Inverse Reinforcement Learning with Graph Neural Networks for IoT Resource Allocation

Conference Paper · June 2023

DOI: 10.1109/ICASSP49357.2023.10096237

6 authors, including:

Guangchen Wang
The University of Sydney
5 PUBLICATIONS **7** CITATIONS
SEE PROFILE

Peng Cheng
La Trobe University
138 PUBLICATIONS **2,530** CITATIONS
SEE PROFILE

Wei Xiang
La Trobe University
479 PUBLICATIONS **12,837** CITATIONS
SEE PROFILE

Branka Vucetic
The University of Sydney
878 PUBLICATIONS **23,333** CITATIONS
SEE PROFILE

ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10096237

INVERSE REINFORCEMENT LEARNING WITH GRAPH NEURAL NETWORKS FOR IOT RESOURCE ALLOCATION

Guangchen Wang[*]*, Peng Cheng*[‡*]*, Zhuo Chen*[†]*, Wei Xiang*[‡]*, Branka Vucetic*[*] *and Yonghui Li*[*]

[*]School of Electrical and Information Engineering, The University of Sydney, Australia
[†]Data 61, CSIRO, Australia
[‡]Department of Computer Science and Information Technology, La Trobe University, Australia
Email: guangchen.wang@sydney.edu.au

ABSTRACT

The rapid development of Internet of Things (IoT) applications requires efficient computing and communication resource allocation strategies to streamline the existing network operations. These strategies could be formulated as mixed-integer nonlinear programming (MINLP) problems, where the optimal branch-and-bound (B&B) with the full strong branching (FSB) variable selection policy features an extremely high complexity. We propose inverse reinforcement learning with graph neural networks (GNNIRL) to generate a new variable selection policy that closely matches the FSB variable selection. Without sacrificing the optimality, the GNNIRL can directly infer the variable selection with a significantly lower complexity, which is also verified by simulation.

1. INTRODUCTION

The massive number of devices in the Internet of Things (IoT) [1] places an ever-increasing demand for scarce spectrum and network resources [2]. Furthermore, many emerging IoT-enabled computation-intensive applications [3] [4] generate an unprecedented volume of data for pattern recognition and call for extremely large computing, storage and caching resources. To meet the requirements of heterogeneous quality-of-service (QoS) in the applications [5], it is desirable to develop resource allocation strategies to significantly streamline the existing network operations.

In essence, the resource allocation over a dynamic IoT environment continuously demands solving an intricate combinatorial optimization problem, which is usually known as mixed-integer nonlinear programming (MINLP) problem. Several contemporary methods are available in literature as potential solutions. The branch-and-bound (B&B) algorithm conducts a systematic search over the discrete variables via an enumeration tree consisting of relaxed sub-problems at nodes and achieves an optimal solution [6]. However, the computational complexity is prohibitively high with a large number of discrete variables, rendering it impractical to handle a large-scale MINLP problem. Consequently, some heuristic optimization algorithms with a low complexity were proposed in the literature, including genetic algorithm (GA) [7], particle swarm optimization algorithm (PSO) [8] and relaxation-based algorithm [9]. However, the aforementioned algorithms are sub-optimal and the performance gap with the optimal B&B algorithm is difficult to quantify and control.

In the B&B algorithm, node selection and pruning policies determine which node is pruned or preserved in the enumeration tree. Consequently, the major efforts in algorithm optimization are placed on improving the efficiency of these two policies. In addition, it is worth noting that the variable selection policy determines the order of the variables for branching, thereby significantly impacting the size of the enumeration tree. The full strong branching policy (FSB) [10], as the optimal variable selection policy, 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.

