International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V73I5P129

Volume 73 Issue 5, 359-368, May 2025

Original Article

A Greedy Constraints-based Fog Computing Model to Optimize Task Scheduling

Monika[1], Harkesh Sehrawat[2], Vikas Siwach[3]

[1,2,3]*University Institute of Engineering and Technology, Maharshi Dayanand University, Rohtak, Haryana, India.*

[2]*Corresponding Author : sehrawat_harkesh@mdurohtak.ac.in*

Received: 09 August 2024 Revised: 08 March 2025 Accepted: 19 May 2025 Published: 31 May 2025

Abstract *- Fog computing ensures effective request processing and service distribution in real-time IoT applications. Instant request processing, reliable execution, and effective QoS are the key requirements of such applications. However, the increasing requests and broad coverage of this network can increase the network load. An effective resource allocation and scheduling method is required to utilize the available resources and to reduce execution failure. Higher delay and makespan are the key challenges of scheduling algorithms and fog computing. In this paper, a multiple-constraint adaptive greedy task scheduler is designed to optimize the functioning of task scheduling. The functioning of the proposed scheduling model is divided into two stages. In the first stage, a multiple constraint-based resource allocation is done. In this stage, task criticality and resource priority-based mapping methods are defined to optimize the resource scheduling. In the final stage, the deadline and delay adaptive greedy method is defined to schedule the requests. The comparative evaluation is done against the FCFS, SJF, GTS, and DPTS methods. The algorithm reduced the average delay and makespan in comparison with state-of-the-art methods.*

Keywords *- Fog computing, Task scheduling, Greedy, Resource allocation, IoT.*

1. Introduction

IoT networks are location-aware, lightweight, real-time, and application-driven networks that support large volumes of data flow and request processing in a restricted timeframe. Security, reliability, efficient data transmission and low power request processing are the key requirements of these networks. Cloud computing is a distributed network that ensures high processing power and storage with centralized control. However, it did not confirm efficient data transmission with lower response time. The limited bottleneck of cloud computing slows down the request processing and increases the error rate in IoT networks. Fog computing is one such lightweight technology that provides processing devices and data centers at the edge of networks [1]. This distributed network is capable of handling real-time situations such as heterogeneous devices, heavy network density, bottleneck situations, etc. This network ensures flexible communication and computing by establishing an interconnection over the resources and devices present at the network edge. Fog computing performs local resource pooling to perform effective data transfer and communication over the network. Fog computing is an extension of cloud computing that can provide interaction with the cloud as well as confirm the request processing at the edge. The service and application-driven network is very adaptive for smart homes, smart grids, V2V networks, smart hospitals and other IoT-based applications [2, 3].

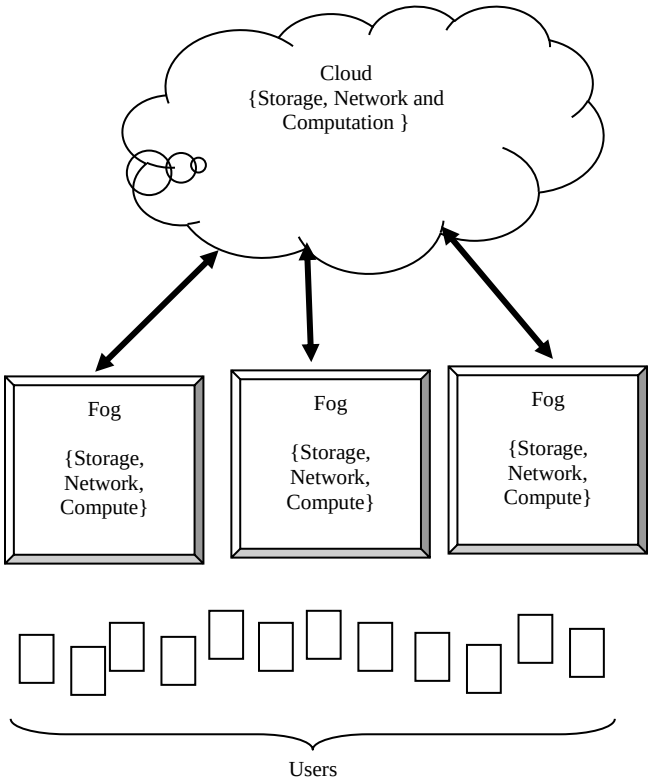

Fig. 1 Layered architecture of fog computing

The decentralized architecture and distributed behaviour of fog computing are provided in Figure 1. It is a layered architecture in which the lowest layer is connected to real-time and application-specific users. These users can generate requests through various IoT devices to perform their requests. These devices capture real-time and environmental information and submit the requests to the fog layer. This layer contains a variety of controller and processing devices, including workstations, gateways, routers, switches, etc. This also contains immediate processing devices and storage units. The top cloud layer collects and controls all these activities over the lower layer and provides centralized processing and storage services [4, 5]. Figure 2 shows the task scheduling model for fog computing. In an application-based distributed environment, a large number of requests are generated by users and the environment through IoT and smart appliances. The intermediate fog layer processes these requests and the request processor to allocate the required processes and to line up for task execution. The objective of the task scheduling algorithm is to maize the execution delay, execution time and cost of execution. The task scheduling algorithm faces various real-time challenges, including heavy load and bottleneck situations, low power resources, power consumption, etc. [6-8].

