Delay-Optimal Distributed Computation Offloading in Wireless Edge Networks

Xiaowen Gong, *Member, IEEE*, Mingyu Chen, *Student Member, IEEE*, Dongsheng Li, *Student Member, IEEE*, and Yang Cao, *Member, IEEE*

***Abstract*— In this paper, we explore distributed edge computation offloading (DECO) that offloads computation to distributed edge devices connected wirelessly, which perform the offloaded computation in parallel. By integrating edge computing with parallel computing, DECO can substantially reduce the total computation delay. In particular, we study the fundamental problem of minimizing the total completion time of DECO. We show that the time-sharing based communication resource allocation always outperforms the bandwidth-sharing scheme, so that it suffices to focus on the time-sharing based communication scheduling. Based on the time-sharing scheme, we first establish some structural properties of the optimal communication scheduling policy. Then, given these properties, we develop an efficient algorithm that finds the optimal allocation of computation workloads. Next, based on the optimal computation allocation, we characterize the optimal scheduling order of communications, which exhibits an elegant structure: the optimal order is in the non-decreasing order of the ratio between a device's computation rate and its communication time. Last, based on the optimal computation allocation and communication scheduling, we show that the optimal device selection problem is a submodular minimization problem, so that it can be solved efficiently using some existing methods. We further extend the study to the setting where devices are subject to maximum computation workload constraints, and develop an efficient algorithm that finds the optimal computation workload allocation. Our results provide useful insights for the optimal computation-communication co-design for DECO. We evaluate the theoretical findings using extensive simulations in both practical settings and controlled settings, which demonstrate the performance of DECO in practice and also the efficiency of our proposed schemes and algorithms for DECO.**

***Index Terms*— Edge computing, distributed and parallel computing, communication scheduling, delay minimization.**

I. Introduction

EDGE computing has emerged as a promising paradigm that performs a substantial amount of computing, storage, networking, and management functions on devices at or close to end users (referred to as "edge devices"). This trend is largely enabled by the proliferation of smart devices with powerful computing capabilities, which are often not fully utilized. Compared to cloud computing which performs computing in remote data centers, edge computing can offload a large amount of data traffic from the core network to the edge network, which can greatly reduce the communication delay incurred in the network. One main driving force for the popularity of edge computing is many emerging applications of AI that require very low service latency, including augmented reality (AR) [2], virtual reality (VR) [3], and autonomous vehicle [4]. These applications are empowered by recent advances in machine learning (ML), which typically rely on computationally intensive processing of large amounts of data.

On the other hand, distributed computing is a traditional computing paradigm that uses distributed devices to perform computing cooperatively in *parallel*. One main advantage of this approach is that it can greatly reduce computation time by parallelizing computation and distributing computation workloads to different devices, rather than performing all the computation on a single device. One well-known example of distributed computing is cloud computing in data centers, which utilizes large clusters of interconnected computer servers for parallel computing.

To exploit the potential of edge computing, it is promising to leverage *distributed edge computation offloading* (DECO), which offloads computation to distributed edge devices connected wirelessly, so that the devices can perform the offloaded computation in parallel. This approach is enabled by widely available edge devices with under-utilized computation capabilities which can be connected by wireless networks in a distributed manner. To accelerate many emerging applications that require very low latency, it is beneficial to *offload* and *distribute* a large computation workload from a single end device to multiple devices at the network edge. One key advantage of this approach is computation parallelization which reduces the computation delay. For example, if a computation workload is offloaded from one device and distributed to N devices with the same computing power, and if communication delays are not counted (which certainly should and will be considered), then the computation delay will reduce by a fold of N, which is very substantial.

In this paper, we study distributed edge computation offloading (DECO) that exploits wirelessly connected edge devices to perform offloaded computation in parallel. Our goal is to minimize the total completion time of DECO. To this end, we will study the optimal allocation of computation workloads (also referred to as "computation allocation") to the edge devices. We will also investigate the optimal scheduling

Manuscript received 27 June 2022; revised 9 February 2023, 21 July 2023, and 3 January 2024; accepted 1 April 2024; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor L. Huang. This work was supported by U.S. NSF under Grant ECCS-2121215. The work of Xiaowen Gong was supported by Auburn University. Preliminary results of this work was published in [DOI: 10.1109/INFOCOM41043.2020.9155272]. *(Corresponding author: Xiaowen Gong.)*

Xiaowen Gong and Dongsheng Li are with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849 USA (e-mail: xgong@auburn.edu; dzl0093@auburn.edu).

Mingyu Chen and Yang Cao are with the School of Electronics and Information Engineering, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China (e-mail: chenmingyuwuhan@gmail.com; ycao@hust.edu.cn).

Digital Object Identifier 10.1109/TNET.2024.3394789

of communications among the devices in a wireless edge network. We will further study the optimal selection of devices that participate in the computation offloading. We will seek fundamental understandings and develop efficient algorithms with provable performance guarantees for DECO.

The cross-layer design, analysis, and optimization of DECO in wireless edge networks are non-trivially different from prior studies. First of all, the computation workload allocation and communication scheduling problems have non-trivial coupling, as the optimal solution to each problem depends on the solution to the other problem. Therefore, these two issues should be designed jointly in a judicious manner. Moreover, compared to existing works on distributed computing, DECO needs to take into account the features of wireless networks, including *interference* among wireless devices and *diverse* communication rates of wireless links. In particular, due to interference constraints among communications, the communication scheduling problem is highly challenging. Furthermore, DECO aims to minimize the total completion time of all computations and communications involved in the computation offloading under their precedence constraints, which is a complex objective and is quite different from existing studies on edge computation offloading and wireless network scheduling.

The main contributions of this paper can be summarized as follows.

- We propose a framework of distributed edge computation offloading (DECO), which offloads computation from an end device to distributed edge devices connected wirelessly, so that the edge devices can perform the offloaded computation in parallel. Based on this framework, we formulate the problem of allocating computation workloads to the devices and scheduling communications between the devices for minimizing the total completion time of the computation offloading.
- We show that the time-sharing based communication resource allocation is always better than the bandwidth-sharing scheme. This result provides the insight that the time-sharing scheme reduces the communication time of each device by using all the bandwidth for that device, so that it increases the available time of that device for computation.
- Based on the time-sharing scheme, we first establish some structural properties of the optimal communication scheduling, which show that it is optimal to be non-preemptive, be non-idle, and schedule forward communications before backward communications. Then, given these properties, we develop an efficient algorithm that finds the optimal allocation of computation workloads. Next, based on the optimal computation allocation, we characterize the optimal scheduling order of communications, which exhibits an elegant structure: the optimal order is in the non-decreasing order of the ratio between a device's computation rate and its communication time. Last, based on the optimal computation allocation and communication scheduling, we show that the problem of selecting the optimal set of participating devices is a submodular minimization problem. Therefore, the optimal device selection can be found efficiently using some existing methods (e.g., the ellipsoid method). Our results provide some useful insights for the optimal computation-communication co-design for DECO.
- We further extend the study to the setting where each device is subject to a maximum computation workload constraint. For this case, we develop an efficient algorithm that finds the optimal computation workload allocation.
- We evaluate the performance of the proposed schemes and algorithms for DECO using extensive simulations in both practical settings and controlled settings. The simulation results demonstrate the performance of DECO compared to benchmarks in practice, and also the efficiency of the proposed optimal schemes and algorithms for DECO.

The rest of this paper is organized as follows. Section II reviews related work. In Section III, we describe a framework of distributed computing offloading in wireless edge networks. We first show in Section IV that the time-sharing based communication resource allocation outperforms the bandwidth-sharing scheme. Then Section V focuses on the optimal computation allocation and communication scheduling. We study the optimal device selection in Section VI. We extend the study to the setting with maximum computation workload constraints in Section VII. Simulation results are discussed in Section VIII. Section IX concludes this paper and discusses future work.

II. Related Work

Edge Computation Offloading. Edge computing has attracted growing research interests in the past few years [5]. Many works have used edge devices for video applications that require low service delays, such as for video rendering [6], and virtual reality [7], [8]. One important application studied in these works is real-time video inference [9], [10], [11]. Computation offloading from mobile devices to edge servers has also been studied [12], [13], [14], [15]. Another major research direction of edge computing is edge caching [16]. Learning users' interested contents for caching has also been studied [17]. Cooperative networks of caches have also been studied [18], [19]. However, existing works on edge computation offloading have not considered offloading computation to *multiple edge devices in parallel*, with the goal of reducing the computation offloading delay.

Distributed and Parallel Computing. There have been many studies on the design of distributed algorithms and computation allocation for reducing computation delays [20], [21], [22], [23], [24], [25]. Some of these works have studied the impacts of the network on computing [26], [27], [28]. Some other works have considered the throughput of networked computers for processing computations [29]. Recent studies have considered the cross-layer design of distributed computing and networking for improving computation delay [30] or throughput [31], [32]. On the other hand, many works have studied communication scheduling in data center networks [33]. A large body of these works have focused on the scheduling of co-flows under distributed computing frameworks, in particular MapReduce [34], [35], [36]. However, existing works on distributed computing have not considered distributing computation by *wireless devices* in parallel for reducing the computation delay, which should take into account the features of wireless networks, such as interference.

Wireless Network Scheduling. Wireless network scheduling has been studied extensively for more than a decade.

Most of the works have focused on the throughput of wireless networks [37], including recent works on deadline-constrained throughput [38] and with distributed scheduling [39]. Many other works have considered the total utility of data flows in the network [40] which depends on the throughput. Much fewer works have studied the delay performance of wireless network scheduling [41]. On the other hand, some works have studied the cross-layer design of scheduling, routing, and/or congestion control for various objectives, including for throughput [42], [43], delay [44], or utility [45]. However, existing works on wireless network scheduling have not considered using distributed wireless devices to *perform computation in parallel*, with the goal of reducing the computation delay.

III. System Model and Problem Formulation

In this section, we first describe a framework of distributed computation offloading in wireless edge networks. Based on this framework, we then formulate the problem of minimizing the total completion time of the computation offloading.

Distributed Edge Computation Offloading (DECO). We consider a wireless end device which aims to perform a computation job. The computation job is divided into multiple computation tasks and offloaded to nearby edge devices in a distributed manner, so that the computations can be performed by the devices in parallel. By exploiting the computing power of distributed edge devices, this parallelization can greatly reduce the total delay of computation offloading.

In this paper, we assume that the computation job has a parallel structure,[1] such that it can be decomposed into multiple computation tasks in parallel, as illustrated in Fig. 1 (a). The parallel structure can capture many applications where the computation job can be parallelized. For example, for graphic rendering [46], a graphic can be partitioned into multiple segments such that different segments can be processed independently in parallel. For another example, an image recognition job can involve recognizing a set of different objects of interest (person, vehicle, building, animal, etc) from an image. Then the job can be divided into parallel tasks, where each task is to recognize a subset of all the objects of interest. For a computation job that cannot be fully parallelized, sometimes they can be partly parallelized. For instance, for image classification using a deep neural network (DNN) model, although different layers of the model have to be processed in order, some layer(s) can be parallelized individually (i.e., one layer can be divided into parallel partitions with each partition consisting of some neurons in the layer). In this case, our proposed DECO can apply to the parallelizable part of the computation job.

