Inverse Reinforcement Learning With Graph Neural Networks for Full-Dimensional Task Offloading in Edge Computing

Guangchen Wang, *Member, IEEE*, Peng Cheng, *Member, IEEE*, Zhuo Chen, *Senior Member, IEEE*, Branka Vucetic, *Fellow, IEEE*, and Yonghui Li, *Fellow, IEEE*

***Abstract*—The ever-increasing number of ubiquitous Internet of Things (IoT) applications entails a high demand for scarce communication and network resources. To meet this stringent requirement, mobile edge computing (MEC) is envisioned as a transformative technique to significantly streamline the existing network operations. Recently, device-to-device (D2D) communication has been proposed as a promising technology in 5G and beyond networks with a significantly increased transmission efficiency, especially suitable for small-packet task exchanges. In this paper, we incorporate D2D communication into the multi-layer computing network and propose a full-dimensional task offloading scheme by jointly optimizing task offloading decisions and computation/communication resource allocation. We formulate it as mixed-integer nonlinear programming (MINLP) problem, where the optimal branch-and-bound (B&B) algorithm with the full strong branching (FSB) variable selection policy features an extremely high complexity. To address this challenge, we propose inverse reinforcement learning with graph neural networks (GIRL) to generate a new variable selection policy that closely matches the FSB variable selection. Without sacrificing the global optimality, the GIRL can directly infer the variable selection with a much lower complexity, significantly accelerating the original B&B algorithm. Simulation results show that the GIRL achieves a lower complexity without sacrificing the global optimality. Furthermore, our proposed full-dimensional task offloading scheme achieves better performance than the existing schemes in terms of average delay for all mobile devices (MDs).**

***Index Terms*—Full-dimensional task offloading optimization, inverse reinforcement learning with graph neural networks (GIRL), Internet of Things (IoT), mobile edge computing (MEC).**

Manuscript received 9 December 2022; revised 15 September 2023; accepted 6 October 2023. Date of publication 13 October 2023; date of current version 7 May 2024. The work of Peng Cheng was supported by Australian Research Council (ARC) under Grants DP210103410 and DP220101634. The work of Yonghui Li was supported by ARC under Grant DP210103410. An earlier version of this work was presented at Inverse reinforcement learning with graph neural networks for IoT resource allocation [doi: 10.1109/ICASSP49357.2023.10096237]. Recommended for acceptance by W. Gao. *(Corresponding authors: Peng Cheng; Yonghui Li.)*

Guangchen Wang, Branka Vucetic, and Yonghui Li are with the School of Electrical and Information Engineering, the University of Sydney, Camperdown, NSW 2050, Australia (e-mail: guangchen.wang@sydney.edu.au; branka.vucetic@sydney.edu.au; yonghui.li@sydney.edu.au).

Peng Cheng is with the Department of Computer Science and Information Technology, La Trobe University, Bundoora, VIC 3086, Australia, and also with the University of Sydney, Camperdown, NSW 2050, Australia (e-mail: p.cheng@latrobe.edu.au).

Zhuo Chen is with CSIRO DATA61, Eveleigh, NSW 2015, Australia (e-mail: drbearsyd@hotmail.com).

Digital Object Identifier 10.1109/TMC.2023.3324332

I. Introduction

A. Multi-Layer Mobile Computing Network

THE ubiquitous Internet of Things (IoT) is envisioned to connect billions of mobile sensors, manufacturing machines, smart devices, industrial utilities, etc., entailing a high demand for scarce communication and network resources [2]. Furthermore, many emerging IoT-enabled computation-intensive applications, such as face recognition, participatory sensing, and autonomous driving [3], [4], generate an unprecedented volume of data for pattern recognition. However, mobile devices (MDs) are usually limited by their computation resources and battery life. Featured by abundant computation and storage resources, mobile cloud computing (MCC) [5] facilitates the handling of computing-intensive tasks via task offloading from MDs. However, due to the remote location of the cloud server (CS), the task offloading will inevitably introduce high transmission latency [6].

As a transformative technique, mobile edge computing (MEC) [7] is proposed to enable delay-sensitive applications. Typically, MEC servers (MSs), are deployed much closer to MDs for task offloading, which significantly reduces transmission delay and energy consumption. This helps decrease traffic congestion and improve the quality of service (QoS) of the whole network. Many task offloading schemes have been proposed in the literature. Earlier work focuses on vertical task offloading, where tasks from an MD can only be offloaded to its associated MS or CS. The authors in [8] consider a multi-user multi-layer mobile computing network (MD-MS-CS), and aim to minimize the energy cost, computation, and delay for all MDs. In [9], the authors strive to minimize the energy consumption for delay-constrained applications, where the task offloading and resource allocation are jointly optimized in a hierarchical network architecture. The authors in [10] propose a greedy optimization approach to minimize communication costs in an MEC network where low computing but high communication capabilities are needed. Recently, some work [11], [12] focused on a two-dimensional task offloading scheme where horizontal cooperation between MSs is considered in a multi-layer network. In [11], the authors propose an alternating direction method of multipliers (ADMM) method to minimize the average task delay in a multi-layer MEC network with vertical and horizontal edge computing cooperation. The authors in [12] propose a

Gaussian process imitation learning (GPIL) method to minimize the average task delay subject to communication/computation resources and energy consumption constraints.

The rapid emergence of new IoT applications such as augmented reality (AR) and mobile healthcare [13] introduce a large number of tasks with a relatively small packet size. In this case, MDs need to frequently offload their tasks to MSs, resulting in increased latency and degraded energy efficiency due to significant wireless communication overhead between MDs and MSs. Recently, device-to-device (D2D) communication has been proposed as a promising technology in 5G and beyond networks [14], where the core idea is to enable direct transmission between proximate devices, without relaying information through the base station. Therefore, D2D can enable low-latency communications with a significantly increased transmission efficiency, especially suitable for small-packet task exchanges in massive machine-type communications [15].

In this paper, we incorporate D2D communication into the multi-layer computing network and propose a new full-dimensional task offloading scheme by jointly optimizing task offloading decisions and communication/computation resources. In the presence of both D2D cooperation and horizontal edge cooperation, a task can be either executed locally or offloaded to proximate MDs, collaborative MSs, or CS. In this case, this scheme can be adaptive to various task sizes, harvesting the benefits of both edge/cloud servers for computation-intensive tasks and D2D cooperation for small-packet ones. Mathematically, we formulate the full-dimensional task offloading into a mixed integer nonlinear programming (MINLP) problem, which involves mutually coupled discrete (task offloading decisions) and continuous variables (communication and computation resource allocation). However, an MINLP problem is usually NP-hard and no optimal solution can be found by a polynomial-time algorithm. This calls for efficient strategies that scale favorably with the MINLP problem size.

B. Solutions to MINLP Problems

Several contemporary methods are available to potentially solve the MINLP problems. The branch-and-bound (B&B) algorithm [16] is a global exhaustive search method that achieves a global optimal solution and provides the performance upper bound, but its computational complexity is prohibitively high with a large number of discrete variables. Consequently, some heuristic optimization algorithms with a low complexity were proposed in the literature. The typical ones in [17] include genetic algorithm (GA), particle swarm optimization algorithm (PSO), and relaxation-based algorithm. These algorithms are fast in solving MINLP problems. However, they are sub-optimal due to their heuristic nature, and the performance gap with the optimal B&B algorithm is difficult to quantify.

Reinforcement learning (RL) [18], which involves an agent making observations and taking actions within an environment to receive rewards, offers a potential avenue to the MINLP problem. However, the MINLP problem combines discrete and continuous variables, which makes it difficult for RL to directly handle a large mixed action space. Furthermore, the MINLP problem involves a number of constraints, and RL cannot guarantee the strict satisfaction of these constraints.

As the B&B algorithm can offer a global optimal solution to the MINLP problem but at the cost of usually unacceptable complexity, in this paper, we develop a new approach to significantly accelerate the B&B algorithm. Basically, the B&B has three fundamental policies, namely, node pruning, node selection, and variable selection policies [16]. Node selection and pruning policies determine which node is pruned or preserved in the enumeration tree. Consequently, the major efforts in accelerating the B&B algorithm are placed on improving the efficiency of these two policies by various heuristic algorithms. On the other hand, it is worth noting that the variable selection policy determines the order of the variables for branching, also having a significant impact on the size of the enumeration tree. The full strong branching policy (FSB) [19], as the optimal variable selection policy of the original B&B algorithm, can achieve a minimum size of an enumeration tree. However, the associated variable selection process is prohibitively tedious, especially for a large number of variables. This hinders its implementation for our multi-layer full-dimensional task offloading scheme. Under the umbrella of the B&B algorithm, we leverage the learning mechanism to generate a new variable selection policy referred to as inverse reinforcement learning with graph neural networks (GIRL), aiming to achieve a comparable performance but with a significantly lower complexity relative to the FSB.

C. Contribution

Our main contributions can be summarized as follows.

- We consider a multi-layer cooperative computing network by leveraging both D2D cooperation and horizontal edge cooperation. In this framework, a task can be either executed locally or offloaded to proximate MDs, collaborative MSs, or CS. We propose a full-dimensional task offloading scheme by jointly optimizing the task offloading and communication/computation resources and formulate it as an MINLP problem.
- We resort to imitation learning and propose GIRL variable selection policy. This approach solves the challenge that the reward function cannot be appropriately designed as the branching order of variables cannot be known in advance. Without sacrificing the global optimality, our approach could achieve much lower complexity than the FSB.
- We develop a graph neural network (GNN) as a parameterized reward function in the GIRL, which directly handles the graphic features in the enumeration tree of the B&B algorithm. We further design an attention mechanism in the GNN and incorporate self-imitation into the GIRL. This results in a strong model generalization capability, empowering the GIRL to easily adapt to the variations of the key network parameters without learning from scratch.
- It is verified by simulation that, the proposed GIRL can significantly accelerate the original B&B algorithm. It

achieves a much lower computational complexity compared to the existing variable selection policies. Furthermore, our proposed full-dimensional task offloading scheme achieves better performance than the existing schemes in terms of average delay for all MDs.

D. Related Work

In addition to the works introduced in Section I-A, other existing related works on computing networks have been done to optimize the latency, the computational cost, and the energy consumption in the mobile computing network. In [20], the authors aim to minimize the average response cost in a D2D-assisted vertical MEC system by jointly optimizing the offloading, transmission scheduling, and computation allocation. The authors in [21] strive to maximize the aggregate offloading benefits (trade-off between delay and energy consumption) in vertical MEC systems with the D2D pairing. The work in [22] jointly optimizes to minimize the computation latency in a three-layer vertical MEC network with D2D cooperation. However, the above task offloading schemes put emphasis on the cooperation between D2D communication and edge computing and ignore the potential computation capacity of collaborative MSs.

In addition to the works introduced in Section I-B, many related works are focused on optimization-based methods to solve optimization problems in multi-layer computing networks. In [23], authors jointly optimize task partitioning and user association in a multi-layer computing network. The formulated problem is divided into two subproblems, where the first one is solved directly under a given user association. The second one is solved by a dual decomposition-based method. In [24], a long-term problem is formulated in a two-layer computing network. It is solved by a regularization and rounding-based method including two sequential phases, where the long-term problem is decomposed series of one-shot fractional problems in the first phase, and the fractional solution is rounded to an integer solution by a dependent rounding scheme in the second phase. In [25], an enders-decomposition-based method is proposed to solve the resource allocation problem, and a significant number of iterations are involved rendering the method of significant complexity. However, these three optimization-based decomposition methods are developed for specific formulated problems in a multi-layer computing network, and cannot be directly applied to solve our MINLP problem. Besides, these methods involve a large number of iterations and suffer significant complexity.

Besides, many related works are focused on ML-based methods in multi-layer computing networks. The authors in [26] propose a deep neural network (DNN) based method to minimize the energy consumption of all the user equipment in the edge computing network. In [27], authors propose a deep reinforcement learning (DRL) based online offloading algorithm to maximize the sum computation rate in a two-layer computing network. The authors in [28] utilize a DRL method to optimize resource scheduling aiming to minimize the task latency of the tasks. However, the above methods decompose the original MINLP problem into a sub-problem with discrete variables addressed by the ML-based method, and the other sub-problem with continuous variables addressed by classical optimization algorithms. The optimality might not be guaranteed due to the decomposition (i.e. not joint optimization).

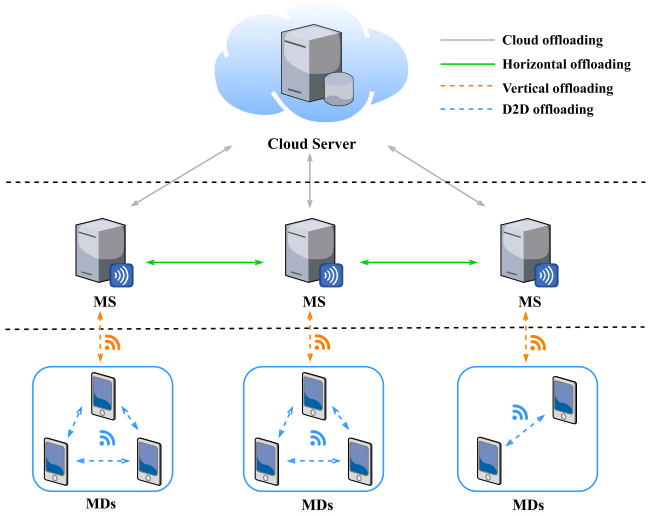


Fig. 1. Multi-layer mobile computing network with full-dimensional task offloading scheme.

E. Organization

The rest of this paper is organized as follows. We first provide system model and formulate an MINLP problem in Section II. Then, we introduce the B&B algorithm in Section III. In Section IV, we elaborate on the design of the proposed GIRL variable selection policy for accelerating the B&B algorithm. Furthermore, we develop an attention GNN as the parameterized reward function in Section V. On this basis, in Section VI, we propose the self-imitation to enhance the model generalization capability, which is followed by performance optimization and complexity analysis. We provide the simulation results in Section VII. Finally, we give the conclusions in Section VIII.

II. System Model and Problem Formulation

A. System Model

As shown in Fig. 1, we consider a multi-layer mobile computing network including N MDs indexed by the set $\mathcal{N} = \{1, 2, \ldots, N\}$, M MSs indexed by the set $\mathcal{M} = \{1, 2, \ldots, M\}$ and a CS. The D2D communication between MDs and the communication between MD and its associated MS are through a wireless network. The inter-connected communication between MSs is through a wired network, and each MS is connected to the CS via a backhaul wired network.

