Multi-User Linearly-Decomposable Distributed Computing

Ali Khalesi

Multi-User Linearly-Decomposable Distributed Computing

Dissertation

submitted to

Sorbonne Université

in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Author:

Ali Khalesi

Jury President/Reviewer	**Prof. Michèle Wigger**	Télécom Paris, FR
Examiner/Reviewer	**Prof. Sundar Rajan**	Indian Institute of Science, IN
Examiner	**Prof. Giuseppe Caire**	Technical University of Berlin, DE
Examiner	**Prof. Derya Malak**	EURECOM, FR
Examiner	**Prof. Daniela Tuninetti**	University of Illinois Chicago, US
Examiner	**Prof. Konstantin Avrachenkov**	INRIA Sophia Antipolis, FR
Thesis Advisor	**Prof. Petros Elia**	EURECOM, FR

The research has been conducted in the Communication Systems Departement at EURECOM (Sophia Antipolis, FR) from January 2020 to June 2024.

Manuscript compiled with pdfLaTeX on September 2, 2024.

Dedication

To the ones who were honest, truthful and loyal to me.
To my friends and my family.

It is the part of an uneducated person to blame others where he himself fares ill; to blame himself is the part of one whose education has begun; to blame neither another nor his own self is the part of one whose education is already complete. Only the educated are free.

Epictetus

Abstract

Distributed computing frameworks are an inevitable part of current telecommunication technologies. They enable service providers to process computation-intensive tasks and to communicate huge amounts of information between servers or end-users via shared networking and storage resources. Many of the prevalent applications can be found in wireless sensor networks, online multi-player games, virtual reality, distributed database management systems, distributed learning algorithms for training large models (such as large language models and federate learning) and parallel computing. This thesis explores multi-user linearly-decomposable distributed computation, where N servers help compute the desired functions (jobs) of K users, and where each desired function can be written as a linear combination of up to L (generally non-linear) subtasks (or subfunctions). Each server computes some of the subtasks, communicates a function of its computed outputs to some of the users, and then each user collects its received data to recover its desired function. Our first study explores the computation and communication costs relationship. For a coefficient matrix $\mathbf{F}$ representing the linearly decomposable form of the set of requested functions, our problem becomes equivalent to the open problem of sparse matrix factorization $\mathbf{F} = \mathbf{DE}$ over any mathematical field, where a sparse communication matrix $\mathbf{D}$ and computing matrix $\mathbf{E}$ imply reduced communication and computation costs respectively. In this thesis, we present our research by modelling the processed databases by the servers and transmitted signals as elements in some arbitrary finite field or the real numbers domain. In the finite field approach, we established a novel relationship between our distributed computing problem, matrix factorization, syndrome decoding and covering codes. To reduce the computation cost per subfunction — Which is the maximum number of servers that a subfunction is assigned to it, across all the subfunctions — the above $\mathbf{D}$ is drawn from covering codes or from a here-introduced class of so-called 'partial covering' codes, whose study here yields reduced computation cost per subfunction. To then reduce the cumulative communication cost — The overall transmitted signals by each server — these coding-theoretic properties are explored in the regime

of codes that have low-density parity check matrices. The thesis reveals — first for the commonly used one-shot scenario — that in the limit of large N, the optimal normalized computation cost per subfunciton $\gamma_f \in (0, 1)$ is in the range $\gamma_f \in (H_q^{-1}(\frac{\log_q(L)}{N}), H_q^{-1}(K/N))$ — where H_q is the q-ary entropy function — and that this can be achieved with normalized communication cost that vanishes as $\sqrt{\log_q(N)}/N$. The above reveals an unbounded coding gain over the uncoded scenario, as well as reveals the role of a certain *functional rate* $\log_q(L)/N$ and *functional capacity* $H_q(\gamma)$ of the system. We also explore the multi-shot scenario, for which we derive bounds on the computation cost per subfunction.

In another effort we aim at reducing the total number of subfunction computations across the servers (cumulative computational cost), as well as the worst-case load which can be a measure of computational delay. Our contribution consists of novel bounds on the two computing costs, where these bounds are linked to the covering and packing radius of classical codes.

One of our findings is that in certain cases, our distributed computing problem is treated optimally when $\mathbf{F}$ is decomposed into a parity check matrix $\mathbf{D}$ of a perfect code, and a matrix $\mathbf{E}$ which has columns as the coset leaders of this same code.

