Efficient IoT Devices Deployment Using Branch and Bound Method

Haesik Kim

VTT Technical Research Centre of Finland
P.O. Box 1100, FI-90570 Oulu, Finland
haesik.kim@vtt.fi

***Abstract*— The Internet Of Thing (IoT) networks are widely investigated in 5G system and will be still a key technical system to drive massive connectivity of 6G systems. As the IoT devices and networks are getting smarter, the IoT ecosystem allows us to bridge between human life and digital life and accelerate the transition towards a hyper-connected world. Optimal and scalable IoT network design has been investigated in many research groups but key challenges in this topic still remain. In this paper, we investigate IoT devices deployment problem to minimize the transmission and computation cost among network nodes. We formulate the IoT devices deployment problem as Mixed-Integer Nonlinear Programming (MILNP) problem. After relaxing the constraints and transforming the problem to a mixed integer linear programming (MILP) problem, we propose a new branch and bound (BB) method with a machine learning function and solve the MILP problem. In the numerical analysis, we evaluate both conventional BB method and the proposed BB method with weighting factors and compare the objective function values, the number of explored nodes, and computational time. The performances of the proposed BB method are significantly improved under the given simulation configuration. We find the optimal mapping of IoT devices to fusion nodes.**

***Index Terms*— IoT, 6G, Mixed Integer Linear Programming, Branch and Bound method, Machine learning, etc.**

I. Introduction

THE Internet Of Thing (IoT) network can be defined as the network that can connect billions of things. It has great potential and enables us to create various new services and offer new solutions in 5G and 6G era. IoT networks are widely investigated in 5G system and will be still a key technical system to drive massive connectivity of 6G systems. The concept of IoT has been extended to internet of everything (IoE) [1]. We have intelligent IoT devices and networks. It enables us to integrate multiple intelligent sensing devices. They will be able to identify, monitor, and decide intelligently. The sensing devices of IoE will be able to obtain various data and these data can be utilized for multiple key 6G use cases such as ehealth, smart cities, transportation, energy, smart factories and so on. As the IoT devices and networks are getting smarter, the IoT ecosystem allows us to bridge between human life and digital life and accelerate the transition towards a hyper-connected world. However, there are still key research challenges to realize the IoT ecosystem. We can summarize them as optimal and scalable IoT devices deployment, low energy efficiency, self-organized IoT networks, security and privacy and so on. In particular, optimal and scalable IoT network design has been investigated in many research groups but this challenge still remains. As increasing the number of IoT devices as well as the number of services, IoT network designers should take into account many design parameters such as complexity, scalability, latency, fault tolerance, privacy and security. They should be optimized but these parameters have a trade-off relationship because they are closely related to each other. Thus, IoT network designers often make a decision subjectively and empirically. It is not easy to find an optimal IoT devices deployments. In this paper, we design IoT networks to achieve optimal transmission and reduce computation among IoT devices and fusion nodes using the branch and bound (BB) method and machine learning algorithm.

The BB methods are widely used to find a solution for convex or non-convex problems that cannot be solved in polynomial time. The BB methods are based on enumerative approaches and suitable for solving mixed integer linear programming (MILP). Land and Doig proposed the BB method [2] to solve discrete optimization problems. The basic idea is to divide the original problem into multiple sub-problems that are easier to solve and enumerates all candidate solutions. Based on the implicit exploration of the feasible region, we can obtain the solution. Four components such as branch variable selection, node selection, node pruning, and cutting-plane selection are the main functions of the BB method. Branch variable selection is a task about which fractional variable will be selected and how to partition a current node into two children nodes in a search tree. Node selection is about which nodes will explore. The node selection rules are typically based on depth-first search, breath-first search, or best bound search. Node pruning is to prevent exploration of sub-optimal region. Cutting plane selection is the rule to add constraints to find cutting planes. They affect to the performance and complexity of the BB methods. Depending on how they are designed, we can reduce the feasible region and the number of the iteration and find optimal solution efficiently. There are many researches about how to design the main functions [3]. One of key research challenges for BB methods is to reduce the computational times and the complexity. Artificial intelligence (AI) algorithms enables us to improve BB methods by decomposing the problem into the sub-problems and learning the policy from the experience. Table 1 summarizes AI algorithms adaptation to improve the BB methods.

