Distributed Edge Computing For Resource Allocation In Mmtc Using Non-Linear Stochastic Optimization

Kezia Rani Burgula[1], Pallavi Sagar Deshpande[2], V.Bhoopathy[3], Keerthana[4], Kusuma Rajasekhar[5], Dhivya Ramasamy[6]

[1]Associate Professor, Department of Information Technology, Vasavi College of Engineering, Hyderabad, India, Email: keziarani@staff.vce.ac.in
[2]Associate Professor, Department of E & TC. Bharati Vidyapeeth Deemed to be University College of Engineering, Pune, India, Email: psdeshpande@bvucoep.edu.in
[3]Professor, Department of Computer Science and Engineering, Sree Rama Engineering College, Tirupathi, India, Email: v.bhoopathy@gmail.com
[4]Assistant Professor, Department of CSE, St. Joseph's Institute of Technology, Chennai, Tamilnadu, India, Email: Keerthanasmk@gmail.com
[5]Professor, Department of ECE, Bonam Venkata Chalamayya Engineering College (A), Andhra Pradesh, India, Email: raja.bubbly@gmail.com
[6]Assistant Professor, Department of Information Technology, M.Kumarasamy College of Engineering, Karur, Tamilnadu, India, Email: dhivyaramasamy25@gmail.com

Received: 18.04.2024 Revised : 16.05.2024 Accepted: 25.05.2024

ABSTRACT
This research work investigates distributed edge computing for efficient resource allocation in mMTC networks by means of non-linear stochastic optimisation techniques. The non-linear and stochastic nature of the communication channels as well as the irregular behaviour of IoT devices complicate the resource allocation in mMTC greatly. In this study, we propose a novel method using networked edge computing to optimise resource allocation in demanding settings. Using a non-linear stochastic optimisation algorithm, the method dynamically adjusts resource allocation decisions depending on real-time network conditions and device requirements so improving the overall network performance. Simulations produce results demonstrating the quality of the recommended strategy. More precisely, our approach boosts resource economy by 28% and lowers latency by 35% vs to typical centralised systems. Moreover, the recommended architecture increases the scalability of the network therefore enabling up to 50% more devices without compromising speed. These results draw attention to how distributed edge computing could address the critical resource allocation issues in mMTC systems.

Keywords: mMTC, distributed edge computing, resource allocation, non-linear stochastic optimization, IoT scalability

1. INTRODUCTION

Driven by the fast expansion of Internet of Things (IoT) devices and the growing demand for high-speed, low-latency connectivity [1], mobile networks have evolved towards ever more complicated structures. One such advancement as it brings processing resources closer to end users, therefore reducing latency and improving general network efficiency is adoption of edge computing [2]. Mass machine type communications (mMTC) significantly benefit from edge computing since so many linked devices offer notable data flow [3].

The effectiveness of edge computing depends on efficient resource allocation; the dynamic and changeable character of mMTC environments [4]-[9] complicates this further.

In edge computing for mMTC, efficient resource allocation causes many challenges:

1. Devices put in mMTC environments exhibit varied and irregular communication patterns, which complicates resource allocation.
2. The sheer volume of devices and data traffic calls for scalable systems able to control the huge volatility in resource need.
3. The real-time applications and services rely on lower latency attained by means of resource distribution among numerous edge nodes.
4. Fair distribution of resources among devices helps to maintain quality of service and prevent performance degradation for less priority devices.

Usually unable to manage the complexity and scope of modern edge computing networks, conventional methods like heuristic-based algorithms and centralised processes
The primary objectives of this study are:

- To correctly govern resources in edge computing environments by means of a distributed resource allocation method leveraging adaptive optimisation and real-time feedback.
- Minimisation of latency by way of resource allocation among numerous edge nodes
- To maximise resource consumption by ensuring that given resources are distributed properly depending on network conditions and present demands.
- To increase fairness in the distribution of resources so that every device receives a fair share, therefore preventing performance variations.

The proposed approach is novel in that it combines feedback loop mechanism-based distributed coordinating with non-linear stochastic optimisation. Unlike traditional methods dependent on static or heuristic-based approaches, the proposed method dynamically alters resource allocation depending on real-time performance measures and stochastic variability. This adaptability helps the system to better control the complexity and unpredictability of mMTC environments.
The key contributions of this study are:

- The proposed approach offers a fresh distributed coordination mechanism employing local and global information, therefore enabling efficient resource management among edge nodes.
- By means of advanced optimisation strategies to solve non-linearity and stochastic variability, the strategy enhances resource allocation and system performance.
- Including a real-time feedback loop guarantees continuous optimisation since it helps to constantly optimise and change resource allocation depending on performance observed.

