An Improved Laxity based Cost Efficient Task Scheduling Approach for Cloud-Fog Environment

Praveen Kumar Mishra
Computer Application Department, Engineering College, Ajmer, BTU, Bikaner, India
mishrapraveen2024@gmail.com (corresponding author)

Amit Kumar Chaturvedi
Computer Application Department, Engineering College, Ajmer, BTU, Bikaner, India
amit0581@gmail.com

Received: 2 August 2024 | Revised: 15 October2024 and 20 October 2024 | Accepted: 21 October 2024

Licensed under a CC-BY 4.0 license | Copyright (c) by the authors | DOI: https://doi.org/10.48084/etasr.8595

ABSTRACT

Task scheduling is critical in fog computing, as it has to assign workloads to fog nodes to save costs and execution times. This study emphasizes the allocation of jobs received from clients to suitable nodes through a proposed scheduling technique, which is deployed on layer 2 servers within a cloud-fog environment. Laxity-based Cost-efficient Task Scheduling (LCTS) is proposed for contemporary task scheduling difficulties, such as balancing cost and delay with optimal energy utilization. The results show that the proposed strategy decreased execution time and cost more than Round Robin (RR) and Genetic Algorithm (GA). Furthermore, the proposed method was less expensive than cloud-based IoT solutions. Compared to GA and RR, the simulation results showed that cost and execution time were reduced by 6.99%-17.36% and 4.58%-9.09%, respectively.

Keywords-execution time; cost; energy consumption; scheduling; fog computing

I. INTRODUCTION

A novel system design that attempts to keep things close to the users is an important step in the three-tier structure of a cloud-fog environment. Processing some data locally or in the fog layer rather than sending them to the cloud may decrease latency and bandwidth. Furthermore, a new administration has been put in place to facilitate fog services, such as virtual machine and task scheduling, and many virtual machines running on fog servers can accommodate all user queries. The work is optimally divided among the fog devices and meets the demands of all the users. In this situation, the fog layer enhances the cloud's processing, networking, and storage capacities to accommodate latency-sensitive requests [1]. Job load balancing ensures that no resource is left idle while others are being used. Businesses no longer face as many security threats while utilizing cloud computing to implement big data solutions [2-3]. The devices at the network's edge are connected with distributed computing systems to provide flexible computing, communication, and storage capabilities. Fog layer nodes are a potent addition to cloud computing, not a replacement, making edge processing and cloud interaction possible, and allowing applications and services to run on devices such as routers and gateways [4].

In the layer 2 computing environment, handling resources is a major challenge in terms of processing, latency, storage, and bandwidth. Response times are reduced when the scheduler moves data to high-productivity resources while scheduling computation-intensive processes. Makespan refers to the time needed to complete all jobs. The response time at the requesting interface is the amount of time that passes between a user's request and receiving the response. This study examined a smart car parking system application that helps reduce fuel consumption, pollution, and traffic congestion when looking for parking areas. As a result, locating parking slots for vehicles and waiting for parking saves time. If a specific parking area is full, the collaboration between fog clusters provides information about neighboring free parking slots for the users [5].

In [6], a time-sensitive and energy-aware scheduling application deployment strategy was presented for fogging, leading to QoS-satisfied apps and reduced energy consumption. In [7], it was shown that a variety of responsive applications for the IoT require rapid handling. This method reduced task delay and maximized the work, involving rigorous timelines to process the large amount of local data and minimize overall latency. In [10], a resource-aware scheduler was proposed to cut expenses and allocate incoming apps to fog nodes to optimize resource use. In [11], a scheduling and resource allocation heuristic method was developed to accelerate response and turnaround times, combining a modified analytical approach with a preemption and bandwidth-aware modular ordering method. In [13], cost, makespan, and bandwidth were reduced by creating a location-aware fuzzy-based scheduler for a cloud fog environment. In [15], a three-tier system was developed to optimize response times and costs for applications such as fitness care in smart homes, considering cloud groups with fog layer nodes and control managers. In [16], an ant colony method considered both the priority and duration of a job for scheduling, effectively reducing energy consumption and managing the impact of task delays.

In [17], two distinct theories of linear arithmetic programming were examined on how to reduce execution time, energy usage, etc. In [18], a deadline-constrained assignment planning method was investigated in fog computing, where tasks may be spread across heterogeneous resources. This study also suggested an algorithm that finds better results in polynomial time using an energy-conscious technique. In [21], the proposed scheduling approach justified the load on all available virtual machines while advancing the effectiveness of the best output using a shift adjustment technique and a meta-analysis method. In [22], an approach was proposed to address the minimization problems associated with fog-based work scheduling. The results showed that the proposed strategy outperformed traditional approaches in terms of workload dimensions, resource costs, and overall job computation time. According to [19], cloud-based systems can benefit from fog computing, as the high degree of mobility of fog nodes affects job completion speed. In addition, an end-user fogging system and a resource-sharing mechanism were proposed. In [8], a load-balancing algorithm was proposed at the fog computing layer for smart applications to reduce latency and bandwidth and improve QoS. In [24], the operations and advantages of genetic algorithms on the Internet of Everything were described.

