjusts load distribution based on real-time system conditions. This approach improves task execution efficiency by optimizing fog resource utilization. [36] presents a hybrid evolutionary algorithm to enhance task scheduling and load balancing in fog computing. The proposed method integrates various optimization strategies to improve system performance and reduce task execution time. Finally, [37] investigates a hybrid algorithm combining particle swarm optimization and simulated annealing for resource allocation in fog-cloud environments. This method aims to optimize load balancing by leveraging the strengths of both algorithms, providing better performance in dynamic fog computing environments.

2- Problem statement

Fog computing, a term coined by Cisco, refers to the use of shared edge or end-user devices close to the users to handle functions such as storage, communication, control, configuration, measurement, and management. This approach helps overcome challenges related to delays and bandwidth limitations. Fog computing provides a variety of services at the network edge, supporting a broad range of IoT applications. It introduces the idea of bringing cloud capabilities closer to the end devices to improve service quality [1]. Essentially, fog computing acts as an intermediary layer between end-user devices and cloud computing. It extends cloud capabilities by utilizing resources located closer to the network edge. Fog computing is particularly beneficial for real-time applications, including healthcare, industrial systems, and smart traffic signage. However, as an emerging computing model, it still requires further standardization, especially regarding load balancing. Load balancing refers to the process of redistributing workload across system nodes to optimize resource usage, enhance task response time, and prevent uneven load distribution. With growing user demand, load balancing at the edge becomes increasingly critical. Effective load balancing is crucial for resource optimization, preventing bottlenecks, and avoiding both overloading and underutilization of resources. It is a key technique used in cloud and fog computing to distribute tasks across resources, improving network performance, optimizing resource utilization, and boosting throughput while reducing response time. Load balancing algorithms can be classified into two types: static and dynamic. Static algorithms work well in stable, homogeneous environments, providing efficient results, but lack flexibility to adapt to system changes during runtime. On the other hand, dynamic algorithms are more adaptable, considering system attributes before and during execution. While various load balancing strategies have been proposed for fog

Received: August 12, 2025

International Journal of Applied Mathematics
Volume 38 No. 9s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

computing [2], they often face limitations, such as low-priority tasks waiting for extended periods, system performance degradation due to continuous task migration, and high network latency. To address these issues, effective load balancing in fog environments can be achieved by selecting suitable algorithms like metaheuristics. This study utilizes simulated annealing and an optimized vector approach to improve load balancing in fog computing, aiming to reduce average completion time and ensure a more balanced load distribution across the system.

3- Proposed method

3.1 Introduction

Load balancing in cloud computing presents a significant challenge, often requiring a distributed solution. It is not always practical or cost-efficient to keep a large number of servers idle to handle potential requests, meaning tasks cannot always be directed to the most suitable server. Effective load balancing in a cloud environment is complex, given the widespread distribution of components across various locations. The process of load balancing involves redistributing workloads across the system's individual nodes to optimize resource usage, enhance response times, and prevent situations where some nodes are overloaded while others remain underutilized. Load balancing algorithms are generally classified into two types: static and dynamic. Static algorithms work best in stable, homogeneous environments, yielding optimal results but lacking the flexibility to adjust to dynamic system changes during execution. On the other hand, dynamic algorithms are more adaptable, taking into account various system characteristics before and during runtime. Load balancing aims to enhance system performance by efficiently redistributing the workload among processors. In the context of cloud systems, several virtual machines with varying characteristics and resources are defined. The problem of evenly distributing the workload across the cloud infrastructure is framed as a search problem and optimized through techniques such as the simulated annealing algorithm. This optimization process improves load balancing in the cloud. Additionally, cloud computing not only manages the assigned tasks but also performs its own internal operations, such as processing datasets like those from NASA.**3-2. Simulated Annealing Algorithm**

The Simulated Annealing (SA) algorithm is a neighborhood search technique commonly applied to optimization problems. Its decision-making process involves randomly generating and evaluating a new neighborhood for each move. The goal of the algorithm is to find a solution that is close to or globally minimal. The algorithm begins with an initial solution, and through a series of iterations, neighboring solutions are generated while maintaining a constant temperature (T = T0). In each iteration, a new solution is derived from the current one, and its objective function value is compared to the current solution. If the new solution yields a better result, it replaces the current one. If the solution is not suitable, it is replaced with the current solution with a probability determined by the Boltzmann function ($exp(\frac{-\Delta}{KT})$). This process ensures that the algorithm does not get trapped in local optima.

In the equation above, the parameter Δ denotes the difference between the objective function values of the current and the new solution. The Boltzmann coefficient, represented by K, is a

Received: August 12, 2025

International Journal of Applied Mathematics
Volume 38 No. 9s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

predetermined value, while T stands for the current temperature. The neighborhood search continues until a predefined number of iterations is reached. Once these iterations are completed, the system's temperature is gradually lowered. As the temperature decreases, the likelihood of accepting suboptimal solutions diminishes in the later stages. This enables the algorithm to converge toward a more optimal solution. The Simulated Annealing algorithm is designed to avoid being trapped in local optima by accepting worse solutions with a small probability. The solutions generated by Simulated Annealing are highly effective for tackling complex combinatorial optimization problems and have wide applicability in various real-world scenarios. The concept of annealing in the algorithm is inspired by the metal annealing process in metallurgy. Unlike other optimization algorithms, Simulated Annealing not only strives for better solutions in each iteration but also has a non-zero probability of accepting solutions with worse objective function values.

Advantages of Using the Simulated Annealing Algorithm in the Proposed Method

1. Very low memory consumption (unlike genetic algorithms, which have high memory consumption).
2. Its implementation is relatively simpler compared to other algorithms of its kind.
3. Due to its focus on local search, it usually finds acceptable solutions.
4. Because of the presence of a guided random process (with a low probability of accepting non-optimal solutions), it has the ability to escape from local optima.

