SITOff: Enabling Size-Insensitive Task Offloading in D2D-Assisted Mobile Edge Computing

Zheyuan Hu, *Graduate Student Member, IEEE*, Jianwei Niu, *Senior Member, IEEE*, Tao Ren, *Member, IEEE*, Xuefeng Liu, *Member, IEEE*, and Mohsen Guizani, *Fellow, IEEE*

***Abstract*—Mobile edge computing (MEC), along with device-to-device (D2D) assisted MEC (D-MEC), are promising technologies that could improve the quality-of-experience for mobile devices (MDs) by offloading their tasks to edge servers or nearby idle MDs. There is a popular trend to develop distributed task offloading algorithms using multi-agent reinforcement learning (MARL), whose adoption of central critics during training makes the offloading still size-sensitive. Therefore, this paper proposes a Size-Insensitive Task Offloading (SITOff) algorithm for D-MEC based on fully-distributed offloading without maintaining any central venue. Specifically, taking advantage of the inherent graph-like structure of D-MEC, SITOff adopts graphs to represent MDs' states and relationships and form each MD's local knowledge about D-MEC through graph computation. Furthermore, considering the limitation of local knowledge in performing whole performance-oriented offloading, each MD utilizes D2D-transmitting to exchange knowledge with its neighbors and form a comprehensive knowledge about D-MEC to enhance the coordination of distributed offloading. Additionally, regarding the different impacts of neighbors' knowledge, each MD leverages attention mechanisms to selectively learn its neighbors' knowledge during knowledge-exchange. Extensive experimental results show the superiority of SITOff over state-of-the-art MARL-based offloading algorithms in D-MEC with various MDs, and the easy collaboration of SITOff with curriculum-learning for large-scale D-MEC offloading.**

***Index Terms*—Device to device, mobile edge computing, task offloading, reinforcement learning.**

Received 24 April 2024; revised 14 September 2024; accepted 11 October 2024. Date of publication 28 October 2024; date of current version 5 February 2025. This work was supported in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LY22F020006, in part by the National Natural Science Foundation of China under Grant U23B2025, Grant 62372027, and Grant 62372028, and in part by Shanghai Aerospace Science and Technology Innovation Foundation under Grant SAST2022-093. Recommended for acceptance by J. Rodrigues. *(Corresponding author: Tao Ren.)*

Zheyuan Hu is with the School of Computer Science and Engineering, Beihang University, Beijing 100191, China (e-mail: huzheyuan18@buaa.edu.cn).

Jianwei Niu is with the State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing 100191, China, also with the Zhongguancun Laboratory, Beijing 100094, China, and also with the Zhengzhou University Research Institute of Industrial Technology, Zhengzhou University, Beijing 100094, China.

Xuefeng Liu is with the State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing 100191, China, and also with the Zhongguancun Laboratory, Beijing 100094, China (e-mail: liu_xuefeng@buaa.edu.cn).

Tao Ren is with the State Key Laboratory of Intelligent Game, Institute of Software Chinese Academy of Sciences, University of Chinese Academy of Sciences, Beijing 100190, China (e-mail: rentao22@iscas.ac.cn).

Mohsen Guizani is with the Machine Learning Department, Mohamed Bin Zayed University of Artificial Intelligence (MBZUAI), Abu Dhabi, UAE (e-mail: mguizani@ieee.org).

Digital Object Identifier 10.1109/TMC.2024.3483951

I. Introduction

MOBILE edge computing (MEC) is a promising technology to accommodate mobile devices (MDs) with both satisfactory computing resources and task latency [1], [2]. One of the key issues in MEC is task offloading [3] that decides the appropriate (part of) task offloaded to base stations (BSs) for edge computing, along with the corresponding transmitting power allocated for task offloading. In addition, MDs with idle computing resources could also be potential candidates for task offloading, resulting in the paradigm of device to device (D2D) assisted MEC [4], [5]. According to the real-world scenarios [6], [7], [8], in D2D-assisted MEC (D-MEC), thin MDs (e.g., smart phones and wearable devices) could offload their computation-intensive tasks (e.g., gesture recognition, speech recognition, augmented reality, and 3D object detection) to thick MDs (e.g., intelligent vehicles, laptops) when the edge server (ES) deployed on the BS is crowded with offloaded tasks, so that computing resources within the whole MEC can be utilized more fully to improve MDs' quality of experience. Fig. 1 illustrates the task-resource conditions of MDs in a D-MEC system in different time slots. Similar to [9], [10] assuming each MD could split the arrived jobs into multiple pieces of tasks and store them in task queues, only one task need to be de-queued and computed at each time slot. However, due to different computational burdens of jobs, each MD may face tasks with different computing resource requirements. Besides, each MD could be deployed with different types of computing unit [5], [11], possessing different amount of computing resources. Taking the MDs and tasks at time slot n in Fig. 1 as an example, MD2 is a thin MD (with small green arcs representing poor computing resources.), but faces a heavy task (with large orange arcs representing heavy tasks). Hence, when MD3 and MD4 both offload tasks to the ES, MD2 could choose to offload its task to MD1 who has rich computing resources (e.g., large computing capacity marked with large green arcs) but light tasks (small orange arcs), in case of task overload in the ES.

Considerable efforts have been made on developing efficient task offloading algorithms in MEC and D-MEC. For instance, the authors in [11], [12], [13] formulate the offloading problem as a mixed integer non-linear optimization problem, which is then decomposed into sub-problems solved using mathematical programming (MP). A crucial issue of MP-based offloading algorithms is its dependence on reliable D-MEC system models, as well as its computational burden of the scale-related iterative

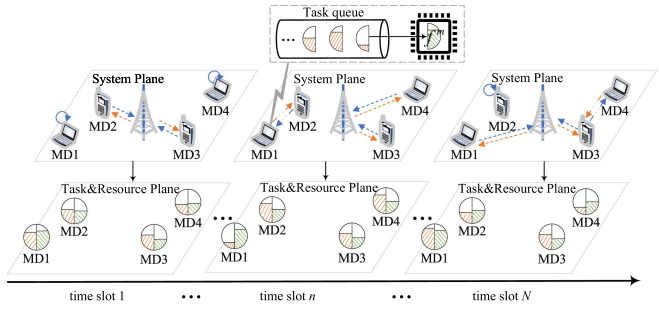


Fig. 1. An example of D2D-assisted MEC. In the System Plane, the blue and orange dashed arrows denote uplink and downlink of task offloading, respectively. In the Task&Resource Plane, the orange and green arcs represent under-processing tasks and available computing resources of MDs at one time slot, respectively.

programming [14]. In view of this, there is a popular trend to design task offloading algorithms using reinforcement learning (RL), which is suitable for solving discrete-time Markov decision process (MDP) problems. In [5], [15], [16], the authors develop centralized offloading algorithms, by first transforming the offloading problem into a MDP, then designing MEC states, actions and reward, and last solving the MDP using state-of-the-art RL algorithms.

Despite the desirable performance achieved by centralized DRL-based offloading algorithms in most cases, the collection of D-MEC states from all MDs for making each offloading decision by a central controller, could inevitably bring challenges in both scalability and survivability [17]. Facing this, multi-agent RL (MARL) is adopted in [18], [19], [20] to develop distributed offloading algorithms, considering each MD as an individual actor that makes offloading decisions according to its local MEC state. On account of the potential limitation of MDs' local states and actions on the whole MEC performance, these MARL-based offloading algorithms also maintain a central critic for each MD during training [21], [22], [23]. Each central critic observes the global state of all MDs and guides the training of its responsible MD, so that each MD can still make desired offloading actions during execution based on the learned knowledge during training.

For MARL-based offloading, the execution performance could be satisfactory so long as the interplay relationships between MDs are similar to those during training, even if each MD only observes its local state. However, MDs could meet decision troubles when the number of MDs, i.e., D-MEC size, changes. Specifically, the introduction of the central critic guides the offloading decision of each MD on one hand, but it also introduces the dependence on the fixed-dimensional global state of all MDs on the other hand. This means the interplay relationships between MDs are also related to the D-MEC size during training, and performance degradation could be observed during execution when the D-MEC size changes and new interplay relationships are brought in. To illustrate this issue, we conduct a pilot experiment, as shown in Fig. 2, on a MARL-based offloading algorithm [24] to investigate the impacts of changing the number of MDs during execution (different from training). As seen in the middle of Fig. 2, the MARL-based offloading

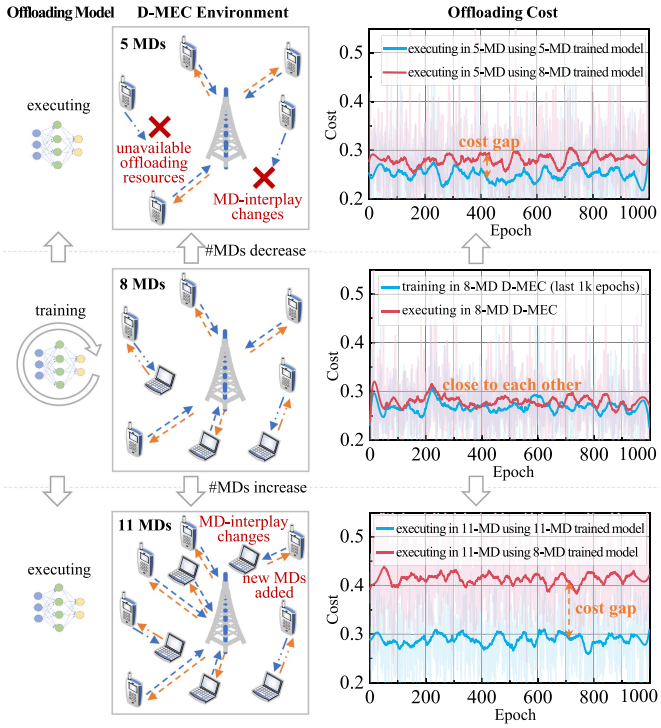


Fig. 2. A pilot experiment investigating the impacts of the change of D-MEC sizes (i.e., #MDs) on MARL-based offloading performances during execution. In the middle, the investigated MARL-based offloading model is trained in 8-MD D-MEC. In the top and bottom, the 8-MD trained offloading model is taken to execute in 5-MD and 11-MD D-MECs, respectively. Obvious performance degradation (compared to the offloading model trained in the native 5/11-MD D-MEC) could be noticed, especially in the 11-MD D-MEC.

algorithm is trained in D-MEC with 8 MDs. When the obtained offloading model executes in 5-MD D-MEC (the top of Fig. 2) and 11-MD D-MEC (the bottom of Fig. 2), obvious performance degradation is noticed (compared to the offloading model trained in the native 5/11-MD D-MEC), especially when the number of MDs increases.

From the above pilot-experiment, it can be seen that the centralized critic kept in existing MARL-based offloading works [24], [25], [26], [27] results in not only the requirements of global communication conditions and centralized training venues, but also the issues of keeping fixed number of MDs during centralized training and same number of MDs during decentralized execution as training. In view of this, we propose to learn distributed offloading policies for each MD via the decentralized training and decentralized execution paradigm. To the best of our knowledge, there has been seldom attention paid to address these issues in D-MEC via decentralized training. However, to achieve an efficient decentralized offloading policy for each MD during decentralized training, the following three problems need to be carefully considered: (1) how to augment the local observation of each MD to enhance its distributed offloading? (2) how to design a structure to represent the augmented observation of each MD? (3) how to efficiently absorb useful information from augmented observations to further facilitate distributed offloading of each MD?

To address these problems, we propose a Size-Insensitive Task Offloading (SITOff) algorithm for D-MEC, that could achieve satisfactory offloading performance irrespective of the change of the number of MDs involved in D-MEC. Specifically, SITOff directly adopts distributing training without the need of maintaining central critics that each MD makes offloading decisions only according to local observation. Taking advantage of the inherent graph-like structure of D-MEC, SITOff adopts graph to represent MDs' states and their relationships and forms each MD's knowledge about the local D-MEC environment through graph computation. Furthermore, given the limitation of each MD's local knowledge on maximizing the offloading performance of whole D-MEC, we design an inter-MD knowledge-exchange mechanism by utilizing MD's D2D transmitting ability that aggregates each MD's own knowledge and neighbors' knowledge and builds its comprehensive knowledge about D-MEC to facilitate coordinated offloading decisions. Besides, regarding the distinct impacts of neighbors' knowledge on each MD's offloading decision, SITOff also leverages attention [28] for the MD to selectively learn its neighbors' knowledge during knowledge-exchange. Extensive experimental results verify the superiority of SITOff against MARL-based offloading algorithms in D-MEC with various numbers of MDs. Additionally, owing to the size-insensitivity, we empirically show SITOff can easily collaborate with curriculum learning [29] to facilitate large-size D-MEC offloading directly training on which from scratch would be difficult.

Our main contributions are summarized as follows:

1) We develop a fully distributed task offloading algorithm that the offloading decision of each MD is made only based on its own knowledge about D-MEC, and thus is size-insensitive.
2) We utilize graph representation to encode MDs' states and relationships so that each MD could form its own knowledge about the local D-MEC environment through graph computation. We further enhance the graph computation with graph mean field to improve the speed, by which the average time cost of graph computation achieves at least 78.15% decline compared to the vanilla graph computation method.
3) We further design an attention-based knowledge-exchange mechanism by utilizing D2D-transmitting to help each MD selectively learn neighbors' knowledge and acquire a comprehensive knowledge about D-MEC to enhance local decision-making, by which the average task success rate (TSR) and epoch success rate (ESR) achieve up to 94.6% and 93.2% performance improvements compared to baselines, respectively.
4) We evaluate SITOff via extensive experiments and indicate its superiority over state-of-the-art MARL-based methods. With the same experimental setup, SITOff can achieve more than 77% performance improvement, and the average cost is reduced to 0.24. We also empirically show that SITOff can easily collaborate with curriculum learning to facilitate large-size D-MEC offloading and reach the average cost at round 0.52, which is difficult for directly learning from scratch.

The rest of the paper is organized as follows. Section II introduces the related work. Section III presents the system model. Section IV formulates the optimization problem. Section V proposes our algorithm, which are then evaluated in Section VI. Finally, Section VII concludes the work.

II. Related Work

In this section, we introduce the related work about DRL-based task offloading in D-MEC from two categories, centralized offloading and decentralized offloading, according to the location of offloading-controllers. In addition, we present existing work on graph representation and computation in MEC relevant to our proposed method.

A. Centralized Offloading

In centralized offloading, all computation offloading decisions are generated and scheduled by a central node. In this mode, all MDs in the MEC environment send their computation task information to this central node, who then decides whether and how many each task should be executed locally or offloaded to the BS. The authors in [5] proposed a centralized deep reinforcement learning framework to solve the joint computation offloading problem in D2D communications. The framework takes a hierarchical actor-critic architecture to cope with the continuous-discrete mixed action spaces. The work in [30] achieved the joint optimization of computation offloading and resource allocation by designing a long short-term deterministic policy gradient approach. The work in [31] introduced a centralized offloading algorithm based on closest observed rewards that can improve the utilization of system resources, to overcome the poor initial quality of experience (QoE) scores problem. A novel proximal policy optimization strategy was introduced in [32] to derive better centralized offloading policy for continuous power allocation.

