Joint Task Partitioning and User Association for Latency Minimization in Mobile Edge Computing Networks

Mingjie Feng, Marwan Krunz, *Fellow, IEEE*, and Wenhan Zhang, *Student Member, IEEE*

***Abstract*—Mobile edge computing (MEC) is a promising solution to support emerging delay-sensitive mobile applications, such as self-driving, augment/virtual reality, and various Internet of Things (IoT) applications. By deploying MEC servers at network edge, e.g., close to cellular base stations (BSs), the computational tasks generated by these applications can be offloaded to edge nodes (ENs) and be quickly executed there. At the same time, with the projected large number of IoT devices, the communication and computational resources allocated to each user can be quite limited, making it challenging to provide low-latency MEC services. In this paper, we investigate the problem of task partitioning and user association in an MEC system, aiming to minimize the average latency of all users. We assume that each task can be partitioned into multiple subtasks that can be executed on local devices (e.g., vehicles), MEC servers, and/or cloud servers; each user can be associated with one of the nearby ENs. The subtasks can be independent of or dependent on each other. For each case, we formulate the joint optimization of task partitioning ratios and user association as a mixed integer programming problem. Each problem is solved by decomposing it into two subproblems. The lower-level subproblem is task partitioning under a given user association, which can be solved optimally. The higher-level subproblem is user association, we propose a dual decomposition-based approach and a matching-based approach to derive near-optimal solutions. Simulation results show that compared to benchmark schemes, the proposed schemes reduce the average latency by about 50% and 40% for the cases of independent and dependent subtasks, respectively.**

***Index Terms*—Mobile edge computing; delay-sensitive IoT applications; task partitioning; user association.**

I. Introduction

Emerging mobile Internet of Things (IoT) applications (e.g., autonomous driving, augmented/virtual reality) require executing computationally intensive tasks with stringent delay requirements [2]. Given the limited processing capability of mobile devices, completing these tasks in a timely manner is challenging. Mobile edge computing (MEC) is a promising solution to support delay-sensitive IoT applications. By deploying MEC servers at the network edge, e.g., close to base stations (BSs) or access points (APs), mobile users can offload their computational tasks to nearby MEC servers for fast processing [3], [4]. Benefiting from the proximity to end-users, low latency can be achieved.

Meanwhile, with the deployment of 5G networks, tens of billions of mobile devices can soon be connected to the Internet [6], many of which are to be supported by future MEC systems. These devices will compete for the limited computational and communication resources, and increase the workload of edge servers, hence making it less likely for the MEC systems to deliver low-latency services to all connected users [20]. To address this challenge, the design of task offloading strategy and the optimization of resource allocation among users served by an MEC server have been explored in the literature (e.g., [8], [20], [22]). The challenge of degraded latency performance caused by increased traffic load can also be addressed via collaboration between multiple MEC servers [17], [18], [26], which enables computational tasks to be transferred between these servers for improved load balancing.

Another approach for latency reduction in MEC is task partitioning. Most existing works on task offloading assume that the computation of a task begins after the whole task has been offloaded to the MEC or cloud server. In contrast, if a given computational task can be partitioned into multiple portions of subtasks and assign various portions to the local device, the MEC server, and/or the cloud server for execution, the workload at each of these entities can be reduced. Besides, the offloading and computing processes can be performed *concurrently*, resulting in lower latency. Obviously, task partitioning ratios need to be optimized based on various system parameters, e.g., computational capabilities of different devices/servers, channel quality between user and edge node (EN)[1], traffic load, etc. Note that, the subtasks can be independent of or dependent on each other. For the latter case, the subtasks have to be executed in a certain order, adding constraints on how the subtasks can be partitioned and assigned.

Manuscript received Feb. 4, 2021; revised Apr. 9, 2021; accepted May 20, 2021. This work was supported in part by NSF (grants CNS-1910348, CNS-1563655, CNS-1731164, CNS-1813401, and IIP-1822071) and by the Broadband Wireless Access & Applications Center (BWAC). Any opinions, findings, conclusions, or recommendations expressed in this paper do not necessarily reflect the views of NSF. An abridged version of this paper was presented at IEEE International Workshop on Intelligent Cloud Computing and Networking (ICCN'21) [1]. The review of this paper was coordinated by Prof. D. Tarchi. (Corresponding author: Mingjie Feng.)

M. Feng is with Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan, 430074 China. He was with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, 85721 USA. M. Krunz and W. Zhang are with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, 85721 USA. Email: mzf0022@auburn.edu, krunz@email.arizona.edu, wenhanzhang@email.arizona.edu.

Digital Object Identifier 10.1109/TVT.2021.XXXXXXX

[1]Here, an EN refers to a combination of a BS/AP and an MEC server.

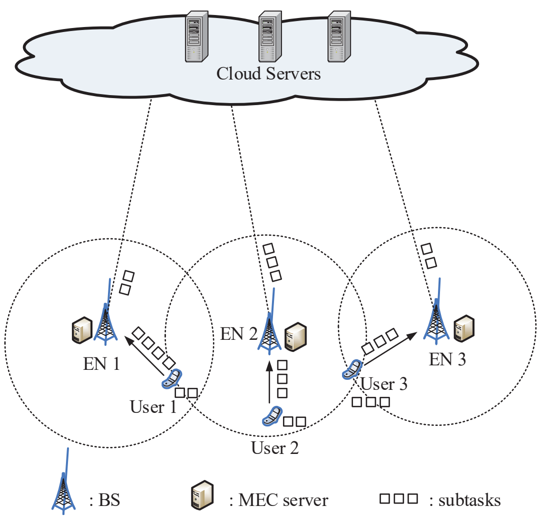

Fig. 1. System model of a multi-user MEC system with one cloud server and multiple MEC servers.

Task partitioning has recently been considered based on the model of a single EN (e.g., [23], [24]) or a single user (e.g. [26]). In an MEC system with multiple ENs serving multiple users (see Fig. 1), user association is a key design factor, as it determines the traffic load at each EN and the latency associated with offloading a task to different ENs. Thus, user association directly impacts the task partitioning strategy, necessitating a joint optimization of the two designs.

In this paper, we investigate joint optimization of task partitioning and user association, aiming to minimize the average latency of users in a cellular network-based MEC system. We develop efficient schemes to obtain near-optimal solutions to the problem. The main contributions are as follows:

- We formulate the problem of joint optimization of task partitioning and user association in MEC systems. Two types of tasks are considered: tasks that are composed of independent subtasks and tasks in which the subtasks follow a sequential dependency structure. For each case, a mixed-integer linear programming (MILP) problem is formulated with the objective of minimizing the average latency of all users.
- For each case of subtask dependency, we decompose the original problem into two subproblems. The lower-level subproblem targets optimizing the task partitioning ratio under a given user association, which can be optimally solved. The higher-level subproblem is user association, for which we develop two schemes to obtain the solution: one is based on dual decomposition and the other is based on a matching between users and ENs.
- To demonstrate the near optimality of our solutions, we derive a lower bound on the average latency.
- We evaluate the performance of the proposed schemes via simulations. The results show that, compared to benchmark schemes, the proposed schemes reduce the average latency by around 50% and 40% for the cases of independent and dependent subtasks, respectively.

The remainder of this paper is organized as follows. We review related literature in Section II. The system model is presented in Section III, followed by latency analysis in Section IV. Afterward, the problem formulation is given in Section V. Algorithmic solutions are presented in Section VI. We present our simulation results and discussion in Section VII. Finally, the paper is concluded in Section VIII.

II. Related Work

MEC has attracted considerable attention from both industry and academia. Standardization efforts by the Industry Specification Group (ISG) of the European Telecommunications Standards Institute (ETSI) were initiated [3], with the first MEC platform developed by Intel in 2014 [7]. From a research perspective, an overview of MEC can be found in [4]. A recent analytical framework that incorporates various components of MEC, including communication, computation, caching, and control, was introduced in [5].

Task assignment in MEC systems was also investigated in prior works (e.g., [8]–[15]). The majority of existing works are based on *binary* task assignment, where a task can either be offloaded to an MEC server or executed locally. In these works, the processing of multiple tasks at the MEC server is performed in parallel [11], [12], or sequentially [13], [14]. For servers that can execute tasks in parallel, the allocation of computational resources is a key factor that impacts execution latency. For sequentially processed tasks, the key design issue is the execution order of these tasks, which directly impacts the queueing delay (the waiting time of a task before execution is initiated). While most existing works consider models based on homogeneous tasks, task assignment for heterogeneous tasks was recently studied [15]. For example, delay-sensitive tasks can be assigned to the MEC server for immediate processing, whereas delay-tolerant tasks can be assigned to the cloud server. In contrast to these works, we extend the notion of task assignment to fully exploit the computational capability of local devices, MEC servers, and cloud servers by allowing individual tasks to be partitioned.

Task partitioning has been considered in some recent works under different partitioning patterns and design objectives [20]–[26], [33]. The partitioning between the local device and the cloud server was considered in [20], while task partitioning between the local device and the MEC server was considered in [21]–[24]. In [21], joint optimization of the task partitioning ratio, device transmit power, and device computational speed was performed to minimize the device's energy consumption and task execution latency. In [25], the impact of task partitioning on the energy consumptions of the local device computing and task offloading was analyzed, and an efficient design was proposed to achieve a good balance between EN energy consumption and task processing delay. The tradeoff between energy and latency was also investigated in [22], where the task partitioning ratio and communication resources were optimized to minimize the total energy consumption under a given latency constraint. In [23], [24], [26], the optimal partitioning ratio and resource allocation were derived with the objective of minimizing the overall offloading latency. Our paper differs from the above works in that it considers the interdependency of subtasks. Recently, dependency-aware task offloading was considered (e.g., in [27]–[30]). In [27], joint optimization of task offloading decision and resource allocation was considered for MEC systems with tasks following a task call graph to be executed, and a deep reinforcement learning-based solution was proposed to capture the time-varying wireless channel and edge computing capability. In [28], task offloading policy was optimized for the scenario of inter-user task dependency, where the input of a task at one user device relies on the output of the final task at another user device. In contrast to these works, we consider the dependency between the subtasks of a given task, rather than dependency between tasks. In particular, the optimal task partitioning ratios are derived based on different dependency structures. Furthermore, previous works targeted a single EN or a single user, hence user association and load balancing among ENs were not considered.

To harness the benefits of utilizing multiple ENs for task offloading, cooperation among ENs has been considered in [16]–[18], [26], [31]. Specifically, a user can offload its tasks to multiple ENs [16], [26], [31], or the ENs can send their workload to each other [17], [18]. Load balancing can also be achieved by optimizing user association in multi-cell-based MEC systems. In [32], [33], joint optimization of user association, computational resource allocation, and power control was carried out to minimize the total energy consumption. In contrast to these works, we aim to minimize the *average latency* in delay-sensitive MEC applications via a joint optimization of task partitioning and user association.

