# Learning to Optimize Resource Assignment for Task Offloading in Mobile Edge Computing

Yurong Qian, *Student Member, IEEE*, Jindan Xu, *Member, IEEE*, Shuhan Zhu, *Student Member, IEEE*, Wei Xu, *Senior Member, IEEE*, Lisheng Fan, *Member, IEEE*, and George K. Karagiannidis, *Fellow, IEEE*

***Abstract*—In this letter, we consider a multiuser mobile edge computing (MEC) system, where a mixed-integer offloading strategy is used to assist the resource assignment for task offloading. Although the conventional branch and bound (BnB) approach can be applied to solve this problem, a huge burden of computational complexity arises which limits the application of BnB. To address this issue, we propose an intelligent BnB (IBnB) approach which applies deep learning (DL) to learn the pruning strategy of the BnB approach. By using this learning scheme, the structure of the BnB approach ensures near-optimal performance and meanwhile DL-based pruning strategy significantly reduces the complexity. Numerical results verify that the proposed IBnB approach achieves optimal performance with complexity reduced by over 80%.**

***Index Terms*—Mobile edge computing (MEC), branch and brand (BnB), offloading assignment, deep learning (DL).**

## I. Introduction

SMART mobile devices (MDs) are indispensable in modern daily life, whereas an increasing number of MDs has led to a huge challenge of handling a massive amount of data. Mobile edge computing (MEC) is an emerging architecture that can potentially address this challenge [1], through providing the ability of computation offloading to edge networks [2], which accelerates data computing, and improves user experience.

Applying MEC, the MDs dynamically offload computation tasks to the edge computing access points (CAPs) [3]. Solving the resource assignment problem of task offloading plays an essential role in the implementation of MEC, which ensures an efficient use of resources, low latency, and reduced energy consumption [4]. Offloading assignment problem has been extensively studied recently [5]–[7]. In [5], the authors considered non-orthogonal multiple access (NOMA) transmission with offloaded workloads and proposed a distributed algorithm.

The problem of resource assignment, however, is a combinatorial optimization problem, whose solution can be achieved optimally by exhaustive search methods [6], or suboptimally by, e.g., relaxation approaches of convex optimization [7]. Specifically in [7], linear-based and semidefinite-based methods were proposed to optimize the offloading problem. The convex optimization based algorithms usually suffered from noticeable performance loss due to the relaxation of discrete assignment variables to continuous values. In order to approach the optimal performance, exhaustive search based algorithms, e.g., the branch and bound (BnB) approach [6], were proposed for MEC. However, exhaustive search methods are known to be computationally complex especially in a dense communication network with blooming mobile terminals.

Recently, deep learning (DL) has attracted much attention for its excellent performance in fitting arbitrary smooth functions [8]–[10]. In particular, the authors of [8] used a deep neural network (DNN) to approximate the channel characteristics. In [9], the authors further made use of convolutional neural network (CNN) to learn a transmit power control strategy. However, these methods directly learnt the resource assignment solutions, which may lead to unsatisfied classification accuracy and insufficient generalization ability.

Motivated by the above, we propose an intelligent BnB (IBnB) approach to solve the problem of offloading resource assignment in a multiuser dynamic MEC system. The contributions of this letter are summarized as follows.

1) We propose an IBnB approach by making use of a joint model-driven structure of the conventional BnB and the data-driven DL technique. This proposed approach uses DL to learn the pruning strategy of BnB rather than directly learning the assignment solution by a block-box DNN, which improves the final prediction performance.
2) With the proposed IBnB approach, we further advice an adaptive threshold procedure which automatically adjusts the pruning threshold. Specifically, it enables automatically adjusting the threshold value when the initial threshold is too large to ensure a feasible solution. By using this adaptive scheme, the proposed approach enhances the model generalization capability and robustness.