In the real numbers domain, we reformulate the real-valued distributed computing problem into a matrix factorization problem and then into a basic sparse recovery problem, where sparsity implies computational savings. Building on this, we first give a simple probabilistic scheme for subfunction assignment, which allows us to upper bound the optimal normalized cumulative computation cost — the overall number of subfunction computations that are done by all of the servers — as $\gamma_c \leq \frac{K}{N}$ that a generally intractable ℓ_0-minimization would give. To bypass the intractability of such optimal scheme, we show that if these optimal schemes enjoy $\gamma_c \leq -r\frac{K}{N}W_{-1}^{-1}(-\frac{2K}{eNr})$ (where $W_{-1}(\cdot)$ is the Lambert function and r calibrates the communication between servers and users), then they can be derived using a tractable Basis Pursuit ℓ_1-minimization. This newly revealed connection opens up the possibility of designing practical distributed computing algorithms by employing tools and methods from compressed sensing.

This thesis also introduces a new framework called tessellated distributed computing, where the aim is for each user to receive their function outputs, allowing for reduced error ϵ, reduced computing cost (γ; the fraction of subfunctions each server must compute), and reduced communication cost (δ; the fraction of users each server must connect to). For any given set of K requested functions — which again is here represented by a coefficient matrix

$\mathbf{F} \in \mathbb{R}^{K \times L}$ — our problem is made equivalent to the open problem of sparse matrix factorization that seeks — for a given parameter T, representing the number of shots for each server — to minimize $\frac{1}{KL}\|\mathbf{F} - \mathbf{DE}\|_F^2$ overall δ-spars and γ-sparse matrices $\mathbf{E} \in \mathbb{R}^{NT \times L}$ and $\mathbf{D} \in \mathbb{R}^{K \times NT}$. With these matrices respectively defining which servers compute each subfunction, and which users connect to each server, we here design our $\mathbf{E}, \mathbf{D}$ by designing tessellated-based and SVD-based fixed support matrix factorization methods that first split $\mathbf{F}$ into properly sized and carefully positioned submatrices, which we approximate and then decompose into properly designed submatrices of $\mathbf{D}$ and $\mathbf{E}$. For the zero-error case and under basic dimensionality assumptions, the thesis reveals achievable computation-vs-communication corner points (γ, δ) which, for various cases, are proven optimal over a very large class of $\mathbf{D}, \mathbf{E}$ using a novel tessellations-based converse. Subsequently, for large N, and under basic statistical assumptions on $\mathbf{F}$, the average achievable error ϵ is concisely expressed using the incomplete first moment of the standard Marchenko-Pastur distribution, where this performance is shown to be optimal over a large class of $\mathbf{D}$ and $\mathbf{E}$. In the end, the work also reveals that the overall achieved gains over baseline methods are unbounded. In summary, this thesis explores the fundamental limits of multi-user linearly-decomposable distributed computing in various domains and various conceptions of communication and computation costs.

One notable aspect is that we treat an extremely broad setting of functions and do so in a promising multi-user setting. Interestingly, we here present never-before-seen connections between distributed computing, coding theory, perfect codes, compressive sensing and tessellation theory, as well as large matrix analysis. As we discuss, there is a rich class of problems that emerges from these connection. We hope that these new directions and connections are useful in the development of the very challenging area of multi-user distributed computing.

Acknowledgements

At this pinnacle of my academic journey, I find it imperative to express my deepest gratitude to all those whose support, guidance, and encouragement have been instrumental in the completion of this thesis. Foremost among them, I am profoundly grateful to my supervisor, Professor Petros Elia, whose invaluable insight, constructive feedback, and unwavering encouragement have shaped this work from its inception to its conclusion. Without his expertise and dedication, the realization of this thesis would not have been possible.

I am also indebted to the jury members for their insightful critiques, precise questions, and invaluable feedback. In particular, I extend my heartfelt thanks to the two reviewers, Pr. Michèle Wigger and Pr. Sundar Rajan, for their time, dedication, and rigorous evaluation of this thesis. I would also like to express my sincere appreciation to Pr. Giuseppe Caire, Pr. Derya Malak, Pr. Daniela Tuninetti, and Pr. Konstantin Avrachenkov, who served as evaluators and jury members. I am especially thankful to Pr. Caire for his encouraging perspective on the thesis and his profound comments on the compressed sensing component.

At this significant juncture, I must also acknowledge those individuals who have profoundly shaped both my character and my academic journey from the very beginning. I owe a deep debt of gratitude to my middle school physics teacher, Dr. Ebadollah Naderi, whose early encouragement to pursue a path in research unveiled my potential in mathematics and physics. His guidance was foundational in shaping my future endeavors, and I sincerely hope he remains in good health and high spirits, wherever he may be.

