IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 18, 2021, accepted March 3, 2021, date of publication March 17, 2021, date of current version April 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3066789

Efficient Computation Offloading in Edge Computing Enabled Smart Home

BOCHENG YU[1], XINGJUN ZHANG[1], ILSUN YOU[2], (Senior Member, IEEE), AND UMER SADIQ KHAN[1]

[1]School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China
[2]Department of Information Security Engineering, Soonchunhyang University, Asan 31538, South Korea

Corresponding author: Ilsun You (ilsunu@gmail.com)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB1001701, and in part by the Soonchunhyang University Research Fund.

ABSTRACT Mobile edge computing which provides computing capabilities at the edge of the radio access network can help smart home reduce response time. However, the limited computing capacity of edge servers is the bottlenecks for the development of edge computing. We integrate cloud computing and edge computing in the Internet of Things to expand the capabilities. Nevertheless, the cost of leasing computing resources has been seldom considered. In this paper, we study the joint transmission power and resource allocation to minimize the users' overhead which is measured by the sum of energy consumption and cost leasing servers. We formulate the problem as a Mixed Integer Linear Programming which is NP-hard and present the Branch-and-Bound to solve it. Due to high complexity, a learning method is proposed to accelerate the algorithm. The branching policy can be learned to reduce the time-cost of the Branch-and-Bound algorithm. Simulation results show our approach can improve the Branch-and-Bound efficiency and performs closely to the traditional branching scheme.

INDEX TERMS Deep learning, integer linear programming, mobile edge computing, smart home, task offloading.

I. INTRODUCTION

With the rapid development of the Internet of things (IoT) [1], the increasing number of smart devices are emerging, such as eldercare and childcare application [2], [3], for the intelligent living environment to bring more convenience to people's lives. However, IoT terminals are usually limited by battery lifetime, computing capabilities, and network connections [4]. At the same time, the IoT devices with limited computation resources have difficulties to deal with the large amount of data generated by smart homes. Therefore, computation offloading is one of the feasible schemes to tackle the above challenges.

Mobile cloud computing (MCC) used to be regarded as a potential approach to solve these challenges [5]. However, mobile and IoT devices may suffer long latency by offloading the computing task. Instead of depending on a remote cloud, a new paradigm called mobile edge computing (MEC) is proposed. The core idea of mobile edge computing is to place computing, storage resources at Internet's edge near the mobile and IoT devices to release the burden on backhaul networks, reduce response latency and promote resource-intensive IoT applications [6]. In particular, idle computing resources, such as laptops, tablets, and smartphones, are deployed at the network edge to process offload tasks from resource-starved devices, e.g., IoT devices. Therefore, edge computing can reduce network bandwidth, response time and the risk of network data leakage [7], [8], which can help improve people's home life in a smart home.

In this paper, we propose a hierarchical network architecture consisting of edge computing and cloud computing to improve the efficiency of cloud resource utilization in a smart home. Our basic idea is to flexibly allocate tasks position to minimize the overhead. For instance, edge computing is a hopeful technology that brings smart home to a new level. A range of smart devices generates a great amount of data stream. It is hard to react promptly to emergencies by transferring data to cloud servers for processing. By offloading critical tasks to the edge server, the latency-sensitive application can get real-time analysis and response. And, non-critical

The associate editor coordinating the review of this manuscript and approving it for publication was Sherali Zeadally.

data is collected and sent to a cloud server for further process. In reality, users also need to consider the lease computing resources (from edge server or cloud server) to meet the offload demand and the energy consumption of a smart home. Hence, we study the joint optimization of user cost and energy of the smart device. It may be the first research on user cost in edge computing so far.

To minimize the overhead and meet the tasks latency requirement, radio resources allocation and server assignment for the offloading task is formulated as a mixed integer linear programming problem (MILP), which is generally considered as NP-hard problem. Although the NP-hard problem can theoretically be solved by exhaustive search, for even small size instances, when the instance size increases, the computational time will be significantly raised, therefore it is hard to find the global optimal solutions [9]. Branch and Bound is seemed as one of the most effective approach to solve MILP problem [10], [11]. The Branch-and-Bound (B&B) algorithm is an iterative algorithm. In each iteration, we solve linear programming (LP) relaxation of each sub-problem and to find an integer feasible solution. However, in each step of the search, the branch of the variable will double the size of the tree resulting in a very large search tree and high computational complexity. An essential challenge of B&B is to appropriately choose variable to branch. Many studies have shown the branching strategy affects the size of the search tree [12], [13]. An effective variable selection scheme may have a huge impact on the search scope of the tree and reduce the time to find the optimal solution. We adopt deep learning to imitate branching policy based on the Strong Branching (SB) approach. Strong branching, can reduce 65% node search compared with hybrid branching, thoroughly test the variables on each node, and select the suitable one based on the gap with the current best feasible solution [14]. Specifically, we study the computation offloading among multiple mobile devices in an integrative network of MEC and MCC, in which we minimize the mobile devices' overhead of cost and energy. We then propose a novel approach to learn branching policy to accelerate the B&B algorithm. To our knowledge, it is the first to joint optimization for devices' cost and energy in multi-layer networks, and attempt to adopt deep learning to mimic variables selection. The main contributions of our paper are as follows,

1) Our work integrates cloud computing and edge computing for computation offloading. We model the overhead of each mobile user as the weighted-sum of the computation cost and energy consumption; We formulate the problem of joint transmission power and resource allocation for minimizing the overhead while complying with the Signal-to-Interference-plus-Noise Ratio (SINR) and task deadline.
2) Because the formulated problem is MILP, the branch and bound is the most effective solution to solve it, while the approach high a time overhead. We present learning branching to improve the performance of B&B. We collected the decisions made by strong branching strategy about variable selection as a training dataset. Next, we train a deep learning model for imitation branching policy. The approach transfers the calculation burden in the training phase to reduce the calculation time.
3) The evaluation shows that our optimization can help mobile users save their costs and decrease energy consumption. Meanwhile, our method improves significantly the B&B algorithm and reduces calculation time over traditional branching policy.

