International Journal of Internet, Broadcasting and Communication Vol.16 No.3 16-26 (2024)
http://dx.doi.org/10.7236/IJIBC.2024.16.3.16

IJIBC 24-3-2

Efficient Task Offloading Decision Based on Task Size Prediction Model and Genetic Algorithm

Quan T. Ngo, Dat Van Anh Duong and Seokhoon Yoon*

Ph.D., Department of Electrical and Computer Engineering, University of Ulsan, Korea
Ph.D., Department of Electrical and Computer Engineering, University of Ulsan, Korea
**Professor, Department of Electrical and Computer Engineering, University of Ulsan, Korea*
tanquan.dn@gmail.com, mrdvad11@gmail.com, seokhoonyoon@ulsan.ac.kr

Abstract

Mobile edge computing (MEC) plays a crucial role in improving the performance of resource-constrained mobile devices by offloading computation-intensive tasks to nearby edge servers. However, existing methods often neglect the critical consideration of future task requirements when making offloading decisions. In this paper, we propose an innovative approach that addresses this limitation. Our method leverages recurrent neural networks (RNNs) to predict task sizes for future time slots. Incorporating this predictive capability enables more informed offloading decisions that account for upcoming computational demands. We employ genetic algorithms (GAs) to fine-tune fitness functions for current and future time slots to optimize offloading decisions. Our objective is twofold: minimizing total processing time and reducing energy consumption. By considering future task requirements, our approach achieves more efficient resource utilization. We validate our method using a real-world dataset from Google-cluster. Experimental results demonstrate that our proposed approach outperforms baseline methods, highlighting its effectiveness in MEC systems.

Keywords: *Task Offloading, Mobile Edge Computing, Predictive Model, Genetic Algorithm.*

1. INTRODUCTION

In the landscape of mobile edge computing, task offloading has gained prominence as a strategic approach to enhance system performance and resource utilization [1-3]. Task offloading refers to the practice of transferring computational tasks from resource-constrained devices (such as mobile phones or IoT devices) to more powerful servers located at the network edge or in the cloud [4-6]. By doing so, we alleviate the burden on local devices, improve response times, and optimize energy consumption.

The proliferation of mobile devices, coupled with the exponential growth in data-intensive applications, underscores the importance of efficient task management. Users demand seamless experiences, whether streaming high-definition videos, running complex machine learning models, or participating in real-time collaborative applications. However, the limited computational capabilities of mobile devices often hinder

Manuscript Received: May. 1, 2024 / Revised: May. 6, 2024 / Accepted: May. 11, 2024
Corresponding Author: seokhoonyoon@ulsan.ac.kr
Tel: +82-52-259-1403, Fax: +82-52-259-1687
Professor, Department of Electrical, Electronic and Computer Engineering, University of Ulsan, Korea

their ability to handle such workloads effectively. Task offloading offers a compelling solution by enabling devices to leverage external resources, such as Unmanned Aerial Vehicles (UAVs), for computation, storage, and communication. UAVs offer dynamic deployment, reducing network latency and improving responsiveness. Their scalability and extended coverage benefit geographically dispersed scenarios and areas with limited edge server infrastructure.

Despite the benefits of task offloading, existing methods predominantly focus on immediate decisions without considering future task requirements. This oversight can lead to suboptimal resource allocation, inefficient energy usage, and missed opportunities for performance improvement. In our paper, we address this limitation by proposing an innovative approach that leverages predictive models to anticipate future task sizes. By incorporating this foresight, we aim to enhance the decision-making process for task offloading, ultimately achieving more efficient resource utilization in mobile edge computing systems.

Our solution combines two key components: predictive task sizing using recurrent neural networks (RNNs) and decision optimization through genetic algorithms (GAs). Here's how it works:

- Predictive Task Size with RNNs: We propose an RNN model to predict task sizes for future time slots. By analyzing historical data and patterns, the RNNs estimate the computational requirements of upcoming tasks. These predictions serve as valuable input for our offloading decisions. Considering future task sizes allows us to allocate resources more effectively, preventing underutilization or overloading of edge servers.
- Genetic Algorithm (GA) for Decision Optimization: To fine-tune our offloading decisions, we employ GAs. These optimization techniques iteratively evolve a population of potential solutions to find the best configuration. Our fitness functions consider both current and future time slots. We aim to minimize total processing time while reducing energy consumption. GA adaptively adjusts the decision parameters based on the predicted task sizes

We validate our method using a real-world dataset from Google-cluster [7]. This dataset provides diverse task profiles, reflecting the complexity and variability of actual workloads. Our experiments demonstrate that our proposed approach outperforms baseline methods. By incorporating future task requirements, we achieve more efficient resource utilization and improved system performance. In summary, our approach leverages predictive modeling and optimization to enhance task offloading decisions in MEC systems. By considering the future, we pave the way for more intelligent and effective resource management.

