Deep Graph Reinforcement Learning for Mobile Edge Computing: Challenges and Solutions

Yixiao Wang, Huaming Wu, *Senior Member, IEEE* and Ruidong Li, *Senior Member, IEEE*

***Abstract*—With the increasing Quality of Service (QoS) requirements of the Internet of Things (IoT), Mobile Edge Computing (MEC) has undoubtedly become a new paradigm for locating various resources in the proximity of User Equipment (UE) to alleviate the workload of backbone IoT networks. Deep Reinforcement Learning (DRL) has gained widespread popularity as a preferred methodology, primarily due to its capability to guide each User Equipment (UE) in making appropriate decisions within dynamic environments. However, traditional DRL algorithms cannot fully exploit the relationship between devices in the MEC graph. Here, we point out two typical IoT scenarios, i.e., task offloading decision-making when dependent tasks in resource-constrained Edge Servers (ESs) are generated in UEs and orchestration of cross-ESs distributed service, where the system cost is minimized by orchestrating hierarchical networks. To further enhance the performance of DRL, Graph Neural Networks (GNNs) and their variants provide promising generalization ability to wide IoT scenarios. We accordingly give concrete solutions for the above two typical scenarios, namely, Graph Neural Networks-Proximal Policy Optimization (GNN-PPO) and Graph Neural Networks-Meta Reinforcement Learning (GNN-MRL), which combine GNN with a popular Actor-Critic scheme and newly developed MRL. Finally, we point out four worthwhile research directions for exploring GNN and DRL for AI-empowered MEC environments.**

***Index Terms*—Graph Reinforcement Learning, Mobil Edge Computing, Internet of Things, Task Offloading, Resource Orchestration.**

I. Introduction

Along with the rapid development of the Internet of Things (IoT), it becomes evident that three pivotal metrics are experiencing remarkable growth: the proliferation of smart services (such as immersive video analysis and the flow of extensive health data), the surge in connections (including millions of demands between UEs and APs), and the exponential growth in mobile data traffic (encompassing data from vehicles and drones [1]. Therefore, computational resources and data from them are increasingly sinking to the edge of the network. This shift has led to a heightened demand for Quality of Service (QoS) that can no longer be adequately fulfilled solely through centralized cloud computing. In recent years, to fill the gap between centralized clouds and IoT devices, Mobile Edge Computing (MEC) has become an important technology for IoT scenarios due to its ability to achieve high QoS with low energy consumption in a distributed manner. Especially for compute-intensive and delay-sensitive applications, the MEC paradigm allows such applications to run on multiple Edge Servers (ESs) with less congestion [2]. Given the proximity of ESs to UEs compared to CSs, they provide significant advantages, including substantially reduced delay and fast feedback through processing procedures, making them an essential component of IoT systems.

Unfortunately, many current studies overlook the dependencies inherent in MEC systems, including those between tasks and communication among devices. This oversight frequently leads to a compromise in Quality of Service (QoS). In this paper, we address this gap by exploring two types of graph-structured MEC scenarios within IoTs that have impacts on academia, industry, and daily life, as depicted in Fig. 1.

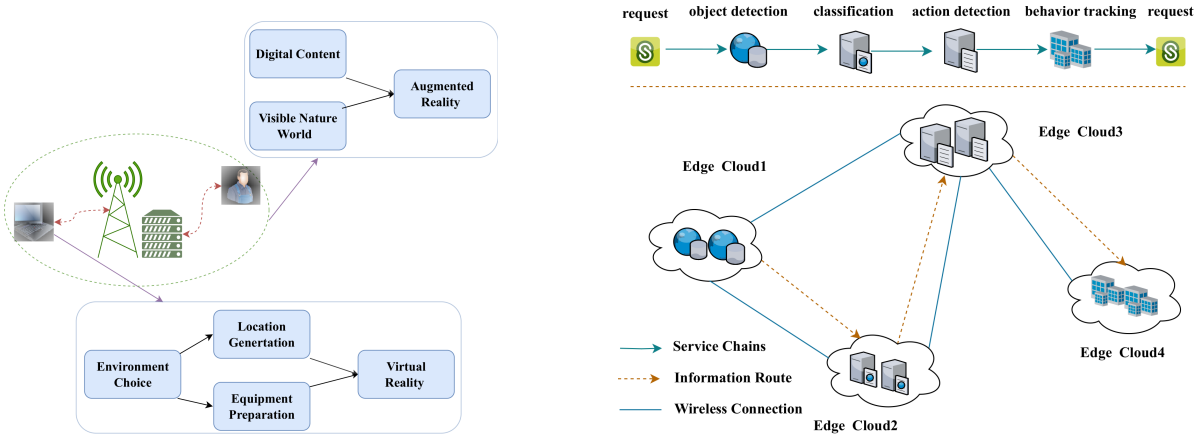
- **Task Offloading:** As shown in Fig. 1(a), numerous compute-intensive and delay-sensitive tasks originating from different users and equipment cannot be entirely processed locally. Therefore, the effective utilization of communication and computational resources can be achieved by making judicious offloading decisions during interactions with the MEC environment [3]. However, there are often obvious dependencies among multiple tasks generated by users and equipment. In other words, certain tasks can only be executed after the completion of specific prerequisite tasks. For instance, a face recognition application entails three sequential steps: face detection, face classification, and face retrieval [4]. In this context, the classification depends on the preceding detection, which subsequently provides information for retrieval from the database if required. We have illustrated this concept using VR/AR.
- **Resources Orchestration:** As illustrated in Fig. 1(b), the exponential expansion of the scope of IoT applications foresees a significant proliferation of connected devices catering to diverse applications. Importantly, the constraint of low cost-efficiency has emerged as a bottleneck, impeding the achievement of agile, flexible, and cost-efficient resource orchestration within the MEC ecosystem, encompassing various aspects. The efficient orchestration of workloads across converged edge platforms remains a challenge, particularly for network functions and multi-IoT services with diverse computation requirements and distinct Service Level Objectives (SLOs). Therefore, strategic cost management plays a crucial role in ensuring sustainable ESs' services [5]. To illustrate, we have used

