A DDPG Hybrid of Graph Attention Network and Action Branching for Multi-Scale End-Edge-Cloud Vehicular Orchestrated Task Offloading

Yejun He, Xiaoxu Zhong, Youhui Gan, Haixia Cui, and Mohsen Guizani

Abstract

With the development of 5G technologies and the wide application of artificial intelligence (AI), the mobile intelligent equipment and the providing services have both seen a significant rise in numbers and types. Some services, such as vehicular tasks, may go beyond the capability of the mobile equipment so that task offloading is required to help deliver such services. However, the graph structure of task offloading data, which can be a key to further improve algorithm's performance, is seldomly considered in future 6G-AI combined communication systems. In this article, we propose an efficient end-edge-cloud orchestration system that combines storage-partition and computation-shared in cloud cluster, cache mechanism, and cybertwin components. At the same time, we model this dynamic system as a graph structure composed of nodes and edges and propose a novel task offloading algorithm that incorporates a graph attention network (GAT) and action branching into deep deterministic policy gradient (DDPG) framework. Numerical results show that our offloading scheme achieves a good performance boost compared with other baseline schemes.

Introduction

Driven by the booming of the wireless communication technology and the penetration of machine learning techniques into almost every industry, three numbers are rocketing: the quantity of smart services, the amount of connections, and the volume of mobile data traffic. To support such dramatic increases, cloud computing and mobile edge computing (MEC) are two complementary and populous paradigms that have been investigated in the past decade. Since 6G aims at a capability increase by a factor of 10-100 [1], and has been envisaged as a "hyper-connected experience for all," it is reasonable to expect the confluence of 6G and various artificial intelligence (AI)-enabled services to be an efficient orchestration among multiend, multiedge and multicloud.

There are three major application scenarios of 5G: enhanced mobile broadband (eMBB), ultra-reliable and low latency communication (uRLLC), and massive machine type communication (mMTC). Based on these application scenarios, the MEC is expected to realize massive device connections, ultra-reliable, and low latency with the combination of AI technology, which is also applicable to future 6G communication systems. The MEC sinks the cloud closer to the edge of Internet of Things (IoT) devices and shares the pressure of the central cloud. However, due to the heterogeneous deployment of edge servers and network dynamics, existing offloading schemes are obsessed with task segmentation and ignore the overall system architecture, which makes it difficult to effectively support a large number of offloading tasks for IoT devices. Therefore, a deeper end-edge-cloud architecture and a more efficient offloading schemes are urgently needed.

Autonomous driving is a typical confluence of 6G and AI-enabled services. If driving tasks including speed control, lane changing, and emergency braking should be conducted in good time, a colossal amount of data collected by radars, lidars, or other vehicular sensors need to be processed within milliseconds. It is difficult for vehicles to process these tasks on their own. Additionally, as the whole world becomes more connected and more intelligent, demands of on-board entertainment or on-board working will be concerned. New and advanced vehicles may not meet requirements of both storage and computation capabilities. Thus, task offloading should be adopted to allow edge servers or cloud servers to process tasks and deliver services with good quality of service (QoS). The core problems of task offloading can be explained from two levels: the level of system architecture and the level of optimization problem.

The Level of System Architecture

In the level of system architecture, the basic questions include whether the system is centralized or distributed and what collaboration is involved to facilitate task offloading. Generally, a centralized system seeks to optimize the system utility from the control center which is responsible for scheduling task offloading, while a distributed system relies on game theory to achieve Nash equilibrium. Since we mainly focus on the centralized system, more details of a distributed system will not be discussed in the article.

The task offloading systems proposed in recent years involve three collaborations: the end-end collaboration, the end-edge collaboration, and the

Yejun He, Xiaoxu Zhong, and Youhui Gan are with Shenzhen University, China; Haixia Cui is with South China Normal University, China; Mohsen Guizani is with Mohamed Bin Zayed University of Artificial Intelligence, UAE.

Digital Object Identifier: 10.1109/MWC.019.2100718

1536-1284/23/$25.00 © 2023 IEEE

The task offloading systems proposed in recent years involve three collaborations: the end-end collaboration, the end-edge collaboration, and the end-edge-cloud collaboration.

end-edge-cloud collaboration. For the end-end collaboration systems, vehicular fog computing (VFC) [2] is a typical end-end collaboration paradigm where vehicles are incentivized to share their surplus computing resource. For example, Zhou *et al.* [3] proposed to sign the vehicles with adequate resource up as fog servers and use fog servers' resource to process tasks offloaded from request vehicles. For the end-edge collaboration systems, Ke *et al.* [4] proposed an end-edge framework that supports partial offloading from the service equipment layer to the MEC service layer. Kiran *et al.* [5] designed a software-defined edge cloudlet framework where multiple mobile users could offload their tasks to edge cloudlets, or to the remote cloud if edge cloudlets are overloaded. However, for the end-edge-cloud collaboration systems, there are relatively few. In addition, existing end-edge-cloud collaboration task offloading architectures are confined to the incumbent communication network. Yu *et al.*[6] raised that current networking paradigms should be upgraded to achieve key performance targets of 6G. Therefore, it is worth exploring the task offloading scheme of the end-edge-cloud collaboration systems based on the future potential network architectures.

