Joint Batching and Scheduling for High-Throughput Multiuser Edge AI with Asynchronous Task Arrivals

Yihan Cang, *Graduate Student Member, IEEE*, Ming Chen, *Member, IEEE*, and Kaibin Huang, *Fellow, IEEE*

Abstract

Edge *artificial intelligence* (AI) in the sixth-generation networks will provide inference services at the network edge to enrich the capabilities of mobile devices and lengthen their battery lives. As a well-known technique in computing, batching can boost the computation throughput at an edge server by assembling multiple tasks into a batch that is fed into a pre-trained prediction model. This reduces the memory-access frequency and hence accelerates the execution of each task. In a multiuser edge-AI system, the end-to-end latency depends not only on computation but also on communication, i.e., multiuser task uploading over a multi-access channel. In this paper, we study joint batching and (task) scheduling to maximise the throughput (i.e., the number of completed tasks) under the practical assumptions of heterogeneous task arrivals and deadlines. The design aims to optimise the number of batches, their starting time instants, and the task-batch association that determines batch sizes. The joint optimisation problem is complex due to multiple coupled variables as mentioned and numerous constraints including heterogeneous tasks arrivals and deadlines, the causality requirements on multi-task execution, and limited radio resources. Underpinning the problem is a basic tradeoff between the size of batch and waiting time for tasks in the batch to be uploaded and executed. Our approach of solving the formulated mixed-integer problem is to transform it into a convex problem via integer relaxation method and ℓ_0-norm approximation. This results in an efficient alternating optimization algorithm for finding a close-to-optimal solution. Specifically, it iterates between solving two sub-problems, optimal

Y. Cang is with Department of Electrical and Electronic Engineering at The University of Hong Kong, Hong Kong, and also with National Mobile Communications Research Laboratory, Southeast University, Nanjing 211111, China (email: yhcang@hku.hk).

M. Chen is with National Mobile Communications Research Laboratory, Southeast University, Nanjing 211111, China, and also with Purple Mountain Laboratories, Nanjing 211100, China (email: chenming@seu.edu.cn).

K. Huang is with Department of Electrical and Electronic Engineering at The University of Hong Kong, Hong Kong (email: huangkb@eee.hku.hk). Corresponding author: K. Huang.

task-batch association and optimal batch starting time. The former is a linear program whose solution can be found using a derived scheme of greedy task selection while that of the latter is derived in closed form. In addition, we also design the optimal algorithm from leveraging *spectrum holes*, which are caused by fixed bandwidth allocation to devices and their asynchronized multi-batch task execution, to admit unscheduled tasks so as to further enhance throughput. Simulation results demonstrate that the proposed framework of joint batching and resource allocation can substantially enhance the throughput of multiuser edge-AI as opposed to a number of simpler benchmarking schemes, e.g., equal-bandwidth allocation, greedy batching and single-batch execution.

Index Terms

Edge AI, edge inference, batching, scheduling, radio resource allocation.

I. Introduction

Edge *Artificial Intelligence* (AI), a key feature of the sixth-generation (6G) mobile networks, will feature ubiquitous deployment of AI algorithms at the network edge to provide inference services to users [1], [2]. Then Internet-of-Things (IoT) devices can rely on the services to acquire intelligent capabilities ranging from visual perception to natural language processing. Realizing efficient edge AI in practice has to overcome both the communication and computing bottlenecks. The former results from many devices uploading high-dimensional data features to an edge server over a resource constrained multi-access channel. The second refers to the well known *von Neumann bottleneck* where frequent data shuttling between memory and processors (e.g., loading of AI model parameters) can incur as much as 90% of total computation latency and energy [3], [4]. The consideration of end-to-end system performance makes it important to simultaneously overcome the two bottlenecks, which motivates this work. To this end, we design a framework of integrating batching (i.e., task execution in batches to alleviate the von Neumann bottleneck) and device scheduling to enhance the throughput of an edge AI system under the practical assumptions of heterogeneous task arrivals and deadlines.

The area of edge AI, also called edge inference, involves cross-disciplinary research integrating wireless communication and AI to improve the end-to-end system performance [1], [5]. Many relevant algorithms are designed based on a popular architecture called *split inference* that partitions a global deep neural network into an on-device and a server sub-models, which are connected by a wireless channel [5], [6]. Given the architecture, a rich set of techniques have been designed to improve the communication efficiency including pruning the features extracted

using the on-device sub-model [7], [8], jointly training the sub-model and channel encoder [5], progressive transmission [9], and distributed data compression using the information-bottleneck approach [10], [11]. Controlling the model splitting point for split inference introduces another dimension for improving the communication efficiency. In [6], the point is jointly optimized with computation-resource allocation for a multi-core CPU to minimize the end-to-end latency of multiuser tasks. From the perspective of implementation, edge AI algorithms can be deployed on the *mobile edge computing* (MEC) platform, a focus of 5G development, to exploit its strengths in enabling latency-critical applications such as virtual reality (see, e.g., [12]). Furthermore, various practical issues for edge AI deployment have been addressed by researchers such as joint management of communication and computation resources (see, e.g., [13]), heterogeneous devices [14], and random task arrivals (see, e.g., [15]).

In the context of multiuser edge AI, batching is mentioned earlier to be an effective technique for breaking the von Neumann bottleneck so that an edge server can serve more users. Specifically, the advantage of batching lies in reusing the part of AI model loaded into a *graphics processing unit* (GPU) for multiple tasks to avoid frequent memory access [16]. As a result, the computation latency per task is reduced and hence the throughput increases [17]. As mentioned, batching should be jointly designed with radio resource allocation to achieve optimal end-to-end performance for multiuser edge AI. Such designs are crucial for 6G AI empowered tactile applications such as augmented reality (AR) and autonomous driving. In particular, AR requires latency lower than 20 ms in order to guarantee an immersive virtual experience for users. However, at its nascent stage, the mentioned area currently has few results [13], [18]. In [18], utilizing the tree-search method, the optimization problem of joint bandwidth allocation and task scheduling to maximize throughput is solved by proposing an efficient tree-search algorithm with intelligent tree pruning. On the other hand, the minimisation of user energy consumption is studied in [13] under inference latency constraints. To solve the problem, different algorithms are presented for joint task scheduling and transmission-time control, which allow both online and offline implementation. For simplicity, backlogged tasks and single-batch optimization are assumed in prior work. On one hand, as queuing time is not accounted for, the existing designs cannot provide a guarantee on end-to-end latency between a task arrival and its completion where tasks may find difficulty in supporting real-time applications which require immediate execution of randomly arriving multiuser tasks. On the other hand, techniques from single-batch optimization are inefficient when dealing with the cases with a large number of concurrent tasks

with asynchronous arrivals or with a low arrival rate. In both cases, they can potentially result in long waiting time for those tasks that arrive earlier than others. Optimally forming multiple batches can perform better in such cases but its joint design with radio resource allocation remains as an open problem.

It is worth mentioning that the issue of asynchronous task arrivals has been addressed in several studies in the MEC area [19]–[21]. Without targeting a specific task or application, these studies are all based on a generic processor model where computing speeds are measured in, for example, the number of clock cycles required for processing a bit [22]. Furthermore, the processor speed is assumed to be controllable by adjusting its clock frequency that changes its energy consumption following a measurement based model [23]. Based on such models, computation-and-radio resources can be jointly managed to maximize the system energy efficiency or throughput under tasks' deadline requirements [24]. Due to model abstraction, computing issues as elaborated by the von Neumann and batching have not been studied in the MEC literature. Thereby, the existing solutions are inadequate for solving the current problem of *joint batching and scheduling* (JBAS) for multiuser edge AI.

In this work, we make an attempt to solve this problem targeting a high-throughput multiuser edge AI systems under the practical assumptions of asynchronous task arrivals and heterogeneous task deadlines. The problem is challenging for two reasons. First, there are numerous batching related parameters to optimize, namely the number of batches, starting time of individual batches, and the task-batch association. Second, meeting the task deadlines requires the control of end-to-end latency of each task that sums its communication and computation latency. This introduces coupling between batching and scheduling as well as radio resource allocation to scheduled devices. By developing efficient approaches to solve the complex problem, we develop a framework for optimal JBAS.

The main contributions of this work are summarized as follows.

- **Optimal Joint Batching and Scheduling:** The framework of JBAS is designed by solving the JBAS optimization problem. First, we simplify the problem by converting it into an equivalent problem where one variable, the number of batches, is removed. Our technique is to set the number of batches equal to its maximum by allowing empty batches. Second, the equivalent problem, which is a mixed-integer non-linear program, is made tractable by approximation through the methods of integer relaxation and ℓ_0-norm approximation. The resultant convex problem can be efficiently solved using a proposed algorithm that

alternatively solves the following two sub-problems.

  - *Optimization of task-batch association:* The sub-problem is a linear program and its solution can be found using a derived scheme of greedy task selection. The scheme assigns each task to the most suitable batch as measured by a derived metric that accounts for different factors such as batching gain and the task's arrival time and uploading latency.
  - *Optimization of batch starting time:* Given the optimal task-batch association, the optimal starting time of each batch is derived in closed form. It is found to be the latest time a batch can start under the deadline and batch causality constraints so that the tasks in the batch use the least radio resources.

- **Exploitation of Spectrum Holes:** The combined effects of synchronized computation duration of tasks in a same batch, their asynchronous arrivals, and heterogeneous uploading durations create spectrum holes that refer to unused frequency-time resource blocks. We design a spectrum-hole allocation algorithm to optimally exploit spectrum holes to enhance the throughput by admitting originally unscheduled tasks. The corresponding optimization problem is transformed into a sequence of single-batch sub-problems, each attempting to jointly add new tasks to a specific batch and distribute spectrum holes among them. The optimal solution for each sub-problem can be found by a linear search that sequentially tests the sub-problem's feasibility given the number of new scheduled tasks.

Simulations verify that the proposed JBAS algorithm yields significant performance gains as opposed to existing schemes, especially in the scenarios with tight resource constraints. Moreover, the proposed spectrum-hole allocation scheme is shown to yield significant throughput enhancement.

The rest of the paper is organized as follows. The system model is described in Section II. The problem of optimal JBAS is formulated in Section III and solved in Section IV. We present the design of the spectrum-hole allocation algorithm in Section V. In Section VI, the extensions to online design with new arriving tasks and frequency-selective channels are discussed. Simulation results are presented in Section VII followed by concluding remarks in Section VIII.