Table 1. Literature review

Target function to improve	Summary	Ref
Branch variable selection	-Learning approach is adopted for regression -Computation time reduction	[4]
Branch variable selection	- Support vector machine is adopted for rank formulation -Consider many good candidate and good performance at medium size problems	[5]
Branch variable selection	- Graph convolutional neural network is adopted to reduce feature computation cost. - Computation time reduction	[6]
Node selection	- Reinforcement learning is adopted for improve the search nodes and the number of iteration. - Smaller search nodes and iteration number reduction.	[7]
Node selection	-Deep neural network is adopted to estimate low bound. -Better solution is obtained than heuristic method.	[8]
Node pruning	-Learning methods is used to achieve better bounds. - Accelerating the pruning process.	[9]
Cutting plane selection	-Reinforcement learning is used. They formulated the process of sequentially selecting cutting planes as a Markov decision process. -Improvement the performance of heuristics.	[10]

In this paper, we investigate IoT device deployment problem to minimize the transmission and computation cost among network nodes. We formulate the IoT device deployment problem as Mixed-Integer Nonlinear Programming (MILNP) problem. After relaxing the constraints and transforming the problem to a mixed integer linear programming (MILP) problem, we propose a new BB method with a machine learning function and solve the MILP problem. The main contributions of this paper can be summarized as follows:

(1) IoT network deployment problem formation as MILP problem to optimize the transmission among network nodes,

(2) New BB method with a machine learning function to reduce the computational complexity,

(3) Performance analysis and evaluation of the proposed BB method.

The remaining part of this paper is organized as follows: In Section II, system model is described and the problem is formulated. In Section III, the proposed BB method is explained. In Section IV, the performances of the proposed method are evaluated and compared with the conventional BB method. Section V contains the conclusion and summary.

II. System Model and Problem Formulation

In this section, we define the IoT network system model and formulate the problem.

A. System Model

We consider 2 dimensional (2D) model of IoT networks. The IoT network consists of network nodes such as IoT devices (end nodes), fusion nodes (gateways) and base stations. The base stations are connected to cloud networks. The type of all IoT devices are same. Their location is fixed. The IoT devices are directly connected with fusion nodes over a wireless link. The multiple hops between IoT devices are not allowed. The fusion nodes are connected to the base stations via a wireless link as well. The network nodes are on separate integer grid points between 1 and *N*. *N* is an integer. The number of IoT devices, fusion nodes and base stations can be expressed as $\lfloor iN^2 \rfloor$, $\lfloor fN^2 \rfloor$ and $\lfloor bN^2 \rfloor$, respectively. i, f and $b \in \mathbb{R}$ represent the density of IoT devices, fusion nodes and base stations, respectively. Their locations are randomly selected in the integer grid points. Figure 1 illustrates an example of 2D IoT network model.

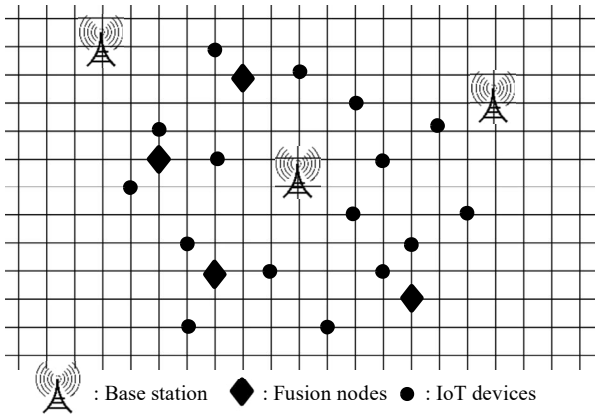


Fig 1 Example of a 2D IoT network model.