This work is supported by the National Natural Science Foundation of China under Grant No. 62071327, and the Tianjin Science and Technology Planning Project under Grant 22ZYYYJC00020.

Y. Wang and H. Wu are with the Center for Applied Mathematics, Tianjin University, Tianjin 300072, China (e-mail: {wang_yixiao, whming}@tju.edu.cn).

R. Li is with the Institute of Science and Engineering, Kanazawa University, Kanazawa 920-1192, Japan (e-mail: liruidong@ieee.org).

(Corresponding author: Huaming Wu)



(a) Task offloading (b) Resources Allocation

Fig. 1: Two typical graph scenarios in IoT environments

video analysis as an example.

There are many challenges in building MEC networks, e.g., how to extract IoT data from a complex and highly-varying environment. Unfortunately, traditional heuristic methods such as genetic algorithms and ant colony algorithms cannot handle these scenarios immediately, while dynamic programming methods and optimization theories are usually trapped into local optimization [2]. DRL, which combines the decision-making ability of reinforcement learning with the perception ability of deep learning, is regarded as the most promising methodology for tackling this problem. However, relying solely on DRL may not fulfill the necessity wherein certain dependent tasks can only be executed after specific designated tasks have been completed [4]. Typically, the state programmed for DRL is usually Euclidean data, which is commonly seen in images and texts sorted like a grid. Undoubtedly, failing to collect adequate graphical information can weaken the performance of the overall algorithm, since the degree of dependency can have a huge influence on many goals. Instead, Graph Neural Network (GNN) is very appropriate for these two split-new MEC environments, since it can extract features of graph-like data while making decisions, and then transfer the reduced intermediate data to centralized cloud servers.

In this paper, two GNN-based architectures are specifically designed for the aforementioned typical graph-structured MEC scenarios. The main contributions can be summarized as:

- We propose a novel Graph Neural Networks-Proximal Policy Optimization (GNN-PPO) framework to tackle the task offloading decision-making problem, where different UEs generate dependent tasks that require offloading to various servers characterized by distinct topologies and task profiles to facilitate real-time computation.
- We propose a novel Graph Neural Networks-Meta Reinforcement Learning (GNN-MRL) framework aimed at orchestrating distributed services across ESs to minimize the overall system cost. In this framework, ESs or UEs are connected as a graph in their respective layers, allowing communication between different layers.
- Through simple experiments conducted on task offloading scenarios, our methods employing bidirectional Graph Convolutional Network (GCN) and Global Attention Pooling (GAP) demonstrate superior performance compared to other approaches, with lower average running costs across varying node numbers, bandwidths, and CPU frequencies. Additionally, the optimal convergence is achieved when the number of layers in our model is set to three.
- In addition, we also discuss existing challenges and potential future research directions concerning the integration of GNN and DRL for processing graphical structure data in IoT environments.

II. Related Work

In this section, we initially provide a brief introduction to the background of RL and GNN. Subsequently, we conduct a comparative analysis of various representative works, highlighting the distinctions in the addressed problems and identifying commonalities in the utilization of GNN and RL. Table I identifies and compares key elements of related works, encompassing scenarios, DRL types, GNN types, and addressed problems.

TABLE I: The qualitative comparison of the current literature. TO and RA are short for "Task Offloading" and "Resource Allocation", respectively.

Work	Scenario	DRL Types	GNN Types	Problem
[6]	EO-DCI	Hierarchical	GGS-NN	RA
[7]	EC system	MAA2C	GCN	RA
[8]	MDC-network	MAPPO	GCN/GAT	RA
[9]	EdgeIoT	A2C	Graph Generation	TO
[10]	SDVN	GraphSAGE	DDPG	TO/RA
[11]	MEC	REINFORCE	GCN/GAT	TO
[12]	CEU system	PPO	GAT	TO

A. Reinforcement Learning and Graph Neural Network

Given its sequential decision-making nature, the optimization challenges in MEC can be effectively addressed using Reinforcement Learning (RL), which effectively tackles NP-hard problems in dynamic and uncertain scenarios by prioritizing the learning of optimal actions through interactions with

the environment to maximize expected rewards. DRL, coupled with the powerful generalization capabilities of Deep Neural Networks (DNNs), has found applications in numerous IoT-based scenarios. A less explored variant, Meta-Reinforcement Learning (MRL), operates with two distinct "loops". The outer loop gradually adjusts the parameters of the meta policy, maintained on the ESs, by accumulating experiences across various contexts. This outer loop also oversees the execution of the "inner loop", responsible for generating the initial model for a specific task. The inner loop exhibits rapid adaptability to new tasks through a small number of gradient updates, enabling fast convergence.