The Level of Optimization Problem

The level of an optimization problem is about formulating and solving the task offloading problem. The objective function is maximized or minimized when a task offloading decision and a resource allocation are optimized. Convex optimizations, heuristic algorithms, and deep reinforcement learning (DRL) [7–10] are common methods used in the task offloading field. In [3], the authors proposed a task offloading mechanism based on matching learning and a vehicle computing resource management mechanism based on contract theory. Ji *et al.* [9] proposed a threshold-based edge-assisted federated learning (EAFL) offloading strategy to reduce the computational burden. Yu *et al.* [10] developed an efficient algorithm based on continuous convex approximation for joint optimization of task offloading and resource allocation problems.

DRL methods, which model the environment as a Markov decision process (MDP), are particularly handy. DRL methods do not assume knowledge of an exact mathematical model of the MDP. In addition, agents in these methods can learn a policy that maximizes the expected cumulative reward by continuously interacting with the environment in discrete time steps. Zhang *et al.* [7] proposed a deep-Q-learning-based scheme to maximize the system utility by choosing target servers and data transmission modes that include vehicle-to-BS mode and the joint mode of vehicle-to-vehicle and vehicle-to-RSU. Instead of optimizing an offloading decision, Ning *et al.* [8] formulated the problem of maximizing a profit function and utilized deep deterministic policy gradient (DDPG)[11] to jointly optimize the allocation of computing resources, caching storage and bandwidth. However, [7] and [8] only considered one aspect of task offloading, namely either offloading decisions or resource allocations. Research attempts that optimized both aspects in a comprehensive way are relatively few. Besides, most of DRL-based task offloading methods are based on deep neural network (DNN) [4] or convolutional neural network (CNN) [8], without considering the graph structure of task offloading data. In some cases, using a graph neural network, such as a graph attention network (GAT) [12], for feature extraction can play an important role in performance boost.

To tackle the above challenges, we propose an efficient DRL task offloading scheme for multi-scale vehicular networks. The main contributions are as follows:

- We propose an end-edge-cloud orchestration system that combines storage-partition and computation-shared in a cloud cluster, cache-considered mechanism and cybertwin components to facilitate vehicular tasks. It is an efficient system that schedules offloading in multiple cross-segments of different system scales.
- We model the dynamic system as a graph that consists of nodes and edges, where the features of nodes and the relation/edge between two nodes vary across time. Based on this model, we formulate the task offloading optimization problem as maximizing the combinational utility boost brought by offloading.
- We propose a novel DDPG-based task offloading algorithm combining GAT and action branching, where the GAT confines offloading destinations to neighborhood and the action branching's shared representation helps coordinate different sub-action branches. Numerical results show that the proposed DDPG hybrid algorithm presents a good performance in small-scale, middle-scale or large-scale system.

The remainder of this article is organized as follows: The system model is introduced in the next section and the problem is formulated following that. Then, a novel task offloading algorithm is presented with details. Following that, numerical results are illustrated and discussed. Finally, a conclusion is given in the last section.

System Model

As aforementioned, we are motivated to design a task offloading scheme for future communication systems. Instead of designing a task offloading scheme from scratch, we build the system on top of a concept model with cybertwin components [13]. Moreover, our system is customized with features required for solving the task offloading problem, including a scheduling center, the collaboration between cloud clusters and segment edges, the trait of storage-partition and computation-share in cloud clusters, and the cache-considered mechanism.

As shown in Fig. 1, the system can be depicted as three layers: the cloud layer, the edge layer, and the service layer. The cloud layer contains a scheduling center and regional clouds. The scheduling center is responsible for receiving and integrating request information from segment edges, and directing the task offloading process. A cloud cluster consists of several regional clouds that are maintained by different service providers. Each regional cloud keeps its own database to cache unique content, however a powerful server is shared to handle offloading tasks within the region. The advantage of the storage-partition and computation-share mechanism allows flexible and efficient infrastructure deployment. For example, a database can be placed according to the volume of the service data, while a cloud server can be deployed based on the QoS requirement without consider-

ing storage capacity. Moreover, the cache-considered mechanism means that if the content required for a task is already cached in the database, the vehicle will be spared from uploading, thus saving time, energy and communication resources.

The main parts of the edge layer are multiple segment edges alongside the road, and these segment edges play a pivotal role in facilitating task offloading and service delivery. Firstly, each segment edge possesses communication resources to support wireless connections between vehicles and segment edges, and between segment edges and the scheduling center. So the cybertwins in each segment edge can connect with their original vehicles, forward the task requests to the scheduling center, and help deliver services to their origins. Secondly, in each region, a cloud cluster is wired-connected to all segment edges, so the segment edges are able to access contents cached in the cloud cluster or offload tasks further to the cloud cluster with a negligible delay. The segment edge situated in the overlapping areas of two regions is wire-connected to the two cloud clusters. Thirdly, light servers placed in segment edges can help process some offloading tasks. Fourthly, neighboring segment edges are wired-connected to provide multiple segment edge server options for offloading.