Computation Task. Each computation task takes some data as input, execute some instructions (e.g., arithmetic operation, comparison, branching such as "if…then…") based on the input data, and produces some data as output. The *workload* of a computation task (e.g., the number of arithmetic operations) generally varies for different computation tasks. For example, for the aforementioned image recognition job, the workload of a computation task can be determined by how many different objects are to be recognized in the task. Some computation tasks cannot be performed in parallel and have to be performed in order. This is the case when the output of a computation task is used as the input of another computation task, such that the latter cannot start until the former is completed. The precedence relations between computation tasks of the computation job can be represented by a directed acyclic graph (as illustrated in Fig. 1 (a)).

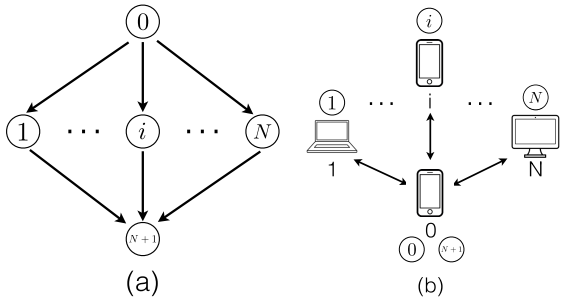

Fig. 1. (a) A computation job can be divided into parallel computation tasks: Each device represents a computation task; each directed edge represents a communication task. (b) The computation tasks constituting the computation job are offloaded to edge devices connected by wireless links.

In this paper, we assume that the total computation workload w of the computation job is fully divisible, such that it can be arbitrarily divided into any workloads w_i, $\forall i \in \mathcal{N}$ of parallel computation tasks i, $\forall i \in \mathcal{N}$ with $\sum_{i\in\mathcal{N}} w_i = w$. For ease of exposition, without loss of generality, we assume that the workloads of computation tasks 0 and $N+1$ are 0. The results of this paper may be extended to case where the total workload w is not fully divisible.[2]

Communication Task. For two computation tasks that have to be performed in order, the output of the first task has to be communicated to the second task as the input. The message passing between two computation tasks is referred to as a communication task. The *workload* of a communication task (typically captured by the amount of data to be transferred) can vary for different communication tasks.

The communication tasks involved in the computation offloading consist of *forward* communications (FMs, from computation task 0 to each computation task $i \in \mathcal{N}$) and *backward* communications (BMs, from each computation task $i \in \mathcal{N}$ to computation task $N+1$). For example, for the aforementioned image recognition job, in forward communications, the end device sends the image and what objects to recognize to each edge device; in backward communications, each edge device sends the result of object recognition to the end device. We assume that the workloads of forward and backward communication are independent of the associated computation tasks' workloads w_i, $i \in \mathcal{N}$.[3] This is the case when the amounts of input and output data of a computation task are independent of the computation workloads.

Wireless Edge Network. We consider a set of edge devices $\mathcal{N} \triangleq \{1, 2, \cdots, N\}$ that perform the offloaded computation tasks. Each computation task i is allocated to and performed

[1] In general, a computation job consists of interdependent computation tasks which can be represented by a directed acyclic graph (DAG). Our future work will explore this general setting which is challenging.

[2] The optimal workload allocation found in this paper can be used to find feasible workload allocation in the case where the total workload is not fully divisible. For example, we can use the feasible workloads that are the closest to the optimal workloads found in the fully divisible case. Under some conditions (e.g., the total workload can be divided into arbitrary partitions where each partition is a multiple of some minimum workload unit), some performance guarantee of the feasible workload allocation can be provided.

[3] We will study the setting where the workload of a communication task depends on that of the associated computation task (e.g., with a proportional relation) in future work.

by edge device i as illustrated in Fig. 1 (b). The computation rate of a device is the computation workload that the device can complete per unit time, which quantifies the computation capability (depending on e.g., CPU, memory) of the device, and can vary for different devices. Let r_i denote the computation rate of each edge device i.

The edge devices are connected by wireless links. Due to interference, only wireless links without mutual interference can transmit data concurrently. The communication rate of a wireless link is the communication workload that the link can complete per unit time, which quantifies the communication capability (depending on e.g., transmit power, channel condition) of the link, and can vary for different wireless links. The communication rates of each device i's FM and BM at time t are determined by and proportional to the bandwidth $b_i(t)$ used by device i at time t for its FM and BM, respectively.

We consider a single-hop wireless network such that each device can transmit data to each other device directly. The network is subject to complete interference constraints (e.g., [38], [39], [40]), such that multiple devices transmitting at the same time must use disjoint frequency bands (i.e., no more than one device can transmit concurrently in the same frequency band). This is a reasonable setting when devices are close to each other, which is usually the case in wireless edge networks (e.g., WiFi network). As a result, the total bandwidth used by all devices at any time must be no greater than the total available bandwidth b of the network (i.e., $\sum_{\in \mathcal{N}} b_i(t) \leq b$, $\forall t$). Let s_i and d_i denote the forward communication time and backward communication time of each edge device i, respectively, when all the available bandwidth b of the network is allocated to edge device i.

We assume that the computation rates of devices and communication rates of wireless links are known[4] (which can be estimated before computation offloading). As a result, the times of performing computation and communication tasks are known. We also assume that communications among the devices are coordinated by a central controller (e.g., WiFi access point), such that there is no contention or interference in the network. This can be achieved, e.g., using the point coordination function protocol of WiFi.

Total Completion Time. The total completion time (also referred to as "total delay") for DECO is the total time it takes to complete all the computation and communication tasks of the computation job, subject to the precedence constraints among computation tasks and communication tasks.

Based on the framework described above, our goal is to solve the following problem.

Definition 1 (Total Completion Time Minimization for DECO): We aim to optimize the allocation of computation workloads w_i, $\forall i \in \mathcal{N}$ to each edge device $i \in \mathcal{N}$, the schedules of communications between the end device and the edge devices (represented by the bandwidth $b_i(t)$, $\forall t$ used by each edge device $i \in \mathcal{N}$ for its FM and BM), and the selection of the edge devices participating in the computation offloading, under the precedence constraints among computations and communications and the total bandwidth constraints of communications, in order to minimize the total completion time of the computation offloading.

Note that, given the computation workload w_i of each device i, the optimal schedule of its computation is obvious: any computation schedule that starts after device i's FM and ends before device i's BM and completes its computation workload w_i is optimal. Therefore, we only consider computation workload allocation in Definition 1 rather than computation scheduling.

[4]We will study the setting where computation and communication rates are unknown and stochastic in future work.

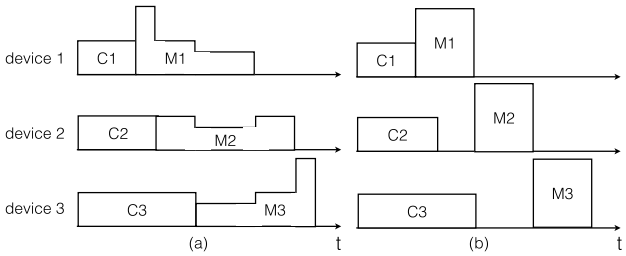

Fig. 2. Schedule of communications based on (a) bandwidth-sharing; (b) time-sharing. The axes in all the figures of this paper represent time, and their labels are omitted for brevity for the rest of the paper.

Also note that although it is possible to mathematically formulate the above problem rigorously, we choose to not do it here because 1) it is a prohibitively complex problem formulation as it involves a huge space of possible policies (including time-sharing and bandwidth-sharing based communication scheduling) and complex constraints (particularly precedence constraints among computations and communications, interference constraints among communications); 2) such a complete mathematical problem formulation is not used to solve the problem in this paper.

IV. Time-Sharing Vs Bandwidth-Sharing Based Communication Resource Allocation

In this section, we study the multiple access scheme for communication resource allocation in the wireless edge network. Due to interference among wireless devices in the network edge, the devices need to share the limited wireless communication resource either in time or in bandwidth to avoid mutual interference. We investigate the performance of the time-sharing versus the bandwidth-sharing scheme in terms of reducing the total delay of DECO.

We first consider a simple bandwidth-sharing scheme where each device is allocated with a fixed proportion of the total bandwidth in the edge network for communication (as illustrated in Fig. 2 (a)). It can be easily seen that this scheme is not efficient, as there are times when the total bandwidth is not fully utilized, such that the algorithm delay can be reduced if some unused bandwidth would have been utilized. Therefore, we consider a more complex bandwidth-sharing scheme where the total bandwidth is always *fully* utilized when some device(s) needs communication (as illustrated in Fig. 2 (b)), and the proportion of bandwidth allocated to a device can vary over time. This scheme *appears* to be efficient. However, for this schedule, we can find a time-sharing based schedule as in Fig. 2 (b), which achieves a total delay *no greater* than that in Fig. 2 (b). Intuitively, communication scheduling based on time-sharing is more efficient than based on bandwidth-sharing, as the former *reduces* the communication time of each device by using all the bandwidth for that device, so that it *increases* the available time of that device for computation before the communication starts. This observation is essentially due to the intricate coupling between computation and communication, as a device's computation should *precede* its communication.

Based on the discussions above, next we formally show that *time-sharing* always performs *at least as well as* bandwidth-sharing based communication scheduling.

Theorem 1: For any bandwidth-sharing based communication schedule, there exists a time-sharing based communication schedule which achieves a total delay no greater than that of the bandwidth-sharing based schedule.

The proofs of all theoretical results of this paper are provided in the appendix (in the order in which the corresponding results are given in this paper). As a result of Theorem 1, it suffices to only consider time-sharing based communication scheduling, which will be the focus in the rest of this paper.

V. Optimal Computation Workload Allocation and Optimal Communication Scheduling

In this section, based on the time-sharing based communication scheduling studied in Section IV, we study the optimal allocation of computation workloads and the optimal scheduling of communications that minimize the total completion time of DECO. We assume that all the available edge devices participate in the computation offloading. In Section VI, we will relax this assumption and study the optimal selection of the participating devices, based on the optimal computation allocation and communication scheduling. Note that once the optimal computation allocation and communication scheduling are determined, the optimal scheduling of computations on the devices can be easily determined: each computation starts once its corresponding FM ends.

We note that there is non-trivial interdependence between computation allocation and communication scheduling: the optimal design for one problem depends on the design for the other problem. In the following, we will first show some general structural properties that are satisfied by the optimal communication scheduling. Then, given any communication scheduling policy with these properties, we will characterize the optimal computation allocation. Next, based on the optimal computation allocation policy, we will characterize the optimal communication scheduling.

A. Structural Properties of Optimal Communication Scheduling

In general, a communication scheduling policy can be preemptive such that the network can interrupt the execution of a communication at any time and start to execute another communication [47]. However, we can show that it suffices to focus on non-preemptive policies.