The effectiveness of DRL and MRL has been well-established across various domains. However, certain scenarios involve explicit relationships that can be naturally modeled as graphs, posing challenges beyond the capabilities of DRL or MRL alone. GNNs have demonstrated superior performance in handling machine learning tasks involving graph-structured data. In contrast, methods like simple DRL and MRL, designed primarily for Euclidean data, are not well-suited for the complexities of emerging IoT scenarios. Undoubtedly, there is a clear need for specialized approaches that can adeptly aggregate information from different agents in these graph-based scenarios.

B. Integration of RL and GNN in Resource Orchestration

Another scenario involves a distributed GNN system, where researchers aim to construct a framework for orchestrating resources across ESs for optimal cost-effectiveness.

Li *et al.* [6] introduces HRLOrch, a hierarchical RL model with a policy neural network based on GNN principles, which is based on the gated graph sequence neural network (GGS-NN), effective in extracting features about a long node sequence in graph-structured data. HRLOrch is devised to minimize blocking probability by orchestrating the allocation of IT resources in data centers and spectrum resources on fiber links within elastic optical data center interconnections (EO-DCI). The hierarchical RL approach incorporates lower-level and upper-level models, finely designed to enhance convergence performance during training in sparse reward environments. Wang *et al.* [7] presented a learning-based approach to adapt to time-varying requests and dynamic service prevalence within edge-cloud systems. This approach involves a two-phase strategy employing GCN-based multi-agent advantage actor-critic (MAA2C), which aims to enhance individual intelligence, thereby facilitating optimal dispatch and orchestration decisions in the edge-cloud system(EC-system). To conquer the challenge of the dynamics and diversity of embedding Service Function Chains (SFC) requests in multi-datacenter (MDC) networks, a two-stage GCN-assisted multi-agent PPO (MAPPO) scheme was proposed in [8]. It mainly maximizes the overall acceptance ratio of SFC requests while minimizing the total cost in an MDC network. This is achieved through an optimization of resource orchestration, focusing on the efficient mapping of SFCs onto the physical network infrastructure in MDC scenarios.

C. Integration of RL and GNN in Task Offloading

The mature integration of GNN and DRL presents a promising paradigm for decision-making in task-offloading. Li *et al.* [9] explored the joint optimization of task offloading and Unmanned Aerial Vehicle (UAV) cruise control to minimize the task dropping rate due to computational task cancellation and offloading errors in harsh environments. GNNs are developed to supervise the training of real-time continuous actions of UAVs within the Advantage Actor-Critic (A2C) framework. Deng *et al.* [10] developed a novel software-defined networking-based vehicular ad hoc network (SDVN)-based system architecture characterized by computation offloading with edge Distributed Denial of Service (DDoS) attack mitigation. A GNN-based collaborative DRL (GCDRL) model to generate the resource provisioning and mitigating strategy, which evaluates the trust value of the vehicles, formulates mitigation of edge DDoS attacks and resource provisioning strategies. GNNs are also used to extract vehicular spatial and structural features of edge nodes, respectively. Aiming to minimize the makespan of a DAG task, Lee *et al.* [11] proposed a DRL-based priority assignment model that leverages both temporal and structural features within a Directed Acycled Graph (DAG). The model effectively learns a priority-based scheduling policy through the use of GCN and policy gradient methods. GCN is specifically employed to handle the complex interdependent task structure To minimize the makespan of user tasks in the Cloud-Edge-User framework (CEU), Cao *et al.* [12] proposed a DRL and Graph Attention Network (GAT)-based scheme. This approach efficiently utilizes the training process on a resourceful cloud to accommodate the numerous data and computation resource requirements associated with the task.

III. Two Typical Graph Scenarios in IoT

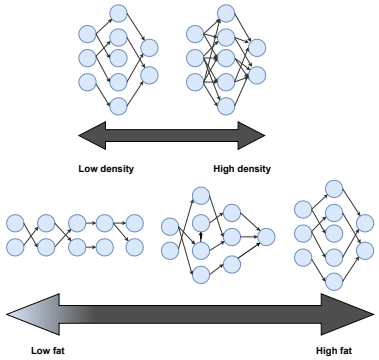
In this section, we continue discussing the two graph-structured scenarios in IoT(Task Offloading and Resource Allocation).

A. Task Offloading

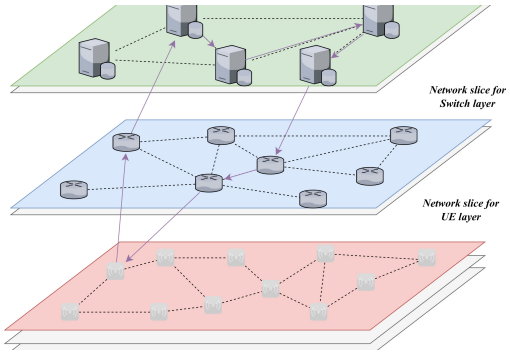
In daily life or industrial production, many IoT devices generate interrelated offloading tasks, however, these tasks cannot simply be performed locally or in the Edge Servers(ESs) for real-time computation. Fig. 1(a) shows AR/VR applications, UEs will generate decomposable jobs with distinct task profiles and dependencies, where child tasks must wait for the completion of parent tasks before they can be executed. For example, to create augmented reality, digital content and the visible nature world should be prepared in advance. This is a simple analogy because it is sure that there are many other steps in the process of digital content and the visible natural world, where the dependent tasks also stand. The dependency information can be effectively modeled as a DAG, where each task is viewed as a node in the graph, and the dependency relationships are represented by directed edges.