II. System Model

Consider a single-cell system including K devices and an edge server that doubles as an access point, as shown in Fig. 1. Each device has a single task that offloads a single data sample

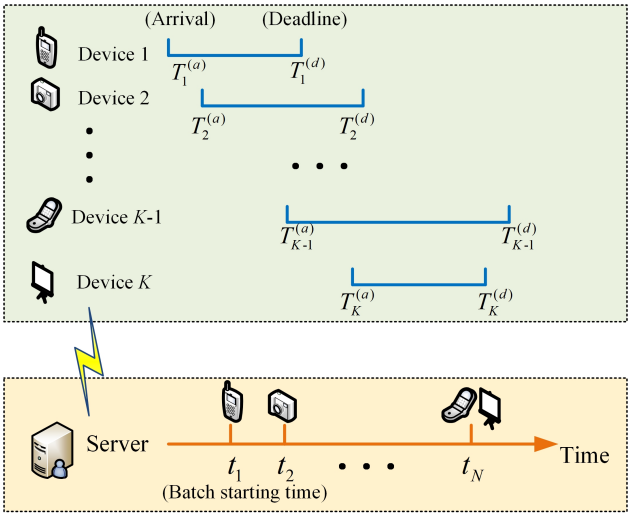

Fig. 1. Edge inference system with asynchronous task arrivals.

(e.g., an image or a video clip) to the server for inference. These tasks are assumed to share a common pre-trained prediction model, such as a large-scale classifier capable of discerning hundreds of object classes [25]. To reduce communication overhead and protect privacy, each scheduled device uploads a feature vector extracted from raw data using a local model. Prior to data uploading, each device communicates to the server over a control channel the profile of its coming task containing the extracted feature size, arrival instant, and deadline requirement. Relevant models and metrics are described in the following sub-sections.

A. Task and Batching Models

Tasks arrive at devices at random time instants with different sizes and distinctive end-to-end delay requirements. An arbitrary device, say device k, has to finish a task within the duration of $\left[T_k^{(a)}, T_k^{(d)}\right]$, where $T_k^{(a)}$ and $T_k^{(d)}$ respectively represent the task-arrival time instant and deadline. The duration consists of three parts: 1) feature uploading phase, 2) task inference phase, and 3) result downloading phase [26]. Due to the relatively small size of inference result (e.g., an object label) and high transmit power of the server, the duration of the result downloading phase is assumed negligible. To enhance the throughput, the server assembles received tasks into a number of batches, denoted as N, which are fed sequentially to the prediction model. Let t_n with $n \in \{1, 2, \cdots, N\}$ denote the time instant when the processing of the n-th batch begins. It follows that $t_1 < t_2 < \cdots < t_N$. To facilitate batching, let $\pi_{k,n}$ represent the association between

task k and the n-th batch. If the task k is included in the n-th batch, $\pi_{k,n} = 1$; otherwise, $\pi_{k,n} = 0$. Since each task should be executed at most once,

$$\sum_{n=1}^{N} \pi_{k,n} \leq 1, \quad \forall k \in \mathcal{K}, \tag{1}$$

where $\mathcal{K}$ denotes the set of devices. The server decision on not sewing a device, say device k, corresponds to $\sum_{n=1}^{N} \pi_{k,n} = 0$. Then batching reduces to determining the association indicators $\{\pi_{k,n}\}$. Upon forming batches, the server sequentially inputs batches of feature vectors into the inference model and downloads results as soon as a batch is executed.

B. Uplink Communication Model

For simplicity, we consider a frequency non-selective channel that emerges as propagation distances keep reducing and the extension to frequency-selective channels is provided in Section VI.B. Its bandwidth B is divided into K sub-channels that are assigned to the scheduled devices. The bandwidth allocated to device k is denoted as B_k. Assume that the channels keep unchanged during the transmission period. The server is assumed to acquire accurate *channel state information* (CSI) useful for resource allocation and device scheduling. The spectrum efficiency of the channel between device k and the server (in bits/second/Hz) is

$$r_k = \log_2 \left(1 + \frac{p_k h_k}{\sigma^2}\right), \quad \forall k \in \mathcal{K}, \tag{2}$$

where p_k represents the transmit power, h_k the channel power gain, and σ^2 the additive white Gaussian noise power. We can write the duration of feature uploading for task/device k as

$$\tau_k^o = \left(\sum_{n=1}^{N} \pi_{k,n} t_n\right) - T_k^{(a)}, \quad \forall k \in \mathcal{K}, \tag{3}$$

where $T_k^{(a)}$ is the task-arrival instant as defined previously. Let ℓ_k represent the number of bits in extracted features for task k. Then $\ell_k = \tau_k^o B_k r_k$. From (3),

$$B_k = \frac{\ell_k}{\left[\left(\sum_{n=1}^{N} \pi_{k,n} t_n\right) - T_k^{(a)}\right] r_k}, \quad \forall k \in \mathcal{K}. \tag{4}$$

C. Inference Model

Consider inference with batching [17]. For the n-th batch, all the uploaded feature vectors satisfying $\pi_{k,n} = 1$ are assembled and input as a batch into the server inference model. The trained

model comprises multiple sequential layers. When processing a batch, the server sequentially loads each layer from the memory and then executes the batch until the batch traverses all layers. As found in the literature, the inference delay increases approximately linearly as the batch size becomes large [17], [27]. Given the association between tasks and batches, $\{\pi_{k,n}\}$, the inference delay of the n-th batch can be modelled as [13], [17], [18]

$$d_n(\pi_n) = a\pi_n + b, \quad \forall n \in \{1, \cdots, N\}, \tag{5}$$

where the batch size $\pi_n = \sum_{k=1}^{K} \pi_{k,n}$ is a positive integer. Note that $d_n(\pi_n)$ is a monotonically increasing function. In the model in (5), a and b depend on the specific inference model [13], [18]. Specifically, a represents the inference delay per task and b the delay of memory access.

III. Problem Formulation

In this section, the design of JBAS is formulated as an optimization problem with the criterion of maximum system throughput, i.e., the number of completed tasks. According to the inference delay model in (5), increasing the batch size can reduce the inference delay per task. However, due to heterogeneous task arrival instants and deadlines, waiting for more tasks to arrive to form a batch hinders the completion of those with early deadlines. On the other hand, to start a batch earlier requires more radio resources so as to finish uploading the associated tasks in time. As a result, there exist two tradeoffs: one between the batch size and batch starting instant and the other between communication and computation resources. Furthermore, the association between tasks and batches also needs to be optimized.

Several practical constraints are considered. The first is the task-causality constraint, namely that the processing of a batch cannot begin until the arrivals of all associated tasks:

$$\pi_{k,n} T_k^{(a)} < t_n, \quad \forall k \in \mathcal{K}, \forall n \in \{1, \cdots, N\}. \tag{6}$$

The second constraint enforces the deadline requirements of scheduled tasks:

$$\pi_{k,n} \left[t_n + d_n(\pi_n)\right] \leq T_k^{(d)}, \quad \forall k \in \mathcal{K}, \forall n \in \{1, \cdots, N\}. \tag{7}$$

Note that when a task, say task k, is not associated with the n-th batch, i.e., $\pi_{k,n} = 0$, constraints (6) and (7) are always satisfied. The third constraint reflects sequential batch processing, namely

that the $(n+1)$-th batch is not processed until the n-th batch finishes its inference:

$$t_n + d_n(\pi_n) \leq t_{n+1}, \quad \forall n \in \{1, \cdots, N-1\}. \tag{8}$$

Last, the bandwidth constraint is given as

$$\sum_{k=1}^{K}\sum_{n=1}^{N} \pi_{k,n}\frac{\ell_k}{r_k \tau_k^o} \leq B. \tag{9}$$

Under the above constraints, we aim at optimizing the bandwidth allocation, the number of batches, their starting instants, as well as the task-batch association. Note that a task that is not assigned to any batch is not scheduled for execution. Then the JBAS optimization problem is formulated as

$$\text{(P1)} \quad \begin{aligned} \max_{\{t_n\},\{\pi_{k,n}\},N} \quad & \sum_{k=1}^{K}\sum_{n=1}^{N} \pi_{k,n}, \\ \text{s.t.} \quad & \pi_{k,n} T_k^{(a)} < t_n, \quad \forall k \in \mathcal{K}, \forall n \in \{1, \cdots, N\}, \\ & \pi_{k,n}\left[t_n + d_n(\pi_n)\right] \leq T_k^{(d)}, \quad \forall k \in \mathcal{K}, \forall n \in \{1, \cdots, N\}, \\ & t_n + d_n(\pi_n) \leq t_{n+1}, \quad \forall n \in \{1, \cdots, N-1\}, \\ & \sum_{k=1}^{K}\sum_{n=1}^{N} \pi_{k,n}\frac{\ell_k}{r_k \tau_k^o} \leq B, \\ & \sum_{n=1}^{N} \pi_{k,n} \leq 1, \quad \forall k \in \mathcal{K}, \\ & \pi_{k,n} \in \{0,1\}, \quad \forall k \in \mathcal{K}, \forall n \in \{1, \cdots, N\}, \\ & N \in \mathbb{Z}^+, N \leq K. \end{aligned}$$

Problem (P1) is non-convex and NP-hard to solve due to the binary task-batch association indicators as well as the coupling between optimization variables [18], [28]. Furthermore, the variable number of batches, N, can change the cardinalities of batch starting instants, $\{t_n\}$, as well as the association indicators, $\{\pi_{k,n}\}$, further complicating this problem.

IV. Optimal JBAS Algorithm

In this section, we design an efficient algorithm for JBAS by approximately solving Problem (P1). The proposed solution approach is to transform the problem to an equivalent, simpler one with the number of bathes fixed. Then applying the method of integer relaxation allows the

equivalent problem to be solved using an alternating optimization algorithm. Its complexity is analyzed.