The rest of this paper is organized as follows Section 2 details the system model and problem formulation, Section 3 presents the methodology, Section 4 discusses the experimental results, and Section 5 concludes the paper.

2. SYSTEM MODEL AND PROBLEM FORMULATION

2.1 System Model:

This paper considers a communication system with a single Unmanned Aerial Vehicle (UAV). The UAV, denoted by U, acts as an aerial base station to provide wireless connectivity for ground-based mobile devices (MDs). These devices, represented by the set M ($M = \{1, ..., |M|\}$), can communicate with the UAV (indicated by dashed lines) for data transmission and reception. This architecture is particularly beneficial in areas lacking infrastructure, such as remote locations or disaster zones. The system leverages advanced wireless technologies like LTE, 5G, and Wi-Fi to establish reliable and adaptable communication channels that can adjust to

changing operational demands and environmental conditions. Figure 1 shows the system model for this paper.

At the beginning of each fixed-time slot, each mobile device m $\in$ M has a computing task with task size K_t^m to complete, where t is the time slot index. The UAV (U) makes offloading decisions at the start of each time slot based on available information.

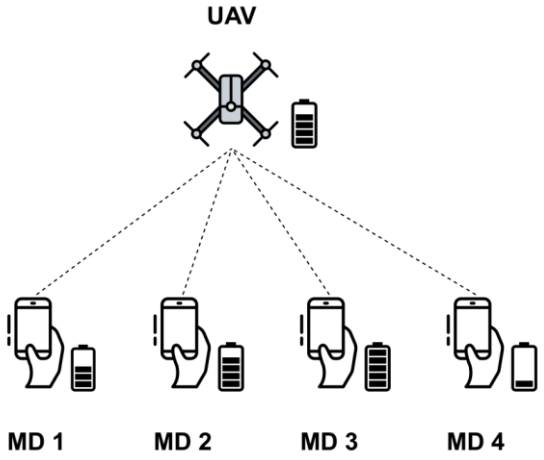


Figure 1. The system model

2.2 Communication Model:

When the MD offloads tasks to UAV. the total task processing time includes several components:

- Control Signal Transmission and Reception: Time for exchanging control signals related to task requests and decisions.
- Decision-Making Time: The duration spent on making offloading decisions.
- Task Transmission Time: Time taken to transmit the task data.
- Propagation Delay: The delay due to signal propagation through the network.
- Task Execution Time: The actual computation time for task execution.
- Output Data Transmission Time: The time needed to send the processed output back to the US.

Since the output data size is typically smaller than the input data, and downlink data rates are higher than uplink rates, the post-processing data transfer time is often omitted. Additionally, control signals are assumed to use a dedicated channel, and decision-making algorithms run on edge servers, making their contribution negligible and thus ignored in this study.

In this study, we adopt orthogonal frequency division multiple access (OFDMA) [8] as the multiple access scheme for uplink communication. In OFDMA, the total available bandwidth is divided into sub-bands, with each sub-band assigned to MD. Specifically, the total bandwidth is divided into |M| sub-bands, each of size B and each MD is allocated to one of these sub-bands.

The communication channels between UAV and the MD follow the free-space path loss model. Since UAV

operate at high altitudes, line-of-sight channels dominate in UAV communication scenarios. The uplink bitrate, denoted as $bitrate^m$, between MD m and UAV can be formulated as follows:

$$bitrate^m = B \times \log_2(1 + \frac{g_0}{(d^m)^2} \times \frac{p}{\sigma^2}) \quad (1)$$

where: σ is the noise power, d^m is the distance between MD m and UAV U, g_0 is the power gain with the reference distance of 1 meter, p the transmission power

2.3 Computational Model:

2.3.1 Local Computing:

When MD m executes a task in time slot t, the task execution time $T^m_{local,t}$ is calculated as:

$$T^m_{local,t} = \frac{C^m_t}{R^m} \quad (2)$$

Where:

- C^m is the computational requirement in CPU cycles for task K^m_t. In this work, we assume C^m is in proportion to the task size K^m_t follow the equation: $C^m = pd \times K^m_t$

  where pd (CPU cycles per bit) is the processing density.

