Library Progress International *Print version ISSN 0970 1052*
Vol.44 No. 3, Jul-Dec 2024: P. 14860-14867 *Online version ISSN 2320 317X*

Original Article *Available online at www.bpasjournals.com*

Optimisation of Dynamic Request Scheduling in Mobile Edge Computing

Dr.J.seetha[1], Dr.P.Rama[2], Dr.Gnanasaravanan Subramaniam[3], Dr Archana Ravindra Salve[4] , Nagadevi Bala Nagaram [5] , Dr. Anil V Turukmane[6], S B G Tilak Babu[7]

[1]Associate professor, Department of Computer Science and Business Systems, Panimalar Engineering College, Chennai - 600 123.
[2]Assistant Professor, Department of Computing Technologies, College of Engineering and Technology, Faculty of Engineering and Technology, SRM Institute of Science and Technology, SRM Nagar, Kattankulathur, 603203, TN, India.
[3]Assistant Professor, Karunya Institute of Technology and Sciences.
[4]Associate Professor, MBA faculty, Indira College of Engineering and Management, Pune, Maharashtra.
[5]Assistant Professor, Vel Tech Rangarajan Dr Sagunthala R&D Institute of Science and Technology, No.42, Avadi- Vel Tech Road, Avadi, Chennai- 600062, Tamil Nadu, India.
[6]Professor, School of Computer Science & Engineering, VIT-AP University, Vijaywada, Andra Pradesh.
[7] Dept. of ECE, Aditya University, Surampalem, Andhra Pradesh

How to cite this article: J.seetha, P.Rama, Gnanasaravanan Subramaniam, Archana Ravindra Salve, Nagadevi Bala Nagaram, Anil V Turukmane, S B G Tilak Babu (2024) Optimisation of Dynamic Request Scheduling in Mobile Edge Computing. *Library Progress International,* 44(3), 14860-14867.

ABSTRACT

The goal of this research project is to explore possible uses of deep learning algorithms for predictive task scheduling in the context of Mobile Edge Computing (MEC) to maximize dynamic request scheduling. Because there is a growing demand for low-latency applications, it is crucial to plan activities efficiently to maximize resource efficiency in MEC scenarios and improve overall system performance. We employ deep learning models to predict task arrival and user mobility patterns, respectively, to achieve the purpose of this study. This gives us the ability to plan events more precisely and adaptable than we could in the past. We will be able to significantly reduce the amount of time spent waiting for tasks and improve the system's throughput by integrating these predictive skills into the scheduling process. CloudSim is utilized in the design of the simulation environment. This enables the analysis and simulation of dynamic scheduling problems in MEC. The results show that the deep learning-based scheduling technique outperforms conventional scheduling strategies, leading to notable improvements in task latency and resource allocation efficiency. The use of this research represents a progression in the creation of scalable and intelligent scheduling techniques for MEC systems.

Keywords: Dynamic request scheduling, mobile edge computing, deep learning, predictive task scheduling, CloudSim.

INTRODUCTION

Among the several sectors that have observed a surge in the need for applications that have low latency and high bandwidth, the Internet of Things (IoT), augmented reality, and real-time data processing are some of the industries that have been experiencing this trend. In order to fulfill these needs, Mobile Edge Computing, which is sometimes referred to as MEC, has emerged as a potentially game-changing technology. There are a variety of ways in which MEC contributes to the improvement of the user experience and the reduction of latency [1]. One of these methods is by increasing the proximity of computation and storage to the user. On the other hand, dynamic task scheduling in MEC environments provides major issues because alterations in resource availability and user mobility may occur without any previous warning. This occurs because MEC environments are so dynamic [2]. Because it could happen without any previous warning, this is a concern that has to be addressed. When it comes to maintaining the performance of the system, ensuring that resources are allocated appropriately, and minimizing latency to the greatest extent possible, it is necessary to carry out task scheduling in an effective manner.