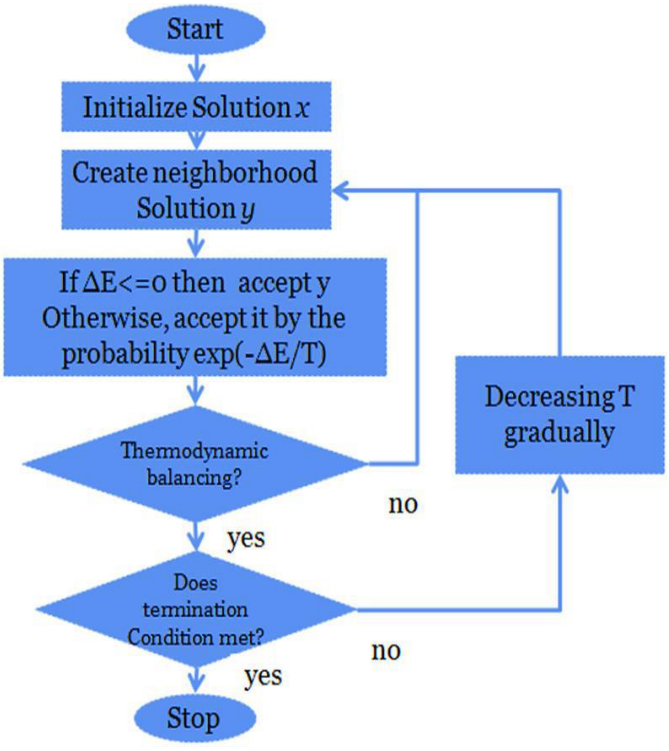

Fig. (1): Flowchart of the Simulated Annealing Algorithm.

3-3 Overview of the Proposed Method

Received: August 12, 2025

International Journal of Applied Mathematics
Volume 38 No. 9s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Load balancing in cloud computing is performed at two levels: the overall system level or the servers deployed within the system. In the proposed method, load balancing is addressed at the overall system level. To achieve load balancing, the simulated annealing optimization algorithm and the optimal vector are used. In the proposed approach, task allocation to the servers deployed in the system is carried out using an optimal vector, where the best values for the vector are selected using the simulated annealing optimization algorithm. Therefore, load balancing in the system is achieved by selecting the appropriate vector (in terms of optimal values). After selecting the most suitable vector, tasks are assigned to the servers. To maintain load balancing, tasks should be allocated to the servers in such a way that the load is evenly distributed across the entire system.

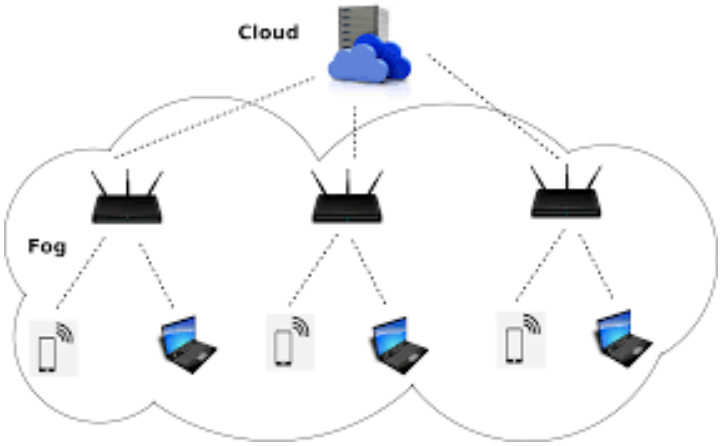

Fig. (2): Overall Architecture of the Proposed Method.

3-3-1 Explanation of the Proposed Method

In the proposed method, user requests are initially directed to the nearest edge server for execution. The requests then enter the execution queue. Simultaneously, another copy of the requests is sent to the cloud management system. The cloud management system estimates and informs how much time it will take for the requests to be executed on the cloud servers. This estimated time is then sent to the edge server. The same process is applied for the edge server, where the time required to execute the requests is determined (i.e., the time it will take to execute the requests). The edge server, by comparing the estimated execution time from both the edge server and the cloud, makes a decision whether to forward the requests to the cloud for execution or to handle them locally. If the estimated time from the cloud is shorter than the edge server's time (i.e., the cloud can execute the requests faster), the requests are sent to the cloud management system for execution. Therefore, if the edge server can execute the requests in the shortest time, it will handle the execution and return the results to the user. Otherwise, the requests are forwarded to the cloud management system. The execution of the requests in the cloud servers follows this process: upon receiving the requests, the cloud management system considers an optimal vector or allocation vector that determines which request should be assigned to which server for execution, ensuring that the pre-determined objectives, namely load balancing in the system, are achieved.

Received: August 12, 2025

International Journal of Applied Mathematics
Volume 38 No. 9s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Therefore, the cloud management system, by considering the optimal vector, makes a decision about which server (or virtual machine) to send the request to, and the simulated annealing algorithm is applied in the decision-making process.

An example of the optimal vector with four servers and ten requests is shown in Fig. 3. The columns of the vector represent the number of requests, and the rows of the vector represent the number of virtual machines.

Servers	Requests									
	1	2	3	4	5	6	7	8	9	10
1										
2										
3										
4										

Fig. (3): An example of the optimal vector in the proposed method.

The initial values of this vector are randomly assigned, and the values of the vector cells are either zero or one. For example, in Fig. 4, which shows an optimal vector with several values, the assignment may be such that request number 1 is assigned to server 3, or request number 2 is assigned to server 2; therefore, the value of this cell in the matrix becomes one.

Servers	Requests									
	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	1	0	0	0	0	0
2	0	1	0	0	0	0	1	0	0	0
3	1	0	0	0	0	0	0	0	1	0
4	0	0	0	0	0	0	0	0	0	1

Fig. (4): Initialization of the optimal vector.

