Coded Computing: Mitigating Fundamental Bottlenecks in Large-Scale Distributed Computing and Machine Learning

Other titles in Foundations and Trends® in Communications and Information Theory

Cache Optimization Models and Algorithms
Georgios Paschos, George Iosifidis and Giuseppe Caire
ISBN: 978-1-68083-702-5

Lattice-Reduction-Aided and Integer-Forcing Equalization: Structures, Criteria, Factorization, and Coding
Robert F. H. Fischer, Sebastian Stern and Johannes B. Huber
ISBN: 978-1-68083-644-8

Group Testing: An Information Theory Perspective
Matthew Aldridge, Oliver Johnson and Jonathan Scarlett
ISBN: 978-1-68083-596-0

Sparse Regression Codes
Ramji Venkataramanan, Sekhar Tatikonda and Andrew Barron
ISBN: 978-1-68083-580-9

Coded Computing: Mitigating Fundamental Bottlenecks in Large-Scale Distributed Computing and Machine Learning

Songze Li
University of Southern California
USA
songzeli@usc.edu

Salman Avestimehr
University of Southern California
USA
avestimehr@ee.usc.edu

now
the essence of knowledge
Boston — Delft

Foundations and Trends® in Communications and Information Theory

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

S. Li and S. Avestimehr. *Coded Computing: Mitigating Fundamental Bottlenecks in Large-Scale Distributed Computing and Machine Learning*. Foundations and Trends® in Communications and Information Theory, vol. 17, no. 1, pp. 1–148, 2020.

ISBN: 978-1-68083-705-6

Foundations and Trends® in Communications and Information Theory
Volume 17, Issue 1, 2020

Editorial Board

Editorial Scope

Topics

Foundations and Trends® in Communications and Information Theory publishes survey and tutorial articles in the following topics:

- Coded modulation
- Coding theory and practice
- Communication complexity
- Communication system design
- Cryptology and data security
- Data compression
- Data networks
- Demodulation and Equalization
- Denoising
- Detection and estimation
- Information theory and statistics
- Information theory and computer science
- Joint source/channel coding
- Modulation and signal design
- Multiuser detection
- Multiuser information theory
- Optical communication channels
- Pattern recognition and learning
- Quantization
- Quantum information processing
- Rate-distortion theory
- Shannon theory
- Signal processing for communications
- Source coding
- Storage and recording codes
- Speech and Image Compression
- Wireless Communications

Information for Librarians

Foundations and Trends® in Communications and Information Theory, 2020, Volume 17, 4 issues. ISSN paper version 1567-2190. ISSN online version 1567-2328 . Also available as a combined paper and online subscription.

Contents

Coded Computing: Mitigating Fundamental Bottlenecks in Large-Scale Distributed Computing and Machine Learning

Songze Li[1] and Salman Avestimehr[2]

[1] *University of Southern California, USA; songzeli@usc.edu*
[2] *University of Southern California, USA; avestimehr@ee.usc.edu*

ABSTRACT

We introduce the concept of "coded computing", a novel computing paradigm that utilizes coding theory to effectively inject and leverage data/computation redundancy to mitigate several fundamental bottlenecks in large-scale distributed computing, namely communication bandwidth, straggler's (i.e., slow or failing nodes) delay, privacy and security bottlenecks. More specifically, for MapReduce based distributed computing structures, we propose the "Coded Distributed Computing" (CDC) scheme, which injects redundant computations across the network in a structured manner, such that in-network coding opportunities are enabled to substantially slash the communication load to shuffle the intermediate computation results. We prove that CDC achieves the optimal tradeoff between computation and communication, and demonstrate its impact on a wide range of distributed computing systems from cloud-based datacenters to mobile edge/fog computing platforms.

Songze Li and Salman Avestimehr (2020), "Coded Computing: Mitigating Fundamental Bottlenecks in Large-Scale Distributed Computing and Machine Learning", Foundations and Trends® in Communications and Information Theory: Vol. 17, No. 1, pp 1–148. DOI: 10.1561/0100000103.

Secondly, to alleviate the straggler effect that prolongs the executions of distributed machine learning algorithms, we utilize the ideas from error correcting codes to develop "Polynomial Codes" for computing general matrix algebra, and "Lagrange Coded Computing" (LCC) for computing arbitrary multivariate polynomials. The core idea of these proposed schemes is to apply coding to create redundant data/computation scattered across the network, such that completing the overall computation task only requires a subset of the network nodes returning their local computation results. We demonstrate the optimality of Polynomial Codes and LCC in minimizing the computation latency, by proving that they require the least number of nodes to return their results.

Finally, we illustrate the role of coded computing in providing security and privacy in distributed computing and machine learning. In particular, we consider the problems of secure multiparty computing (MPC) and privacy-preserving machine learning, and demonstrate how coded computing can be leveraged to provide efficient solutions to these critical problems and enable substantial improvements over the state of the art.

To illustrate the impact of coded computing on real world applications and systems, we implement the proposed coding schemes on cloud-based distributed computing systems, and significantly improve the run-time performance of important benchmarks including distributed sorting, distributed training of regression models, and privacy-preserving training for image classification. Throughout this monograph, we also highlight numerous open problems and exciting research directions for future work on coded computing.

1

Introduction

Recent years have witnessed a rapid growth of large-scale machine learning and big data analytics, facilitating the developments of data-intensive applications like voice/image recognition, real-time mapping services, autonomous driving, social networks, and augmented/virtual reality. These applications are supported by cloud infrastructures composed of large datacenters. Within a datacenter, a massive amount of users' data are stored distributedly on hundreds of thousands of low-end commodity servers, and any application of big data analytics has to be performed in a distributed manner within or across datacenters. This has motivated the fast development of scalable, interpretable, and fault-tolerant distributed computing frameworks (see, e.g., [42, 56, 129, 171, 175]) that efficiently utilize the underlying hardware resources (e.g., CPUs and GPUs).

In this monograph, we focus on addressing the following three major performance bottlenecks for large-scale distributed machine learning/data analytics systems.

- *Communication bottleneck*: Excessive data shuffling between compute nodes.

- *Straggler bottleneck*: Delay of computation caused by slow or failing compute nodes, which are referred to as stragglers.
- *Security bottleneck*: Vulnerability to eavesdroppers and attackers.

To alleviate these bottlenecks, we take an unorthodox approach by employing ideas and techniques from coding theory, and propose the concept of "coded computing", whose core spirit is described as follows.

> Exploiting coding theory to optimally **inject** and **leverage** data/task **redundancy** in distributed computing systems, creating coding opportunities to overcome communication, straggler, and security bottlenecks.

Guided by this core spirit, we propose and evaluate a rich class of coded distributed computing frameworks, for computation tasks ranging from general MapReduce primitives to fundamental polynomial algebra, and for computation systems ranging from conventional cloud-based datacenters to emerging (mobile) edge/fog computing systems. In the rest of this section, we describe our contributions on utilizing coded computing to mitigate the communication, straggler, and security bottlenecks, and discuss related works.

Before proceeding with the overview of coded computing, we would like to also point out an important remark. In order to enable redundant computations in coded computing, we need to also redundantly store the datasets over which the computations are done. This would impose a certain communication and storage cost to the system. However, in many applications this cost can be ignored due to the following two reasons. First, in many computation scenarios we are interested in *many* computations over the *same* dataset (e.g., database query, keyword search, loss calculation in machine learning, etc.). In those cases the cost of encoding and redundantly storing the dataset in the network can be amortized over many computations. Second, in many scenarios, the encoding and storage of the dataset can happen at a different time than the desired computations. For example, one can use the off-peak

network times to properly encode and store the dataset, so as to be ready for computations during the peak times.

1.1 Coding for Bandwidth Reduction

It is well known that communicating intermediate computation results (or data shuffling) is one of the major performance bottlenecks for various distributed computing applications, including self-join [3], TeraSort [58], and many machine learning algorithms [36]. For instance, in a Facebook's Hadoop cluster, it is observed that 33% of the overall job execution time is spent on data shuffling [36]. Also as is observed in [174], 70% of the overall job execution time is spent on data shuffling when running a self-join job on Amazon EC2 clusters. This bottleneck is becoming worse for training deep neural networks with millions of model parameters (e.g., ResNet-50 [65]), where partial gradients with millions of entries are computed at distributed computing nodes and passed across the network to update the model parameters [33].