- R^m represent the computing capacity of the MD m.

The energy consumption $E^m_{local,t}$ for completing task K^m_t locally is calculated as follows:

$$E^m_{local,t} = \alpha \times C^m_t \quad (3)$$

where α denotes the energy consumed by MD per CPU cycle.

2.3.2 Edge Computing in UAV:

When task K^m_t of MD m is offloaded to the UAV U, the total task processing time $T^m_{UAV,t}$ is the total of transmission time and the task execution time and is calculated as follows:

$$T^m_{UAV,t} = \frac{K^m_t}{bitrate^m} + \frac{C^m_t}{R^U} \quad (4)$$

Where: R^U represent the computing capacity of the UAV U

The energy consumption $E^m_{UAV,t}$ for completing task K^m_t in UAV U is calculated as follows:

$$E^m_{UAV,t} = P_{trans} \times \frac{K^m_t}{bitrate^m} + \beta \times C^m_t \quad (5)$$

where: P_{trans} is the transmission power, β is the energy consumed by UAV per CPU cycle

2.4 Problem Formulation:

This work focuses on optimizing the total cost associated with processing tasks from MDs. Our objective is to minimize the combined time and energy consumption required to complete all the tasks. The main problem can be formulated as follows:

$$minimize\ System\ cost\ C_t = \sum_m [w \times (T^m_{local,t} + T^m_{UAV,t}) + (1-w)(E^m_{local,t} + E^m_{UAV,t})]$$

The weight parameter w allows us to balance the trade-off between time and energy costs. Adjusting w based on system requirements ensures flexibility in achieving the desired balance.

3. METHODOLOGY

3.1 Task Size Prediction Model:

Current offloading decisions in computational systems rely heavily on limited information, like current workload, neglecting the dynamic nature of tasks. Task size can vary significantly, leading to unnecessary offloading and reduced efficiency. This work proposes predicting future task size and incorporating it into the decision process. This allows for reduced unnecessary offloading, improved system efficiency, and enhanced energy savings.

While individual tasks may seem random, the overall workload exhibits patterns over time. We leverage historical data to predict future task sizes. In this work, we propose utilizing Long Short-Term Memory (LSTM) networks for this prediction task. LSTMs are well-suited for analyzing sequential data like time series, making them ideal for capturing the temporal dependencies within MD workloads. By training an LSTM model on historical task sizes, we aim to achieve accurate predictions of the next time slot's task size. This predicted information will then be incorporated into the offloading decision-making process.

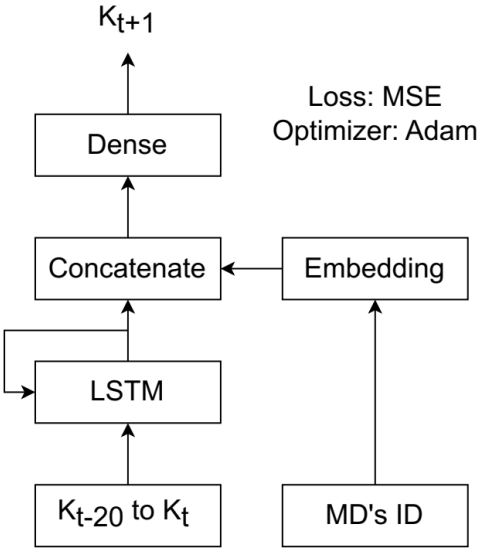


Figure 2. The task size prediction model

The architecture of our proposed model for predicting MD task size is illustrated in Figure 2. The model leverages two key inputs. Historical Task Size: The last 20 historical task sizes of the current MD simulation are fed into a Long Short-Term Memory (LSTM) layer. The LSTM layer is adept at capturing temporal dependencies within sequential data, allowing it to learn patterns from the past workload to predict future task sizes. MD ID Embedding: The ID of the specific MD simulation is fed into an embedding layer. This layer transforms the categorical ID into a dense vector representation, potentially capturing information about the

characteristics of different MD simulations that might influence their task size. The outputs from the LSTM layer and the embedding layer are then concatenated (combined) before being fed into a final dense layer. This dense layer performs a linear transformation to predict the task size of the MD simulation for the next time slot.