Manuscript received February 21, 2022; revised March 11, 2022; accepted March 12, 2022. Date of publication March 16, 2022; date of current version June 10, 2022. This work was supported in part by the National Key Research and Development Program 2020YFB1806600, in part by the NSFC under Grant 62022026 and Grant 61871139, and in part by the Natural Science Foundation of Jiangsu Province for Distinguished Young Scholars under Grant BK20190012. The work of George. K. Karagiannidis was supported by the European Union and Greek National Funds through the Competitiveness, Entrepreneurship and Innovation Operational Program (EPAnEK) by the Special Actions Aquaculture–Industrial Materials–Open Innovation in Culture under Project T6YBP-00134. The associate editor coordinating the review of this letter and approving it for publication was S. Mazuelas. *(Corresponding authors: Jindan Xu; Wei Xu.)*

Yurong Qian, Jindan Xu, and Shuhan Zhu are with the National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China (e-mail: qianyr@seu.edu.cn; jdxu@seu.edu.cn; shzhu@seu.edu.cn).

Wei Xu is with the National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China, and also with the Purple Mountain Laboratories, Nanjing 211111, China (e-mail: wxu@seu.edu.cn).

Lisheng Fan is with the School of Computer Science, Guangzhou University, Guangzhou 510006, China (e-mail: lsfan@gzhu.edu.cn).

George K. Karagiannidis is with the Electrical and Computer Engineering Department, Aristotle University of Thessaloniki, 54 124 Thessaloniki, Greece (e-mail: geokarag@auth.gr).

Digital Object Identifier 10.1109/LCOMM.2022.3159742

3) Simulation results verify the superiority of the proposed approach. It reduces the complexity by nearly an order-of-magnitude while retaining the optimality of the solution in high probability.

## II. System Model and Problem Formulation

We consider a MEC system with time-varying channel consisting of a CAP and $S$ MDs, denoted by set $\mathcal{S} = \{1, 2, \cdots, S\}$, with insufficient resources. The MEC is applied in an orthogonal frequency division multiple access (OFDMA) scheme with time division duplexed (TDD) mode. There are $N$ time frames and $K$ orthogonal subchannels, denoted by $\mathcal{K} = \{1, 2, \cdots, K\}$. During each frame, all the tasks, coming up in the $S$ MDs, are suggested to be computed at the edge CAP through the $K$ subchannels. Let $\mathbf{X}$ be an indicator matrix of size $S \times K$ corresponding to this task offloading. The $(s, k)$th element of $\mathbf{X}$ is defined as

$$[\mathbf{X}]_{sk} = \begin{cases} 1, & \text{if task } s \text{ is offloaded through subchannel } k, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Assuming that one subchannel can only handle the task from a single MD at a time, we have the constraint for $\mathbf{X}$ as

$$\sum_{s \in \mathcal{S}} [\mathbf{X}]_{sk} \leq 1. \tag{2}$$

In addition, we consider a data partitioned oriented task model, e.g., virus scan and file/figure compression, where the tasks appear at MDs can be flexibly divided into several parts as subtasks [11]. Define $L_s$ as the size of the task generated at MD $s$, and denoted by $l_{sk}$ as the size of one of the subtasks that is offloaded through subchannel $k$. Then, all the subtasks must be computed at the CAP, yielding,

$$\sum_{k \in \mathcal{K}} l_{sk} = L_s. \tag{3}$$

The parameter $l_{sk}$ can be reshaped as a matrix $\mathbf{L}$ of size $S \times K$ and $[\mathbf{L}]_{sk} = l_{sk}$. In general, we have $l_{sk} \leq L_s$ where the equality $l_{sk} = L_s$ occurs when the conveyed subtasks through subchannel $k$ happens to be the entire task.