My undergraduate years were further enriched under the supervision of Dr. Bahare Akhbari and through the teachings of Pr. Mahmoud Ahmadian Attari in channel coding course. Their profound understanding of information theory and coding theory, coupled with their masterful exposition of these subjects, galvanized my decision to specialize in this field. It was their recommendation that led me to pursue my Master's degree under the mentorship of two distinguished professors at Sharif University. In particular, I had the privilege of working with Pr. Mahtab Mirmohseni and Pr. Mohammad Ali

Maddah-Ali, two of the most astute and knowledgeable academics I have encountered. I am deeply grateful to them for laying the groundwork of my research methodologies and strategies and for their rigorous expectations, which ultimately prepared me for further academic pursuits. This foundational experience paved the way for my acceptance at Sorbonne University, one of the world's most venerable and esteemed institutions, where I was fortunate to continue my studies under the mentorship of Pr. Elia, a profound lover of mathematics and an exemplary teacher.

I am particularly grateful to my friend and colleague, Sajad Daei, and to Pr. Marios Kountouris for their instrumental role in introducing me to the concept of compressed sensing. Additionally, I extend my sincere thanks to my colleagues at EURECOM, Ahmad Tanha and Reza Deylam Salehi, for their unwavering support throughout my doctoral journey. Special acknowledgement is due to my best friend, Mohammad Saeed Masiha, a true connoisseur of mathematics and a brilliant thinker, for the countless intellectually stimulating discussions that greatly enriched my PhD experience.

I am also deeply appreciative of my long-standing friends in my home country—Amirhossein Afshar, Mohammad Zolfaghari, Mostafa Parsapour, and Alireza Dehghani—whose friendship has spanned over 15 years. Their virtual presence during my PhD defense and their shared joy in my success have meant more to me than words can convey.

Moreover, I would like to extend my heartfelt gratitude to my colleagues and staff at the EURECOM Institute, particularly the former and current directors, Ulrich Finger and David Gesbert, for their relentless commitment to maintaining EURECOM as a premier institution for both fundamental and applied research. I deeply appreciate their efforts to ensure the well-being of all staff during the COVID-19 pandemic and beyond, as well as their provision of an environment conducive to cutting-edge research.

Lastly, I am profoundly grateful to my family for their unwavering support, patience, and love throughout this academic endeavor. Their belief in me has been my greatest source of motivation and strength.

Nice, September 02, 2024 Ali Khalesi

Contents

List of Figures

Notations

$\mathbb{N}$	Natural numbers
$\mathbb{R}$	Real numbers
$\mathbf{GF}(q)$	Finite Field of size q where q is a power of a prime number
$\mathbb{F}$	An arbitrary Finite Field
$\mathbf{A}$	Matrix
$[\mathbf{A}, \mathbf{B}]$	Indicates the horizontal concatenation of two matrices $\mathbf{A}, \mathbf{B}$
$\mathbf{A}(i, j)$	The element of matrix $\mathbf{A}$ in i-th Row and j-th column
$\mathbf{A}(i, :)$	Row vector representing the i-th row of matrix $\mathbf{A}$
$\mathbf{A}(:, j)$	Column vector representing the j-th column of matrix $\mathbf{A}$
$\mathbf{a}$	Column vector a
$\mathbf{a}^\intercal$	A vertical vector and transpose of a
$\mathcal{I}$	Set I
$\mathbf{A}(\mathcal{I}, \mathcal{J})$	Submatrix of $\mathbf{A}$ comprised of elements where their row indices are in $\mathcal{I}$ and their column indices are in $\mathcal{J}$
$\omega(\mathbf{X})$	The number of nonzero elements of some matrix $\mathbf{X}$
$\omega(\mathbf{x})$	The number of nonzero elements of some vector $\mathbf{x}$
$\mathcal{C} \subseteq \mathbb{F}^n$	Represents a code
$d(\mathbf{x}, \mathcal{C})$	The Hamming distance of $\mathbf{x} \in \mathbb{F}^n$, to the nearest codeword in $\mathcal{C} \subseteq \mathbb{F}^n$
$\rho, \rho(\mathcal{C})$	The normalized covering radii(of code $\mathcal{C} \subseteq \mathbb{F}^n$) in Chapter 2
$\mathcal{C}(k, n)$	Linear code of dimension (the message length) k or with codeword length n
$\mathcal{C}_{\mathbf{H}}$	Linear code whose Parity-Check matrix is $\mathbf{H}$
$\mathbf{H}_{\mathcal{C}}$	Matrix that serves as the parity-check matrix of a specific linear code $\mathcal{C}$
$[\mathcal{C}_1, \mathcal{C}_2]$	The code resulting from direct product of $\mathcal{C}_1$ and $\mathcal{C}_2$
$< \mathbf{x}, \mathcal{C}_1 >$	A code whose basis is the union of $\mathbf{x} \in \mathbb{F}^n$ with the basis of $\mathcal{C}_1$
$V_q(n, \rho)$	The volume of a Hamming ball in $\mathbb{F}^n$ of radius ρn
$H_q(x)$	For $0 \leq x \leq 1 - \frac{1}{q}, x \in \mathbb{R}$, represents the q-ary entropy function which is equal to $x \log_q(q-1) - x \log_q(x) - (1-x) \log_q(1-x)$ $H(x)$.
$H(x)$	is $H_2(x)$.