III. System Model

A. Problem Setup

As shown in Fig. 1, we consider a multi-user MEC system that consists of one cloud server and multiple MEC servers that are placed next to or integrated into the BSs of a wireless cellular network. The combination of a BS and an MEC server is regarded as an edge node (EN), which is connected to the cloud server via a backhaul connection. There are J ENs, indexed by $j \in \{1, \ldots, J\} \triangleq \mathcal{J}$. These ENs collectively serve K mobile user equipments (UE), indexed by $k \in \{1, \ldots, K\} \triangleq \mathcal{K}$. User associations are represented by the following binary variables:

$$x_{k,j} \triangleq \begin{cases} 1, & \text{if UE } k \text{ is associated with EN } j \\ 0, & \text{otherwise,} \end{cases} \qquad k \in \mathcal{K},\ j \in \mathcal{J}. \tag{1}$$

We consider a scenario in which each UE can be associated with at most one EN. Thus, $\sum_{j=1}^{J} x_{k,j} \leq 1$ for $k \in \mathcal{K}$. For UEs associated with EN j, their tasks can be executed at EN j and/or forwarded by EN j to the cloud server for execution. We assume that each UE generates one task at a time. Each task can be partitioned into multiple subtasks, each with its own data. An example of such a task is object recognition, which is based on videos taken by cameras mounted on an autonomous vehicle. Each video clip can be partitioned into multiple segments and processed at the UE, EN, and cloud server, respectively. To fully exploit the computational capabilities of different computing units for latency reduction, the subtasks can be grouped into three sets that are executed by UE, EN, and cloud server, respectively. Suppose that $x_{k,j} = 1$, the ratios of subtasks assigned to UE k, EN j, and the cloud server are denoted by α_k, $\beta_{k,j}$, and $\gamma_{k,j}$, respectively. Similar to the definitions in [23], [24], α_k, $\beta_{k,j}$, and $\gamma_{k,j}$ are the fractions of input data (e.g., file sizes of video segments) of the task generated by UE k, which are determined by the number of subtasks to be executed by UE k, EN j, and cloud server, respectively.[2] By definition, if $x_{k,j} = 0$, then $\beta_{k,j} = \gamma_{k,j} = 0$. Thus, $\beta_{k,j} \leq x_{k,j}$ and $\gamma_{k,j} \leq x_{k,j}$ for $k \in \mathcal{K}, j \in \mathcal{J}$. Considering that $\sum_{j=0}^{J} x_{k,j} \leq 1$ for every UE k, we have $\alpha_k + \sum_{j=1}^{J} \beta_{k,j} + \sum_{j=1}^{J} \gamma_{k,j} = 1$.

To capture the mobility of UEs, we assume that their locations are updated at every time slot, where each UE may stay unmoved or move in a random fashion. Accordingly, the system configurations, including task partitioning and user association, are optimized and updated on a per-slot basis. Specifically, $\{\alpha_k\}$, $\{\beta_{k,j}\}$, $\{\gamma_{k,j}\}$, and $\{x_{k,j}\}$ are optimized based on the locations of UEs at the current time slot. When UE locations change, the optimization will be performed again based on the new locations at the next time slot.

B. Computational Model

The task generated by any UE k is characterized by the size of the input data s_k (in bits) and the computational complexity z_k, expressed in the number of CPU cycles required to execute one bit of the task. Then, the number of CPU cycles required to complete the whole task is $s_k z_k$.

1) Local UE Computing Time: Let $c_k^{(\mathrm{L})}$ be the computational capability of UE k, measured in CPU cycles per second. Given α_k, the number of CPU cycles required to complete the subtasks assigned to UE k is $\alpha_k s_k z_k$. Then, the execution time (in seconds) at UE k is given by:

$$t_{\mathrm{comp},k}^{(\mathrm{L})} = \frac{\alpha_k s_k z_k}{c_k^{(\mathrm{L})}}. \tag{2}$$

2) MEC Server Computing Time: We assume that each EN is equipped with a multi-core processor, allowing it to execute the subtasks received from multiple UEs concurrently. For fairness, the computational capability of an EN is equally split between all UEs associated with that EN within each time slot. Once the input data of the subtasks have been uploaded to an EN, the EN immediately executes these subtasks. This way, there is no waiting time incurred at each EN.

We refer to the number of UEs associated with EN j as the traffic load of EN j, given by $Q_j \triangleq \sum_{k=1}^{K} x_{k,j}$. Let $c_j^{(\mathrm{E})}$ be the computational capability of EN j. As this capability is equally shared among all associated UEs, the computational capacity allocated to UE k by EN j is given by $c_{k,j}^{(\mathrm{E})} = \frac{c_j^{(\mathrm{E})}}{\sum_{k=1}^{K} x_{k,j}}$.

[2]Because a task cannot be partitioned into arbitrarily small subtasks, α_k, $\beta_{k,j}$, and $\gamma_{k,j}$ can only take a finite number of values. For example, if a task can be partitioned into 10 comparable subtasks, the partitioning ratios may take values in the set $\{0, 0.1, \ldots, 0.9, 1\}$. In this paper, we first obtain the optimal α_k, $\beta_{k,j}$, and $\gamma_{k,j}$ in the continuous domain $[0, 1]$ and then round them to the closest feasible values.

Given $\beta_{k,j}$, the execution time for the subtasks of UE k at EN j is given by:[3]

$$t^{(\mathrm{E})}_{\mathrm{comp},k,j} = \frac{\beta_{k,j} s_k z_k}{c^{(\mathrm{E})}_{k,j}} = \frac{\beta_{k,j} s_k z_k Q_j}{c^{(\mathrm{E})}_j}, k \in \mathcal{K},\ j \in \mathcal{J}. \quad (3)$$

Remarks: As the subtasks offloaded by various UEs differ in size and complexity, equal allocation of computational capability at EN results in under-utilization of computing resources. For example, the computational resource allocated to some subtasks that are completed earlier cannot be released to execute the ongoing subtasks. On the other hand, if the values of $\{\beta_{k,j}\}$ are known to EN j in advance, the computational resource allocation can be optimized in a way that maximizes the utilization of computing resource of EN j. Specifically, if the subtasks of all UEs associated with EN j are completed at the same time, the computational capability of EN j would not be "wasted". To achieve this goal, the computational capability allocated to UE k should be set to $c^{(\mathrm{E})}_{k,j} = c^{(\mathrm{E})}_j \frac{x_{k,j}\beta_{k,j}s_k z_k}{\sum_{k=1}^K x_{k,j}\beta_{k,j}s_k z_k}$. However, as indicated later in the solutions section, the values of $\{\beta_{k,j}\}$ are unknown to EN j in advance. In fact, $\{\beta_{k,j}\}$ are optimized by EN j based on various parameters including $c^{(\mathrm{E})}_{k,j}$. Therefore, the abovementioned optimization of EN computing resource cannot be directly applied. An intuitive approach to obtain an optimized solution is to iteratively update $\{\beta_{k,j}\}$ and $c^{(\mathrm{E})}_{k,j}$ until convergence. However, as the configurations of different UEs are coupled via the total computational capability $c^{(\mathrm{E})}_j$, a significant number of iterations are required to achieve convergence. Due to such uncertainty and high complexity, we adopt equal allocation of the computational capability.

3) Cloud Server Computing Time: We assume that the cloud server provides a fixed computational capability to UE k, given by $c^{(\mathrm{C})}_k$. The value of $c^{(\mathrm{C})}_k$ is based on the plan of service purchased by UE k. Suppose the subtasks of UE k is offloaded to EN j and then forwarded to the cloud server ($x_{k,j} = 1$). The execution time at the cloud server is given by:

$$t^{(\mathrm{C})}_{\mathrm{comp},k,j} = \frac{\gamma_{k,j} s_k z_k}{c^{(\mathrm{C})}_k}, k \in \mathcal{K},\ j \in \mathcal{J}. \quad (4)$$

C. Communication Model

The cellular network considered in this paper adopts an orthogonal time-frequency resource allocation, e.g., OFDMA, as used in LTE and 5G systems. We assume that the communication resource is equally allocated among all UEs associated with an EN to achieve sum-logarithmic rate maximization [35]. We assume that each EN can measure the uplink signal-to-interference-plus-noise ratio (SINR) of UEs associated with it [5], [22]–[24]. This can be implemented by having UEs send pilot or beacon signals to the EN at the beginning of each time slot. Such an approach has been widely used in cellular systems, where multiple orthogonal OFDM symbols are used as pilots and are sent from UEs to BSs for channel estimation. With the estimated channel state information (CSI), the SINRs of different UEs can be calculated. Let W be the bandwidth of the access channel for each EN, and let $\theta_{k,j}$ be the SINR for the uplink from UE k to EN j. The data rate of UE k when associated with EN j is given by:

$$R_{k,j} = \frac{W \log(1 + \theta_{k,j})}{Q_j}. \quad (5)$$

Note that ENs may use the same spectrum band (universal frequency reuse) or different spectrum bands (fractional frequency reuse). For both cases, the data rate expression in (5) is applicable as long as the SINR can be obtained. We assume that the connection between UE k and EN j can be established only when $\theta_{k,j}$ is greater than or equal to a given threshold θ_{th}. Let π_k be the set of ENs that can be employed by UE k for task offloading, $\pi_k = \{j\,|\theta_{k,j} \geq \theta_{\mathrm{th}}\}$. Then, we have:

$$x_{k,j} = 0, \forall j \notin \pi_k. \quad (6)$$

We assume that the number of UEs that can be served by EN j is upper-bounded by S_j (e.g., S_j is the number of channels). The programs of user applications are pre-installed in MEC and cloud servers. Thus, a UE only needs to send the input data of subtasks to the associated EN. The subtasks to be executed by the cloud server are offloaded by a UE via its associated EN, along with the subtasks to be executed by the EN. Then, the offloading time from UE k to EN j is given by:

$$t^{(\mathrm{E})}_{\mathrm{off},k,j} = \frac{(\beta_{k,j} + \gamma_{k,j})s_k}{R_{k,j}} = \frac{(\beta_{k,j} + \gamma_{k,j})s_k Q_j}{W \log(1 + \theta_{k,j})}. \quad (7)$$

We consider a wired backhaul link of rate M_j between EN j and the cloud sever. The backhaul capacity is sufficiently large so that there is no congestion, and this capacity is equally split between all UEs served by the EN. Thus, the total time required for offloading the subtasks of UE k to the cloud server via EN j is given by:

$$t^{(\mathrm{C})}_{\mathrm{off},k,j} = t^{(\mathrm{E})}_{\mathrm{off},k,j} + t^{(\mathrm{B})}_{\mathrm{off},k,j} \quad (8)$$

where $t^{(\mathrm{B})}_{\mathrm{off},k,j}$ is the backhaul transmission time, given by:

$$t^{(\mathrm{B})}_{\mathrm{off},k,j} = \frac{\gamma_{k,j} s_k Q_j}{M_j}. \quad (9)$$

Due to the small size of the output data, the latency for sending the outcome of a task back to a UE is neglected in our analysis [22], [31]. In case such latency is non-negligible, the time for downloading the task outcome can be calculated in the same way as in (7).

