Heterogeneous Coded Computation across Heterogeneous Workers

Yuxuan Sun*, Junlin Zhao†, Sheng Zhou*, Deniz Gündüz†
*Beijing National Research Center for Information Science and Technology
Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
†Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2BT, UK
Email: {sunyx15@mails., sheng.zhou@}tsinghua.edu.cn, {j.zhao15, d.gunduz}@imperial.ac.uk

***Abstract*—Coded distributed computing framework enables large-scale machine learning (ML) models to be trained efficiently in a distributed manner, while mitigating the straggler effect. In this work, we consider a multi-task assignment problem in a coded distributed computing system, where multiple masters, each with a different matrix multiplication task, assign computation tasks to workers with heterogeneous computing capabilities. Both *dedicated* and *probabilistic* worker assignment models are considered, with the objective of minimizing the average completion time of all tasks. For dedicated worker assignment, greedy algorithms are proposed and the corresponding optimal load allocation is derived based on the Lagrange multiplier method. For probabilistic assignment, successive convex approximation method is used to solve the non-convex optimization problem. Simulation results show that the proposed algorithms reduce the completion time by 80% over uncoded scheme, and 49% over an unbalanced coded scheme.**

I. Introduction

Machine learning (ML) techniques are penetrating into many aspects of human lives, and boosting the development of new applications from autonomous driving, virtual and augmented reality, to Internet of things [1]. Training complicated ML models requires computations with massive volumes of data, e.g., large-scale matrix-vector multiplications, which cannot be realized on a single centralized server. Distributed computing frameworks such as MapReduce [2] enable a centralized *master* node to allocate data and update global model, while tens or hundreds of distributed computing nodes, called *workers*, train ML models in parallel using partial data. Since task completion time depends on the slowest worker, a key bottleneck in distributed computing is the *straggler effect*: experiments on Amazon EC2 instances show that some workers can be 5 times slower than the typical ones [3].

Straggler effect can be mitigated by adding redundancy to the distributed computing system via coding [2]–[8], or by scheduling computation tasks [9]–[11]. Maximum distance separable (MDS) codes are widely applied for matrix multiplications [2]–[7], which can reduce the task completion time by $O(\log N)$, where N is the number of workers [2]. A unified coded computing framework for straggler mitigation is proposed in [4]. Heterogeneous workers are considered in [5], and an asymptotically optimal load allocation scheme is proposed. Although the stragglers are slower than the typical workers, they can still make non-negligible contributions to the system [6], [8]. A hierarchical coded computing framework is thus proposed in [6], where tasks are partitioned into multiple levels so that stragglers contribute to subtasks in the lower levels. Multi-message communication with Lagrange coded computing is used in [8] to exploit straggler servers.

The straggler effect can be mitigated even with uncoded computing, via redundant scheduling of tasks and multi-message communications. A batched coupon's collector scheme is proposed in [9], and the expected completion time is analyzed in [10]. The input data is partitioned into batches, and each worker randomly processes one at a time, until the master collects all the results. Deterministic scheduling orders of tasks at different workers are proposed in [11], specifically cyclic and staircase scheduling, and the relation between redundancy and task completion time is characterized.

Existing papers mainly consider a single master. However, in practice, workers may be shared by more than one masters to carry out multiple large-scale computation tasks in parallel. Therefore, in this work, we focus on a multi-task assignment problem for a heterogeneous distributed computing system using MDS codes. As shown in Fig. 1, we consider multiple masters, each with a matrix-vector multiplication task, and a number of workers with *heterogeneous* computing capabilities. The goal is to design centralized worker assignment and load allocation algorithms that minimize the completion time of all the tasks. We consider both *dedicated* and *probabilistic* worker assignment policies, and formulate a non-convex optimization problem under a unified framework. For dedicated assignment, each worker serves one master. The optimal load allocation is derived, and the worker assignment is transformed into a max-min allocation problem, for which NP-hardness is proved and greedy algorithms are proposed. For probabilistic assignment, each worker selects a master to serve based on an optimized probability, and a successive convex approximation (SCA) based algorithm is proposed. Simulation results show that the proposed algorithms can drastically reduce the task completion

This work is sponsored in part by the European Research Council (ERC) under Starting Grant BEACON (grant No. 725731), the Nature Science Foundation of China (No. 61871254, No. 91638204, No. 61571265, No. 61861136003, No. 61621091), National Key R&D Program of China 2018YFB0105005, and Intel Collaborative Research Institute for Intelligent and Automated Connected Vehicles.

978-1-7281-0962-6/19/$31.00 ©2019 IEEE

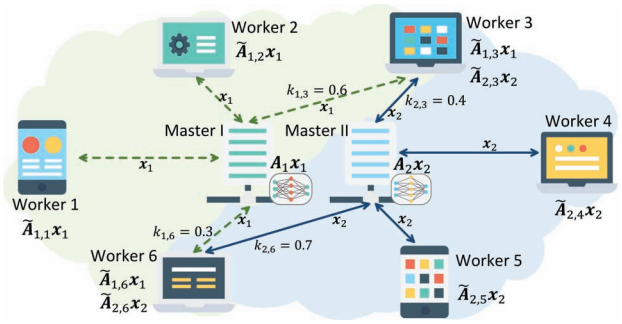

Fig. 1. Illustration of a distributed computing system with multiple master nodes and worker nodes.

time compared to uncoded and unbalanced coded schemes.