Each vehicle has its own corresponding cybertwin at each segment edge which is responsible for security authentication, passing mission information to the control center, forwarding mission data, delivering missions, etc. Segment edges are the hosts of cybertwins. As part of the edge layer, cybertwins are the facilitators and windows of a task offloading. The processing of task information and the final delivery of services must go through cybertwins. In addition, a connection between the vehicles and the cybertwins needs to go through a certification process that relies on identification devices deployed at the segment edges.

In the service layer, a bi-directional lane is divided into multiple overlapping segments, and each segment represents the coverage of one segment edge. So when a task is generated, the vehicle will send the request information through its cybertwin to the scheduling center and deal with the task according to the instruction from the scheduling center. In the overlapping area where multiple segment edges are available, each vehicle chooses a cybertwin according to its preference setting which is assumed the nearest distance in this article.

The whole task offloading process can be summarized into four steps. First, vehicles connect with their cybertwins and send the task request information. Second, those cybertwins forward the task request information to the scheduling center. Third, the scheduling center integrates this information and returns instructions about where and how to offload, and how to allocate computing and communication resources. Finally, according to the instructions, tasks are offloaded, a resource is allocated, and services are delivered.

Problem Formulation

In the system, there is a task set $V = \{1, 2, ..., N\}$ and a server set $S = \{1, 2, ..., M\}$. Assume that the arrival rate of vehicular tasks in a time step follows a Poisson distribution with a parameter λ. λ is time-varying and is obtained by taking a bigger value between 0 and z, where z follows a Gaussian distribution $\mathcal{N}(\mu, \sigma)$. Since the traffic flow is often stable over a short period in real world, it is assumed that μ remains the same over a certain number of time steps and updated periodically. Each task $n \in V$ is modeled by task size $L^{(n)}$, computing density $D^{(n)}$, the ratio of input to output ϑ_n and the maximum tolerable delay $T_{\max}$. The workload of a task can be represented by the product of $L^{(n)}$ and $D^{(n)}$. The benefit of introducing $D^{(n)}$ and ϑ_n is implicitly considering various task types. Moreover, the parameter ϑ_n can model a situation where the delay of returning results is non-negligible, which is seldomly considered in existing research.

The task offloading proportion o_n is introduced to divide the task into a local computing part and an offloading part. Since the two parts are processed in parallel, the total delay T_n is the bigger value between the local computing delay and the offloading delay (including the delay of uploading data, remote computing and returning results), and the total energy consumption E_n is summing the energy consumption of the local computing and the uplink transmission.

In the local computing part, the local workload divided by the local computing capability $f_n^{(l)}$ equals the local computing delay, and the local computing energy consumption is calculated as in [14] with a parameter $\kappa = 10^{-26}$ that depends on the chip architecture.

In the offloading part, we adopt the cache-considered mechanism with a binary number η_n, where $\eta_n = 0$ indicates that the content required for task n is already cached in cloud clusters and this vehicle is free from uploading. If the content required is not cached, the offloading part would be composed by uploading data, remote computing and returning results. Therefore, the offloading delay is summing the delay of the three aspects. The size of transmitting data divided by transmission rates equals to the transmission delay. Since we consider an uplink-downlink decoupled communication system as [13], the achievable uplink rate $r_n^{(u)}$ and the achievable downlink rate $r_n^{(d)}$ depend on the product of the respective bandwidth $W_n^{(u)}$, $W_n^{(d)}$ and spectral efficiency $S_n^{(u)}$, $S_n^{(d)}$. Meanwhile, we only consider the energy consumption of uploading which is associated with the uplink power $P_n^{(u)}$.

In the task offloading problem, the delay and energy consumptions are the two main concerns, and the two aspects depend on the task model, the local computing model and the offloading model. It is assumed that a remote server resource is used at a financial cost $\mathcal{F}$. To the build the connection between the financial cost and the efficiency boost, we introduce the delay efficiency $\mathcal{T}$ that represents how many giga CPU cycles of the task are processed in one unit time, and the energy efficiency $\mathcal{E}$ that indicates how many giga CPU cycles of the task are computed when consuming one unit energy. Further, the utility of a delay efficiency $\mathcal{U}_t$ is defined as the average delay efficiency increase brought by one unit price compared with the all local computing. Likewise the utility of energy efficiency $\mathcal{U}_e$ is defined as the average energy efficiency improvement brought by one unit price compared with the all local computing. We introduce a timeout flag ρ_n and an excessive energy consumption flag ξ_n, where $\rho_n = 0$ indicates that the total delay T_n exceeds T_{max} and $\xi_n = 0$ indicates that the actual energy consumption E_n is bigger than the energy consumption of all local comput-

Each vehicle has its own corresponding cybertwin at each segment edge which is responsible for security authentication, passing mission information to the control center, forwarding mission data, delivering missions, etc.