In this paper, we resort to imitation learning and propose inverse reinforcement learning with graph neural networks (GNNIRL) 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. Specifically, we design the inverse reinforcement learning (IRL) strategy to recover the unknown reward function from the expert demonstrations. We develop a graph neural network (GNN) as a parameterized reward function in the GNNIRL, which directly handles the graphic features in the enumeration tree of the B&B algorithm. Besides, we adopt the maximum entropy (MaxEnt) gradients [11] to match the reward function and the graphic features. Without sacrificing the optimality, our approach could achieve much lower complexity than the FSB. This offers a unified tool to a broad class of MINLP problems in IoT networks by accelerating the optimal B&B algorithm. It is verified by simulation that, given an MINLP problem in task offloading scheduling for MEC, the proposed GNNIRL can significantly accelerate the conventional B&B algorithm. It achieves a much lower computational complexity compared to the existing variable selection policies.

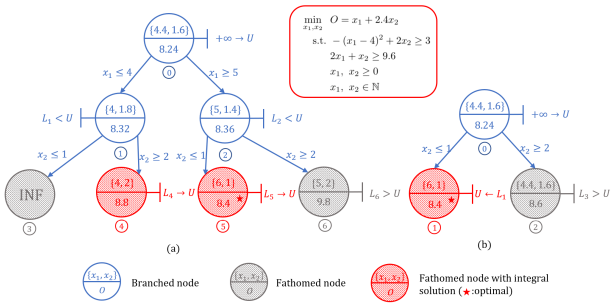


Fig. 1. An example of the B&B algorithm. (a) The variable x_1 is first selected to branch. (b) The variable x_2 is first selected to branch.

2. RESOURCE ALLOCATION WITH THE BRANCH-AND-BOUND ALGORITHM

Many resource allocation problems in IoT networks could be formulated as an MINLP problem, which can be mathematically expressed by

$$\begin{aligned} \boldsymbol{P}: \quad & \min_{\boldsymbol{\alpha},\boldsymbol{\beta}} O(\boldsymbol{\alpha},\boldsymbol{\beta}) \quad \text{s.t. } C(\boldsymbol{\alpha},\boldsymbol{\beta}) \leq 0, \\ & \alpha_i \in \mathbb{N}, i = 1,\ldots,N, \quad \beta_j \in \mathbb{C}, j = 1,\ldots,M, \end{aligned} \tag{1}$$

where $\boldsymbol{\alpha} \triangleq [\alpha_1,\ldots,\alpha_N]^T$ represents N discrete variables, and $\boldsymbol{\beta} \triangleq [\beta_1,\ldots,\beta_M]^T$ represents M continuous variables; $O(\boldsymbol{\alpha},\boldsymbol{\beta})$ and $C(\boldsymbol{\alpha},\boldsymbol{\beta})$ represent the nonlinear objective function and constrains, respectively. Generally, $\boldsymbol{P}$ is NP-hard, and there is no polynomial-time algorithm for finding an optimal solution. This calls for efficient strategies that scale favorably with the problem size.

The B&B algorithm is capable of solving reasonably sized MINLP problems within an acceptable time while guaranteeing optimality, and its key idea is to sequentially partition the feasible set $\mathcal{F}_t$ into smaller ones with relaxations to control the exponential search. The B&B algorithm determines the discrete variables via a recursive method and it could be represented as an enumeration tree as shown in Fig. 1. In the enumeration tree, nodes correspond to convex nonlinear programming problems (NLPs) and are added until the optimal solution is found. The construction complies with three policies: node selection policy, pruning policy, and variable selection policy. Node selection policy determines which node to process, and pruning policy [6] determines whether a node should be fathomed. To improve the basic B&B routine, more efforts have been placed on node selection and pruning. However, the variable selection has been less considered, although it significantly impacts the complexity of the B&B algorithm. For example, in Fig. 1(a), if x_1 is selected first, there will be 7 nodes attached in the tree $(0, ..., 6)$, in stark contrast with only 3 nodes in Fig. 1(b) if x_2 is selected first. This illustrates how important the order of selection is, not to mention the case with more variables. Known as the optimal variable selection policy, the FSB has the highest node creation efficiency [12] and achieves the minimum enumeration tree.

Algorithm 1 Inverse Reinforcement Learning with Graph Neural Networks.