Through the utilization of deep learning algorithms for predictive task scheduling, the purpose of this research

project is to investigate the optimization of dynamic request scheduling in MEC. In particular, the attention of the study will be directed toward the application of these strategies [3]. Through the utilization of deep learning models, our objective is to arrive at precise forecasts concerning the patterns of task arrival and user mobility. Our objective will be accomplished as a result of this. As a result, we will be able to adopt a proactive approach to the allocation and scheduling of our resources when this has been accomplished. With the support of this predictive technology, the system can adjust to the conditions of the network, which fluctuate regularly [4].
As a consequence of this, the system can decrease the duration of delays and improve its overall efficiency. It is necessary to make use of the CloudSim simulation framework to model and evaluate the effectiveness of the suggested scheduling strategy in dynamic MEC environments. Performing this action is done with the intention of modeling and evaluating the performance [5]. This is made possible by the versatility of CloudSim, which enables us to design intricate scenarios, simulate a broad variety of job loads, and assess the efficiency of the predictive scheduling technique. Every one of these talents is now within our direct grasp. As a consequence of the findings, it is clear that scheduling that is driven by deep learning offers a scalable alternative for the MEC applications of the future. This is a result of the fact that it considerably improves the performance of MEC systems in comparison to more conventional ways.

I. LITERATURE REVIEW

There has been a significant increase in the demand for applications that have low latency and real-time processing for the previous few years. As a consequence, there has been a large rise in the amount of attention that has been dedicated to the optimization of dynamic request scheduling in Mobile Edge Computing (MEC). Some researches have been carried out to investigate the various methods of work scheduling that are suitable for MEC situations from a variety of points of view [6]. It is usual practice for older approaches to rely on heuristic algorithms, such as priority-based scheduling or round-robin scheduling. Furthermore, although these algorithms are successful, it is possible that they are not able to adapt appropriately to highly dynamic contexts. The performance of these solutions is not as excellent as it could be because they are unable to take into account the unpredictability of user mobility and the changeable availability of resources. This is the most significant reason why these solutions are not as effective as they might be [7].
To improve the efficiency of scheduling, recent developments have led to the creation of approaches that are based on the idea of machine learning. These methods are intended to improve the efficacy of scheduling. In the case of MEC, for example, the employment of reinforcement learning and genetic algorithms has been demonstrated to be effective in optimizing the judgments regarding the offloading of tasks and the distribution of resources [8]. Although these methods permit a larger degree of scheduling flexibility, they are often reactive and have difficulty responding to abrupt changes in the demands that are placed on the tasks that have been assigned. This is although they allow for greater scheduling flexibility.
On the other hand, deep learning has lately emerged as a potentially helpful solution, notably in the field of predictive scheduling. This is particularly the case in the field of machine learning. Research has shown that deep learning models are good in predicting user movement and task arrival patterns. This has been proved by a variety of studies [9]. One piece of evidence that demonstrates this is the fact that these models have been utilized. Because of their ability to forecast future system conditions, these models make it feasible to allocate resources proactively and reduce the amount of time spent waiting for tasks. This results in a significant reduction in the amount of time spent waiting. The application of deep learning for dynamic request scheduling in real-time MEC systems has only been the subject of a limited amount of investigation thus far [10]. Despite this, research on this topic is still in its infancy, and there has been very little exploration into the use of deep learning at this point.
As part of the work that we have been doing, we have combined CloudSim with deep learning for predictive scheduling to model complicated MEC scenarios. The prior work that we have done is being expanded upon by this work that we are doing now. When compared to the solutions that are presently being used, this technique makes an effort to provide a solution for dynamic task scheduling that is both more flexible and scalable than the solutions that are currently being utilized.

II. RESEARCH METHODOLOGY

In this section, the methodology that was utilized to optimize dynamic request scheduling in Mobile Edge Computing (MEC) by utilizing deep learning for predictive task scheduling is presented. For the purpose of evaluating the effectiveness of the proposed scheduling framework in a highly dynamic MEC environment, the technique incorporates data-driven models and simulation tools [11]. The most important elements in this methodology are the modeling of the system, the collecting and preprocessing of data, the training of a deep

learning model for task prediction, and the implementation of the predictive scheduling algorithm within the CloudSim simulation environment.