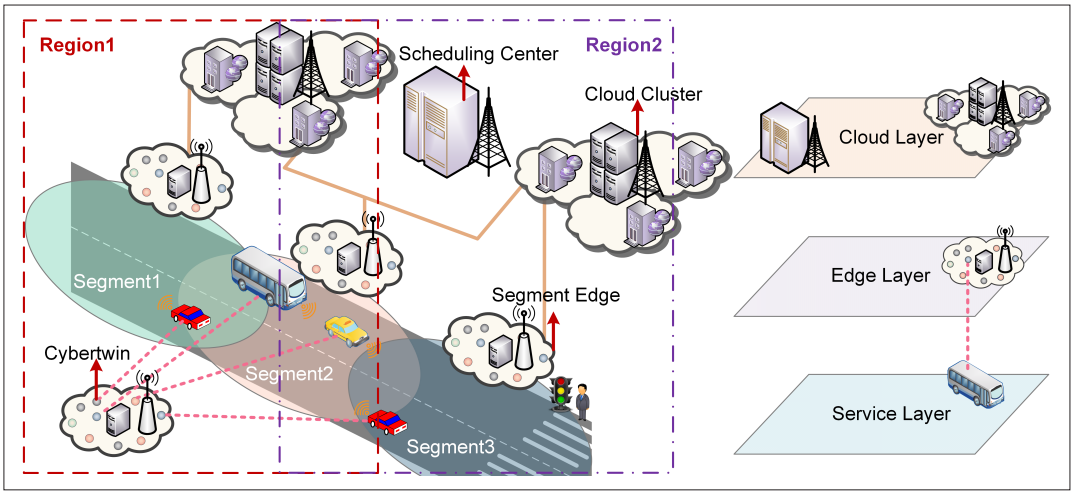

FIGURE 1. End-edge-cloud orchestrated system with cybertwins.

ing $E_{AL,n}$. It is to be noted that the time delay T or the energy consumption E divided by the product of the task size L and the computing density D is the time efficiency $\mathcal{T}$ or energy efficiency $\mathcal{E}$. Therefore, the overall utility $\mathcal{U}$ is given by a trade-off between $\mathcal{U}_t$ and Υ_e with a coefficient $\omega \in (0, 1)$:

$$\begin{aligned}\mathcal{U} &= \frac{1}{N}\sum_{n\in V} \omega\rho_n \mathcal{U}_t + (1-\omega)\rho_n\xi_n\mathcal{U}_e \\ &= \frac{1}{N}\sum_{n\in V} \omega\frac{\rho_n(\mathcal{T}_{AL,n}-\mathcal{T}_n)}{\mathcal{T}_{AL,n}\mathcal{F}_n} \\ &\quad +(1-\omega)\frac{\rho_n\xi_n(\mathcal{E}_{AL,n}-\mathcal{E}_n)}{\mathcal{E}_{AL,n}\mathcal{F}_n}, \\ s.t.\quad \rho_n &= \begin{cases}0, & if\ T_n > T_{max}, \\ 1, & otherwise,\end{cases} \\ \xi_n &\begin{cases}0, & if\ E_n > E_{AL,n}, \\ 1, & otherwise.\end{cases}\end{aligned} \tag{1}$$

The problem becomes: how to maximize the overall utility by optimizing an offloading decision of all tasks and resource allocation of segment edges and cloud clusters. It is to be noted that each segment edge or cloud cluster only facilitates tasks within the coverage and the total allocating resource of each segment edge or cloud cluster must not exceed the total resources owned by the cloud cluster itself.

The Hybrid of GAT and Action Branching under DDPG Framework

The DDPG-based Framework

Although DDPG has seen many successful implementations in a continuous control area, challenges such as the scale of the system, the availability of remote resources to each vehicle, and the constraints of resources allocation need to be further tackled. To cope with these challenges, we combine DDPG, multi-agent reinforcement learning and action branching [15] to form the DDPG hybrid.

Like other DRL methods, the proposed DDPG hybrid also contains three basic components: state, action and reward, which are described as follows

- *State:* At time step t, the state of the environment, denoted as s_t, includes $L^{(n)}$, $D^{(n)}$, $\mathcal{U}_n$, $T_{\max}$, η_n, $f_n^{(l)}$, $P_n^{(u)}$, $W_n^{(u)}$, $W_n^{(d)}$, $S_n^{(u)}$, $S_n^{(d)}$, $f_m^{(r)}$, $C_m^{(r)}$ and $\mathcal{A}$, where $f_m^{(r)}$ stands for the computing resource of server m, $C_m^{(r)}$ denotes the unit price of renting server m and $\mathcal{A}$ represents the availability of each server to each vehicle.
- *Action*: The action a_t involves five sub-actions: which server to offload, how much to offload, and how to allocate the uplink resource, how to allocate the remote computing resource, and how to allocate the downlink resource.
- *Reward:* Affected by a_t, the environment transfers to the next state s_{t+1} and returns an immediate reward r_t, where r_t is defined as $\mathcal{U}$.