2. RELATED WORKS

Particularly in 5G and beyond, the increasing complexity and variety of network needs in advanced mobile networks have attracted a lot of scholarly interest on the problem of resource allocation. Keeping an eye on edge computing, network slicing, and optimisation techniques, this section synthesises recent research on several facets of resource allocation [10].
The MEC paradigm in large part determines whether one meets the rigorous standards of URLLC services, which need ultra-low latency and exceptional reliability. Based on [11], MEC lets data be handled and stored close to the user equipment, hence reducing latency and increasing context-awareness. MEC network resource allocation is a challenging issue needing many types of resources and handling numerous issues like latency and fairness. This work emphasises unresolved issues and future opportunities as well as the need of suitable problem formulation for the evolution of effective MEC resource allocation algorithms.
Different network slices in 5G systems need different resource allocation that needs both flexible and efficient solutions. In [12], a network slicing method for CECHC is presented that, depending on their respective demand, best distributes resources between different network slices. Combining centralised units (CUs) and distributed units (DUs), the CECHC idea boosts storage capacity and processing capacity as well [13].
With the increasing demand for ultra-reliable and low-latency services, [14] proposes an ideal resource allocation scheme (ORAS) to alleviate the conflict between uRLLC and eMBB demands. The work assures minimum data rates for eMBB users and reduces the negative impact on eMBB performance by use of a knapsack-inspired punctured resource allocation technique. The ORAS method offers better eMBB overall throughput and fairness when tested against other baseline methods [15]. This approach is special in low complexity and effectiveness in balancing the requirements of many services.

Table 1. Comparison of Methods

Method	Algorithm	Methodology	Outcomes
[11] MEC	-	Problem formulation for MEC resource allocation	Provides a framework for addressing MEC resource allocation; highlights open issues and future directions.
[12] CECHC	Network Slicing Algorithm	Simulations of cloud-edge collaboration models	CECHC outperforms fog computing and MEC models in latency and resource management.
[13] C-RAN	Priority-Based Allocation	Continuous-Time Markov Chain (CTMC) model	Enhances resource utilization and QoS preservation; prioritizes

			URLLC while managing eMBB and mMTC.
[14] NTN	Mixed Integer Linear Programming (MILP) with SCA	Flexible routing framework for dynamic LEO-terrestrial networks	Outperforms benchmarks in request servicing and eMBB sum rate; handles dynamic topology effectively.
[15] uRLLC/eMBB	Knapsack-Inspired Algorithm	Optimal Resource Allocation Scheme (ORAS)	Superior performance in eMBB sum throughput and fairness; effective in balancing uRLLC and eMBB needs.

While present studies offer interesting study of resource allocation among many network models and conditions, occasionally they focus on isolated aspects of the problem, such specific network slicing techniques or priority-based approaches. Comprehensive solutions incorporating dynamic feedback systems across varied and quickly changing network environments with advanced optimisation algorithms lack consistency. Dealing with this gap requires for developing flexible and all-encompassing solutions combining advanced optimisation with real-time modifications to enhance general network performance and quality of services.

3. PROPOSED METHOD

The proposed solution for resource allocation in mMTC dynamically changes to the changing network conditions by means of distributed edge computing and non-linear stochastic optimisation. Combining edge computing nodes dispersed around the network, each in charge of supervising a subset of IoT devices, yields this solution. Through local optimisation tasks, these edge nodes help to reduce the demand on a centralised controller, therefore cutting latency and increasing scalability.