The rest of the paper is organized as follows. The system model and problem formulation is introduced in Sec. II. Dedicated and probabilistic worker assignments, and the corresponding load allocation algorithms are proposed in Sec. III and Sec. IV, respectively. Simulation results are presented in Sec. V, and the conclusions are summarized in Sec. VI.

II. System Model and Problem Formulation

A. System Architecture

We consider a heterogeneous distributed computing system with M masters $\mathcal{M} = \{1, 2, ..., M\}$, and N workers $\mathcal{N} = \{1, 2, ..., N\}$, with $N > M$. We assume that each master has a matrix-vector multiplication task[1]. The task of master m is denoted by $\boldsymbol{A}_m\boldsymbol{x}_m$, where $\boldsymbol{A}_m \in \mathbb{R}^{L_m \times s_m}$, $\boldsymbol{x}_m \in \mathbb{R}^{s_m}$, $L_m, s_m \in \mathbb{Z}^+$. The masters can use the workers to complete their computation tasks in a distributed manner.

To deal with the straggling workers, we adopt MDS coded computation, and encode the rows of $\boldsymbol{A}_m$. Define the coded version of $\boldsymbol{A}_m$ as $\tilde{\boldsymbol{A}}_m$, which is further divided into N sub-matrices $\tilde{\boldsymbol{A}}_m = \left[\tilde{\boldsymbol{A}}_{m,1}^T, \tilde{\boldsymbol{A}}_{m,2}^T, \cdots, \tilde{\boldsymbol{A}}_{m,N}^T\right]^T$, where $\tilde{\boldsymbol{A}}_{m,n} \in \mathbb{R}^{l_{m,n} \times s_m}$ is assigned to worker n, and $l_{m,n}$ is a non-negative integer representing the load allocated to worker n. Vector $\boldsymbol{x}_m$ is multicast from master m to the workers with $l_{m,n} > 0$, and worker n calculates the multiplication of $l_{m,n}$ coded rows of $\boldsymbol{A}_m$ (which is $\tilde{\boldsymbol{A}}_{m,n}$) and $\boldsymbol{x}_m$. Matrix $\boldsymbol{A}_m$ is thus $\left(\sum_{n=1}^N l_{m,n}, L_m\right)$-MDS-coded, with the requirement of $\sum_{n=1}^N l_{m,n} \geq L_m$. Upon aggregating the multiplication results for any L_m coded rows of $\boldsymbol{A}_m$, master m can recover $\boldsymbol{A}_m\boldsymbol{x}_m$.

B. Task Processing Time

The processing times of the assigned computation tasks at the workers are modeled as mutually independent random variables. Following the literature on coded computing [2], [4]–[6], the processing time at each worker is modeled by a shifted exponential distribution[2]. The processing time, $T_{m,n}^{[l_{m,n}]}$, for worker n to calculate the multiplication of $l_{m,n} > 0$ coded rows of $\boldsymbol{A}_m$ and $\boldsymbol{x}_m$ has the cumulative distribution function:

$$\mathbb{P}\left[T_{m,n}^{[l_{m,n}]} \leq t\right] = \begin{cases} 1 - e^{-\frac{u_{m,n}}{l_{m,n}}(t - a_{m,n}l_{m,n})}, & t \geq a_{m,n}l_{m,n}, \\ 0, & \text{otherwise}, \end{cases} \quad (1)$$

where $a_{m,n} > 0$ is a parameter indicating the minimum processing time for one coded row, and $u_{m,n} > 0$ is the parameter modeling the straggling effect.

We consider a *heterogeneous environment* by assuming that $u_{m,n}$ and $a_{m,n}$ are different over different master-worker pairs (m, n), for $\forall m \in \mathcal{M}$ and $\forall n \in \mathcal{N}$. This assumption is due to the fact that workers may have different computation speeds, and the dimensions of $\boldsymbol{A}_m$ and $\boldsymbol{x}_m$ vary over m.

C. Worker Assignment Policy

We consider two worker assignment policies:

1) *Dedicated worker assignment:* In this policy, each worker n is assigned computation tasks from a single master $m \in \mathcal{M}$. Let indicator $k_{m,n} = 1$ if worker n provides computing service for master m, and $k_{m,n} = 0$ otherwise. Since a worker serves at most one master, we have $\sum_{m=1}^M k_{m,n} \leq 1, \forall n \in \mathcal{N}$.

2) *Probabilistic worker assignment:* In this policy, each worker randomly selects which master to serve according to probability $k_{m,n} \in [0, 1]$. For each worker $n \in \mathcal{N}$, we have $\sum_{m=1}^M k_{m,n} \leq 1$. In Fig. 1, worker 3 selects master 1 to serve with probability 0.6, and master 2 with probability 0.4.

