This is the peer reviewd version of the followng article:

Coded Computation Across Shared Heterogeneous Workers With Communication Delay / Sun, Y., Zhang, F., Zhao, J., Zhou, S., Niu, Z., Gunduz, D.. - In: IEEE TRANSACTIONS ON SIGNAL PROCESSING. - ISSN 1053-587X. - 70:(2022), pp. 3371-3385. [10.1109/TSP.2022.3185905]

Terms of use:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

24/10/2026 05:36

(Article begins on next page)

Coded Computation across Shared Heterogeneous Workers with Communication Delay

Yuxuan Sun, *Member, IEEE,* Fan Zhang, Junlin Zhao, *Member, IEEE,*
Sheng Zhou, *Member, IEEE,* Zhisheng Niu, *Fellow, IEEE,* Deniz Gündüz, *Senior Member, IEEE*

arXiv:2109.11246v1 [cs.DC] 23 Sep 2021

***Abstract*—Distributed computing enables large-scale computation tasks to be processed over multiple workers in parallel. However, the randomness of communication and computation delays across workers causes the straggler effect, which may degrade the performance. Coded computation helps to mitigate the straggler effect, but the amount of redundant load and their assignment to the workers should be carefully optimized. In this work, we consider a *multi-master heterogeneous-worker* distributed computing scenario, where multiple matrix multiplication tasks are encoded and allocated to workers for parallel computation. The goal is to minimize the communication plus computation delay of the slowest task. We propose worker assignment, resource allocation and load allocation algorithms under both dedicated and fractional worker assignment policies, where each worker can process the encoded tasks of either a single master or multiple masters, respectively. Then, the non-convex delay minimization problem is solved by employing the Markov's inequality-based approximation, Karush-Kuhn-Tucker conditions, and successive convex approximation methods. Through extensive simulations, we show that the proposed algorithms can reduce the task completion delay compared to the benchmarks, and observe that dedicated and fractional worker assignment policies have different scopes of applications.**

***Index Terms*—Coded computation, communication delay, Markov's inequality, convex optimization, successive convex approximation.**

I. Introduction

With the fast development of artificial intelligence technologies and the explosion of data, computation tasks for the training and inference of machine learning (ML) models are becoming increasingly complex and demanding, which are almost impossible to be realized on a single machine. Distributed computing frameworks have been developed to parallelize these computations [2], [3], where a centralized *master* node takes charge of task partition, data dissemination, and result collection, and distributed computing nodes, called *workers*, process partial computation tasks in parallel.

While parallel processing across multiple workers speeds up computation, the overall delay depends critically on the slowest worker. According to the experiments on the commercial Amazon elastic compute cloud (EC2) platform, some workers might experience much longer computation and communication delays than the average [4]–[6]. This fact is mainly due to the randomness of the system, e.g., time-varying stochastic workloads of the workers, or the traffic over the communication network connecting the workers to the master. Such randomness leads to the so-called *straggler effect*, which substantially increases the overall computation delay and becomes a major bottleneck in distributed computing.

The key idea to mitigate the straggler effect is to add redundancy to the computation tasks, so that the computation result does not rely on receiving results from all the workers. State-of-the-art approaches mainly include redundant scheduling of computation tasks [7]–[9], and various coding schemes [10], such as maximum distance separable (MDS) coding [4]–[6], [11]–[13], gradient coding [14]–[16], and polynomial coded computation [17], [18]. Among them, the easiest policy is to replicate each task to multiple workers upon its arrival, and the optimal number of replicas can be derived under exponential [7] or general service time distributions [8]. The orders of partitioned tasks at different workers are designed in [9], and the impact of redundancy on the task completion delay under different scheduling orders is characterized.

Compared with simple task replication, coding can further improve the efficiency of computation. MDS coding schemes under different system settings have been widely investigated for matrix multiplication, which is the most common type of computation task in the distributed computing system. With N homogeneous workers, it is proved in [4] that MDS coding can reduce the computation delay by $O(\log N)$ compared to uncoded computation. Considering that workers have heterogeneous computing capabilities, the load allocation algorithms are proposed in [5] and [12] for a single-task scenario, both with asymptotic optimality. Based on [5], an online, recursive load allocation algorithm is further proposed in [13] for the random task arrival case, where cancellation is enabled to clear the unfinished parts of each task upon its completion, so as to avoid unnecessary computations.

Although stragglers are slower than the average computation speed, it is still possible for them to provide partial results. This can be achieved by the hierarchical coded computation framework [6], or multi-message communications [18]–[21]. Specifically, in the hierarchical framework, the coded task at each worker is partitioned into multiple layers. Stragglers are able to finish the lower layer sub-tasks and thus the coding redundancy in the lower layers can be reduced to

Y. Sun, F. Zhang, S. Zhou and Z. Niu are with the Beijing National Research Center for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (e-mail: sunyuxuan@tsinghua.edu.cn, zhang-f17@tsinghua.org.cn, sheng.zhou@tsinghua.edu.cn, niuzhs@tsinghua.edu.cn).

J. Zhao is with the Chinese University of Hong Kong (Shenzhen), Shenzhen 518172, China (e-mail: zhaojunlin@cuhk.edu.cn).

D. Gündüz is with the Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2BT, UK (e-mail: d.gunduz@imperial.ac.uk).

Part of this work has been presented in IEEE GLOBECOM 2019 [1].

improve system efficiency [6]. Multiple messages that include partial computation result are allowed to transmit from each worker to the master at each time slot, and thus stragglers can contribute a few messages, not none, to the system [19]. Multi-message communication may introduce additional transmission overhead, and the corresponding trade-off of communication and computation delay is investigated in [20]. Bivariate polynomial coding is introduced in [18], and is shown to reduce the average computation delay with respect to univariate polynomial alternatives. Such method is further combined with the concept of age of information for timely distributed computing in [21].

The papers above mainly address the straggler effect caused by the randomness of computation delay. Meanwhile, as the communication data volume between the master and worker nodes is usually high, the communication delay cannot be ignored either. Particularly, master and worker nodes might be base stations, mobile phones and smart vehicles at the edge of the wireless network, where the communication delay through wireless links may be highly stochastic and non-negligible. A scalable framework is proposed in [22] for coded distributed computing over wireless networks, where the communication load does not scale with the number of workers. Considering an MDS-coded distributed computing system with homogeneous workers, the impact of packet erasure channel on the delay of tasks is analyzed in [23]. Under heterogeneous settings, fixed transmission rate is considered in [24], and the load allocation of MDS-coded tasks is optimized. A cooperative transmission scheme for coded matrix multiplication is proposed in [25] to reduce the inter-cell interference, while a joint coding and node scheduling algorithm is proposed in [26] based on reinforcement learning.

Most existing papers on distributed coded computing only consider a single-master scenario, and the impact of communication delay on the load allocation has not been sufficiently investigated. In this work, we consider a multi-master heterogeneous-worker distributed computing scenario, where multiple matrix multiplication tasks are encoded with MDS codes, and allocated to workers for parallel computation, with random communication and computation delay. The goal is to jointly design worker assignment and load allocation algorithms to minimize the completion delay of all the tasks. The main contributions of this work are summarized as follows:

1) We consider both dedicated and fractional worker assignment policies, where each worker can process the encoded tasks of either a single master or multiple masters, respectively. Considering the randomness of communication and computation delays, we formulate a unified delay minimization problem for the joint allocation of computing power, communication bandwidth and task load.

2) For dedicated worker assignment, we obtain a non-convex mixed-integer non-linear programming problem (MINLP). The load allocation problem is solved first by deriving a convex approximation problem with Markov's inequality. Worker assignment is then transformed to a max-min allocation problem, which is NP-hard and solved with greedy heuristics. A successive convex approximation (SCA) based algorithm is proposed to further enhance the load allocation.

3) For fractional worker assignment, the optimization problem is non-convex. We again use Markov's inequality to simplify the problem, and transform the fractional worker assignment and resource allocation problem to max-min allocation by deriving its optimality condition. A greedy algorithm is proposed accordingly.

4) Simulations under various settings verify the feasibility of the proposed Markov's inequality based approximation, and show the significant delay reduction of the proposed algorithms over benchmarks. In particular, when using Amazon EC2 for delay evaluation, about 82% and 30% delay reductions are achieved by the proposed algorithms compared to the uncoded and coded benchmarks, respectively.

The rest of the paper is organized as follows. In Section II, we introduce the system model and formulate the problem. In Section III, we propose load allocation and worker assignment algorithms under the dedicated case. In Section IV, we further consider the fractional assignment case. Simulation results are shown in Section V, and conclusions are given in Section VI.

II. System Model and Problem Formulation

As shown in Fig. 1, we consider a distributed computing system with M master nodes and N worker nodes, denoted by $\mathcal{M} = \{1, 2, ..., M\}$ and $\mathcal{N} = \{1, 2, ..., N\}$, respectively. Each master m has a matrix-vector multiplication task, denoted by $\boldsymbol{A}_m \boldsymbol{x}_m$, where $\boldsymbol{A}_m \in \mathbb{R}^{L_m \times S_m}$, $\boldsymbol{x}_m \in \mathbb{R}^{S_m \times 1}$, and L_m, S_m are positive integers. Each task can be partitioned and allocated to a subset of workers and processed by them in parallel. Local computation at the master is also available, and thus the set of nodes that can serve master m is defined as $\mathcal{N}' \triangleq \mathcal{N} \cup \{0\}$, where index 0 represents local processing.

To reduce the straggler effect brought by the randomness of communication and computation, we introduce redundancy to each task through MDS coded computation. Specifically, each master encodes matrix $\boldsymbol{A}_m$ in units of rows to get its coded version $\widetilde{\boldsymbol{A}}_m \in \mathbb{R}^{\widetilde{L}_m \times S_m}$, where $\widetilde{L}_m \geq L_m$ denotes the number of coded rows. Then, the coded matrix $\widetilde{\boldsymbol{A}}_m$ is divided into $N+1$ disjoint sub-matrices $\widetilde{\boldsymbol{A}}_{m,0}$, $\widetilde{\boldsymbol{A}}_{m,1}$, $\ldots$, $\widetilde{\boldsymbol{A}}_{m,N}$, where $\widetilde{\boldsymbol{A}}_{m,n}$ has $l_{m,n} \in \mathbb{N}$ rows, i.e., $\widetilde{\boldsymbol{A}}_{m,n} \in \mathbb{R}^{l_{m,n} \times S_m}$. Note that, $l_{m,n} = 0$ indicates no assignment from master m to worker n. Let $\Omega_m \triangleq \{n | n \in \mathcal{N}, l_{m,n} > 0\}$ be the set of workers to serve master m.

After task encoding and assignment, each master m transmits $\widetilde{\boldsymbol{A}}_{m,n}$ and $\boldsymbol{x}_m$ to worker $n \in \Omega_m$ through their communication channel. We assume that the channel of each worker is orthogonal with that of others, and each worker can allocate its channel bandwidth to multiple masters and communicate with them simultaneously. This assumption is suitable for many realistic scenarios, e.g., the communication link is wired, or each worker is a base station with orthogonal wireless bandwidth. Each worker n calculates the multiplication of $\widetilde{\boldsymbol{A}}_{m,n}$ and $\boldsymbol{x}_m$, and transmits back the result. Finally, master m can recover the result of the original task $\boldsymbol{A}_m \boldsymbol{x}_m$ upon receiving the inner products of any L_m out of $\widetilde{L}_m$ coded rows of $\widetilde{\boldsymbol{A}}_m$ and vector $\boldsymbol{x}_m$.