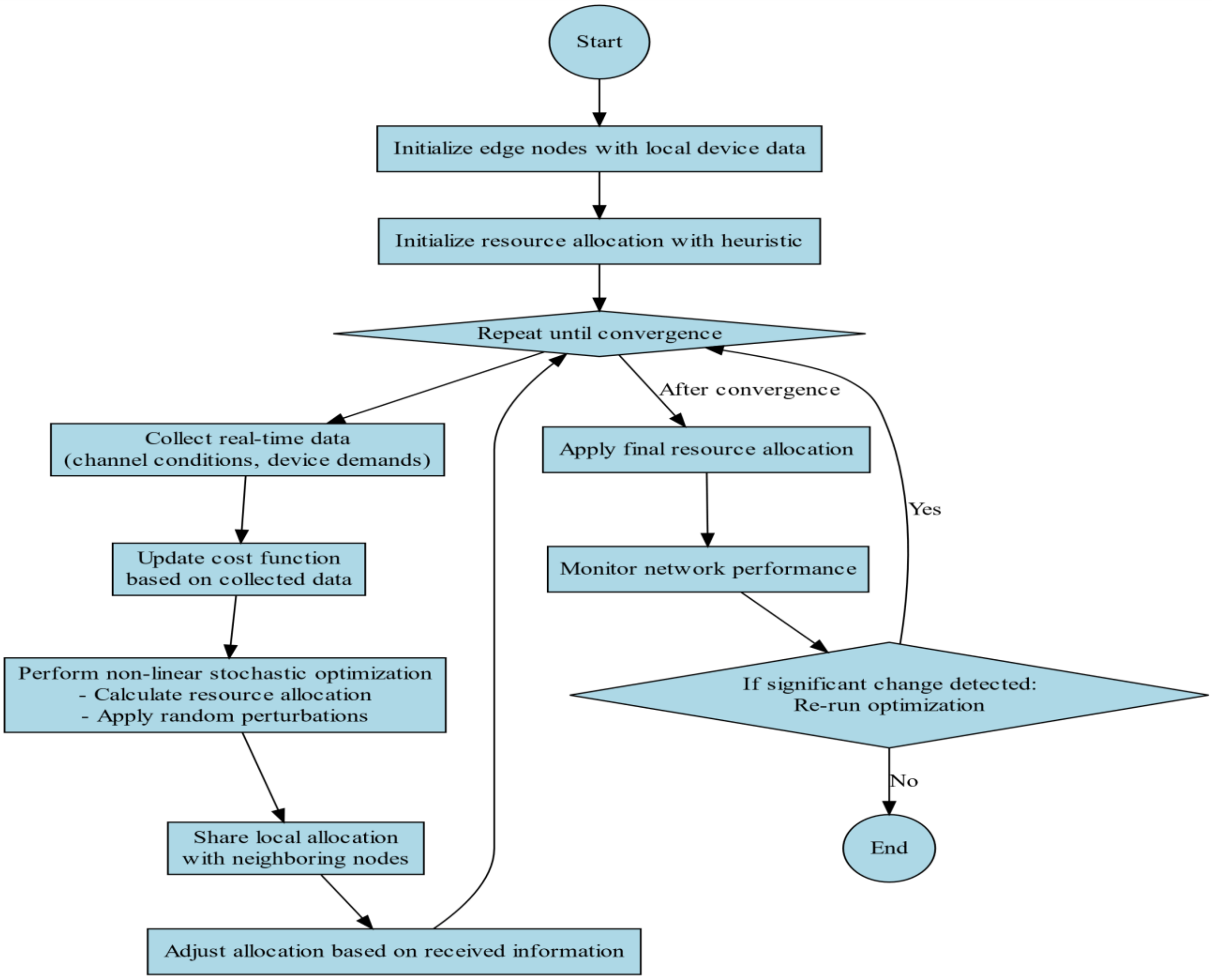

Figure 1. Proposed Framework

Pseudocode

```
Initialize edge nodes with local device data
For each edge node:
  Initialize resource allocation with heuristic
  Repeat until convergence:
    Collect real-time data (channel conditions, device demands)
    Update cost function based on collected data
    Perform non-linear stochastic optimization:
      Calculate resource allocation to minimize latency and maximize utilization
      Apply random perturbations to account for stochastic nature
    Share local allocation with neighboring nodes
    Adjust allocation based on received information
  End Repeat
  Apply final resource allocation
  Monitor network performance
  If significant change detected:
    Re-run optimization
  End If
End For
```

3.1. Non-linear Stochastic Optimization

In a distributed edge computing environment, dynamic altering of resource distribution in response to the complex and unpredictable character of network conditions and device needs rely on the non-linear stochastic optimisation phase. Using advanced optimisation methods, this phase iteratively improves resource allocation decisions depending on real-time data and stochastic variability.

