Work-in-Progress: Deadline-Constrained Multi-Resource Allocation in Edge-Cloud System

Chuanchao Gao, Arvind Easwaran
School of Computer Science and Engineering, *Nanyang Technological University*, Singapore
gaoc0008@e.ntu.edu.sg, arvinde@ntu.edu.sg

***Abstract*—In an edge-cloud system, end devices can offload computation intensive tasks to servers for processing, to satisfy deadline requirements of time-critical tasks, or maintain a good quality of service. Because the system has limited bandwidth and computation resource, it can be very challenging to determine where tasks should be offloaded and processed (task mapping), and how much bandwidth and computation resource should be allocated to each task (resource allocation). In this paper, we propose a task mapping and multi-resource allocation problem with both communication and computation contentions in an edge-cloud system, which aims to maximize the total profit gained by the system while meeting the deadlines of mapped tasks. Besides, the backhaul network of the proposed edge-cloud system is modeled as a directed incomplete graph with bandwidth contention on every edge of the graph. We formulate the problem into a nonconvex Mixed-Integer Nonlinear Programming (MINLP) problem and provide a linearization method to reformulate the MINLP problem into an Integer Linear Programming (ILP) problem formulation, which can be solved with ILP solvers.**

***Index Terms*—deadline requirement, multi-resource allocation, communication and computation contentions**

I. Introduction

With the rapid development of the Internet of Things and Artificial Intelligence technologies, end devices are required to process time-critical computation intensive tasks. However, limited by the computation resource and power supply, end devices become less capable of meeting the deadlines of such tasks to guarantee safety or quality of service. The edge-cloud system is proposed to assist end devices to handle computation intensive tasks. In an edge-cloud system, computation intensive tasks can be offloaded from end devices to edge servers through a wireless network. Upon receiving tasks from end devices, edge servers can either process these tasks by themselves, or forward these tasks to other servers for processing through the backhaul network.

In this paper, we focus on the deadline-constrained task mapping and resource allocation problem in an edge-cloud system, with both communication and computation contentions. *Task mapping* determines where each task is offloaded and processed, and the path in the backhaul network to transmit the task from its offloading server to its processing server. *Resource allocation* determines how much bandwidth and computation resource will be allocated to each task by servers and the backhaul network. Each task can contribute a pre-defined profit to the system when its deadline requirement is met, and this problem aims to maximize the system total profit. Due to the limited bandwidth of wireless/backhaul networks and the limited computation resource of servers, it can be very difficult to determine both task mapping and resource allocation, while meeting task deadlines.

Studies on this problem have generally considered a simplified backhaul network modeling without bandwidth contention [1]–[4]. Li *et al.* [2] and Vu *et al.* [3] assumed that the backhaul network was a complete graph. Yang *et al.* [1] assumed that the bandwidth allocation was constant for all tasks in the backhaul network. Gao *et al.* [4] assumed that the data transmission delay between a given server pair was constant in the backhaul network, regardless of the transmitted data size. These simplified modelings of the backhaul network make the task mapping and resource allocation problem easier to solve, but such settings can also limit the application of the edge-cloud system in real-world systems.

In this paper, we propose an edge-cloud system with a more general and practical backhaul network setting, where the backhaul network is modeled as a directed incomplete graph. In our system, each server is connected to a few nearby servers, and a unidirectional data transmission mode is used. Thus, between any connected server A and server B, there exists one channel for transmitting data from A to B, and another channel for transmitting data from B to A. These two opposite channels are modeled as two directed edges in the backhaul network graph. Each edge of the graph has a bandwidth capacity and can allocate varying bandwidth to tasks. Because a task is not necessarily processed on its offloading server, and the backhaul network graph is not a complete graph, multiple paths will exist from its offloading server to its processing server, and one or more directed edges might exist in each path.