A. Worker Assignment Policy

We consider two worker assignment policies in this work:

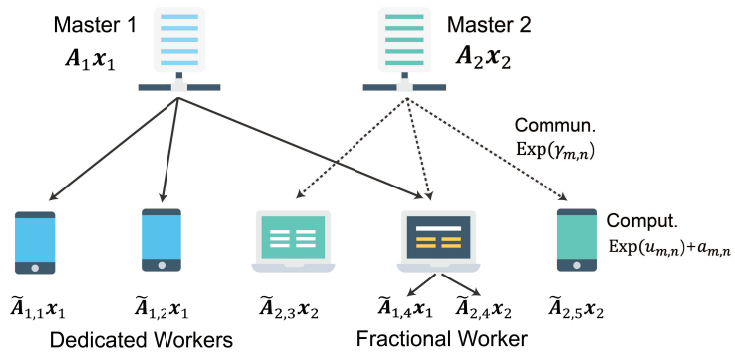


Fig. 1. Illustration of a distributed computing system with multiple master and worker nodes.

1) *Dedicated worker assignment:* Each worker only serves a single master. For $\forall n \in \mathcal{N}, m \in \mathcal{M}$, let $k_{m,n} \in \{0,1\}$ be the worker assignment indicator, where $k_{m,n} = 1$ if worker n is assigned a coded task by master m, and $k_{m,n} = 0$ otherwise. We have $\sum_{m=1}^{M} k_{m,n} \leq 1, \forall n \in \mathcal{N}$.

2) *Fractional worker assignment:* We allow each worker to serve multiple masters simultaneously through processor sharing. Let $k_{m,n} \in [0,1]$ be the fraction of computing power of worker n allocated to master m, with $\sum_{m=1}^{M} k_{m,n} \leq 1, \forall n \in \mathcal{N}$. Define $b_{m,n} \in [0,1]$ as the fraction of bandwidth allocated to the link between master m and worker n, with $\sum_{m=1}^{M} b_{m,n} \leq 1, \forall n \in \mathcal{N}$.

We assume that a master is always dedicated, i.e., it only computes local task but not helping others. Therefore, for $\forall m \in \mathcal{M}$, we have $k_{m,0} = 1$ and $b_{m,0} = 1$. Also note that, for dedicated worker assignment, the bandwidth allocation variable $b_{m,n}$ is binary, with $b_{m,n} = k_{m,n}$.

B. Communication and Computation Delays

We consider the delay of transmitting $\widetilde{\boldsymbol{A}}_{m,n}$ from master m to worker n, and ignore the transmission delays of $\boldsymbol{x}_m$ and the computation results. This is because the size of $\widetilde{\boldsymbol{A}}_{m,n}$ is typically much larger than that of $\boldsymbol{x}_m$ and the result vector. Moreover, as $\boldsymbol{x}_m$ is shared among multiple workers that serve master m, $\boldsymbol{x}_m$ can be transmitted in a more efficient way, such as broadcast or multicast.

The communication delay to transmit a single coded row from master m to worker n using the whole bandwidth is modeled by an exponential distribution [23], with rate parameter $\gamma_{m,n}$. Define the total communication delay of transmitting $\widetilde{\boldsymbol{A}}_{m,n}$ using $b_{m,n}$ of the bandwidth as $T_{m,n}^{[\mathrm{tr}]}$, whose cumulative distribution function (CDF) is given by:

$$\begin{aligned}&\mathbb{P}\left[T_{m,n}^{[\mathrm{tr}]}(l_{m,n}, b_{m,n}; \gamma_{m,n}) \leq t\right]\\ &= 1 - e^{-\frac{b_{m,n}\gamma_{m,n}}{l_{m,n}}t}, \ t \geq 0, \forall n \in \mathcal{N}, \forall m \in \mathcal{M}. \end{aligned} \tag{1}$$

At each master, local processing does not need communication, and thus $T_{m,0}^{[\mathrm{tr}]} = 0, \ \forall m$.

Following the literature [4]–[6], [13], [24], the delay of computing the inner product of one coded row of $\widetilde{\boldsymbol{A}}_m$ and vector $\boldsymbol{x}_m$ at worker n or master m ($n = 0$) is modeled by a shifted exponential distribution, with shift parameter $a_{m,n}$ and rate parameter $u_{m,n}$. For $\forall n \in \mathcal{N}', \ \forall m \in \mathcal{M}$, define the total computation delay of $\widetilde{\boldsymbol{A}}_{m,n}\boldsymbol{x}_m$ as $T_{m,n}^{[\mathrm{cp}]}$, with CDF

$$\begin{aligned}&\mathbb{P}\left[T_{m,n}^{[\mathrm{cp}]}(l_{m,n}, k_{m,n}; a_{m,n}, u_{m,n}) \leq t\right]\\ &= \begin{cases} 1 - e^{-\frac{k_{m,n}u_{m,n}}{l_{m,n}}\left(t - \frac{a_{m,n}l_{m,n}}{k_{m,n}}\right)}, & t \geq \frac{a_{m,n}l_{m,n}}{k_{m,n}}, \\ 0, & \text{otherwise.} \end{cases}\end{aligned} \tag{2}$$

Let $T_{m,n} \triangleq T_{m,n}^{[\mathrm{tr}]} + T_{m,n}^{[\mathrm{cp}]}$ be the total communication plus computation delay of the task assigned from master m to worker n, where $T_{m,n}^{[\mathrm{tr}]}$ and $T_{m,n}^{[\mathrm{cp}]}$ are two independent random variables. Then, if $b_{m,n}\gamma_{m,n} \neq k_{m,n}u_{m,n}$ and $t \geq \frac{a_{m,n}l_{m,n}}{k_{m,n}}$, the CDF of $T_{m,n}$ is given as follows:

$$\begin{aligned}\mathbb{P}[T_{m,n} \leq t] = 1 &- \frac{b_{m,n}\gamma_{m,n}e^{-\frac{k_{m,n}u_{m,n}}{l_{m,n}}\left(t - \frac{a_{m,n}l_{m,n}}{k_{m,n}}\right)}}{b_{m,n}\gamma_{m,n} - k_{m,n}u_{m,n}}\\ &+ \frac{k_{m,n}u_{m,n}e^{-\frac{b_{m,n}\gamma_{m,n}}{l_{m,n}}\left(t - \frac{a_{m,n}l_{m,n}}{k_{m,n}}\right)}}{b_{m,n}\gamma_{m,n} - k_{m,n}u_{m,n}}.\end{aligned} \tag{3}$$

If $b_{m,n}\gamma_{m,n} = k_{m,n}u_{m,n}$ and $t \geq \frac{a_{m,n}l_{m,n}}{k_{m,n}}$, the CDF of $T_{m,n}$ is

$$\begin{aligned}&\mathbb{P}\left[T_{m,n} \leq t\right] = 1 -\\ &\left[1 + \frac{k_{m,n}u_{m,n}}{l_{m,n}}\left(t - \frac{a_{m,n}l_{m,n}}{k_{m,n}}\right)\right]e^{-\frac{k_{m,n}u_{m,n}}{l_{m,n}}\left(t - \frac{a_{m,n}l_{m,n}}{k_{m,n}}\right)}.\end{aligned} \tag{4}$$

Otherwise, if $t < \frac{a_{m,n}l_{m,n}}{k_{m,n}}$, $\mathbb{P}\left[T_{m,n} \leq t\right] = 0$.

For local computation, we have $T_{m,0} = T_{m,0}^{[\mathrm{cp}]}, \ \forall m$. When $t \geq a_{m,0}l_{m,0}$, the CDF is given by

$$\mathbb{P}\left[T_{m,0} \leq t\right] = 1 - e^{-\frac{u_{m,0}}{l_{m,0}}(t - a_{m,0}l_{m,0})}, \ t \geq a_{m,0}l_{m,0}, \tag{5}$$

otherwise, $\mathbb{P}\left[T_{m,0} \leq t\right] = 0$.

C. Problem Formulation

Our objective is to minimize the task completion delay, by jointly optimizing the allocation of task load $\boldsymbol{l} \triangleq \{l_{m,n} | n \in \mathcal{N}', m \in \mathcal{M}\}$, computing power $\boldsymbol{k} \triangleq \{k_{m,n} | n \in \mathcal{N}, m \in \mathcal{M}\}$, and communication bandwidth $\boldsymbol{b} \triangleq \{b_{m,n} | n \in \mathcal{N}, m \in \mathcal{M}\}$. As the communication and computation delays are with random, we aim to minimize the delay t, upon which the probability that all the masters can recover their computations

is higher than a given threshold ρ_s. The optimization problem is formulated as:

$$\mathcal{P}1: \min_{\{\boldsymbol{l},\boldsymbol{k},\boldsymbol{b},t\}} \quad t \tag{6a}$$

$$\text{s.t.} \quad \mathbb{P}\left[X_m(t) \geq L_m\right] \geq \rho_s, \ \forall m \in \mathcal{M}, \tag{6b}$$

$$\sum_{m=1}^{M} k_{m,n} \leq 1, \sum_{m=1}^{M} b_{m,n} \leq 1, \forall n \in \mathcal{N}, \tag{6c}$$

$$k_{m,n} \in \mathcal{K}, b_{m,n} \in \mathcal{K}, \forall m \in \mathcal{M}, \forall n \in \mathcal{N}, \tag{6d}$$

$$l_{m,n} \in \mathbb{N}, \quad \forall m \in \mathcal{M}, \ \forall n \in \mathcal{N}'. \tag{6e}$$

In constraint (6b), $X_m(t)$ is defined as the number of computation results that can be received by the master m until time t, where a unit result refers to the inner product of one coded row of $\widetilde{\boldsymbol{A}}_m$ and vector $\boldsymbol{x}_m$. Constraint (6b) guarantees that each task can be recovered with probability ρ_s. Equation (6c) is the resource allocation constraint of each worker. In constraint (6d), we have $\mathcal{K} = \{0, 1\}$ for dedicated worker assignment, while $\mathcal{K} = [0, 1]$ for fractional worker assignment. In constraint (6e), $\mathbb{N}$ represents the set of non-negative integers.

Since workers have heterogeneous computing and communication capabilities, their loads $l_{m,n}$ will be different in general. To derive $\mathbb{P}\left[X_m(t) \geq L_m\right]$, we need to find all the combinations of $\{l_{m,n}, n \in \mathcal{N}'\}$ that satisfy $X_m(t) \geq L_m$, and further derive their joint probability distributions, which is intractable. As a result, problem $\mathcal{P}1$ is hard to solve.