Like the original DDPG, this approach also adopts an actor-critic framework, a replay buffer that offers training samples and soft updates on target networks. We add a Gaussian noise $\mathcal{N}(0, 0.01)$ to each sub-action for exploration. Moreover, instead of using one agent with five branches, which presents poor performance, we arrange two agents with different branches to learn the offloading policy and the resource allocation policy, respectively. In each action-branching actor, there are two layers: the first layer that includes one GAT layer is a shared representation module and the second layer consists of two/three GAT layers, as shown in Fig. 2.

The Hybrid of GAT and Action Branching

Although we have analyzed the state, action and reward function, there are still many difficulties:

- The state data is complex.
- The action has multiple dimensions.
- The dynamic change of the dimension.
- The lightness of the algorithm and the training effect.

For example, among the features listed previously, $\mathcal{A}$ is especially important because it directly affects which server a vehicle can offload to and which vehicle a cloud cluster or a segment edge should assign its resources to. However, such relation feature presents a great challenge to DNN or CNN in feature extraction because the data of DNN is sequence in nature and that of CNN is grid-like structure, while the data in this problem can not be simply represented by these two types. Thus, we model the task offloading data as a graph and reorganize the aforesaid features into three categories:

- The features of vehicles: $L^{(n)}$, $D^{(n)}$, $\mathcal{U}_n$, $T_{\max}$, η_n, $f_n^{(l)}$, and $P_n^{(u)}$.
- The features of cloud clusters or segment edges: $W_n^{(u)}$, $W_n^{(d)}$, $S_n^{(u)}$, $S_n^{(d)}$, $f_m^{(r)}$, and $C_m^{(r)}$.

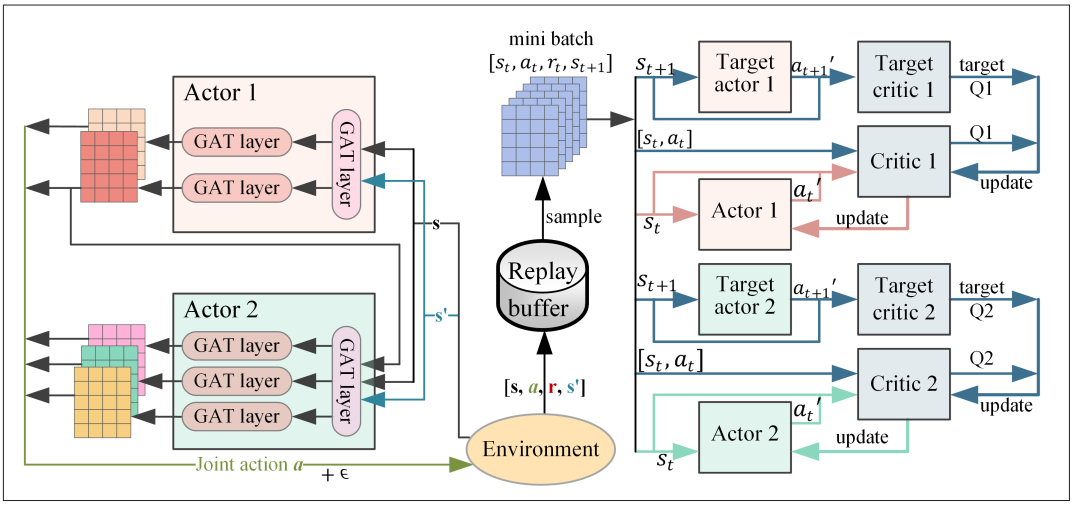

FIGURE 2. Dual-Agent DRL Framework.

- The availability matrix of cloud clusters or segment edges to vehicles: $\mathcal{A}$.

Then, GAT with multi-head attention is used as the basic structure of actor networks, as shown in Fig. 2. The shared representation module is a standard GAT layer and the branches are the modified GAT layers. The critic networks and target critic networks are similar to the actor networks, however the third layer, namely a fully-connected layer, is concatenated in the critic networks and target critic networks.

One sub-action of the actor network 1, namely the decision about where to offload, is input as an adjacency matrix into the actor network 2. This is to confine a server's resource allocation to the vehicles that have chosen the server as the offloading destination.

Simulation Results

Parameter Settings

In the experiment, we consider two cloud clusters and seven segment edges. The computing resource of each segment edge is randomly distributed from 150 to 200 GHz. $W_n^{(u)}$ is randomly distributed from 50 to 200 MHz, and W(d)n from 50 to 100 MHz. Both $S_n^{(u)}$ and $S_n^{(d)}$ are randomly distributed from 50 to 100 bit/s/Hz. In addition, one cloud cluster connects with the first four segment edges and the other cloud cluster connects with the last four. The computing resource of the two cloud clusters are set as 300 and 400 GHz, respectively. Besides, it takes 0.5 unit price when consuming one GHz of an segment edge and 1 unit price when using one GHz of a cloud cluster.