A. A Tractable Solution Approach

1) An Equivalent Problem: First, Problem (P1) can be transformed into the following equivalent problem:

$$
\begin{aligned}
\max_{\{t_n\},\{\pi_{k,n}\},N} \quad & \sum_{k=1}^{K}\sum_{n=1}^{N}\pi_{k,n}, \\
\text{s.t.} \quad & \pi_{k,n}T_k^{(a)} < t_n, \quad \forall k \in \mathcal{K}, \forall n \in \{1,\cdots,N\}, \\
& t_n + d_n\left(\pi_n\right) \le T_k^{(d)} + \left(1-\pi_{k,n}\right)\Xi, \quad \forall k \in \mathcal{K}, \forall n \in \{1,\cdots,N\}, \\
& t_n + d_n\left(\pi_n\right) \le t_{n+1}, \quad \forall n \in \{1,\cdots,N-1\}, \\
& \sum_{k=1}^{K}\sum_{n=1}^{N}\pi_{k,n}\frac{\ell_k}{r_k\left(t_n - T_k^{(a)}\right)} \le B, \\
& \sum_{n=1}^{N}\pi_{k,n} \le 1, \quad \forall k \in \mathcal{K}, \\
& \pi_{k,n} \in \{0,1\}, \quad \forall k \in \mathcal{K}, \forall n \in \{1,\cdots,N\}, \\
& N \in \mathbb{Z}^+, N \le K.
\end{aligned}
\tag{P2}
$$

where the constant $\Xi \triangleq \max_{k\in\mathcal{K}} T_k^{(d)} + d_N\left(K\right)$. Problem (P2) is different from the conventional *mixed integer nonlinear programming* (MINLP) problem since the number of variables varies with the number of batches, N. Without loss of generality, we propose to mend the difference by fixing N as $N = K$ by allowing the existence of empty batches. This results in the following MINLP problem:

$$
\begin{aligned}
\max_{\{t_n\},\{\pi_{k,n}\}} \quad & \sum_{k=1}^{K}\sum_{n=1}^{N}\pi_{k,n}, \\
\text{s.t.} \quad & \pi_{k,n}T_k^{(a)} < t_n, \quad \forall k \in \mathcal{K}, \forall n \in \{1,\cdots,N\}, \\
& t_n + d_n\left(\pi_n\right) \le T_k^{(d)} + \left(1-\pi_{k,n}\right)\Xi, \quad \forall k \in \mathcal{K}, \forall n \in \{1,\cdots,N\}, \\
& t_n + d_n\left(\pi_n\right) \le t_{n+1}, \quad \forall n \in \{1,\cdots,N-1\}, \\
& \sum_{k=1}^{K}\sum_{n=1}^{N}\pi_{k,n}\frac{\ell_k}{r_k\left(t_n - T_k^{(a)}\right)} \le B,
\end{aligned}
\tag{P3}
$$

$$\sum_{n=1}^{N} \pi_{k,n} \leq 1, \quad \forall k \in \mathcal{K},$$

$$\pi_{k,n} \in \{0,1\}, \quad \forall k \in \mathcal{K}, \forall n \in \{1, \cdots, N\},$$

where the corresponding inference delay evolves as:

$$d_n(\pi_n) = \begin{cases} a\pi_n + b, & \text{if } \pi_n > 0, \\ 0, & \text{if } \pi_n = 0, \end{cases} \tag{10}$$

for all n. The following theorem gives the equivalence between Problems (P2) and (P3).

Theorem 1. *Problems (P3) and (P2) are equivalent in the sense that their optimal objectives are identical.*

The proof is provided in Appendix A. □

Theorem 1 allows us to solve Problem (P2) by solving Problem (P3) that leverages MINLP.

2) Integer Relaxation and Alternating Optimization: To solve (P3), the method of integer relaxation is adopted to obtain an approximate solution (see e.g., [29]). Specifically, the binary variables $\{\pi_{k,n}\}$ are relaxed as continuous ones belonging to $[0,1]$. It should be emphasized that the relaxation does not compromise the optimaliy as discussed in Remark 2. Due to the existence of empty batches, the inference delay function $d_n(\pi_n)$ has a step at $\pi_n = 0$, making Problem (P3) nonconvex. To address the issue, it can be rewritten in a form comprising ℓ_0-norm as

$$d_n(\pi_n) = a\pi_n + b\mathbb{1}_{\{\pi_n\}} = a\pi_n + b\|\pi_n\|_0, \tag{11}$$

where $\|\cdot\|_0$ is ℓ_0-norm and $\mathbb{1}_{\{x\}}$ is the indicator function that is 1 if $x > 0$ and 0 otherwise. The non-smooth ℓ_0-norm can be well approximated by a series of convex weighted ℓ_1-norms, which is a commonly used technique in compressive sensing (see e.g., [30], [31]). Using this technique, the ℓ_0-norm term in (11) can be approximated by an asymptotically equivalent term as

$$\left\| \sum_{k=1}^{K} \pi_{k,n} \right\|_0 = \lim_{\delta \to 0} \frac{\ln\left(1 + \delta^{-1} \sum_{k=1}^{K} \pi_{k,n}\right)}{\ln\left(1 + \delta^{-1}\right)}. \tag{12}$$

Since the logarithmic function is concave and upper bounded by the first-order term of Taylor's expansion, we have

$$\left\|\sum_{k=1}^{K}\pi_{k,n}\right\|_0 \leq \theta_n^{(r)}\sum_{k=1}^{K}\pi_{k,n}+\psi_n^{(r)}, \tag{13}$$

with

$$\theta_n^{(r)}=\frac{\delta^{-1}\left(1+\delta^{-1}\sum_{k=1}^{K}\pi_{k,n}^{(r)}\right)^{-1}}{\ln\left(1+\delta^{-1}\right)}, \tag{14}$$

and

$$\psi_n^{(r)}=\frac{\ln\left(1+\delta^{-1}\sum_{k=1}^{K}\pi_{k,n}^{(r)}\right)+\left(1+\delta^{-1}\sum_{k=1}^{K}\pi_{k,n}^{(r)}\right)^{-1}-1}{\ln\left(1+\delta^{-1}\right)}, \tag{15}$$

where $\pi_{k,n}^{(r)}$ represents the value of $\pi_{k,n}$ at the previous iteration and δ is a sufficiently small constant. The equality in (13) holds if and only if $\pi_{k,n}=\pi_{k,n}^{(r)}$ for all (n,k). Through the above iterative updates of $\theta_n^{(r)}$ and $\psi_n^{(r)}$, the difference between $\|\pi_n\|_0$ and its first-order term of Taylor's expansion diminishes until the equality in (13) holds. Then substituting (13) into (11), the inference delay function can be approximated as

$$d_n\left(\pi_n\right)\approx\left(a+b\theta_n^{(r)}\right)\sum_{k=1}^{K}\pi_{k,n}+b\psi_n^{(r)},\quad \forall n, \tag{16}$$

which is continuous and linear. Using (16), Problem (P3) can be approximated as

$$\begin{aligned}
\max_{\{t_n\},\{\pi_{k,n}\}}\quad & \sum_{k=1}^{K}\sum_{n=1}^{N}\pi_{k,n},\\
\text{s.t.}\quad & \pi_{k,n}T_k^{(a)}<t_n,\quad \forall k\in\mathcal{K},\forall n\in\{1,\cdots,N\},\\
& t_n+\left(a+b\theta_n^{(r)}\right)\sum_{k=1}^{K}\pi_{k,n}+b\psi_n^{(r)}\leq T_k^{(d)}+\left(1-\pi_{k,n}\right)\Xi,\forall k\in\mathcal{K},\forall n\in\{1,\cdots,N\},\\
& t_n+\left(a+b\theta_n^{(r)}\right)\sum_{k=1}^{K}\pi_{k,n}+b\psi_n^{(r)}\leq t_{n+1},\quad \forall n\in\{1,\cdots,N-1\},\\
& \sum_{k=1}^{K}\sum_{n=1}^{N}\pi_{k,n}\frac{\ell_k}{r_k\left(t_n-T_k^{(a)}\right)}\leq B,\\
& \sum_{n=1}^{N}\pi_{k,n}\leq 1,\quad \forall k\in\mathcal{K},\\
& 0\leq\pi_{k,n}\leq 1,\quad \forall k\in\mathcal{K},\forall n\in\{1,\cdots,N\}.
\end{aligned} \tag{P4}$$

This problem is convex and can be readily solved utilizing the approach of alternating optimization. We propose to alternate solving two reduced-dimension sub-problems as described in the following sub-sections. As a result, the complexity is dramatically reduced as opposed to directly solving Problem (P4) and furthermore useful insight can be obtained. It is worth mentioning that alternating optimization provides no guarantee on reaching the global optimal point since the constraints in Problem (P4) are not *box constraints* (see, e.g., [32]). The complete algorithm is presented in Algorithm 1.

B. Optimal Task-Batch Association

The first sub-problem results from fixing the starting time of batches, $\{t_n\}$, in Problem (P4). Then it reduces to a linear program. The dual problem of (P4) with respect to task-batch association, $\{\pi_{k,n}\}$, is given as

$$\min_{\{\beta_{k,n}\},\{\gamma_{k,n}\},\rho} G\left(\beta_{k,n},\gamma_{k,n},\rho\right), \tag{17}$$

where $G\left(\beta_{k,n},\gamma_{k,n},\rho\right)$ is the dual function that solves

$$\begin{aligned} \max_{\{\pi_{k,n}\}} \quad & \mathcal{L}\left(\pi_{k,n},\beta_{k,n},\gamma_{k,n},\rho\right), \\ \text{s.t.} \quad & \sum_{n=1}^{N}\pi_{k,n}\leq 1, \quad \forall k\in\mathcal{K}, \\ & 0\leq\pi_{k,n}\leq 1, \quad \forall k\in\mathcal{K},\forall n\in\{1,\cdots,N\}. \end{aligned} \tag{18}$$

In (18), $\mathcal{L}\left(\pi_{k,n},\beta_{k,n},\gamma_{k,n},\rho\right)$ denotes the partial Lagrangian function of Problem (P4):

$$\begin{aligned} &\mathcal{L}\left(\pi_{k,n},\alpha_{k,n},\beta_{k,n},\gamma_{k,n},\rho\right)=\sum_{k=1}^{K}\sum_{n=1}^{N}\pi_{k,n} \\ &-\sum_{k=1}^{K}\sum_{n=1}^{N}\beta_{k,n}\left[t_n+\left(a+b\theta_n^{(r)}\right)\left(\sum_{k=1}^{K}\pi_{k,n}\right)+b\psi_n^{(r)}-T_k^{(d)}-\left(1-\pi_{k,n}\right)\Xi\right] \\ &-\sum_{n=1}^{N-1}\gamma_n\left[t_n+\left(a+b\theta_n^{(r)}\right)\left(\sum_{k=1}^{K}\pi_{k,n}\right)+b\psi_n^{(r)}-t_{n+1}\right]-\rho\left(\sum_{k=1}^{K}\sum_{n=1}^{N}\pi_{k,n}\frac{\ell_k}{r_k\left(t_n-T_k^{(a)}\right)}-B\right) \end{aligned} \tag{19}$$

where $\beta_{k,n}$, $\gamma_{k,n}$, and ρ are non-negative Lagrange multipliers associated with the deadline, batch causality, and bandwidth allocation constraints, respectively. Besides, we let $\gamma_0=\gamma_N=0$ for

consistency. We can observe that (19) is linear with respect to $\pi_{k,n}$. Therefore, to maximize the Lagrange function with fixed multipliers, the optimal $\pi_{k,n}$ is either zero or one. Specifically, for all n, if $\pi_{k,n}$ are less than or equal to zero, task k is not scheduled, i.e., $\pi_{k,n} = 0$; otherwise, this task is associated with the batch that has the largest coefficient:

$$\pi_{k,n}^{*} = \begin{cases} 1, & \text{if } n = \arg \max\limits_{n \in \{1,\cdots,N\}} \mu_{k,n}, \\ 0, & \text{otherwise}, \end{cases} \tag{20}$$

where $\mu_{k,n} = 1 - \left(a + b\theta_n^{(r)}\right) \sum_{k=1}^{K} \beta_{k,n} - \left(a + b\theta_n^{(r)}\right) \gamma_n - \Xi\beta_{k,n} - \rho \frac{\ell_k}{r_k\left(t_n - T_k^{(a)}\right)}$. If there are multiple batches satisfying $\arg\max_{n \in \{1,\cdots,N\}} \mu_{k,n}$, we can choose any of them due to the non-strict convexity of Problem (P3). Then substituting (20) into (18), we can obtain $G\left(\beta_{k,n}, \gamma_{k,n}, \rho\right)$.