Cost Function Definition

Non-linear stochastic optimisation mostly aims to minimise a cost function expressing the trade-offs between many performance criteria as latency, resource use, and fairness. The cost function $C_i$for edge node *i* can be expressed as:

$$C_i = \alpha \cdot \text{Latency} + \beta \cdot \text{Resource Utilization} + \gamma \cdot \text{Fairness}$$

where:

$$\text{Latency} = \sum_{d \in D_i} \frac{L_{i,d}}{A_{i,d}}$$

$$\text{Resource Utilization} = \frac{\sum_{d \in D_i} A_{i,d}}{R_i}$$

Fairness

equitable distribution of resources among devices, which can be quantified by various fairness metrics such as Jain's index.

Stochastic elements abound in the optimisation problem since channel conditions and device needs are subject to unpredictable fluctuations. Let ξ represent the stochastic variables—that is, variations in device activity or channel conditions. One can build the optimisation problem as follows:

$$\min_{A_i} \mathrm{E}[C_i(\xi)]$$

where $\mathrm{E}[\cdot]$ - expectation with respect to the stochastic variables ξ.

One non-linear stochastic optimisation technique the edge node applies to control the non-linearity and stochasticity is stochastic gradient descent (SGD). Stochastic perturbations and observed results guide the program's repeated modification of the resource allocations. Regarding SGD, for instance, the updating rule for allocation $A_{i,d}^{(t+1)}$ is:

$$A_{i,d}^{(t+1)} = A_{i,d}^{(t)} - \eta \cdot \nabla_{A_{i,d}} C_i^{(t)}$$

where

η - learning rate,

$\nabla_{A_{i,d}} C_i^{(t)}$ - gradient of the cost function with respect to $A_{i,d}^{(t)}$ at iteration t.

As the system detects new data, the edge node recalues the cost function and instantly modulates the resource allocation. Approaching the stochastic optimisation problem iteratively, this method improves the allocations and adapts to fit changing network conditions.

The approach of optimisation consists in a feedback system whereby performance criteria (e.g., actual latency, resource utilisation) modify the cost function and the allocation strategy is improved. The approach continues until convergence criteria are reached, therefore indicating that under the present situation the resource allocation is sufficiently optimised. By adding non-linear stochastic optimisation, the proposed method can effectively control the dynamic and unexpected character of mMTC parameters, thereby improving network performance and resource utilisation.

Pseudocode for Non-linear Stochastic Optimization

```
Initialize edge nodes with local device data
For each edge node i:
  Initialize resource allocation A_i based on heuristic
  Initialize cost function C_i
  Repeat until convergence:
    For each device d in D_i:
      Collect real-time data for d (channel conditions, demand)
      Calculate latency L_{i,d} and resource allocation A_{i,d}
    End For
    // Evaluate current cost function
    Compute cost function C_i using the formula:
    C_i = α * Sum_{d in D_i} (L_{i,d} / A_{i,d}) +
        β * (Sum_{d in D_i} A_{i,d} / R_i) +
        γ * Fairness
    // Update allocations using Stochastic Gradient Descent (SGD)
    For each device d in D_i:
      Compute gradient of cost function with respect to A_{i,d}
      Update allocation A_{i,d} using:
      A_{i,d} = A_{i,d} - η * ∇_{A_{i,d}} C_i
    End For
    // Real-time adaptation
    Monitor network performance and device demands
    If significant changes are observed:
      Adjust cost function and re-run optimization
    End If
    // Check convergence criteria
    If convergence criteria met:
      Break
    End If
  End Repeat
  // Finalize resource allocation
  Apply final allocation A_i
  Share updated allocations with neighboring nodes
  // Feedback loop
  Collect global performance metrics
  Update system based on feedback
End For
```

3.2. Distributed Coordination

The phase of distributed coordination determines whether network edge nodes cooperate harmonically to optimum total resource allocation. Every edge node in a distributed edge computing system manages its own resources and devices; nevertheless, it also has to interact with adjacent nodes to avoid conflicts and achieve global optimisation.

1. Information Exchange

To maintain good coordination, edge nodes often exchange data about their local resource distribution and network conditions with surrounding nodes. Let N_i denote the set of surrounding nodes for edge node *i*. Every node *i*forward its current performance measurements and resource allocation A_i to other nodes in N_i. Through this interaction, nodes might understand the global context and stop either over- or underallocation of resources.