IV. Latency Analysis

A. Independent Subtasks

We first consider the scenario in which a computational task can be partitioned into multiple independent subtasks. An example of such a task is object recognition using video processing, where a video clip can be segmented into multiple episodes and separately processed at the local device, EN, and cloud server. Let $t^{(\mathrm{L})}_k$, $t^{(\mathrm{E})}_k$, and $t^{(\mathrm{C})}_k$ be the total elapsed time

[3]Here, we assume that the computational resource allocated to each task is fixed within a time slot. Specifically, if some subtasks are completed earlier than others, the unused computational capacity would not be allocated to the tasks that are still being executed, due to the short period of a time slot and the overhead for reallocation. The length of a time slot is set to be a value such that all tasks can be completed during one time slot.

until the subtasks of UE k are completed at the local device, EN j, and cloud server, respectively. We have:

$$\begin{aligned} t_k^{(\mathrm{L})} &= t_{\mathrm{comp},k}^{(\mathrm{L})} \\ t_{k,j}^{(\mathrm{E})} &= t_{\mathrm{off},k,j}^{(\mathrm{E})} + t_{\mathrm{comp},k,j}^{(\mathrm{E})} \\ t_{k,j}^{(\mathrm{C})} &= t_{\mathrm{off},k,j}^{(\mathrm{C})} + t_{\mathrm{comp},k,j}^{(\mathrm{C})}. \end{aligned} \tag{10}$$

Because subtasks are independent, they can be concurrently executed. Thus, the latency for completing the whole task is the latency of the latest completed part, which is given by:

$$T_k = \max\left\{ t_k^{(\mathrm{L})}, \sum_{j=1}^{J} x_{k,j} t_{k,j}^{(\mathrm{E})}, \sum_{j=1}^{J} x_{k,j} t_{k,j}^{(\mathrm{C})} \right\}. \tag{11}$$

B. Dependent Subtasks

In some applications, the computational task is composed of multiple inter-dependent subtasks. For example, the task of a navigation system can be divided into multiple subtasks, including localization, map downloading and updating, traffic information acquisition, and route planning. Obviously, the last subtask relies on the outcomes of previous subtasks. The inter-dependency could also lie in the program code, where the code of a task consists of multiple stages that need to be executed in a certain order.

In this paper, we consider a task with subtasks that need to be executed sequentially in different stages. For such a task, the output of one subtask is an input to the subsequent subtask. Based on the required order of execution, we divide the subtasks into three parts, which are sequentially executed in three stages: the beginning stage, the middle stage, and the final stage. Specifically, subtasks in the beginning stage are to be executed first, and then the output of the beginning stage is used to trigger the execution of subtasks in the middle stage. The same applies to subtasks in the middle and final stages. The subtasks in the beginning stage are assigned to the local device for execution. Once the local device starts executing, it also starts offloading the rest of the subtasks to the EN at the same time. After the local device completes executing its subtasks, it sends the output to the EN. Upon receiving the output, the EN begins executing the subtasks in the middle stage. At the same time, the EN sends the subtasks to be executed at the final stage to the cloud server. After the EN has completed the subtasks in the middle stage, it sends the output to the cloud server. Lastly, the cloud server returns the final output of the task to the UE after it completes executing its part. Let $\tilde{t}_k^{(\mathrm{L})}$ be the time unit the EN starts executing its subtasks, $\tilde{t}_{k,j}^{(\mathrm{E})}$ be the time from the beginning of execution at the EN to the beginning of execution at the cloud server, and $\tilde{t}_{k,j}^{(\mathrm{C})}$ be execution time at the cloud server. These quantities are calculated as follows:

$$\begin{aligned} \tilde{t}_k^{(\mathrm{L})} &= \max\left\{ t_{\mathrm{comp},k}^{(\mathrm{L})}, t_{\mathrm{off},k,j}^{(\mathrm{E})} \right\}, \\ \tilde{t}_{k,j}^{(\mathrm{E})} &= \max\left\{ t_{\mathrm{comp},k,j}^{(\mathrm{E})}, t_{\mathrm{off},k,j}^{(\mathrm{B})} \right\}, \\ \tilde{t}_{k,j}^{(\mathrm{C})} &= t_{\mathrm{comp},k,j}^{(\mathrm{C})}. \end{aligned} \tag{12}$$

Then, the latency of completing the whole task is given by:

$$\tilde{T}_k = \tilde{t}_k^{(\mathrm{L})} + \sum_{j=1}^{J} x_{k,j} \tilde{t}_{k,j}^{(\mathrm{E})} + \sum_{j=1}^{J} x_{k,j} \tilde{t}_{k,j}^{(\mathrm{C})}. \tag{13}$$

V. Problem Formulation

In this paper, we aim to minimize the average latency of UEs, which is equivalent to minimizing the sum latency of all UEs. Let $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, and $\mathbf{x}$ denote the vector $[\alpha_k]_{k\in\mathcal{K}}$, the matrix $[\beta_{k,j}]_{k\in\mathcal{K},j\in\mathcal{J}}$, the matrix $[\gamma_{k,j}]_{k\in\mathcal{K},j\in\mathcal{J}}$, and the matrix $[x_{k,j}]_{k\in\mathcal{K},j\in\mathcal{J}}$, respectively. For the case of independent subtasks, the problem is formulated as:

$$\mathbf{P1}: \min_{\{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{x}\}} \sum_{k=1}^{K} T_k \tag{14}$$

$$\text{s.t.:} \quad \alpha_k + \sum_{j=1}^{J} \beta_{k,j} + \sum_{j=1}^{J} \gamma_{k,j} = 1, \ k \in \mathcal{K} \tag{15}$$

$$\sum_{j=1}^{J} x_{k,j} \le 1, \ k \in \mathcal{K} \tag{16}$$

$$\sum_{k=1}^{K} x_{k,j} \le S_j, \ j \in \mathcal{J} \tag{17}$$

$$\beta_{k,j}, \gamma_{k,j} \le x_{k,j}, \ k \in \mathcal{K}, \ j \in \mathcal{J} \tag{18}$$

$$0 \le \alpha_k, \beta_{k,j}, \gamma_{k,j} \le 1, \ k \in \mathcal{K}, \ j \in \mathcal{J} \tag{19}$$

$$x_{k,j} \in \{0,1\}, \ k \in \mathcal{K}, \ j \in \mathcal{J} \tag{20}$$

$$x_{k,j} = 0, \ k \in \mathcal{K}, \ \forall j \notin \pi_k. \tag{21}$$

The constraints in (15) come directly from the definitions of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$; the constraints in (16) indicate that each UE can be associated with at most one EN; the constraints in (17) specify the upper bound on the number of UEs that can be served by EN j; the constraints in (18) are due to the fact that a UE can assign a certain ratio of its task to EN j and/or to the cloud server via EN j only when it is associated with EN j; and finally the constraints in (21) result from the SINR constraint as described in (6).

For the case of dependent subtasks, the problem is formulated as:

$$\mathbf{P2}: \min_{\{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\mathbf{x}\}} \sum_{k=1}^{K} \tilde{T}_k \tag{22}$$
$$\text{s.t.:} \ (15) - (21)$$

VI. Solution Algorithms

In this section, we present solutions to the formulated problems. For both cases of subtask dependency, we decompose the formulated problems (**P1** and **P2**) into two levels of subproblems. The lower-level subproblem determines the task partitioning for a given user association, which can be optimally solved. The higher-level subproblem determines the user association given that optimal task partitioning has been applied. Two schemes are developed to solve the user association problem, one is based on dual decomposition and the other is based on a matching between UEs and ENs.

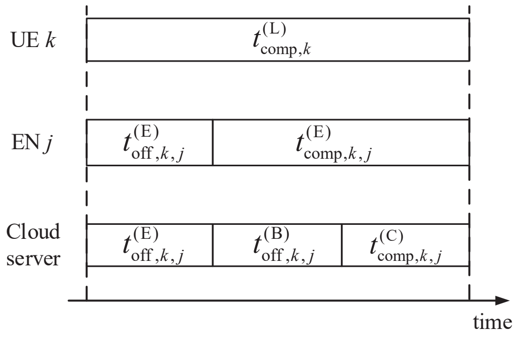
UE k
$t^{(\mathrm{L})}_{\mathrm{comp},k}$
EN j
$t^{(\mathrm{E})}_{\mathrm{off},k,j}$
$t^{(\mathrm{E})}_{\mathrm{comp},k,j}$
Cloud server
$t^{(\mathrm{E})}_{\mathrm{off},k,j}$
$t^{(\mathrm{B})}_{\mathrm{off},k,j}$
$t^{(\mathrm{C})}_{\mathrm{comp},k,j}$
time

Fig. 2. Optimal task partitioning for the case of independent subtasks.