$\text{supp}(\mathbf{x})$	The support of some vector $\mathbf{x} \in \mathbb{F}^n$ or $\mathbf{x} \in \mathbb{R}^n$, which is the set of indices of non-zero elements.
$\epsilon(n)$	An expression which, in the limit of large n, vanishes to zero
$\|.\|_0$	L_0 norm operator, represents the number of non-zero elements of a factor or matrix defined in the real domain
$\otimes$	Kronecker Product Operator
$\text{vec}(\mathbf{X})$	The vectorization of $\mathbf{X}$ matrix
$\mathbf{A}^*$	Conjugate of $\mathbf{A}$ in the complex domain
$\mathbf{A}_{\mathcal{S}}$	A matrix that consists of only the columns of $\mathbf{A}$ indexed in $\mathcal{S}$
$\mathbf{I}_{m\times m}$	An $m \times m$ identity matrix.
$\mathbb{I}(\mathcal{X}, \mathcal{Y})$	A matrix that all of its elements are zero except the elements, $\mathbb{I}(i,j), i \in \mathcal{X}, j \in \mathcal{Y}$
$\text{Supp}(\mathbf{A})$	A binary matrix from $\{0,1\}^{m\times n}$ representing the locations of non-zero elements of $\mathbf{A}$
$\vee$	Logical "or"
$\wedge$	Logical "and"
$\neg$	Logical "negation"
$\mathbf{I} \cup \mathbf{J}$	Defined for two binary matrices $\mathbf{I}, \mathbf{J} \in \{0,1\}^{m\times n}$, having the property $(\mathbf{I} \cap \mathbf{J})(i,j) = \mathbf{I}(i,j) \vee \mathbf{J}(i,j)$
$\mathbf{I} \cap \mathbf{J}$	Defined for two binary matrices $\mathbf{I}, \mathbf{J} \in \{0,1\}^{m\times n}$, having the property $(\mathbf{I} \cap \mathbf{J})(i,j) = \mathbf{I}(i,j) \wedge \mathbf{J}(i,j)$
$\mathbf{I}'$	Defined for a binary matrix $\mathbf{I} \in \{0,1\}^{m\times n}$, having the property $\mathbf{I}'(i,j) = \neg\mathbf{I}(i,j)$
$\mathbf{I} \backslash \mathbf{J}$	Defined for two binary matrices $\mathbf{I}, \mathbf{J} \in \{0,1\}^{m\times n}$ and is equal to $\mathbf{I} \cap \neg\mathbf{J}$
$\mathbf{1}_n$	An all-one n dimensional column vector in real numbers
$\mathbf{0}_m$	An all-zero m dimensional column vector
$\odot$	Hadamard product operation
$\|\mathbf{A}\|_F$	The Frobenius norm of the matrix $\mathbf{A}$ which is equal to $\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\mathbf{A}^2(i,j)}$.
$\lceil x \rceil, \lfloor x \rfloor$	The ceiling and floor function of a real number x
$\text{mod}(a,b)$	For two numbers $a, b \in \mathbb{N}$ is the remainder of division of a by b.
$\Phi_{\text{MP},\lambda}(t,r)$	The *incomplete first moment* of the Marchenko–Pastur distribution with ration λ which can also be described as $\int_r^t x f_{\text{MP},\lambda}(x)dx$, where $f_{\text{MP},\lambda}(x)$ is the probability density function of a Marchenko-Pasture distribution
$a \vert b$	For two natural numbers a, b, it means that a divide b
$a \nmid b$	For two natural numbers a, b, it means that a does not divide b

Chapter 1

Introduction

As continuous data streams become more prevalent, the limitations of individual computing nodes in handling large-scale computation tasks are increasingly apparent. In recent years, distributed computing has emerged as a preferred solution due to its multitude of advantages over centralized computing. Distributed computing involves a collaborative network of computing nodes working together as a unified system to tackle computation tasks, utilizing shared networking and storage resources [1].