Key Issues: The advantages of the centralized offloading approach is that there is only one central task scheduling node, which facilitates the deployment of the offloading algorithm. Also, the centralized offloading algorithm receives fully observable states of all MDs for unified decision making, easily obtaining stable offloading policies [5]. However, despite the desirable performance achieved by centralized DRL-based offloading in most cases, as the number of MDs increases the algorithm complexity for centralized offloading also increases which makes it difficult to solve. Additionally, the requirement of collecting D-MEC states from all MDs for making each offloading decision by a central controller, inevitably brings challenges in both scalability and survivability [17]. Therefore, we propose SITOff that is suitable for size-insensitive and decentralized task offloading.

B. Decentralized Offloading

In decentralized offloading, offloading decisions are made by each MD itself, independent of any central node. Decentralized offloading avoids the requirement of central controller and states collection, showing higher flexibility and scalability. The work

in [33] proposed a utility maximization algorithm by introducing a centralized training distributed execution (CTDE) paradigm in the heterogeneous vehicular and cellular networks, where a central critic is employed to guide the training for decentralized offloading policies. The authors in [34] also applied the CTDE paradigm to maximize the weighted-sum system throughput. A decentralized MARL-based resource allocation algorithm in [35] was proposed to maximize the long-term average energy efficiency. The paper in [9] proposed the heterogeneous multi-agent task offloading algorithm based on distributed RL framework with homogeneous MDs and heterogeneous BS to enable MDs to choose actions independently at the execution stage.

Key Issues: For CTDE-based decentralized offloading in most existing work, the advantage is that each MD makes offloading decisions independently without the need for unified management by the central node, which improves the efficiency of decision making in the D-MEC system [35]. Meanwhile, the execution performance could be satisfactory so long as the interplay relationships between MDs are similar to those during training, even if each MD only observes its local state. However, MDs could meet decision troubles when the number of MDs, i.e., D-MEC size, changes. Specifically, the introduction of the central critic guides the offloading decision of each MD on one hand, but it also introduces the dependence on the fixed-dimensional global state of all MDs on the other hand. This means the interplay relationships between MDs are also related to the D-MEC size during training, and performance degradation could be observed during execution when the D-MEC size changes and new interplay relationships are brought in. To this end, we propose a fully distributed computation offloading algorithm that can adapt to different numbers of MDs without the need for central critic guidance.

C. Graph Representation and Computation in MEC

Recent years, graph representation and computation have received a lot of focus in MEC system. The work in [36] modeled the wireless network as a directed graph in a D-MEC system. A supervised learning method was proposed to enhance the robustness and maximize the data transmitting rate of the network. However, the method requires pre-built datasets with fine labeling, which can lead to high training costs for the model. The work in [37] applied graph representation and computation to vehicular D-MEC and combined features of graph representation with MARL to optimize spectrum allocation. This method, however, only optimizes the transmitting rate of the network and lacks consideration of task offloading. In this respect, the authors in [38] proposed a meta-reinforcement learning task offloading algorithm based on graph neural network (GNN) and seq2seq network to fast adapt to MEC environments with different dependency tasks. However, the seq2seq network in [38], which accepts fixed input dimensions, makes it difficult to apply the algorithm to the environments with varying numbers of MDs. The work in [39] built MEC as a shared MD agents-ESs graph and converted the overall information into a fixed-size embedding to address the training difficulties induced by the large amounts of MDs and ESs. But it focused only on task offloading between ES and MD and did not consider the D-MEC case.

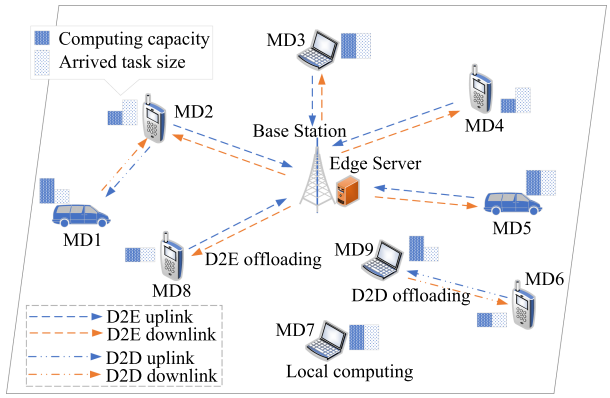


Fig. 3. An example of the D2D-assistd MEC network. Thin MDs (e.g., MD3, MD4) could offload their tasks to ES. Meanwhile, thin MDs (e.g., MD2, MD6) could also offload their tasks to thick MDs (e.g., MD1, MD9).

Key Issues: The advantages of graph representation and computation are that they can well capture and encode the actual network topology and make efficient resource allocation or offloading decisions based on the encoded features, while being able to handle dynamic changes in the network topology. However, the current research suffers from high training costs, requires the global knowledge of network topology [36], [37], as well as lacks consideration of D-MEC scenarios. In contrast, SITOff enables each MD to make size-insensitive offloading decision based on the partial network topology within its visibility in D-MEC system.

D. Motivation

Compared to existing work, either centralized or decentralized offloading, this work proposes to learn the offloading policy for each MD based on the decentralized training and decentralized execution (DTDE) paradigm, fully avoiding the dependence of centralized venues whether during execution or training. By doing so, this work alleviates the dependence of task offloading on: (1) global states collection from MDs and offloading actions dispatching to MDs, (2) centralized critic for training decentralized offloading policies or centralized controller for generating all MDs' offloading decisions, (3) fixed number of MDs during centralized training and same number of MDs during decentralized execution as training. These independences also imply that each MD could join in or leave out of the D-MEC dynamically, insensitive to the number of MDs in D-MEC during training or execution. Hence, this work could also collaborate with curriculum learning, as shown in the section of experiments, to achieve large-size D-MEC offloading, directly training on which using existing methods could be difficult.

III. System Model

As shown in Fig. 3, we consider a D-MEC consisting of a BS (deployed with an ES), and multiple MDs $\mathcal{M} = \{1, 2, \ldots, M\}$.

TABLE I
SUMMARY OF PRIMARY NOTATIONS

Notation	Description
$\mathscr{M}$	The set of MDs
N	The number of time slots
l_n^m	The location of MD m in slot n
Γ_n^m	The task arriving at MD m in slot n
d_n^m	The task size of Γ_n^m
c	The amount of CPU cycles per bit of tasks
e^{B}	The MD's initial energy budget
f^{ES}	The computing capacity of ES
f^m	The computing capacity of MD m
b_n^m	The offloading decision variable for Γ_n^m
α_n^m	The percentage of local-computing Γ_n^m
β_n^m	The percentage of D2D-computing Γ_n^m
γ_n^m	The percentage of edge-computing Γ_n^m
i_n^m	The chosen MD to D2D-computing Γ_n^m
$r_n^{m,\cdot}$	The D2D/D2E transmitting rate of MD m
$p^{m,\cdot}$	The D2D/D2E transmitting power of MD m
$t_n^{m,\mathrm{loc}}$	The task latency for local-computing α_n^m
$t_n^{m,\mathrm{D2D}}$	The task latency for D2D-computing β_n^m
$t_n^{m,\mathrm{D2E}}$	The task latency for D2E-computing γ_n^m
$t_n^{m,\Gamma}$	The task latency of Γ_n^m
$e_n^{m,\mathrm{loc}}$	The energy consumption for local-computing α_n^m
$e_n^{m,\mathrm{D2D}}$	The energy consumption for D2D-computing β_n^m
$e_n^{m,\mathrm{D2ETran}}$	The energy consumption of MD m to transmit γ_n^m
$e_n^{m,\Gamma}$	The energy consumption of Γ_n^m
$e_n^{m,\mathrm{MD}}$	The energy consumption of MD m at slot n
$e_n^{m,\mathrm{B}}$	The available energy of MD m at slot n
c_n^m	The cost for executing Γ_n^m
c_n	The average cost of all MDs at slot n
c	The average system cost of an episode

Each MD $m \in \mathscr{M}$ is powered by battery with energy budget e^{B}. The time horizon is equally divided into $\mathscr{N} = \{1, 2, \ldots, N\}$ slots, each length of which equals to τ. At each slot $n \in \mathscr{N}$, a task $\Gamma_n^m = \{d_n^m, c, t^{\max}\}$ arrives at each MD m, where d_n^m, c and $t^{\max}$ represent the task size, required amount of CPU cycles per bit, and maximum tolerable latency, respectively. Additionally, each MD moves randomly[1] at each slot n, into or out of the BS-serviced area which will increase/decrease the number of service devices in the system, resulting in the dynamic change of M in the D-MEC system. Under euclidean coordinates, the positions of MD m and BS at slot n are denoted as $l_n^m = (x_n^m, y_n^m, 0)$ and $l^{\mathrm{BS}} = (x^{\mathrm{BS}}, y^{\mathrm{BS}}, h^{\mathrm{BS}})$, respectively. The ES and MD m's computing capacities are denoted as f^{ES} and f^m, where generally $f^{\mathrm{ES}} \gg f^m$ for $\forall m \in \mathscr{M}$, thus MDs tend to offload their tasks to the ES. Simultaneously, MDs' computing capacities f^m are of large variety, so thin MDs like smart phones could also choose to offload their tasks to thick MDs like intelligent vehicles. The primary notations used in this paper are summarized in Table I.

[1]The random movements of MDs could be implemented using different mobility models, that are featured by moving speeds, directions, etc.

A. Partial Offloading Mode

As most recent works [10], [40], [41] in MEC, we consider partial task offloading in this work, which could also be easily extended to binary offloading by constraining the offloading proportion to be two values (0 and 1). At each slot n, the task offloading decision for Γ_n^m is denoted by a four-tuple variable,

$$b_n^m = \{\alpha_n^m, \beta_n^m, \gamma_n^m, i_n^m\}, \tag{1}$$

$$\text{s.t.} \quad \alpha_n^m + \beta_n^m + \gamma_n^m = 1, \tag{2}$$

$$\alpha_n^m, \beta_n^m, \gamma_n^m \geq 0, \tag{3}$$

$$i_n^m \in \mathscr{M}_m^R, \tag{4}$$

where $\alpha_n^m, \beta_n^m, \gamma_n^m$ stand for the task percentages of local computing, D2D-computing and edge computing, respectively, and i_n^m denotes the receiving MD for D2D-computing when $\beta_n^m \neq 0$. These variables are defined to support partial offloading and computing locally, on other MD or ES. Here, MDs are assumed to be equipped with both cellular and Wi-Fi Direct/Bluetooth antennas that allow for simultaneous D2E and D2D offloading [12], thus β_n^m, γ_n^m could be both larger than 0. Additionally, Wi-Fi Direct/Bluetooth antennas are generally endowed with short transmission distance R [10] that i_n^m could only be within available MDs $\mathscr{M}_m^R = \{m' \,|\, m' \in \mathscr{M} \setminus \{m\}, \, ||l_n^m - l_n^{m'}||_2 < R\}$. In this work, b_n^m of each MD at each slot n needs careful decision to achieve desirable MEC performance.

B. Communication Model

When a task Γ_n^m is scheduled for D2D-computing ($\beta_n^m \neq 0$) or edge computing ($\gamma_n^m \neq 0$), Γ_n^m should be first offloaded to MD i_n^m or ES via wireless communication. Without loss of generality, each D2D or D2E wireless channel is accessed using orthogonal frequency division multiple-access with sub-channel bandwidth B, so that task offloading through different sub-channels could avoid interfere with each other [11], [42]. Hence, the D2D/D2E transmitting rate of MD m at slot n can be calculated as

$$r_n^{m,\cdot} = B \log\left(1 + \frac{p_n^{m,\cdot} g_n^{m,\cdot}}{\sigma^2}\right), \tag{5}$$

where $\cdot$ could be 'D2D' (if $\beta_n^m \neq 0$) or 'D2E' (if $\gamma_n^m \neq 0$). Here, $p_n^{m,\cdot}$ is MD's D2D/D2E transmitting power that needs careful decision to assure both desirable offloading performance and energy constraint, σ^2 is the white Gaussian background noise, and $g_n^{m,\cdot}$ is the D2D/D2E channel gains. Due to the different wireless transmitting conditions for D2D and D2E links, $g_n^{m,\cdot}$ is modeled in different forms as in existing research [10] as follows,

$$g_n^{m,i_n^m} = 148 + 40 \times \log \mathscr{d}_n^{m,i_n^m}, \tag{6}$$

$$g_n^{m,\mathrm{BS}} = 128.1 + 37.6 \times \log \mathscr{d}_n^{m,\mathrm{BS}}, \tag{7}$$

where $\mathscr{d}_n^{m,i_n^m}$ and $\mathscr{d}_n^{m,\mathrm{BS}}$ are the D2D/D2E distances between MD $m \leftrightarrow$ MD i_n^m and MD $m \leftrightarrow$ BS. In addition, the transmitting power $p^{m,\cdot}$ should not exceed the maximum available power $p^{\max}$, i.e.,

$$p^{m,\cdot} \leq p^{\max}. \tag{8}$$

C. Local Computing Model

If $\alpha_n^m \neq 0$, the α_n^m part of Γ_n^m (abbr. α_n^m) will be executed by MD m locally. The computing latency of α_n^m is given by

$$t_n^{m,\text{loc}} = \frac{\alpha_n^m d_n^m c}{f^m}, \tag{9}$$

and the corresponding energy consumption is represented as

$$e_n^{m,\text{loc}} = \xi (f^m)^2 \alpha_n^m d_n^m c, \tag{10}$$

where ξ is the energy efficiency coefficient of MD's chip [43].

D. D2D Computing Model

If $\beta_n^m \neq 0$, the β_n^m part of Γ_n^m (abbr. β_n^m) will be executed by MD i_n^m through D2D offloading. The task latency of β_n^m could be expressed as

$$t_n^{m,\text{D2D}} = t_n^{m,\text{D2DTran}} + t_n^{m,\text{D2DWait}} + t_n^{m,\text{D2DExe}} + t_n^{m,\text{D2DBack}}, \tag{11}$$

where $t_n^{m,\text{D2DTran}} = \beta_n^m d_n^m / r_n^{m,\text{D2D}}$ is the D2D-transmitting time of β_n^m, $t_n^{m,\text{D2DWait}} = \sum_{m' \neq m, i_n^{m'} = i_n^m, \beta_n^{m'} \neq 0} \mathbb{1}(t_n^{m',\text{D2DTran}} < t_n^{m,\text{D2DTran}}) \beta_n^{m'} d_n^{m'} c / f^{i_n^m}$ is the queue-waiting time[2] of β_n^m on MD i_n^m for the execution of tasks finishing D2D-transmission before β_n^m, $t_n^{m,\text{D2DExe}} = \beta_n^m d_n^m c / f^{i_n^m}$ is the execution time of β_n^m on MD i_n^m, and $t_n^{m,\text{D2DBack}} = d_n^{m,\text{Back}} / r_n^{m,\text{D2D}}$ is the results backhaul time of β_n^m. Here, when a MD m chooses to offload part of its task to its neighbor i_n^m, the D2D-offloaded task needs to wait execution on the neighbor if other MD m' has also chosen to offload to the same neighbor (i.e., $i_n^m = i_n^{m'}$) and finish task transmission before m (i.e., $t^{m',\text{D2DTran}_n} < t^{m,\text{D2DTran}_n}$). Hence, the D2D-wait time of MD m includes all waiting times (each of which is $\beta_n^{m'} d_n^{m'} c / f^{i_n^m}$) for other m'.