The rest of this paper is structured as follows. In Section 2, related works are surveyed; in Section3, the system model is presented and the performance metric is considered; then in Section 4, we formulate the system as an Integer Linear optimization problem; Section 5 details the proposed solution; simulation results are shown in Section 6. Finally, Section 7 concludes the paper.

II. RELATED WORK

Smart home technology combines sensors, monitors, interfaces, applications, and devices, and realize the automation, localization, and remote control through the network [15]. In recent years, with the development of Internet technology, a large number of novel intelligent applications are emerging. For example, in healthcare, people are monitored by various sensors and actuators, and doctors can make a proper diagnosis and decisions based on the data generated by these devices [16]. At the same time, it also brings further demands for much more computational capabilities and low-latency [17].

Mobile cloud computing, providing computation and storage performance, can help to solve the above problems [18], [19]. In [20], a new application is developed as an integration of a smart home, IoT, and cloud computing. The new benefits of the composite systems are compared to individual components. In [21], a scheme is proposed to manage power consumption monitoring through IoT and cloud computing. In [22], A novel multi-layer cloud architecture is developed to realize the efficient interaction of heterogeneous devices in smart homes based on the Internet of Things, and ontology is adopted as an effective method to alleviate the heterogeneity issues.

However, offloading tasks from large-scale IoT devices will cause backhaul congestion for a cloud server. Furthermore, due to the long transmission latency, cloud computing cannot meet the latency-critical applications [23]. Edge computing as a promising technology can overcome the weakness of traditional cloud computing [24]. Recently, several works have been carried out to study the applying of edge computing to a smart home. Reference [25] designed a method named edgeIoT to solve the scalability problem. The method can effectively distribute packets to reduce the data stream and delay via SDN technology. Aim to the resource limitation of smart home in computing resources, a technology based on containerization is proposed. With the help of

technology, a deep learning model depends on fewer hardware resources [26]. Due to data processed localized, edge computing is also considered a secure paradigm to protect privacy. An intrusion detection system based on a convolutional neural network is presented to categorize traffic and generate intrusion detection dataset [27]. Due to limited computational capability and complex topology, the edge computing network needs to be optimized to maximize its effectiveness. Reference [28] considered a computation offloading problem that aims to minimize the delay and energy consumption. Specifically, the authors formulated the problem as the curse-of-dimensionality and solved it via dynamic programming techniques. In [29], authors designed an energy management strategy of edge computing. The scheme enabled edge computing while reducing the energy consumption up to 86% in a smart home. Reference [30] proposed an offloading algorithm COM to optimize the execution time and energy consumption. However, few studies have considered the cost of using edge servers. This article will consider the joint optimization of energy consumption and cost in edge computing, and the problem is formulated as MILP problems, which are generally considered as NP-hard.

To solve the integer programming, most effective approach is B&B algorithm [31]. B&B alogirthm set up an optimization tree, and each node of the tree denotes an initial problem with integer constraints which is from the relaxed fractional variables. Which fractional variable is selected for branching determines the size of the optimized tree. Meanwhile, it is necessary to find a tradeoff between time spend of decision made by branching strategy and the overall optimization time. Therefore, the branching strategy is one of the important components that determine the efficiency of the B&B algorithm. To reduce the time cost and guarantee the accuracy, machine learning has been introduced to imitate branching strategy. Reference [32] learned the node searching stategy to improve B&B. The approach achieve better performance compared with open-source solvers. In [33], authors use machine learning for configuration of search tree. The experiments show the method can reduce the tree size. Reference [34] adopted machine learning to mimic the branching strategy, which can significantly reduce the size of search tree. However, they applied traditional machine learning algorithms, which need human intervention to decide features of data set. We adopt deep learning for automatic learning of branching strategies to reduce human intervention.

III. SYSTEM MODEL

We consider the integration of MCC and MEC to meet the computation-intensive and latency-sensitive applications in smart home. As depicted in Fig. 1, we denote the set of base station (BS) as $S = \{1, \ldots, S\}$ and the set of mobile users as $N = \{1, 2, \ldots, N\}$, respectively. Each BS $j \in S$ equipped with server of f_j^e (CPU frequency) which provide offloading services for mobile users. Every user has a task, which can be executed either at local or offloading on a server. The decision binary variable is x_{ij}, $\forall\{i, j\} \in \{N, S\}$, where $x_{ij} = 0$ means

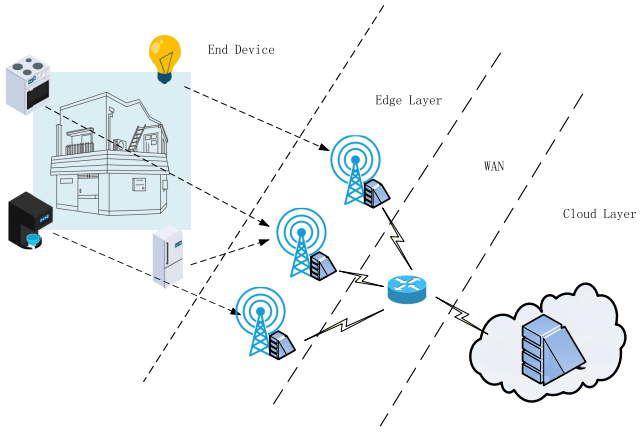

FIGURE 1. The system of edge computing networks.