We thus consider an approximation of $\mathcal{P}1$, where the probability constraint (6b) is substituted by an expectation constraint, shown as follows:

$$\mathcal{P}2: \min_{\{\boldsymbol{l},\ \boldsymbol{k},\ \boldsymbol{b},\ t\}} \quad t \tag{7a}$$

$$\text{s.t.} \quad \mathbb{E}[X_m(t)] \geq L_m, \ \forall m \in \mathcal{M}, \tag{7b}$$

$$l_{m,n} \geq 0, \quad \forall m \in \mathcal{M}, \ \forall n \in \mathcal{N}, \tag{7c}$$

$$\text{Constraints (6c), (6d).}$$

Constraint (7b) states that master m is expected to receive sufficient computation results to recover $\boldsymbol{A}_m \boldsymbol{x}_m$ until time t. Similar approximation approach is also used in [5], [13], [24], and the performance gap under a single master case can be bounded [5]. As $\boldsymbol{A}_m$ is with high dimension and thus the non-zero $l_{m,n}$ are typically large, we further relax $l_{m,n} \in \mathbb{N}$ to $l_{m,n} \geq 0$ in (7c), and ignore the rounding error in the following.

To simplify the system workflow as well as the theoretical analysis, we assume that each encoded task $\widetilde{\boldsymbol{A}}_{m,n} \boldsymbol{x}_m$, either being processed locally or allocated to a worker, is processed as a whole without any further partition. Accordingly, each master can only receive $l_{m,n}$ computation results from node $n \in \mathcal{N}'$ upon the completion. As computations on workers are independent, $\mathbb{E}[X_m(t)]$ can be written as follows:

$$\mathbb{E}[X_m(t)] = \sum_{n=0}^{N} \mathbb{E}\left[l_{m,n} \mathbb{I}_{\{T_{m,n} \leq t\}}\right] = \sum_{n=0}^{N} l_{m,n} \mathbb{P}\left[T_{m,n} \leq t\right],$$

where $\mathbb{I}_{\{x\}}$ denotes the indicator function with $\mathbb{I}_{\{x\}} = 1$ if event x is true, and $\mathbb{I}_{\{x\}} = 0$ otherwise. For $n \in \mathcal{N}$, $\mathbb{P}\left[T_{m,n} \leq t\right]$ is given in (3) or (4), and for $n = 0$, $\mathbb{P}\left[T_{m,n} \leq t\right]$ is given in (5).

In the following two sections, we design solutions to $\mathcal{P}2$ under dedicated and fractional worker assignments, respectively. We will further show in Section V that a good solution to $\mathcal{P}2$ can also achieve low delay under the constraints of $\mathcal{P}1$.

III. Dedicated Worker Assignment

In this section, we solve problem $\mathcal{P}2$ under the dedicated worker assignment policy, where $\mathcal{K} = \{0, 1\}$ and $b_{m,n} = k_{m,n}$. Accordingly, problem $\mathcal{P}2$ is a non-convex MINLP, which is very challenging to solve in general.

We decouple the binary worker assignment variable $\boldsymbol{k}$ and the continuous load allocation variable $\boldsymbol{l}$ to seek a solution. First, given any worker assignment decision, the load allocation problem is still non-convex. We use Markov's inequality to provide a convex approximation to the non-convex constraint, and derive the optimal load allocation for this subproblem. We also show that, when either the computation or communication delay plays a leading role, the original load allocation problem is convex, and the optimal solution can be derived. Then, based on the optimal load allocation, we transform the worker assignment problem into a max-min allocation problem, which is still NP-hard and thus solved with greedy heuristics. Finally, after optimizing the worker assignment, we further provide an enhanced load allocation algorithm by solving the original non-convex problem with the SCA method.

A. Load Allocation for the General Case

Given the set of workers $\Omega_m = \{n | k_{m,n} = 1, n \in \mathcal{N}\}$ that serve master m, the optimal load allocation problem aims to minimize the task completion delay t_m for each master m:

$$\mathcal{P}3: \min_{\{\boldsymbol{l}_m,\ t_m\}} \quad t_m \tag{8a}$$

$$\text{s.t.} \quad \mathbb{E}[X_m(t_m)] \geq L_m, \tag{8b}$$

$$l_{m,n} \geq 0, \ \forall n \in \Omega'_m, \tag{8c}$$

where $\Omega'_m \triangleq \Omega_m \cup \{0\}$ includes the master m itself, and $\boldsymbol{l}_m = \{l_{m,n} | n \in \Omega'_m\}$ denotes the load allocation vector. For $n \in \Omega_m$, the CDF of the total delay $T_{m,n}$ under dedicated assignment can be obtained by letting $k_{m,n} = 1$ and $b_{m,n} = 1$ in (3) and (4). Accordingly, $\mathbb{E}[X_m(t_m)] = \sum_{n \in \Omega'_m} l_{m,n} \mathbb{P}\left[T_{m,n} \leq t_m\right]$ is a non-convex function, making problem $\mathcal{P}3$ hard to solve.

We provide an approximation to $\mathbb{E}[X_m(t_m)]$ based on Markov's inequality, i.e., for $n \in \Omega_m$,

$$\mathbb{P}[T_{m,n} \geq t_m] \leq \frac{E[T_{m,n}]}{t_m} = \frac{l_{m,n}}{t_m}\left(\frac{1}{\gamma_{m,n}} + \frac{1}{u_{m,n}} + a_{m,n}\right). \tag{9}$$

At the master, $\mathbb{P}\left[T_{m,0} \geq t_m\right] \leq \frac{l_{m,0}}{t_m}\left(\frac{1}{u_{m,0}} + a_{m,0}\right)$. Let

$$\theta_{m,n} \triangleq \frac{1}{\gamma_{m,n}} + \frac{1}{u_{m,n}} + a_{m,n}, \ \theta_{m,0} \triangleq \frac{1}{u_{m,0}} + a_{m,0}. \tag{10}$$

Then we have

$$\mathbb{E}[X_m(t_m)] = \sum_{n\in\Omega'_m} l_{m,n}\mathbb{P}\left[T_{m,n}\le t_m\right] \ge \sum_{n\in\Omega'_m} l_{m,n}\left(1-\frac{\theta_{m,n}l_{m,n}}{t_m}\right). \quad (11)$$

Substituting inequality (11) into (8b), we obtain a tighter constraint, and an approximation to $\mathcal{P}3$ is given by

$$\mathcal{P}4: \min_{\{\boldsymbol{l}_m,t_m\}} \quad t_m \quad (12a)$$

$$\text{s.t.} \quad \sum_{n\in\Omega'_m} l_{m,n}\left(1-\frac{\theta_{m,n}l_{m,n}}{t_m}\right)\ge L_m, \quad (12b)$$

$$l_{m,n}\ge 0,\ \forall n\in\Omega'_m. \quad (12c)$$

Problem $\mathcal{P}4$ is a convex optimization problem, and the optimal solution is given as follows.

Theorem 1. *For a given subset of workers Ω_m that serves a master $m\in\mathcal{M}$, the optimal load allocation $\boldsymbol{l}^*_m$ and the corresponding task completion delay t^*_m to $\mathcal{P}4$ are*

$$l^*_{m,n} = \frac{L_m}{\theta_{m,n}\sum_{n\in\Omega'_m}\frac{1}{2\theta_{m,n}}},\ n\in\Omega'_m, \quad (13a)$$

$$t^*_m = \frac{L_m}{\sum_{n\in\Omega'_m}\frac{1}{4\theta_{m,n}}}. \quad (13b)$$

Proof. See Appendix A. □

As shown in (10), $\theta_{m,n}$ represents the expected total delay for worker n to handle a unit coded task of master m, and thus $\frac{1}{\theta_{m,n}}$ indicates the average communication plus computation rate. As shown in Theorem 1, the optimal load allocated to each worker n is proportional to $\frac{1}{\theta_{m,n}}$, while inversely proportional to the overall communication plus computation rates of workers.

B. Load Allocation for the Computation Delay Dominant Case

When computation delay is much larger than the communication delay, we ignore the latter and get $T_{m,n} = T^{[\text{cp}]}_{m,n}, \forall n\in\mathcal{N}'$. The CDF of $T_{m,n}$ is given in (2). It is easy to see that the optimal solution of $\mathcal{P}3$ must satisfy $t^*_m > \max_{\{n\in\Omega'_m\}}\{a_{m,n}l^*_{m,n}\}$. In fact, if there is a worker $n_0\in\Omega'_m$ such that $t^*_m\le a_{m,n_0}l_{m,n_0}$, then $l_{m,n_0}\mathbb{P}\left[T_{m,n_0}\le t^*_m\right]=0$, meaning that the master m cannot expect to obtain the computation results from worker n_0. By reducing l_{m,n_0} to satisfy $t^*_m > a_{m,n_0}l_{m,n_0}$, constraint (8b) can be strictly satisfied, and thus t^*_m can be further reduced.

Based on this observation, constraint (8b) of $\mathcal{P}3$ can be written as

$$\mathbb{E}[X_m(t_m)] = \sum_{n\in\Omega'_m} l_{m,n}\mathbb{P}\left[T_{m,n}\le t_m\right] = \sum_{n\in\Omega'_m} l_{m,n}\left(1-e^{-\frac{u_{m,n}}{l_{m,n}}(t_m-a_{m,n}l_{m,n})}\right). \quad (14)$$

The following theorem provides the optimal solution to $\mathcal{P}3$.

Theorem 2. *When computation delay dominates the total delay, $\mathcal{P}3$ is a convex optimization problem, and the optimal load allocation $\boldsymbol{l}^*_m$ and task completion delay t^*_m are*

$$l^*_{m,n} = \frac{L_m}{\phi_{m,n}\sum_{n\in\Omega'_m}\frac{u_{m,n}}{1+u_{m,n}\phi_{m,n}}},\ n\in\Omega'_m, \quad (15a)$$

$$t^*_m = \frac{L_m}{\sum_{n\in\Omega'_m}\frac{u_{m,n}}{1+u_{m,n}\phi_{m,n}}}, \quad (15b)$$

where $\phi_{m,n}\triangleq\frac{1}{u_{m,n}}\left[-\mathcal{W}_{-1}(-e^{-u_{m,n}a_{m,n}-1})-1\right]$, and $\mathcal{W}_{-1}(x)$ denotes the lower branch of Lambert W function, with $x\le -1$ and $\mathcal{W}_{-1}(xe^x)=x$.

Proof. See Appendix B. □

Similar results can be derived for the communication delay dominant case, by substituting $u_{m,n}$ with $\gamma_{m,n}$ and letting $a_{m,n}=0$.

C. Dedicated Worker Assignment Algorithms

In this subsection, we design worker assignment algorithms, aiming to assign workers to masters in a balanced manner and minimize the completion delay of the slowest task.

According to Theorem 1, the minimum task completion delay that can be achieved under a given subset of workers is

$$t^*_m = \frac{L_m}{\sum_{n\in\Omega'_m}\frac{1}{4\theta_{m,n}}} = \frac{L_m}{\frac{1}{4\theta_{m,0}}+\sum_{n=1}^{N}\frac{k_{m,n}}{4\theta_{m,n}}}, \quad (16)$$

where we recall that $k_{m,n}\in\{0,1\}$ is the worker assignment indicator.