1: Initialize $\boldsymbol{\theta}$.
2: **while** $\boldsymbol{\theta}$ not Converge **do**
3: Update reward function $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$ via GNN $\mathcal{G}$.
4: Update policy π_t via Q-learning with Model Learning.
5: **for** ς in $\mathcal{S}$ **do**
6: Calculate $\mathbb{E}[\mu_\varsigma(\mathbf{s}_t, a_t)]$ based on the policy π_t.
7: **end for**
8: Determine Maximum-Entropy gradients
9: $\dfrac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)} \leftarrow \sum_{\varsigma \in \mathcal{S}} \Big(\mu_{\mathcal{D}_\varsigma}(\mathbf{s}_t, a_t) - \mathbb{E}[\mu_\varsigma(\mathbf{s}_t, a_t)]\Big)$.
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 $R^*(\mathbf{s}_t, a_t)$ via GNN $\mathcal{G}$.
17: Obtain policy π_t^* via Q-learning with Model Learning.

3. THE PROPOSED GNNIRL VARIABLE SELECTION POLICY

It is clear that, although the FSB can achieve the minimum enumeration tree, it needs to solve two NLPs for each variable candidate at one node. Therefore, the FSB variable selection is extremely costly. To address this problem, we propose a novel data-driven variable selection policy referred to as the GNNIRL that closely matches the FSB variable selection but without involving the tedious computations in the FSB. The B&B algorithm is a sequential decision process with an episodic variable selection, so this process can be formulated as a Markov Decision Process (MDP) [13]. It is well known that reinforcement learning (RL) can handle a general MDP, however, RL is inapplicable here because the reward function is challenging to be designed. Before obtaining the optimal solution, branching which variable generates a smaller enumeration tree cannot be known. To address this challenge, we resort to imitation learning and design an IRL-based framework. Instead of trying to manually specify a reward function, we obtain $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$ by a parameterized GNN $\mathcal{G}$, where $\boldsymbol{\theta}$ is the neural network parameters of $\mathcal{G}$. The workflow of the proposed GNNIRL is summarized in **Algorithm 1**.

3.1. Q-Learning with Modeling Learning

In the MDP, the action a_t selects a variable from the unbranched variable list $\mathscr{L}_\mathscr{V}$ based on the state $\mathbf{s}_t$, but the state transition probability $\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$ is unknown in advance. We use the Q-Learning with Modeling Learning [14] to update the expected state-action visitation counts $\mathbb{E}[\mu_\varsigma(\mathbf{s}_t, a_t)]$ and estimate $\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$ based on the state-action-state triple $(\mathbf{s}_t, a_t, \mathbf{s}_{t+1})$. We denote the state-action value function as $Q(\mathbf{s}_t, a_t)$, and simplify $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$ as R_t in this section. It represents the cumulative expected re-

ward function since t, and can be written as $Q(\mathbf{s}_t, a_t) = \mathbb{E}\left[R_t + \gamma Q(\mathbf{s}_{t+1}, a_{t+1}) \mid \mathbf{s}_t, a_t, \pi\right]$, where $\pi = \pi(a_t|\mathbf{s}_t)$ is the policy of the Q-learning with Model Learning, which maps the state $\mathbf{s}_t$ to the action a_t, and γ is the discount rate. When the value of the $Q(\mathbf{s}_t, a_t)$ converges to the maximum, the optimal policy could be determined by $\pi^* = \arg\max_\pi Q(\mathbf{s}_t, a_t)$. The corresponding optimal state-action value function could be denoted as $Q^*(\mathbf{s}_t, a_t)$, and the update process of $Q(\mathbf{s}_t, a_t)$ is

$$\Delta Q(\mathbf{s}_t, a_t) = \lambda\big(R_t + \gamma \max_{a_{t+1}\in\mathcal{A}} Q(\mathbf{s}_{t+1}, a_{t+1}) - Q(\mathbf{s}_t, a_t)\big), \quad (2)$$

where λ is the learning rate. Finally, $Q(\mathbf{s}_t, a_t)$ will converge to $Q^*(\mathbf{s}_t, a_t)$. Consequently, the optimal policy π^* can be derived by the approximate value iteration [11]. Then the policy π^* would be used to generate the expected state-action visitation counts $\mathbb{E}[\mu_\varsigma(\mathbf{s}_t, a_t)]$.