3.2 Genetic Algorithm for Task Offloading Decision-making:

Evolutionary algorithms, particularly genetic algorithms (GAs) [9], are well known for their capability of solving complex optimization problems. In this work, a GA is proposed to optimize the offloading decision.

- Solution Representation:

  We represent solutions as vectors $y = (y^1, y^2, \dots, y^{|M|})$, where |M| is the number of MDs.

  Each element y^i represents the offloading decision for a particular task:

  - $y^i = 0$ indicates the task will be processed locally on the MD.
  - $y^i = 1$ indicates the task will be offloaded to UAV.

  Key Distinction from Standard GAs: Unlike typical GAs that focus on a single solution, our proposed GA generates two solutions for each individual in the population:

  - The first solution (y_t) represents the offloading decisions for the current time slot (t).
  - The second solution (y_{t+1}) represents the predicted offloading decisions for the next time slot $(t+1)$.

- Fitness Function: The fitness function evaluates the combined system cost for both time slots $(t \; and \; t+1)$. This cost typically considers factors like execution time and energy consumption.

  A lower system cost translates to a higher fitness score, guiding the GA towards solutions that optimize resource utilization.

  The fitness function can be mathematically represented as $F(y_t, y_{t+1})$, where F represents the cost calculation for both time slots based on the offloading decisions in $y_t \; and \; y_{t+1}$.

- Selection Process: The selection process identifies the top n individuals (solutions) with the highest fitness scores from the current population. This selection can be achieved using various techniques like roulette wheel selection or tournament selection.

- Crossover Operation: We employ single-point crossover for generating offspring solutions. During crossover, two parent solutions exchange genetic material (offloading decisions for specific tasks) at a randomly chosen point within the vector representation. This process creates new offspring solutions with potentially improved characteristics.

- Mutation Operation: To maintain diversity and prevent premature convergence, a single bit (offloading decision for a single task) is randomly flipped within each offspring solution with a predetermined mutation rate.

- Replacement Strategy: The offspring solutions generated through crossover and mutation compete with the existing population. In this work, the worst individuals in the populations are replaced with the best offspring.

Please note that, after obtaining the best solutions y_t and y_{t+1}, only the offloading decision y_t is applied to the current time slot t. The above process is repeated at the beginning of each time slot.

4. EXPERIMENTAL RESULTS

4.1 Experimental Setup

In our research, we utilize a dataset sourced from Google Cluster [7]. This dataset contains crucial information about various tasks, including their arrival times, data sizes, processing durations, and associated deadlines. Notably, these tasks encompass a wide spectrum, ranging from substantial tasks like big data analysis and real-time video processing to smaller-scale tasks such as image processing within virtual reality environments.

To ensure compatibility with our established model, we preprocess the raw data. This involves normalization and denormalization techniques tailored to the specific characteristics of the data. By doing so, we align the data sizes with our model's requirements, allowing for more accurate analysis and predictions.

Drawing insights from relevant literature [10, 11], we establish key parameter settings. These parameters play a pivotal role in shaping our model's performance and overall effectiveness. For a comprehensive overview, you can refer to Table 1, where we outline these essential parameter values.

Table 1. Experimental settings

Parameter	Value
Number for MDs $(\|M\|)$	25
Number of UAVs	1
Task size group (5 groups)	[1-2], [2-3], [3-4], [4-5], [5-6] Mbits
Processing density (pd)	0.297 gigacycles per Mbits
MD CPU computing capacity (R^m)	2.5 GHz
UAV CPU computing capacity (R^U)	41.8 GHz
Transmission power (P_{trans})	2 W
Bandwidth (B)	20 MHz
g_0	-50 dBm
σ^2	-100 dBm
Energy consumed by MD per CPU cycle (α)	5.625 J per gigacycle
Energy consumed by UAV per CPU cycle (β)	2.708 J per gigacycle
Number of evaluation time slots	80

Since the dataset has no location information, we randomly generated MD locations within a 500 meters by 500 meters area. The UAV hovers at an altitude of 100 meters, directly above the center of the experimental

area.