Many optimization methods have been proposed to alleviate the communication bottleneck in distributed computing systems. For example, from the algorithm perspective, when the function that reduces the final result is commutative and associative, it was proposed to pre-combine intermediate results before data shuffling, cutting off the amount of data movement [42, 125]. On the other hand, from the system perspective, optimal flow scheduling across network paths has been designed to accelerate the data shuffling process [53, 57], and distributed cache memories were utilized to speed up the data transfer between consecutive computation stages [49, 173]. Recently, motivated by the fact that training algorithms exhibit tolerance to precision loss of intermediate results, a family of lossy compression (or quantization) algorithms for distributed learning systems have been developed to compress the intermediate results (e.g., gradients), and then the compressed results are communicated to achieve a smaller bandwidth consumption (see, e.g., [5, 19, 138, 158]).

The above mentioned approaches are designed for specific computations and network structures, and difficult to generalize to handle arbitrary computation tasks. To overcome these difficulties, we focus

on a general MapReduce-type distributed computing model [42], and propose to utilize coding theory to slash the communication bottleneck in running MapReduce applications. In particular, in this computing model, each input file is mapped into multiple intermediate values, one for each of the output functions, and the intermediate values from all input files for each output function are collected and reduced to the final output result. For this model, we propose a coded computing scheme, named "coded distributed computing" (CDC), which trades extra local computations for more network bandwidth. For some design parameter r, which is termed as "communication load", the CDC scheme places and maps each of the input files on r carefully chosen distributed computing nodes, injecting r times more local computations. In return, the redundant computations produce side information at the nodes, which enable the opportunities to create *coded multicast* packets during data shuffling that are simultaneously useful for r nodes. That is, the CDC scheme trades r times more redundant computations for an r times reduction in the communication load. Furthermore, we theoretically demonstrate that this inversely proportional tradeoff between computation and communication achieved by CDC is fundamental, i.e., for a given computation load, no other schemes can achieve a lower communication load than that achieved by CDC.

Having proposed the CDC framework and characterized its optimal performance in trading extra computations for communication bandwidth, we also empirically demonstrate its impact on speeding up practical workloads. In particular, we integrate the principle of CDC into the widely used Hadoop sorting benchmark, `TeraSort` [62], developing a novel distributed sorting algorithm, named `CodedTeraSort`. At a high level, `CodedTeraSort` imposes *structured* redundancy in the input data, enabling in-network coding opportunities to significantly slash the load of data shuffling, which is a major bottleneck of the run-time performance of `TeraSort`. Through extensive experiments on Amazon EC2 [7] clusters, we demonstrate that `CodedTeraSort` achieves $1.97\times\sim3.39\times$ speedup over `TeraSort`, for typical settings of interest. Despite the extra overhead imposed by coding (e.g., generation of the coding plan, data encoding and decoding), the practically

achieved performance gain approximately matches the gain theoretically promised by `CodedTeraSort`.

Beyond the conventional wireline networks in datacenters, we also introduce the concept of coded computing to tackle the scenarios of mobile edge/fog computing, where the communication bottleneck is even more severe due to the low data rate and the large number of mobile users. In particular, we consider a wireless distributed computing platform, which is composed of a cluster of mobile users scattered around the network edge, connected wirelessly through an access point. Each user has a limited storage and processing capability, and the users have to collaborate to satisfy their computational needs that require processing a large dataset. This ad hoc computing model, in contrast to the centralized cloud computing model, is becoming increasingly common in the emerging edge computing paradigm for Internet-of-Things (IoT) applications [28, 32]. For this model, following the principle of the CDC scheme, we propose a coded wireless distributed computing (CWDC) scheme that jointly designs the local storage and computation for each user, and the communication schemes between the users. The CWDC scheme achieves a constant bandwidth consumption that is independent of the number of users in the network, which leads to a *scalable* design of the platform that can simultaneously accommodate an arbitrary number of users. Moreover, for a more practically important decentralized setting, in which each user needs to decide its local storage and computation independently without knowing the existence of any other participating users, we extend the CWDC scheme to achieve a bandwidth consumption that is very close to that of the centralized setting.

1.2 Coding for Straggler Mitigation

Other than data shuffling, another major performance bottleneck of distributed computing applications is the effect of stragglers. That is, the execution time of a computation consisting of multiple parallel tasks is limited by the slowest task run on the straggling processor. These stragglers significantly slow down the overall computations, and have been widely observed in distributed computing systems (see, e.g., [8, 41, 172]). For instance, it was experimentally demonstrated in [172] that

this straggler effect can prolong the job execution time by as much as five times.

Conventionally, in the original open-source implementation of Hadoop MapReduce [10], the stragglers are constantly detected and the slow tasks are speculatively restarted on other available nodes. Following this idea of straggler detection, more timely straggler detection algorithms and better scheduling algorithms have been developed to further alleviate the straggler effect (see, e.g., [9, 172]). Apart from straggler detection and speculative restart, another straggler mitigation technique is to schedule the clones of the same task (see, e.g., [8, 30, 55, 91, 139]). The underlying idea of cloning is to execute redundant tasks such that the computation can proceed when the results of the fast-responding clones have returned. Recently, it has been proposed to utilize error correcting codes for straggler mitigation in distributed matrix-vector multiplication [47, 89, 96, 106]. The main idea is to partition the data matrix into K batches, and then generate N coded batches using the maximum-distance-separable (MDS) code [101], and assign multiplication with each of the coded batches to a worker node. Benefiting from the "any K of N" property of the MDS code, the computation can be accomplished as long as *any* K fastest nodes have finished their computations, providing the system the robustness to up to $N - K$ arbitrary stragglers. This coded approach was shown to significantly outperform the state-of-the-art cloning approaches in straggler mitigation capability, and minimize the the overall computation latency.

Our first contribution on this topic is the development of optimal codes, named *polynomial codes*, to deal with stragglers in distributed high-dimensional matrix–matrix multiplication. More specifically, we consider a distributed matrix multiplication problem where we aim to compute $C = A^\top B$ from input matrices A and B. The computation is carried out using a distributed system with a master node and N worker nodes that can each stores a fixed fraction of A and B respectively (possibly in a coded manner). For this problem, we aim to design computation strategies that achieve the minimum possible *recovery threshold*, which is defined as the minimum number of workers that the master needs to wait for in order to compute C. While the prior works,

Table 1.1: Comparison of recovery threshold for distributed high-dimensional matrix multiplication, over a system consisting of a master node, and N worker nodes

	1D MDS Code	**Product Code**	**Polynomial Code**
Recovery threshold	$\Theta(N)$	$\Theta(\sqrt{N})$	$\Theta(1)$

i.e., the *one dimensional MDS code* (*1D MDS code*) in [89], and the *product code* in [90] apply MDS codes on the data matrices, they are sub-optimal in minimizing the recovery threshold. The main novelty and advantage of the proposed polynomial code is that, by carefully designing the algebraic structure of the coded storage at each worker, we create an MDS structure on the *intermediate computations*, instead of only the coded data matrices. This allows polynomial code to achieve order-wise improvement over state of the arts (see Table 1.1). We also prove the optimality of polynomial code by showing that it achieves the information-theoretic lower bound on the recovery threshold. As a by-product, we also prove the optimality of polynomial code under several other performance metrics considered in previous literature.

Going beyond matrix algebra, we also study the straggler mitigation strategies for scenarios where the function of interest is an *arbitrary multivariate polynomial* of the input dataset. This significantly broadens the scope of the problem to cover many computations of interest in machine learning, such as various gradient and loss-function computations in learning algorithms and tensor algebraic operations (e.g., low-rank tensor approximation). In particular, we consider a computation task for which the goal is to compute a function f over a large dataset $X = (X_1, \ldots, X_K)$ to obtain K outputs $Y_1 = f(X_1), \ldots, Y_K = f(X_K)$. The computation is carried over a system consisting of a master node and N worker nodes. Each worker i stores a coded dataset $\tilde{X}_i$ generated from X, computes $f(\tilde{X}_i)$, and sends the obtained result to the master. The master decodes the output $Y_1, \ldots, Y_K$ from the computation results of the group of the fastest workers.

