IJICTR
International Journal of Information & Communication Technology Research

Volume 17- Number 4 – 2025 (8-15)

Multi-Stage Heuristic Approach for Resource Allocation in Edge Computing: Enhancing Efficiency and Load Balancing

Hossein Etedadi
Advanced Computer Networks Research Laboratory,
Amirkabir University of Technology
(Tehran Polytechnic)
Tehran, Iran
hossein.eetedadi@aut.ac.ir

Masoud Sabaei *
Department of Computer Engineering
Amirkabir University of Technology
(Tehran polytechnic)
Tehran, Iran
sabaei@aut.ac.ir

Received: 6 November 2024 – Revised: 5 March 2025 - Accepted: 16 April 2025

Abstract—**Resource allocation and task scheduling in edge computing environments are crucial for optimizing overall system performance. This paper introduces a multi-stage heuristic approach that combines local search, Genetic Algorithm (GA), and Particle Swarm Optimization (PSO) to address the complexities of resource management in a three-layer architecture consisting of IoT devices, edge nodes, and cloud servers. The proposed method aims to minimize latency, reduce energy consumption, and maintain load balance by intelligently distributing computational tasks among the available nodes. The multi-objective optimization framework dynamically adapts to changes in workload and network conditions, thereby enhancing the efficiency of the system. Experimental results show that the proposed approach outperforms traditional methods in terms of reducing response time and energy usage while achieving a balanced load distribution across edge nodes, making it an effective solution for real-time and resource-intensive applications in edge computing environments.**

Keywords: Edge Computing, Resource Allocation, Multi-Objective Optimization, Latency

Article type: Research Article

I. INTRODUCTION

With the rapid advancement of technologies such as the Internet of Things (IoT)[1], 5G networks[2], and cloud computing, a massive volume of data is continuously being generated and transmitted. This data requires rapid processing and responsive actions close to its production site, especially in applications such as smart cities[3], digital health, autonomous vehicles[4], smart homes and healthcare[5]. In such environments, traditional cloud computing is not feasible due to its inherent limitations in bandwidth, transfer delay, and the need for extensive communication. In this context, Edge Computing has emerged as a novel approach that shifts data processing closer to the source, enabling reduced latency and improved efficiency. In Edge Computing [6], edge nodes positioned near the data-generating devices process computational tasks locally and only send significant information to cloud servers. This approach not only reduces the burden of data transmission but also decreases processing latency and enhances Quality of Service (QoS). However, Edge

* Corresponding Author

Computing faces several challenges, including optimal allocation of limited resources, energy management, and maintaining load balance among edge nodes.

Resource allocation in edge computing environments involves optimal selection of processing nodes, allocation of computational power and memory, and bandwidth management. In this regard, developing efficient resource allocation algorithms that can dynamically adapt to environmental conditions and workload variations is of paramount importance. So far, various resource allocation methods have been proposed, including classical optimization techniques, metaheuristic algorithms, and machine learning approaches.

The proposed method incorporates a combination of local search algorithms, genetic optimization, Particle Swarm Optimization (PSO), and Tabu Search. In the first stage, local search is used to assign initial tasks to edge nodes. Subsequently, the genetic algorithm refines the initial allocation and generates new solutions. Next, PSO is employed for precise resource allocation tuning, and finally, Tabu Search is used to maintain load balance among the nodes. The proposed model demonstrates its effectiveness in terms of reducing latency, energy consumption, and achieving load balance in a simulated environment and is compared with simpler methods such as Random Allocation, MAS[7], Min-Delay, and Min-Energy. Results indicate that, in comparison with simpler methods, the proposed approach reduces latency by 0.15 seconds, brings energy consumption down to 50 watts, and increases the load balance index to 0.8. These achievements indicate that the proposed heuristic method strikes an appropriate balance among various criteria and outperforms existing approaches.

In Section II, we provide a review of the related works and discuss the various approaches proposed in the literature for resource allocation and task scheduling in edge computing. Section III presents the system model and the proposed methodology in detail, outlining the multi-stage heuristic approach used for optimizing task allocation. In Section IV, the proposed solution is evaluated through experimental analysis, comparing its performance with existing methods. Finally, Section V concludes the paper with a summary of the findings and potential future research directions.

