An Online Framework for Ephemeral Edge Computing in the Internet of Things

Gilsoo Lee, *Member, IEEE*, Walid Saad, *Fellow, IEEE*, Mehdi Bennis, *Fellow, IEEE*, Cheonyong Kim, *Member, IEEE*, and Minchae Jung, *Member, IEEE*

***Abstract*—In the Internet of Things (IoT) environment, edge computing can be initiated at anytime and anywhere. However, in an IoT environment, edge computing sessions are often *ephemeral*, i.e., they last for a short period of time and can often be discontinued once the current application usage is completed or the edge devices leave the system due to factors such as mobility. Therefore, in this paper, the problem of *ephemeral edge computing* in an IoT is studied by considering scenarios in which edge computing operates within a limited time period. To this end, a novel online framework is proposed in which a source edge node offloads its computing tasks from sensors within an area to neighboring edge nodes for distributed task computing, within the limited period of time of an ephemeral edge computing system. The online nature of the framework allows the edge nodes to optimize their task allocation and decide on which neighbors to use for task processing, even when the tasks are revealed to the source edge node in an online manner, and the information on future task arrivals is unknown. The proposed framework essentially maximizes the number of computed tasks by jointly considering the communication and computation latency. To solve the joint optimization, an online greedy algorithm is proposed and solved by using the primal-dual approach. Since the primal problem provides an upper bound of the original dual problem, the competitive ratio of the online approach is analytically derived as a function of the task sizes and the data rates of the edge nodes. Simulation results show that the proposed online algorithm can achieve a near-optimal task allocation with an optimality gap that is no higher than 7.1 % compared to the offline, optimal solution with complete knowledge of all tasks.**

***Index Terms*—Competitive ratio, edge computing, internet of things (IoT), online optimization, task allocation.**

I. Introduction

Next-generation wireless networks will bring in new Internet of Things (IoT) services that can potentially transform people's daily lives [2], [3]. Much of these emerging IoT and 5G (fifth generation of wireless communications) services require low latency in terms of both communication and computing. To deliver low-latency IoT services, one can resort to edge computing [4], [5] techniques that can use radio and computing resources at a network edge[1].

In particular, by using local computing resources, edge computing can significantly reduce the distance of data transmission, thus inducing smaller communication latency. To enable large-scale and distributed edge computing among heterogeneous devices, there is a need to enable edge devices to pool their computing resources by instantaneously forming a local edge network to process the computational tasks received from various user applications [4]. Clearly, if properly deployed, edge computing will bring forth key benefits for low-latency IoT services by ensuring that a local edge network is instantaneously deployed by edge devices. Therein, fundamental challenges include joint radio and computing resource management and application-oriented edge computing system and architecture design.

A. Related Work

1) Edge computing in general IoT environments: Edge computing enables a diverse set of IoT services ranging from real-time IoT applications running on user devices to safety applications operating on connected vehicles [32]. Recently, a number of edge computing proof of concepts have been implemented for various IoT applications such as network resource management [12], IoT application deployment [13], and multimedia data caching [14]. The work in [16] showed how one can deploy, in the real world, edge devices with powerful computing resources and an inherent capability of running computation intensive applications. Recent prior works in [6]–[11], [15] studied deployment scenarios and resource allocation problems for standard edge computing in static or low-mobility networks. In particular, the work in [6] proposed an edge computing platform deployed in network infrastructure nodes such as base stations to provide contents to users while maintaining a required quality-of-service. Meanwhile, the authors in [7] studied the problem of joint computational task offloading and radio resource allocation in a wireless powered edge computing system by using deep learning. The work in [11] introduced a caching scheme so as to maximize fairness for an edge computing environment consisting of heterogeneous devices with different communication and computing resources. The authors

A preliminary version of this paper was presented in [1].

This research was supported by the U.S. National Science Foundation under Grant CNS-1814477 and by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2021R1C1C1012950).

G. Lee and W. Saad are with Wireless@VT, Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061 USA (e-mail: {gilsoolee, walids}@vt.edu).

M. Bennis is with Centre for Wireless Communications, University of Oulu, Finland (email: mehdi.bennis@oulu.fi).

C. Kim and M. Jung (corresponding author) are with the Department of Electronics and Information Engineering, Sejong University, Seoul (e-mail: {cykim0807, mcjung}@sejong.ac.kr).

[1]According to the network environment and application scenario, the network's edge can include various entities such as border routers, access points, base stations, mobile devices, and connected vehicles. In this study, we focused on an edge network consisting of mobile nodes.

TABLE I: Comparison with related works in edge computing. (✓: considered, -: not considered)

	Radio resource allocation	Multiple edges	Edge mobility	Computation heterogeneity	Time constraints
[6]	-	-	-	-	-
[7]–[10]	✓	-	-	-	-
[11]	-	-	-	✓	-
[12]–[14], [15]	✓	✓	-	-	-
[16]	✓	✓	-	✓	-
[17], [18], [19]	✓	-	✓	-	-
[20], [21]	-	-	✓	-	-
[22]	✓	✓	✓	-	-
[23], [24]	✓	-	✓	✓	-
[25], [26]	✓	✓	✓	✓	-
[27]–[31]	-	✓	-	-	-
Our work	✓	✓	✓	✓	✓

in [8] proposed a Lyapunov optimization-based computation offloading algorithm to jointly control transmit power and CPU (Central Processing Unit)-clock speeds when edge computing devices are powered by energy harvesting techniques. The work in [9] studied a partial computational task offloading and radio allocation problems are jointly studied. Moreover, in [15], a joint strategy of computational offloading and content caching is proposed to maximize the utilization of each edge node radio and computing resources when the statistical information on the content request is previously known. In [10], the authors used edge computing for enhancing virtual reality services.

2) Edge computing with high mobility: The works in [17]–[31] studied various problems related to edge computing in IoT networks that integrate highly mobile devices such as unmanned aerial vehicles (UAVs) and connected vehicles. First, in [17]–[26], the authors studied the use of UAVs for wireless and computing scenarios. For instance, the authors in [17] proposed a framework that jointly optimizes UAV placement and uplink power control so that UAVs can collect edge data from ground sensors. In [20], the authors employed UAVs as edge message ferries that collect information in wireless sensor networks and carry the data to the destination. In [18], [19], [21]–[26], the authors proposed various use cases for deploying airborne edge computing using a UAV. In [18], the authors investigated a UAV-mounted cloudlet in which UAVs equipped with a computing processor offload and compute the tasks offloaded from ground devices. The work in [19] studied a UAV-enabled mobile edge computing system in which the users harvest the energy from the signal transmitted by the UAV in downlink, and the harvested energy is used to transmit in uplink. The work in [21] investigated a UAV-enabled edge computing system in which a UAV offloads computational tasks from users and decides whether to compute the tasks or transmit the tasks to a remote server. In [23] and [24], the authors proposed a UAV-aided multi-access edge computing (MEC) system in which a UAV acts as an edge server (or cloudlet) providing computation service for the ground devices. On the other hand, in [25] and [26], multiple UAVs are assumed to act as edge computing devices which cooperatively compute tasks offloaded by ground devices. Also, the authors in [22] studied the joint problem of user association and computational task allocation in a mobile edge computing system where UAVs act as edge computing devices. Hence, the role of UAVs is changeable and determined depending on the considered network environment. In this paper, we focus on a scenario in which one of UAVs acts as a edge server and the rest of them act as edge computing devices. This scenario implies that the considered UAVs are not as powerful as a high performance computing server which can compute all tasks alone, however, they can compute a few tasks faster than other IoT devices such as sensors.

Next, edge computing is investigated in various scenarios incorporating connected vehicles [27]–[31]. The authors in [27] developed a distributed reputation management system in which the edge computing resources are allocated in a way to optimize security. The work in [28] proposed a low-complexity computation offloading algorithm that minimizes the computing cost at connected vehicles. Also, the work in [29] proposed the use of edge computing techniques to process the computational tasks required in a blockchain system by using the local computing resources of vehicular nodes. The authors in [30] developed a smart contract deployed on an edge computing system to enable connected vehicles to store and share the data securely. In [31], the authors applied a software-defined networking concept to develop an edge computing architecture in which the control plane protocol is designed to cluster a set of neighboring vehicles and a centralized edge computing server is used to optimize the data transmission path.

3) Limited time constraints within edge computing: The aforementioned prior works [6]–[11], [15], [17]–[31] assume that edge computing operates during a relatively long time period, and they do not consider a constraint on the total edge computing time period. However, in IoT scenarios, edge computing can be initiated and discontinued at any time due to the completion of running an application or the mobility of the edge nodes such as drones and vehicles. To capture such use cases, we propose the concept of *ephemeral edge computing* in which edge computing occurs among IoT devices that have a stringent time constraints within which they can perform edge computing. In Table I, we provide a comprehensive comparison between our work and the existing works on computation offloading in edge computing.

Next, we first provide the real-world examples of ephemeral edge computing scenarios and, then, we outline our key contributions in this area.

This article has been accepted for publication in IEEE Transactions on Wireless Communications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TWC.2022.3208096

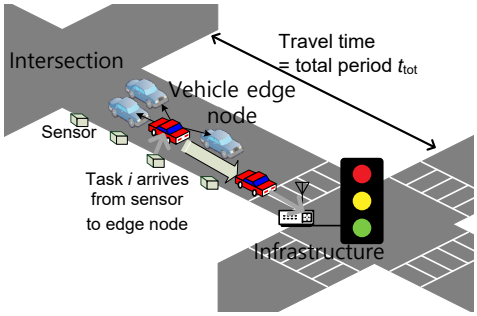

Fig. 1: Illustrative example of ephemeral edge computing framework in intelligent transporation systems.

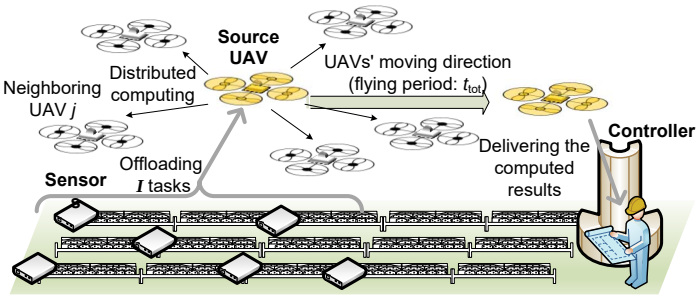

Fig. 2: Illustrative example of ephemeral edge computing framework in smart factory.

B. Ephemeral Edge Computing

In real world applications, various edge devices can be used to form a local edge network spontaneously and process computational tasks of different applications. One common observation here is that the total time period is limited in real-world IoT examples. In particular, the running time of a local edge network can be limited due to mobility of edge devices. Also, when edge computing is initiated to operate an IoT user's application, the usage time of the application can be finite. Therefore, we introduce a notion of *ephemeral edge computing* to capture cases in which edge computing occurs in a relatively short time period. Here, we note that there exists a suite of industry products related to edge computing (e.g., from Nokia or Amazon). However, these products are mostly related to infrastructure-based edge computing, and to our knowledge, they have not been yet exploited to deploy a concept such as ephemeral edge computing. Meanwhile, the emerging O-RAN standard [33] will have capabilities to support short-lived computing transactions, however, O-RAN does not provide any ephemeral edge computing solution that can leverage these capabilities, as such solutions are left to the research community, which motivates the timeliness and need for this work. As discussed next, the concept of ephemeral edge computing admits many real-world IoT applications in several industrial and civilian areas in which total time period available for the use of edge computing is constrained.