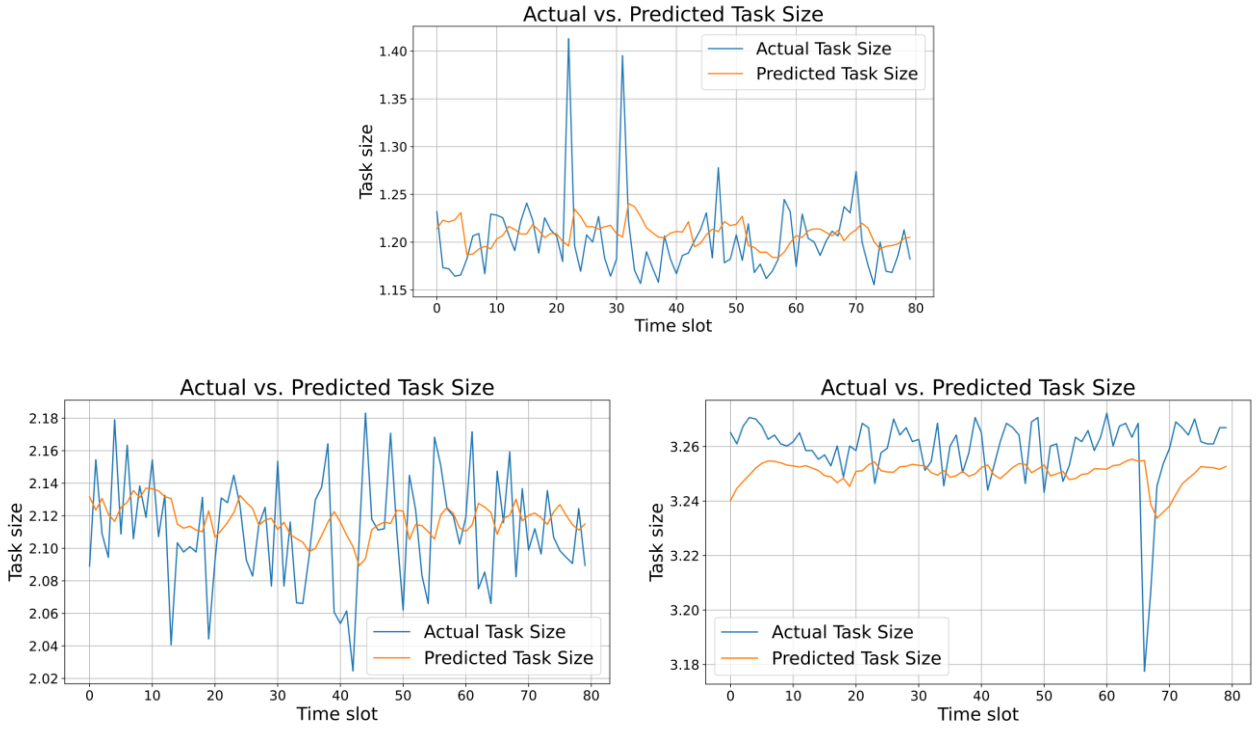
To assess the effectiveness of our proposed prediction-based Genetic Algorithm (GA), we compare its performance with two baseline approaches:

- **Greedy Algorithm:** This baseline implements a simple greedy strategy. It first sorts all tasks based on their task sizes. Then, it iterates through each task, comparing the cost of processing it locally versus offloading it to a UAV. If offloading is cheaper based on estimated costs, the task is marked for offloading; otherwise, it is processed locally.
- **Standard GA:** This baseline utilizes a standard GA similar to our proposed approach, but with a key difference. It focuses solely on optimizing offloading decisions for the current time slot. It does not consider predicted task sizes for the next time slot. This allows us to isolate the impact of incorporating future predictions into the offloading decision process.

By comparing our proposed prediction-based GA with these baselines, we can evaluate the benefits of including predicted task sizes in the offloading decision-making process. The greedy algorithm provides a basic benchmark for cost-aware offloading, while the standard GA serves as a reference point for the effectiveness of incorporating future predictions.

4.2 Task Size Prediction Performance

In a stable real edge computing scenario, the tasks generated by terminal devices are highly correlated with time and have a certain continuity and regularity. Generally, several times history window is used to predict the task at $t + 1$ times. In the experiments, we set the history window to 20 and trained one model for all task size groups. Figure 3 shows the predicted and actual task size of five random MD from 5 task size groups. The prediction model can effectively predict the future task size with a small MSE.