In applications, latency and energy consumption are two key criteria in MEC networks [3]. Typically, the offloading resource assignment can be formulated in the following problem with the objective function

$$\min_{\mathbf{X}, \mathbf{L}} \Psi(\mathbf{X}, \mathbf{L}) \triangleq \lambda_t T(\mathbf{X}, \mathbf{L}) + \lambda_e E(\mathbf{X}, \mathbf{L}), \tag{4}$$

where $\lambda_t$ and $\lambda_e$ are the weights to balance the importance between latency, $T(\mathbf{X}, \mathbf{L})$, and energy consumption, $E(\mathbf{X}, \mathbf{L})$. The two weights can be determined by the specific systems status, e.g., remaining battery life and tolerable latency.

As we consider the problem of offloading resource assignment, the system latency $T(\mathbf{X}, \mathbf{L})$ in (4) is the transmission latency during offloading. Then, the transmission latency of subchannel $k$ due to the offloading is calculated as

$$t_k = \sum_{s \in \mathcal{S}} \frac{[\mathbf{X}]_{sk} l_{sk}}{R_{sk}}, \tag{5}$$

where $R_{sk}$ represents the uplink data rate of MD $s$ over the $k$th subchannel, which is defined as,

$$R_{sk} = B \log_2 \left(1 + \frac{P_s h_{sk}}{N_0}\right), \tag{6}$$

where $B$ is the communication bandwidth of the subchannel, $P_s$ is the transmit power of the $s$th MD, $N_0$ is the variance of the zero-mean additive white Gaussian noise (AWGN), and $h_{sk}$ is the channel gain between the edge CAP and the $s$th MD through the $k$th subchannel. Without loss of generality, the channel gains are assumed to be block-independent, i.e., invariant within one time frame and varying independently from one frame to another.

Considering the latency of the entire procedure of MEC, all the subchannels transmit their corresponding subtask data simultaneously. Then, the latency is in fact determined by the maximum value among all $t_k{}'s$. It follows

$$T(\mathbf{X}, \mathbf{L}) = \max_{k \in \mathcal{K}} t_k. \tag{7}$$

The energy consumption $E(\mathbf{X}, \mathbf{L})$ in (4) is defined as the sum of consumed energy of all the serving MDs for task data transmission. Accordingly, the energy consumption can be evaluated as

$$E(\mathbf{X}, \mathbf{L}) = \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}} P_s \frac{[\mathbf{X}]_{sk} l_{sk}}{R_{sk}}, \tag{8}$$

Considering equations (1), (2), (3), (7), and (8), the optimization problem of the offloading resource assignment is a mixed integer optimization problem formulated as

$$\begin{aligned} \min_{\mathbf{X}, \mathbf{L}} \Psi(\mathbf{X}, \mathbf{L}) = \lambda_t \max_{k \in \mathcal{K}} & \left( \sum_{s \in \mathcal{S}} \frac{[\mathbf{X}]_{sk} l_{sk}}{R_{sk}} \right) \\ & + \lambda_e P_s \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}} \frac{[\mathbf{X}]_{sk} l_{sk}}{R_{sk}} \qquad \text{(9a)} \\ \text{s.t.} \quad & \sum_{s \in \mathcal{S}} [\mathbf{X}]_{sk} \leq 1, \quad \forall k \in \mathcal{K} \qquad \text{(9b)} \\ & \sum_{k \in \mathcal{K}} l_{sk} = L_s, \quad \forall s \in \mathcal{S} \qquad \text{(9c)} \\ & 0 \leq l_{sk} \leq L_s, \quad \forall k \in \mathcal{K},\ \forall s \in \mathcal{S} \qquad \text{(9d)} \\ & [\mathbf{X}]_{sk} \in \{0, 1\}, \quad \forall k \in \mathcal{K},\ \forall s \in \mathcal{S}. \qquad \text{(9e)} \end{aligned}$$

## III. The Proposed IBnB Approach

Note that the problem in (9) is a mixed integer nonlinear programming (MINLP) problem. To solve this problem, we propose a DL-based IBnB approach. By intelligently learning the sample data, the trained DNN helps determine the future resource assignment strategies, which significantly reduces complexity and ensures near-optimal performance.