1) Intelligent transportation systems: As shown in Fig. 1, edge computing can be applied to an urban road environment in which a number of sensors monitor the status of the road traffic, vehicle flow, and pedestrian generating a large data volume [31]. For example, the generated sensory data from the road environment can be used to detect the current traffic status or to predict safety hazards. Moreover, the generated data can also be used to decide the signal light timing and schedule the vehicles at a merging ramp or intersection [27]. Therefore, processing the sensory data from a road environment is essential to optimize and control the various physical components of transportation systems. In a road environment, since the road sensors have a low computational capability, edge computing on the vehicles can be used to offload the sensory data from environment. Then, the data is processed to extract meaningful information such as traffic forecast and safety warnings [34], [35]. Once the data is processed by the vehicles' on-board computers, the vehicles can transmit the processed information to adjacent road side unit (RSU) that can then use the processed information to control traffic flows. Therefore, intelligent transportation systems provide an important use case for ephemeral edge computing. In an urban environment such as the one shown in Fig. 1, a set of vehicles move from an intersection to the next intersection while maintaining a formation. When edge computing is implemented on the vehicles, it can only be maintained for a limited time period due to mobility. Those vehicles can cooperatively process the offloaded data within a limited time period that is the travel time between two intersections. Therefore, these vehicles will form an ephemeral edge computing network. In this case, the total time period dedicated to edge computing in a vehicular network will be affected by the vehicles' speed and trajectory. In particular, the vehicles can share the information on the destination and trajectory to estimate the time period during which a set of vehicles moving the same direction. This is just one example of edge computing among many others in the context of transportation systems.

2) Smart factory: In emerging smart factory scenarios, also known as Industry 4.0 [36], sensors can detect malfunctions and send diagnostics signals to actuators in the factory. Therefore, factory systems must be optimized to manage the process of sensory data transmission, low-latency computation, and proactive decision making in order to quickly react to new situations [37]. Some key challenges for enabling the smart factory vision include effective in-network computing and improvement of wireless connectivity to integrate physical and digital systems, i.e., networking and computation. *Computing sensory data in a timely manner* is essential to operate a physical factory system. To this end, the concept of ephemeral edge computing can be applied in cyber-physical smart factory systems where UAVs, robots, and drones are deployed and perform key functions such as data storage, computing, control, and transmission [38].

As shown in Fig. 2, we consider a smart factory in which sensors monitor the status of the manufacturing process and generate a large data volume. For example, the generated sensory data can be used as an input to machine learning algorithms, e.g., for classification, to predict any abnormality in the manufacturing process. Hence, a number of computational tasks must be processed in order to make a decision on how

to control the physical systems of the factory based on the information extracted from the data. However, due to the low computational capability of the sensors, it is not possible to compute those tasks locally at the sensors. Also, sensors are not able to transmit data over a long distance, and, hence, a flexible relay is necessary [39]. For example, edge-enabled UAVs can be used in a smart factory to gather the tasks from the sensors, compute the tasks, and deliver the computed results to the destination, e.g., a central factory controller that can control the actuators. This is a meaningful use case of ephemeral edge computing in that the local edge network can be maintained until the UAVs arrive at the destination. Here, the total time period of ephemeral edge computing corresponds to the moving time from the source location to destination.

3) IoT sensor systems for end users: Consider an IoT environment in which the generated sensory data from the IoT devices is used to control and monitor the status of home appliances, to detect a user's motion and voice [40], or to run gaming and augmented reality applications at a museum, sport events, and sightseeing places [32]. Those applications require processing and analysis of the real-time IoT data. In particular, augmented reality and gaming applications must process the data depending on the user's location and orientation. In this case, the time duration within which a user's device is at a stable location in space can be relatively short, and ephemeral edge computing is needed to process the IoT data in a limited time period.

As a result, the aforementioned examples in this section show that: a) Ephemeral edge computing admits a diverse set of IoT applications and b) in these applications, the time period dedicated to ephemeral edge computing can be limited depending on the various factors such as mobility and usage patterns of applications. When the total time period of ephemeral edge computing is limited, there is a need for a new approaches to efficiently allocate the radio and computing resources to process a maximum number of computational tasks while considering the time-sensitive nature of the system.

C. Contributions

In all of these existing works on edge computing [6]–[31], it is generally assumed that edge computing is formed and used for a relatively long time period, and, therefore, the total computing time of edge computing is not considered. As shown in the real-world examples of ephemeral edge computing, edge computing can be initiated and discontinued at any time, resulting in the finite total time period to use edge computing. Therefore, we propose the concept of *ephemeral edge computing* in which the total edge computing time is limited. Also, the prior art on edge computing employing both communications and computing [6]–[11], [15], [17]–[31], generally assumes that information on prospective computing tasks such as data size and arriving order is completely known. However, in practice, the information on tasks can be revealed gradually over time since sensory data is randomly generated. Hence, when a series of tasks are offloaded to a neighboring edge node, predicting prospective future tasks is often not possible. Moreover, instead of offloading the computational tasks to base stations that are connected the servers, as done in [6]–[8], and [27]–[31], the tasks can be offloaded to neighboring edge devices by using device-to-device (D2D) communications so as to reduce a communication latency. Furthermore, instead of relying on a single edge node for computing, as done in [18], [19], [21], it is beneficial to leverage multiple, neighboring edge nodes for distributed computing of tasks. Consequently, unlike the existing literature [6]–[11], [15], [17]–[31] which assumes full information knowledge on tasks and adopts either single edge node computing models or the models placing edge computing at the base stations, our goal is to design an *online approach* to maximize the number of computed tasks on a network of multiple end-user edge nodes engaged in an ephemeral edge computing network in which there is a strict and limited total edge-computing time, when the information on tasks is revealed in an online manner.

The main contribution of this paper is a *novel framework for distributed ephemeral edge computing* that can be operated within a limited time period, as needed in the applications of Figs. 1 and 2. In particular, our framework allows tasks from sensors to be offloaded to a source edge node, which can subsequently allocate tasks to neighboring edge nodes for computation before the source node finishes edge computing. When the exact information on the offloaded tasks is unknown to the source node, it is challenging to decide which neighboring edge node has to compute which task. If a prior information on the task size is known to the source node, the computation delay at each neighboring edge node can be determined and the source node will allocate the tasks to the edge nodes according to their computational speed and the size of the tasks. However, in practice, the computational tasks arrive dynamically to the source edge node under a real-time process (i.e., online process) and their different data sizes cannot be known in advance. Therefore, we formulate an online optimization problem whose goal is to maximize the number of computed tasks when the total time period dedicated to ephemeral edge computing is constrained. To solve this problem without any prior information on the future task size, we propose a new online greedy algorithm that is used by the source edge node to make an on-the-fly decision for selecting one of the neighboring node upon the sequential arrival of the computational tasks while a prior information on the task size is unknown. Then, we analyze the performance of the proposed algorithm by using the notion of competitive ratio; defined as the ratio between the number of computed tasks achieved by the proposed algorithm and the optimal number of computed tasks that can be achieved by an offline algorithm. To this end, we apply the concept of primal-dual approach where the ratio between the dual problem and the original problem constitutes a competitive ratio. Therefore, we derive dual problem so as to analyze the worst-case performance of the proposed online algorithm. By doing so, the worst-case competitive ratio can be derived as a function of the task sizes and the communication and computing performance of the neighboring edge nodes. Simulation results show that the proposed online algorithm can maximize the number of computed tasks and achieve a performance that is near-optimal compared to an offline solution that has full information on

tasks.

The rest of this paper is organized as follows. In Section II, we present the system model. Section II-B formulates the proposed online problem. Section III presents our proposed solution and performance analysis. Simulation results are analyzed in Section IV while conclusions are drawn in Section V.

II. System Model and Problem Formulation

A. System Model

We consider an ephemeral edge computing system in which sensors generate a set $\mathcal{I}$ of I tasks[2] that are offloaded to a given edge node that we refer to hereinafter as the *source edge node*. The source edge node can be seen as a node with mobility such as vehicles and UAVs. Also, the source edge node can be a static node. While the scenarios that can use ephemeral edge computing are diverse, the role of the source edge node is to offload the computational task data from the sensors and allocate them to neighboring edge nodes. Then, each neighboring edge node directly delivers the computed result to the destination, such as a central controller in a smart factory or an RSU in intelligent transportation systems. Finally, the destination collects the computed tasks from the neighboring edge nodes and makes a decision on how to control the physical systems of the factory based on the collected data. When tasks reach the source edge node, they are labeled by their order of arrival. Thus, a task that arrives a time instant i is denoted as task $i \in \mathcal{I}$. Since the source edge node processes the tasks using a first-input-first-output policy, it will sequentially compute its tasks. The set $\mathcal{J}$ denotes the set of J edge nodes that are neighbors to the source. Each edge node $j \in \mathcal{J}$ is used to compute some allocated task i from the source edge node. We also consider that the set of neighboring edge nodes $\mathcal{J}$ is initially selected by the source edge node. In this regard, the source edge node selects the neighboring edge nodes that are moving towards its same destination. Note that the term "one task" used here can be seen as a reference to a bundle of small tasks, making it possible to execute multiple tasks at each edge node. Furthermore, the set $\mathcal{J}$ can include multiple virtual entities of an actual edge node when the number of edge nodes is too small to accommodate all of the tasks. In this case, $|\mathcal{J}| = kJ$ where k is the number of virtual entities and J is the number of actual edge nodes. The virtual entities of an actual edge node would then have to share the decision variables to have the same priority, and the edge nodes would execute their tasks in a round-robin manner. The association between the source edge node and neighboring edge nodes can be established based on the clustering algorithm proposed in [41], in which the cluster, cluster head, and cluster members correspond, respectively, to the ephemeral edge computing system, source edge node, and neighboring edge nodes. In the considered clustering algorithm, the source edge nodes exchange their link information, such as link states, computation capacity, and mobility, and each source edge node selects the qualified neighboring edge nodes that can maintain the connectivity during t_{tot} with no computation in progress, based on the exchanged information. The source edge node is assumed to select J neighboring edge nodes that are qualified to join a local edge computing network to process the computational tasks in terms of residual battery level and computation speed. If there are no edge nodes (i.e., $J = 0$), a sensor computes its tasks by itself and transmits the results to the destination. In this paper, we use the edge node essentially for boosting the computation speed rather than for carrying data between sensors and controllers. Note that, the case in which the node is static, can easily be accommodate into our framework. For instance, a static source edge node offloads the computational task data from the sensors and allocates them to neighboring static edge nodes. Then, each static neighboring edge node calculates the allocated task and directly transmits the result to a destination. Moreover, mobile edge nodes can be dispatched to any location such as mountains and rural areas where the fixed infrastructure is not readily accessible.