II. RELATED WORKS

Recent studies have explored various methodologies for optimizing resource allocation in edge computing environments, so in this section some studies are discussed. The proposed solution in [7] combines three scheduling models: FCFS, delay-priority, and real-time allocation. The FCFS mechanism schedules tasks based on their arrival order, while the delay-priority mechanism prioritizes tasks according to required delays. The real-time scheduling approach allocates tasks without considering node capacity. The main weaknesses include high computational costs and the inefficiency of deterministic algorithms for large state-space problems. Heuristic approaches are suggested as alternatives for addressing scheduling issues. In [5], a mobility-aware scheduling algorithm is designed for healthcare applications, focusing on reducing delay and response time. The scheduling process is divided into two stages: task ranking based on urgency and allocation using a heuristic method considering patient mobility. The paper uses the Received Signal Strength (RSS) metric for handling mobility, adapting the concept of handover to fog computing nodes. The Weighted Sum Model (WSM) is used to prioritize tasks, but the algorithm does not consider other important factors like resource usage and energy efficiency. Metaheuristic algorithms are suggested to address these limitations. In [8] present a two-tier model that offers a solution to the energy optimization problem through task scheduling to computing nodes. Specifically, this paper attempts to select suitable computing nodes for each task based on the relationship between IoT devices and computing nodes. Despite not considering latency, which is crucial in delay-sensitive applications, an important point in this system architecture is the omission of cloud processing space as a significant resource for applications that do not urgently need rapid processing. The Fireworks Algorithm (FWA) introduced in [9] is used to solve optimization problems by simulating the behavior of fireworks explosions. It correlates fitness values with explosion characteristics to find optimal solutions. In [10], FWA is applied to task scheduling in a three-layer fog computing model. Tasks are categorized based on Euclidean distance to minimize execution time and maintain load balance. The proposed solution shows improved results compared to other methods but has limitations, such as potential single points of failure and lack of consideration for random environmental factors. In [11] propose a three-tier architecture where fog servers are located within the edge network. This paper focuses on energy consumption in edge computing nodes and introduces a meta-heuristic method to reduce energy consumption. One weakness highlighted in this paper is the consideration of only one edge server, whereas in reality, the relationship between edge servers is a crucial evaluation criterion. In [12] propose an Ant Colony Optimization (ACO)-based task offloading algorithm for IoT-based applications in fog computing environments. Their approach formulates the task offloading challenge as a combinatorial optimization problem, aiming to minimize overall system latency by selecting optimal task-node mappings. The algorithm incorporates a dynamic pheromone updating mechanism that adapts to changing network conditions, enhancing responsiveness in real-time scenarios. Simulation results demonstrate that their method effectively reduces task completion time compared to conventional heuristics. However, a key limitation of their work lies in its focus on single-objective optimization, prioritizing only latency.

In contrast, the proposed multi-stage heuristic method in this study integrates Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Tabu Search to jointly optimize latency, energy consumption, and load balancing. This multi-objective formulation allows for a more balanced and practical solution in real-world edge computing scenarios, where trade-offs between performance, efficiency, and resource utilization must be carefully managed. As such, the proposed method addresses a broader range of system

requirements and operational constraints than the ACO-based approach.

III. PROPOSED ALGORITHM

In this section, a heuristic-based optimization method is introduced for resource allocation in edge computing environments. This method leverages metaheuristic algorithms such as Genetic Algorithm, Particle Swarm Optimization (PSO), and local search algorithms to perform an optimal search within the decision space. In this paper, a multi-stage heuristic method for optimal resource allocation in edge computing is presented. The novelty of the proposed approach lies not in the mere application of existing heuristic algorithms, but in the strategic integration and orchestration of multiple complementary optimization techniques—Hill Climbing, Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Tabu Search—into a unified, multi-stage scheduling framework tailored specifically for edge computing environments. Each algorithm was carefully selected based on its inherent strengths in addressing different sub-problems within the task allocation pipeline. Other heuristic algorithms such as Ant Colony Optimization (ACO), Simulated Annealing (SA), or Firefly Algorithm were considered; however, empirical and literature-based evaluations showed that they either lack convergence speed in high-dimensional search spaces (e.g., ACO), or are prone to premature convergence without hybrid reinforcement (e.g., SA). Moreover, PSO and GA are widely acknowledged for their robustness, scalability, and adaptability in handling discrete task scheduling problems, making them well-suited to the dynamic and heterogeneous nature of edge computing. Therefore, the novelty of this work is not only in choosing well-known algorithms but in how they are sequenced, tuned, and adapted within a multi-objective context for real-time, energy-aware, and balanced resource scheduling—a contribution that is distinctly different from mono-objective or single-layer optimization approaches previously explored in the literature.