### A. Review of Conventional BnB Approach

The BnB approach is based on an exhaustive search mechanism, which systematically enumerates all feasible solutions by traversing the BnB search tree. Specifically, the original optimization problem in (9) is regarded as the root node of the BnB tree, and then branching and bounding are implemented to construct and search the tree. *Branching* is the process of dividing a big parent problem into two subproblems by adding mutually exclusive and complete constraints. This process is regarded as adding child nodes to the tree continuously. *Bounding* is the process of solving and checking the upper and lower bounds of the subproblems in the process of branching. When the subproblem produces a better solution than the current bound, the current node is reserved to branch. Otherwise, this node is pruned.

**Algorithm 1** The BnB Approach

```
1  **Initialize:** Set of the tree nodes: $\mathcal{N} = \{N_0\}$, Upper
   bound of the objective value $\Psi$ in (9a): $Z_{\mathrm{UB}} = \infty$
2  **While** $\mathcal{N} \neq \emptyset$ **do**
3    $j \leftarrow j + 1$;
4    $N_j \leftarrow$ pop the first node from $\mathcal{N}$;
5    Without considering the integer constraints in (9e),
     calculate $(\boldsymbol{x}^{(j)}, \boldsymbol{l}^{(j)}, \Psi^{(j)})$ using (9a), s.t. (9b), (9c),
     (9d), and the special constraint $\mathcal{C}_t$ of this node $N_j$;
6    **if** $\boldsymbol{x}^{(j)}$ is not integer **then**
7      **if** $\Psi^{(j)} \leq Z_{\mathrm{UB}}$ **then**
8        Find the first non-integer component $x_i^{(j)}$ in $\boldsymbol{x}^{(j)}$;
9        Branch the node into two child nodes $N_j^{(1)}$, $N_j^{(2)}$
         by adding two constraints $\mathcal{C}_j^{(1)}$, $\mathcal{C}_j^{(2)}$ respectively,
         $\mathcal{C}_j^{(1)}$: $\mathcal{C}_j \cup \{x_i \leq \lfloor x_i^{(j)} \rfloor\}$,
         $\mathcal{C}_j^{(2)}$: $\mathcal{C}_j \cup \{x_i \geq \lfloor x_i^{(j)} \rfloor + 1\}$;
10       $\mathcal{N} \leftarrow \mathcal{N} \cup \{N_j^{(1)}, N_j^{(2)}\}$;
11     **end if**
12   **else if** $\Psi^{(j)} < Z_{\mathrm{UB}}$
13     $Z_{\mathrm{UB}} \leftarrow \Psi^{(j)}$;
14     $\boldsymbol{x}^*, \boldsymbol{l}^* \leftarrow \boldsymbol{x}^{(j)}, \boldsymbol{l}^{(j)}$;
15   **end if**
16 **end while**
```

Algorithm 1 exemplifies the procedure of applying the BnB approach, where we vectorize the solution matrices $\mathbf{X}$ and $\mathbf{L}$ in (9) as equivalent vector variables $\boldsymbol{x} = (x_1, x_2, \cdots, x_{S\times K})^{\mathrm{T}}$ and $\boldsymbol{l} = (l_1, l_2, \cdots, l_{S\times K})^{\mathrm{T}}$. Denote $\boldsymbol{x}^{(j)} = (x_1^{(j)}, x_2^{(j)}, \cdots, x_{S\times K}^{(j)})^{\mathrm{T}}$ and $\boldsymbol{l}^{(j)} = (l_1^{(j)}, l_2^{(j)}, \cdots, l_{S\times K}^{(j)})^{\mathrm{T}}$ as the optimal relaxation solutions to the subproblem of the $j$th node, and $\Psi^{(j)}$ is the corresponding objective value in (9a), where $j \in \{0, 1, \cdots, J\}$, and $J$ is the number of nodes of the BnB search tree.