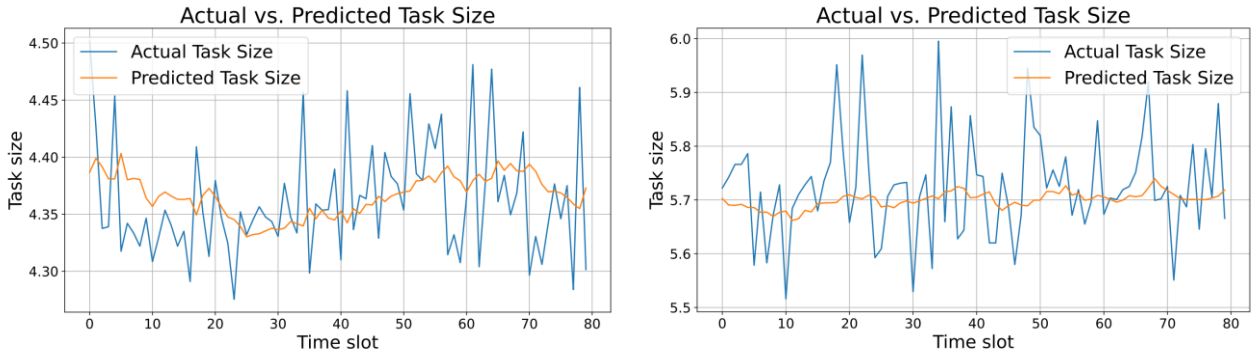


Figure 3. The task size prediction model performance

4.3 Impact of Task Size:

This experiment investigates the relationship between task size and system cost (processing time and energy consumption). We collected data from various MD simulations with different task sizes. The results are presented in Figure 4.

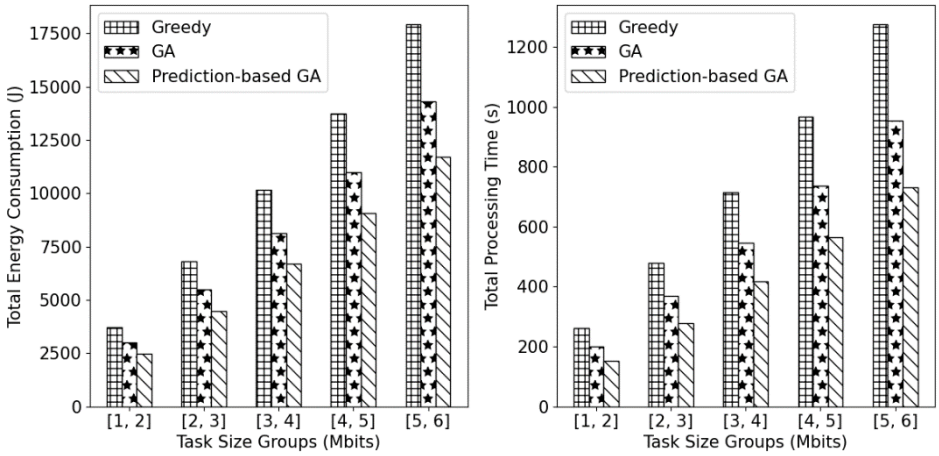


Figure 4. Effect of different task sizes

As expected, the overall energy consumption and processing time increase with increasing task size. This aligns with the intuition that larger tasks require more computational resources, leading to higher energy expenditure and longer execution times. Figure 4 shows that the proposed model achieves the lowest cost across different task sizes. This advantage is from the model's ability to incorporate future task size predictions into its offloading decisions. By anticipating the computational demands of upcoming tasks, the GA can strategically choose between local processing and offloading to remote resources.

4.4 Impact of Number of MDs:

This experiment explores how the number of concurrent MD simulations affects system cost (processing time and energy consumption). We evaluated performance with varying numbers of MDs, ranging from 5 to 25 ([3-4] Mbits task size) simulations running simultaneously. The results are presented in Figure 5.

As the number of MDs increases, we anticipate a rise in both total energy consumption and processing time. This is because managing multiple simulations simultaneously places a higher burden on computational resources. Again, the proposed prediction-based GA demonstrates its effectiveness in handling this growing workload.