As shown in Fig. 2(a), the topology of the generated DAGs is controlled by three parameters: *task number*, *fat*, and *density*. Specifically, *fat* influences the width and height of the

DAG, while *density* determines the number of edges between two levels of the DAG [4]. This is a very quantitative way of describing granularity. Each node is characterized by its size, representing specific features such as task size, maximum latency, and the demand for various computational resources. Recently, it has been popular among many researchers to combine features extracted from tasks (modeled as DAGs) [9], [11], [12] with certain state information embedded into a Multi-Layer Perception (MLP), both of which will serve as the input and backbone of the DRL.



(a) Features of generated DAGs



(b) Information flow of cross-ESs distributed service

Fig. 2: The illustration for two typical scenarios in IoT

B. Resource Allocation

The pivotal strategy lies in orchestrating a network's entire resources. Network operators must consistently upgrade their service provisioning strategies to fully leverage the limited and distributed resources, including the spectral resources of fiber links, IT services, and storage on ESs [6].

At various hierarchical levels, distributed devices establish connections with each other. For instance, ESs are interconnected, and their communication can be represented as an undirected symmetric graph. The top and bottom layer of Fig. 2(b) shows how both ESs and UEs form a data graph. On one hand, ESs create an extensive edge network in collaboration with a series of APs, such as 5G base stations and IoT gateways, to facilitate data transportation. This data graph allows ESs to effectively cover a range of UEs, enhancing network coverage and service availability. On the other hand, the connected APs can also form a data graph, albeit not necessarily, allowing ESs to cover a range of UEs.

When a query from the bottom layer(Application Layer) is raised and obtained through the AP layer, each ES first aggregates a subset of the graph data via region-specific APs and computes the embedding through a given GNN model. At runtime, ESs are coordinated in a collaborative manner, where they are committed to exchanging necessary graph data with each other, executing GNN computations individually in parallel, and repeating this routine.

As is shown in Fig. 1(b), when new requests for video analysis from APs arrive in ES layers, the resource provisions according to the distributed GNN structure will happen within the ESs layer. To carry out the complete information flow from the request, object detection, classification, action detection, and behavior tracking are sequentially performed by being sent among wirelessly connected edge clouds, using information about the requests by paralleling model execution and exchanging data mutually. The key question is how to choose the information route under the maximization of the traffic cost and constraints on some resources.

When the information flow is finished, the finished information will spread back from the last ES, through the AP layer, and finally reach the request UE.

IV. Solutions

A. Motivation

The integration of Deep Reinforcement Learning (DRL) and Graph Neural Networks (GNN) is chosen to address the challenges discussed in Section III due to its inherent suitability. This combination is well-suited for the unique characteristics and complexities presented by the discussed scenarios.

First and foremost, these tasks necessitate optimization for the long-term rewards by selecting appropriate actions that adapt to evolving environments. However, these environments are often intricate and multifaceted. Consequently, researchers employ Deep Neural Networks (DNN) to extract pertinent environmental features. However, DRL methods are traditionally well-suited for Euclidean spaces, whereas the network state in the aforementioned problems is inherently graph-structured. Consequently, DNNs struggle to effectively process graph-structured states, leading to the loss of crucial information. For instance, in Section III-A, jobs consist of interdependent tasks that form a DAG structure. Similarly, in Section III-B, the network state topology exhibits a graph structure. Lastly, although seemingly straightforward, deterministic heuristics fall short in making offloading decisions or orchestrating resources adaptively in response to the rapidly changing and intricate non-Euclidean network states. This compelling challenge motivates us to embrace the power of Deep Graph Reinforcement Learning [6] as a solution.

B. GNN-PPO for Task Offloading Decision-Making

In the first scenario, each UE generates a unique job for computation offloading. Typically, for the graphical characteristics of a certain job, it usually carries out binary offloading rather than partial offloading. It's important to emphasize that the adoption of binary offloading can induce changes in the

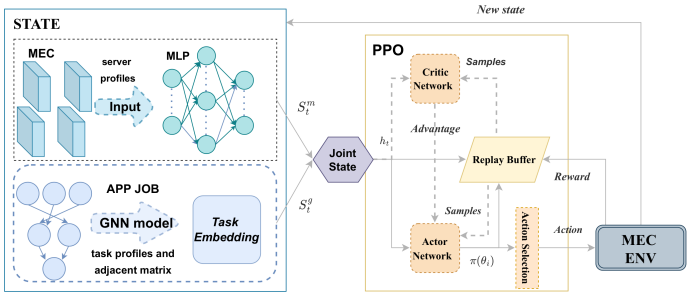


Fig. 3: The framework of GNN-PPO

features of the task environment. For instance, it may lead to variations in the availability of computational resources during specific time slots.

As displayed in Fig. 3, we take our proposed scheme GNN-PPO as an example to illustrate the procedures. We assume that the local computing ability is inaccessible and there exists only a single UE within the entire system. At the beginning of each time-slot t, we concatenate the state of the MEC system S_t^m with the state of the current decision-making job S_t^v to create a global state for offloading actions and rewards based on the customized objective function. In this network, each individual GNN layer (comprising various GNN variants) is employed to embed a specific job, while MLP layers are utilized to generate the embedding for the present MEC system. UEs employ this global state for a policy network that generates actions. The current state S_t undergoes a transition to the subsequent state S_{t+1} following the action A_t. It is important to note that we primarily emphasize the state embedding process. The state of the MEC environment solely depends on features such as ESs' capacity and energy levels, as well as the computational requirements of tasks.