As the number of nodes in the BnB search tree is the number of iterations to solve the MINLP problem in (9), the complexity of the BnB approach is directly determined by the number of searched nodes. However, in the BnB search tree, only a few branches point to the optimal solution. A large number of redundant nodes lead to a high computational complexity. A better pruning strategy can make it possible to find the optimal solution with a lower complexity. In the next subsection, we introduce a DL-based method to learn an intelligent pruning strategy, which achieves the optimal performance with significantly reduced complexity.

## *B. The Proposed Low-Complexity IBnB Approach*

Based on recent researches, DL performs well in solving some NP-hard and nonconvex problems [8]–[10]. However, for combinational problems with binary, nonbinary and continuous variables, like the problem in (9), a direct application of DL can hardly achieve a satisfactory solution. Moreover, DL usually requires a large amount of training data, which is a huge challenge for mobile communication applications especially with MEC [12].

To solve the above problems, we propose a low-complexity IBnB approach. Specifically, we design a novel pruning strategy by applying DL, which avoids branching (almost) all nodes as in the conventional BnB approach. We use a DNN to approximate the unknown mapping between the attributes of BnB tree nodes and the pruning decisions in the BnB search tree. In particular, the proposed DNN consists of 6 layers, which respectively has $\{m, 256, 256, 256, 256, 1\}$ neurons in each layer, where $m$ is the input dimension of the training sample data. At each layer except for the last one, the hyperbolic tangent function, tanh $f(x) = (e^x - e^{-x})/(e^x + e^{-x})$, is used as the activation function. In the last layer, we use the sigmoid function, $f(x) = 1/(1 + e^{-x})$, to map the output to the interval $(0, 1)$. The Adam algorithm is chosen as the optimizer in the training phase, and the cross-entropy is used as the loss function.

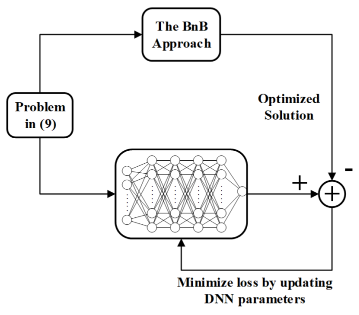
To train the DNN, we generate the dataset through the conventional BnB approach, denoted by $\mathcal{D} = \{\mathcal{I}, \mathcal{O}\}$, where $\mathcal{I}$ is the input dataset and $\mathcal{O}$ is the output dataset. The input data is the attributes of the node, including the variables $j$, $g$, $\boldsymbol{x}^{(j)}$, $\boldsymbol{l}^{(j)}$, $\Psi^{(j)}$, and $f$, where $j$ is the node number, $g$ is the level of the node in the tree, $\boldsymbol{x}^{(j)}$ and $\boldsymbol{l}^{(j)}$ are the relaxation solutions of the subproblem of this node, $\Psi^{(j)}$ is the objective value corresponding to the solutions, and $f$ is the flag indicating whether the subproblem is solvable. The output data in $\mathcal{O}$ is a binary variable $\{0, 1\}$ indicating whether the input node is a parent node of the node corresponding to the optimal solution. If it is true, the output is 1, which means that the node is reserved to branch. Otherwise it is 0, which means that the node needs to be pruned. Fig. 1(a) depicts a diagram of the training procedure.

The training process is periodically performed offline to update the parameters of the DNN. The trained DNN serves as a classifier in the process of the branching. The input of the trained DNN is the attributes of the node, while the output of the trained DNN is a scalar variable ranging in the interval $(0, 1)$, which is used to indicate the pruning strategy. Denote by $\hat{y}$ as the output of the trained DNN. A threshold, denoted by $\theta$, is set to distinguish the pruning decision $\rho$. By comparing the values of $\hat{y}$ and $\theta$, we set