IoT devices require specific application services. Service providers define the IoT device groups of the specific services and the IoT device groups are operated for specific applications. The applications can be computed at the base stations. When a base station computes specific application to reduce the network traffics at the core network, we can reduce the latency and improve the service quality. However, due to the limited computing resources, some applications can be computed at cloud-computing servers while not violating specific latency conditions. In this system model, base stations have enough computational power. In mobile network systems, IoT networks are widely referred to as graphs. The graph theory can represent symmetric or asymmetric relations among nodes and edges. We express IoT networks as a graph $G = (V, E)$ where V and E denote network nodes (vertices) and network links (edges), respectively. The node has a 2D position component $v \in V = (v_1, v_2) \in \mathbb{R}$. A hub (a node with high degree) becomes a fusion node. The service demand for each application $a \in \mathbb{R}$ at an IoT device v is $D_{v,a}$. It represents how much it can be served in a time interval. In the IoT network, applications from a base station and cloud should be distributed to IoT devices in the demand. The instances are independent of each other. Maximum number of instances N_a represents how application can be horizontally scaled. Namely, it indicates how much we cope with the new demands by adding nodes to the networks. Resource demand function $f_a^r(w)$ is defined as the average amount of resources r required by an instance of application with a workload w. The workload is defined as the average arrival rate of the requests for an instance of application. The workload rate w_a is the average request rate of IoT devices for the application a. It can be expressed as a Poisson distribution function. It indicates how application can be vertically scaled. Vertical scaling means to add more resources to existing networks. We assume that each IoT device requests for only one application and the instance of applications handles their

requests. We define the capacities of base stations and fusion nodes as $C_{u,a}$ and $C_{v,a}$, respectively, where u denotes a base station node and v denotes a fusion node. The total data rate of the application a from a fusion node to an IoT device is less than $C_{v,a}W_a$ where W_a is the bandwidth of the application a to transmit from a fusion node to an IoT device. We assume that each IoT device receives an application service from one fusion node. We need to decide the most efficient transmission from an IoT device to a fusion node. In the practical IoT network deployment and management, there are the resource limitations and latency requirements in terms of applications and network configuration. The distance between nodes and the transmission power affect to the transmission latency and reliability. We simply represent the transmission cost as $\alpha d_{u,v} p_a^{u,v}$ where α is constant, $d_{u,v}$ is distance between the node u and the node v, and $p_a^{u,v}$ is the transmission power when transmitting the application a from the node u to the node v. The computational cost to process the application a at the base station is represented as γ_a^u where u is a base station node. We consider the transmission latency and reliability and the computational cost as the costs of the problem.

B. Problem Formulation

Given the network node locations and the demands, we optimize the transmission among the network nodes so that IoT devices receive an application service satisfactorily. Namely, we find the pair between IoT devices and fusion nodes to minimize the transmission and computation cost. In order to formulate the problem, we define two variables as follows: $x_a^{u,v} \in \mathbb{R}$ represents the volume of the application a transmitting from a node u to a node v. $y_{u,v} \in \{0,1\}$ represents a binary number indicating whether or not a node u is associated with a node v. The objective function to minimize can be expressed as follows:

$$\min \sum_{a,u,v} f(a,u,v) + \sum_{a,v,l} g(a,v,l) \tag{1}$$

where

$$f(a,u,v) = x_a^{u,v}\left(\gamma_a^u + \alpha d_{u,v} p_a^{u,v}\right) \tag{2}$$

represents the distribution cost from a base station to and a fusion node and

$$g(a,v,l) = y_{v,l}\left(\alpha d_{v,l} p_a^{v,l} D_{l,a}\right) \tag{3}$$

represents the distribution cost from a fusion node to an IoT device node in order to satisfy the demands, where $u \in U$, $v \in V$, and $l \in L$ denotes a base station node, a fusion node and an IoT device node, respectively. The constraint C_1 of the problem is the capacity limitation of a base station. It is defined as follows:

$$\mathrm{C}_1: \sum_v x_a^{u,v} \le C_{u,a}, \qquad \forall a \in A. \tag{4}$$

The constraint C_2 is the demands for the IoT devices for the application a. It is defined as follows:

$$\mathrm{C}_2: \sum_u x_a^{u,v} = \sum_l D_{l,a} y_{v,l}, \qquad \forall a \in A. \tag{5}$$