We first formulate the profit maximization problem into a nonconvex Mixed-Integer Nonlinear Programming (MINLP) problem. By assuming that minimum resource units exist, and any resource allocation is an integer multiple of the corresponding minimum resource unit, we reformulate the MINLP problem into an Integer Linear Programming (ILP) problem, which can be solved with existing ILP solvers.

II. System Model and Problem Formulation

A. Edge-Cloud System Model

The multi-layer edge-cloud system comprises end devices, edge and cloud servers, as shown in Fig 1. *End Devices* are units that have specific functionalities and can communicate

This work was supported by the MoE Tier-2 grant MOE-T2EP20221-0006.

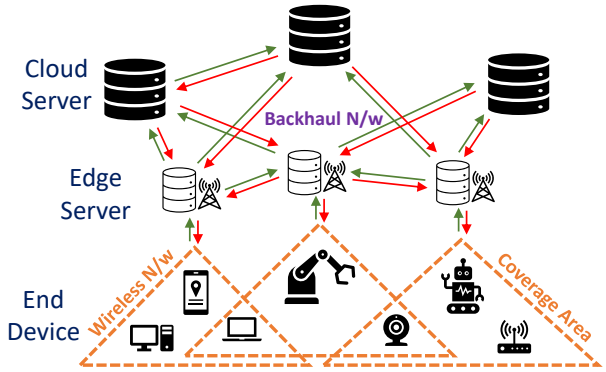

Fig. 1. Edge-Cloud System

with nearby edge servers through the wireless network. Denote the set of tasks generated by end devices as $\mathcal{I}$, and each task $i \in \mathcal{I}$ is associated with four parameters: $\{s_i, q_i, \Delta_i, g_i\}$. s_i is the required data size for task i offloading, q_i is the number of CPU cycles needed to process task i, Δ_i is the deadline of task i, and g_i is the profit gained by completing task i before its deadline. If task i misses its deadline, the system will not gain any profit from this task. For example, for a moving autonomous vehicle, the object detection task should be completed within a predefined time window such that the vehicle has enough time to take actions. The profit can be considered as the priority for time-critical tasks. We assume that tasks cannot be split, *i.e.*, each task must be entirely offloaded to one edge server and processed on one edge/cloud server. We further assume that tasks can only be processed on servers, thus all tasks must be offloaded to be processed.

Edge Servers are deployed near the end devices to provide instant responses to the requests of end devices. Edge servers can collect tasks from the end devices within their coverage area through a wireless network, and either process these tasks by themselves, or forward these tasks to other servers for processing through the backhaul network. We denote the set of edge servers as $\mathcal{J}$. Each task is covered only by a few nearby edge servers, and we denote the subset of edge servers to which task i can be offloaded as $\mathcal{J}_i$. In a wireless network, there exist an uplink for transmitting data from end devices to the edge server and a downlink for transmitting data from the edge server to end devices. We denote the uplink bandwidth capacity of the edge server $j \in \mathcal{J}$ as b_j.

Cloud Servers are servers deployed far away from end devices, which can cause a significant data transmission delay when tasks are processed on cloud servers. However, cloud servers usually have a significantly larger computation resource capacity, which can enhance the capability of an edge-cloud system in handling computation intensive tasks. Cloud servers can only communicate with other servers through the backhaul network. We denote the set of servers, including the edge and cloud servers, as $\mathcal{K}$. Note that $\mathcal{J} \subset \mathcal{K}$. We denote the computation resource capacity of a server $k \in \mathcal{K}$ as c_k.