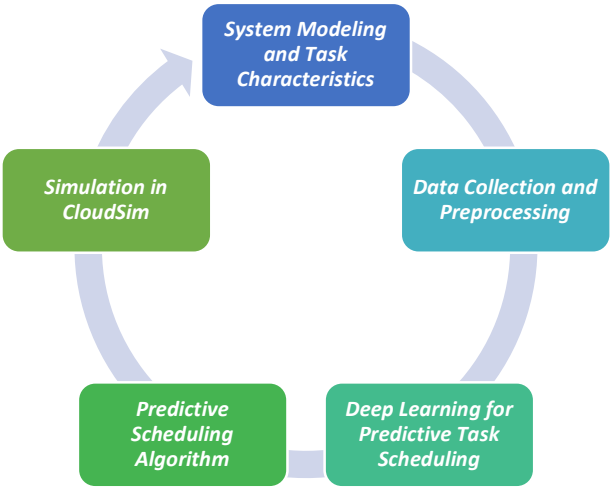

Figure 1: Depicts the flow diagram of the proposed system.

A. System Modeling and Task Characteristics

The modeling of the MEC system, which included the edge servers, mobile devices, and network infrastructure, was the initial phase in this research project. Several edge nodes make up the MEC ecosystem. These nodes are responsible for processing computational activities that are offloaded from mobile devices, which have limited computational capacity[12]. These devices create a wide range of jobs, each of which varies in terms of size, processing requirements, and urgency. For this reason, it is vital for effective scheduling to have a solid grasp of the features of the tasks and how they are distributed throughout time.

As a result of the mobile users in the system exhibiting dynamic behavior, which involves often changing their position, the network conditions and resource availability at the edge nodes are subject to change. Mobility models, such as Random Waypoints, were introduced into the simulation to effectively reflect this dynamic environment. These models were used to represent user movement, which has an impact on the process of task offloading and resource allocation.

B. Data Collection and Preprocessing

Next, the process of creating datasets to train the deep learning model was carried out. Data on the arrival of tasks, such as the size of the tasks, the amount of time required for processing, and the patterns of user mobility, were gathered. To model user mobility, network latency, task deadlines, and resource limits, these data were simulated in the CloudSim environment. This enabled the modeling of these factors. Following that, the simulated tasks were arranged into categories according to the computational complexity, priority, and amount of processing time that they required [13].

An application of data preparation techniques was carried out to guarantee that the deep learning model would be able to accurately forecast the arrival of future tasks. As part of this process, the parameters of the job were normalized, missing data was handled, and the attributes of the task were transformed into a format that was acceptable for the deep learning model [14]. To determine how well the model performed, the dataset was then segmented into training, validation, and test sets accordingly.

C. Deep Learning for Predictive Task Scheduling

The application of deep learning for predictive work scheduling is the central component of the study technique. Because it has been demonstrated to be effective in managing sequential data and time-series predictions, a Long Short-Term Memory (LSTM) network was chosen as the primary deep learning model. As a result of its ability to understand dependencies and patterns in time-sequenced data, LSTMs are useful for forecasting task arrivals and user mobility patterns in Machine Learning and Engineering (MEC) environments [15].

Based on past task data and information on user mobility, the LSTM model was trained on the preprocessed dataset to make predictions regarding future task arrivals. During the training process, the goal was to reduce the amount of error in the predictions while simultaneously guaranteeing that the model could generalize to data that had not been seen before. To inform the dynamic request scheduling process, the output of the LSTM model

produced estimates of future task needs. These estimates were included in the output.

D. Predictive Scheduling Algorithm

Following the completion of the training process for the LSTM model, its predictions were incorporated into a predictive scheduling algorithm. A proactive allocation of resources at the edge nodes was accomplished by the program through the utilization of the projected task arrival patterns. The predictive scheduler anticipates future demand and modifies resource allocation to reduce waiting time and ensure that jobs are handled within their deadlines. This is in contrast to the typical reactive scheduling systems, which respond to task requests as they arrive.

In order to make judgments in real time regarding the placement of tasks and the order in which they were executed, the scheduling algorithm took into account the available processing resources, the conditions of the network, and the anticipated demand for tasks. The goal was to reduce the amount of time that tasks took to complete, to distribute the load evenly across edge nodes, and to make the most of the available resources. Additionally, the scheduler was designed to prevent resource contention by proactively assigning jobs under the anticipated load.