The constraint C_3 is about the resource capability at the network node. The resource demand at the network node v should be less than the resource capability as follows:

$$\mathrm{C}_3: \sum_a \rho_a^v f_a^r(\lambda_a^v) \le C_{v,r}, \qquad \forall r \in R, v \in V \tag{6}$$

where resource capability $C_{v,r}$ represents the total resource capability $r \in R$ on the network node v. The binary variable ρ_a^v indicate a node v computes an instance of the application a. The requested arrival rate λ_a^v for the application a on the node v is defined as the sum of all requested arrival at the node v.
The constraint C_4 represents that one IoT device node is connected to only one fusion node as follows:

$$\mathrm{C}_4: \sum_v y_{v,l} = 1. \tag{7}$$

The constraints C_5 and C_6 are a non-negative amount of applications and a binary number, respectively. They can be defined as follows:

$$\mathrm{C}_5: x_a^{u,v} \ge 0 \tag{8}$$

and

$$\mathrm{C}_6: y_{v,l} \in \{0,1\}. \tag{9}$$

Now, we can formulate the problem as follows:

$$\min \sum_{a,u,v} f(a,u,v) + \sum_{a,v,l} g(a,v,l)$$

Subject to (10)

$$\mathrm{C}_1, \mathrm{C}_2, \mathrm{C}_3, \mathrm{C}_4, \mathrm{C}_5, \mathrm{C}_6.$$

The optimization problem (10) includes continuous and discrete variables and nonlinear functions in the constraints. We can regards this problem as a Mixed-Integer Nonlinear Programming (MINLP) problem. This problem is known as a NP-hard problem. The solution of the MINLP problem typically requires searching huge candidates. Due to the high complexity [11], it is difficult to solve the problem.

C. Relaxation of the MILNP problem

There are many approaches to solve the MINLP problem, such as heuristic algorithms, branch and bound algorithms and so on [12]. Our approach is to relax the constraints, transform (10) to a mixed integer linear programming (MILP) problem and then solve the problem. In the constraint 3, the resource demand function $f_a^r(\lambda_a^v)$ is not a linear function. In order to simplify the resource capability at the network node, we express the constraint $\mathrm{C'}_3$ as the capacity limitation of a fusion node. It is defined as follows:

$$\mathrm{C'}_3: \sum_{a,l} \frac{D_{l,a} y_{v,l}}{W_a} \le C_{v,a}. \tag{11}$$

We can re-formulate the problem as follows:

$$\min \sum_{a,u,v} f(a,u,v) + \sum_{a,v,l} g(a,v,l) \tag{12}$$

Subject to

$$C_1, C_2, C'_3, C_4, C_5, C_6.$$

As we can observe the above problem formulation, two variables $x_a^{u,v}$ and $y_{v,l}$ are linear in the objective functions and constraints function. Since $y_{v,l}$ is restricted to integer, we can regard this problem as a mixed integer linear programming (MILP) problem. The MILP problem is basically a decision making problem. The BB method is widely employed to solve general MILP problems but the computational complexity and time is exponential in the size of the MILP problem. Thus, it is a key research challenge to reduce the complexity.