D. Problem Formulation

Let $X_{m,n}(t)$ denote the number of multiplication results (one result refers to the multiplication of one coded row of $\boldsymbol{A}_m$ with $\boldsymbol{x}_m$) master m collects from worker n till time t. We assume that worker n computes $\tilde{\boldsymbol{A}}_{m,n}\boldsymbol{x}_m$ and then sends the result to the master m upon completion, without further dividing it into subtasks or transmitting any feedbacks before completion. Therefore, master m can either receive $l_{m,n}$ results or none from worker n till time t. We denote the number of aggregated results at master m until time t by $X_m(t)$, and we have $X_m(t) = \sum_{n=1}^N X_{m,n}(t)$.

Our objective is to minimize the average completion time t, upon which all the masters can aggregate sufficient results from the workers to recover their computations with high probability. We aim to design a centralized policy that optimizes worker assignment $\{k_{m,n}\}$ and load allocation $\{l_{m,n}\}$. The optimization problem is formulated as:

$$\mathcal{P}1: \min_{\{l_{m,n}, k_{m,n}, t\}} \quad t \quad (2a)$$

$$\text{s.t.} \quad \mathbb{P}\left[X_m(t) \geq L_m\right] \geq \rho_s, \ \forall m, \quad (2b)$$

$$\sum_{m=1}^M k_{m,n} \leq 1, \quad \forall n, \quad (2c)$$

$$k_{m,n} \in \mathcal{K}, \quad l_{m,n} \in \mathbb{N}, \quad \forall m, n, \quad (2d)$$

where we have $\mathcal{K} = \{0, 1\}$ for dedicated worker assignment, while $\mathcal{K} = [0, 1]$ for probabilistic worker assignment, and $\mathbb{N}$ is the set of non-negative integers. In constraint (2b), ρ_s is defined as the probability that master m receives no less than

[1]In training ML models, e.g., linear regression, matrix-vector multiplication tasks are carried out at each iteration of the gradient descent algorithm. These tasks are independent over iterations, thus we focus on one iteration here.

[2]In this work, the worker assignment and load allocation algorithms are designed based on the assumption of shifted exponential distribution. However, the proposed methods can also be applied to other distributions, as long as the corresponding function $-l_{m,n}\mathbb{P}\left[T_{m,n}^{[l_{m,n}]} \leq t\right]$ is convex.

L_m results until time t; that is, the probability of $\boldsymbol{A}_m\boldsymbol{x}_m$ being recovered. Constraint (2c) states that under dedicated assignment, each worker serves at most one master, and under probabilistic assignment, the total probability rule is satisfied.

The key challenge to solve $\mathcal{P}1$ is that, constraint (2b) cannot be explicitly expressed, since it is difficult to find all the combinations that satisfy $X_m(t) \geq L_m$ in a heterogeneous environment with non-uniform loads $\{l_{m,n}\}$. Therefore, we instead consider an approximation to this problem, by substituting constraint (2b) with an expectation constraint:

$$\mathcal{P}2: \min_{\{l_{m,n},k_{m,n},t\}} \quad t \tag{3a}$$

$$\text{s.t.} \quad L_m - \mathbb{E}[X_m(t)] \leq 0, \ \forall m, \tag{3b}$$

$$\text{Constraints (2c), (2d),}$$

where constraint (3b) states that the expected number of results master m receives until time t is no less than L_m. A similar approach is used in [5], where the gap between the solutions of $\mathcal{P}1$ and $\mathcal{P}2$ is proved to be bounded when there is a single master. We will design algorithms that solve $\mathcal{P}2$ in the following two sections.

Constraint (3b) can be explicitly expressed. Let $\mathbb{I}_{\{x\}}$ be an indicator function with value 1 if event $\{x\}$ is true, and 0 otherwise. If $k_{m,n} > 0$ (and thus $l_{m,n} > 0$),

$$\begin{aligned}\mathbb{E}[X_{m,n}(t)] &= \mathbb{E}\left[k_{m,n}l_{m,n}\mathbb{I}_{\left\{T_{m,n}^{[l_{m,n}]} \leq t\right\}}\right] \\ &= \begin{cases} k_{m,n}l_{m,n}\left[1-e^{-\frac{u_{m,n}}{l_{m,n}}(t-a_{m,n}l_{m,n})}\right], t \geq a_{m,n}l_{m,n}, \\ 0, \quad \text{otherwise.} \end{cases}\end{aligned} \tag{4}$$

If $k_{m,n} = 0$ (and thus $l_{m,n} = 0$), $\mathbb{E}[X_{m,n}(t)] = 0$. And we have $\mathbb{E}[X_m(t)] = \sum_{n=1}^{N} \mathbb{E}[X_{m,n}(t)]$.