From $\mathcal{P}2$, the objective of worker assignment is $\min_{\boldsymbol{k}}\max_{m\in\mathcal{M}} t^*_m$. As $t^*_m>0,\forall m$, the objective is equivalent to $\max_{\boldsymbol{k}}\min_{m\in\mathcal{M}}\frac{1}{t^*_m}$. Let $v_{m,n}\triangleq\frac{1}{4L_m\theta_{m,n}},\forall m\in\mathcal{M},\forall n\in\mathcal{N}'$, and thus

$$\frac{1}{t^*_m} = \frac{1}{L_m}\left[\frac{1}{4\theta_{m,0}}+\sum_{n=1}^{N}\frac{k_{m,n}}{4\theta_{m,n}}\right] = v_{m,0}+\sum_{n=1}^{N}k_{m,n}v_{m,n}. \quad (17)$$

The worker assignment problem can be transformed into the following form:

$$\mathcal{P}5: \min_{\boldsymbol{k}} \max_{m\in\mathcal{M}} \ v_{m,0}+\sum_{n=1}^{N}k_{m,n}v_{m,n} \quad (18a)$$

$$\text{s.t.} \quad \sum_{m=1}^{M}k_{m,n}\le 1,\ \forall n\in\mathcal{N}, \quad (18b)$$

$$k_{m,n}\in\{0,1\},\quad \forall m\in\mathcal{M},\ \forall n\in\mathcal{N}. \quad (18c)$$

Note that, for the computation delay dominant case, we only need to set $v_{m,n}=\frac{u_{m,n}}{L_m(1+u_{m,n}\phi_{m,n})}$, while the rest of the derivation still holds.

Problem $\mathcal{P}5$ is called *max-min allocation* problem, which is proposed for the fair assignment of items [27], [28]. In the original max-min allocation problem, each of the N items has a unique value for an agent, and can be assigned to one of the M agents. The objective is to assign all the items to the agents as fairly as possible, by maximizing the minimum total value of agents. In $\mathcal{P}5$, each worker n is an item with value $v_{m,n}$ for master m, and each master corresponds to an agent. The

Algorithm 1 Iterated Greedy Algorithm for Dedicated Worker Assignment

Input: Let $v_{m,n} = \frac{1}{4L_m\theta_{m,n}}, \forall m, n$, $V_m = v_{m,0}, \forall m$ and $\Omega_m = \emptyset, \forall m$.
for $n = 1, ..., N$ **do** ▷ *Initialization*
  Find $m^* = \arg\max_{m\in\mathcal{M}} v_{m,n}$, and update $V_{m^*} = V_{m^*} + v_{m^*,n}$, $\Omega_{m^*} = \Omega_{m^*} \cup \{n\}$.
end for
while termination condition is not satisfied **do** ▷ *Main iteration*
  for $n = 1, ..., N$ **do** ▷ *Insertion*
    Let m_1 be master that worker n is serving, and $m_2 = \arg\min_{m\in\mathcal{M}/\{m_1\}} V_m$.
    $V'_{m_1} = V_{m_1} - v_{m_1,n}$, $V'_{m_2} = V_{m_2} + v_{m_2,n}$, and $V'_m = V_m, \forall m \in \mathcal{M}/\{m_1, m_2\}$.
    if $\min_{m\in\mathcal{M}} V'_m > \min_{m\in\mathcal{M}} V_m$ **then**
      $\Omega_{m_1} = \Omega_{m_1} - \{n\}$, $\Omega_{m_2} = \Omega_{m_2} + \{n\}$.
    end if
  end for
  for $n_1, n_2 = 1, ...N$, and $n_1 \neq n_2$ **do** ▷ *Interchange*
    Masters m_1, m_2 served by workers n_1, n_2, $V'_{m_1} = V_{m_1} - v_{m_1,n_1} + v_{m_1,n_2}$, and $V'_{m_2} = V_{m_2} - v_{m_2,n_2} + v_{m_2,n_1}$.
    if $m_1 \neq m_2$, $v_{m_1,n_1} + v_{m_2,n_2} < v_{m_1,n_2} + v_{m_2,n_1}$, $V'_{m_1} > V_{\min}$, and $V'_{m_2} > V_{\min}$ **then**
      $\Omega_{m_1} = \Omega_{m_1} - \{n_1\} + \{n_2\}$, $\Omega_{m_2} = \Omega_{m_2} - \{n_2\} + \{n_1\}$.
    end if
  end for
  Randomly remove some workers in $\mathcal{N}_s \in \mathcal{N}$, and update V_m accordingly. ▷ *Exploration*
  while $\mathcal{N}_s \neq \emptyset$ **do**
    Find $\{m^*, n^*\} = \arg\max_{m\in\mathcal{M}, n\in\mathcal{N}_s} v_{m,n}$.
    Update $V_{m^*} = V_{m^*} + v_{m^*,n^*}$, $\Omega_{m^*} = \Omega_{m^*} \cup \{n^*\}$, $\mathcal{N}_s = \mathcal{N}_s - \{n^*\}$.
  end while
end while

Algorithm 2 Simple Greedy Algorithm for Dedicated Worker Assignment

Input: $\mathcal{N}_0 = \{1, 2, ..., N\}$, $v_{m,n} = \frac{1}{4L_m\theta_{m,n}}, \forall m, n$, $V_m = v_{m,0}, \forall m$, and $\Omega_m = \emptyset, \forall m$.
while $\mathcal{N}_0 \neq \emptyset$ **do**
  Find $m^* = \arg\min_{m\in\mathcal{M}} V_m$.
  Find $n^* = \arg\max_{n\in\mathcal{N}_0} v_{m^*,n}$.
  $V_{m^*} = V_{m^*} + v_{m^*,n^*}$.
  $\Omega_m = \Omega_m \cup \{n^*\}$, $\mathcal{N}_0 = \mathcal{N}_0 - \{n^*\}$.
end while

max-min allocation problem can be reduced to the partitioning problem [29], when considering only 2 agents and assuming that each item has the same value for each agent. Since the partitioning problem is NP-complete, the max-min allocation problem is NP-hard.

Although some polynomial-time algorithms have been proposed for the max-min allocation problem with worst-case performance guarantee [27], [28], they are very complex and difficult to implement. Instead, we propose two greedy algorithms in the following.

Inspired by [30], an iterated greedy algorithm is proposed, as shown in Algorithm 1. In the initialization phase, we assign each worker to the master with highest $v_{m,n}$, in order to maximize the contribution of workers. Then, we iterate among the insertion, interchange, and exploration phases, until the termination condition is met. To be specific, in the insertion phase, each worker is re-assigned to a master m_2 with the minimum sum value V_{m_2} if the minimum sum value of the masters is improved. In the interchange phase, any two workers exchange the masters they are serving, if the minimum sum values of both masters, and the total value of the workers are improved. In the exploration phase, a subset of workers are randomly removed from the current assignment, and allocated to the masters in a greedy manner. If the number of iterations reaches a preset maximum value, or the minimum sum value of the masters does not improve any more, the iteration is terminated. Note that, the final output is the worker assignment after the interchange phase.

As shown in Algorithm 2, we also propose a simple greedy algorithm that does not require iterations for performance improvement, inspired by the largest-value-first algorithm [31]. The initial value of each master is related to its local computation capability, given by $V_m = v_{m,0}$. During the main loop, we select a master m whose current sum value is the minimum, and allocate an available worker n with highest $v_{m,n}$ for master m. The algorithm terminates when all the workers are allocated.

D. SCA-Enhanced Load Allocation

The main purpose of using Markov's inequality for load allocation in the general case is to provide an explicit form for the worker assignment problem. After that, we can get back to the original load allocation problem $\mathcal{P}3$ to further improve the performance. We observe that the non-convex constraint (8b) in $\mathcal{P}3$ has a structure of the difference of convex functions, and thus we implement the SCA method to further optimize the load allocation.

When $\gamma_{m,n} \neq u_{m,n}$,

$$\mathbb{E}[X_m(t_m)] = \sum_{n\in\Omega'_m} l_{m,n}\mathbb{P}\left[T_{m,n} \leq t_m\right]$$
$$= l_{m,0}\left[1 - e^{-\frac{u_{m,0}}{l_{m,0}}(t_m - a_{m,0}l_{m,0})}\right] + \sum_{n\in\Omega_m} l_{m,n}\left[1 - \frac{\gamma_{m,n}e^{-\frac{u_{m,n}}{l_{m,n}}(t-a_{m,n}l_{m,n})} - u_{m,n}e^{-\frac{\gamma_{m,n}}{l_{m,n}}(t-a_{m,n}l_{m,n})}}{\gamma_{m,n} - u_{m,n}}\right]. \quad (19)$$

Let $\boldsymbol{w}_m \triangleq \{\boldsymbol{l}_m, t_m\}$. Without loss of generality, we assume $\gamma_{m,n} > u_{m,n}$, and let

$$h^+_{m,n}(\boldsymbol{w}_m) \triangleq \frac{\gamma_{m,n}l_{m,n}e^{-\frac{u_{m,n}}{l_{m,n}}(t-a_{m,n}l_{m,n})}}{\gamma_{m,n} - u_{m,n}},$$
$$h^-_{m,n}(\boldsymbol{w}_m) \triangleq \frac{u_{m,n}l_{m,n}e^{-\frac{\gamma_{m,n}}{l_{m,n}}(t-a_{m,n}l_{m,n})}}{\gamma_{m,n} - u_{m,n}}.$$

Algorithm 3 SCA-Enhanced Load Allocation Algorithm

1: **Input**: Given master m and its worker assignment Ω_m, find a feasible point $\boldsymbol{z}_0$ of $\mathcal{P}3$, and set $\gamma_0 = 1$, $r = 0$, $\alpha \in (0, 1)$.
2: **while** $\boldsymbol{z}_r$ is not a stationary solution **do**
3: Solve the optimal solution $\boldsymbol{w}_r$ to the optimization problem $\mathcal{P}(\boldsymbol{z}_r)$.
4: $\boldsymbol{z}_{r+1} = \boldsymbol{z}_r + \gamma_r(\boldsymbol{w}_r - \boldsymbol{z}_r)$.
5: $\gamma_{r+1} = \gamma_r(1 - \alpha\gamma_r)$, $r \leftarrow r + 1$.
6: **end while**

Otherwise, we can exchange $h^+_{m,n}(\boldsymbol{w}_m)$ with $h^-_{m,n}(\boldsymbol{w}_m)$, and the following solution still works. Let $h_{m,0}(\boldsymbol{w}_m) \triangleq -l_{m,0}\left[1 - e^{-\frac{u_{m,0}}{l_{m,0}}(t_m - a_{m,0}l_{m,0})}\right]$. From Appendix B, we know that $h_{m,0}(\boldsymbol{w}_m)$, $h^+_{m,n}(\boldsymbol{w}_m)$, and $h^-_{m,n}(\boldsymbol{w}_m)$ are all convex functions. Accordingly,

$$L_m - \mathbb{E}[X_m(t_m)] = L_m - \sum_{n\in\Omega_m} l_{m,n} + h_{m,0}(\boldsymbol{w}_m) + \sum_{n\in\Omega_m} \left(h^+_{m,n}(\boldsymbol{w}_m) - h^-_{m,n}(\boldsymbol{w}_m)\right), \quad (20)$$

that is, $L_m - \mathbb{E}[X_m(t_m)]$ can be decomposed into the difference of convex functions.