III. The Proposed BB Method

As we discussed in section I, branch variable selection is a key task of the BB method. The branching of the BB method is to choose a fractional variable in the search tree. If the branching is not efficient, it fails the sub-problem simplification and increase the size of the BB method tree. The objective of branch variable selection is to reduce the number of search nodes. There are multiple approaches to determine the branching rule. For example, one of the common approaches is the most fractional variable. We score candidate variables and measure their effectiveness. Among them, we select one candidate with fractional part closest to 0.5. Another approach is to give them a branching priority in terms of different target metrics. Another approach is to check the progress before actual branching and then branch on fractional variable with strong branch by tracking. We call this strong branching. In terms of the children node expansion, it is the most efficient way but the main disadvantage is a huge computational cost. AI techniques can used to learn and estimate the scoring. They enable us to improve the computational time and reduce the number of the explored nodes. This process can be regarded as supervised learning. We call this imitation learning. Consider a search tree for a MILP and a node M with objective function value $\check{z}$, solution $\check{s}$, variable s_j and candidate variable set $\mathcal{C}$ in the search tree. Two children nodes M_j^- and M_j^+ are obtained from branching on j up-fractionality and down-fractionality. They have feasible values $\check{z}_j^-$ and $\check{z}_j^+$. If M_j^- and M_j^+ are in feasible, $\check{z}_j^-$ and $\check{z}_j^+$ are set to a large value. The change can be expressed as $\Delta_j^- = \check{z}_j^- - \check{z}$ and $\Delta_j^+ = \check{z}_j^+ - \check{z}$. It is important to make a decision on which fractional variable to branch. A branching rule is defined by its score function $\zeta()$ as follows:

$$\mathcal{B}_j = \zeta(\max\{\Delta_j^-, \epsilon\}, \max\{\Delta_j^+, \epsilon\}) \quad (13)$$

where ϵ is a small constant (For example, 10^{-6}) [13]. The product as the score function, $\zeta(a, b) = ab$, was proposed in [13] in order to make a balance between the subtree sizes. It attempts to find the variable with the maximum score of (13) by tracking the branching process in the candidate set $\mathcal{C}$. We propose a weighting factor ω_j and (13) is rewritten as follows:

$$\mathcal{B}_j = \omega_j \zeta\big(\max\{\Delta_j^-, \epsilon\}, \max\{\Delta_j^+, \epsilon\}\big). \quad (14)$$

The weighting factor $0 < \omega_j < 2$ is determined by machine learning algorithms. Machine learning algorithms (ex. a linear regression, support vector machine, deep learning and so on) that are able to perform prediction can be used for the weight factor selection. In this paper, a linear regression model is used. Using the training data sets about variable selection, a machine learning model is trained. After training machine learning model enough, we can predict the variable selection at test data sets. Depending on the training results, each variable is weighted differently. The value of ω_j, has greater than 1 when a variable is highly likely selected. The weight factors satisfy $E[\boldsymbol{\omega}] = 1$.

Assume that we solve an optimization problem $\mathcal{P} = (\mathcal{S}, \mathcal{F})$ where $\mathcal{S}$ and $\mathcal{F}$ are a search space and an objective function, respectively. The search space means a set of candidate solutions. The objective function is $\mathcal{F}: \mathcal{S} \to \mathbb{R}$. We find an optimal solution $s^* \in \arg\min_{s \in \mathcal{S}} \mathcal{F}(s)$. A search tree T of sub-problem is constructed by the BB method. A feasible solution $\check{s} \in \mathcal{S}$ is globally stored. We find a new subset of the search space $\mathcal{C} \subset \mathcal{S}$ in order to explore from a queue $\mathcal{Q}$ of unexplored subsets. If a solution $\hat{s}' \in \mathcal{C}$ has a better value than $\hat{s}$ (Namely, $\mathcal{F}(\hat{s}') < \mathcal{F}(\hat{s})$), the solution is updated. If there is no solution better than $\hat{s}$ (Namely, $\mathcal{F}(\hat{s}') \geq \mathcal{F}(\hat{s}), \forall s$), the subset is pruned. Otherwise, we divide the subset $\mathcal{C}$ into $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_r$ that are pushed onto $\mathcal{Q}$. If there is no unexplored subset in the queue $\mathcal{Q}$, we terminate the BB method and return the solution. The followings are the pseudo codes of the proposed BB method and variable selection with machine learning.