We observe that: 1) From constraint (3b), we can infer that for $\forall m \in \mathcal{M}$, the optimal task completion time t^* satisfies $t^* \geq \max_{\{n\in\Omega_m\}}\{a_{m,n}l_{m,n}\}$, where $\Omega_m \subset \mathcal{N}$ is the subset of workers serving master m. In fact, if there exists $n_0 \in \Omega_m$ such that $t^* < a_{m,n_0}l_{m,n_0}$, we have $\mathbb{E}[X_{m,n_0}(t^*)] = 0$, i.e., master m cannot expect to receive any results from worker n_0. By reducing l_{m,n_0} to satisfy $\mathbb{E}[X_{m,n_0}(t^*)] > 0$, it is possible to further reduce t^*. 2) Due to the high dimension of input matrix $\boldsymbol{A}_m$, $l_{m,n}$ is usually in the order of hundreds or thousands. So we relax the constraint $l_{m,n} \in \mathbb{N}$ to $l_{m,n} \geq 0$, and ignore the effect of rounding in the following derivations.

Therefore, by substituting (4), we can simplify (3b) as:

$$L_m - \sum_{n=1}^{N} k_{m,n}l_{m,n}\left(1 - e^{-\frac{u_{m,n}}{l_{m,n}}(t-a_{m,n}l_{m,n})}\right) \leq 0. \tag{5}$$

And problem $\mathcal{P}2$ can be simplified as follows:

$$\mathcal{P}3: \min_{\{l_{m,n},k_{m,n},t\}} \quad t \tag{6a}$$

$$\text{s.t.} \quad \text{Constraints (2c), (5),}$$

$$k_{m,n} \in \mathcal{K}, \quad l_{m,n} \geq 0, \quad \forall m, n. \tag{6b}$$

Problem $\mathcal{P}3$ is a non-convex optimization problem due to the non-convexity of (5), which is in general difficult to solve. In the following two sections, we will propose algorithms for dedicated and probabilistic worker assignments and corresponding load allocations, respectively.

III. Dedicated Worker Assignment

In this section, we solve $\mathcal{P}3$ for dedicated worker assignment, where $\mathcal{K} = \{0, 1\}$. Given the assignment of workers, we first derive the optimal load allocation. Then the worker assignment can be transformed into a max-min allocation problem, for which NP-hardness is shown and two greedy algorithms are developed.

A. Optimal Load Allocation for a Given Worker Assignment

We first assume that the subset of workers that serve master m is given by $\Omega_m \subset \mathcal{N}$, and derive the optimal load allocation for master m, that minimizes the approximate completion time. The problem is formulated as:

$$\mathcal{P}4: \min_{\{l_{m,n},\ t_m\}} \quad t_m \tag{7a}$$

$$\text{s.t.} \quad L_m - \mathbb{E}[X_m(t_m)] \leq 0, \tag{7b}$$

$$l_{m,n} \geq 0, \forall n \in \Omega_m, \tag{7c}$$

where t_m is the approximate completion time of master m, and $\mathbb{E}[X_m(t_m)] = \sum_{n\in\Omega_m} l_{m,n}\left(1 - e^{-\frac{u_{m,n}}{l_{m,n}}(t_m - a_{m,n}l_{m,n})}\right)$.

Lemma 1. *Problem $\mathcal{P}4$ is a convex optimization problem.*

Define $\mathcal{W}_{-1}(x)$ as the lower branch of Lambert W function, where $x \leq -1$ and $\mathcal{W}_{-1}(xe^x) = x$. Let $\phi_{m,n} \triangleq \frac{1}{u_{m,n}}\left[-\mathcal{W}_{-1}(-e^{-u_{m,n}a_{m,n}-1}) - 1\right]$. By using the Lagrange multiplier method and solving the Karush-Kuhn-Tucker (KKT) conditions of $\mathcal{P}4$, we get the following theorem.

Theorem 1. *For master $m \in \mathcal{M}$, and a given subset of workers $\Omega_m \in \mathcal{N}$ serving this master, the optimal load allocation $l_{m,n}^*$ derived from $\mathcal{P}4$, and the corresponding minimum approximate completion time t_m^* are given by:*

$$l_{m,n}^* = \frac{L_m}{\phi_{m,n}\sum_{n\in\Omega_m}\frac{u_{m,n}}{1+u_{m,n}\phi_{m,n}}}, \ t_m^* = \frac{L_m}{\sum_{n\in\Omega_m}\frac{u_{m,n}}{1+u_{m,n}\phi_{m,n}}}.$$

B. Greedy Worker Assignment Algorithms

Define $V_m \triangleq \frac{1}{t_m^*}$, and let

$$v_{m,n} \triangleq \frac{u_{m,n}}{L_m(1 + u_{m,n}\phi_{m,n})}. \tag{8}$$

Based on Theorem 1, $V_m = \frac{1}{L_m}\sum_{n\in\Omega_m}\frac{u_{m,n}}{1+u_{m,n}\phi_{m,n}} = \sum_{n=1}^{N} k_{m,n}v_{m,n}$. Since $\min_{\{k_{m,n}\}}\max_{m\in\mathcal{M}} t_m^*$ is equivalent to $\max_{\{k_{m,n}\}}\min_{m\in\mathcal{M}} V_m$, we have the proposition below.