In our three-level full-dimensional cooperation architecture, MD $i \in \mathcal{N}$ has a computation task $Q_i = \{D_i, C_i\}$, where D_i is the data size and C_i is the required computation resources (in CPU cycles). Specifically, Q_i can be processed locally (Level 0 local computing) or by an adjacent peer MD $j \in \hat{\mathcal{N}}_i = \{\mathcal{N} \setminus i\}$ (Level 0 D2D cooperation computing) when MD j has extra computing resources. Alternatively, the task Q_i can be offloaded to the associated MS $l_i \in \mathcal{M}$. From here, the task can be either processed locally (Level 1 vertical edge computing), or

forwarded to the other free MSs $k \in \hat{\mathcal{M}}_i = \{\mathcal{M} \setminus l_i\}$ (Level 1 horizontal edge computing). In addition, the task could be offloaded to the CS for processing (Level 2 cloud computing). In summary, the task offloading binary decision could be classified as the following five indicators

- Level 0 local computing indicator $x_{i,i} \in \{0,1\}$: task Q_i is processed by the MD i when $x_{i,i} = 1$.
- Level 0 D2D cooperation computing indicator $x_{i,j} \in \{0,1\}$: task Q_i is processed by the peer MD j when $x_{i,j} = 1$.
- Level 1 vertical edge computing indicator $y_{i,l_i} \in \{0,1\}$: task Q_i is vertically offloaded to the associate MS l_i for processing when $y_{i,l_i} = 1$.
- Level 1 horizontal edge computing indicator $y_{i,k} \in \{0,1\}$: task Q_i is forwarded to the MS k by MS l_i horizontally for processing when $y_{i,k} = 1$.
- Level 2 cloud computing indicator $z_i \in \{0,1\}$: task Q_i is offloaded to the CS for processing when $z_i = 1$.

Overall, the offloading decision for the task Q_i could be constrained by

$$x_{i,i} + \sum_{j \in \hat{\mathcal{N}}_i} x_{i,j} + \sum_{k \in \mathcal{M}} y_{i,k} + z_i = 1. \tag{1}$$

B. Communication Model

In this paper, the communication model consists of the wireless link and the wired link. The communication of level 0 D2D cooperation computing and level 1 vertical edge computing is through the wireless link. For level 0 D2D cooperation, the data rate between MD i and the peer MD j could be calculated as

$$r_{i,j} = b_i \log\left(1 + \frac{p_i g_{i,j}}{\sum_{i' \in \hat{\mathcal{N}}_i} p_{i'} g_{i',j} + \sigma_2}\right), \tag{2}$$

where b_i is the allocated bandwidth of MD i, p_i is the transmission power, $g_{i,j}$ is the channel gain between MD i and the peer MD j, and σ^2 is the power of the additive white Gaussian noise. Here, $\sum_{i' \in \hat{\mathcal{N}}_i} p_{i'} g_{i',j}$ represents the interference caused by D2D transmission.

Similarly, for the level 1 vertical transmission, the uplink data rate between the MD i and associated MS l_i could be calculated as

$$r_{i,l_i} = b_i \log\left(1 + \frac{p_i g_{i,l_i}}{\sum_{i' \in \mathcal{U}_{l_i}, i' \neq i} p_{i'} g_{i',l_i} + \sigma_2}\right), \tag{3}$$

where $\mathcal{U}_{l_i}$ is the set of MDs associated to MS l_i, and $\sum_{i' \in \mathcal{U}_{l_i}, i' \neq i} p_{i'} g_{i',l_i}$ represents the interference from MDs that offload resources to MS l_i. The total bandwidth for MS k should not exceed its maximum bandwidth capacity $B_k^{\max}$, so we have the constraint

$$\sum_{i \in \mathcal{U}_k} b_i \leq B_k^{\max}, \tag{4}$$

where $\mathcal{U}_k$ is the set of MDs associated to MS k. With the data rate, the transmission delay could be calculated as

$$d_{i,(\cdot)}^{\text{tr}} = \frac{D_i}{r_{i,(\cdot)}}, \ (\cdot) \in \{j, l_i\}, \tag{5}$$

and the corresponding energy consumption could be obtained as

$$e_{i,(\cdot)}^{\text{tr}} = d_{i,(\cdot)}^{\text{tr}} p_i^{\text{tr}}, \ (\cdot) \in \{j, l_i\}. \tag{6}$$

For level 1 horizontal edge computing, task Q_i is transmitted from MS l_i to MS $k \in \hat{\mathcal{M}}_i$ via the wired link, commonly established through an optical fiber connection with a notably high link capacity. This round-trip delay is denoted as $d_{i,k}^{\text{rrt}}$ which could be assumed to be fixed and estimated with the historical data beforehand [29]. For level 2 cloud computing, task Q_i would be offloaded from MS l_i to the CS via the backhaul wired link, and this transmission delay is denoted as d^{BC}.

C. Computation Model

1) Level 0 Local Computation With D2D Cooperation: In Level 0, task Q_i could be processed locally or by a peer MD with D2D cooperation. For local computing, the required computation delay for Q_i could be obtained as

$$d_i^{\text{L}} = d_i^{\text{comp}} = \frac{C_i}{f_i^{\text{L}}}, \tag{7}$$

where f_i^{L} is the computation capability of the local MD i (in CPU frequency). The corresponding energy consumption for processing Q_i could be obtained as

$$e_i^{\text{L}} = d_i^{\text{L}} p_i^{\text{L}}, \tag{8}$$

where p_i^{L} is the computation power of the local MD i (in watt).

For the D2D cooperation computing, task Q_i is transferred from MD i to the peer MD $j \in \hat{\mathcal{N}}_i$ for processing, and the total delay d_i^{D} is composed of the computation delay d_j^{comp} and the transmission delay $d_{i,j}^{\text{tr}}$, and we have

$$d_i^{\text{D}} = d_{i,j}^{\text{tr}} + d_j^{\text{comp}}. \tag{9}$$

Also, the energy consumption for this D2D cooperation could be calculated as

$$e_i^{\text{D}} = d_{i,j}^{\text{tr}} p_i^{\text{tr}} + d_j^{\text{comp}} p_j^{\text{L}}. \tag{10}$$

2) Level 1 Edge Computation With Horizontal Edge Cooperation: In Level 1, task Q_i is vertically offloaded to its associated MS l_i, and then it can be processed by MS l_i itself or other MS $k \in \hat{\mathcal{M}}_i$ with horizontal edge cooperation. For vertical edge computing, the required computation delay for Q_i in MS l_i could be calculated as

$$d_{i,l_i}^{\text{comp}} = \frac{C_i}{f_{i,l_i}}, \tag{11}$$

where f_{i,l_i} is the computation capability of MS l_i for task Q_i. And the total delay d_{i,l_i}^{V} for vertical edge computing consists of the transmission delay d_{i,l_i}^{tr} between MD i and MS l_i and the computation delay d_{i,l_i}^{comp}. Combining (11) and Section II-B, we can obtain

$$d_{i,l_i}^{\text{V}} = d_{i,l_i}^{\text{tr}} + d_{i,l_i}^{\text{comp}}. \tag{12}$$

Similarly, for the horizontal edge computing between MS l_i and the MS K, the total delay $d_{i,k}^{\text{H}}$ is composed of the computation delay for task Q_i in MS k ($d_{i,k}^{\text{comp}} = \frac{C_i}{f_{i,k}}$), the transmission delay

d^{tr}_{i,l_i} between from the MD i, and the round-trip delay $d^{\text{rrt}}_{l_i,k}$ between MS l_i and k, and we have

$$d^{\text{H}}_{i,k} = d^{\text{tr}}_{i,l_i} + d^{\text{rrt}}_{l_i,k} + d^{\text{comp}}_{i,k}. \tag{13}$$

Besides, for MS k, the sum of the computation resource allocated to all MDs in $\mathcal{N}$ should not be more than its total computational capacity $f^{\max}_k$, and we have

$$\sum_{i\in\mathcal{N}} f_{i,k} \le f^{\max}_k, \quad k \in \mathcal{M}. \tag{14}$$

3) Level 2 Cloud Computation: In Level 2, task Q_i is first transmitted to the associated MS l_i, and then it is transmitted to the CS. The required computation delay by the CS could be obtained as

$$d^{\text{comp}}_{i,c} = \frac{C_i}{f_{i,c}}, \tag{15}$$

where $f_{i,c}$ is the computation capability of the CS, which is a predetermined value according to the cloud computing service [5]. And the CS has a sufficiently large capacity than those of MSs, so we have $f_{i,c} \gg f_{i,k}$. Here, the total delay for the cloud computation is composed of the transmission delay d^{tr}_{i,l_i}, the backhaul delay d^{BC}, and the computation delay $d^{\text{comp}}_{i,c}$, and we have

$$d^{\text{C}}_{i,c} = d^{\text{tr}}_{i,l_i} + d^{\text{BC}} + d^{\text{comp}}_{i,c}. \tag{16}$$

D. Problem Formulation

Here, we formulate the joint task offloading and resource allocation problem[1] to minimize the average delay for all MDs subject to communication, computation, and energy constraints. Accordingly, the full-dimensional task offloading optimization problem is formulated as

$$\begin{aligned}\boldsymbol{P}: \min_{\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{f},\mathbf{b}} \sum_{i\in\mathcal{N}} \Bigg(& x_{i,i} d^{\text{comp}}_i + \sum_{j\in\hat{\mathcal{N}}_i} x_{i,j}(d^{\text{tr}}_{i,j} + d^{\text{comp}}_j) \\ & + y_{i,l_i}(d^{\text{tr}}_{i,l_i} + d^{\text{comp}}_{i,l_i}) + \sum_{k\in\hat{\mathcal{M}}_i} y_{i,k}(d^{\text{tr}}_{i,l_i} + d^{\text{rrt}}_{l_i,k} + d^{\text{comp}}_{i,k}) \\ & + z_i(d^{\text{tr}}_{i,l_i} + d^{\text{BC}} + d^{\text{comp}}_{i,c}) \Bigg), \end{aligned} \tag{17}$$

$$\text{s.t.} \quad x_{i,i} d^{\text{comp}}_i + \sum_{j\in\hat{\mathcal{N}}_i} x_{i,j} d^{\text{tr}}_{i,j} p^{\text{tr}}_i + (1 - \sum_{j\in\mathcal{N}} x_{i,j}) d^{\text{tr}}_{i,l_i} p^{\text{tr}}_i \le e^{\max}_i, \tag{17a}$$

$$x_{i,i} + \sum_{j\in\hat{\mathcal{N}}_i} x_{i,j} + \sum_{k\in\mathcal{M}} y_{i,k} + z_i = 1, \tag{17b}$$

$$\sum_{i\in\mathcal{U}_k} b_i \le B^{\max}_k,\ b_i > 0, \tag{17c}$$

$$\sum_{i\in\mathcal{N}} f_{i,k} \le f^{\max}_k,\ f_{i,k} > 0,\ k \in \mathcal{M}, \tag{17d}$$

$$\mathbf{x}, \mathbf{y}, \mathbf{z} \in \{0,1\}, \tag{17e}$$

where $\mathbf{x} = [x_{1,1},\ldots,x_{1,N},\ldots,x_{N,1},\ldots,x_{N,N}]^T$, $\mathbf{y} = [y_{1,1},\ldots,y_{1,M},\ldots,y_{N,1},\ldots,y_{N,M}]^T$, $\mathbf{z} = [z_1,\ldots,z_N]^T$, $\mathbf{f} = [f_{1,1},\ldots,f_{1,M},\ldots,f_{N,1},\ldots,f_{N,M}]^T$, and $\mathbf{b} = [b_1,\ldots,b_N]^T$. For convenience, we denote the discrete task offloading as $\boldsymbol{\alpha} = [\mathbf{x}^T, \mathbf{y}^T, \mathbf{z}^T]^T$, and resource allocation as $\boldsymbol{\beta} = [\mathbf{f}^T, \mathbf{b}^T]^T$. The objective function $\boldsymbol{O}(\boldsymbol{\alpha},\boldsymbol{\beta})$ and $\boldsymbol{C}(\boldsymbol{\alpha},\boldsymbol{\beta})$ in (17) represents the total delay for all MDs subject to constraints $\boldsymbol{C}(\boldsymbol{\alpha},\boldsymbol{\beta})$ in (17a)–(17e). In practice, a centralized controller executes the optimization algorithm to solve $\boldsymbol{P}$. The controller is much closer to MSs than to the CS benefiting from low latency transmission via high-speed wired links. The controller requires two types of information: system information (e.g., local computation capability f^{L}_i, local energy capability $e^{\max}_i$, MS computation capability $B^{\max}_k$, and CS computation capability $f_{i,k}$) and task-related information (e.g., data size D_i, channel gain between MD and the associated MS g_{i,l_i}, and channel gains between MDs $\{g_{i,j} \mid j = 1,\ldots,N\}$). The controller has pre-stored the system information, so it only needs to collect task-related information, typically in kilobytes (KB), which leads to a very low latency of probing. Therefore, the latency of probing is not considered in $\boldsymbol{P}$.

The problem $\boldsymbol{P}$ is an MINLP with discrete variables $\boldsymbol{\alpha}$ and continuous variables $\boldsymbol{\beta}$. In fact, $\boldsymbol{P}$ consists of $2^{N(N+M+1)}$ sub-problems. In other words, its computational complexity increases exponentially with the dimension of $\boldsymbol{\alpha}$. As one of the state-of-the-art algorithms, B&B can find a global optimal solution to $\boldsymbol{P}$, and it can fathom sub-problems with relaxations to control the exponential nature of the search. Besides, due to constraint (17b), the complexity of the original B&B can be further reduced. For example, if $x_{i,j} = 1$, naturally we have $y_{i,j} = 0$ and $z_i = 0$. Therefore, in the worst case, the complexity has been reduced to solving $2^{N(M+1)}$ sub-problems. Despite the complexity reduction, it is still of exponential nature. Consequently, no optimal solution could be found by a polynomial-time algorithm. Therefore, our optimization problem $\boldsymbol{P}$ is NP-hard. To address this challenge, with relaxation, we translate our formulated $\boldsymbol{P}$ to a convex optimization problem $\boldsymbol{P}'$ which is suitable to be solved with the B&B algorithm. Specifically, we can translate $\boldsymbol{P}$ into

$$\begin{aligned}\boldsymbol{P}': \min_{\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{f},\mathbf{b}} \sum_{i\in\mathcal{N}} \Bigg(& \sum_{j\in\mathcal{N}} x^2_{i,j} \left(\frac{D_i}{r_{i,j}} + \frac{C_i}{f^{\text{L}}_i}\right) + y^2_{i,l_i} \left(\frac{D_i}{r_{i,l_i}} + \frac{C_i}{f_{i,l_i}}\right) \\ & + \sum_{k\in\hat{\mathcal{M}}_i} \left(y^2_{i,k} \left(\frac{D_i}{r_{i,l_i}} + \frac{C_i}{f_{i,k}}\right) + y_{i,k} d^{\text{rrt}}_{l_i,k} \right) \\ & + z^2_i \left(\frac{D_i}{r_{i,l_i}} + \frac{C_i}{f_{i,c}}\right) + z_i d^{\text{BC}} \Bigg), \end{aligned} \tag{18}$$

[1]Note that a single objective (average latency) is considered in this paper. Our algorithm could also be adopted to solve the multi-objective problem by transforming it into a single objective [30].

$$\text{s.t. } x_{i,i}^2 \frac{C_i p_i^{\mathrm{L}}}{f_i^{\mathrm{L}}} + \sum_{j\in\mathcal{N}} x_{i,j}^2 \frac{D_i p_i^{\mathrm{tr}}}{r_{i,j}}$$

$$+ (y_{i,l_i}^2 + \sum_{k\in\hat{\mathcal{M}}_i} y_{i,k}^2 + z_i^2) \frac{D_i p_i^{\mathrm{tr}}}{r_{i,l_i}} \le e_i^{\max}, \quad (18a)$$

$$\sum_{j\in\mathcal{N}} x_{i,j} + \sum_{k\in\mathcal{M}} y_{i,k} + z_i = 1, \quad (18b)$$

$$\sum_{i\in\mathcal{U}_k} b_i \le B_k^{\max},\ b_i > 0, \quad (18c)$$

$$\sum_{i\in\mathcal{N}} f_{i,k} \le f_k^{\max},\ f_{i,k} > 0,\ k \in \mathcal{M}, \quad (18d)$$

$$\mathbf{x}, \mathbf{y}, \mathbf{z} \in \{0, 1\}, \quad (18e)$$

where we assume that $r_{i,i} = \infty$ ($d_{i,i}^{\mathrm{tr}} = 0$), and the local computing could be a special D2D cooperation. For the binary variables $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$, we implement the quadratic expression $\mathbf{x}^2$, $\mathbf{y}^2$ and $\mathbf{z}^2$ to replace them. As a result, the form like $x_{i,j}\frac{C_i}{f_i^{\mathrm{L}}}$ is translated to a convex function $x_{i,j}^2\frac{C_i}{f_i^{\mathrm{L}}}$. When binary variables $\boldsymbol{\alpha}$ are relaxed, $\boldsymbol{P}'$ is a convex optimization problem. Though we can obtain the optimal solution of $\boldsymbol{P}'$ with the B&B algorithm, there are $2^{N(M+1)}$ sub-problems to be solved in the worst case and the computational complexity is extremely high. This calls for efficient strategies that scale favorably with the problem size.