TABLE 1. Summary of notation.

Symbol	Meaning
N	Set of mobile devices
S	Set of servers
T_n	Set of offloading task
d_n	Size of offloading task
c_n	Task demanded computational resource
l_n	Latency of Task
p_n	Transmission Power
λ^e	Weight of energy
λ^c	Weight of cost
δ_s	Resource unit cost of edge server
δ	Resource unit cost of cloud server
B	Radio spectrum Bandwith
$h_n s$	The channel gain between device n to server s
σ^2	The noise variance
α_i	The fraction of the bandwidth
x_{ij}	Indicates task is processed locally or offloaded through BS j
y_i	Indicates the task offload to edge server or cloud server

that the task is processed at local device, $x_{ij} = 1$ implies that the mobile user i offload the task through the BS j. The task T_n ($n \in N$) cannot be divided into subtask, represented by tuple $\{d_n, c_n, l_n\}$, where d_n denotes input data size, c_n is the workload of the task i.e., the number of CPU cycles to accomplish the task, and l_n is the latency thresholds. The key notations used in the paper are summarized as follow,

A. LOCAL COMPUTING

If the task T_n is executed locally, the decision variable $x_{ij} = 0$. Let f_n^l (CPU cycles per second) represents the computing capability of local device u. The execution time of the task T_n is:

$$t_n^l = \frac{c_n}{f_n^l} \tag{1}$$

When the task is performed on the user device, we calculate the energy consumption via the execution energy model $E = kf^2$ [35], where k is energy coefficient and f is frequency of the CPU. Therefore, the energy consumption for executing the task T_n that requires CPU cycles c_n to complete the task

is:

$$E_n^l = k(f_n^l)^2 c_n, \quad \forall n \in N \tag{2}$$

B. OFFLOAD COMPUTING

Offloading task T_n to servers consists of two main phases: data transmission and task processing. Since the size of the result is often small, we neglect to consider the download step [36]. Each stage of task offloading incurs an additional overhead such as energy consumption, delay, and cost.

1) TRANSMISSION MODEL

In this work, OFDMA is used for uplink between mobile devices and servers, in which the whole spectrum B Hertz is divided into N sub-carrier and each device is assigned to one sub-band. A fraction α_n ($\alpha_n \geq 0$, where $\sum_{n\in N} \alpha_n = 1$) of bandwidth is scheduled for mobile devices to transmit the task. Since the devices offload tasks to the same BS through their allocated sub-band, we assume that each device and BS has only one antenna like [37]. Let p_n and h_{ns} denote the transmission power of user n and channel gain between device n and server s, respectively. In this case, the users also suffer from the inter-cell interference. The Signal-to-interference-plus-noise ratio from mobile device n to server s is:

$$\gamma_u s = \frac{p_n h_{ns}}{\sigma^2} \tag{3}$$

where σ^2 is the noise variance. Since one task only offloading on one sub-band, the achievable transmission rate bits/seconds can be written as,

$$R_n(p_n, \alpha_n) = \alpha_n B \sum_{s\in S} \log_2(1 + \frac{p_n h_{ns}}{\sigma^2}) \tag{4}$$

User n can offload task to only one BS, the constraint of the relationship between user and BS is given as,

$$\sum_j x_{ij} \leq 1 \tag{5}$$

2) LATENCY MODEL

Both MEC servers at the Base Stations and remote cloud server can provide computing service to multi-devices. When a task T_n is offloaded, computational resources $f_n[cycles/s] > 0, n \in N$ is allocated to execute the task and return the output result to the user. Hence, given the computation capability, the execution latency expressed as,

$$t_n^{exe} = \frac{c_n}{f_n} \tag{6}$$

where more allocated resource f_n will decrease the execution time of task T_n, but will increase the cost. The time of offloading task from the user n to the BS s depends on the transmit power and bandwidth. We also assume that BS transmit data to the remote cloud server via backhaul links with enough high bandwidth, and the transmission delay can be ignored [38]. Therefore, the transmission time can be calculated as (where d_n is the data size of task),

$$t_n^{up} = \frac{d_n}{R_n(p_n, \alpha_n)} \tag{7}$$

3) ENERGY MODEL

The consumed energy (millijoule) of offload task T_n equals to product of transmission power and latency [39]–[41], which can be derived as,

$$E_n^{up} = \frac{p_n t_n^{up}}{\varepsilon_n}, \quad \forall n \in N \tag{8}$$

where ε_n is the power amplifier efficiency of user n. Without loss of generality, we assume that $\varepsilon_n = 1$. The transmission energy consumption of user T_n is presented as,

$$E_n^{up}(p_n, \alpha_n) = p_n t_n^{up} = \frac{d_n p_n}{R_n(p_n, \alpha_n)} \tag{9}$$

Therefore, according (2) and (9), we can compute the energy consumption of mobile user as,

$$E_n(x_{ij}, p_n, \alpha_n) = x_{ij} E_n^l + (1 - x_{ij}) E_n^{up}(p_n, \alpha_n) \tag{10}$$

4) COST MODEL

The offloading cost depends on the unit cost of computing resources δ_s at edge server and δ at cloud server. The cost model can prevent the task from being offloaded too concentrated on the edge server with high computation capability and cloud server. For instance, more computing resources will be used in a cloud server, thus $\delta_s < \delta$. Offloading the tasks to remote cloud servers will incur a high cost, which may go over budget. We define binary variable $y_i = \{0, 1\}, i \in N$, where $y_i = 0$ means the task i is offloaded to cloud server; $y_i = 1$ means the task is executed at the edge server. The computational cost can be evaluated as,

$$C_n(x_i, y_i, f_n) = f_n(1 - y_i)\delta + \sum_{j\in S} x_{ij} y_i f_u \delta_s \tag{11}$$