In this model, resource allocation is conducted in a decentralized and distributed manner, with the primary objectives being minimizing latency, reducing energy consumption, and maintaining load balance among edge nodes and central servers. As shown in Fig.1, the proposed system model in this paper consists of three layers: IoT devices, edge computing nodes, and cloud computing. The IoT device layer comprises all devices that generate computational tasks, such as sensors, cameras, and other IoT devices. These devices send their tasks to edge nodes for initial processing. The edge computing layer includes nodes that are positioned close to the IoT devices and possess higher computational power compared to them. The primary objective of this layer is to process tasks and temporarily store data to reduce the traffic sent to the cloud layer. If the resources in this layer are insufficient, the tasks can be offloaded to cloud servers. The cloud computing layer consists of data centers with significantly higher computational power. This layer is generally used for processing more complex tasks that require substantial computational capabilities.

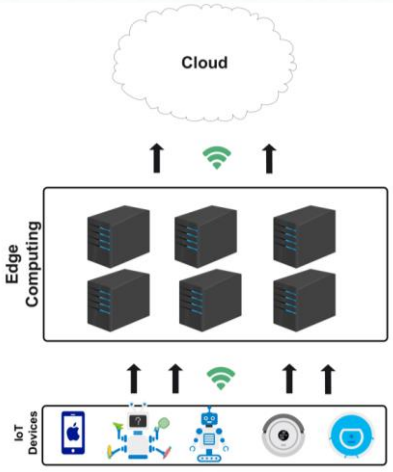

Figure 1. System Model

After providing an overview of the system model, we will delve into the specific mathematical formulations and relationships within each layer to provide a more detailed analysis. The proposed resource allocation model is formulated as a multi-objective optimization problem, targeting various objectives such as minimizing latency, reducing energy consumption, and maintaining load balance among the nodes.

To achieve these goals, the resource allocation problem is defined mathematically, where decision variables indicate the assignment of tasks to either edge nodes or cloud servers. In this system model, tasks are represented as $T = \{t_1, t_2, \dots, t_n\}$, the set of edge computing nodes is represented as $E = \{e_1, e_2, \dots, e_m\}$, and the set of cloud servers is denoted as $C = \{c_1, c_2, \dots, c_n\}$.

A. *Initial Task Allocation Using the "Hill Climbing" Algorithm*

In this stage, initial task allocation to the edge nodes is performed using a local search algorithm, such as the "Hill Climbing" algorithm. The objective of this stage is to find a suitable initial solution for task allocation based on the processing capacities and network latency between nodes. The Hill Climbing algorithm iteratively searches for a better solution by making small changes to the current allocation and accepting the changes if they improve the objective function. Let *T* be the set of tasks and *E* be the set of edge nodes. The initial allocation is conducted in a way that each task is assigned to the nearest edge node in terms of network latency and computational capacity. The Hill Climbing algorithm begins by randomly assigning tasks to the available edge nodes. It then explores neighboring solutions by moving tasks between nodes or swapping tasks and calculates the objective function for each new configuration. The goal is to reduce the total network delay and ensure that the computational capacities of the nodes are not exceeded.

This process continues until no further improvement in the objective function can be found, resulting in an initial task allocation that serves as the starting point for further optimization in the next stages. Objective function is shown as

$$\boldsymbol{min} \sum_{i=1}^{n} \sum_{j=1}^{m} \boldsymbol{x_{ij}}.(\boldsymbol{D_{ij}} + \boldsymbol{\lambda}.\boldsymbol{P_{ij}}) \quad (1)$$

$$\boldsymbol{subject\ to}:$$

$$\sum_{j=1}^{m} \boldsymbol{x_{ij}} = \boldsymbol{1} \quad \forall\, \boldsymbol{i} = \boldsymbol{1}, \boldsymbol{2}, \dots, \boldsymbol{n} \quad (a)$$

$$\boldsymbol{x_{ij}} \in \{\boldsymbol{0}, \boldsymbol{1}\}, \quad \forall \boldsymbol{i}, \boldsymbol{j} \quad (b)$$

In Eq(1) x_{ij} is a binary decision variable indicating the allocation of task T_i to edge node E_j. If T_i is allocated to E_j then $x_{ij} = 1$; otherwise, $x_{ij} = 0$. D_{ij} represents the delay associated with transferring task T_i to edge node E_j. P_{ij} denotes the processing power required to execute task T_i on edge node E_j. λ is balancing coefficient adjusting the trade-off between latency and computational resources. The local search algorithm aims to minimize the following objective function. At each step, the allocation of tasks to the nodes is modified, and the value of the objective function is evaluated. If the new allocation results in a lower objective value, it is selected as the optimal allocation. This process continues iteratively until no further improvements can be made.

