香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Distributed Gradient Descent Algorithm with Fractional Repetition Code

Li Haojun
李浩骏

A Thesis Submitted in Partial Fulfilment

of the Requirements for the Degree of

Master of Philosophy

in

Computer and Information Engineering

The Chinese University of Hong Kong, Shenzhen

December 2024

Thesis Assessment Committee

Professor Kenneth Shum (Thesis Supervisor)

Professor Yi Chen (Thesis Co-supervisor)

Professor Simon Pun (Committee Member)

Professor Kaiming Shen (Committee Member)

Professor Cheuk Ting Li (Examiner from CUHK)

Professor Linqi Song (External Examiner)

Abstract

of thesis entitled:

Distributed Gradient Descent Algorithm with Fractional Repetition Code

Submitted by Li Haojun

for the degree of Master of Philosophy

at The Chinese University of Hong Kong, Shenzhen in December 2024

Distributed Gradient Descent (DGD) algorithms are essential for large-scale machine learning, leveraging multiple worker nodes to accelerate training. Gradient descent is an optimization method that iteratively adjusts model parameters in the direction of the steepest descent of a loss function. However, the presence of stragglers —workers with delayed computations—poses a significant challenge in distributed systems, leading to inefficiencies.

This thesis addresses the straggler problem using a Fractional Repetition Codes (FRC) based gradient coding, a technique designed to mitigate straggler effects by distributing redundant data assignments among workers. By incorporating probabilistic models of worker response times, the study formulates the gradient approximation problem to minimize the expected recovery error. Key contributions include the development of optimal data assignment schemes for homogeneous systems and heuristic algorithms for heterogeneous systems, where worker capabilities vary.

Simulation results demonstrate that the proposed methods achieve near-optimal recovery error, significantly reducing training time in distributed machine learning tasks. These findings advance the design of robust and efficient DGD algorithms, enhancing their applicability to real-world heterogeneous computing clusters.

摘要

基于部分重复编码的分布式梯度下降算法

分布式梯度下降（DGD）算法对于大规模机器学习至关重要，它利用多个工作节点来加速训练。梯度下降是一种优化方法，迭代地在损失函数的最陡下降方向上调整模型参数。然而，滞后工人（计算延迟的工人）的存在在分布式系统中构成了一个重大挑战，导致效率低下。

本论文利用基于部分重复编码（FRC）的梯度编码技术解决了滞后问题，旨在通过在工人之间分配冗余数据来减轻滞后效应。通过引入工人响应时间的概率模型，研究提出了梯度近似问题，以最小化预期的恢复误差。主要贡献包括为同质系统开发最佳数据分配方案，以及为异质系统开发启发式算法，其中工人的能力各不相同。

仿真结果表明，所提出的方法实现了近乎最优的恢复误差，大大减少了分布式机器学习任务中的训练时间。这些发现推进了鲁棒且高效的 DGD 算法的设计，增强了它们在现实异构计算集群中的适用性。

Contents

List of Figures

List of Tables

Symbols and Acronyms

In general, we denote a scalar by an italic lowercase letter, a vector by an italic lowercase bold letter, and a matrix by an italic uppercase bold letter, respectively. For example, $a \in \mathbb{R}$, $\boldsymbol{v} \in \mathbb{R}^n$, and $\boldsymbol{M} \in \mathbb{R}^{p\times q}$, with any exceptions to be mentioned in the context case by case.

An all-one vector is written as $\mathbf{1}$. Specifically, an $1 \times k$ all-one vector is written as $\mathbf{1}_{1\times k}$. Specialized symbols and major acronyms are defined as follows.

Table 1: Summary of Notation

Notation	Description
n	Number of workers in the distributed system
k	Total number of data blocks
d	Number of data blocks assigned to each worker
s	Number of stragglers in a given straggler scenario
i	Index for data blocks
j	Index for workers
$\boldsymbol{g}_i$	Partial gradient of the i-th data block
p_j	Probability that worker j becomes a straggler
$\boldsymbol{B}$	Data assignment matrix
$\text{err}(\cdot)$	Recovery error
$\text{err}^{abs}(\cdot)$	Absence error
$\text{err}^{los}(\cdot)$	Loss error
$\boldsymbol{B}^{fr}$	Fractional repetition matrix
$\boldsymbol{B}^{bf}$	Balanced fractional repetition matrix
m	Number of worker groups in $\boldsymbol{B}^{fr}$
$\mathcal{U}_i$	Set of indices of workers assigned to compute the partial gradient of the i-th data block
$\boldsymbol{\mathcal{V}}$	Partition of $\{1, 2, \ldots, n\}$

Chapter 1

Introduction

Summary

This chapter provides an introduction to the background of distributed gradient descent and related work. Additionally, it outlines the contributions of the thesis and gives a brief sketch of its overall structure.

1.1 Background

Gradient descent is a common optimization algorithm used in machine learning for training models. It works by iteratively updating a set of model parameters in the direction of the steepest decrease of a cost function, namely the opposite direction of the gradient of the cost function. By applying gradient descent, machine learning models can learn from data, adjust their parameters, and improve their predictions through an iterative process.

When the size of the data set is large, we can expedite the learning process by distributing the calculations to a number of computing devices. We refer to the computing devices as “workers” and the centralized controller as “master”. The master node divides the data set into parts, with each worker storing one part and calculating the gradient for its assigned data. After completing the calculations, the workers return the results to the master node, which can then combine the results to recover the full gradient. Because the calculations are done in parallel, the learning speed can be increased multi-fold.

However, implementing distributed gradient descent in a network with varying worker speeds, communication capacities, and delays can diminish the benefits of parallelization. Some workers may be disconnected from the network and unresponsive. Additionally, some workers might be preempted by other urgent tasks, which slows down the computation speed. Workers who are slow to provide feedback or do not communicate with the master node are referred to as “stragglers”. A straggler can slow down the entire training job and is undesirable. With the prevalence of large models today, stragglers have become the norm rather than the exception, as training these models requires large-scale AI clusters [Jiang et al., 2024]. Addressing the problem of stragglers is crucial for making large-model training efficient.

This work was initially presented at the IEEE International Symposium on Information Theory (ISIT)[Li et al., 2023], showcasing preliminary results and methodologies. A more comprehensive version of this study has been submitted to the IEEE Transactions on Communications (TCom) for peer review.

1.2 Organization

The remainder of the thesis is organized as follows. We introduce some relate works of gradient coding in Chapter 2. In Chapter 3, we formulate the problem of approximate gradient coding with a probabilistic straggler model and introduce two crucial concepts relevant to this problem: absence error and loss error. In Chapter 4, we characterize the optimal data assignment pattern under our formulation when the workers are homogeneous and illustrate that in the heterogeneous case, the optimization problem we aim to solve is NP-hard and provide a heuristic algorithm. Simulation results and comparisons with other methods are discussed in Chapter 5.

Summary

This chapter provides an introduction to the background of distributed gradient descent and related work. Additionally, it outlines the contributions of the thesis and gives a brief sketch of its overall structure.

☐ **End of chapter.**

Chapter 2

Relate Work

Summary

This chapter presents some background information about Gradient Coding and lists some variants of it.

Tandon *et al.* [Tandon et al., 2017] first introduced the concept of gradient coding to mitigate the impact of stragglers by assigning more data to each worker and creating overlapping data subsets among them. This redundant computation ensures that even if some workers become stragglers, the master node can still gather enough information to recover the necessary gradient. The main challenge in gradient coding lies in designing the data assignment among workers and the coding scheme each worker uses to encode its computed results before returning them to the master node. In the literature, gradient coding solutions are generally classified into two categories: exact gradient coding and approximated gradient coding. They differ primarily in their tolerance for stragglers and the accuracy of the gradient estimates.

We illustrate the central concept of gradient coding through the example in Figure 2.1. Consider the scenario of employing synchronous Gradient Descent (SGD) across three workers (W_1, W_2, W_3). The initial conventional system is presented in the upper diagram and functions as follows: Each of the three workers possesses distinct segments of locally stored labeled data ($\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$), while all of them share the prevailing model. Worker 1 calculates the gradient of the model using examples from partition $\mathcal{D}_1$, denoted as $\boldsymbol{g}_1$. Similarly, Workers 2 and 3 compute $\boldsymbol{g}_2$ and $\boldsymbol{g}_3$, respectively. These three gradient vectors are then transmitted to master, which aggregates them to form the complete gradient $\boldsymbol{g}_1 + \boldsymbol{g}_2 + \boldsymbol{g}_3$. Subsequently, the master updates the model through a gradient step. The updated model is communicated back to the workers, and the system progresses to the subsequent round.

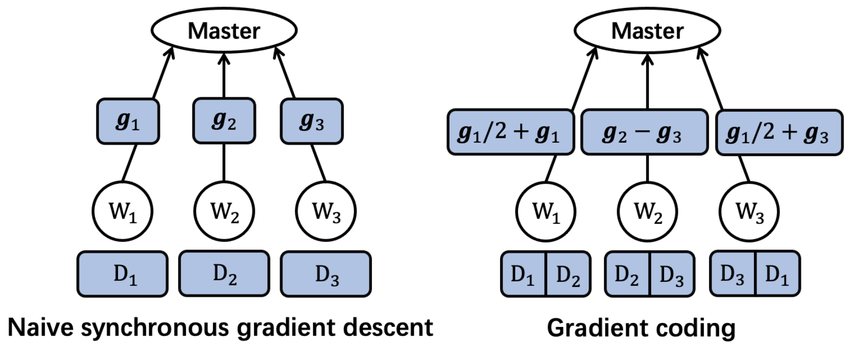


Figure 2.1: An example of Gradient Coding. The vector $\boldsymbol{g}_1 + \boldsymbol{g}_2 + \boldsymbol{g}_3$ lies within the subspace spanned by any two of the vectors $\boldsymbol{g}_1/2 + \boldsymbol{g}_2$, $\boldsymbol{g}_2 - \boldsymbol{g}_3$, and $\boldsymbol{g}_1/2 + \boldsymbol{g}_3$.

More generally, gradient coding considers a system with one master and n workers. The data is partitioned into k parts, and each worker is assigned several different parts. The master wants to obtain the gradient exactly, even if there s non-responding workers. By employing a cyclic repetition scheme, the authors of [Tandon et al., 2017] prove that exact recovery of the complete gradient is possible if each worker stores $s + 1$ data parts (out of k parts), where s denotes the number of stragglers.

In the literature, gradient coding solutions are generally classified into two categories: exact gradient coding and approximated gradient coding. They differ primarily in their tolerance for stragglers and the accuracy of the gradient estimates.

2.1 Exact Gradient Coding

The exact gradient coding aims to enable the master node to accurately reconstruct the full gradient. Research in this area typically assumes a fixed and known number of stragglers, denoted as s, among n workers. Various coding schemes have been developed to tolerate up to s stragglers, allowing the master node to recover the exact gradient from the feedback of any $n-s$ non-straggling workers. Notable studies have introduced schemes based on fractional repetition codes [Tandon et al., 2017], cyclic MDS codes [Raviv et al., 2020], and Reed Solomon codes [Halbawi et al., 2018]. These studies consistently show that as the number of stragglers increases, more computational redundancy becomes necessary. This line of work was further extended in [Cao et al., 2021; Kadhe et al., 2020; Ozfatura et al., 2019, 2020; Ye and Abbe, 2018], which explores the trade-offs between computational redundancy, straggler tolerance, and communication costs, the latter referring to the overhead involved in transmitting messages between the master node and the workers. It has been shown that computational redundancy can be reduced at the expense of increased communication costs. Furthermore, heterogeneity-sensitive gradient coding has been discussed in [Buyukates et al., 2022; Charalambides et al., 2024a; Jahani-Nezhad and Maddah-Ali, 2021; Wang et al., 2022], addressing scenarios where workers have different computing or communication capabilities. We summarize the exact gradient coding schemes in Table 2.1.

However, in many practical scenarios, the real-time number of stragglers is variable and often unpredictable. For instance, consider a situation where the master node sets a fixed deadline for worker feedback. Those who fail to return their results on time are regarded as stragglers. The reasons for straggling can be transient. For example, a worker's connection to the master may experience traffic congestion, incurring a higher transmission delay. Alternatively, a worker might prioritize other urgent tasks over gradient computation, leading to late feedback to the master. As a result, coding schemes designed for a fixed s straggler scenario may waste computational and communication resources when fewer stragglers are present, despite their ability to guarantee exact gradient recovery. Conversely, if the actual number of stragglers exceeds s, most of the previously mentioned coding schemes fail to provide guaranteed error bounds.

2.2 Approximate Gradient Coding

The above limitation highlights the need for approximated gradient coding, which focuses on recovering gradient estimates rather than exact values. Another rationale for approximated gradient coding is that, in distributed machine learning applications, approximated gradients can often be sufficient. In gradient descent, particularly during the initial stages of training, an exact calculation of the full gradient is not always essential [Bottou, 2010]. A deliberate and slight deviation from the true gradient can actually facilitate exploration of the parameter space. Moreover, recent work in [Charalambides et al., 2024b] suggests that concepts from randomized linear algebra can be used to introduce a controlled random error, thereby accelerating the training process. Motivated by these benefits, this thesis focuses on approximate gradient coding. Our methods offer a trade-off between estimation error and training time. Achieving zero estimation error, i.e., exact gradient recovery, is possible, but it comes at the cost of longer training durations.