Remark 1. (Favourable Task Conditions) According to (20), one can infer that for a task, as the channel condition becomes worse, its likelihood of being scheduled reduces as uploading the task requires more radio resources or else incurs higher latency. Moreover, early task-arrival time increases the probability that a task is successfully executed due to the following two reasons: 1) the larger batching gain, and 2) the longer communication time that increases the probability of successful feature uploading. Last, by combining (14) and (20), we can observe that a task prefers a larger batch as its inference delay per task is smaller due to the batching gain.

Given $G\left(\beta_{k,n}, \gamma_{k,n}, \rho\right)$, we attempt to solve the dual problem (17) to get the the optimal dual variables $\{\beta_{k,n}\}, \{\gamma_{k,n}\}, \rho$. Note that $G\left(\beta_{k,n}, \gamma_{k,n}, \rho\right)$ is not differentiable in general due to the discontinuous selection operations in obtaining the optimal $\pi_{k,n}^{*}$. To this end, the value of dual variables is updated by the sub-gradient method [33]. Thus, through iteratively optimizing primal variables and dual variables, the optimal tasks and batches association $\pi_{k,n}$ with fixed t_n can be obtained directly without rounding according to the following remark.

Remark 2. (Optimality of Task-Batch Association) We can observe that for task k, there exists at most a single element among $\{\pi_{k,n}\}$ that is equal to one while others are set as zero according to (20). This indicates that although the feasible range of $\pi_{k,n}$ is relaxed to be continuous, the optimal solution to Problem (P4) with respect to $\pi_{k,n}$ always satisfies the binary constraint $\pi_{k,n} \in \{0, 1\}$ for all (n, k). Hence, the relaxation of $\pi_{k,n}$ does not compromise the optimality of the original Problem (P3).

C. Optimal Batch Starting Time

The other sub-problem results from fixing the task-batch association, $\{\pi_{k,n}\}$, in Problem (P4). As a result, the sub-problem is written as

$$\text{(P5)} \quad \begin{aligned} \max_{\{t_n\}} \quad & \sum_{k=1}^{K}\sum_{n=1}^{N} \pi_{k,n}, \\ \text{s.t.} \quad & \pi_{k,n} T_k^{(a)} < t_n, \quad \forall k \in \mathcal{K}, \forall n \in \{1, \cdots, N\}, \\ & t_n + d_n(\pi_n) \le T_k^{(d)} + (1 - \pi_{k,n})\,\Xi, \quad \forall k \in \mathcal{K}, \forall n \in \{1, \cdots, N\}, \\ & t_n + d_n(\pi_n) \le t_{n+1}, \quad \forall n \in \{1, \cdots, N-1\}, \\ & \sum_{k=1}^{K}\sum_{n=1}^{N} \pi_{k,n} \frac{\ell_k}{r_k\left(t_n - T_k^{(a)}\right)} \le B. \end{aligned}$$

Theorem 2. *The optimal starting time of the n-th batch, which solves Problem (P5), is given as*

$$t_n^* = \begin{cases} \min\left\{\chi_n^{(d)}, t_{n+1}\right\} - d_n(\pi_n), & \textit{if } \pi_n > 0, \\ t_{n+1}^*, & \textit{otherwise}, \end{cases} \quad (n = N, \cdots, 1), \tag{21}$$

where $\chi_n^{(d)} = \min_{k \in \mathcal{K}_n} T_k^{(d)}$ denotes the minimum deadline among all the tasks processed in the n-th batch, i.e., $\mathcal{K}_n = \{k | \pi_{k,n} = 1\}$ $(\forall n \in \{1, \cdots, N\})$, and $t_{N+1}^* = \Xi$.

The proof is provided in Appendix B. ■

From Theorem 2, we can observe that with fixed $\pi_{k,n}$, t_n is only determined by the deadlines of tasks in the n-th batch and t_{n+1}. The starting time of a batch, t_n, is set as the latest starting time that can ensure the latency and batch causality constraint such that the scheduled tasks occupy the least radio resources.

D. Complexity Analysis

The complexity of Algorithm 1 is largely attributed to solving the two subproblems solved in the preceding subsections. The complexity in optimizing the task-batch association is $\mathcal{O}\left(K^2/\sqrt{\epsilon}\right)$ based on (20), where ϵ represents the predefined accuracy of the dual method [34]. The complexity of calculating the optimal batch starting instants is $\mathcal{O}(K)$ according to (21). The overall complexity is given by $\mathcal{O}\left(L\left(K^2/\sqrt{\epsilon} + K\right)\right)$, where L denotes the average number of iterations in Algorithm 1.

Algorithm 1: JBAS Algorithm

Initialize $t_n = \frac{\max_{k\in\mathcal{K}} T_k^{(d)} - \min_{k\in\mathcal{K}} T_k^{(a)}}{N-1} \times (n-1) + \min_{k\in\mathcal{K}} T_k^{(a)}$ $(\forall n \in \{1,\cdots,N\})$, $t_{N+1} = \Xi$, $\pi_{k,n}^{(r)} = 0$ $(\forall k \in \mathcal{K}, \forall n \in \{1,\cdots,N\})$ and required precision.
repeat
  Initialize $\{\beta_{k,n}\}$, $\{\gamma_n\}$, ρ.
  repeat
    Obtain the association between tasks and batches $\{\pi_{k,n}\}$ according to (20).
    Update dual vairables $\{\beta_{k,n}\}$, $\{\gamma_n\}$, ρ using the sub-gradient method.
  until the objective of problem (9) converges;
  for $n = N,\cdots,1$ **do**
    Obtain the startup time of batches $\{t_n\}$ according to Theorem 2.
  end
  Update $\theta_n^{(r)}$ and $\psi_n^{(r)}$ according to (14) and (15), respectively.
until the objective of Problem (P3) converges;
Output the optimal $\{\pi_{k,n}\}$ and $\{t_n\}$.

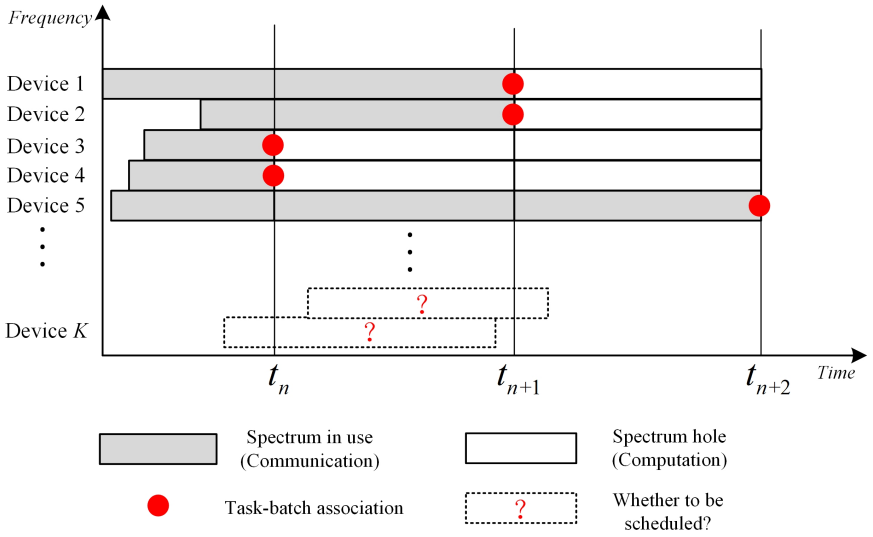

Fig. 2. Illustration of spectrum-holes.

Last, the complexity of Algorithm 1 is much lower than the conventional interior point method for directly solving Problem (P4) whose complexity is $\mathcal{O}\left(L\left(K^2+K\right)^{3.5}\right)$ [34].

V. Exploiting Spectrum Holes

In Section II, individual devices are assigned dedicated frequency bands for identical uploading durations to allow tractable design. Nevertheless, due to heterogeneous task arrivals and communication latency, there exist *spectrum holes* that can be exploited to further increase the throughput. As defined, a spectrum hole refers to an unused frequency-time resource block. In this section, an algorithm for spectrum hole allocation algorithm is designed by formulating and solving a corresponding throughput maximization problem.

A. Spectrum-Hole Allocation Problem

As shown in Fig. 2, upon arriving time instant t_n for all n, the tasks inferred in the n-th batch finish their transmission, and thus spectrum holes (i.e., the bandwidth left by scheduled tasks) can be allocated to the unscheduled tasks to improve the throughput or the scheduled tasks to reduce the latency. It should be noted that fixing $\{t_n\}$ as computed using Algorithm 1, makes it difficult to insert additional tasks for inference due to the tight deadlines according to (21). This means that even those unscheduled tasks can be uploaded to the server exploiting spectrum holes, they cannot be executed in the original batches without interrupting originally tasks. The challenge faced in the current problem lies in adjusting $\{t_n\}$ to accommodate new tasks without causing the failure of any existing task to meet its deadline. Denote $\mathcal{F}$ as the set of unscheduled tasks based on Algorithm 1. For each t_n, let $\mathcal{S}_n$ denote the set of tasks associated with the n-th batch. Hence, at each arrival time, say t_n, the total bandwidth of spectrum holes, denoted by $\bar{B}_n = \sum_{i=1}^{n} \sum_{k\in\mathcal{S}_i} B_k$, can be allocated to unscheduled tasks in $\mathcal{F}$ so as to improve the throughput. Moreover, we have to adjust starting time of the $(n+1)$-th batch such that new scheduled tasks can be inserted into current batch without causing any original scheduled tasks to miss its deadline. Mathematically, at each checkpoint t_n with $n \in \{1, \cdots, N-1\}$, we let $\tilde{\ell}_k$ represent the data size to be transmitted for task k in the duration from t_n to t_{n+1}. Specifically, for tasks in $\mathcal{S}_{n+1}$, $\tilde{\ell}_k$ is given by $B_k r_k \left(\bar{t}_{n+1} - \max\left\{T_k^{(a)}, \bar{t}_n\right\}\right)$, and for tasks in $\mathcal{F}$, $\tilde{\ell}_k$ is equivalent to ℓ_k. Optimization variables $\bar{t}_{n+1}$ and $\bar{\mathcal{K}}_{n+1}$ represent the adjusted startup instant of the $(n+1)$-th batch and new-scheduled tasks in the $(n+1)$-th batch, respectively. Besides, we let $\bar{t}_1 = t_1$. Then, we solve the following optimization problem:

$$
\text{(P6)} \quad
\begin{aligned}
\max_{\bar{t}_{n+1}, \bar{\mathcal{K}}_{n+1}} \quad & |\bar{\mathcal{K}}_{n+1}|, \\
\text{s.t.} \quad & \max\left\{\max_{k\in\bar{\mathcal{K}}_{n+1}\cup\mathcal{S}_{n+1}} T_k^{(a)}, \bar{t}_n + d(|\mathcal{S}_n|)\right\} < \bar{t}_{n+1}, \\
& \bar{t}_{n+1} + d\left(|\bar{\mathcal{K}}_{n+1}\cup\mathcal{S}_{n+1}|\right) \le \min\left\{\min_{k\in\bar{\mathcal{K}}_{n+1}\cup\mathcal{S}_{n+1}} T_k^{(d)}, t_{n+2}\right\}, \\
& \sum_{k\in\bar{\mathcal{K}}_{n+1}\cup\mathcal{S}_{n+1}} \frac{\tilde{\ell}_k}{r_k\left(\bar{t}_{n+1} - \max\left\{T_k^{(a)}, \bar{t}_n\right\}\right)} \le \sum_{i=1}^{n+1}\sum_{k\in\mathcal{S}_i} B_k, \\
& \bar{\mathcal{K}}_{n+1} \subseteq \mathcal{F}.
\end{aligned}
$$

In (P6), the first and second constraints specify the causality of the new-scheduled tasks $\bar{\mathcal{K}}_{n+1}$ and previously determined to be scheduled tasks $\mathcal{S}_{n+1}$ for the $(n+1)$-th batch. The third constraint

implies that the allocated bandwidth to tasks in $\bar{\mathcal{K}}_{n+1}$ and $\mathcal{S}_{n+1}$ should not exceed the sum of spare bandwidth and previously determined bandwidth for tasks in $\mathcal{S}_{n+1}$, which ensures that bandwidth reallocation does not influence the bandwidth allocated to other tasks being transmitted. To this end, the spectrum-hole allocation problem is formulated as a sequence of subproblems, each corresponding to a one-batch optimization problem of tasks scheduling and bandwidth allocation for the $(n+1)$-th batch $(\forall n \in \{1, \cdots, N-1\})$ at checkpoint t_n. By solving Problem (P6), we attempt to increase the number of scheduled tasks and decrease the total delay simultaneously.

B. Solution Approach

Problem (P6) is difficult to solve due to the combinatorial nature of this problem. In the following, we show that the optimal solution of (P6) can be obtained by solving a series of feasibility problems each corresponding to a fixed number of new-scheduled tasks $|\bar{\mathcal{K}}_{n+1}| = \Pi \in \{0, 1, \cdots, |\mathcal{F}|\}$:

$$
\text{(P7)} \quad \begin{aligned}
&\text{find} \quad \bar{t}_{n+1}, \bar{\mathcal{K}}_{n+1}, \\
&\text{s.t.} \quad \max\left\{\max_{k\in\bar{\mathcal{K}}_{n+1}\cup\mathcal{S}_{n+1}} T_k^{(a)}, \bar{t}_n + d(|\mathcal{S}_n|)\right\} < \bar{t}_{n+1}, \\
&\qquad \bar{t}_{n+1} + d\left(\Pi + |\mathcal{S}_{n+1}|\right) \le \min\left\{\min_{k\in\bar{\mathcal{K}}_{n+1}\cup\mathcal{S}_{n+1}} T_k^{(d)}, t_{n+2}\right\}, \\
&\qquad \sum_{k\in\bar{\mathcal{K}}_{n+1}\cup\mathcal{S}_{n+1}} \frac{\tilde{\ell}_k}{r_k\left(\bar{t}_{n+1} - \max\left\{T_k^{(a)}, \bar{t}_n\right\}\right)} \le \sum_{i=1}^{n+1}\sum_{k\in\mathcal{S}_i} B_k, \\
&\qquad \bar{\mathcal{K}}_{n+1} \subseteq \mathcal{F}, |\bar{\mathcal{K}}_{n+1}| = \Pi.
\end{aligned}
$$

Proposition 1. *Denote the optimal solution of Problem (P6) by $\bar{\mathcal{K}}_{n+1}^*$, $\bar{t}_{n+1}^*$ and let $|\bar{\mathcal{K}}_{n+1}^*| = \Pi^*$. Problem (P7) is feasible if and only if $\Pi < \Pi^*$.*

The proof is provided in Appendix C. □

Proposition 1 implies that we can adopt the bisection method to solve Problem (P6) as described below. Denote the upper bound and lower bound of Π by Π_{up} and Π_{low}, respectively. Let $\Pi = \lfloor(\Pi_{\text{up}} + \Pi_{\text{low}})/2\rfloor$, where $\lfloor\cdot\rfloor$ denotes the round down operation, and solve Problem (P7). If (P7) is feasible, which means that the optimal Π^* is no smaller than Π, we set $\Pi_{\text{low}} = \Pi$. Otherwise, the optimal Π^* is no larger than Π, thus setting $\Pi_{\text{up}} = \Pi$. Repeating these procedures until $\Pi_{\text{up}} - \Pi_{\text{low}} \le 1$. Then if Problem (P7) is feasible when $\Pi = \Pi_{\text{up}}$, the optimal $\Pi^* = \Pi_{\text{up}}$; otherwise, we have $\Pi^* = \Pi_{\text{low}}$.

However, it remains to solve Problem (P7) with given Π. Note that with given $\min_{k\in\bar{\mathcal{K}}_{n+1}} T_k^{(d)}$, the optimal $(n+1)$-th batch starting time $\bar{t}_{n+1}^*$ should be given as $\min\left\{\min_{k\in\bar{\mathcal{K}}_{n+1}\cup\mathcal{S}_{n+1}} T_k^{(d)}, t_{n+2}\right\} - d\left(\Pi + |\mathcal{S}_{n+1}|\right)$ since the required bandwidth for tasks decreases with $\bar{t}_{n+1}$. To this end, a tentative policy is proposed to solve Problem (P7). The principles behind this policy is that $\min_{k\in\bar{\mathcal{K}}_{n+1}} T_k^{(d)}$ only takes values from a finite discrete set $\left\{T_k^{(d)}\middle| k\in\mathcal{F}\right\}$ such that we can judge the feasibility of each value of $\min_{k\in\bar{\mathcal{K}}_{n+1}} T_k^{(d)}$ sequentially. Specifically, with given Π, define a set $\mathcal{G}$ containing all the values of unscheduled tasks' deadlines, i.e., $\mathcal{G} = \left\{T_k^{(d)}\middle| k\in\mathcal{F}\right\}$. At each time, we check whether the unscheduled task with the earliest deadline can be added in $\bar{\mathcal{K}}_{n+1}$. Set $T^{(d)}$ as the latest completion instant for the $(n+1)$-th batch, that is $T^{(d)} = \min\left\{\min\left\{i|i\in\mathcal{G}\right\}, \min_{k\in\mathcal{S}_{n+1}} T_k^{(d)}, t_{n+2}\right\}$. Thus, the optimal batch starting instant is given as $\bar{t}_{n+1}^* = T^{(d)} - d\left(\Pi + |\mathcal{S}_{n+1}|\right)$. Let $\tilde{\mathcal{S}} = \left\{k\middle| T_k^{(a)} < \bar{t}_{n+1}^*, T_k^{(d)} \geq T^{(d)}, k\in\mathcal{F}\right\}$ denote the set of tasks that not only satisfy the uploading causality constraint but also their deadlines is no earlier than $T^{(d)}$. This indicates that the tasks in $\tilde{\mathcal{S}}$ can meet their deadlines requirements even with an earlier deadline. Next, we judge whether there exist Π new-scheduled tasks in $\tilde{\mathcal{S}}$ satisfying bandwidth constraint. Sort the tasks belonging in $\tilde{\mathcal{S}}$ in ascending order according to the value of minimum bandwidth required, i.e., $\frac{\tilde{\ell}_k}{r_k\left(\bar{t}_{n+1} - \max\left\{T_k^{(a)}, \bar{t}_n\right\}\right)}$ and assemble the first Π tasks in set $\bar{\mathcal{K}}_{n+1}$. If the total bandwidth of tasks in $\bar{\mathcal{K}}_{n+1}\cup\mathcal{S}_{n+1}$ is no larger than $\sum_{i=1}^{n+1}\sum_{k\in\mathcal{S}_i} B_k$ and $\max\left\{\max_{k\in\bar{\mathcal{K}}_{n+1}\cup\mathcal{S}_{n+1}} T_k^{(a)}, \bar{t}_n + d(|\mathcal{S}_n|)\right\}$ is smaller than $\bar{t}_{n+1}$, i.e., bandwidth and task-causality constraints in Problem (P7) are satisfied, Problem (P7) under current $T^{(d)}$ is feasible, thus making Problem (P7) with current Π feasible. Hence, we set $\Pi_{\text{low}} = \Pi$. Otherwise, it is infeasible with current $T^{(d)}$ indicating that task $k = \arg\min\left\{i\in\mathcal{G}\right\}$ cannot be scheduled in current batch. In this case, we should delete the minimum value in $\mathcal{G}$. Then update $T^{(d)} = \min\left\{\min\left\{i|i\in\mathcal{G}\right\}, \min_{k\in\mathcal{S}_{n+1}} T_k^{(d)}, t_{n+2}\right\}$ and repeat the above steps until the number of elements in $\tilde{\mathcal{S}}$ is less than Π. This implies that problem with $|\bar{\mathcal{K}}_{n+1}| = \Pi$ is infeasible and we set $\Pi_{\text{up}} = \Pi$. At each checkpoint t_n for $n = 1, \cdots, N-1$, we solve Problem (P7) and update $\mathcal{F}$ as $\mathcal{F}\setminus\bar{\mathcal{K}}_{n+1}$, $\mathcal{S}_{n+1}$ as $\mathcal{S}_{n+1}\cup\bar{\mathcal{K}}_{n+1}$ until $\mathcal{F}$ is empty.