3.2. MaxEnt Gradients

The expert demonstration $\mathcal{D}_\varsigma = \{\tau_\varsigma^i \mid i = 1, \ldots, N_\varsigma\}$ consists of expert trajectories τ_ς^i obtained in advance by solving many MINLP problem instances. The path space of τ_ς^i is denoted by Ω_ς which includes all trajectories starting from ς. The reward function R_t, a feedback indicating whether a_t follows the branching strategy of the FSB, is correlated to the probability of the action attending in the demonstrations ($\mathbb{P}(a_t|\boldsymbol{\theta})$). In addition, $\mathbb{P}(a_t|\boldsymbol{\theta})$ is equivalent to the probability of the trajectory τ over all paths including a_t, and we have $\mathbb{P}(a_t|\boldsymbol{\theta}) \propto \sum_{\tau:a_t\in\tau} \mathbb{P}(\tau|\boldsymbol{\theta})$. Under the MaxEnt [15], over Ω_ς subject to the feature constraints from $\mathcal{D}_\varsigma$, $\mathbb{P}(\tau|\boldsymbol{\theta})$ could be approximated as

$$\mathbb{P}(\tau|\boldsymbol{\theta}) \approx \frac{1}{Z_\varsigma(\boldsymbol{\theta})} e^{\sum_\tau R_t} \prod_\tau \mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t), \quad (3)$$

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

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{\varsigma\in\mathcal{S}} \frac{1}{N_\varsigma} \sum_{\tau\in\mathcal{D}_\varsigma} \log\mathbb{P}(\tau|\boldsymbol{\theta}) = \sum_{\varsigma\in\mathcal{S}} \Big(\frac{1}{N_\varsigma}\Big(\sum_{\tau\in\mathcal{D}_\varsigma} \Big(\sum_{\mathbf{s}_t\in\tau} R_t + \mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)\Big)\Big) - \log Z_\varsigma(\boldsymbol{\theta})\Big). \quad (4)$$

Under the MaxEnt, we can obtain the optimal parameters $\boldsymbol{\theta}$ by maximizing $\mathcal{L}(\boldsymbol{\theta})$, given by $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$. To achieve the maximum $\mathcal{L}(\boldsymbol{\theta})$, we take the partial derivative of (4) with respect to $\boldsymbol{\theta}$, and we have $\frac{\partial\mathcal{L}(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}} = \sum_{\mathbf{s}_t\in\mathcal{S}} \sum_{a_t\in\mathcal{A}} \frac{\partial\mathcal{L}(\boldsymbol{\theta})}{\partial R_t} \cdot \frac{\partial R_t}{\partial\boldsymbol{\theta}}$. Here, $\frac{\partial R_t}{\partial\boldsymbol{\theta}}$ could be obtained via the back propagating the GNN $\mathcal{G}$. In this case, we need to derive $\frac{\partial\mathcal{L}(\boldsymbol{\theta})}{\partial R_t}$, which can be written as

$$\begin{aligned}\frac{\partial\mathcal{L}(\boldsymbol{\theta})}{\partial R_t} &= \sum_{\varsigma\in\mathcal{S}} \frac{1}{N_\varsigma}\Big(\frac{\partial\sum_{\tau\in\mathcal{D}_\varsigma}\sum_{\mathbf{s}_t\in\tau} R_t}{\partial R_t} - \frac{1}{Z_\varsigma(\boldsymbol{\theta})} \\ &\times \sum_{\tau\in\Omega_\varsigma} e^{\sum_{\mathbf{s}_t\in\tau} R_t} \prod_{\mathbf{s}_t\in\tau} \mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t) \frac{\partial\sum_{\mathbf{s}_t\in\tau} R_t}{\partial R_t}\Big) \quad (5) \\ &= \sum_{\varsigma\in\mathcal{S}} \Big(\mu_{\mathcal{D}_\varsigma}(\mathbf{s}_t, a_t) - \mathbb{E}[\mu_\varsigma(\mathbf{s}_t, a_t)]\Big) \triangleq \pi_E - \pi_t.\end{aligned}$$