B. *Allocation Optimization Using the Genetic Algorithm*

After the initial allocation, the Genetic Algorithm is used for further optimization of resource allocation. At this stage, chromosomes represent the allocation of tasks to nodes, and genetic operators such as mutation and crossover are employed to generate new generations. The fitness function at this stage is defined based on minimizing latency and energy consumption. After determining the initial allocation, the Genetic Algorithm is used to improve resource allocation. At this stage, each chromosome represents a possible allocation of tasks to edge nodes and cloud servers. The structure of the chromosome is $Choromosome = [x_{11}, x_{12}, \dots, x_{nm}]$. As all knows, the Genetic Algorithm employs two main operators: crossover and mutation. In the crossover operation, two chromosomes are randomly selected, and parts of these chromosomes are exchanged to generate new offspring. The mutation operation involves randomly altering the allocation of one or more tasks, which may include reassigning a task from one node to another or shifting it from the edge layer to the cloud layer. The fitness function is defined based on these changes to evaluate and optimize the overall allocation. The fitness function is defined as:

$$Fitness = \frac{1}{\alpha \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij}.D_{ij} + \beta \sum_{j=1}^{m} E_j + \gamma\ unbalanced\ load} \quad (2)$$

In Eq(2), E_j is the energy consumption of edge node E_j for processing the allocated tasks. unbalanced load is the load balance index among the nodes, defined as the difference in workload between nodes. α,β,γ are weight coefficients that determine the relative importance of latency, energy consumption, and load balance, respectively.

C. *Precise Optimization Using Particle Swarm Optimization (PSO)*

In the third stage, precise optimization of resource allocation is performed using the Particle Swarm Optimization (PSO) algorithm. This algorithm fine-tunes the allocation of resources by simulating the social behavior of particles (solutions) as they explore the search space. Each particle adjusts its position based on its own experience and the experience of neighboring particles, ultimately converging towards an optimal or near-optimal solution for minimizing latency, reducing energy consumption, and achieving load balance across nodes.
In this method, the particle update is performed according to the following equation:

$$v_i^{t+1} = w.v_i^t + c_1.r_1.\left(P_i^{best} - x_i^t\right) + c_2.r_2.(g^{best} - x_i^t) \quad (3)$$

Where v_i^{t+1} represents the velocity of particle i at iteration t+1, www is the inertia weight, and c_1 and c_2 are learning coefficients. The variables r_1 and r_2 are random values in the range [0, 1]. Additionally, P_i^{best} denotes the best local position (Local Best) of particle i, while g^{best} refers to the best global position (Global Best) found by the swarm. Accordingly, the position update of the particle can be calculated using the following equation:

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (4)$$

Particle positions are discretized to binary allocations in task-node assignments for practical implementation.

D. *Load Balancing Among Nodes Using the Tabu Search Method*

In this stage, load balancing among the nodes is achieved using the Tabu Search algorithm. Tabu Search is a metaheuristic optimization method designed to explore the solution space beyond local optimal points by using a memory structure known as the "Tabu List." This list stores recently visited solutions or specific attributes of those solutions, preventing the algorithm from revisiting them and thus avoiding cycles and local traps. The main goal of this stage is to achieve a balanced load distribution across all edge nodes by shifting tasks between nodes based on their current load and capacity. The algorithm searches the neighborhood of the current solution by

reallocating tasks from overloaded nodes to underutilized ones. Each candidate solution is evaluated based on a load balancing objective function, and the best solution that is not in the Tabu List is selected for the next iteration. The evaluation function at this stage is defined as follows:

$$Load\ balancing\ score = \sum_{j=1}^{m} \left|1 - \frac{L_j}{L^-}\right| \quad (5)$$

where $L_j$represents the current load of edge node E_j, and L^- denotes the average load across all edge nodes.

$$L^- = \frac{1}{m}\sum_{j=1}^{m} L_j \quad (6)$$

A lower value of the Load Balancing Score indicates a more balanced distribution of tasks among the nodes. The objective of the Tabu Search algorithm is to minimize this score, thus ensuring an optimal load balance throughout the system and preventing any single node from becoming overloaded.