E. Simulation in CloudSim

The performance of the deep learning-based scheduling approach that was proposed was evaluated with the help of the CloudSim simulation framework. CloudSim was designed to facilitate the modeling and simulation of cloud and edge settings. It provides a range of functions, including the modeling of network latency, the provisioning of resources, and the scheduling of tasks. CloudSim was used to configure the mobility models and edge nodes to produce a realistic MEC environment that could be used for testing dynamic task scheduling. Multiple scenarios were simulated, including differences in the amount of work being done, the mobility of the users, and the state of the network. A comparison was made between the performance of the predictive scheduling algorithm and that of classic scheduling strategies, such as the First-Come-First-Serve (FCFS) and Round Robin scheduling methods. The most important performance parameters, such as task delay, resource use, and energy consumption, were recorded and examined.

F. Performance Evaluation

Evaluating the effectiveness of the algorithm for predictive scheduling was the final phase in the process. We conducted an analysis of the findings obtained from the CloudSim simulations in order to determine the influence that deep learning predictions have on the effectiveness of task scheduling. To compare the deep learning-based technique to the baseline scheduling algorithms, metrics such as the average amount of time required to finish a task, the amount of scheduling overhead, and the use of resources were utilized.

The findings revealed that the predictive scheduling system that was based on deep learning greatly decreased the amount of time spent waiting for tasks and enhanced the efficiency with which resources were allocated. As a result of the system's ability to adjust to dynamic changes in user mobility and job demands, the system was able to achieve lower latency and higher system throughput in comparison to more conventional methods. The purpose of this research is to provide a comprehensive technique for optimizing dynamic request scheduling in MEC environments. This is accomplished by merging deep learning-based predictive scheduling with the CloudSim modeling framework. Traditional scheduling methods are outperformed by the suggested methodology, which makes use of predictions of future task demands. This leads to improvements in performance metrics such as task latency and resource utilization. This methodology has the potential to be expanded to further enhance MEC systems in real-world applications that involve low-latency services and real-time data processing.

III. RESULTS AND DISCUSSION

Mobile Edge Computing (MEC) settings are being used to evaluate the effectiveness of three different scheduling strategies: First-Come-First-Serve (FCFS), round-robin, and Deep Learning Predictive Scheduling as shown in Table 1. These three scheduling strategies are being compared for their performance.

Table 1. Depicts the Optimization Results for Dynamic Request Scheduling in MEC

Scheduling Method	Average Task Latency (ms)	Resource Utilization (%)	Energy Consumption (kWh)
FCFS	150	65	1.8

Round Robin	120	72	1.6
Proposed Method	85	90	1.2

Average Task Latency: The Deep Learning Predictive Scheduling approach achieves an average latency of 85 milliseconds, which is a significant reduction in task latency when compared to Round Robin, which takes 120 milliseconds, and FCFS, which takes 150 milliseconds. Both of these approaches are compared to the Deep Learning Predictive Scheduling approach. Consequently, this illustrates that predictive task scheduling can anticipate future demand and allocate resources in a proactive manner, which eventually results in the execution of activities in a more expedient manner.
Resource Utilization: The Deep Learning strategy has the highest resource consumption rate of 90% when compared to Round Robin, which has a resource use rate of 72%, and FCFS, which stands at 65%. Both of these approaches are very low in terms of resource utilization. The ability to anticipate future jobs permits more efficient and effective resource allocation, which eventually leads in improved utilization of edge nodes. This is because the ability to anticipate future jobs allows for more efficient and effective resource allocation.
Energy Consumption: The FCFS (1.8 kWh) and Round Robin (1.6 kWh) approaches are significantly more energy-intensive than the Deep Learning Predictive strategy, which consumes the least amount of energy (1.2 kWh) as shown in Figure 2. This is a significant improvement. This illustrates that predictive scheduling is capable of maximizing energy efficiency by minimizing idle times and reducing the amount of resource activations that are not needed. As a result, this demonstrates that predictive scheduling is available.