IV. PROBLEM FORMULATION

Our goal is to minimize the sum of energy consumption and cost while meeting the task's deadline. We optimize the parameters of devices, like transmission power, sub-band, and assigned computational resources, to reduce the overhead. This can be formulated as follows,

$$min \quad \sum_{n\in N}((\lambda^e E_n(x_{ij}, p_n, \alpha_n)) + \lambda^c C_n(x_i, y_i, f_n)) \tag{12a}$$

$$\text{s.}t. \quad (1 - x_{ij})t_n^l + x_{ij}(t_n^{up} + t_n^{exe}) \leq l_n, \quad n \in N \tag{12b}$$

$$\sum_{n\in N} \alpha_n = 1 \tag{12c}$$

$$x_{ij} y_i f_u \leq f^t h_s, \quad j \in S \tag{12d}$$

$$x_{ij}, y_i, \in \{0, 1\}, \quad \forall\{i, j\} \in \{N, S\} \tag{12e}$$

$$p_n, \alpha_n, f_n \geq 0, \quad n \in N \tag{12f}$$

where λ^e and λ^c are the weights for energy consumption and cost, respectively. Constraint (10b) meets the latency

of each task. Constraint (10c) means the whole bandwidth is allocated to the users. Constraint (10d) make sure the computational resources of each edge server is respect. It can be seen that the objective (12a) and (12e) indicate the problem as a mixed-integer program, which is hard to solve.

V. PROPOSED SOLUTION

We adopt the B&B algorithm to solve the problem (12). Since the Problem (12) is NP-hard, the B&B algorithm can obtain the global optimal solution through iteration search the binary tree. The lower bound can be found among the feasible solutions in the given region. And we linearize the integer problem by relaxing the constraints to find the upper bound. The basic idea of B&B is to constantly search and update the upper and lower bound of the problem. If the upper and lower value is equal, the algorithm terminates. That is, a feasible solution is optimal. If the upper and lower bound are not equal, the algorithm reselects the node to partition and repeats the search process.

A. OPTIMAL BRANCH-AND-BOUND ALGORITHM

We summarize the B&B algorithm for the formulated problem(12). B&B algorithm sets up a search tree in which the root node is a problem(12). In each iteration, B&B uses a node selection scheme (we adopt Best-bound-first search in this paper [42]) to select a leaf node and then adopt a branching policy to choose a variable x_i to partition. Let Q_i^+ (Q_i^-) denote the reformulated problem with the constraint $x_i = 1$ ($x_i = 0$) as a linear programming problem. Then we solve the linear problem and get the solutions.

1) The optimal solution to Linear programming of Q_i^+ (Q_i^-) is also the feasible solution of original problem, then update the lower bound to the objective value of Q_i^+ (Q_i^-)
2) If the Q_i^+ (Q_i^-) is infeasible, the node will be pruned.
3) The objective vale of Q_i^+ (Q_i^-) is smaller the current lower bound, the node will be pruned.

B. LEARN TO BRANCHING

In the B&B search process in Algorithm 1, there are three main parts: node selection, calculation of solution and variable branching [43]. Most of the time is spent in the branching process, and a good branching strategy can significantly improve the efficiency of the algorithm. The more accurate branching variable is found; the less computational time is spent. Our goal is to find branching variables as accurately as possible to speed up the algorithm's B&B search process. However, the current branching strategy is either spend much more time to calculate or not accurate. In the following, we propose a deep learning method to assist in branch scheme. Deep learning approach transfers the computational burden to the training phase, therefore it can significantly reduce the time cost of B&B algorithm.

The B&B search process can be expressed as a decision-making process. In each search node, we need to determine which variable in the node to choose for branching. The set of all the fractional variables is V. We propose a learning-assisted branching framework to provide an effective and efficient approach. The idea is to apply learning-based methods to help optimize algorithms to solve the most difficult and time-consuming part of optimization. With the increasing of variables, it will spend a lot of time to find a proper variable to partition like Strong Branching policy. We are training a deep neural network (DNN) for Strong Branching. We fed the best solution to train the model. A well-trained DNN can be used to provide a prediction of the branching variable to limit the scope of the search. For instance, the set of the fractional variable V is regarded as input and the branching variable can be obtained as output. The algorithm is shown in Algorithm 2.

Algorithm 1 Optimal B&B Algorithm

initial a search tree T, the root of the tree consists of the problem (12);
Let $L^* = -\infty$ be the objective value
Set list of leaf nodes $= N$
while $N \neq \emptyset$ **do**
  Select a leaf n in N through a *node selection policy*
  if Q corresponding to n is infeasible **then**
    Go to the step 2
  else
    Using *learning method* to choose a variable x_i
    Let Q_i^+ (Q_i^-) be the MILP problem with the constraint $x_i = 1$ ($x_i = 0$)
    for $\hat{Q} \in \{Q_i^+, Q_i^-\}$ **do**
      if $\hat{Q}$ is feasible **then**
        $\bar{x}$ is an optimal solution and $\bar{L}$ is the objective value
        if $L^* < \bar{L}$ **then**
          Set $L^* = \bar{L}$
        else if $L^* > \bar{L}$ **then**
          $\bar{x}$ is not the optimum solution. Prune the branch
        end if
      else
        Prune the node
      end if
    end for
  end if
end while