IV. EVALUATION

In this section, the performance of the proposed method is compared with three conventional methods: Random Allocation, Min-Delay Allocation, and Min-Energy Allocation. The evaluation criteria include Delay, Energy Consumption, and Load Balance, which respectively indicate the system's efficiency in reducing latency, optimizing energy consumption, and maintaining load balance across the edge nodes. The simulations were conducted on a system equipped with an Intel Core i5-11400H CPU, 16 GB of RAM, and an NVIDIA GeForce RTX 3050 GPU. This configuration ensured sufficient computational resources for running the simulation scenarios efficiently. Table 1 demonstrate the detail of experimental evaluation.

TABLE I. DETAIL OF EVALUATION

Experimental Variable	Description	Values/Range
Number of Tasks	Total number of tasks allocated to edge nodes	50, 100, 150, 200
Number of Edge Nodes	Number of edge computing nodes used for task allocation	10, 20, 30
Task Arrival Rate	Frequency of task arrival at the edge nodes	Low, Medium, High
Processing Capacity	Computational power of each edge node (in GHz)	1.5 GHz, 2.0 GHz, 2.5 GHz
Energy Consumption	Power consumed by each edge node for processing tasks (in Watts)	40W, 50W, 60W
Network Latency	Delay experienced during data transfer between nodes (in seconds)	0.15s, 0.25s, 0.35s

In the conducted experiments, the proposed method used a balanced configuration of α=0.5, β=0.3, and γ=0.2, prioritizing latency while maintaining good energy performance and load distribution. This balance is reflected in the results which will be explained deeply. We acknowledge the importance of validating the proposed algorithm against real-world datasets or standardized benchmarks to assess its practical applicability and generalizability. In this study, we focused on a simulated edge computing environment where the parameters such as task sizes, node capacities, energy levels, and network delays were modeled based on ranges and distributions observed in existing literature. While our current evaluation demonstrates the effectiveness of the proposed multi-stage heuristic algorithm in a controlled simulation, we recognize the value of benchmarking against widely accepted datasets.

A. Latancy

Fig. 2 illustrates the comparative performance of different resource allocation methods in terms of total task execution time for 150 tasks. This metric is crucial for evaluating scheduling efficiency, especially in latency-sensitive edge computing applications. The chart clearly highlights the superiority of the Proposed Method, which completes all tasks in just 22.5 seconds. This substantial performance gain results from its carefully designed multi-stage heuristic strategy, integrating Hill Climbing, Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Tabu Search to optimize both initial allocation and fine-tuning. Among the alternatives, MAS [7], a mobility-aware scheduling method for IoT-based environments, records a total execution time of 46.8 seconds. While MAS [7] addresses mobility considerations and reduces task delays relative to random strategies, it does not incorporate energy efficiency or load balancing into its optimization, limiting its adaptability in broader edge contexts. Min-Delay Allocation, focused solely on minimizing delay without considering system-wide balance, completes execution in 42.0 seconds, nearly 87% longer than the proposed method. The Random Allocation strategy performs worse, requiring 52.5 seconds due to its lack of intelligent task placement, which causes inefficient resource use and congestion. Finally, the Min-Energy Allocation approach, which prioritizes power savings, takes the longest at 60.0 seconds, clearly reflecting the trade-off between energy efficiency and execution speed.

Overall, the latency chart clearly demonstrates the effectiveness of the proposed approach in minimizing delays and enhancing the responsiveness of edge computing systems, making it an ideal solution for latency-sensitive applications.