The energy consumption $e_n^{m,\text{D2D}}$ for D2D-computing β_n^m can be calculated as

$$e_n^{m,\text{D2D}} = e_n^{m,\text{D2DTran}} + e_n^{i_n^m,\text{D2DExe}} + e_n^{i_n^m,\text{D2DBack}}, \tag{12}$$

where $e_n^{m,\text{D2DTran}} = p_n^{m,\text{D2D}} \beta_n^m d_n^m / r_n^{m,\text{D2D}}$ and $e_n^{i_n^m,\text{D2DExe}} = \xi (f^{i_n^m})^2 \beta_n^m d_n^m c$ are the consumed energy for MD m transmitting β_n^m to MD i_n^m and MD i_n^m executing the transmitted β_n^m, respectively. $e_n^{i_n^m,\text{D2DBack}} = p_n^{i_n^m,\text{D2D}} d_n^{m,\text{Back}} / r_n^{i_n^m,\text{D2D}}$ is the backhaul energy consumption of the task results.

E. Edge Computing Model

If $\gamma_n^m \neq 0$, the γ_n^m part of Γ_n^m (abbr. γ_n^m) will be executed on ES via D2E offloading. The task latency of γ_n^m is given by

$$t_n^{m,\text{D2E}} = t_n^{m,\text{D2ETran}} + t_n^{m,\text{D2EWait}} + t_n^{m,\text{D2EExe}} + t_n^{m,\text{D2EBack}}, \tag{13}$$

where $t_n^{m,\text{D2ETran}} = \gamma_n^m d_n^m / r_n^{m,\text{D2E}}$ is the D2E-transmitting time of γ_n^m, $t_n^{m,\text{D2EWait}} = \sum_{m' \neq m, \gamma_n^{m'} \neq 0} \mathbb{1}(t_n^{m',\text{D2ETran}} < t_n^{m,\text{D2ETran}}) \gamma_n^{m'} d_n^{m'} c / f^{\text{ES}}$ is the queue-waiting time of γ_n^m on ES for the execution of tasks finishing D2E-transmitting before γ_n^m, and $t_n^{m,\text{D2EExe}} = \gamma_n^m d_n^m c / f^{\text{ES}}$ is the execution time of γ_n^m on ES. $t_n^{m,\text{D2EBack}} = p_n^{m,\text{D2E}} d_n^{m,\text{Back}} / r_n^{m,\text{D2E}}$ is the results backhaul time of γ_n^m. Here, the meaning of D2E-wait time is similar to D2D-wait time in (11).

The energy consumption of MD m to D2E-transmit γ_n^m to ES can be calculated as

$$e_n^{m,\text{D2ETran}} = p_n^{m,\text{D2E}} \frac{\gamma_n^m d_n^m}{r_n^{m,\text{D2E}}}. \tag{14}$$

Here, the energy consumption of the ES for computing and sending back γ_n^m is omitted similar to [42], [44], since ES is generally endowed with sufficient power supply.

F. System Cost Model

The cost of each task Γ_n^m involves the time cost (the latency of Γ_n^m) and energy cost (the consumed energy of Γ_n^m).

1) Time Cost: Since Γ_n^m could be executed simultaneously on three venues: MD m ($\alpha_n^m \neq 0$), MD i_n^m ($\beta_n^m \neq 0$), and ES ($\gamma_n^m \neq 0$). Hence, the task latency $t_n^{m,\Gamma}$ of Γ_n^m is the maximum time on three venues that can be given by,

$$t_n^{m,\Gamma} = \max\{t_n^{m,\text{loc}}, t_n^{m,\text{D2D}}, t_n^{m,\text{D2E}}\}, \tag{15}$$

which should not exceed the maximum tolerable latency, i.e.,

$$t_n^{m,\Gamma} \leq t^{\max}. \tag{16}$$

2) Energy Cost: Different from the time cost, the consumed energy of Γ_n^m should be the energy consumption on all executed venues (if α_n^m, β_n^m or γ_n^m is not 0), given by

$$e_n^{m,\Gamma} = e_n^{m,\text{loc}} + e_n^{m,\text{D2D}} + e_n^{m,\text{D2ETran}}. \tag{17}$$

Besides, from the perspective of each MD, the energy consumption $e_n^{m,\text{MD}}$ of MD m at slot n can be expressed as,

$$e_n^{m,\text{MD}} = e_n^{m,\text{loc}} + e_n^{m,\text{D2DTran}} + e_n^{m,\text{D2ETran}} + \sum_{m' \neq m} \mathbb{1}(i_n^{m'} = m) e_n^{m',\text{D2DExe}}, \tag{18}$$

where the first three parts represent the consumed energy of MD m for locally-computing α_n^m, D2D-transmitting β_n^m and D2E-transmitting γ_n^m, respectively, and the last is the consumption of MD m for D2D-computing tasks from other MDs.

Let $e_n^{m,\text{B}}$ denote the available energy budget of MD m at slot n, then $e_n^{m,\text{B}}$ can be calculated iteratively as follows,

$$e_n^{m,\text{B}} = e_{n-1}^{m,\text{B}} - e_n^{m,\text{MD}}, \tag{19}$$

where $e_0^{m,\text{B}} = e^{\text{B}}$ is MD's initial energy budget. At each slot n, the available energy budget $e_n^{m,\text{B}}$ should not be less than 0, i.e.,

$$e_n^{m,\text{B}} \geq 0, \quad \forall n \in \mathcal{N}. \tag{20}$$

3) System Cost: The cost c_n^m for executing Γ_n^m is given by,

$$c_n^m = \eta t_n^{m,\Gamma} + (1 - \eta) e_n^{m,\Gamma}, \tag{21}$$

where η is the weight of time cost. Then, the average cost of all MDs at slot n could be expressed as

$$c_n = \frac{1}{M} \sum_{m=1}^{M} c_n^m, \tag{22}$$

[2]First-in-first-serve is assumed for MD's execution of simultaneous D2D-offloading tasks, as well as ES's execution of simultaneous D2E-offloading tasks.

and the average system cost over N slots is calculated as

$$c = \frac{1}{N}\sum_{n=1}^{N} c_n. \tag{23}$$

IV. Problem Formulation and Transformation

A. Problem Formulation

Taking the offloading decision b_n^m and transmitting power $p_n^{m,\cdot}$ as optimizing variables, c can be written as a function of b_n^m and $p_n^{m,\cdot}$, i.e., $c(b_n^m, p_n^{m,\cdot})$. Aiming at minimizing the average system cost, we formulate the optimization problem as

$$\mathscr{P}: \min_{b_n^m,\, p_n^{m,\cdot}} c\,(b_n^m,\, p_n^{m,\cdot}),$$
$$\text{s.t. Eqs. (2), (3), (4), (8), (16), (20).} \tag{24}$$

Theorem 1: The optimization problem $\mathscr{P}$ is a mixed-integer nonlinear programming (MINLP) problem w.r.t. the optimizing variables b_n^m and $p_n^{m,\cdot}$.

Proof: According to the definition of MINLP [45], we analyze the mixed-integer and nonlinear properties of $\mathscr{P}$ from the following aspects.

- *Mixed-Integer of Variables:* For the problem $\mathscr{P}$ in (24), the optimizing variables include b_n^m and p_n^m, where b_n^m is a vector containing integer variables (as the element $i_n^m \in \mathscr{M}_m^R$ could only be integers) and p_n^m is a continuous variable.
- *Nonlinearity of Objectives:* In the optimization problem $\mathscr{P}$, the objective c is a non-linear function because $c(b_n^m, p_n^m)$ includes the non-linear components, $t_n^{m,\text{D2DTran}}$, $t_n^{m,\text{D2ETran}}$, $e_n^{m,\text{D2DTran}}$, $e_n^{m,\text{D2ETran}}$, whose denominators involve the nonlinear function $r_n^{m,\cdot}$ w.r.t. p_n^m.
- *Nonlinearity of Constraints:* In the optimization problem $\mathscr{P}$, the constraints (2), (3), (4), and (8) are linear, while the constraints (16) and (20) are non-linear due to the involvement of $r_n^{m,\cdot}$.

Hence, according to the definition of MINLP, the optimization problem $\mathscr{P}$(24) is an MINLP problem.

Theorem 2: The optimization problem $\mathscr{P}$ in (24) is NP-hard.

Proof: We first give a brief description of the capacitated facility location problem (CFLP) which is a well-known NP-hard optimization problem [46], then induce our problem $\mathscr{P}$ into a CFLP (denoted as $\mathscr{P}_{\text{CFLP}}$), showing the NP-hardness of $\mathscr{P}$.

- In the CFLP, there are $\mathbb{N}$ customers and $\mathbb{M}$ facilities. Let d^n represents the demand of customer n ($n \in \{1, 2, \ldots, \mathbb{N}\}$), who can divide their demands into several parts and send them to various facilities for production. Additionally, each facility m ($m \in \{1, 2, \ldots, \mathbb{M}\}$) is assigned a production capacity c^m, which is the maximum producible quantity of the facility m, ensuring facilities operate within their capacity constraints. Let o_m denote the cost of operating facility m, and g_m^n denote the cost of transporting the product from facility m to customer n. Therefore, the overall cost is the sum of the facility-operating cost and the product-transporting cost. $\mathscr{P}_{\text{CFLP}}$ needs to decide 1) how many parts should each customer's demand be divided, 2) which facility should be sent to for each customer's demand-part, 3) which demand-part should be chosen by each facility to meet (i.e., conduct operation and transportation), with the lowest overall cost.
- By analogizing the factors of problem $\mathscr{P}$ to the corresponding ones in $\mathscr{P}_{\text{CFLP}}$, the problem $\mathscr{P}$ can be seen as a CFLP.
  1) Analogize the number of BSs and MDs in the problem $\mathscr{P}$ to the number of facilities and customers in $\mathscr{P}_{\text{CFLP}}$.
  2) View the task size d^m of MD m in $\mathscr{P}$ as the demand of customer n in $\mathscr{P}_{\text{CFLP}}$.
  3) View the computing capacity f^{ES} of BS and $f^{i_n^m}$ of D2D-computing MD i_n^m in $\mathscr{P}$ as the production capacity c^m in $\mathscr{P}_{\text{CFLP}}$.
  4) Consider the system cost c for transmitting and executing MDs' tasks in $\mathscr{P}$ as the facility-operating and product-transporting costs in $\mathscr{P}_{\text{CFLP}}$.

Therefore, the problem $\mathscr{P}$ can be induced to $\mathscr{P}_{\text{CFLP}}$, thus is also NP-hard. □

B. Problem Transformation

On account of the hardness in directly and rapidly solving the above MINLP problem using traditional programming methods, RL could be a more appropriate paradigm to solve (24). Hence, we first transform (24) into a MDP (a mathematical framework of RL), then propose a RL-based offloading algorithm in the following section.

According to (23), the minimization of c is equal to minimize accumulated costs of c_n over N slots, similar to the objective of discrete-time Markov decision process (MDP) that maximizes accumulated rewards over multiple steps [47]. Therefore, we take a MDP, generally defined by a 5-tuple $(s_n, a_n, P, \gamma, r_n)$, to reformulate $\mathscr{P}$ as follows,

$$\mathscr{P}': \max_{a_n} \frac{1}{N}\sum_{n=1}^{N} \gamma^{n-1} r_n, \tag{25}$$

where $s_n = \{d_n^m, l_n^m, e_n^{m,\text{B}}, f^m \mid m \in \mathscr{M}\}$ is the *system state*[3] of D-MEC at slot n, $a_n = \{b_n^m, p_n^{m,\cdot} \mid m \in \mathscr{M}\}$ is the *offloading and transmitting actions* of all MDs at slot n, $P(s_{n+1}|s_n, a_n)$ is the *transition probability* for D-MEC transiting from s_n to s_{n+1} after taking action a_n (usually not known as a prior), r_n is the *immediate reward* at slot n when D-MEC transits from s_n to s_{n+1} after taking action a_n, and γ is the *discount factor* of future rewards.

Considering the constraints of (2), (3), (4), (8), (16), (20), we define the immediate reward r_n, by taking into account not only the reward related to maximize $-c_n$ but also the penalty due to violating these constraints, i.e.,

$$r_n = w_{\text{r}}(-c_n) + w_{\text{p}}(\rho_{\text{vp}} + \rho_{\text{vl}} + \rho_{\text{ve}}), \tag{26}$$

where w_{r} and w_{p} are the weights for the reward and penalties, respectively, and $\rho_{\text{vp}}, \rho_{\text{vl}}, \rho_{\text{ve}}$ are the penalties that depend on the extent to which a_n violates the transmitting power, task

[3]Time-invariant variables, e.g., $f^{\text{ES}}, e^{\text{B}}, c, t^{\max}, l^{\text{BS}}, R$, are omitted for brevity. Though these variables do not change with time slots, they could facilitate offloading algorithms to understand the system properties of D-MEC, without the need of learning them from massive offloading experiences, thus promoting the training of offloading algorithms.

latency and energy budget constraints. Concretely speaking, the $\wp_{\text{vp}}, \wp_{\text{vl}}, \wp_{\text{ve}}$ are defined as follows:

$$\wp_{\text{vp}} = \mathbb{1}(p^{m,\cdot} > p^{\max}) w_{\text{p}} \cdot (p^{m,\cdot} - p^{\max}), \tag{27}$$

$$\wp_{\text{vl}} = \mathbb{1}(t_n^{m,\Gamma} > t^{\max}) w_{\text{p}} \cdot (t_n^{m,\Gamma} - t^{\max}), \tag{28}$$

$$\wp_{\text{ve}} = \mathbb{1}(e_n^{m,\text{B}} < 0) w_{\text{p}} \cdot |e_n^{m,\text{B}}|. \tag{29}$$

V. Offloading Algorithm

With the problem $\mathscr{P}$, RL can be directly used to design centralized offloading algorithms for D-MEC, like most existing works [15], [16]. Concerning the scalability of D-MEC, MARL is a more promising paradigm for distributed offloading, such as D-MEC works using MADDPG in [24], [25], MATD3 in [48] and MA-DDQN in [27]. However, these distributed offloading algorithms are primarily based on centralized training and distributed execution where a central critic is built for each MD to guide its training. This prevents the offloading algorithm from easily adapting to D-MEC with different $\mathscr{M}$s, since the dimension of the central critic is fixed to M. Therefore, we design our MARL-based offloading algorithm directly using distributed training without maintaining any central venues, i.e., each MD determines its offloading action $a_n^m = \{b_n^m, p_n^{m,\cdot}\} \leftarrow \pi(s_n^m)$ using a policy π according to its local state $s_n^m = \{d_n^m, l_n^m, e_n^{m,\text{MD}}, f^m\}$. Taking advantage of the inherent graph-like structure of D-MEC, we use graph to represent D-MEC and form each MD's local knowledge about D-MEC through graph computation, so that the topoligical feature of D-MEC with varying numbers of MDs could be learned dynamically. Besides, considering the limitation of each MDs' local knowledge on making distributed offloading decisions to maximize the whole D-MEC performance, we design a knowledge-exchange mechanism by utilizing D2D transmitting, and aggregate each MD's own knowledge and neighbors' knowledge to enhance the coordination of MDs' offloading decisions. Furthermore, concerning the different importance[4] of neighbors' knowledge on each MD's local action a_n^m, we adopt attention mechanism to selectively learn its neighbors' knowledge.