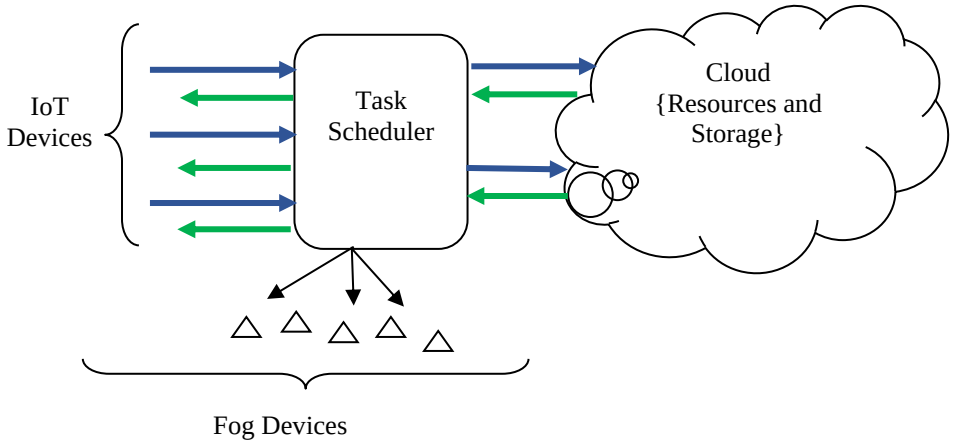

Fig. 2 Standard task scheduling architecture for fog computing

A smart grid is one such critical application of IoT that is responsible for the distribution of power and energy-related services. In this network, the data is collected in the form of power requests and the appliance's power usage. Several utilities related to IoT techniques are included in the smart grid environment. Various elements and behaviours of the smart grid are handled by integrating the IoT cloud, data governance, and sensor fusion. Business analytics may be applied to the obtained data to make decisions about monitoring and allocating energy sources. In the case of smart grids, IoT is a larger network that communicates at various levels to enhance energy distribution. The communication traffic in the network also suffers from numerous challenges, which may be addressed by conducting traffic maintenance and restoration in the IoT network. The Internet of Things (IoT) configures networks and devices and gathers data from distribution networks. The smart grid system collects various sorts of data at different periods. The acquired systematic data may be handled and processed in this smart grid to maximize the use of available energy and avoid severe load situations. A vast amount of data may be acquired since IoT devices capture data at regular intervals. The frequency of data collection is also an issue when working with a smart grid. It can cause complications with storage and consumption. Effective infrastructure and storage systems are necessary to handle and process this data. If the data is greater, it may have an impact on peak-hour performance or the ability to give meaningful advantages to customers. The obtained data may be examined to determine peak load and price scenarios. The network is fitted with smart meters for frequent monitoring and identifying consumer behaviour [9, 10].

1.1. Research Contribution

This paper provides a greedy constraint adaptive fog computing model for optimizing the functioning of task scheduling. The main contributions of the proposed model are listed below:

- A criticality and deadline adaptive resource allocation method is integrated with the prior stage to reduce the failure probability.
- A multi-objective greedy algorithm is applied within the second stage to obtain the sequence for each resource and to reduce the execution delay.
- The model was applied under deadline consideration so that starvation situations are avoided.
- The proposed model is validated against conventional and state-of-the-art methods under delay, execution time and execution ratio measures.

In this paper, a greedy and constraint adaptive model is presented to optimize the performance of the task scheduling

algorithm for fog computing. In this section, a brief introduction to fog computing architecture and the requirements of task scheduling is provided. Various challenges of task scheduling and fog computing are discussion discussed in this section. In section 2, the research contributions of various researchers to optimize the behaviour of task scheduling and fog computing are provided. Various optimization methods and models investigated by the researchers are discussed. In section 3, the proposed multiple-constraints adaptive task scheduling model is provided and discussed. The algorithm and functional process of the proposed model are discussed. In section 4, the experimental analysis and results are provided. The significance of the proposed model is validated in this section. In section 5, the conclusion and future scope are provided.

2. Related Work

This section provides a study on various optimization models and techniques proposed by the researchers to optimize the effectiveness and reliability of task scheduling. Saif et al. [11] proposed a multi-objective grey wolf optimization method to reduce energy consumption and delay in cloud computing. This model incorporated a non-Pareto dominance solution within the conventional hunting method of GWO to optimize its behaviour within task scheduling. The dynamic positioning of prey and wolves was considered under parametric constraints. The fitness function is defined with delay and energy consumption parameters. This work did not consider heterogeneity in requests and resources. The load imbalance and resource utilization were not analyzed in this work. Yeh et al. [12] proposed a local search analysis-based Bi-objective simplified swarm optimization system to enhance the effectiveness of task scheduling. This algorithm is integrated with a fast elite selecting mechanism to set up a non-dominating sorting of tasks with lesser time complexity. This hybrid algorithm generated a time-effective solution to the task scheduling problem and reduced the execution time and delay. The author did not investigate the work against a heavy load situation. The task priority and heterogeneity of resources were not considered in this work. Khan et al. [13] proposed a delay-sensitive and modified particle swarm optimization (MPSO) based method to optimize the task scheduling functioning in load balancing constraints. This algorithm enhanced the performance of task processing under energy consumption, cost, network bandwidth and delay parameters. The resource utilization was improved by up to 80% in this algorithm. The request and resource level heterogeneity was not considered in this research. Dai et al. [14] addressed the complex modeling challenges of fog computing and provided a multi-objective solution for this problem. The uncertainty problem of fog computing was resolved by using a dynamic priority mechanism. The multi-objective optimization model is integrated using weight adjustment to reduce the delay and energy consumption. This multi-objective model reduced the task execution cost and improved resource utilization in a heterogeneous environment. Jakwa et al. [15] investigated a deterministic spanning tree and modified the particle swarm optimization algorithm to optimize the task scheduling. This scheduler performed the resource allocation and managed the resources. This algorithm provided a load-balanced method to handle the heavy load situation of the IoT network. The algorithm was effectively defined for a heterogeneous environment and improved the service execution. The proposed algorithm reduced the energy consumption and response time. Liu et al. [16] provided a meta-heuristic hybrid scheduling algorithm to reduce the delay and energy consumption in fog computing. This hybrid algorithm is the combination of an artificial bee colony algorithm and a particle swarm optimization algorithm. Particle swarm optimization was used for optimizing the resource allocation under load balancing. This algorithm reduced the energy and computation time while processing a single fog cluster. In the second stage, an artificial bee colony algorithm was used to optimize fog computing. This resource scheduling method reduced the energy consumption and execution delay effectively against state-of-the-art methods.