III. Branch-and-Bound Algorithm

In this section, we introduce the B&B algorithm, an optimal approach to solving combinatorial optimization MINLP problems, and provide an example to illustrate the process of the B&B algorithm. Then we introduce the variable selection policy of the B&B algorithm.

A. Workflow of the Branch-and-Bound Algorithm

The B&B algorithm, combining a diverse mixture of heuristics, is capable of solving reasonably sized MINLP problems within an acceptable time duration while guaranteeing optimality. The key idea is to sequentially partition the original MINLP problem into smaller problems with relaxations to control the exponential nature of the search. The B&B algorithm could be represented as an enumeration tree composed of multiple edges and nodes (e.g., the enumeration tree shown in Fig. 2(a)). The nodes in the enumeration tree consist of a root node (e.g., node 0) and multiple sub-tree nodes (e.g., nodes 1-6). The root node comes from the relaxed original MINLP problem by relaxing all the discrete variables to continuous ones, and it corresponds to a convex nonlinear programming problem (NLP), which can be solved by many convex optimization algorithms such as the interior-point method [31]. The enumeration tree starts at the root node, and then sub-tree nodes are added sequentially until the optimal solution is found. The construction complies with three policies: node selection policy, node pruning policy, and variable selection policy all to be elaborated later.

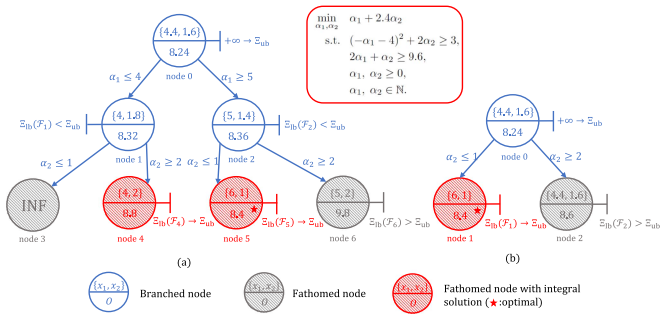


Fig. 2. Example of the B&B algorithm. (a) The variable α_1 is first selected to branch. (b) The variable α_2 is first selected to branch.

Node selection policy determines which node to process, and is commonly based on search methods, such as depth-first search and best-first search [16]. Assume that the l-th node first pops out from the unbranched node list, denoted by $\mathscr{L}_{\mathrm{n}}$, based on a node selection policy. Here, the general MINLP problem can be expressed by

$$\min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \boldsymbol{O}(\boldsymbol{\alpha},\boldsymbol{\beta}) \quad \text{s.t. } (\boldsymbol{\alpha},\boldsymbol{\beta}) \in \mathcal{F}_l, \quad (19)$$

where $\mathcal{F}_l$ represents the feasible set of the problem at l-th node. The optimal solution of (19) is denoted as $\{\boldsymbol{\alpha}^*(l), \boldsymbol{\beta}^*(l)\}$, and its corresponding optimal objective value is denoted as $\boldsymbol{\Xi}^*(\mathcal{F}_l)$. We further denote the lower bound of $\boldsymbol{\Xi}^*(\mathcal{F}_l)$ as $\boldsymbol{\Xi}_{\mathrm{lb}}(\mathcal{F}_l)$. Meanwhile, we denote the upper bound of $\boldsymbol{\Xi}^*(\mathcal{F}_l)$ as $\boldsymbol{\Xi}_{\mathrm{ub}}$ that is updated iteratively. Clearly, we have

$$\boldsymbol{\Xi}_{\mathrm{lb}}(\mathcal{F}_l) \le \boldsymbol{\Xi}^*(\mathcal{F}_l) \le \boldsymbol{\Xi}_{\mathrm{ub}}. \quad (20)$$

Next, we elaborate on the node selection, node pruning, as well as the rule to update $\boldsymbol{\Xi}_{\mathrm{ub}}$. Initially, $\boldsymbol{\Xi}_{\mathrm{ub}}$ is assumed to be infinity. Then, the pruning policy would determine whether this node will be fathomed if one of the following three conditions is met. 1) (19) is infeasible. 2) $\boldsymbol{\Xi}_{\mathrm{lb}}(\mathcal{F}_l) > \boldsymbol{\Xi}_{\mathrm{ub}}$, which means that all solutions of child nodes below cannot be better than $\boldsymbol{\Xi}_{\mathrm{ub}}$ in this branch. 3) $\boldsymbol{\Xi}_{\mathrm{lb}}(\mathcal{F}_l) < \boldsymbol{\Xi}_{\mathrm{ub}}$ and all variables in $\boldsymbol{\alpha}^*(l)$ are integers, which means that $\boldsymbol{\alpha}^*(l)$ is the optimal solution in this enumeration tree. In the meantime, the upper bound is updated as $\boldsymbol{\Xi}_{\mathrm{lb}}(\mathcal{F}_l) \to \boldsymbol{\Xi}_{\mathrm{ub}}$. If the l-th node is reserved, there will be some variables that need to be branched, and they constitute an unbranched variable list $\mathscr{L}_{\mathrm{v}}$.

The next step in the B&B algorithm is the variable selection process. In this process, we need to determine which fractional variable in $\mathscr{L}_{\mathrm{v}}$ should be selected for further branching. The variable α_n is chosen based on the variable selection policy to be elaborated in Section III-B. The optimal value of α_n, denoted as $\alpha_n^{(l)}$, can be found in $\boldsymbol{\alpha}^*(l)$. Given $\alpha_n^{(l)}$, the feasible set $\mathcal{F}_l$ is partition into two subsets

$$\begin{cases} \mathcal{F}_l^- &= \mathcal{F}_l \cap \{(\boldsymbol{\alpha},\boldsymbol{\beta}) \mid \alpha_n \le \lfloor \alpha_n^{(l)} \rfloor\} \\ \mathcal{F}_l^+ &= \mathcal{F}_l \cap \{(\boldsymbol{\alpha},\boldsymbol{\beta}) \mid \alpha_n \ge \lceil \alpha_n^{(l)} \rceil\}. \end{cases} \quad (21)$$

Accordingly, (19) is divided into two sub-problems

$$\min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \boldsymbol{O}(\boldsymbol{\alpha},\boldsymbol{\beta}) \ \text{s.t.}(\boldsymbol{\alpha},\boldsymbol{\beta}) \in \mathcal{F}_l^-,$$
$$\min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \boldsymbol{O}(\boldsymbol{\alpha},\boldsymbol{\beta}) \ \text{s.t.}(\boldsymbol{\alpha},\boldsymbol{\beta}) \in \mathcal{F}_l^+. \tag{22}$$

These two sub-problems are assigned to the next two child nodes which then will be added to the list $\mathscr{L}_{\text{n}}$. The enumeration tree grows after every iteration until there are no nodes left in the list $\mathscr{L}_{\text{n}}$.

Next, we provide an example in Fig. 2(a) to illustrate the process aforementioned.

- We start from the root node 0. The relaxed problem corresponds to $\mathcal{F}_0$ where α_1 and α_2 are relaxed to the continuous variables. We have $\alpha_1^{(0)} = 4.4$, $\alpha_2^{(0)} = 1.6$, and $\Xi^*(\mathcal{F}_0) = 8.24$. Assuming that α_1 is selected for further branching, we obtain two child nodes 1 and 2. For node 1, we have $\alpha_1^{(1)} = 4$, $\alpha_2^{(1)} = 1.8$, and $\Xi^*(\mathcal{F}_1) = 8.32$. As $\Xi_{\text{lb}}(\mathcal{F}_1) = \Xi^*(\mathcal{F}_1) < \Xi_{\text{ub}}$, and $\alpha_2^{(1)}$ is fractional, node 1 will be branched. Similarly, node 2 will also be branched.
- For node 1, as only α_2 is left in $\mathscr{L}_{\text{v}}$, we branch it, resulting in nodes 3 and 4. For node 3, the corresponding problem is infeasible, so this node is fathomed. For node 4, the relaxed solutions are $\alpha_1^{(4)} = 4$, $\alpha_2^{(4)} = 2$, and $\Xi^*(\mathcal{F}_4) = 8.8$. As $\alpha_1^{(4)}$ and $\alpha_2^{(4)}$ are integers while $\Xi_{\text{lb}}(\mathcal{F}_4) = \Xi^*(\mathcal{F}_4) < \Xi_{\text{ub}}$, we set $\Xi_{\text{ub}} = 8.8$. Then, node 4 is fathomed.
- For node 2, node 5 is first selected to branch. Clearly, $\alpha_1^{(5)}$ and $\alpha_2^{(5)}$ are integers and $\Xi_{\text{lb}}(\mathcal{F}_5) = \Xi^*(\mathcal{F}_5) = 8.4 < \Xi_{\text{ub}} = 8.8$, thus optimal node shifts from node 4 to 5. Then only node 6 is left in $\mathscr{L}_{\text{n}}$, where $\Xi_{\text{lb}}(\mathcal{F}_6) > \Xi_{\text{ub}}$, meaning that the last node has been fathomed, and the B&B process has finished. Thereby we reach the conclusion that the global optimal node is node 5 and the optimal solution is 8.4.

B. Variable Selection Policy

To improve the basic B&B routine, efforts in the literature have been placed on more efficient node selection and pruning. However, the variable selection has been less considered, although it significantly impacts the computational complexity of the B&B algorithm. For example, in Fig. 2(a), if α_1 is selected first, there will be 7 nodes attached in the tree (0,...,6), in stark contrast with only 3 nodes in Fig. 2(b) if α_2 is selected first. This illustrates how important the order of selection is, not to mention the case with more variables.

In essence, the variable selection policy has a direct impact on the size of the enumeration tree. The FSB known as the optimal variable selection policy has the highest node creation efficiency. Compared to any other policy, it is able to find the minimum size of an enumeration tree. The FSB works as follows. In node l, the variable α_n with the best $\mathbf{score}_n^{(l)*}$ is selected to branch, where $\mathbf{score}_n^{(l)*} = \max\{\mathbf{score}_n^{(l)} \,|\, a_n \in \mathscr{L}_{\text{v}}\}$. Here, the score function of the FSB is given by

$$\mathbf{score}_n^{(l)} = (1-\mu)\cdot \min\{\Delta_n^{(l)-}, \Delta_n^{(l)+}\} + \mu \cdot \max\{\Delta_n^{(l)-}, \Delta_n^{(l)+}\}, \tag{23}$$

where $0 \le \mu \le 1$ is the score factor that can be adjusted dynamically. In addition, we have $\Delta_n^{(l)-} = \Xi^*(\mathcal{F}_{l+1}) - \Xi^*(\mathcal{F}_l)$ and $\Delta_n^{(l)+} = \Xi^*(\mathcal{F}_{l+2}) - \Xi^*(\mathcal{F}_l)$. Clearly, $\Xi^*(\mathcal{F}_{l+1})$ and $\Xi^*(\mathcal{F}_{l+2})$ could be obtained from two NLPs with new constraints $\alpha_n \le \lfloor \alpha_n^{(l)} \rfloor$ and $\alpha_n \ge \lceil \alpha_n^{(l)} \rceil$, respectively. For example, in Fig. 2(a), child nodes of node 0 are nodes 1 and 2, where $\Xi^*(\mathcal{F}_0) = 8.24$, $\Xi^*(\mathcal{F}_1) = 8.32$ and $\Xi^*(\mathcal{F}_2) = 8.36$. If we choose $\mu = 1/6$, we have the score $\mathbf{score}_1^{(1)} = 0.11$ for α_1. In Fig. 2(b), we can calculate the score $\mathbf{score}_2^{(1)} = 0.33$ for α_2, therefore selecting α_2 for branching in this problem can achieve a smaller enumeration tree and a lower computation complexity compared with selecting α_1 for branching.

Reliability pseudo-cost branching policy (RPB) [19] is a representative low-complexity variable selection method compared to FSB. Essentially, it is a hybrid branching method that uses strong branching for variable selection at the initialization and pseudo-cost branching for the rest variables.

Remark 1: Note that the variable selection policy determines the order of the variables for branching, and does not involve the optimal node pruning. Therefore, the B&B algorithm with various variable selection policies (including FSB) all can get the global optimal solution (with different complexity).

IV. The Proposed GIRL Variable Selection Policy

In this section, we first provide our motivation to accelerate the optimal B&B algorithm with significantly reduced complexity but without sacrificing the global optimality. Then, we build the variable selection policy as an MDP (Markov Decision Process) [32]. Finally, we introduce the framework of the GIRL variable selection policy, including Q-learning with modeling learning and the maximum entropy (MaxEnt) gradients.

A. Motivation

It is clear that, although the FSB can achieve the minimum size of the enumeration tree and the lowest computational complexity of the node solving process (NLPs solving), it needs to solve two NLPs for each candidate in $\mathscr{L}_{\text{v}}$ of one node. Therefore, the variable selection process of the FSB is extremely costly, especially for a large number of variables. This hinders the implementation of the FSB for our multi-layer full-dimensional task offloading scheme. To address this problem, we propose a novel data-driven variable selection policy referred to as the GIRL. The essence of our approach lies in the exploitation of the learning mechanism to generate a new variable selection policy that closely matches the FSB variable selection, but without involving the tedious computations in the FSB.

B. Markov Decision Process

The B&B algorithm is a sequential decision process with an episodic variable selection. In this case, we formulate this process as an MDP from a model-free perspective. Here, we take the variable selector and the sub-problem solver as the agent and the environment, respectively. The major components of the MDP are summarized as follows.