A. Algorithm Overview

As shown in Fig. 4, the offloading algorithm of MD m receives its neighbor states $s_n^{m'}$ (via D2D transmission with MD $m' \in \mathscr{M}_m^R$) along with its own state s_n^m (locally read from MD m itself), from the D-MEC environment. Then, the neighbor states $s_n^{m'}$ are aggregated into a fixed-size GNN-based [49] representation $e_n^{m,\text{neigh}}$ and the own state s_n^m is encoded into a fixed-size embedding e_n^m, which two are subsequently concatenated together to form the knowledge k_n^m of MD m representing its understanding about its local D-MEC environment. Next, the knowledge k_n^m of MD m, along with its neighbors knowledge

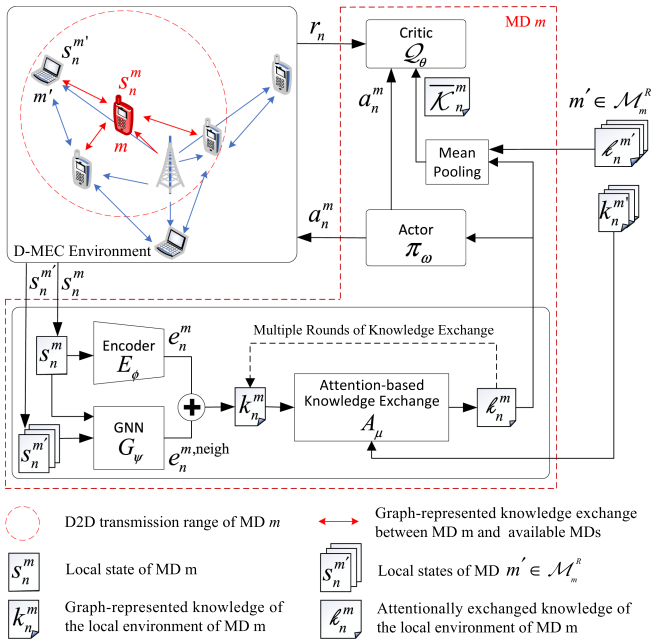


Fig. 4. The algorithm overview of SITOff.

$k_n^{m'}$,[5] $m' \in \mathscr{M}_m^R$, are taken by an Attention-based Knowledge Exchange (AKE) mechanism to selectively absorb its neighbors' knowledge and form a comprehensive knowledge $\hbar_n^m$ about its local D-MEC environment. In addition, the AKE could run multiple rounds, so that MD m could perceive a wider range of its local D-MEC environment. Finally, the comprehensive knowledge $\hbar_n^m$ is fed into the Actor π_ω to make an offloading action a_n^m, and into the Critic $\mathcal{Q}_\theta$ to evaluate the long-term reward (mean pooling is adopted to aggregate $\hbar_n^m$ of MD m and $\hbar_n^{m'}$[6] of available neighbors).

B. Graph-Represented Knowledge of Local States

Constrained by the D2D-transmitting distance, each MD m could only observe the local states of MDs within the range $\mathscr{M}_m^R$, as shown by the red circle in Fig. 4. In view of the inherent graph structure between MD m and observable MDs, we define a D2D graph $\mathscr{G}_n^m = (\mathscr{V}_n^m, \mathscr{E}_n^m)$ for each MD m at slot n to dynamically represent its local states.

1) Graph Representation and Computation: The nodes $\mathscr{V}_n^m$ denote the observable MDs, and each node $\upsilon_n^{m'} \in \mathscr{V}_n^m$ is represented by the node's task size, location, left energy budget and computing capacity, i.e., $\upsilon_n^{m'} = \{d_n^{m'}, l_n^{m'}, e_n^{m',\text{B}}, f^{m'}\}$.

The edges $\mathscr{E}_n^m$ stand for the D2D transmission capabilities, and each edge $\epsilon_n^{mm'}$ is represented by the distance $\mathscr{d}_n^{m,m'}$ between MD m and its neighbor MD m'.

MD m aggregates all the neighbors' states into a fixed-size embedding $e_n^{m,\text{neigh}}$ by using a GNN, each layer of which is

[4]The neighbors m' of a MD m has different capacities $f^{m'}$, task sizes $d_n^{m'}$, locations $l_n^{m'}$ and energy budgets $e_n^{m',\text{B}}$ that play distinct impacts on the D2D-action elements β_n^m, i_n^m, $p_n^{m,\text{D2D}}$, as well as some extent of impacts on the D2E-action elements γ_n^m, $p_n^{m,\text{D2E}}$, of MD m.

[5]The neighbors' knowledge $k_n^{m'}$ comes from other MDs $m' \in \mathscr{M}_m^R$ via D2D communication.

[6]The neighbors' comprehensive knowledge $\hbar_n^{m'}$ comes from other MDs $m' \in \mathscr{M}_m^R$ via D2D communication.

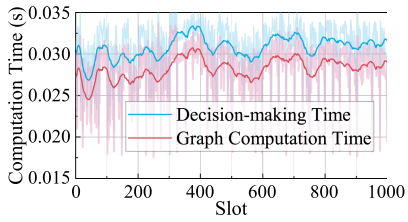


Fig. 5. Cost evaluation on the time taken for graph construction and computation of each MD's local knowledge.

defined as

$$e_n^{m,\text{neigh}} = G_\psi(s_n^m, s_n^{m'}) = \sum_{m' \in \mathscr{M}_m^R} \kappa^{mm'} W_2\, s_n^{m'}, \tag{30}$$

with $\kappa^{mm'}$ being calculated as

$$\kappa^{mm'} = \text{softmax}\left(\frac{(W_3\, s_n^m)^\mathsf{T}(W_4\, s_n^{m'} + W_5\, e_n^{mm'})}{|d^\text{G}|}\right), \tag{31}$$

where $\{W_\iota | \iota = 1, \ldots, 5\}$ are the neural network parameters of $\mathscr{G}$, and $|d^\text{G}|$ is the output dimension of $\mathscr{G}$. For brevity, we denote all the parameters of $\mathscr{G}$ as ψ, i.e., $\psi = \{W_\iota | \iota = 1, \ldots, 5\}$.

Apart from the neighbors' aggregated states embedding $e_n^{m,\text{neigh}}$, MD m also encodes its own state s_n^m into a fixed-size embedding e_n^m using an Encoder E_ϕ (approximated by neural network parameters ψ), which is expressed as $e_n^m = E_\phi(s_n^m)$. Then, e_n^m is concatenated with $e_n^{m,\text{neigh}}$ to form the graph-represented knowledge k_n^m about the local D-MEC environment of MD m, which is formulated as

$$k_n^m = E_\phi(s_n^m) \oplus e_n^{m,\text{neigh}}. \tag{32}$$

2) Time Cost Evaluation of Vanilla Graph Computation: We evaluate the time consumption of the graph construction and computation in SITOff with 30 MDs, using a single Nvidia RTX3080 GPU, as shown in Fig. 5. We find the average time taken for the graph computation of each MD's local knowledge during one offloading decision-making is 0.02816 s, as shown by the red line. At the same time, we also report the average time taken by SITOff to generate one offloading decision, as shown by the blue line. We calculate the proportion of graph-computation time to decision-making time, and find that the graph-computation time accounts for almost 91.64% of the decision-making time.

Thus, to reduce the time cost of the graph structure during building each MD's local knowledge, we adopt the idea of Mean Field (MF), whose efficacy has been verified in recent studies [50][51] on MARL, and replace the vanilla time-consuming graph computation with MF-based method.

3) Low-Cost Graph Computation Based on Mean Field: In view of the large time cost in the vanilla graph computation for each MD's local knowledge, we adopt the weighted MF method to approximate the state embeddings of MDs' neighbors $e_n^{m,\text{neigh}}$ as follows,

$$e_n^{m,\text{neigh}} = \frac{1}{W_m} \sum_{m' \in \mathscr{M}_m^R} w^{mm'} s_n^{m'},$$

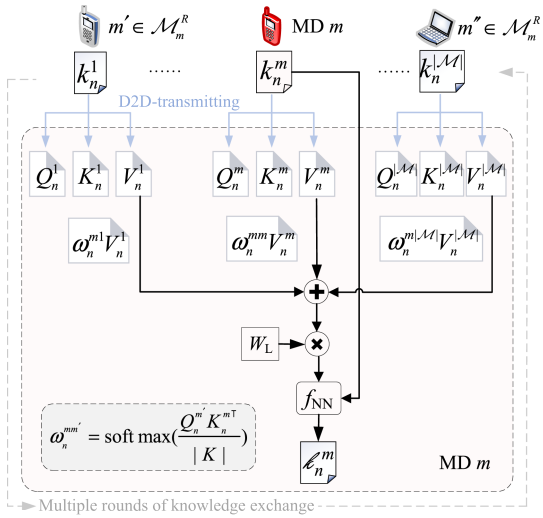


Fig. 6. The schematic of attention-based knowledge exchange.

$$W_m = \sum_{m' \in \mathscr{M}_m^R} w^{mm'}. \tag{33}$$

The $w^{mm'}$ in (33) is a weight of the embedding of the neighbor m' of MD m, which reflects the importance of m' on MD m and can be calculated by Attention-based Knowledge Exchange in the next subsection. The computation of $e_n^{m,\text{neigh}}$ for each MD in the vanilla SITOff in (30), involves multiple matrices ψ ($W_\iota, \iota = 1, \ldots, 5$). In contrast, the low-cost SITOff using weighted MF alleviates the heavy burden on complex matrix calculations but also guarantees the knowledge quality of each MD after graph computation.

C. Attention-Based Knowledge Exchange

For each MD m, after forming the knowledge k_n^m, we propose an Attention-based Knowledge Exchange (AKE) mechanism, as shown in Fig. 6, to build a comprehensive knowledge κ_n^m for each MD by selectively learning neighbors' knowledge. Specifically, AKE includes the following steps:

1) Knowledge Decomposition: Each MD m calculates a key $K_n^m = W_K k_n^m$, query $Q_n^m = W_Q k_n^m$ and value $V_n^m = W_V k_n^m$ based on its own knowledge k_n^m, where W_K, W_Q and W_V are linear matrix parameters.

2) Knowledge Transmitting: Each MD m receives the queries $Q_n^{m'}$ and values $V_n^{m'}$ from MD $m' \in \mathscr{M}_m^R$ via D2D-transmitting, where $Q_n^{m'}$ encodes the extent of impacts of MD m' on other MDs and $V_n^{m'}$ encodes the intended exchanging knowledge of MD m', respectively.

3) Attention Weighting: Each MD m uses its own key K_n^m (encoding its knowledge-absorbing intention from different neighbors) to compute the dot products with all received queries $Q_n^{m'}$, each of which is then scaled by $1/|K|$ where $|K|$ is the key's dimension. Corresponding to (33), the scaled dot products are then used to compute the attention weights $\omega_n^{mm'} = \text{softmax}(Q_{n,\tilde{\mathcal{Q}}}^{m'} K_{n,\tilde{\mathcal{Q}}}^{m\mathsf{T}} / |K|)$ and $u_n^{mm'} =$

softmax$(Q^{m'}_{n,\hat{\pi}} {K^{m}_{n,\hat{\pi}}}^{\top}/|K|)$ for each received knowledge of neighbors $m' \in \mathscr{M}^R_m$.

4) Knowledge Aggregation: Each MD m selectively aggregates the knowledge of its neighbors by performing the weighted-sum of $V^{m'}_n$, $m' \in \mathscr{M}^R_m$, along with a linear transformation, i.e., $V^{m,\text{agg}}_n = W_{\text{L}} \sum \omega^{mm'}_n V^{m'}_n$. Then, MD m forms its comprehensive knowledge $\mathscr{k}^m_n$ about the D-MEC through a non-linear transformation f_{NN} on both the aggregated knowledge $V^{m,\text{agg}}_n$ and original knowledge k^m_n, by using a neural network f_{NN}.

With multiple rounds of the above four steps of knowledge exchange, each MD m could learn knowledge from MDs that might not be observable (within the direct D2D-transmission range R) and perceive a wider environment of the D-MEC, substantially benefiting its local distributed offloading decision-making. Finally, we denote the AKE as A_μ where μ stands for all the involved parameters in the above four steps, including $W_K, W_Q, W_V, W_{\text{L}}$ and the parameter of f_{NN}.

D. Distributed DRL-Based Offloading

After knowledge exchange, we can adopt a completely distributed DRL algorithm to make decisions for each MD's offloading action, during both training and running irrespective of the dynamic variation of $\mathscr{M}$ in D-MEC.

1) Actor-Critic DRL framework: An Actor π_ω (approximated by neural network parameters ω) is built to take each MD m's knowledge $\mathscr{k}^m_n$ as input and generate the desirable action $a^m_n = \pi(\mathscr{k}^m_n)$ for MD m, and a Critic $\mathscr{Q}_\theta$ (approximated by neural network parameters θ) is built to take as input the generated action a^m_n, along with the mean pooled knowledge $\bar{\mathscr{K}}^m_n$ of MD m and its neighbors, to estimate the long-term reward $\mathscr{Q}(\bar{\mathscr{K}}^m_n, a^m_n)$ of the knowledge-action pair. With the designed homogeneous model structures of Actor and Critic, MDs can share the parameters of π_ω and $\mathscr{Q}_\theta$, along with the parameters of Encoder E_ϕ, GNN G_ψ and AKE A_μ, leading to improved learning efficiency. Besides, keeping only one set of models for all MDs, makes the building of these models independent of the number of MDs during whether training or execution, leading to promising scalability.