With the initialization of the vector and based on the simulated annealing algorithm, the initial temperature is determined, which in the proposed method is set to 2000. In each iteration (or execution) of the algorithm, the temperature is reduced by one unit. The execution of the simulated annealing algorithm in the proposed method is as follows: in each iteration, all the neighboring states related to the initial vector are generated. Then, the best neighbor is selected (based on the cost function value), and Eq.3-1 is calculated:

$$dif = Cost(Best\ neighbor) - Cost(current\ state) \quad (1)$$

$$Cost = \sqrt{Var(LPC)} \quad (2)$$

Received: August 12, 2025

International Journal of Applied Mathematics
Volume 38 No. 9s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

By calculating the desired equation, if the value of dif is negative, the neighbor is accepted and considered as the current state. However, if dif is not negative, the neighbor is considered as a worse neighbor, and Eq. 3 is calculated:

$$if(dof)$$
$$e^{\frac{-dif}{T}} > random\ number \tag{3}$$

Where T is the current temperature. If the above condition holds, the worse neighbor is accepted.

The simulated annealing algorithm, considering the cost function and the above equations, determines the optimal vector, which is considered the best allocation vector for requests. Therefore, finding the best values for the vector is done using the simulated annealing algorithm. After selecting the optimal vector with appropriate values, the requests are transferred to the cloud servers for execution.

The capacity of a server is defined as:

$$C_j = Pe_{numj} \times Pe_{mipsi} + VM_{bwj} \tag{4}$$

Where $Pe_{numj}$is the number of processors in VM_j, $Pe_{mipsi}$is the million instructions per second (MIPS) of all processors in VM_j, and $VM_{bwj}$is the bandwidth capability of the communication for VM_j.

The load on a server

The load of virtual machine is defined as the number of tasks in the service queue of VM_i(N), divided by the service rate of VMi at time t (denoted as S):

$$L = \frac{N}{S} \tag{5}$$

The total load across all virtual machines is defined as:

$$L = \sum_{i=1}^{K} L_i \tag{6}$$

Where i represents the number of virtual machines. The unit capacity of each load is defined as:

$$LPC = \frac{L}{\sum_{i=1}^{M} C_i} \tag{7}$$

Where C_i is the capacity of the server.

This section presented a novel method for load balancing in cloud computing based on the simulated annealing optimization algorithm. The proposed method focused on achieving load balancing at the system level, addressing which tasks should be assigned to which virtual machines to improve load distribution in the system. The main task of the proposed method, considering the simulated annealing optimization algorithm, is to determine the optimal allocation. Therefore, the processes involved in load balancing in the system are aimed at

Received: August 12, 2025

International Journal of Applied Mathematics
Volume 38 No. 9s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

calculating the overall system balance. The role of the simulated annealing optimization algorithm is to select the best values for a one-dimensional array. In the proposed system, the objective is to provide load balancing while focusing on improving execution time in the cloud environment.

4- Results and Analysis

4-1- Introduction

In the previous section, a method for load balancing in a fog computing environment was presented, utilizing the simulated annealing algorithm and an optimal vector. This section evaluates the performance of the proposed method. The implementation was carried out using MATLAB version 2021, with a two-layer system combining both fog and cloud computing (incorporating multiple servers). To assess the effectiveness of the proposed approach, its results were compared with other algorithms, including SHC, GHC, Min-Min, Max-Min, PSO, and Throttled, across various metrics such as load values, imbalance degree, response time, and total execution time. The results demonstrate the superior performance of the proposed method for load balancing in both fog and cloud computing environments.

The Throttled algorithm maintains an index table of virtual machines and their statuses. In contrast, the proposed algorithm seeks to enhance response time while optimizing the use of available virtual machines. It utilizes a selection process to choose a virtual machine for handling client service requests, and if the selected virtual machine is available, the request is assigned to it. The MIN-MIN algorithm prioritizes the shortest tasks when assigning servers. For each task, the minimum completion time is determined, and the task with the shortest completion time is allocated to the appropriate processor. The MIN-MAX algorithm is used in batch processing, where tasks are scheduled at predefined intervals. These exploratory methods enable batch processing to learn and adjust based on real-time execution times. The SHC (Stochastic Hill Climbing) algorithm randomly selects moves toward the peak, with the probability of selection varying according to the steepness of the climb. The GHC (Greedy Hill Climbing) algorithm, on the other hand, greedily selects moves toward the peak, aiming to find the best or a sufficiently optimal solution.

4-2- Evaluation of Implementation Results

In this section, the performance of the proposed method is assessed by analyzing the results obtained from its implementation. The data used for the implementation were sourced from NASA, covering the months of July and August 1995. As mentioned earlier, the experimental setup involves a system that integrates both cloud and fog computing environments. In the implementation, the specifications for each virtual machine in the fog computing environment include several factors: the number of servers, the number of CPUs per server, CPU frequency, server memory, bandwidth, storage capacity, processing cost per time unit, and the data center ID. For instance, the specifications for 10 virtual machines are detailed in Table 1. Table 2, on the other hand, presents the specifications related to the data centers utilized in the proposed method's implementation.

Table 1: Specifications of servers in fog computing.

International Journal of Applied Mathematics
Volume 38 No. 9s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

ServerID	CPU Cnt	Freq (GHz)	RAM (GB)	BW (MB)	HDD (GB)	Proc Cost Per Time Unit	DataCenterID
1	8	2.4	4	100	1000	1.8	1
2	4	1.2	4	100	1000	1.2	2
3	2	2.4	4	100	1000	1.6	3
4	2	2.4	4	100	1000	1.2	4
5	2	2.8	4	100	1000	1.4	5
6	4	2.4	4	100	1000	1.8	6
7	4	2.8	4	100	1000	1.8	7
8	4	1.2	4	100	1000	1.6	8
9	8	2.4	4	100	1000	1.2	9
10	8	2.4	4	100	1000	1.6	10

Table 2: Specifications of data centers in the cloud environment.