After rigorous study, it was observed that there is a trade-off between delay and cost. Therefore, this study proposes a Laxity-based Cost-effective Task Scheduling (LCTS) approach for assigning tasks to virtual machines at cloud fog nodes with the least cost and make-span. The proposed scheduling method used a multi-objective model and an improved version of the Whale Optimization Approach (WOA). The multi-objective model determines the cost function by summing the costs of each virtual machine's RAM and CPU. The main contribution of this study is the LCTS algorithm. The iFogSim simulator was used to simulate the proposed approach for smart car parking applications. Table I shows the scheduling methods in cloud fog environments, along with previous research methods, advantages, and disadvantages. The key features of this study include:

- The design of a three-layer smart car parking model system.
- Designing a scheduling algorithm to assign jobs to fog nodes.

TABLE I. SUMMARY OF RELATED WORKS

Study	Technique used	Advantages	Limitations/Future work	Objective criteria
[6]	Deadline & energy-aware task scheduling algorithm	Reduced overall system performance such as energy consumption and cost.	Need to investigate the optimal number of servers at the fog layer.	Makespan, power consumption
[7]	Heuristic-based scheduling algorithm (dEDA)	Metrically, the technique performs significantly well. It is a heuristic technique for both the success rate and aggregate tardiness.	It is important to examine how costs could impact other quality service requirements such as lowering service latency.	Deadline, completion time
[8]	Distributing the job while using the fog server.	Reduces latency and improves quality of service.	Need to focus on cost minimization.	Response time, power consumption
[10]	RACE algorithm.	This scheduling model helps minimize execution time and bandwidth in cloud resource uses.	Require to reduce energy and resource use.	Monetary cost, execution time
[12]	MPA algorithm.	Reduces carbon emissions and speeds up flow.	The computing cost of this strategy is higher than that of alternative algorithms.	Flow time, energy consumption
[13]	Location-aware fuzzy-based task allocation algorithm.	Reduces cost and network utilization and application makespan.	Need to minimize operational cost.	Cost, network utilization
[14]	Whale optimization technique	Reduced energy and cost compared to the SJF, RR, and PSO approaches.	Ignored the impact of laxity-based task scheduling on the fog-cloud domain.	Execution time, energy consumption, cost
[15]	Metaheuristic scheduling strategies in edge computing.	Focuses on assigning the task using task assignment algorithm on fog layer to minimize response time.	Requirement to create multi-objective optimization techniques to minimize multiple parameters of scheduling.	Cost, response time
[16]	Ant colony optimization algorithm.	Minimizes energy consumption for all processes at the fog layer.	Needs implementation on a real-world system to schedule the task.	Energy consumption
[17]	Mathematical Programming (MPM) models in fogging	Aimed for the lowest feasible total cost and power consumption.	Needs to optimize multiple parameters, such as task finishing time, energy consumption, etc.	Cost, energy minimization
[18]	Energy-conscious approach with an iterative path method.	Enhance fog node performance with energy-efficient job scheduling based on prioritization.	Hybrid computational resources to reduce energy use need to be solved.	Delay, energy consumption
[19]	Time-effective scheduler with contract-based service exchange.	Fewer resources are required to finish more jobs using fog in the allocated time.	An energy-efficient resource allocation strategy is required.	Makespan, total cost
[20]	Priority-based fair scheduling.	Provided a target platform assessment for the equal, random, and Gaussian distribution scenarios.	Neglected essential elements such as energy consumption.	Execution time, priority
[21]	Metaheuristic approach based on the local search strategy.	IoT devices can provide better user service quality using an energy-conscious metaheuristic approach.	Ignores essential elements such as cost and makespan.	Response time, energy consumption
[22]	A heuristic task scheduling approach	The heuristic method optimization theory provides the global optimal solution to improve parameters.	Does not ascertain the whole job with cost execution time in the fog environment.	Cost, execution time

II. PROPOSED SYSTEM

This section presents the model of a smart vehicle parking system and a laxity-based task scheduling scheme. The three components that make up this system are the cloud, fog, and the end-user layer.

A. Proposed Smart Parking Model

This section describes the three-layer design of a fog-based smart car parking model. The initial component of the proposed model consists of an ultrasonic sensor installed in the parking areas to measure the distance of any object or car present in the parking slot and transmit the vehicle's presence information to IoT-enabled devices at layer 1. Fog Nodes (FN), as well as the Master Fog Server (MFS), are installed at layer 2. This fog layer can process the data collected by IoT devices to identify whether the car parking slot is occupied or unoccupied, and finally relay it to the actuators.