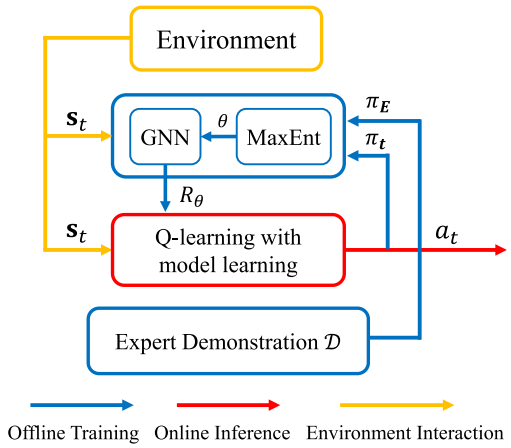


Fig. 3. Framework of the GIRL.

1) State and Action: The state $\mathbf{s}_t$ is related to the entire already solved enumeration tree at t-th iteration, and the state space is $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \ldots\}$ consisting of variable, constraint and edge features to be specified in Section V-A. Based on $\mathbf{s}_t$, the action a_t selects a variable from $\mathscr{L}_\mathrm{v}$, and the action space could be represented as $\mathcal{A} = \{a_1, a_2, a_3 \ldots\}$. After a_t is taken, the current problem would be branched, and $\mathbf{s}_t$ transits to $\mathbf{s}_{t+1}$ with the state transition probability $\mathbb{P}(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a_t)$.

2) Trajectory: The trajectory from the root node to the final fathomed node is represented as $\tau = \{(\mathbf{s}_t, a_t) \mid t = 1, 2, 3, \ldots\}$, capturing the sequence of state-action pairs. When referring to the trajectory starting from the root node ς, it is denoted as τ_ς. The trajectory space of τ_ς is denoted by $\Omega_\varsigma = \{\tau_\varsigma^1, \tau_\varsigma^2, \tau_\varsigma^3 \ldots\}$.

3) Reward and Policy: We denote the immediate reward as $R(\mathbf{s}_t, a_t)$ for the state-action pair $(\mathbf{s}_t, a_t)$, which tells the agent whether the action a_t is good or not. Given the state $\mathbf{s}_t$, policy $\pi : (\mathcal{S} \to \mathcal{A})$ maps the state to the action.

C. The Framework of the GIRL

It is well known that the RL can handle a general MDP in an online manner with a low computational complexity. Typically, given a reward function, the RL attempts to learn an optimal policy to map the state to the action. However, the RL is inapplicable to the formulated MDP, because in our problem it is very challenging to design an appropriate reward function. Before obtaining the optimal solution, the branching order of variables cannot be known in advance. That is, we cannot know that branching which variable can generate a smaller enumeration tree. For example, in Fig. 2, before we get the optimal solution, we can not decide to branch variable α_1 or α_2 in advance.

To address this challenge, we resort to imitation learning[2] and design an IRL-based framework as shown in Fig. 3. Instead of trying to manually specify a reward function, we design a GNN $\mathcal{G}$ with neural network parameters $\boldsymbol{\theta}$ as the parameterized reward function, which is denoted as $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$. Subsequently, $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$ is integrated into the formulated MDP described in Section IV-B, ultimately yielding a policy that aligns with that of the expert demonstration $\mathcal{D}$ to be elaborated in Section VI-B2.

The workflow of the proposed GIRL is summarized in Algorithm 1. Specifically, after initializing the network parameters $\boldsymbol{\theta}$, each iteration can be divided into three phases: 1) updating reward function $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$ via the GNN $\mathcal{G}$ (Line 3); 2) updating the policy (Lines 4–7); 3) updating $\boldsymbol{\theta}$ based on the expert demonstration $\mathcal{D}$ (Lines 8–15). In the first phase, we first update the GNN $\mathcal{G}$ based on $\boldsymbol{\theta}$, and then the GNN $\mathcal{G}$ will generate a new $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$. In the second phase, we update the policy π_t and obtain $\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$ via the Q-learning with the model learning, and π_t is implemented to calculate the expected state-action visitation counts $\mathbb{E}[\mu_\varsigma(\mathbf{s}_t, a_t)]$ (Lines 5–7). In the final phase, $\mu_{\mathcal{D}_\varsigma}(\mathbf{s}_t, a_t)$ is calculated based on the expert demonstration $\mathcal{D}_\varsigma$, and then MaxEnt gradients $\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)}$ are obtained. We can obtain $\frac{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)}{\partial \boldsymbol{\theta}}$ by backward propagating the GNN $\mathcal{G}$. Next, $\boldsymbol{\theta}$ is updated based on the network gradients $\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$. When $\boldsymbol{\theta}$ converges to the optimal value $\boldsymbol{\theta}^*$, the GNN $\mathcal{G}$ could obtain the optimal reward function $R_{\boldsymbol{\theta}^*}(\mathbf{s}_t, a_t)$, and the optimal policy could be trained based on $R_{\boldsymbol{\theta}^*}(\mathbf{s}_t, a_t)$.

[2]Imitation learning [33] can be categorized into behavior cloning and inverse reinforcement learning (IRL). Behavior cloning involves learning a mapping from inputs (states) to outputs (actions). Clearly, it does not work for our full-dimensional task offloading scheme. Instead, our GIRL algorithm, based on IRL, learns a reward function that leads to a good policy aligned with the expert demonstration.

TABLE I
TABLE OF SYMBOLS

Variable	Description
$\boldsymbol{\alpha}, \boldsymbol{\beta}$	The discrete and the continuous variable sets
$\boldsymbol{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$	The constraints with $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$
$\boldsymbol{O}(\boldsymbol{\alpha}, \boldsymbol{\beta})$	The objective function with $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$
$\mathcal{F}_l$	The feasible set of the problem at l-th node
$\Xi^*(\mathcal{F}_l)$	The optimal objective value at l-th node
$\Xi_\mathrm{lb}(\mathcal{F}_l)$	The lower bound of $\Xi^*(\mathcal{F}_l)$
Ξ_ub	The upper bound of $\Xi^*(\mathcal{F}_l)$
$\mathscr{L}_\mathrm{n}$, $\mathscr{L}_\mathrm{v}$	Unbranched node and variable lists
α_n	The n-th variable in $\mathscr{L}_\mathrm{v}$
$\alpha_n^{(l)}$	The optimal solution of α_n at l-th node
$\mathcal{A}$, $\mathcal{S}$	The action and the state space
ς	The root node of the enumeration tree
τ_ς	The trajectory starting from the root node ς
Ω_ς	The trajectory space of τ_ς
$\mathcal{D}$	The expert demonstration
$\mathcal{D}_\varsigma$	The expert demonstration with ς
$\mathcal{D}'$	The self-imitation demonstration
γ, λ	The discount rate and the learning rates
π_t, π_E	The policy of the GIRL and the expert policy

Algorithm 1: Inverse Reinforcement Learning With Graph Neural Networks.

Input: Expert policy $\mu_{\mathcal{D}_\varsigma}$
1: Initialize $\boldsymbol{\theta}$.
2: **while** $\boldsymbol{\theta}$ not Converge **do**
3: Update reward function $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$ via the GNN $\mathcal{G}$.
4: Update policy π_t and obtain $\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$ via Q-learning with model learning.
5: **for** ς in $\mathcal{S}$ **do**
6: Calculate the expected state-action visitation counts $\mathbb{E}[\mu_\varsigma(\mathbf{s}_t, a_t)]$ based on the policy π_t.
7: **end for**
8: Determine MaxEnt gradients
9: $\dfrac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)} \leftarrow \sum_{\varsigma \in \mathcal{S}} (\mu_{\mathcal{D}_\varsigma}(\mathbf{s}_t, a_t) - \mathbb{E}[\mu_\varsigma(\mathbf{s}_t, a_t)])$.
10: Obtain $\dfrac{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)}{\partial \boldsymbol{\theta}}$ backward propagating neural-network.
11: Compute network gradients
12: $\dfrac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \leftarrow \sum_{\mathbf{s}_t \in \mathcal{S}} \sum_{a_t \in \mathcal{A}} \dfrac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)} \cdot \dfrac{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)}{\partial \boldsymbol{\theta}}$.
13: Update neural-network parameters $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$.
14: **end while**
15: Return the optimal $\boldsymbol{\theta}^*$.
16: Obtain the optimal reward function $R_{\boldsymbol{\theta}^*}(\mathbf{s}_t, a_t)$ via the GNN $\mathcal{G}$.
17: Obtain policy π_t^* via Q-learning with model learning.
Output: Optimal policy π_t^*

1) Q-Learning With Modeling Learning: Q-learning with model learning not only infers the action a_t given the state $\mathbf{s}_t$, but also learns a model by generating the transition probability $\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$. In the proposed GIRL, $\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$ plays a crucial role as it serves as input to the MaxEnt for updating the parameterized reward function (GNN $\mathcal{G}$).

Next, we elaborate on Q-Learning with modeling learning. According to [34], following policy π, the Bellman equation could be expressed as

$$V^\pi(\mathbf{s}_t) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} \mid \mathbf{s}_t, \pi\right] = \mathbb{E}[R_t + \gamma V^\pi(\mathbf{s}_{t+1}) \mid \mathbf{s}_t, \pi], \tag{24}$$

where

$$R_t = R(\mathbf{s}_t, a_t) = \sum_{\mathbf{s}_{t+1} \in \mathcal{S}} \mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t) R(\mathbf{s}_t, a_t, \mathbf{s}_{t+1}), \tag{25}$$

and $\gamma \in [0, 1)$ is the discount rate. Here, $V^\pi(\mathbf{s}_t)$ equivalently represents the expected discounted cumulative reward starting at state $\mathbf{s}_t$. Next, following policy π, we formulate the state–action value as

$$Q^\pi(\mathbf{s}_t, a_t) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} \mid \mathbf{s}_t, a_t, \pi\right] = \mathbb{E}\left[R_t + \gamma Q^\pi(\mathbf{s}_{t+1}, a_{t+1}) \mid \mathbf{s}_t, a_t, \pi\right]. \tag{26}$$

Compared with $V^\pi(\mathbf{s}_t)$, $Q^\pi(\mathbf{s}_t, a_t)$ equivalently represents the expected discounted cumulative reward starting at state $\mathbf{s}_t$ after taking action a_t. Substituting (26) into (24), the Bellman equation could be written as

$$V^\pi(\mathbf{s}_t) = \sum_{a_t \in \mathcal{A}} \pi(\mathbf{s}_t, a_t) Q^\pi(\mathbf{s}_t, a_t). \tag{27}$$

The objective of the Q-learning with model learning is to learn the optimal policy π^*. In our approach, π^* could be obtained when $V^\pi(\mathbf{s}_t)$ converges to maximum, or, equivalently, when $Q^\pi(\mathbf{s}_t, a_t)$ converges to maximum. Mathematically, we have

$$\pi^* = \arg\max_\pi Q^\pi(\mathbf{s}_t, a_t). \tag{28}$$

Clearly, π^* corresponds to the optimal state–action value $Q^*(\mathbf{s}_t, a_t)$, which satisfies the Bellman optimality equation [35] as follows

$$Q^*(\mathbf{s}_t, a_t) = \mathbb{E}\left[R_t + \gamma \max_{a_{t+1} \in \mathcal{A}} Q^*(\mathbf{s}_{t+1}, a_{t+1}) \mid \mathbf{s}_t, a_t, \pi\right]. \tag{29}$$

To obtain the optimal state–action value $Q^*(\mathbf{s}_t, a_t)$, we should follow the iterative process based on the updated law

$$Q^\pi(\mathbf{s}_t, a_t) \leftarrow Q^\pi(\mathbf{s}_t, a_t) + \lambda(R_t + \gamma \max_{a_{t+1} \in \mathcal{A}} Q^\pi(\mathbf{s}_{t+1}, a_{t+1}) - Q^\pi(\mathbf{s}_t, a_t)), \tag{30}$$

where λ is the learning rate. By iteratively updating the Q-values using (30), the Q-learning progressively approaches $Q^*(\mathbf{s}_t, a_t)$, which maximizes the expected cumulative reward for every state-action pair. Consequently, the expected state-action visitation counts $\mathbb{E}[\mu_\varsigma(\mathbf{s}_t, a_t)]$ could be generated based on $Q^*(\mathbf{s}_t, a_t)$ according to [36]. Besides, the transition probability $\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$ could be estimated by the following equation

$$\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t) = \frac{\upsilon(\mathbf{s}_t, a_t, \mathbf{s}_{t+1})}{\sum_{\mathbf{s}_{t+1} \in \mathcal{S}} \upsilon(\mathbf{s}_t, a_t, \mathbf{s}_{t+1})}, \tag{31}$$

where $\upsilon(\mathbf{s}_t, a_t, \mathbf{s}_{t+1})$ is the number of state transition from $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$.

2) MaxEnt Gradients: In this step, we present our approach to update $\boldsymbol{\theta}$ in the GNN $\mathcal{G}$ by leveraging MaxEnt gradients [36]. We first introduce some definitions as follows. The expert demonstration $\mathcal{D}$ is grouped into $\mathcal{D}_\varsigma$, and we have $\mathcal{D} = \{\mathcal{D}_\varsigma \mid \varsigma \in \mathcal{S}\}$. Specifically, $\mathcal{D}_\varsigma = \{\tau_\varsigma^i \mid i = 1, 2, 3, \ldots, N_\varsigma\}$ consists of all trajectories starting from the root node ς and ending at the optimal node, where N_ς is the number of trajectories starting from ς. With these definitions, then we will maximize the entropy of $\boldsymbol{\theta}$ subject to the expert demonstration.

The reward function $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$ indicates whether a_t follows the branching strategy of the FSB. It quantifies the extent to which the actions in the expert demonstration align with the prescribed FSB strategy. On this basis, $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$ is correlated to $\mathbb{P}(a_t|\boldsymbol{\theta})$, the probability of the action a_t in the expert demonstration, thus, we have $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t) \propto \mathbb{P}(a_t|\boldsymbol{\theta})$. In addition, $\mathbb{P}(a_t|\boldsymbol{\theta})$ is equivalent to the probability of the trajectory τ with a_t in the expert demonstration, and we have $\mathbb{P}(a_t|\boldsymbol{\theta}) \propto \sum_{\tau: a_t \in \tau} \mathbb{P}(\tau|\boldsymbol{\theta})$.