Existing approaches to approximated gradient coding include [Bitar et al., 2020; Charles et al., 2017; Glasgow and Wootters, 2021; Jahani-Nezhad and Maddah-Ali, 2021; Johri et al., 2021; Raviv et al., 2020; Sakorikar and Wang, 2022; Sarmasarkar et al., 2023; Wang et al., 2019a,b], which are summarized in Table 2.1. The goal is to provide a robust solution for a random number of stragglers when we allow for some reconstruction error in the gradient. The works in [Raviv et al., 2020; Sakorikar and

Wang, 2022; Sarmasarkar et al., 2023] introduce schemes based on expander graphs, balanced incomplete block designs, and cyclic assignments, respectively, which guarantee worst-case recovery error under the s-out-of-n straggler model. The authors in [Wang et al., 2019b] propose a scheme using Batched Raptor codes, which offers a probabilistic upper bound on the recovery error within the s-out-of-n straggler model. However, the worst-case scenario in the s-out-of-n model may occur with only a small probability, and designing for the worst case in this sense may not be optimal when we have information about which workers are fast and which are slow.

Table 2.1: Classification of existing works.

	Exact gradient recovery	Approximate gradient recovery
Homogeneous workers	• [Tandon et al., 2017] • [Ye and Abbe, 2018] • [Ozfatura et al., 2019] • [Halbawi et al., 2018] • [Raviv et al., 2020] • [Ozfatura et al., 2020] • [Kadhe et al., 2020] • [Cao et al., 2021] • [Charalambides et al., 2024a]	• [Raviv et al., 2020] • [Sakorikar and Wang, 2022] • [Sarmasarkar et al., 2023] • [Wang et al., 2019b] • [Charles et al., 2017] • [Wang et al., 2019a] • [Bitar et al., 2020] • [Glasgow and Wootters, 2021] • This thesis
Heterogeneous workers	• [Wang et al., 2022] • [Buyukates et al., 2022] • [Charalambides et al., 2024a] • [Jahani-Nezhad and Maddah-Ali, 2021]	• [Jahani-Nezhad and Maddah-Ali, 2021] • [Johri et al., 2021] • [Wang et al., 2022] • This thesis

Summary

This chapter presents some background information about Gradient Coding and lists some variants of it.

□ **End of chapter.**

Chapter 3

Problem Formulation

Summary

This chapter presents the problem of approximate gradient coding, introduces relevant concepts, discusses optimal data assignment patterns under different worker conditions, and presents simulation results and comparisons with other methods.

3.1 Problem Formulation

The master node wants to train a model using a data set $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots\}$ by minimizing the loss function in the form

$$\sum_{(\boldsymbol{x},y)\in\mathcal{D}} \ell(\boldsymbol{x}, y; \boldsymbol{\theta}), \tag{3.1}$$

where $\boldsymbol{\theta}$ denotes the vector of parameters, and $\ell(\boldsymbol{x}, y; \boldsymbol{\theta})$ is a function of the sample $(\boldsymbol{x}, y)$ and the parameter vector $\boldsymbol{\theta}$. By using the method of gradient descent, the master node computes the gradient

$$\boldsymbol{g}(\boldsymbol{\theta}^{(t)}) := \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \nabla_{\boldsymbol{\theta}}\ell(\boldsymbol{x},y;\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}^{(t)}}, \tag{3.2}$$

where $\nabla_{\boldsymbol{\theta}}$ is the gradient with respect to the variables in $\boldsymbol{\theta}$, and $\boldsymbol{\theta}^{(t)}$ is the value of the parameter vector at training iteration t.

To speed up the computation of the gradient vector, the master node distributes the data to n worker nodes, so that each worker node computes the gradient pertaining to the assigned block of data. We index the workers by $\mathcal{N} = \{1, \dots, n\}$. We assume that the storage of the worker nodes is limited, and each worker can store no more than a fraction, say d/k, of the total amount of data, where d and k are relatively prime positive integers. The whole data set is divided into k non-overlapping subsets $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_k$ of the same size. We refer to a data subset as a *data block*. Each worker node is assigned with at most d distinct data blocks. The system parameter d represents the workload of the workers. In the literature, it is sometimes termed as the *redundancy*, or *computational overhead*, when it quantifies the additional tasks a worker has to perform in order to mitigate the effect of stragglers.

For $i = 1, 2, \dots, k$, denote the *partial gradient* of the loss function evaluated over $\mathcal{D}_j$ at the t-th iteration by

$$\boldsymbol{g}_i(\boldsymbol{\theta}^{(t)}) := \sum_{(\boldsymbol{x},y)\in\mathcal{D}_i} \nabla_{\boldsymbol{\theta}}\ell(\boldsymbol{x},y;\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}^{(t)}}. \tag{3.3}$$

It is easy to check that $\boldsymbol{g}^{(t)} = \boldsymbol{g}_1^{(t)} + \boldsymbol{g}_2^{(t)} + \cdots + \boldsymbol{g}_k^{(t)}$. We will regard the full gradient $\boldsymbol{g}^{(t)}$, and the partial gradients $\boldsymbol{g}_i^{(t)}$ as row vectors, with dimension equal to the number of parameters in $\boldsymbol{\theta}$. At iteration t, each worker computes the partial gradients of its assigned data blocks, and sends a linear combination of them to the master.

A worker may not be able to complete the task of computing the gradient of the assigned data block before the deadline. In this case we say that the worker is a *straggler* in iteration t. We will denote the number of stragglers in an iteration by a variable s. The coefficients for the linear combination at the workers must be carefully selected so that the master can approximately recover the full gradient, no matter which workers become the stragglers. Once the master recovers the gradient and updates the parameter $\boldsymbol{\theta}$, it broadcasts the latest $\boldsymbol{\theta}$ to the workers for the next round of iteration.

We give a more formal representation of the problem below. For simplicity, we will drop the iteration index t and will not write $\boldsymbol{\theta}^{(t)}$, because they are fixed within an iteration. We stack the partial gradients together and form a $k \times \dim(\boldsymbol{\theta})$ matrix, given below

$$\bar{\boldsymbol{g}} := \begin{pmatrix} \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ \vdots \\ \boldsymbol{g}_k \end{pmatrix}.$$

We will also view $\bar{\boldsymbol{g}}$ as a $k \times 1$ column vector, with each entry being a row vector.

The gradient coding can be represented by a *data assignment matrix* $\boldsymbol{B} \in \{0, 1\}^{n \times k}$ and a *local combination coefficient matrix* $\boldsymbol{C} \in \mathbb{R}^{n \times k}$. Each row of the matrix corresponds to a worker and each column corresponds to a data block. The matrix $\boldsymbol{B}$ is a binary matrix, where the entry in row j and column i is one if the i-th data block is assigned to the j-th worker. Each row of $\boldsymbol{B}$ contains at most d ones. The j-th row of $\boldsymbol{C}$ contains the coefficients for the linear combination of locally computed partial gradients at worker j. Let supp$(\cdot)$ denote the support of a matrix or vector, i.e., the set of indices of nonzero components. We require that supp$(\boldsymbol{C}) \subseteq$ supp$(\boldsymbol{B})$. In other words, the (j, i)-th entry of $\boldsymbol{C}$ equals zero if the (j, i)-th entry of $\boldsymbol{B}$ equals zero. Then the gradient information transmitted from worker j to the master is the j-th component (i.e. row vector) of $\boldsymbol{C}\bar{\boldsymbol{g}}$.

With the presence of s stragglers, the master can only receive $n - s$ components of $\boldsymbol{C}\bar{\boldsymbol{g}}$. Let $\mathcal{S} \subseteq \mathcal{N}$ denote the index set of straggler workers. Define $\mathcal{S}^{\mathrm{c}} = \mathcal{N} \setminus \mathcal{S}$. The master estimates the full gradient by linearly combing the received results, namely, $\boldsymbol{a}\boldsymbol{C}\bar{\boldsymbol{g}}$, where $\boldsymbol{a} \in \mathbb{R}^n$ includes the coefficients for the linear combination, and $\mathrm{supp}(\boldsymbol{a}) \subseteq \mathcal{S}^{\mathrm{c}}$. Given $\boldsymbol{C}$ and $\bar{\boldsymbol{g}}$, the optimal $\boldsymbol{a}$ can be found by minimizing the squared error:

$$\min_{\substack{\boldsymbol{a} \\ \mathrm{supp}(\boldsymbol{a}) \subseteq \mathcal{S}^{\mathrm{c}}}} \|\mathbf{1}_{1\times k}\bar{\boldsymbol{g}} - \boldsymbol{a}\boldsymbol{C}\bar{\boldsymbol{g}}\|^2. \tag{3.4}$$

However, in practice, the master has no knowledge of $\bar{\boldsymbol{g}}$. Consequently, it chooses $\boldsymbol{a}$ by solving the following problem:

$$\boldsymbol{a}^* = \arg \min_{\substack{\boldsymbol{a} \\ \mathrm{supp}(\boldsymbol{a}) \subseteq \mathcal{S}^{\mathrm{c}}}} \|\mathbf{1}_{1\times k} - \boldsymbol{a}\boldsymbol{C}\|^2. \tag{3.5}$$

Definition 1. Given a data assignment matrix $\boldsymbol{B}$, a local combination coefficient matrix $\boldsymbol{C}$ that satisfies $\mathrm{supp}(\boldsymbol{C}) \subseteq \mathrm{supp}(\boldsymbol{B})$, and a straggler set $\mathcal{S}$, we define the *recovery error* as

$$\mathrm{err}(\boldsymbol{B}, \boldsymbol{C}, \mathcal{S}) := \min_{\substack{\boldsymbol{a} \\ \mathrm{supp}(\boldsymbol{a}) \subseteq \mathcal{S}^{\mathrm{c}}}} \|\mathbf{1}_{1\times k} - \boldsymbol{a}\boldsymbol{C}\|^2. \tag{3.6}$$

Example 1. Consider a system comprising $n = 4$ workers and $k = 4$ data blocks, with each worker storing $d = 2$ data blocks. A feasible $\boldsymbol{B}$ is provided as follows

$$\boldsymbol{B} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}. \tag{3.7}$$

It indicates that worker 1 is assigned data blocks 1 and 2, worker 2 is assigned data blocks 2 and 3, and so forth. We can simply set $\boldsymbol{C} = \boldsymbol{B}$. Suppose workers 3 and 4 become stragglers, i.e., $\mathcal{S} = \{3, 4\}$. By solving (3.5), the optimal $\boldsymbol{a}^*$ is determined to be $(\frac{2}{3}, \frac{2}{3}, 0, 0)$. Note that $\text{supp}(\boldsymbol{a}^*) = \mathcal{S}^c$. Then we can have $\boldsymbol{a}^*\boldsymbol{C} = (\frac{2}{3}, \frac{4}{3}, \frac{2}{3}, 0)$, and the recovery error is:

$$\text{err}(\boldsymbol{B}, \boldsymbol{C}, \mathcal{S}) = \left\|(\frac{2}{3}, \frac{4}{3}, \frac{2}{3}, 0) - (1, 1, 1, 1)\right\|^2 \tag{3.8}$$

$$= (\frac{1}{3})^2 + (\frac{1}{3})^2 + (\frac{1}{3})^2 + 1^2 = \frac{4}{3}. \tag{3.9}$$

As noted before, the number of stragglers is a random variable, and which workers become stragglers is also random. We consider a probability model on all possible straggler scenarios. Let $\mathcal{P}(\mathcal{N})$ denote the power set of $\mathcal{N}$. Each straggler scenario corresponds to an element $\mathcal{S} \in \mathcal{P}(\mathcal{N})$. We assume that each worker becoming a straggler is independent, and the probability that worker j becomes a straggler is p_j for $j \in \mathcal{N}$. Without loss of generality, assume $0 \leq p_j < 1$. Therefore, the probability associated with a straggler scenario defined by $\mathcal{S}$ can be expressed as:

$$\text{Pr}(\mathcal{S}) := \prod_{j \in \mathcal{S}^c} (1 - p_j) \prod_{j \in \mathcal{S}} p_j. \tag{3.10}$$

In this thesis, our goal is to construct an optimal pair of $\boldsymbol{B}$ and $\boldsymbol{C}$ that minimizes the expected recovery error, which is given below

$$\sum_{\mathcal{S} \in \mathcal{P}(\mathcal{N})} \text{Pr}(\mathcal{S}) \cdot \text{err}(\boldsymbol{B}, \boldsymbol{C}, \mathcal{S}) = \mathbb{E}_{\mathcal{S}}[\text{err}(\boldsymbol{B}, \boldsymbol{C}, \mathcal{S})]. \tag{3.11}$$

Here we use $\mathbb{E}_{\mathcal{S}}[\cdot]$ to represent the average over $\mathcal{S} \in \mathcal{P}(\mathcal{N})$.

3.2 Absence Error and Loss Error

In Example 1, it is observed that data block 4 is assigned to workers 3 and 4, who happen to be stragglers. Consequently, the fourth component in the vector $\boldsymbol{a}^*\boldsymbol{C}$ is 0, resulting in an increase of 1 in the recovery error. This observation suggests that if a data block is not computed by any of the non-stragglers, it adds 1 to the recovery error. On the other hand, if a data block is computed by at least one non-straggler, for instance, data block 1 computed by worker 1, then the corresponding component in $\boldsymbol{a}^*\boldsymbol{C}$ is not necessary to be 0, and the error arises from its deviation from 1. We distinguish these two cases and define two types of error, namely *absence error* and *loss error* below.

Given a data assignment matrix $\boldsymbol{B}$, for each data block $i = 1, 2, \dots, k$, let $\mathcal{U}_i$ denote the set of indices of the workers assigned to compute the i-th data block, i.e., the index set of the nonzero entries in the i-th column of $\boldsymbol{B}$. We call $\mathcal{U}_i$ the *worker assignment set*. To simplify notation, we do not explicitly write out the dependency of $\mathcal{U}_i$ on $\boldsymbol{B}$.