The detailed steps for spectrum-hole allocation scheme is summarized in Algorithm 2 whose computational complexity lies in solving Problem (P7) with given Π. Specifically, with given Π, the complexity for (P7) is $\mathcal{O}\left(K^2\right)$. Since we have to solve (P7) with each Π and n, the total computational complexity for the proposed spectrum-hole allocation algorithm is estimated as $\mathcal{O}\left((N-1)K^2\log_2(K)\right)$. It should be noted that Algorithm 2 can obtain a globally optimal

Algorithm 2: Spectrum-Hole Allocation Algorithm

Initialize $\{T_k^{(a)}\}$, $\{T_k^{(d)}\}$, $\{A_k\}$, $\{B_k\}$, $\mathcal{F}$, $\bar{t}_1 = t_1$, and $\{\mathcal{S}_n\}$.
for $n = 1, \cdots, N-1$ **do**
  Set $\Pi_{\text{up}} = |\mathcal{F}|$, and $\Pi_{\text{low}} = 0$.
  repeat
    Set $\Pi = \lfloor (\Pi_{\text{up}} + \Pi_{\text{low}})/2 \rfloor$.
    Set $\mathcal{G} = \left\{ T_k^{(d)} \middle| k \in \mathcal{F} \right\}$.
    repeat
      Set $T^{(d)} = \min\{\min\{i | i \in \mathcal{G}\}, \min_{k \in \mathcal{S}_{n+1}} T_k^{(d)}, t_{n+2}\}$.
      Set $\bar{t}_{n+1} = T^{(d)} - d(\Pi + |\mathcal{S}_{n+1}|)$.
      Set $\tilde{\mathcal{S}} = \left\{ k \middle| T_k^{(a)} < \bar{t}_{n+1}, T_k^{(d)} \geq T^{(d)}, k \in \mathcal{F} \right\}$
      Sort the elements in $\tilde{\mathcal{S}}$ in an ascending order according to $\frac{\tilde{\ell}_k}{r_k\left(\bar{t}_{n+1} - \max\{T_k^{(a)}, \bar{t}_n\}\right)}$.
      Assemble the first Π elements of $\tilde{\mathcal{S}}$ in set $\bar{\mathcal{K}}_{n+1}$.
      if *the bandwidth and task-causality constraints are satisfied* **then**
        $\Pi_{\text{low}} \leftarrow \Pi$.
        Break.
      else
        $\mathcal{G} \leftarrow \mathcal{G} \setminus \min\{i | i \in \mathcal{G}\}$
      end
    until $|\tilde{\mathcal{S}}| < \Pi$;
    if $|\tilde{\mathcal{S}}| < \Pi$ **then**
      $\Pi_{\text{up}} \leftarrow \Pi$.
    end
  until $\Pi_{up} - \Pi_{low} \leq 1$;
  if Π_{up} *is feasible* **then**
    $\Pi^* = \Pi_{\text{up}}$
  else
    $\Pi^* = \Pi_{\text{low}}$
  end
  Update $\mathcal{F} \leftarrow \mathcal{F} \setminus \bar{\mathcal{K}}_{n+1}$, $\mathcal{S}_{n+1} \leftarrow \mathcal{S}_{n+1} \cup \bar{\mathcal{K}}_{n+1}$.
end

solution for Problem (P6) with low complexity.

VI. Extensions and Discussion

A. Online Task Admission

During the process of task uploading and inference, new tasks may arrive and submit service requests [35]. In this scenario, the proposed Algorithm 2 can be modified to support online admission of new tasks to leverage spectrum holes. Specifically, the proposed Algorithm 1 is executed as batching initialization for existing tasks. Then new tasks arriving during the inference process are first stored locally at devices. Similar to Algorithm 2, at each checkpoint t_n, we update the set of active tasks, $\mathcal{F}$, to include selected new tasks that are deemed feasible for successful

execution using spare resources. To this end, Algorithm 2 can be executed again to update the resource allocation to accommodate the new tasks.

B. Frequency-Selective Channels

The current assumption of frequency non-selective can be relaxed as follows. A frequency selective channel can be partitioned using orthogonal frequency division multiplexing (OFDM) into sub-channels with heterogeneous gains. Following [36], [37], new indicator variables can be introduced to denote the association between sub-channels and tasks. Then the throughput maximization problem can be formulated as a MINLP problem containing two kinds of binary optimization variables for sub-channel allocation and task-batch association, respectively. Despite being more complex, the problem can be solved efficiently using conventional MINLP methods such as convex relaxation and branch-and-bound, or latest approach using machine learning (see e.g., [38]).

VII. Simulation Results

A. Simulation Settings

The default settings are as follows. There are $K = 100$ devices, with task arrivals uniformly and independently generated in the time interval of $[0, 1]$ s. The size of feature vectors is set as 10 KBytes. The delay requirements of tasks follow the uniform distribution in $[0.05, 2]$ s. The inference latency profile with respect to the batch size as reported in [39] is adopted, which is generated from a ResNet-50 model implemented on JETSON TX1 and the ImageNet dataset. The channel gains between devices and server follow independent Rayleigh fading with the average power loss being 10^{-3}. The transmit *signal-to-noise ratio* (SNR) of devices is set as 20 dB. The constant δ in ℓ_0-norm approximation (13) is 10^{-15}. The following schemes are considered in performance comparison:

- *Proposed Algorithm:* See Algorithm 1.
- *Equal Bandwidth Allocation Scheme:* The total bandwidth is evenly allocated to devices while task scheduling follows Algorithm 1.
- *Spectrum-Hole Allocation Scheme:* Algorithm 1 enhanced with spectrum-hole allocation using Algorithm 2.
- *Greedy Batching Scheme:* Upon finishing executing the previous batching, the server greedily assembles all tasks that arrived during the previous batch into a new batch and makes

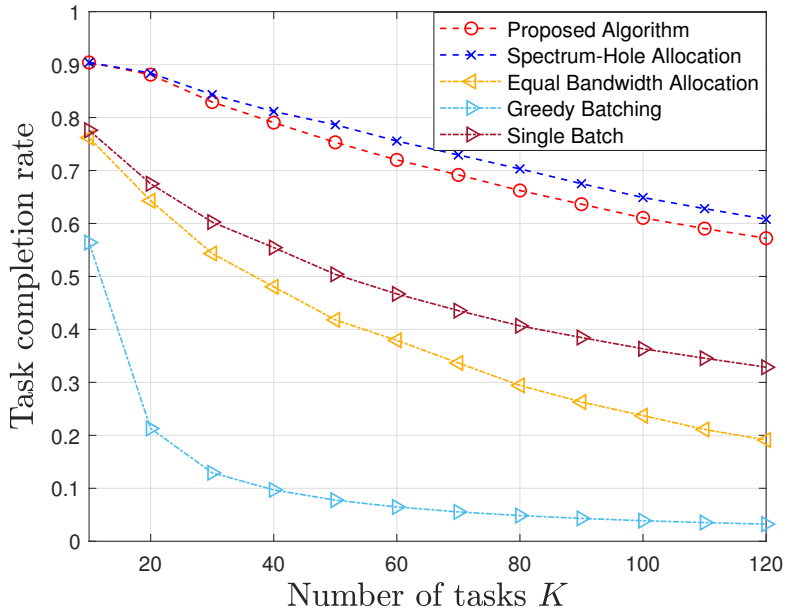

Fig. 3. Task completion rate versus number of tasks.

inferences on them [40]. When the inference is finished, those tasks that do not satisfy deadline requirements are discarded.

- *Single Batch Scheme:* The optimal single batch scheme in [18] is modified for asynchronous task arrivals. In particular, the optimal batch starting instant is determined using an algorithm similar to Algorithm 2.

The performance metric of task completion rate is defined as the ratio between completed tasks and all tasks. Note that the metric measures system throughput.

B. Effect of Task Number

Fig. 3 compares task completion rates between different schemes for a varying number of tasks. The proposed JBAS scheme and its enhanced version with spectrum-hole exploitation achieve the highest rates. This shows the advantages of jointly optimizing batching, scheduling, and bandwidth allocation so as to accommodate the heterogeneity of task arrivals and deadlines. In contrast, the three benchmarking schemes are less effective in accounting for the different delay requirements of tasks and balancing the tradeoff between batch size and batch startup instants. As a result, they suffer loss on system throughput that is larger as the number of tasks grows. On the other hand, we can observe that the Spectrum-Hole Allocation Scheme

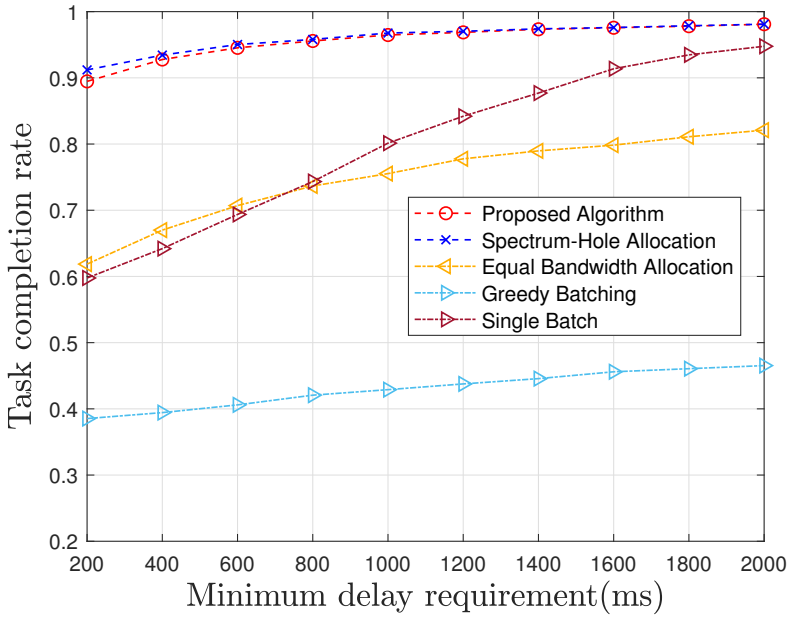

Fig. 4. Task completion rate versus the minimum delay requirement.

can enhance the throughput of the Proposed Scheme by an average of 2.8%. Furthermore, as observed from Fig. 3, the task completion rates decrease as the number of tasks grows. This indicates that the limited communication and computation source leads to an increasing slower in the number of completed tasks as the total number grows.