A. Optimal Task Partitioning for a Given User Association

1) Independent Subtasks: For the case of independent subtasks, the latency of a task equals to the latency of the set of subtasks that are last to complete among the local device, EN, and the cloud server. Since $\alpha_k + \sum_{j=1}^{J} x_{k,j}\beta_{k,j} + \sum_{j=1}^{J} x_{k,j}\gamma_{k,j} = 1$, a decrease of one ratio would cause an increase of at least one of the other ratios. Thus, the optimal task partitioning is achieved when the subtasks executed at the local device, the EN, and cloud server are completed at the same time, as shown in Fig. 2. Then, we have the following equation:

$$t_k^{(\mathrm{L})} = \sum_{j=1}^{J} x_{k,j} t_{k,j}^{(\mathrm{E})} = \sum_{j=1}^{J} x_{k,j} t_{k,j}^{(\mathrm{C})}. \tag{23}$$

Let α_k^*, $[\beta_{k,j}^*]_{j\in\mathcal{J}}$, and $[\gamma_{k,j}^*]_{j\in\mathcal{J}}$ be the optimal task partitioning ratios of UE k. Applying the expressions in (10) to (23), we have:

$$\begin{aligned}\frac{\alpha_k^* s_k z_k}{c_k^{(\mathrm{L})}} &= \frac{(\sum_{j=1}^{J}\beta_{k,j}^* + \sum_{j=1}^{J}\gamma_{k,j}^*) s_k Q_j}{W\log(1+\theta_{k,j})} + \frac{\sum_{j=1}^{J}\beta_{k,j}^* s_k z_k Q_j}{c_j^{(\mathrm{E})}} \\ &= \frac{(\sum_{j=1}^{J}\beta_{k,j}^* + \sum_{j=1}^{J}\gamma_{k,j}^*) s_k Q_j}{W\log(1+\theta_{k,j})} + \frac{\sum_{j=1}^{J}\gamma_{k,j}^* s_k Q_j}{M_j} + \frac{\sum_{j=1}^{J}\gamma_{k,j}^* s_k z_k}{c_k^{(\mathrm{C})}}.\end{aligned} \tag{24}$$

Combine (24) with the equation $\alpha_k^* + \sum_{j=1}^{J}\beta_{k,j}^* + \sum_{j=1}^{J}\gamma_{k,j}^* = 1$, we can solve for α_k^*, $\sum_{j=1}^{J}\beta_{k,j}^*$, and $\sum_{j=1}^{J}\gamma_{k,j}^*$. Finally, based on $[x_{k,j}]_{j\in\mathcal{J}}$, the optimal α_k, $[\beta_{k,j}]_{j\in\mathcal{J}}$, and $[\gamma_{k,j}]_{j\in\mathcal{J}}$ can be obtained.

2) Sequentially Dependent Subtasks: For the case of sequentially dependent subtasks, the optimal task partitioning is achieved when: (1) the execution at the local device and the offloading from UE to EN are completed at the same time; (2) the execution at EN and the offloading from EN to cloud server are completed at the same time. The reason behind the first condition is that if the local device execution is completed before or after the offloading from UE to EN, the execution at the EN cannot be initiated until the latter of the two is completed. To minimize the elapsed time before the execution at the EN starts, the task partitioning ratios should be set such that the local execution time and offloading time

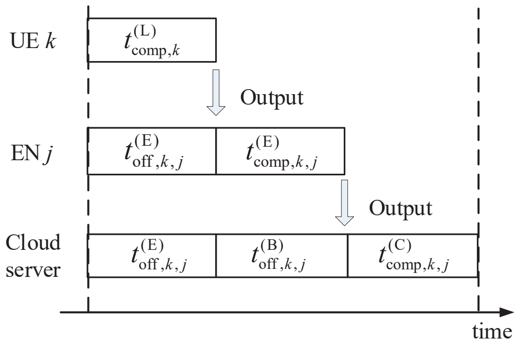
UE k
$t^{(\mathrm{L})}_{\mathrm{comp},k}$
Output
EN j
$t^{(\mathrm{E})}_{\mathrm{off},k,j}$
$t^{(\mathrm{E})}_{\mathrm{comp},k,j}$
Output
Cloud server
$t^{(\mathrm{E})}_{\mathrm{off},k,j}$
$t^{(\mathrm{B})}_{\mathrm{off},k,j}$
$t^{(\mathrm{C})}_{\mathrm{comp},k,j}$
time

Fig. 3. Optimal task partitioning for the case of sequentially dependent subtasks.

(from UE to EN) are equal. The same argument extends to the EN execution and the offloading from EN to the cloud server. As illustrated in Fig. 3, we have:

$$t_{\mathrm{comp},k}^{(\mathrm{L})} = t_{\mathrm{off},k,j}^{(\mathrm{E})},\ \ t_{\mathrm{comp},k,j}^{(\mathrm{E})} = t_{\mathrm{off},k,j}^{(\mathrm{B})}. \tag{25}$$

Applying (10), the optimal task partitioning ratios are calculated with the following:

$$\begin{aligned}\frac{\alpha_k^* s_k z_k}{c_k^{(\mathrm{L})}} &= \frac{(\sum_{j=1}^{J}\beta_{k,j}^* + \sum_{j=1}^{J}\gamma_{k,j}^*) s_k Q_j}{W\log(1+\theta_{k,j})}, \\ \frac{\sum_{j=1}^{J}\beta_{k,j}^* s_k z_k Q_j}{c_j^{(\mathrm{E})}} &= \frac{\sum_{j=1}^{J}\gamma_{k,j}^* s_k Q_j}{M_j}.\end{aligned} \tag{26}$$

The solution for α_k^*, $[\beta_{k,j}^*]_{j\in\mathcal{J}}$, and $[\gamma_{k,j}^*]_{j\in\mathcal{J}}$ can be obtained in the same way as described above.

B. Dual Decomposition-Based User Association

In this part, we present the dual decomposition-based user association scheme. We first derive the latency expression of a UE as a function of $\mathbf{x}$, given that optimal task partitioning has been applied. Based on this expression, a dual decomposition algorithm is applied to obtain the solution for user association. Due to page limits, we only present the solution for the case of dependent subtasks. This solution can be easily customized to the case of independent subtasks.

For the case of dependent subtasks, the solution for the optimal task partitioning with a given $\mathbf{x}$ can be obtained by solving the equations in (26). The solution is given by:

$$\begin{aligned}\alpha_k^* &= \frac{c_k^{(\mathrm{L})} Q_j}{z_k W\log(1+\theta_{k,j}) + c_k^{(\mathrm{L})} Q_j} \\ \beta_{k,j}^* &= \frac{z_k W\log(1+\theta_{k,j}) c_j^{(\mathrm{E})}}{\left(z_k W\log(1+\theta_{k,j}) + c_k^{(\mathrm{L})} Q_j\right)\left(c_j^{(\mathrm{E})} + z_k M_j\right)} \\ \gamma_{k,j}^* &= \frac{z_k^2 W\log(1+\theta_{k,j}) M_j}{\left(z_k W\log(1+\theta_{k,j}) + c_k^{(\mathrm{L})} Q_j\right)\left(c_j^{(\mathrm{E})} + z_k M_j\right)}.\end{aligned} \tag{27}$$

Let $\Gamma_{k,j}$ be the latency of UE k when associated with EN j under optimal task partitioning. This $\Gamma_{k,j}$ is calculated by applying the optimal partitioning ratios in (27) to the latency expression in (13). The closed-form expression for $\Gamma_{k,j}$ is given by (28) (see next page). This expression is

$$\Gamma_{k,j} = s_k z_k \frac{\left[c_j^{(\mathrm{E})} + z_k M_j + z_k W \log(1+\theta_{k,j})\right] Q_j + W \log(1+\theta_{k,j}) \frac{z_k^2 M_j}{c_k^{(\mathrm{C})}}}{c_k^{(\mathrm{L})} \left(c_j^{(\mathrm{E})} + z_k M_j\right) Q_j + z_k W \left(c_j^{(\mathrm{E})} + z_k M_j\right) \log(1+\theta_{k,j})}. \tag{28}$$

only applicable to the case when UE k is associated with an EN. For the case when UE k is not associated with any EN, i.e., $\sum_{j=1}^{J} x_{k,j} = 0$, the latency of UE k is given by $\frac{s_k z_k}{c_k^{(\mathrm{L})}}$. Combining the two cases, the latency of UE k under optimal task partitioning $\tilde{T}_k^*$ is given by:

$$\tilde{T}_k^* = \sum_{j=1}^{J} x_{k,j} \Gamma_{k,j} + \left(1 - \sum_{j=1}^{J} x_{k,j}\right) \frac{s_k z_k}{c_k^{(\mathrm{L})}}. \tag{29}$$

Then, the objective function of the user association problem is given by:

$$\sum_{k=1}^{K} \tilde{T}_k^* = \sum_{k=1}^{K} \frac{s_k z_k}{c_k^{(\mathrm{L})}} + \sum_{k=1}^{K} \sum_{j=1}^{J} x_{k,j} \left(\Gamma_{k,j} - \frac{s_k z_k}{c_k^{(\mathrm{L})}}\right). \tag{30}$$

Let $\Delta_{k,j} \triangleq \frac{s_k z_k}{c_k^{(\mathrm{L})}} - \Gamma_{k,j}$. This $\Delta_{k,j}$ can be interpreted as the achievable latency reduction for UE k when it is associated with EN j, compared to UE k executing the task by itself. From (30), we can see that minimizing sum latency $\sum_{k=1}^{K} \tilde{T}_k^*$ is equivalent to maximizing the sum latency reduction $\sum_{k=1}^{K} \sum_{j=1}^{J} \Delta_{k,j}$. Then, the user association problem can be formulated as:

$$\mathbf{P3}: \max_{\{\mathbf{x}\}} \sum_{k=1}^{K} \sum_{j=1}^{J} x_{k,j} \Delta_{k,j} \tag{31}$$

$$\text{s.t.:} \quad \sum_{j=1}^{J} x_{k,j} \leq 1,\ k \in \mathcal{K},\ j \in \mathcal{J} \tag{32}$$

$$\sum_{k=1}^{K} x_{k,j} \leq S_j,\ j \in \mathcal{J} \tag{33}$$

$$x_{k,j} \in \{0,1\},\ k \in \mathcal{K},\ j \in \mathcal{J} \tag{34}$$

$$x_{k,j} = 0,\ k \in \mathcal{K},\ \forall j \notin \pi_k. \tag{35}$$

Problem **P3** is an integer programming problem that is difficult to solve directly. To derive an effective solution algorithm, we relax the integer constraint by allowing all $x_{k,j}$ to take any values in $[0,1]$. Although the relaxed problem, **P3-Relexted**, is non-convex, we can apply a dual decomposition approach to obtain a near-optimal solution for it.

Let $\mathbf{Q} = \{Q_1, \ldots, Q_J\}$ be a set of auxiliary variables and add constraints $\sum_{k=1}^{K} x_{k,j} = Q_j,\ j \in \mathcal{J}$. With the new constraints added, we have the following problem:

$$\mathbf{P4}: \max_{\{\mathbf{x}\}} \sum_{k=1}^{K} \sum_{j=1}^{J} x_{k,j} \Delta_{k,j} \tag{36}$$

$$\text{s.t.:} \quad (32), (33), \text{ and } (35)$$

$$\sum_{k=1}^{K} x_{k,j} = Q_j,\ j \in \mathcal{J} \tag{37}$$

$$x_{k,j} \in [0,1],\ k \in \mathcal{K},\ j \in \mathcal{J}. \tag{38}$$

We apply a partial relaxation on the constraints $\sum_{k=1}^{K} x_{k,j} = Q_j, j \in \mathcal{J}$. The corresponding Lagrangian function is given by:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = \sum_{k=1}^{K} \sum_{j=1}^{J} x_{k,j} \Delta_{k,j} + \sum_{j=1}^{J} \lambda_j \left(\sum_{k=1}^{K} x_{k,j} - Q_j\right) \tag{39}$$

where $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_J\}$ are the Lagrangian multipliers for the constraints in (37). Then, the dual problem of **P4** is given by:

$$\textbf{P4-Dual:} \ \min_{\{\boldsymbol{\lambda}\}} g(\boldsymbol{\lambda}) \tag{40}$$

where $g(\boldsymbol{\lambda})$ is given by:

$$g(\boldsymbol{\lambda}) = \max_{\{\mathbf{x}\}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}). \tag{41}$$

The problems in (40) and (41) are solved iteratively. At each iteration, $\mathbf{x}$ and $\boldsymbol{\lambda}$ are updated by UEs and ENs, respectively.