The proposed fog-based system comprises several devices, including sensors, IoT-enabled devices, fog nodes, intelligent actuators, and a Cloud Data Center (CDC). Subsequently, IoT-enabled devices collect data from parking sensors (PS1, PS2, …, PS9) deployed in parking areas and transmit information to the fog layer. The FN receives information from the MFS, processes the request, passes it to the end device layer, and displays it through actuators to show the status of the vacant parking slots. Using IoT-enabled devices, the fog layer receives data from the IoT layer's sensors, which can measure an object's distance in the parking slots. FN has placed closer IoT devices at the edge to ensure that consumers receive fast responses in an instantaneous environment. The results of the empty/occupied car parking slot status are also stored in the data center situated in layer 3. The communication link between the layer 2 nodes and the CDC is established via an application-level gateway. The primary objective of the CDC setup in this framework is to provide massive data centers for storage and processing in heavy traffic loads.

Figure 1 illustrates the design of the proposed three-layer car parking system based on fog computing. Every MFS in the fog layer has implemented this LCTS algorithm. The job deadline is initially estimated using this approach. Next, MFS divides the jobs into many tasks and ensures that the expected completion time is within the deadline. The job is then executed on the closest accessible FN. FNs forward the results to MFS. Finally, MFS forwards combined results to end users. The parking space information is stored and managed by a CDS in layer 3 for extended periods. Users are therefore directed and informed to utilize the parking slot near the parking space entrance to reduce congestion and fuel consumption.

Cloud computing offers more extended processing and administration processes for data. However, frequent transmission and data access over the cloud increase costs, adversely affecting other requests. In contrast to the proposed approach, obtaining actuator information, processing data to locate parking slots, and often transferring data to the cloud are time-consuming operations.

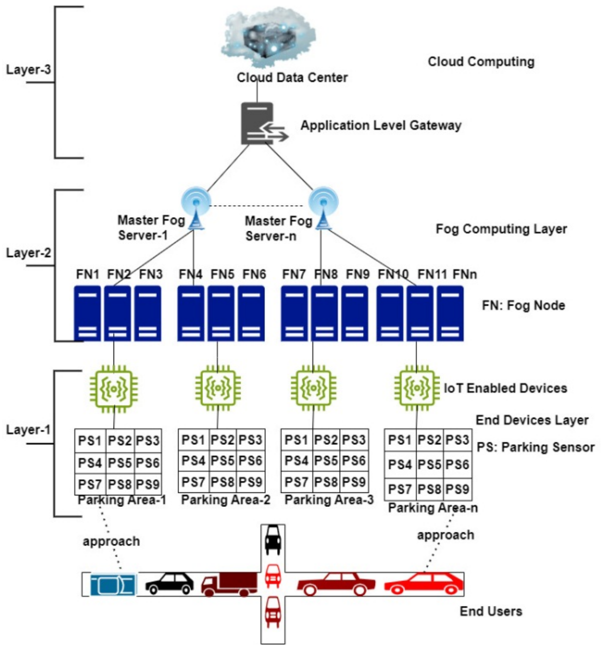

Fig. 1. Three-layer system of smart car parking model.

B. Sensors and IOT Enabled Devices

The initial stage of the intelligent parking system consists of detectors with IoT-enabled devices. Sensors rely on the object's distance data for the parked object to identify vacant parking slots. This study uses the proposed algorithm to minimize cost, energy, and time for executing tasks on fog servers.

C. The Fog Node

Ultrasonic sensors are deployed near IoT-enabled devices to identify vacant parking slots. PS1 denotes a parking slot at layer 1. IoT-enabled devices forward the parking slot vacant/occupied information to the fog computing layer. If the object's length matches the parking spot's size, the area is considered unoccupied, and if not, it means it is a vacant parking slot. The processing node at layer 2 initially sends the results to the end-user tier, that is, layer 1, for a certain amount of time before transferring the information to layer 3, i.e., CDS. The layer 2 infrastructure between the end user and the CDS processes real-time parking slots at the edge, by collecting, handling, and evaluating data.

D. The Cloud Data Center

In the proposed system, proxy servers help fog and cloud communicate with each other. After processing, the fog node sends information about the parking area for storage in the cloud layer. If the processing nodes at layer 2 require additional information, the cloud layer provides it. By connecting layer 2 to the CDC through an application-level gateway, traffic and latency are reduced, and the response to layer 1 improves.