Primarily, distributed computing offers enhanced reliability and fault tolerance, ensuring seamless operation even in the face of node failures. Additionally, it boasts accelerated computation speed by distributing the workload across multiple nodes. Furthermore, it exhibits inherent scalability, facilitating the effortless addition of computing nodes as required. Moreover, distributed computing proves cost-effective, leveraging economical hardware for computing nodes. This approach finds widespread adoption in cloud computing and other emerging services.

Given these advantages, distributed computing finds diverse applications in various real-world scenarios. These include telecommunication networks (such as telephone networks and wireless sensor networks), network applications (such as World Wide Web networks, massively multiplayer online games, virtual reality communities, distributed database management systems, and network file systems), real-time process control (including aircraft control systems), and parallel computation (such as cluster computing, grid computing, and computer graphics)[2]–[4].

Let's examine one of the most prevalent distributed computation frameworks, known as MapReduce [5]. MapReduce serves as a software framework and programming model designed to process computation tasks across extensive datasets utilizing a multitude of computing nodes, also known as workers. These computing nodes are often grouped into clusters. Typically,

the computation task is divided into three distinct phases: the "Map" phase, "Shuffle" phase, and "Reduce" phase. During the Map phase, a master node partitions the computation task into numerous subtasks and assigns them to the computing nodes. These nodes then execute the subtasks based on allocated Map functions, yielding intermediate results. Subsequently, the intermediate results undergo exchange among the computing nodes, a process referred to as "data shuffling," occurring in the Shuffle phase. Finally, in the Reduce phase, the computing nodes utilize these results to compute the outcome in a distributed manner, employing their designated Reduce functions.

In this distributed computing framework, we face two primary challenges. Firstly, computing nodes must exchange numerous intermediate results over the network to calculate the final result. This leads to significantly increased communication overheads. This problem also degrades the performance of distributed computing applications such as Self-Join, Terasort [6], and Orchestra [7] and other Distributed Machine Learning frameworks [8]. For instance, in the Hadoop cluster [9] at Facebook, the data shuffling phase typically consumes 33% of the overall job execution time. Similarly, when running TeraSort and Self-Join applications on a heterogeneous Amazon EC2 cluster, approximately 65% and 70% of the overall job execution time, respectively, is spent on the Shuffle phase [10]. The communication bottleneck is particularly pronounced in training convolutional neural networks (CNNs), such as Resnet-50 [11] and AlexNet [12], which involve updating millions of model parameters. Secondly, distributed computing involves a large number of computing nodes with varying computing and networking resources, resulting in straggler nodes that unintentionally run slower than others. These stragglers increase the overall time required to complete computing tasks. Traditional approaches like work exchange and naive replication have been used to mitigate straggler effects, but they either introduce redundancy or require coordination among nodes, increasing communication costs and computational load. This underscores the need for novel techniques to effectively address straggler effects and communication load in distributed computing.

Coding theoretic techniques, such as low-density parity-check (LDPC) coding, have been extensively used in WiFi and cellular systems to counteract channel noise and impairments. They have also found applications in distributed storage systems and cache networks to reduce storage costs and network traffic [13]–[15]. These techniques introduce redundancy in messages or signals before transmission, allowing receivers to correct errors caused by channel noise. Recently, coding theoretic techniques have been recognized as promising solutions for overcoming challenges in distributed computing. For instance, they can encode the Map tasks of computing nodes to enable the

master to recover the final result from partially finished nodes, mitigating straggler effects [16], [17]. Additionally, coding theoretic techniques facilitate coding opportunities across intermediate results of distributed computation tasks, reducing communication load by minimizing the number and size of data transmissions among computing nodes [18]. The amalgamation of coding techniques and distributed computing is termed coded distributed computing (CDC). Beyond reducing communication load and mitigating straggler effects, the CDC offers fault tolerance, privacy preservation, and improved security in distributed computing, garnering significant attention. For a detailed survey, we refer the reader to [19] and [20].