DataCenterID	Server Count	CPU Count	Freq (GHz)	RAM (GB)	BW (MB)	HDD (GB)	Proc Cost Per Time Unit	RegionID
1	4	8	2.4	4	100	1000	1.8	1
2	8	4	1.2	4	100	1000	1.2	1
3	2	2	2.4	4	100	1000	1.6	1
4	4	2	2.4	4	100	1000	1.2	1
5	4	2	2.8	4	100	1000	1.4	1
6	6	4	2.4	4	100	1000	1.8	2
7	3	4	2.8	4	100	1000	1.8	2
8	4	4	1.2	4	100	1000	1.6	2
9	4	8	2.4	4	100	1000	1,2	2
10	8	8	2.4	4	100	1000	1.6	2
11	4	8	2.8	4	100	1000	1.2	3
12	8	8	2.4	4	100	1000	1.4	3

Received: August 12, 2025

International Journal of Applied Mathematics
Volume 38 No. 9s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

13	4	4	1.2	4	100	1000	1.8	3
14	4	4	2.4	4	100	1000	1.8	3
15	4	4	2.8	4	100	1000	1.6	3
16	4	4	2.8	4	100	1000	1.2	4
17	4	2	2.4	4	100	1000	1.8	4
18	2	2	2.4	4	100	1000	1.8	4

Figs. 5 and 6 show the comparison of workload values on cloud and fog computing servers for the proposed algorithm and the SHC, GHC, Min-Min, Max-Min, Throttled, and PSO algorithms, respectively. This metric represents the amount of load on the data centers in the cloud environment as well as the load on the servers in the fog environment, with the unit being measured in megabytes. Based on the obtained results, the workload values on fog and cloud servers for the proposed algorithm are lower compared to the other algorithms. This indicates an effective distribution of tasks across the servers, particularly the fog servers.

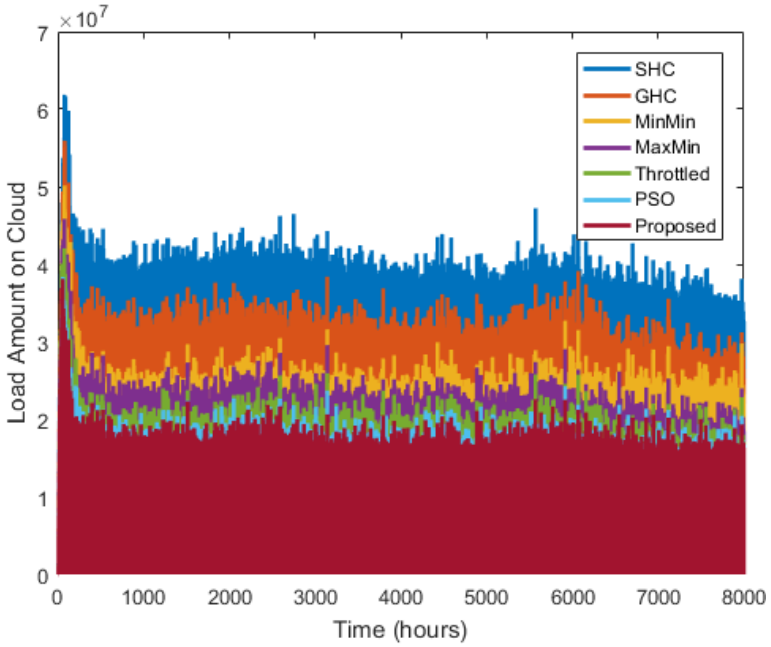

Fig. 5: Comparison of workload values on the cloud.

Received: August 12, 2025

International Journal of Applied Mathematics
Volume 38 No. 9s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

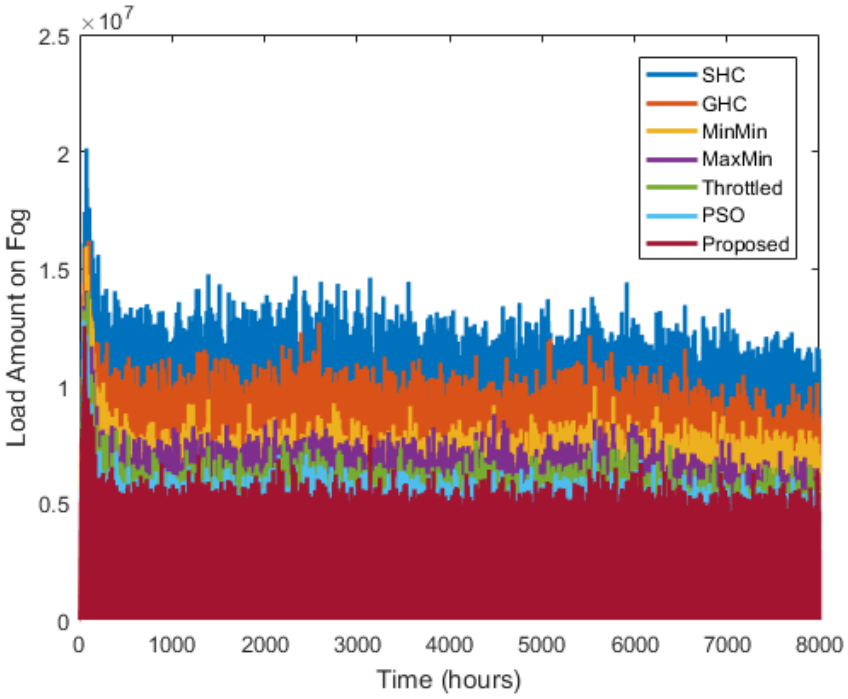

Fig. 6: Comparison of workload values on the fog.