1) INPUT DATA

The input data set V is obtained from Strong Branching. To get a proper variable x_i from the candidate variable set, consider the node with LP relaxation value Q, with solution L^*. The two subproblem of varible i are Q_i^+ and Q_i^-, which have the LP value L_i^+ and L_i^-, respectively. If the Q_i^+ (Q_i^-), is infeasible, we set the L_i^+ (L_i^-) to a very large value. The changes of the objective value

Algorithm 2 Learning-Assisted Branching Policy

Input: The set of fractional variable V
The current lower bound L^*
1: Training Phase:
2: Generate training set, validation set and test set from Strong Branching
3: Training the DNN
4: Online Operation Phase:
5: **while** $V \neq \emptyset$ **do**
6: Branch on the variable and solve the LP relaxation problem
7: **if** objective value < lower bound or no feasible solution **then**
8: Prune the branch, and choose another variable
9: **else**
10: Update the lower bound to objective value. If objective value equals upper bound, stop and obtain the optimum solution
11: **end if**
12: **end while**
Output: The best solution of the node

is $\delta_i^- = L_i^- - L^*$ and $\delta_i^+ = L_i^+ - L^*$.

$$SB_i = sore(max(\delta_i^-, \epsilon), max(\delta_i^+, \epsilon)) \tag{13}$$

where ϵ is small constant (e.g. 10^{-6}). A product is adopted for $sore(a, b) = a \times b$, SB try to find the variable with maximum score.

2) OUTPUT DATA

The learning model is aimed to forecast the probability of each candidate variable to get the best feasible solution. hence, the output layer can be viewed as vector with the same length as the number of fraction variable of the node.

For the output layer, we just need only one variable, we use binary value to define the output vector, whis is,

$$y = \begin{cases} 1 & \textit{the variable is used to branch} \\ 0 & \textit{otherwise} \end{cases} \tag{14}$$

The binary value indicates that we focus more on whether a variable will be selected to partition. For our learning model, the length of output vector corresponds to the input data. If there are N input variables, the output vector will be a N dimensional vector, where only 1 element represents the suitable variable, and all other elements are 0. We assume a DNN network with a 3-layer network, and the number of neurons in each layer is N, N_1, and N. Taking a sample $N * 1$ for training, two matrix multiplications require $N * N_1$ and $N_1 * N$ calculations respectively. The time complexity of Forward propagation for a sample should be $O(N_1)$. In the Back propagation stage, assuming there are a total of V training samples, and each sample is trained only once, the time complexity of training a neural network should be $O(V * N_1)$.

3) TEST SAMPLES

We generated the test samples using SB policy. The value of output unit is between 0 and 1 in our model. The output value is represented to probability distribution via normalization or softmax function. The transformed values indicate the possibilities that the variable can be partitioned to obtain the best feasible solution. In our problem, the output can be used to express the probability of whether a variable will be branched and the sum of output values equals 1. We can choose the maximum value of the variable as the suitable one. We compare the selected variable with the test samples, and adjust the parameters based on the result.

VI. PERFORMANCE EVALUATION

In this section, simulation results are presented to evaluate the performance following different scenarios. We consider a hierarchical network architecture consisting of edge computing and cloud computing. There are 6 mobile devices which can offload the task to 4 edge servers or cloud server. The bandwidth is randomly divided into n sub-channels. The main simulation parameters are presented in Table 2.

TABLE 2. Simulation parameters.

Notation	Description	Value
N	The set of mobile user	6
S	The set of servers	4
d_n	The size of task	300 Kilobits
c_n	Task computational demand	120 Megacycles
l_n	Latency requirement	50 milliseconds
p_i	Transmission Power of device	23dBm [44]
δ	Computational resource unit cost of edge server	6 cent/gigahertz
δ_s	Computational resource unit cost of cloud server	3 cent/gigahertz
B	Bandwidth	20 MHz
σ^2	Noise	-120 dBm/Hertz [45]
f^e	The maximum CPU frequency of edge server	10 GHz
f^l	The CPU frequency of local device	1 GHz

We adopt IBM ILOG CPLEX 12.6.1 to implement various strategies using control callbacks, in single-thread mode. In this paper, we compare the branching scheme based on deep learning with 3 strategies, which is shown in Table3. CPLEX is the default variable selection of the solver. SB refers to Strong Branching, and PC refers to pseudocost branching [46]. From Table 1, we can see that the computation time based on our optimal algorithm is considerably reduced than the computation time of the conventional optimization algorithm. And as the scale of the instance increases, the computation time of the conventional algorithm increases significantly, while the time increase of the algorithm based on our algorithm is not obvious.

TABLE 3. Time cost (seconds) of branching strategies.

Mobile Devices	Servers	SB	PC	CPLEX	SB+DL
15	10	2.4	1.6	1.6	0.7
20	10	2.7	1.7	1.5	0.65
20	15	3.4	1.95	1.83	0.58
25	10	3.64	2.31	1.96	0.72
25	15	4.0	2.26	1.89	0.68

In Fig. 2, we compare the convergence performance between the B&B algorithm and the Deep learning assisted

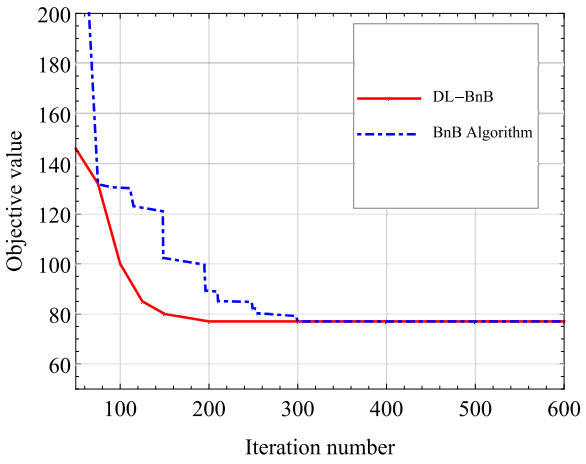

FIGURE 2. Convergence between the B&B and Deep learning method.

method. There are 6 mobile devices and 4 edge server in the instance. As expected, the deep learning-based algorithm takes about 100 iterations to keep steady. The traditional B&B with the default sets of CPLEX spends around 300 iterations to converge, which means B&B slowly obtain the optimal solution.