The source edge node allocates the computational tasks to other neighboring edge nodes. Such distributed computing can reduce the overall computational latency when multiple tasks are computed. Also, to prevent an excessive energy consumption at neighboring edge nodes, we assume that only one task is allocated to one edge node. Therefore, when neighboring edge node j computes task i, the decision variable is set as $y_{ij} = 1$. The other edge nodes are not used to process the same task i, i.e., if $y_{ij} = 1$ then $y_{ij'} = 0, \forall j' \in \mathcal{J} \setminus \{j\}$, $\forall i \in \mathcal{I}$. Task allocation to neighboring edge node incurs a *transmission latency*. The data rate pertaining to the transmission of the data of task i to neighboring edge node j will be: $r_j = F(B, g_j, P_t, \sigma)$, where F is a general transmission rate function, P_t is the transmit power of the source edge node, B is the bandwidth, σ^2 is the noise power, and g_j is the channel gain between the source edge node and neighboring edge node j. Therefore, when the data size of task i is d_i bits, the transmission latency becomes d_i/r_j. Once task i is received by neighboring edge node j, it will be processed within a *computational latency*[3] d_i/f_j where f_j is the computation power of edge node j.

In the proposed ephemeral edge computing system, the time period that the source edge node actively uses edge computing is given by t_{tot}. To determine t_{tot}, key features of the edge nodes can be considered. From the aforementioned cases of ephemeral edge computing, the total time period t_{tot} can be determined as the moving time period of a set of edge computing vehicles on the road or UAVs in a smart factory. For example, t_{tot} can depend on the mobility that is characterized by the speed and moving distance of the source edge node. t_{tot} could also depend on the different trajectories of the source edge node. In the IoT scenarios, the total time period can be

[2]For consistency, we use the term "task" to indicate both the data generated by a sensor and the computational job that will be used to process data.

[3]One way to estimate computation latency is to define how many CPU cycles are needed for computing a bit of data. In this paper, the computation latency is defined as $\alpha \cdot d_i/F_j$ where α is the required number of CPU cycles per bit (i.e., computation complexity) and F_j is the CPU speed of edge node j in Hz. For notational simplicity, we introduce the computation power as $f_j = F_j/\alpha$. Therefore, the computation delay can be simply expressed as d_i/f_j.

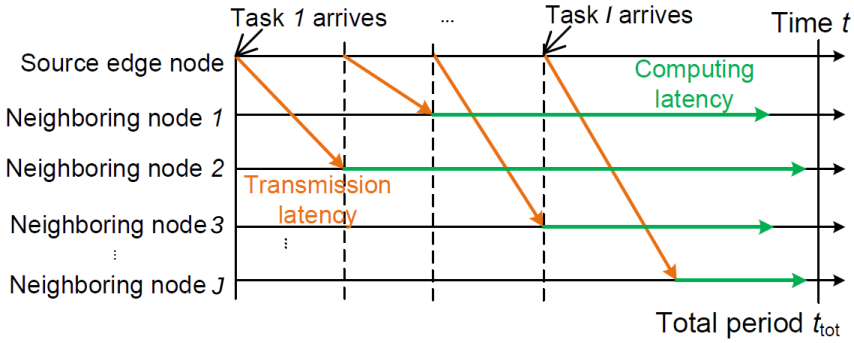

Fig. 3: Online edge computing framework to offload computational tasks and allocate the offloaded tasks to neighboring edge nodes in total edge computing period t_{tot} within an ephemeral edge computing system.

given as the running time of an application or the time period where a smart device is staying near the other edge devices to deploy an edge computing network. For a given t_{tot}, if a certain number of tasks is processed as shown in Fig. 3, then the tasks' transmission and computation must be completed within t_{tot}. We define the duration between the arrival of the first task and completion of task i as the *completion time* of task i. As shown in Fig. 3, the tasks are sequentially offloaded from the source edge node to one of neighboring nodes. For instance, when the first task, $i = 1$, is being allocated, the completion time including transmission and computation of task 1 will be:

$$\sum_{j=1}^{J} d_1 \left(\frac{1}{r_j} + \frac{1}{f_j} \right) y_{1j} \leq t_{\text{tot}}. \tag{1}$$

Subsequently, since tasks are sequentially transmitted to the neighboring edge nodes in the order of index i, there will be $i-1$ transmissions before task i is transmitted. Therefore, the completion time of any task i, $\forall i \in \mathcal{I} \setminus \{1\}$,

$$\sum_{i'=1}^{i-1} \sum_{j=1}^{J} d_{i'} \left(\frac{1}{r_j} \right) y_{i'j} + \sum_{j=1}^{J} d_i \left(\frac{1}{r_j} + \frac{1}{f_j} \right) y_{ij} \leq t_{\text{tot}}, \tag{2}$$

where the first term is the sum of the transmission latency of $i-1$ tasks, and the second term is the transmission and computation latency of task i. Given that tasks are allocated to and computed by neighboring edge nodes in the order of index i, the completion time of task i in (2) includes the summation of the transmission latency for the previous $i-1$ tasks. Next, we formulate an online task allocation problem to study how tasks are distributed within an edge computing network.

B. Problem Formulation

Our goal is to allocate tasks to neighboring edge nodes in order to complete the maximum number of tasks during the period t_{tot} needed for the source edge node to reach its destination. To compute the tasks, the source edge node must allocate each task to a neighboring edge node that can yield low latency. In practice, when the computational tasks arrive dynamically to the source edge node, their different data sizes cannot be known in advance. As a result, the source edge node will be unable to know a priori the information on future tasks, and, therefore, optimizing the task distribution process under this uncertainty is very challenging. Under such uncertainty, selecting a neighboring edge node that computes a current task must also account for potential arrival of future tasks. When the future information is revealed sequentially, the arrival of information can be captured within an online optimization framework. In particular, by using online optimization techniques such as those in [42], it is possible to make an on-the-fly decision while the future information is given in an online manner. To cope with the uncertainty of the future task arrivals while considering the data rate and computing capabilities of given neighboring edge nodes, we will thus propose a rigorous *online optimization framework* that can handle the problem of task allocation under uncertainty.

First, we formulate the following online task allocation problem whose goal is to maximize the number of computed tasks when the total latency is limited by t_{tot}:

$$\begin{aligned}
(\text{D}): \max_{\boldsymbol{y}} \quad & \sum_{i=1}^{I} \sum_{j=1}^{J} y_{ij} && (3)\\
\text{s.t.} \quad & (1), (2), \\
& \sum_{i=1}^{I} y_{ij} \leq 1, \forall j \in \mathcal{J}, && (4)\\
& \sum_{j=1}^{J} y_{1j} \leq 1, && (5)\\
& \sum_{j=1}^{J} (-y_{i-1j} + y_{ij}) \leq 0, \forall i \in \mathcal{I} \setminus \{1\}. && (6)
\end{aligned}$$

where $\boldsymbol{y}$ is the vector of decision variables $y_{ij}, \forall i \in \mathcal{I}, \forall j \in \mathcal{J}$. Hereinafter, this problem is called the dual problem. Constraints (1) and (2) show that task i's completion time must be smaller than t_{tot} and tasks that cannot satisfy those constraints will not be offloaded. (4) implies that each neighboring edge node can compute at most one task to prevent an excessive energy consumption at any given edge node. In constraint (5), the first task is allocated to one of the neighboring edge nodes. Constraint (6) implies that task i can be allocated to a neighboring edge node if the task allocation of task $i-1$ is successful, i.e., $\sum_{j=1}^{J} y_{i-1j} = 1$. Otherwise, if $\sum_{j=1}^{J} y_{i-1j} = 0$, then, task i cannot be allocated to any edge node, and $\sum_{j=1}^{J} y_{ij} = 0$. Due to (5) and (6), we have $\sum_{j=1}^{J} y_{ij} \leq 1, \forall i \in \mathcal{I}$, and, thus, each task is allocated to only one of neighboring

edge nodes. Given that the demand for mobile device has been growing exponentially in recent years, mainly driven by various emerging IoT applications, we assume that $J \geq I$ and all tasks can be completed during a limited time t_{tot} using the edge computing network, and thus, there exist some feasible solutions that satisfy all the constraints in problem (D).

Note that problem (D) is an *online optimization problem* and is challenging to solve using conventional offline approaches. This is because the value of $d_i, \forall i$, is sequentially revealed. When the tasks that different sensors send to the source edge node have a random size, the arrival sequence of d_i is assumed to be unpredictable and unknown. At the moment when d_i is disclosed, the source edge node knows only the current and past tasks. However, the source edge node must make an *instant and irrevocable online decision* on which neighboring edge node will compute task i. Under such uncertainty on d_i, allocating tasks to existing neighboring edge nodes must also account for potential arrival of new tasks. In fact, even if a given task allocation can compute an existing task successfully, it may have a detrimental effect on the allocation of incoming tasks. In particular, if an edge node having a high data rate and high computational speed is already assigned to compute a previous task, it may not be possible to compute a future task having a large size. Therefore, it is challenging to optimize the task allocation between incoming tasks and neighboring edge nodes.

In an online setting, the ad-auction problem in [42] shows a generalized structure of an online linear programming problem and its algorithmic solution. We observe that the ad-auction problem and our problem have a key difference in the dependency of the constraints. In particular, the ad-auction problem includes the independent constraints about the maximum allocation size for each buyer that corresponds to the edge node in our problem. However, in our problem, the constraints about the maximum allocation size of edge nodes are dependent on each other. For instance, in (1) and (2), the sum of the transmission latency of the previous tasks and the processing latency of the current task should be less than t_{tot}. The total time period is a function of the task allocation decisions of all edge nodes while each edge node has an independent task allocation size. Therefore, if the given budget of total time period is previously spent to offload and compute previous tasks, the source node cannot offload a new task to a neighboring node that is still available to accept a task. Additionally, our problem assumes that the arriving tasks are sequentially allocated to the neighboring node. For instance, the current task cannot be allocated to any node, if the previous task is not allocated due to constraints (5) and (6). Due to the aforementioned differences, we need to develop a novel online task allocation strategy to solve problem (D).

III. Proposed Online Task Allocation Framework

Our goal is to determine the vector of decision variables $\boldsymbol{y}$ so that the maximum number of sequentially arriving tasks is successfully computed by our distributed ephemeral edge computing system. When task size d_i is unpredictable, the decision is not trivial since the current decision may affect the task allocation of future tasks, and all tasks cannot be computed due to the limited time resource t_{tot}. In this case, making an on-the-fly online decision, can process a smaller number of tasks than that of offline decision in which the complete information on all tasks is initially known. Therefore, the gap between the results achieved by online and offline cases must be minimized. To this end, the notion of *competitive ratio* [42] from competitive analysis can be used to measure the performance of our online algorithm. It is an effective metric that compares the ratio between the the objective function's value achieved by an online algorithm and that of the offline optimal solution. In particular, the upper bound of the competitive ratio can be defined as a constant γ such that

$$1 \leq \frac{\mathrm{D}_{\text{IP,OPT}}}{\mathrm{D}_{\text{IP}}} \leq \gamma, \tag{7}$$

where $\mathrm{D}_{\text{IP,OPT}}$ denotes the offline optimal solution (OPT) of problem (D) in the form of integer programming (IP), i.e., the maximum number of computed tasks with the integer solution of y_{ij}. We will measure the performance of our proposed algorithm by observing the upper bound value defined by γ.