The imbalance metric indicates the degree of imbalance in the system (fog and cloud). Load imbalance refers to the improper distribution of tasks across servers, resulting in an imbalance between the tasks and servers in terms of their power and capacity to execute them. Fig. 7 shows the degree of load imbalance on servers for the proposed method and other algorithms. According to the obtained results, the load distribution in the proposed method is more balanced compared to the other methods, and a logical load balance exists within the system. Additionally, in the proposed method, the workload on the servers is lower compared to the other methods. This is because the proposed algorithm, using an efficient approach, logically allocates workloads to servers at each time step (hour), which aligns with one of the objectives of the proposed method: improving the system's performance.

This metric represents the workload on servers in the cloud layer, with the unit measured in megabytes. Figure 8 shows the workload values on servers for the proposed method and the methods SHC, GHC, Min-Min, Max-Min, Throttled, and PSO. As shown in Figure 8, in the proposed method, the workload values on servers remain low throughout the entire time steps, and the tasks assigned to servers are executed earlier, preventing workload accumulation on the servers.

Received: August 12, 2025

International Journal of Applied Mathematics
Volume 38 No. 9s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

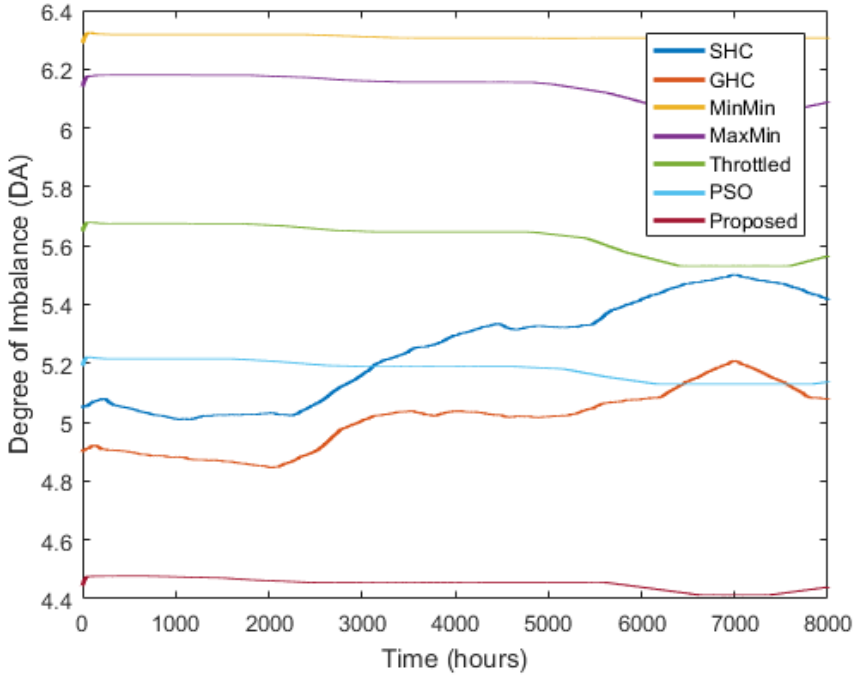

Fig. 7: Comparison of the load imbalance degree metric on servers between the proposed method and other methods.

However, for the methods SHC, GHC, Min-Min, Max-Min, Throttled, and PSO, the workload values on servers are higher than those of the proposed method. These values do not decrease significantly as the time steps increase up to 600, but beyond time step 600, the workload decreases in both the proposed method and other methods. The highest workload on servers is observed with the SHC method, indicating the inefficiency of this method in properly distributing the workload across servers.

The MakeSpan metric represents the total execution time of a job. It refers to the time taken from when a job is injected into the network until the computations related to that job are completed and the job exits the network. Proper distribution of tasks across servers and data centers leads to optimization and a reduction in the total execution time. Fig. 9 shows a comparison of the total execution time results for the proposed method and other methods. According to the results obtained, the use of the simulated annealing algorithm and the optimal vector for workload balancing has reduced the MakeSpan time compared to other methods.

The total execution time is the most commonly used parameter for measuring optimization methods. Optimizing this parameter indicates that tasks are completed in a reasonable amount of time and all tasks are executed within an acceptable timeframe. As seen in the implementation results, using the proposed method reduces the time required to complete the longest job compared to other protocols. This is because the proposed method focuses on the efficient allocation of tasks in the cloud.

Received: August 12, 2025

International Journal of Applied Mathematics
Volume 38 No. 9s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

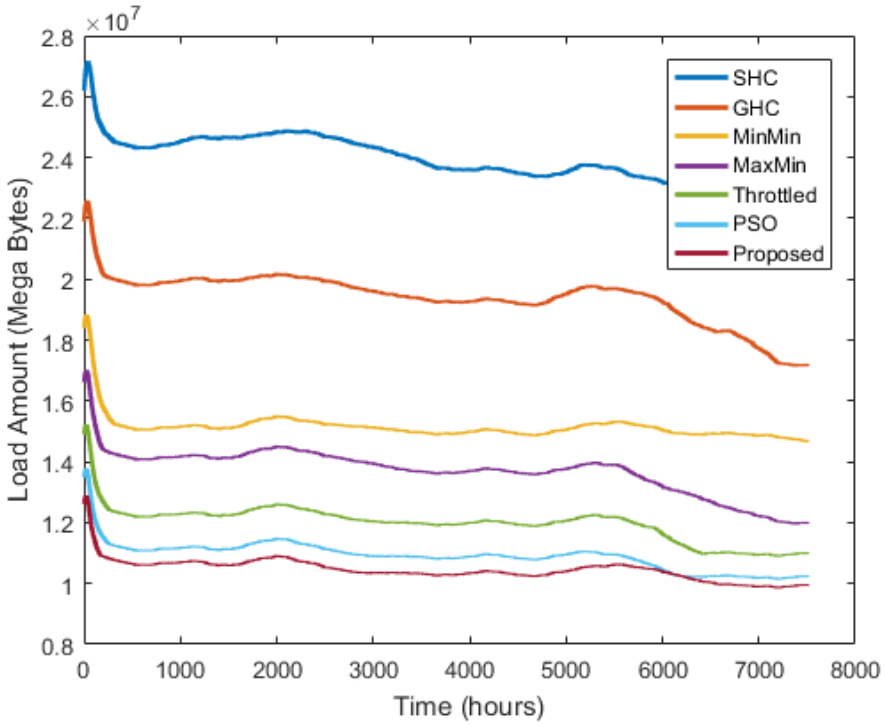

Fig. 8: Comparison of workload values on servers between the proposed method and other methods.