Under the principle of the MaxEnt, $\mathbb{P}(\tau|\boldsymbol{\theta})$ could be approximated as

$$\mathbb{P}(\tau|\boldsymbol{\theta}) \approx \frac{1}{Z_\varsigma(\boldsymbol{\theta})} e^{\sum_\tau R_{\boldsymbol{\theta}}(\mathbf{s}_t,a_t)} \prod_{(\mathbf{s}_t,a_t,\mathbf{s}_{t+1})\in\tau} \mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t,a_t), \tag{32}$$

where $Z_\varsigma(\boldsymbol{\theta}) = \sum_{\tau\in\Omega_\varsigma} e^{\sum_{(\mathbf{s}_t,a_t)\in\tau} R_{\boldsymbol{\theta}}(\mathbf{s}_t,a_t)}$ is the partition distribution function. Based on $\mathbb{P}(\tau|\boldsymbol{\theta})$, the average log-likelihood of $\boldsymbol{\theta}$ over $\mathcal{D}_\varsigma$ could be represented as

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \sum_{\varsigma\in\mathcal{S}} \frac{1}{N_\varsigma} \sum_{\tau\in\mathcal{D}_\varsigma} \log\mathbb{P}(\tau|\boldsymbol{\theta}) \\ &= \sum_{\varsigma\in\mathcal{S}} \left(\frac{1}{N_\varsigma} \left(\sum_{\tau\in\mathcal{D}_\varsigma} \left(\sum_{(\mathbf{s}_t,a_t)\in\tau} R_{\boldsymbol{\theta}}(\mathbf{s}_t,a_t) \right.\right.\right. \\ &\quad \left.\left.\left. + \sum_{(\mathbf{s}_t,a_t,\mathbf{s}_{t+1})\in\tau} \mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t,a_t) \right) \right) - \log Z_\varsigma(\boldsymbol{\theta}) \right)\end{aligned} \tag{33}$$

Note that in this context $-\mathcal{L}(\boldsymbol{\theta})$ represents cross-entropy. The minimum cross-entropy could be referred to as the MaxEnt [37]. Under the MaxEnt, we can obtain the optimal parameters $\boldsymbol{\theta}$ by maximizing $\mathcal{L}(\boldsymbol{\theta})$, given by

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}). \tag{34}$$

To achieve the maximum $\mathcal{L}(\boldsymbol{\theta})$, we take the partial derivative of (33) with respect to $\boldsymbol{\theta}$, and we have

$$\frac{\partial\mathcal{L}(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}} = \sum_{\mathbf{s}_t\in\mathcal{S}} \sum_{a_t\in\mathcal{A}} \frac{\partial\mathcal{L}(\boldsymbol{\theta})}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t,a_t)} \cdot \frac{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t,a_t)}{\partial\boldsymbol{\theta}}. \tag{35}$$

Here, $\frac{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t,a_t)}{\partial\boldsymbol{\theta}}$ could be obtained via the back propagating the GNN $\mathcal{G}$. In this case, we need to derive $\frac{\partial\mathcal{L}(\boldsymbol{\theta})}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t,a_t)}$, which can be written as

$$\begin{aligned}\frac{\partial\mathcal{L}(\boldsymbol{\theta})}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t,a_t)} &= \sum_{\varsigma\in\mathcal{S}} \frac{1}{N_\varsigma} \left(\frac{\partial \sum_{\tau\in\mathcal{D}_\varsigma} \sum_{(\mathbf{s}_t,a_t)\in\tau} R_{\boldsymbol{\theta}}(\mathbf{s}_t,a_t)}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t,a_t)} \right. \\ &\quad - \frac{1}{Z_\varsigma(\boldsymbol{\theta})} \sum_{\tau\in\Omega_\varsigma} e^{\sum_{(\mathbf{s}_t,a_t)\in\tau} R_{\boldsymbol{\theta}}(\mathbf{s}_t,a_t)} \prod_{(\mathbf{s}_t,a_t,\mathbf{s}_{t+1})\in\tau} \mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t,a_t) \\ &\quad \left. \times \frac{\partial \sum_{(\mathbf{s}_t,a_t)\in\tau} R_{\boldsymbol{\theta}}(\mathbf{s}_t,a_t)}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t,a_t)} \right) \\ &= \sum_{\varsigma\in\mathcal{S}} \frac{1}{N_\varsigma} \left(\frac{\partial \sum_{\tau\in\mathcal{D}_\varsigma} \sum_{(\mathbf{s}_t,a_t)\in\tau} R_{\boldsymbol{\theta}}(\mathbf{s}_t,a_t)}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t,a_t)} \right. \\ &\quad \left. - \sum_{\tau\in\Omega_\varsigma} \mathbb{P}(\tau|\boldsymbol{\theta}) \frac{\partial \sum_{(\mathbf{s}_t,a_t)\in\tau} R_{\boldsymbol{\theta}}(\mathbf{s}_t,a_t)}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t,a_t)} \right) \\ &= \sum_{\varsigma\in\mathcal{S}} \left(\mu_{\mathcal{D}_\varsigma}(\mathbf{s}_t,a_t) - \mathbb{E}[\mu_\varsigma(\mathbf{s}_t,a_t)] \right) \triangleq \pi_E - \pi_t.\end{aligned} \tag{36}$$

Furthermore, we define $\pi_E \triangleq \mu_{\mathcal{D}_\varsigma} \triangleq \sum_{\varsigma\in\mathcal{S}}(\mu_{\mathcal{D}_\varsigma}(\mathbf{s}_t,a_t))$ as the expert policy which has the same behavior as the demonstration $\mathcal{D}$, and define $\pi_t \triangleq \sum_{\varsigma\in\mathcal{S}}(\mathbb{E}[\mu_\varsigma(\mathbf{s}_t,a_t)])$ as the GIRL policy. Then parameters $\boldsymbol{\theta}$ could be updated based on $\frac{\partial\mathcal{L}(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}$, and the update process of $\boldsymbol{\theta}$ is simplified as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}, \tag{37}$$

which is elaborated later in Section VI-B1. At the end of the iterations, the optimal $\boldsymbol{\theta}^*$ would be finally obtained. Taking the optimal $\boldsymbol{\theta}^*$ back into the GNN $\mathcal{G}$, the optimal reward function $R_{\boldsymbol{\theta}^*}(\mathbf{s}_t,a_t)$ can be generated. Finally, the optimal policy could be trained based on $R_{\boldsymbol{\theta}^*}(\mathbf{s}_t,a_t)$. After the offline training, the learned GIRL could directly infer the variable selection and achieve a global optimal solution (c.f. Remark 1) to the formulated MINLP problem with significantly low complexity.

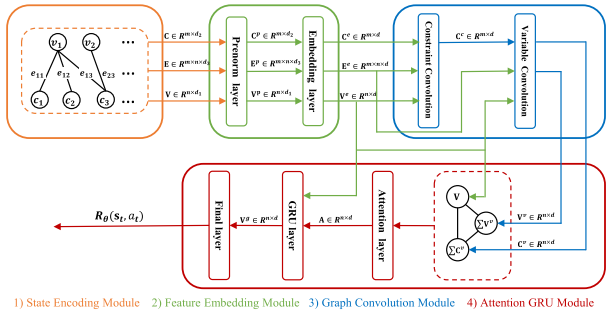


Fig. 4. Functional structure of the proposed GNN.

V. The Proposed Graph Neural Network

In this paper, we propose GIRL by generating a new variable selection policy to accelerate the B&B algorithm as shown in Fig. 3. The GNN $\mathcal{G}$, as the core component of the GIRL, is proposed as a parameterized function to recover the unknown reward. The conventional DNN can also be used to achieve this goal. However, the process of the B&B could be represented as an enumeration tree composed of multiple edges and nodes (e.g., the enumeration tree shown in Fig. 2). Compared to DNN, the GNN $\mathcal{G}$ is able to aggregate enough information from graphic structures of the B&B [38], and empower GIRL to easily adapt to dynamic parameters without learning from scratch.

Next, we elaborate on our proposed GNN $\mathcal{G}$ as shown in Fig. 4. In the offline training, the neural network parameters $\boldsymbol{\theta}$ are updated by MaxEnt gradients as shown in Fig. 3. In the online inference, the GIRL takes action a_t based on the given state $\mathbf{s}_t$. To indicate whether the GIRL follows the branching strategy of the FSB, the GNN $\mathcal{G}$ generates a reward $R_{\boldsymbol{\theta}}(\mathbf{s}_t,a_t)$ based on the input state action pair $(\mathbf{s}_t,a_t)$. We draw on the algorithmic structure of the B&B algorithm and take advantage of its strong graphic structure represented in Fig. 2, and the structure consists of four modules, namely, state encoding, feature embedding, graph convolution, and attention gated recurrent unit (GRU).

A. State Encoding Module

In the state encoding module, the graphic features are extracted from the enumeration tree and encoded as the state $\mathbf{s}_t = \{\mathbf{V},\mathbf{C},\mathbf{E}\}$, including variable features, constraint features, and edge features. The variable feature $\mathbf{v} \in \mathbb{R}^{d_v}$ captures the

variable information of the branching nodes and the node information except for the constraints $\boldsymbol{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ in $\boldsymbol{P}$. There are n fractional variables left in the list $\mathscr{L}_{\mathrm{v}}$, and the variable feature matrix is defined as $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_n]^T \in \mathbb{R}^{n \times d_v}$, where $(\cdot)^T$ denotes the transpose matrix operation. The objective value $\Xi^*(\mathcal{F}_t)$ is also included in $\mathbf{v}$. In addition, $\mathbf{v}$ consists of the lower bound $\Xi_{\mathrm{lb}}(\mathcal{F}_t)$ and upper bound Ξ_{ub} of $\Xi^*(\mathcal{F}_t)$. The constraint feature $\mathbf{c} \in \mathbb{R}^{d_c}$ captures the constraints $\boldsymbol{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ in $\boldsymbol{P}$. The number of the constraints is m, and the constraint feature matrix is defined as $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_m]^T \in \mathbb{R}^{m \times d_c}$. Clearly, $\mathbf{c}$ consists of the constraint normalized by the euclidean norm of the coefficients, and the number of iterations since the last time the constraint is set. The edge feature $\mathbf{e} \in \mathbb{R}^{d_e}$ indicates the connection between $\mathbf{v}$ and $\mathbf{c}$. For example, the variable feature $\mathbf{v}_j$ and one of its constraint features $\mathbf{c}_i$ has the edge feature $\mathbf{e}_{i,j}$. In our design, the edge feature $\mathbf{e}_{i,j}$ consists of its index (i, j), the dimension of which is $\mathbb{R}^1$. The edge feature matrix $\mathbf{E}$ consists of all the edge features, so we have $\mathbf{E} \in \mathbb{R}^{n \times m}$.

B. Feature Embedding Module

The feature embedding module takes $\mathbf{s}_t$ as its input and normalizes it to a fixed dimension. In the feature embedding module, the encoded state $\mathbf{s}_t = \{\mathbf{V}, \mathbf{C}, \mathbf{E}\}$ is initialized and normalized by the prenorm layer and the embedding layer, respectively. This module is a pre-training procedure to improve the model generalization capability of the proposed GNN $\mathcal{G}$, which facilitates handling a large-scale problem.

In standard convolutional neural networks, a weight initiation [39] is usually adopted to normalize the input features $\{\mathbf{V}, \mathbf{C}, \mathbf{E}\}$. The initial weight depends on the number of input features, a parameter unknown prior to the training process in our proposed GNN $\mathcal{G}$. Therefore, instead of weight initiation, we adopt a prenorm layer. In the prenorm layer, the affine transformation $p(\cdot)$ is employed to normalize $\{\mathbf{V}, \mathbf{C}, \mathbf{E}\}$. Mathematically, we have $\mathbf{x}^p = p(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu})/\sigma$, where $\mathbf{x} \in \{\mathbf{V}, \mathbf{C}, \mathbf{E}\}$, $\boldsymbol{\mu}$ is the vector of the empirical mean, and σ is the sample variance. Then the initialized features are introduced to the embedding layer to normalize their dimensions and decrease the computational complexity. Specifically, the embedding layer consists of three neural networks with $relu$ activation functions. The embedding module for $\mathbf{v}$ and $\mathbf{c}$ can be expressed as

$$\mathbf{v}^e = g^e\left(p\left(\mathbf{v}\right)\right), \ \mathbf{c}^e = g^e\left(p\left(\mathbf{c}\right)\right), \ \mathbf{e}^e = f^e\left(p\left(\mathbf{e}\right)\right), \tag{38}$$

where g^e and f^e are 2-layer and 1-layer perceptrons (neural networks) in the feature embedding module, respectively. Here, $\mathbf{e}$ is the edge feature, whose linear data structure is simple, so a 1-layer perceptron suffices to extract the features from $\mathbf{e}$. Furthermore, the 1-layer perceptron is easy to train and provides higher training efficiency compared to multi-layer perceptron. In contrast, the constraint feature $\mathbf{c}$ is non-linear and far more complex than $\mathbf{e}$. In this case, the 1-layer structure is no longer effective in feature extraction. Therefore, we resort to a more complex but more effective 2-layer perceptron to fully extract the features from the complex data structure of $\mathbf{c}$.

After the feature embedding, the dimensions d_v, d_c and d_e are normalized to be a fixed d. Therefore, in the GNN $\mathcal{G}$, regardless of the size of the enumeration trees, the same topology can be processed by the graph convolution module. The embedding module could reduce the computational complexity with negligible information loss.

C. Graph Convolution Module

The graph convolution module aggregates features of $\mathbf{s}_t$ and captures the spatial and temporal graphic information. It plays a key role in our proposed GNN $\mathcal{G}$. It has been observed that the deep GNN $\mathcal{G}$, even with just three layers, might suffer from overfitting and over-smoothing, rendering it difficult to gather the required information [40]. Specifically, the graph convolution module [41] is decomposed into two successive sub-layers, the constraint convolution layer and the variable convolution layer.

In the constraint convolution layer, the embedded features in $\{\mathbf{V}^e, \mathbf{C}^e, \mathbf{E}^e\}$ serve as the input. When MINLP problems are solved, constraints are used to determine the value of the variables. Due to the strong correlation between constraint features and variable features, we use a 2-layer perceptron g^c with $relu$ activation functions to form a joint feature $\mathbf{j}^c$ with the same dimension as the embedded constraint feature $\mathbf{c}^e$. Mathematically, we have

$$\begin{aligned} \mathbf{j}_j^c &= \sum_{i=1}^{n} g^c\left(\mathbf{v}_i^e, \mathbf{c}_j^e, \mathbf{e}_{i,j}^e\right) \in \mathbb{R}^d, \\ \mathbf{J}^c &= [\mathbf{j}_1^c, \mathbf{j}_2^c, \ldots, \mathbf{j}_m^c]^T \in \mathbb{R}^{m \times d}. \end{aligned} \tag{39}$$

Next, we combine $\mathbf{c}^e$ and $\mathbf{j}^c$, and apply a 2-layer perceptron g^c with $relu$ activation functions. We have

$$\begin{aligned} \mathbf{c}_j^c &= g^c\left(\mathbf{c}_j^e, \mathbf{j}_j^c\right) \in \mathbb{R}^d, \\ \mathbf{C}^c &= [\mathbf{c}_1^c, \mathbf{c}_2^c, \ldots, \mathbf{c}_m^c]^T \in \mathbb{R}^{m \times d}, \end{aligned} \tag{40}$$

where $\mathbf{C}^c$ is the new constraint feature matrix. In the variable convolution layer, the input is the new constraint features in $\mathbf{C}^c$ and the embedded features in $\{\mathbf{V}^e, \mathbf{E}^e\}$. To capture the potential node features, we combine features in $\{\mathbf{V}^e, \mathbf{C}^e, \mathbf{E}^c\}$, and rearrange them based on the order of variables and then transfer them to the neural network. Then, we have

$$\begin{aligned} \mathbf{v}_i^v &= \sum_{j=1}^{m} g^v\left(\mathbf{v}_i^e, \mathbf{c}_j^c, \mathbf{e}_{i,j}^e\right) \in \mathbb{R}^d, \\ \mathbf{V}^v &= [\mathbf{v}_1^v, \mathbf{v}_2^v, \ldots, \mathbf{v}_n^v]^T \in \mathbb{R}^{n \times d}, \end{aligned} \tag{41}$$

where g^v is a 2-layer perceptron in the variable convolution layer. Regardless of their features, $\mathbf{C}^e$ and $\mathbf{C}^c$ are arranged following the order of the original constraint features in $\mathbf{C}$. In order to strengthen the link between variable features and constraint features, we rearrange the constraint features following the order of the variable features, and the new constraint feature $\mathbf{c}_i^v$ is related to all the constraint features connecting to $\mathbf{v}_i$. We have

$$\begin{aligned} \mathbf{c}_i^v &= \sum_{j=j_a^i}^{j_b^i} f^v\left(\mathbf{c}_j^e\right) \in \mathbb{R}^d, \\ \mathbf{C}^v &= [\mathbf{c}_1^v, \mathbf{c}_2^v, \ldots, \mathbf{c}_n^v]^T \in \mathbb{R}^{n \times d}, \end{aligned} \tag{42}$$

where j_a^i and j_b^i is the index of the constraint features connecting to $\mathbf{v}_i$. For example, in the state encoding of Fig. 4, $\mathbf{v}_1$ connects to $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_3$, so j_a^i and j_b^i are 1 and 3, respectively. The outputs of the variable convolution layer are $\mathbf{V}^v$ and $\mathbf{C}^v$, which have the potential node features, and each feature consists of information for its neighbors.