E. Proposed Scheduling Model

This section describes cloud fog LCTS. The proposed scheduling mechanism uses the WOA. The scheduler uses the existing solution to carry out the search procedure. It repeats this procedure until the best answer is found, where $T1, T2, \dots, T_k$ are the first task, second task, and k^{th} task, respectively.

$$Task = \{T1, T2 \dots T_k \dots, T_{500}\} \quad (1)$$

F. Used Energy

The fog layer nodes can be active or idle. The idle mode consumes 50% less power. The total energy consumed by each fog node is:

$$E(FN_j) = \left(Et_j * b_j + (MS - Et_j) * a_j\right) * PS_j \quad (2)$$

where $a_j = 0.5 * b_j$, Et_j is the execution time, b_j is the energy consumed by the active virtual machine, MS is the makespan, and PS is the processing speed of FN [12].

G. Cost

The cost displays the entire cost of the CPU required for fog node machine job scheduling. The variables in (3) are as follows: $MR_{cost}(i)$ is the fog nodes' memory cost, fsc is the iFogSim clock, lut is the used time, cpm shows the number of cycles per millisecond, mpn is milliseconds per node, and lu is the last utilization [9, 14]. M_{base} is the base cost, M_i is the fog node's virtual memory, and t_{ki} is the amount of time required for task T_k to be finished at X_i. MR_{trans} is the transmission cost of the memory, where $MR_{trans} = 0.5$ and $M_{base} =$ 0.05GB/hr.

$$Cost = MR_{cost}(i) + fsc * lut * cpm * lu * mpn \quad (3)$$

$$MR_{cost}(i) = M_{base} * M_i * t_{ki} + MR_{trans} \quad (4)$$

H. Execution Time

Equation 6 determines the entire execution duration (E_{total}) of the algorithm to complete the job.

$$E_{total} = max\{BT_{ti},\ ti \in T\} - min\{ET_{ti},\ ti \in T\} (5)$$

where BT_{ti} and ET_{ti} are the beginning and ending times, respectively, for a specific task ti.

I. Fitness Value Calculation

The fitness value determines the quality of optimal solutions, which must have low energy consumption, cost, and execution time. Combining energy usage, cost, and execution time yields the fitness value as follows:

$$G(\alpha) = E(\alpha) + Cost(\alpha) + E_{total}(\alpha) \quad (6)$$

where $E(\alpha)$ represents energy consumption, $Cost(\alpha)$ denotes cost, and $E_{total}(\alpha)$ represents execution time.

J. Suggested Task scheduling algorithm

The WOA is used to specify the optimal work allocation for fog nodes [23]. This technique begins with the collection of random solutions. The underlying assumption of this process is that the current solution is optimal. This approach is repeated until the best response is discovered.

- X_i $(i = 1, 2, \dots, n)$ represents the initial population, and X^* represents the best search agent.
- Equation 6 represents the fitness value.
- Search agents adjust their positions based on which search agent is now in the best position. This may be shown as follows:

$$\vec{E} = |\vec{Z} * \overrightarrow{X^*}(\alpha) - \vec{X}(\alpha)| \quad (7)$$

$$\vec{X}(\alpha + 1) = \overrightarrow{X^*} - \vec{A} * \vec{E} \quad (if\ \theta < 0.5) \quad (8)$$

where, θ is a random number in [-1, 1], α denotes the present repetition, $\vec{X}$ represents the position vector, $\overrightarrow{X^*}$ represents the position vector of the finest result, and $\vec{Z}$ denotes the coefficient vector.

$$\vec{A} = 2\vec{\beta} * \vec{\gamma} - \vec{\beta} \quad (9)$$

$$\vec{Z} = 2 * \vec{\gamma} \quad (10)$$

where $\vec{\gamma}$ represents the position vector in [0, 1] and $\vec{A}$ represents the coefficient vector.

- Phase of exploitation where the value of $\vec{A}$ is set to [-1, 1].

$$\vec{X}(\alpha + 1) = E' * c^{wt} * \cos(2\Pi t) + \overrightarrow{X^*}(\alpha) \text{ (if } \theta > 0.5) \quad (11)$$

$$E' = |\overrightarrow{X^*}(\alpha) - \vec{X}(\alpha)| \quad (12)$$

where w represents a constant value and t is the value in [-1, 1].

- In the exploration phase, $\vec{X}_{rand}$ represents a randomly chosen agent. The following formula is used to update the search agent's location.

$$\vec{E} = |\vec{Z} * \vec{X}_{rand} - \vec{X}| \quad (13)$$

$$\vec{X}(\alpha + 1) = \vec{X}_{rand} - \vec{A} * \vec{E} \quad (14)$$

K. Laxity of the Task

Higher-priority tasks have less laxity. Task laxity (l_i) is the least amount of time that should be added to a task's deadline. It is related to [16] and used to assess a task's priority.

$$laxity\,(l_i) = \begin{cases} l_{i\ deadline} - \overline{(l_i)} \\ 0, \quad if\ l_i = l_{exit} \end{cases} \quad (15)$$