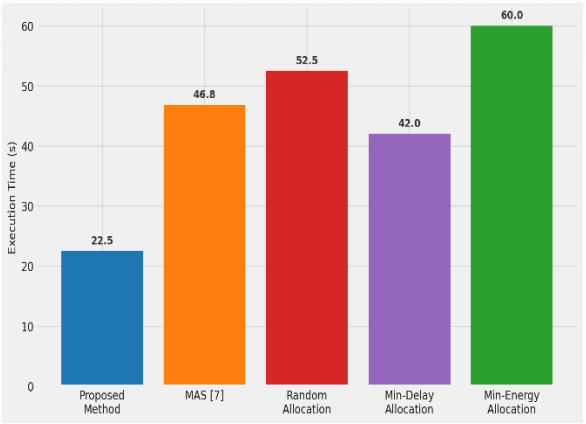

Figure 2. Delay Evaluation

B. *Energy Consumption*

Fig. 3 compares the efficiency of different resource allocation methods in terms of the amount of energy required to process tasks in edge nodes. Energy efficiency is a core concern in distributed computing, particularly at the network edge where devices often operate on limited power budgets. The energy consumption plot reveals the efficiency spectrum of each method. Min-Energy Allocation, as expected, achieves the lowest power draw at 40W, aligning with its single-objective goal of reducing energy expenditure. However, this comes at the cost of a substantial increase in execution time (60.0s), which could render it unsuitable for real-time applications. In contrast, the Proposed Method consumes a modest 50W, only 10W more than Min-Energy, but completes execution almost three times faster. This balance between performance and energy usage showcases the practicality of the proposed solution in edge computing, where both power and responsiveness must be optimized simultaneously. The Random Allocation approach is the least energy-efficient, consuming 80W. Its lack of decision-making logic leads to inefficient node usage, increased idle power consumption, and frequent task migration. Min-Delay Allocation, while more targeted, still consumes 70W, indicating that focusing solely on minimizing latency can also lead to energy waste if not carefully controlled.

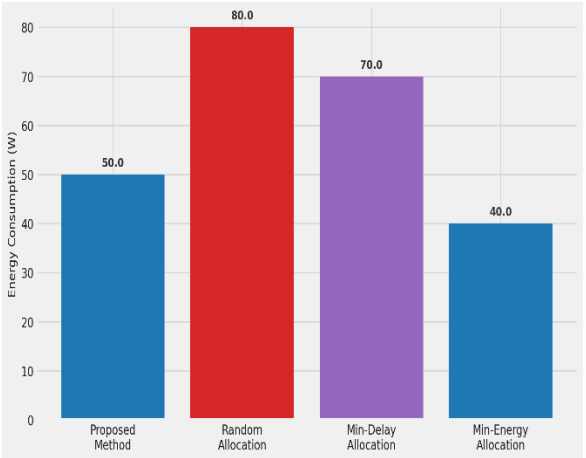

Figure 3. Energy Consumption evaluation

In summary, the proposed method presents a highly effective trade-off: low energy consumption with high performance, making it a compelling choice for scenarios where both metrics are mission-critical, such as wearable health monitors or real-time surveillance systems.

C. *Load Balancing*

Fig. 4 illustrates how effectively different resource allocation methods distribute tasks among available edge nodes. Load balancing is a pivotal factor in ensuring long-term system stability and resource fairness. Uneven task distribution can lead to overloaded nodes, increased failure rates, and delayed response times—especially detrimental in dynamic edge environments. The load balancing plot illustrates how each method distributes the system workload.

The Proposed Method achieves a Load Balance Index of 0.80, the highest among all compared strategies. This value reflects a near-uniform distribution of computational loads across all edge nodes, indicating effective coordination between its multiple optimization stages. The use of Tabu Search in the final step allows it to escape local optima and find globally efficient allocations, contributing significantly to this result. In contrast, Random Allocation, which lacks any balancing logic, has a low index of 0.50, showing considerable imbalance and potential node overloading. Min-Delay Allocation, although focused on reducing latency, improves slightly to 0.60 but still suffers from bias in node selection, often overloading faster nodes while underutilizing others. The Min-Energy Allocation method performs relatively well, achieving an index of 0.70. This suggests that while it doesn't directly optimize for load balancing, its energy-saving logic indirectly avoids concentration of tasks on a few powerful nodes. Nevertheless, it still falls short of the proposed method, which explicitly includes load balancing as an optimization goal.

Ultimately, the high load balance index of the proposed method not only improves responsiveness but also extends the lifespan of the edge network by preventing localized overheating and hardware degradation. This makes it an excellent fit for scalable, sustainable, and fault-tolerant edge infrastructures.