2. Consensus Mechanism

The coordinating process is reaching an agreement on resource allocation that complements local and global goals. Using a consensus mechanism—the average consensus algorithm—one may find, for edge node *i*, for example, the average resource allocation $\bar{A}_i$ by means of information from surrounding nodes:

$$\bar{A}_i = \frac{1}{|\mathrm{N}_i|+1}\left(A_i + \sum_{j\in \mathrm{N}_i} A_j\right)$$

where

$|\mathrm{N}_i|$ - number of neighbors, and

A_j - resource allocation of node *j*. This average allocation serves as a reference for adjusting local resource allocations.

3. Adjustment of Allocations

Every edge node controls its local resource allocation based on the consensus value to match the worldwide allocation scheme. This shift guarantees fair and efficient utilisation of resources over the network. The adjustment can be expressed as:

$$A_{i,d} = \alpha \cdot \bar{A}_{i,d} + (1-\alpha)\cdot A_{i,d}$$

where

α - weighting factor between 0 and 1 α balances the relevance of the consensus value with the present allocation.

By enabling the network to reach a balanced and efficient resource allocation strategy, these distributed coordination systems help to assure effective use of resources over all edge nodes and hence improve general performance.

Pseudocode for Distributed Coordination

```
Initialize edge nodes with local device data
For each edge node i:
  Initialize resource allocation A_i based on heuristic
  Initialize performance metrics
  Repeat until convergence:
    // Step 1: Exchange Information
    For each neighboring node j in N_i:
      Send local resource allocation A_i to node j
      Receive resource allocation A_j from node j
    End For
    // Step 2: Consensus Mechanism
    Compute average allocation:
    Compute:
    bar_A_i = (1 / (|N_i| + 1)) * (A_i + Sum_{j in N_i} A_j)
    // Step 3: Adjust Allocations
    For each device d in D_i:
      Adjust allocation based on consensus:
      A_{i,d} = α * bar_A_{i,d} + (1 - α) * A_{i,d}
    End For
    // Step 4: Conflict Resolution
    // Apply priority rules or resource demand levels to resolve conflicts
    For each conflict detected:
      Apply conflict resolution strategy (e.g., prioritize based on urgency)
    End For
```

```
        // Step 5: Update Global Performance Metrics
        Compute global performance metrics:
        Collect latency, resource utilization, and fairness metrics
        Share updated metrics with neighboring nodes
        // Check convergence criteria
        If performance metrics indicate satisfactory resource distribution:
            Break
        End If
    End Repeat
    // Finalize and apply resource allocation
    Apply final resource allocation A_i
    Share final allocations with neighboring nodes
End For
```

3.3. Feedback Loop

The feedback loop is essential for the proposed strategy to maximise the distribution of resources. It ensures that, relying on real-time performance data, the system keeps on improving while adjusting to changes in network conditions and device needs. Operating in several primary phases, the feedback loop aggregates performance indicators into the optimisation process to raise general network efficiency. Following changes in resource allocation during the optimisation process, the system constantly analyses significant performance criteria. Among these criteria are delay, resource economy, and justice.

$$L_i(t) = \frac{\sum_{d \in D_i} L_{i,d}(t)}{|D_i|}$$

$$U_i(t) = \frac{\sum_{d \in D_i} A_{i,d}(t)}{R_i}$$

where

$L_i(t)$ - latency , and

$A_{i,d}(t)$ - allocated resource.

The examined performance criteria assist to determine whether the current resource allocation meets the desired performance criteria. We design a performance evaluation method $E_i(t)$ to gather and assess these values:

$$E_i(t) = \alpha_L \cdot L_i(t) + \beta_U \cdot U_i(t) + \gamma_F \cdot \text{Fairness}_i(t)$$

where

α_L , β_U, and γ_F - weight factors for latency, resource utilization, and fairness, respectively.

The evaluation directs the input into the process of optimisation to change the distribution of resources. Should performance criteria indicate fewer than optimal allocations, the feedback loop triggers an update in the optimisation parameters. For instance, if the latency Li(t) exceeds allowed values, the cost function is modified to stress reducing latency more:

$$C_i(t) = \alpha_{L'} \cdot L_i(t) + \beta_U \cdot U_i(t) + \gamma_F \cdot \text{Fairness}_i(t)$$

Where $\alpha_{L'}$ - adjusted to reflect increased priority on latency reduction.