C. Effect of Delay Requirement

To investigate the effect of delay requirements on system throughput, we vary the minimum delay requirement from 50 to 1450 ms while the maximum delay requirement is fixed at 2000 ms. The curves of task completion rate versus the minimum delay requirements are depicted in Fig. 4. One can observe that the task completion rates of all schemes gradually increase as the minimum delay requirement relaxes. The reason is that less bandwidth is required for each task for uploading and the server has more computation time. From Fig. 4, we can observe that the throughput improvement of the Spectrum-Hole Allocation Scheme on top of the Proposed Scheme reduces from 1.7% to zero as the minimum delay increases from 200 to 2000 ms. This can be explained by that as the minimum delay increases, the radio resource constraints are relaxed and the communication bottleneck is dominated by the computation counterpart. In contrast, the throughput improvement of Spectrum-Hole Allocation Scheme is more significant

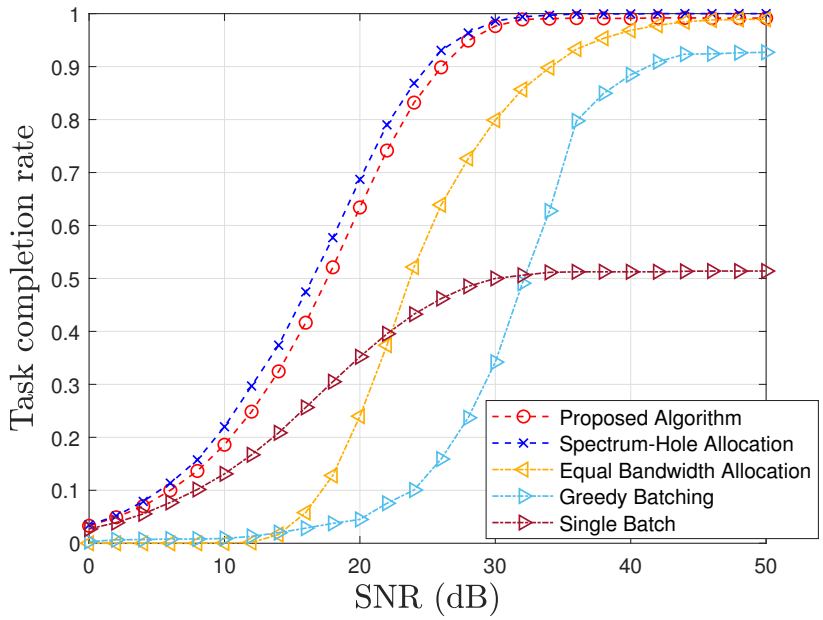

Fig. 5. Task completion rate versus transmit SNR.

in spectrum constrained scenarios, (i.e., a large number of tasks and tight delay requirements) as shown in Fig. 3 and Fig. 4. Furthermore, as the minimum delay increases, the throughput of the One Batch Scheme improves rapidly as the loss on synchronizing tasks' starting time reduces.

D. Effect of Transmit SNR

In Fig. 5, the curves of task completion rate performance versus transmit SNR are plotted. As the transmit SNR increases, the task completion rate first improves rapidly and then saturates. The early rapid improvement reflects the overcoming of the communication bottleneck. As the SNR is further increased, the bandwidth constraint becomes inactive, leading to throughput saturation. In this operation regime, the computation bottleneck dominates and limits system throughput. One can observe that with sufficiently large SNR (e.g., 50 dB), the Proposed, Spectrum-Hole Allocation, and Equal Bandwidth Allocation Schemes can complete almost all tasks, while the Greedy Batching Scheme only reaches 92% task completion rate, which verifies the need of batching optimization. On the other hand, the Single Batch Scheme performs worst at a large SNR, i.e., less than 48% task completion rate, indicating the importance of multiple batches for asynchronous tasks arrivals.

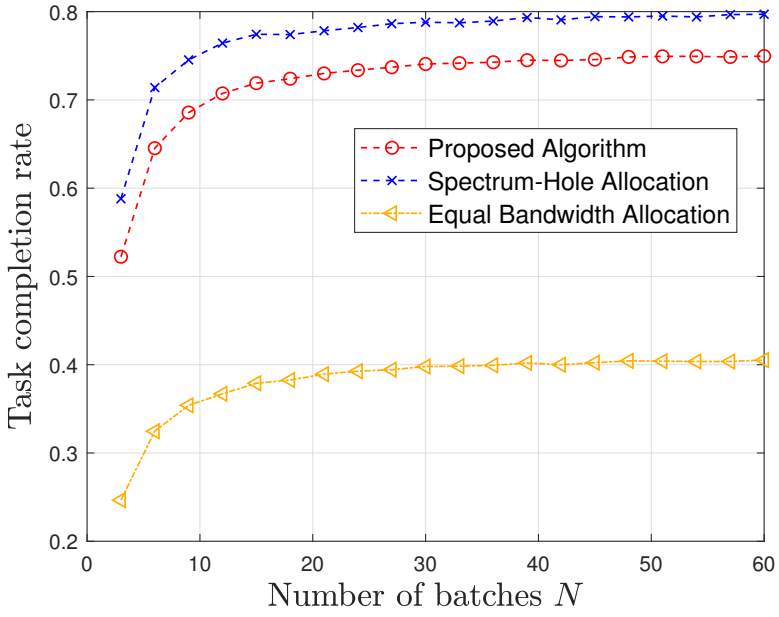

Fig. 6. Task completion rate versus number of batches with $K = 60$.

E. Effect of Batch Number

Fig. 6 shows the curves of task completion rate versus number of batches from solving Problem (P3). As can be observed, as the number of batches grows, the system throughput increases and then saturates as the effective number of batches, namely the non-empty ones, converges to a fixed value. Besides, the proposed Scheme and Spectrum-Hole Allocation Scheme achieve the throughput improvement of 39.34% and 47.50%, respectively, compared with Equal Bandwidth Allocation Scheme when $K = 60$.

F. Effect of Bandwidth

In Fig. 7, the curves of task completion rate versus total bandwidths are plotted. As can be seen, as the bandwidth increases, the throughput of all schemes increases due to the communication resources getting more abundant. The Proposed Scheme achieves 29.80%, 66.48%, and 35.89% throughput gains compared with Equal Bandwidth Allocation Scheme, Greedy Batching Scheme, and One Batch Scheme, respectively. Moreover, the Spectrum-Hole Allocation Scheme can further improve the throughput by 4.61%.

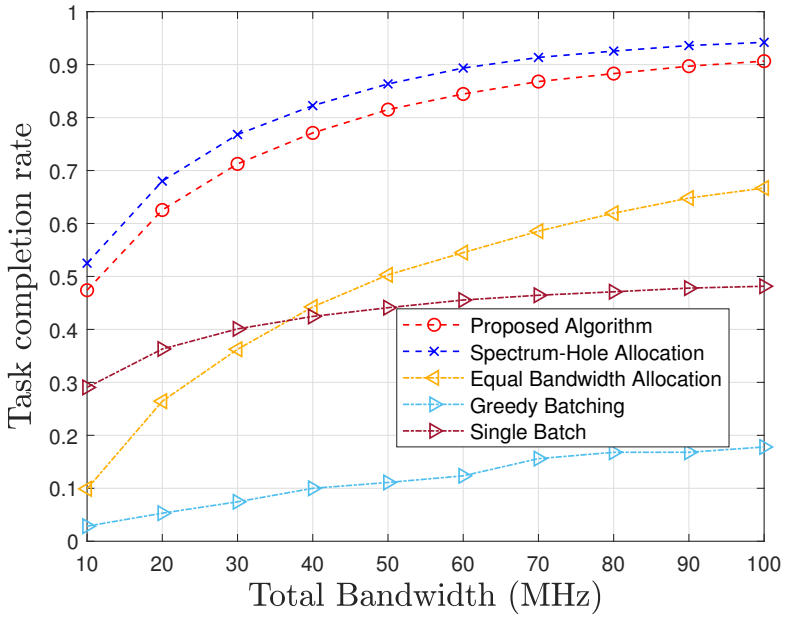

Fig. 7. Task completion rate versus total bandwidth.

VIII. Conclusion

In this paper, we have presented a JBAS framework for high-throughput multiuser edge AI in the practical scenarios with heterogeneous task arrivals and deadlines. The number of batches, batch startup instants, task-batch association, as well as bandwidth allocation have been jointly optimized to maximize the system throughput. Moreover, spectrum holes have been exploited to further increase the throughput. We find that judiciously assembling tasks into multiple batches is important to ensure high throughput in practice. However, the communication model considered in this paper is simple for the sake of tractability. For future works, it is promising to integrate batching with advanced transmission techniques such as *non-orthogonal multiple access* (NOMA) and *multiple-input multiple-output* (MIMO). In another interesting direction, multiple-cell edge AI with batching couples communications in different cells and computation at different servers and hence is more challenging to design.

Appendix

A. Proof of Theorem 1

Denote the optimal solution of Problem (P2) by $\left(\{t_n^*\}, \{\pi_{k,n}^*\}, N^*\right)$. We first prove that for the optimal solution of Problem (P2), the optimal value of (P3) is no less than that of (P2).

Consider the following two cases:

Case 1: $N^* = K$. If the optimal $N^* = K$, $\left(\left\{t_n^*\right\}, \left\{\pi_{k,n}^*\right\}\right)$ is feasible to Problem (P3) since that (P3) is the case when $N = K$.

Case 2: $N^* < K$. In this case, introducing new variables $t_n^* = \Xi$ $(\forall n \in \{N^* + 1, \cdots, K\})$ and $\pi_{k,n}^* = 0$ $(\forall k \in \mathcal{K}, \forall n \in \{N^* + 1, \cdots, K\})$. Then combing the optimal solution of (P2) and the new introduced variables, the constructed variables $\left(\left\{t_n^*\right\}_{\forall n \in \mathcal{K}}, \left\{\pi_{k,n}^*\right\}_{\forall k \in \mathcal{K}, \forall k \in \mathcal{K}}\right)$ satisfy all the constraints in Problem (P3). Moreover, the optimal value of (P2) is equal to the value of (P3). Therefore, the optimal value of (P3) is no less than that of (P2).

Next, since the optimal solution of (P3) always satisfies the constraints of (P2). Hence, the feasibility of (P3) is included in that of (P2). In other words, the optimal solution of (P3) is feasible to (P2). Thus, the optimal value of (P2) is no less than that of (P3).

Combining that the optimal value of (P2) is no less than and also no larger than that of (P3), we can conclude that Problem (P3) is equivalent to Problem (P2).