Definition 2. Given a data assignment matrix $\boldsymbol{B}$, a local combination coefficient matrix $\boldsymbol{C}$ that satisfies supp$(\boldsymbol{C}) \subseteq$ supp$(\boldsymbol{B})$, and a straggler set $\mathcal{S}$, the *absence error* is defined as

$$\text{err}^{\text{abs}}(\boldsymbol{B}, \mathcal{S}) := \sum_{i=1}^{k} \mathbb{1}(\mathcal{U}_i \subseteq \mathcal{S}), \tag{3.12}$$

where $\mathbb{1}(\cdot)$ is the indicator function.

We define the *loss error* as

$$\text{err}^{\text{los}}(\boldsymbol{B}, \boldsymbol{C}, \mathcal{S}) := \text{err}(\boldsymbol{B}, \boldsymbol{C}, \mathcal{S}) - \text{err}^{\text{abs}}(\boldsymbol{B}, \mathcal{S}). \tag{3.13}$$

The recovery error err$(\boldsymbol{B}, \boldsymbol{C}, \mathcal{S})$ is decomposed into the absence error and the loss error. We use Example 1 to illustrate the two types of error. For data block $i = 1, 2, 3, 4$, we have

$$\mathcal{U}_1 = \{1, 4\},\ \mathcal{U}_2 = \{1, 2\},\ \mathcal{U}_3 = \{2, 3\},\ \mathcal{U}_4 = \{3, 4\}. \tag{3.14}$$

Since $\mathcal{U}_1 \nsubseteq \mathcal{S}, \mathcal{U}_2 \nsubseteq \mathcal{S}, \mathcal{U}_3 \nsubseteq \mathcal{S}$ and $\mathcal{U}_4 \subseteq \mathcal{S}$, the absence error is 1. The loss error is $1/3$.

It is worth mentioning that the absence error is determined by $\boldsymbol{B}$ and $\mathcal{S}$, rather than by $\boldsymbol{C}$. Moreover, for any $\mathcal{S} \in \mathcal{P}(\mathcal{N})$, we have

$$\mathrm{err}(\boldsymbol{B}, \boldsymbol{C}, \mathcal{S}) \geq \mathrm{err}^{\mathrm{abs}}(\boldsymbol{B}, \mathcal{S}). \tag{3.15}$$

The absence error serves as a lower bound on the recovery error. In the rest of this section, we present some general findings concerning the expected absence error, i.e., $\mathbb{E}_{\mathcal{S}}[\mathrm{err}^{\mathrm{abs}}(\boldsymbol{B}, \mathcal{S})]$.

Lemma 1. The expected absence error is given by

$$\mathbb{E}_{\mathcal{S}}[\mathrm{err}^{\mathrm{abs}}(\boldsymbol{B}, \mathcal{S})] = \sum_{i=1}^{k} \prod_{j \in \mathcal{U}_i} p_j. \tag{3.16}$$

Proof. We expand left-hand side of the equality as

$$\mathbb{E}_{\mathcal{S}}[\mathrm{err}^{\mathrm{abs}}(\boldsymbol{B}, \mathcal{S})] = \sum_{\mathcal{S} \in \mathcal{P}(\mathcal{N})} \Pr(\mathcal{S}) \sum_{i=1}^{k} \mathbb{1}(\mathcal{U}_i \subseteq \mathcal{S}) \tag{3.17}$$

$$= \sum_{i=1}^{k} \sum_{\mathcal{S} \in \mathcal{P}(\mathcal{N})} \Pr(\mathcal{S}) \mathbb{1}(\mathcal{U}_i \subseteq \mathcal{S}). \tag{3.18}$$

The equality (3.17) holds in accordance with (3.12). We observe that the event $\{\mathcal{U}_i \subseteq \mathcal{S}\}$ happens when the workers with indices in $\mathcal{U}_i$ are among the stragglers in $\mathcal{S}$. The inner sum in (3.18) is equal to the probability that the workers in $\mathcal{U}_i$ are stragglers, and this happens with probability $\prod_{j \in \mathcal{U}_i} p_j$. Therefore, the left-hand side of (3.16) equals

$$\sum_{i=1}^{k} \prod_{j \in \mathcal{U}_i} p_j. \tag{3.19}$$

$\square$

The minimization of absence error is intimately related to the theory of majorization [Marshall et al., 2011]. We recall some terminology from majorization theory below.

Given a real vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_k)$, we order the components in a non-increasing order and define $x_{[1]}$ as the largest value among the components of $\boldsymbol{x}$, $x_{[2]}$ as the second largest, and so on, so that $(x_{[1]}, x_{[2]}, \ldots, x_{[k]})$ is a permutation of vector $\boldsymbol{x}$ satisfying

$$x_{[1]} \geq x_{[2]} \geq \cdots \geq x_{[k]}. \tag{3.20}$$

We say that a vector $\boldsymbol{x} \in \mathbb{R}^k$ is *weakly majorized by* $\boldsymbol{y} \in \mathbb{R}^k$ if

$$\sum_{i=1}^{j} x_{[i]} \leq \sum_{i=1}^{j} y_{[i]} \tag{3.21}$$

for $j = 1, 2, \ldots, k$. Furthermore, if vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ satisfy the additional condition that

$$\sum_{i=1}^{k} x_i = \sum_{i=1}^{k} y_i, \tag{3.22}$$

then we say that $\boldsymbol{x}$ is *majorized* by $\boldsymbol{y}$. We will use the notation $\boldsymbol{x} \prec_w \boldsymbol{y}$ if $\boldsymbol{x}$ is weakly majorized by $\boldsymbol{y}$, and $\boldsymbol{x} \prec \boldsymbol{y}$ if $\boldsymbol{x}$ is majorized by $\boldsymbol{y}$.

We will also need a multiplicative version of majorization. If $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors with positive components, then we say that $\boldsymbol{x}$ is *weakly log-majorized by* $\boldsymbol{y}$ if

$$\prod_{i=1}^{j} x_{[i]} \leq \prod_{i=1}^{j} y_{[i]} \tag{3.23}$$

for $j = 1, 2, \ldots, k$.

We state a major result for weak majorization below.

Theorem 2 ([Marshall et al., 2011] Theorem 3.C.1.b)**.** If $g : \mathbb{R} \to \mathbb{R}$ is a convex and increasing function, then for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^k$, we have

$$\boldsymbol{x} \prec_w \boldsymbol{y} \Rightarrow \sum_{i=1}^{k} g(x_i) \leq \sum_{i=1}^{k} g(y_i). \tag{3.24}$$

Applying this theorem to absence error, we have the following theorem.

Theorem 3. For $j = 1, 2, \dots, n$, let p_j be the straggling probability of worker j. Consider two data assignment matrices $\boldsymbol{B}$ and $\boldsymbol{B}'$. For $i = 1, 2, \dots, k$, we let $\mathcal{U}_i$ (resp. $\mathcal{U}_i'$) be the index set of the nonzero entries in the i-th column of $\boldsymbol{B}$ (resp. $\boldsymbol{B}'$). Define two vectors $\boldsymbol{x} = (x_1, \dots, x_k)$ and $\boldsymbol{y} = (y_1, \dots, y_k)$ of dimension k by

$$x_i := \prod_{j \in \mathcal{U}_i} p_j, \qquad y_i := \prod_{j \in \mathcal{U}_i'} p_j, \tag{3.25}$$

for $i = 1, 2, \dots, k$. Then, if $\boldsymbol{x}$ is weakly log-majorized by $\boldsymbol{y}$, we have

$$\mathbb{E}_{\mathcal{S}}[\text{err}^{\text{abs}}(\boldsymbol{B}, \mathcal{S})] \leq \mathbb{E}_{\mathcal{S}}[\text{err}^{\text{abs}}(\boldsymbol{B}', \mathcal{S})] \tag{3.26}$$

Proof. From Lemma 1, the expected value of the absence error pertaining to matrix $\boldsymbol{B}$ is given by $\sum_{i=1}^{k} \prod_{j \in \mathcal{U}_i} p_j$. Likewise, the expected value of the absence error of $\boldsymbol{B}'$ is given by a similar expression with $\mathcal{U}_i$ replaced by $\mathcal{U}_i'$.

We can replace the probabilities p_j by $\log p_j$, and write the absence error as

$$\sum_{i=1}^{k} \prod_{j \in \mathcal{U}_i} p_j = \sum_{i=1}^{k} \exp\Big(\sum_{j \in \mathcal{U}_i} (\log p_j) \Big). \tag{3.27}$$

We apply Theorem 2 with $g(x)$ being the exponential function e^x, which is a convex and increasing function. The rest of the proof is simple verification. The main step is to observe that the vector $(\sum_{j \in \mathcal{U}_i} \log p_j)_{i=1}^{k}$ is weakly majorized by the vector

$(\sum_{j\in\mathcal{U}'_i}\log p_j)_{i=1}^k$ if and only if $\boldsymbol{x}$ is weakly log-majorized by $\boldsymbol{y}$. □

Example 2. We continue with Example 1, and suppose that the straggling probabilities of workers 1 to 4 are

$$p_1 = 0.1,\ p_2 = 0.2,\ p_3 = 0.3,\ p_4 = 0.4. \tag{3.28}$$

Considering the data assignment matrix $\boldsymbol{B}$ as given in (3.7), by Lemma 1, the expected absence error is

$$0.1\times 0.4 + 0.1\times 0.2 + 0.2\times 0.3 + 0.3\times 0.4 = 0.24. \tag{3.29}$$

Consider another data assignment matrix

$$\boldsymbol{B}' = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}. \tag{3.30}$$

The expected absence error becomes

$$0.1\times 0.4 + 0.2\times 0.3 + 0.2\times 0.3 + 0.1\times 0.4 = 0.2, \tag{3.31}$$

which is smaller than that under $\boldsymbol{B}$.

Define vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ for $\boldsymbol{B}$ and $\boldsymbol{B}'$ correspondingly, as per (3.25). Then

$$\boldsymbol{x} = (0.04, 0.06, 0.06, 0.04) \tag{3.32}$$

$$\boldsymbol{y} = (0.04, 0.02, 0.06, 0.12). \tag{3.33}$$

It can be checked that $\boldsymbol{x}$ is weakly log-majorized by $\boldsymbol{y}$.

The next two lemmas are easy consequences that will be used later.

Lemma 4. Consider the set of data assignment matrices that have at most d ones in each row and suppose $nd \geq k$. Then a matrix $\boldsymbol{B}$ that minimizes the expected absence error in this set must have at least one "1" in each column.

Proof. We prove by contradiction. Suppose $\boldsymbol{B}$ achieves the minimum expected absence error and there exits some columns in $\boldsymbol{B}$ whose entries are all zero, say column i_0. We consider two cases.

Case 1. There are some rows in $\boldsymbol{B}$ that contain less than d ones, say the j_0-th row. We construct a data assignment matrix $\boldsymbol{B}'$ by letting $\boldsymbol{B}' = \boldsymbol{B}$ except for the (j_0, i_0)-th entry, which is set to one. Let $\mathcal{U}_i$ and $\mathcal{U}'_i, i = 1, \dots, k$ be the worker assignment sets of $\boldsymbol{B}$ and $\boldsymbol{B}'$, respectively. We have $\mathcal{U}_i = \mathcal{U}'_i$ for $i \neq i_0$, $\mathcal{U}_{i_0} = \emptyset$ and $\mathcal{U}'_{i_0} = \{j_0\}$. According to Lemma 1,

$$\begin{aligned}
\mathbb{E}_{\mathcal{S}}[\mathrm{err}^{\mathrm{abs}}(\boldsymbol{B}, \mathcal{S})] &= \sum_{i \neq i_0} \prod_{j \in \mathcal{U}_i} p_j + 1 && (3.34)\\
&> \sum_{i \neq i_0} \prod_{j \in \mathcal{U}'_i} p_j + p_{j_0} && (3.35)\\
&= \mathbb{E}_{\mathcal{S}}[\mathrm{err}^{\mathrm{abs}}(\boldsymbol{B}', \mathcal{S})]. && (3.36)
\end{aligned}$$

It contradicts that $\boldsymbol{B}$ achieves the minimum expected absence error.

Case 2. All rows of $\boldsymbol{B}$ have exactly d ones. Then $\boldsymbol{B}$ has a total of nd ones. Since $nd \geq k$ and column j_0 in $\boldsymbol{B}$ has no ones, there exits a column in $\boldsymbol{B}$ that contains two or more ones, say column i_1. That is, $|\mathcal{U}_{i_1}| \geq 2$. Suppose $j_0 \in \mathcal{U}_{i_1}$. We construct a data assignment matrix $\boldsymbol{B}'$ by letting $\boldsymbol{B}' = \boldsymbol{B}$ but changing the entries at positions (j_0, i_0) and (j_0, i_1) to one and zero, respectively. It can be verified that each row of $\boldsymbol{B}'$ has d ones. Moreover, $\mathcal{U}_i = \mathcal{U}'_i$ for $i \neq i_0, i_1, \mathcal{U}_{i_0} = \emptyset, \mathcal{U}'_{i_0} = \{j_0\}$, and $\mathcal{U}_{i_1} \setminus \{j_0\} = \mathcal{U}'_{i_1}$.

We have

$$\begin{aligned}
&\mathbb{E}_{\mathcal{S}}[\mathrm{err}^{\mathrm{abs}}(\boldsymbol{B},\mathcal{S})] - \mathbb{E}_{\mathcal{S}}[\mathrm{err}^{\mathrm{abs}}(\boldsymbol{B}',\mathcal{S})] && (3.37)\\
&= 1 + p_{j_0} \prod_{j\in\mathcal{U}_{i_1}\setminus\{j_0\}} p_j - p_{j_0} - \prod_{j\in\mathcal{U}_{i_1}\setminus\{j_0\}} p_j && (3.38)\\
&= (1-p_{j_0})(1 - \prod_{j\in\mathcal{U}_{i_1}\setminus\{j_0\}} p_j) > 0, && (3.39)
\end{aligned}$$

which contradicts the assumption that $\boldsymbol{B}$ achieves the minimum expected absence error. □

We then derive a lower bound for the expected recovery error.

