Collaborative Task Offloading with Digital Twin in Multi-Vehicle and Multi-Edge Environments

Anqi Gu*, Huaming Wu*, Yixiao Wang*, Ruidong Li† and Chaogang Tang‡
*Center for Applied Mathematics, Tianjin University, Tianjin 300072, China
†Institute of Science and Engineering, Kanazawa University, Kanazawa 920-1192, Japan
‡School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China
Emails: {guanqi9_, whming, wang_yixiao}@tju.edu.cn, liruidong@ieee.org, cgtang@cumt.edu.cn

GLOBECOM 2024 - 2024 IEEE Global Communications Conference | 979-8-3503-5125-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/GLOBECOM52923.2024.10901293

***Abstract*—In recent years, the effective utilization of edge servers to assist vehicles in handling compute-intensive and latency-sensitive tasks has emerged as a pivotal concern in Vehicular Edge Computing (VEC). In this paper, we adopt a cooperative approach that leverages the collective capabilities of multiple edge servers. This strategy is designed to effectively manage tasks and alleviate the computational burden imposed on these servers. Specifically, Graph Neural Network (GNN) is applied to extract and classify features such as the geographical locations and communication statuses of multiple edge servers, enabling the selection of the most suitable servers for collaborative task execution. We have utilized solar energy for local computing, effectively achieving environmental protection and reducing the local energy burden on vehicles. Moreover, a novel edge attraction formula is defined to refine the rationality of clustering. In addition, Deep Reinforcement Learning (DRL) is employed to make real-time offloading decisions. To ensure experimental accuracy while mitigating costs, we establish a corresponding digital twin environment to acquire experimental data. By conducting a comparative analysis against three other baseline methods, we effectively reduce task completion time and thus meet the stringent demands of time-sensitive tasks.**

***Index Terms*—Vehicular edge computing, Internet of vehicles, Digital twin, Task offloading, Graph neural network, Deep reinforcement learning**

I. Introduction

In the Internet of Vehicles (IoV) environments, many real-time processing challenges arise, including tasks such as obstacle detection and fault handling [1]. On the one hand, relying solely on the vehicle's onboard CPU for these tasks can be impractical in terms of timely execution. On the other hand, cloud servers may incur high transmission latency due to their distant locations. Therefore, the concept of offloading tasks from vehicles to edge servers located near the road has emerged as a promising solution [2]. Given the dynamic nature of vehicles operating in a real-time environment, the development of effective task-offloading strategies has become a central focus of research.

To address this issue, Wang *et al.* [3] proposed a meta-learning-based adaptive approach for task offloading decisions. This approach allows for partial task offloading decisions considering the task's inherent topology to minimize the task completion time. However, the data employed lacks sufficient representatives of real-world scenarios. Cao *et al.* [4] proposed multi-objective task offloading models for vehicles within corresponding digital twin environments, which involves simulating vehicle movement and significantly reducing the costs associated with practical experiments. On actual roads, relying on a single edge server for tasks often results in high latency when handling a large volume of vehicle tasks.

In [5], Multi-Agent Deep Deterministic Policy Gradient (MADDPG) was employed to make task offloading decisions and optimize resources for vehicles with varying speed and latency requirements, achieving commendable performance. However, it's worth noting that traditional Deep Reinforcement Learning (DRL) lacks the inherent capability to account for concealed or hidden features within the system. Instead, researchers have incorporated Graph Neural Networks (GNNs) into task offloading decisions to handle the graphical features of the system. For instance, GNN-TSO [6] leverages GNN to capture the inherent characteristics of task applications, thereby enabling more precise decision-making and reducing task latency. However, this approach does not consider the potential collaboration among edge servers. Some recent studies [7, 8] integrated GNN to explore the system status of MEC servers and mobile devices, resulting in more accurate and efficient task decisions. Similarly, Wu *et al.* [9] combined DRL and GNN to explore task diversity and minimize task completion time. Unfortunately, these approaches have not yet been expanded into the domain of vehicular networking. Furthermore, they do not encompass the intricacies of task offloading in scenarios involving multiple MEC servers. To improve the accuracy of offloading decision-making and multi-edge collaborative computing, extracting and classifying features such as geographical location, communication status, and computing power of multiple edges has become a focus. The information of multiple edges can be abstracted into a graph. This paper introduces an innovative approach by utilizing GNNs to extract and classify features of these edges, leveraging them for multi-edge collaboration to address the issue of insufficient edge computing power.