Each server is directly connected with few nearby servers, and we denote the communication network among all servers as the *backhaul network*. We assume that the data transmission is unidirectional. Thus, the backhaul network is modeled as a directed incomplete graph $\mathcal{G} = (\mathcal{K}, \mathcal{E})$. The node set of $\mathcal{G}$ consists of all servers. For directly connected server k_1 and server k_2, both edge (k_1, k_2) and edge (k_2, k_1) are included in edge set $\mathcal{E}$. Each edge $e \in \mathcal{E}$ has a bandwidth capacity, denoted as b_e. The number of possible acyclic paths from one server to another server can be exponential in the number of server nodes in graph $\mathcal{G}$. Thus, we consider only the paths that contain no more than M edges, where M is a constant. We denote the set of such paths as $\mathcal{P}$, which can be predetermined with a breadth-first search algorithm. When $M = 1$, it means a task can only be processed on its offloading server or the adjacent servers of its offloading server. We denote the set of paths from server k_1 to server k_2 as $\mathcal{P}_{k_1 k_2}$, where $\mathcal{P}_{k_1 k_2} \in \mathcal{P}$. Besides, we set $\mathcal{P}_{kk} = \{\emptyset\}, \forall k \in \mathcal{K}$, which means that no edges inside the path from server k to itself. We also denote the set of paths in $\mathcal{P}$ which contain edge $e \in \mathcal{E}$ as $\mathcal{P}_e$.

TABLE I
NOTATION (PARAMETERS AND VARIABLES)

Notation	Definition
$\mathcal{I}$	Taskset, where $i \in \mathcal{I}$ denotes a task
$\mathcal{J}$	Edge server set, where $j \in \mathcal{J}$ denotes an edge server
$\mathcal{J}_i$	Edge server subset to which task i can be offloaded
$\mathcal{K}$	Server set, $k \in \mathcal{K}$ denotes an edge or cloud server, $\mathcal{J} \subset \mathcal{K}$
$\mathcal{G}$	$\mathcal{G} = (\mathcal{K}, \mathcal{E})$, the directed backhaul network graph
$\mathcal{E}$	Edge set of graph $\mathcal{G}$, where $e \in \mathcal{E}$ denotes an edge
$\mathcal{P}$	Set of paths from servers to servers with no more than M edges, where M is a constant
$\mathcal{P}_{k_1 k_2}$	Set of paths from server k_1 to server k_2, where $p \in \mathcal{P}_{k_1 k_2}$ denotes one feasible path
$\mathcal{P}_e$	Set of paths in $\mathcal{P}$ that contain edge $e \in \mathcal{E}$
s_i	Required data size for task i offloading
q_i	CPU cycles required to process task i
Δ_i	Deadline of task i
g_i	Profit gained by completing task i within its deadline
b_j	Uplink bandwidth capacity of edge server $j \in \mathcal{J}$
b_e	Bandwidth capacity of edge $e \in \mathcal{E}$
c_k	Computation resource capacity of server $k \in \mathcal{K}$
b_{ij}	Variable for wireless uplink bandwidth allocated to task i by edge server j
b_{ie}	Variable for bandwidth allocated to task i by edge e in the backhaul network
c_{ik}	Variable for computation resource allocated to task i by server k
x_{ij}	Binary offloading decision variable, $x_{ij} = 1$ if task i is offloaded to edge server j
y_{ik}	Binary processing decision variable, $y_{ik} = 1$ if task i is processed on server k
z_{ip}	Binary path decision variable, $z_{ip} = 1$ if task i is transmitted from server k_1 to server k_2 through path $p \in \mathcal{P}_{k_1 k_2}$ in the backhaul network

B. Problem Formulation

For task mapping, we use a binary variable x_{ij} to denote the offloading decision of task i, where $x_{ij} = 1$ only when task i is offloaded to edge server $j \in \mathcal{J}_i$. Similarly, we use a binary variable y_{ik} to denote the processing decision of task i, where $y_{ik} = 1$ only when task i is processed on server $k \in \mathcal{K}$. Besides, we use a binary variable z_{ip} to denote the path decision of task i, and $z_{ip} = 1$ only when task i is

transmitted from its offloading server k_1 to its processing server k_2 through path $p \in \mathcal{P}_{k_1k_2}$ in the backhaul network. For resource allocation, we use the variable b_{ij} to denote the amount of bandwidth that will be allocated to task i by edge server $j \in \mathcal{J}$, and use the variable c_{ik} to denote the amount of computation resource that will be allocated to task i by server $k \in \mathcal{K}$. Besides, we use the variable b_{ie} to denote the bandwidth resource that will be allocated to task i by edge $e \in \mathcal{E}$. The notations used in this paper is summarized in Table I.