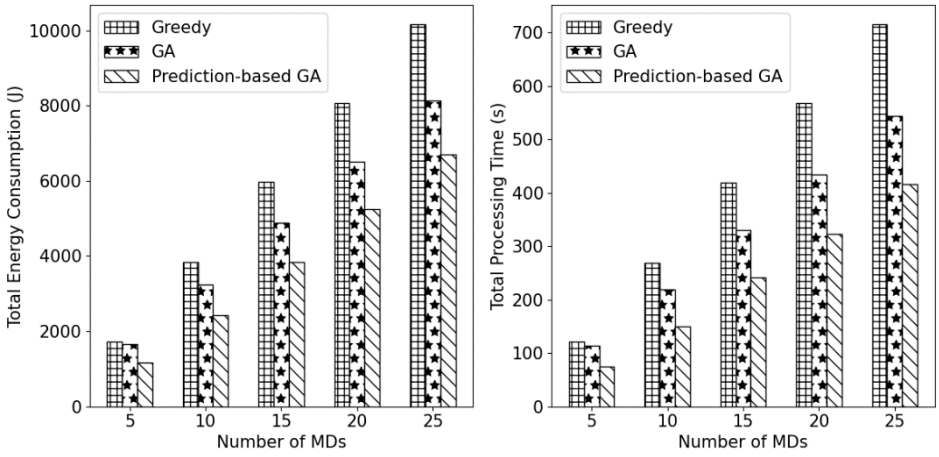


Figure 5. Effect of different number of MDs

5. CONCLUSION

In this study, we proposed an innovative approach to enhance task offloading in mobile edge computing systems. Our contributions address critical limitations in existing methods and pave the way for more efficient resource utilization. Our predictive task size method, leveraging recurrent neural networks (RNNs), provides foresight into future task requirements. By considering upcoming computational demands, we enable more informed offloading decisions. Additionally, our GA-based optimization fine-tunes parameters, aiming to minimize energy consumption while improving overall system performance. Experimental validation using real-world data underscores the effectiveness of our approach. However, further research is needed to explore advanced prediction models and energy-aware strategies. By bridging the gap between immediate decisions and future task requirements, our work contributes to the ongoing evolution of MEC systems.

ACKNOWLEDGEMENT

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant 2021R1I1A3051364.

REFERENCES

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys and Tutorials.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017. DOI: doi.org/10.1109/COMST.2017.2745201

[2] M. T. Beck, M. Werner, S. Feld, and T. Schimper, "Mobile edge computing: A taxonomy," in *Proc. 6th International Conference on Advances in Future Internet*, pp. 48–54, 2014.

[3] X. Lyu, H. Tian, P. Zhang, and C. Sengul, "Multi-user joint task offloading and resources optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.
DOI: doi.org/10.1109/TVT.2016.2593486

[4] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, Jan. 2019.
DOI: doi.org/10.1109/TVT.2018.2881191

[5] L. Tan, Z. Kuang, L. Zhao, and A. Liu, "Energy-Efficient Joint Task Offloading and Resource Allocation in OFDMA-Based Collaborative Edge Computing," *IEEE Transactions on Wireless Communications*, vol. 21, no. 3, pp. 1960-1972, Mar. 2022.
DOI: doi.org/10.1109/TWC.2021.3108641

[6] T. W. Jung, J. Y. Lee and K. D. Jung, "Traffic-based reinforcement learning with neural network algorithm in fog computing environment," *International journal of internet, broadcasting and communication: IJIBC*, vol. 12, no. 1, pp.144-150, 2020.

[7] H. Yuan, G. Tang, X. Li, D. Guo, L. Luo, and X. Luo, "Online dispatching and fair scheduling of edge computing tasks: A learning-based approach," *IEEE Internet of Things Journal*, 8(19), pp.14985-14998, 2021.
DOI: doi.org/10.1109/JIOT.2021.3073034

[8] E. Dahlman, S. Parkvall, and J. Skold, "4G: LTE/LTE-Advanced for Mobile Broadband," New York, NY, USA: Academic, 2013.

[9] M. Mitchell, "An Introduction to Genetic Algorithms," Cambridge, MA, USA: MIT Press, 1996.

[10] M. Tang and V.W. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Transactions on Mobile Computing*, 21(6), pp.1985-1997, 2020.
DOI: doi.org/10.1109/TMC.2020.3036871

[11] J. Chen, H. Xing, Z. Xiao, L. Xu, and T. Tao, "A DRL agent for jointly optimizing computation offloading and resource allocation in MEC," *IEEE Internet of Things Journal*, 8(24), pp.17508-17524, 2021.
DOI: doi.org/10.1109/JIOT.2021.3081694