To address these challenges, we introduce a novel optimization approach that improves decision-making for task offloading in complex collaborative scenarios by leveraging the strengths of GNNs and DRL. Our method effectively reduces prolonged task processing times and high energy consumption during vehicle operations, while alleviating the high latency caused by resource shortages in edge computing

through the innovative concept of "edge gravity". In addition, we integrate clean energy into the vehicle's local computing power, reducing the vehicle's energy consumption and achieving overall energy savings. The main contributions can be summarized as follows:

- We alleviate computational stress through a novel approach to multilateral collaborative computing by designing an edge attraction formula. This approach reduces task latency by efficiently scheduling computing power across multiple edges.
- We propose an innovative solution involving using clean solar energy to charge vehicles and serve as a local computing power source, significantly reducing local computing energy consumption.
- We design a novel approach by integrating GNN with DRL and digital twin within the domain of Vehicular Edge Computing (VEC). Simulating in a virtual environment lowers experimentation costs and real-world risks while offering a realistic and accurate setting for training and testing autonomous driving algorithms.

II. System Model and Problem Formulation

A. System Model

Fig. 1 illustrates the network scenario involving vehicle-to-vehicle cooperation. The scenario consists of N vehicles and M edge servers, where each vehicle generates a task during every time slot. The communication model employed in this study encompasses two critical types of communication: vehicle-to-vehicle (V2V) communication and vehicle-to-infrastructure (V2I) communication [10]. It is important to note that communication between base stations is conducted via remote channels using microwave transmission lines, which offer low signal loss, high transmission speeds, and strong anti-interference capabilities. Consequently, the information transmission delay between base stations is significantly lower than the vehicle-to-edge delay.

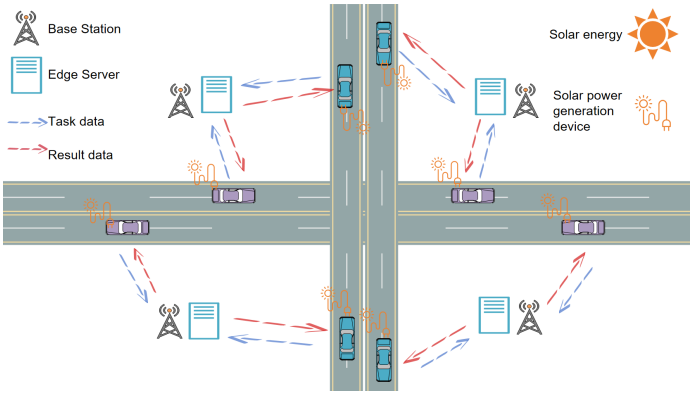

Fig. 1. System model

The vehicles are equipped with antennas for transmitting and receiving data and photovoltaic panels and absorbers to harvest energy for local task execution. These vehicles upload task-related information to the edge servers for decision-making, which involves determining whether the task should be executed locally on the vehicle or offloaded to the edge for execution. Meanwhile, Roadside Units (RSUs) can cooperate. If the offloaded RSU lacks the computational resources required for task execution, it can transfer the task to other available edge servers for processing. The decision to offload tasks is represented by the variable $x_{i,j}$, where i corresponds to each of the N vehicles. When $x_{i,j}$ equals 1, the task is offloaded to the nearby j-th edge server, while a value of 0 indicates that the task is executed locally. It is crucial to emphasize that tasks are considered indivisible by default.

1) Local Computing: In the case where we opt for local task computation, denoted by $x_{i,j} = 0$, the completion time is solely determined by the execution time, as expressed by:

$$T_i^{loc} = \frac{r_i^t}{f_i^t}, \tag{1}$$

where r_i^t denotes the computational workload required for the task of the i-th vehicle at time t, representing the number of CPU cycles required to complete the task, and f_i^t denotes the computational capacity of the i-th vehicle, expressed in the number of CPU cycles per second.