To find the upper bound of problem (D), we use the structure of the primal and dual approach [42]. To this end, the optimization variables y_{ij} are relaxed to be linear, i.e., $y_{ij} \in [0, 1]$. By using the duality of linear programming, problem (D) can be rewritten as:

$$(\mathrm{P}): \min_{\boldsymbol{x},\boldsymbol{z},u_1} \quad \sum\nolimits_{i=1}^{I} t_{\text{tot}} x_i + \sum\nolimits_{j=1}^{J} z_j + u_1, \tag{8}$$

$$\text{s.t.} \quad \left(\frac{1}{r_j} + \frac{1}{f_j}\right) d_i x_i + \left(\frac{d_i}{r_j}\right) \sum_{i'=i+1}^{I} x_{i'} + z_j + u_i - u_{i+1} \geq 1,$$
$$\forall i \in \mathcal{I} \setminus \{I\}, \forall j \in \mathcal{J}, \tag{9}$$

$$\left(\frac{1}{r_j} + \frac{1}{f_j}\right) d_I x_I + z_j + u_I \geq 1, \forall j \in \mathcal{J}, \tag{10}$$

$$x_i \geq 0, z_j \geq 0, u_i \geq 0, \tag{11}$$

where $\boldsymbol{x}$ and $\boldsymbol{z}$ are vectors with elements $x_i, \forall i \in \mathcal{I}$, and $z_j, \forall j \in \mathcal{J}$, respectively. This problem is called the *primal problem*. In problem (8), x_1, $x_{i\geq 2}$, z_j, u_1 and $u_{i\geq 2}$ are the dual variables associated, respectively, with constraints (1), (2), (4), (5), and (6) in problem (D).

The values of (3) and (8) are denoted by D_{IP} and P_{LP}, respectively. With D_{IP} and P_{LP}, a *competitive ratio* in (7) is derived. From the dual and primal problem formulation, it can be shown that $\mathrm{D}_{\text{IP}} \leq \mathrm{D}_{\text{LP}} \leq \mathrm{D}_{\text{LP,OPT}} \leq \mathrm{P}_{\text{LP,OPT}} \leq \mathrm{P}_{\text{LP}}$. The first inequality is due to the fact that a linear relaxation allows problem (D), which is in the form of linear programming (LP), to have a higher value. The second inequality indicates that the offline optimal solution always achieves a value higher than or equal to the online solution of problem (D). The third inequality captures the slackness of the primal and dual problems. In the fourth inequality, the offline optimal solution of problem (P), i.e., $\mathrm{P}_{\text{LP,OPT}}$ is smaller than or equal to any online solution of problem (P), i.e., P_{LP}. Also, we have $\mathrm{D}_{\text{IP}} \leq \mathrm{D}_{\text{IP,OPT}} \leq \mathrm{D}_{\text{LP,OPT}}$. The first inequality follows from the optimality gap between the online and offline solutions when y_{ij} is an integer. The second inequality shows that linear

Algorithm 1 Online Task Allocation Algorithm

Initialize $y_{ij} = x_i = z_j = u_i = 0, \forall i, j$.
for $i \in \mathcal{I}$
  Task i arrives at source node.
  Select edge node by using (12).
  if (1) and (2) are satisfied, and $\sum_j y_{i-1j} = 1$,
    $y_{ij} \leftarrow 1$.
    Allocate task i to edge node j_i^* defined in (12).
    Update z_j, x_i, and u_i, respectively, by using (13), (14), and (15).
  otherwise,
    $y_{ij} \leftarrow 0$.
    Set $\Delta u_i = 1$ and update $u_{i'}, \forall i' \leq i$
  end if
end for

relaxation of y_{ij} allows us to have a higher value in problem (D). Thus, the ratio in (7) becomes: $\frac{\mathrm{D_{IP,OPT}}}{\mathrm{D_{IP}}} \leq \frac{\mathrm{P_{LP}}}{\mathrm{D_{IP}}}$, where $\mathrm{P_{LP}}/\mathrm{D_{IP}}$ corresponds to γ in (7). Therefore, $\mathrm{P_{LP}}/\mathrm{D_{IP}}$ becomes the upper bound of the competitive ratio.

A. Online Greedy Algorithm

To find the ratio $\mathrm{P_{LP}}/\mathrm{D_{IP}}$, we develop a new online greedy algorithm (Algorithm 1) specifically designed to solve problems (D) and (P), based on a general online optimization framework using the primal and dual approach of [42]. In Algorithm 1, the decision variables y_{ij}, x_i, z_j, and u_i are updated while observing the new value of d_i. In particular, when task i arrives to the source edge node, the original dual problem is solved by determining the value of y_{ij}. Also, other dual variables x_i, z_j, and u_i are updated in order to find the performance bound of the proposed online algorithm. At the initial step of Algorithm 1, all variables are set to 0. The algorithm selects which edge node should compute task i. Since it is beneficial to offload task i from the source node to the neighbor with a high data rate and computing speed, this decision rule can be designed to select an edge node with the shortest communication and computing latency to process the task i. To this end, edge node j^* is selected by following the decision rule:

$$j^* = \arg\max_{\forall j} \frac{(1-z_j)^\alpha}{\left(\frac{1}{r_j} + \frac{1}{f_j}\right) d_i}, \tag{12}$$

where $\alpha \geq 1$ is a constant used to guarantee that at least one of the tasks can be fairly allocated among the neighboring edge nodes. The detailed derivation of the decision rule in (12) is presented in Appendix A. Since z_j is initially zero, the decision rule in (12) only considers the latency required to process task i. In Algorithm 1, if a neighboring node j accepts a task, the value of z_j is updated to become positive. By doing so, $1-z_j$ is reduced, and, hence, another neighboring node can be selected when the next task arrives. However, if the neighboring node j results in $(1-z_j)^\alpha / \left(\frac{1}{r_j} + \frac{1}{f_j}\right) d_i \geq (1-z_{j'})^\alpha / \left(\frac{1}{r_{j'}} + \frac{1}{f_{j'}}\right) d_i, \forall j' \in \mathcal{J} \setminus \{j\}$, the same node j can be selected again. This can violates constraint (4) that restricts each neighbor to accept one task. Therefore, a large value of α can be used to make $(1-z_j)^\alpha$ close to zero. Then, at the arrival of a new task, a different neighboring node is selected as j^* by using decision rule (12).

After a neighboring node j^* is selected for task i, if the time budget is still available for the current task i from constraints (1) and (2), neighbor node j^* finally receives task i from the source node and performs processing. At this moment, the dual and primal variables are updated in Algorithm 1. The algorithm sets $y_{ij^*} = 1$ showing that task i is allocated to edge node j^*. Next, the value of z_{j^*} must be updated since z_{j^*} is the primal variable associated with the dual problem's constraint (4) with $j = j^*$. When a neighboring node initially does not have any accepted task, all $z_j, \forall j \in \mathcal{J}$ are set to zero. However, if a neighboring node j accepts a task i, z_j will be updated as follows:

$$z_j = z_j \left(1 + \left(\frac{1}{r_j} + \frac{1}{f_j}\right) \frac{d_i}{t_{\mathrm{tot}}}\right) + \left(\frac{1}{r_j} + \frac{1}{f_j}\right) \frac{d_i}{t_{\mathrm{tot}}} \left(\frac{1}{c-1}\right), \tag{13}$$

where $c > 1$ is a positive constant that will be defined later. Also, the total time period t_{tot} is assumed to be enough to process at least one task, and, thus, $\left(\frac{1}{r_j} + \frac{1}{f_j}\right) \frac{d_i}{t_{\mathrm{tot}}} < 1$. Meanwhile, the update of x_i must satisfy constraints (9) and (10). The value of x_i is updated by using the rule:

$$x_i = \frac{(1-z_j)^\alpha}{\left(1/r_j + 1/f_j\right) d_i}. \tag{14}$$

Moreover, the values of $u_{i'}, \forall i' \leq i$, is updated as follows:

$$u_{i'} = u_{i'} + \Delta u_i, \forall i' \leq i, \tag{15}$$

where we define, $\forall j' \in \mathcal{J}$,

$$\Delta u_i = \max_{j' \in \mathcal{J} \setminus \{j^*\}} \left(1 - \left(\left(\frac{1}{r_{j'}} + \frac{1}{f_{j'}}\right) \frac{(1-z_{j^*})^\alpha}{\left(1/r_{j^*} + 1/f_{j^*}\right)} + z_{j'}\right), 0\right).$$

Otherwise, if the edge nodes in $\mathcal{J}$ do not satisfy (1) and (2), then, the tasks arriving after task i cannot be computed, i.e., $y_{ij} = 0$, and those tasks will not be offloaded. In this case, to satisfy constraints (9) and (10), Algorithm 1 updates any z_j that has a value of 0 to 1 if $J \leq I$, or, otherwise, Δu_i is set to 1. This update is intended to satisfy the constraints (9) and (10) for all $i \in \mathcal{I}$ and $j \in \mathcal{J}$. For the arrival of each task, the proposed algorithm is a one-shot decision making process to find a feasible solution. Therefore, by iterating the proposed algorithm for all arriving tasks during t_{tot}, our algorithm converges to a feasible solution of problem (D).

B. Performance Analysis

For the analysis hereinafter, we assume that $\alpha = 1$ for analytical tractability. In practice, this assumption implies that the decision rule (12) tends to select the neighboring node with a high data rate and computing speed. As α increases, the decision rule selects a new neighboring node that has not been used to process any previous task. Now, as a first step to derive the competitive ratio of the proposed algorithm, we find the following result.

Lemma 1. *The constraints of the primal problem* (9) *and* (10) *will be satisfied if* z_j, x_i, *and* u_i *are updated by* (14), (13), *and* (15), *respectively.*

Proof. See Appendix B. □

The next step of our analysis is to check whether the constraints in problem (D) is satisfied. In particular, since it is observable that the upper bound of the left-hand side of the constraint (4) can be greater than one, (4) is not satisfied for $\alpha = 1$, as shown next.

Lemma 2. *In* (4), $\sum y_{ij}$ *is violated by at least 2.*

Proof. See Appendix C. □

This result implies that more than two tasks can be offloaded to the same neighboring node. However, there exists a condition under which constraint (4) is satisfied.

Lemma 3. (4) *is satisfied if* $d_i > \left(\left(1/r_{j_i^*} + 1/f_{j_i^*}\right)^{-1} - \left(1/r_{j_I^*} + 1/f_{j_I^*}\right)^{-1}\right) t_{tot}(c-1)$, *where* j_i^* *is the node selected to process task* $i, \forall i \in \mathcal{I}$.

Proof. After task i is offloaded to node j_i^*, Algorithm 1 updates $z_{j_i^*} = \left(1/r_{j_i^*} + 1/f_{j_i^*}\right) \frac{d_i}{t_{\text{tot}}(c-1)}$. Next, when task $i+1$ arrives, the condition above yields the inequality $\frac{1}{\left(1/r_{j_{i+1}^*} + 1/f_{j_{i+1}^*}\right) d_{i+1}} > \frac{1}{\left(1/r_{j_I^*} + 1/f_{j_I^*}\right) d_{i+1}} > \frac{(1-z_{j_i^*})^{\alpha}}{\left(1/r_{j_i^*} + 1/f_{j_i^*}\right) d_{i+1}}, \forall i \in \mathcal{I}$ with $\alpha = 1$. Therefore, (12) is used to select a new node j_{i+1}^* to process task $i+1$. Hence, a different neighboring node is selected for each task. □

For instance, the condition in Lemma 3 can be satisfied if the value of d_i is decreasing over time. In that case, every neighboring node can be used to process different tasks, thus satisfying constraint (4). As a last step, we derive the increment rate of the $\Delta P/\Delta D$ when a new task i arrives in an online manner.