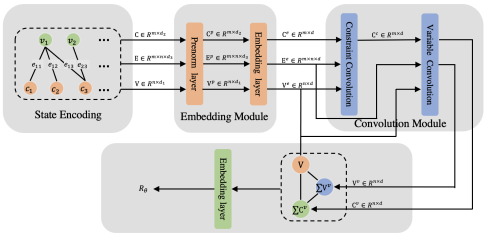


Fig. 2. The functional structure of the proposed GNN.

Furthermore, we define $\pi_E \triangleq \sum_{\varsigma\in\mathcal{S}}(\mu_{\mathcal{D}_\varsigma}(\mathbf{s}_t, a_t))$ as the expert policy with the same behavior as the demonstrations, and define $\pi_t \triangleq \sum_{\varsigma\in\mathcal{S}}(\mathbb{E}[\mu_\varsigma(\mathbf{s}_t, a_t)])$ as the GNNIRL policy.

4. THE PROPOSED GRAPH NEURAL NETWORK

In this section, we elaborate on the proposed GNN $\mathcal{G}$. After updating $\boldsymbol{\theta}$ by MaxEnt gradients, the GNN $\mathcal{G}$ recovers the reward function $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$ based on $\mathbf{s}_t$ and a_t. In specific, the GNN $\mathcal{G}$ is able to capture characteristics of the nodes and their relationship (edges), and aggregate enough information from graphic structures [16]. Besides, $\mathcal{G}$ is permutation-invariant, regardless of the input graph size, $\mathcal{G}$ can embed input features and generalize the different enumeration trees to graph-structured data, which makes $\mathcal{G}$ as an ideal choice for processing graph-structured data. Fig. 2 illustrates the functional structure of the proposed GNN $\mathcal{G}$. 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, constraint, and edge features. The feature embedding module normalizes $\mathbf{s}_t$ to a fixed dimension d, and its outputs serve as the input of the graph convolution module. The graph convolution module aggregates features of $\mathbf{s}_t$ and captures the spatial and temporal graphic information. When MINLP problems are solved, constraints are used to determine the value of the variables. With the embedded constraint feature $\mathbf{c}^e$, a 2-layer perceptron g^c with $relu$ activation functions is used to form a joint feature

$$\mathbf{j}_j^c = \sum_{i=1}^{n} g^c\Big(\mathbf{v}_i^e, \mathbf{c}_j^e, \mathbf{e}_{i,j}^e\Big). \quad (6)$$

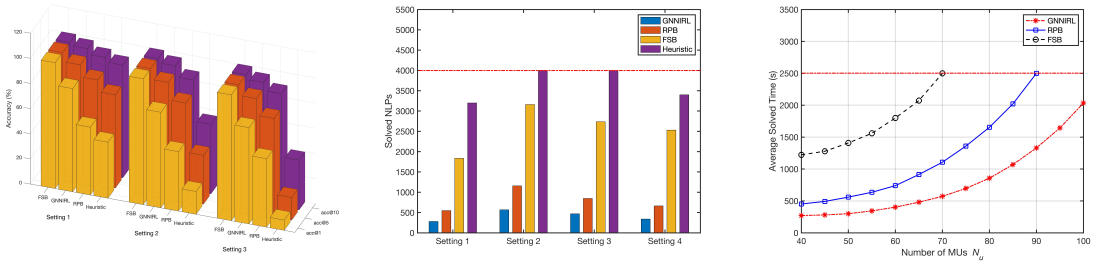
Next, we combine $\mathbf{c}^e$ and $\mathbf{j}^c$, and apply a 2-layer perceptron g^c to obtain new constraint features $\mathbf{c}_j^c = g^c\Big(\mathbf{c}_j^e, \mathbf{j}_j^c\Big)$. To capture the potential node features, we combine $\mathbf{c}^c$, $\mathbf{v}^e$ and $\mathbf{e}^e$, and rearrange them based on the order of variables, and we have

$$\mathbf{v}_i^v = \sum_{j=1}^{m} g^v\Big(\mathbf{v}_i^e, \mathbf{c}_j^c, \mathbf{e}_{i,j}^e\Big). \quad (7)$$