Lemma 5. For any $\boldsymbol{B}$ and $\boldsymbol{C}$, we have

$$\mathbb{E}_{\mathcal{S}}[\mathrm{err}(\boldsymbol{B},\boldsymbol{C},\mathcal{S})] \geq k\cdot(\prod_{j=1}^{n} p_j)^{d/k}. \tag{3.40}$$

Proof. According to (3.15) and Lemma 1, we have

$$\mathbb{E}_{\mathcal{S}}[\mathrm{err}(\boldsymbol{B},\boldsymbol{C},\mathcal{S})] \geq \mathbb{E}_{\mathcal{S}}[\mathrm{err}^{\mathrm{abs}}(\boldsymbol{B},\mathcal{S})] = \sum_{i=1}^{k}\prod_{j\in\mathcal{U}_i} p_j. \tag{3.41}$$

Since the product of the summands $(\prod_{j\in\mathcal{U}i} p_j)$ over all i is larger than or equal to $(\prod_{j=1}^{n} p_j)^d$, we can apply the the Arithmetic-Mean-Geometric-Mean (AM-GM) inequality to obtain the lower bound on the right-hand side of (3.40), □

□ End of chapter.

Chapter 4

Data allocation scheme and error analysis

Summary

This chapter presents the problem of approximate gradient coding, introduces relevant concepts, discusses optimal data assignment patterns under different worker conditions, and presents simulation results and comparisons with other methods.

4.1 Homogeneous Setting

We continue with the assumption that there are n workers and k data blocks in total. We say that the workers are *homogeneous* if each of them stores the same number of data blocks, and have the same straggling probability. In this section, we consider that each worker stores *exactly* d data blocks, and d is a divisor of k. Furthermore, we assume that each worker straggles with probability $\bar{p}$, independently of the others, for some constant $0 \leq \bar{p} < 1$. Let $r = \lfloor nd/k \rfloor$.

We identify a special data allocation structure that minimizes the expected absence error in the homogeneous case.

Definition 3. (Balanced Assignment Matrix) We say that a data assignment matrix $\boldsymbol{B}$ is *valid* if every row contains exactly d ones. A valid assignment matrix is said to be *balanced* if each column contains either r or $r + 1$ ones. That is, the data blocks are distributed to workers as evenly as possible.

Theorem 6. Consider a homogeneous system. Let $\boldsymbol{B}^{ba}$ be a balanced assignment matrix, then for any valid data assignment matrix $\boldsymbol{B}$, we have

$$\mathbb{E}_{\mathcal{S}}[\text{err}^{\text{abs}}(\boldsymbol{B}^{ba}, \mathcal{S})] \leq \mathbb{E}_{\mathcal{S}}[\text{err}^{\text{abs}}(\boldsymbol{B}, \mathcal{S})]. \tag{4.1}$$

Proof. Let $\boldsymbol{u}^{ba} = (u_1^{ba}, \ldots, u_k^{ba})$ and $\boldsymbol{u} = (u_1, \ldots, u_k)$ be vectors of dimension k, indicating the count of ones in each column of $\boldsymbol{B}^{ba}$ and $\boldsymbol{B}$, respectively. By Theorem 2 with $g(x) = \bar{p}^x$, it suffices to show that $\boldsymbol{u}^{ba}$ is weakly majorized by $\boldsymbol{u}$. Because both matrices $\boldsymbol{B}^{ba}$ and $\boldsymbol{B}$ are valid, we have $\sum_{i=1}^k u_i^{ba} = \sum_{i=1}^k u_i = n \cdot d$, We want to prove that $\boldsymbol{u}^{ba}$ is majorized by vector $\boldsymbol{u}$.

Consider a vector $\boldsymbol{u}$ whose components are nonzero, non-increasing, and sum to nd. We apply the following action repeatedly to $\boldsymbol{u}$: find the smallest index i such that there exists another index $j > i$ with $u_i - u_j \geq 2$, replace u_i by $u_i - 1$ and replace u_j by $u_j + 1$. It is obvious that the sum of the components in the new vector remains constant. Moreover, the new vector is majorized by the original one. We repeat this process until the difference between any two components in the vector is either 0 or 1.

We note that this process must stop. It is because the sequence of vectors produced by this process is decreasing in the lexicographical order.

Hence, an arbitrary vector $\boldsymbol{u}$ with nonzero components summing to nd majorizes a vector $\boldsymbol{u}'$ whose components consists of two distinct integers $a+1$ and a. Since the sum of the components is given, we can determine the value of a, and how many times $a+1$ and a appear in the vector. Suppose $nd = kr + t$ for some integers r and t, where t is the unique integer between 0 and $k-1$ such that $nd - t$ is divisible by k. The vector $\boldsymbol{u}'$ that we get at the end of this process must be the vector

$$(\underbrace{r+1, \ldots, r+1}_{t}, \underbrace{r, \ldots, r}_{k-t}),$$

which is a permutation of the vector $\boldsymbol{u}^{ba}$.

This proves that vector $\boldsymbol{u}^{ba}$ is majorized by vector $\boldsymbol{u}$. $\square$

We then introduce a class of matrices called fractional repetition matrix.

Definition 4. (Fractional Repetition Matrix) A data assignment matrix $\boldsymbol{B}$ is said to be a *fractional repetition matrix* if it satisfies the condition that

$$\mathcal{U}_i \cap \mathcal{U}_j \neq \emptyset \Rightarrow \mathcal{U}_i = \mathcal{U}_j \tag{4.2}$$

for any $i, j = 1, 2, \ldots, n$, and

$$\bigcup_{i=1}^{k} \mathcal{U}_i = \{1, 2, \ldots, n\}. \tag{4.3}$$

We will denote a matrix in this form by $\boldsymbol{B}^{fr}$.

In other words, for a fractional repetition matrix $\boldsymbol{B}^{fr}$, the set of sets $\{\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_k\}$ forms a partition of the index set $\{1, 2, \ldots, n\}$. Given a $\boldsymbol{B}^{fr}$, $\boldsymbol{C}$ can simply be assigned the same values as $\boldsymbol{B}^{fr}$. We present Algorithm 1 to construct the $\boldsymbol{a}$, that is, the linear combination coefficients at the master.

Algorithm 1 Construction of $\boldsymbol{a}$ at the master

```
1: Initialize $\boldsymbol{a} = (0, \dots, 0), \mathcal{W} = \emptyset$.
2: for $i = 1, 2, \dots, k$ do
3:     if $\mathcal{U}_i \nsubseteq \mathcal{S}$ and $\mathcal{W} \cap \mathcal{U}_i = \emptyset$ then
4:         Pick $j$ such that $j \in \mathcal{U}_i$ and $j \notin \mathcal{S}$, let $a_j = 1$
5:     end if
6:     $\mathcal{W} \leftarrow \mathcal{W} \cup \{j\}$
7: end for
```

Lemma 7. Given a fractional repetition matrix $\boldsymbol{B}^{fr}$ and $\boldsymbol{C} = \boldsymbol{B}^{fr}$, for any $\mathcal{S} \in \mathcal{P}(\mathcal{N})$, the $\bar{\boldsymbol{a}}$ yielded by Algorithm 1 solves (3.5). Moreover, the loss error is zero, that is

$$\mathrm{err}^{\mathrm{los}}(\boldsymbol{B}^{fr}, \boldsymbol{C}, \mathcal{S}) = 0. \tag{4.4}$$

Proof. Let $\boldsymbol{b}_i^{fr}$ denote the i-th column of $\boldsymbol{B}^{fr}$ for $i = 1, \dots, k$. For any $\boldsymbol{a}$ such that $\mathrm{supp}(\boldsymbol{a}) \subseteq \mathcal{S}^{\mathrm{c}}$, we have

$$\|\mathbf{1}_{1\times k} - \boldsymbol{a}\boldsymbol{C}\|^2 = \|\mathbf{1}_{1\times k} - \boldsymbol{a}\boldsymbol{B}^{fr}\|^2 \tag{4.5}$$

$$= \sum_{i=1}^{k} \mathbb{1}(\mathcal{U}_i \subseteq \mathcal{S}) + \sum_{i=1}^{k} \mathbb{1}(\mathcal{U}_i \nsubseteq \mathcal{S})(\boldsymbol{a}\boldsymbol{b}_i^{fr} - 1)^2 \tag{4.6}$$

$$\geq \sum_{i=1}^{k} \mathbb{1}(\mathcal{U}_i \subseteq \mathcal{S}) + \sum_{i=1}^{k} \mathbb{1}(\mathcal{U}_i \nsubseteq \mathcal{S})(\bar{\boldsymbol{a}}\boldsymbol{b}_i^{fr} - 1)^2. \tag{4.7}$$

The last inequality holds as $\sum_{i=1}^{k} \mathbb{1}(\mathcal{U}_i \nsubseteq \mathcal{S})(\bar{\boldsymbol{a}}\boldsymbol{b}_i^{fr} - 1)^2 = 0$ due to the construction of $\bar{\boldsymbol{a}}$. So $\bar{\boldsymbol{a}}$ is optimal to (3.5) and thus

$$\mathrm{err}^{\mathrm{los}}(\boldsymbol{B}^{fr}, \boldsymbol{C}, \mathcal{S}) = \sum_{i=1}^{k} \mathbb{1}(\mathcal{U}_i \nsubseteq \mathcal{S})(\bar{\boldsymbol{a}}\boldsymbol{b}_i^{fr} - 1)^2 = 0. \tag{4.8}$$

□

We introduce a matrix class that combines the advantages of both the balanced assignment matrix and fractional repetition matrix. This matrix is designed to minimize both absence error and loss error.

Definition 5. (Balanced Fractional Repetition Matrix) A data assignment matrix $\boldsymbol{B}$ is said to be a *balanced fractional repetition matrix* if it satisfies the conditions in Definitions 3 and 4. We will use the notation $\boldsymbol{B}^{bf}$ to indicate that a matrix is a balanced fractional repetition matrix.

Example 3. Consider a system with $n = 5$ workers, $k = 6$ data blocks and $d = 3$. Then $r = \lfloor nd/k \rfloor = 2$. A balanced fraction repetition matrix is given as

$$\boldsymbol{B}^{bf} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}. \tag{4.9}$$

It can be checked that every row contains d ones and each column contains either r or $r+1$ ones. Moreover, $\mathcal{U}_1 = \mathcal{U}_2 = \mathcal{U}_3, \mathcal{U}_4 = \mathcal{U}_5 = \mathcal{U}_6$, and $\cup_{i=1}^{6} \mathcal{U}_i = \{1, \ldots, 5\}$.

In the following, we show that balanced fractional repetition matrix is an optimal choice of data assignment matrix when the workers are homogeneous.

Theorem 8. Consider a homogeneous system. Let $\boldsymbol{B}^{bf}$ be a balanced fraction repetition matrix and $\boldsymbol{C}^{bf} = \boldsymbol{B}^{bf}$. Then for any valid data assignment matrix $\boldsymbol{B}$ and any $\boldsymbol{C}$ that satisfies $\text{supp}(\boldsymbol{C}) \subseteq \text{supp}(\boldsymbol{B})$, we have

$$\mathbb{E}_{\mathcal{S}}[\text{err}(\boldsymbol{B}, \boldsymbol{C}, \mathcal{S})] \geq \mathbb{E}_{\mathcal{S}}[\text{err}(\boldsymbol{B}^{bf}, \boldsymbol{C}^{bf}, \mathcal{S})]. \tag{4.10}$$

Proof.

$$\begin{aligned}
\mathbb{E}_{\mathcal{S}}[\text{err}(\boldsymbol{B}, \boldsymbol{C}, \mathcal{S})] &\geq \mathbb{E}_{\mathcal{S}}[\text{err}^{\text{abs}}(\boldsymbol{B}, \mathcal{S})] && (4.11)\\
&\geq \mathbb{E}_{\mathcal{S}}[\text{err}^{\text{abs}}(\boldsymbol{B}^{bf}, \mathcal{S})] && (4.12)\\
&= \mathbb{E}_{\mathcal{S}}[\text{err}(\boldsymbol{B}^{bf}, \boldsymbol{C}^{bf}, \mathcal{S})] && (4.13)
\end{aligned}$$

where (4.11) holds according to (3.15), (4.12) holds by Theorem 6, and (4.13) holds as $\mathbb{E}_{\mathcal{S}}[\text{err}^{\text{los}}(\boldsymbol{B}^{bf}, \boldsymbol{C}^{bf}, \mathcal{S})] = 0$ according to Lemma 7. □

4.2 Heterogeneous Setting

In previous section, we show that balanced fractional repetition matrices are optimal for homogeneous systems in the sense that they minimize the expected recovery error. However, in a heterogeneous system where workers may have different probabilities of straggling, the task of identifying an optimal data assignment matrix becomes significantly more complex.