2) Training of SITOff: During training, since the objective of the problem $\mathscr{P}$ is to minimize the average system cost c, the immediate reward r_n in (26) is generated only when each MD has executed the π_ω model that produces each individual action a^m_n and the D-MEC environment has seen the joint action $a_n = \{a^m_n | m \in \mathscr{M}\}$ of all MDs. It means all MDs share the global reward r_n that guides them to cooperatively solve the problem $\mathscr{P}$. The whole training process of SITOff is presented in Algorithm 1. Similar to [24], [25], we can update Actor π_ω using the sample policy gradient from the experience buffer $\mathscr{B}$ by,

$$\nabla_\omega J = \mathbb{E}_{\mathscr{B}} \left[\nabla_{\pi(\mathscr{k}^m_n)} \mathscr{Q}(\bar{\mathscr{K}}^m_n, \pi(\mathscr{k}^m_n)) \nabla_\omega \pi(\mathscr{k}^m_n) \right]. \tag{34}$$

Meanwhile, we optimize the Critic $\mathscr{Q}_\theta$ by minimizing the loss,

$$L(\theta^Q) = \mathbb{E}_{\mathscr{B}} \left[\left(\mathscr{Q}(\bar{\mathscr{K}}^m_n, a^m_n) - y^m_n \right)^2 \right], \tag{35}$$

Algorithm 1: The Training Algorithm of SITOff.

Initialize parameters $\omega, \theta, \phi, \psi, \mu$ for $\pi_\omega, \mathscr{Q}_\theta, E_\phi, G_\psi, A_\mu$.
Initialize experience buffer $\mathscr{B}$
repeat
  Initialize the D-MEC environment ζ
  ζ generates the initial global state s_0
  for each slot n **do**
    // Graph Knowledge of Local States in Section V-B
    for each MD m **do**
      Observes its state s^m_n
      Receives neighbor states $s^{m'}_n$, $m' \in \mathscr{M}^R_m$
      Forms knowledge k^m_n by (32)
    end for
    // Attention-based Knowledge Exchange in Section V-C
    Perform multiple rounds of AKE: $\mathscr{k}^m_n = A_\mu(k^m_n)$
    // Actor-Critic DRL framework in Section V-D1
    For each MD m: Generate an action $a^m_n = \pi_\omega(\mathscr{k}^m_n)$
    Form the global action $a_n = \{a^m_n \,|\, m \in \mathscr{M}\}$
    Interact with ζ using a_n
    Receive a global reward r_n and state s_{n+1} from ζ
    // Training of SITOff in Section V-D2
    for each MD m **do**
      Observes its new state s^m_{n+1}
      Stores $\{s^m_n, a^m_n, r_n, s^m_{n+1}\}$ into $\mathscr{B}$
      Periodically:
        Samples a batch of experience $\mathscr{E}$ from $\mathscr{B}$
        Update ω, ϕ, ψ, μ by (34) and θ by (35) and (36) using $\mathscr{E}$
    end for
  end for
until convergence
return $\omega, \theta, \phi, \psi, \mu$

where

$$y^m_n = r_n(s^m_n, a^m_n) + \gamma \mathscr{Q}(\bar{\mathscr{K}}^m_{n+1}, \pi(\mathscr{k}^m_{n+1})). \tag{36}$$

According to the design of the Encoder, GNN and AKE models, integrated with the Actor-Critic DRL framework, the whole size-insensitive offloading algorithm SITOff can be trained in an end-to-end manner, after which the parameters $\omega, \theta, \phi, \psi, \mu$ for $\pi_\omega, \mathscr{Q}_\theta, E_\phi, G_\psi, A_\mu$ models are output.

3) Execution of SITOff: When SITOff finishes training, we can perform size-insensitive offloading action generation using the output offloading model π_ω, encoding model E_ϕ, GNN model G_ψ and AKE model A_μ. According to Algorithm 2, each MD observes its local state s^m_n from the environment and generates its action a^m_n. The D-MEC environment receives the joint action $a_n = \{a^m_n | m \in \mathscr{M}\}$ and generates the next state $s_{n+1} = \{s^m_{n+1} | m \in \mathscr{M}\}$ and the global reward r_n. Each MD then observes its new local state s^m_{n+1} and generates the next action a^m_{n+1}, and so on until the end of the epoch. During execution, although MDs are deployed with the same offloading model and receive the same global reward, it does not prevent each

Algorithm 2: The Execution Algorithm of SITOff.

1: Input parameters $\omega, \theta, \phi, \psi, \mu$ for $\pi_\omega, \mathcal{Q}_\theta, E_\phi, G_\psi, A_\mu$ and the number of MDs M.
2: Initialize the D-MEC environment $\bar{\zeta}$
3: **repeat**
4: $\bar{\zeta}$ generates the initial global state s_0
5: **for** each slot n **do**
// Graph Knowledge of Local States in Section V-B
6: **for** each MD m **do**
7: Observes its state s_n^m
8: Receives neighbor states $s_n^{m'}$, $m' \in \mathscr{M}_m^R$
9: Forms knowledge k_n^m by (32)
10: **end for**
// Attention-based Knowledge Exchange in Section V-C
11: Perform multiple rounds of AKE: $\mathscr{k}_n^m = A_\mu(k_n^m)$
// Actor-Critic DRL framework in Section V-D1
// Execution of SITOff in Section V-D3
12: For each MD m: Generate an action $a_n^m = \pi_\omega(\mathscr{k}_n^m)$
13: Form the global action $a_n = \{a_n^m \mid m \in \mathscr{M}\}$
14: Interact with ζ using a_n
15: Receive a global reward r_n and state s_{n+1} from $\bar{\zeta}$
16: **end for**
17: **until** Epoch end
18: **return**offloading action a in current epoch

MD from individually observing its local state, perceiving the local D-MEC environment, exchanging its desired knowledge and making its own offloading decision.

E. Convergence and Complexity Analysis

Since SITOff is a distributed DRL-based offloading algorithm, and each MD m learns the offloading policy through attention-based knowledge exchange to achieve a promising performance for the whole system, it is necessary to analyze the convergence and computational complexity of the Algorithm 1. SITOff employs neural networks, i.e., nonlinear functions, to approximate the optimal policy. Thus, as in [52], we show that SITOff can converge at a rate of $\mathcal{O}(1/\mathscr{T})$, where $\mathscr{T}$ is the number of iterations of parameter updates during algorithm training. The convergence rate of $\mathcal{O}(1/\mathscr{T})$ means that the performance gap between the algorithm solution and the stationary solution is bounded and inversely proportional to the number of iterations of the algorithm. As the number of training iterations increases SITOff is able to gradually converge to a stable solution and eventually achieve convergence. Specifically, we first rewrite (25) in the form of long-term discounted accumulative reward:

$$V^\pi(s_0) = \mathbb{E}\left[\frac{1}{M}\sum_{n=0}^{\infty}\gamma^n\sum_{m=1}^{M} r_n(s_n, a_n)|s_0, \pi_\omega\right]. \tag{37}$$

The Bellman equation of (37) $V^{\mathrm{B}}(s_n) \triangleq \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}[r(s_n, a_n)] + \gamma\sum_{s_{n+1}} P(s_{n+1}|s_n, a_n)V^{\mathrm{B}}(s_{n+1})$ can be solved by minimizing mean-squared projected bellman error (MSPBE) $\frac{1}{2}\|\mathbb{E}_s[(V_\Theta^{\mathrm{B}}(s) - \tilde{V}_\Theta^\pi)\nabla_\Theta \tilde{V}_\Theta^\pi(s)^\top\|^2_{\mathbf{K}_\Theta^{-1}}$ [53], where Θ is the overall parameter of the neural network $\tilde{V}_\Theta^\pi$ proposed above and $\mathbf{K}_\Theta = \mathbb{E}_s[\nabla_\Theta \tilde{V}_\Theta^\pi(s)\nabla_\Theta \tilde{V}_\Theta^\pi(s)^\top]$. According to the Fenchel's duality, we can transform the MSPBE minimization problem into primal-dual minimax problem as follows,

$$\min_{\Theta\in\mathbb{R}}\max_{\Omega\in\mathbb{R}} F(\Theta, \Omega) = \mathbb{E}_s\langle \boldsymbol{\delta}\cdot\nabla_\Theta \tilde{V}_\Theta^\pi(s), \Omega\rangle - \frac{1}{2}\Omega^\top \mathbf{K}_\Theta \Omega, \tag{38}$$

where $\boldsymbol{\delta} \triangleq \frac{1}{M}\sum_{m=1}^{M} r_n(s_n, a_n) + \gamma\tilde{V}_\Theta^\pi(s_{n+1}) - \tilde{V}_\Theta^\pi(s_n)$ and Ω is dual variable. Let $J(\Theta) \triangleq F(\Theta, \Omega^*) = \max_{\Omega\in\mathbb{R}} F(\Theta, \Omega)$ where $\Omega^* = \arg\max_{\Omega\in\mathbb{R}} F(\bar{\Theta}, \Omega)$. We can find the stationary point of $F(\Theta, \Omega)$ by minimizing $J(\Theta)$ to achieve convergence of the algorithm. Based on the assumptions established in [54], We define the convergence rate $\mathscr{R}$ of the algorithm as follows:

$$\mathscr{R} \triangleq \|\nabla J(\bar{\Theta})\|^2 + 2\|\Omega^* - \bar{\Omega}\|^2 + \left(\|\Theta - \bar{\Theta}\|^2 + \|\Omega - \bar{\Omega}\|^2\right), \tag{39}$$

where $\bar{\Theta}$ is a first-order stationary point for $J(\cdot)$, and Ω^* denotes $\Omega^*(\bar{\Theta}) = \arg\max_{\Omega\in\mathbb{R}} F(\bar{\Theta}, \Omega)$. The first term in (39) denotes the convergence of primal variable Ω, i.e., it satisfies $\|\nabla J(\bar{\Theta})\|^2 = 0$. The second term denotes $\bar{\Omega}$'s convergence to the maximizer Ω^* for $F(\bar{\Theta}, \cdot)$. The last term is the average consensus error of each MD m. Similar to the proof of Theorem 1 in [54], we derive the convergence result of Algorithm 1 as:

$$\frac{1}{\mathscr{T}+1}\sum_{n=0}^{\mathscr{T}}\mathbb{E}[\mathscr{R}_n] \le \frac{2\mathbb{E}[\mathscr{C}_0 - J^*]}{\min\{1, L_F^2\}(\mathscr{T}+1)\gamma}, \tag{40}$$

where the potential function $\mathscr{C}_t$ is defined as,

$$\mathscr{C}_t \triangleq J(\bar{\Theta}_t) + \frac{8\lambda L_f^2}{\rho\nu}\|\Omega_t^* - \bar{\Omega}_t\|^2 + (\|\Theta_t - \bar{\Theta}_t\|^2 + \|\Omega_t - \bar{\Omega}_t\|^2 + \lambda\|\mathbf{p}_t - \bar{\mathbf{p}}_t\|^2 + \nu\|\mathbf{d}_t - \bar{\mathbf{d}}_t\|^2). \tag{41}$$

Variables $\mathbf{p}$ and $\mathbf{d}$ are aggregations of neighbor about the graph $\mathscr{G}_n$ corresponding to Ω and Θ, respectively, which are related to the topology of the graph $\mathscr{G}_n$. λ and ν are the update step-sizes of the parameters Ω and Θ. L_f^2 and ρ denote the $L_f^2 - LipschitzSmoothness$ constant and positive constant of Strong Concavity in Dual, respectively. From (40) and (41), it is shown that the convergence rate of the Algorithm 1 is $\mathcal{O}(1/\mathscr{T})$ and can be affected by the topology of the graph $\mathscr{G}_n$. Therefore, it also reveals that we need to set the update step-sizes about λ and ν properly (i.e., the learning rate of the neural network). Furthermore, in order to achieve an ϵ^2-stationary solution [55] of Algorithm 1, i.e., satisfying $\frac{1}{\mathscr{T}+1}\sum_{n=0}^{N}\mathbb{E}[\mathscr{R}_n] \le \epsilon^2$, the corresponding communication complexity is $\mathcal{O}(\epsilon^{-2})$, where ϵ is convergence accuracy. It represents the total rounds in which each MD sends or receives knowledge k from its neighbors until algorithm converges to the accuracy of ϵ. It is worth noting that the knowledge k is a $|k|$-dimension vector, and its data size is much smaller than the data size d_n^m arrived at MD m, so it will not impose an excessive communication burden on the whole system.

For the computation complexity of Algorithm 1, it mainly depends on the operations related to the graph $\mathscr{G}_n$. When the state of M MDs encoded in graph $\mathscr{G}_n$, the feature projection

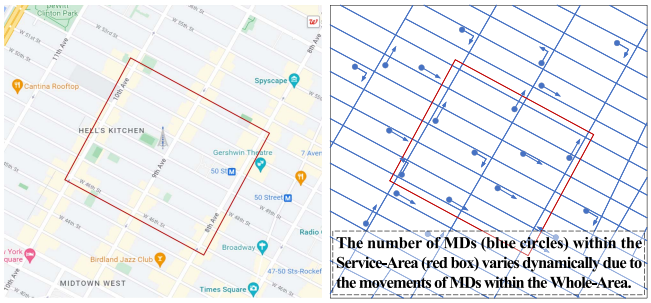


Fig. 7. The investigated area of Manhattan city map.

complexity is $\mathcal{O}(M \times S \times |k_n^m|)$, where S and $|k_n^m|$ are dimension of state and knowledge k_n^m of each MD m respectively. When attention-based knowledge exchange is performed, the computational complexity is $\mathcal{O}(\mathbf{E} \times |k_n^m| \times |\mathcal{K}_n^m|)$, where $\mathbf{E}$ denotes the total number of edges in graph $\mathcal{G}_n$ and $|\mathcal{K}_n^m|$ denotes the dimension of knowledge k_n^m of each MD m respectively. In summary, the total computational complexity is $\mathcal{O}(M \times S \times |k_n^m| + \mathbf{E} \times |k_n^m| \times |\mathcal{K}_n^m|)$.