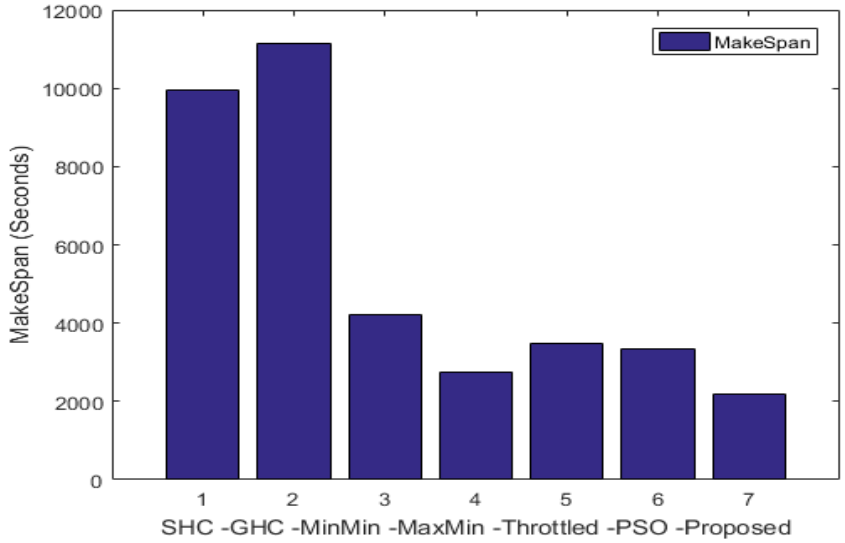

Fig. 9: Comparison of MakeSpan results between the proposed method and other methods.

The response time metric represents the time taken to respond to tasks. It is clear that the more tasks are properly distributed across data centers and servers within the system, the shorter and more optimized their response time will be. The unit for this metric in the implementation of the proposed method is considered in seconds. Fig. 10 shows a comparison of the response time results for the proposed method and other methods. As evident from the results of the proposed method's implementation, the use of the simulated annealing algorithm and the optimal vector for workload balancing leads to a reduction in response time by effectively allocating tasks among servers. Therefore, the reduction in response time in the proposed

Received: August 12, 2025

International Journal of Applied Mathematics
Volume 38 No. 9s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

method is due to the fact that effective task allocation and the selection of appropriate servers have increased the system's response speed to requests. Thus, the workload balancing algorithm aims to minimize task execution time through the proper distribution of tasks across servers.

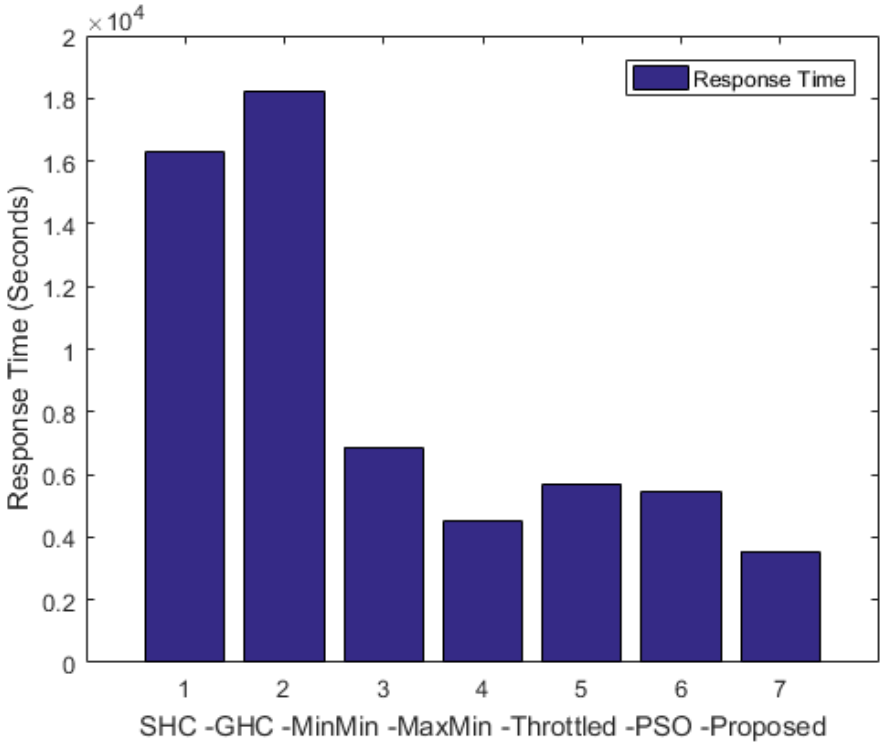

Fig. 10: Comparison of response time results between the proposed method and other methods.

In this section, the results of the proposed method for workload balancing in the cloud computing environment were studied and evaluated. The proposed method was implemented using the NASA dataset for the months of August and July 1995. The implementation was carried out using MATLAB 2021 software, and the analysis of the results demonstrated effective task allocation between servers, leading to an improved system state in both cloud and fog environments. Various parameters were considered for evaluation, and the proposed method was compared with other methods such as SHC, GHC, Min-Min, Max-Min, Throttled, and PSO. Experimental results show that the proposed method significantly reduces response time and improves the system's performance.