The problem of maximizing $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$, as described in (41), can be decomposed into K subproblems, each solved by the corresponding UE. To obtain the optimal solution of each subproblem at iteration t, each UE k selects EN $j^{*[t]}$ that satisfies:

$$j^{*[t]} = \arg\max_{j \in \pi_k} \left\{\Delta_{k,j}(Q_j^{[t]}) - \lambda_j^{[t]}\right\}. \tag{42}$$

After the selection, each UE sends a notice to its selected EN. Upon receiving the selections from various UEs, each EN j updates $\mathbf{x}_j = [x_{1,j}, \ldots, x_{K,j}]$ with the following rule:

$$x_{k,j}^{[t]} = \begin{cases} 1, & j = j^{*[t]} \\ 0, & \text{otherwise} \end{cases} \tag{43}$$

On the other hand, Problem **P4-Dual** can be decomposed into J subproblems, each solved by the corresponding EN. For each EN j, it updates $\lambda_j^{[t]}$ with the following gradient approach:

$$\lambda_j^{[t+1]} = \lambda_j^{[t]} - \rho_j^{[t]} \eta_j^{[t]} \tag{44}$$

where $\eta_j^{[t]}$ is the gradient of $\lambda_j^{[t]}$, given by:

$$\eta_j^{[t]} = Q_j^{[t]} - \sum_{k=1}^{K} x_{k,j}^{[t]} \tag{45}$$

and $\rho_j^{[t]}$ is the step size, given by:

$$\rho_j^{[t]} = \frac{g(\boldsymbol{\lambda}^{[t]}) - g(\boldsymbol{\lambda}^*)}{\left\|\boldsymbol{\eta}^{[t]}\right\|^2}. \tag{46}$$

After updating $\lambda_j^{[t]}$, EN j updates $Q_j^{[t]}$ as follows:

$$Q_j^{[t+1]} = \min\{\sum_{k=1}^{K} x_{k,j}^{[t]}, S_j\}. \tag{47}$$

Algorithm 1: Dual Decomposition-Based User Association Algorithm

```
1  Initialize $\mathbf{Q}$ and $\boldsymbol{\lambda}$ do
2      for $k = 1 : K$ do
3          UE $k$ selects the optimal EN according to (42) and informs the selected EN
4      end
5      for $j = 1 : J$ do
6          EN $j$ updates $\mathbf{x}_j$ according to (43) ;
7          Updates $\eta_j$ according to (45) ;
8          Updates $\lambda_j$ according to (44) ;
9          Updates $Q_j$ according to (47) ;
10     end
11     $t++$
12 while ($\mathbf{x}$ does not converge);
```

Finally, EN j broadcasts the updated values of $\lambda_j^{[t]}$ and $Q_j^{[t]}$ to nearby UEs. The UEs then initiate the next iteration of EN selection. The procedure of the dual decomposition-based user association algorithm is summarized in Algorithm 1. At each iteration of Algorithm 1, the signaling overhead includes: (1) the notification message sent by each UE to its selected EN; (2) the broadcasting messages sent by each EN that indicate the updated values of the Lagrangian variable and traffic load. It can be seen that the overhead of information exchange between ENs and UEs at each iteration is quite small. Thus, the computational complexity of Algorithm 1 is dominated by the number of iterations required to achieve convergence.

Lemma 1. *Algorithm 1 converges faster than the sequence* $\{1/\sqrt{t}\}$.

Proof. Consider the optimality gap of $\boldsymbol{\lambda}$, we have:

$$\begin{aligned}
&\|\boldsymbol{\lambda}^{[t+1]} - \boldsymbol{\lambda}^*\|^2 \\
&= \left\|\boldsymbol{\lambda}^{[t]} - \frac{g(\boldsymbol{\lambda}^{[t]}) - g(\boldsymbol{\lambda}^*)}{\|\boldsymbol{\eta}^{[t]}\|^2}\boldsymbol{\eta}^{[t]} - \boldsymbol{\lambda}^*\right\|^2 \\
&= \left\|\boldsymbol{\lambda}^{[t]} - \boldsymbol{\lambda}^*\right\|^2 - 2\left(\boldsymbol{\lambda}^{[t]} - \boldsymbol{\lambda}^*\right)^{\mathrm{T}}\frac{g(\boldsymbol{\lambda}^{[t]}) - g(\boldsymbol{\lambda}^*)}{\left\|\boldsymbol{\eta}^{[t]}\right\|^2}\boldsymbol{\eta}^{[t]} \\
&\quad + \left(\frac{g(\boldsymbol{\lambda}^{[t]}) - g(\boldsymbol{\lambda}^*)}{\left\|\boldsymbol{\eta}^{[t]}\right\|^2}\right)^2\left\|\boldsymbol{\eta}^{[t]}\right\|^2 \\
&\overset{(a)}{\leq} \|\boldsymbol{\lambda}^{[t]} - \boldsymbol{\lambda}^*\|^2 - 2\frac{\left(g(\boldsymbol{\lambda}^{[t]}) - g(\boldsymbol{\lambda}^*)\right)^2}{\|\boldsymbol{\eta}^{[t]}\|^2} \\
&\quad + \left(\frac{g(\boldsymbol{\lambda}^{[t]}) - g(\boldsymbol{\lambda}^*)}{\|\boldsymbol{\eta}^{[t]}\|^2}\right)^2\|\boldsymbol{\eta}^{[t]}\|^2 \\
&\leq \|\boldsymbol{\lambda}^{[t]} - \boldsymbol{\lambda}^*\|^2 - \frac{\left(g(\boldsymbol{\lambda}^{[t]}) - g(\boldsymbol{\lambda}^*)\right)^2}{\hat{\boldsymbol{\eta}}^2}.
\end{aligned}$$

Inequality (a) is due to the convexity of problem **P4-dual**, i.e., $g(\boldsymbol{\lambda}^{[t]}) - g(\boldsymbol{\lambda}^*) \leq \left(\boldsymbol{\lambda}^{[t]} - \boldsymbol{\lambda}^*\right)^{\mathrm{T}}\boldsymbol{\eta}^{[t]}$. $\hat{\boldsymbol{\eta}}$ is an upper bound on $\boldsymbol{\eta}^{[t]}$. Since $\lim_{t\to\infty}\boldsymbol{\lambda}^{[t+1]} = \lim_{t\to\infty}\boldsymbol{\lambda}^{[t]}$, it follows that $\lim_{t\to\infty} g(\boldsymbol{\lambda}^{[t]}) = g(\boldsymbol{\lambda}^*)$. Summing the above inequalities over t, we have:

$$\sum_{t=1}^{\infty}\left(g(\boldsymbol{\lambda}^{[t]}) - g(\boldsymbol{\lambda}^*)\right)^2 \leq \hat{\boldsymbol{\eta}}^2\left\|\boldsymbol{\lambda}^{[1]} - \boldsymbol{\lambda}^*\right\|^2. \tag{48}$$

The rest of the proof proceeds by contradiction. Suppose that $g(\lambda^{[\mathbf{t}]})$ converges slower than $\{1/\sqrt{t}\}$. Then, we have $\lim_{t\to\infty}\sqrt{t}\left(g(\boldsymbol{\lambda}^{[t]}) - g(\boldsymbol{\lambda}^*)\right) > 0$. As a result, there must be a positive number ε and a sufficiently large t' such that:

$$\sqrt{t}\left(g(\boldsymbol{\lambda}^{[t]}) - g(\boldsymbol{\lambda}^*)\right) \geq \varepsilon, \forall t > t'. \tag{49}$$

Taking the square sum of (49) from t' to ∞, we have:

$$\sum_{t=t'}^{\infty}\left(g(\boldsymbol{\lambda}^{[t]}) - g(\boldsymbol{\lambda}^*)\right)^2 \geq \varepsilon^2\sum_{t=t'}^{\infty}\frac{1}{t} = \infty. \tag{50}$$

This contradicts (48). Thus, we conclude that $g(\boldsymbol{\lambda}^{[t]})$ converges faster than the sequence $\{1/\sqrt{t}\}$. □

Lemma 2. *An upper bound on the complexity of Algorithm 1 is* $1/\kappa^2$*, where* κ *is the threshold for the convergence of* $g(\boldsymbol{\lambda}^{[t]})$.

Proof. From Lemma 1 and for a sufficiently large t, we have $g(\boldsymbol{\lambda}^{[t]}) - g(\boldsymbol{\lambda}^*) < 1/\sqrt{t}$. Then, for a sufficiently small $\kappa < 1/\sqrt{t}$, $g(\boldsymbol{\lambda}^{[t]}) - g(\boldsymbol{\lambda}^*)$ is guaranteed to be smaller than κ. This means when the sequence $g(\boldsymbol{\lambda}^{[t]})$ achieves an optimality gap that is less than κ, the number of iterations is less than $1/\kappa^2$. Thus, given the convergence threshold κ, the total number of variable updates is upper bounded by $1/\kappa^2$, which serves as an upper bound on the complexity of Algorithm 1. □

Remarks: An effective approach to reduce the number of iterations in Algorithm 1 is to properly select the initial values. In a low mobility environment, the distribution of UEs does not change dramatically. Then, the values of $\mathbf{Q}$ obtained at the end of the previous iteration can be used as the initial values of $\mathbf{Q}$ in the current iteration.

To obtain a near-optimal solution and also reduce the number of iterations, the initial values of $\{\lambda_j\}$ can be set to be close to their optimal values. Note that λ_j can be interpreted as a price for associating with EN j. When λ_j is larger than its optimal value, according to (41), fewer users will select EN j. Then, the value of η_j will increase, causing λ_j to decrease in the next iteration. In the same way, when λ_j is smaller than its optimal value, it will increase in the next iteration. Based on this property, we can set λ_j to a relatively large value when the current traffic load of EN j is higher than the empirical value, and vice versa when the traffic load is low.