2) Edge Computing: When a vehicle is offloaded for computation at the edge, the data needs to be transferred to the edge server first, and then the computation is performed by the edge server. The transmission time can be calculated by:

$$T_{i,j}^{tr} = \frac{D_i}{R_{i,j}^{tr}}, \tag{2}$$

where D_i refers to the size of the data that the i-th vehicle needs to upload, and $R_{i,j}^{tr}$ refers to the transmission rate at which the i-th vehicle transfers data from its local device to the j-th edge server. Using Shannon's formula [11], the transmission rate can be calculated by:

$$R_{i,j}^{tr} = B_{i,j}^t \log\left(1 + \frac{p_i g_{i,j}}{\sigma^2}\right), \tag{3}$$

where $B_{i,j}^t$ represents the channel transmission bandwidth between vehicle i and the j-th edge server, while pi denotes the transmission power of the i-th vehicle. The wireless gain is indicated by gi, j, and σ corresponds to the power of Gaussian white noise. The channel operates under a Rayleigh fading model. We define the transmission time between the j-th and k-th edge servers as $T_{j,k}^{tr}$. In scenarios where the computing power of the j-th edge server is insufficient, tasks are offloaded to the k-th edge server. The calculation method for this offloading time is similar to that used for $T_{i,j}^{tr}$.

The computation time of the i-th task on the j-th server can be formulated as:

$$T_{i,j}^{c} = \frac{r_i^t}{f_j^t}, \tag{4}$$

where f_j^t represents the computational capacity of the j-th server, which corresponds to its CPU speed.

Therefore, the total time required for task completion in edge computing mode is T_i^e.

$$T_i^e = T_{i,j}^{tr} + T_{i,j}^{c} + T_{j,k}^{tr}. \tag{5}$$

3) Local Energy Consumption: To minimize energy consumption during task computation on the vehicle and adhere to the principles of energy conservation and emission reduction, we employ solar power generation to provide energy support for these tasks. Solar panels and absorbers will be installed on the vehicle to convert solar energy. The electrical energy derived from solar energy collected by the vehicle at time t is represented as [12]:

$$E_i^{t,u} = a_s b_s c \exp\left(\frac{-k}{\cos(a_t^{solar})}\left(1 - 2.2556 \times 10^{-5} z\right)^{5.2561}\right), \tag{6}$$

where c is a constant with a value between 0 and 1, representing the efficiency of the solar absorber panel, a_s denotes the charging size, b_s is the constant power intensity of the solar beam, a_t^{solar} denotes the solar zenith angle at time t, z denotes the height of the vehicle roof, and $k > 0$ denotes the total amount of atmospheric sunlight.

When a task is executed locally, the associated energy consumption is denoted as:

$$E_i^{t,c} = \frac{r_i^t}{f_i^t} g_i^{cm}, \tag{7}$$

where g_i^{cm} represents the energy consumption level of the vehicle per second.

To achieve our energy-saving and emission-reduction objectives, the amount of solar energy collected for power generation must equal or exceed the computational energy consumed by the vehicle.

$$E_i^{t,u} - E_i^{t,c} \geq 0, \tag{8}$$

which ensures that the vehicle's computational needs are consistently met through solar energy generation

B. Problem Formulation

To achieve task-offloading decisions from a practical perspective, we formulate the problem as follows:

$$\mathcal{P}: \min_{x_{i,j}} T_{total} = x_{i,j}\left(T_{i,j}^{tr} + T_{i,j}^{c} + T_{j,k}^{tr}\right) + \left(1 - x_{i,j}\right)\left(T_i^c\right) \tag{9}$$

$$s.t. \quad E_i^{t,u} - E_i^{t,c} \geq 0, \tag{9a}$$

$$0 \leq f_i^t \leq f_i^{\max}, \tag{9b}$$

$$0 \leq f_j^t \leq f_j^{\max}, \tag{9c}$$

$$x_{i,j} \in \{0, 1\}, \tag{9d}$$

where Eq. (9a) imposes an energy constraint, ensuring that the energy consumption for local execution does not exceed the amount of clean energy collected. If this threshold is surpassed, the task must be offloaded to the edge server for execution. This ensures that the energy used for the vehicle's task execution remains within the limits of the collected clean energy, thereby achieving the objective of reducing the vehicle's overall energy consumption. Eqs. (9b) and (9c) impose constraints on computational capacity, ensuring that the computational load of each vehicle does not exceed its maximum capacity, and similarly, that the edge servers operate within their own capacity limits. Eq. (9d) defines the constraint on the decision variables, where the decision variables x_i in this study are binary, taking values of either 0 or 1. It can be seen that $\mathcal{P}$ is a mixed-integer 0-1 programming problem, a well-known NP-hard problem that is not easily solvable with traditional methods.