To extract sufficient information, we embed the MEC environment state and task information to be decided, respectively.

- **MEC Embedding:** In each time slot, we gather information from the MEC environment to make an offloading decision for the current task. The MEC environment primarily comprises ESs, and occasionally, cloud servers as well. Taking ESs as an example, crucial indicators for DRL in making offloading decisions include whether the ES is connected to the current UE, the availability of CPU resources, available bandwidth, and the distance between the current user and the relevant servers. If the task is offloaded to an MEC server, it is imperative to consider Channel State Information (CSI), and the current overall channel gain of the respective ES is incorporated into the MEC environment information. The features extracted from both MEC servers and users are then input into the MLP to learn the embedding of the entire MEC environment.
- **Task Embedding:** Taking into account the dependencies among tasks, a GNN network is used to confine the overall structure of a job, facilitating the extraction of information necessary for making offloading decisions for each individual task. In essence, the learned embedding through the GNN can be regarded as the feature of the task, eliminating the need for manual feature engineering.

C. GNN-MRL for Orchestration of Distributed Cross-ESs Service

For the second scenario, we characterize the system costs in terms of the life cycle of distributed GNNs over ES tasks [13]. For ease of presentation, concentrating the GNN processing workload on inference, we specifically highlight two types of graphs: i) the edge network that hosts distributed model execution, and ii) the data graph formed from the associated data of UEs, which feeds the GNN model as the input graph.

Take Fig. 2(b) as an example, we represent the control decisions as $\pi(t) = x_{v,i}(t)|v \in \mathcal{V}_t, i \in \mathcal{D}_t$. These decisions orchestrate the operation of the ES network $\mathcal{T}$ and the UE network $\mathcal{G}$, determining whether UE i is provisioned in ES v during time slot t. It's important to note that each UE is limited to selecting only one ES. Furthermore, ESs collaborate by sharing their collected graph data, thereby eliminating the need for data replication and reducing storage overhead. This collaborative approach optimizes data management within the network.

Given control decisions, we decompose the total cost incurred in the cross-ES system, including the activating cost, data collection cost, GNN computation cost and cross-ES routing cost. The flow of a distributed ES system starts from collecting data derived from distributed clients. Before collecting data, a newly launched UE or ES will activate some devices (e.g. launching a virtual machine image), so there is an activating cost. After these two steps, as we mentioned before, computing the whole UE network through the GNN model consists of aggregation and update steps, so the cost consideration on GNN is of great importance. To describe GNN processing, in addition to computing GNN over their resident graphs, we also need to consider the data transferred across ESs. What's more, there are some other costs, e.g., edge server maintenance cost [13], cloud outsourcing cost [14], and operation cost [15].

Given that different UEs generate various types of data during different time slots, a one-size-fits-all strategy for orchestration decisions that applies to all UEs is insufficient. Therefore, we provide a comprehensive description of the training process for GNN-MRL in the distributed ESs system, as depicted in Fig. 4.

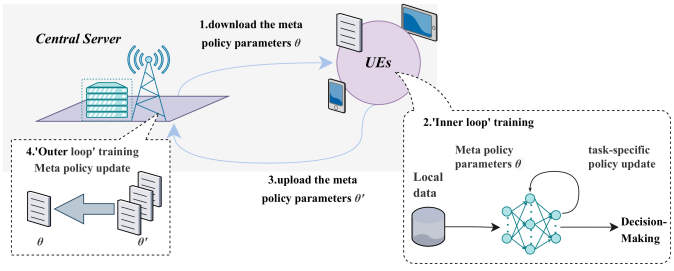


Fig. 4: The framework of GNN-MRL

- **Step 1**: Each UE downloads the parameters of the meta policy from the central server.
- **Step 2**: An inner loop training is conducted on every UE with the local experiences captured from both the whole

network and the meta policy to learn a tasks-specific policy.

- **Step 3**: The UE uploads the parameters of the task-specific policy to the central server.
- **Step 4**: The CS assembles the parameters of task-specific policies and starts an outer loop training to update the meta policy.

After completing these four steps, another round of training will be started by the central server.

V. Performance Evaluation

To validate the efficiency of our methods, we carry out straightforward experiments for Section IV-B, which illustrates GNN-PPO. It becomes evident that, in scenarios where a job comprises N tasks to be offloaded, with each task offering M choices, the total action space expands exponentially, reaching a scale of $\mathcal{O}(N^M)$. In this context, we focus on the simplest case, where each task presents only two options: either local computation or remote execution on a designated edge server. It's important to note that this edge server is accessible only when there are no ongoing tasks currently utilizing its resources.

A. Parameters Setup

Our simulations were conducted on laptops equipped with NVIDIA 4G GeForce RTX 3050 GPUs. Specifically, we set the number of task points to 15, and we varied the "fat" (the width of a Directed Acyclic Graph) and the "density" (the dependency density) within the set {0.3, 0.4, 0.5, 0.6, 0.7, 0.8}. We configured the upload and download bandwidth to 7 Mbps, while the local and remote capable CPU resources were set to 1 GHz and 10 GHz, respectively. The size of data that needed to be processed and received if executed remotely for each node was randomly generated within the range [5, 50] kB. In the GNN model depicted in Fig. 3, responsible for generating the task embedding, our model is constructed by incorporating both the bidirectional GCN module and the GAP module, as illustrated in Fig. 5.