Performance Bound: To show that a near-optimal solution can be achieved by the proposed schemes, we derive a lower bound that will be used for comparison in simulations. First, we exhaustively search all feasible traffic load vectors $\mathbf{Q}$. For each $\mathbf{Q}$, we relax the feasibility constraints in (34) and (35) from **P4** and solve the following linear programming (LP) problem:

$$\mathbf{P5} : \max_{\{\mathbf{x}\}}\sum_{k=1}^{K}\sum_{j=1}^{J} x_{k,j}\Delta_{k,j} \tag{51}$$
$$\text{s.t.:}\ (32), (37) \text{ and } (38).$$

Algorithm 2: Strategy of EN j During the Matching Process

```
1 while (matching is not converged) do
2     if (EN $j$ receives more than $S_j$ applications) then
3         Keep the top $S_j$ UEs with largest $\Delta_{k,j}$ in the
          waiting list and reject the rest
4     else
5         Keep all UEs that applied to EN $j$ in the waiting list
6     end
7 end
```

Since each element in $\mathbf{Q}$ has S_j possible values, the total number of LPs to be solved is $\prod_{j=1}^{J} S_j$. Among the $\prod_{j=1}^{J} S_j$ LPs, we find the one with the largest value for the objective function $\sum_{k=1}^{K}\sum_{j=1}^{J} x_{k,j}\Delta_{k,j}$. Then, the largest value of $\sum_{k=1}^{K}\sum_{j=1}^{J} x_{k,j}\Delta_{k,j}$ is an upper bound on the sum latency reduction. Subtracting this value from the sum of UE local computing time $\sum_{k=1}^{K}\frac{s_k z_k}{c_k^{(\mathrm{L})}}$, the outcome is a lower bound on the sum latency of all UEs.

C. Matching-based User Association

In the dual decomposition-based user association scheme, a considerable number of iterations is required to achieve convergence, which incurs a significant amount of information exchange and overhead. In this section, we propose a matching-based user association approach to reduce the overhead. We formulate a many-to-many matching based on the model of *college admission problem*, and demonstrate that the matching process will converge to a stable matching.

In the college admission problem, multiple students are applying to multiple colleges. Each student has a preference list over all the colleges, which indicates the order of willingness to attend these colleges. Each college also has a preference list over the students and a capacity that limits the number of admitted students [37]. During the matching process, each college has a waiting list that indicates its temporary selection of students.

In our problem, we regard the UEs and ENs as the students and colleges, respectively. The ENs in the preference list of UE k follow the descending order of $\Delta_{k,j}$. From the perspective of UE k, the value of $\Delta_{k,j}$ is determined by multiple factors, including the channel gain $\theta_{k,j}$, UE k's computing capability $c_k^{(\mathrm{L})}$, UE k's available cloud computational capability $c_k^{(\mathrm{C})}$, and the size and computational complexity of the task requested by UE k (s_k and z_k), as indicated in (28). To minimize the latency (equivalently maximize the latency reduction), UE k selects EN according to:

$$x_{k,j^*} = 1, \;\; j^* = \arg\max_{j\in\pi_k} \Delta_{k,j}. \tag{52}$$

From the perspective of EN j, UEs in its preference list follow the descending order of $\Delta_{k,j}$, and the maximum number of UEs that can be admitted is S_j. To minimize the sum latency, EN j decides to hold or reject a UE according to Algorithm 2.

Before the matching starts, UEs and ENs establish their preference lists with the following steps. First, UE k sends an offloading request to nearby ENs j ($j \in \pi_k$). The request is sent along with the task size s_k and complexity z_k, as well as the computational capability of UE k's device $c_k^{(\mathrm{L})}$ and the computational capability that the cloud server is allocating to UE k (which is determined by the computing service plan purchased by UE k and is known to UE k in advance), given by $c_k^{(\mathrm{C})}$. Upon receiving the offloading request and various parameters, EN j calculates the achievable latency reduction $\Delta_{k,j}$ using (28), based on the received parameter values as well as its current traffic load Q_j, the observed channel gain $\theta_{k,j}$, the backhaul data rate M_j, and its computational capability $c_j^{(\mathrm{E})}$. Then, EN j creates its preference list based on the rank of $\Delta_{k,j}$ values of different UEs. Finally, EN j sends the calculated $\Delta_{k,j}$ to the UEs, which will be used by them to create their preference lists. Along with $\Delta_{k,j}$, the values of $\{\alpha_k^*, \beta_{k,j}^*, \gamma_{k,j}^*\}$ are sent to the UEs, which are later used to perform task partitioning after the matching process converges.

The matching process is based on multiple rounds of message exchange between UEs and ENs. This process starts with each UE sending an offloading application to the ENs according to the strategy given in (52), i.e., the UE always applies to the EN at the top of its preference list. Upon receiving applications of different UEs, an EN decides whether to put the UEs in its waiting list according to Algorithm 2. After the decisions are made, the EN sends rejection notices to the UEs it decided to reject. The EN also updates its Q_j and broadcasts it to nearby UEs. Each UE then updates its preference list by calculating $\Delta_{k,j}$ with the updated Q_j. A UE would make another round of applications under one of the following cases:

Case (a): Its application has been rejected;

Case (b): It is currently in the waiting list of EN j, but a higher latency reduction can be achieved by switching to another EN j' and such switching is feasible. A switching is feasible only when one of the following conditions is satisfied:

Condition (i): The waiting list of EN j' is not full, i.e., $Q_{j'} < S_{j'}$;

Condition (ii): The waiting list of EN j' is full, i.e., $Q_{j'} = S_{j'}$, but there is another UE k' currently in the waiting list of EN j' who is less competitive than UE k, i.e., $\Delta_{k',j'} < \Delta_{k,j'}$.

After receiving another round of applications, each EN makes decisions by comparing the new applicants with ones that are already in the waiting list according to Algorithm 2. With the decisions made by ENs, the UEs update their preference lists and submit another round of applications if they are under Case (a) or Case (b). Such a matching process continues until convergence is achieved.

Compared to a standard college admission problem with static preference lists for both students and colleges, the preference lists of UEs and ENs in our problem change over time. Despite such a difference, we show that the matching process converges to a stable matching.

Definition 1. *In a stable matching, there is no UE-EN pair such that: the UE can be matched to a better EN than the current associated EN, and the EN can be matched to a UE who is better than one of the UEs that are currently associated* *with it.*

The convergence of the matching is given in Theorem 1.

Theorem 1. *The proposed matching process converges and the outcome is a stable matching.*

Proof. Suppose for contradiction, a stable matching cannot be achieved. By definition, if the matching is not stable, there must be at least one pair of UE k and EN j such that UE k is currently associated with EN j and UE k can switch to another EN j' that is better than EN j. Then, we have $\Delta_{k,j'} > \Delta_{k,j}$ and it is feasible for UE k to switch from EN j to EN j'. As a result, either Condition (i) or Condition (ii) is satisfied. If Condition (i) is satisfied, i.e., $Q_{j'} < S_{j'}$, then UE k should have already switched to EN j', contradicting the fact that it is currently associated to EN j. If Condition (ii) is satisfied, i.e., $Q_{j'} = S_{j'}$, then there must be at least one UE k' that is associated with EN j' who is less competitive than UE k from the perspective of EN j', i.e., $\Delta_{k,j'} > \Delta_{k',j'}$. Meanwhile, as UE k' is currently in the waiting list of EN j' while UE k is not, the only reason is that UE k has never submitted an application to EN j' before. However, since UE k prefers EN j' over EN j, it must have applied to EN j' prior to EN j, which leads to a contradiction. Thus, we conclude that the matching process converges to a stable matching. $\square$

VII. Simulation Results

We evaluate the performance of the proposed schemes via simulations. We consider a 500 m $\times$ 500 m area with 10 randomly located ENs. UEs are uniformly distributed in the area. We adopt the parameter values in [24], [31]. Specifically, the channel is modeled as a combination of distance-dependent path loss $140.7+36.7\log_{10}d$ in dB and Rayleigh fading, where d is the distance in meters. The UE transmission power is set to 20 dBm and the noise density is -174 dBm/Hz. The uplink bandwidth is 10 MHz. The data rate for the backhaul link is uniformly distributed in $[20, 80]$ Mbps. S_j is set to 15 for all ENs. Unless otherwise stated, the default number of users is 100, and the default size and complexity of the tasks are $s_k = 200$ KB and $z_k = 1000$ CPU cycles/bit, respectively. The computational capabilities of the local device, EN, and the cloud server are 1 GHz, 50 GHz, and 100 GHz, respectively. Each task can be partitioned into 50 subtasks, yielding a resolution of 0.02 for the task partitioning ratios. When evaluating the impact of a parameter on the latency performance, the value of this parameter is varied while all other parameters are set to be their default values.

We consider two schemes for the proposed approaches. The first scheme is called *DD-UA*, which applies the optimal task partitioning and the dual decomposition-based user association. The second scheme is called *matching-UA*, which applies the optimal task partitioning and the matching-based user association. The two proposed schemes are compared with several benchmark schemes. To demonstrate the effectiveness of the proposed task partitioning solution, four benchmark schemes are considered. The first two schemes are called *edge-only* and *cloud-only*, where all subtasks are executed at the EN and the cloud server, respectively. These two schemes are based on *binary offloading* (i.e., offloading whole tasks), which has been considered in exiting works on adaptive offloading. The other two schemes are called *local & edge* and *local & cloud*, which are similar to the solutions presented in existing works on task partitioning. The local & edge scheme is based on the proposed task partitioning solution, but only applies partitioning between the local device and EN; the local & cloud scheme is based on the proposed task partitioning solution, but only applies partitioning between the local device and cloud server. To provide a fair comparison, the dual decomposition-based user association is applied to these four schemes. We show the effectiveness of the proposed user association solution by comparing it with a heuristic user association scheme called *heuristic-UA*, in which each UE is associated with the EN that has the maximum SINR. For a fair comparison, the optimal task partitioning is also applied in the heuristic-UA scheme. Finally, the derived performance lower bound is plotted.

The latency performance of different schemes is shown in Figs. 4–7. In Figs. 4 and 5, we plot the average latency versus the number of users for the cases of independent and dependent subtasks, respectively. Expectedly, the average latency increases as more users are served, because less communication and computational resources are allocated to each user. Benefiting from the proximity of MEC servers

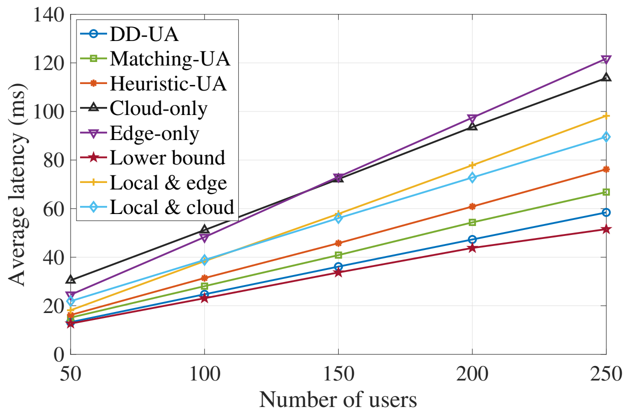

Fig. 4. Average latency vs. number of users (independent subtasks).