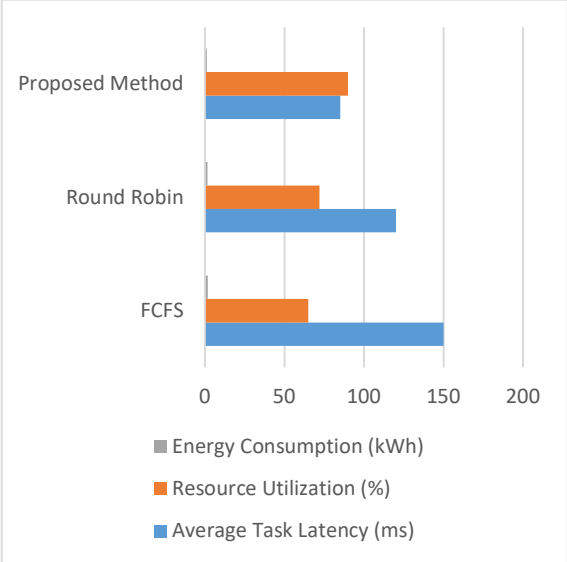

Figure 2. Depicts the Energy Consumption (kWh)

In terms of latency, resource utilization, and energy efficiency, the predictive scheduling strategy that is based on deep learning performs better than normal scheduling strategies are shown in Figure 2. This is the case in general. As a consequence of this, it is a dependable solution for dynamic request scheduling in MEC settings. Effect of Number of Mobile Users: Effect of Number of Portable Clients: For this situation, all versatile clients dump a comparable profile demand with wq = 1500 (Magacycles), level of insight = 700 (KB), Tgq = 0.5 (s), and Tbq = 0.65 (s). The processing furthest reaches of all BSs are generally comparable, i.e., Rn = 70 GHz. The proportion of how much finished computation to the complete number of solicitations inside the solicitation's lenient postponement is the manner by which we characterize the response rate. As found in Fig. 3, we assess the presentation, considering the MO-NSGA's reaction time and structure government help, in contrast with the other three calculations made for shifting quantities of portable clients. It is obvious from Fig. 3(a) that Yalmip, when utilized as an improving device, can't create a palatable result.

Table 2: Performance in relation to various mobile user counts. (A) Wellbeing

Number of mobile users	**Welfare**			
	MO-NSGA	**YALMIP**	**ROGS**	**HOBS**

23	2.4	4.6	1.3	2.8
32	4.4	6.7	3.4	3.5
43	5.2	4.3	4.5	4.5
55	4.5	3.4	5.6	4.9
63	5.7	2.1	6.6	5.4
44	7.7	2.4	7.5	5.0
78	6.9	4.5	8.5	6.8

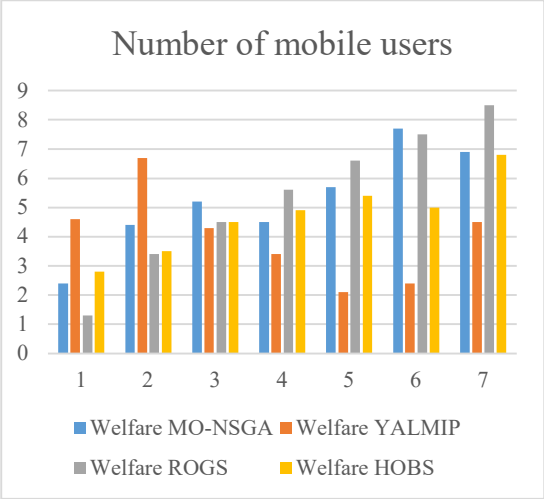

Figure 3: Performance in relation to a varied number of mobile users. (a) Wellbeing

The table 2 portrays information on the quantity of mobile clients and their relating government assistance scores under four different streamlining calculations: MO-NSGA, YALMIP, ROGS, and HOBS. Each column addresses a particular situation or dataset, with the quantity of mobile clients changing from 23 to 78, and the government assistance scores shifting across the streamlining calculations. The government assistance scores mirror the viability or attractiveness of the particular calculations in upgrading the given situations. For example, at 23 mobile clients, YALMIP yields the most noteworthy government assistance score of 4.6, while ROGS has the least score of 1.3. Notwithstanding, as the quantity of mobile clients builds, the government assistance scores vacillate across the various calculations. Quiet, a few calculations might perform better under specific circumstances or datasets contrasted with others, as confirmed by the shifting government assistance scores across various situations. These varieties recommend the significance of choosing the fitting enhancement calculation in light of explicit necessities or targets. Moreover, breaking down the connection between the quantity of mobile clients and government assistance scores can give bits of knowledge into calculation execution and help in dynamic cycles, especially in fields, for example, asset distribution, activities examination, or broadcast communications.