Lemma 4. *When the dual problem's objective function increases by one, the primal problem's objective function increases by* $\frac{t_{tot}}{\left(1/r_j + 1/f_j\right) d_i} \left(1 + \frac{1}{c-1}\right) + \Delta u_i$ *for any given* $c > 1$.

Proof. See Appendix D. □

Now, to derive a competitive radio for the proposed algorithm, we will adopt a primal-dual online analysis analogous to the one done in [42]. In Lemma 1, it is shown that the primal variable is updated while satisfying the constraints (9) and (10). Then, we show that the dual constraints from (1) to (6) are satisfied under the derived condition in Lemma 3. Finally, the increment rates of the primal and dual problems are, respectively, derived in Lemma 4. As a result, from Lemmas 1, 3, and 4, we obtain the following key result:

Theorem 1. *The upper bound of the competitive ratio in Algorithm 1 is* $O(1/\min_i \beta_{ij})$ *where* $\beta_{ij} \triangleq \left(\frac{1}{r_j} + \frac{1}{f_j}\right) \frac{d_i}{t_{tot}}$ *if* $d_i > \left(\left(1/r_{j_i^*} + 1/f_{j_i^*}\right)^{-1} - \left(1/r_{j_I^*} + 1/f_{j_I^*}\right)^{-1}\right) t_{tot}(c-1)$.

Proof. Lemma 1 first shows that the constraints of problem (P) are satisfied for all tasks that are assigned to the set of edge nodes. At each iteration, Lemma 3 shows that the increment of $\Delta P/\Delta D$ is at most

$$\frac{\Delta P}{\Delta D} \leq \frac{1}{\min_i \beta_{ij}} \left(1 + \frac{1}{(1+\delta)^{\frac{1}{\delta}} - 1}\right) + \max_i \Delta u_i, \qquad (16)$$

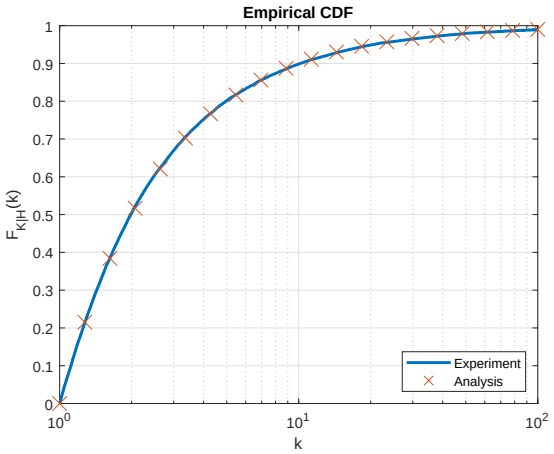

Fig. 4: Example of the cumulative probability distribution of $F_{K|H}(k)$.

where $\beta_{ij} = \left(\frac{1}{r_j} + \frac{1}{f_j}\right) \frac{d_i}{t_{\text{tot}}}$. Also, (16) has an upper bound at $\delta = 1$. Since $D_{IP} = \sum_{\forall i,j} y_{ij}$, the future tasks $i > D_{IP}$ cannot be allocated to any neighbor. In that case, Algorithm 1 sets $\Delta u_i = 1$. Then, all values of $u_{i'}, \forall i' \leq i$ increase by one, resulting in $\Delta D = 0$ and $\Delta P = 1$. Thus, we have $\gamma \leq \frac{\Delta P}{\Delta D} + (I - D_{IP})$. We observe that $\Delta P/\Delta D$ increases with the rate of $O(1/\min_i \beta_{ij})$ as $\beta_{ij} \to 0$. At the same time, $I - D_{IP}$ can decrease with D_{IP} when the number of processed tasks increases. Hence, the ratio γ can be bounded by $O(1/\min_i \beta_{ij})$. □

This result characterizes the online performance bound achieved by Algorithm 1 in which γ can decrease as $\min_i \beta_{ij}$ approaches 1. If $\min \left(\frac{1}{r_j} + \frac{1}{f_j}\right) \frac{d_i}{t_{\text{tot}}} \approx 1$, we have an environment in which all neighboring edge nodes have similar communication and computing performance, thus resulting in the smallest γ close to 1. In such a case, the online and offline performance gap is minimized. Also, Algorithm 1 can be usefully converted into another simple algorithm that updates $\Delta u_i = 0$ for all tasks $i \in \mathcal{I}$ so that problem (P) has a value of I, by assuming $\left(\frac{1}{r_j} + \frac{1}{f_j}\right) \frac{d_i}{t_{\text{tot}}}, \forall i, j$, equals to 1. This algorithm shows that the competitive ratio is inherently upper bounded by $P_{\text{LP}}/D_{\text{IP}} = I$ in the worst case.

As shown in Theorem 1, it is essential to investigate how the value of $1/\beta_{ij}$ is determined when measuring a realistic performance of the proposed ephemeral edge computing system. We conduct a statistical analysis to derive the probability corresponding to different values of $1/\beta_{ij}$. To this end, it is assumed that the data rate and task size are randomly determined. In particular, the size of data d_i is generated by following a uniform distribution random variable $D \sim U(0, D_{\max})$ where $D_{\max}$ is the maximum size of a task. We assume that the data rate is denoted by a random variable $R \triangleq \log_2(1+P)$ where P is the received power in a fading channel modeled as an exponential distribution with parameter λ, i.e., $P \sim \exp(\lambda)$. This statistical model is a simplified version of our edge computing system model. This statistical modeling facilitates the observation of factors that affect the performance of the proposed algorithm. Then, we derive the probability to have a certain value of $1/\beta_{ij}$.

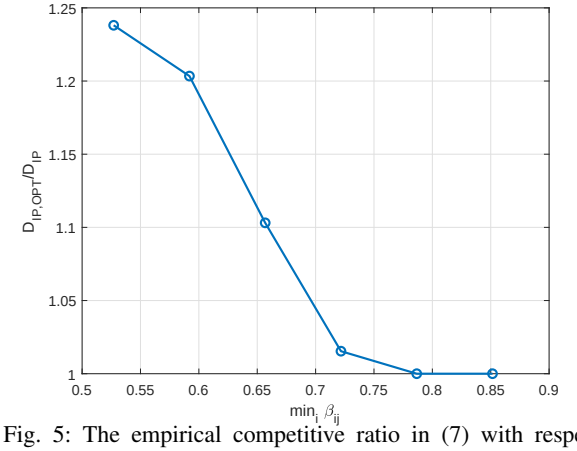

Fig. 5: The empirical competitive ratio in (7) with respect to the different values of $\min_i \beta_{ij}$ when $\alpha = 1$.

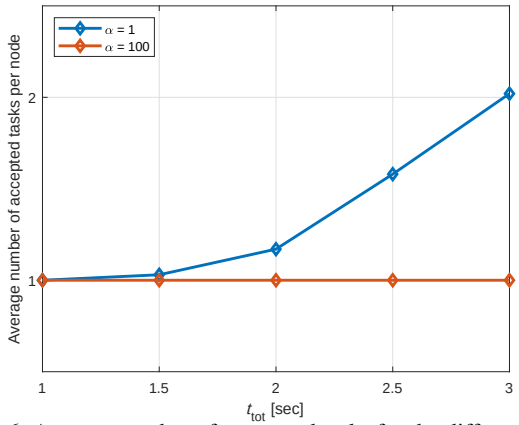

Fig. 6: Average number of computed tasks for the different total times when $\alpha = 1$ and 100.

Theorem 2. *If* $k \geq \frac{t_{\text{tot}} f}{D_{\max}}$, *the probability that* $1/\beta_{ij} \leq k$ *is* $(F_K(k) - F_K(1))/(1 - F_K(1))$ *where*

$$F_K(k) = \frac{1}{D_{max}} \left[\int_0^{\frac{t_{tot} f}{k}} \left(1 - \exp\left(-\lambda \left(2^{\frac{1}{\frac{t_{tot}}{kx} - \frac{1}{f}}} - 1 \right) \right) \right) dx + \left(D_{max} - \frac{t_{tot} f}{k} \right) \right]. \quad (17)$$

Proof. We define a random variable $K \triangleq \frac{t_{\text{tot}}}{\left(\frac{1}{\log_2(1+P)} + \frac{1}{f} \right) D}$. Therefore, if $k \geq \frac{t_{\text{tot}} f}{D_{\max}}$, the cumulative density function of a random variable K is shown as:

$$\begin{aligned} F_K(k) &= \Pr\left(\frac{t_{\text{tot}}}{\left(\frac{1}{\log_2(1+P)} + \frac{1}{f} \right) D} > k \right) \\ &= \int_0^{D_{\max}} \Pr\left(\frac{t_{\text{tot}}}{\left(\frac{1}{\log_2(1+P)} + \frac{1}{f} \right) x} > k | D = x \right) \Pr(D = x) dx \\ &= \frac{1}{D_{\max}} \left[\int_0^{\frac{t_{\text{tot}} f}{k}} \left(1 - \exp\left(-\lambda \left(2^{\frac{1}{\frac{t_{\text{tot}}}{kx} - \frac{1}{f}}} - 1 \right) \right) \right) dx \right. \\ &\quad \left. + \left(D_{\max} - \frac{t_{\text{tot}} f}{k} \right) \right]. \end{aligned} \quad (18)$$

When H is defined as the event in which $K \geq 1$, the cumulative density function of a random variable K conditioned on H is

$$F_{K|H}(k) = \frac{\Pr(K \leq k \cap K \geq 1)}{\Pr(K \geq 1)} \quad (19)$$

$$= (F_K(k) - F_K(1))/(1 - F_K(1)). \quad (20)$$

□

When the tasks are randomly generated and wireless performance dynamically changes, Fig. 4 shows an example of the cumulative probability distribution of $F_{K|H}(k)$ when $t_{\text{tot}} = 2$, $1/f = 0.5$, and $D_{\max} = 4$. In Fig. 4, if $k = 2$, the probability that $k = 1/\beta$ is less than 2 is around 50 %. Therefore, the probability that k becomes the empirical value of a competitive ratio in Theorem 2 is: $\Pr(1/\min_i \beta_{ij} \leq k) = \Pr(\max_i 1/\beta_{ij} \leq k) = (F_{K|H}(k))^I$. Also, from Theorem 2, the derived probability does not change if the total time period is equal to the processing time of the maximum task size, i.e., $t_{\text{tot}} = D_{\max}/f$. Hence, if an ephemeral edge computing system is designed to use the maximum task size given by $t_{\text{tot}} f$, it is possible to expect the empirical value of the competitive ratio when the data rate and task size are randomly determined in a wireless environment.

IV. Simulation Results and Analysis

For our simulations, we use a MATLAB simulator in which we consider that the source edge node initially forms a network with $J = 10$ neighboring edge nodes uniformly distributed within a circular area of radius between 10 m and 100 m. For instance, this can be seen as a generalized scenario in which an edge-enabled UAV (or vehicle) forms an edge network with J neighboring nodes in a smart factory (or on a road environment). The task size follows a uniform distribution between 50 and 100 Mbits, and the number of tasks is $I = 10$. The power spectral density of the noise is -174 dBm/Hz, the carrier frequency is 2.1 GHz, and $P_t = 20$ dBm. The computational speed of each neighboring edge node is randomly determined from a uniform distribution between 1×10^8 and 5×10^8 bits/sec and we assume $r_j = B \log_2\left(1 + \frac{g_j P_t}{\sigma^2}\right)$. The offline optimal solution is calculated by using a mixed-integer linear programming (MILP) solver with the assumption that the size d_i of task i, $\forall i \in \mathcal{I}$, is completely known. All simulations are statistically averaged over 5000 independent runs.