Our model consists of three separate Graph Convolution layers for the actor network and critic network. The hidden size in our model is set to 16, and the learning rate for the commonly shared layer is always one-tenth of that for the two separate layers, set at 0.001. We trained our model for 100 epochs with a batch size of 128 and a total data amount of 512. To facilitate PPO, we employed Generalized Advantage Estimation (GAE) with a discount factor of 0.95 and a weighting factor of 0.99. The data resources can be accessed here[1].

B. Results Discussion

Our model's performance evaluation focuses on two key aspects: the training parameters of DRL networks and the system parameters of MEC environments. We measure performance primarily through the running cost, defined as the completion time of the last node within a given job. Additionally, we assess the impact of DRL network training parameters, including action diversity evaluated through entropy loss, as shown in Fig. 6(a). It is evident that the 3-layer GCN outperforms the others in terms of both running cost and entropy loss. This configuration exhibits faster convergence and a broader ability for action exploration, making it the optimal choice. In Fig. 6(b), we averaged the improvement effect across different sample sizes compared to alternative methods, including Local (all tasks performed locally), Remote (all tasks executed on the edge server), Random, Round Robin, and Greedy. Our method demonstrates improvements of 26.66%, 22.03%, 15.14%, 12.23%, and 7.97%, respectively.

As shown in Fig.7, regarding the node numbers, after averaging the improvement effects across different node numbers, our GCN-based method demonstrated average performance improvements of 29.84%, 25.29%, 17.96%, 15.85%, and 8.74% when compared to Local, Remote, Random, Round Robin, and Greedy strategies, respectively. Similarly, for varying bandwidth scenarios, our GCN-based method showed the best average performance improvements. Likewise, across different local CPU capabilities, our GCN-based method delivered average performance improvements of 27.57%, 23.31%, 16.78%, 14.10%, and 6.06% when compared to Local, Remote, Random, Round Robin, and Greedy, respectively.

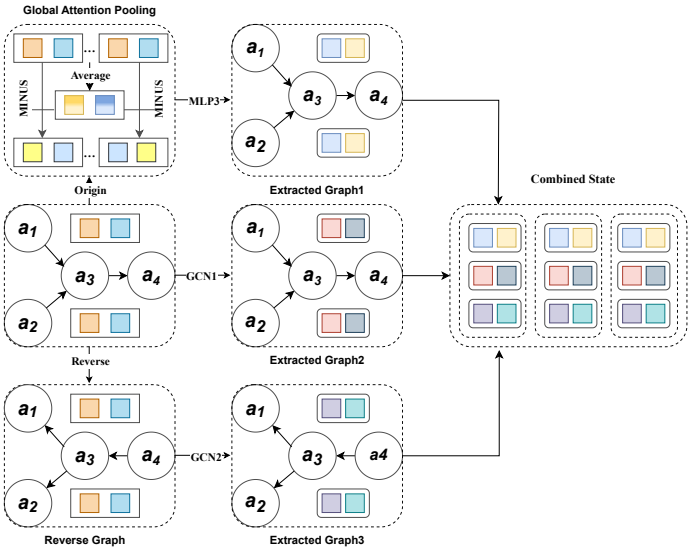
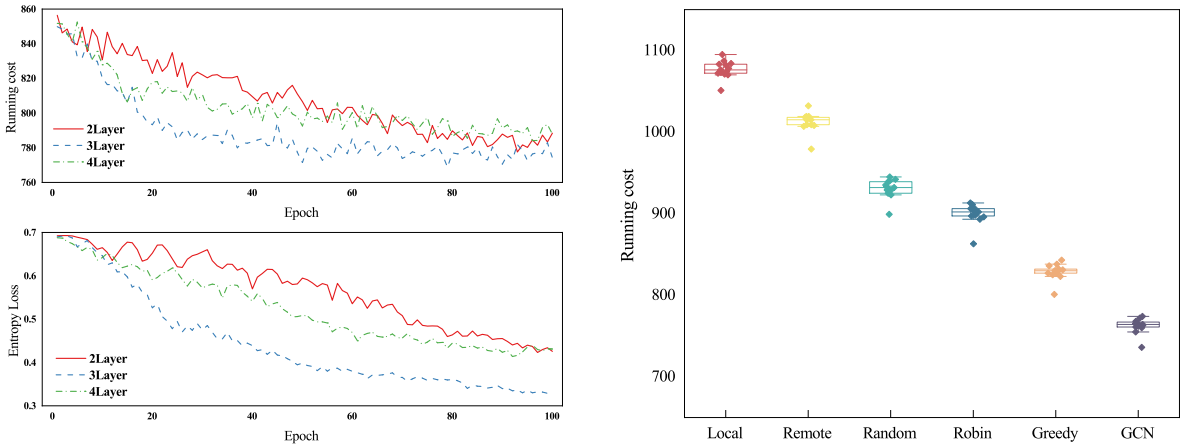


Fig. 5: The GNN model. The bottom two rows are the bidirectional GCN module, where different GCN components are utilized for the original graphs and reverse graphs, extracting two distinct sets of GCN features. The top row represents the GAP module, where all node features are averaged to obtain global features. These global features are then subtracted from each node's feature, resulting in each node's new feature. Subsequently, we combine these three types of features as input for the Actor-Critic Network.