III. Proposed Approach

To tackle the above challenge, we decompose the problem into two main components. One involves the extraction and clustering of edge server features using GNN, which leads to creating a classification scheme for mutually collaborative servers, and the other one focuses on devising task offloading strategies while adhering to constraints, utilizing an enhanced DRL approach. The structure is shown in Fig. 2.

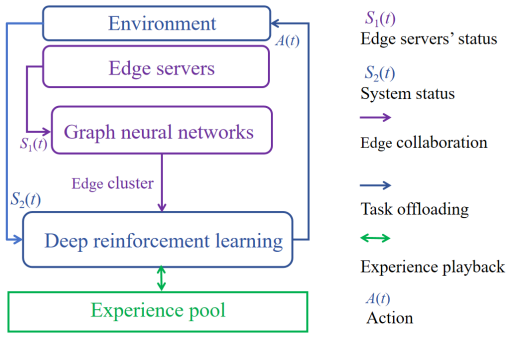

Fig. 2. The framework of multilateral edge collaborative offloading algorithm

A. GNN-Assisted Edge Collaboration

GNN is primarily employed for graph processing, feature extraction from graphs, and performing operations on them. In this paper, we treat the M edge servers as individual nodes, collectively forming the vertex set denoted as V. We assume the presence of directed edges between nodes, representing servers capable of communicating with each other. These edges collectively constitute the edge set, denoted as E. Moreover, we assign weights to these edges, represented as W. This results in the formation of the graph G that requires processing.

The node features are defined as the coordinates of the nodes. To comprehensively account for factors such as the communication environment, computing capabilities, and inter-server distances, we introduce an edge attraction formula, which enables us to calculate the edge attraction between nodes, and these values are subsequently utilized as the weights on the edges connecting the nodes.

$$F(j,k) = f_k^t \frac{B_{j,k}}{\left(dis\left(j,k\right)\right)^{\alpha}}, \tag{10}$$

where $F(j,k)$ represents the edge attraction between the j-th server and the k-th server, f_k^t represents the computing capability of the k-th edge server at time t, and $B_{j,k}$ represents the communication capability between the j-th and the k-th servers. Moreover, $dis\left(j,k\right)$ represents the Euclidean distance between the j-th server and the k-th server, with α serving as a weight hyperparameter. Eq. (10) resembles the universal

gravitation formula. A larger f_k indicates a stronger attraction between two servers, meaning server j is more likely to offload tasks to server k when their communication capability is higher, leading to positive feedback. Conversely, as the distance between servers increases, the attraction decreases, leading to higher transmission costs. The stronger the calculated attraction between two servers, the more likely tasks are to be offloaded between them. These attraction values are then used as input weights, enhancing the GNN's ability to capture latent information and improve clustering accuracy.

The detailed procedures for calculating edge attraction and performing GNN clustering are outlined in Alg. 1 and Alg. 2, respectively. In Alg. 2, the graph, node features, edges, and computed edge attraction weights are input into a multi-layer. This process yields learned node embeddings, which include details such as model architecture, embedding dimensions, and parameters for clustering. Finally, k-means clustering is applied to the node embeddings to generate the clustering results. We use the third-party library PyClustering to configure a custom distance function for k-means clustering based on edge gravity. Edge servers within the same cluster can then collaborate effectively and offload tasks to one another, enhancing synergy among the edge servers.

Algorithm 1 Edge gravity

Input: Central edge v_j and neighborhood $N(v_j)$, gravitational constant f_i^t, weight coefficient α, matrix B.
Output: Score, v_j and $N(v_j)$.

while $v_k \in N(v_j)$ **do**
  $dis(j,k) = \sqrt{(x_i - x_k)^2 - (y_i - y_k)^2}$;
  $F(j,k) = f_k^t \frac{B_{j,k}}{(dis(j,k))^\alpha}$;
end while
$F_{max} = 0$, $F_{min} = 0$;
while $v_k \in N(v_j)$ **do**
  $F_{max} = \max(\max F, F_{j,k})$;
  $F_{min} = \min(\min F, F_{j,k})$;
end while
while $v_k \in N(v_j)$ **do**
  $G_{j,k} = \frac{F_{j,k} - F_{min}}{F_{max} - F_{min}}$;
end while
return G matrix;

Algorithm 2 Edge clustering

Input: $g(v, \varepsilon, w)$, v(Vertex set, ε edge, edge weight G, node characteristics: (spatial coordinates)).