In Fig. 3, we study the objective value with different latency threshold and the computational capacity of the edge server. First, increasing the latency threshold results in a decrease in the objective value. The reason is that mobile devices will consume more computational resources to reduce its task execution time to keep up with the large threshold, which will incur higher costs. Additionally, increasing the latency threshold will allow users to use fewer resources at the server, thereby reducing overall costs. Secondly, the objective value decreases as the edge servers have a high computational capacity. The edge server with high computational capability will afford more resources and lower cost (Compared with cloud server) to mobile devices, which reduces the need to offload tasks to the cloud server.

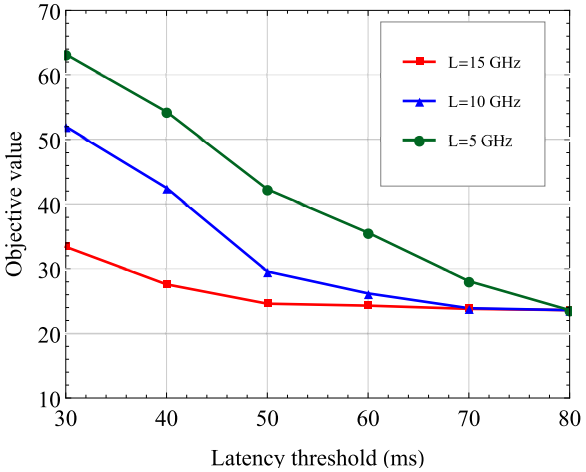

FIGURE 3. Comparison of latency threshold.

Meanwhile, edge servers with less computational resources are easy congestion, thereby mobile devices have to seek more computation service at a high cost. That is why the gap between curves gets smaller and overlap under the high latency.

In Fig. 4, we study the objective value under a different number of devices. First, the objective value is advanced with the growth of devices. This is caused by the increase in the number of resources: energy and computing. Additionally, it means more devices offload tasks to servers, which incurs higher costs. Secondly, adding more edge servers may be useful to address high device requests in dense areas, which will decrease cost by reducing the high utilization of the cloud server.

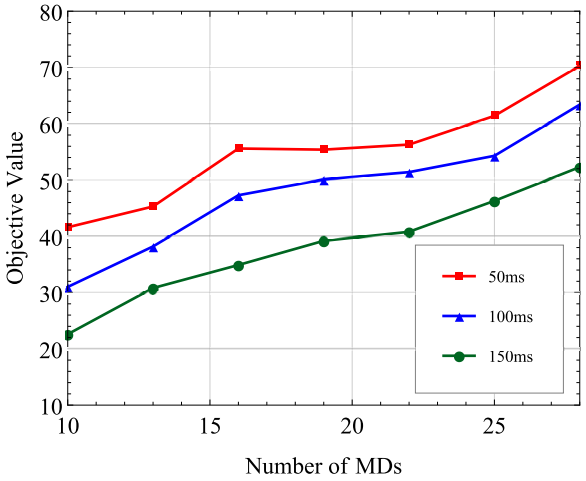

FIGURE 4. Comparison of number of devices.

As show in Fig. 5, we show the changes in the weight λ^c affect the cost. Specifically, we compare the cost under different λ^c. As the λ^c increase, the cost on each device decrease. A higher weight is given to the cost goal, the function will give priority to allocating devices with small computational resources with lower unit costs, thereby reducing the overall cost. But on the other hand, this may lead to an increase in energy consumption in terms of wireless channels. The channel state is not the best to communicate with the edge server of low capability, so the devices will be forced to increase its transmission power to keep up with the required deadline of the task, resulting in higher energy consumption in the process.

The Fig. 6 shows the impact of weight λ^e on energy consumption. We vary the weight λ^e and study the change on energy consumption of device under different values of the latency. As we can see, with decrease in λ^e, the energy consumption increase. These observations are due to the fact that when the weight λ^e on sub-target energy of objective function is increased, the optimization solutions will make device to choose a base station with better channel conditions to reduce the transmission power, therefore the device's energy consumption is reduced.

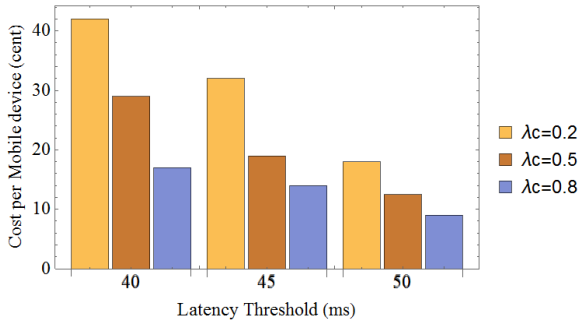

FIGURE 5. Comparison of cost of computational resource.