VI. Performance Evaluation

A. Experimental Setup

We take a 1×1 km^2 area (termed *Whole-Area*) of the Manhattan city as the investigated area, along with the MDs analogous to mobile device-holding walkers or vehicles moving along roads following the Manhattan mobility model [56], as shown in Fig. 7. The BS is located according to the data from Opencellid [57], and we further select the internal 0.5×0.5 km^2 area (termed *Service-Area*) around the BS as the MEC service area where the MEC size dynamically varies due to the mobility of MDs into or out of the BS-serviced area. We visualize the simulation using SUMO [58], [59] based on the above map description, where each main stem is set up with 3 lanes and the locations of MDs are saved at each time slot. The settings of the number of MDs, the moving speed and the crossroad-turning probability at each intersection, along with their impacts on offloading performances, are described in Sections VI-B1, VI-B2 and VI-B3, respectively. The MEC system is built using Python 3.6 and PyTorch 1.10, and primary parameters are listed in Table II. The whole experiment was trained and executed on a desktop equipped with the Intel Xeon Gold 5218 and Nvidia RTX 3080. MDs are deployed in the Whole-Area and categorized according to computing frequencies into two types, namely, *Thick-MDs* and *Thin-MDs*. The number of MDs in the Whole-Area and the ratio of Thick-MDs to Thin-MDs are 30 and 1:1, and the computing frequencies of Thick-MDs and Thin-MDs are 2 GHz and 1GHz, respectively, unless otherwise specified. Concerning the super-parameters of SITOff, we set the DRL-related parameters as $\gamma = 0.95$, $w_{\mathrm{r}} = 100$, $w_{\mathrm{p}} = -1000$.

Performance evaluation metrics include three types. (1) The first is the *average cost* c defined in (23) of all success epochs that represents the quality of service of each offloading approach in MEC. (2) The second is the average *task success rate (TSR)* of all MDs over multiple epochs, defined as $\mathcal{S}^{\mathrm{TSR}} = \sum \mathbb{1}(t_n^{m,\Gamma} < t^{\max}) / \sum \mathbb{1}(t_n^{m,\Gamma})$. (3) The third is the average *epoch success rate (ESR)* of all MDs, defined as $\mathcal{S}^{\mathrm{ESR}} = \sum \mathbb{1}(n = N) / \sum \mathbb{1}(n = 1)$, where n could fail to reach the end N from start 1 of an epoch due to any violation of the constraints in (2), (3), (4), (8), (16), (20).

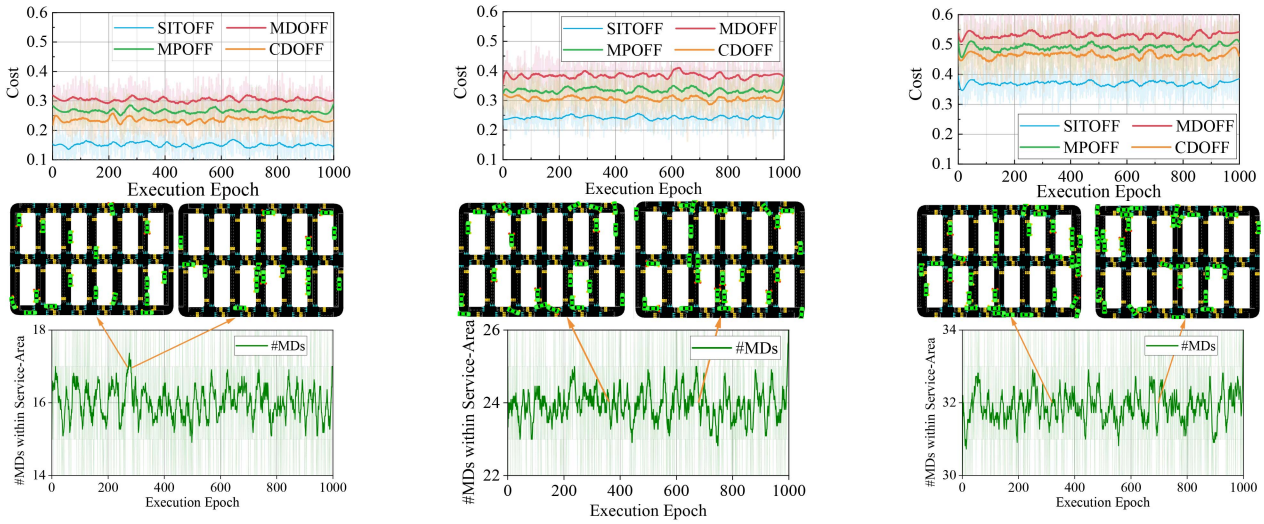
TABLE II
Primary Parameters of MEC System

Parameter	Description	Value
f^{ES}	ES computing capacity	5 GHz
f^m	MD computing capacity	$\{2, 1\}$ GHz
E_{train}	Number of training epochs	2×10^4
E_{exe}	Number of execution epochs	1×10^3
M	Number of MDs	[20, 40]
N	Number of slots	10
N_{batch}	Training batch size	32
B	Bandwidth	10 MHz
τ	Slot length	0.5 s
e^{B}	MD energy budget	7 J
d_n^m	MD task size	[50, 100] kb
c	CPU cycles/bit	900
$t^{\max}$	maximum task delay	0.9 s
h^{BS}	BS height	50 m
R	D2D transmission distance	30 m
L_{lane}	Number of lanes in stem	3
v_{MD}	Moving speed of MDs	{1, 5, 10, 15} m/s
$p^{\max}$	Maximum transmitting power	0.1 W
ξ	Energy efficiency coefficient	10^{-27}
σ^2	White Gaussian noise	-100 dBm
η	Weight of time cost	0.8
w_{r}	Weight of cost	100
w_{p}	Weight of penalty	-1000

Baselines include three state-of-the-art DRL-based offloading approaches, involving both centralized and distributed. (1) MADDPG-based distributed offloading [24], named *MDOff*, that maintains a centralized critic with fixed number of MDs during training and adopts the DDPG as each MD's offloading framework. (2) MAPPO-based distributed offloading [60], named *MPOff*, that is the same as MDOff except for the adoption of the PPO as the MD's offloading framework. Due to the size-sensitivity of MDOff and MPOff, they are trained in D-MEC with 30 MDs. (3) Centralized DDPG-based offloading [15], named *CDOff*, that receives global MEC states and adopts DDPG to make centralized offloading decisions. CDOff assumes a maximum number of MDs as input and pads zeros when the number of MDs is not full.

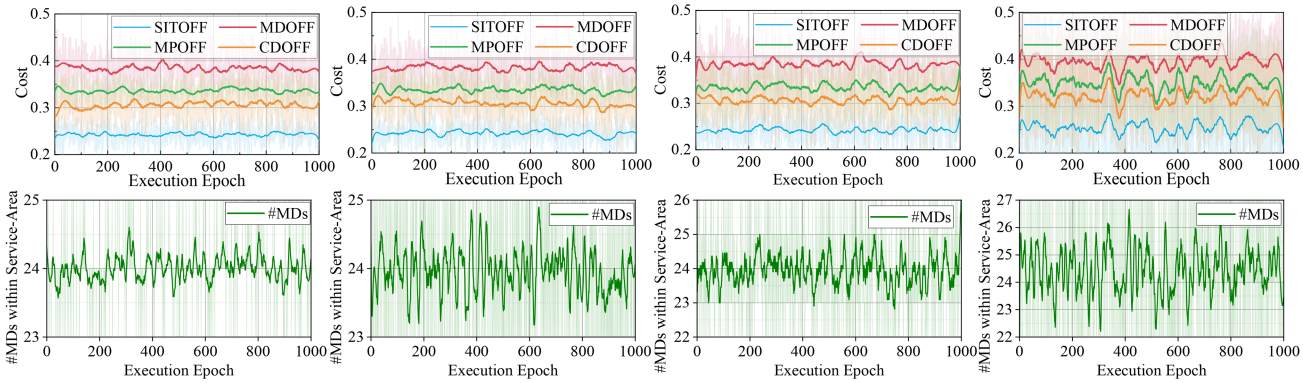
B. Performance Comparison With Dynamic D-MEC Sizes

The D-MEC size, i.e., the number of MDs within the Service-Area, varies dynamically due to various factors of the Whole-Area, e.g., the number of MDs, MD moving speeds, MD crossroad-turning probabilities. Hence, we compare the performance of SITOff with baselines under different settings of these factors. In addition, the compared metric c of different approaches in this part, refers to the cost during execution, where the D-MEC environment could be of different sizes or topology compared to training.



(a) 20 MDs in the Whole-Area. (b) 30 MDs in the Whole-Area. (c) 40 MDs in the Whole-Area.

Fig. 8. Performance comparison between SITOff and baselines, with different numbers of MDs in the Whole-Area (20/30/40).

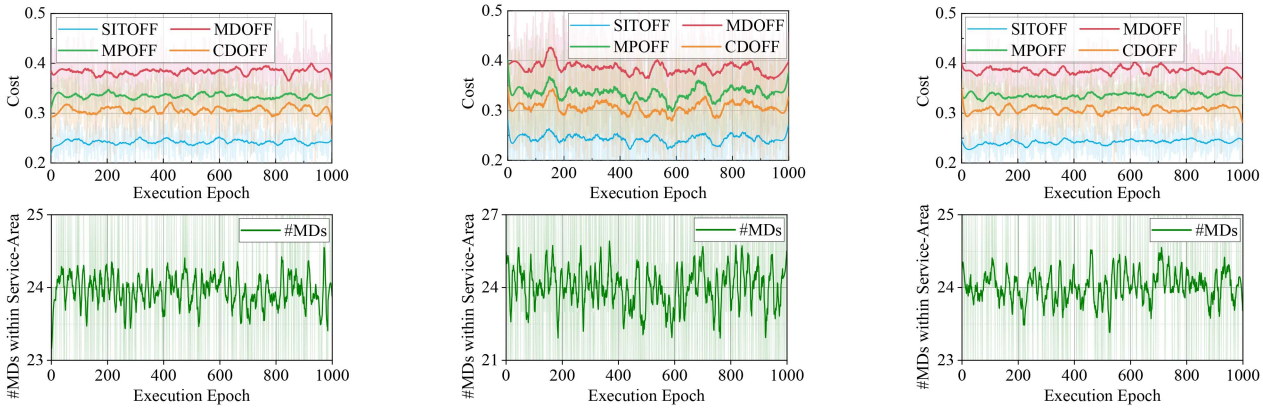


(a) MD moving speeds at 1 m/s. (b) MD moving speeds at 5 m/s. (c) MD moving speeds at 10 m/s. (d) MD moving speeds at 15 m/s.

Fig. 9. Performance comparison between SITOff and baselines, with different MD moving speeds. (1/5/10/15 m/s).

1) Different Numbers of MDs in Whole-Area: As shown in Fig. 8(a), (b), and (c), we deploy 20, 30, 40 MDs in the Whole-Area, respectively. As a result, the number of MDs within the Service-Area exhibits diverse densities, around 16, 24, and 32 in Fig. 8(a), (b), and (c), respectively. Additionally, the topology of MDs in the Service-Area also appears to be time-varying even with the same number of MDs in the Service-Area, as shown by the green vehicles visualized using SUMO [58], [59] in the subplots of the bottom row. It can be seen that MDOff meets severe performance degradation under time-varying MDs, with its cost c up to 0.31, 0.39, and 0.54 in Fig. 8(a), (b), and (c), respectively. Similar cases are also encountered by MPOff, with a little lower costs, probability due to its better adaptability to time-varying MDs than MDOff resulted from its adoption of probabilistic offloading decision-making. CDOff is observed to achieve lower costs than MDOff and MPOff, with its cost around 0.23, 0.30, and 0.49, in the 20/30/40 MDs Whole-Area. However, our approach SITOff is found to outperform CDOff, achieving the lowest costs in all three cases, around 0.15, 0.24, and 0.38 in Fig. 8(a), (b), and (c), respectively. The superiority of SITOff against centralized offloading CDOff could be due to the reason that the access to global states by CDOff, instead of intentionally learned local graph-like knowledge by SITOff, may not help much for making D2D offloading decisions.

2) Different MD Moving Speeds in Whole-Area: In Fig. 9, we deploy 30 MDs in the Whole-Area and set the MDs' moving speeds at 1m/s, 5 m/s, 10 m/s, and 15 m/s, respectively, where the 1 m/s MDs could be smart phones or wearable devices, and the 5 m/s, 10 m/s, 15 m/s MDs could be mobile robots or automated guided vehicles. With different moving speeds, the number of MDs, as well as the topology of MDs, within the Service-Area vary with different rates, as shown by the subplots in (a), (b), (c) and (d), respectively. It can be seen that as the MDs' speeds increase, the cost-fluctuations of different approaches grow gradually. Specifically, When the moving speed is low at 5 m/s in Fig. 9(a), MDOff and MPOff are both observed to achieve the costs around 0.39 and 0.34, along with low fluctuations (standard deviations are 0.031 and 0.027,



(a) Large turning left probabilities. (b) Large going straight probabilities. (c) Large turning right probabilities.

Fig. 10. Performance comparison between SITOff and baselines, with different large turning left/straight/right probabilities of MDs at crossroads.

respectively). However, when the moving speed becomes large to 15 m/s in Fig. 9(c), both MDOff and MPOff see obvious large cost fluctuations (standard deviations are 0.062 and 0.079, respectively), primarily because of their fixed number of MDs during training and lack of adaptability to time-varying D-MEC sizes. Similarly, SITOff shows more satisfactory adaptability than MDOff, MPOff and CDOff to the change of MD moving speeds, with both the lowest cost around 0.25 and standard deviation not larger than 0.059, in all four types of MD speeds.

3) Different MD Crossroad-Turning Probabilities in Whole-Area: In Fig. 10, we deploy 30 MDs in the Whole-Area and set the MDs' turning left, going straight, and turning right probabilities at crossroads as 6/3/1, 2/6/2, and 1/3/6 in (a), (b), and (c), respectively. With different turning probabilities, the number of MDs in the Service-Area is found to be different, with the high going-straight probability showing the largest variance (21-27 around 24) and the turning-left/right probability smaller variance (23-25 around 24). Accordingly, MDOff, MPOff and CDOff are found to achieve large cost fluctuations in (b) than (a) and (c), revealing their weak adaptability to the variation of D-MEC sizes. Similarly, SITOff is noticed to achieve both the lowest costs and cost fluctuations in Fig. 10(a), (b), and (c), suggesting its promising size-insensitivity in D-MEC.

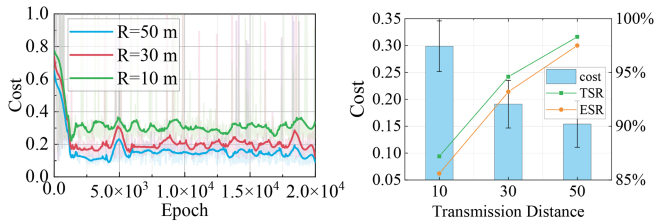
According to the above three experiments, it shows that SITOff can achieve more promising performance than both MARL-based offloading algorithms and centralized DRL-based offloading algorithms, primarily benefiting from its adoption of the graph representation of each MD's local state and the attention-based knowledge exchange with neighbor MDs to facilitate more coordinated offloading of the whole D-MEC.