The proposed LCTS task scheduler is represented in Algorithm 1. The inputs are tasks and fog nodes. The proposed task scheduler optimally distributes tasks among fog nodes. First, the population of search agents is initialized. Then, values are initialized by setting the optimal search agent X^*. Next, the fog layer's resources are retrieved for task allocation using the Available Resource method (AR) approach. This process is repeated until the most suitable solution is found. After that, the sub-method computes cost function, execution time, fitness value, and task laxity. Algorithm 2 checks the closest fog node's resource availability at the fog layer. Before allocating a job for execution, this method verifies that the fog node's RAM, CPU, and storage are available, where VM denotes virtual memory, Fog node Processor (FP), Edge node Processor (EP), Fog devices Random Access Memory (FRAM), Edge devices Random Access Memory (ERAM), Fog device Storage (FS) and Edge Devices Storage (ES). This approach can identify the optimal fog node for task assignment in a cloud fog environment.

```
Algorithm 1: Proposed LCTS Algorithm
  Input: Tasks T, Fog Node FN
  Output: Mapping of Task on Fog Node
  Parameters: X*,  α, θ,  A
Begin
The population of the search agent is
initialized as $X_i$ ($i$ = 1,2,..)
Determine the fitness value by applying
(5) - Initialize the current best search
agent $X*$
Call(Procedure AR(fog node FN, end devices
node e))
If (θ<0.5)
  If(|A|<1)
    Equation 8 updates the search agent
    variable's value.
  Else if(|A|≥1)
    Update the value of the exploration
    Agent variable by (14)
  End if
End if
If($\theta$≥0.5)
  The value of the exploration agent
  variable, is updated by (11)
End if
If(any search agent is outside of the
search area)
  Update $X*$
  α= α+1
End if
End

Submethod:
  Input: Mapping of Task on FN
  Output: Fitness value
Begin
For (every node)
  Calculate energy consumption using (2)
  Calculate the cost function of the CPU
  according to (3) and (4).
  Calculate the Execution time by (5).
  Calculate the fitness value by (6).
  Calculate task laxity ($l_i$) according to
  (15).
End for
End

Algorithm 2: Check processing nodes'
availability
  Input: Fog node, End devices node
  Output: Teturn True or False
Begin
Procedure AR(fog node f, end devices
node e)
If(f.FP≥e.EP && f.FRAM≥e.ERAM && f.FS≥
e.ES) then
  f.FP = f.FP - e.EP
  f.FRAM = f.FRAM - e.ERAM,
  Return 1;
Else
  Return 0;
End
```

III. EXPERIMENTAL SETUP

Distance-measuring sensors determine the car's distance in the parking area. The fog node receives the sensor data once IoT-enabled end devices deliver it. Layer 2 analyses the data to determine the parking slot's condition, provides it to actuators attached through a Wi-Fi connection, and clients are informed via actuators. This study evaluated the cost, energy usage, and execution time of the proposed scheduling approach with the implementation of a smart car parking system. In this experimental condition, four parking spaces were established. Sensors were installed at each parking place, and actuators were used to display information about available parking slots. Fog nodes are connected to layer 3, which has been set up via the application-level gateway. The smart Wi-Fi-enabled sensors are linked through IoT-capable devices. The scenarios generated in the simulation to assess the proposed algorithm outcomes are shown in Figure 2. Sensors are represented as S1, S2, S3…..S8 and actuators are represented as A1, A2, A3….A12. As the results are also shown on client IoT devices, actuators A1, A4, A7, and A10 are placed in the parking area and linked to P_IoT_D1, P_IoT_D2, P_IoT4_D3, and P_IoT_D4 respectively. Actuators 2, 3, 5, 6, 8, 9, 11, and 12 are coupled with client C1 to C8 accordingly.

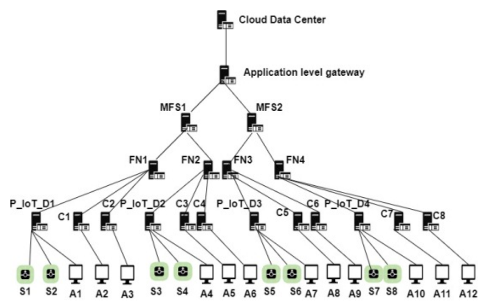

Fig. 2. Topology of fog computing for the proposed algorithm.

Fog nodes are shown as FN1, FN2, FN3, and FN4. Clients are represented as C1, C2….C8, where P_IoT_D means Parking IoT Device. This design has four fog nodes, two master fog nodes, eight sensors, 12 actuators, and one application-level gateway linked to the CDC. Application level gateways connect with the MFS1 and MFS2 layer 2 to CDC layer 3. The module for calculating object distance was built and included in fog nodes to analyze data and calculate vacant parking slots. Furthermore, the linked smart actuators receive and display the parking slots available in the parking areas. Table II presents the parameter values for the proposed algorithm.