Example 4. We continue with Example 2, but the straggling probabilities of workers 1 to 4 become

$$p_1 = 0.01,\ p_2 = 0.2,\ p_3 = 0.3,\ p_4 = 0.4. \tag{4.14}$$

Considering the balanced fractional repetition matrix $\boldsymbol{B}'$ as given in (3.30), by Lemma 7, the expected loss error is 0. Hence, the expected recovery error equals the expected absence error, which by Lemma 1 is given by

$$0.01 \times 0.4 + 0.2 \times 0.3 + 0.2 \times 0.3 + 0.01 \times 0.4 = 0.128. \tag{4.15}$$

Consider a data assignment matrix which is not balanced:

$$\boldsymbol{B}'' = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}. \tag{4.16}$$

The expected recovery error is

$$0.01 \times 2 + 0.2 \times 0.3 \times 0.4 \times 2 = 0.068. \tag{4.17}$$

From above example, it becomes evident that in a heterogeneous system, the balanced assignment matrix is not necessary to be optimal. In view of the advantageous property of the fractional repetition matrix, which simplifies the design of $\boldsymbol{C}$ and $\boldsymbol{a}$ and results in zero loss error, we investigate the optimal fractional repetition matrix for a heterogeneous system. The problem becomes

$$\min_{\boldsymbol{B}^{fr}} \mathbb{E}_{\mathcal{S}}[\text{err}^{\text{abs}}(\boldsymbol{B}^{fr}, \mathcal{S})] = \min_{\boldsymbol{B}^{fr}} \sum_{i=1}^{k} \prod_{j \in \mathcal{U}_i} p_j. \tag{4.18}$$

For analytical convenience, we consider that each worker stores d data blocks, and d is a divisor of k. Let $m = k/d$. Optimizing $\boldsymbol{B}^{fr}$ reduces to optimizing a partition of n workers into m groups, where each group handles a unique set of d data blocks. Let $\mathcal{V}_q$ denote the set of worker indices in the q-th group, for $q = 1, \dots, m$. The sets $\mathcal{U}_i$ for $i = 1, \dots, k$ can be obtained by replicating $\mathcal{V}_q$ for $q = 1, \dots, m$ d times. Therefore, the problem in (4.18) reduces to

$$\min_{\mathcal{V}_1,\ldots,\mathcal{V}_m} \quad \sum_{q=1}^{m} d \cdot \prod_{j\in\mathcal{V}_q} p_j \tag{4.19}$$

$$s.t. \quad \mathcal{V}_j \cap \mathcal{V}_i = \emptyset, \forall i \neq j \tag{4.20}$$

$$\cup_{q=1}^{m} \mathcal{V}_q = \mathcal{N}. \tag{4.21}$$

We call this problem the Fractional Repetition Code-Grouping (FRC-Grouping) problem. As will be proved shortly, this problem is generally NP-hard. Prior to that, we examine some special cases.

4.2.1 When $n = k$, $d = 2$ and $|\mathcal{V}_i| = 2$ for $i = 1, \ldots, m$

Proposition 9. Consider $n = k$, $d = 2$ and $|\mathcal{V}_i| = 2$ for $i = 1, \ldots, m$. Assume that the workers are arranged in ascending order based on their straggling probabilities, i.e., $p_1 \leq p_2 \leq \cdots \leq p_n$. Then, the group solution

$$\mathcal{V}_i = \{i, n-i+1\}, \quad i = 1, \ldots, m \tag{4.22}$$

minimizes $\sum_{q=1}^{m} 2 \cdot \prod_{j\in\mathcal{V}_q} p_j$.

The proposition is obtained directly from the rearrangement inequality [Marshall et al., 2011, Chapter 6]. It suggests that the optimal grouping involves pairing the top-performing worker with the lowest-performing worker, the second-best with the second-worst, and so on.

4.2.2 When $n = k = 4$ and $d = 2$

Theorem 10. When $n = k = 4$ and $d = 2$, the minimum expected recovery error for heterogeneous workers can be achieved by a fractional repetition matrix.

We introduce another concept called fractional grouped matrix, which will play a key role in the proof of Theorem 10.

Definition 6. (fractional grouped matrix) Suppose k is divisible by d and let $m = k/d$. A data assignment matrix $\boldsymbol{B}$ is said to be a *fractional grouped matrix* if its columns can be divided in to m groups of size d, such that in each group, there are exactly one nonzero entry in each row. We denote a matrix in this form by $\boldsymbol{B}^{fg}$.

Example 5. Consider a system with $n = 4$ workers, and $k = 4$ data blocks. Suppose each worker stores $d = 2$ data block. The following matrix

$$\boldsymbol{B}^{fg} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \tag{4.23}$$

is a fractional grouped matrix. The first and fourth columns form the first group, and the second and third columns form the second group. In each group, each row contains exactly one nonzero entry.

We illustrate the definition using Example 5. For the first group (columns 1 and 4), the the sum of individual absence error is calculated as $p_1p_4 + p_2p_3$ because workers 1 and 4 handle the first data block, and worker 4 handles the fourth data block. For the second group (columns 2 and 3), the sum of individual absence error is $p_1 + p_2p_3p_4$ since worker 1 handle the second data block, and workers 2, 3 and 4 handle the third data block.

We next show that fractional grouped matrix can be ignored in the search of optimal structure that minimize the expected recovery error.

Lemma 11. Suppose k is divisible by d and let $m = k/d$. Given any fractional grouped matrix $\boldsymbol{B}^{fg}$, we can find a *fractional repetition* matrix $\boldsymbol{B}^{fr}$ such that:

$$\mathbb{E}_{\mathcal{S}}[\mathrm{err}(\boldsymbol{B}^{fg}, \mathcal{S})] \geq \mathbb{E}_{\mathcal{S}}[\mathrm{err}(\boldsymbol{B}^{fr}, \mathcal{S})]. \tag{4.24}$$

Proof. Let $\boldsymbol{\mathcal{U}}_q$, for $q \in [1, m]$, denote the set of the $\mathcal{U}_i$ of i-th data block within group q. We first compute the sum of individual absence error for each data block within groups, i.e., for group q, the sum of individual absence error is given by

$$\sum_{\mathcal{U}_i \in \boldsymbol{\mathcal{U}}_q} \prod_{j \in \mathcal{U}_i} p_j.$$

By applying (3.15) and aggregating the individual absence errors across all m groups , we obtain

$$\mathbb{E}_{\mathcal{S}}[\mathrm{err}(\boldsymbol{B}^{fg}, \mathcal{S})] \geq \mathbb{E}_{\mathcal{S}}[\mathrm{err}^{abs}(\boldsymbol{B}^{fg}, \mathcal{S})] = \sum_{q=1}^{m} \sum_{\mathcal{U}_i \in \boldsymbol{\mathcal{U}}_q} \prod_{j \in \mathcal{U}_i} p_j$$

Let $\boldsymbol{\mathcal{U}}^*$ denote the set of the group with the lowest individual absence error. Construct a data assignment matrix $\boldsymbol{B}^{fr}$ by replicating the data assignment strategy from $\boldsymbol{\mathcal{U}}^*$ across all m groups. This matrix has the structure of fractional repetition matrix. Consequently, the average recovery error of matrix B is lower bounded by

$$\sum_{q=1}^{m} \sum_{\mathcal{U}_i \in \boldsymbol{\mathcal{U}}_q} \prod_{j \in \mathcal{U}_i} p_j \geq m \cdot \sum_{\mathcal{U}_i \in \boldsymbol{\mathcal{U}}^*} \prod_{j \in \mathcal{U}_i} p_j = \mathbb{E}_{\mathcal{S}}[\mathrm{err}(\boldsymbol{B}^{fr}, \mathcal{S})].$$

□

Proof of Theorem 10. We show that for any data assignment matrix $\boldsymbol{B}$, we can find another fractional repetition matrix $\boldsymbol{B}^{fr}$ with less recovery errors. In the followings, we assume without loss of generality that $p_1 < p_2 < p_3 < p_4$.

We divide the proof in three parts by considering the different assignment of worker 1 and 2. Noted that we can always rotate the column to make worker 1 compute task 1 and 2.

Workers 1 and 2 are assigned the same tasks

Suppose the first two rows of the matrix $\boldsymbol{B}$ are given by:

$$\boldsymbol{B} = \begin{pmatrix} * & * & 0 & 0 \\ * & * & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

We will show that by changing the assignment of workers 3 and 4, we can obtain a fractional repetition matrix that has smaller expected recovery error.

There are 36 possible configurations. According to Lemma 4 and Lemma 11, we can ignore the data assignment matrix with empty column, fractional group matrix, and data assignment matrix that is already in the form of fractional repetition matrix.

For instance, the matrix

$$\boldsymbol{B} = \begin{pmatrix} * & * & 0 & 0 \\ * & * & 0 & 0 \\ 0 & 0 & * & * \\ * & 0 & 0 & * \end{pmatrix}$$

is in the form of a fractional grouped matrix. Columns 1 and 3 form a group, and columns 2 and 4 form another group. By the result in Lemma 11, we can find another fractional repetition matrix with smaller expected recovery error. Indeed, any data assignment matrix in the form

$$\begin{pmatrix} * & * & 0 & 0 \\ * & * & 0 & 0 \\ 0 & 0 & * & * \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \text{ or } \begin{pmatrix} * & * & 0 & 0 \\ * & * & 0 & 0 \\ * & * & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

can be seen as a fractional group matrix, regardless of the assignment to worker 4.

It remains to consider the case that worker 3 computes one task in $\{1, 2\}$ and one task in $\{3, 4\}$. Then, worker 4 has to compute task 3 given Lemma 4. We only need to consider

$$\boldsymbol{B} = \begin{pmatrix} * & * & 0 & 0 \\ * & * & 0 & 0 \\ 0 & * & 0 & * \\ 0 & * & * & 0 \end{pmatrix} \tag{4.25}$$

because the other possible data assignment pattern will produce a fractional grouped matrix, which can be neglected by Lemma 11. We can modify the data assignment to worker 3, and compare the expected recovery error the matrix in (4.25) with that of

$$\begin{pmatrix} * & * & 0 & 0 \\ * & * & 0 & 0 \\ 0 & 0 & * & * \\ 0 & * & * & 0 \end{pmatrix} \tag{4.26}$$

The difference of expected recovery error is equal to

$$\begin{aligned} & p_1p_2 + p_1p_2p_3p_4 + p_4 + p_3 - (p_1p_2 + p_1p_2p_4 + p_3p_4 + p_3) \\ & = p_4(1 - p_3)(1 - p_1p_2) \end{aligned}$$

which is a positive number. Since the matrix in (4.26) is in the form of fractional group matrix, we can further reduce the expected recovery error by Lemma 11.

Workers 1 and 2 are assigned different tasks

Suppose the first two rows of the matrix $\boldsymbol{B}$ are given by:

$$\boldsymbol{B} = \begin{pmatrix} * & * & 0 & 0 \\ 0 & 0 & * & * \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

By appealing to Lemma 4 and 11 again, we only need to consider data assignment matrix that are not fractional group matrix and do not contain any all-zero column. When workers 1 and 2 are assigned different tasks, the two cases that we need to consider are

$$\begin{pmatrix} * & * & 0 & 0 \\ 0 & 0 & * & * \\ * & 0 & * & 0 \\ * & 0 & 0 & * \end{pmatrix} \text{ and } \begin{pmatrix} * & * & 0 & 0 \\ 0 & 0 & * & * \\ * & 0 & * & 0 \\ 0 & * & * & 0 \end{pmatrix}.$$

We can modify the first one by changing the data assignment of worker 4,

$$\begin{pmatrix} * & * & 0 & 0 \\ 0 & 0 & * & * \\ * & 0 & * & 0 \\ * & 0 & 0 & * \end{pmatrix} \rightarrow \begin{pmatrix} * & * & 0 & 0 \\ 0 & 0 & * & * \\ * & 0 & * & 0 \\ 0 & 0 & * & * \end{pmatrix}$$

It will reduce the expected recovery error, because the difference

$$\begin{aligned}&p_1p_3p_4 + p_1 + p_2p_3 + p_2p_4 - (p_1p_2 - p_1 - p_2p_3p_4 - p_2p_4)\\&= p_3(1-p_4)(p_2-p_1)\end{aligned}$$

is positive. Since the matrix on the right is a fractional grouped matrix, we can apply Lemma 11.

Likewise, the second matrix can be treated by

$$\begin{pmatrix} * & * & 0 & 0 \\ 0 & 0 & * & * \\ * & 0 & * & 0 \\ 0 & * & * & 0 \end{pmatrix} \to \begin{pmatrix} * & * & 0 & 0 \\ 0 & 0 & * & * \\ * & 0 & * & 0 \\ 0 & 0 & * & * \end{pmatrix}.$$

The calculations are similar and omitted.

Workers 1 and 2 have one task in common

By permuting the columns, we can suppose the first two rows of the matrix $\boldsymbol{B}$ are given by

$$\boldsymbol{B} = \begin{pmatrix} * & * & 0 & 0 \\ 0 & * & * & 0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}.$$

Again, there are 36 ways to fill in the third and fourth row of the matrix. Some of them have zero column, and some of them result in fractional grouped matrices. All these case can be shown to be non-optimal by Lemma 4 and 11.

For the remaining cases, we can always modify the assignment of worker 2, and reduce the argument to parts A or B.

For example, the matrix on the left

$$\begin{pmatrix} * & * & 0 & 0 \\ 0 & * & * & 0 \\ * & 0 & 0 & * \\ * & 0 & * & 0 \end{pmatrix} \to \begin{pmatrix} * & * & 0 & 0 \\ 0 & 0 & * & * \\ * & 0 & * & 0 \\ 0 & 0 & * & * \end{pmatrix}.$$

is not a fractional grouped matrix, but we can change the data assignment in second row as shown above. The difference of expected recovery error is positive, because

$$\begin{aligned} &p_1p_3p_4 + p_1p_2 + p_2p_4 + p_3 - (p_1p_3p_4 + p_1 + p_2p_4 + p_2p_3) \\ &= (p_3 - p_1)(1 - p_2). \end{aligned}$$

Therefore, the matrix on the right has smaller expected recovery error. We can now repeat the argument in Part 4.2.2. □

4.2.3 General Case

Then, we give the main theorem under general cases of this section.