D. Attention GRU Module

The attention GRU module applies an offset to the convolution module and converts its output features to the temporal-spatial-frequency domain, and obtains the output reward function $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$. Specifically, the convolution module is able to integrate different types of relevant information (e.g., node information from $\mathbf{v}$ and $\mathbf{c}$ and edge information from $\mathbf{e}$) via the node-edge-node relationship in the state encoding module of Fig. 4. The three different inputs, $\mathbf{V}$, $\mathbf{C}$ and $\mathbf{E}$, are integrated with equal weights before being processed by g^c and g^v. Note that the condition feature in $\mathbf{v}$ directly determines the variable selection in the conventional B&B algorithm. Consequently, $\mathbf{V}$ plays a more significant role, and a more appropriate weight should be allocated. Therefore, there should be a more sensible way to train a set of proper weights for various types of input features.

1) Mix State: The mix state is based on some matrix operation on $\mathbf{V}^e$, $\mathbf{V}^v$, and $\mathbf{C}^v$. Clearly, $\mathbf{V}^e$ represents the embedded original information of $\mathbf{V}$. In addition, $\mathbf{V}^v$ includes the combined information of $\{\mathbf{V}, \mathbf{C}, \mathbf{E}\}$, and $\mathbf{C}^v$ consists of the constraint information of each variable node. We have the three matrices $\mathbf{V}^e$ $\mathbf{V}^v$ and $\mathbf{C}^v$, where the order of vectors is consistent with the order of variable features in the bipartite graph of the state encoding module as shown in Fig. 4. Specifically, we extract the vectors of these three matrices with the same index k to form a mix state matrix $\mathbf{M}_k = [\mathbf{v}_k^e, \mathbf{v}_k^v, \mathbf{c}_k^v]^T$. Here, $\mathbf{v}_k^e$, $\mathbf{v}_k^v$, and $\mathbf{c}_k^v$ can be regarded as vectors of three new nodes, so we express the t-th row vector in $\mathbf{M}_k$ as a node vector $\mathbf{n}_{k,t} = \mathbf{M}_k[t] \in \mathbb{R}^d$. This mix state matrix can be expressed as a new node-edge graph shown in the attention GRU module of Fig. 4. In comparison to the bipartite graph in the state encoding module, this graph has a fixed number of nodes and edges which facilitates the implementation of the ensuing attention mechanism.

2) Attention Layer: The attentioned information of the node in the new graph is the sum of edge-weighted information of its neighbor node vectors, which is expressed as an attentional vector An attentioned vector [42], aggregating edge-weighted information of one node and its neighbor nodes, is the output of the attention layer and can be expressed as $\mathbf{A}_k = \sum_{\mathbf{n}_{k,j} \in \mathbf{M}_k} w_{\mathbf{n}_{k,j} \to \mathbf{n}_{k,i}} \cdot \mathbf{n}_{k,j}$, where $\mathbf{n}_{k,i}$ is the target node vector to update, while $\mathbf{n}_{k,j}$ is one of its neighbor node vectors (we write this neighboring relationship as $\mathbf{n}_{k,j} \to \mathbf{n}_{k,i}$). Furthermore, $w_{\mathbf{n}_{k,i} \to \mathbf{n}_{k,j}}$ is the edge weight from $\mathbf{n}_{k,j}$ to $\mathbf{n}_{k,i}$, and it is calculated as

$$\begin{aligned} w_{\mathbf{n}_{k,j} \to \mathbf{n}_{k,i}} &= \sigma\left(g^a\left([\mathbf{n}_{k,i}; \mathbf{n}_{k,j}]^T \cdot \mathbf{W}_w\right)\right) \\ &= \frac{e^{g^a\left([\mathbf{n}_{k,i}; \mathbf{n}_{k,j}]^T \cdot \mathbf{W}_w\right)}}{\sum_{l=1}^{3} e^{g^a\left([\mathbf{n}_{k,i}; \mathbf{n}_{k,l}]^T \cdot \mathbf{W}_w\right)}}, \end{aligned} \tag{43}$$

where ; represents the operation of concatenation, and $\mathbf{W}_w \in \mathbb{R}^{2d}$ is the weight matrix. Besides, a 2-layer perceptron with $LeakyRelu$ activation function g^a and softmax function $\sigma(\cdot)$ is applied to realize a rational weights distribution by comparing one neighbor node to all its neighbor nodes.

3) GRU Layer: In the GRU layer, the attentioned vector $\mathbf{a}_k$ will be updated by a 1-layer GRU layer, which is based on the previous work [43]. Two inputs of the GRU layer include a target node vector $\mathbf{v}_i^e$ and a vector $\mathbf{a}_i$ containing aggregated neighbor information. Thus, the output of the GRU layer, known as the gated variable feature vector, can be shown as $\mathbf{v}_i^g = \mathrm{GRU}(\mathbf{v}_i^e, \mathbf{A}_i)$. The detailed update process of the GRU layer obeys the following formulas

$$\begin{aligned} \mathbf{z}_i^g &= S(\mathbf{W}_z \mathbf{a}_i + \mathbf{U}_z \mathbf{v}_i^e + \mathbf{b}_z), \\ \mathbf{r}_i^g &= S(\mathbf{W}_r \mathbf{a}_i + \mathbf{U}_r \mathbf{v}_i^e + \mathbf{b}_r), \\ \bar{\mathbf{v}}_i^g &= \tanh(\mathbf{W}_v \mathbf{a}_i + \mathbf{U}_v(\mathbf{r}_i^g \odot \mathbf{v}_i^e) + \mathbf{b}_v), \\ \mathbf{v}_i^g &= \mathbf{v}_i^e \odot (1 - \mathbf{z}_i^g) + \bar{\mathbf{v}}_i^g \odot \mathbf{z}_i^g, \end{aligned} \tag{44}$$

where $\mathbf{W}_z$, $\mathbf{W}_r$ and $\mathbf{W}_v$ are the GRU weights, $\mathbf{b}_z$, $\mathbf{b}_r$ and $\mathbf{b}_v$ are the GRU bias, and $\mathbf{z}_i^g$ and $\mathbf{r}_i^g$ are updated and reset gates. Furthermore, $S(\cdot)$ is the logistic sigmoid function, and $\odot$ is the Schur product operation. Finally, we apply a 2-layer perception with $relu$ activation functions to the features in $\mathbf{V}^g$, and we obtain $\mathbf{V}^f$. A softmax function is employed to obtain the reward

$$R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_i) = \sigma\left(v_i^f\right) = \frac{e^{v_i^f}}{\sum_{i=1}^{n} e^{v_i^f}},\ i \in [1, n], \tag{45}$$

where a_i represents that variable α_i is selected to branch. Note that a_t is to select a variable to branch from $\mathscr{L}_{\mathrm{v}}$ that includes n variables, thus we have $a_t \in \{a_1, \ldots a_i, \ldots a_n\}$. Accordingly, with the input $(\mathbf{s}_t, a_t)$, the GNN $\mathcal{G}$ would generate a reward $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t) \in [0, 1]$. Note that the higher value of $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$, the larger probability that the GIRL and the B&B with FSB can select the same variable.

In summary, regardless of the input graph size, enumeration trees are encoded into the graph-structured features $\{\mathbf{V}, \mathbf{C}, \mathbf{E}\}$ in a unified way. It is clear that the operations (38)–(45) do not rely on the size of the input graph. Consequently, the output of the GNN $\mathcal{G}$ exhibits permuted equivariance, which empowers the GIRL to easily adapt to the variations of the system parameters without learning from scratch. Besides, the computational complexity is only related to the graph density, which makes the GNN $\mathcal{G}$ as an ideal choice for processing graph-structured data of the B&B algorithm.

VI. Model Generalization and Performance Analysis

A. Model Generalization

In practical multi-layer computing networks, parameters such as the number of MDs and required CPU cycles can change dynamically over time. As a key advantage, our GIRL, although trained offline in a specific parameter setting, can be directly applied to a new scenario with dynamic parameters without learning from scratch due to the designed GNN $\mathcal{G}$ and incorporated self-imitation.

Under the GNN $\mathcal{G}$, various enumeration trees with different network parameters are encoded into the graph-structured features $\{\mathbf{V}, \mathbf{C}, \mathbf{E}\}$ in a unified way. Then, the features could be rearranged to a fixed dimension d, resulting in the same topology to be processed by the GNN $\mathcal{G}$. Consequently, the output of the GNN $\mathcal{G}$ is guaranteed to exhibit permuted equivariance and is independent of different parameters.

Self-imitation in the offline training of the GIRL collects some additional problems as the self-imitation demonstration $\mathcal{D}'$, which are not in the original expert demonstration $\mathcal{D} = \{\mathcal{D}_\varsigma \mid \varsigma \in \mathcal{S}\}$. Because $\mathcal{D}'$ has not been learned by the previous policy π_{old}, a better policy π_{new} can be obtained based on the new demonstration $\mathcal{D} \cup \mathcal{D}'$. In other words, the generalization performance of the GIRL is improved iteratively by repeating this process. For our GIRL, extensive simulations have been conducted, revealing that a mere three self-imitation iterations yield remarkable model generalization.

B. Performance Optimization

To further optimize the GIRL variable selection policy and accelerate the training process, we aim to evaluate the impact of several dominant factors on performance. The training performance of the GIRL depends on the speed of convergence, or the variation $\Delta\boldsymbol{\theta}$ in (37). The speed of convergence is affected by two major factors, the learning rate, and the expert demonstration.

1) Learning Rate: In the update process of $\boldsymbol{\theta}$ in (37), the learning rate λ_θ determines how much the newly acquired information $\Delta\boldsymbol{\theta}$ covers the current state of $\boldsymbol{\theta}$. For a large λ_θ, $\boldsymbol{\theta}$ is likely to converge without a performance guarantee. On the contrary, a small λ_θ would extend the training process and increase the computational complexity. Instead of a fixed learning rate setting, we adopt an adaptive learning rate approach referred to as adaptive moment estimation (ADAM) [44] in our GIRL. ADAM aims to optimize the first-order gradient-based stochastic objective functions, based on adaptive estimates of lower-order moments. To evaluate the impact of the learning rate in the update process, we implement three different update schemes in Section VII-C, including standard gradient ascent (SGA) with large λ_θ, SGA with small λ_θ, and ADAM. Specifically, the SGA could be expressed as $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \lambda_\theta \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$.

2) Expert Demonstration: In Section IV, our proposed GIRL aims to obtain a policy with the same behavior as the observed expert demonstration $\mathcal{D}$. In our problem, the expert demonstration $\mathcal{D}$ is composed of the optimal trajectories obtained in advance by solving many MINLP problem instances via the B&B algorithm with the FSB, instead of generating the data progressively. The collected $\mathcal{D}$ would be implemented the expert policy π_E by calculating the expectation of state-action visitation counts $\mu_{\mathcal{D}}(\mathbf{s}_t, a_t)$ over $\mathcal{D}$. Therefore, $\mathcal{D}$ plays a significant role in the learning process and should be carefully generated.

Note that the increasing number of training samples leads to enhanced training performance and therefore reduces complexity. In this paper, to ensure that $\mathcal{D}$ is both substantial and falls within an acceptable range, we collect 100,000 training samples as $\mathcal{D}$ by solving 20,000 problem instances. This guarantees a robust representation of diverse scenarios while maintaining a manageable scale for effective training. To enhance the model generalizability of the GIRL, we collect an additional 10,000 samples as the self-imitation demonstration $\mathcal{D}'$ by solving 2,000 problem instances in each self-imitation iteration. Furthermore, if noise is mingled into $\mathcal{D}$, the term $\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)}$ is easy to diverge. Therefore, the collected $\mathcal{D}$ would be checked before being implemented to train the GIRL.

C. Computational Complexity Analysis

In this section, we analyze the computational complexity for several variable selection policies, namely, the FSB, the RPB and the GIRL. In general, the complexity of each policy mainly consists of two parts: the variable selection process and the node solving process.

For the FSB, in the variable selection process, if there are K fractional variables to branch, there are $2K$ NLPs to be solved, each with the complexity $\mathcal{O}(n^3)$ (n is the number of discrete variables). As such, the complexity for variable selection process is $\mathcal{C}_{\text{vs}}^{\text{FSB}} = \mathcal{O}(2n^3 K N_{\text{v}})$, where N_{v} is the number of times to select variables. It is also straightforward to see that, for the node solving process with N_{n} nodes in the enumeration tree, the complexity is $\mathcal{C}_{\text{ns}}^{\text{FSB}} = \mathcal{O}(n^3 N_{\text{n}})$. The RPB can also obtain an optimal solution with lower complexity than the FSB. The variable selection of the RPB features a lower complexity than the FSB ($\mathcal{C}_{\text{vs}}^{\text{RPB}} < \mathcal{C}_{\text{vs}}^{\text{FSB}}$). However, its enumeration tree is much larger than that of FSB, leading to higher complexity in the node solving process ($\mathcal{C}_{\text{ns}}^{\text{RPB}} > \mathcal{C}_{\text{ns}}^{\text{FSB}}$). Overall, we have $\mathcal{C}_{\text{vs}}^{\text{RPB}} + \mathcal{C}_{\text{ns}}^{\text{RPB}} < \mathcal{C}_{\text{vs}}^{\text{FSB}} + \mathcal{C}_{\text{ns}}^{\text{FSB}}$.