5- Conclusion

Load balancing refers to the dynamic redistribution of workloads between servers or to underutilized machines. Given its significant impact on the efficiency of both cloud and fog computing systems, load balancing has garnered substantial attention from researchers in recent years, leading to extensive studies in this field. The ongoing goal is to improve system performance by enhancing computational speed and optimizing the use of available resources. Load balancing methods are typically classified into static and dynamic approaches. Static algorithms are more effective in homogeneous and stable environments, delivering excellent results in these contexts. However, they are less adaptable and cannot accommodate dynamic changes in system characteristics during execution. On the other hand, dynamic algorithms are

Received: August 12, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

more flexible, considering various task-related features in the system both before and during execution.

Load balancing plays a crucial role in improving system performance by redistributing workloads across processors. It ensures that the load is distributed fairly among nodes, preventing any single node from becoming overloaded. Effective load balancing leads to optimal resource utilization, reduced response times, increased throughput, and high system efficiency. In recent years, several load balancing algorithms have been proposed for fog computing environments. While these algorithms offer benefits, they also have certain limitations. To address these challenges, this paper introduces a method based on simulated annealing and an optimal vector for load balancing in fog computing environments. The proposed system performs load balancing at the overall system level, aiming to achieve the optimal allocation of tasks to servers to maintain load balance, particularly in the cloud. The method uses an optimal vector to assign tasks to servers, with the values for this vector determined by a simulated annealing optimization algorithm, ensuring a balanced load distribution in the cloud. Once the appropriate vector values are identified, tasks are dispatched to the servers.

The proposed method was implemented using MATLAB 2021 and the NASA dataset. The performance of the method was evaluated using various parameters and compared with other algorithms, including SHC, GHC, Min-Min, Max-Min, PSO, and Throttled. The results showed a reduction in response time and overall time compared to the other methods. Moreover, the load balancing state was effectively maintained, and the proposed method achieved logical load distribution within the system. Future research could focus on enhancing the proposed method by considering timing issues, improving resource selection and discovery for request execution, accounting for request loads in decision-making for forwarding tasks, addressing the execution level of specific tasks based on user decisions, and applying the proposed approach to other metaheuristic algorithms, such as cuckoo and firefly.

REFERENCES

[1] Lavanya, R. (2019). Fog Computing and Its Role in the Internet of Things. In Advancing Consumer-Centric Fog Computing Architectures (pp. 63-71). IGI Global.

[2] Xu, X., Fu, S., Cai, Q., Tian, W., Liu, W., Dou, W., ... & Liu, A. X. (2018). Dynamic resource allocation for load balancing in fog environment. *Wireless Communications and Mobile Computing*, *2018*.

[3] Yousefpour, A., Ishigaki, G., & Jue, J. P. (2017). Fog computing: Towards minimizing delay in the internet of things. In Edge Computing (EDGE), 2017 IEEE International Conference on (pp. 17-24). IEEE.

[4]. Atlam, H.F.; Alenezi, A.; Walters, R.J.; Wills, G.B. An Overview of Risk Estimation Techniques in Risk-based Access Control for the Internet of Things. In Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security (IoTBDS 2017), Porto, Portugal, 24–26 April 2017; pp. 254–260.

Received: August 12, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[5]. Atlam, H.F.; Alenezi, A.; Hussein, R.K.; Wills, G.B. Validation of an Adaptive Risk-Based Access Control Model for the Internet of Things. *Int. J. Comput. Netw. Inf. Secur.* 2018, *10*, 26–35.

[6] F. Liu et al., "NIST cloud computing reference architecture," NIST Spec. Publ., vol. 500, no. 2011, p. 292, 2011.

[7] Heena I. Naghma A. Survey On Cloud Computing. International Journal of Emerging Technology and Advanced Engineering, Volume 3, Issue 4, April 2013.

[8] Bonomi, Flavio, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. "Fog computing: A platform for internet of things and analytics." In *Big data and internet of things: A roadmap for smart environments*, pp. 169-186. Springer, Cham, 2014.

[9] Ilyas, M. U., Ahmad, M., & Saleem, S. (2020). Internet-of-Things-Infrastructure-as-a-Service: The democratization of access to public Internet-of-Things Infrastructure. International Journal of Communication Systems, e4562.

[10] Zhang, PeiYun, MengChu Zhou, and Giancarlo Fortino. "Security and trust issues in Fog computing: A survey." *Future Generation Computer Systems* 88 (2018): 16-27. 5.008.

[11] Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012, August). Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing* (pp. 13-16). ACM.

[12] Liu, Y.; Fieldsend, J.E.; Min, G. A Framework of Fog Computing: Architecture, Challenges and Optimization. *IEEE Access* 2017, *4*, 1–10.

[13] Sarkar, S., Chatterjee, S., & Misra, S. (2015). Assessment of the Suitability of Fog Computing in the Context of Internet of Things. *IEEE Transactions on Cloud Computing*, *6*(1), 46-59.

[14] Mukherjee, M., Matam, R., Shu, L., Maglaras, L., Ferrag, M.A., Choudhury, N., & Kumar, V. (2017). Security and Privacy in Fog Computing: Challenges. IEEE Access, 5, 19293-19304.

[15] Aazam, M., & Huh, E. N. (2016). Fog computing: The cloud-iot\/ioe middleware paradigm. *IEEE Potentials*, *35*(3), 40-44.

[16] Atlam, Hany & Walters, Robert & Wills, Gary. (2018). Fog Computing and the Internet of Things: A Review. Big Data and Cognitive Computing. 2.

[17] Souza, V. B. C., Ramírez, W., Masip-Bruin, X., Marín-Tordera, E., Ren, G., & Tashakor, G. (2016, May). Handling service allocation in combined fog-cloud scenarios. In *2016 IEEE international conference on communications (ICC)* (pp. 1-5). IEEE.