C. Performance Comparison w.r.t. Costs and Success Rates

We further compare the execution performances of SITOff and baselines with 20/30/40 MDs in the Whole-Area (baselines are trained with 30 MDs due to their sensitivity to D-MEC sizes), by additionally considering TSR and ESR, as shown in Table III (The name suffix "Off" in the second row is omitted due to space limits). The costs of SITOff and baselines are consistent with those in Fig. 8. In Table III, Due to the consistency on the number of MDs with training, MDOff, MPOff and CDOff

TABLE III
PERFORMANCE COMPARISON W.R.T. COST, TSR, AND ESR

MDs	Cost				TSR				ESR			
	MD	MP	CD	SIT	MD	MP	CD	SIT	MD	MP	CD	SIT
20	0.31	0.26	0.23	**0.15**	82.6	84.2	85.7	**99.4**	80.4	81.7	82.5	**98.1**
30	0.39	0.33	0.30	**0.24**	93.4	93.7	94.7	94.6	92.7	92.5	93.3	93.2
40	0.54	0.52	0.49	**0.38**	79.8	80.3	81.5	**91.8**	77.3	78.4	77.0	**90.5**



(a) Training costs. (b) Execution costs/TSR/ESR.

Fig. 11. Performance evaluation of SITOff, with different D2D transmission distances (R=10/30/50 m).

achieve the highest TSR and ESR in the execution D-MEC with 30 MDs (in the second row) compared to other execution D-MECs. Whereas, for TSR and ESR, we can see an obvious performance degradation (down from 90+% to 80+% and 77+%) of MDOff, MPOff, and CDOff when they execute in D-MEC with 20/40 MDs (different from the 30 MDs during training). Comparatively, SITOff could still achieve the highest owing to its promising size-insensitivity.

D. Impacts of Different D-MEC Settings on SITOff

We further evaluate the performance of SITOff with different D-MEC settings, including the D2D transmission distance R, thick/thin MD ratios, and thick/thin MD computing frequencies.

1) Different D2D Transmission Distances: The D2D transmission distances R plays crucial roles in not only the D2D offloading ability but also the knowledge exchange mechanism of SITOff. In Fig. 11, we set three different R (10/30/50 m) and see its impacts on the offloading performance of SITOff. It can be seen that the cost of SITOff reduces dramatically from 0.29 to 0.19 when R changes from 10 m to 30 m, and slightly

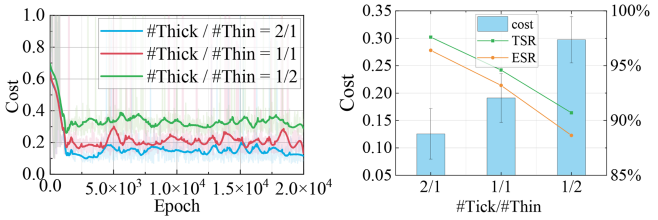


Fig. 12. Performance evaluation of SITOff, with different ratios of Thick-MDs to Thin-MDs (2/1, 1/1, 1/2).

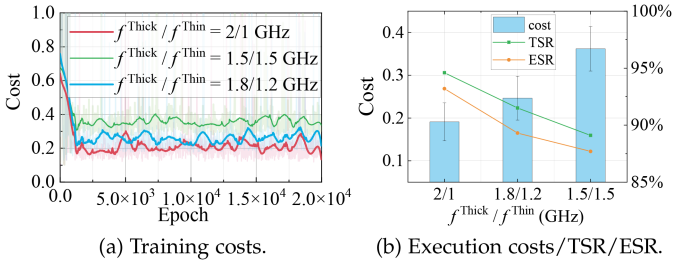


Fig. 13. Performance evaluation of SITOff, with different computing frequencies of Thick-MDs and Thin-MD.

from 0.19 to 0.15 when R grows from 30 m to 50 m, as shown in Fig. 11(a) (. It suggests that the increment of R could give rise to obvious performance improvements when R is small and only slight improvements when R is large. When R grows to the extent capable of covering the Service-Area, the SITOff would degrade into a kind of centralized offloading algorithm like CDOff, except for the individual offloading decision-making by each MD with the global knowledge of the D-MEC formed by attention-based knowledge exchange. Hence, the increment of R when R is large would help little for further improving the offloading performance. During execution, the most desirable cost, TSR, and ESR are also observed when $R = 50$, reaching about 0.15, 98%, and 97%, respectively, as shown in Fig. 11(b).

2) Different Ratios of Thick-MDs to Thin-MDs: In Fig. 12, we deploy 30 MDs in the Whole-Area and set different ratios of thick MDs to thin MDs (2/1, 1/1, and 1/2). When the ratio of thick MDs to thin MDs is 2/1, i.e., 20 thick MDs and 10 thin MDs, we can find the cost is low at about 0.13, mainly due to the large number of thick MDs that meets the requirements of computation-intensive tasks. When the thick MDs is the same as thin MDs, i.e., 1/1, SITOff is also noticed to achieve desirable cost, at about 0.19, which could be resulted from the efficient D2D offloading between thick MDs and thin MDs with the help of the attention-based knowledge exchange between MDs. When the ratio becomes 1/2, the cost sees obvious rise to about 0.30, which could be largely due to the lack of whole available computing resources in the D-MEC. During execution in Fig. 12(b), TSR and ESR are also noticed to achieve the highest rates, 97.6% and 96.4%, when the ratio is 2/1.

3) Different Frequencies of Thick-MDs and Thin-MDs: We also set thick MDs and thin MDs with different computing frequencies, 2/1 GHz, 1.8/1.2 GHz and 1.5/1.5GHz (the whole available computing resources are still the same), as shown in Fig. 13. When f^{Thick} is much larger than f^{Thin} (i.e., 2/1 GHz), we can see the cost reaches at about 0.19. When the frequency

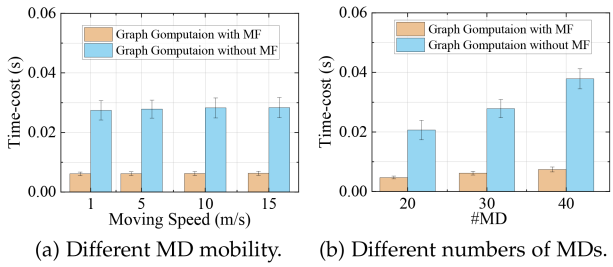


Fig. 14. Time-cost evaluation of the graph computation in SITOff, with different speeds and numbers of MDs.

gap between f^{Thick} and f^{Thin} reduces to 0.6 Ghz, the offloading cost sees obvious increment to about 0.25. This suggests that SITOff can make better use of exchanged knowledge to offload intensive tasks of thin MDs to thick MDs when their frequency gap is large, and the performance improvements resulting from D2D offloading could be small when the gap is small. During execution in Fig. 13(b), the performance superiority w.r.t. the cost, TSR, and ESR of large $f^{\text{Thick}}/f^{\text{Thin}}$ over small one is more obvious.

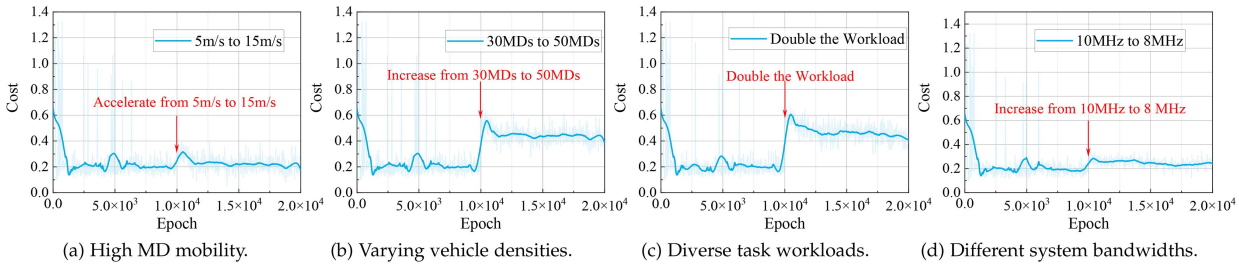
E. Time-Cost Evaluation of Graph Computation in SITOff

As shown in Fig 14, we investigate the time cost of the MF-based graph computation compared to the vanilla graph computation in SITOff. We record the average graph computation time during one offloading decision-making with different mobilities and numbers of MDs. It can be seen that the graph computation time is significantly reduced and the decision-making efficiency of SITOff is improved by the MF-based method. As shown in Fig. 14(a), different MD movement speeds do not significantly affect the graph computation time, and the average time cost of MF-based graph computation reduces to around 0.00615 s with different MD moving speeds, at least 78.15% decline compared to the vanilla graph computation method. In Fig. 14(b), the graph computation time is affected by the number of different MDs. As the number of MDs increases, the MF-based method exhibits a more obvious advantage in reducing graph computation time. When the number of MDs increases from 20 to 40, the time taken for MF-based graph computation only grows from 0.006154 s to 0.00741 s, while the time cost in the vanilla method increases from 0.0206 s to 0.0378 s. The main reason for the significant reduction in graph computation time could resulted from (33) that simplifies the computation by calculating the weighted average of the neighbouring MDs' state features while guaranteeing the performance, which avoids the complex high-dimensional matrices operation of the vanilla method in (30).

F. Robustness of SITOff Under Different Network Conditions

We evaluate the robustness of SITOff under different network conditions, including high MD mobility, varying vehicle densities, diverse task workloads and different system bandwidths, as shown in Fig. 15.

1) Robustness of SITOff Under High MD Mobility: We accelerate the speed of MDs from 5 m/s to 15 m/s at the epoch 1×10^4, as shown in Fig 15(a). It can be seen that the cost fluctuates slightly from 0.2 to 0.3, due to the reason that the increase of



(a) High MD mobility. (b) Varying vehicle densities. (c) Diverse task workloads. (d) Different system bandwidths.

Fig. 15. Performance evaluation on the robustness of SITOff under different network conditions.

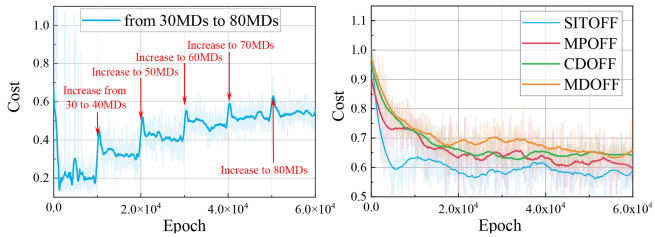
MDs' speeds in the Whole-Area leads to the increased variance of the number of MDs within the Service-Area. But after about 1K epochs, the cost is observed to recover back to 0.21, a stable and reasonable performance for 15 m/s speed of all MDs. It indicates that SITOff is not largely susceptible to the changes of MDs' speeds and shows a strong robustness. The main reason is that the graph representation can better cope with the rapid change of MDs in the Service-Area, and transform the rapidly changing graph topology into suitable MDs' state features to facilitate the learning convergence of the offloading policy.

2) Robustness of SITOff Under Varying Vehicle Densities: In Fig. 15(b), we investigate SITOff in D-MEC with different vehicle densities, where the number of MDs increases from 30 to 50 at the epoch 1×10^4. The results show that the cost first fluctuates from 0.2 to 0.56, but later returns back to 0.42 after less than 1K epochs. The cost keeps stable around 0.42 after the epch 1.1×10^4, which is a reasonable performance[7] for the D-MEC with 50 MDs. The reason could be that SITOff is able to aggregate the features of neighbouring MDs through graph representation thereby performing size-insensitive distributed task policy learning. It demonstrates the robustness of SITOff under varying vehicle densities.

3) Robustness of SITOff Under Diverse Task Workloads: As shown in Fig. 15(c), we evaluate the robustness of SITOff under two different degrees of task workloads, which grow from uniform distribution [50, 100] kb to [100, 200] kb at the epoch 1×10^4. The results show that despite the significant increment of task workloads, SITOff is able to return to a stable performance level around 0.41 after a short fluctuation from 0.2 to 0.6. This is mainly because when the state of the task workloads is changed, each MD can quickly converge to a stable cost by simply updating its own neural network model for its individual task via the fully decentralized training framework. Similarly, since the workload doubles while the ES computing capacity and system bandwidth remain unchanged, it is reasonable for SITOff to converge to 0.41.

4) Robustness of SITOff Under Different System Bandwidths: We investigate the robustness of SITOff when changing the system bandwidth from 10 MHz to 8 MHz at the epoch 1×10^4, as shown in Fig. 15(d). SITOff is found to encounter a slight cost increase from 0.2 to 0.29, and soon keep stable around 0.24 after 1.05×10^4. The slight cost increase could be due to that the change of system bandwidths leads to the variation of system dynamics, bringing in troubles for the offloading policy. But, SITOff is able to update the model parameters according to Algorithm 1 (lines 17-23) at each time slot, that soon adapt its offloading decision to the changed system dynamics. This suggests that SITOff is robust to the changes of system bandwidths, and can utilize limited bandwidths to achieve reasonable costs.

[7]The task workloads increase with the number of MDs, but the ES computing resources and system bandwidths do not increase. Hence, the time cost of tasks also increases and is reasonable around 0.42.



(a) Curriculum learning in 30-80 MDs. (b) Direct learning in 80 MDs.

Fig. 16. Collaboration between SITOff and curriculum learning.

G. Collaboration Between SITOff and Curriculum Learning

Due to the insensitivity of SITOff on the D-MEC size during whether training or running, we adopt curriculum learning to aid SITOff for large-scale D-MEC by gradually increasing the number of MDs from 30 to 80, as shown in Fig. 16(a) (. When SITOff is trained gradually from 30 to 80 MDs, during which the output ω, ϕ, ψ, μ of previous training is taken as the input of the following training of Algorithm 1, we can see that the cost could converge to about 0.52 at the end of curriculum learning, demonstrating the efficient collaboration of SITOff with curriculum learning for large-scale D-MEC. Nonetheless, when both SITOff and MARL-based baselines are training directly in D-MEC with 80 MDs, with the same number of epochs 6×10^4, it is found to be hard for them to converge to satisfactory costs until the end of training, as shown in Fig. 16(b).

VII. Conclusion

This paper addresses the size-sensitive problem of most DRL-based offloading algorithms in D2D-assisted MEC, and proposes a size-insensitive task offloading algorithm, named SITOff, that is fully based on distributed offloading without the need of maintaining any central venue. SITOff adopts graph to represent local knowledge for each MD, designs a D2D transmission-aided knowledge-exchange mechanism to enhance

the coordination of MDs' distributed offloading, and leverages attention mechanism for selective knowledge-exchange to further improve the whole offloading performance. We conduct extensive experiments to demonstrate the superiority of SITOff against state-of-the-art DRL-based algorithms, as well as the possibility for SITOff to collaborate with curriculum learning to facilitate large-scale MEC offloading.

Beyond of this work, future studies could include: size-insensitive task offloading in more complex MEC scenarios (multi-BS MEC, UAV-assisted MEC, etc.), size-insensitive task offloading on account of more decision variables (computing resources allocation, system bandwidth allocation, etc.), and performance evaluation of SITOff in real-world testbed.

REFERENCES

[1] M. Tang and V. W. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 1985–1997, Jun. 2020.