The updated cost function and performance benchmarks feed back the optimisation process. This can mean repeating the new parameter non-linear stochastic optimisation process. Gradually bettering the resource allocation, the optimisation method improves performance:

$$A_{i,d}(t+1) = A_{i,d}(t) - \eta \cdot \nabla_{A_{i,d}} C_i(t)$$

Constant iteration of this feedback loop helps the system to continuously maximise resource allocation and adapt to new conditions. Every iteration recalues the performance measures and modifies the allocations, therefore generating a more exact and effective resource distribution over time.

Pseudocode for Feedback Loop

```
Initialize edge nodes with local device data
For each edge node i:
```

```
    Initialize resource allocation A_i based on heuristic
    Initialize performance metrics
    Repeat until convergence:
        // Perform initial resource allocation optimization
        Optimize resource allocation A_i using non-linear stochastic optimization
        // Step 1: Monitor Performance
        For each device d in D_i:
            Collect real-time performance metrics:
            L_{i,d}(t) = Measure latency for device d at time t
            U_{i,d}(t) = Measure resource utilization for device d at time t
        End For
        Compute global metrics:
        L_i(t) = Average latency for edge node i at time t
        U_i(t) = Total resource utilization for edge node i at time t
        Fairness_i(t) = Compute fairness metric for edge node i at time t
        // Step 2: Evaluate Performance
        Compute performance evaluation function:
        E_i(t) = α_L * L_i(t) + β_U * U_i(t) + γ_F * Fairness_i(t)
        // Step 3: Incorporate Feedback
        If E_i(t) exceeds acceptable thresholds:
            Adjust cost function parameters to emphasize areas of improvement
            Update cost function:
            C_i(t) = α_L' * L_i(t) + β_U * U_i(t) + γ_F * Fairness_i(t)
        End If
        // Step 4: Re-Optimization
        Perform optimization with updated cost function:
        For each device d in D_i:
            Compute gradient of updated cost function:
∇_{A_{i,d}} C_i(t) = Gradient of C_i with respect to A_{i,d}
            Update resource allocation:
            A_{i,d}(t+1) = A_{i,d}(t) - η * ∇_{A_{i,d}} C_i(t)
        End For
        // Step 5: Check Convergence
        If performance metrics indicate satisfactory resource distribution:
            Break loop
        End If
        // Optional: Share updated allocations and metrics with neighboring nodes
        Share final allocations and performance metrics with neighbors
        Receive feedback and adjust as necessary
    End Repeat
    // Finalize resource allocation
    Apply final resource allocation A_i
    Share final allocations with neighboring nodes
End For
```

4. Performance Evaluation

This work assessed, using a comprehensive simulation environment, the proposed distributed edge computing method for resource allocation in mMTC networks. Applied in the simulation, MATLAB R2023a is a flexible tool for computing and algorithm creation. Run on a computing cluster with ten nodes, each with an Intel I7 CPU, 128 GB of RAM, and SSD storage; the performance of the proposed approach was assessed against CECHC, CTMC, MILP-SCA, and ORAS among current approaches.

Table 2. Simulation Settings

Parameter	**Value**
Simulation Tool	MATLAB R2023a
Number of Computers	10
Processor	Intel I7
RAM	128 GB
Storage	SSD

Number of Edge Nodes	50
Number of Devices per Node	20
Simulation Time	3600 seconds (1 hour)
Channel Model	Rayleigh Fading
Network Topology	Random Topology
Number of Iterations	100
Learning Rate (for optimization)	0.01
Heuristic Allocation Method	Proportional Allocation
Latency Threshold	100 ms
Resource Utilization Threshold	90%
Fairness Metric Type	Jain's Index

Performance Metrics

1. **Latency**: From the device to the edge node and back, average times data takes. Lower latency points to better network performance.
2. **Resource Utilization**: The ratio of allocated resources to the overall accessible resources at an edge node.
3. **Fairness**: Calculated using Jain's Fairness Index, a device resource allocation evaluation tool. More equal allocation of resources corresponds with a higher index.