The total time to complete task i, denoted as T_i, consists of four parts: task offloading time T_i^o, data transmission time in the backhaul network T_i^c, task processing time T_i^p, and result return time T_i^r. The result return time T_i^r includes the time spent in both the backhaul network and the wireless network downlink. The data size of the returning result is usually negligible, thus, the time spent in the backhaul network and the wireless network downlink can be ignored. We assume $T_i^r = 0, \forall i \in \mathcal{I}$, in this paper. If task i is offloaded to edge server j, processed on server k, and path $p \in \mathcal{P}_{jk}$ is chosen for data transmission in the backhaul network, then $T_i^o = s_i/b_{ij}$, $T_i^c = \sum_{e\in p} s_i/b_{ie}$, and $T_i^p = q_i/c_{ij}$. Note that when $j = k$, $p = \emptyset$ and $T_i^c = \sum_{e\in\emptyset} s_i/b_{ie} = 0$. The total task completion time is

$$T_i = T_i^o + T_i^c + T_i^p + T_i^r = \frac{s_i}{b_{ij}} + \sum_{e\in p} \frac{s_i}{b_{ie}} + \frac{q_i}{c_{ij}}.$$

The deadline-constrained task mapping $\{x_{ij}, y_{ik}, z_{ip}\}$ and resource allocation $\{b_{ij}, c_{ik}, \{b_{ie} : e \in \mathcal{E}\}\}$ problem $(\mathbf{P}_0)$, which aims to maximize the total system profit, can be formulated as follows.

$$(\mathbf{P}_0) \quad \max \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}_i} \sum_{k\in\mathcal{K}} \sum_{p\in\mathcal{P}_{jk}} x_{ij}y_{ik}z_{ip}g_i \tag{1}$$

subject to:

$$\sum_{j\in\mathcal{J}_i} x_{ij}\frac{s_i}{b_{ij}} + \sum_{j\in\mathcal{J}_i}\sum_{k\in\mathcal{K}}\sum_{p\in\mathcal{P}_{jk}}\sum_{e\in p} x_{ij}y_{ik}z_{ip}\frac{s_i}{b_{ie}} + \sum_{k\in\mathcal{K}} y_{ik}\frac{q_i}{c_{ik}} \le \Delta_i, \quad \forall i \in \mathcal{I} \tag{1a}$$

$$\sum_{j\in\mathcal{J}_i} x_{ij} \le 1, \quad \forall i \in \mathcal{I} \tag{1b}$$

$$\sum_{j\in\mathcal{J}\setminus\mathcal{J}_i} x_{ij} = 0, \quad \forall i \in \mathcal{I} \tag{1c}$$

$$\sum_{k\in\mathcal{K}} y_{ik} \le 1, \quad \forall i \in \mathcal{I} \tag{1d}$$

$$\sum_{j\in\mathcal{J}_i}\sum_{k\in\mathcal{K}}\sum_{p\in\mathcal{P}_{jk}} z_{ip} \le 1, \ \forall i \in \mathcal{I} \tag{1e}$$

$$\sum_{j\in\mathcal{J}_i}\sum_{k\in\mathcal{K}}\sum_{p\in\mathcal{P}\setminus\mathcal{P}_{jk}} x_{ij}y_{ik}z_{ip} = 0, \ \forall i \in \mathcal{I} \tag{1f}$$

$$\sum_{i\in\mathcal{I}} x_{ij}b_{ij} \le b_j, \quad \forall j \in \mathcal{J} \tag{1g}$$

$$\sum_{i\in\mathcal{I}} y_{ik}c_{ik} \le c_k, \quad \forall k \in \mathcal{K} \tag{1h}$$

$$\sum_{i\in\mathcal{I}}\sum_{p\in\mathcal{P}_e} z_{ip}b_{ie} \le b_e, \ \forall e \in \mathcal{E} \tag{1i}$$

$$x_{ij}, y_{ik}, z_{ip} \in \{0,1\}, \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \forall k \in \mathcal{K}, \forall p \in \mathcal{P} \tag{1j}$$