Procedure the proposed BB method
Set $\mathcal{Q} = \mathcal{S}$
Initialize $\hat{s}$
while $\mathcal{Q} \neq \emptyset$ **do**
  Choose a subset $\mathcal{C}$ from $\mathcal{Q}$ to explore
  if $\hat{s}' \in \{s \in \mathcal{C} | \mathcal{F}(s) < \mathcal{F}(\hat{s}\}$ can be found **then**
    Set $\hat{s} = \hat{s}'$
  if $\mathcal{C}$ can't be pruned **then**
    Divide $\mathcal{C}$ into $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_r$ by variable selection with weight prediction of machine learning
    Put $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_r$ into $\mathcal{Q}$
  end if
    Get rid of $\mathcal{C}$ from $\mathcal{Q}$
end while
return $\hat{s}$
End procedure

Procedure the variable selection with weight prediction of machine learning
Input Subset of the current node $\mathcal{C}$ with its optimal solution
  - Define branching candidate set $\mathcal{G} = \{i \in I | \hat{s}_i \notin \mathbb{Z}\}$
  - Compute the score $\mathcal{B}_j = \omega_j \zeta\big(\max\{\Delta_j^-, \epsilon\}, \max\{\Delta_j^+, \epsilon\}\big)$ for each candidate $i \in \mathcal{G}$, where ω_j is given by machine learning prediction.
Return $i \in \arg\min_{i \in \mathcal{G}} \mathcal{B}_j$
End procedure

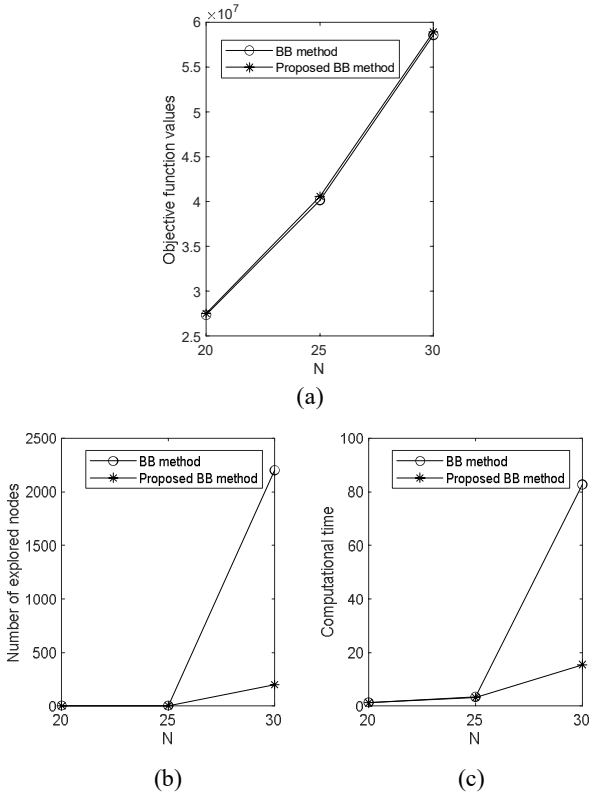
IV. Numerical Analysis

We formulated the IoT network deployment problem as MILP problem (12) in order to minimize the transmission and computation cost. Using a MILP solver of Matlab [14] and

YALMIP [15], we solve the IoT device deployment problem to minimize the transmission and computation cost between networks nodes. Hardware specification is 1.7GHz dual core CPU, 8GB RAM, and 64 bit OS system. The key simulation configurations are summarized as follows:

- MILP solvers: the conventional BB method and the proposed BB method.
- Weighting factor $0 < \omega_j < 2$
- Machine learning algorithm: linear regression
- Integer grid points: N = 20, 25, 30
- Density of IoT devices, Fusion nodes and base stations: i=0.1, f=0.05 and b =0.05
- Network component nodes (IoT devices, Fusion nodes and base stations) locations are randomly selected in the integer grid points
- Total demand of applications by a base station: $\sum D_{v,a} = 20$
- Computational cost range at base stations : $20 \leq \gamma_a^b \leq 100$
- Capacity range of base stations: $500 \leq C_{u,a} \leq 1500$
- Capacity range of fusion stations: $8000 \leq C_{v,a} \leq 16000$
- Bandwidth range for application a: $1 \leq W_a \leq 3$
- Transmission power range for application a : $5 \leq p_a^{u,v} \leq 10$

We evaluate both conventional BB method and the proposed BB method with weighting factor and compare the objective values, the number of explored nodes in the search tree and computational time as shown in figure 2.