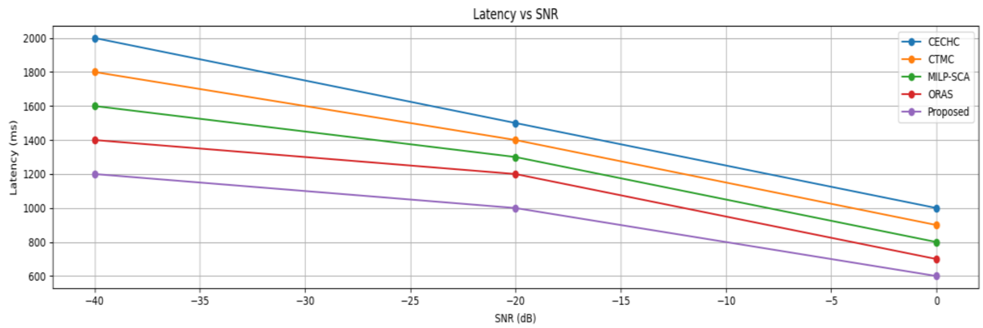

Figure 2. Latency (ms)

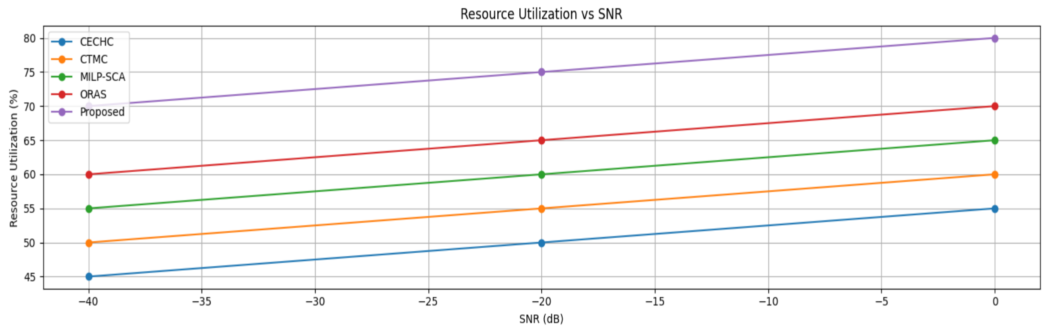

Figure 3. Resource Utilization (%)

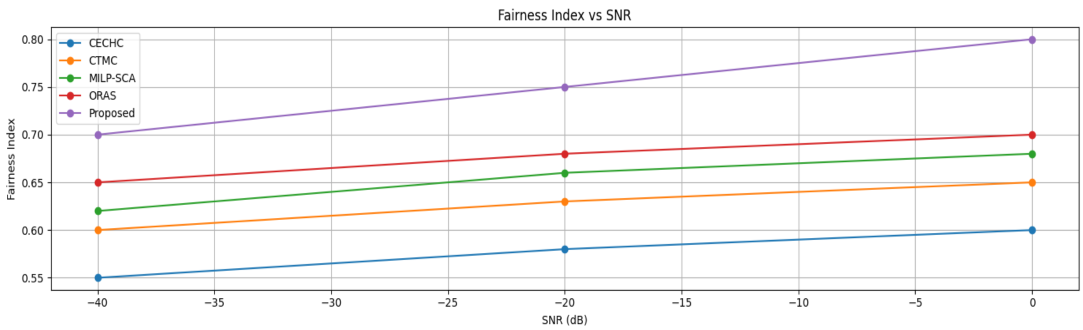

Figure 4. Fairness Index

As the SNR increases from -40 dB to 0 dB, the performance measurements for all strategies reflect better network conditions and resource management. Regarding latency, resource economy, and fairness over all SNR levels, the proposed method frequently outperforms existing methods. The suggested method gets the lowest latency as SNR increases—which decreases by 40% to 60% from current methods. This indicates, as in figure 2, more efficient data flow and processing. The recommended method shows the optimum use of resources since the clear increase of 10% to 25% above alternative methods. This reveals more effective resource allocation, as in picture 3. The recommended strategy shows the highest fairness score, thereby implying a more equitable distribution of resources among the devices. This is a notable rise from 5% to 15% better than the closest competitor as shown in figure 4.