The objective function of problem $\mathbf{P}_0$ indicates that the system can gain the task profit only when the task is offloaded, transmitted through a valid backhaul network path and processed. Constraint (1a) is the deadline constraint, which guarantees that the total completion time of any task cannot exceed its deadline. Note that when a task is offloaded to edge server j and processed on the same server, the path $p = \emptyset$, and the time spent in the backhaul network is also 0. Constraints (1b) $\sim$ (1f) ensure that each task can be offloaded to at most one edge server, transmitted through at most one path in the backhaul network, and processed on at most one server. Constraints (1g) and (1i) are the bandwidth constraints of the edge servers and the backhaul network edges, and constraint (1h) is the computation resource constraint of all servers.

Because the second derivatives of the terms $x_{ij}\frac{1}{b_{ij}}$ and $y_{ik}\frac{1}{c_{ik}}$ are negative, constraint (1a) is a nonconvex constraint. Besides, because terms $x_{ij}y_{ik}z_{ip}, x_{ij}b_{ij}, y_{ik}c_{ik}, z_{ip}b_{ie}$ are nonlinear, and variables x_{ij}, y_{ik}, z_{ip} are binary variables, problem $\mathbf{P}_0$ is a nonconvex MINLP problem.

III. Methodology

In problem $\mathbf{P}_0$, we must determine both the decision binary variable (x_{ij}, y_{ik}, z_{ip}) and the continuous resource allocation variable $(b_{ij}, c_{ik}, \{b_{ie} : e \in \mathcal{E}\})$, while satisfying the task end-to-end deadlines. In the proposed edge-cloud system, the bandwidth contention exists in edge servers and the backhaul network edges, and the computation resource contention exists in all servers, which make problem $\mathbf{P}_0$ very difficult to solve.

Nonconvex MINLP problems are more complex than convex MINLP problems, and convex MINLP problems are more complex than ILP problems [5]. To solve problem $\mathbf{P}_0$, we can first reformulate (or convert) problem $\mathbf{P}_0$ into an ILP problem. We can use the linearization method introduced by Gao *et al.* [4], where they assumed that the minimum resource units existed, and any resource allocation was the integer multiple of the corresponding minimum resource unit.

We use $\tilde{b}$ and $\tilde{c}$ to denote the minimum bandwidth and computation resource units, and assume that any resource allocation is an integer multiple of the corresponding minimum resource unit. Let $u_j = \lfloor b_j/\tilde{b} \rfloor$, $u_e = \lfloor b_e/\tilde{b} \rfloor$, and $v_k = \lfloor c_k/\tilde{c} \rfloor$. We also let $b_{ij} = u_{ij}\tilde{b}, b_{ie} = u_{ie}\tilde{b}$ and $c_{ik} = v_{ik}\tilde{c}$, where u_{ij}, u_{ie} and v_{ik} are integer variables. $u_{ij} \in \{0, 1, ..., u_j\}$, and for every non-zero possible value of u_{ij}, we associate it with a new binary variable x_{ijm}, where $\sum_{m=1}^{u_j} x_{ijm} = x_{ij}$ and $x_{ijm} = 1$ only if $u_{ij} = m$. Thus,

$$x_{ij}\frac{1}{b_{ij}} = \sum_{m=1}^{u_j} x_{ijm}\frac{1}{m}, \ x_{ij}b_{ij} = \sum_{m=1}^{u_j} x_{ijm}m.$$

Constraint (1g) can be replaced with following linear constraint.

$$\sum_{i\in\mathcal{I}}\sum_{m=1}^{u_j} x_{ijm}m \le b_j, \forall j \in \mathcal{J} \tag{2}$$