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 we have $\mathbf{c}_i^v = \sum_{j=j_a^i}^{j_b^i} f^v\Big(\mathbf{c}_j^e\Big)$, where j_a^i and j_b^i is the index of the constraint features connecting to $\mathbf{v}_i$. Finally, we design a mix state matrix $\mathbf{M}_k = [\mathbf{v}_k^e, \mathbf{v}_k^v, \mathbf{c}_k^v]^T$ to extract the vectors with the same index k and aggregating edge-weighted information from nodes. Finally, a softmax function is employed to obtain the reward function $R_{\boldsymbol{\theta}}(\mathbf{s}_t, a_t)$.

5. SIMULATION RESULTS

In this section, we take the two-dimensional task offloading problem in [17] as an example and verify the performance of our proposed GNNIRL variable selection policy by simulation.



(a) (b) (c)

Fig. 3. Performance of different scenarios (a) Accuracy for different variable selection policies. (b) Complexity comparison of different variable selection policies in terms of the searched NLPs. (c) Average solved time with different numbers of MUs.

5.1. Policies for Comparison and Performance Metrics

To illustrate the significance of adopting the variable selection policy in the B&B algorithm, we compare our proposed GNNIRL with the FSB, the reliability pseudo-cost branching policy (RPB) [12] and a heuristic policy with random variable selection, and all of them can eventually obtain an optimal solution to the MINLP problem [12]. Here, accuracy is adopted as one of the performance metrics, which quantifies the similarity between other variable selection policies and the FSB in terms of the variable selection order. 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.

5.2. Simulation Results

We set that the number of mobile users (MUs) and mobile servers (MSs) are $N_u = 50$ and $M_s = 20$, respectively, and the remaining parameters are the same as those in [17]. We first demonstrate the accuracy performance of the GNNIRL, the FSB, the RPB, and the heuristic policy in Fig. 3(a). There are 40 online testing instances, and 3 instances are randomly selected and presented in the figure as Setting 1-3. It can be observed that the GNNIRL has higher acc@1, acc@5 and acc@10 than the RPB and the heuristic policy. Specifically, it achieves the accuracy of 78.2%, 92.0% and 96.8% 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 GNNIRL over the RPB in imitating the performance of the FSB. Then, the computational complexity comparison is carried out in Fig. 3(b) in terms of the solved NLPs. Here, 4 instances are randomly selected and presented in the figure as Setting 1-4. It is observed that the GNNIRL significantly reduces the number of the solved NLPs compared with the FSB across all four settings. A simple calculation finds that such a reduction stands at 84.09% on average. Compared with the RPB, the GNNIRL achieves a reduction of approximately 47.99% on average. Besides, we compare the model generalization capability of these policies in Fig. 3(c) considers the original setting ($N_u = 50$) transferring to different MU numbers. Across the whole range of N_u, the GNNIRL has the lowest searched NLPs and the FSB has the highest searched NLPs, consistent with the previous observation in Fig. 3(b). Next, we compare the average delay in terms of the objective function values for four algorithms: the B&B algorithm with the GNNIRL, the B&B algorithm with the FSB, the B&B algorithm with FGPIL variable selection policy [17] and the GA. It is observed that the B&B algorithms with the GNNIRL and the FSB have the same average delay, as they can obtain the optimal solution (c.f. Section 5.1). Compared with the GNNIRL and the FSB, the FGPIL cannot achieve the optimal solution in this large-scale scenario ($N_u = 50$ and $M_s = 20$). In addition, the GA has a significantly higher delay than the other three algorithms. Overall, Figs. 3(b) and 4 show that the B&B algorithm with the GNNIRL can achieve the best average delay while featuring a low computational complexity. This demonstrates the potential of the GNNIRL for solving a large-scale MINLP problem and achieving efficient network resource allocation.