For this setting, a naive repetition scheme would repeat the computation for each data block X_k onto N/K workers, yielding a recovery threshold of $N - N/K + 1 = \Theta(N)$. We propose the "Lagrange Coded

Computing" (LCC) framework to minimize the recovery threshold. In particular, denoting the degree of the function f as $\deg f$, LCC promises the recovery of all output results at the master as soon as it receives computation results from $(K-1)\deg f+1$ workers. That is, LCC achieves a recovery threshold of $(K-1)\deg f+1$. Note that the recovery threshold of LCC is $\Theta(K)$, which is independent of the total number of workers N. Hence, as the network expands (i.e., N grows), compared with the naive repetition scheme, LCC benefits much more from the abundant computation resources in alleviating the negative effects caused by slow or failed nodes, which leads to a much lower computation latency. In fact, we demonstrate through proving a matching information-theoretic converse that LCC achieves the minimum possible recovery threshold among all distributed computing schemes.

The key idea of LCC is to encode the input dataset using the well-known Lagrange interpolation polynomial, in order to create computation redundancy in a novel coded form across the workers. This redundancy can then be exploited to provide resiliency to stragglers. Additionally, we emphasize on the following two salient features of the data encoding of LCC:

- *Universal*: The data encoding is oblivious of the output function f. Therefore, the coded data placement can be performed offline without knowing which operations will be applied on the data.

- *Incremental*: When new data become available and coded data batches need to be updated, we only need to encode the new data and append them to the previously coded batches, instead of accessing the entire uncoded data and re-encoding them to update the coded data.

Finally, we specialize our general theoretical guarantees for LCC in the context of least-squares linear regression, which is one of the elemental learning tasks, and demonstrate its performance gain by optimally suppressing stragglers. Leveraging the algebraic structure of gradient computations, several strategies have been developed recently to exploit data and gradient coding for straggler mitigation in the training process (see, e.g., [78, 89, 94, 106, 147]). We implement LCC

for regression on Amazon EC2 clusters, and empirically compare its performance with the conventional uncoded approaches, and two state-of-the-art straggler mitigation schemes: gradient coding (GC) [63, 127, 147, 164] and matrix-vector multiplication (MVM) based approaches [89, 106]. Our experiment results demonstrate that compared with the uncoded scheme, LCC improves the run-time by $6.79\times\sim13.43\times$. Compared with the GC scheme, LCC improves the run-time by $2.36\times\sim4.29\times$. Compared with the MVM scheme, LCC improves the run-time by $1.01\times\sim12.65\times$.

1.3 Coding for Security and Privacy

Data privacy has become a major concern in the information age. The immensity of modern datasets has popularized the use of third-party cloud services, and as a result, the threat of privacy infringement has increased dramatically. In order to alleviate this concern, techniques for private computation are essential [25, 38, 102, 114]. Additionally, third-party service providers often have an interest in the result of the computation, and might attempt to alter it for their benefit [23, 24]. In particular, we consider a common and important scenario where a user wishes to disperse computations over a large network of workers, subject to the following privacy and security constraints.

- *Privacy constraint*: Sets of colluding workers cannot infer anything about the input dataset in the information-theoretic sense.
- *Security constraint*: The computation must be accomplished successfully even if some workers return purposefully erroneous results.

The problem of secure and private distributed computing has been studied extensively from various perspectives in the past, mainly within the scope of secure *multiparty computation* (MPC) [18, 37, 38, 64]. Most notably, the celebrated BGW scheme [18], which adapts the Shamir secret sharing scheme [141] to the realm of computation, has been a reference point for several decades. The key idea of BGW scheme is to view any computation task as composed by linear and bilinear functions

to be handled in multiple rounds. It applies the Shamir secret sharing scheme to generate coded data shares with security guarantees, and computes the function on the coded shares. We generalize the proposed Lagrange Coded Computing (LCC) scheme designed for straggler mitigation purposes to also provide security and privacy guarantees to MPC systems. Specifically, similarly as before, we consider the problem of evaluating a multivariate polynomial f over dataset $X = (X_1, \ldots, X_K)$. We employ a distributed computing network with a master and N workers, and aim to compute $Y_1 = f(X_1), \ldots, Y_K = f(X_K)$. For this computing system, we propose modifications to the data encoding and computation decoding processes of LCC, and demonstrate that LCC provides a T-private and A-secure computation of f (i.e., keeping the dataset private amidst collusion of any T workers, and the computation secure amidst the presence of A Byzantine adversarial workers), for any pair (T, A) satisfying

$$N \geq (K + T - 1) \deg f + 2A + 1. \tag{1.1}$$

Furthermore, we also demonstrate that LCC achieves an optimal tradeoff between privacy and security, and requires a minimal amount of added randomness to preserve privacy.

In the presence of Byzantine workers, a subset of computation results received at the master can be arbitrarily erroneous. In order to correctly recover the computation results, during the decoding process, instead of mere polynomial interpolation, the master applies an error correcting decoding algorithm for a Reed–Solomon code of dimension $(K - 1) \deg(f) + 1$ and length N. This allows LCC to tolerate A malicious workers as long as $2A \leq N - (K - 1) \deg f - 1$. Obtaining information-theoretic privacy against colluding workers, i.e., keeping small sets of workers oblivious to the dataset does not require altering the encoding nor decoding algorithm. However, prior to encoding, the dataset X is padded by T random elements $R_1, \ldots, R_T$, where T is the maximum size of sets of workers that cannot infer anything about X.

We note from (1.1) that when $N \geq (K + T - 1) \deg f + 2A + 1$, the LCC scheme *simultaneously* achieves

1. *Resiliency* against $N - ((K + T - 1) \deg f + 2A + 1)$ straggler workers that prolong computations;

2. *Security* against A malicious workers, with no computational restriction, that deliberately send erroneous data in order to affect the computation for their benefit; and

3. *(Information-theoretic) Privacy* of the dataset amidst possible collusion of up to T workers.

We also note that the number of workers the master needs to wait for does *not* scale with the total number of workers N, hence the key property of LCC is that adding one additional worker can increase its resiliency to stragglers by 1, or increase its robustness to malicious worker by 1/2, while maintaining the privacy constraint. Hence, this result essentially extends the well-known optimal scaling of error-correcting codes (i.e., adding one parity can provide robustness against 1 erasure or 1/2 error in optimal maximum distance separable codes) to the distributed computing paradigm. Compared with the state-of-the-art BGW-based designs, we also show that LCC significantly improves the storage, communication, and secret-sharing overhead needed for secure and private multiparty computing (see Table 1.2).

Finally, we will also discuss the problem of privacy-preserving machine learning. In particular, we consider an application scenario in which a data-owner (e.g., a hospital) wishes to train a logistic regression model by offloading the large volume of data (e.g., healthcare records) and computationally-intensive training tasks (e.g., gradient computations) to N machines over a cloud platform, while ensuring that any collusions between T out of N workers do not leak information about

Table 1.2: Comparison between BGW based designs and LCC. The computational complexity is normalized by that of evaluating f; randomness, which refers to the number of random entries used in encoding functions, is normalized by the length of X_i

	BGW	**LCC**
Complexity/worker	K	1
Frac. data/worker	1	$1/K$
Randomness	KT	T
Min. num. of workers	$2T+1$	$\deg f \cdot (K+T-1)+1$

the dataset. We discuss a recently proposed scheme [145], named CodedPrivateML, which leverages coded computing for this problem. More specifically, we show how one can leverage coded computing to both provide strong information-theoretic privacy guarantees and enable fast training by distributing the training computation load effectively across several workers.