Proposition 1. *Problem $\mathcal{P}3$ is equivalent to*

$$\mathcal{P}5: \max_{\{k_{m,n}\}}\min_{m\in\mathcal{M}} \sum_{n=1}^{N} k_{m,n}v_{m,n} \tag{9a}$$

$$\text{s.t.} \ \sum_{m=1}^{M} k_{m,n} \leq 1, \ k_{m,n} \in \{0, 1\}, \ \forall m, n. \tag{9b}$$

Algorithm 1 Iterated Greedy Algorithm for Dedicated Worker Assignment

Input: $\Omega_m = \emptyset$, $V_m = 0$, and $\{v_{m,n}\}$ according to (8).
for $n = 1, ..., N$ **do** ▷ *Initialization*
  $m^* = \arg\max_{m \in \mathcal{M}} v_{m,n}$.
  $V_{m^*} = V_{m^*} + v_{m^*,n}$, $\Omega_{m^*} = \Omega_{m^*} \cup \{n\}$.
end for
while iteration is not terminated **do** ▷ *Main iteration*
  for $n = 1, ..., |\mathcal{N}|$ **do** ▷ *Insertion*
    Find master m_1 that worker n is serving.
    $m_2 = \arg\min_{m \in \mathcal{M}/\{m_1\}} V_m$.
    $V'_{m_1} = V_{m_1} - v_{m_1,n}$, $V'_{m_2} = V_{m_2} + v_{m_2,n}$.
    $V'_m = V_m, \forall m \in \mathcal{M}/\{m_1, m_2\}$.
    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 $n_1, n_2 = 1, ...|\mathcal{N}|$ **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 $m_1 \neq m_2$, $n_1 \neq n_2$, $v_{m_1,n_1} + v_{m_2,n_2} < v_{m_1,n_2} + v_{m_2,n_1}$, $V'_{m_1} > V_{\min}$, 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 a subset of $\mathcal{N}_s$ workers, and update V_m based on the current assignment. ▷ *Exploration*
  while $\mathcal{N}_s \neq \emptyset$ **do**
    $\{m^*, n^*\} = \arg\max_{m \in \mathcal{M}, n \in \mathcal{N}_s} v_{m,n}$.
    $V_{m^*} = V_{m^*} + v_{m^*,n^*}$.
    $\Omega_{m^*} = \Omega_{m^*} \cup \{n^*\}$, $\mathcal{N}_s = \mathcal{N}_s - \{n^*\}$.
  end while
end while

Algorithm 2 Simple Greedy Algorithm for Dedicated Worker Assignment

Input: $\mathcal{M}_0 = \{1, 2, ..., M\}$, $\mathcal{N}_0 = \{1, 2, ..., N\}$, $\Omega_m = \emptyset$, $V_m = 0$, and $\{v_{m,n}\}$ according to (8).
while $\mathcal{M}_0 \neq \emptyset$ **do** ▷ *Initialization*
  $\{m^*, n^*\} = \arg\max_{m \in \mathcal{M}_0, n \in \mathcal{N}_0} v_{m,n}$.
  $V_{m^*} = V_{m^*} + v_{m^*,n^*}$.
  $\Omega_m = \Omega_m \cup n^*$, $\mathcal{M}_0 = \mathcal{M}_0 - \{m^*\}$, $\mathcal{N}_0 = \mathcal{N}_0 - \{n^*\}$.
end while
while $\mathcal{N}_0 \neq \emptyset$ **do** ▷ *Main loop*
  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

Problem $\mathcal{P}5$ is a combinatorial optimization problem called *max-min allocation*, which is motivated by the fair allocation of indivisible goods [12]–[14]. Specifically, there are M agents and N items. Each item has a unique value for each agent, and can only be allocated to one agent. The goal is to maximize the minimum sum value of agents, by allocating items as fairly as possible. In our problem, *each master corresponds to an agent with sum value V_m, and each worker n can be considered as an item with value $v_{m,n}$ for master m*. The problem can be reduced to a NP-complete partitioning problem [15], when considering only 2 agents and that each item has identical value for both agents. Therefore, problem $\mathcal{P}5$ is NP-hard. An $O(N^\epsilon)$-approximation algorithm in time $N^{O(\frac{1}{\epsilon})}$ is proposed in [13] for max-min allocation, with $\epsilon \geq \frac{9 \log \log N}{\log N}$. Another polynomial-time algorithm is proposed in [14], guaranteeing $O(\frac{1}{M \log^3 M})$ approximation to the optimum. However, these algorithms are complex and difficult to implement. We propose two low-complexity greedy algorithms as follows.

An iterated greedy algorithm is proposed in Algorithm 1, which is inspired by [16], where a similar min-max fairness problem is investigated. In the initialization phase, each worker is assigned to the master for which its value $v_{m,n}$ is the highest. The main iteration has the following three phases:

1) *Insertion:* We remove each worker n from the current master m_1, and assign it to a master $m_2 \neq m_1$ with minimum sum value V_{m_2}. If the minimum sum value V_m is improved, let worker n serve master m_2. The complexity is $O(MN)$.