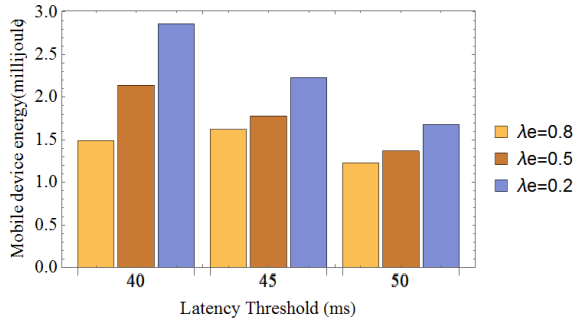

FIGURE 6. Comparison of energy consumption.

VII. CONCLUSION

We study the cost and energy efficiency of task offloading in a hierarchical network architecture that consists of edge computing and cloud computing. To the best of our knowledge, it is the first time to study the problem of cost and energy efficiency in a multi-layer network for smart home. We expoit a learning approach to improve the B&B algorithm to solve the formulated MILP problem. The learning method addresses the variable selection issue to speed up the most time-consuming process of the B&B algorithm. Simulation experiment showed the results of our proposed algorithm closely to the B&B algorithm and significantly reduce the calculation time. Meanwhile, we evaluated the key factors in energy and cost of task offloading in different environments.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.
[2] L. C. De Silva, C. Morikawa, and I. M. Petra, "State of the art of smart homes," *Eng. Appl. Artif. Intell.*, vol. 25, no. 7, pp. 1313–1321, Oct. 2012.
[3] Y. A. Shichkina, G. V. Kataeva, Y. A. Irishina, and E. S. Stanevich, "The use of mobile phones to monitor the status of patients with parkinson's disease," *J. Wireless Mobile Netw., Ubiquitous Comput., Dependable Appl.*, vol. 11, no. 1, pp. 55–73, 2020.
[4] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of MEC in the Internet of Things," *IEEE Consum. Electron. Mag.*, vol. 5, no. 4, pp. 84–91, Oct. 2016.
[5] I. Kholod, A. Shorov, and S. Gorlatch, "Efficient distribution and processing of data for parallelizing data mining in mobile clouds," *J. Wireless Mobile Netw., Ubiquitous Comput., Dependable Appl.*, vol. 11, no. 1, pp. 2–17, 2020.
[6] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge cloud offloading algorithms: Issues, methods, and perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–23, Feb. 2019.
[7] H. Kim, "5G core network security issues and attack classification from network protocol perspective," *J. Internet Services Inf. Secur.*, vol. 10, no. 2, pp. 1–15, 2020.
[8] M. Alizadeh, K. Andersson, and O. Schelén, "A survey of secure Internet of Things in relation to blockchain," *J. Internet Services Inf. Secur.*, vol. 10, no. 3, pp. 47–75, 2020.
[9] G. J. Woeginger, "Exact algorithms for NP-hard problems: A survey," in *Combinatorial Optimization—Eureka, You Shrink*. Berlin, Germany: Springer, 2003, pp. 185–207.
[10] A. H. Land and A. G. Doig, "An automatic method for solving discrete programming problems," in *50 Years of Integer Programming 1958-2008*. Berlin, Germany: Springer, 2010, pp. 105–132.
[11] A. N. Letchford and A. Lodi, "An augment-and-branch-and-cut framework for mixed 0-1 programming," in *Combinatorial Optimization—Eureka, You Shrink*. Heidelberg, Germany: Springer, 2003, pp. 119–133.
[12] T. Achterberg. (2007). *Constraint Integer Programming*. [Online]. Available: http://opus.kobv.de/tuberlin/volltexte/2007/1611/
[13] T. Achterberg and R. Wunderling, "Mixed integer programming: Analyzing 12 years of progress," in *Facets of Combinatorial Optimization*. Berlin, Germany: Springer, 2013, pp. 449–481.
[14] T. Achterberg and T. Berthold, "Hybrid branching," in *Proc. Int. Conf. AI OR Techn. Constriant Program. Combinat. Optim. Problems*. Berlin, Germany: Springer, 2009, pp. 309–311.
[15] D. J. Cook, "How smart is your home?" *Science*, vol. 335, no. 6076, pp. 1579–1581, Mar. 2012.
[16] H. Wang, J. Gong, Y. Zhuang, H. Shen, and J. Lach, "Healthedge: Task scheduling for edge computing with health emergency and human behavior consideration in smart homes," in *Proc. Int. Conf. Netw., Archit., Storage (NAS)*, Aug. 2017, pp. 1213–1222.
[17] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the edge: A scalable IoT architecture based on transparent computing," *IEEE Netw.*, vol. 31, no. 5, pp. 96–105, Sep. 2017.
[18] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, no. 18, pp. 1587–1611, Dec. 2013.
[19] E. Ahmed, A. Gani, M. Sookhak, S. H. A. Hamid, and F. Xia, "Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges," *J. Netw. Comput. Appl.*, vol. 52, pp. 52–68, Jun. 2015.
[20] M. Domb, "Smart home systems based on Internet of Things," in *Internet Things (IoT) for Automated Smart Applications*. Rijeka, Croatia: Intech, 2019.
[21] E. Orsi and S. Nesmachnow, "Smart home energy planning using IoT and the cloud," in *Proc. IEEE URUCON*, Oct. 2017, pp. 1–4.
[22] M. Tao, J. Zuo, Z. Liu, A. Castiglione, and F. Palmieri, "Multi-layer cloud architectural model and ontology-based security service framework for IoT-based smart homes," *Future Gener. Comput. Syst.*, vol. 78, pp. 1040–1051, Jan. 2018.
[23] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.
[24] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, May 2019.
[25] X. Sun and N. Ansari, "EdgeIoT: Mobile edge computing for the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 22–29, Dec. 2016.
[26] N. Gupta, K. Anantharaj, and K. Subramani, "Containerized architecture for edge computing in smart home: A consistent architecture for model deployment," in *Proc. Int. Conf. Comput. Commun. Inform. (ICCCI)*, Jan. 2020, pp. 1–8.
[27] A. S. Almogren, "Intrusion detection in edge-of-things computing," *J. Parallel Distrib. Comput.*, vol. 137, pp. 259–265, Mar. 2020.
[28] L. Lei, H. Xu, X. Xiong, K. Zheng, and W. Xiang, "Joint computation offloading and multiuser scheduling using approximate dynamic programming in NB-IoT edge computing system," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5345–5362, Jun. 2019.
[29] X. Chang, W. Li, C. Xia, J. Ma, J. Cao, S. U. Khan, and A. Y. Zomaya, "From insight to impact: Building a sustainable edge computing platform for smart homes," in *Proc. IEEE 24th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2018, pp. 928–936.
[30] X. Xu, Q. Liu, Y. Luo, K. Peng, X. Zhang, S. Meng, and L. Qi, "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Gener. Comput. Syst.*, vol. 95, pp. 522–533, Jun. 2019.
[31] T. Achterberg, T. Koch, and A. Martin, "Branching rules revisited," *Oper. Res. Lett.*, vol. 33, no. 1, pp. 42–54, Jan. 2005.