Some researchers analyze the task processing history as the key factor in making decisions on resource allocation and task scheduling. They used machine learning and deep learning models and methods to identify the patterns and optimize the task scheduling. Iftikhar et al. [17] provided an effective task-scheduling model called Hunterplus for fog computing. This model integrated a recurrent bidirectional unit within a Gated Graph Convolutional Network (GGCN) to optimize the resource allocation and task scheduling methods. The experimental results confirm the reduction in energy consumption and task completion time by 17% and 10.4%. This work did not provide experiments under scalability and reliability considerations. Sharma et al. [18] investigated a two-stage model to optimize the task scheduling for a smart home-based fog computing architecture. In the first stage of this model, the usage history analysis and power requirement prediction were done using the Naive Bayes model. In the second stage, the Ant colony Optimization and Particle swarm optimization-based hybrid model is applied to optimize the behaviour of task scheduling. The analysis results identified a significant improvement achieved in terms of reduced latency, low power consumption and effective utilization of network resources. This work did not handle heavy loads and bottleneck situations. Imbrahim et al. [19] provided an effective solution by utilizing the limited capacity of computing resources. The author used a multi-objective deep reinforcement learning method to handle the load, task priority and node distance parameters. The author worked on task allocation scheduling behaviour by using agents of deep reinforcement learning. The method was analyzed against challenging scenarios and achieved significant results in terms of effective task completion time, transmission delay, propagation delay, processing delay, makespan and storage utilization.

Raju et al. [20] used the reinforcement learning model with mixed integer and non-linear programming to optimize fog computing for vehicular networks. The algorithm considered the vehicle mobility, resource limit and task deadline challenges in this work. The adaptive and intelligent modeling-based method reduced the delay by 12% and energy consumption by 14% against state-of-the-art methods. Choppara et al. [21] used the reinforcement learning model by effective placement of nodes. The model was defined for optimizing the delivery of healthcare services. The positional behaviour of architecture was improved under a limited resource specification. The model also computed the resource demand. The model was tested on different scenarios, and a dynamic learning method was applied to optimize the task execution. Sui et al. [22] proposed Mayfly based fusion algorithm under multiple parameter-based analysis. The proposed algorithm improved the diversity and efficiency by including global learning and coefficient-driven mapping. The algorithm reduced the operating cost and time. The energy consumption and communication delay were also reduced. Some of the recent methods proposed by research for optimizing task scheduling methods are provided in Table 1.

Table 1. Recent task scheduling meta-heuristic and optimization methods

Author	Task Scheduler	Significance	Limitation
Kumar et al. [23]	Improved Dingo Optimization	Optimized makespan time, reduced VM failure rate, reduced degree of imbalance.	Heterogeneity was not considered in tasks and resources.
Tran-Dang et al. [24]	Dynamic Collaborative Task Offloading with parallel computation to optimize task execution.	Reduced Average Delay, Effective for Heterogeneous environment, Improved Utilization ratio	Scalability was not considered in the experimental evaluation.
Mohammadzadeh et al. [25]	Hybrid discrete Symbiotic Organisms Search-Grasshopper Optimization Algorithm (HDSOS-GOA) was proposed. SOS improved search capability, and GOA improved workflow scheduling.	Reduced the energy consumption and number of VMs required, Improved Energy Utilization	Work was not validated in a heterogeneous environment.
Khan et al. [26]	Ripple Induced Whale Optimization Algorithm for utilizing the ripple effect to schedule independent tasks	Minimize makespan and energy consumption, maximize throughput	A solution for the Load imbalance problem was not provided
Tian et al. [27]	Hybrid Ant Lion Optimization was defined with a generation hopping stage to handle a diversity of environments.	Improved convergence speed, improved accuracy, reduced latency and energy consumption.	Varied and heavy load situations were not experimented with.
Kumar et al. [28]	Combined Electric Fish Optimization (EFO) and Earthworm Optimization Algorithm (EOA) to handle heterogeneous workload and QoS	Improved efficiency, Reduced Energy Consumption, and Reduced Total Cost	No predictive method was included, and workflow analysis was not considered
Saad et al. [29]	Particle swarm optimization and Genetic algorithm-based hybrid methods were provided to provide a compelling solution to task scheduling.	Reduced Execution time, Improved response time, Improved performance of task completion	Heavy load situations and priority in tasks and resources were not adopted.

3. Proposed Multiple-Constraint Adaptive Greedy Task Scheduler

In this paper, a multiple-constraint adaptive greedy method is proposed to optimize the efficiency and reliability of fog computing. The functional capability of this model is divided into two main aspects. The first aspect is the constraints that include performing effective resource allocation that will avoid switching between the fog devices and considering the deadline so that the task will be executed before any cut-off time. The second aspect associated with this work is the greedy approach that is applied to the key parameters or objectives of the task scheduling method. The greedy approach is applied to the average delay and execution time parameters. The functional stages of this proposed model are provided in Figure 3.

This model is implemented between the end-user layer and the fog layer. The input to this model is in the form of user requests that are generated either by the users or captured by the IoT devices from the environment. Each of the generated requests or tasks is defined with multiple parameters, including task_time, task_criticality and task_length.