B. Proof of Theorem 2

Startup time t_n is lower bounded by the task-arrival time instants of its associated tasks and upper bounded by deadlines and start time of the next batch t_{n+1}. Since the allocated bandwidth B_k decreases with t_n, we should set t_n as large as possible in order to satisfy the bandwidth constraint. Through solving t_n sequentially from $n = N$ to $n = 1$, we can obtain the optimal solution of t_n. Specifically, for the N-th batch, we consider the following two cases: 1) If the N-th batch is non-empty, i.e., $\sum_{k=1}^{K} \pi_{k,N} > 0$, the time instant that the N-th batch finishes its inference $t_N + d_N(\pi_N)$ is upper bounded by the deadlines of its associated tasks. Hence, we should let $t_N = \min_{k \in \mathcal{K}_N} T_k^{(d)} - d_N(\pi_N)$. 2) If the N-th batch is empty, i.e., $\sum_{k=1}^{K} \pi_{k,N} = 0$, we should set t_N as large as possible such that it will not affect the value of t_{N-1}. Without loss of generality, we set $t_N = \Xi$. Subsequently, consider the $(N-1)$-th batch. Similarly, two cases are considered. If it is non-empty, time instant $t_{N-1} + d_{N-1}(\pi_{N-1})$ is restricted not only by the deadlines of its associated tasks but also by the startup time of the N-th batch. Therefore, t_{N-1} is set to $\min\left\{\min_{k \in \mathcal{K}_{N-1}} T_k^{(d)}, t_N\right\} - d_{N-1}(\pi_{N-1})$. If the $(N-1)$-th batch is empty, in order to mitigate the influence on the startup time of the $(N-2)$-th batch, we set $t_{N-1} = t_N$. Following this procedure until the startup time of the first batch t_1 is obtained, completing the proof.

C. Proof of Proposition 1

First, when $\Pi < \Pi^*$, we have $d(\Pi+|\mathcal{S}_{n+1}|) \leq d(\Pi^*+|\mathcal{S}_{n+1})$ since the inference delay function is non-decreasing. Therefore, let $\bar{t}_{n+1} = \bar{t}^*_{n+1}$ and $\bar{\mathcal{K}}_{n+1}$ be an arbitrary subset of $\bar{\mathcal{K}}^*_{n+1}$ with the size of Π. We can deduce $\bar{t}_{n+1}$ and $\bar{\mathcal{K}}_{n+1}$ satisfy all the constraints of (P7). Thus, Problem (P7) is feasible.

Subsequently, when $\Pi > \Pi^*$, Problem (P7) is always feasible, otherwise, the optimal $|\bar{\mathcal{K}}^*_{n+1}|$ of (P6) is larger than Π^*, which contradicts that $|\bar{\mathcal{K}}^*_{n+1}| = \Pi^*$. That completes the proof.

REFERENCES

[1] K. B. Letaief, Y. Shi, J. Lu, and J. Lu, "Edge Artificial Intelligence for 6G: Vision, Enabling Technologies, and Applications," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 5–36, Jan. 2022.

[2] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Toward an Intelligent Edge: Wireless Communication Meets Machine Learning," *IEEE Commun. Magazine*, vol. 58, no. 1, pp. 19–25, Jan. 2020.

[3] J. Backus, "Can programming be liberated from the von neumann style? a functional style and its algebra of programs," *Communications of the ACM*, vol. 21, no. 8, pp. 613–641, 1978.

[4] X. Zou, S. Xu, X. Chen, L. Yan, and Y. Han, "Breaking the von Neumann bottleneck: architecture-level processing-in-memory technology," *Sci. China Inf. Sci.*, vol. 64, no. 6, p. 160404, Apr. 2021.

[5] J. Shao and J. Zhang, "Communication-Computation Trade-off in Resource-Constrained Edge Inference," *IEEE Commun. Mag.*, vol. 58, no. 12, pp. 20–26, Dec. 2020.

[6] X. Tang, X. Chen, L. Zeng, S. Yu, and L. Chen, "Joint Multiuser DNN Partitioning and Computational Resource Allocation for Collaborative Edge Intelligence," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9511–9522, June 2021.

[7] W. Shi, Y. Hou, S. Zhou, Z. Niu, Y. Zhang, and L. Geng, "Improving device-edge cooperative inference of deep learning via 2-step pruning," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Paris, France, Apr 29–May 2, 2019.

[8] M. Jankowski, D. Gündüz, and K. Mikolajczyk, "Joint device-edge inference over wireless links with pruning," in *Proc. IEEE Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Atlanta, GA, USA, May 26–29, 2020.

[9] Q. Lan, Q. Zeng, P. Popovski, D. Gündüz, and K. Huang, "Progressive Feature Transmission for Split Classification at the Wireless Edge," *IEEE Trans. Wirel. Commun.*, vol. 22, no. 6, pp. 3837–3852, June 2023.

[10] J. Shao, Y. Mao, and J. Zhang, "Task-Oriented Communication for Multidevice Cooperative Edge Inference," *IEEE Trans. Wirel. Commun.*, vol. 22, no. 1, pp. 73–87, Jan. 2023.

[11] ——, "Learning Task-Oriented Communication for Edge Inference: An Information Bottleneck Approach," *IEEE J. Selected Areas Commun.*, vol. 40, no. 1, pp. 197–211, June 2022.

[12] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 1, pp. 447–457, Jan. 2020.

[13] W. Shi, S. Zhou, Z. Niu, M. Jiang, and L. Geng, "Multiuser Co-Inference With Batch Processing Capable Edge Server," *IEEE Trans. Wirel. Commun.*, vol. 22, no. 1, pp. 286–300, Jan. 2023.

[14] L. Zeng, X. Chen, Z. Zhou, L. Yang, and J. Zhang, "CoEdge: Cooperative DNN Inference With Adaptive Workload Partitioning Over Heterogeneous Edge Devices," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 595–608, Apr. 2021.

[15] B. Lu, J. Yang, J. Xu, and S. Ren, "Improving QoE of Deep Neural Network Inference on Edge Devices: A Bandit Approach," *IEEE Internet Things J.*, vol. 9, no. 21, pp. 21 409–21 420, Nov. 2022.

[16] D. Zhang, N. Vance, Y. Zhang, M. T. Rashid, and D. Wang, "EdgeBatch: Towards AI-Empowered Optimal Task Batching in Intelligent Edge Systems," in *Proc. IEEE Real-Time Systems Symposium (RTSS)*, Hong Kong, China, Dec. 2019, pp. 366–379.

[17] Y. Xu, J. Sun, S. Zhou, and Z. Niu, "SMDP-Based Dynamic Batching for Efficient Inference on GPU-Based Platforms," Jan. 2023. [Online]. Available: https://arxiv.org/abs/2301.12865

[18] Z. Liu, Q. Lan, and K. Huang, "Resource Allocation for Multiuser Edge Inference With Batching and Early Exiting," *IEEE J. Selected Areas Commun.*, vol. 41, no. 4, pp. 1186–1200, Apr. 2023.

[19] F. Qi, L. Zhuo, and C. Xin, "Deep Reinforcement Learning Based Task Scheduling in Edge Computing Networks," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Chongqing, China, Nov. 2020, pp. 835–840.

[20] C. You, Y. Zeng, R. Zhang, and K. Huang, "Asynchronous Mobile-Edge Computation Offloading: Energy-Efficient Resource Management," *IEEE Trans. Wirel. Commun.*, vol. 17, no. 11, pp. 7590–7605, Nov. 2018.

[21] S. Eom, H. Lee, J. Park, and I. Lee, "Asynchronous Protocol Designs for Energy Efficient Mobile Edge Computing Systems," *IEEE Trans. Veh. Techn.*, vol. 70, no. 1, pp. 1013–1018, Jan. 2021.

[22] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Commun. Surv. & Tutor.*, vol. 19, no. 3, pp. 1628–1656, Thirdquarter 2017.

[23] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices," *IEEE J. Selected Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[24] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-Efficient Offloading for Mobile Edge Computing in 5G Heterogeneous Networks," *IEEE Access*, vol. 4, pp. 5896–5907, Aug. 2016.

[25] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang, "Large-scale image classification: Fast feature extraction and SVM training," in *CVPR 2011*, Colorado Springs, CO, USA, June 2011, pp. 1689–1696.

[26] J. Li, W. Liang, Y. Li, Z. Xu, X. Jia, and S. Guo, "Throughput Maximization of Delay-Aware DNN Inference in Edge Computing by Exploring DNN Model Partitioning and Inference Parallelism," *IEEE Trans. Mobile Comput.*, vol. 22, no. 5, pp. 3017–3030, May 2023.

[27] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Weinberger, "Multi-scale dense networks for resource efficient image classification," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, Apr 30–May 3, 2018.

[28] B. Korte and R. Schrader, "On the existence of fast approximation schemes," in *Nonlinear Programming*. Academic Press, 1981, vol. 4, pp. 415–437.

[29] I. A. Elgendy, W.-Z. Zhang, Y. Zeng, H. He, Y.-C. Tian, and Y. Yang, "Efficient and Secure Multi-User Multi-Task Computation Offloading for Mobile-Edge Computing in Mobile IoT Networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 4, pp. 2410–2422, Dec. 2020.

[30] B. Dai and W. Yu, "Energy Efficiency of Downlink Transmission Strategies for Cloud Radio Access Networks," *IEEE J. Selected Areas Commun.*, vol. 34, no. 4, pp. 1037–1050, Apr. 2016.

[31] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang, "A Survey of Sparse Representation: Algorithms and Applications," *IEEE Access*, vol. 3, pp. 490–530, May 2015.

[32] L. Grippo and M. Sciandrone, "On the convergence of the block nonlinear Gauss–Seidel method under convex constraints," *Operations research letters*, vol. 26, no. 3, pp. 127–136, Apr. 2000.

[33] D. P. Bertsekas, *Convex optimization Theory*. Athena Scientific Belmont, 2009.

[34] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[35] Z. Chen, S. Zhang, Z. Ma, S. Zhang, Z. Qian, M. Xiao, J. Wu, and S. Lu, "An Online Approach for DNN Model Caching and Processor Allocation in Edge Computing," in *Proc. IEEE/ACM Intern. Symp. Quality Serv. (IWQoS)*, Oslo, Norway, June 2022, pp. 1–10.

[36] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading," *IEEE Trans. Wirel. Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[37] C. Xiong, G. Y. Li, S. Zhang, Y. Chen, and S. Xu, "Energy- and Spectral-Efficiency Tradeoff in Downlink OFDMA Networks," *IEEE Trans. Wirel. Commun.*, vol. 10, no. 11, pp. 3874–3886, Nov. 2011.

[38] S. Bi, L. Huang, H. Wang, and Y.-J. A. Zhang, "Lyapunov-Guided Deep Reinforcement Learning for Stable Online Computation Offloading in Mobile-Edge Computing Networks," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 11, pp. 7519–7537, Nov. 2021.

[39] NVIDIA, "NVIDIA Jetson TX2 Delivers Twice the Intelligence to the Edge," [Online] https://devblogs.nvidia.com/jetson-tx2-delivers-twiceintelligence-edge/, 2019.

[40] A. Ali, R. Pinciroli, F. Yan, and E. Smirni, "BATCH: Machine Learning Inference Serving on Serverless Platforms with Adaptive Batching," in *Proc. Intern. Conf. High Perform. Comput., Netw., Storage Anal.*, Atlanta, GA, USA, Nov. 2020, pp. 1–15.