2) *Interchange:* We pick two workers n_1, n_2 that serve two masters m_1, m_2 respectively, and interchange their assignments. If the minimum sum $\min V_m$ is improved, and the overall system performance is improved (i.e., $v_{m_1,n_1} + v_{m_2,n_2} < v_{m_1,n_2} + v_{m_2,n_1}$), the interchange is kept. The complexity is $O(N^2)$. Note that the insertion and interchange are repeated for multiple times within each iteration, in order to obtain a local optimum.

3) *Exploration:* We randomly remove some workers from the current assignment, and allocate them in a greedy manner. This operation is an exploration which prevents the algorithm to be stuck in a local optimum. When the number of iterations reach a predefined maximum, or the performance does not improve any more, the main loop is terminated.

While Algorithm 1 still requires iterations to obtain a good assignment, Algorithm 2, which is inspired by the *largest-value-first* algorithm in [12], is even simpler with only one round. In a homogeneous case with $v_{1,n} = \cdots = v_{M,n}$, the algorithm finds an agent m with minimum sum value V_m, and assigns a remaining item with the largest value $v_{m,n}$. The algorithm provides $\frac{4}{3}$ approximation to the optimum. We extend the algorithm to our heterogeneous case. As shown in Algorithm 2, in the initialization phase, we find a master without any workers assigned, and allocate an available worker with largest contribution for it. In the main loop, we always select master m with the minimum sum value V_m, and allocate a remaining worker that has the maximum value $v_{m,n}$ for this master. The overall complexity of Algorithm 2 is $O(N^2)$.

IV. Probabilistic Worker Assignment

In this section, we solve $\mathcal{P}3$ for the probabilistic worker assignment, where $\mathcal{K} = [0, 1]$. The key challenge is that

constraint (3b) is non-convex. We decompose (3b) into the difference of convex functions, and adopt SCA method to jointly solve the worker assignment and load allocation problems.

Let $\boldsymbol{w} \triangleq \{l, k, t\}$, $g(\boldsymbol{w}) \triangleq -kl$, and $h(\boldsymbol{w}) \triangleq kle^{-\frac{ut}{l}}$. It is easy to see that $g^+(\boldsymbol{w}) \triangleq \frac{1}{2}\left(k^2+l^2\right)$, $g^-(\boldsymbol{w}) \triangleq \frac{1}{2}(k+l)^2$, $h^+(\boldsymbol{w}) \triangleq \frac{1}{2}\left(k+le^{-\frac{ut}{l}}\right)^2$, and $h^-(\boldsymbol{w}) \triangleq \frac{1}{2}\left(k^2+l^2e^{-\frac{2ut}{l}}\right)$ are all convex, and we have $g(\boldsymbol{w}) = g^+(\boldsymbol{w}) - g^-(\boldsymbol{w})$, $h(\boldsymbol{w}) = h^+(\boldsymbol{w}) - h^-(\boldsymbol{w})$. Given any two points $\boldsymbol{w}$, $\boldsymbol{z}$, the convex upper approximations of $g(\boldsymbol{w})$ and $h(\boldsymbol{w})$ can be obtained [17]: $\tilde{g}(\boldsymbol{w}, \boldsymbol{z}) \triangleq g^+(\boldsymbol{w}) - g^-(\boldsymbol{z}) - \nabla_{\boldsymbol{w}} g^-(\boldsymbol{z})^T(\boldsymbol{w}-\boldsymbol{z}) \geq g(\boldsymbol{w})$, $\tilde{h}(\boldsymbol{w}, \boldsymbol{z}) \triangleq h^+(\boldsymbol{w}) - h^-(\boldsymbol{z}) - \nabla_{\boldsymbol{w}} h^-(\boldsymbol{z})^T(\boldsymbol{w}-\boldsymbol{z}) \geq h(\boldsymbol{w})$.

Let subscript $\{m, n\}$ denote the variables, parameters and functions related to master m and worker n, e.g., $\boldsymbol{w}_{m,n} = \{l_{m,n}, k_{m,n}, t\}$, $h_{m,n}(\boldsymbol{w}_{m,n}) = l_{m,n}k_{m,n}e^{-\frac{u_{m,n}t}{l_{m,n}}}$. Let $\boldsymbol{w}_m \triangleq \{\boldsymbol{w}_{m,1}, ..., \boldsymbol{w}_{m,N}\}$, $\boldsymbol{z}_m \triangleq \{\boldsymbol{z}_{m,1}, ..., \boldsymbol{z}_{m,N}\}$.