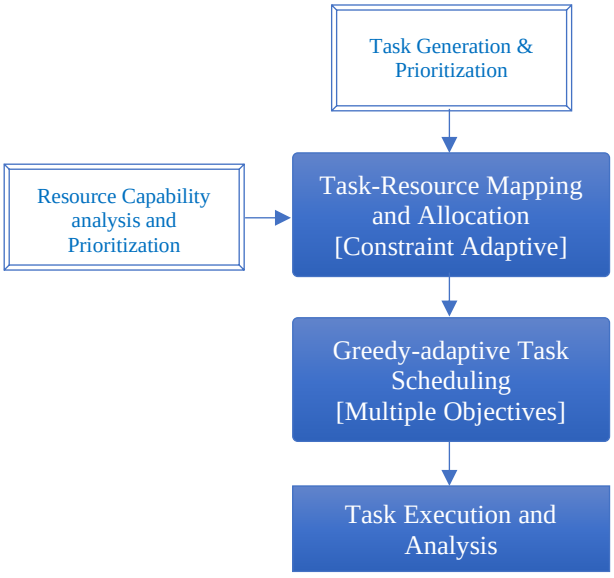

Fig. 3 Multiple-constraint adaptive greedy task scheduling model for fog computing

These requests are processed, and the task priority is computed in the first stage of the proposed algorithm. The prioritization method used in this work is provided in equation (1).

Task.Priority = task_criticality * (1 - task_deadline /max_deadline) (1)

The task_criticality is a key factor used to decide the priority of the task. The tasks associated with some real-time phenomenon or under a hard deadline constraint are considered high-priority tasks. These tasks are required to be allocated and executed on fog devices. The fog devices are defined within the fog layer with specifications of certain memory, processing speed and capacity. Another dynamic measure computed for each fog device is the task completion ratio. It is a history analysis-based measure that identifies the number of tasks successfully executed and before the deadline. Based on these capacity and dynamic measures, the priority of the fog devices is computed. The priority computation of fog devices is provided in equation (2).

Set FNodes.Priority = w1* Memory + w2 * Task Completion Ratio + w3 * FNodes.Procss (2)

The Fog device with higher priority, processing speed and higher task completion ratio is considered a highly reliable and effective fog device. A fog device with higher priority can guarantee the successful execution of user tasks. Here, w1, w2 and w3 are the weights used to decide the priority of tasks. To identify the best values of w1,w2, and w3, multiple experiments are conducted, and the task completion ratio for each experiment is computed. The experimentation results for identifying the weights with the maximum task execution ratio are provided in Table 2. The results show that the maximum execution ratio achieved for this work is 96.43% for weights w1=0.4, w2=0.3, w3=0.3.

Table 2. Experimental results for effective weight evaluation

w1	w2	w3	Task Execution Ratio
0.1	0.1	0.8	45.76
0.1	0.2	0.7	49.67
0.1	0.3	0.6	51.45
0.2	0.1	0.7	53.56
0.2	0.2	0.6	56.77
0.2	0.3	0.5	59.32
0.3	0.1	0.6	65.31
0.3	0.2	0.5	73.47
0.3	0.3	0.4	78.49
0.4	0.1	0.5	83.54
0.4	0.2	0.4	87.64
0.4	0.3	0.3	96.43
0.5	0.1	0.4	88.43
0.5	0.2	0.3	82.21
0.5	0.3	0.2	74.35
0.6	0.1	0.3	70.04
0.6	0.2	0.2	64.67
0.6	0.3	0.1	51.34
0.7	0.1	0.2	48.97
0.7	0.2	0.1	46.33
0.8	0.1	0.1	42.84

Now, to perform the resource allocation, the request for resource mapping is under the multiple constraint-based method. According to these constraints, the requests are categorized based on the priority. While allocating the tasks to resources, a ratio of highly critical, medium critical, and low critical tasks is considered. The tasks with more number of critical and high-priority tasks are allocated to the high-priority resources. Similarly, the tasks with low priority and criticality are allocated to the low-priority resources. Once allocation is done, the greedy-based scheduling method is applied to each fog device to set up the order of task execution. For this greedy rule, each task is described in the ratio of deadline and delay parameters. This task ratio feature is provided in equation (3).

Task.Ratio = task.Delay / Task.DeadLineDiff (3)

The objective of this greedy method is to minimize the delay and the failure ratio. According to the proposed algorithm, the allocated tasks are executed in order of the Ratio parameter. The algorithmic detail and behaviour of this greedy approach are provided in Algorithm 1. The statistical measures and environmental setup are considered under some observations and assumptions. The assumptions considered in this work are provided below.

- The tasks executed by the users are independent, and their execution will not be affected by other tasks.

- A user can generate multiple requests, so several requests and users can be different.
- The deadline is considered a critical factor. The tasks executed after the deadline will be considered as failed execution.
- The re-execution of tasks is not considered in this work. In case of failure, the task will be re-generated by the user.
- The application of the task is not defined.

The tasks with a lower ratio are executed first. The proposed algorithm is simulated on iFogSim environment with multiple tasks and fog nodes. Multiple experiments are conducted with different configurations and tasks. The analysis results are provided in the next section.

Algorithm 1: Multiple-constraint Adaptive Greedy Scheduler (Tasks, FNodes)

1. Acquire Memory, Task Completion Ratio, and Processing Speed for FNodes
   [Obtain the resource features]
2. Set FNodes.Priority = w1* Memory + w2 * Task Completion Ratio + w3 * FNodes.Process
   [Compute the priority factor for each task]
3. Acquire Deadline, Criticality, and Expected Task Duration for Tasks
   [Obtain Task Features]
4. Categorize the tasks under deadline and criticality parameters.
5. Apply constraints for allocating tasks to available fog devices with a specific ratio of different criticality levels of tasks.
6. For i=1 to FNodes.Length
   {
7. Set the Greedy Rule Minimize $\sum_{j=1}^{Fnode(i).Tasks.lenth} FNode(i).Tasks(j).$ Delay/FNode(i).Tasks(j).DeadLineDiff
9. Arrange FNodes(i).Tasks in order of increasing Greedy Factor
10. Execute FNodes(i).Tasks and Record the Performance Measures
    }

Algorithm-1 defines the multi-constraint-based greedy scheduler for optimizing the performance of task execution in fog computing. This algorithm accepts the resources as FogNodes and tasks as user requests. The algorithm uses the statistical measures on both the resource and user ends to perform effective allocation and scheduling. The resource features, such as capacity, processing speed and history record, were acquired for resources. Based on this, the priority of fog nodes is computed. Another analysis is performed on tasks by obtaining the criticality level, task duration and deadline. The tasks are categorized based on deadline and criticality measures. Now the ratio-specific analysis is applied for allocating the tasks to specific fog devices. The greedy rule is defined to minimize the delay and increase the execution ratio on the same fog device. The greedy factor is computed, and tasks are executed in this greedy order. The experimental analysis and evaluation to analyze the performance of the proposed model is provided in the next section.