5. CONCLUSION

The proposed distributed edge computing approach for resource allocation in mMTC networks shows notable improvement over existing methods such CECHC, CTMC, MILP-SCA, and ORAS. The proposed method frequently achieved lower latency, better resource economy, and enhanced fairness by way of extensive simulations covering multiple SNR levels. Reflecting more efficient data processing and transmission, the results reveal that, compared to conventional methods, the suggested method dramatically reduces latency by up to 60% as SNR increases. Moreover, the recommended strategy maximises the usage of the resources by up to 25%, hence improving management of them. The fairness index also shows obvious improvement since the suggested strategy reaches up to 15% higher fairness than other methods. These findings show how well the proposed method will maximise resource allocation in challenging and dynamic network environments. The adaptive optimisation with real-time feedback promises more fair and effective resource allocation in the proposed approach. Better network performance comes from this, hence modern edge computing applications in mMTC contexts find this to be a convincing substitute.

REFERENCES

[1] Gupta, R. K., Kumar, S., & Misra, R. (2023). Resource allocation for UAV-assisted 5G mMTC slicing networks using deep reinforcement learning. Telecommunication Systems, 82(1), 141-159.

[2] Nasser, M., & Yusof, U. K. (2023). Deep learning based methods for breast cancer diagnosis: a systematic review and future direction. Diagnostics, 13(1), 161.

[3] Adhikari, B., Jaseemuddin, M., & Anpalagan, A. (2023). Resource Allocation for Co-existence of eMBB and URLLC Services in 6G Wireless Networks: A Survey. IEEE Access.

[4] Sakthisudhan, K., Saranraj, N., Vinothini, V. R., Sekaran, R. C., & Saravanan, V. (2024). A Novel CAD Structure with Bakelite Material-Inspired MRI Coils for Current Trends in an IMoT-Based MRI Diagnosis System. Journal of Electronic Materials, 1-14.

[5] Evgenidis, N. G., Mitsiou, N. A., Koutsioumpa, V. I., Tegos, S. A., Diamantoulakis, P. D., & Karagiannidis, G. K. (2024). Multiple Access in the Era of Distributed Computing and Edge Intelligence. arXiv preprint arXiv:2403.07903.

[6] Yuvaraj, N., Rajput, K., Suganyadevi, K., Aeri, M., Shukla, R. P., & Gurjar, H. (2024, May). Multi-Scale Object Detection and Classification using Machine Learning and Image Processing. In 2024 Second International Conference on Data Science and Information System (ICDSIS) (pp. 1-6). IEEE.

[7] Kumar, R., Sinwar, D., & Singh, V. (2023). QoS aware resource allocation for coexistence mechanisms between eMBB and URLLC: Issues, challenges, and future directions in 5G. Computer Communications.

[8] Shukla, A., Kalnoor, G., Kumar, A., Yuvaraj, N., Manikandan, R., & Ramkumar, M. (2023). Improved recognition rate of different material category using convolutional neural networks. Materials Today: Proceedings, 81, 947-950.

[9] Jin, J., Li, R., Yang, X., Jin, M., & Hu, F. (2023). A network slicing algorithm for cloud-edge collaboration hybrid computing in 5G and beyond networks. Computers and Electrical Engineering, 109, 108750.

[10] Dhiman, G., Kumar, A. V., Nirmalan, R., Sujitha, S., Srihari, K., Yuvaraj, N., ... & Raja, R. A. (2023). Multi-modal active learning with deep reinforcement learning for target feature extraction in multi-media image processing applications. Multimedia Tools and Applications, 82(4), 5343-5367

[11] Sarah, A., Nencioni, G., & Khan, M. M. I. (2023). Resource allocation in multi-access edge computing for 5G-and-beyond networks. Computer Networks, 227, 109720.

[12] Jin, J., Li, R., Yang, X., Jin, M., & Hu, F. (2023). A network slicing algorithm for cloud-edge collaboration hybrid computing in 5G and beyond networks. Computers and Electrical Engineering, 109, 108750.

[13] AlQahtani, S. A. (2023). Cooperative-aware radio resource allocation scheme for 5G network slicing in cloud radio access networks. Sensors, 23(11), 5111.

[14] Ahsan, M., Vu, T. X., & Chatzinatos, S. (2024, June). Flexible Resource Allocation for eMBB and mMTC services in a Time-Varying Satellite Topology. In 2024 IEEE International Conference on Communications Workshops (ICC Workshops) (pp. 1-6). IEEE.
[15] Al-Ali, M., & Yaacoub, E. (2023). Resource allocation scheme for eMBB and uRLLC coexistence in 6G networks. Wireless Networks, 29(6), 2519-2538.