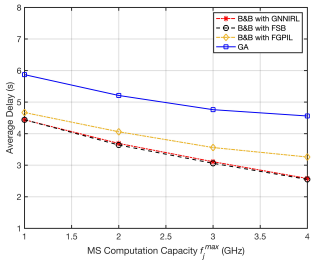


Fig. 4. Average delay of different algorithms.

6. CONCLUSION

Large-scale resource allocation plays a pivotal role in meeting the stringent requirement of massive complex IoT applications. In general, it could be formulated as an MINLP problem, and the B&B algorithm with the FSB represents an optimal solution but incurs a high computational complexity. In this paper, we proposed the GNNIRL that offers a unified solution to a broad class of MINLP problems in IoT networks by generating a new variable selection policy to accelerate the optimal B&B algorithm with significantly reduced complexity but without sacrificing the optimality. It is verified that the GNNIRL achieves a lower computational complexity compared to the existing variable selection policies.

7. REFERENCES

[1] Lalit Chettri and Rabindranath Bera, "A comprehensive survey on internet of things (iot) toward 5g wireless systems," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 16–32, Jan. 2020.

[2] James Adu Ansere, Guangjie Han, Li Liu, Yan Peng, and Mohsin 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] Andreas Kamilaris and Andreas 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] Abdulhamid A. Adebayo, Danda B. Rawat, and Min Song, "Energy-efficient multivariate privacy-aware rf spectrum reservation in wireless virtualization for wireless internet of things," *IEEE Trans. on Green Commun. and Netw.*, vol. 5, no. 2, pp. 682–692, Jun. 2021.

[5] Xuemei Li and Li Da Xu, "A review of internet of things—resource allocation," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 8657–8666, Jun. 2021.

[6] A. H. Land and A. G. Doig, "An automatic method for solving discrete programming problems," *Springer Berlin Heidelberg*, 2010.

[7] Fengxian Guo, Heli Zhang, Hong Ji, Xi Li, and Victor C. M. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2651–2664, Dec. 2018.

[8] Nianyin Zeng, Zidong Wang, Weibo Liu, Hong Zhang, Kate Hone, and Xiaohui Liu, "A dynamic neighborhood-based switching particle swarm optimization algorithm," *IEEE Transactions on Cybernetics*, pp. 1–12, Oct. 2020.

[9] Feng Wang, Jie Xu, and Zhiguo Ding, "Multi-antenna noma for computation offloading in multiuser mobile edge computing systems," *IEEE Trans. Commun.*, vol. 67, no. 3, pp. 2450–2463, Mar. 2019.

[10] T Achterberg, T Koch, and A Martin, "Branching rules revisited," *Operations Research Letters*, vol. 33, no. 1, pp. 42–54, Apr. 2004.

[11] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, 13-17*, Jul. 2008.

[12] T. Achterberg and R. Wunderling, "Mixed integer programming: Analyzing 12 years of progress," *Springer Berlin Heidelberg*, 2013.

[13] S. A. Lippman, *Dynamic Programming and Markov Decision Processes*, The New Palgrave Dictionary of Economics, 1987.

[14] Richard S Sutton and Andrew G Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

[15] Edwin T Jaynes, "Information theory and statistical mechanics," *Physical review*, vol. 106, no. 4, pp. 620, 1957.

[16] Jintao Zhang and Quan Xu, "Attention-aware heterogeneous graph neural network," *Big Data Mining and Analytics*, vol. 4, no. 4, pp. 233–241, Dec 2021.

[17] Zun Yan, Peng Cheng, Zhuo Chen, Branka Vucetic, and Yonghui 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.