Fig. 5 first shows the empirical ratio between the offline optimal and online solutions, $D_{IP,OPT}/D_{IP}$ for the different values of $\min_i \beta_{ij}$ when $t_{\text{tot}} = 1$, $\alpha = 1$, and $f_j \in [7 \times 10^7, 10 \times 10^7]$. The numerical results in Fig. 5 confirm that the ratio $D_{IP,OPT}/D_{IP}$ decreases as $\min_i \beta_{ij}$ increases as shown in Theorem 1. For example, the empirical competitive ratio can be reduced up to 19.2 % if the smallest β_{ij} increases from 0.58 to 0.85. Also, in Fig. 5, the cases in which the ratio is one correspond to scenarios in which the proposed algorithm finds the optimal solution. For instance, when $\min_i \beta_{ij}$ is greater

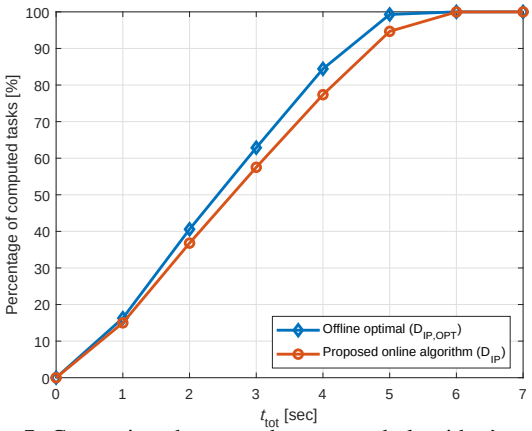

Fig. 7: Comparison between the proposed algorithm's result and the offline optimal solution in terms of percentage of computed tasks for different t_{tot}.

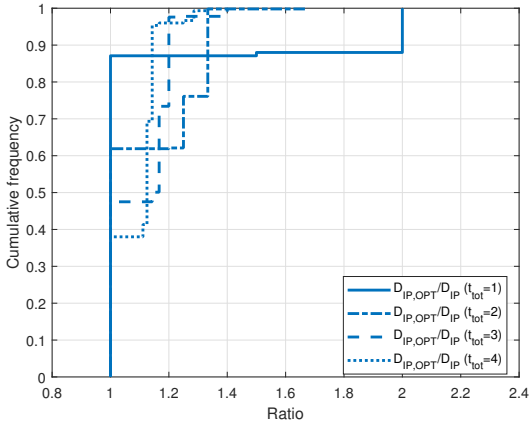

Fig. 8: The empirical competitive ratios $D_{IP,OPT}/D_{IP}$ when $t_{\text{tot}} = 1, 2, 3, 4$.

than 0.79, Fig. 5 shows that the empirical ratio becomes one since $D_{IP,OPT} = D_{IP}$.

Fig. 6 shows the average number of accepted tasks per node, i.e., $\sum_i y_{ij}$, for two values of $\alpha = 1$ and 100. In Fig. 6, the number of accepted tasks per node needs to be one due to constraint (6). When $0 < z_j < 1$, the selection rule in (12) can decide to offload a new task to a neighboring node that already accepted a task. In particular, Fig. 6 shows that the average number of accepted tasks per node increases with t_{tot} for $\alpha = 1$. This is due to the fact that the selection rule in (12) is affected by two factors, i.e., $(1 - z_j)^\alpha$ and $1/\left((1/r_j + 1/f_j)\frac{d_i}{t_{\text{tot}}}\right)$ where $(1 - z_j)^\alpha$ prevents the algorithm from choosing the same node multiple times. It is observable that $1/\left((1/r_j + 1/f_j)\frac{d_i}{t_{\text{tot}}}\right)$ increases as t_{tot} increases. Therefore, with a large t_{tot}, the selection rule in (12) is determined by $1/\left((1/r_j + 1/f_j)\frac{d_i}{t_{\text{tot}}}\right)$, rather than $(1-z_j)^\alpha$. For example, Fig. 6 shows the average number of accepted tasks can reach up to 2 when t_{tot} increases from 1 to 3. Thus, to avoid offloading more than one task to the same neighboring node, a large α is used in Fig. 6. If α is set to a large value, e.g., 100, Fig. 6 shows that the selection rule in (12) only offloads the tasks to different nodes. This is due to the fact that $(1 - z_j)^\alpha$ is close to zero for a large α when $0 < z_j < 1$. For instance, when $\alpha = 100$, the average number of accepted tasks is 1 for all t_{tot}. To evaluate Algorithm 1 in a general task arrival, $\alpha = 100$ is used for the rest of our simulations.

Fig. 7 shows the percentage of computed tasks for different values of t_{tot} from 0 to 7 seconds when the total bandwidth is 10 MHz. For comparison, we calculate the offline optimal solution of the dual integer problem, i.e., $D_{IP,OPT}$, by assuming that all task sizes, d_i, $\forall i$, are known in advance. The offline optimal $D_{IP,OPT}$ shows that the percentage of computed tasks increases with t_{tot} that is a given parameter in problem (D). The design goal of our online algorithm is to achieve a performance that is similar to the offline optimal when the task size d_i is revealed one by one. To this end, in Fig. 7, we can observe that the optimal solution and the solution found by Algorithm 1 are very close for all values of t_{tot}. This demonstrates the effectiveness of the proposed algorithm that can select properly neighboring edge nodes to offload tasks while maximizing the number of computed tasks. For instance, Fig. 7 shows that the maximum gap between the offline optimality and the online solution is only 7.1 % when $t_{\text{tot}} = 4$. Also, in Fig. 7, as t_{tot} increases, more tasks can be readily processed within a given time period, and, therefore, the percentage of computed tasks approaches to 100 %. In particular, when $t_{\text{tot}} = 7$, Fig. 7 shows that all computational tasks are processed on the edge computing network in both online and offline cases, respectively.

Fig. 8 shows the cumulative frequency of the empirical ratio, $D_{IP,OPT}/D_{IP}$, for both the offline optimal and online solutions when $t_{\text{tot}} = 1, 2, 3, 4$. In Fig. 8, the ratio $D_{IP,OPT}/D_{IP}$ is shown to have a step-like shape since both $D_{IP,OPT}$ and D_{IP} are integers, and there exists a limited number of possible values for $D_{IP,OPT}/D_{IP}$ for specific settings of the simulations. In Fig. 8, the cases in which the ratio is one correspond to scenarios in which the proposed algorithm finds the optimal solution. For example, in Fig. 8, about $38 - 88$ % iterations result in the slope of 1 where the optimal solution is achieved by running the proposed algorithm. By the definition of γ in (7), the number of computed tasks with the proposed algorithm is at least $D_{IP,OPT}/\gamma$. For instance, in Fig. 8, the largest empirical competitive ratio is shown to be 2 which implies that the number of computed task is at least $D_{IP,OPT}/2$ when the proposed algorithm is executed with the given simulation parameters.

Fig. 9 shows the percentage of computed tasks for two different ranges of computational speeds of the edge nodes and different task sizes when the bandwidth is changed from 3 to 10 MHz with $t_{\text{tot}} = 7$ and distance randomly distributed in range from 10 m to 70 m. In Fig. 9, neighboring edge nodes with low computational speeds are represented by $f_j \in [5 \times 10^7, 8 \times 10^7]$, whereas edge nodes with high computational speeds are assumed to have $f_j \in [5 \times 10^8, 8 \times 10^8]$. Also, we consider two scenarios with small-size tasks $d_i \in [50{\times}10^6, 70{\times}10^6]$ and large-size tasks $d_i \in [70{\times}10^6, 90{\times}10^6]$,

This article has been accepted for publication in IEEE Transactions on Wireless Communications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TWC.2022.3208096

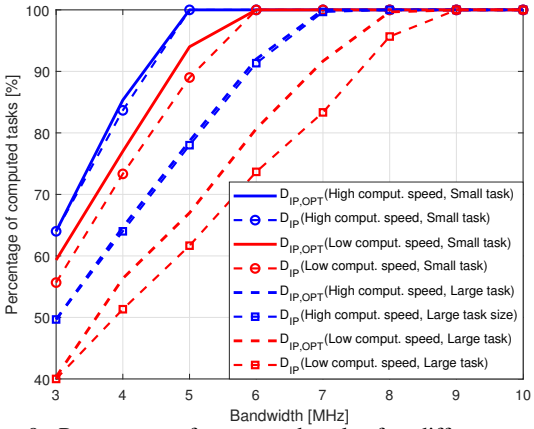

Fig. 9: Percentage of computed tasks for different computational speeds of neighboring edge nodes and different task sizes when bandwidth is varying between 3 and 7 MHz.

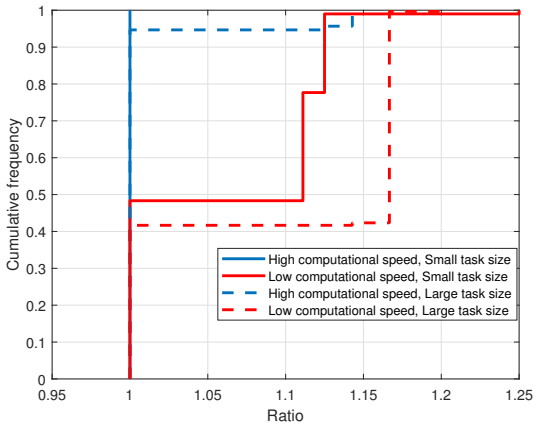

Fig. 10: The empirical competitive ratios $D_{IP,OPT}/D_{IP}$ for different computational speeds of neighboring edge nodes and different task sizes when bandwidth is 5 MHz.

respectively. From Fig. 9, we can see that the number of computed tasks increases with more bandwidth. This is due to the fact that a higher bandwidth can increase the data rate and reduces tasks' transmission latency. Therefore, more tasks can be allocated to neighboring edge nodes. For instance, the number of computed tasks can increase about two-fold if the bandwidth changes from 3 MHz to 10 MHz in the case of edge nodes with low computational speeds and large-size tasks. Also, Fig. 9 shows that using edge nodes with high computational speeds increases the number of computed tasks. For example, the percentage of computed tasks increases from 88 % to 99.5 % by using edge nodes having high computational speeds when bandwidth is 5 MHz and the task sizes are small. Moreover, Fig. 9 shows that more tasks can be computed as task sizes become smaller; for example, small-sized tasks result in 32.8 % more computed tasks compared to that of large-sized tasks in the case of 4 MHz in a high computational speed case.

Fig. 10 shows the empirical competitive ratio $D_{IP,OPT}/D_{IP}$ for different computational speeds of neighboring edge nodes and different task sizes when bandwidth is 5 MHz. We can observe that the proposed algorithm in both cases of edge nodes having high computational speeds almost achieves the optimal performance that can be achieved by the offline optimal solution, i.e., $D_{IP,OPT}$. However, as shown in Fig. 9, since $D_{IP,OPT}$ in case of edge nodes having high computational speeds with large-sized tasks is lower than that in case of edge nodes having low computational speeds with small-sized tasks, the percentage of computed tasks in case of edge nodes having low computational speeds with small-sized tasks is higher than that in case edge nodes having high computational speeds with large-sized tasks. A higher computational capability can be achieved in a larger edge network than in a small one. However, establishing a large network will increase the signaling overhead as the number of participating nodes increases. Hence, between a large and small edge network, there clearly exists a tradeoff between signaling overhead and computing capability.