[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Fourth Quarter 2017.

[3] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, Third Quarter 2017.

[4] H. Ko and S. Pack, "Distributed device-to-device offloading system: Design and performance optimization," *IEEE Trans. Mobile Comput.*, vol. 20, no. 10, pp. 2949–2960, Oct. 2021.

[5] W. Jiang, D. Feng, Y. Sun, G. Feng, Z. Wang, and X.-G. Xia, "Joint computation offloading and resource allocation for D2D-assisted mobile edge computing," *IEEE Trans. Services Comput.*, vol. 16, no. 3, pp. 1949–1963, May/Jun. 2023.

[6] F. H. Cabrini et al., "Enabling the industrial internet of things to cloud continuum in a real city environment," *Sensors*, vol. 21, no. 22, 2021, Art. no. 7707.

[7] C. T. Garrocho, M. J. da Silva, and R. A. Oliveira, "D2D pervasive communication system with out-of-band control autonomous to 5G networks: Project and evaluation of a middleware for networking and content exchange to D2D communication without human interaction," *Wireless Netw.*, vol. 26, no. 1, pp. 373–386, 2020.

[8] M. S. M. Gismalla et al., "Survey on device to device (D2D) communication for 5GB/6G networks: Concept, applications, challenges, and future directions," *IEEE Access*, vol. 10, pp. 30792–30821, 2022.

[9] J. Yu, Y. Li, X. Liu, B. Sun, Y. Wu, and D. H.-K. Tsang, "IRS assisted noma aided mobile edge computing with queue stability: Heterogeneous multi-agent reinforcement learning," *IEEE Trans. Wireless Commun.*, vol. 22, no. 7, pp. 4296–4312, Jul. 2023.

[10] J. Peng, H. Qiu, J. Cai, W. Xu, and J. Wang, "D2D-assisted multi-user cooperative partial offloading, transmission scheduling and computation allocating for MEC," *IEEE Trans. Wireless Commun.*, vol. 20, no. 8, pp. 4858–4873, Aug. 2021.

[11] Y. He, J. Ren, G. Yu, and Y. Cai, "D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 3, pp. 1750–1763, Mar. 2019.

[12] U. Saleem, Y. Liu, S. Jangsher, X. Tao, and Y. Li, "Latency minimization for D2D-enabled partial computation offloading in mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4472–4486, Apr. 2020.

[13] O. Karatalay, I. Psaromiligkos, and B. Champagne, "Energy-efficient resource allocation for D2D-assisted fog computing," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 4, pp. 1990–2002, Dec. 2022.

[14] B. Kar, W. Yahya, Y.-D. Lin, and A. Ali, "Offloading using traditional optimization and machine learning in federated cloud–edge–fog systems: A survey," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 2, pp. 1199–1226, Second Quarter 2023.

[15] H. Xiao, C. Xu, Y. Ma, S. Yang, L. Zhong, and G.-M. Muntean, "Edge intelligence: A computational task offloading scheme for dependent IoT application," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 7222–7237, Sep. 2022.

[16] X. Dai et al., "Task co-offloading for D2D-assisted mobile edge computing in industrial internet of things," *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 480–490, Jan. 2023.

[17] J. Tan, Y.-C. Liang, L. Zhang, and G. Feng, "Deep reinforcement learning for joint channel selection and power control in D2D networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1363–1378, Feb. 2021.

[18] Z. Cao, P. Zhou, R. Li, S. Huang, and D. Wu, "Multiagent deep reinforcement learning for joint multichannel access and task offloading of mobile-edge computing in industry 4.0," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6201–6213, Jul. 2020.

[19] Z. Cheng, M. Min, Z. Gao, and L. Huang, "Joint task offloading and resource allocation for mobile edge computing in ultra-dense network," in *Proc. IEEE Glob. Commun. Conf.*, 2020, pp. 1–6.

[20] X. Zhu, Y. Luo, A. Liu, M. Z. A. Bhuiyan, and S. Zhang, "Multiagent deep reinforcement learning for vehicular computation offloading in IoT," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9763–9773, Jun. 2021.

[21] Z. Ye, K. Wang, Y. Chen, X. Jiang, and G. Song, "Multi-UAV navigation for partially observable communication coverage by graph reinforcement learning," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 4056–4069, Jul. 2023.

[22] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1226–1252, Second Quarter 2021.

[23] T. Li et al., "Applications of multi-agent reinforcement learning in future internet: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 2, pp. 1240–1279, Second Quarter 2022.

[24] Z. Li and C. Guo, "Multi-agent deep reinforcement learning based spectrum allocation for D2D underlay communications," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1828–1840, Feb. 2020.

[25] D. Shi, L. Li, T. Ohtsuki, M. Pan, Z. Han, and H. V. Poor, "Make smart decisions faster: Deciding D2D resource allocation via stackelberg game guided multi-agent deep reinforcement learning," *IEEE Trans. Mobile Comput.*, vol. 21, no. 12, pp. 4426–4438, Dec. 2022.

[26] N. Zhao, Z. Ye, Y. Pei, Y.-C. Liang, and D. Niyato, "Multi-agent deep reinforcement learning for task offloading in UAV-assisted mobile edge computing," *IEEE Trans. Wireless Commun.*, 2022.

[27] Y. Ju et al., "Joint secure offloading and resource allocation for vehicular edge computing network: A multi-agent deep reinforcement learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 5, pp. 5555–5569, May 2023.

[28] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[29] X. Wang, Y. Chen, and W. Zhu, "A survey on curriculum learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 4555–4576, Sep. 2022.

[30] X. Lei, Q. Li, P. Bo, Y. Z. Zhou, C. Chen, and S. L. Peng, "Long short-term deterministic policy gradient for joint optimization of computational offloading and resource allocation in MEC," in *Proc. Int. Conf. Algorithms Architectures Parallel Process.*, Springer, 2023, pp. 329–348.

[31] D. B. Son, T. H. Binh, H. K. Vo, B. M. Nguyen, H. T. T. Binh, and S. Yu, "Value-based reinforcement learning approaches for task offloading in delay constrained vehicular edge computing," *Eng. Appl. Artif. Intell.*, vol. 113, 2022, Art. no. 104898.

[32] A. Sadiki, J. Bentahar, R. Dssouli, A. En-Nouaary, and H. Otrok, "Deep reinforcement learning for the computation offloading in mimo-based edge computing," *Ad Hoc Netw.*, vol. 141, 2023, Art. no. 103080.

[33] J. Tian, Q. Liu, H. Zhang, and D. Wu, "Multiagent deep-reinforcement-learning-based resource allocation for heterogeneous QoS guarantees for vehicular networks," *IEEE Internet Things J.*, vol. 9, no. 3, pp. 1683–1695, Feb. 2022.

[34] S. Wang, T. Lv, W. Ni, N. C. Beaulieu, and Y. J. Guo, "Joint resource management for MC-NOMA: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 20, no. 9, pp. 5672–5688, Sep. 2021.

[35] Y. Xiao, Y. Song, and J. Liu, "Multi-agent deep reinforcement learning based resource allocation for ultra-reliable low-latency internet of controllable things," *IEEE Trans. Wireless Commun.*, vol. 22, no. 8, pp. 5414–5430, Aug. 2023.

[36] T. Chen, X. Zhang, M. You, G. Zheng, and S. Lambotharan, "A gnn-based supervised learning framework for resource allocation in wireless IoT networks," *IEEE Internet Things J.*, vol. 9, no. 3, pp. 1712–1724, Feb. 2022.

[37] Z. He, L. Wang, H. Ye, G. Y. Li, and B.-H. F. Juang, "Resource allocation based on graph neural networks in vehicular communications," in *Proc. IEEE Glob. Commun. Conf.*, 2020, pp. 1–5.

[38] Z. Sun, Y. Mo, and C. Yu, "Graph-reinforcement-learning-based task offloading for multiaccess edge computing," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3138–3150, Feb. 2023.
[39] Z. Gao, L. Yang, and Y. Dai, "Fast adaptive task offloading and resource allocation in large-scale MEC systems via multiagent graph reinforcement learning," *IEEE Internet Things J.*, vol. 11, no. 1, pp. 758–776, Jan. 2024.
[40] M. Sheng, Y. Wang, X. Wang, and J. Li, "Energy-efficient multiuser partial computation offloading with collaboration of terminals, radio access network, and edge server," *IEEE Trans. Commun.*, vol. 68, no. 3, pp. 1524–1537, Mar. 2020.
[41] Z. Ning et al., "Partial computation offloading and adaptive task scheduling for 5G-enabled vehicular networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 4, pp. 1319–1333, Apr. 2022.
[42] Z. Xiao et al., "Multi-objective parallel task offloading and content caching in D2D-aided MEC networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 11, pp. 6599–6615, Nov. 2023.
[43] T. Zhu, J. Li, Z. Cai, Y. Li, and H. Gao, "Computation scheduling for wireless powered mobile edge computing networks," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 596–605.
[44] C. Xu, G. Zheng, and X. Zhao, "Energy-minimization task offloading and resource allocation for mobile edge computing in noma heterogeneous networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16001–16016, Dec. 2020.
[45] F. Boukouvala, R. Misener, and C. A. Floudas, "Global optimization advances in mixed-integer nonlinear programming, minlp, and constrained derivative-free optimization, cdfo," *Eur. J. Oper. Res.*, vol. 252, no. 3, pp. 701–727, 2016.
[46] G. Cornuéjols, R. Sridharan, and J.-M. Thizy, "A comparison of heuristics and relaxations for the capacitated plant location problem," *Eur. J. Oper. Res.*, vol. 50, no. 3, pp. 280–297, 1991.
[47] O. Hernández-Lerma and J. B. Lasserre, *Discrete-Time Markov Control Processes: Basic Optimality Criteria*, vol. 30. Berlin, Germany: Springer Science & Business Media, 2012.
[48] N. Zhao, Z. Ye, Y. Pei, Y.-C. Liang, and D. Niyato, "Multi-agent deep reinforcement learning for task offloading in UAV-assisted mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 6949–6960, Sep. 2022.
[49] L. Cotta, C. Morris, and B. Ribeiro, "Reconstruction for powerful graph representations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 1713–1726.
[50] Q. Hao, W. Huang, T. Feng, J. Yuan, and Y. Li, "GAT-MF: Graph attention mean field for very large scale multi-agent reinforcement learning," in *Proc. 29th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2023, pp. 685–697.
[51] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 5571–5580.
[52] C. Qu, S. Mannor, H. Xu, Y. Qi, L. Song, and J. Xiong, "Value propagation for decentralized networked deep multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 1184–1193.
[53] H.-T. Wai, M. Hong, Z. Yang, Z. Wang, and K. Tang, "Variance reduced policy evaluation with smooth function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 5776–5787.
[54] X. Zhang, Z. Liu, J. Liu, Z. Zhu, and S. Lu, "Taming communication and sample complexities in decentralized policy evaluation for cooperative multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 18825–18838.
[55] H. Sun, S. Lu, and M. Hong, "Improving the sample and communication complexity for decentralized non-convex optimization: Joint gradient estimation and tracking," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2020, pp. 9217–9228.
[56] F. Bai and A. Helmy, "A survey of mobility models," *Wireless Adhoc Netw. Univ. Southern California, USA*, vol. 206, 2004, Art. no. 147.
[57] M. Ulm, P. Widhalm, and N. Brändle, "Characterization of mobile phone localization errors with opencellid data," in *Proc. 4th Int. Conf. Adv. Logistics Transport*, 2015, pp. 100–104.
[58] P. Dai, K. Hu, X. Wu, H. Xing, and Z. Yu, "Asynchronous deep reinforcement learning for data-driven task offloading in MEC-empowered vehicular networks," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.
[59] G. Luo et al., "Software-defined cooperative data sharing in edge computing assisted 5G-vanet," *IEEE Trans. Mobile Comput.*, vol. 20, no. 3, pp. 1212–1229, Mar. 2021.
[60] X. Mi and H. He, "Multi-agent deep reinforcement learning for D2D-assisted MEC system with energy harvesting," in *Proc. 25th Int. Conf. Adv. Commun. Technol.*, 2023, pp. 145–153.

Zheyuan Hu (Graduate Student Member, IEEE) received the BS degree in computer science and engineering from Northeastern University, Shenyang, China, in 2017, and the MS degree in computer science and engineering from Beihang University, Beijing, China, in 2021. He is currently workin toward the PhD degree with the School of Computer Science and Engineering, Beihang University, Beijing, China. His research interests include mobile edge computing and distributed computing system.

Jianwei Niu (Senior Member, IEEE) received the MS and PhD degrees in computer science from Beihang University, Beijing, China, in 1998 and 2002, respectively. He was a visiting scholar with the School of Computer Science, Carnegie Mellon University, USA, from 2010 to 2011. He is currently a professor with the School of Computer Science and Engineering, BUAA. His current research interests include mobile and pervasive computing, and distributed computing.

Tao Ren (Member, IEEE) received the BS, MS, and PhD degrees in computer science from the Information Engineering University, Zhengzhou, China, in 2004, 2007, and 2011, respectively. He is currently an associate research fellow with the Institute of Software Chinese Academy of Sciences. His research interests include mobile edge computing, intelligent game.

Xuefeng Liu (Member, IEEE) received the MS degree from the Beijing Institute of Technology, China, in 2003, and the PhD degree from the University of Bristol, U.K., in 2008. He is currently an associate professor with Beihang University. His research interests include intelligent signal processing and deep learning. He has served as a reviewer for several international journals/conference proceedings.

Mohsen Guizani (Fellow, IEEE) received the BS (with distinction), MS, and PhD degrees in electrical and computer engineering from Syracuse University, Syracuse, NY, USA, in 1985, 1987, and 1990, respectively. He is currently a professor of machine learning with the Mohamed Bin Zayed University of Artificial Intelligence (MBZUAI), Abu Dhabi, UAE. Previously, he worked in different institutions in the USA. His research interests include applied machine learning and artificial intelligence, smart city, Internet of Things (IoT), intelligent autonomous systems, and cybersecurity. He became an IEEE fellow in 2009 and 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 Communications Society Best Survey Paper Award", the Best ComSoc Journal Paper Award in 2021 as well 5 Best Paper Awards from ICC and Globecom Conferences. He is the author of 11 books, more than 1000 publications and several US patents. He is also the recipient of the 2017 IEEE Communications Society Wireless Technical Committee (WTC) Recognition Award, the 2018 AdHoc Technical Committee Recognition Award, and the 2019 IEEE Communications and Information Security Technical Recognition (CISTC) Award. He served as the editor-in-chief of IEEE Network and is currently serving on the editorial boards of many IEEE Transactions and Magazines. He was the chair of the IEEE Communications Society Wireless Technical Committee and the chair of the TAOS Technical Committee. He served as the IEEE Computer Society distinguished speaker and is currently the IEEE ComSoc distinguished lecturer.