VI. Challenges and Prospects of GNN-based MEC

This section identifies the challenges preventing widespread adoption and suggests future directions of combined GNN-

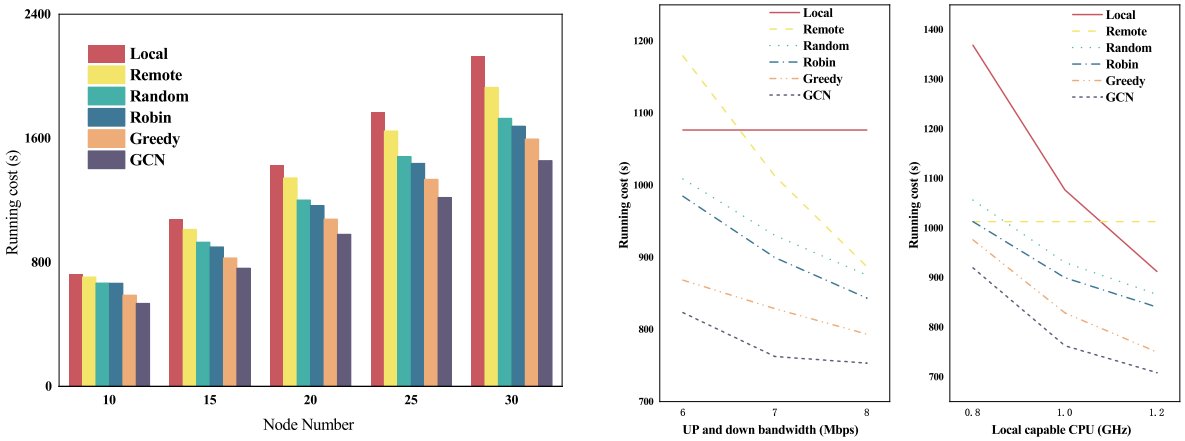
[1] https://github.com/linkpark/RLTaskOffloading



(a) The influence of model structure on running cost

(b) The influence of training batch size on running cost

Fig. 6: Running cost under different training parameters



(a) The influence of the number of job nodes on running cost

(b) The influence of bandwidth and local capable CPU resources on running cost

Fig. 7: Running cost under different system parameters

DRL frameworks in MEC scenarios to unlock the combination's full potential.

A. Large Graph Network Learning

In contrast to the contemporary large networks found in the real world, most current studies focus on small-scale networks, typically featuring fewer than 100 nodes and 10,000 edges. However, simulating the topology of larger graph-structured networks within these small-scale datasets proves insufficient to meet the significant computing demands.

The widely adopted divide-and-conquer approach allows for the decomposition of any extensive network into smaller, more manageable components. As a result, the availability of local graph information becomes essential. For instance, the process of graph partitioning effectively divides the ES and UE networks into smaller ones, allowing for a more detailed focus on their local structures. Batch sampling and importance sampling play a vital role in capturing the local structure, thereby contributing to enhanced representation performance in IoT networks. More importantly, the UE or ES in IoT can collaboratively act, rather than ignoring the interests of others, and they can also use partitioned strategies to solve large graph problems. However, for some middle-scale graphs, it is uncertain whether it is worthy of gaining narrow improvement margins at the cost of the local information burden incurred by these algorithms.

B. Generalization Across Problems

Enhancing the generalization ability of DRL and GNNs is one of the key research directions for AI-empowered MEC, especially when DNNs are suffering from over-fitting in the training environment. Meanwhile, there also exists the problem of over-smoothing for GNNs.

Many methods have been proposed in the recent literature, e.g., ResGCN and Edge Pooling, which remains an edge-cutting issue that confuses graph-structured data and environments. One possible research orientation is to develop graph MRL frameworks for MEC, where different UEs are suitable for different tasks or environments, with less experience that can be provided by various applications. For example, in the case of task offloading decision-making, the number or density of tasks in a job may be unpredictable for each time slot, the computational resources of MEC may vary from time to time, and even the availability of certain resources may change over time. More importantly, data augmentation techniques can be

utilized for graph-structured scenarios if UEs can be exposed to multiple graph environments.

C. Seamless Connection From Simulation to Deployment

Most of the prevailing GNN-DRL methods are developed based on synthetic datasets and conducted on popular simulated platforms. The scarcity of data for RL tasks, particularly for graph-structured data and even more so in IoT scenarios, poses a significant challenge. Generative Adversarial Networks (GANs) emerge as a potential solution in scenarios where training data is limited or the collection of real data proves to be expensive. GANs can play a crucial role in generating synthetic data to augment datasets and facilitate more robust training for RL tasks. While the capability and scale to replicate real-world networks have improved, real-world modeling demands a level of tuning that surpasses that of simulation platforms and synthetic datasets, some of which are publicly available.Rigorous validation and testing become imperative, especially for life-critical applications in the Internet of Health Things (IoHT) and Internet of Vehicles (IoV), before deploying DRL algorithms in certain real-world scenarios. The gap between training on simulated environments and actual applications persists, particularly when dealing with graph-structured data. In summary, achieving a seamless transition from simulation to reality, ensuring safety and productivity, remains a key focus for future research directions.