1.4 Related Works

The problem of characterizing the minimum communication for distributed computing has been previously considered in several settings in both computer science and information theory literature. In [163], a basic computing model is proposed, where two parities have x and y and aim to compute a Boolean function $f(x, y)$ by exchanging the minimum number of bits between them. Also, the problem of minimizing the required communication for computing the modulo-two sum of distributed binary sources with symmetric joint distribution was introduced in [85]. Following these two seminal works, a wide range of communication problems in the scope of distributed computing have been studied (cf. [16, 88, 116, 120, 121, 126]).

The idea of efficiently creating and exploiting *coded multicasting* for bandwidth reduction was initially proposed in the context of cache networks in [104, 105], and extended in [72, 79], where caches prefetch part of the content in a way to enable coding during the content delivery, minimizing the network traffic. Generally speaking, we can also view the data shuffling of the considered distributed computing framework as an instance of the index coding problem [15, 20], in which a central server aims to design a broadcast message (code) with minimum length to simultaneously satisfy the requests of all the clients, given the clients' side information stored in their local caches. Note that while a randomized linear network coding approach (see e.g., [2, 66, 83]) is sufficient to implement any multicast communication where messages are intended by all receivers, it is generally sub-optimal for index coding problems where every client requests different messages. Although the index coding problem is still open in general, for the considered distributed computing scenario where we are given the

flexibility of designing Map computation (thus the flexibility of designing side information), we can prove *tight* lower bounds on the minimum communication loads, demonstrating the optimality of the proposed Coded Distributed Computing scheme.

We would like to also point out that the main focus of the index coding problem/literature is to design the optimal delivery scheme for a given (often fixed) side information at the nodes. On the other hand, the key novelty of our scheme/framework is the *design of side information (or redundant computations)* at the nodes in order to maximize the index coding (or coded multicast) opportunities. So, while index coding focused on the design of best delivery strategies, we focus on the design of best side information structure. In that sense they are complementary to each other and we can leverage any of the delivery schemes developed in the index coding literature (e.g., the schemes based on local clique cover [142], partial and fractional clique cover [1, 20], interference alignment [107], and many other schemes [11]) in the shuffling phase.

Other than designing coded computing strategies for bandwidth reduction, there has recently been a surge of interest in developing coded computing frameworks for straggler mitigation. Initiated in [89], many following works has focused on designing data encoding strategies, mainly inspired by the concepts of erasure/error correcting codes for communication systems, to minimize the recovery threshold, in distributed computation of matrix-vector and matrix–matrix multiplications (e.g., [47, 52, 155, 167, 168]). Coded computing also finds its application in distributed machine learning, specifically for running distributed stochastic gradient descent (SGD) on a master/worker architecture. For general machine learning tasks, data encoding is not applicable due to the complicated structure of gradient computation (e.g., gradients are computed numerically using back-propagation for deep neural networks). In this scenario, "gradient coding" techniques [63, 94, 127, 147, 164] have been designed to code across partial gradients computed from uncoded data, such that the master can recover the total gradient as the sum of all partial gradients, after receiving the computation results from the minimum possible number of workers.

The proposed Lagrange Coded Computing (LCC) scheme improves and expands these prior works in a few aspects: *Generality* – LCC

significantly expands the computation class for which we know how to design coded computing to go beyond linear and bilinear computations that have so far been the main research focus. In particular, it can be applied to more general multivariate polynomial computations that arise in machine learning applications. *Universality* – once the data has been coded, any polynomial up to a certain degree can be computed distributedly via LCC. In other words, data encoding of LCC can be *universally* used for any polynomial computation. This is in stark contrast to previous task-specific coding techniques in the literature. *Security and privacy* – other than straggler mitigation, LCC also extends the application of coded computing to secure and private computing for general polynomial computations.

The security and privacy issue of distributed computing has been extensively studied in the literature of secure multiparty computing (MPC) and secure machine learning/data mining, [18, 37, 38, 64, 68, 102]. As a representative example, we briefly describe the celebrated BGW MPC scheme [18]. Given data inputs $\{X_i\}_{i=1}^K$, the problem is to compute outputs $\{f(X_i)\}_{i=1}^K$ using N workers in a privacy-preserving manner (i.e., colluding workers cannot infer anything about the dataset using their local data). To do that, BGW first uses Shamir's scheme [141] to encode each X_i as a polynomial $P_i(z) = X_i + Z_{i,1}z + \cdots + Z_{i,T}z^T$, where $Z_{i,j}$'s are i.i.d. uniformly random variables and T is the number of colluding workers that should be tolerated. Then, each worker ℓ stores the coded data $\{P_i(\alpha_\ell)\}_{i=1}^K$, for a distinct α_ℓ, and computes $\{f(P_i(\alpha_\ell))\}_{i=1}^K$. Hence, for each i, each worker provides the evaluation of the degree-$(\deg f \cdot T)$ polynomial $f(P_i(z))$ at a distinct point α_ℓ. The polynomial $f(P_i(z))$ can be interpolated using computation results from $\deg f \cdot T + 1$ workers, and $f(X_i)$ is obtained by taking the constant term of $f(P_i(z))$.[1] In the proposed LCC scheme, instead of hiding X_i's individually in data encoding, we code *across* X_i's together with some added random inputs. This gives rise to significant reduction on storage overhead, computational complexity, and the amount of padded

[1] It is also possible to use the conventional multi-round BGW, which only requires $N \geq 2T + 1$ workers to ensure T-privacy. However, multiple rounds of computation and communication ($\Omega(\log(\deg f))$ rounds) are needed, which further increases its communication overhead.

randomness. However, under the same condition, LCC scheme requires $N \geq \deg f \cdot (K + T - 1) + 1$ number of workers, which is larger than that of the BGW scheme. So, in some sense LCC achieves reduction in storage overhead, computational complexity, and the amount of padded randomness, at the expense of increasing the number of needed workers (or reducing the fraction of Byzantine workers that can be tolerated). We refer to Table 1.2 for a detailed comparison between BGW and LCC.

Coding techniques have been recently developed to provide security and privacy guarantees to distributed computing. Specifically, staircase codes [21] were proposed to combat stragglers in linear computations (e.g., matrix-vector multiplications) while preserving data privacy, improving the computation latency of the conventional secure computing schemes based on secret sharing [110, 141]. The proposed LCC scheme generalizes the staircase codes beyond linear computations. Even for the linear case, LCC guarantees data privacy against T colluding workers by introducing less randomness than [21] (T rather than $TK/(K-T)$). Beyond linear computations, a recent work [117] has combined ideas from the BGW scheme and the polynomial code [167] to form *polynomial sharing*, a private coded computing scheme for arbitrary matrix polynomials. However, polynomial sharing inherits the undesired BGW property of performing a communication round for *every* bilinear operation in the polynomial; a feature that drastically reduces communication efficiency, and is circumvented by the one-shot approach of LCC. *DRACO* [31] was proposed as a secure distributed training algorithm that is robust to Byzantine workers. Since DRACO is designed for general gradient computations, it employs a blackbox approach, i.e., the coding is applied on the gradients computed from uncoded data, but not on the data itself, which is similar to the gradient coding techniques [63, 94, 127, 147, 164] designed primarily for stragglers. For this approach, [31] show that a $2A + 1$ *multiplicative* factor of redundant computations is needed to be robust to A Byzantine workers. For the proposed LCC however, the blackbox approach is disregarded in favor of an algebraic one, and consequently, a $2A$ *additive* factor suffices.

References

[1] Agarwal, A. and A. Mazumdar (2016). "Local partial clique and cycle covers for index coding". In: *2016 IEEE Globecom Workshops (GC Wkshps)*. 1–6.

[2] Ahlswede, R., N. Cai, S.-Y. R. Li, and R. W. Yeung (2000). "Network information flow". *IEEE Transactions on Information Theory*. 46(4): 1204–1216.

[3] Ahmad, F., S. T. Chakradhar, A. Raghunathan, and T. Vijaykumar (2012). "Tarazu: Optimizing MapReduce on heterogeneous clusters". *ACM SIGARCH Computer Architecture News*. 40(1): 61–74.