Similarly, for every non-zero possible value of $v_{ik} \in \{0, 1, ..., v_k\}$, we associate it with a new binary variable y_{ikn}, where $\sum_{n=1}^{v_k} y_{ikn} = y_{ik}$ and $y_{ikn} = 1$ only if $v_{ik} = n$. Thus,

$$y_{ik}\frac{1}{c_{ik}} = \sum_{n=1}^{v_k} y_{ikn}\frac{1}{n},\ y_{ik}c_{ik} = \sum_{n=1}^{v_k} y_{ikn}n.$$

Constraint (1h) can be replaced with following linear constraint.

$$\sum_{i\in\mathcal{I}}\sum_{n=1}^{v_k} y_{ikn}n \le c_k, \forall k \in \mathcal{K} \tag{3}$$

For every non-zero possible value of $u_{ie} \in \{0, 1, ..., u_e\}$, we associate it with a new binary variable w_{ier}, where $\sum_{r=1}^{u_e} w_{ier} = \sum_{p\in\mathcal{P}_e} z_{ip}$ and $w_{ier} = 1$ only if path p is chosen by task i ($z_{ip} = 1$) and $u_{ie} = r$ for $e \in p$. Note that $\sum_{r=1}^{u_e} w_{ier} = \sum_{r=1}^{u'_e} w_{ie'r}$ if $e, e' \in p$. Thus,

$$\frac{1}{u_{ie}} = \sum_{r=1}^{u_e} w_{ier}\frac{1}{r},\ z_{ip}u_{ie} = \sum_{r=1}^{u_e} w_{ier}r.$$

Constraint (1i) can be replaced with following linear constraint.

$$\sum_{i\in\mathcal{I}}\sum_{p\in\mathcal{P}_e}\sum_{r=1}^{u_e} w_{ier}r \le u_e, \forall e \in \mathcal{E} \tag{4}$$

For constraint (1f), $x_{ij}y_{ik}z_{ip}$ should be rewritten as $x_{ij}y_{ik}z_{ip}w_{ier}$ after we use $\sum_{r=1}^{u_e} w_{ier}\frac{1}{r}$ to replace $\frac{1}{u_{ie}}$. We can use a new binary variable h_{ijkper} to replace $x_{ij}y_{ik}z_{ip}w_{ier}$, where $h_{ijkper} = 1$ only when task i is offloaded to edge server j, transmitted through path p in the backhaul network, $u_{ie} = r$ for edge $e \in p$, and processed on server k. For simplicity of notation, let $\forall(i, j, k, p, e, r)$ represents $\forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \forall k \in \mathcal{K}, \forall p \in \mathcal{P}, e \in p, r \in \{0, 1, ..., u_e\}$. h_{ijkper} can be defined with the following linear constraints.

$$h_{ijkper} \ge x_{ij} + y_{ik} + z_{ip} + w_{ier} - 3, \forall(i, j, k, p, e, r) \tag{5}$$

$$h_{ijkper} \le x_{ij}, \forall(i, j, k, p, e, r) \tag{6}$$

$$h_{ijkper} \le y_{ik}, \forall(i, j, k, p, e, r) \tag{7}$$

$$h_{ijkper} \le z_{ip}, \forall(i, j, k, p, e, r) \tag{8}$$

$$h_{ijkper} \le w_{ier}, \forall(i, j, k, p, e, r) \tag{9}$$

$$h_{ijkper} \in \{0, 1\}, \forall(i, j, k, p, e, r) \tag{10}$$

Constraint (5) ensures that $h_{ijkper} = 1$ only when $x_{ij} = y_{ik} = z_{ip} = w_{ier} = 1$. Thus, constraint (1a) can be rewritten as follows.