In Fig. 11, the percentage of computed tasks is shown for different numbers of neighboring edge nodes ranging from 10 to 60. The scenario in Fig. 11 assumes that neighboring edge nodes are randomly distributed within a maximum distance that is varied in range from 30 m to 110 m with $I = 10$, $B = 5$ MHz, and $t_{tot} = 7$. Simulations assume that the small-size tasks are in the range of $d_i \in [40\times10^6, 70\times10^6]$. Also, the neighboring nodes use low computational speeds in the range of $f_j \in [5\times10^7, 8\times10^7]$. In Fig. 11, it is clear that the number of computed tasks increases with the number of neighboring edge nodes. As the set of neighboring edge nodes becomes larger, the source edge node has a higher probability to allocate its tasks to the neighboring edge nodes having a high data rate and computational speed. For instance, the number of computed tasks can increase by about 8.2 % if the number of edge nodes increases from 10 to 60 when the maximum distance is 110 m. Fig. 11 also shows that the number of computed tasks increases if the maximum communication distance between edge nodes is reduced. For example, the percentage of computed tasks increases from 91.8 % to 99.6 % by reducing the maximum distance between neighboring edge nodes and the source edge node.

In Fig. 12, the percentage of computed tasks is shown for different transmit powers from 20 dBm to 25 dBm when the neighboring nodes use identical computing speed that varies from 10^8 to 7.5×10^8. Fig. 12 shows that the number of computed tasks increases with the transmit power of the source edge node. This is due to the fact that the increased data rate reduces the wireless transmission latency, and, therefore, more tasks can be processed within a limited time period. For example, the percentage of computed tasks increases by up to 10.7 % if the transmit power changes from 20 dBm to 25 dBm with $f_j = 10^8$. Also, Fig 12 shows that increasing a computing speed is beneficial to process notably more tasks. For instance, if the computing speed of edge nodes increases from 10^8 to 7.5×10^8, the edge computing network can process up to about 20% more tasks. Thus, Fig 12 shows that reducing the computing latency by using a high computing speed is needed while reducing the transmission latency with a high power.

This article has been accepted for publication in IEEE Transactions on Wireless Communications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TWC.2022.3208096

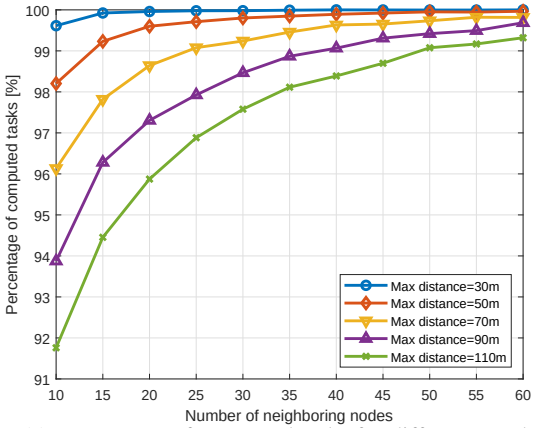

Fig. 11: Percentage of computed tasks for different number of neighboring edge nodes and different maximum communication distances.

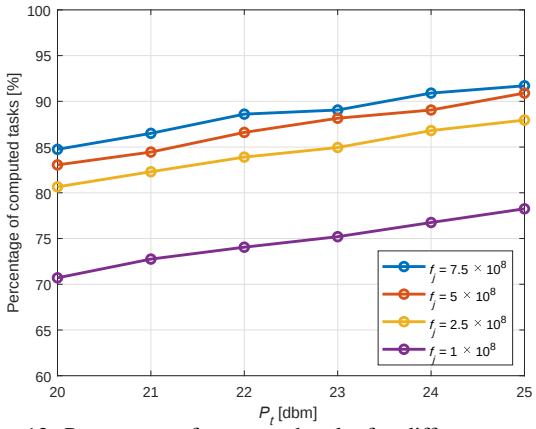

Fig. 12: Percentage of computed tasks for different transmit powers with respect to different computing speeds of neighboring nodes.

V. Conclusion

In this paper, we have proposed a new concept of ephemeral edge computing in which the total time period dedicated to edge computing is limited. This concept of ephemeral edge computing is applicable to a wide range of scenarios including Industry 4.0 smart factory, intelligent transportation systems, and smart homes. By modeling a generalized scenario of ephemeral edge computing, we have proposed a novel framework to maximize the number of successful computations over an edge computing network within a limited time period. This framework allows a source edge node to offload tasks from sensors and distributed tasks to neighboring edge nodes in order to compute the tasks before the source edge node discontinues its current edge computing network. When the exact information on the offloaded tasks is unknown to the source edge node, it is challenging to optimize the decision of which neighboring edge node has to compute each task. Therefore, we have formulated an online optimization problem that jointly optimizes the communication and computation latency is formulated and introduced an online greedy algorithm to solve the problem. Then, by using the structure of the primal-dual problem formulation, we have derived a feasible competitive ratio as a function of the task sizes and the data rates of the edge nodes. Simulation results have shown that the empirical competitive ratio defined as the ratio between the number of computed tasks achieved by the proposed online algorithm and offline optimal case is at most 2 in a given simulation setting. Thus, the simulation results confirm that the proposed online algorithm can efficiently allocate tasks to neighboring edge nodes under uncertainty. Our future work will include extending our results to additional practical scenarios in which multiple tasks can be allocated to edge nodes under the consideration of the lifetime of an ephemeral edge computing network.

Appendix A
Derivation of (12)

In (3), problem (D) is formulated to minimize the sum delay. In this regard, it is beneficial to offload task i from the source node to the neighbor with a high data rate and computing speed to minimize the sum delay. Therefore, the decision rule to select edge node j^* needs to be designed to select an edge node with the shortest communication and computing latency to process the task i, i.e., $\left(\frac{1}{r_j}+\frac{1}{f_j}\right)d_i$. In (8), problem (P) is formulated to minimize the cost objective function $\sum_{i=1}^{I} t_{\text{tot}} x_i + \sum_{j=1}^{J} z_j + u_1$. Then, the decision rule to select edge node j^* need to be designed to select an edge node with the smallest z_j which is equivalent to select an edge node with the largest $\left(1-z_j\right)^{\alpha}$. Thus, the decision rule to select edge node j^* is obtained by (12).

Appendix B
Proof of Lemma 1

We will show that the first constraint is always satisfied for all i when using the updating rule. When allocating task i', $x_i = 0, \forall i \geq i'$ and $u_i = 0, \forall i$ due to the initialization. From the constraints in (9) and (10), we have that $\left(1/r_j + 1/f_j\right) d_i x_i + \left(d_i/r_j\right) \sum_{i'=i+1}^{I} x_{i'} + z_j + u_i - u_{i+1} = \left(1/r_j + 1/f_j\right) d_i (1 - z_j) \frac{1}{(1/r_j+1/f_j)d_i} + z_j = 1$. Then, we consider the constraints regarding other edge nodes $j \in \mathcal{J} \setminus \{j^*\}$ for a given task $\forall i \in \mathcal{I}$. When u_i is updated, $u_i - u_{i+1}$ is equal to Δu_i. Therefore, we can show that edge node $\forall j \in \mathcal{J}$ satisfy the constraint (5) as follows:

$$\begin{aligned}
&\left(\frac{1}{r_j}+\frac{1}{f_j}\right) d_i (1-z_{j^*})^{\alpha} \frac{1}{\left(1/r_{j^*}+1/f_{j^*}\right) d_i} + z_j + \Delta u_i \\
&= \frac{\left(1/r_j+1/f_j\right)}{\left(1/r_{j^*}+1/f_{j^*}\right)} (1-z_{j^*})^{\alpha} + z_j \\
&\quad + \max_{j' \in \mathcal{J}} \left(1 - \left(\frac{\left(1/r_{j'}+1/f_{j'}\right)}{\left(1/r_{j^*}+1/f_{j^*}\right)} (1-z_{j^*})^{\alpha} + z_{j'}\right), 0\right) \quad (21) \\
&\geq \frac{\left(1/r_j+1/f_j\right)}{\left(1/r_{j^*}+1/f_{j^*}\right)} (1-z_{j^*})^{\alpha} + z_j \\
&\quad + \left(1 - \left(\frac{\left(1/r_j+1/f_j\right)}{\left(1/r_{j^*}+1/f_{j^*}\right)} (1-z_{j^*})^{\alpha} + z_j\right)\right) = 1. \quad (22)
\end{aligned}$$

Hence, the primal constraints (9) and (10) are satisfied.

APPENDIX C
PROOF OF LEMMA 2

For a given j, the upper bound of $\sum_{\forall i} y_{ij}$ in (4) is derived by using the fact that the proposed algorithm does not update z_j if $\sum_i y_{ij} \geq 1$. In particular, when the task is indexed by i', suppose that the task allocation is not possible for the first time, i.e., $y_{ij} = 0$, $\forall i > i'$. Before the last task i' arrives, the value of $\sum_{\forall i} y_{ij}$ is still less than the total budget of edge node j. However, after allocating task i' to edge node j, $\sum_{\forall i} y_{ij}$ can be greater then one. The violation of the constraint (4) makes the value of z_j be greater than 1. Therefore, for any $c > 1$, the inequality $z_j \geq \frac{1}{c-1}\left(c^{\sum_{i=1}^{i'} y_{ij}} - 1\right)$ is used to derive the upper bound of $\sum_{\forall i} y_{ij}$. From this relationship, if $\sum_{i=1}^{i'} y_{ij} \geq 1$, we have $\frac{1}{c-1}\left(c^{\sum_{i=1}^{i'} y_{ij}} - 1\right) \geq 1$, then $z_j \geq 1$.

When we define $\beta_{i'j} = \left(\frac{1}{r_j} + \frac{1}{f_j}\right)\frac{d_{i'}}{t_{\text{tot}}}$, the update rule of z_j in (13) is used as following:

$$
\begin{aligned}
z_j &= z_j(1+\beta_{i'j}) + \beta_{i'j}\frac{1}{c-1} && (23)\\
&\geq \frac{1}{c-1}\left(c^{\sum_{i=1}^{i'-1} y_{ij}} - 1\right)(1+\beta_{i'j}) + \beta_{i'j}\frac{1}{c-1} && (24)\\
&= \frac{1}{c-1}\left(c^{\sum_{i=1}^{i'-1} y_{ij}}(1+\beta_{i'j}) - 1\right) && (25)\\
&\overset{(a)}{\geq} \frac{1}{c-1}\left(c^{\sum_{i=1}^{i'-1} y_{ij}+\beta_{i'j}} - 1\right), && (26)
\end{aligned}
$$

where $c \triangleq (1+\delta)^{1/\delta}$ for a constant $\delta \geq \beta_{i'j}$. From the definition of c, (a) holds due to the relationship $1 + \beta_{i'j} \geq \left((1+\delta)^{1/\delta}\right)^{\beta_{i'j}} = \left((1+\delta)^{1/\delta}\right)^{\beta_{i'j}}$ when $0 \leq \beta_{i'j} \leq \delta \leq 1$. Also, the definition of z_j in (13) has an upper bound $z_j \leq \bar{z} \triangleq (1+\delta) + \frac{\delta}{c-1}$, and, therefore, we can rewrite (26) as following: $\sum_{i=1}^{i'-1} y_{ij} \leq \log_c(\bar{z}(c-1)+1) - \beta_{i'j}$. Thus, an upper bound of $\sum_{i=1}^{i'} y_{ij}$ is derived as:

$$
\begin{aligned}
\sum\nolimits_{i=1}^{i'} y_{ij} &\overset{(b)}{\leq} \log_c(\bar{z}(c-1)+1) - \beta_{i'j} + 1\\
&\leq 1 + \log_c \frac{(1+\delta)c}{c^{\beta_{i'j}}} && (27)
\end{aligned}
$$

where (b) hold since $\sum_{i=1}^{i'} y_{ij} = \sum_{i=1}^{i'-1} y_{ij} + 1$ if task i' is allocated. Then, if $\delta = \beta_{i'j} = 0$, we can have a lower bound $1 + \log_c \frac{(1+\delta)c}{c^{\beta_{i'j}}} = 2$.