Lemma 1: It is optimal for the communication scheduling policies to be non-preemptive.

Lemma 1 shows that it is not beneficial to preempt an ongoing communication to execute another communication. Intuitively, this is because preemptive scheduling is typically better than non-preemptive scheduling when tasks become available at different times and the objective is to minimize the total delay of tasks [47]. In contrast, for the problem here, the communications are always available (subject to that a BM is after the corresponding FM), and the objective is to minimize the algorithm delay which is equal to the maximum delay (i.e., total completion time) of communications.

Then we show that it is optimal to schedule all the FMs before all the BMs.

Lemma 2: It is optimal to schedule all FMs before all BMs.

Lemma 2 provides the insight that it is always beneficial to schedule any FM before any BM compared to the other way, as it allows for more time to execute the computations associated with these communications.

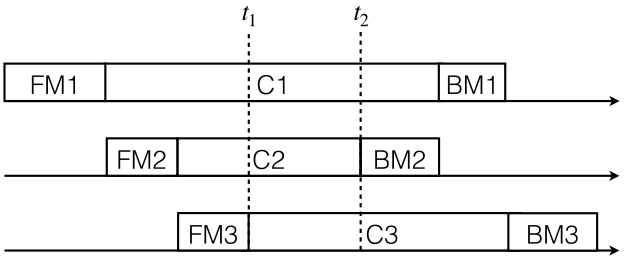

Fig. 3. Each axis shows the schedule of computations (Cs) and communications (FMs and BMs) for a device.

Next we show that it is optimal for the wireless network to keep busy between FMs and between BMs.

Lemma 3: It is optimal for the communication scheduling policies to be non-idle between FMs and between BMs, respectively.

The non-idle optimal policy in Lemma 3 means that the wireless network has no idle period between any two FMs and between any two BMs. However, there can be some idle period between the last FM and the first BM (i.e., between time t_1 and time t_2 in Fig. 3). Lemma 3 provides the insight that the wireless network should keep performing communications without any idle period, so as to complete communications as soon as possible, unless it is necessary to wait for some period during which the devices can perform computations.

B. Optimal Computation Workload Allocation

In this subsection, we study the optimal allocation of computation workloads to devices, given any communication scheduling policy that satisfies the structural properties discussed in Section V-A.

The optimal computation allocation can be found by an efficient algorithm that consists of up to three phases as described in Algorithm 1. Note that o_i^f (or o_i^b) denotes the index of the device whose FM (or BM, respectively) is at the ith position among all devices. In particular, in Phase 1, we first allocate computation workloads as much as possible to devices such that all these workloads can be completed before the last FM ends (i.e., before time t_1 in Fig. 3). If the total workload of the algorithm can be fully allocated in this way, we have found the optimal computation allocation. Otherwise, in Phase 2, we allocate computation workloads as much as possible to devices such that all these workloads can be completed after the first BM starts (i.e., after time t_2 in Fig. 3). If the remaining workload of the algorithm can be fully allocated in this way, we have found the optimal computation allocation. Otherwise, in Phase 3, we allocate the further remaining workload of the algorithm to devices such that it can be completed after the last FM ends and before the first BM starts (i.e., between time t_1 and time t_2 in Fig. 3). It can be seen that the computational complexity of Algorithm 1 is $O(N)$, as each phase of the algorithm involves at most N iterations. We establish the optimality of Algorithm 1 as follows.

Proposition 1: For any communication scheduling policy that satisfies the structural properties in Lemmas 1, 2, and 3, Algorithm 1 finds the optimal allocation of computation workloads to devices.

Proposition 1 provides some interesting insights regarding the optimal computation allocation characterized by

Algorithm 1 Optimal Computation Workload Allocation

input: computation rates $\{r_i\}$, FM order $\{o_i^f\}$, BM order $\{o_i^b\}$, FM and BM times $\{s_i\}$ and $\{d_i\}$, total computation workload w;
$w_i = 0, \forall i \in \mathcal{N}$, $j = 1$;
// Phase 1;
while $w > 0$ *and* $j \leq N-1$ **do**
  $w_{o_j^f} = \min(r_{o_j^f} \sum_{i=j+1}^{N} s_{o_i^f}, w)$;
  $w = w - w_{o_j^f}$;
  $j = j + 1$;
end
// Phase 2;
$j = N$;
while $w > 0$ *and* $j \geq 2$ **do**
  $w' = \min(r_{o_j^b} \sum_{i=1}^{j-1} d_{o_i^b}, w)$;
  $w_{o_j^b} = w_{o_j^b} + w'$;
  $w = w - w'$;
  $j = j - 1$;
end
// Phase 3;
if $w > 0$ **then**
  for $j \in \mathcal{N}$ **do**
    $w_j = w_j + w r_j / \sum_{i \in \mathcal{N}} r_i$;
  end
end
output: optimal computation workload $\{w_i\}$.

Algorithm 1. Intuitively, the optimal policy should reduce the idle computing periods of devices as much as possible, so as to minimize the algorithm delay. To this end, it allocates workloads to the idle period of each device after its FM ends and before the last FM (among all devices) ends, and to the idle period after the first BM (among all devices) starts and before its BM starts, until there is no such idle period. The workloads allocated to these idle periods do not increase the algorithm delay, as it remains equal to the total delay of all forward and BMs. If there is some workload of the algorithm that remains unallocated after the above allocation, it is allocated to all devices in proportional to their computation rates, such that it incurs an equal computation delay to all the devices. This delay increases the algorithm delay beyond the delay incurred by communications. As a result of Proposition 1 and Algorithm 1, we can see that when the total workload of the algorithm is sufficient (above some threshold), each device keeps performing its computation between its forward and BMs. Otherwise, some device is forced to be idle between its forward and BMs.

C. Optimal Communication Order

In this subsection, based on the structural properties of the optimal communication scheduling in Section V-A and the optimal computation allocation in V-B, we study the optimal scheduling order of communications. Due to symmetry between FMs and BMs, we focus on the scheduling order of BMs, as the results for FMs follow similarly. In particular, we consider the optimal scheduling order that minimizes the algorithm delay, given the total computation workload w of the algorithm. Based on the optimal computation allocation found by Algorithm 1, we can transform this problem to an equivalent "dual" form: how to schedule the BMs, such that the total computation workload v that can be completed *after* the first BM starts (i.e., after time t_2 in Fig. 3) is maximized? This equivalent problem is given by

$$\max_{\{o_i^b, \forall i \in \mathcal{N}\}} \sum_{i=2}^{N} r_{o_i^b} \Big(\sum_{j=1}^{i-1} d_{o_i^b}\Big).$$

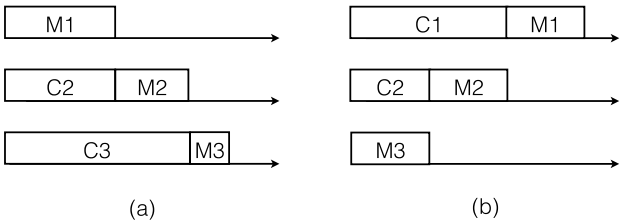

Fig. 4. (a) It is optimal that the longest communication M1 is scheduled first; (b) It is optimal that the fastest computing device (which is device 1) finishes its computation last.

We first consider the case where devices have uniform computation rates but can have diverse communication times. The optimal scheduling order is given as follows.

Proposition 2: When devices have uniform computation rates, it is optimal to schedule communications in the descending order of devices' communication times.

Proposition 2 means that the optimal policy schedules the longest communication (i.e., with the largest delay) first, and the second longest next, etc, as illustrated in Fig. 4(a). This result provides the following insight: it is better to schedule a longer communication earlier than a shorter one, since it allows for more time for other device(s) to perform computing. As a result, the computing capabilities of devices are most efficiently utilized and thus the total delay is minimized.

Then we consider the case where devices have uniform communication times but can have diverse computation rates. The optimal scheduling order is given below.

Proposition 3: When devices have uniform communication times, it is optimal to schedule communications in the ascending order of devices' computation rates.

Proposition 3 means that the optimal policy schedules the communication of the slowest device first, and that of the second slowest device next, etc, as illustrated in Fig. 4(b). The insight from this result is as follows: it is better to utilize a faster-computing device for a longer period than a slower-computing device, so that the computing capabilities of devices are most efficiently utilized and thus the total delay is minimized.

Next we consider the general case where devices can have diverse computation rates and also diverse communication times.

Theorem 2: It is optimal to schedule communications in the non-increasing order of the ratio between a device's computation rate and its communication time, i.e., r_i/d_i.

Theorem 2 provides an elegant result: the optimal communication order is simply in the non-increasing order of the computation-rate-to-communication-time *ratio* r_i/d_i. Intuitively, similar to Propositions 2 and 3, to maximize the total computation workload that can be completed, we can expect that a device with a higher computation rate or a smaller

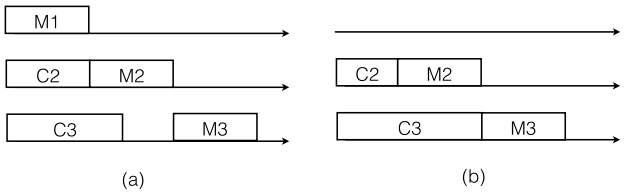

Fig. 5. It is optimal to use 2 devices as in (b) rather than 3 devices as in (a).

communication time should be more likely to have its communication scheduled earlier. Indeed, these two properties are satisfied by the optimal communication order which is according to the ratio r_i/d_i. This general result also degenerates to those in the special cases where devices have homogeneous computation rates or communication times. However, the proof for that the optimal communication order is exactly according to the form of the ratio between r_i and d_i is non-trivial. The key idea of the proof is a switching argument (described in the appendix).

D. Comparison of DECO and Cloud Offloading and Non-Offloading

In this subsection, we provide some theoretical analysis to compare DECO with cloud offloading and non-offloading. For ease of exposition, consider a simplified case of our proposed DECO scheme, where a total workload of w is divided and offloaded from an end device to N edge devices. Suppose the end device and all the edge devices have the same computation rate r, and all the edge devices have the same communication time d with the end device. Let d_0 and r_0 be the cloud's communication time with the end device and the cloud's computation rate, respectively, with $d_0 > d$ and $r_0 > r$. We obtain that, under the optimal workload allocation and the optimal communication order, the optimal total completion time for DECO is $2Nd+\max\{w/(Nr)-(N-1)d, 0\}$. We also obtain that the total completion time is $2d_0 + w/r_0$ for cloud offloading, and is w/r for non-offloading. Then we can see that, given N, if $Nd < d_0$ and $d_0 - Nd$ is large enough, and $Nr < r_0$ and $r_0 - Nr$ is small enough, and w is small enough, then DECO outperforms cloud offloading. otherwise, if $d_0 < Nd$ and $r_0 > Nr$, then cloud offloading outperforms DECO. We can also see that, given N, if d is small enough and w is large enough (or r is small enough), then DECO outperforms non-offloading; otherwise, non-offloading outperforms DECO.