Lemma 2. *The left-hand side of constraint* (5) *can be approximated by a convex function as follows:*

$$L_m - \mathbb{E}[X_m(t)] \leq L_m + \sum_{n=1}^{N} \left[\tilde{g}_{m,n}(\boldsymbol{w}_{m,n}, \boldsymbol{z}_{m,n}) + e^{u_{m,n}a_{m,n}}\tilde{h}_{m,n}(\boldsymbol{w}_{m,n}, \boldsymbol{z}_{m,n})\right] \triangleq \tilde{q}_m(\boldsymbol{w}_m, \boldsymbol{z}_m). \quad (10)$$

Let $\boldsymbol{z} \triangleq \{\boldsymbol{z}_1, ..., \boldsymbol{z}_M\}$ be a feasible point of $\mathcal{P}3$. The convex approximation to $\mathcal{P}3$ at point $\boldsymbol{z}$, defined as $\mathcal{P}(\boldsymbol{z})$, is given by:

$$\mathcal{P}(\boldsymbol{z}): \min_{\{l_{m,n}, k_{m,n}, t\}} \quad t \quad (11a)$$

$$\text{s.t.} \quad \tilde{q}_m(\boldsymbol{w}_m, \boldsymbol{z}_m) \leq 0, \quad \forall m, \quad (11b)$$

$$\text{Constraints (2c), (6b).}$$

A probabilistic worker assignment and load allocation algorithm is proposed in Algorithm 3 based on the SCA method. A diminishing step-size rule is adopted with decreasing ratio $\alpha \in (0, 1)$, guaranteeing the convergence of the SCA method [17], and in line 5, γ_r is the step-size in the rth iteration. Starting from a feasible point $\boldsymbol{z}_0$ of $\mathcal{P}3$, we iteratively solve convex optimization problems $\mathcal{P}(\boldsymbol{z}_r)$, in which constraint (5) is replaced by its upper convex approximation (11b). The iteration terminates when the solution is stationary (e.g., $\|\boldsymbol{w}_r - \boldsymbol{z}_r\|_2 \leq \epsilon$), and according to Theorem 2 in [17], the stationary solution is a local optimum.

A. Comparison of Dedicated and Probabilistic Assignments

We remark that the completion time of probabilistic worker assignment is a lower bound on what is achieved by dedicated worker assignment, since any feasible point of dedicated assignment is also feasible for probabilistic assignment. However, dedicated assignment simplifies the connections between workers and masters, and requires less communication for multicasting $\boldsymbol{x}_m$ and less storage at each worker. Moreover, the proposed dedicated assignment algorithms have lower computational complexity and are easier to implement.

V. Simulation Results

In this section, we evaluate the average task completion time of the proposed dedicated and probabilistic worker assignment algorithms, in both small-scale and large-scale scenarios.

Algorithm 3 SCA-based Probabilistic Worker Assignment and Load Allocation Algorithm

1: **Input**: find a feasible point of $\mathcal{P}3$, $\boldsymbol{z}_0$, 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 $\boldsymbol{w}_r$ of $\mathcal{P}(\boldsymbol{z}_r)$.
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**

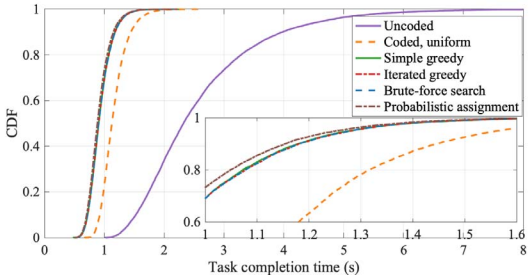

Fig. 2. The CDF of task completion time achieved by different worker assignment and load allocation algorithms with 2 masters and 20 workers.

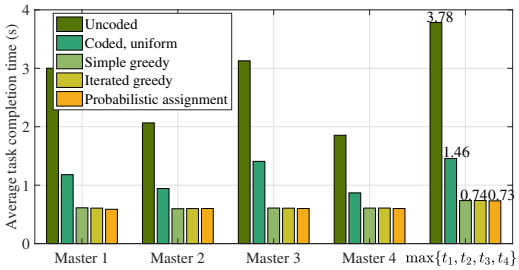

Fig. 3. Average task completion time achieved by different worker assignment and load allocation algorithms with 4 masters and 50 workers.

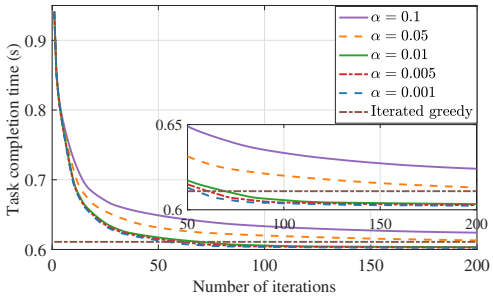

Fig. 4. Convergence of the SCA-based probabilistic worker assignment algorithm with 4 masters and 50 workers.

In the small-scale scenario, we consider $M = 2$ masters and $N = 20$ workers, and three benchmarks: 1) *Uncoded computing with uniform dedicated 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, each with $\frac{L_m M}{N}$ rows. 2) *Coded computing with uniform dedicated worker assignment* [5]: each master is assigned an equal number of $\frac{N}{M}$ workers, and the load is allocated according to Theorem 1. 3) *Brute-force search for dedicated worker assignment*: the oracle solution for dedicated worker assignment is obtained by searching all possible combinations, and the load is allocated according to Theorem 1. In the large-scale scenario, we consider $M = 4$ masters and $N = 50$

workers, and only use the first two benchmarks, due to the high complexity of the brute-force search.