APPENDIX D
PROOF OF LEMMA 4

By using the definition of z_j and x_i, we derive the change of the objective function of problem (P), denoted by ΔP. When a task i is allocated to an edge node j, z_j and x_i are updated, and, therefore, the objective function of problem (P) increases. In particular, ΔP increase with Δz_j since we want to observe the increment of z_j at current interation while the value of z_j can be updated multiple time. Also, ΔP increase with x_i since x_i is initially given by 0 and updated only once. Thus, we have $\Delta P = \Delta z_j + t_{\text{tot}} x_i + \Delta u_i$ and

$$
\begin{aligned}
\Delta P &= \left(\frac{1}{r_j} + \frac{1}{f_j}\right)\frac{d_i}{t_{\text{tot}}}\left(z_j + \frac{1}{c-1}\right)\\
&\quad + t_{\text{tot}}(1-z_j)^{\alpha}\frac{1}{\left(1/r_j + 1/f_j\right)d_i} + \Delta u_i && (28)\\
&\overset{(c)}{\leq} \frac{t_{\text{tot}}}{\left(1/r_j + 1/f_j\right)d_i}\left(z_j + \frac{1}{c-1}\right)\\
&\quad + (1-z_j)\frac{t_{\text{tot}}}{\left(1/r_j + 1/f_j\right)d_i} + \Delta u_i\\
&= \frac{t_{\text{tot}}}{\left(1/r_j + 1/f_j\right)d_i}\left(1 + \frac{1}{c-1}\right) + \Delta u_i, && (29)
\end{aligned}
$$

where (c) holds due to $\left(1/r_j + 1/f_j\right)\frac{d_i}{t_{\text{tot}}} \leq 1$ with $\alpha = 1$. Next, the objective function of problem (D) is increases by one, and it is denoted by $\Delta D = 1$. This is due to the fact that y_{ij} is initially set to zero, and we update $y_{ij} = 1$ when task i is assigned to edge node j. Hence, we have $\frac{\Delta P}{\Delta D} \leq \frac{t_{\text{tot}}}{\left(1/r_j + 1/f_j\right)d_i}\left(1 + \frac{1}{c-1}\right) + u_i$.

REFERENCES

[1] G. Lee, W. Saad, and M. Bennis, "Online optimization for UAV-assisted distributed fog computing in smart factories of Industry 4.0," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, United Arab Emirates, Dec. 2018, pp. 1–6.

[2] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Network*, vol. 34, no. 3, pp. 134–142, 2020.

[3] M. Karimzadeh, W. Saad, and M. Debbah, "Common language for goal-oriented semantic communications: A curriculum learning framework," in *Proc. IEEE International Conference on Communications (ICC)*, Seoul, South Korea, May 2022.

[4] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[5] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 269–283, 2021.

[6] S. Wang, Y. Zhao, L. Huang, J. Xu, and C.-H. Hsu, "QoS prediction for service recommendations in mobile edge computing," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 134–144, May 2019.

[7] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581–2593, 2020.

[8] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[9] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial offloading scheduling and power allocation for mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6774–6785, 2019.

[10] M. Chen, W. Saad, and C. Yin, "Virtual reality over wireless networks: Quality-of-service model and learning-based resource management," *IEEE Transactions on Communications*, vol. 66, no. 11, pp. 5621–5635, 2018.

[11] Y. Huang, X. Song, F. Ye, Y. Yang, and X. Li, "Fair caching algorithms for peer data sharing in pervasive edge computing environments," in *Proc. IEEE International Conference on Distributed Computing Systems (ICDCS)*, Atlanta, GA, USA, 2017, pp. 605–614.

[12] F. M. F. Wong, C. Joe-Wong, S. Ha, Z. Liu, and M. Chiang, "Improving user QoE for residential broadband: Adaptive traffic management at the network edge," in *Proc. IEEE International Symposium on Quality of Service (IWQoS)*, Portland, OR, USA, 2015, pp. 105–114.

[13] E. Yigitoglu, M. Mohamed, L. Liu, and H. Ludwig, "Foggy: A framework for continuous automated IoT application deployment in fog computing," in *Proc. IEEE International Conference on AI & Mobile Services (AIMS)*, Honolulu, HI, USA, 2017, pp. 38–45.

[14] S.-R. Yang, Y.-J. Tseng, C.-C. Huang, and W.-C. Lin, "Multi-access edge computing enhanced video streaming: Proof-of-concept implementation and prediction/QoE models," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1888–1902, 2019.

[15] 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 Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, 2017.

[16] G. Darling, "IoT vs. edge computing: What's the difference?" IBM, Sep. 9, 2021. [Online]. Available: https://developer.ibm.com/articles/iot-vs-edge-computing/

[17] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Mobile unmanned aerial vehicles (UAVs) for energy-efficient internet of things communications," *IEEE Transactions on Wireless Communications*, vol. 16, no. 11, pp. 7574–7589, 2017.

[18] S. Jeong, O. Simeone, and J. Kang, "Mobile cloud computing with a UAV-mounted cloudlet: Optimal bit allocation for communication and computation," *IET Communications*, vol. 11, no. 7, pp. 969–974, May 2017.

[19] F. Zhou, Y. Wu, H. Sun, and Z. Chu, "UAV-enabled mobile edge computing: Offloading optimization and trajectory design," in *Proc. IEEE International Conference on Communications (ICC)*, Kansas City, MO, USA, 2018, pp. 1–6.

[20] Y. Wang, W. Peng, Q. Dou, and Z. Gong, "Energy-constrained ferry route design for sparse wireless sensor networks," *Journal of Central South University*, vol. 20, no. 11, pp. 3142–3149, Nov. 2013.

[21] X. Hu, K.-K. Wong, K. Yang, and Z. Zheng, "UAV-assisted relaying and edge computing: Scheduling and trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 4738–4752, 2019.

[22] Z. Yang, C. Pan, K. Wang, and M. Shikh-Bahaei, "Energy efficient resource allocation in UAV-enabled mobile edge computing networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 9, pp. 4576–4589, 2019.

[23] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao, and G. Y. Li, "Joint offloading and trajectory design for UAV-enabled mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1879–1892, 2019.

[24] Y. K. Tun, Y. M. Park, N. H. Tran, W. Saad, S. R. Pandey, and C. S. Hong, "Energy-efficient resource management in UAV-assisted mobile edge computing," *IEEE Communications Letters*, vol. 25, no. 1, pp. 249–253, 2021.

[25] L. Yang, H. Yao, J. Wang, C. Jiang, A. Benslimane, and Y. Liu, "Multi-UAV-enabled load-balance mobile-edge computing for IoT networks," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6898–6908, 2020.

[26] J. Zhang, L. Zhou, F. Zhou, B.-C. Seet, H. Zhang, Z. Cai, and J. Wei, "Computation-efficient offloading and trajectory scheduling for multi-UAV assisted mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2114–2125, 2020.

[27] X. Huang, R. Yu, J. Kang, and Y. Zhang, "Distributed reputation management for secure and efficient vehicular edge computing and networks," *IEEE Access*, vol. 5, pp. 25 408–25 420, 2017.

[28] Y. Liu, S. Wang, J. Huang, and F. Yang, "A computation offloading algorithm based on game theory for vehicular edge networks," in *Proc. IEEE International Conference on Communications (ICC)*, Kansas City, MO, USA, May 2018, pp. 1–6.

[29] G. Lee, J. Park, W. Saad, and M. Bennis, "Performance analysis of blockchain systems with wireless mobile miners," *IEEE Networking Letters*, vol. 2, no. 3, pp. 111–115, 2020.

[30] J. Kang, R. Yu, X. Huang, M. Wu, S. Maharjan, S. Xie, and Y. Zhang, "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4660–4670, 2018.

[31] X. Huang, R. Yu, J. Kang, Y. He, and Y. Zhang, "Exploring mobile edge computing for 5G-enabled software defined vehicular networks," *IEEE Wireless Communications*, vol. 24, no. 6, pp. 55–63, Dec. 2017.

[32] 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, no. 11, pp. 1–16, 2015.

[33] "Near-Real-time RAN Intelligent Controller Architecture & E2 General Aspects and Principles," O-RAN Alliance, Alfter, Germany, Technical Specification, Jul. 2022.

[34] T. S. J. Darwish and K. Abu Bakar, "Fog based intelligent transportation big data analytics in the internet of vehicles environment: Motivations, architecture, challenges, and critical issues," *IEEE Access*, vol. 6, pp. 15 679–15 701, 2018.

[35] A. Ferdowsi, U. Challita, and W. Saad, "Deep learning for reliable mobile edge analytics in intelligent transportation systems: An overview," *IEEE Vehicular Technology Magazine*, vol. 14, no. 1, pp. 62–70, 2019.

[36] S. Wang, J. Wan, D. Li, and C. Zhang, "Implementing smart factory of Industrie 4.0: An outlook," *International Journal of Distributed Sensor Networks*, vol. 12, no. 1, p. 3159805, Jan. 2016.

[37] D. Zuehlke, "SmartFactory—towards a factory-of-things," *Annual Reviews in Control*, vol. 34, no. 1, pp. 129–138, Mar. 2010.

[38] A. Majd, G. Sahebi, M. Daneshtalab, J. Plosila, and H. Tenhunen, "Placement of smart mobile access points in wireless sensor networks and cyber-physical systems using fog computing," in *Proc. IEEE International Conference on Scalable Computing and Communications*, Toulouse, France, Jul. 2016, pp. 680–689.

[39] I. Jawhar, N. Mohamed, J. Al-Jaroodi, and S. Zhang, "A framework for using unmanned aerial vehicles for data collection in linear wireless sensor networks," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 1, pp. 437–453, Apr 2014.

[40] M. Schneider, J. Rambach, and D. Stricker, "Augmented reality based on edge computing using the example of remote live support," in *Proc. IEEE International Conference on Industrial Technology (ICIT)*, Toronto, Canada, 2017, pp. 1277–1282.

[41] C. Cooper, D. Franklin, M. Ros, F. Safaei, and M. Abolhasan, "A comparative survey of VANET clustering techniques," *IEEE Communications Surveys Tutorials*, vol. 19, no. 1, pp. 657–681, 2017.

[42] N. Buchbinder and J. S. Naor, "The design of competitive online algorithms via a primal–dual approach," *Foundations and Trends in Theoretical Computer Science*, vol. 3, no. 2-3, pp. 93–263, May 2009.