TABLE II. PARAMETER VALUES FOR SIMULATION

Parameters	Values
t	-1 , 1
i	1,2….n
θ	[-1, 1]
A	[-1, 1]
Number of tasks	500
Quantity of iterations	100
Size of the population	100
Quantity of executions	500
Number of FNs	4
RAM	4096 MB
BW (Mbps)	250-1500
Processor speed	10,000
Processor	Intel Core i3, 2.53GHz
OS	Windows 7 (64-bit)

IV. RESULTS AND DISCUSSION

This section determines the performance of the proposed fog-based algorithm in execution time, total energy consumed by each virtual machine, and total cost of CPU and memory, and compares these results with those of the RR and GA approaches. Figure 3 presents the proposed method's energy consumption. As can be observed, the energy consumption is lowest when the number of tasks is 400.

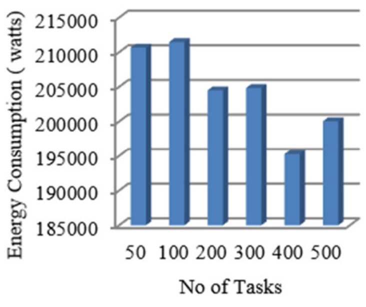

Fig. 3. Energy consumption.

Figure 4 presents the cost of the proposed solution, showing that costs directly increase with the number of tasks.

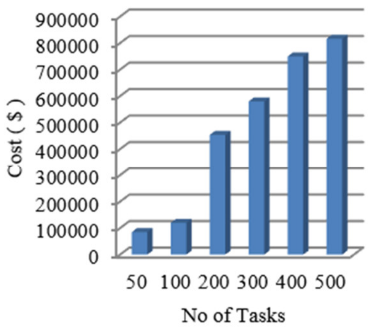

Fig. 4. Cost.

Figure 5 shows the results on execution time in ms. The simulation shows that execution time increases with 50 to 500 tasks.

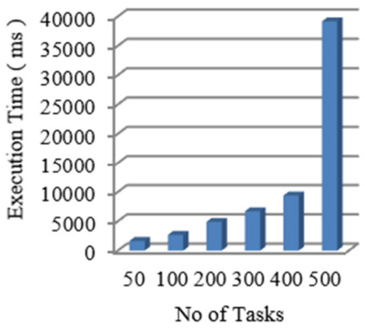

Fig. 5. Execution time.

Figure 6 shows energy consumption comparisons of the proposed solution in fog environments. Energy consumption was minimized compared to RR but was increased compared to GA for different sets of inputs.

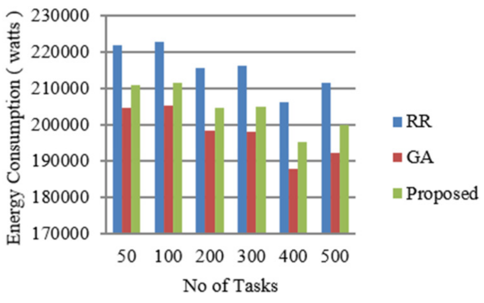

Fig. 6. Comparison of energy consumption.

Figure 7 shows the results of the proposed approach in the fog environment and compares the cost consumption with the existing approaches. When jobs range from 50 to 500, the proposed approach outperforms the RR and GA techniques, being 17.36% less expensive than RR and 6.99% less expensive than GA.

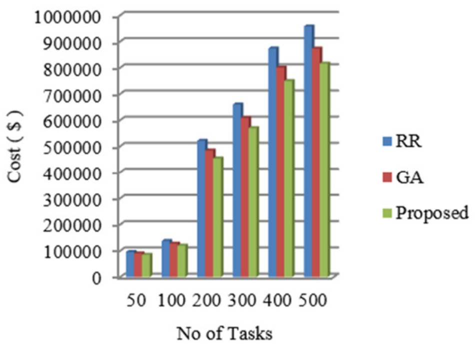

Fig. 7. Cost consumption comparison.

Figure 8 shows the variations in execution times in the fog environment. When tasks ranging from 50 to 500 are given as input, the proposed method yields better results in execution time than the RR and GA algorithms, reducing it by 4.58% compared to GA and 9.09% compared to RR.

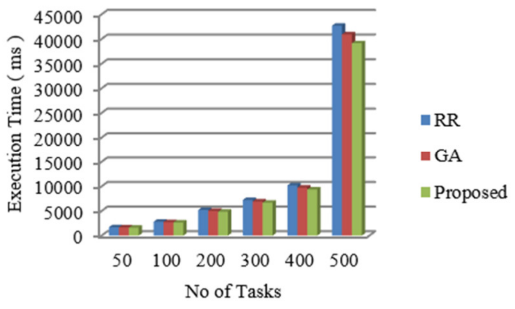

Fig. 8. Execution time comparison.