$$\sum_{j\in\mathcal{J}_i}\frac{s_i}{\tilde{b}}\Big(\sum_{m=1}^{u_j} x_{ijm}\frac{1}{m}\Big) + \sum_{j\in\mathcal{J}_i}\sum_{k\in\mathcal{K}}\sum_{p\in\mathcal{P}_{jk}}\sum_{e\in p}\frac{s_i}{\tilde{b}}\Big(\sum_{r=1}^{u_e} h_{ijkper}\frac{1}{r}\Big) + \sum_{k\in\mathcal{K}}\frac{q_i}{\tilde{c}}\Big(\sum_{n=1}^{v_k} y_{ikn}\frac{1}{n}\Big) \le \Delta_i, \quad \forall i \in \mathcal{I} \tag{11}$$

With the above linearization method, we can reformulate problem $\mathbf{P}_0$ into an ILP problem. The ILP problem can then be solved using some existing ILP solvers. Note that the runtimes of these ILP solvers are not bounded within polynomial time. When the minimum resource units become smaller, a result with higher accuracy can be obtained, but more integer variables exist in the ILP problem, which further increases the runtime of the ILP solver.

For the approximation algorithms of problem $\mathbf{P}_0$, one possible direction is the rounding method. We can first relax the ILP problem into a Linear Programming (LP) problem, which can be solved within polynomial time. Then, we can use some polynomial-time rounding algorithms to round the fractional solution of the LP problem into an integral solution. Rounding methods are commonly used in solving ILP problems, such as the Knapsack Problems (KP) or the Generalized Assignment Problems (GAP). Based on partial enumeration, Frieze and Clarke [6] proposed a polynomial-time rounding algorithm to solve the multidimensional 0-1 knapsack problem. By constructing the bipartite graph, Shmoys and Tardos [7] developed a polynomial-time rounding algorithm to solve the GAP.

IV. Conclusion

In this presented work, we proposed a deadline-constrained multi-resource allocation problem, with both communication and computation contentions, in a multi-layer edge-cloud system, which aimed to maximize the total system profit. Different from other studies on this problem, we modeled the backhaul network as an incomplete directed graph, and bandwidth contention existed in every edge of the backhaul network. We formulated this problem as a nonconvex MINLP problem and provided a linearization method to reformulate the MINLP problem into an ILP problem, which could be solved with ILP solvers. We also introduced a possible direction to develop a polynomial-time approximation algorithm for the proposed problem. In the future, we want to explore possible polynomial-time approximation algorithms and distributed solutions for the proposed problem.

References

[1] C. Yang, Y. Liu, X. Chen, W. Zhong, and S. Xie, "Efficient mobility-aware task offloading for vehicular edge computing networks," *IEEE Access*, vol. 7, pp. 26 652–26 664, 2019.

[2] Q. Li, J. Zhao, and Y. Gong, "Cooperative computation offloading and resource allocation for mobile edge computing," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2019, pp. 1–6.

[3] T. T. Vu, D. N. Nguyen, D. T. Hoang, E. Dutkiewicz, and T. V. Nguyen, "Optimal energy efficiency with delay constraints for multi-layer cooperative fog computing networks," *IEEE Transactions on Communications*, vol. 69, no. 6, pp. 3911–3929, 2021.

[4] C. Gao, A. Shaan, and A. Easwaran, "Deadline-constrained multi-resource task mapping and allocation for edge-cloud systems," *arXiv preprint arXiv:2206.05950*, 2022.

[5] An introduction to mixed integer nonlinear optimization. [Online]. Available: https://www.ima.umn.edu/2015-2016/ND8.1-12.16/25419

[6] A. M. Frieze, M. R. Clarke *et al.*, "Approximation algorithms for the m-dimensional 0-1 knapsack problem: worst-case and probabilistic analyses," *European Journal of Operational Research*, vol. 15, no. 1, pp. 100–109, 1984.

[7] D. B. Shmoys and É. Tardos, "An approximation algorithm for the generalized assignment problem," *Mathematical programming*, vol. 62, no. 1, pp. 461–474, 1993.