VI. Optimal Device Selection

In the previous section, it is assumed that all edge devices participate in the computation offloading (i.e., perform offloaded computation), so that each device performs forward and backward communications regardless of the computation workload allocated to that device (e.g., even when no workload is allocated). However, it is important and interesting to note that it may not be optimal to use as many devices as possible to perform offloaded computation. This is because using an additional device incurs extra communication times which can increase the total completion time. This increase can outweigh the decrease in the total completion time of computations due to utilizing the additional device for computing, as illustrated in Fig. 5. Thus motivated, in this section, we investigate how to select participating devices of computation offloading to minimize the total completion time. The optimal device selection problem is given by

$$\min_{\mathcal{S}\subset\mathcal{N}} \; g(\mathcal{S}) \tag{1}$$

where $g(\mathcal{S})$ is the optimal total completion time under the optimal computation workload allocation and optimal communication order given in Section V when the set of participating devices is $\mathcal{S}$, which is given by

$$g(\mathcal{S})=\frac{\max\{w-\sum_{i=1}^{|\mathcal{S}|-1} r_{o_i^f}(\sum_{j=i+1}^{|\mathcal{S}|} s_{o_j^f})-\sum_{i=2}^{|\mathcal{S}|} r_{o_i^b}(\sum_{j=1}^{i-1} d_{o_j^b}),0\}}{\sum_{i=1}^{|\mathcal{S}|} r_i} + \sum_{i=1}^{|\mathcal{S}|} s_i + \sum_{i=1}^{|\mathcal{S}|} d_i$$

where $\{o_i^f\}$ and $\{o_i^b\}$ are the optimal communication orders of the FMs and BMs, respectively, for the devices in $\mathcal{S}$ according to Theorem 2.

We first consider the cases when devices have uniform computation rates or uniform communication times. For these two cases, we can show that the optimal set of devices to select are those with the smallest communication times or the highest computation rates.

Lemma 4: When devices have uniform computation rates (or uniform communications times), if a device is in the optimal set of participating devices, then each device with a smaller communication time (or higher computation rate, respectively) is also in the optimal set.

Lemma 4 is intuitive as a device with a smaller communication time or higher computation rate should be preferred over one with a larger communication time or lower computation rate, respectively. Based on this result, we can use an efficient exhaustive search to find the optimal set of devices. In particular, we can calculate the minimum total offloading delay for all possible numbers of the "best" devices (i.e., the k devices with the smallest communication times or highest computation rates where $k \in \{1, 2, \cdots, N\}$), and then find the optimal number of the "best" devices. The computational complexity of this linear exhaustive search is $O(N)$.

Then we consider the general case where devices can have diverse computation rates and also diverse communication times. To determine the optimal set of devices, we may use an exhaustive search that calculates the minimum offloading delay for all possible sets of devices (using the optimal computation allocation given in Algorithm 1 and the optimal communication order given in Theorem 2), and then finds the optimal set among them. However, the computational complexity of the exhaustive search is $O(2^N)$ which can be too high. Therefore, we use a greedy algorithm instead as follows: we start with the empty set, and in each iteration we add to this set the device not selected that can reduce the offloading delay the most, until no such device exists. The computational complexity of this greedy algorithm is $O(N)$.

Next we observe the following important property of the device selection problem.

Lemma 5: The total completion time $g(\mathcal{S})$ is a non-negative, non-monotonic submodular function of the set of participating devices $\mathcal{S}$.

The proof of Lemma 5 is non-trivial due to the complex dependence of $g(\mathcal{S})$ on the set of participating devices $\mathcal{S}$. Lemma 5 provides the insight that the marginal decrease of the total completion time $g(\mathcal{S})$ by adding more participating

devices is diminishing. Due to the submodular property of $g(\mathcal{S})$ given in Lemma 5, we immediately have the result below (the proof is omitted as it follows directly from Lemma 5).

Theorem 3: The optimal device selection problem (1) is a submodular minimization problem.

Since problem (1) is a submodular minimization problem, it can be equivalently transformed into a convex optimization problem via the Lovasz extension. Then this convex problem can be solved efficiently in polynomial time using some existing methods (such as the ellipsoid method), and this solution can be easily converted to the solution to the original problem (1).

VII. Optimal Computation Workload Allocation Under Maximum Workload Constraints

In the previous two sections, it is assumed that any computation workload can be allocated to an edge device. However, a device's computation resource may be limited, e.g., the device can only perform computation up to some time or some amount of energy consumption. In this case, each device has a maximum computation workload. In this section, based on the framework and problem formulation in Section III, we study the optimal computation workload allocation and communication scheduling when devices' computation workload is upper bounded.

First, it can be shown that the structural properties of the optimal communication scheduling without the workload constraints, which are time-sharing, non-preemptive, non-idle, and scheduling all forward communications before all backward communications, still hold in the case with the workload constraints (following similar proofs for Theorem 1 and Lemmas 1, 2, and 3). For ease of exposition, we assume that each device's forward communication time and backward communication time are the same. We also assume that the total maximum computation workload of devices (i.e., $\sum_{i\in\mathcal{N}} l_i$) is larger than the total workload to be completed W.

Under the maximum workload constraints, the optimal workload allocation can be found by an efficient algorithm as described in Algorithm 2. Similar to Algorithm 1, Algorithm 2 first allocates computation workloads as much as possible to devices such that they can be completed before the last FM ends and after the first BM starts, under the maximum workload constraint of each device. If the total workload W cannot be fully allocated in this way, it allocates the remaining workload to devices such that it is completed after the last FM ends and before the first BM starts. In particular, it iteratively allocates the workload to all the devices that have not reached their maximum workload constraints, until the constraint is reached for some device, or the total workload W is fully allocated. We can see that the computational complexity of Algorithm 2 is $O(N)$. The optimality of Algorithm 2 is given below.

Proposition 4: For any given communication order, Algorithm 2 finds the optimal computation workload allocation under the maximum workload constraints of devices.

Similar to Proposition 1, Proposition 4 provides some insights on the optimal workload allocation given by Algorithm 2: the optimal policy should reduce the idle computing periods of devices as much as possible, under the maximum workload constraints.

In general, under the maximum workload constraints, the optimal communication order is difficult to find.

Algorithm 2 Optimal Computation Workload Allocation Under Maximum Workload Constraints

input: computation rates $\{r_i\}$, FM order $\{o_i^f\}$, BM order $\{o_i^b\}$, maximum workload constraints $\{l_i\}$, total computation workload w;
for $i = 1, \cdots, N$ **do**
  $w_i = \min\{r_k(\sum_{i=o_i^f+1}^{N} d_i + \sum_{i=1}^{o_i^b-1} d_i),\ l_k\}$;
end
$w' = \sum_{i=1}^{N} w_i$;
if $w' \le w$ **then**
  for $i = 1, \cdots, N$ **do**
    $w_i = w_i + r_i(w - w')/(\sum_{s\in\mathcal{N}} r_s)$;
  end
else
  while $W > w$ **do**
    $\mathcal{N}' \leftarrow$ the set of devices with $l_i > w_i$
    $j \leftarrow i$ meets $\arg\min_{i\in\mathcal{N}'}(l_i - w_i)/r_i$;
    if $W - w > (\sum_{s\in\mathcal{N}'} r_s)(l_j - w_j)/r_j$ **then**
      for $i \in \mathcal{N}'$ **do**
        $w_i = w_i + (l_j - w_j)r_i/r_j$;
      end
      $w = \sum_{i\in\mathcal{N}} w_i$;
    end
    else
      for $i \in \mathcal{N}'$ **do**
        $w_i = w_i + r_i(W - w)/(\sum_{s\in\mathcal{N}'} r_s)$;
      end
    end
  end
end
output: optimal workload allocation $\{w_i\}$.

VIII. Simulation Results

In this section, we evaluate the performance of the proposed DECO approach via extensive simulations. The simulation results consist of two parts. In the first part, to demonstrate the performance of DECO in practice, we evaluate DECO for a realistic application in practical settings, by comparing its performance with that of two benchmarks as below.

- Non-offloading: The end device performs all computations without offloading any computation to the edge devices.
- Cloud offloading: The end device offloads all computations to a cloud server which performs all computations.

In the second part, to demonstrate the performance of the schemes and algorithms proposed in this paper for DECO, we evaluate various design components of DECO (i.e., time-sharing based communication resource allocation, computation workload allocation, communication order, device selection) in controlled settings.

A. Comparing DECO With Non-Offloading and Cloud Offloading

In this subsection, we compare the performance of DECO with the benchmarks of non-offloading and cloud offloading in

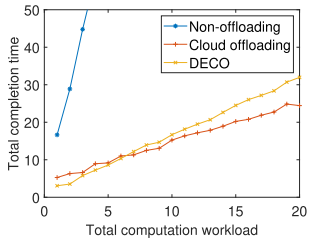

Fig. 6. DECO vs benchmarks: impact of total computation workload.

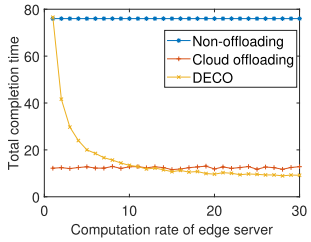

Fig. 7. DECO vs benchmarks: impact of edge computation rate.

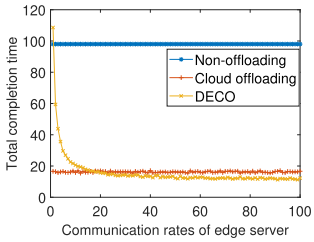

Fig. 8. DECO vs benchmarks: impact of edge communication rate.

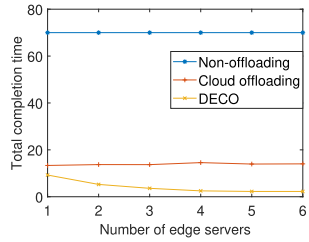

Fig. 9. DECO vs benchmarks: impact of edge device number.

practice. We use the optimal workload allocation and optimal communication order for DECO. We evaluate the impacts of various system parameters on the comparison between DECO and the benchmarks.

We consider a realistic application in practical settings described as below. We consider the image classification task using the residual neural network (ResNet) [48] based on the CIFAR-10 [49] datasets. Some system parameters are set as follows by default: the computation rates of the end device, edge devices, and the cloud are set to 1.36 trillion FLOPS, 10 trillion FLOPS, and 20 trillion FLOPS, respectively; the end-to-edge communication rate and the end-to-cloud communication rate are set as 100 Mbps and 40 Mbps, respectively. The CIFAR-10 datasets consist of 10 classes of natural images with a total of 50K training images and a total of 10K testing images, with each image of size 32 x 32. Running the ResNet with 34 parameter layers for one image requires about $3.4 - 3.6$ billion FLOPs (we follow the same pre-processing of global contrast normalization and Zero component analysis (ZCA) whitening as [50]). We set 1 computation workload unit as 4K image classification tasks. After offloading tasks to the edge/cloud, the edge/cloud sends the results of classification back to the client, and the size of the results is randomly distributed from $1 - 3$ MB [9].