Table III compares the proposed LCTS method with RR and GA. The execution time for LCTS was 39153, for GA it was 40946.2074, and for RR it was 42712.0077. LCTS costs 816128.81, significantly less than the other procedures, as RR costs 957808.7714 and GA 873176.2138.

TABLE III. COMPARISON BETWEEN RR, GA, AND LCTS

	LCTS	RR	GA
Execution time	39153	42712.0077	40946.2074
Cost	816128.81	957808.7714	873176.2138

V. CONCLUSION AND FUTURE SCOPE

Due to their delayed execution and high cost, cloud-based solutions are unsuitable for time-sensitive applications like smart vehicle parking systems. Processing for fog computing is near the network edge. The limited storage and processing capabilities of fog nodes require job scheduling to reduce cost, execution time, and data processing time without missing deadlines. This study provides a comprehensive, effective multi-objective job scheduling system, called LCTS, that advances fog computing. The proposed method reduces cost and minimizes execution time, outperforming RR and GA for various parameters such as execution time, cost, energy, and task deadline. According to the simulation results, LCTS was 17.36% less expensive than RR and 6.99% less expensive than GA. On the other hand, the execution time was reduced by 9.09% and 4.58% compared to RR and GA, respectively. The projected cost and execution time in a fog environment were determined by running several test cases. For varying input sets, the energy consumption was simultaneously reduced compared to RR and increased compared to GA. The simulated smart parking system yielded the best results with the proposed method. LCTS can also be used with other intelligent real-world applications.

ACKNOWLEDGMENT

The authors acknowledges the Engineering College, Ajmer, for its intellectual support during his studies.

REFERENCES

[1] P. Hosseinioun, M. Kheirabadi, S. R. Kamel Tabbakh, and R. Ghaemi, "aTask scheduling approaches in fog computing: A survey," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 3, 2022, Art. no. e3792, https://doi.org/10.1002/ett.3792.

[2] S. A. Alshaya, "IoT Device Identification and Cybersecurity: Advancements, Challenges, and an LSTM-MLP Solution," *Engineering, Technology & Applied Science Research*, vol. 13, no. 6, pp. 11992–12000, Dec. 2023, https://doi.org/10.48084/etasr.6295.

[3] K. S. Awaisi *et al.*, "Towards a Fog Enabled Efficient Car Parking Architecture," *IEEE Access*, vol. 7, pp. 159100–159111, 2019, https://doi.org/10.1109/ACCESS.2019.2950950.

[4] M. R. Alizadeh, V. Khajehvand, A. M. Rahmani, and E. Akbari, "Task scheduling approaches in fog computing: A systematic review," *International Journal of Communication Systems*, vol. 33, no. 16, 2020, Art. no. e4583, https://doi.org/10.1002/dac.4583.

[5] K. Matrouk and K. Alatoun, "Scheduling Algorithms in Fog Computing: A Survey," *International Journal of Networked and Distributed Computing*, vol. 9, no. 1, pp. 59–74, Jan. 2021, https://doi.org/10.2991/ijndc.k.210111.001.

[6] A. Alwabel and C. K. Swain, "Deadline and Energy-Aware Application Module Placement in Fog-Cloud Systems," *IEEE Access*, vol. 12, pp. 5284–5294, 2024, https://doi.org/10.1109/ACCESS.2024.3350171.

[7] C. Wu and L. Wang, "A Deadline-Aware Estimation of Distribution Algorithm for Resource Scheduling in Fog Computing Systems," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, Wellington, New Zealand, Jun. 2019, pp. 660–666, https://doi.org/10.1109/CEC.2019.8790305.

[8] H. A. Khattak *et al.*, "Utilization and load balancing in fog servers for health applications," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, Apr. 2019, Art. no. 91, https://doi.org/10.1186/s13638-019-1395-3.

[9] D. Rahbari, S. Kabirzadeh, and M. Nickray, "A security aware scheduling in fog computing by hyper heuristic algorithm," in *2017 3rd Iranian Conference on Intelligent Systems and Signal Processing (ICSPIS)*, Shahrood, Iran, Dec. 2017, pp. 87–92, https://doi.org/10.1109/ICSPIS.2017.8311595.

[10] J. U. Arshed and M. Ahmed, "RACE: Resource Aware Cost-Efficient Scheduler for Cloud Fog Environment," *IEEE Access*, vol. 9, pp. 65688–65701, 2021, https://doi.org/10.1109/ACCESS.2021.3068817.

[11] M. B. Gawali and S. K. Shinde, "Task scheduling and resource allocation in cloud computing using a heuristic approach," *Journal of Cloud Computing*, vol. 7, no. 1, Feb. 2018, Art. no. 4, https://doi.org/10.1186/s13677-018-0105-8.