Theorem 12. The FRC-Grouping problem is NP-hard.

Proof. We show that the FRC-Grouping problem with $m = 2$ is NP-hard by reduction from the following problem [Ng et al., 2010, Theorem 3].

Minimize-Generalized Product Partition:

Instance: A positive number n, a list of positive integer numbers $a_1, a_2, \ldots, a_n$. Let P denote the product $\prod_{j=1}^{n} a_j$, $\mathcal{N} := \{1, 2, \ldots, n\}$, and $A(\mathcal{N}) := \left\{\prod_{j\in\mathcal{X}} a_j \middle| \mathcal{X} \subset \mathcal{N}\right\}$.

Question: Minimize $f(x) = x + P/x$ for $x \in A(\mathcal{N})$.

Given an instance of Minimize-Generalized Product Partition, we construct the instance of FRC-Grouping Problem by setting $p_j = 1/a_j$ for all $j = 1, 2, \ldots, n$.

We have

$$\min_{\{\mathcal{V}_1,\mathcal{V}_2\}} \prod_{j\in\mathcal{V}_1} p_j + \prod_{j\in\mathcal{V}_2} p_j \tag{4.27}$$

$$= \min_{\{\mathcal{V}_1,\mathcal{V}_2\}} 1/\prod_{j\in\mathcal{V}_1} a_j + 1/\prod_{j\in\mathcal{V}_2} a_j \tag{4.28}$$

$$\overset{(a)}{=} \min_{\{\mathcal{V}_1,\mathcal{V}_2\}} (\prod_{j\in\mathcal{V}_2} a_j + \prod_{j\in\mathcal{V}_1} a_j)/\prod_{j\in\mathcal{N}} a_j \tag{4.29}$$

$$\overset{(b)}{=} \min_{x\in A(\mathcal{N})} (x + P/x)/P, \tag{4.30}$$

where (a) is obtained from $\prod_{j\in\mathcal{V}_1} a_j \cdot \prod_{j\in\mathcal{V}_2} a_j = \prod_{j\in\mathcal{N}} a_j$ since $\mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{N}$, and (b) is obtained by substituting x by $\prod_{j\in\mathcal{V}_2} a_j$. A solution to the minimization problem in (4.30) yields a solution to the MINIMIZE-GENERALIZED PRODUCT PARTITION. □

As the general FRC-GROUPING problem is NP-hard, we consider a heuristic algorithm in the remainder of this section. We consider a two-stage, greedy-swapping algorithm. By re-labelling the workers if necessary, we can assume without loss of generality that $p_1 \le p_2 \le \cdots \le p_n$.

The first stage involves a greedy algorithm. We start by setting up m empty sets $\mathcal{V}_1, \ldots, \mathcal{V}_m$. Then for $j = 1, 2, \ldots, n$, we in turn identify the group $q \in [1, m]$ with the maximum product $\prod_{i\in\mathcal{V}_q} p_i$, and add j to the set $\mathcal{V}_q$. Ties are resolved arbitrarily, and an empty product is considered as 1 as per convention.

In the second stage, we assess all potential pairs of workers to determine if exchanging two workers from distinct groups could reduce the expected recovery error. Should an improvement be feasible through swapping, the workers are exchanged. If not, we move on to the subsequent pair of workers. This process is repeated until no further improvement via swapping is achievable. The complete algorithm is detailed in Algorithm 2. Convergence of the algorithm is assured due to the finite number of worker pairs under consideration. In the next section, we will show by simulation that the proposed algorithm achieves performance very close to the lower bound of the expected recovery error under some scenarios.

Algorithm 2 Two-Stage Heuristic Algorithm for FRC-G

```
Input: Number of workers $n$, number of groups $m$, straggler probabilities
    $p_1, p_2, \ldots, p_n$. Assume $p_1 \le p_2 \le \cdots \le p_n$.
Output: A partition of $\{1, 2, \ldots, n\}$
 1: // Greedy Stage:
 2: Initialize $m$ empty sets $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_m$.
 3: for $j = 1, 2, \ldots, n$ do
 4:     Find index $q \in \{1, \ldots, m\}$ such that $\prod_{i \in \mathcal{V}_q} p_i$ is maximized (with tie broken
        arbitrarily)
 5:     Update $\mathcal{V}_q \leftarrow \mathcal{V}_q \cup \{j\}$;
 6: end for
 7: // Swap Stage:
 8: Calculate the expected recovery error $d \cdot \sum_{q=1}^{m} \prod_{j \in \mathcal{V}_q} p_j$.
 9: repeat
10:     Set $Improved \leftarrow False$;
11:     for pair of workers $(j_1, j_2)$ with $j_1$ and $j_2$ in two distinct groups do
12:         if swapping workers $j_1$ and $j_2$ results in a reduction of expected recovery
            error then
13:             Swap workers $j_1$ and $j_2$
14:             Update the current expected recovery error
15:             Set $Improved \leftarrow True$
16:         end if
17:     end for
18: until $Improved = False$
19: return $\{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_m\}$
```

It worth mentioning that the heuristic algorithm requires prior knowledge of the straggling probabilities $p_1, \ldots, p_n$ of the workers. In practical applications, p_j can be estimated through some observation of worker j's behavior. We will also discuss one way to estimate p_j in the next chapter.

□ End of chapter.

Chapter 5

Experiment Results and Discussion

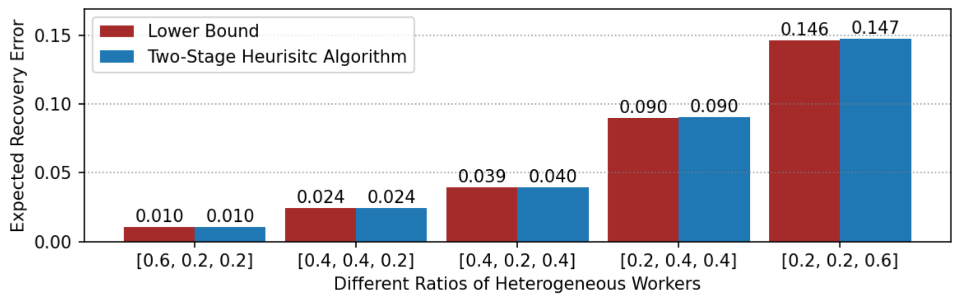
Summary

This chapter presents simulation results and comparisons with other methods. Results show that our proposed heuristic can achieve an expected recovery error very close to the lower bound.

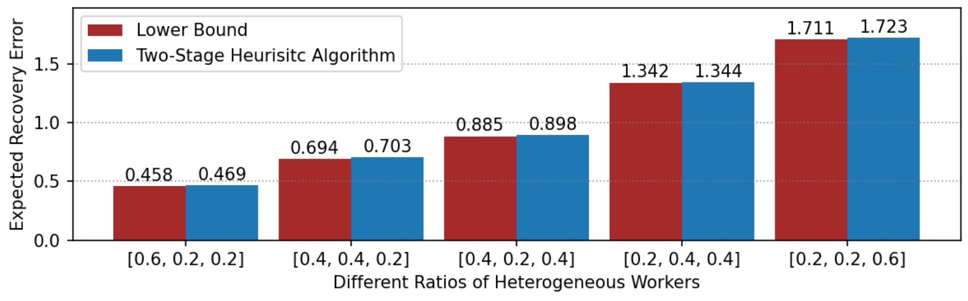
In this chapter, we present some experimental results.

5.1 Performance of two-stage heuristic algorithm for FRC-G

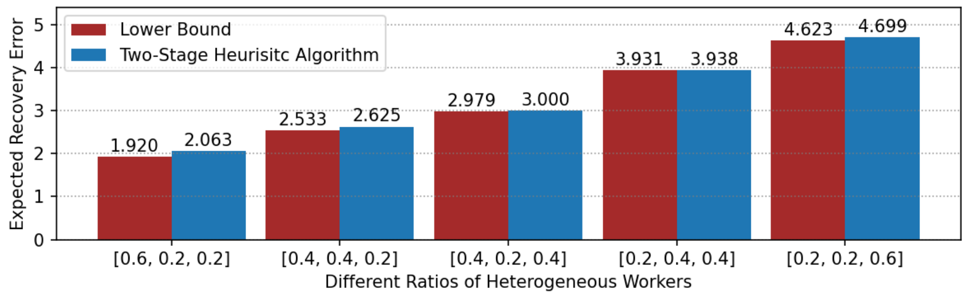
We first evaluate the performance of our two-stage heuristic algorithm as outlined in Algorithm 2 for addressing the FRC-G problem. We consider three types of workers, distinguished by three different straggling probabilities: p_{low}, p_{med} and p_{high}. Each worker $j = 1, 2, \ldots, n$ belongs to a specific type and has a straggling probability p_j set as p_{low}, p_{med}, or p_{high}. The evaluation includes analyzing the performance across



(a) $n = 30, m = 5$



(b) $n = 30, m = 10$



(c) $n = 30, m = 15$

Figure 5.1: Expected recovery error under different ratios of heterogeneous workers.

various ratios of these three worker types.

Set $n = k = 30$, $p_{\text{low}} = 0.25, p_{\text{med}} = 0.5$ and $p_{\text{high}} = 0.75$. Fig. 5.1 presents the theoretical lower bound of the expected recovery error, calculated using (3.40), and the expected recovery error obtained by the heuristic algorithm under different values of m (i.e. the number of worker groups) and worker type ratios. The ratios of worker types shown in the square brackets on the x-axis correspond to the ratios of low, medium and high types. For instance, $[0.6, 0.2, 0.2]$ indicates that 60% of the workers have a low straggling probability, 20% have a medium probability, and 20% have a high probability.

From the figure, it can be seen that in all cases the expected recovery error achieved by the heuristic algorithm closely aligns with the lower bound on the expected recovery error. This implies that the data assignment generated by the heuristic algorithm is nearly optimal in the sense of minimizing the expected recovery error. In addition, the expected recovery error increases as the proportion of worker types with higher straggling probability increases and as m increases. The latter phenomenon occurs because, as m increases, the potential number of workers in each worker group decreases, thereby reducing the capability to mitigate the effect of stragglers.

5.2 Time consumption for exact gradient recovery

We consider a setting of heterogeneous workers and compare the per-iteration time consumption for exact gradient recovery under different gradient coding schemes.

We first give a formal definition of the per-iteration time consumption. We define the response time of a worker by the time elapsed from the beginning of an iteration to the time the master receives its computational result. The response time of worker j, denoted by T_j for $j = 1, \ldots, n$, is a random variable that can vary with each training iteration. Followed [Reisizadeh et al., 2019] and [Lee et al., 2017], we assume that $T_1, T_2, \ldots, T_n$ are independent of each other and the distribution of T_j follows a shift-exponential distribution. Specifically, for worker j, the cumulative distribution function of T_j is given by

$$\Pr(T_j \leq t) = \begin{cases} 1 - e^{-\mu_j(t-\tau_j)} & t \geq \tau_j, \\ 0 & t < \tau_j, \end{cases} \tag{5.1}$$

where $\mu_j > 0$ is the rate parameter and $\tau_j > 0$ is the shift parameter. The mean response time of worker j is $1/\mu_j + \tau_j$. The per-iteration time consumption, denoted by T, is defined as the duration from the start of an iteration to when the master receives enough information to recover the exact full gradient. T is also a random variable and depends on $T_1, \ldots, T_n$ and the gradient coding schemes.

We evaluate T under four different gradient coding schemes:

- Fractional Repetition Code with Heuristic Grouping (FRC-HG): This is our scheme that applies the two-stage heuristic algorithm to decide the data assignment matrix $\boldsymbol{B}$, sets $\boldsymbol{C} = \boldsymbol{B}$ and solves $\boldsymbol{a}$ using Algorithm 1. As noted before, in the heuristic algorithm, prior knowledge of the straggling probabilities $p_1, \ldots, p_n$ of the workers is needed. We estimate p_j by observing the response time T_j for multiple iterations. Given a time threshold γ, in each iteration, if $T_j > \gamma$, the worker is treated as a straggler. In the simulation, p_j is a statistical result of 5000 observations. It needs to emphasize that for exact gradient recovery, the master may need to wait for information from the stragglers, i.e., T could be larger than γ.

- Cyclic Repetition Scheme (CRS)[Tandon et al., 2017]: The first row of the data assignment matrix $\boldsymbol{B}$ has its first d entries assigned as one. Subsequently, starting from $j = 2$, the j-th row is a cyclic shift of the $(j-1)$-th row by one step to the right. The coefficient matrix $\boldsymbol{C}$ is constructed using [Tandon et al., 2017, Algorithm 2]. The master solves (3.5) to recover the gradient. It is proved in [Tandon et al., 2017, Theorem 3] that for exact gradient recovery, the master needs to receive information from at most $n - d + 1$ workers.

- Gradient Coding with Static Clustering (GC-SC)[Buyukates et al., 2022]: In this setup, the workers are divided into disjoint equal-size clusters, each cluster handling a different set of data blocks. Within each cluster, CRS is used. The schemes assumes prior knowledge of the computation rate, i.e., μ_j, and organizes workers into clusters sequentially based on their computation rates. At the master, the full gradient is recovered by solving (3.5). It is worth noting that when the number of clusters is one, GC-SC simplifies to CRS.

- Balanced Fractional Repetition Code with Shuffling (BFRC-S)[Johri et al., 2021]: This scheme initially determines a balanced fractional repetition matrix. In each iteration, the data assignment matrix $\boldsymbol{B}$ is obtained by randomly permuting the rows of this matrix and the coefficient matrix $\boldsymbol{C}$ is set the same as $\boldsymbol{B}$. The original motivation for shuffling the rows, as discussed in [Johri et al., 2021], is to help the master access the datasets uniformly in presence of stragglers. We use Algorithm 1 to find $\boldsymbol{a}$ used in the decoding process.