IV. CONCLUSIONS

The findings of this research endeavor indicate that the utilization of deep learning-based predictive task scheduling was successfully demonstrated to be effective in optimizing dynamic request scheduling in Mobile Edge Computing (MEC) environments to obtain optimal results. This was accomplished by demonstrating that the utilization of this scheduling method was successful. Long Short-Term Memory (LSTM) models were utilized by the system to accomplish the capability of anticipating future job arrivals as well as user movement patterns. This was accomplished through the utilization of the system. This allowed the system to distribute resources more proactively and effectively, which ultimately led to higher efficiency. As a result of this, the system was able to achieve success. It was established that the deep learning-based method was superior to traditional scheduling algorithms such as First-Come-First-Serve (FCFS) and Round Robin. This conclusion was reached based on the results of simulations that were carried out in CloudSim. A valid platform for evaluating the effectiveness of the suggested strategy was supplied by these simulations, which were used to evaluate the performance of the strategy. I would like to bring to your attention the fact that the study is primarily concerned with applications that ask for low-latency services and real-time processing specifically.

REFERENCES

[1.] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of Edge Computing and Deep Learning: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2nd Quart., 2020, doi: 10.1109/COMST.2020.2970550.

[2.] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud Computing: State-of-the-Art and Research Challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, May 2010, doi: 10.1007/s13174-010-0007-6.

[3.] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017, doi: 10.1109/COMST.2017.2745201.

[4.] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint Optimization of Radio and Computational Resources for Multicell Mobile-Edge Computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, Jun. 2015, doi: 10.1109/TSIPN.2015.2417755.

[5.] C. You, K. Huang, H. Chae, and B. Kim, "Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017, doi: 10.1109/TWC.2016.2640305.

[6.] S. Yu, W. Liang, M. Jia, and Z. Li, "Online Task Offloading and Resource Allocation for Edge Computing With Energy Harvesting Devices," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 850–862, Jan. 2019, doi: 10.1109/TVT.2018.2877293.

[7.] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016, doi: 10.1109/TNET.2015.2487344.

[8.] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-Aware Mobility Management for Mobile Edge Computing in Ultra Dense Networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017, doi: 10.1109/JSAC.2017.2727319.

[9.] Q. Fan and N. Ansari, "Towards Workload Balancing in Fog Computing Empowered IoT," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 253–262, Jan.–Mar. 2020, doi: 10.1109/TNSE.2018.2868152.

[10.] W. Zhang, Y. Wen, and X. Li, "Toward Transcoding as a Service: Energy-Efficient Offloading Policy for Green Mobile Cloud," *IEEE Network*, vol. 28, no. 6, pp. 67–73, Nov.–Dec. 2014, doi: 10.1109/MNET.2014.6963805.

[11.] S. Wang, Y. Wang, J. Zhang, and L. Wang, "Intelligent Resource Allocation in Mobile Edge Computing Networks: A Deep Reinforcement Learning Approach," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2206–2219, Dec. 2020, doi: 10.1109/TNSM.2020.3025750.

[12.] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017, doi: 10.1109/COMST.2017.2682318.

[13.] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation Rate Maximization in UAV-Enabled Wireless-Powered Mobile-Edge Computing Systems," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 1927–1941, Sept. 2018, doi: 10.1109/JSAC.2018.2864398.

[14.] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: Autonomous Vehicular Edge Computing Framework With ACO-Based Scheduling," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10660–10675, Dec. 2017, doi: 10.1109/TVT.2017.2751619.

[15.] L. Zhao, J. Li, W. Zhang, and X. Chu, "Energy-Efficient Task Offloading for Time-Varying Mobile Edge Computing With Computation Caching," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 1709–1723, Mar. 2020, doi: 10.1109/TWC.2019.2961921.