4. Results and Discussion

This paper proposes a multiple-constraints adaptive greedy method to optimize the task scheduling for fog computing. The algorithm was simulated using the iFogSim simulator. When operating in this environment, the input is obtained via Internet of Things devices or nodes that are connected to the application environment. In addition to being non-preemptive, the requests that are approved are diverse. The simulation is carried out on a system with an Intel-I5 processor operating at 3.1 GHz. This device comes equipped with 8 gigabytes of random access memory (RAM), 4 megabytes of cache, and an integrated iFogSim Toolkit. The makespan, failure rate, and average delay metrics are used in the comparative evaluation that is carried out. The initial metric examined in this study is Average Delay. Delay is defined as the disparity between the time of arrival and the time of execution of a job. Equations (4) and (5) denote the calculation of Delay and Average Delay.

$$\text{Delay} = \text{Time-of-Execution} - \text{Time-of-Arrival} \quad (4)$$

$$\text{Average delay} = \frac{\sum_{i=1}^{N} Delay^{i}}{N} \quad (5)$$

N represents the number of tasks.

Another performance metric utilized in this experiment is the average turnaround time or average execution time. The turnaround time is the difference between job completion time and the time of arrival. The execution time and average execution time are denoted in equations (6) and (7).

$$\text{Execution Time} = \text{Time of Completion} - \text{Arrival Time} \quad (6)$$

$$\text{AverageExecutionTime} = \frac{\sum_{i=1}^{N} Execution\ Time^{i}}{N} \quad (7)$$

N represents the number of tasks or requests created during a session.

A task is deemed unsuccessful if it is not completed before the deadline. The Failure Rate (FR) metric quantifies the ratio of failed tasks to the total tasks throughout the session. In order to estimate the performance of scheduling algorithms, the most frequent measure that is used is called makespan. The real amount of time that the algorithm required to complete the task is what is being referred to here. In this work, the average makespan is taken into consideration. By calculating the average number of tasks that are not completed before the deadline, the failure rate may be determined. The comparative evaluation is conducted against First Come First

Serve (FCFS), Shortest Job First (SJF), Greedy based Task Scheduler (GTS) and Dynamic Programming based Task Scheduler (DPTS).

Figure 4 provides the comparative evaluation of the proposed multiple-constraint-based greedy method against the average delay measure in varied task load-based scenarios. For this evaluation, multiple experiments are conducted with 100 to 500 tasks and 30 fog devices. The line graph shows that the average delay obtained for FSCS and SJF is are worst. The average delay obtained for all these experiments is 284.12 ms for FCFS, 272.68 ms for SJF, 234.57 ms for GTS and 216.58 ms for DPTS. The proposed algorithm minimized the average delay of 185.15 ms and outperformed all the existing methods.

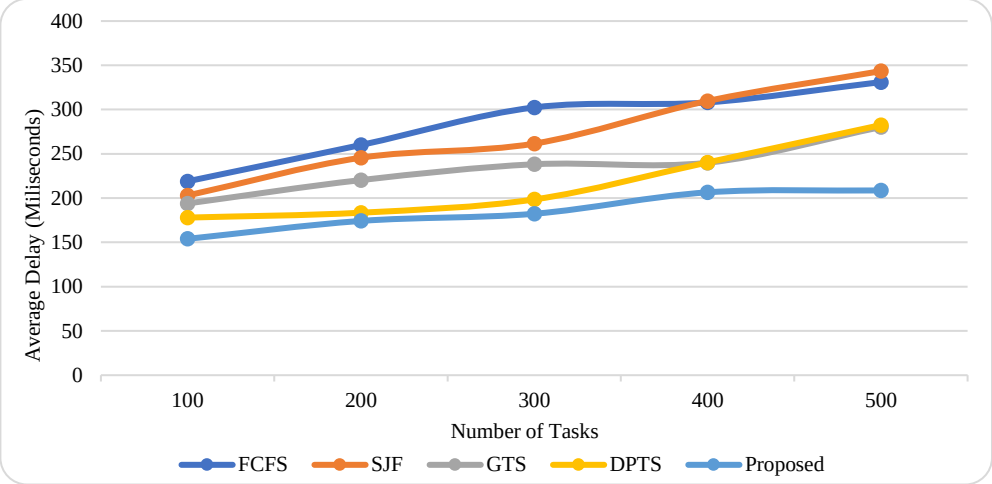

Fig. 4 Average delay analysis (Varied task load)

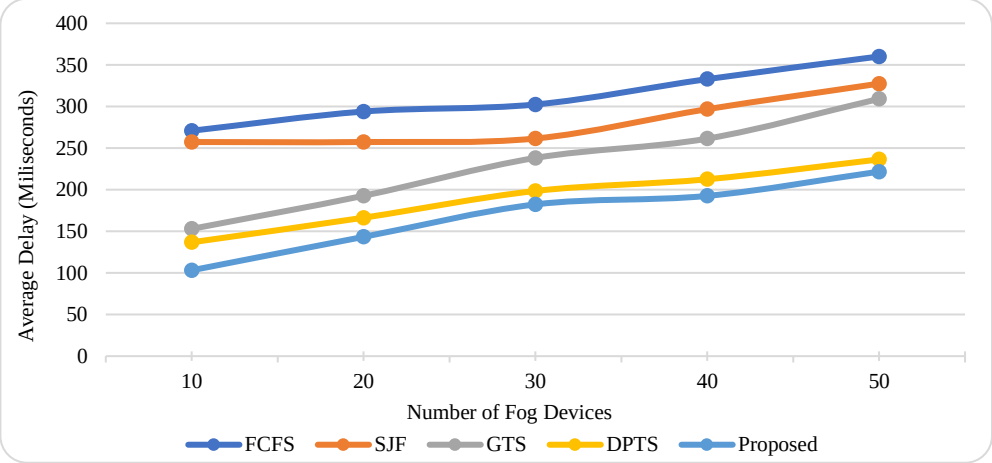

Fig. 5 Average delay analysis (Varied number of fog devices)