For any given point $\boldsymbol{z}$, a convex upper bound of $h^+_{m,n}(\boldsymbol{w}) - h^-_{m,n}(\boldsymbol{w})$ can be obtained by linearizing $h^-_{m,n}(\boldsymbol{w})$:

$$h^+_{m,n}(\boldsymbol{w}) - h^-_{m,n}(\boldsymbol{w}) \leq h^+_{m,n}(\boldsymbol{w}) - h^-_{m,n}(\boldsymbol{z}) - \nabla_{\boldsymbol{w}} h^-_{m,n}(\boldsymbol{z})^T(\boldsymbol{w} - \boldsymbol{z}). \quad (21)$$

Let $\tilde{h}_{m,n}(\boldsymbol{w}, \boldsymbol{z}) \triangleq h^+_{m,n}(\boldsymbol{w}) - h^-_{m,n}(\boldsymbol{z}) - \nabla_{\boldsymbol{w}} h^-_{m,n}(\boldsymbol{z})^T(\boldsymbol{w} - \boldsymbol{z})$. A convex approximation problem to $\mathcal{P}3$ under point $\boldsymbol{z}$, denoted by $\mathcal{P}(\boldsymbol{z})$, is given by

$$\mathcal{P}(\boldsymbol{z}): \min_{\{\boldsymbol{w}_m\}} \quad t_m \quad (22a)$$

$$\text{s.t.} \quad h_{m,0}(\boldsymbol{w}_m) + \sum_{n\in\Omega_m} \tilde{h}_{m,n}(\boldsymbol{w}_m, \boldsymbol{z}) \leq \sum_{n\in\Omega_m} l_{m,n} - L_m, \quad (22b)$$

$$l_{m,n} \geq 0, \ \forall n \in \Omega'_m. \quad (22c)$$

Based on the SCA method proposed in [32], we develop an SCA-enhanced load allocation algorithm, as shown in Algorithm 3. For each master m and the corresponding worker assignment Ω_m by Algorithm 1 or 2, the SCA algorithm starts from a feasible point of $\boldsymbol{z}_0$ of $\mathcal{P}3$. Note that, the Markov's inequality provides a tighter approximation to constraint (8b), and thus Theorem 1 directly provides $\boldsymbol{z}_0$. Then, we iteratively solve convex optimization problems $\mathcal{P}(\boldsymbol{z}_r)$ until convergence, where in the r-th iteration, $\boldsymbol{z}_r$ is updated according to Line 4 using step-size γ_r. According to [32], we update γ_r with a decreasing ratio $\alpha \in (0, 1)$, so as to guarantee the convergence to a local optimum.

As a summary, we would like to provide the following remarks.

Remark 1. *Scope of application:* While we assumed certain delay distributions in the system model, the Markov's inequality based approximate load allocation and the corresponding worker assignment algorithms, introduced in Section III-A and Section III-C, *do not* rely on these distributions. Instead, the proposed solution can be applied to any communication and computation delay distributions with broad adaptivity, as long as their mean values are known. To further carry out the SCA-enhanced load allocation, we need to specify the delay distributions.

Remark 2. *Iterated matrix multiplication:* Distributed matrix-vector multiplication is often needed for the training of large ML models, where matrix $\boldsymbol{A}_m$ corresponds to the data and vector $\boldsymbol{x}_m$ to the model [4], [19]. Using a common training algorithm such as distributed gradient descent, the coded data is transmitted to the workers at the beginning, while multiple iterations of computations are required with the updated model vector. In this scenario, we can use the result of the computation-delay dominant case for worker assignment and load allocation, or modify the communication delay distribution of $\boldsymbol{x}_m$ by removing the load variable $l_{m,n}$.

IV. Fractional Worker Assignment

While dedicated worker assignment only needs a simple communication connection topology between masters and workers, it may lead to an unbalanced worker assignment, particularly when a few workers are much more powerful than the others, or the number of workers is relatively small. Therefore, in this section, we further consider fractional worker assignment, by allowing each worker to serve multiple masters simultaneously. In this case, we have $\mathcal{K} = [0, 1]$, $k_{m,n}, b_{m,n} \in \mathcal{K}$, $\forall m, n$, and the CDF of the total delay $T_{m,n}$ is given in (3) and (4). Accordingly, problem $\mathcal{P}2$ is a non-convex optimization problem, which is difficult to solve directly.

Similarly to Section III, we use Markov's inequality to derive an approximation to problem $\mathcal{P}2$, and further simplify the resultant optimization problem by analyzing its optimality condition. We show that the joint bandwidth and computing power allocation under fractional assignment can also be transformed to a max-min allocation problem, and propose a greedy algorithm based on Algorithms 1 and 2.

A. Markov's Inequality based Approximation and its Optimality Condition

Using the Markov's inequality, $\forall m \in \mathcal{M}$ and $\forall n \in \mathcal{N}$ with $b_{m,n} \neq 0$ and $k_{m,n} \neq 0$,

$$\mathbb{P}\left[T_{m,n} \leq t\right] = 1 - \mathbb{P}\left[T_{m,n} \geq t\right] \geq 1 - \frac{E[T_{m,n}]}{t} = 1 - \frac{l_{m,n}}{t}\left(\frac{1}{b_{m,n}\gamma_{m,n}} + \frac{1}{k_{m,n}u_{m,n}} + \frac{a_{m,n}}{k_{m,n}}\right). \quad (23)$$

In the fractional assignment case, the expected total delay for worker $n \in \mathcal{N}$ to handle a unit coded task of master $m \in \mathcal{M}$ is given by

$$\theta_{m,n} = \begin{cases} \frac{1}{b_{m,n}\gamma_{m,n}} + \frac{1}{k_{m,n}u_{m,n}} + \frac{a_{m,n}}{k_{m,n}}, & k_{m,n}, b_{m,n} > 0, \\ \infty, \quad k_{m,n} = 0 \text{ or } b_{m,n} = 0. \end{cases} \quad (24)$$

For local computation at each master m, we still have $\theta_{m,0} = \frac{1}{u_{m,0}} + a_{m,0}$. Considering the inherent feature of the system, $k_{m,n}$, $b_{m,n}$ and $l_{m,n}$ are either all non-zero or all zero.

Algorithm 4 Greedy Algorithm for Fractional Worker Assignment

1: **Input**: Get an initial dedicated worker assignment according to Algorithm 1 or 2. Let $b_{m,n} = k_{m,n}, \forall m, n$. Initialize $\theta_{m,n}$ according to (24), and let $V_m = \frac{1}{L_m}\sum_{n=0}^{N}\frac{1}{4\theta_{m,n}}, \forall m$.
2: **while** $\max_m V_m > \min_m V_m$ **do**
3: $m_1 = \arg\max V_m$, $m_2 = \arg\min V_m$, $\mathcal{N}_{\text{tmp}} = \{n | k_{m_1,n} > 0, k_{m_2,n} = 0\}$.
4: Calculate $\theta'_{m_2,n} = \frac{1}{b_{m_1,n}\gamma_{m_2,n}} + \frac{1}{k_{m_1,n}u_{m_2,n}} + \frac{a_{m_2,n}}{k_{m_1,n}}$, $\forall n \in \mathcal{N}_{\text{tmp}}$.
5: Find worker $n_1 = \arg\max_{n\in\mathcal{N}_{\text{tmp}}} \frac{1}{\theta'_{m_2,n}}$ with maximum performance gain for master m_2.
6: **if** $V_{m_1} - \frac{1}{4\theta_{m_1,n_1}L_{m_1}} \leq V_{m_2} + \frac{1}{4\theta'_{m_2,n_1}L_{m_2}}$ **then**
7: Find $k_{m_1,n_1}, b_{m_1,n_1}, k_{m_2,n_1}, b_{m_2,n_1}$, such that $V_{m_1} = V_{m_2}$.
8: **else**
9: Assign all the resource of worker n_1 for master m_1 to master m_2, i.e., let $k_{m_2,n_1} = k_{m_1,n_1}$, $b_{m_2,n_1} = b_{m_1,n_1}$, and then let $k_{m_1,n_1} = 0$, $b_{m_1,n_1} = 0$.
10: **end if**
11: Update θ_{m_1,n_1}, θ_{m_2,n_1}, V_{m_1}, V_{m_2} accordingly.
12: **end while**

Substituting (23) into (8b), an approximation to problem $\mathcal{P}2$ under the fractional worker assignment policy is given by

$$\mathcal{P}6: \min_{\{\boldsymbol{l},\ \boldsymbol{k},\ \boldsymbol{b},\ t\}} \quad t \tag{25a}$$

$$\text{s.t.} \quad L_m - \sum_{n=0}^{N} l_{m,n}\left(1 - \frac{l_{m,n}\theta_{m,n}}{t}\right) \leq 0, \forall m, \tag{25b}$$

$$\sum_{m=1}^{M} k_{m,n} \leq 1, \quad \sum_{m=1}^{M} b_{m,n} \leq 1, \quad \forall n, \tag{25c}$$

$$k_{m,n}, b_{m,n} \in [0,1], l_{m,n} \geq 0, \quad \forall m, n. \tag{25d}$$

Compared to $\mathcal{P}4$, problem $\mathcal{P}6$ needs to jointly optimize load allocation $\boldsymbol{l}$ and the resource allocation $\boldsymbol{k}$ and $\boldsymbol{b}$, which is still non-convex. In the following theorem, we derive the KKT optimality condition for $\mathcal{P}6$.

Theorem 3. *Given any resource allocation $k_{m,n}$ and $b_{m,n}$, the optimal load allocation $l^*_{m,n}$ to problem $\mathcal{P}6$ that minimizes delay t^* must satisfy the following condition:*

$$l^*_{m,n} = \frac{t^*}{2\theta_{m,n}}, \ n \in \mathcal{N}', \tag{26}$$

where $\theta_{m,n}$ is derived from (24) *according to $k_{m,n}$ and $b_{m,n}$.*

Proof. See Appendix C. □

B. Fractional Worker Assignment Algorithm

Based on Theorem 3, without loss of optimality, constraint (25b) can be simplified to

$$L_m - \sum_{n=0}^{N} \frac{t}{4\theta_{m,n}} \leq 0, \ \forall m. \tag{27}$$

Therefore, problem $\mathcal{P}6$ is equivalent to:

$$\mathcal{P}7: \max_{\{\boldsymbol{k},\ \boldsymbol{b}\}} \min_{m\in\mathcal{M}} \frac{1}{L_m}\sum_{n=0}^{N}\frac{1}{4\theta_{m,n}} \tag{28a}$$

$$\text{s.t.} \quad \text{constraints (25c), (25d).} \tag{28b}$$

We can see that $\mathcal{P}7$ is very similar to the max-min allocation problem $\mathcal{P}5$ under dedicated worker assignment, except that $\theta_{m,n}$ can further change with respect to the computing power allocation $k_{m,n} \in [0,1]$ and communication bandwidth allocation $b_{m,n} \in [0,1]$. Therefore, we adopt the dedicated assignment as an initialization, and iteratively balance the resource allocation between the master $m_1 = \arg\max V_m$ with maximum sum value and master $m_2 = \arg\min V_m$ with minimum sum value. To balance their sum values, we select a worker n_1 that serves m_1 but not m_2 for the moment, with maximum potential performance gain for m_2, as shown in Lines 3-5. Then, part or all of the computing power and communication bandwidth of worker n_1 are re-assigned to master m_2, as shown in Lines 6-10. Note that, in practice, we may not want to make the topology of masters and workers too complicated. In that case, we can limit the maximum number of masters each worker can serve in Algorithm 4.