Create a GCN model and initialize the model network parameters;
Model=GCN model created ()
for $episode = 1$ to ep **do**
  Update coefficient $\nabla\tau(\theta)$;
  Calculate loss coefficient;
  Obtain node embedding for learning columns (model architecture, embedding dimensions, parameters for constructing clustering);
end for
Use k-means clustering to cluster node embedding, with $k = 2$ and the distance metric changed to edge weight;
Output: Clustering results;

B. DRL-Assisted Edge Offloading

Considering the significant mobility and uncertainty in vehicular environments, we leverage unsupervised learning with DRL to facilitate real-time decision-making in dynamically changing scenarios. We formulate the problem as a Markov Decision Process (MDP), where the definitions of state, action, and reward are as follows.

1) State: The state is defined by several key components, including the vehicle's current position, its communication status with the base station, the real-time task load of the vehicle, and the amount of energy absorbed and stored by the vehicle. The aim is to make accurate decisions based on the changes in task load and energy storage of the vehicle, considering the dynamic positioning and communication status with the base station. The system's state space, denoted as $S(t)$, includes various parameters such as the computing capacity of vehicles and Small Cell Base Station (SCBS), as well as the positions of the vehicles.

$$S(t) = \big\{f_1(t), f_2(t), \cdots, f_i(t), F_1(t), F_2(t), \cdots, F_j(t), p_1(t), p_2(t), \cdots, p_i(t)\big\}, \tag{11}$$

where $f_i(t)$, $F_j(t)$, and $p_i(t)$ denote the computing capacity of the vehicle and MEC, and the position of the vehicle, respectively.

2) Action: We set the action variable as the decision variable, and the action space, denoted as $A(t)$, comprises decisions on whether to retain tasks for local processing or offload them to other servers, expressed as:

$$A(t) = \{x_{1,1}(t), x_{1,2}(t), \cdots x_{1,j}(t), \cdots, x_{i,j}(t)\}, \tag{12}$$

where $x_{i,j}(t)$ represents the decision for the i^{th} task. Specifically, $x_{i,j}(t) = 0$ corresponds to local processing, while $x_{i,j}(t) = 1$ corresponds to offloading for MEC processing.

3) Reward: To minimize latency, the reward is defined as the negative value of latency. This choice is based on the principle that DRL algorithms tend to adjust in the direction of larger rewards.

$$R(t) = -T_{total}. \tag{13}$$

The details of the algorithmic process are shown in Alg. 3.

C. Computational Complexity

In Alg. 1, the traversal of nodes and their neighboring nodes entails the determination of edge attractions between each node and its neighbors. The complexity can be expressed as $\mathcal{O}(M^2)$. The complexity of Alg. 2 is determined by the number of iterations ep, and it amounts to $\mathcal{O}(ep)$. As for Alg. 3, it contains two loops: one in the first line and the other in the fourth line. These loops correspond to the number of episodes in the interaction between the agent and the environment and the number of times the vehicles are traversed, respectively. Thus, the complexity for Alg. 3 is $\mathcal{O}(PT)$.

IV. Performance Evaluation

A. Parameter Settings

We utilize PanoSim software to simulate the digital twin environment. In our setup, a 200×200 m^2 area contains five

Algorithm 3 Policy selection algorithms based on DRL

Input: Vehicle location, task volume, edge computing resources, vehicle computing resources.