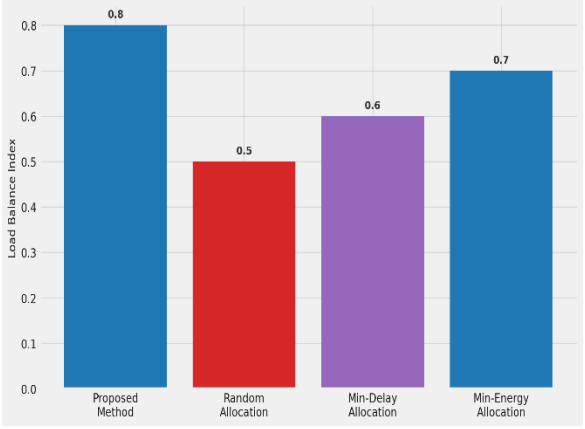

Figure 4. Load Balancing Evaluation

TABLE II. GENERAL COMPARISON

Method	Latency(s)	Energy(W)	Load Balance
Proposed method	22.5	50	0.8
MAS[7]	46.8	N/A	Qualitative
Random	52.5	80	0.5
Min-Delay	42	70	0.6
Min-Energy	60	40	0.7

By achieving a higher load balance index, the proposed method ensures that no single node becomes a bottleneck, thus enhancing resource utilization and maintaining optimal performance across the entire network. Table 2 presents a comparative analysis of five task scheduling methods based on three key metrics: total execution time, energy consumption, and load balance index. To ensure fairness and consistency, all latency figures have been normalized as total execution time for 150 tasks—aligned with the evaluation format used in MAS [7].

The Proposed Method achieves the most balanced performance, completing all tasks in just 22.5 seconds, which is significantly faster than MAS [7] (46.8s), Random (52.5s), Min-Delay (42.0s), and Min-Energy (60.0s). This demonstrates its strength in low-latency task offloading suitable for real-time edge computing scenarios.

In terms of energy efficiency, the Min-Energy approach leads with the lowest energy usage (40W), as expected. However, the proposed method maintains a strong balance, consuming only 50W—notably lower than Random and Min-Delay—while achieving much faster execution time.

Regarding load balancing, the proposed method again outperforms others with a Load Balance Index of 0.80, indicating effective and fair distribution of workloads across edge nodes. This is higher than Random (0.50), Min-Delay (0.60), and even the specialized Min-Energy approach (0.70). MAS [7] provides only a qualitative assessment of load, lacking measurable data.

In summary, the proposed multi-stage heuristic method offers a compelling trade-off across all three critical dimensions—latency, energy, and resource balance—surpassing both traditional and state-of-the-art alternatives in overall effectiveness.

D. Computational Complexity

The computational complexity of the proposed multi-stage heuristic algorithm is derived by analyzing each of its constituent phases: Hill Climbing, Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Tabu Search. Let n represent the number of tasks, m the number of edge nodes, and g the number of iterations or generations in the metaheuristic phases. The initial task allocation is performed using a Hill Climbing approach, which iteratively explores neighboring solutions by modifying task-to-node assignments. This phase incurs a complexity of $O(n \cdot m)$. The GA phase operates over a population of size p for g generations, with each chromosome evaluated across n tasks and m nodes, resulting in a complexity of $O(g \cdot p \cdot n \cdot m)$. Subsequently, PSO refines the solution using a swarm of s particles over g iterations, where each particle update and evaluation also require $O(n \cdot m)$ operations, leading to $O(g \cdot s \cdot n \cdot m)$. Finally, Tabu Search adjusts the task distribution for load balancing, with each of the g iterations considering n possible reassignments, resulting in $O(g \cdot n)$ complexity.

Implementing a multi-objective optimization framework in edge computing environments involves several practical challenges, particularly when balancing conflicting objectives such as latency and energy consumption. One major challenge lies in the heterogeneity of edge devices, which often differ in computational power, energy profiles, and network latency. Designing an algorithm that generalizes well across such variability requires careful abstraction of system parameters and real-time adaptability. Additionally, implementing multi-stage heuristics like Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) in distributed settings imposes computational overhead and may require parallelization or lightweight adaptations to remain feasible for resource-constrained edge nodes. In edge computing environments, achieving optimal performance often requires navigating inherent trade-offs among latency, energy efficiency, and load balancing. These three objectives are interdependent and, in many cases, conflicting. For instance, minimizing latency typically involves assigning tasks to the most powerful or closest nodes, which may lead to node overloading and uneven resource utilization, thereby degrading overall system stability. Conversely, balancing load across all available nodes may require offloading tasks to more distant or less efficient devices, inadvertently increasing communication delay and processing time. Similarly, aggressively minimizing energy consumption—by favoring low-power nodes or throttling computational speed—can delay task completion and concentrate tasks on specific nodes, negatively affecting both latency and load distribution.