We select two groups of tasks: group A consumes more computing resources and group B uses more communication resources. Specifically, the task size of group A follows a uniform distribution between 8 to 10 kb, the corresponding computing density follows a uniform distribution between 10 to 12 kcycle/bit, and the arrival rate of these tasks is assumed to have a Poisson distribution with parameter λ_1. The task size of group B follows a uniform distribution between 10 to 12 kb, the corresponding computing density follows a uniform distribution between 8 to 10 kcycle/bit, and the arrival rate is assumed to have a Poisson distribution with parameter λ_2. Specifically, $\lambda_1 = \max(0, z_1)$ and $\lambda_2 = \max(0, z_2)$, where $z_1 \sim \mathcal{N}(\mu_1, 1)$ and $z_2 \sim \mathcal{N}(\mu_2, 1)$. Both μ_1 and μ_2 range from 5 to 20. The two parameters remain the same in one episode and get updated when a new episode begins. The computing capability of vehicles is randomly distributed from 0 to 1 GHz and $\mathcal{U}_n$ is from 0 to 0.5. $P_n^{(u)}$ is fixed as 0.5 W and ω is set as 0.5. $T_{\max}$ is set as 10 ms.

The first layer in actor 1, actor 2, target actor 1, and target actor 2 is standard GAT layer based on four attention heads and the corresponding hidden feature's size is 16. The second layer in these actors is a modified GAT layer with one attention head, where the hidden feature's size depends on the quantity of segment edges and cloud clusters. We use Adam optimizer with the same learning rate of 0.0005 for the actor and the critic and train with a minibatch size of 128. The target networks are updated with $\tau = 0.005$. We employ a replay buffer of size 50000 and adopt a reward discount factor of 0.99. The average $\mathcal{U}$ is considered as a key performance index and obtained by averaging over 1000 episodes.

Results and Discussions

As shown in Fig. 3, the DDPG hybrid presents a good convergence, and its average $\mathcal{U}$ varies between 17.5 and 19.3 after sufficient training episodes. As discussed previously, $\mathcal{U}$ stands for the combined efficiency improvement of the delay and energy consumption brought by one unit price compared with the complete local computing. More intuitively, take 19 for example, it means that by spending every one unit price for offloading, the average utility of the policy generated by the DDPG hybrid is 19 times better than the average utility when tasks are all processed by the vehicles themselves. At the same time, we can see that the task offloading and resource allocation-deep deterministic policy gradient-action branching (TORA-DDPG-AB) algorithm based on CNN and DNN performs poorly. The reason is that the CNN and DNN cannot effectively extract relational features and fail to limit the object of resource allocation to the corresponding neighborhood, resulting in some idle resources, while the effective task exceeds the delay or energy consumption limit by not being allocated sufficient resources.

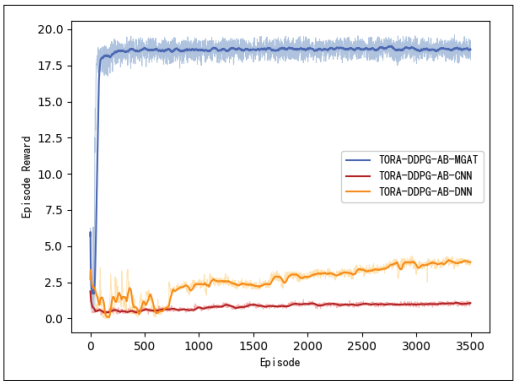

FIGURE 3. Training curves of the DDPG hybrid.

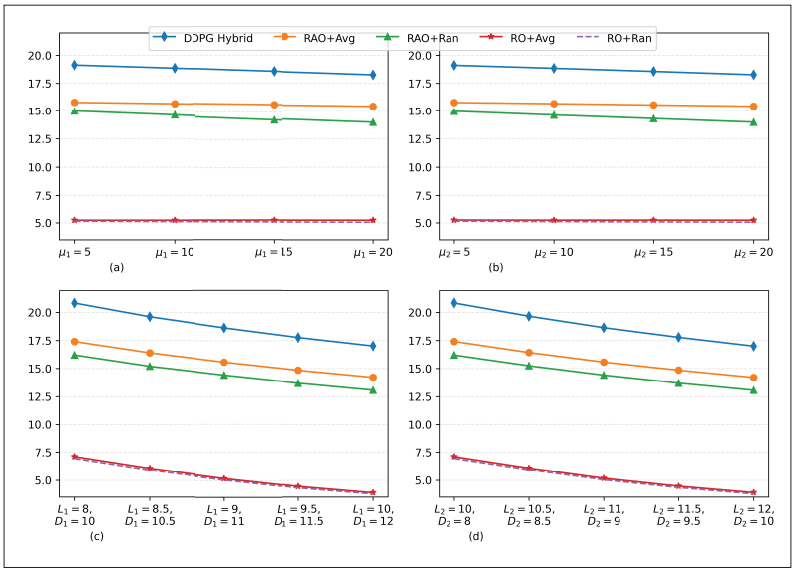

FIGURE 4. Average $\mathcal{U}$ over different arrival rates and workloads.