[18] Masip-Bruin, X., Marín-Tordera, E., Tashakor, G., Jukan, A., & Ren, G. J. (2016). Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems. *IEEE Wireless Communications*, *23*(5), 120-128.

Received: August 12, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[19] Deng, R., Lu, R., Lai, C., & Luan, T. H. (2015, June). Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing. In *2015 IEEE International Conference on Communications (ICC)* (pp. 3909-3914). IEEE.

[20] Baek, J. Y., Kaddoum, G., Garg, S., Kaur, K., & Gravel, V. (2019). Managing Fog Networks using Reinforcement Learning Based Load Balancing Algorithm. *arXiv preprint arXiv:1901.10023*.

[21] Zahid, M., Javaid, N., Ansar, K., Hassan, K., Khan, M. K., & Waqas, M. (2018). Hill Climbing Load Balancing Algorithm on Fog Computing. In International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (pp. 238-251). Springer, Cham.

[22] Nazir, S., Shafiq, S., Iqbal, Z., Zeeshan, M., Tariq, S., & Javaid, N. (2018). Cuckoo optimization algorithm based job scheduling using cloud and fog computing in smart grid. In International Conference on Intelligent Networking and Collaborative Systems (pp. 34-46). Springer, Cham.

[23] Saif, T., Javaid, N., Rahman, M., Butt, H., Kamal, M. B., & Ali, M. J. (2018). Round Robin Inspired History Based Load Balancing Using Cloud Computing. In International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (pp. 496-508). Springer, Cham.

[24] Ahmad, N., Javaid, N., Mehmood, M., Hayat, M., Ullah, A., & Khan, H. A. (2018). Fog-cloud based platform for utilization of resources using load balancing technique. In International Conference on Network-Based Information Systems (pp. 554-567). Springer, Cham.

[25] Talaat, F. M., Ali, S. H., Saleh, A. I., & Ali, H. A. (2019). Effective Load Balancing Strategy (ELBS) for Real-Time Fog Computing Environment Using Fuzzy and Probabilistic Neural Networks. Journal of Network and Systems Management, 1-47.

[26] Rahim, M. H., Javaid, N., Rahim, S., Naz, M., Akbar, M., & Javed, F. (2018). Weighted Cuckoo Search Based Load Balanced Cloud for Green Smart Grids. In International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (pp. 252-264). Springer, Cham.

[27] Wan, J., Chen, B., Wang, S., Xia, M., Li, D., & Liu, C. (2018). Fog computing for energy-aware load balancing and scheduling in smart factory. IEEE Transactions on Industrial Informatics, 14(10), 4548-4556.

[28] Xu, X., Liu, Q., Qi, L., Yuan, Y., Dou, W., & Liu, A. X. (2018). A Heuristic Virtual Machine Scheduling Method for Load Balancing in Fog-Cloud Computing. In 2018 IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing,(HPSC) and IEEE International Conference on Intelligent Data and Security (IDS) (pp. 83-88). IEEE.

[29] Zubair, M., Javaid, N., Ismail, M., Zakria, M., Zaheer, M. A., & Saeed, F. (2018). Integration of Cloud-Fog Based Platform for Load Balancing Using Hybrid Genetic

Received: August 12, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Algorithm Using Bin Packing Technique. In International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (pp. 279-292). Springer, Cham.

[30] Beraldi, R., & Alnuweiri, H. (2018). Sequential Randomization load balancing for Fog Computing. In 2018 26th International Conference on Software, Telecommunications and Computer Networks (SoftCOM) (pp. 1-6). IEEE.

[31] Puthal, D., Obaidat, M. S., Nanda, P., Prasad, M., Mohanty, S. P. (2018). Secure and sustainable load balancing of edge data centers in fog computing. IEEE Communications Magazine, 56(5), 60-65.

[32] Manju, A. B., & Sumathy, S. (2019). Efficient Load Balancing Algorithm for Task Preprocessing in Fog Computing Environment. In Smart Intelligent Computing and Applications (pp. 291-298). Springer, Singapore.

[33] Yin, Chao, Qin Fang, Hongyi Li, Yingjian Peng, Xiaogang Xu, and Dan Tang. "An optimized resource scheduling algorithm based on GA and ACO algorithm in fog computing." *The Journal of Supercomputing* 80, no. 3 (2024): 4248-4285.

[34] Ogundoyin, Sunday Oyinlola, and Ismaila Adeniyi Kamil. "Optimal fog node selection based on hybrid particle swarm optimization and firefly algorithm in dynamic fog computing services." *Engineering Applications of Artificial Intelligence* 121 (2023): 105998.

[35] Singh, Prabhdeep, Rajbir Kaur, Junaid Rashid, Sapna Juneja, Gaurav Dhiman, Jungeun Kim, and Mariya Ouaissa. "A fog-cluster based load-balancing technique." *Sustainability* 14, no. 13 (2022): 7961.

[36] Yu, Dongxian, and Weiyong Zheng. "A hybrid evolutionary algorithm to improve task scheduling and load balancing in fog computing." *Cluster Computing* 28, no. 1 (2025): 1-26.

[37] Shaik, Mahaboob Basha, Kunam Subba Reddy, K. Chokkanathan, Sardar Asad Ali Biabani, P. Shanmugaraja, and DR Denslin Brabin. "A Hybrid Particle Swarm Optimization and Simulated Annealing with Load Balancing Mechanism for Resource Allocation in Fog-Cloud Environments." *IEEE Access* (2024).

[38] Saoud, Afaf, and Abdelmadjid Recioui. "Hybrid algorithm for cloud-fog system-based load balancing in smart grids." *Bulletin of Electrical Engineering and Informatics* 11, no. 1 (2022): 477-487.

Received: August 12, 2025