Ever since the groundbreaking studies on employing coding techniques in distributed computing [18], [21], numerous schemes for coded distributed computing have been introduced to tackle various tasks in machine learning applications. In this thesis, we begin our analysis by focusing on one of the basic and abstract models of distributed computing, introduced in [22]. This framework, although simplified, generalized many distributed computing frameworks such as:

- The distributed gradient coding problem examined in [23], [24], [25].
- The distributed linear transform problem analyzed in [26].
- In the scenarios of distributed matrix-vector multiplication explored in [27]–[29], distributed matrix-matrix multiplication discussed in [30]–[36], and distributed multivariate polynomial computation examined in [37], where coded assignments are allowed. This means that each worker can receive linear combinations of all input datasets. In contrast, the problem under consideration in this paper involves uncoded data assignments, meaning each worker is only capable of computing functions based on the datasets assigned to it.

In Section1.1, we elaborate more on the single user distributed linearly-separable introduced in [22] and discuss their method of analysis and its advantages and disadvantages concerning the practical concerns. Then, in Section 1.2, we present the multi-user linearly-decomposable (separable) distributed computing problem, inspired by [22], and finally, we summarize the contributions of this thesis in Section 1.2. In Section 1.2, each contribution will be accompanied by a reference to the corresponding chapter and section, where detailed formulations and proofs are presented.

1.1 Single-User Distributed Linearly-Separable Computation

As mentioned earlier, distributed computation systems partition computational tasks into multiple subtasks, which are then allocated to distributed workers. This approach significantly reduces computing time by leveraging parallel computing methods, enabling the processing of large-scale data. However, while large-scale distributed computing holds promise for achieving remarkable accuracy and insights into complex phenomena, it also presents technical challenges. Firstly, the presence of stragglers, where certain workers experience extended processing times or fail to return completed subtasks, introduces undesirable and unpredictable latency. Secondly, the transfer of data and computed results between the user orchestrating the task and the workers poses another bottleneck, especially when communication bandwidth is limited. To address these challenges, coding techniques have been integrated into distributed computing algorithms. These techniques aim to enhance tolerance to stragglers and reduce communication costs between the user and workers.

In [22], a user aims to compute a linearly separable function f (such as linear Mapreduce, Fourier transform, convolution, etc.) on L datasets $(D_1, \ldots, D_L)$, which can be written as

$$f(D_1, \ldots, D_L) = g\Big(f_1(D_1), \ldots, f_L(D_L)\Big) = g(W_1, \ldots, W_L).$$

$W_l = f_l(D_l)$ for all $l \in \{1, \ldots, L\}$ is the outcome of the component function $f_l(\cdot)$ applied to dataset D_l, and it is represented as a string of M symbols on an appropriate sufficiently large alphabet. For example, W_l can be the intermediate value in linear MapReduce, an input signal in Fourier Transform, etc. In fact, $g(\cdot)$ is a linear map defined by K_c linear combinations of the messages $W_1, \ldots, W_L$ with uniform i.i.d. coefficients over some large enough finite field; i.e., $g(W_1, \ldots, W_L)$ can be seen as the matrix product $\mathbf{FW}$, where $\mathbf{F}$ is the coefficient matrix and $\mathbf{W} = [W_1; \ldots; W_L]$. In particular, this model is chosen to capture the following applications

- As matrix multiplication is one of the key building blocks underlying many data analytics, machine learning algorithms and engineering problems, the considered model also has potential applications in those areas, where $f_1, \ldots, f_L$ represent the pretreatment of the datasets.

- Each dataset D_l where $l \in \{1, \ldots, L\}$ represents a raw dataset and needs to be processed through some filters, where W_l represents the

filtered dataset of D_l. For the sake of linear transforms (e.g., wavelet transform, discrete Fourier transform), we need to compute multiple linear combinations of the filtered datasets, which can be expressed as $g(W_1, \ldots, W_L)$.

- For another example, $D_1, \ldots, D_L$ are the L "input channels" of a Convolutional Neural Networks (CNN) stage. Each input channel D_l where $l \in \{1, \ldots, L\}$ is filtered individually by a convolution operation yielding W_l. Then the convolutions are linearly mixed by the coefficients of $g(W_1, \ldots, W_L)$ producing K_c new layers in the feature space.
- if $\mathbf{F}$ represents a MIMO precoding matrix, the considered model can also be used in the MIMO systems.