1) Impact of Total Computation Workload: Fig. 6 shows the performance of DECO and the benchmarks as the total computation workload varies. We see that DECO always outperforms non-offloading, and outperforms cloud offloading when the workload is less than 5. This is because the cloud has more computation power than the edge, and thus is the better choice when the computation workload is large.

2) Impact of Edge Computation Rate: We set the end-to-edge communication rate as 50 Mbps, the end-to-cloud communication rate as 30 Mbps, and the total workload as 5. Fig. 7 shows that the total completion time of DECO decreases as the edge computation rate increases, which is because less computation time is needed. We see that DECO is better than non-offloading and cloud offloading, when the edge computation rate is larger than 15.

3) Impact of Edge Communication Rate: We set the end-to-edge communication rate as 30 Mbps, and the total workload as 7. Fig. 8 shows that the total completion time of DECO decreases as the edge communication rate increases, which is because less communication time is needed. We see that DECO outperforms cloud offloading and non-offloading when the edge communication rate is larger than 20.

4) Impact of Edge Device Number: We set the end-to-cloud communication rate as 30 Mbps, and the total workload as 7. Fig. 9 shows that DECO always outperforms cloud offloading and non-offloading. We also see that the total completion time of DECO decreases as the number of edge devices increases from 1 to 4. However, when there are too many edge devices (i.e., 5 or 6 edge devices), the total completion time no longer decreases. This is because too many edge devices takes too much time for communications, which cannot be compensated from fast computations.

B. Evaluating Design Components of DECO

In this subsection, we evaluate the impacts of various design components (i.e., time-sharing based communication resource allocation, computation workload allocation, communication order, and device selection) on the performance of DECO using simulation results. To thoroughly demonstrate the performance, we choose various settings for system parameters.We set the default values of system parameters as follows: $N = 3$, $w = 10$, $s_i = 1$, $d_i = 1$, $r_i = 1$, $\forall i$. For the case of diverse $\{s_i\}$ and $\{d_i\}$ or diverse $\{r_i\}$, they are set by sampling from a uniform distribution over $[0, 2]$. The maximum workload constraints $\{l_i\}$ are set by sampling from a uniform distribution over $[0, 4]$.

1) Time-Sharing Based Communication Scheduling: To show the optimal communication schedule, we compare the total completion time of the bandwidth-sharing based scheme with the time-sharing based scheme.

Fig. 10 illustrates the total delay as the total computation workload varies, where the bandwidth-sharing scheme evenly allocates the total bandwidth to 3 devices. We note that the time-sharing scheme significantly and steadily outperforms the bandwidth-sharing scheme. Fig. 11 illustrates the comparison when the number of devices changes. From Fig. 11, we can see that the total completion time of the time-sharing scheme

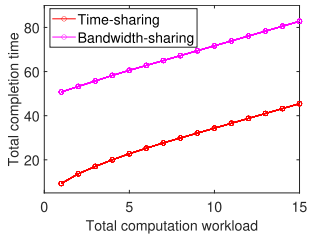

Fig. 10. Time-sharing vs bandwidth-sharing based communication scheduling.

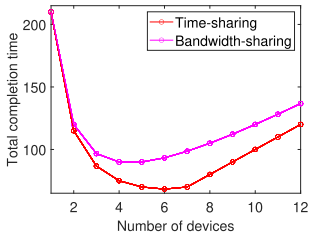

Fig. 11. Time-sharing vs bandwidth-sharing based communication scheduling.

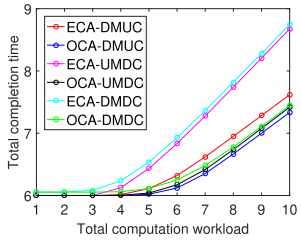

Fig. 12. Impact of computation workload allocation.

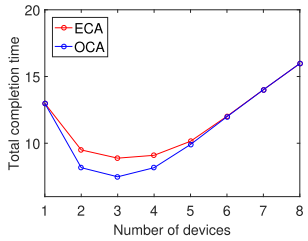

Fig. 13. Impact of computation workload allocation.

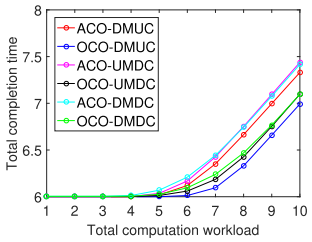

Fig. 14. Impact of communication order.

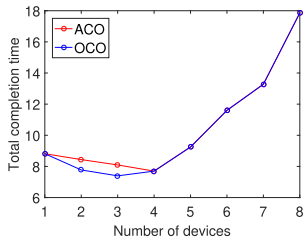

Fig. 15. Impact of communication order.

is always no greater than the bandwidth-sharing scheme. In addition, we observe that the total delay can increase for both of the schemes when the number of devices is too large. This implies that it is not optimal to select as many participating devices, since the bottleneck of reducing the total delay can change from computation times to communication times.

2) Computation Workload Allocation: To illustrate the efficiency of the optimal computation allocation (OCA), we compare the total delay under OCA found by Algorithm 1, and under equal computation allocation (ECA) that allocates equal workloads w/N to all devices.

Fig. 12 illustrates the total delay under OCA and under ECA, when the total computation workload w varies. We can see that the delay under OCA is always no greater than that under ECA, which demonstrates the better performance of OCA. We can also see that the delay is non-decreasing with w, which is because a larger workload takes more time to complete. We note that the performance gap is 0 when w is small. This is because in this case, all the workloads can be completed before the last FM ends or after the first BM starts, such that the delay is equal to the total time of FMs and BMs, which is the same for both allocations. We further observe that the performance gain of OCA compared to ECA for diverse communication times and computation rates (DMDC) is more than that for uniform communication times and diverse computation rates (UMDC), or for diverse communication times and uniform computation rates (DMUC). This is because when communication times or computation rates are diverse rather than uniform, OCA is more different from ECA, so that OCA is more beneficial. Fig. 13 illustrates the total delay when the number of devices varies. We note that OCA always outperforms ECA. We also note that when more than 1 device (but not too many devices) are used, the total delay is substantially smaller than when only 1 device is used, which is the conventional edge computation offloading strategy.

3) Communication Order: To illustrate the efficiency of the optimal communication order (OCO), we compare the total delay under OCO given by Theorem 2, and under an arbitrary communication order (ACO) that schedules communications in the ascending order of devices' indices.

Fig. 14 illustrates the total delay under OCO and under ACO, as the the total computation workload w varies. As expected, we can see that OCO always outperforms ACO, and the delay is non-decreasing with w. Similar to Fig. 12, we note that the performance gap is 0 when w is small, which is because in this case the delay is equal to the total time of communications, which is the same for both scheduling orders. We can also observe that the performance gain of OCO compared to ACO for DMDC is more than that for UMDC or DMUC. Similar to Fig. 12, the reason is that when communication times or computation rates are diverse rather than uniform, OCO is more different from ECA and thus is more beneficial. Fig. 15 illustrates the total delay when the number of devices varies. We note that OCO always outperforms ACO. Fig. 16 illustrates the total delay under the optimal computation allocation (OCA) or/and the optimal communication order (OCO), as the total workload increases. We note that OCA+OCO always outperforms OCA only, which demonstrates that OCO is beneficial when OCA is used.

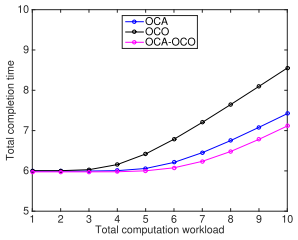

Fig. 16. Impact of communication order.

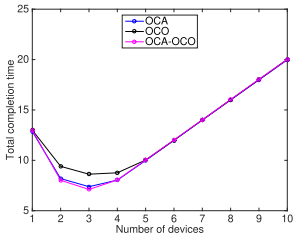

Fig. 17. Impact of device selection.

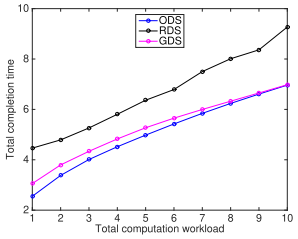

Fig. 18. Impact of device selection.

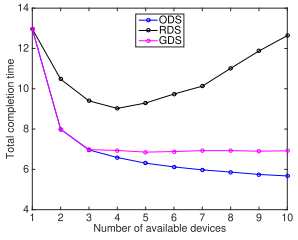

Fig. 19. Impact of device selection.

4) Device Selection: To illustrate the impact of the selection of participating devices, we compare the total delay as the number of selected devices varies. Fig. 17 illustrates the comparison as the number of selected devices N varies. We can see that the delay first decreases and then increases with N. This is because the delay reduction due to the computation workload completed by an additional device first outweighs the delay increase due to the communications of that device, and then the former effect is dominated by the second effect. We can also observe that the delay under OCA-OCO is better than under OCA or OCO only.

To illustrate the performance of the optimal device selection (ODS), we compare its performance with that of random device selection (RDS) and greedy device selection (GDS), when the optimal computation workload allocation and optimal communication order are used. As the benchmark, RDS randomly selects the set of participating devices from all available devices, while GDS greedily adds a device from available devices to the set of participating devices, until the total completion time does not decrease. Both Figs. 18 and 19 show that GDS outperforms RDS, and ODS outperforms both GDS and RDS, which demonstrates the benefits of the proposed ODS. Moreover, Fig. 19 shows that the total delay of ODS is always decreasing as the number of available devices increases, while the total delay of RDS and GDS saturates or even becomes worse when the number of available devices is large.

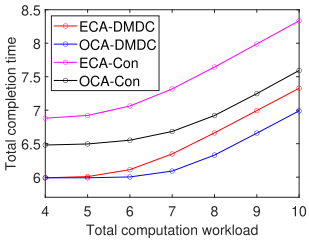

Fig. 20. Impact of computation allocation under maximum constraints.

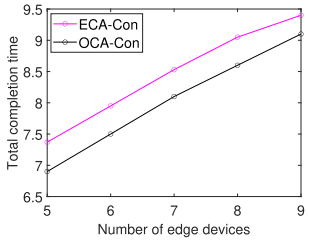

Fig. 21. Impact of computation allocation under maximum constraints.

5) Impacts of Maximum Workload Constraints: To illustrate the efficiency of the optimal computation allocation under devices' maximum workload constraints (OCA-Con), we compare the total delay under OCA-Con given by Algorithm 2 with the equal computation allocation under the maximum workload constraints (ECA-Con) that allocates equal workloads to all devices. The total workload is set to 6 and the maximum workload constraint is set to 1.2.