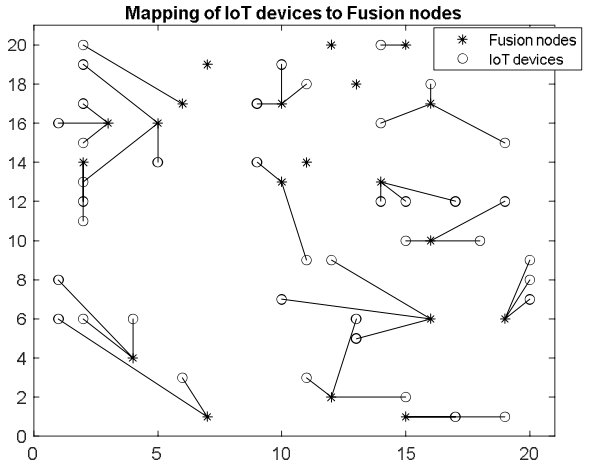


(a)

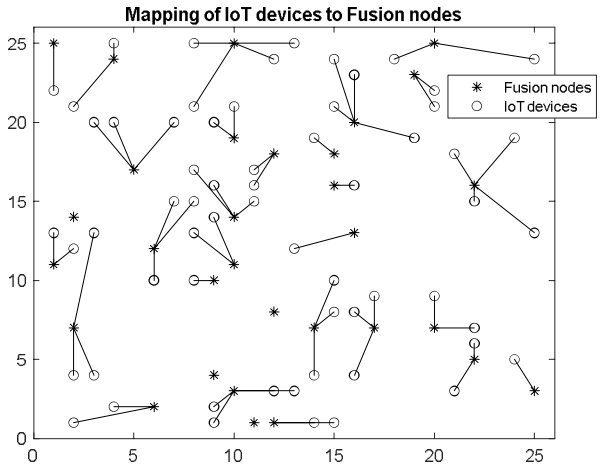
(b) (c)

Fig 2 Objective values (a), number of explored nodes (b) and computational time (c)

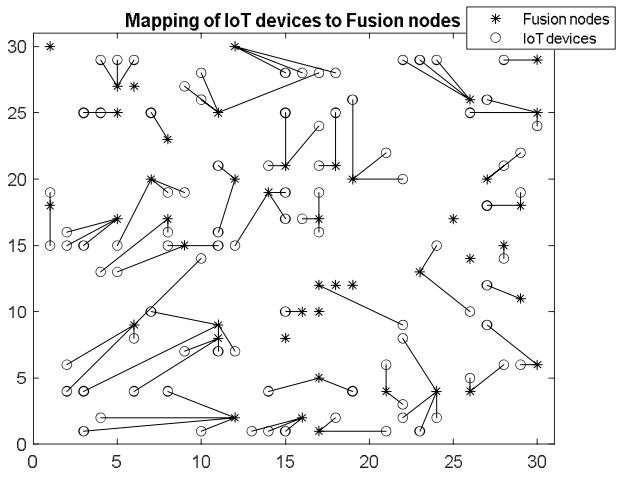
As we can observe figure 2, the proposed BB method significantly reduces the number of explored nodes and computational time at N = 30. In the proposed BB method, we need only 10% of explored nodes and 18% of computational time to find the optimal solution for the MILP problem. This approach will be useful when deploying the medium size of IoT devices. However, we assume the linear regression model is fully trained and the weighting factors are accurate. If we do not have the fully trained model, the performances are similar to the conventional BB method. If it is not well trained with the enough number of the training set, the performance of the proposed BB method is even worse. In addition, the formulated MILP problem can be solved well for the medium size of network nodes and applications. When we have a large number of network nodes (N>40) and applications ($\sum D_{v,a}$>30), it is not solvable well. Figure 3 illustrates the mapping of IoT devices to fusion nodes after solving the MILP problem.