For the GIRL, its complexity lies in offline training and online inference. Offline training is a one-off process and its complexity can be ignored from the overall perspective. After the offline training, we directly apply the optimal policy to select variables for a new MINLP problem instance in the online inference, and the complexity is only $\mathcal{O}(n)$. As such, the computational complexity is $\mathcal{C}_{\text{vs}}^{\text{GIRL}} = \mathcal{O}(nN_{\text{v}})$ for the whole variable selection process. Clearly, we have $\mathcal{C}_{\text{vs}}^{\text{GIRL}} \ll \mathcal{C}_{\text{vs}}^{\text{FSB}}$. Ideally, if the offline training is performed with a sufficient number of training samples, the GIRL would generate the same enumeration tree as the FSB ($\mathcal{C}_{\text{ns}}^{\text{GIRL}} = \mathcal{C}_{\text{ns}}^{\text{FSB}}$). Overall, the complexity of the GIRL is much lower than FSB ($\mathcal{C}_{\text{vs}}^{\text{GIRL}} + \mathcal{C}_{\text{ns}}^{\text{GIRL}} \ll \mathcal{C}_{\text{vs}}^{\text{FSB}} + \mathcal{C}_{\text{ns}}^{\text{FSB}}$). In Section VII, the complexity of these three policies will be numerically compared.

VII. Simulation Results

A. Experimental Setting

In Figs. 5–8 and Figs. 10–12, we consider a task offloading scenario with 20 MDs and 5 MSs. The distance DIS_1 between MD and its associated MS is sampled from the uniform distribution $\mathcal{U}(0.1, 0.5)$ (in kilometers). We assume that each MD has one D2D link with a peer MD, and the distance DIS_2 between MDs is sampled from the distribution $\mathcal{U}(0.01, 0.05)$ (in kilometers). The path-loss model between MDs and MSs follows $g = 128.1 + 37.6 \log(\text{DIS}_1)$ (DIS_1 in kilometers), and the path-loss model for the D2D link follows $g = 148.1 + 40 \log(\text{DIS}_2)$ (DIS_2 in kilometers) [45]. For task Q_i, its data size D_i is

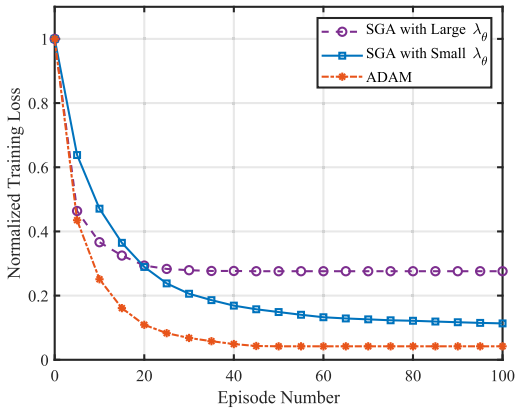


Fig. 5. Loss and convergence speed comparison for different λ_θ settings.

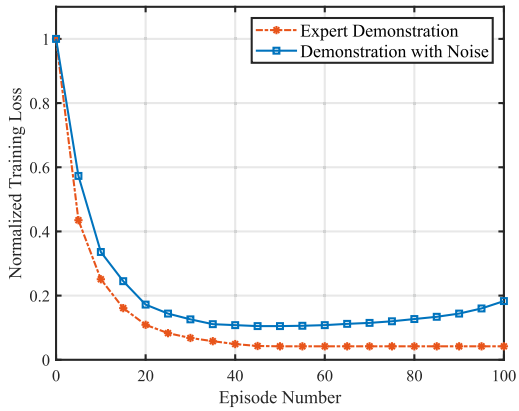


Fig. 6. Loss and convergence speed comparison for different demonstrations.

TABLE II
MEC SYSTEM PARAMETERS

Local transmission power p_i^{tr}	23 dBm
Local computation power p_i^{L}	0.5 W
Local computation capability f_i^{L}	0.6 GHz
Local energy capability $e_i^{\max}$	3 J
MS Computation capability $f_k^{\max}$	1 GHz
MS Bandwidth capacity $B_k^{\max}$	1 MHz
Round-trip transmission time d_{l,l_i}^{rrt}	0.5 s
Cloud computation capability $f_{i,c}$	20 GHz
Backhaul time d^{BC}	5 s
Noise power spectrum density σ^2	-174 dBm/Hz

sampled uniformly from $\mathcal{U}(0.8, 1.2)$ (in Mb) and the number of required computation resources C_i is sampled uniformly as $\mathcal{U}(2, 5)$ (in Gigacycles). Besides, the other related parameters are summarized in Table II.

B. Policies for Comparison and Performance Metrics

From the algorithm perspective, to illustrate the significance of adopting the variable selection policy in the B&B algorithm, we compare our proposed GIRL with the FSB and the RPB introduced in Section III-B. A heuristic policy with random variable selection is also included to provide baseline performance for comparison. According to Remark 1, all the above variable selection policies can eventually reach the optimal node and obtain an optimal solution to the MINLP problem.

We first adopt accuracy as one of the algorithm performance metrics, which quantifies the similarity between other variable selection policies and the FSB in terms of the variable selection order. This will have an impact on the computational complexity rather than the optimality. The accuracy includes three representative indicators: top-one accuracy (acc@1), top-five accuracy (acc@5) and top-ten accuracy (acc@10). Generally speaking, the top-x accuracy is the percentage of the top-x ($x \in \{\text{one, five, ten}\}$) variables selected by other policies among all the unbranched variables. Specifically, for a given node, all unbranched variables are marked by the FSB. If a policy selects a variable (e.g., α_1) and its FSB score is the highest one among the variables, this variable will be marked as the top-one variable. If a policy selects a variable and its FSB score is one of the highest five among the unbranched variables, this variable will be marked as the top-five variable. To further evaluate the computational complexity, the number of solved NLPs is adopted as another metric. In order to control the training and testing time, we set a number limitation (2000), beyond which the process of the B& B algorithm will be terminated.

From the network perspective, we compare our proposed full-dimensional task offloading scheme with the non-cooperation offloading scheme, the vertical offloading scheme, and the two-dimensional offloading scheme. For the non-cooperation scheme, the task Q_i of the MD i can only be offloaded to its associated MS l_i, and the computation resources $\mathbf{f}$ and bandwidth $\mathbf{b}$ are optimized with $x_{i,i} + y_{i,l_i} = 1$ for all $i \in \mathcal{N}$. For the vertical offloading scheme, Q_i can be offloaded to its associated MS l_i and CS. Here, $\mathbf{f}$ and $\mathbf{b}$ are optimized with $x_{i,i} + y_{i,l_i} + z_i = 1$ for all $i \in \mathcal{N}$. Furthermore, for the two-dimensional offloading scheme, Q_i not only can be offloaded vertically among MD i, MS j and CS but offloaded horizontally from MS j to MS $k \in \hat{\mathcal{M}}_i$, thus, we have $x_{i,i} + \sum_{k \in \mathcal{M}} y_{i,k} + z_i = 1$.

C. Simulation Results

1) Training Performance of the GIRL Variable Selection Policy: We first demonstrate the performance of the GIRL variable selection policy during the training process in our full-dimensional task offloading scheme. The performance is evaluated by the normalized loss. In Fig. 5, we compare the loss and convergence speed of the GIRL for three learning rate settings (e.g., SGA with large λ_θ (0.2), SGA with small λ_θ (0.001), and ADAM introduced in Section VI-B1). We can observe that the SGA with the larger learning rate (0.2) leads to much faster convergence and a larger loss relative to the SGA with the smaller learning rate (0.001). It can also be seen that ADAM performs best in terms of the loss, and its convergence speed is only slightly inferior to the learning rate 0.2 at the beginning. This demonstrates that ADAM achieves an excellent training performance. A similar comparison is carried out in Fig. 6 for the expert and imperfect demonstrations. Note that the imperfect demonstration is mingled with some artificial data (noise). It is observed that the loss of the imperfect demonstration

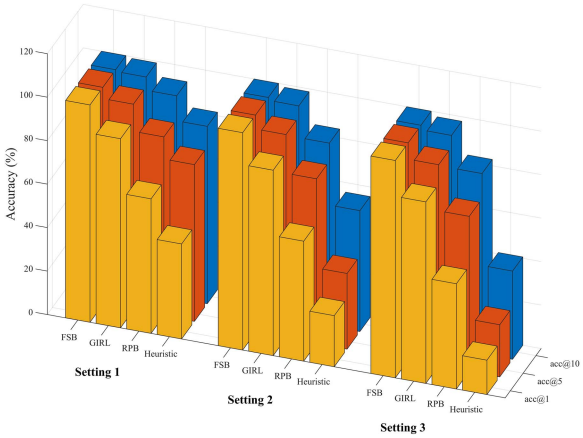


Fig. 7. Accuracy for different variable selection policies.

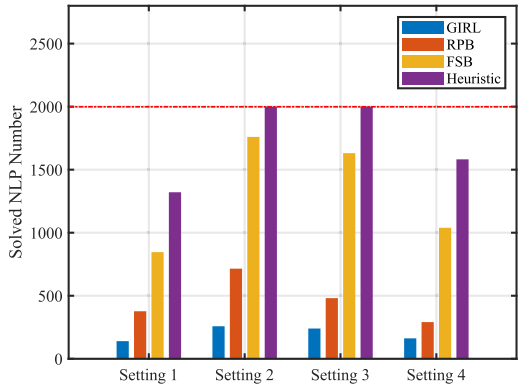


Fig. 8. Complexity comparison of the four variable selection policies in terms of searched NLP number.

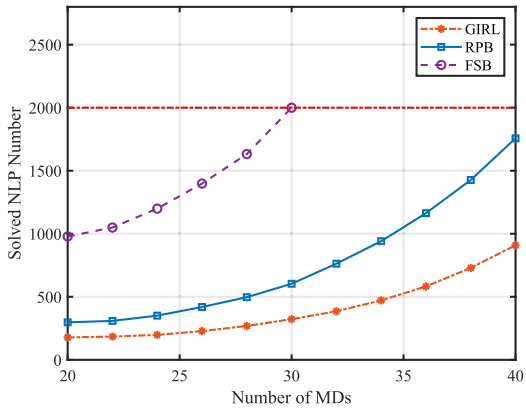


Fig. 9. Average solved NLP numbers with different numbers of MDs.

does not converge. In contrast, convergence is achieved for the expert demonstration, and its loss is always lower than that of the imperfect demonstration. This verifies the significant role that the demonstration plays in the learning process as discussed in Section VI-B2.

2) Optimality Comparison: In Fig. 7, we compare the accuracy performance of the GIRL, the FSB, the RPB, and the heuristic policy. There are 40 online testing problem instances, and 3 out of 40 testing problem instances are randomly selected and presented in the figure as Setting 1-3. Note that the accuracy for the FSB is always 100%, as all the variable selection policies are to imitate the behavior of the FSB. It can be observed that the GIRL has higher acc@1, acc@5 and acc@10 than the RPB and the heuristic policy. Specifically, it achieves the accuracy of 85.2%, 93.6% and 98.6% after averaging across the three settings for acc@1, acc@5 and acc@10, respectively. For the RPB, on average, the acc@10 approaches 90%, while acc@1 is about 50%. This clearly verifies the superiority of our proposed GIRL over the RPB in imitating the performance of the FSB. It is also clear that, for the heuristic policy, the acc@1 fluctuates significantly across the three settings (15.8%, 23.5% and 43.7%), so it is with acc@5 and acc@10. Clearly, Fig. 7 leads to the conclusion that the GIRL best imitates the behavior of the FSB.

3) Complexity Comparison: The computational complexity comparison is carried out for the GIRL, the FSB, the RPB and the heuristic policies in Fig. 8 in terms of the number of the solved NLPs. Among 40 online testing problem instances, 4 are randomly selected and presented in both figures as Setting 1-4. In Fig. 8, it is observed that the GIRL significantly reduces the solved NLP number compared with the FSB across all four settings. A simple calculation finds that such a reduction stands at 84.8% on average. Compared with the RPB, the GIRL achieves a reduction of approximately 54.2% on average. Finally, it is clear that the heuristic policy has the highest complexity, and the number limitation is even triggered for Setting 2 and Setting 3.

4) Model Generalization Capability of the GIRL Variable Selection Policy: Having considered the scenario with a fixed value for N, next we compare the model generalization capability of these policies (c.f. Section VI-A). Here, the GIRL is offline trained with an original expert demonstration and three self-imitation demonstrations, and then directly infers the variable selection to solve the formulated problems with different settings. By contrast, RPB and FSB need to solve both the variable selection order and the sub-problems at each setting. Fig. 9 considers the original setting ($N = 20$) transferring to different MD numbers. Across the whole range of N, the GIRL has the lower average solved NLP number and the FSB has the highest average solved NLP number, consistent with the previous observation in Fig. 8. In Fig. 7, Setting 1 is the accuracy of Setting $N = 20$ in Fig. 9. It is observed from Figs. 7 and 9 that the GIRL has higher top-one accuracy than the RPB, and the GIRL achieves a low complexity than the RPB. Note that the NLP number limitation has reached for the FSB ($N = 30$). It is also observed that the average solved NLP number gap between the RPB and the GIRL significantly increases with N. In summary, the GIRL could directly infer the solutions to the problems with different settings without learning from scratch, which confirms the timeliness and effectiveness of the GIRL.

5) Average Delay Comparison: In Figs. 10–12, we compare our proposed full-dimensional task offloading scheme with the non-cooperation offloading scheme, the vertical offloading

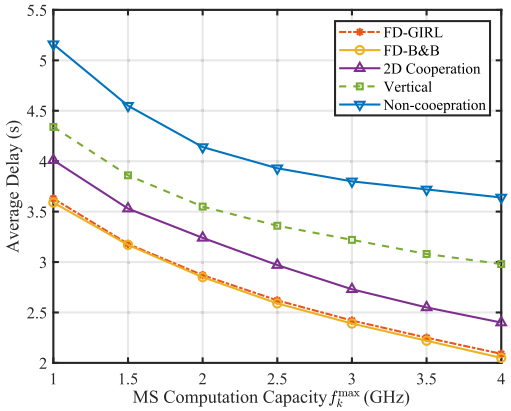


Fig. 10. Average delay performance comparison versus different $f_k^{\max}$.

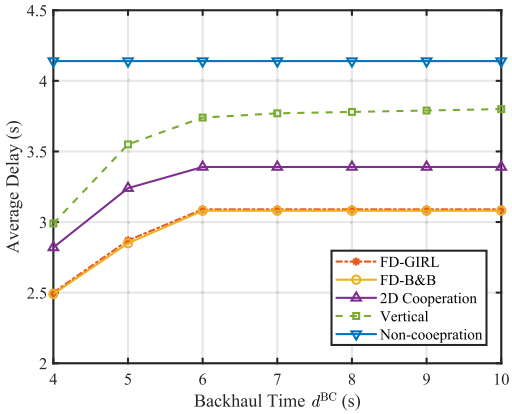


Fig. 11. Average delay performance comparison versus different d^{BC}.

scheme, and the two-dimensional offloading scheme. The full-dimensional offloading schemes are achieved by either the B&B algorithm with the FSB or the accelerated B&B with the GIRL, which are denoted as FD-B&B and FD-GIRL, respectively.

In Fig. 10, we compare the variation of average delay with the MS computation capability $f_k^{\max}$. As expected, it is observed that the average delay decreases as $f_k^{\max}$ increases, due to the increased total computation capacity in the computing networks. Clearly, FD-B&B and FD-GIRL have similar performance, consistent with the observation in Fig. 7, and both of them achieve the shortest average delay among different offloading schemes. In particular, when $f_k^{\max} = 4$ GHz, improvements of the FD-GIRL are about 42.6%, 29.8% and 12.9% compared with the non-cooperation offloading, the vertical offloading, and the two-dimensional offloading schemes, respectively.