	Small-scale	Middle-scale	Large-scale
SE Qty.	7	13	25
$f_{SE}^{(e)}$ (GHz)	[150, 200]	[450, 500]	[750, 800]
$f_{CC}^{(e)}$ (GHz)	300, 400	900, 1000	1500, 1600
μ_1	[5, 20]	[50, 70]	[100, 120]
μ_2	[5, 20]	[50, 70]	[100, 120]

TABLE 1. Parameters of different systems.

Since common DNNs and CNNs cannot naturally confine the resource allocation within a certain neighborhood, the DNN- or CNN-based DRL frameworks have difficulty in satisfying allocation constraints meanwhile guaranteeing performance. Thus, the proposed DDPG hybrid is compared with the following schemes:

- RAO + Avg: random all offloading plus average resource allocation.
- RAO + Ran: random all offloading plus random resource allocation.
- RO + Avg: random offloading at random proportion plus average resource allocation.
- RO + Ran: random offloading at random proportion plus random resource allocation.

Figures 4a and b demonstrate how the arrival rate affects the system utility. Though the average $\mathcal{U}$ of the DDPG hybrid slides slightly when the task number increases, the DDPG hybrid outperforms the other four schemes. Figures 4c and d show how the utility of different schemes changes along with a growing workload. The utility of all five schemes falls as the workload of each task rises, meanwhile the DDPG hybrid presents a better performance compared to others. It is to be noted that Figs. 4a and c show how different schemes perform as parameters of group A change, and Figs. 4b and d correspond to tasks of group B. Similar utility performance and trend indicate that the system can balance the two groups of tasks.

We then extend the experiment to systems of a larger scale. Some parameters are specified in Table 1, where SE is the abbreviation of a segment edge and CC is the abbreviation of a cloud cluster. Other parameters that are not specified remain the same as specified in the last subsection. As shown in Fig. 5, the proposed method performs well even when the number of vehicles becomes more than two hundred.

Challenges and Future Directions

This article provides a new solution for the end-edge-cloud collaboration system with dynamic multi-to-multi relationship, but the proposed solution still has room for improvement and optimization.

Novel Graph Neural Networks

The DRL algorithm proposed in this article is based on improving the GAT network, but the GAT network is the most basic of many graph neural networks. A graph neural network is a research hotspot in recent years. Many researchers have proposed new graph neural networks suitable for different needs, such as Heterogeneous Graph Attention Network (HAN), Heterogeneous Graph Structural Attention Neural Network (HetSANN), Knowledge Graph Attention Network (KGAT), etc. Whether these new graph neural networks can be used to further improve performance or solve problems beyond the scope of the system model proposed in this article is worthy of further exploration and research.

Ultra-Dense Multi-to-Multi Relational Systems

This article conducts simulation experiments on small-scale and extended systems and achieves good performance, but for systems with ultra-dense multi-to-multi relationships, the performance of the algorithm is not ideal. How to design the task offloading and resource allocation optimization algorithm suitable for ultra-dense systems is also the focus of the next work.

Dependencies between Tasks

This article only considers some offloading tasks, but does not consider the graph task offloading class that has interdependence between subtasks and task segmentation constraints. Since this article has modeled the system as a graph, if we want to consider the graph task on the basis of this system graph, it is equivalent to further modeling each node of the system graph as a graph. How to deal with the relationship of nested graphs and achieve efficient feature extraction is a complex task, which will be further studied in the follow-up work.

Conclusion

In this article, we investigated the main factors limiting a task offloading performance and proposed an end-edge-cloud orchestration scheme for vehicular task offloading. To cope with dynamics and delay-sensitiveness of the system, to capture the availability features between vehicles and segment edges or cloud clusters, and to confine the resource allocation of segment edges or cloud clusters within their respective neighborhoods, we proposed a novel DDPG hybrid with GAT and action branching. Numerical results demonstrated that the proposed scheme achieves a good performance and can be applied to a different-scale system.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grants 62071306 and 61871433, in part by the Mobility Program for Taiwan Young Scientists under Grant RW2019TW001, and in part by the Shenzhen Science and Technology Program under Grants JCYJ20200109113601723, JSGG20210420091805014 and JSG 2021080215 4203011.