D. Dynamic/Heterogeneous Graph-structured Environments

It is well known that existing GNN models [2]–[5]in many IoT scenarios perform feature extractions or communications over homogeneous fixed graphs. In a fixed graph, the addition and removal of nodes/edges are disregarded. However, in practical deployment, it is impossible for only a single device or server that satisfy all computational resource allocation requirements. For example, in social networks, there may be hospitals, mobile devices and automobiles for UEs and ESs with different computational resources, and even fog servers, the dynamic spatial relationship may evolve continuously, so learning on heterogeneous dynamic graphs is indispensable if we make further research into the real world.

Such graphs cannot be analyzed easily. Consequently, new models and algorithms capable of learning from heterogeneous dynamic graphs will be highly beneficial in real-world MEC systems. DRL can be viewed as one of the potential future research directions with heterogeneous dynamic graphs. Last but not least, current research on graph-based deep learning for MEC still has much room for improvement. To the best of our knowledge, no previous studies have concentrated on graph-structured MEC scenarios for IoT applications. The problems we posed above are still far from completely addressed, existing GNN or DRL approaches still fail to meet the requirements of a range of IoT scenarios, face many challenges and require urgent attention from researchers.

VII. Conclusions

In this paper, we illustrate two typical graph-structured IoT scenarios within MEC environments, intending to optimize performance through decision-making via DRL. Additionally, we underscore the broad applicability and significance of merging GNN and DRL. We introduce the task offloading scenario and the concept of distributed hierarchical ESs. We then propose GNN-PPO as a solution for the former situation and GNN-MRL for the latter. Simulation results demonstrate the robustness of our approach across different variables, including varying numbers of nodes within a job, different upload and download bandwidths, and varying local CPU capabilities. Furthermore, we point out four promising research directions, including problem generalization, the seamless transition from simulation to deployment, and adaptability in dynamic/heterogeneous graph-structured environments.

References

[1] Y. He, X. Zhong, Y. Gan, H. Cui, and M. Guizani, "A ddpg hybrid of graph attention network and action branching for multi-scale end-edge-cloud vehicular orchestrated task offloading," *IEEE Wireless Communications*, vol. 30, no. 4, pp. 147–153, 2023.

[2] 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.

[3] H. Zhou, T. Wu, X. Chen, S. He, D. Guo, and J. Wu, "Reverse auction-based computation offloading and resource allocation in mobile cloud-edge computing," *IEEE Transactions on Mobile Computing*, vol. 22, no. 10, pp. 6144–6159, 2023.

[4] J. Wang, J. Hu, G. Min, W. Zhan, A. Y. Zomaya, and N. Georgalas, "Dependent task offloading for edge computing based on deep reinforcement learning," *IEEE Transactions on Computers*, vol. 71, no. 10, pp. 2449–2461, 2022.

[5] F. Guim, T. Metsch, H. Moustafa, T. Verrall, D. Carrera, N. Cadenelli, J. Chen, D. Doria, C. Ghadie, and R. G. Prats, "Autonomous lifecycle management for resource-efficient workload orchestration for green edge computing," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 1, pp. 571–582, 2022.

[6] B. Li and Z. Zhu, "Gnn-based hierarchical deep reinforcement learning for nfv-oriented online resource orchestration in elastic optical dcis," *Journal of Lightwave Technology*, vol. 40, no. 4, pp. 935–946, 2022.

[7] Z. Wang, Y. Zhao, C. Qiu, Q. He, X. Wang, X. Wang, and Q. Hu, "Socialedge: Socialized learning-based request scheduling for edge-cloud systems," in *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, 2023, pp. 853–863.

[8] D. Xiao, J. A. Zhang, X. Liu, Y. Qu, W. Ni, and R. P. Liu, "A two-stage gcn-based deep reinforcement learning framework for sfc embedding in multi-datacenter networks," *IEEE Transactions on Network and Service Management*, vol. 20, no. 4, pp. 4297–4312, 2023.

[9] 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.

[10] Y. Deng, H. Jiang, P. Cai, T. Wu, P. Zhou, B. Li, H. Lu, J. Wu, X. Chen, and K. Wang, "Resource provisioning for mitigating edge ddos attacks in mec-enabled sdvn," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 24 264–24 280, 2022.

[11] H. Lee, S. Cho, Y. Jang, J. Lee, and H. Woo, "A global dag task scheduler using deep reinforcement learning and graph convolution network," *IEEE Access*, vol. 9, pp. 158 548–158 561, 2021.

[12] Z. Cao and X. Deng, "Dependent task offloading in edge computing using gnn and deep reinforcement learning," *ArXiv*, vol. abs/2303.17100, 2023.

[13] L. Zeng, C. Yang, P. Huang, Z. Zhou, S. Yu, and X. Chen, "Gnn at the edge: Cost-efficient graph neural network processing over distributed edge servers," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 3, pp. 720–739, 2023.

[14] Z. Zhou, Q. Wu, and X. Chen, "Online orchestration of cross-edge service function chaining for cost-efficient edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1866–1880, 2019.

[15] T. Dong, Q. Qi, J. Wang, Z. Zhuang, H. Sun, J. Liao, and Z. Han, "Standing on the shoulders of giants: Cross-slice federated meta learning for resource orchestration to cold-start slice," *IEEE/ACM Transactions on Networking*, vol. 31, no. 2, pp. 828–845, 2023.