In Fig. 11, we compare the average delay versus the backhaul time d^{BC}. Except for the non-cooperation offloading scheme, the average delay of the other three offloading schemes increases with d^{BC} increases. It is clear that FD-B&B and FD-GIRL achieve the shortest average delay. In particular, when $d^{\mathrm{BC}} = 10$ s, improvements of the FD-B&B are about 25.4%, 18.7% and 8.9% compared with non-cooperation offloading, the vertical offloading, and the two-dimensional offloading schemes, respectively.

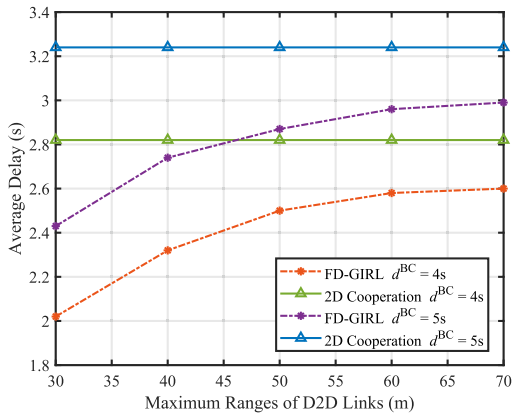


Fig. 12. Average delay performance comparison versus different maximum ranges of D2D links.

In Fig. 12, we compare the average delay versus the different maximum ranges of D2D links for the FD-GIRL and the two-dimensional offloading schemes. Clearly, it is observed that FD-GIRL achieves a lower average delay than the two-dimensional offloading scheme. When the maximum ranges of D2D links increase, the average delay increases, the reason for which is MDs have a lower potential to offload tasks with the increased ranges of D2D links. It is also observed that a lower average delay can be achieved with a lower backhaul time d^{BC} consistent with the observation in Fig. 11.

Overall, the simulations in Figs. 10–12 illustrate that our proposed full-dimensional offloading scheme can significantly improve the average delay performance by jointly optimizing task offloading decisions and communication/computation resources. In addition, it is verified that our GIRL could effectively solve the formulated problems with different dynamics.

6) Discussions: In practice, our full-dimensional task offloading scheme in a multi-layer computing network involves assigning computing, storage, and communication resources to the different layers of the network to improve its efficiency and performance. The local layer comprises the devices, such as smartphones and tablets, that are close to the end users. The edge layer consists of small data centers, usually located at the edge of the network, that provide intermediate processing and storage capabilities. Finally, the cloud layer consists of large-scale data centers that offer massive processing and storage capabilities. There are several challenges in implementing our scheme, including determining traffic patterns [46], determining the resource requirements [47], and monitoring resource allocation [48]. By implementing task offloading and resource allocation, the efficiency and performance of the network can be improved, and end-users can enjoy a better user experience.

VIII. Conclusion

In this paper, we incorporated D2D communication into the multi-layer computing network and proposed a full-dimensional task offloading scheme by jointly optimizing task offloading decisions and communication/computation resources. We formulated it as an MINLP problem to minimize the average

delay for all the MDs and proposed the GIRL that offers a global optimal solution to the MINLP problem by generating a new variable selection policy to accelerate the optimal B&B algorithm with significantly reduced complexity but without sacrificing the global optimality. It has been verified by simulations that the GIRL achieves a significantly lower computational complexity compared to the existing variable selection policies. Furthermore, the superiority of our proposed full-dimensional task offloading scheme over the existing schemes has been verified by simulation.

REFERENCES

[1] G. Wang, P. Cheng, Z. Chen, W. Xiang, B. Vucetic, and Y. Li, "Inverse reinforcement learning with graph neural networks for iot resource allocation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2023, pp. 1–5.

[2] J. A. Ansere, G. Han, L. Liu, Y. Peng, and M. Kamal, "Optimal resource allocation in energy-efficient internet-of-things networks with imperfect CSI," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5401–5411, Jun. 2020.

[3] A. Kamilaris and A. Pitsillides, "Mobile phone computing and the Internet of Things: A survey," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 885–898, Dec. 2016.

[4] A. A. Adebayo, D. B. Rawat, and M. Song, "Energy-efficient multivariate privacy-aware RF spectrum reservation in wireless virtualization for wireless Internet of Things," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 2, pp. 682–692, Jun. 2021.

[5] C.-H. Hong and B. Varghese, "Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms," *ACM Comput. Surv.*, vol. 52, no. 5, pp. 1–37, Sep. 2019.

[6] Z. Xu, W. Gong, Q. Xia, W. Liang, O. F. Rana, and G. Wu, "NFV-enabled IoT service provisioning in mobile edge clouds," *IEEE Trans. Mobile Comput.*, vol. 20, no. 5, pp. 1892–1906, May 2021.

[7] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[8] M.-H. Chen, B. Liang, and M. Dong, "Multi-user multi-task offloading and resource allocation in mobile cloud systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 10, pp. 6790–6805, Oct. 2018.

[9] K. Guo, M. Sheng, J. Tang, T. Q. S. Quek, and Z. Qiu, "Hierarchical offloading for delay-constrained applications in fog RAN," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4257–4270, Apr. 2020.

[10] A. Naouri, H. Wu, N. A. Nouri, S. Dhelim, and H. Ning, "A novel framework for mobile-edge computing by optimizing task offloading," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 13065–13076, Aug. 2021.

[11] Y. Wang, X. Tao, X. Zhang, P. Zhang, and Y. T. Hou, "Cooperative task offloading in three-tier mobile computing networks: An ADMM framework," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2763–2776, Mar. 2019.

[12] Z. Yan, P. Cheng, Z. Chen, B. Vucetic, and Y. Li, "Two-dimensional task offloading for mobile networks: An imitation learning framework," *IEEE/ACM Trans. Netw.*, vol. 29, no. 6, pp. 2494–2507, Dec. 2021.

[13] C. Suraci, S. Pizzi, A. Molinaro, and G. Araniti, "MEC and D2D as enabling technologies for a secure and lightweight 6G eHealth system," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11524–11532, Jul. 2022.

[14] R. I. Ansari et al., "5G D2D networks: Techniques, challenges, and future prospects," *IEEE Syst. J.*, vol. 12, no. 4, pp. 3970–3984, Dec. 2018.

[15] C. V. Anamuro, N. Varsier, J. Schwoerer, and X. Lagrange, "Distance-aware relay selection in an energy-efficient discovery protocol for 5G D2D communication," *IEEE Trans. Wireless Commun.*, vol. 20, no. 7, pp. 4379–4391, Jul. 2021.

[16] A. H. Land and A. G. Doig, *An Automatic Method for Solving Discrete Programming Problems*, Berlin, Germany: Springer, 2010, pp. 105–132.

[17] X. Li, L. Huang, H. Wang, S. Bi, and Y.-J. A. Zhang, "An integrated optimization-learning framework for online combinatorial computation offloading in mec networks," *IEEE Wireless Commun.*, vol. 29, no. 1, pp. 170–177, Feb. 2022.

[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[19] T. Achterberg, T. Koch, and A. Martin, "Branching rules revisited," *Operations Res. Lett.*, vol. 33, no. 1, pp. 42–54, Jan. 2005.

[20] 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.

[21] T. Fang, F. Yuan, L. Ao, and J. Chen, "Joint task offloading, D2D pairing, and resource allocation in device-enhanced MEC: A potential game approach," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3226–3237, Mar. 2022.

[22] M. Sun, X. Xu, X. Tao, and P. Zhang, "Large-scale user-assisted multi-task online offloading for latency reduction in D2D-enabled heterogeneous networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2456–2467, Fourth Quarter 2020.

[23] M. Feng, M. Krunz, and W. Zhang, "Task partitioning and user association for latency minimization in mobile edge computing networks," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2021, pp. 1–6.

[24] Z. Zhou, Q. Wu, and X. Chen, "Online orchestration of cross-edge service function chaining for cost-efficient edge computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1866–1880, Aug. 2019.

[25] R. Yu, G. Xue, Y. Wan, J. Tang, D. Yang, and Y. Ji, "Robust resource provisioning in time-varying edge networks," in *Proc. 21st Int. Symp. Theory, Algorithmic Found., Protoc. Des. Mobile Netw. Mobile Comput.*, New York, NY, USA, 2020, pp. 21–30.

[26] F. Jiang, K. Wang, L. Dong, C. Pan, W. Xu, and K. Yang, "Deep-learning-based joint resource scheduling algorithms for hybrid MEC networks," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6252–6265, Jul. 2020.

[27] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.

[28] F. Jiang, K. Wang, L. Dong, C. Pan, and K. Yang, "Stacked autoencoder-based deep reinforcement learning for online resource scheduling in large-scale MEC networks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9278–9290, Oct. 2020.

[29] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 11, Aug. 2018, Art. no. e3493.

[30] W. Xiang, J. Li, Y. Zhou, P. Cheng, J. Jin, and K. Yu, "Digital twin empowered industrial IoT based on credibility-weighted swarm learning," *IEEE Trans. Ind. Informat.*, early access, Apr. 04, 2023, doi: 10.1109/TII.2023.3264289.

[31] D. Castanheira and A. Gameiro, "Low complexity and high-resolution line spectral estimation using cyclic minimization," *IEEE Trans. Signal Process*, vol. 67, no. 24, pp. 6285–6300, Dec. 2019.

[32] S. A. Lippman, "Dynamic programming and markov decision processes," in *The New Palgrave Dictionary of Economics*, London, U.K.: Palgrave Macmillan, 1987.

[33] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Comput. Surv.*, vol. 50, no. 2, pp. 1–35, Apr. 2017. [Online]. Available: https://doi.org/10.1145/3054912

[34] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[35] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-learning algorithms: A comprehensive classification and applications," *IEEE Access*, vol. 7, pp. 133653–133667, 2019.

[36] M. Wulfmeier, P. Ondruska, and I. Posner, "Maximum entropy deep inverse reinforcement learning," Jul. 2015, *arXiv:1507.04888*.

[37] E. T. Jaynes, "Information theory and statistical mechanics," *Phys. Rev.*, vol. 106, no. 4, May 1957, Art. no. 620.

[38] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[39] G. Hinton et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[40] H. Cheng, J. T. Zhou, W. P. Tay, and B. Wen, "Graph neural networks with triple attention for few-shot learning," *IEEE Trans. Multimedia*, Jan. 02, 2023, doi: 10.1109/TMM.2022.3233442.

[41] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

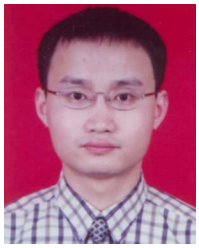
[42] X. Wang et al., "Heterogeneous graph attention network," in *Proc. World Wide Web Conf.*, New York, NY, USA, May 2019, pp. 2022–2032.

[43] L. Ruiz, F. Gama, and A. Ribeiro, "Gated graph recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 68, pp. 6303–6318, 2020.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Jan. 2014, *arXiv:1412.6980*.
[45] R. Zhang, Y. Li, Y. Ruan, H. Zhang, W. Wang, and W. Wang, "CQI-based interference management scheme for D2D communication underlaying cellular networks," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops*, 2015, pp. 347–351.
[46] Y.-H. Liu and K. C.-J. Lin, "Traffic-aware resource allocation for multi-user beamforming," *IEEE Trans. Mobile Comput.*, vol. 22, no. 6, pp. 3677–3690, Jun. 2023.
[47] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Edge intelligence for energy-efficient computation offloading and resource allocation in 5g beyond," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12175–12186, Oct. 2020.
[48] Y. He, W. Zhu, and L. Guan, "Optimal resource allocation for pervasive health monitoring systems with body sensor networks," *IEEE Trans. Mobile Comput.*, vol. 10, no. 11, pp. 1558–1575, Nov. 2011.

Guangchen Wang (Member, IEEE) received the BS degree in electrical engineering from the University of Sydney, Sydney, NSW, Australia, and the BS degree in electrical engineering from the Harbin Institute of Technology, Harbin, China, in 2019. He is currently working toward the PhD degree in telecommunication from the University of Sydney. He holds an international postgraduate research scholarship. His research with Prof. Y. Li and Dr. P. Cheng involves machine learning, wireless communications, and Internet of Things (IoTs).

Peng Cheng (Member, IEEE) received the BS and MS degrees with great honors in communication and information systems from University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2006 and 2009, and the PhD degree from Shanghai Jiao Tong University, Shanghai, China, in 2013. From 2014 to 2017, he was a postdoctoral research scientist in CSIRO, Sydney, Australia. From 2017 to 2020, he was an ARC DECRA fellow/lecturer with the University of Sydney, Australia. He is currently an ARC DECRA fellow and a senior lecturer (tenured associate professor in U.S. systems) with the Department of Computer Science and Information Technology, La Trobe University, Australia, and is affiliated with the University of Sydney, Australia. He has published more than 70 peer-reviewed research papers in leading international journals and conferences. His current research interests include wireless AI, machine learning, IoT, millimeter-wave communications, and compressive sensing theory.



Zhuo Chen (Senior Member, IEEE) received the BS degree in electrical engineering from Shanghai Jiao Tong University, Shanghai, China, in 1997 and the MS and PhD degrees from the School of Electrical and Information Engineering, the University of Sydney, Sydney, Australia, in 2001 and 2004, respectively. He was a senior research scientist with Commonwealth Scientific and Industrial Research Organization (CSIRO), Sydney, Australia. His research interests include wireless communications, machine learning, and wireless sensor networks.

Branka Vucetic (Fellow, IEEE) is an ARC laureate fellow and director of the Centre of Excellence for IoT and Telecommunications, University of Sydney. Her current research work is in wireless networks and the Internet of Things. In the area of wireless networks, she works on communication system design for millimeter wave frequency bands. In the area of the Internet of Things, Vucetic works on providing wireless connectivity for mission critical applications. She is a fellow of the Australian Academy of Technological Sciences and Engineering and the Australian Academy of Science.

Yonghui Li (Fellow, IEEE) received the PhD degree from the Beijing University of Aeronautics and Astronautics, in November 2002. From 1999-2003, he was affiliated with Linkair Communication Inc, where he held a position of project manager with responsibility for the design of physical layer solutions for the LAS-CDMA system. Since 2003, he has been with the Centre of Excellence in Telecommunications, the University of Sydney, Australia. He is now a professor with the School of Electrical and Information Engineering, University of Sydney. He is the recipient of the Australian Queen Elizabeth II Fellowship in 2008 and the Australian Future Fellowship in 2012. His current research interests are in the area of wireless communications, with a particular focus on MIMO, millimeter wave communications, machine to machine communications, coding techniques and cooperative communications. He holds a number of patents granted and pending in these fields. He is now an editor for *IEEE Transactions on Communications* and *IEEE Transactions on Vehicular Technology*. He was also the guest editor for IEEE JSAC Special issue on Millimeter Wave Communications for Future Mobile Networks. He received the best paper awards from IEEE International Conference on Communications (ICC) 2014, IEEE PIMRC 2017, and IEEE Wireless Days Conferences (WD) 2014.