for $episode = 1, 2, \cdots, P$ **do**
  Receive initial state $S(t)$;
  Initialize a random process for action;
  for $t = 1, 2, \cdots, T$ **do**
    Execute actions $A(t)$ and obtain the reward $R(t)$;
    Obtain the action $A(t)$, new state $S'(t)$ and reward $R(t)$;
    Store ($S(t)$, $A(t)$, $S'(t)$, $R(t)$) in replay buffer D;
    Sample a random mini-batch of samples (S_j, A_j, R_j, S_j') from D;
    Update the evaluation network by minimizing the loss function;
    Update target network using the sampled policy gradient at every 10 steps;
    Update the target network parameters: $\theta^{'} \leftarrow \delta\theta + (1-\delta)\,\theta^{'}$
  end for
end for

vehicles, with four edge servers positioned at each corner of the road. The vehicles are programmed to travel at a speed of 30 km/h. The number of CPUs required for the task is randomly distributed between 10 and 200. The computing capacities of the local vehicle and edge server follow a random distribution within the range of $[10, 100]$ GHz and $[500, 1000]$ GHz, respectively. The coverage radius of each edge server is 150 meters and the wireless communication parameters are set as follows: Gaussian white noise is $N_0 = -100$ dB, small fading factor is -5 dB, vehicle transmission power is 100 mW, and edge server communication bandwidth is 80 MHz [4].

Fig. 3. The constructed vehicular digital twin environment

The established digital twin environment, as shown in Fig. 3, offers real-time information about vehicle positions and their distances to the VEC server. Data collected from sensors and road test units is mapped into this digital twin, where it interacts with the proposed algorithms to determine optimal offloading decisions. Throughout this process, the digital twin provides precise inputs and environmental perception capabilities for intelligent driving algorithms by continuously collecting real-time sensor data and updating models. By simulating the vehicle's operating environment, the digital twin generates real-time data that serves as input for the algorithms and is displayed in graphical form, allowing users to monitor the vehicle's status in real time. This setup creates a realistic and accurate environment for algorithm training and testing.

B. Baselines

To compare the performance of our approach, we implement the following three baseline algorithms:

- **One Edge Computing (OEC)**: This approach involves task computation using a single server, without employing collaborative methods, and makes offloading decisions solely through reinforcement learning.
- **Full Local Computing (FLC)**: Tasks are processed entirely using local resources without offloading to edge servers.
- **Greedy-based Scheme (GBS)**: This approach uses the original greedy method for task offloading without any additional enhancements.

C. Convergence Performance

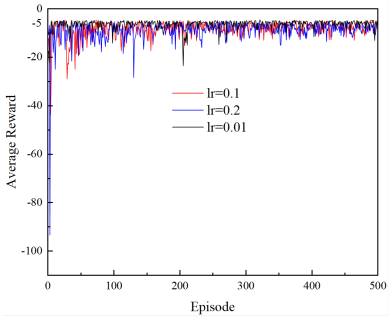

Fig. 4. Training rewards

As depicted in Fig. 4, a clear convergence trend becomes evident after 200 rounds. Notably, setting the learning rate to 0.1 results in faster convergence with reduced fluctuations. Hence, it is vital to carefully choose an appropriate learning rate, as excessively large or small values are undesirable.

D. Impact of Numbers of Vehicles

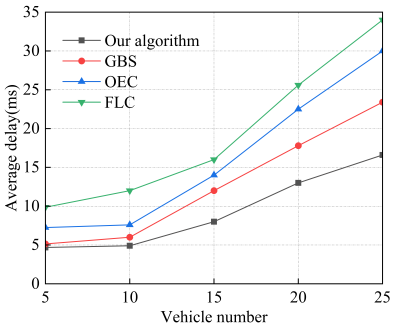

Fig. 5. Comparison of different schemes with varying numbers of vehicles

In Fig. 5, we can observe the performance differences among the three baselines and the proposed approach at varying numbers of vehicles. As the number of vehicles increases, the latency of all approaches also increases. This is because, with the increase in the number of vehicles, a higher volume of tasks is generated, resulting in a collective increase in the total time required to complete these tasks. With the growing number of vehicles, the waiting time for tasks on the edge server also increases significantly in the OEC scheme. This increase in waiting time contributes to a more substantial

performance gap when compared to the approach proposed in this paper. In the FLC scheme, all tasks are executed locally, and even with limited local computing capacity, it still takes more time to complete tasks than our proposed approach.

E. Impact of Numbers of MEC Servers

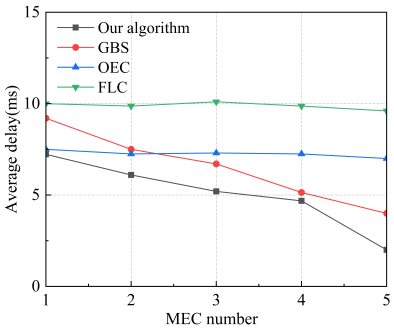

Fig. 6. Comparing different schemes with varying numbers of MEC servers.