[4] Aktas, M. F., P. Peng, and E. Soljanin (2018). "Straggler mitigation by delayed relaunch of tasks". *ACM SIGMETRICS Performance Evaluation Review*. 45(2): 224–231.

[5] Alistarh, D., D. Grubic, J. Li, R. Tomioka, and M. Vojnovic (2017). "QSGD: Communication-efficient SGD via gradient quantization and encoding". *Advances in Neural Information Processing Systems (NIPS)*: 1707–1718.

[6] Alpatov, P., G. Baker, C. Edwards, J. Gunnels, G. Morrow, J. Overfelt, R. van de Geijn, and Y.-J. J. Wu (1997). "PLAPACK: Parallel linear algebra package design overview". In: *Proceedings of the 1997 ACM/IEEE Conference on Supercomputing*. ACM. 1–16.

[7] "Amazon Elastic Compute Cloud (EC2)" (n.d.). https://aws.amazon.com/ec2/. Accessed on Jan. 30, 2018.

[8] Ananthanarayanan, G., A. Ghodsi, S. Shenker, and I. Stoica (2013). "Effective straggler mitigation: Attack of the clones". In: *10th USENIX Symposium on Networked Systems Design and Implementation*. 185–198.

[9] Ananthanarayanan, G., S. Kandula, A. G. Greenberg, I. Stoica, Y. Lu, B. Saha, and E. Harris (2010). "Reining in the outliers in map-reduce clusters using Mantri". In: *OSDI*. Vol. 10. No. 1. 24.

[10] "Apache Hadoop" (n.d.). http://hadoop.apache.org. Accessed on Jan. 30, 2018.

[11] Arbabjolfaei, F. and Y. Kim (2018). "Fundamentals of index coding". *Foundations and Trends® in Communications and Information Theory*. 14(3–4): 163–346.

[12] Attia, M. A. and R. Tandon (2016). "Information theoretic limits of data shuffling for distributed learning". In: *IEEE Global Communications Conference (GLOBECOM)*. 1–6.

[13] Baktir, S. and B. Sunar (2006). "Achieving efficient polynomial multiplication in fermat fields using the fast fourier transform". In: *Proceedings of the 44th Annual Southeast Regional Conference*. ACM. 549–554.

[14] Ballard, G., E. Carson, J. Demmel, M. Hoemmen, N. Knight, and O. Schwartz (2014). "Communication lower bounds and optimal algorithms for numerical linear algebra". *Acta Numerica*. 23: 1–155.

[15] Bar-Yossef, Z., Y. Birk, T. Jayram, and T. Kol (2011). "Index coding with side information". *IEEE Transactions on Information Theory*. 57(3): 1479–1494.

[16] Becker, K. and U. Wille (1998). "Communication complexity of group key distribution". In: *Proceedings of the 5th ACM Conference on Computer and Communications Security*. 1–6.

[17] Beerliova-Trubiniova, Z. and M. Hirt (2008). "Perfectly-secure MPC with linear communication complexity". In: *Theory of Cryptography Conference*. Springer. 213–230.

[18] Ben-Or, M., S. Goldwasser, and A. Wigderson (1988). "Completeness theorems for non-cryptographic fault-tolerant distributed computation". In: *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*. ACM. 1–10.

[19] Bernstein, J., Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar (2018). "signSGD: Compressed optimisation for non-convex problems". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. Vol. 80. *Proceedings of Machine Learning Research*. Stockholmsmässan, Stockholm Sweden: PMLR. 560–569. URL: http://proceedings.mlr.press/v80/bernstein18a.html.

[20] Birk, Y. and T. Kol (2006). "Coding on demand by an informed source (ISCOD) for efficient broadcast of different supplemental data to caching clients". *IEEE Transactions on Information Theory*. 52(6): 2825–2830.

[21] Bitar, R., P. Parag, and S. E. Rouayheb (2020). "Minimizing latency for secure coded computing using secret sharing via staircase codes". *IEEE Transactions on Communications*.

[22] Blackford, L. S., J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley (1997). *ScaLAPACK Users' Guide*. SIAM.

[23] Blanchard, P., E.-M. El Mhamdi, R. Guerraoui, and J. Stainer (2017a). "Byzantine-tolerant machine learning". *preprint arXiv:1703.02757*.

[24] Blanchard, P., E.-M. El Mhamdi, R. Guerraoui, and J. Stainer (2017b). "Machine learning with adversaries: Byzantine tolerant gradient descent". In: *Advances in Neural Information Processing Systems*. 118–128.

[25] Bogdanov, D., S. Laur, and J. Willemson (2008). "Sharemind: A framework for fast privacy-preserving computations". In: *Proceedings of the 13th European Symposium on Research in Computer Security: Computer Security. ESORICS '08*. Spain: Springer-Verlag. 192–206.

[26] Bonawitz, K., V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth (2016). "Practical secure aggregation for federated learning on user-held data". In: *Conference on Neural Information Processing Systems*.

[27] Bonawitz, K., V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth (2017). "Practical secure aggregation for privacy-preserving machine learning". In: *ACM SIGSAC Conference on Computer and Communications Security*. ACM. 1175–1191.

[28] Bonomi, F., R. Milito, J. Zhu, and S. Addepalli (2012). "Fog computing and its role in the internet of things". In: *Proceedings of the 1st Edition of the MCC Workshop on Mobile Cloud Computing*. ACM. 13–16.

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

[30] Chaubey, M. and E. Saule (2015). "Replicated data placement for uncertain scheduling". In: *IEEE International Parallel and Distributed Processing Symposium Workshop*. 464–472.

[31] Chen, L., H. Wang, Z. Charles, and D. Papailiopoulos (2018). "DRACO: Robust distributed training via redundant gradients". *e-print arXiv:1803.09877*.

[32] Chiang, M. and T. Zhang (2016). "Fog and IoT: An overview of research opportunities". *IEEE Internet of Things Journal*. 3(6): 854–864.

[33] Chilimbi, T. M., Y. Suzue, J. Apacible, and K. Kalyanaraman (2014). "Project Adam: Building an efficient and scalable deep learning training system". In: *11th USENIX Symposium on Operating Systems Design and Implementation*. Vol. 14. 571–582.

[34] Choi, B., J.-Y. Sohn, D.-J. Han, and J. Moon (2019). "Scalable network-coded PBFT consensus algorithm". In: *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 857–861.

[35] Choi, J., J. Dongarra, and D. Walker (1996). "PB-BLAS: A set of parallel block basic linear algebra subprograms". *Concurrency: Practice and Experience*. 8(7): 517–535.

[36] Chowdhury, M., M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica (2011). "Managing data transfers in computer clusters with orchestra". *ACM SIGCOMM Computer Communication Review*. 41(4): 98–109.

[37] Cramer, R., I. Damgård, and J. B. Nielsen (2001). "Multiparty computation from threshold homomorphic encryption". In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 280–300.

[38] Cramer, R., I. B. Damgrd, and J. B. Nielsen (2015). *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press.

[39] Dalcin, L. D., R. R. Paz, P. A. Kler, and A. Cosimo (2011). "Parallel distributed computing using python". *Advances in Water Resources*. 34(9): 1124–1139.

[40] Damgård, I. and J. B. Nielsen (2007). "Scalable and unconditionally secure multiparty computation". In: *International Cryptology Conference*. Springer. 572–590.

[41] Dean, J. and L. A. Barroso (2013). "The tail at scale". *Communications of the ACM*. 56(2): 74–80.

[42] Dean, J. and S. Ghemawat (2004). "MapReduce: Simplified data processing on large clusters". *Sixth USENIX Symposium on Operating System Design and Implementation*.

[43] Demmel, J., L. Grigori, M. Hoemmen, and J. Langou (2012). "Communication-optimal parallel and sequential QR and LU factorizations". *SIAM Journal on Scientific Computing*. 34(1): A206–A239.

[44] Dimakis, A. G., J. Wang, and K. Ramchandran (2007). "Unequal growth codes: Intermediate performance and unequal error protection for video streaming". In: *Multimedia Signal Processing, 2007. MMSP 2007. IEEE 9th Workshop on*. IEEE. 107–110.