Another experiment is conducted in this work against a varied number of fog devices and a fixed number of requests. In this experiment, Multiple tests are carried out for this evaluation, each with 10 to 50 fog devices and 300 requests. Fig 5 illustrates that the average latency for FSCS and SJF is the worst, with a higher average delay. The average delay derived from all of these studies is 312.08 ms for FCFS, 280.12 ms for SJF, 230.91 ms for GTS, and 190.15 ms for DPTS. The suggested strategy reduced the average latency to 168.65 ms and surpassed all existing solutions.

Another parameter considered to validate the reliability of the proposed model is the Successful execution ratio. A task execution is called a failure if the allocated fog device is unable to execute it or the execution is not performed within the specified deadline. An experiment is carried out in this work to validate the reliability of the proposed algorithm under a varied number of requests and a fixed number of fog devices. This experiment involves many tests, each with 100 to 500 requests and 30 fog devices. Figure 6 shows that the successful execution rate of FSCS and SJF is the worst. The average successful execution rate calculated from these trials is 72.31% for FCFS, 77.33% for SJF, 82.16% for GTS, and 86.85% for DPTS. The recommended technique improved the successful execution rate up to 90.47% by outperforming all other alternatives.

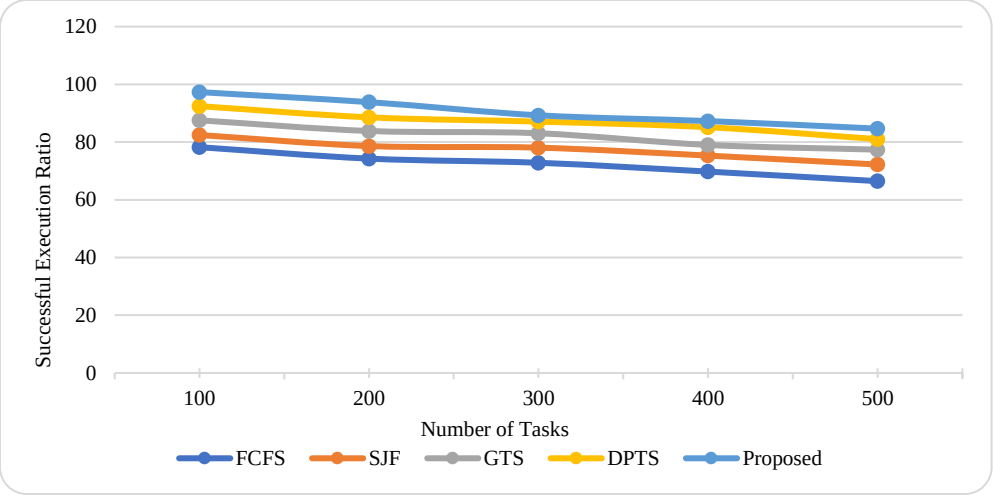

Fig. 6 Successful execution ratio analysis

Another measure examined to assess the suggested model's efficiency and reliability is the makespan. This paper includes an experiment to evaluate the proposed algorithm's dependability under varying numbers of queries and a fixed number of fog devices. This experiment includes many tests, each with 100 to 500 requests and 30 fog devices.

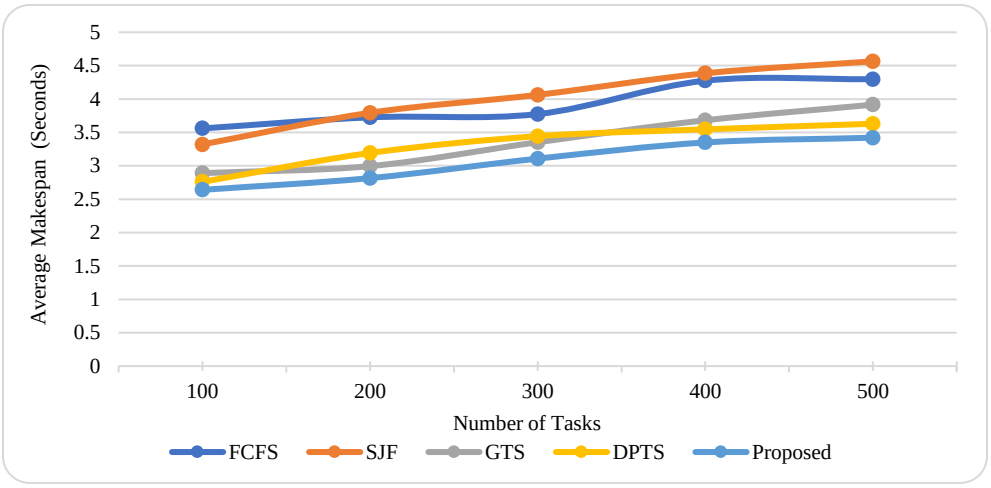

Fig. 7 Average makespan analysis (Varied task load)