[12] M. Abdel-Basset, N. Moustafa, R. Mohamed, O. M. Elkomy, and M. Abouhawwash, "Multi-Objective Task Scheduling Approach for Fog Computing," *IEEE Access*, vol. 9, pp. 126988–127009, 2021, https://doi.org/10.1109/ACCESS.2021.3111130.

[13] A. Markus, J. D. Dombi, and A. Kertesz, "Location-aware Task Allocation Strategies for IoT-Fog-Cloud Environments," in *2021 29th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, Valladolid, Spain, Mar. 2021, pp. 185–192, https://doi.org/10.1109/PDP52278.2021.00037.

[14] K. P. N. Jayasena and B. S. Thisarasinghe, "Optimized task scheduling on fog computing environment using meta heuristic algorithms," in *2019 IEEE International Conference on Smart Cloud (SmartCloud)*, Tokyo, Japan, Dec. 2019, pp. 53–58, https://doi.org/10.1109/SmartCloud.2019.00019.

[15] H. K. Apat, B. S. Compt, K. Bhaisare, and P. Maiti, "An Optimal Task Scheduling Towards Minimized Cost and Response Time in Fog Computing Infrastructure," in *2019 International Conference on Information Technology (ICIT)*, Bhubaneswar, India, Dec. 2019, pp. 160–165, https://doi.org/10.1109/ICIT48102.2019.00035.

[16] J. Xu, Z. Hao, R. Zhang, and X. Sun, "A Method Based on the Combination of Laxity and Ant Colony System for Cloud-Fog Task Scheduling," *IEEE Access*, vol. 7, pp. 116218–116226, 2019, https://doi.org/10.1109/ACCESS.2019.2936116.

[17] Z. He, Y. Zhang, B. Tak, and L. Peng, "Green Fog Planning for Optimal Internet-of-Thing Task Scheduling," *IEEE Access*, vol. 8, pp. 1224–1234, 2020, https://doi.org/10.1109/ACCESS.2019.2961952.

[18] H. Tan, W. Chen, L. Qin, J. Zhu, and H. Huang, "Energy-aware and Deadline-constrained Task Scheduling in Fog Computing Systems," in *2020 15th International Conference on Computer Science & Education (ICCSE)*, Delft, Netherlands, Aug. 2020, pp. 663–668, https://doi.org/10.1109/ICCSE49874.2020.9201710.

[19] H. Sun, H. Yu, and G. Fan, "Contract-Based Resource Sharing for Time Effective Task Scheduling in Fog-Cloud Environment," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 1040–1053, Jun. 2020, https://doi.org/10.1109/TNSM.2020.2977843.

[20] A. Madej, N. Wang, N. Athanasopoulos, R. Ranjan, and B. Varghese, "Priority-based Fair Scheduling in Edge Computing," in *2020 IEEE 4th International Conference on Fog and Edge Computing (ICFEC)*, Melbourne, Australia, May 2020, pp. 39–48, https://doi.org/10.1109/ICFEC50348.2020.00012.

[21] M. Abdel-Basset, D. El-Shahat, M. Elhoseny, and H. Song, "Energy-Aware Metaheuristic Algorithm for Industrial-Internet-of-Things Task Scheduling Problems in Fog Computing Applications," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12638–12649, Dec. 2021, https://doi.org/10.1109/JIOT.2020.3012617.

[22] M. Yang, H. Ma, S. Wei, Y. Zeng, Y. Chen, and Y. Hu, "A Multi-Objective Task Scheduling Method for Fog Computing in Cyber-Physical-Social Services," *IEEE Access*, vol. 8, pp. 65085–65095, 2020, https://doi.org/10.1109/ACCESS.2020.2983742.

[23] S. Mirjalili and A. Lewis, "The Whale Optimization Algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, May 2016, https://doi.org/10.1016/j.advengsoft.2016.01.008.

[24] N. K. Rathore and S. Pande, "IoE-Based Genetic Algorithms and Their Requisition," in *Computational Intelligent Security in Wireless Communications*, CRC Press, 2023.

AUTHORS PROFILE

Praveen Kumar Mishra is currently attending BTU Bikaner to complete his Ph.D. He earned his B.E. degree from Dr. B. R. A. University in Agra. He graduated from Punjab University's NITTTR Chandigarh with a Master's of Engineering in Computer Science & Engineering. His areas of expertise include image processing and cloud fog computing.

Amit K. Chaturvedi is an Assistant Professor at the Government Engineering College, Ajmer. He holds a Ph.D. from Bhagwant University in Ajmer, Rajasthan, and an M.Sc. from M.P. Bhoj University in Bhopal. He has four chapters in edited books and 65 articles published in reputable journals. He has also organized several seminars and workshops at the national and international levels. He has also contributed significantly to the study of cloud fog computing and ad-hoc networks.