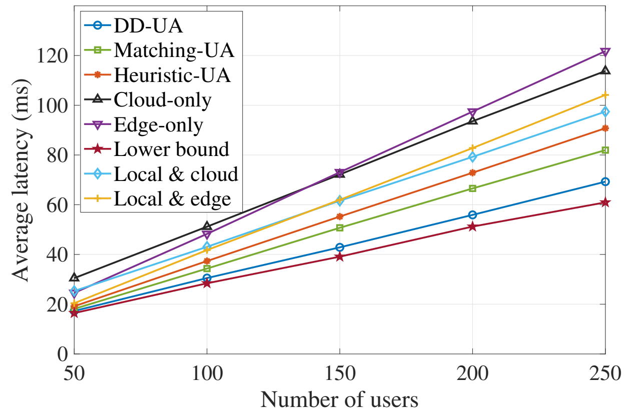

Fig. 5. Average latency vs. number of users (sequentially dependent subtasks).

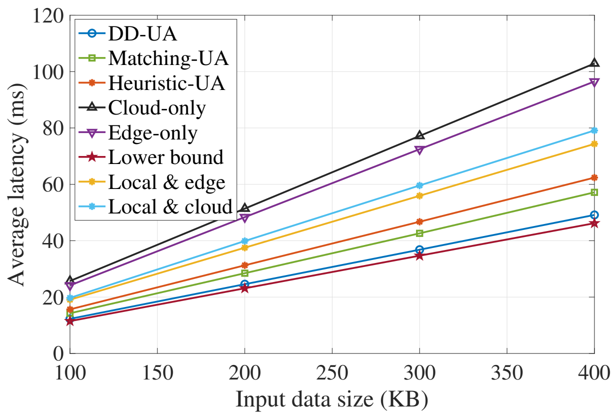

Fig. 6. Average latency vs. input data size (independent subtasks).

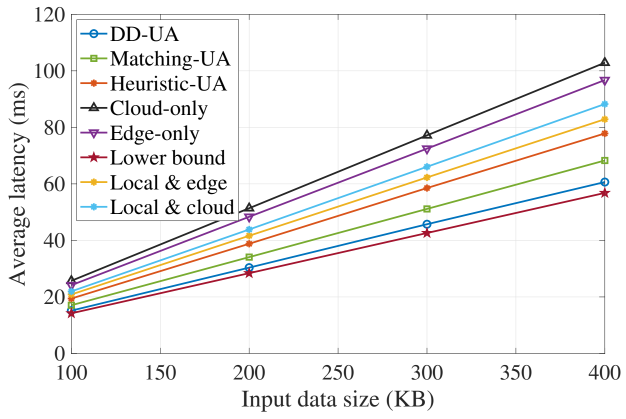

Fig. 7. Average latency vs. input data size (sequentially dependent subtasks).

to end-users, the average latency of the edge-only scheme is lower than the cloud-only scheme when the traffic load is low. However, when the traffic load is high (i.e., more users are served), the average latency of the edge-only scheme becomes higher than the cloud-only scheme, as the computing latency at the EN is significantly increased when more users are sharing the computational resource. The proposed schemes (DD-UA and matching-UA) and the heuristic-UA scheme achieve lower latencies than the edge-only and cloud-only schemes, as the tasks are properly partitioned and assigned to local devices, ENs, and cloud servers. Comparing different user association approaches, we can see that the proposed schemes outperform the heuristic-UA scheme, since the set of UEs served by each EN are optimized and a good load balancing among ENs is achieved. It is also observed that the performance gaps between the proposed schemes and the lower bound are relatively small, showing that a near-optimal solution for user association can be achieved by the proposed schemes. Comparing Fig. 4 with Fig. 5, it can be seen that the latency reduction achieved by task partitioning is more significant for the case of independent subtasks than the case of dependent subtasks. This is because, for the case of independent subtasks, the communication and computing processes can be concurrently performed at the local device, the EN, and the cloud server; while for the case of dependent subtasks, only part of these processes can be performed in parallel, which limits the benefit of task partitioning.

The impact of the input data size of tasks on average latency is presented in Figs. 6 and 7, where similar trends among different schemes are observed. As expected, the latencies of all schemes increase linearly with the input data size. When the data size is small, the latency reduction achieved by the proposed schemes is small, since the local device is able to execute the tasks in a timely manner. As the input data size increases, a higher reduction in latency can be achieved. We plot the average latency versus the task computational complexity in Fig. 8. As expected, the complexity has the same impact on the latency as the input data size.

The optimal task partitioning ratios under different system settings are shown in Figs. 9–12. In Fig. 9 and 10, we plot the average optimal partitioning ratios over all users versus the number of users. As the number of users increases, the ratios assigned to the local device and cloud server increase, while the ratio assigned to the edge server decreases. This is because when the traffic load increases, both the offloading and computing times at the EN become higher. In contrast, the computing times at local devices and cloud servers are not impacted by the traffic load. To minimize the total latency, the workload assigned to the edge server should be reduced.

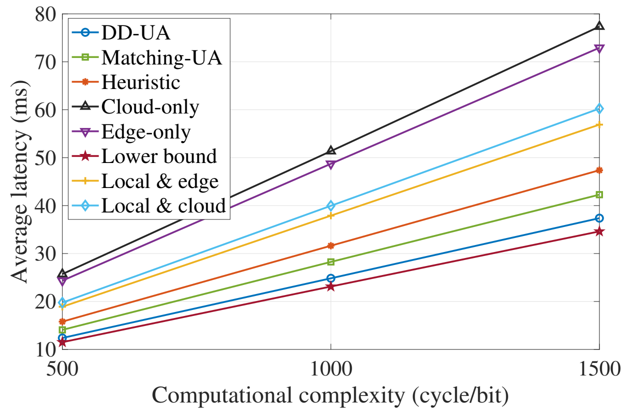

Fig. 8. Average latency vs. task complexity (independent subtasks).

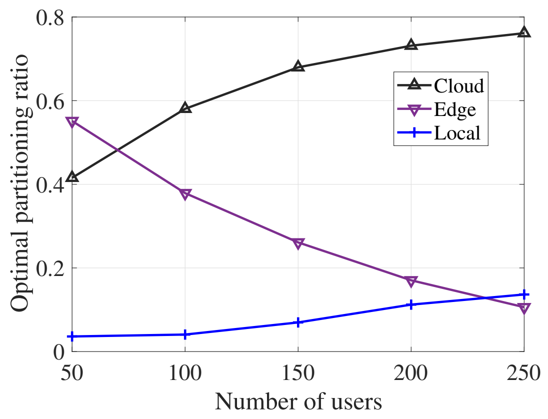

Fig. 9. Optimal partitioning ratios vs. number of users (independent subtasks).

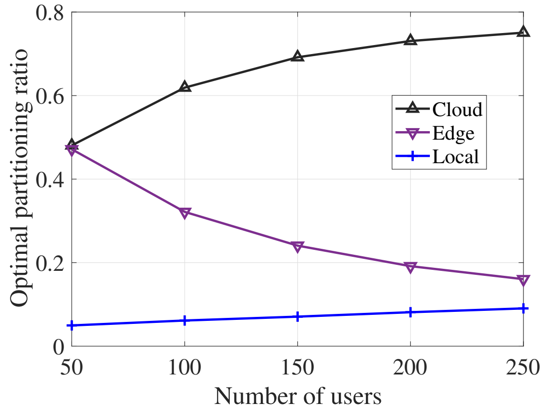

Fig. 10. Optimal partitioning ratio vs. number of users (sequentially dependent subtasks).

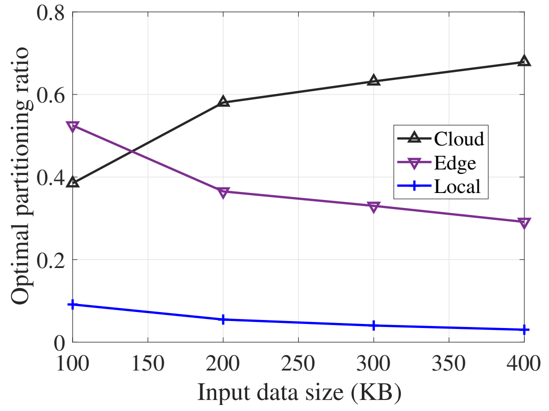

Fig. 11. Optimal partitioning ratio vs. input data size (independent subtasks).

We investigate the impact of input data size on the optimal partitioning ratios for the cases of independent and dependent subtasks in Figs. 11 and 12, respectively. For both cases, when the input data size increases, smaller ratios are assigned to the local device and edge server while a larger ratio is assigned to the cloud server. This is because the total computational workload (measured in CPU cycles) becomes larger as the input data size increases, hence assigning more subtasks to the cloud server can fully exploit its powerful computational capability to reduce the execution time. Comparing Figs. 11 and 12, it can be seen that the overall partitioning patterns of the two cases are similar, since the dominant parameters that impact the optimal partitioning (e.g., channel condition, computational capabilities of different computing units, user distribution, etc.) are the same. However, there is still a noticeable difference between the two figures. Specifically, the ratios of EN and local device in the case of independent subtasks are more than 10% higher than those in the case of dependent subtasks when the data size is small. This is because the computations at a local device and cloud server can be performed concurrently with other processes in the case of independent subtasks. Thus, assigning more subtasks to local devices and edge servers reduces the latency.

Finally, we show in Fig. 13 the effectiveness of the proposed user association schemes in terms of load balancing. It can be seen that the matching-based scheme achieves better load balancing than the heuristic user association scheme (i.e., where UEs are associated with the EN that provides the maximum SINR). This is because, during the matching process, an overloaded (underloaded) EN would provide a relatively high (low) latency to UEs, making the EN rank lower (higher) in the preference lists of UEs. This discourages (attracts) UEs to select this EN in the next round of matching, resulting in balanced load distribution among ENs. The dual decomposition-based scheme achieves the most balanced load among ENs, as the values of $\{Q_j\}$ are optimized via iterative interactions between ENs and UEs. As a result, the traffic loads are properly assigned to various ENs in a way that reduces congestion and improves resource utilization.