We also remark that, by substituting $\gamma_{m,n}$, $u_{m,n}$ and $a_{m,n}$ in (19) with $b_{m,n}\gamma_{m,n}$, $k_{m,n}u_{m,n}$ and $\frac{a_{m,n}}{k_{m,n}}$, respectively, the SCA-enhanced load allocation (Algorithm 3) can also be implemented for the fractional worker assignment problem after Algorithm 4.

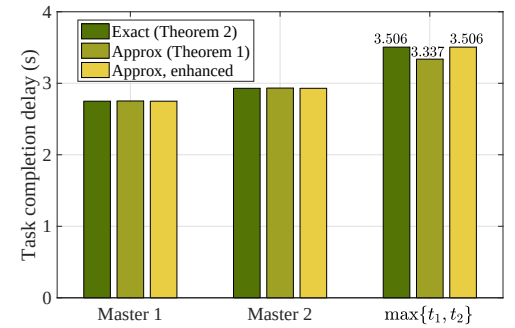
V. Simulation Results

In this section, we show the simulation results of the proposed algorithms under various settings. We first verify the feasibility of the Markov's inequality based approximation, and then evaluate the task completion delay of the proposed algorithms and benchmarks. Finally, we sample the task completion delay from commercial compute platform Amazon EC2, and use the measured data to further validate the proposed algorithms.

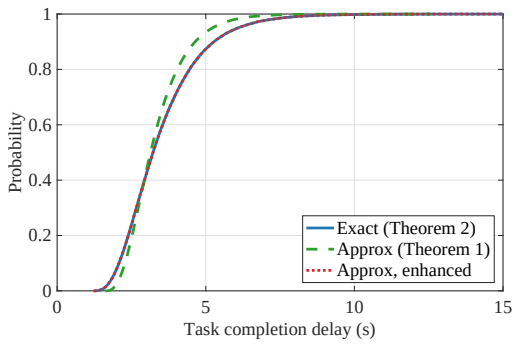
A. Validation of Markov's Inequality based Approximation

We first show in Fig. 2 and Fig. 3 that, the task completion delay achieved by solving the Markov's inequality based approximation problem is similar to the original problem. To achieve this verification, we consider the computation delay dominant case, where the optimal load allocation to the original problem $\mathcal{P}3$ (*'Exact'*) can be derived from Theorem 2, and the approximate load allocation (*'Approx'*) is derived from Theorem 1. Based on the two load allocation results, iterated greedy algorithm, i.e., Algorithm 1, is adopted to assign the workers in a dedicated manner. Corresponding to the SCA-enhanced load allocation in Section III-D, we further use Theorem 2 to improve the performance after obtaining the worker assignment based on the approximate load allocation, labeled as *'Approx, enhanced'*.

We consider two scenarios with different scales. In the small-scale scenario, there are $M = 2$ masters and $N = 5$ workers. For each worker, the shift parameter $a_{m,n}$

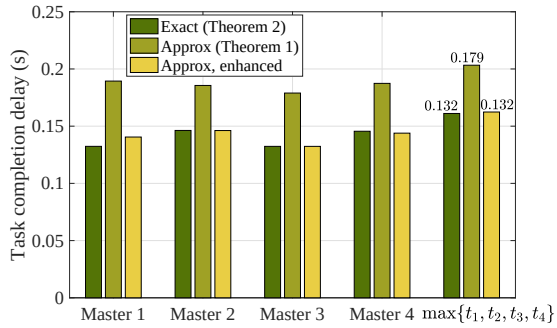


(a) Average task completion delay.

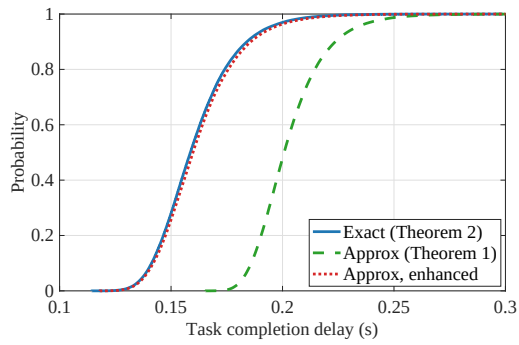


(b) CDF of task completion delay.

Fig. 2. Validation of the Markov's inequality based approximation in the 2-master, 5-worker case.



(a) Average task completion delay.



(b) CDF of task completion delay.

Fig. 3. Validation of the Markov's inequality based approximation in the 4-master, 50-worker case.

of the computation delay distribution is randomly selected from $\{0.2, 0.25, 0.3\}$ ms, while for each master, $a_{m,0} \in \{0.4, 0.5\}$ ms. The rate parameter is $u_{m,n} = \frac{1}{a_{m,n}}, \forall m, n$, and the load of the original task is set to $L_m = 10^4, \forall m$ [5]. In the large-scale scenario, there are $M = 4$ masters and $N = 50$ workers. Parameter $a_{m,n}$ is randomly chosen from $[0.05, 0.5]$ ms, while $u_{m,n}$ and L_m remain the same. After deriving the load allocation and worker assignment from the corresponding theorems and algorithms, we run Monte Carlo realizations for 10^6 times and present the average value and the CDF of the empirical task completion delay.

Fig. 2 and Fig. 3 show the validation results under the small-scale and large-scale scenarios, respectively. In each histogram, the first M groups of bars show the average task completion delay of each master under different solutions. The last group of bars show the average delay of all the tasks, which is what we aim to minimize in $\mathcal{P}2$, obtained by taking the maximum delay among M masters in each Monte Carlo realization and then taking the average. Overall, the gap between the Markov's inequality based approximate solution and the optimal solution is acceptable, while the enhanced approximate solution has almost the same performance as the optimal one, in terms of both the average and the CDF of task completion delay under both scenarios. We can also see from Fig. 2(a) that, the approximate solution can achieve even lower average delay when the number of workers is small. This is because, the approximate problem $\mathcal{P}4$ provides a tighter constraint and thus increases the redundancy of load, making the system more robust to stragglers in some cases.

B. Performance of the Proposed Dedicated and Fractional Worker Assignment Algorithms

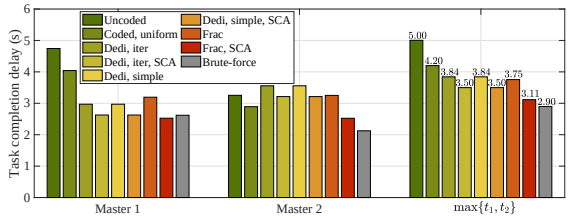
Now we take the communication delay into account and evaluate the proposed algorithms. The simulation settings for both small-scale and large-scale scenarios remain the same as the previous subsection, while the communication rate parameter of each worker is set to $\gamma_{m,n} = 2u_{m,n}, \forall m, n$. For the SCA algorithm, the step-size decreasing ratio is set to $\alpha = 0.995$. We compare the delay performance of the proposed algorithms with the following benchmarks:

1) Uncoded computation with uniform worker assignment: Each master is assigned an equal number of $\frac{N}{M}$ workers, and $\boldsymbol{A}_m$ is equally partitioned into $\frac{N}{M}$ sub-matrices without coding.

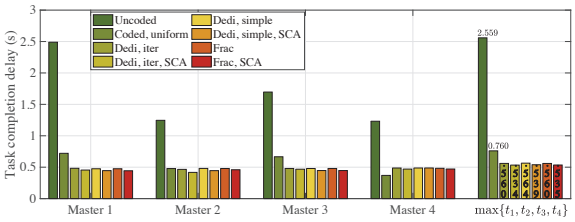
2) Coded computation with uniform worker assignment: Each master is assigned an equal number of $\frac{N}{M}$ workers, and the load allocation is given by Theorem 2. This benchmark can be regarded as the scheme presented in [5], where only the computation delay is considered under a single master scenario.

3) Brute-force search for optimal fractional worker assignment: The optimal benchmark is obtained by traversing all possible $k_{m,n}$ and $b_{m,n}$ at a step-size of 0.01. SCA-enhanced load allocation is further implemented after getting the optimal fractional worker assignment. Note that, as the brute-force search is with extremely high complexity, we can only provide this result in the small-scale scenario.

The average task completion delay in the two scenarios are shown in Fig. 4. We use 'Dedi, iter', 'Dedi, simple' and 'Frac' to represent the worker assignment results from Algorithms

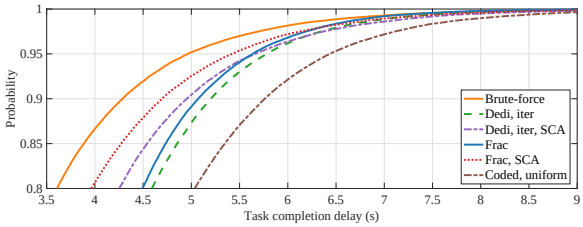


(a) $M = 2$ masters, $N = 5$ workers.

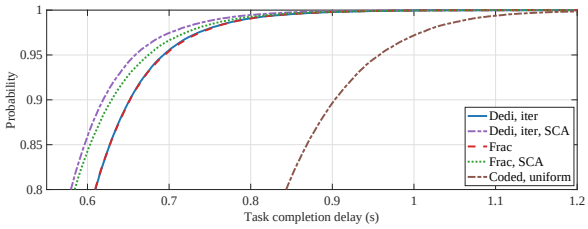


(b) $M = 4$ masters, $N = 50$ workers.

Fig. 4. Average task completion delay of the proposed algorithms and benchmarks.



(a) $M = 2$ masters, $N = 5$ workers.



(b) $M = 4$ masters, $N = 50$ workers.

Fig. 5. CDF of the task completion delay of the proposed algorithms and benchmarks.

1, 2 and 4, respectively. The legend with 'SCA' indicates that SCA-enhanced load allocation is further implemented. As shown in Fig. 4(a), in the small-scale scenario, our proposed algorithms outperform the uncoded and coded benchmarks by balancing the worker assignment, while the fractional assignment is slightly better than the dedicated one. With SCA enhancement, the average delay can be decreased by 8.85% under dedicated worker assignment, while the delay can be substantially decreased by 17.1% with fractional assignment. We can also see that, the delay performance of the SCA-enhanced fractional assignment is close-to-optimal. As shown in Fig. 4(b), in the large-scale scenario, iterated greedy algorithm can seek a better assignment compared to the simple greedy algorithm under the dedicated case. On the other hand, fractional assignment achieves the same performance as iterated greedy, since dedicated algorithm can already balance the worker assignment when the number of workers is large. With SCA-enhancement, the delay performance can be further decreased by over 4.4%, but we should also be aware that the complexity of the SCA algorithm is high under the large-scale scenario. Compared to the uncoded and coded benchmarks, up to 79% and 30% delay reduction can be achieved by the proposed algorithm, respectively.