[32] H. He, H. Daume, and J. M. Eisner, "Learning to search in branch and bound algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 3293–3301.
[33] M.-F. Balcan, T. Dick, T. Sandholm, and E. Vitercik, "Learning to branch," 2018, *arXiv:1803.10150*. [Online]. Available: http://arxiv.org/abs/1803.10150
[34] E. B. Khalil, P. Le Bodic, L. Song, G. Nemhauser, and B. Dilkina, "Learning to branch in mixed integer programming," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 1–8.
[35] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.
[36] S. Josilo and G. Dan, "A game theoretic analysis of selfish mobile computation offloading," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.
[37] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.
[38] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.
[39] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.
[40] X. Lyu, H. Tian, W. Ni, Y. Zhang, P. Zhang, and R. P. Liu, "Energy-efficient admission of delay-sensitive tasks for mobile edge computing," *IEEE Trans. Commun.*, vol. 66, no. 6, pp. 2603–2616, Jun. 2018.
[41] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, "TOFFEE: Task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing," *IEEE Trans. Cloud Comput.*, early access, Jun. 20. 2019, doi: 10.1109/TCC.2019.2923692.
[42] T. Ibaraki, "Theoretical comparisons of search strategies in branch-and-bound algorithms," *Int. J. Comput. Inf. Sci.*, vol. 5, no. 4, pp. 315–344, Dec. 1976.
[43] J. Clausen, "Branch and bound algorithms-principles and examples," Dept. Comput. Sci., Univ. Copenhagen, København, Denmark, Tech. Rep., 1999, pp. 1–30.
[44] S. M. Lopez, "An overview of D2D in 3GPP LTE standard," in *Proc. Indofrench Workshop D2D Commun. 5G IoT Netw.*, 2016, pp. 1–34.
[45] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 1451–1455.
[46] J. T. Linderoth and M. W. P. Savelsbergh, "A computational study of search strategies for mixed integer programming," *INFORMS J. Comput.*, vol. 11, no. 2, pp. 173–187, May 1999.

BOCHENG YU received the M.S. degree in software engineering from the University of Southampton, U.K., in 2011. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Xi'an Jiaotong University. His current research interests include mobile edge computing, network optimization, and machine learning.

XINGJUN ZHANG received the Ph.D. degree in computer architecture from Xi'an Jiaotong University, China, in 2003. From January 2004 to December 2005, he was a Postdoctoral Fellow with the School of Computer, Beihang University, China. From February 2006 to January 2009, he was a Research Fellow with the Department of Electronic Engineering, Aston University, U.K. He is currently a Full Professor and the Dean of the School of Computer Science and Technology, Xi'an Jiaotong University. His research interests include high performance computing, big data storage systems, and machine learning acceleration.

ILSUN YOU (Senior Member, IEEE) received the M.S. and Ph.D. degrees in computer science from Dankook University, Seoul, South Korea, in 1997 and 2002, respectively, and the second Ph.D. degree from Kyushu University, Japan, in 2012. From 1997 to 2004, he was with Thin Multimedia Inc., Internet Security Company Ltd., and Hanjo Engineering Company Ltd., as a Research Engineer. He is currently an Associate Professor with the Department of Information Security Engineering, Soonchunhyang University. His current research interests include the Internet security, authentication, access control, and formal security analysis. He is also a Fellow of the IET (based on document published on September 2019). He has been serving as a Main Organizer for international conferences and workshops, such as MobiWorld, MIST, SeCIHD, AsiaARES, and IMIS. He is also the Editor-in-Chief of the *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications* (JoWUA). He is also on the Editorial Board of *Intelligent Automation and Soft Computing* (AutoSoft), the *Journal of Network and Computer Applications* (JNCA), the *International Journal of Ad Hoc and Ubiquitous Computing* (IJAHUC), *Computing and Informatics* (CAI), *Journal of High Speed Networks* (JHSN), and *Security and Communication Networks* (SCN).

UMER SADIQ KHAN received the Master of Computer Science degree from the Kohat University of Science and Technology Kohat, Pakistan, in 2009. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, China. His research interests include computer vision, image processing, pattern recognition, and computer graphics.