In Fig. 6, the performance comparison of all approaches under different numbers of servers is illustrated. As the number of servers increases, the total latency of OEC and FLC remains constant. This is because a single server's local computation and task computation are independent of the number of servers. However, an increase in the number of servers means more coordinated server resources, resulting in a decrease in task completion latency and an increased gap compared to the OEC and FLC schemes. With an increasing number of servers, the proposed strategy can efficiently schedule more server resources, resulting in a more significant performance gap when compared to the GBS scheme.

V. Conclusion

This paper proposes a multi-edge coordinated task offloading scheme in a digital twin environment, introducing an edge attraction formula to calculate the attraction between different edge servers. It utilizes GNN to extract and integrate information from multiple edges, enabling the exploration of latent information, and then applies DRL to obtain real-time offloading decisions, ultimately reducing task latency. Additionally, using clean energy helps reduce the cost of local computing and contributes to environmental sustainability. The digital twin environment ensures the practical applicability of the model. Simulation results highlight the superior performance of this scheme compared to other comparative approaches. Future research will focus on expanding collaborative scenarios for vehicular edges, considering additional constraint conditions, and further enhancing optimization performance.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (No. 62071327) and the Tianjin Science and Technology Planning Project (No. 22ZYYYJC00020). Huaming Wu is the corresponding author.

References

[1] R. Zhang, L. Wu, S. Cao, D. Wu, and J. Li, "A vehicular task offloading method with eliminating redundant tasks in 5g hetnets," *IEEE Transactions on Network and Service Management*, vol. 20, no. 1, pp. 456–470, 2023.

[2] G. Perin, M. Berno, T. Erseghe, and M. Rossi, "Towards sustainable edge computing through renewable energy resources and online, distributed and predictive scheduling," *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 306–321, 2022.

[3] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast adaptive task offloading in edge computing based on meta reinforcement learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 242–253, 2021.

[4] B. Cao, Z. Li, X. Liu, Z. Lv, and H. He, "Mobility-aware multiobjective task offloading for vehicular edge computing in digital twin environment," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 10, pp. 3046–3055, 2023.

[5] X. Huang, L. He, and W. Zhang, "Vehicle speed aware computing task offloading and resource allocation based on multi-agent reinforcement learning in a vehicular edge computing network," in *2020 IEEE International Conference on Edge Computing (EDGE)*, 2020, pp. 1–8.

[6] Y. Bian, Y. Sun, M. Zhai, W. Wu, Z. Wang, and J. Zeng, "Dependency-aware task scheduling and offloading scheme based on graph neural network for mec-assisted network," in *2023 IEEE/CIC International Conference on Communications in China*, 2023, pp. 1–6.

[7] Z. Sun, Y. Mo, and C. Yu, "Graph-reinforcement-learning-based task offloading for multiaccess edge computing," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3138–3150, 2023.

[8] T. Pamuklu, A. Syed, W. S. Kennedy, and M. Erol-Kantarci, "Heterogeneous gnn-rl-based task offloading for uav-aided smart agriculture," *IEEE Networking Letters*, vol. 5, no. 4, pp. 213–217, 2023.

[9] T. Wu, W. Jing, X. Wen, Z. Lu, and S. Zhao, "A scalable computation offloading scheme for mec based on graph neural networks," in *2021 IEEE Globecom Workshops (GC Wkshps)*, 2021, pp. 1–6.

[10] M. S. Bute, P. Fan, G. Liu, F. Abbas, and Z. Ding, "A cluster-based cooperative computation offloading scheme for c-v2x networks," *Ad Hoc Networks*, vol. 132, p. 102862, 2022.

[11] Y. Chen, F. Zhao, X. Chen, and Y. Wu, "Efficient multi-vehicle task offloading for mobile edge computing in 6g networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 4584–4595, 2022.

[12] K. Li, W. Ni, X. Yuan, A. Noor, and A. Jamalipour, "Deep-graph-based reinforcement learning for joint cruise control and task offloading for aerial edge internet of things (edgeiot)," *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 21 676–21 686, 2022.