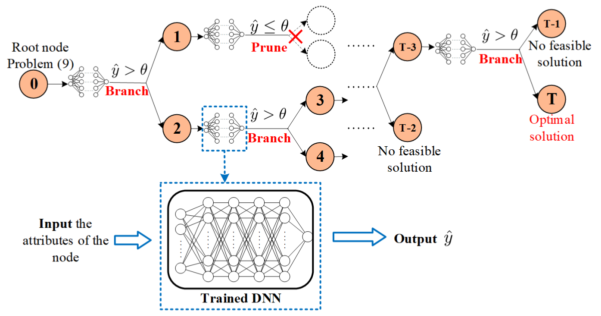
$$\rho = \begin{cases} 1, & \text{if } \hat{y} > \theta, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

Note that $\rho = 1$ implies that the current node is decided to branch into two child nodes, otherwise $\rho = 0$ corresponds to a pruned node. Fig. 1(b) elaborates the DL-based branching process with an example.

Obviously, the threshold $\theta$ has an impact on the performance of the proposed IBnB approach. A lower threshold results in more nodes to be searched, which lifts the computation complexity, while a higher threshold may prune the branch



(a) Training stage



(b) The DL-based branching process of the IBnB approach

Fig. 1. Schematic diagram of the proposed IBnB approach.

**Algorithm 2** The Proposed IBnB Approach

```
1  Initialize: $\mathcal{N} = \{N_0\}$, $\theta = \theta_0$, $Z_{\rm UB} = \infty$
2  While $Z_{\rm UB} = \infty$ do
3    While $\mathcal{N} \neq \emptyset$ do
4      Same as Step 3-5 of Algorithm 1;
5      if $\boldsymbol{x}^{(j)}$ is not integer then
6        if $\Psi^{(j)} < Z_{\rm UB}$ then
7          Input $j, g, f, \boldsymbol{x}^{(j)}, \boldsymbol{l}^{(j)}, \Psi^{(j)}$ to the trained DNN;
8          Output $\hat{y}$;
9          Calculate $\rho$ using (10);
10           if $\rho = 1$ then
11             Same as Step 8-9 of Algorithm 1;
12           end if
13         end if
14     else if $\Psi^{(j)} < Z_{\rm UB}$
15       $Z_{\rm UB} \leftarrow \Psi^{(j)}$;
16       $\boldsymbol{x}^*, \boldsymbol{l}^* \leftarrow \boldsymbol{x}^{(j)}, \boldsymbol{l}^{(j)}$;
17     end if
18   end while
19   $\theta_0 \leftarrow \theta_0 \times \Delta\theta$;
20 end while
```

with the optimal solution. By choosing an appropriate $\theta$, we flexibly reduce the number of searched nodes and guarantee the optimal performance of the proposed IBnB. It is clear that the best case is to prune all the nodes that do not lead to the optimal solution. However, an extreme case is over-pruning, which can lead to no feasible solution. Another extreme case is that a bitty threshold prunes few nodes, and has almost the same complexity as the conventional BnB.

Here, we design an adaptive threshold procedure to guarantee a feasible solution. This procedure enables to adjust the threshold automatically when the initial value of the threshold is too large to ensure a feasible solution for the optimization problem in (9). Specifically, an initial threshold is set, which in most cases can find the optimal solution but prune redundant nodes as much as possible. When the initial threshold is too large, excessive nodes are pruned and there can be no feasible solution left. To fix this, the threshold is lifted by a small step, i.e., $\theta \leftarrow \theta \cdot \Delta\theta$, and then those search trace back a layer to check the IBnB tree to find a solution.

The IBnB algorithm is summarized in Algorithm 2. In steps 4 and 11, the proposed IBnB performs the same steps to solve the relaxation problem of (9) and branch as the conventional BnB. In steps 7-9, the trained DNN is used to prune some redundant nodes with an appropriate threshold $\theta$,

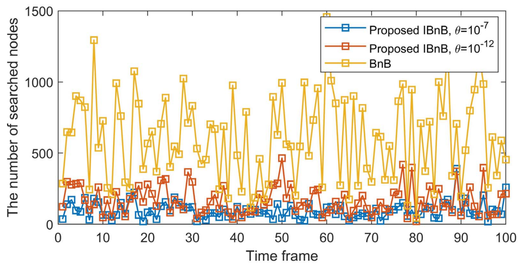


Fig. 2. Comparison of the number of searched nodes of the BnB and the proposed IBnB algorithms in different time frames.

thereby reducing the number of searched nodes to significantly reduce complexity. Step 19 shows the process of adaptive threshold adjustment. The threshold adjustment and learning procedure repeats until a feasible solution is reached.