Fig. 20 illustrates the total delay as the total computation workload varies. We note that OCA and ECA are always better than OCA-Con and ECA-Con, respectively, which is because of the maximum workload constraints. Moreover, OCA-Con always outperforms ECA-Con, which demonstrates the benefit of the OCA-Con. Fig. 21 illustrates the total delay as the number of devices varies. Note that the total workload can be completed by 5 or more devices under the maximum workload constraints. We see that OCA-Con always outperforms ECA-Con. This is because ECA-Con must perform computation after the last FM ends and before the first BM starts, while OCA-Con does not need to do so which reduces the total delay.

IX. Conclusion and Future Work

In this paper, we have explored DECO by studying the minimization of the total completion time of computation offloading using distributed edge devices connected by a wireless network. In particular, we have shown the benefit of time-sharing based communication resource allocation, and characterized the optimal communication scheduling, the optimal computation allocation, and the optimal selection of participating devices for minimizing the total completion time. The optimal policies have been developed by addressing the non-trivial coupling between these issues, while taking into

account the features of wireless networks. The results have provided useful insights into the optimal policies.

For future work, one immediate direction is to investigate the case where communication and computation delays are unknown and stochastic. Another important setting is when the computation workload of the offloaded job is not arbitrarily divisible. We will also study the setting when the workloads of communications depend the corresponding workloads of computations.

APPENDIX

Proof of Theorem 1

Given any computation schedule, consider any feasible bandwidth-sharing based communication schedule. We show that for any pair of two devices' FMs or BMs, we can construct a new feasible time-sharing based schedule for this pair of two communications such that the total completion time is not increased. We consider three cases as follows.

Case 1) One FM and one BM. Suppose WLOG that device i's FM FM_i and device j's BM BM_j are considered. If FM_i ends at t_i^e no later than when BM_j starts at t_j^s (i.e., $t_i^e \leq t_j^s$), then they are already based on time-sharing. Now suppose FM_i ends after BM_j starts (i.e., $t_j^s < t_i^e$). Since function $f(t) \triangleq \int_{t_j^s}^{t} b_j(t)dt - \int_{t}^{t_i^e} b_i(t)dt$ is continuous and non-decreasing in t and satisfies $f(t_j^s) = -\int_{t_j^s}^{t_i^e} b_i(t)dt < 0$ and $f(t_i^e) = \int_{t_j^s}^{t_i^e} b_j(t)dt > 0$, there must exist some t_0 with $t_j^s \leq t_0 \leq t_i^e$ such that $f(t_0) = \int_{t_j^s}^{t_0} b_j(t)dt - \int_{t_0}^{t_i^e} b_i(t)dt = 0$. Therefore, the total amount of bandwidth-time unit of BM_j before t_0 (given by $\int_{t_j^s}^{t_0} b_j(t)dt$) is equal to that of FM_i after t_0 (given by $\int_{t_0}^{t_i^e} b_i(t)dt$). Then we can construct a new schedule of FM_i and BM_j by reallocating the bandwidth of BM_j for each time t before t_0 to FM_i (i.e., $b_i'(t) = b_i(t) + b_j(t)$, $b_j'(t) = 0$, $\forall t \leq t_0$) and reallocating the bandwidth of FM_i for each time t after t_0 to BM_j (i.e., $b_j'(t) = b_j(t) + b_i(t)$, $\forall t \geq t_0$). Since $\int_{t_j^s}^{t_0} b_j(t)dt = \int_{t_0}^{t_i^e} b_i(t)dt$, the total communication workload of each of FM_i and BM_j in the new schedule remains the same. Moreover, as FM_i in the new schedule ends earlier than in the old schedule while BM_j in the new schedule starts later than in the old schedule, they are compatible with the computation schedule of devices i and j. Furthermore, since the new schedule only reallocates bandwidths from BM_j to FM_i, FM_i and BM_j in the new schedule are compatible with the communications of other devices. Therefore, the new schedule is feasible. Since the new FM_i ends at t_0 and the new BM_j starts at t_0, the new schedule is also based on time-sharing.

Case 2) Two BMs. Suppose WLOG that device i's BM BM_i starts before device j's BM BM_j starts. If BM_i ends no later than when BM_j starts, then they are already based on time-sharing. Now suppose BM_i ends at t_i^e after BM_j starts at t_j^s (i.e., $t_j^s < t_i^e$). Using a similar argument as in Case 1, there must exist some t_0 with $t_j^s \leq t_0 \leq t_i^e$ such that $\int_{t_j^s}^{t_0} b_j(t)dt - \int_{t_0}^{t_i^e} b_i(t)dt = 0$. Then we can construct a new schedule of BM_i and BM_j by reallocating the bandwidth of BM_j for each time t before t_0 to BM_i (i.e., $b_i'(t) = b_i(t) + b_j(t)$, $b_j'(t) = 0$, $\forall t \leq t_0$) and reallocating the bandwidth of BM_i for each time t after t_0 to BM_j (i.e., $b_j'(t) = b_j(t) + b_i(t)$, $\forall t \geq t_0$), where $b_i'(t)$ is device i's bandwidth at time t in the new schedule. Using similar arguments as in Case 1, we can show that the new schedule is feasible and also based on time-sharing.

Case 3) Two FMs. We can use similar arguments as in Case 2 to prove this case. The difference is that we use the arguments in a reverse manner in time. Specifically, we can construct a new schedule of FM_i and FM_j (where FM_i ends after FM_j starts and before FM_j ends) by reallocating the bandwidth of FM_i for each time t after some time t_0 to FM_j and reallocating the bandwidth of FM_j for each time t before t_0 to FM_i. We can show that the new schedule is feasible and also based on time-sharing.

Next we show that we can use the construction arguments in the above three cases iteratively to eventually find a new schedule of all devices' forward and BMs which is feasible and based on time-sharing. It consists of the following three steps.

Step 1. We first apply the argument in Case 1 iteratively to each pair of one device's FM and another device's BM, in an arbitrary order. For example, we can apply it first for (FM_1, BM_2), and then (FM_1, BM_3), ...and then (FM_1, BM_N), and then (FM_2, BM_3), ..., until (FM_N, BM_{N-1}). We note that each use of the argument results in the new FM ending before the old FM ends and also before the new BM starts. Therefore, it can be seen that after the sequence of arguments, each FM ends before each BM starts.

Step 2. Based on the schedule obtained in Step 1, we then apply the argument in Case 2 iteratively to each pair of two devices' BMs in a certain order. In particular, we first find the BM that starts the earliest among all BMs (say, BM_1). Then we apply the argument for (BM_1, BM_2), and then (BM_1, BM_3), ..., until (BM_1, BM_N). We note that each use of the argument results in the new BM_1 starting at the same time as the old BM_1, while ending before the old BM_1 ends and also before the new BM starts. Thus, after this sequence of arguments, BM_1 remains the earliest to start among all BMs, while it is based on time-sharing with respect to each other BM. Next, based on the new schedule, we find the BM that starts the second earliest among all the BMs (say, BM_2). Then we apply the argument for (BM_2, BM_3), and then (BM_2, BM_4), ..., until (BM_2, BM_N). Following a similar argument as above, BM_2 remains the second earliest to start, while it is based on time-sharing with respect to each other BM. We continue this argument for the remaining BMs. Eventually, we obtain a feasible schedule of all BMs which are based on time-sharing with respect to each other.

Step 3. Based on the schedule obtained in Step 2, we then can use similar arguments as in Step 2 for all FMs, and thus can obtain a feasible schedule of all FMs which are based on time-sharing with respect to each other. This completes the proof.

Proof of Lemma 1

The main idea of the proof is an exchange argument. WLOG, suppose FM M1 is interrupted by FM M2 into two parts, such that the 2nd part of M1 starts after M2 (or part of M2) is completed, as illustrated in Fig. 22 (a). If M1 is interrupted for multiple times, the proof follows by applying the exchange argument multiple times. Now we exchange the scheduling order of the 1st part of M1 and M2 (or the interrupting part of M2), as illustrated in Fig. 22 (b). We can see that the order exchange does not affect the schedules of

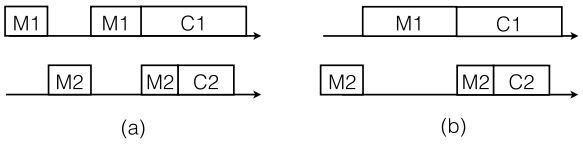

Fig. 22. An exchange argument for the proof of Lemma 1.

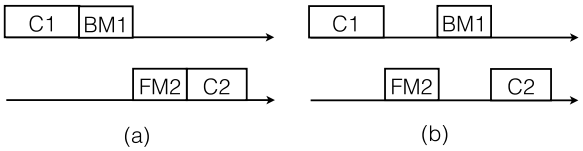

Fig. 23. An exchange argument for the proof of Lemma 2.

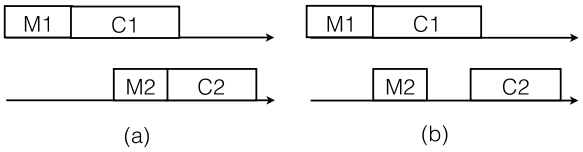

Fig. 24. A shifting argument for the proof of Lemma 3.

computations C1 and C2, as well as the schedule of any communication or computation on the devices other than devices 1 and 2. As a result, the delay of the algorithm remain the same. This completes the proof.

Proof of Lemma 2

The main idea of the proof is an exchange argument. According to Lemma 1, it suffices to focus on non-preemptive scheduling. WLOG, suppose FM FM1 is scheduled after BM BM2, as illustrated in Fig. 23 (a). Now we exchange the scheduling order of FM1 and BM2, as illustrated in Fig. 23 (b). We can see that the order exchange does not affect the schedules of computations C1 and C2, as well as the schedule of any communication or computation on the devices other than devices 1 and 2. As a result, the delay of the algorithm remain the same. This completes the proof.

Proof of Lemma 3

The main idea of the proof is a shifting argument. According to Lemma 1 and Lemma 2, it suffices to focus on non-preemptive scheduling policies that schedule all FMs before all BMs. WLOG, suppose there is an idle period of the wireless network between FM M1 and FM M2, as illustrated in Fig. 24 (a). If there are multiple idle periods, the proof follows by applying the exchange argument multiple times. Now we shift M2 to be right after M1, as illustrated in Fig. 24 (b). We can see that the shifting does not affect the schedules of computations C1 and C2, as well as the schedule of any communication or computation on the devices other than devices 1 and 2. As a result, the delay of the algorithm remain the same. If M1 and M2 are BMs, we can shift M1 to be right before M2, and the same argument applies. This completes the proof.