The proposed multi-stage heuristic method effectively manages these trade-offs by integrating Hill Climbing, Genetic Algorithm, PSO, and Tabu Search in a synergistic framework, each stage focusing on optimizing one or more objectives. More importantly, the method incorporates a weighted fitness function that allows fine-grained control over the importance of each objective via the coefficients α, β, and γ. Through experimental calibration (e.g., giving higher priority to α for real-time applications), the algorithm dynamically adjusts its decision-making process, seeking a solution that minimizes delay without significantly compromising energy usage or load distribution. The final outcomes demonstrate that it is possible to maintain low latency (22.5s total for 150 tasks) while keeping energy consumption moderate (50W) and achieving superior load balance (index =

0.80). This indicates that, although the objectives are in tension, they need not be mutually exclusive. With a carefully designed optimization strategy, a high-performing equilibrium can be achieved, making the system both responsive, efficient, and scalable.

V. CONCLUSION

In this paper, a multi-stage heuristic approach was proposed for optimal resource allocation in edge computing environments. The method integrated multiple heuristic techniques, including local search, Genetic Algorithm, Particle Swarm Optimization (PSO), and Tabu Search, to minimize latency, reduce energy consumption, and maintain load balance across nodes. The proposed method was evaluated against conventional approaches such as Random Allocation, Min-Delay, and Min-Energy. The results demonstrated that the proposed method significantly outperformed the conventional methods by achieving lower latency, balanced energy consumption, and improved load distribution. These findings indicate that the proposed multi-stage heuristic approach provides an effective solution for complex resource allocation problems in edge computing, offering a balanced trade-off among different performance metrics. Also as part of our future work, we plan to incorporate real-world datasets such as those from edge computing testbeds (e.g., EdgeDroid, Grid5000) or open benchmarks like Google Cluster Data and Alibaba Cluster Trace, to further validate and fine-tune our approach under realistic workloads and system conditions.

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] G. Sreekanth, S. A. N. Ahmed, M. Sarac, I. Strumberger, N. Bacanin, and M. Zivkovic, "Mobile Fog Computing by Using SDN/NFV on 5G Edge Nodes," *Comput. Syst. Sci. Eng.*, vol. 41, no. 2, pp. 751–765, 2022.

[3] O. Sharma, G. Rathee, C. A. Kerrache, and J. Herrera-Tapia, "Two-stage optimal task scheduling for smart home environment using fog computing infrastructures," *Applied Sciences*, vol. 13, no. 5, p. 2939, 2023.

[4] B. Ji *et al.*, "Survey on the internet of vehicles: Network architectures and applications," *IEEE Communications Standards Magazine*, vol. 4, no. 1, pp. 34–41, 2020.

[5] R. M. Abdelmoneem, A. Benslimane, and E. Shaaban, "Mobility-aware task scheduling in cloud-Fog IoT-based healthcare architectures," *Computer networks*, vol. 179, p. 107348, 2020.

[6] B. Jamil, H. Ijaz, M. Shojafar, K. Munir, and R. Buyya, "Resource allocation and task scheduling in fog computing and internet of everything environments: A taxonomy, review, and future directions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 11s, pp. 1–38, 2022.

[7] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 26–35, 2017.

[8] X. Huang, W. Fan, Q. Chen, and J. Zhang, "Energy-efficient resource allocation in fog computing networks with the candidate mechanism," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8502–8512, 2020.

[9] Y. Tan and Y. Zhu, "Fireworks algorithm for optimization," in *International conference in swarm intelligence*, 2010: Springer, pp. 355–364.

[10] S. Wang, T. Zhao, and S. Pang, "Task scheduling algorithm based on improved firework algorithm in fog computing," *IEEE Access*, vol. 8, pp. 32385–32394, 2020.

[11] 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, 2020.

[12] M. K. Hussein and M. H. Mousa, "Efficient task offloading for IoT-based applications in fog computing using ant colony optimization," *IEEe Access*, vol. 8, pp. 37191–37201, 2020.

Hossein Etedadi received the B.E. degree from Guilan University, Guilan, Iran, in 2021 and M.Sc. in computer networks from Amirkabir University of Technology (Tehran Polytechnic), Iran, in 2024. His research interests are cluster computing, the Internet of Things, wireless networks and beyond-5G and 6G networks.

Masoud Sabaei is an Associate Professor with the Computer Engineering department at Amirkabir University of Technology, in Tehran, Iran. Dr. Sabaei received his Ph.D. in computer engineering from Amirkabir University of Technology, Tehran, Iran, in 2000. His research interests are wireless networks and software-defined networks.