(a)



(b)



(c)

Fig 3 Mapping of IoT devices to fusion nodes at N = 20 (a), 25 (b) and 30 (c)

As we can observe figure 3, we solve the MILP problem and find the optimal connections between IoT devices and fusion nodes. The mapping of both BB methods is same. In figure 3, some fusion nodes are not connected to IoT devices. It represents that the fusion nodes are not in use in the optimal solution and go sleep. Sometime, the randomly generated demands and capacities can exceed the constraints and lead to infeasible problems in particular at the large number of IoT devices and applications. We can't find the optimal solution of the MILP problem.

V. Conclusion and Summary

In this paper, we investigate the IoT device deployment problem in order to minimize the transmission and computation cost. The optimization problem is formulated as a MILNP problem. After relaxing the constraints and transforming to a MILP problem, we solve the problem using the proposed BB method. The proposed BB method includes a machine learning function and give a weighting factor to score function of branching rule. When having a well trained machine learning model, the weighting factor guides to find us through accurate branch variable selection. Thus, it reduces the number of explored nodes and computational time significantly. However, we assumed that the machine learning model is fully trained and the weighting factor is accurate. When we have moving IoT devices, varying channel models, and flexible network configuration, the training will be very challenging. Thus, as further works, how to efficiently train the machine learning model should be investigated. In addition, the proposed method is limited when we have a large number of network nodes and applications. Thus, we should investigate further generalization of the proposed method.

Acknowledgement

This work was supported by the European Commission in the framework of the H2020-ICT-19-2019 project 5G-HEART (Grant agreement no. 857034).

References

[1] D. Vaya and T. Hadpawat, "Internet of Everything (IoE): A new era of IoT," in Proc. *ICCCE*, 2020, pp. 1–6.

[2] A. H. Land and A. G. Doig, "An automatic method for solving discrete programming problems," in *Econometrica*, 1960, vol. 28, no. 3, pp. 497-520.

[3] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell, "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning," *Discrete Optimization*, vol. 19, pp. 79{102, 2016.

[4] A. M. Alvarez, Q. Louveaux, L. Wehenkel, "A machine learning-based approximation of strong branching," *INFORMS Journal on Computing*, vol. 29, no. 1, pp. 185-195, 2017.

[5] E. Khalil, P. Le Bodic, L. Song, G. Nemhauser, and B. Dilkina, "Learning to branch in mixed integer programming," Proceedings of the *AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, Feb. 2016.

[6] M. Gasse, D. Chetelat, N. Ferroni, L. Charlin, and A. Lodi, "Exact combinatorial optimization with graph convolutional neural networks," arXiv preprint, arXiv:1906.01629, 2019.

[7] A. Sabharwal, H. Samulowitz, and C. Reddy, "Guiding combinatorial optimization with uct," in *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming,* Springer, 2012, pp. 356-361.

[8] A. Hottung, S. Tanaka, and K. Tierney, "Deep learning assisted heuristic tree search for the container pre-marshalling problem," *Computers & Operations Research*, vol. 113, p. 104781, 2020

[9] H. He, H. Daume III, and J. M. Eisner, "Learning to search in branch and bound algorithms," *Advances in neural information processing systems*, vol. 27, pp. 3293-3301, 2014.

[10] Y. Tang, S. Agrawal, and Y. Faenza, "Reinforcement learning for integer programming: Learning to cut," in *Proceedings of the 37th International Conference on Machine Learning*, vol. 119. PMLR, 13-18 Jul 2020

[11] S. Burer and A. N. Letchford, "Non-convex mixed-integer nonlinear programming: A survey," *Surveys in Operations Research and Management Science*, vol. 17, no. 2, pp. 97 – 106, 2012.

[12] R.J. Dakin RJ, "A tree–search algorithm for mixed integer programming problems," *Comp Journal*, Vol.8, pp250–255, 1965.

[13] T. Achterberg, T. Koch, and A. Martin, "Branching rules revisited," *Operations Research Letters*, 33(1):42–54, 2004.

[14] https://mathworks.com/

[15] https://yalmip.github.io/