[45] "Distributed Algorithms and Optimization Lecture Notes" (n.d.). https://stanford.edu/~rezab/classes/cme323/S16/notes/Lecture16/Pregel_GraphX.pdf. Accessed on July 11, 2018.

[46] Dutta, S., Z. Bai, T. M. Low, and P. Grover (2019). "CodeNet: Training large scale neural networks in presence of soft-errors". *preprint arXiv:1903.01042*.

[47] Dutta, S., V. Cadambe, and P. Grover (2016). "Short-dot: Computing large linear transforms distributedly using coded short dot products". *Advances in Neural Information Processing Systems (NIPS)*: 2100–2108.

[48] Dutta, S., V. Cadambe, and P. Grover (2017). "Coded convolution for parallel and distributed computing within a deadline". In: *IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2403–2407.

[49] Ekanayake, J., H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox (2010). "Twister: A runtime for iterative MapReduce". *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. June: 810–818.

[50] Ezzeldin, Y. H., M. Karmoose, and C. Fragouli (2017). "Communication vs. distributed computation: An alternative trade-off curve". In: *IEEE Information Theory Workshop (ITW)*. IEEE. 279–283.

[51] Fahim, M. and V. R. Cadambe (2019). "Numerically stable polynomially coded computing". In: *2019 IEEE International Symposium on Information Theory (ISIT)*. 3017–3021.

[52] Fahim, M., H. Jeong, F. Haddadpour, S. Dutta, V. Cadambe, and P. Grover (2017). "On the optimal recovery threshold of coded matrix multiplication". In: *55th Annual Allerton Conference*. IEEE. 1264–1270.

[53] Al-Fares, M., S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat (2010). "Hedera: Dynamic flow scheduling for data center networks". *7th USENIX Symposium on Networked Systems Design and Implementation*. Apr.

[54] Ferdinand, N. and S. C. Draper (2018). "Hierarchical coded computation". In: *2018 IEEE International Symposium on Information Theory (ISIT)*. 1620–1624.

[55] Gardner, K., S. Zbarsky, S. Doroudi, M. Harchol-Balter, and E. Hyytia (2015). "Reducing latency via redundant requests: Exact analysis". *ACM SIGMETRICS Performance Evaluation Review*. 43(1): 347–360.

[56] Gemulla, R., E. Nijkamp, P. J. Haas, and Y. Sismanis (2011). "Large-scale matrix factorization with distributed stochastic gradient descent". In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 69–77.

[57] Greenberg, A., J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta (2009). "VL2: A scalable and flexible data center network". *ACM SIGCOMM Computer Communication Review*. 39(4): 51–62.

[58] Guo, Y., J. Rao, and X. Zhou (2013). "iShuffle: Improving Hadoop performance with shuffle-on-write". In: *Proceedings of the 10th International Conference on Autonomic Computing*. 107–117.

[59] Gupta, V., S. Wang, T. Courtade, and K. Ramchandran (2018). "OverSketch: Approximate matrix multiplication for the cloud". In: *2018 IEEE International Conference on Big Data (Big Data)*. 298–304.

[60] Guyon, I., S. Gunn, A. Ben-Hur, and G. Dror (2005). "Result analysis of the NIPS 2003 feature selection challenge". *Advances in Neural Information Processing Systems (NIPS)*: 545–552.

[61] Haddadpour, F. and V. R. Cadambe (2018). "Codes for distributed finite alphabet matrix-vector multiplication". In: *2018 IEEE International Symposium on Information Theory (ISIT)*. 1625–1629.

[62] "Hadoop TeraSort" (n.d.). https://hadoop.apache.org/docs/r2.7.1/api/org/apache/hadoop/examples/terasort/package-summary.html. Accessed on Jan. 30, 2018.

[63] Halbawi, W., N. Azizan-Ruhi, F. Salehi, and B. Hassibi (2017). "Improving distributed gradient descent using Reed–Solomon codes". *e-print arXiv:1706.05436*.

[64] Halpern, J. and V. Teague (2004). "Rational secret sharing and multiparty computation". In: *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*. ACM. 623–632.

[65] He, K., X. Zhang, S. Ren, and J. Sun (2016). "Deep residual learning for image recognition". *IEEE Conference on Computer Vision and Pattern Recognition*: 770–778.

[66] Ho, T., R. Koetter, M. Medard, D. R. Karger, and M. Effros (2003). "The benefits of coding over routing in a randomized setting". *IEEE International Symposium on Information Theory*. June: 442.

[67] Huang, K.-H. and J. A. Abraham (1984). "Algorithm-based fault tolerance for matrix operations". *IEEE Transactions on Computers*. C-33(6): 518–528.

[68] Huang, L., A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar (2011). "Adversarial machine learning". In: *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*. ACM. 43–58.

[69] Huang, W. (2017). "Coding for security and reliability in distributed systems". *PhD thesis*. California Institute of Technology.

[70] Jahani-Nezhad, T. and M. A. Maddah-Ali (2019). "CodedSketch: Coded distributed computation of approximated matrix multiplication". In: *2019 IEEE International Symposium on Information Theory (ISIT)*. 2489–2493.

[71] Jeong, H., T. M. Low, and P. Grover (2018). "Masterless coded computing: A fully-distributed coded FFT algorithm". In: *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. 887–894.

[72] Ji, M., G. Caire, and A. F. Molisch (2016). "Fundamental limits of caching in wireless D2D networks". *IEEE Transactions on Information Theory*. 62(2): 849–869.

[73] Joshi, G., E. Soljanin, and G. Wornell (2017). "Efficient redundancy techniques for latency reduction in cloud systems". *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*. 2(2): 12.

[74] Jou, J.-Y. and J. A. Abraham (1986). "Fault-tolerant matrix arithmetic and signal processing on highly concurrent computing structures". *Proceedings of the IEEE*. 74(5): 732–741.

[75] Kadhe, S., J. Chung, and K. Ramchandran (2019). "SeF: A secure fountain architecture for slashing storage costs in blockchains". *preprint arXiv:1906.12140*.

[76] Kairouz, P., H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, S. El Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao (2019). "Advances and open problems in federated learning". *preprint arXiv:1912.04977*.

[77] Kamra, A., V. Misra, J. Feldman, and D. Rubenstein (2006). "Growth codes: Maximizing sensor network data persistence". *ACM SIGCOMM Computer Communication Review*. 36(4): 255–266.

[78] Karakus, C., Y. Sun, S. Diggavi, and W. Yin (2017). "Straggler mitigation in distributed optimization through data encoding". *Advances in Neural Information Processing Systems (NIPS)*: 5440–5448.

[79] Karamchandani, N., U. Niesen, M. A. Maddah-Ali, and S. Diggavi (2014). "Hierarchical coded caching". *IEEE International Symposium on Information Theory*. June: 2142–2146.

[80] Kedlaya, K. S. and C. Umans (2011). "Fast polynomial factorization and modular composition". *SIAM Journal on Computing*. 40(6): 1767–1802.

[81] Kiamari, M., C. Wang, and A. S. Avestimehr (2017). "On heterogeneous coded distributed computing". *IEEE GLOBECOM*. Dec.

[82] Kim, S. and S. Lee (2009). "Improved intermediate performance of rateless codes". In: *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on*. Vol. 3. IEEE. 1682–1686.

[83] Koetter, R. and M. Medard (2003). "An algebraic approach to network coding". *IEEE/ACM Transactions on Networking*. 11(5): 782–795.

[84] Konstantinidis, K. and A. Ramamoorthy (2018). "Leveraging coding techniques for speeding up distributed computing". *e-print arXiv:1802.03049*.

[85] Korner, J. and K. Marton (1979). "How to encode the modulo-two sum of binary sources". *IEEE Transactions on Information Theory*. 25(2): 219–221.

[86] Kosaian, J., K. Rashmi, and S. Venkataraman (2018). "Learning a code: Machine learning for approximate non-linear coded computation". *preprint arXiv:1806.01259*.

[87] Krizhevsky, A. and G. Hinton (2009). "Learning multiple layers of features from tiny images". *Tech. rep.* Citeseer.