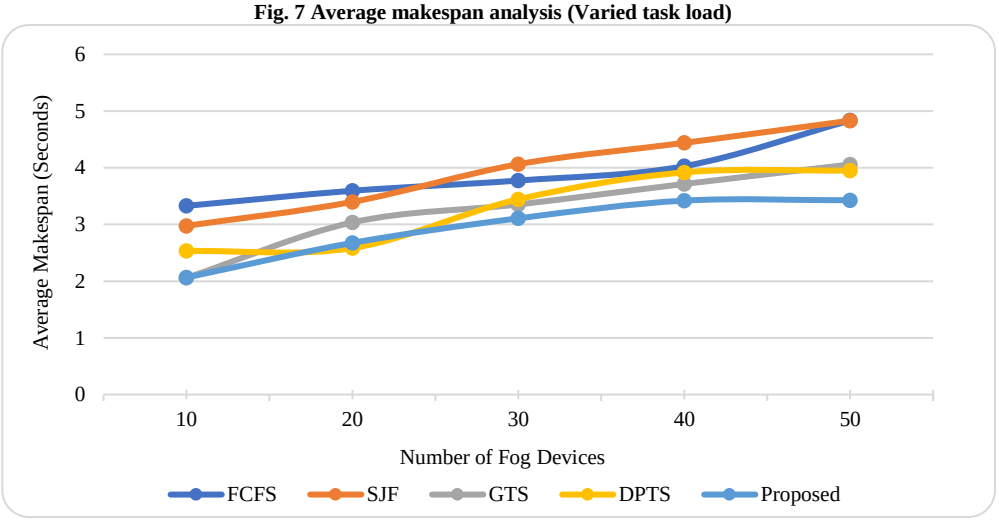

Fig. 8 Average makespan analysis (varied number of fog devices)

According to Fig 7, the FSCS and SJF have recorded the lowest makespan. The average makespan based on these trials is 3.93 sec for FCFS, 4.03 sec for SJF, 3.37 sec for GTS, and 3.31 sec for DPTS. The proposed strategy reduced the makespan by up to 3.07 and improved the efficiency of the proposed algorithm.

Another experiment is carried out in this work with a variable number of fog devices and a set number of requests. This experiment involves many tests, each with 10 to 50 fog devices and 300 requests. Figure 8 shows that the average makespan for FSCS and SJF is the worst, with the longest average makespan. The average makespan calculated from these trials is 3.91 sec for FCFS, 3.94 sec for SJF, 3.24 sec for GTS, and 3.29 sec for DPTS. The recommended technique lowered the average makespan to 2.94 sec, outperforming all other alternatives.

5. Conclusion

This paper investigates the key challenges of fog computing and presents a delay and makespan adaptive task scheduling algorithm. The paper provided a multiple-constraint adaptive greedy algorithm to maximize resource allocation and task scheduling in a heterogeneous fog computing environment. The model's functional process is divided into two components. During the initial step, requests for resource mapping and allocation are conducted based on task criticality and resource priority. In this stage, multiple constraints are defined to perform the allocation. In the final stage, the delay and deadline adaptive greedy algorithm is defined for scheduling the tasks of all fog devices. The model is compared to the FCFS, SJF, GTS, and DPTS algorithms. The assignment is duplicated in six different situations to represent differing load circumstances. The experiments are conducted based on different load conditions. Two scenarios were built to validate the performance in different load conditions. In the first scenario, the number of resources is fixed, i.e. 30, and the number of requests varies from 100 to 500. The mean delay recorded across all trials is 284.12 ms for FCFS, 272.68 ms for SJF, 234.57 ms for GTS, and 216.58 ms for DPTS. The suggested approach reduced the average latency to 185.15 ms and surpassed all current methods. The mean successful execution rates derived from these trials are 72.31% for FCFS, 77.33% for SJF, 82.16% for GTS, and 86.85% for DPTS. The proposed approach enhanced the success rate to 90.47% by surpassing all previous options.

The second scenario is defined with a fixed number of tasks and a varied number of resources. The number of resources varied between 10 and 50. The number of requests in this experiment is 300. The mean delay calculated from these trials is 312.08 ms for FCFS, 280.12 ms for SJF, 230.91 ms for GTS, and 190.15 ms for DPTS. The proposed technique decreased the average latency to 168.65 ms and exceeded all current solutions. The mean successful execution rates derived from these trials are 72.31% for FCFS, 77.33% for SJF, 82.16% for GTS, and 86.85% for DPTS.

The proposed approach enhanced the success rate to 90.47% by surpassing all previous options. The mean makespan from these trials is 3.93 seconds for FCFS, 4.03 seconds for SJF, 3.37 seconds for GTS, and 3.31 seconds for DPTS. The proposed technique decreased the makespan by up to 3.07 and enhanced the efficiency of the proposed algorithm. The results show that the proposed model reduced the makespan and average delay effectively and improved the performance and reliability against existing methods. In the future, the work can be extended by integrating a deep learning model to analyze the history and to make predictive decisions about resource allocation and scheduling.

References

[1] Mohammad Reza Alizadeh et al., "Task Scheduling Approaches in Fog Computing: A Systematic Review," *International Journal of Communication Systems*, vol. 33, no. 16, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[2] Lokman Altin, Haluk Rahmi Topcuoglu, and Fikret Sadik Gürgen, "Network Congestion Aware Multi-Objective Task Scheduling in Heterogeneous Fog Environments," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 2, pp. 3015-3024, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[3] Taybeh Salehnia et al., "An Optimal Task Scheduling Method in IoT-Fog-Cloud Network Using Multi-Objective Moth-Flame Algorithm," *Multimedia Tools and Applications*, vol. 83, no. 12, pp. 34351-34372, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[4] Mehdi Hosseinzadeh et al., "Task Scheduling Mechanisms for Fog Computing: A Systematic Survey," *IEEE Access*, vol. 11, pp. 50994-51017, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[5] Zulfiqar Ali Khan et al., "A Review on Task Scheduling Techniques in Cloud and Fog Computing: Taxonomy, Tools, Open Issues; Challenges, and Future Directions," *IEEE Access*, vol. 11, pp. 143417-143445, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[6] Javid Misirli, and Emiliano Casalicchio, "An Analysis of Methods and Metrics for Task Scheduling in Fog Computing," *Future Internet*, vol. 16, no. 1, pp. 1-22, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[7] Prashanth Choppara, and Sudheer Mangalampalli, "An Effective Analysis on Various Task Scheduling Algorithms in Fog Computing," *EAI Endorsed Transactions on Internet of Things*, vol. 10, pp. 1-7, 2024. [Google Scholar] [Publisher Link]