References

[1] U. of OULU, "Key Drivers and Research Challenges for 6G Ubiquitous Wireless Intelligence," 6G white paper, 2015.
[2] J. Shi *et al.*, "Priority-Aware Task Offloading in Vehicular Fog Computing Based on Deep Reinforcement Learning," *IEEE Trans. Vehic. Tech.*, vol. 69, no. 12, Dec. 2020, pp. 16067–81.
[3] Z. Zhou *et al.*, "When Vehicular Fog Computing Meets Autonomous Driving: Computational Resource Management and Task Offloading," *IEEE Network*, vol. 34, no. 6, Nov./Dec. 2020, pp. 70–76.
[4] H. Ke *et al.*, "Deep Reinforcement Learning-Based Adaptive Computation Offloading for MEC in Heterogeneous Vehicular Networks," *IEEE Trans. Vehic. Tech.*, vol. 69, no. 7, Jul. 2020, pp. 7916–29.
[5] N. Kiran *et al.*, "Joint Resource Allocation and Computation Offloading in Mobile Edge Computing for SDN Based Wireless Networks," *J. Commun. Networks*, vol. 22, no. 1, Feb. 2020, pp. 1–11.
[6] Q. Yu *et al.*, "A Cybertwin Based Network Architecture for 6G," *Proc. 2nd 6G Wireless Summit*, 2020, Levi, Finland, pp. 1–5.
[7] K. Zhang *et al.*, "Deep Learning Empowered Task Offloading for Mobile Edge Computing in Urban Informatics," *IEEE Internet of Things J.*, vol. 6, no. 5, Oct. 2019, pp. 7635–47.
[8] Z. Ning *et al.*, "Joint Computing and Caching in 5G-Envisioned Internet of Vehicles: A Deep Reinforcement Learning-Based Traffic Control System," *IEEE Trans. Intelligent Transportation Systems*, vol. 22, no. 8, Aug. 2021, pp. 5201–12.
[9] Z. Ji *et al.*, "Computation Offloading for Edge-Assisted Federated Learning," *IEEE Trans. Vehic. Tech.*, vol. 70, no. 9, Sept. 2021, pp. 9330–44.
[10] Z. Yu *et al.*, "Joint Task Offloading and Resource Allocation in UAV-Enabled Mobile Edge Computing," *IEEE Internet of Things J.*, vol. 7, no. 4, Apr. 2020, pp. 3147–59.
[11] T. P. Lillicrap *et al.*, "Continuous Control With Deep Reinforcement Learning," *Proc. Int'l. Conf. Learning Representations*, May 2–4, 2016, San Juan, Puerto Rico, pp. 1–14.
[12] P. Veličković *et al.*, "Graph Attention Networks," *Proc. 2018 Int'l. Conf. Learning Representations*, Apr. 30–May 3, 2018, Canada, pp. 1–12.
[13] Q. Yu *et al.*, "Cybertwin: An Origin of Next Generation Network Architecture," *IEEE Wireless Commun.*, vol. 26, no. 6, Dec. 2019, pp. 111–17.
[14] L. Huang *et al.*, "Deep Reinforcement Learning for Online Computation Offloading in Wireless Powered Mobile-Edge Computing Networks," *IEEE Trans. Mobile Computing*, vol. 19, no. 11, 1 Nov. 2020, pp. 2581–93.
[15] A. Tavakoli *et al.*, "Action Branching Architectures for Deep Reinforcement Learning," *Proc. Association for the Advancement of Artificial Intelligence*, Feb. 2–7, 2018, USA, pp. 1–9.

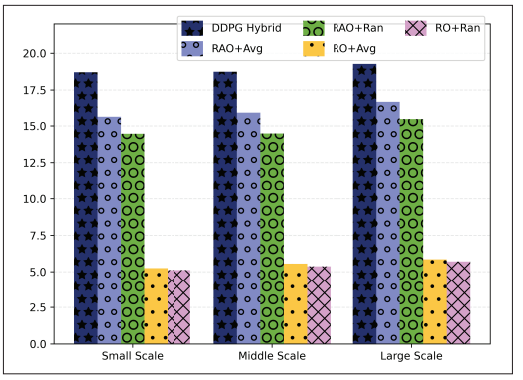

FIGURE 5. Average $\mathcal{U}$ over various-scale systems.

Biographies

Yejun He [SM'09] (heyejun@126.com) is a Full Professor with College of Electronics and Information Engineering, Shenzhen University, Shenzhen, China, where he is the Director of Guangdong Engineering Research Center of Base Station Antennas and Propagation, and the Director of Shenzhen Key Laboratory of Antennas and Propagation. His research interests include wireless communications, antennas, and radio frequency. He is a Fellow of IET

Xiaoxu Zhong (1031772642@qq.com) obtained her Master Degree of Information and Communication Engineering in Shenzhen University, Shenzhen, China, in 2022. Her research interests include wireless communications, mobile edge computing and AI.

Youhui Gan (youhuigan@qq.com) is currently pursing his Master Degree of Communication Engineering in Shenzhen University, Shenzhen, China. His research interests include wireless communications, mobile edge computing, and AI.

Haixia Cui [SM'22] (cuihaixia@m.scnu.edu.cn) is a Full Professor with the School of Electronics and Information Engineering, South China Normal University, Foshan 528225, China, and also with the School of Physics and Telecommunication Engineering, South China Normal University, Guangzhou 510006, China. Her research interests are in the areas of cooperative communication, wireless resource allocation, 5G/6G, and antennas.

Mohsen Guizani [S'85, M'89, SM'99, F'09] (mguizani@ieee.org) is currently a Professor of Machine learning at Mohamed Bin Zayed University of Artificial Intelligence (MBZUAI), UAE. He has authored/co-authored over 1,000 technical papers in top journals and conferences. He has been granted more than 10 U.S. patents. He was listed as a Clarivate Analytics Highly Cited Researcher in computer science in 2019, 2020, 2021, and 2022. He has won several research awards, including the 2015 IEEE Comunications Society Best Transaction Paper Award, and Best Paper Awards from top conferences such as IEEE ICC and IEEE Globecom.