## C. Complexity Analysis

As shown in Section III-A, as long as there are enough iterations, the optimal solution to (9) can be found. As the training process is usually performed offline to update the parameters of the DNN, the structure and training process of DNN have a marginal impact on the complexity of the proposed approach when it is used online. In both the conventional BnB and the proposed IBnB approaches, the complexity comes from the process of repeatedly solving the MINLP problem in (9). Denote by $\mathcal{O}(S)$ as the complexity of solving the objective function once during the iteration. Then, the complexity of the conventional BnB is $\mathcal{O}(S \times J)$, and with an increasing number of MDs, $J$ will grow exponentially. Denote $T$ as the number of nodes of the IBnB search tree, so the complexity of the proposed IBnB is $\mathcal{O}(S \times T)$. By setting an appropriate threshold, the IBnB approach prunes most redundant nodes, and $T$ becomes proportional to the number of MDs. Obviously, the IBnB approach prune a large number of redundant nodes through an intelligent pruning strategy to greatly reduce the complexity.

## IV. Experimental Results

This section validates the efficiency of the proposed IBnB approach through simulations. In the simulations, the transmit power $P_s$ is chosen from a uniform distribution in the range of $[1.0, 1.5]$ W. The communication bandwidth $B$ is 10 MHz. The channel gain $h_{sk}$ is generated from a Rayleigh fading channel model as in [13], and $N_0$ is set to $-110$ dBm. The CPU

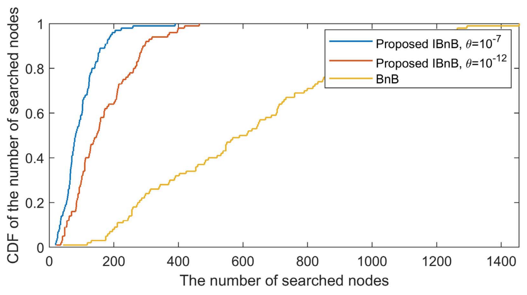


Fig. 3. Complexity comparison.

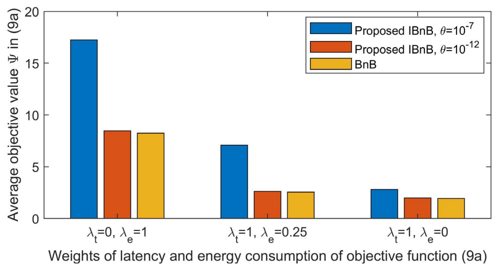


Fig. 4. Performance comparison of the BnB and the proposed IBnB algorthms under different weights of latency and energy consumption.

frequency of the CAP is $r = 2 \times 10^9$ cycles/sec. The weight $\lambda_t$ is set to 1, and $\lambda_e$ is set to 0.25. Two initial thresholds are set to $\theta = 10^{-7}$ and $\theta = 10^{-12}$, and the adaptive threshold step parameter is set to $\Delta\theta = 10^{-5}$.

As the number of searched nodes in the BnB approach directly affects the complexity of the approach, we compare the number of searched nodes to reflect the complexity of the two approaches in Fig. 2. It shows that the complexity of the proposed IBnB is much less than that of the conventional BnB in all time frames. Moreover, from Fig. 2, it is verified that the variance of the complexity of our proposed IBnB is also greatly reduced compared to the conventional BnB. This signifies that the proposed IBnB is robust and appropriate for the delay-turbulance-sensitive tasks.