The straggling parameter $u_{m,n}$ is randomly selected within $[1, 5]$ ms^{-1}, the shift parameter is set as $a_{m,n} = \frac{1}{u_{m,n}}$ ms, and $L_m = 10^5$, $\forall m$ [5]. In Algorithm 1, we randomly remove $\frac{N}{M}$ workers for each exploration. In Algorithm 3, we set the convergence criteria as $|1 - \frac{t'}{t}| < 10^{-6}$, decreasing ratio $\alpha = 10^{-3}$, and use CVX toolbox[3] to solve each convex approximation problem. We obtain the worker assignment and load allocation from the algorithms that minimize the approximate completion time. Then we carry out 10^5 Monte Carlo realizations and calculate the empirical cumulative distribution function (CDF) and the average of task completion time.

Fig. 2 shows the CDFs of the task completion time. The proposed greedy dedicated assignment and SCA-based probabilistic assignment algorithms outperform the uncoded and coded benchmarks with uniform assignment of dedicated workers. The CDFs achieved by iterated and greedy algorithms are very close, and both performances are close to the optimal brute-force search algorithm. Specifically, when the successful probability $\rho_s = 0.98$, the three dedicated assignment algorithms all achieve task completion time 1.40s. Probabilistic assignment further outperforms the dedicated assignment, which is consistent with the fact that it is a lower bound for dedicated assignment. When $\rho_s = 0.98$, probabilistic assignment achieves task completion time 1.38s.

Fig. 3 compares the average task completion time achieved by the proposed algorithms and benchmarks. The first four groups of bars show the average time each master needs to finish its own task using different algorithms. The fifth group of bars show the average task completion time of the system, which is what we aim to minimize, obtained by averaging the maximum time of each realization. From the fifth group of bars, we can see that all the proposed algorithms reduce the delay performance by more than 80% over uncoded benchmark, and more than 49% over coded benchmark. The performance gain is mainly achieved by taking into account the heterogeneity of the system. From the first four groups of bars, we can see that the average delay of each master achieved by our proposed algorithms are very close, indicating that the workers and loads are assigned in a balanced manner.

In Fig. 4, the impact of the decreasing ratio α on the convergence of SCA-based probabilistic assignment algorithm is evaluated, in the scenario with 4 masters and 50 workers. The decreasing ratio α decides the step-size γ_r, and thus the convergence rate of the SCA algorithm. We can see that by choosing a proper α, the proposed SCA algorithm can converge after 100 iterations, and outperforms the iterated greedy algorithm for dedicated worker assignment.

VI. Conclusions

We have considered a joint worker assignment and load allocation problem in a distributed computing system with heterogeneous computing servers, i.e., workers, and multiple master nodes competing for these workers. MDS coding has been adopted by the masters to mitigate the straggler effect, and both dedicated and probabilistic assignment algorithms have been proposed, in order to minimize the average task completion time. Simulation results show that the proposed algorithms can reduce the task completion time by 80% compared to uncoded task assignment, and 49% over an unbalanced coded scheme. While probabilistic assignment is more general, we have observed through simulations that the two have similar delay performances. We have noted that dedicated assignment has lower computational complexity and lower communication and storage requirements, beneficial for practical implementations. As future work, we plan to take communication delay into consideration, and develop decentralized algorithms.

[3]http://cvxr.com/cvx/

References

[1] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," [Online] Available: https://arxiv.org/abs/1812.02858, Dec. 2018.

[2] 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.

[3] 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.

[4] 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.

[5] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," [Online] Available: https://arxiv.org/abs/1701.05973, Jan. 2017.

[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] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," [Online] Available: https://arxiv.org/abs/1801.10292, May 2018.

[8] E. Ozfatura, D. Gündüz, and S. Ulukus, "Speeding up distributed gradient descent by utilizing non-persistent stragglers," [Online] Available: https://arxiv.org/abs/1808.02240, Aug. 2018.

[9] S. Li, S. M. M. Kalan, A. S. Avestimehr, and M. Soltanolkotabi, "Near-optimal straggler mitigation for distributed gradient methods," *IEEE Int. Parallel and Distributed Processing Symp. Workshops*, Vancouver, BC, Canada, May 2018, pp. 857-866.

[10] A. Behrouzi-Far and E. Soljanin, "On the effect of task-to-worker assignment in distributed computing systems with stragglers," *56th Annual Allerton Conf. on Commun., Control, and Comput.*, Monticello, IL, USA, Oct. 2018, pp. 560-566.

[11] M. Mohammodi Amiri, and D. Gündüz, "Computation scheduling for distributed machine learning with straggling workers," [Online] Available: https://arxiv.org/abs/1810.09992, Oct. 2018.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.