[88] Kushilevitz, E. and N. Nisan (2006). *Communication Complexity*. Cambridge University Press.

[89] Lee, K., M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran (2018). "Speeding up distributed machine learning using codes". *IEEE Transactions on Information Theory*. 64(3): 1514–1529.

[90] Lee, K., C. Suh, and K. Ramchandran (2017). "High-dimensional coded matrix multiplication". In: *2017 IEEE International Symposium on Information Theory (ISIT)*. 2418–2422.

[91] Lee, K., R. Pedarsani, and K. Ramchandran (2015). "On scheduling redundant requests with cancellation overheads". In: *53rd Annual Allerton Conference on Communication, Control, and Computing*. IEEE. 99–106.

[92] Li, S., M. A. Maddah-Ali, and A. S. Avestimehr (2018a). "Compressed coded distributed computing". In: *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2032–2036.

[93] Li, S., M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr (2018b). "A fundamental tradeoff between computation and communication in distributed computing". *IEEE Transactions on Information Theory*. 64(1): 109–128.

[94] Li, S., S. M. M. Kalan, A. S. Avestimehr, and M. Soltanolkotabi (2018c). "Near-optimal straggler mitigation for distributed gradient methods". *IPDPSW*. May.

[95] Li, S., S. M. M. Kalan, Q. Yu, M. Soltanolkotabi, and A. S. Avestimehr (2018d). "Polynomially coded regression: Optimal straggler mitigation via data encoding". *e-print arXiv:1805.09934*.

[96] Li, S., M. A. Maddah-Ali, and A. S. Avestimehr (2016a). "A unified coding framework for distributed computing with straggling servers". *IEEE NetCod*. Dec.

[97] Li, S., M. A. Maddah-Ali, and A. S. Avestimehr (2015). "Coded MapReduce". *53rd Annual Allerton Conference on Communication, Control, and Computing*. Sept.

[98] Li, S., M. A. Maddah-Ali, and A. S. Avestimehr (2016b). "Coded distributed computing: Straggling servers and multistage dataflows". *54th Allerton Conference*. Sept.

[99] Li, S., M. Yu, C.-S. Yang, A. S. Avestimehr, S. Kannan, and P. Viswanath (2018e). "PolyShard: Coded sharding achieves linearly scaling efficiency and security simultaneously". *arXiv: 1809.10361 [cs.CR]*.

[100] Liberty, E. and S. W. Zucker (2009). "The mailman algorithm: A note on matrix–vector multiplication". *Information Processing Letters*. 109(3): 179–182.

[101] Lin, S. and D. J. Costello (2004). *Error Control Coding*. Pearson.

[102] Lindell, Y. (2005). "Secure multiparty computation for privacy preserving data mining". In: *Encyclopedia of Data Warehousing and Mining*. IGI Global. 1005–1009.

[103] Low, Y., D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein (2012). "Distributed GraphLab: A framework for machine learning and data mining in the cloud". *Proceedings of the VLDB Endowment*. 5(8): 716–727.

[104] Maddah-Ali, M. A. and U. Niesen (2014a). "Decentralized coded caching attains order-optimal memory-rate tradeoff". *IEEE/ACM Transactions on Networking*. Apr.

[105] Maddah-Ali, M. A. and U. Niesen (2014b). "Fundamental limits of caching". *IEEE Transactions on Information Theory*. 60(5): 2856–2867.

[106] Maity, R. K., A. S. Rawat, and A. Mazumdar (2018). "Robust gradient descent via moment encoding with LDPC codes". *SysML Conference*.

[107] Maleki, H., V. R. Cadambe, and S. A. Jafar (2014). "Index coding—An interference alignment perspective". *IEEE Transactions on Information Theory*. 60(9): 5402–5432.

[108] Malewicz, G., M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski (2010). "Pregel: A system for large-scale graph processing". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM. 135–146.

[109] Mallick, A., M. Chaudhari, U. Sheth, G. Palanikumar, and G. Joshi (2019). "Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication". *Proceedings of the ACM on Measurement and Analysis of Computing Systems*. 3(3): 1–40.

[110] McEliece, R. J. and D. V. Sarwate (1981). "On sharing secrets and Reed–Solomon codes". *Communications of the ACM*. 24(9): 583–584.

[111] McMahan, H. B., E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas (2017). "Communication-efficient learning of deep networks from decentralized data". In: *International Conference on Artificial Intelligence and Statistics*. 1273–1282.

[112] Melis, L., C. Song, E. D. Cristofaro, and V. Shmatikov (2019). "Exploiting unintended feature leakage in collaborative learning". *arXiv:1805.04049*.

[113] Mitra, D. and L. Dolecek (2019). "Patterned erasure correcting codes for low storage-overhead blockchain systems". In: *2019 53rd Asilomar Conference on Signals, Systems, and Computers*. IEEE. 1734–1738.

[114] Mohassel, P. and Y. Zhang (2017). "SecureML: A system for scalable privacy-preserving machine learning". In: *2017 IEEE Symposium on Security and Privacy (SP)*. 19–38.

[115] Narra, K. G., Z. Lin, M. Kiamari, S. Avestimehr, and M. Annavaram (2019). "Slack squeeze coded computing for adaptive straggler mitigation". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC '19*. Denver, Colorado: Association for Computing Machinery.

[116] Nazer, B. and M. Gastpar (2007). "Computation over multiple-access channels". *IEEE Transactions on Information Theory*. 53(10): 3498–3516.

[117] Nodehi, H. A. and M. A. Maddah-Ali (2018). "Limited-sharing multi-party computation for massive matrix operations". In: *IEEE International Symposium on Information Theory (ISIT)*. 1231–1235.

[118] OMalley, O. (2008). "TeraByte sort on apache hadoop". *Tech. rep.* Yahoo.

[119] "Open MPI: Open source high performance computing" (n.d.). https://www.open-mpi.org/.

[120] Orlitsky, A. and A. El Gamal (1990). "Average and randomized communication complexity". *IEEE Transactions on Information Theory*. 36(1): 3–16.

[121] Orlitsky, A. and J. Roche (2001). "Coding for computing". *IEEE Transactions on Information Theory*. 47(3): 903–917.

[122] Pawar, S., S. El Rouayheb, and K. Ramchandran (2011). "Securing dynamic distributed storage systems against eavesdropping and adversarial attacks". *IEEE Transactions on Information Theory*. 57(10): 6734–6753.

[123] Poulson, J., B. Marker, R. A. van de Geijn, J. R. Hammond, and N. A. Romero (2013). "Elemental: A new framework for distributed memory dense matrix computations". *ACM Transactions on Mathematical Software*. 39(2): 13:1–13:24.

[124] Prakash, S., A. Reisizadeh, R. Pedarsani, and S. Avestimehr (2018). "Coded computing for distributed graph analytics". *IEEE ISIT*.

[125] Rajaraman, A. and J. D. Ullman (2011). *Mining of Massive Datasets*. Cambridge University Press.

[126] Ramamoorthy, A. and M. Langberg (2013). "Communicating the sum of sources over a network". *IEEE Journal on Selected Areas in Communications*. 31(4): 655–665.

[127] Raviv, N., I. Tamo, R. Tandon, and A. G. Dimakis (2017). "Gradient coding from cyclic MDS codes and expander graphs". *e-print arXiv:1707.03858*.

[128] Rawat, A. S., O. O. Koyluoglu, N. Silberstein, and S. Vishwanath (2014). "Optimal locally repairable and secure codes for distributed storage systems". *IEEE Transactions on Information Theory*. 60(1): 212–236.

[129] Recht, B., C. Re, S. Wright, and F. Niu (2011). "Hogwild: A lock-free approach to parallelizing stochastic gradient descent". *Advances in Neural Information Processing Systems (NIPS)*: 693–701.

[130] Reisizadeh, A., S. Prakash, R. Pedarsani, and A. S. Avestimehr (2019). "Coded computation over heterogeneous clusters". *IEEE Transactions on Information Theory*. 65(7): 4227–4242.