Among these four schemes, the data assignment in FRC-HG, CRS, GC-SC are static, meaning it remains fixed during the whole training process. In contrast, BFRC-S employs a dynamic data assignment strategy, necessitating greater storage capacity for the worker nodes. Moreover, FRC-HG and GC-SC make use of the straggling statistics of the workers which are ignored in CRS and BFRC-S.

In the simulation, we set $n = k = 30$ and $d = 3$. The parameter μ_j of each worker j is assigned values of μ_{high}, μ_{med} or μ_{low}. In specific, we set $\mu_{\text{low}} = 0.1$, $\mu_{\text{med}} = 0.3$, $\mu_{\text{high}} = 10$ and $\tau_j = 1$ for all j. The corresponding mean response time are 11s, 4.333s, and 1.1s. We consider three scenarios for the workers' speeds. In Scenario A, the ratios of low, medium and high-speed workers are $[1/3, 1/3, 1/3]$, whereas in Scenario B and C, the ratios are $[1/6, 2/3, 1/6]$ and $[1/10, 8/10, 1/10]$, respectively. Scenario A exhibits greater heterogeneity among the workers than Scenario B and C. The threshold γ is chosen as 1.4s for FRC-HG.

In Table 5.1, the confidence interval of the average time consumption for FRC-HG, CRS, GC-SC with 10 and 5 clusters, and BFRC-S in Scenario A, B and C is summarized. In all scenarios, FRC-HG achieves the lowest time consumption. In Scenario

Table 5.1: Average time consumption for exact gradient recovery under different gradient coding schemes

	Scenario A	Scenario B	Scenario C
CRS	9.21±0.10s	7.63±0.07s	7.15±0.10s
GC-SC(5)	5.81±0.10s	4.81±0.07s	4.90±0.10s
BFRC-S	4.31±0.11s	4.26±0.07s	4.20±0.08s
GC-SC(10)	1.28±0.005s	4.29±0.07s	4.21±0.09s
FRC-HG	**1.28**±0.005s	**2.71**±0.04s	**3.41**±0.08s

A, FRC-HG divides the workers into 10 groups, each consisting of a fast, medium and slow worker, tasked with processing identical data blocks. Consequently, performance hinges primarily on the fast workers. GC-SC with 10 clusters results in worker clustering akin to that of FRC-HG, yielding similar performance. In scenario B, FRC-HG divides the workers into 10 groups: five groups consist of one fast worker each, while the other five consist of 4 medium workers and one slow worker each. In GC-SC with 10 clusters, each cluster comprises a fast, medium and slow worker. So FRC-HG excels in grouping workers to ensure balanced performance across groups. In scenario C, workers become more homogeneous, and the advantages of FRC-HG is not as pronounced. The cumulative distribution function (CDF) of the time consumption in Scenario B is also plotted in Fig. 5.2.

5.3 Performance for distributed machine learning task

We evaluate the performance of different gradient coding schemes on a practical machine learning task. The task is to train a logistic regression model on the MNIST dataset [LeCun et al., 1998] to classify handwritten digits. The cross-entropy loss function is used as well as the gradient descent algorithm. The sizes of the training dataset and test dataset are 60000 and 10000, respectively.

In the simulation, we set $n = k = 30$ and continue to employ the shift-exponential

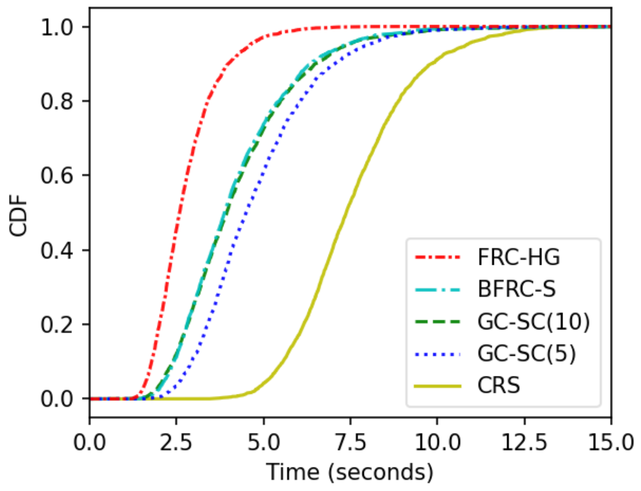


Figure 5.2: The cumulative distribution function of the time consumption for exact gradient recovery under Scenario B.

distribution given in (5.1) to model the workers' response time T_j. We define a deadline γ. During each training iteration, the master uses the information received before γ to recover the gradient, and workers are regarded as stragglers if $T_j > \gamma$. Therefore the recovered gradient is an approximated gradient and may not be the exact gradient.

Homogeneous Workers

We begin by examining a homogeneous scenario, where all workers have the same computational capacities and workload, and thus $\mu_j = \mu$ and $\tau_j = \tau$ for all $j = 1, \ldots, n$. We compare the performance of our scheme, namely balanced fractional repetition code (BFRC), with CRS and the following scheme:

- Expander-graph-based Coding Scheme (EGCS)[Raviv et al., 2020]: EGCS is designed for approximate gradient recovery. The data assignment matrix $\boldsymbol{B}$ is constructed from the normalized adjacency matrix of a randomly generated d-regular Ramanujan graph [Lubotzky et al., 1988], which is known to have good expander properties. We generate $\boldsymbol{B}$ by repeatedly creating random d-regular Ramanujan graphs of size $n \times n$, until we confirm that its second largest eigenvalue is less than or equal to $2\sqrt{d-1}$. During the decoding process, the master solves (3.5) to obtain the estimated gradient.

Given μ and τ, the straggling probability of a worker is a function of γ and can be derived as $\bar{p}(\gamma) = \Pr(T_j > \gamma) = e^{-\mu(\gamma-\tau)}$ for $\gamma \geq \tau$. The smaller the γ, the larger the $\bar{p}$. In the simulation, we keep μ and τ constant, while adjusting γ. For each gradient coding scheme and each value of γ, we execute the training algorithm for 5000 iterations and record the recovery error as defined in (3.6). The average recovery error across the 5000 samples versus $\bar{p}(\gamma)$ is plotted in Fig. 5.3. Two values of d, specifically $d = 3$ and $d = 10$ are taken into account. It can be seen that BFRC consistently achieves the smallest average recovery error for all values of d and $\bar{p}$, which confirms the result established in Theorem 8.

In Fig. 5.4, with fixed values of $d = 3$ and $\bar{p} = 0.55$, the logistic regression model's training set cross-entropy loss and test set prediction accuracy are displayed as the training progresses. BFRC outperforms EGCS and CRS.

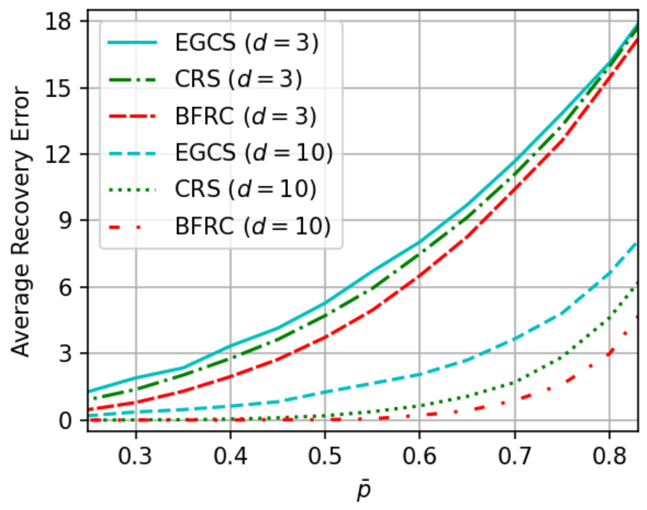


Figure 5.3: Average recovery error with homogeneous workers.

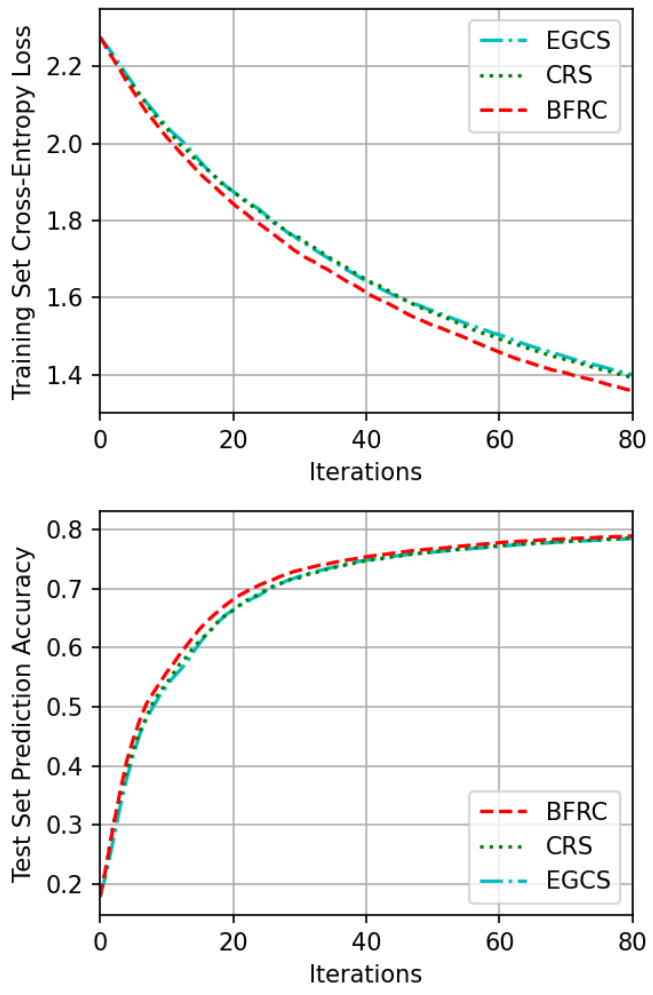


Figure 5.4: Performance of logistic regression model trained on MNIST dataset with homogeneous workers.

Heterogeneous Workers

Now we consider a heterogeneous scenario, where workers may have different μ_j. Specifically, the simulation results are shown for Scenario B which has been introduced in Section 5.2. Recall that $n = k = 30, d = 3$ and the ratios of low ($\mu_{\text{low}} = 0.1$), medium ($\mu_{\text{med}} = 0.3$) and high-speed ($\mu_{\text{high}} = 10$) workers are $[1/6, 2/3, 1/6]$. We observe similar results for Scenario A and C and thus do not include them in the thesis.

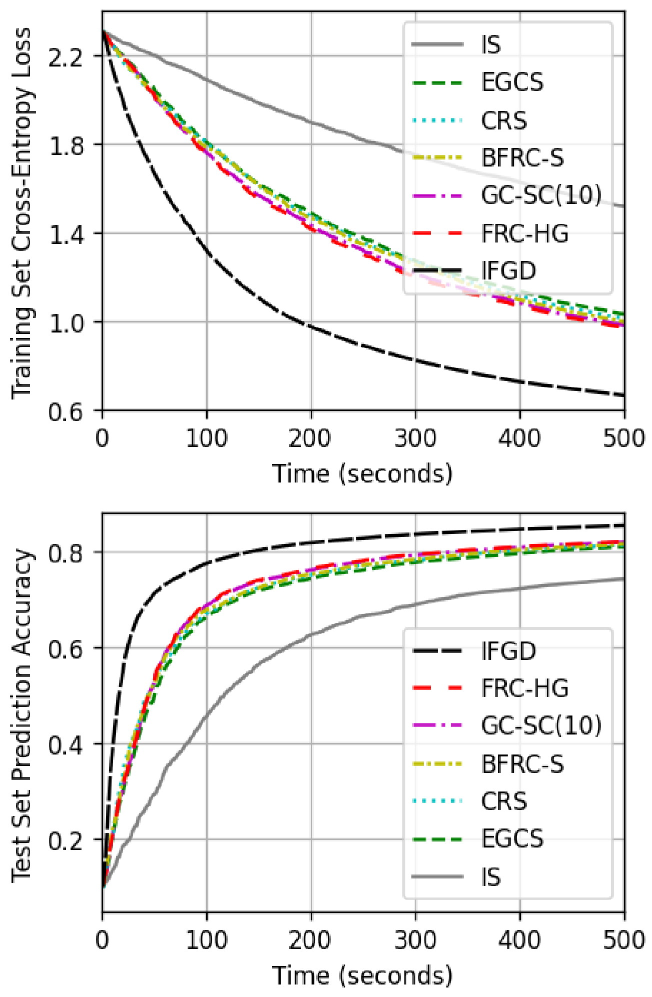


Figure 5.5: Performance for $\gamma = 1.05$ under Scenario B with MNIST

We compare the learning performance under different gradient coding schemes, including the aforementioned CRS, GC-SC with 10 clusters, BFRC-S, EGCS and our FRC-HG. In addition, two benchmark schemes are considered:

- Ideal Fully Gradient Descent (IFGD): consider an ideal scenario that in each training iteration, the master can recover an exact full gradient from the computations of the top $k/d = 10$ fastest workers. The time taken per iteration is regarded as the response time of the worker ranked as the tenth fastest.