Proof of Theorem 2

Based on problem, it suffices to show that the total computation workload scheduled after the first BM starts is maximized when BMs are scheduled in the non-increasing order of the ratio d_i/r_i. Consider any order of devices $\mathcal{O} \triangleq (o_1, o_2, \cdots, o_N)$. Then the total computation workload scheduled after the first BM starts under order $\mathcal{O}$ is given by

$$f(\mathcal{O}) = \sum_{i=1}^{N-1} d_{o_i} \sum_{j=i+1}^{N} r_{o_j}$$

Suppose the order $\mathcal{O}$ is not in the non-increasing order of d_i/r_i. Then there must exist a pair of two neighbor devices o_i and o_{i+1} in order $\mathcal{O}$ with $d_{o_i}/r_{o_i} < d_{o_{i+1}}/r_{o_{i+1}}$. Then consider a new order $\mathcal{O}' = (o_1, \cdots, o_{i+1}, o_i, \cdots, o_N)$ obtained from order $\mathcal{O}$ by swapping the positions of o_i and o_{i+1}. Then we have that the total computation workload scheduled after the first BM starts under order $\mathcal{O}'$ is given by

$$f(\mathcal{O}') = d_{o_1} \sum_{j=2}^{N} r_{o_2} + \cdots + d_{o_{i+1}} (r_{o_i} + \sum_{j=i+2}^{N} r_{o_j}) + d_{o_i} \sum_{j=i+2}^{N} r_{o_j} + \cdots + d_{o_{N-1}} \sum_{j=N}^{N} r_{o_j}.$$

Since we have

$$f(\mathcal{O}') - f(\mathcal{O}) = d_{o_{i+1}} r_{o_i} - d_{o_i} r_{o_{i+1}} > 0$$

where the inequality follows from $d_{o_i}/r_{o_i} < d_{o_{i+1}}/r_{o_{i+1}}$, $\mathcal{O}'$ is a better order than $\mathcal{O}$ in terms of the total computation workload. Based on order $\mathcal{O}'$, we can use the argument above iteratively such that the total computation workload improves after each iteration, until there does not exist two neighbor devices o_i and o_{i+1} with $d_{o_i}/r_{o_i} < d_{o_{i+1}}/r_{o_{i+1}}$. This completes the proof.

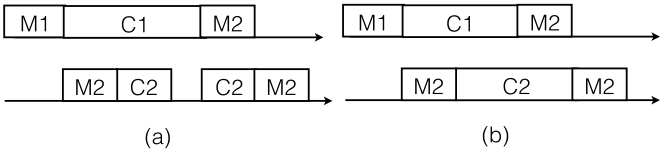

Fig. 25. A shifting argument for the proof of Proposition 1.

Proof of Proposition 1

The main idea of the proof is a shifting argument. We first note that a lower bound of the algorithm delay is the total delay D of all forward and BMs. Therefore, if Algorithm 1 terminates in Phase 1 or Phase 2, then it finds a feasible schedule of all the computation workloads of the algorithm, such that the algorithm delay is equal to D. Suppose the Algorithm 1 terminates in Phase 3, and the remaining total unallocated computation workloads is not allocated to the devices in proportional to their computation rates. Then there must be some device that is idle for some period after all FMs and before all BMs, as illustrated in Fig. 25 (a). In this case, we can always shift some workload from some other device to this device without increasing the algorithm delay, until there is no such idle period, as illustrated in Fig. 25 (b). This completes the proof.

Proof of Lemma 5

For ease of exposition, we ignore the computation workloads that can be completed before the last FM ends (i.e., before time t_1 in Fig. 3). Let $g'(S)$ be the time difference

between when the last FM ends and when the last BM ends (i.e., between time t_1 and time t_2 in Fig. 3), when the set of selected devices is S. Then it suffices to show that $g'(S)$ is a submodular function of S.

Let k_i be the ratio between device i's communication time and its computation rate, i.e., $k_i = d_i/r_i$. Let f be the total workload to be allocated to the selected devices. Consider two devices $i \neq k$ and assume WLOG that $k_i > k_k$. Let f_1, $f_1{}'$, f_2, and $f_2{}'$ be the total workload that can be allocated after the first BM starts (i.e., time t_2 in Fig. 3), when the set of selected devices is S, $S \cup \{i\}$, $S \cup \{k\}$, and $S \cup \{k, i\}$, respectively. Then we have

$$
\begin{aligned}
f_1 &= \sum_{j,l\in S, j\neq l} k_{jl} r_j r_l, \\
f_1{}' &= \sum_{j,l\in S, j\neq l} k_{jl} r_j r_l + \sum_{j\in S} k_{ij} r_i r_j, \\
f_2 &= \sum_{j,l\in S, j\neq l} k_{jl} r_j r_l + \sum_{j\in S} k_{kj} r_k r_j, \\
f_2{}' &= \sum_{j,l\in S, j\neq l} k_{jl} r_j r_l + \sum_{j\in S} k_{kj} r_k r_j + \sum_{j\in S} k_{ij} r_i r_j + k_{ik} r_i r_k,
\end{aligned}
$$

where $k_{ij} \triangleq \max\{k_i, k_j\}$, $\forall i, j$.

For ease of exposition, assume WLOG that $f \geq f_2{}'$. Note that we have

$$
g'(S) - g'(S \cup \{i\}) = \underbrace{\frac{f - f_1}{\sum_{j\in S} r_j} - \frac{f - f_1{}'}{\sum_{j\in S} r_j + r_i}}_{(a)} - d_i,
$$

$$
g'(S \cup \{k\}) - g'(S \cup \{k, i\}) = \underbrace{\frac{f - f_2}{\sum_{j\in S} r_j} - \frac{f - f_2{}'}{\sum_{j\in S} r_j + r_i}}_{(b)} - d_i.
$$

Then it suffices to show that $(a) \geq (b)$. Multiplying (a) and (b) with the four denominators in them, we have that

$$
\begin{aligned}
&(f - f_1)(\sum_{j\in S} r_j + r_i)(\sum_{j\in S} r_j + r_k)(\sum_{j\in S} r_j + r_i + r_k) \\
&\quad - (f - f_1{}')(\sum_{j\in S} r_j)(\sum_{j\in S} r_j + r_k)(\sum_{j\in S} r_j + r_i + r_k) \\
&= \underbrace{(f_1{}' - f_1)\sum_{j\in S} r_j (\sum_{j\in S} r_j + r_k)(\sum_{j\in S} r_j + r_i + r_k)}_{(c)} \\
&\quad + \underbrace{(f - f_1) r_i (\sum_{j\in S} r_j + r_k)(\sum_{j\in S} r_j + r_i + r_k)}_{(d)}
\end{aligned}
$$

and

$$
\begin{aligned}
&(f - f_2)(\sum_{j\in S} r_j)(\sum_{j\in S} r_j + r_i)(\sum_{j\in S} r_j + r_i + r_k) \\
&\quad - (f - f_2{}')(\sum_{j\in S} r_j)(\sum_{j\in S} r_j + r_i)(\sum_{j\in S} r_j + r_k) \\
&= \underbrace{(f_2{}' - f_2)(\sum_{j\in S} r_j + r_k)(\sum_{j\in S} r_j)(\sum_{j\in S} r_j + r_i)}_{(e)} \\
&\quad + \underbrace{(f - f_2) r_i (\sum_{j\in S} r_j)(\sum_{j\in S} r_j + r_i)}_{(f)}.
\end{aligned}
$$

Thus we have

$$
\begin{aligned}
(c) - (e) = &-k_{ik} r_i r_k (\sum_{j\in S} r_j)(\sum_{j\in S} r_j + r_k)(\sum_{j\in S} r_j + r_i) \\
&+ (\sum_{j\in S} r_j) k_{ij} r_i r_j (\sum_{j\in S} r_j)(\sum_{j\in S} r_j + r_k) r_k \quad (2)
\end{aligned}
$$

and

$$
\begin{aligned}
(d) - (f) = &(\sum_{j\in S} r_j) k_{kj} r_k r_j (\sum_{j\in S} r_j)(\sum_{j\in S} r_j + r_i) r_i \\
&+ \underbrace{(f - f_1) r_i (2\sum_{j\in S} r_j r_k + r_k r_i + {r_k}^2)}_{(g)}. \quad (3)
\end{aligned}
$$

Next we have

$$
\begin{aligned}
(2) + (3) = &\, k_i r_i r_k (\sum_{j\in S} r_j)(\sum_{j\in S} r_j)(\sum_{j\in S} r_j + r_k) \\
&+ k_k r_i r_k (\sum_{j\in S} r_j)(\sum_{j\in S} r_j)(\sum_{j\in S} r_j + r_i) \\
&- k_{ik} r_i r_k (\sum_{j\in S} r_j)(\sum_{j\in S} r_j + r_k)(\sum_{j\in S} r_j + r_i) + (g) \\
= &\, [k_i (\sum_{j\in S} r_j)(\sum_{j\in S} r_j + r_k) + k_k (\sum_{j\in S} r_j)(\sum_{j\in S} r_j + r_i) \\
&- k_{ik} (\sum_{j\in S} r_j + r_i)(\sum_{j\in S} r_j + r_k)] r_i r_k (\sum_{j\in S} r_j) + (g) \\
= &\, [k_k (\sum_{j\in S} r_j)(\sum_{j\in S} r_j + r_i) - k_i r_i (\sum_{j\in S} r_j + r_k)] \\
&r_i r_k (\sum_{j\in S} r_j) + (g) \quad (4)
\end{aligned}
$$

where the last equality follows from $k_{ik} = k_i$ since $k_i > k_k$. Since

$$
\begin{aligned}
f - f_1 &= (\sum_{j\in S} r_j) k_{ij} r_i r_j + (\sum_{j\in S} r_j) k_{kj} r_k r_j \\
&\quad + k_{ik} r_i r_k + (f - f_2') \\
&\geq (\sum_{j\in S} r_j) k_{ij} r_i r_j + (\sum_{j\in S} r_j) k_{kj} r_k r_j + k_{ik} r_i r_k,
\end{aligned}
$$

we have

$$
\begin{aligned}
(g) &\geq (\sum_{j\in S} r_j) k_{ij} r_i r_j + (\sum_{j\in S} r_j) k_{kj} r_k r_j + k_{ik} r_i r_k \\
&\quad r_i r_k (2\sum_{j\in S} r_j + r_i + r_k) \\
&\geq (2k_i \sum_{j\in S} r_j r_i + 2k_i r_k r_i) r_k r_i (\sum_{j\in S} r_j).
\end{aligned}
$$

Therefore we can see that (4)> 0. This completes the proof.

Proof of Proposition 4

When there is no time gap between the end of the last FM and the start of the first BM (i.e., $t_1 = t_2$ as illustrated in Fig 26), we say that there is no extra computation time.

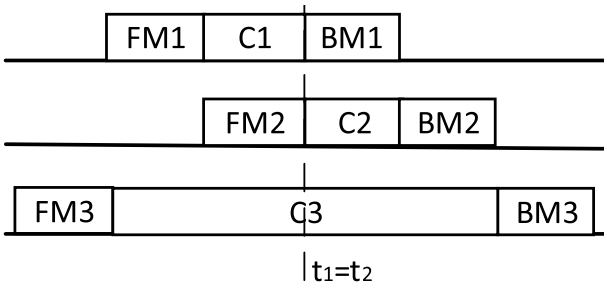

Fig. 26. There is no extra computation time between the end of last FM (FM2) and the start of the first BM (BM1).