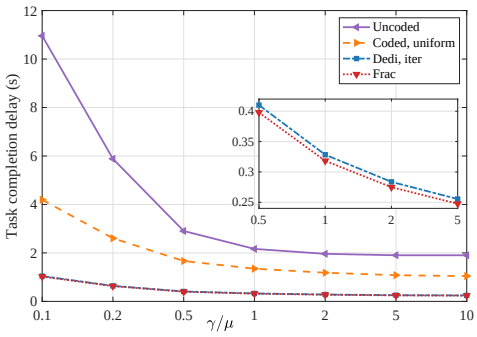
In order to observe the delay performance of the initial optimization problem $\mathcal{P}1$, we further plot the CDF of the task completion delay in Fig. 5. Given the probability threshold ρ_s, we can obtain the corresponding delay from the x-axis, such that constraint (6b) is satisfied. This figure shows the tail distribution of the task completion delay, and can reflect the robustness of the system under different algorithms. As shown in Fig. 5(b), given $\rho_s = 0.95$, the delays achieved by the SCA-enhanced dedicated assignment, dedicated assignment and the coded benchmark are 0.658s, 0.694s and 0.957s, respectively. That is, over 30% delay reduction can be achieved by the proposed algorithm compared to the coded benchmark. We can also see that, a good solution to the approximation problem $\mathcal{P}2$ also leads to a good delay performance for the original problem $\mathcal{P}1$ in general, and thus solving $\mathcal{P}2$ is reasonable.

The impact of communication rate on the average task completion delay and the load allocation is investigated in Fig. 6, by varying $\frac{\gamma_{m,n}}{u_{m,n}}$ while fixing $u_{m,n}$. As shown in Fig. 6(a), when $\frac{\gamma_{m,n}}{u_{m,n}}$ is small, the communication rate between each master and worker is low, and thus the average task completion delay is high. Meanwhile, the proposed dedicated and fractional worker assignment algorithms always achieve significantly lower delay compared to the benchmarks. Fig. 6(b) plots the ratio of load allocated to the master itself to the total load, i.e., $\frac{l_{m,0}}{\sum_{n\in\mathcal{N}'} l_{m,n}}$. As the two benchmarks do not take the communication delay into account, the ratio remains the same over different communication rates. With the proposed algorithms, this ratio decreases as $\frac{\gamma_{m,n}}{u_{m,n}}$ increases, since more computation load is allocated to the workers when communication is faster.

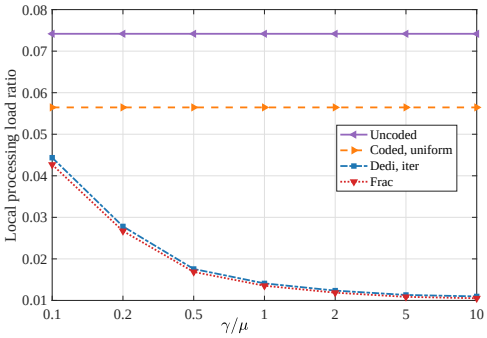
C. Delay Sampling on Amazon EC2 Instances and the Corresponding Algorithm Evaluation

We further evaluate the delay performance of the proposed algorithms based on measured computation delays on the commercial compute platform Amazon EC2. To do so, we sample the computation delay on two types of Amazon EC2 instances called t2.micro and c5.large, by feeding each instance a 10^6-dimension, float-number vector multiplication task for 10^6 times. We plot the cumulative frequency distribution of the sampled computation delay in Fig. 7, and then fit the data with shifted exponential distribution. For the t2.micro instance, the shift parameter is $a = 1.36$ ms, and the rate parameter is $u = 4.976$ ms^{-1}. For c5.large instance, $a = 0.97$ ms and $u = 19.29$ ms^{-1}. We can see that, in terms of the computation capability, the c5.large instance is more powerful than the t2.micro instance, and the fitting of the shifted exponential distribution is accurate.

Finally, we use the measured data to evaluate the proposed algorithms. We consider a computation delay dominant scenario with 4 masters and 50 workers. All the masters and 40 workers are considered as t2.micro instances, while the

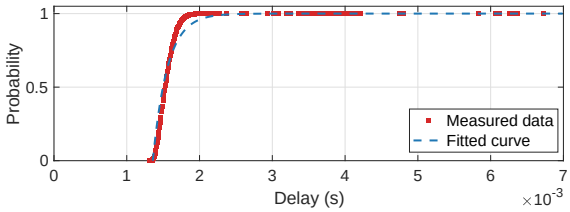


(a) Average task completion delay.

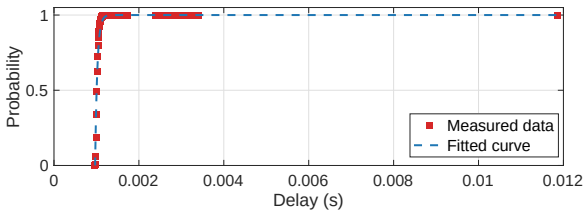


(b) The ratio of local processing load to total load.

Fig. 6. Performance evaluation under different communication rates in the 4-master, 50-worker case.



(a) t2.micro instance.



(b) c5.large instance.

Fig. 7. The cumulative frequency distribution of the measured delay and its fitted curve based on shifted exponential distribution, on different types of Amazon EC2 instances.

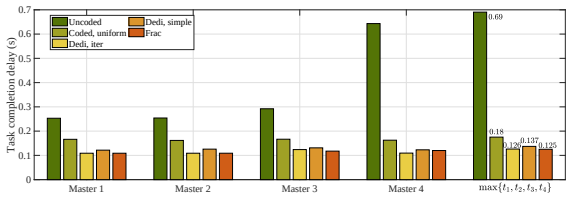


Fig. 8. Average task completion delay under the 4-master, 50-worker case.

remaining 10 workers are c5.large instances. We use the fitted distribution for load allocation and worker assignment, and then use the measured data to simulate the average task completion delay with the Monte Carlo method. As shown in Fig. 8, the proposed dedicated and fractional worker assignment algorithms still outperform the uncoded and coded benchmarks, with up to 82% and 30% delay reductions, respectively. Comparing the two dedicated assignment algorithms, the iterated greedy algorithm achieves a much lower delay under this practical scenario. Meanwhile, fractional assignment can slightly decrease the average task completion delay compared with the iterated dedicated assignment.

Summarizing all the simulation results, we remark that, in the small-scale scenario where the number of master and worker nodes is small, SCA-enhanced fractional assignment is the best algorithm with great advantage over other alternatives. On the other hand, in the large-scale scenario, dedicated assignment by the iterated greedy algorithm is satisfactory when considering the delay performance together with the complexity of the algorithm and the network topology.

VI. Conclusions

We have investigated a joint worker assignment, resource allocation, and load allocation problem in an MDS-coded distributed computing scenario with multiple masters and heterogeneous workers, aiming to minimize the communication and computation delay of tasks. Dedicated and fractional worker assignment and load allocation algorithms have been proposed, employing Markov's inequality-based approximation, Karush-Kuhn-Tucker conditions, and SCA techniques for the analysis and optimization of these algorithms. Simulations under various settings have shown that the proposed algorithms can significantly reduce the task completion delay compared to the benchmark algorithms, while the SCA-enhanced fractional assignment algorithm can achieve close-to-optimal delay performance when the number of master and worker nodes is small. Considering measured data on Amazon EC2 platform for delay evaluation, we have shown that about 82% and 30% delay reductions can be achieved by the proposed algorithms compared to the uncoded and coded benchmarks, respectively. We have observed that SCA-enhanced fractional assignment significantly outperforms the other proposed algorithms under small-scale scenarios. Meanwhile, the dedicated policy with iterated greedy assignment can be a practical alternative for large-scale scenarios, when the delay performance, resultant communication network topology, and algorithm complexity are jointly taken into account.

As future directions, multi-message communication schemes [20] as well as the costs of encoding and decoding can be further incorporated into the current optimization framework.

Appendix A
Proof of Theorem 1

For $x > 0,\ y > 0,\ f(x,y) = \frac{x^2}{y}$ is convex. Therefore, problem $\mathcal{P}4$ is a convex optimization problem. The Lagrangian

of $\mathcal{P}4$ is given by

$$\mathcal{L}(\boldsymbol{l}_m, t_m, \lambda_m) = t_m + \lambda_m \left[L_m - \sum_{n\in\Omega'_m} \left(l_{m,n} - \frac{\theta_{m,n} l_{m,n}^2}{t_m} \right) \right],$$

where $\lambda_m \geq 0$ is the Lagrange multiplier associated with (12b).

The partial derivatives of $\mathcal{L}(\boldsymbol{l}_m, t_m, \lambda_m)$ can be derived as

$$\frac{\partial \mathcal{L}}{\partial l_{m,n}} = -\lambda_m + \lambda_m \theta_{m,n} \frac{2 l_{m,n}}{t_m}, \tag{29a}$$

$$\frac{\partial \mathcal{L}}{\partial t_m} = 1 - \lambda_m \sum_{n\in\Omega'_m} \frac{\theta_{m,n} l_{m,n}^2}{t_m^2}. \tag{29b}$$

The Karush-Kuhn-Tucker (KKT) conditions are written as follows:

$$\frac{\partial \mathcal{L}}{\partial l^*_{m,n}} = 0,\ \forall n \in \Omega'_m, \quad \frac{\partial \mathcal{L}}{\partial t^*_m} = 0, \tag{30a}$$

$$\lambda^*_m \left[L_m - \sum_{n\in\Omega'_m} \left(l^*_{m,n} - \frac{\theta_{m,n} {l^*_{m,n}}^2}{t^*_m} \right) \right] = 0, \tag{30b}$$

$$\lambda^*_m \geq 0, \quad l^*_{m,n} \geq 0. \tag{30c}$$

By solving the KKT conditions, we get the optimal load allocation and task completion delay to $\mathcal{P}4$, as shown in Theorem 1.

Appendix B
Proof of Theorem 2

When the computation delay dominates the total delay, the optimization problem $\mathcal{P}3$ is given by

$$\begin{aligned} \mathcal{P}3(1): \min_{\{\boldsymbol{l}_m,\ t_m\}} \quad & t_m \\ \text{s.t.} \quad & L_m \leq \sum_{n\in\Omega'_m} l_{m,n} \left(1 - e^{-\frac{u_{m,n}}{l_{m,n}}(t_m - a_{m,n} l_{m,n})} \right), \\ & l_{m,n} \geq 0,\ \forall n \in \Omega'_m. \end{aligned}$$

We first prove that $\mathcal{P}3(1)$ is a convex optimization problem. Let $f(x,t) = -x\left(1 - e^{-\frac{u}{x}(t-ax)}\right)$, with variables $x > 0$, $t \geq ax$, and parameters $u > 0$, $a > 0$. The Hessian matrix of $f(x,t)$ is:

$$\boldsymbol{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial t} \\ \frac{\partial^2 f}{\partial t \partial x} & \frac{\partial^2 f}{\partial t^2} \end{bmatrix} = e^{-\frac{u}{x}(t-ax)} \begin{bmatrix} \frac{u^2 t^2}{x^3} & -\frac{u^2 t}{x^2} \\ -\frac{u^2 t}{x^2} & \frac{u^2}{x} \end{bmatrix}. \tag{32}$$

The eigenvalues of $\boldsymbol{H}$ are 0 and $\frac{u^2(x^2+t^2)}{x^3} > 0$. Thus $\boldsymbol{H} \succeq 0$, and $f(x,t)$ is convex. As the summation of convex functions are still convex, (32) is convex. Therefore, $\mathcal{P}3(1)$ is convex.