[131] Reisizadeh, A., S. Prakash, R. Pedarsani, and S. Avestimehr (2017). "Coded computation over heterogeneous clusters". *IEEE ISIT*: 2408–2412.

[132] Renteln, P. (2013). *Manifolds, Tensors, and Forms: An Introduction for Mathematicians and Physicists*. Cambridge University Press.

[133] Roth, R. (2006). *Introduction to Coding Theory*. Cambridge University Press.

[134] Sahraei, S. and A. S. Avestimehr (2019). "INTERPOL: Information theoretically verifiable polynomial evaluation". In: *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 1112–1116.

[135] Sahraei, S., M. A. Maddah-Ali, and S. Avestimehr (2019). "Interactive verifiable polynomial evaluation". *preprint arXiv:1907.04302*.

[136] Sanghavi, S. (2007). "Intermediate performance of rateless codes". In: *Information Theory Workshop, 2007. ITW'07*. IEEE. 478–482.

[137] Schölkopf, B., R. Herbrich, and A. J. Smola (2001). "A generalized representer theorem". In: *International Conference on Computational Learning Theory*. Springer. 416–426.

[138] Seide, F., H. Fu, J. Droppo, G. Li, and D. Yu (2014). "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns". In: *Fifteenth Annual Conference of the International Speech Communication Association*.

[139] Shah, N. B., K. Lee, and K. Ramchandran (2016). "When do redundant requests reduce latency?" *IEEE Transactions on Communications*. 64(2): 715–722.

[140] Shah, N. B., K. Rashmi, and P. V. Kumar (2011). "Information-theoretically secure regenerating codes for distributed storage". In: *Global Telecommunications Conference (GLOBECOM 2011), 2011*. IEEE. 1–5.

[141] Shamir, A. (1979). "How to share a secret". *Communications of the ACM*. 22(11): 612–613.

[142] Shanmugam, K., A. G. Dimakis, and M. Langberg (2013). "Local graph coloring and index coding". In: *2013 IEEE International Symposium on Information Theory*. 1152–1156.

[143] Singleton, R. (1964). "Maximum distance q-nary codes". *IEEE Transactions on Information Theory*. 10(2): 116–118.

[144] So, J., B. Guler, and A. S. Avestimehr (2020). "Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning". *arXiv: 2002.04156 [cs.LG]*.

[145] So, J., B. Guler, A. S. Avestimehr, and P. Mohassel (2019). "CodedPrivateML: A fast and privacy-preserving framework for distributed machine learning". *CoRR*. abs/1902.00641. *arXiv: 1902.00641*.

[146] Song, L., C. Fragouli, and T. Zhao (2017). "A pliable index coding approach to data shuffling". *e-print arXiv:1701.05540*.

[147] Tandon, R., Q. Lei, A. G. Dimakis, and N. Karampatziakis (2017). "Gradient coding: Avoiding stragglers in distributed learning". In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. *Proceedings of Machine Learning Research*. International Convention Centre, Sydney, Australia: PMLR. 3368–3376.

[148] Tang, L., K. Konstantinidis, and A. Ramamoorthy (2019). "Erasure Coding for distributed matrix multiplication for matrices with bounded entries". *IEEE Communications Letters*. 23(1): 8–11.

[149] "tc – show/manipulate traffic control settings" (n.d.). http://lartc.org/manpages/tc.txt.

[150] Ullman, J. D., A. V. Aho, and J. E. Hopcroft (1974). *The Design and Analysis of Computer Algorithms*. Vol. 4. Addison-Wesley, Reading. 1–2.

[151] Van De Geijn, R. A. and J. Watts (1997). "SUMMA: Scalable universal matrix multiplication algorithm". *Concurrency-Practice and Experience*. 9(4): 255–274.

[152] Wan, K., D. Tuninetti, M. Ji, and P. Piantanida (2018). "Fundamental limits of distributed data shuffling". *e-print arXiv:1807.00056*.

[153] Wang, D., G. Joshi, and G. Wornell (2014). "Efficient task replication for fast response times in parallel computation". *ACM SIGMETRICS Performance Evaluation Review*. 42(1): 599–600.

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

[155] Wang, S., J. Liu, and N. Shroff (2018a). "Coded sparse matrix multiplication". *e-print arXiv:1802.03430*.

[156] Wang, S., J. Liu, and N. Shroff (2019b). "Fundamental limits of approximate gradient coding". *Proceedings of the ACM on Measurement and Analysis of Computing Systems*. 3(3): 52.

[157] Wang, S., J. Liu, N. Shroff, and P. Yang (2018b). "Fundamental limits of coded linear transform". *e-print arXiv:1804.09791*.

[158] Wen, W., C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li (2017). "TernGrad: Ternary gradients to reduce communication in distributed deep learning". *Advances in Neural Information Processing Systems (NIPS)*: 1508–1518.

[159] Woolsey, N., R.-R. Chen, and M. Ji (2018). "A new combinatorial design of coded distributed computing". *e-print arXiv:1802.03870*.

[160] Yang, H. and J. Lee (2019). "Secure distributed computing with straggling servers using polynomial codes". *IEEE Transactions on Information Forensics and Security*. 14(1): 141–150.

[161] Yang, Y., M. Interlandi, P. Grover, S. Kar, S. Amizadeh, and M. Weimer (2019). "Coded elastic computing". In: *2019 IEEE International Symposium on Information Theory (ISIT)*. 2654–2658.

[162] Yang, Y., P. Grover, and S. Kar (2017). "Coded distributed computing for inverse problems". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc. 709–719. URL: http://papers.nips.cc/paper/6673-coded-distributed-computing-for-inverse-problems.pdf.

[163] Yao, A. C.-C. (1979). "Some complexity questions related to distributive computing (preliminary report)". In: *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*. 209–213.

[164] Ye, M. and E. Abbe (2018). "Communication-computation efficient gradient coding". *e-print arXiv:1802.03475*.

[165] Yu, M., S. Sahraei, S. Li, S. Avestimehr, S. Kannan, and P. Viswanath (2020). "Coded Merkle tree: Solving data availability attacks in blockchains". In: *Financial Cryptography and Data Security (FC)*.

[166] Yu, Q., S. Li, M. A. Maddah-Ali, and A. S. Avestimehr (2017a). "How to optimally allocate resources for coded distributed computing?" *IEEE International Conference on Communications (ICC)*. May: 1–7.

[167] Yu, Q., M. A. Maddah-Ali, and A. S. Avestimehr (2017b). "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication". *Advances in Neural Information Processing Systems (NIPS)*: 4406–4416.

[168] Yu, Q., M. A. Maddah-Ali, and A. S. Avestimehr (2018a). "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding". *e-print arXiv:1801.07487*.

[169] Yu, Q., M. A. Maddah-Ali, and A. S. Avestimehr (2018b). "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding". In: *IEEE International Symposium on Information Theory (ISIT)*. 2022–2026.

[170] Yu, Q., N. Raviv, J. So, and A. S. Avestimehr (2018c). "Lagrange coded computing: Optimal design for resiliency, security and privacy". *e-print arXiv:1806.00939*.

[171] Zaharia, M., M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica (2010). "Spark: Cluster computing with working sets". In: *Proceedings of the 2nd USENIX HotCloud*. Vol. 10. 10.

[172] Zaharia, M., A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica (2008). "Improving MapReduce performance in heterogeneous environments". *Operating Systems Design and Implementation*. 8(4): 7.

[173] Zhang, S., J. Han, Z. Liu, K. Wang, and S. Feng (2009). "Accelerating MapReduce with distributed memory cache". *15th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*. Dec.: 472–478.

[174] Zhang, Z., L. Cherkasova, and B. T. Loo (2013). "Performance modeling of MapReduce jobs in heterogeneous cloud environments". In: *IEEE Sixth International Conference on Cloud Computing*. 839–846.

[175] Zhuang, Y., W.-S. Chin, Y.-C. Juan, and C.-J. Lin (2013). "A fast parallel SGD for matrix factorization in shared memory systems". In: *Proceedings of the 7th ACM Conference on Recommender Systems*. ACM. 249–256.