In Fig. 3, we provide the cumulative distribution function (CDF) of the number of searched nodes for the conventional BnB and proposed IBnB under different values of $\theta$. From Fig. 3, the proposed IBnB with the threshold of $10^{-7}$ can reduce the complexity of the conventional BnB to about $10\%$, while the proposed IBnB with the threshold of $10^{-12}$ can reduce the complexity to about $20\%$. Hence, we can conclude that the proposed IBnB is computationally efficient.

The average cost of the objective value $\Psi$ reflects the performance of the two approaches. In Fig. 4, we compare the performance of the two approaches under different weights of latency and energy consumption of (9a). The different weights may suit for different application scenarios. We can find from Fig. 4 that the proposed IBnB with the threshold of $10^{-12}$ approaches $98\%$ of the optimal performance by the conventional BnB, while at the complexity of only $20\%$. When the threshold is $10^{-7}$, the approach can also approach $80\%$ of the optimal performance. The experimental results show that our proposed approach improves the model's generalization capability under different scenarios.

By adaptively adjusting the threshold, we can always find a feasible solution. In fact, if the initial threshold is low, e.g., lower than $10^{-12}$, the number of searched nodes becomes high and the complexity increases. On the other hand, if the initial threshold is high, e.g., higher than $10^{-7}$, the probability of achieving the optimal solution, or a feasible solution, decreases. After setting the initial threshold based on specific scenario parameters, e.g., the parameters of variable channels and the size of the offloading tasks, the trained DNN is in general able to find a feasible solution approaching the optimal performance, consuming less computation resource.

## V. Conclusion

In this letter, we presented a model-and-data-driven offloading resource assignment approach in a MEC system. The proposed IBnB approach learnt the pruning strategy of the decision-making tree to significantly reduce the complexity. Simulation results were demonstrated to validate that the performance of the proposed IBnB is very close to the optimal one, and the complexity is only one-fifth or even lower compared to that of the conventional BnB.

## References

[1] H. Li, G. Shou, Y. Hu, and Z. Guo, "Mobile edge computing: Progress and challenges," in *Proc. 4th IEEE Int. Conf. Mobile Cloud Comput., Services, Eng. (MobileCloud)*, Mar. 2016, pp. 83–84.

[2] S. S. D. Ali, H. Ping Zhao, and H. Kim, "Mobile edge computing: A promising paradigm for future communication systems," in *Proc. IEEE Region 10 Conf.*, Oct. 2018, pp. 1183–1187.

[3] B. Dab, N. Aitsaadi, and R. Langar, "A novel joint offloading and resource allocation scheme for mobile edge computing," in *Proc. 16th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2019, pp. 1–2.

[4] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[5] Y. Wu, K. Ni, C. Zhang, L. P. Qian, and D. H. K. Tsang, "Noma-assisted multi-access mobile edge computing: A joint optimization of computation offloading and time allocation," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12244–12258, Dec. 2018.

[6] G. Yang, L. Hou, X. He, D. He, S. Chan, and M. Guizani, "Offloading time optimization via Markov decision process in mobile-edge computing," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2483–2493, Feb. 2021.

[7] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.

[8] G. Gui, H. Huang, Y. Song, and H. Sari, "Deep learning for an effective nonorthogonal multiple access scheme," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8440–8450, Sep. 2018.

[9] W. Lee, M. Kim, and D.-H. Cho, "Deep power control: Transmit power control scheme based on convolutional neural network," *IEEE Commun. Lett.*, vol. 22, no. 6, pp. 1276–1279, Jun. 2018.

[10] S. Zhu, W. Xu, L. Fan, K. Wang, and G. K. Karagiannidis, "A novel cross entropy approach for offloading learning in mobile edge computing," *IEEE Wireless Commun. Lett.*, vol. 9, no. 3, pp. 402–405, Mar. 2020.

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

[12] L. Liang, H. Ye, G. Yu, and G. Y. Li, "Deep-learning-based wireless resource allocation with application to vehicular networks," *Proc. IEEE*, vol. 108, no. 2, pp. 341–356, Feb. 2020.

[13] 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.