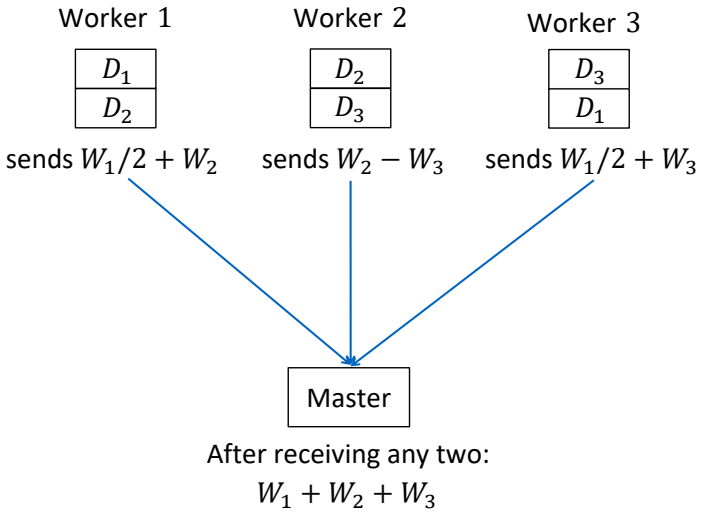
The distributed computation scenario involves computing $f(D_1, \ldots, D_L)$ in a distributed manner by a group of N workers. Each dataset is assigned uncoded to a subset of workers, with the number of datasets assigned to each worker not exceeding Γ, referred to as the computation cost. It's assumed that the complexity of computing the messages from the datasets is much higher than computing the desired linear combinations of the messages. The computation cost is denoted by Γ. Each worker computes and sends coded messages in terms of the datasets assigned to it, such that from the answers of any N_r workers, the user can recover the task function with high probability. Given (L, N, N_r, K_c, D), the master node aims to find the optimal distributed computing scheme with data assignment, computing, and decoding phases, minimizing the communication cost (i.e., the number of downloaded symbols by the user, normalized by D).

Two examples presented in [22] illustrate the formulated distributed scenario in Fig. 1.1 where $K_c = 1$ and $K_c = 2$, respectively. In both examples, $L = N = 3, N_r = 2$, and the number of datasets assigned to each worker is $\Gamma = 2$. The characteristic of number $\mathbf{GF}(q)$ (the field that the alphabet is designed) is assumed to be larger than 3.

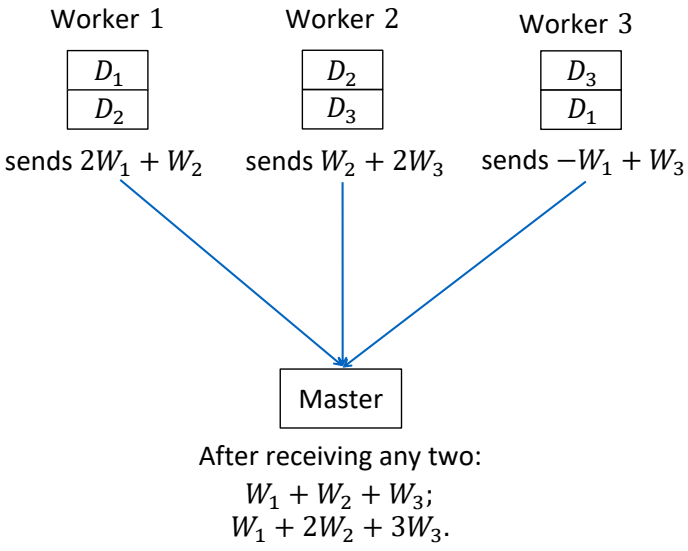
The detailed comparison between the considered distributed linearly separable computation problem and each of the related existing works is provided in Section II-B of [22].

In that paper, the author formulates the problem of distributed computation for linearly separable functions, focusing on scenarios where N divides L and the computation cost is minimized, specifically $\Gamma = \frac{L}{N}(N - N_r + 1)$. The key contributions outlined in that paper are as follows:

- An information-theoretic converse bound on the minimum communication cost is introduced, inspired by the converse bound utilized in the coded caching problem with uncoded cache placement.



(a) $L_c = 1$.



(b) $L_c = 2$.

Figure 1.1: Distributed linearly separable computation with $L = N = 3$ and $N_r = 2$. The number of datasets assigned to each worker is $\Gamma = 2$ (adapted from [22]). In our terminology, master refers to the user and worker nodes are servers.

- Leveraging the cyclic assignment method, a widely adopted approach in existing literature addressing distributed gradient coding problems, such as [23], [38], a novel distributed computing scheme based on linear space intersection is proposed. The decodability of this scheme is established through the Schwartz-Zippel Lemma.