[8] Nikita Sehgal, Savina Bansal, and RK Bansal, "Task Scheduling in Fog Computing Environment: An Overview," *International Journal of Engineering Technology and Management Sciences*, vol. 7, no. 1, pp. 47-54, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[9] Nikita Sehgal, Savina Bansal, and RK Bansal, "Optimizing Fog Computing Efficiency: Exploring the Role of Heterogeneity in Resource Allocation and Task Scheduling," *International Journal of Computing and Digital Systems*, vol. 15, no. 1, pp. 1119-1133, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[10] Maryssa Trabelsi, and Samir Ben Ahmed, "Energy and Cost-Aware Real-Time Task Scheduling with Deadline-Constraints in Fog Computing Environments," *Proceedings of the 19th International Conference on Evaluation of Novel Approaches to Software Engineering ENASE*, Angers, France, vol. 1, pp. 434-441, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[11] Faten A. Saif et al., "Multi-Objective Grey Wolf Optimizer Algorithm for Task Scheduling in Cloud-Fog Computing," *IEEE Access*, vol. 11, pp. 20635-20646, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[12] Wei-Chang Yeh, Zhenyao Liu, and Kuan-Cheng Tseng, "Bi-Objective Simplified Swarm Optimization for Fog Computing Task Scheduling," *International Journal of Industrial Engineering Computations*, vol. 14, no. 4, pp. 723-748, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] Salman Khan et al., "Optimal Resource Allocation and Task Scheduling in Fog Computing for Internet of Medical Things Applications," *Human-Centric Computing and Information Sciences*, vol. 13, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[14] Zhiming Dai et al., "ME-AWA: A Novel Task Scheduling Approach Based on Weight Vector Adaptive Updating for Fog Computing," *Processes*, vol. 11, no. 4, pp. 1-18, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[15] Ali Garba Jakwa et al., "Performance Evaluation of Hybrid Meta-Heuristics-Based Task Scheduling Algorithm for Energy Efficiency in Fog Computing," *International Journal of Cloud Applications and Computing (IJCAC)*, vol. 13, no. 1, pp. 1-16, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[16] Weimin Liu et al., "Fog Computing Resource-Scheduling Strategy in IoT Based on Artificial Bee Colony Algorithm," *Electronics*, vol. 12, no. 7, pp. 1-24, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[17] Sundas Iftikhar et al., "HunterPlus: AI Based Energy-Efficient Task Scheduling for Cloud–Fog Computing Environments," *Internet of Things*, vol. 21, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[18] Oshin Sharma et al., "Two-Stage Optimal Task Scheduling for Smart Home Environment Using Fog Computing Infrastructures," *Applied Sciences*, vol. 13, no. 5, pp. 1-15, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[19] Media Ali Ibrahim, and Shavan Askar, "An Intelligent Scheduling Strategy in Fog Computing System Based on Multi-Objective Deep Reinforcement Learning Algorithm," *IEEE Access*, vol. 11, pp. 133607-133622, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[20] Mekala Ratna Raju, Sai Krishna Mothku, and Manoj Kumar Somesula, "DRL-based Task Scheduling Scheme in Vehicular Fog Computing: Cooperative and Mobility Aware Approach," *Ad Hoc Networks*, vol. 173, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[21] Prashanth Choppara, and Bommareddy Lokesh, "Efficient Task Scheduling and Load Balancing in Fog Computing for Crucial Healthcare through Deep Reinforcement Learning," *IEEE Access*, vol. 13, pp. 26542-26563, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[22] Xiao-Fei Sui et al., "Multi-Strategy Fusion Mayfly Algorithm on Task Offloading and Scheduling for IoT-Based Fog Computing Multi-Tasks Learning," *Artificial Intelligence Review*, vol. 58, no. 5, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[23] Santhosh Kumar Medishetti, and Ganesh Reddy Karri, "An Improved Dingo Optimization for Resource Aware Scheduling in Cloud Fog Computing Environment," *Majlesi Journal of Electrical Engineering*, vol. 17, no. 3, pp. 31-41, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[24] Hoa Tran-Dang, and Dong-Seong Kim, "Dynamic Collaborative task Offloading for Delay Minimization in the Heterogeneous Fog Computing Systems," *Journal of Communications and Networks*, vol. 25, no. 2, pp. 244-252, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[25] Ali Mohammadzadeh et al., "Energy-Aware Workflow Scheduling in Fog Computing Using A Hybrid Chaotic Algorithm," *The Journal of Supercomputing*, vol. 79, no. 16, pp. 18569-18604, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[26] Zulfiqar Ali Khan, and Izzatdin Abdul Aziz, "Ripple-Induced Whale Optimization Algorithm for Independent Tasks Scheduling on Fog Computing," *IEEE Access*, vol. 12, pp. 65736-65753, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[27] Fengqing Tian et al., "Fog Computing Task Scheduling of Smart Community Based on Hybrid Ant Lion Optimizer," *Symmetry*, vol. 15, no. 12, pp. 1-21, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[28] M. Santhosh Kumar, and Ganesh Reddy Karri, "EEOA: Cost and Energy Efficient Task Scheduling in a Cloud-Fog Framework," *Sensors*, vol. 23, no. 5, pp. 1-20, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[29] Muhammad Saad, Rabia Noor Enam, and Rehan Qureshi, "Optimizing Multi-Objective Task Scheduling in Fog Computing with GA-PSO Algorithm for Big Data Application," *Frontiers in Big Data*, vol. 7, 2024. [CrossRef] [Google Scholar] [Publisher Link]