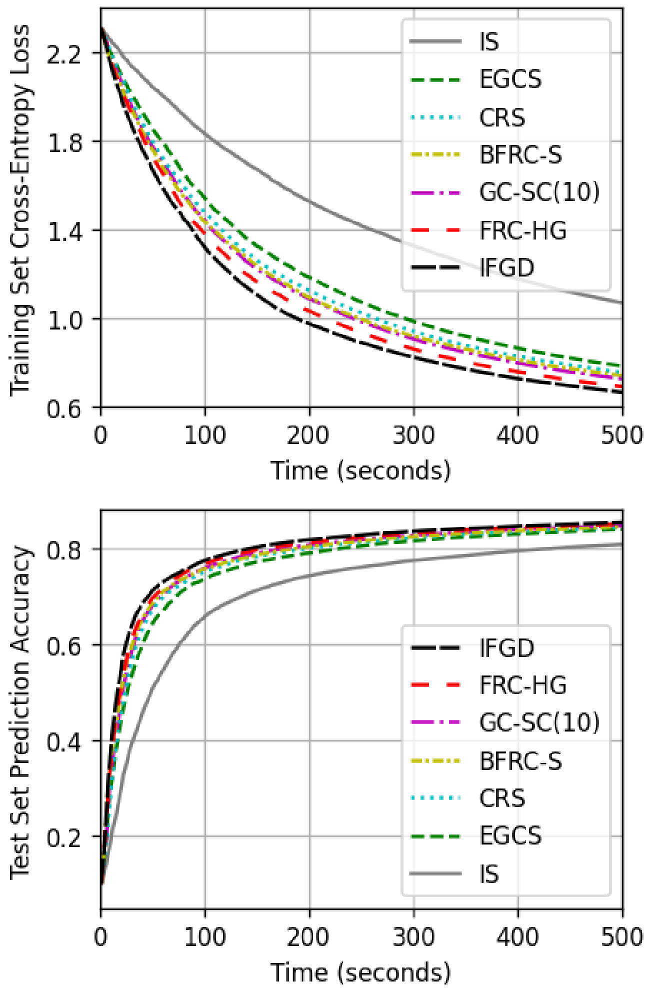


Figure 5.6: Performance for $\gamma = 1.5$ under Scenario B with MNIST

- Ignore Straggler (IS): The data is allocated equally among all workers without replication. In each training iteration, the master uses the information from the non-stragglers to recover a gradient.

Fig. 5.5, 5.6 and 5.7 show the learning curves of logistic regression models on the MNIST data set under three different values of γ, specifically 1.05, 1.5, and 3. Note that the horizontal axis represents time instead of training iteration, and the duration covered by each training iteration may differ. Here, we make the assumption that

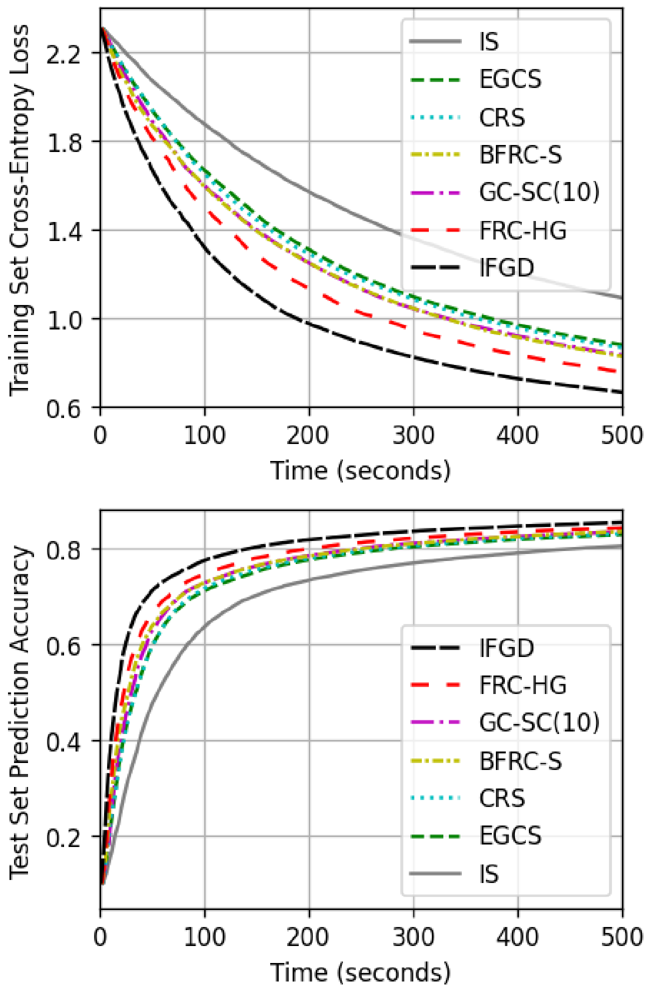


Figure 5.7: Performance for $\gamma = 3$ under Scenario B with MNIST

the processing time at the master is significantly shorter than the response time of workers, and therefore, is disregarded. For fair comparison, the initialization and learning rate of the gradient decent algorithm are set the same for all schemes.

We first exam the influence of γ on learning efficiency. For IFGD, due to the ideal assumption, its learning efficiency remains unaffected by γ. However, for the other schemes, we observe that they learn slowly when $\gamma = 1.05$ and faster when $\gamma = 1.5$. However, when $\gamma = 3$, they tend to slow down again. For example, to achieve a training cross-entropy loss of 1, FRC-HG requires around 500s at $\gamma = 1.05$s, 210s at

$\gamma = 1.5$s, and 260s at $\gamma = 3$s. The reason is that γ plays a role in controlling the trade-off between per-iteration time and gradient accuracy. A larger γ potentially extends the per-iteration time, but also allows for gathering more information to reconstruct a more accurate gradient. This, in turn, may facilitate the convergence of the gradient descent process. When γ is large enough, all schemes except EGCS implement the exact gradient recovery. This further indicates that approximated gradient coding can lead to faster learning compared to exact gradient when accounting for the workers' response time.

Next we compare the performance among the schemes. It is evident that across all values of γ, the schemes with gradient coding perform much better than the one without gradient coding, namely IS. Among the gradient coding schemes, FRC-HG performs the best. Additionally, as γ increases, the difference in learning speed between FRG-HG and other schemes is getting bigger. For example, when $\gamma = 3$s, to reach a training cross-entropy loss of 1, FRC-HG requires approximately 260s, while GC-SC(10) requires 340s and EGCS requires 380s. Lastly, at $\gamma = 1.5$s, the performance of FRC-HG is close to that of IFGD which serves as a benchmark for ideal performance.

Summary

This chapter presents simulation results and comparisons with other methods. Results show that our proposed heuristic can achieve an expected recovery error very close to the lower bound.

□ **End of chapter.**

Chapter 6

Conclusion and future work

Summary

This chapter summarizes the contributions of this thesis.

6.1 Contribution

Our first contribution is a problem formulation that accounts for the statistical characteristics of the workers. We model the workers' response times as independent random variables, each following its own probability distributions. These distributions do not need to be the same. This framework embraces the inherent heterogeneity of modern clusters, where differences between computing nodes are common. We assume that the parameters of the probabilistic model can be estimated through observation. Therefore, we can leverage the statistical information of the workers to design a more effective gradient coding scheme.

The gradient coding schemes in [Bitar et al., 2020; Charles et al., 2017; Glasgow and Wootters, 2021; Wang et al., 2019a] also employ a probabilistic model for stragglers and assume that a worker fails to return the computational result with probability p, for some constant p. As a result, these schemes are limited to homogeneous settings. In [Johri et al., 2021], workers are divided into two classes: fast workers and slow workers. In this thesis, we allow for more than two classes of workers. While [Jahani-Nezhad and Maddah-Ali, 2021; Wang et al., 2022] consider heterogeneous settings, their gradient coding schemes are analyzed in the s-out-of-n model without incorporating the statistical information of the workers.

The second contribution of this thesis is an in-depth study of the data assignment problem aimed at minimizing the expected recovery error. We show that the recovery error can be decomposed into the absence error and the loss error, and the minimization of the expected absence error can be tackled using the theory of majorization. Based on this finding, we derive a lower bound on the expected recovery error. Furthermore, for independent and homogeneous workers, we prove that the optimal solution can be achieved by the fractional repetition code. In the general case of heterogeneous workers, we prove that the worker grouping problem associated with the fractional repetition code is NP-hard. To address this, we design a heuristic algorithm. Numerical results show that our proposed heuristic can achieve an expected recovery error very close to the lower bound.

□ End of chapter.

Bibliography

Bitar, R., Wootters, M., and El Rouayheb, S. (2020). Stochastic gradient coding for straggler mitigation in distributed learning. *IEEE J. Sel. Areas Inf. Theory*, 1(1):277–291.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In Lechevallier, Y. and Saporta, G., editors, *19th International Conference on Computational Statistics (COMPSTAT)*, pages 177–186.

Buyukates, B., Ozfatura, E., Ulukus, S., and Gündüz, D. (2022). Gradient coding with dynamic clustering for straggler-tolerant distributed learning. *IEEE Transactions on Communications.*

Cao, H., Yan, Q., Tang, X., and Han, G. (2021). Adaptive gradient coding. *IEEE/ACM Transactions on Networking*, 30(2):717–734.

Charalambides, N., Mahdavifar, H., and Hero III, A. O. (2024a). Generalized fractional repetition codes for binary coded computations. arXiv:2109.10484v2 [cs.IT].

Charalambides, N., Pilanci, M., and Hero, A. O. (2024b). Gradient coding with iterative block leverage score sampling. *IEEE Trans. Inf. Theory.*

Charles, Z., Papailiopoulos, D., and Ellenberg, J. (2017). Approximate gradient coding via sparse random graphs. *arXiv preprint arXiv:1711.06771.*

Glasgow, M. and Wootters, M. (2021). Approximate gradient coding with optimal decoding. *IEEE Journal on Selected Areas in Information Theory*, 2(3):855–866.

Halbawi, W., Azizan, N., Salehi, F., and Hassibi, B. (2018). Improving distributed gradient descent using reed-solomon codes. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 2027–2031.

Jahani-Nezhad, T. and Maddah-Ali, M. A. (2021). Optimal communication-computation trade-off in heterogeneous gradient coding. *IEEE Journal on Selected Areas in Information Theory*, 2(3):1002–1011.

Jiang, Z., Lin, H., Zhong, Y., et al. (2024). MegaScale: Scaling large language model training to more than 10,000 GPUs. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 745–760.

Johri, A., Yardi, A., and Bodas, T. (2021). Approximate gradient coding for heterogeneous nodes. In *2021 IEEE Information Theory Workshop (ITW)*, pages 1–6. IEEE.

Kadhe, S., Koyluoglu, O. O., and Ramchandran, K. (2020). Communication-efficient gradient coding for straggler mitigation in distributed learning. In *IEEE Int. Symp. Inf. Theory (ISIT)*, pages 2634–2639.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, K., Lam, M., Pedarsani, R., Papailiopoulos, D., and Ramchandran, K. (2017). Speeding up distributed machine learning using codes. *IEEE Transactions on Information Theory*, 64(3):1514–1529.

Li, H., Chen, Y., Shum, K. W., and Sung, C. W. (2023). Data allocation for approximate gradient coding in edge networks. In *IEEE Int. Symp. Inf. Theory (ISIT)*, pages 2541–2546.

Lubotzky, A., Phillips, R., and Sarnak, P. (1988). Ramanujan graphs. *Combinatorica*, 8(3):261–277.

Marshall, A. W., Olkin, I., and Arnold, B. C. (2011). *Inequalities: theory of majorization and its applications.* Springer, 2nd edition.

Ng, C. T., Barketau, M., Cheng, T. E., and Kovalyov, M. Y. (2010). "Product partition" and related problems of scheduling and systems reliability: Computational complexity and approximation. *European Journal of Operational Research*, 207(2):601–604.

Ozfatura, E., Gündüz, D., and Ulukus, S. (2019). Gradient coding with clustering and multi-message communication. In *2019 IEEE Data Science Workshop (DSW)*, pages 42–46. IEEE.

Ozfatura, E., Ulukus, S., and Gündüz, D. (2020). Straggler-aware distributed learning: communication–computation latency trade-off. *Entropy*, 22(5).

Raviv, N., Tamo, I., Tandon, R., and Dimakis, A. G. (2020). Gradient coding from cyclic MDS codes and expander graphs. *IEEE Transactions on Information Theory*, 66(12):7475–7489.

Reisizadeh, A., Prakash, S., Pedarsani, R., and Avestimehr, A. S. (2019). Coded computation over heterogeneous clusters. *IEEE Trans. Inf. Theory*, (7):4227–4242.

Sakorikar, A. and Wang, L. (2022). Soft BIBD and product gradient codes. *IEEE J. Sel. Areas Inf. Theory*, 3(2):229–240.

Sarmasarkar, S., Lalitha, V., and Karamchandani, N. (2023). On gradient coding with partial recovery. *IEEE Trans. on Commun.*, (2):644–657.

Tandon, R., Lei, Q., Dimakis, A. G., and Karampatziakis, N. (2017). Gradient coding: Avoiding stragglers in distributed learning. In *International Conference on Machine Learning*, pages 3368–3376. PMLR.

Wang, H., Charles, Z., and Papailiopoulos, D. (2019a). ErasureHead: Distributed gradient descent without delays using approximate gradient coding. *arXiv preprint arXiv:1901.09671*.

Wang, H., Guo, S., Tang, B., Li, R., Yang, Y., Qu, Z., and Wang, Y. (2022). Heterogeneity-aware gradient coding for tolerating and leveraging stragglers. *IEEE Trans. Comput.*, 71(4):779–794.

Wang, S., Liu, J., and Shroff, N. (2019b). Fundamental limits of approximate gradient coding. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(3):1–22.

Ye, M. and Abbe, E. (2018). Communication-computation efficient gradient coding. In *International Conference on Machine Learning*, pages 5610–5619. PMLR.