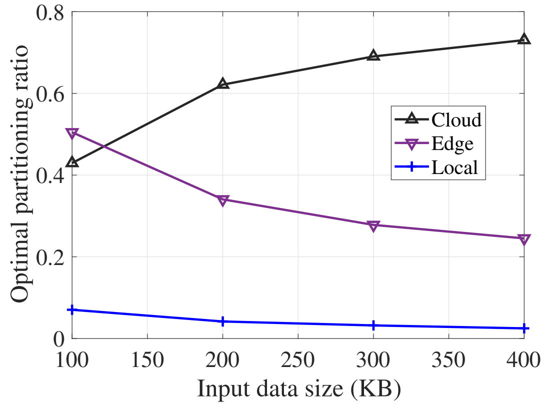

Fig. 12. Optimal partitioning ratio vs. input data size (sequentially dependent subtasks).

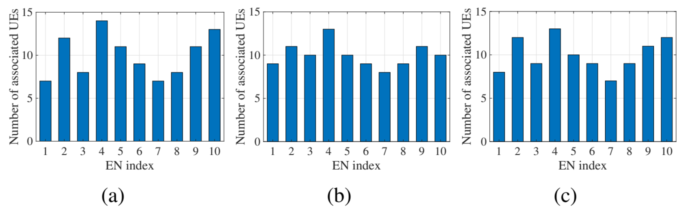

Fig. 13. Traffic load distribution among ENs. (a) under heuristic user association, (b) under the proposed dual decomposition-based user association, (c) under the proposed matching-based user association.

VIII. Conclusions

In this paper, we considered joint optimization of task partitioning and user association to minimize the average latency of users in an MEC system. We formulated a mixed integer programming problem for the cases of both independent and dependent subtasks. For each case, the original problem was decomposed into two subproblems: a lower-level subproblem for task partitioning under a given user association, and a higher-level subproblem for user association. We first derived the optimal task partitioning solutions for both cases of subtask dependency. Then, we proposed a dual decomposition-based user association scheme that achieves a near-optimal solution. We also proposed a matching-based user association scheme with proven convergence. Simulation results show that the proposed schemes outperform several benchmark schemes.

Specifically, the average latency is reduced by about 50% and 40% for the cases of independent and dependent subtasks, respectively. For future work, we will explore optimal task partitioning under general subtask dependency (which can be indicated by a subtask call graph). In addition, new methods for reducing the number of iterations in the proposed user association algorithms will be developed.

References

[1] M. Feng, M. Krunz, and W. Zhang, "Task partitioning and user association for latency minimization in mobile edge computing networks," in *IEEE ICCN WKSHPS* in conjunction with *INFOCOM 2021,* May 2021.

[2] Huawei, "5G Vision: 100 Billion Connections, 1 ms Latency, and 10 Gbps Throughput," Accessed: June 2021. [Online]. Available: http://support.huawei.com/huaweiconnect/carrier/en/thread-357441-1-1.html

[3] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing: A key technology towards 5G," *ETSI White Paper,* vol. 11, 2015.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surv. Tuts.,* vol. 19, no. 4, pp. 2322–2358, Sept.–Dec. 2017.

[5] A. Ndikumana, N. H. Tran, T. M. Ho, Z. Han, W. Saad, D. Niyato, and C. S. Hong, "Joint communication, computation, caching, and control in big data multi-access edge computing," *IEEE Trans. Mobile Comput.,* vol. 19, no. 6, pp. 1359–1374, June 2020.

[6] Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022" White Paper, Feb. 2019. Accessed: June 2021. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html

[7] Intel, "Increasing mobile operators' value proposition with edge computing," 2014.

[8] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.,* vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[9] Y. H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *IEEE Trans. Mobile Comput.,* vol. 16, no. 11, pp. 3056–3069, Nov. 2017.

[10] 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.

[11] M.-H. Chen, B. Liang, M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in *Proc. IEEE INFOCOM'17,* Atlanta, GA, May 2017, pp. 1–9.

[12] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.,* vol. 36, no. 3, pp. 587–597, Mar. 2018.

[13] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing," *IEEE J. Sel. Areas Commun.,* vol. 37, no. 3, pp. 668–682, Mar. 2019.

[14] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things J.,* vol. 6, no. 3, pp. 4005–4018, June 2019.

[15] X. Lyu et al., "Selective offloading in mobile edge computing for the green Internet of Things," *IEEE Netw.,* vol. 32, no. 1, pp. 54–60, Jan./Feb. 2018.

[16] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Commun. Mag.,* vol. 55, no. 4, pp. 54–61, Apr. 2017.

[17] Y. Xiao and M. Krunz, "QoE and power efficiency tradeoff for fog computing networks with fog node cooperation," in *Proc. IEEE INFOCOM'17,* Atlanta, GA, May 2017, pp. 1–9.

[18] Y. Xiao and M. Krunz, "Distributed optimization for energy-efficient fog computing in the tactile Internet," *IEEE J. Sel. Areas Commun.,* vol. 36, no. 11, pp. 2390–2400, Nov. 2018.

[19] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proc. IEEE INFOCOM'16,* San Francisco, CA, Apr. 2016, pp. 1–9.

[20] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Trans. Comput.,* vol. 64, no. 8, pp. 2253–2266, Aug. 2015.

[21] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.,* vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[22] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.,* vol. 16, no. 3, Mar. 2017.

[23] J. Ren, G. Yu, Y. Cai, and Y. He, "Latency optimization for resource allocation in mobile-edge computation offloading," *IEEE Trans. Wireless Commun.,* vol. 17, no. 8, pp. 5506–5518, Aug. 2018.

[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] A. Bozorgchenani, D. Tarchi, and G. E. Corazza, "Centralized and distributed architectures for energy and delay efficient fog network-based edge computing services," *IEEE Trans. Green Commun. Netw.,* vol. 3, no. 1, pp. 250–263, Mar. 2019.

[26] J. Liu and Q. Zhang, "Offloading schemes in mobile edge computing for ultra-reliable low latency communications," *IEEE Access,* vol. 6, pp. 12825–12837, 2018.

[27] J. Yan, S. Bi, and Y. J. A. Zhang, "Offloading and resource allocation with general task graph in mobile edge computing: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.,* vol. 19, no. 8, pp. 5404–5419, Aug. 2020.

[28] J. Yan, S. Bi, Y. J. A. Zhang, and M. Tao, "Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency," *IEEE Trans. Wireless Commun.,* vol. 19, no. 1, pp. 235–250, Jan. 2020.

[29] C. Shu, Z. Zhao, Y. Han, and G. Min, "Dependency-aware and latency-optimal computation offloading for multi-user edge computing networks," in *Proc. IEEE SECON'19,* Boston, MA, June 2019, pp. 1–9.

[30] S. Pan, Z. Zhang, Z. Zhang, and D. Zeng, "Dependency-aware computation offloading in mobile edge computing: A reinforcement learning approach," *IEEE Access,* vol. 7, pp. 134742–134753, Sept. 2019.

[31] 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.

[32] S. Sardellitti, M. Merluzzi, and S. Barbarossa, "Optimal association of mobile users to multi-access edge computing resources," in *Proc. IEEE ICC'18,* Kansas City, MO, May 2018, pp. 1–6.

[33] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Trans. Veh. Technol.,* vol. 67, no. 12, pp. 12313–12325, Oct. 2018.

[34] M. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? The bandwidth and energy costs of mobile cloud computing," in *Proc. IEEE INFOCOM'13,* Turin, Italy, Apr. 2013, pp. 1285–1293.

[35] Q. Ye, B. Rong, Y. Chen, M.A.-Shalash, C. Caramanis, and J. G. Andrews, "User association for load balancing in heterogeneous cellular networks," *IEEE Trans. Wireless Commun.,* vol. 12, no. 6, pp. 2706–2716, June 2013.

[36] M. Feng, S. Mao, and T. Jiang, "Joint frame design, resource allocation and user association for massive MIMO heterogeneous networks with wireless backhaul," *IEEE Trans. Wireless Commun.,* vol. 17, no. 3, pp. 1937–1950, Mar. 2018.

[37] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *American Mathematical Monthly.,* vol. 69 no. 1, pp. 9–14, Jan. 1962.

Mingjie Feng [S'15] is currently an associate professor at Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan, China. He was a postdoctoral research associate in the Department of Electrical and Computer Engineering at the University of Arizona, Tucson, AZ, USA. He received his Ph.D. degree in Electrical and Computer Engineering from Auburn University, Auburn, AL, USA, in 2018. He received his B.E. and M.E. degrees from the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China, in 2010 and 2013, respectively. He served or is serving as technical program committee membership of several IEEE conferences, including IEEE MASS, IEEE ICC, IEEE WCSP, and IEEE CCNC etc. His research focuses on the design and optimization of wireless networks, the detailed directions include heterogeneous networks, mmWave communications, massive MIMO, mobile edge computing, and full-duplex communications. He is a recipient of Woltosz Fellowship at Auburn University and Best Reviewer of IEEE Transactions on Wireless Communications.

Marwan Krunz [S'93-M'95-SM'04-F'10] is a Regents Professor at the University of Arizona. He holds the Kenneth VonBehren Endowed Professorship in ECE and is also a professor of computer science. He directs the Broadband Wireless Access and Applications Center (BWAC), a multi-university NSF/industry center that focuses on next-generation wireless technologies. He also holds a courtesy appointment as a professor at University Technology Sydney. He previously served as the site director for the Connection One center. Dr. Krunz's research is on resource management, network protocols, and security for wireless systems. He has published more than 300 journal articles and peer-reviewed conference papers, and is a named inventor on 12 patents. His latest h-index is 61. He is an IEEE Fellow, an Arizona Engineering Faculty Fellow, and an IEEE Communications Society Distinguished Lecturer (2013-2015). He received the NSF CAREER award. He served as the Editor-in-Chief for the IEEE Transactions on Mobile Computing. He also served as editor for numerous IEEE journals. He was the TPC chair for INFOCOM'04, SECON'05, WoWMoM'06, and Hot Interconnects 9. He was also the general vice-chair for WiOpt 2016 and general co-chair for WiSec'12. Dr. Krunz is an entrepreneur, served/currently serving as chief scientist for two startup companies that focus on 5G and beyond systems and machine learning for wireless communications.

Wenhan Zhang [S'19] Wenhan Zhang received his B.S. degree in Electrical Engineering and Automation from Hefei University of Technology, Heifei, China, in 2016. He earned his M.S. degree in Electrical Engineering from Syracuse University, Syracuse, NY, USA, in 2018. He is working toward his Ph.D. in Electrical and Computer Engineering at the University of Arizona, Tucson, AZ, USA. His research interests include mobile edge computing, wireless communications, and applications of machine learning in wireless networks.