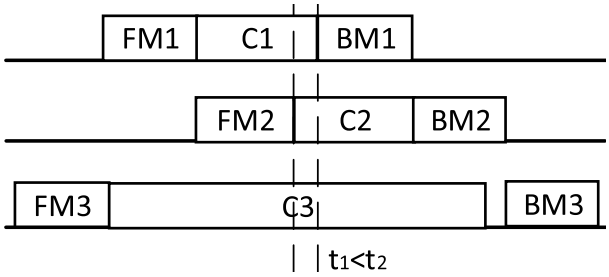

Fig. 27. There is an extra computation time between the end of last FM (FM2) and the start of the first BM (BM1).

Otherwise, we say that there is an extra computation time (i.e., $t_1 < t_2$ as illustrated in Fig 27).

From the above definition, we can find that for a given communication order, if the total workload can be completed during all devices' FM and BM times, then there is no extra computation time. This workload is referred to as the total maximum workload that can be completed without incurring an extra computation time under this communication order, as in Fig 26. Given any communication order, the total maximum workload without incurring an extra computation time is given by

$$w = \sum_{k=1}^{N} \Big[\min\{r_k(\sum_{i=o_i^f+1}^{N} d_i + \sum_{i=1}^{o_i^b-1} d_i),\ l_k\} \Big].$$

In particular, each device i's maximum workload allocation without incurring an extra computation time is given by

$$w_i = \min\{r_k(\sum_{i=o_i^f+1}^{N} d_i + \sum_{i=1}^{o_i^b-1} d_i),\ l_k\}.$$

Next, comparing the total workload to be completed W with the total maximum workload without incurring an extra computation time w, we consider two cases:

Case 1: $W \leq w$. In this case, all the workload can be fully allocated to devices without incurring an extra computation time. Thus each device i's optimal workload is

$$w = \max\{w_i + (W - w)/N,\ 0\}.$$

Case 2: $W > w$. In this case, we have to allocate the total workload to devices such that it incurs an extra computation time. In particular, we iteratively allocate the remaining workload $W - w$ to the devices that have not reached their maximum workload constraints, until the constraint is reached for some device, or the total workload is fully allocated. This procedure results in Algorithm 2, which completes the proof.

REFERENCES

[1] X. Gong, "Delay-optimal distributed edge computing in wireless edge networks," in *Proc. IEEE INFOCOM - Conf. Comput. Commun.*, Jul. 2020, pp. 2629–2638.

[2] *Google Glas*. Accessed: May 2022. [Online]. Available: https://x.company/glass/

[3] *Oculus*. Accessed: May 2022. [Online]. Available: https://www.oculus.com

[4] *Autonomous Flight*. Accessed: May 2022. [Online]. Available: https://autonomousflight.com

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

[6] C. Wu et al., "ButterFly: Mobile collaborative rendering over GPU workload migration," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9.

[7] W. Zhang, J. Chen, Y. Zhang, and D. Raychaudhuri, "Towards efficient edge cloud augmentation for virtual reality MMOGs," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, p. 8.

[8] L. Wang, L. Jiao, T. He, J. Li, and M. Mühlhäuser, "Service entity placement for social virtual reality applications in edge computing," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2018, pp. 468–476.

[9] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2018, pp. 756–764.

[10] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "DeepDecision: A mobile deep learning framework for edge video analytics," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Jun. 2018, pp. 1421–1429.

[11] L. Cheng and J. Wang, "ViTrack: Efficient tracking on the edge for commodity video surveillance systems," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Sep. 2018, pp. 1052–1060.

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

[13] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proc. IEEE INFOCOM 35th Annu. Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[14] H. Tan, Z. Han, X.-Y. Li, and F. C. M. Lau, "Online job dispatching and scheduling in edge-clouds," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9.

[15] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2019, pp. 1459–1467.

[16] P. Yang, N. Zhang, Y. Bi, L. Yu, and X. S. Shen, "Catalyzing cloud-fog interoperation in 5G wireless networks: An SDN approach," *IEEE Netw.*, vol. 31, no. 5, pp. 14–20, Sep. 2017.

[17] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5G systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 131–139, Feb. 2014.

[18] W. C. Ao and K. Psounis, "Distributed caching and small cell cooperation for fast content delivery," in *Proc. 16th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*. New York, NY, USA: ACM, Jun. 2015, pp. 127–136, doi: 10.1145/2746285.2746300.

[19] A. Liu and V. K. N. Lau, "Exploiting base station caching in MIMO cellular networks: Opportunistic cooperation for video streaming," *IEEE Trans. Signal Process.*, vol. 63, no. 1, pp. 57–69, Jan. 2015.

[20] K. W. Ross and D. D. Yao, "Optimal load balancing and scheduling in a distributed computer system," *J. ACM*, vol. 38, no. 3, pp. 676–689, Jul. 1991.

[21] H. Chang, M. Kodialam, R. R. Kompella, T. V. Lakshman, M. Lee, and S. Mukherjee, "Scheduling in mapreduce-like systems for fast completion time," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 3074–3082.

[22] Y. Zheng, N. B. Shroff, and P. Sinha, "A new analytical technique for designing provably efficient MapReduce schedulers," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 1600–1608.

[23] Y. Zhu et al., "Minimizing makespan and total completion time in MapReduce-like systems," in *Proc. IEEE INFOCOM - IEEE Conf. Comput. Commun.*, Apr. 2014, pp. 2166–2174.

[24] Y. Zheng, N. B. Shroff, R. Srikant, and P. Sinha, "Exploiting large system dynamics for designing simple data center schedulers," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 397–405.

[25] S. Im, M. Naghshnejad, and M. Singhal, "Scheduling jobs with non-uniform demands on multiple servers without interruption," in *Proc. IEEE INFOCOM 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[26] Y.-C. Cheng and T. G. Robertazzi, "Distributed computation for a tree network with communication delays," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 26, no. 3, pp. 511–516, May 1990.

[27] J. Tan, X. Meng, and L. Zhang, "Coupling task progress for MapReduce resource-aware scheduling," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 1618–1626.

[28] J. Jiang, S. Ma, B. Li, and B. Li, "Symbiosis: Network-aware task scheduling in data-parallel frameworks," in *Proc. IEEE INFOCOM 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[29] Q. Xie, A. Yekkehkhany, and Y. Lu, "Scheduling with multi-level data locality: Throughput and heavy-traffic optimality," in *Proc. IEEE INFOCOM 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[30] F. Chen, M. Kodialam, and T. V. Lakshman, "Joint scheduling of processing and shuffle phases in MapReduce systems," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 1143–1151.

[31] J. Liu, C. H. Xia, N. B. Shroff, and X. Zhang, "On distributed computation rate optimization for deploying cloud computing programming frameworks," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 4, pp. 63–72, Apr. 2013.

[32] J. Ghaderi, S. Shakkottai, and R. Srikant, "Scheduling storms and streams in the cloud," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 43, no. 1, pp. 439–440, Jun. 2015.

[33] M. Shafiee and J. Ghaderi, "A simple congestion-aware algorithm for load balancing in datacenter networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3670–3682, Dec. 2017.

[34] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 98–109, 2011.

[35] Y. Zhao et al., "Rapier: Integrating routing and scheduling for coflow-aware data center networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 424–432.

[36] Y. Li et al., "Efficient online coflow routing and scheduling," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Jul. 2016, pp. 161–170.

[37] A. Eryilmaz, R. Srikant, and J. R. Perkins, "Stable scheduling policies for fading wireless channels," *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 411–424, Apr. 2005.

[38] I.-H. Hou, V. Borkar, and P. R. Kumar, "A theory of QoS for wireless," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 486–494.

[39] L. Jiang and J. Walrand, "A distributed CSMA algorithm for throughput and utility maximization in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 3, pp. 960–972, Jun. 2010.

[40] X. Liu, E. K. P. Chong, and N. B. Shroff, "A framework for opportunistic scheduling in wireless networks," *Comput. Netw.*, vol. 41, no. 4, pp. 451–474, 2003.

[41] G. R. Gupta and N. B. Shroff, "Delay analysis and optimality of scheduling policies for multihop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 1, pp. 129–141, Feb. 2011.

[42] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Automatic Control.*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.

[43] X. Lin and S. B. Rasool, "Distributed and provably efficient algorithms for joint channel-assignment, scheduling, and routing in multichannel ad hoc wireless networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 6, pp. 1874–1887, Dec. 2009.

[44] D. Xue and E. Ekici, "Delay-guaranteed cross-layer scheduling in multihop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 6, pp. 1696–1707, Dec. 2013.

[45] Y. Yi and M. Chiang, "Stochastic network utility maximisation—A tribute to Kelly's paper published in this journal a decade ago," *Eur. Trans. Telecommun.*, vol. 19, no. 4, pp. 421–442, Jun. 2008.

[46] A. Munshi, D. Ginsburg, and D. Shreiner, *OpenGL ES 2.0 Programming Guide*. London, U.K.: Pearson, 2008.

[47] M. Pinedo, *Scheduling*, vol. 29. Cham, Switzerland: Springer, 2012.

[48] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016.

[49] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[50] M. Lin, Q. Chen, and S. Yan, "Network in network," Nat. Univ. Singapore, Singapore, Tech. Rep., 2014.

Xiaowen Gong (Member, IEEE) received the Ph.D. degree in electrical engineering from Arizona State University (ASU) in 2015. From 2015 to 2016, he was a Post-Doctoral Researcher with the Department of ECE, The Ohio State University. He is currently an Associate Professor with the Department of Electrical and Computer Engineering (ECE), Auburn University. His research interests include wireless networks and their applications, with current focuses on machine learning, AI in wireless networks, and edge computing. He received the IEEE Internet of Things Journal Best Paper Runner-Up Award in 2022 as the coauthor, the IEEE INFOCOM 2014 Runner-Up Best Paper Award as the coauthor, the ASU ECEE Palais Outstanding Doctoral Student Award in 2015, and the NSF CAREER Award in 2022. He served as an Associate Editor for IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the Guest Editor for IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, and the Lead Guest Editor for IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY.

Mingyu Chen (Student Member, IEEE) received the bachelor's degree in electronics and information engineering from Huazhong University of Science and Technology, Hubei, China, in June 2021. He is currently with the Department of Electrical and Computer Engineering, Boston University, USA. His research interests include mobile edge computing and optimization.

Dongsheng Li (Student Member, IEEE) received the B.S. degree in electronic science and technology from Tongji University, Shanghai, China, in 2016, and the M.S. degree in communication engineering from Inner Mongolia University, Hohhot, China, in 2019. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Auburn University, USA. His main research interests include bilevel optimization, federated learning, and wireless communication scheduling.

Yang Cao (Member, IEEE) is currently an Associate Professor with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China. From 2011 to 2013, he was with the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ, USA, as a Visiting Scholar. He has coauthored 50 papers on refereed IEEE journals and conferences. His research interests include video transmission and edge/distributed computing. He was awarded the CHINACOM Best Paper Award in 2010 and the Microsoft Research Fellowship in 2011.