The Lagrangian is given by

$$\begin{aligned} &\mathcal{L}(\boldsymbol{l}_m, t_m, \lambda_m) = t_m + \lambda_m \left(L_m - \mathbb{E}[X_m(t_m)] \right) \\ &= t_m + \lambda_m \left[L_m - \sum_{n\in\Omega'_m} l_{m,n} \left(1 - e^{-\frac{u_{m,n}}{l_{m,n}}(t_m - a_{m,n} l_{m,n})} \right) \right]. \end{aligned} \tag{33}$$

The partial derivatives of $\mathcal{L}$ can be derived as

$$\frac{\partial \mathcal{L}}{\partial l_{m,n}} = \lambda_m \left[\left(1 + \frac{u_{m,n} t_m}{l_{m,n}} \right) e^{-\frac{u_{m,n}}{l_{m,n}}(t_m - a_{m,n} l_{m,n})} - 1 \right], \tag{34a}$$

$$\frac{\partial \mathcal{L}}{\partial t_m} = 1 - \lambda_m \sum_{n\in\Omega'_m} u_{m,n} e^{-\frac{u_{m,n}}{l_{m,n}}(t_m - a_{m,n} l_{m,n})}. \tag{34b}$$

The optimal solution $(\boldsymbol{l}^*_m, t^*_m, \lambda^*_m)$ needs to satisfy the following KKT conditions

$$\frac{\partial \mathcal{L}}{\partial l^*_{m,n}} = 0,\ \forall n \in \Omega'_m, \quad \frac{\partial \mathcal{L}}{\partial t^*_m} = 0, \tag{35a}$$

$$\lambda^*_m \left[L_m - \sum_{n\in\Omega'_m} l^*_{m,n} \left(1 - e^{-\frac{u_{m,n}}{l^*_{m,n}}(t^*_m - a_{m,n} l^*_{m,n})} \right) \right] = 0, \tag{35b}$$

$$\lambda^*_m \geq 0,\ l^*_{m,n} \geq 0. \tag{35c}$$

By jointly considering (34b) and (35a), we get $\lambda^*_m > 0$. Substituting (34a) into $\frac{\partial \mathcal{L}}{\partial l^*_{m,n}} = 0$ yields

$$\begin{aligned} &\left(1 + \frac{t^*_m u_{m,n}}{l^*_{m,n}} \right) e^{u_{m,n}\left(a_{m,n} - \frac{t^*_m}{l^*_{m,n}} \right)} - 1 = 0, \\ &-\left(1 + \frac{t^*_m u_{m,n}}{l^*_{m,n}} \right) e^{-\left(1 + \frac{t^*_m u_{m,n}}{l^*_{m,n}} \right)} = -e^{-u_{m,n} a_{m,n} - 1}, \end{aligned}$$

Let $\mathcal{W}_{-1}(x)$ be the lower branch of Lambert W function, where $x \leq -1$ and $\mathcal{W}_{-1}(xe^x) = x$. Then we have

$$\frac{t^*_m}{l^*_{m,n}} = \frac{-\mathcal{W}_{-1}(-e^{-u_{m,n} a_{m,n} - 1}) - 1}{u_{m,n}} \triangleq \phi_{m,n}. \tag{36}$$

Substituting (36) into (35b):

$$L_m - \sum_{n\in\Omega_m} \frac{t^*_m}{\phi_{m,n}} \left(1 - \frac{1}{1 + u_{m,n}\phi_{m,n}} \right) = 0. \tag{37}$$

Then, t^*_m and $l^*_{m,n}$ can be derived, as shown in Theorem 2.

Appendix C
Proof of Theorem 3

Given any $\boldsymbol{k}$ and $\boldsymbol{b}$, the Lagrangian of $\mathcal{P}6$ is given by

$$\begin{aligned} &\mathcal{L}(\boldsymbol{l}, t, \lambda_m) \\ &= t + \lambda_m \left[L_m - \sum_{n\in\mathcal{N}'} \left(l_{m,n} - \frac{\theta_{m,n} l_{m,n}^2}{t} \right) \right], \forall m. \end{aligned} \tag{38}$$

For the non-convex optimization problem, the optimal solution $\{\boldsymbol{l}^*, t^*\}$ must satisfy the KKT conditions. By solving

$$\begin{aligned} &\frac{\partial \mathcal{L}}{\partial l^*_{m,n}} = -\lambda^*_m + \lambda^*_m \theta_{m,n} \frac{2 l^*_{m,n}}{t^*} = 0, \\ &\frac{\partial \mathcal{L}}{\partial t^*} = 1 - \lambda^*_m \sum_{n\in\mathcal{N}'} \frac{\theta_{m,n} (l^*_{m,n})^2}{(t^*)^2} = 0, \\ &\lambda^*_m \left[L_m - \sum_{n\in\mathcal{N}'} \left(l^*_{m,n} - \frac{\theta_{m,n} (l^*_{m,n})^2}{t^*} \right) \right] = 0, \\ &\lambda^*_m \geq 0,\ l^*_{m,n} \geq 0, \end{aligned} \tag{39a}$$

we derive the optimality condition, as shown in Theorem 3.

References

[1] Y. Sun, J. Zhao, S. Zhou, and D. Gunduz, "Heterogeneous coded computation across heterogeneous workers," in *Proc. IEEE Global Commun. Conf.*, Waikoloa, HI, USA, Dec. 2019.

[2] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp.107-113, Jan. 2008.

[3] J. Dean, et al. "Large scale distributed deep networks," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, Dec. 2012.

[4] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514-1529, Mar. 2018.

[5] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Trans. Inform. Theory*, vol. 65, no. 7, pp. 4227-4242, July 2019.

[6] N. Ferdinand and S. C. Draper, "Hierarchical coded computation," in *Proc. IEEE Int. Symp. on Inform. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 1620-1624.

[7] K. Gardner, M. Harchol-Balter, A. Scheller-Wolf, M. Velednitsky, and S. Zbarsky, "Redundancy-d: The power of d choices for redundancy," *Operations Research*, vol. 65, no. 4, pp. 1078-1094, Apr. 2017.

[8] G. Joshi, E. Soljanin, and G. Wornell, "Efficient redundancy techniques for latency reduction in cloud systems," *ACM Trans. Modeling Perform. Eval. Comput. Syst. (TOMPECS)* vol. 2, no. 2, pp. 1-30, Apr. 2017.

[9] M. Mohammodi Amiri and D. Gunduz, "Computation scheduling for distributed machine learning with straggling workers," *IEEE Trans. Signal Process.*, vol. 67, no. 24, pp. 6270-6284, Dec. 2019.

[10] J. S. Ng, W. Y. B. Lim, N. C. Luong, Z. Xiong, A. Asheralieva, D. Niyato, C. Leung, and C. Miao, "A survey of coded distributed computing," *arXiv preprint arXiv:2008.09048*, Aug. 2020.

[11] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "A unified coding framework for distributed computing with straggling servers," *IEEE Global Commun. Conf. Workshop*, Washington, DC, USA, Dec. 2016.

[12] D. Kim, H. Park, and J. K. Choi, "Optimal load allocation for coded distributed computation in heterogeneous clusters," *IEEE Trans. Commun.*, vol. 69, no. 1, pp. 44-58, Jan. 2021.

[13] F. Zhang, Y. Sun, and S. Zhou, "Coded computation over heterogeneous workers with random task arrivals," *IEEE Commun. Lett.*, vol. 25, no. 7, pp. 2338-2342, July 2021.

[14] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: avoiding stragglers in distributed learning," in *Proc. Int. Conf. on Machine Learning*, Sydney, Australia, Aug. 2017, pp. 3368-3376.

[15] A. Reisizadeh, S. Prakash, R. Pedarsani, and S. A. Avestimehr, "Tree gradient coding," in *Proc. IEEE Int. Symp. on Inform. Theory (ISIT)*, Paris, France, July 2019.

[16] R. Bitar, M. Wootters, and S. El Rouayheb, "Stochastic gradient coding for straggler mitigation in distributed learning," *IEEE J. Sel. Areas Commun.*, vol. 1, no. 1, pp. 277-291, May 2020.

[17] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security, and privacy," in *Proc. 22rd Int. Conf. on Artificial Intelligence and Statistics*, PMLR, vol. 89, pp. 1215-1225, Apr. 2019.

[18] B. Hasircioglu, J. Gomez-Vilardebo and D. Gunduz, "Bivariate polynomial coding for efficient distributed matrix multiplication," in *IEEE J. Sel. Areas Inform. Theory*, early access, Aug. 2021.

[19] E. Ozfatura, D. Gunduz, and S. Ulukus, "Speeding up distributed gradient descent by utilizing non-persistent stragglers," in *Proc. IEEE Int. Symp. on Inform. Theory (ISIT)*, Paris, France, July 2019.

[20] E. Ozfatura, S. Ulukus, and D. Gunduz, "Straggler-aware distributed learning: Communication-computation latency trade-off," *Entropy*, 22(5):544, May 2020.

[21] B. Buyukates and S. Ulukus, "Timely distributed computation with stragglers," *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5273-5282, Sept. 2020.

[22] S. Li, Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "A scalable framework for wireless distributed computing," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2643–2654, May 2017.

[23] D. J. Han, J. -y. Sohn, and J. Moon, "Coded distributed computing over packet erasure channels," in *Proc. IEEE Int. Symp. on Inform. Theory (ISIT)*, Paris, France, July 2019.

[24] F. Wu and L. Chen, "Latency optimization for coded computation straggled by wireless transmission," *IEEE Wireless Commun. Lett.*, vol. 9, no. 7, pp. 1124-1128, July 2020.

[25] K. Li, M. Tao, J. Zhang and O. Simeone, "Coded computing and cooperative transmission for wireless distributed matrix multiplication," *IEEE Trans. Commun.*, vol. 69, no. 4, pp. 2224-2239, Apr. 2021.

[26] N. Van Huynh, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "Joint coding and scheduling optimization for distributed learning over wireless edge networks," *arXiv preprint arXiv:2103.04303*, Mar. 2021.

[27] D. Chakrabarty, J. Chuzhoy, and S. Khanna, "On allocating goods to maximize fairness," *50th Annual IEEE Symposium on Foundations of Computer Science*, Atlanta, GA, USA, Oct. 2009, pp. 107-116.

[28] A. Asadpour, and A. Saberi, "An approximation algorithm for max-min fair allocation of indivisible goods," *SIAM J. Algebraic Discrete Methods*, vol. 39, no. 7, pp. 2970-2989, May 2010.

[29] B. Hayes, "Computing science: the easiest hard problem," *American Scientist*, vol. 90, no. 2, pp. 113-117, Apr. 2002.

[30] L. Fanjul-Peyro, R. Ruiz, "Iterated greedy local search methods for unrelated parallel machine scheduling," *European Journal of Operational Research*, vol. 207, no. 1, pp. 55-69, Nov. 2010.

[31] B. Deuermeyer, D. Friesen, and M. Langston, "Scheduling to maximize the minimum processor finish time in a multiprocessor system," *SIAM J. Algebraic Discrete Methods*, vol. 3, no. 2, pp. 190-196, Jun. 1982.

[32] G. Scutari, F. Facchinei, and L. Lampariello, "Parallel and distributed methods for constrained nonconvex optimization -part I: theory," *IEEE Trans. Signal Process.*, vol. 65, no. 8, pp. 1929-1944, Apr. 2017.