- The paper demonstrates that the achievable scheme is optimal compared to the proposed converse bound under the conditions when $N = L$, or $L \in \left\{1, \ldots, \left\lceil \frac{\frac{L}{N}}{\left(N-N_r+1\right)} \right\rceil \right\}$, or $K_c \in \left\{\frac{L}{N}N_r, \ldots, L\right\}$. Additionally, the paper establishes that the achievable scheme is optimal when considering the constraint of the cyclic assignment across all system parameters.

- An interesting observation stemming from the derived optimality results is highlighted in that paper: when $K = N_r$, for any $K_c \in \{1, \ldots, N_r\}$, the optimal communication cost consistently remains N_r. Consequently, by adopting the same communication cost as the optimal gradient coding scheme proposed in [23] for the distributed gradient coding problem (which aligns with the case $K_c = 1$ in our problem), the proposed scheme enables the master to recover any additional $N_r - 1$ linear combinations with uniformly i.i.d. coefficients over $\mathbf{GF}(q)$ with high probability.

Finally, various extensions of the single-user linearly separable distributed computation problem and its related problems have been investigated in [39]–[42], from different aspects and but similar approaches. In the next section, we elaborate on the multi-user linearly-decomposable distributed computing, which not only extends the single-user aspects of the problem but also generalizes the computation and communication costs and the assignment phase where the master node is not required to assign the datasets or subfunctions cyclically.

1.2 Multi-User Linearly-Decomposable Distributed Computing

As mentioned earlier linearly-decomposable (separable) functions appear in several classes of problems such as in training large-scale machine learning algorithms and deep neural networks with massive data [8], where indeed both computation and communication costs are crucial [43], [44]. To first characterize the communication and computation relationship between multi-user, multi-server computation of linearly-decomposable functions, we modelled a setting that consists of a master node that manages N server nodes that

must contribute in a distributed manner to the computation of the desired functions of K different users. Under the linearly-decomposable assumption (cf. [22]), we consider that user $k \in \{1, 2, \ldots, K\}$ demands a function $F_k(.)$ that can be decomposed as

$$F_k(.) = \sum_{\ell=1}^{L} f_{k,\ell} f_\ell(.) = \sum_{\ell=1}^{L} f_{k,\ell} W_\ell$$

where in the above, $W_\ell = f_\ell(.)$ denotes the computed output of a subfunction and where $f_{k,\ell}$ are the combining coefficients which belong, together with the entries of W_ℓ, in some field. This modeling nicely captures linearly separable functions where each $F_k(.)$, taking L subfunction as input, can be written as a linear combination of L *univariate* subfunctions. In this thesis, these subfunctions need not be univariate. Also, note that the setting nicely includes the case where each F_k itself is a linear combination of some linearly separable functions, i.e., where F_k can itself be written as $F_k(.) = \sum_{\ell=1}^{L} \sum_{i=1}^{M} f_{k,\ell,i} f_{\ell,i}(.) = \sum_{\ell=1}^{L} \sum_{i=1}^{M} f_{k,\ell,i} W_{\ell,i}$, corresponding to some set of basis subfunctions $f_{\ell,i}(.)$. Upon notification of the users' requests — where these requests are jointly described by the $K \times L$ matrix $\mathbf{F}$ that contains the different coefficients $f_{k,\ell}$ — the master instructs the servers to compute some of the subfunctions $f_\ell(.)$. Each server may compute a different number of functions. Upon completing its computations, each server communicates linear combinations of its locally computed outputs (files) to carefully selected subsets of users. Each user can then only linearly combine what it receives from the servers that have been transmitted to it, and the goal is to guarantee that each user can recover its desired function. The problem is completed when every user k retrieves its desired $F_k(.)$.

We note that there is a clear differentiation between the server nodes that are asked to compute hard (generally non-linear) component functions (subfunctions), and the users that can only linearly combine their received outputs. Generating the so-called output files $W_\ell = f_\ell(.), \ell \in \{1, 2, \ldots, L\}$, can be the result of a computationally intensive task that may for example relate to training a deep learning model on a dataset, or it can relate to the distributed gradient coding problem [23]–[25], [38], the distributed linear-transform computation problem [26], [45], or even the distributed matrix multiplication and the distributed multivariate polynomial computation problems [18], [27]–[29], [31]–[33], [35], [36], [46], Mapreduce with linear reduce function [47] and naturally, the problems that are included in the single-user linearly-separable scenario.

$$\epsilon \triangleq \frac{\mathcal{E}}{KL}, \quad \gamma \triangleq \frac{\Gamma}{L}, \quad \delta \triangleq \frac{\Delta}{K}. \tag{1.1}$$

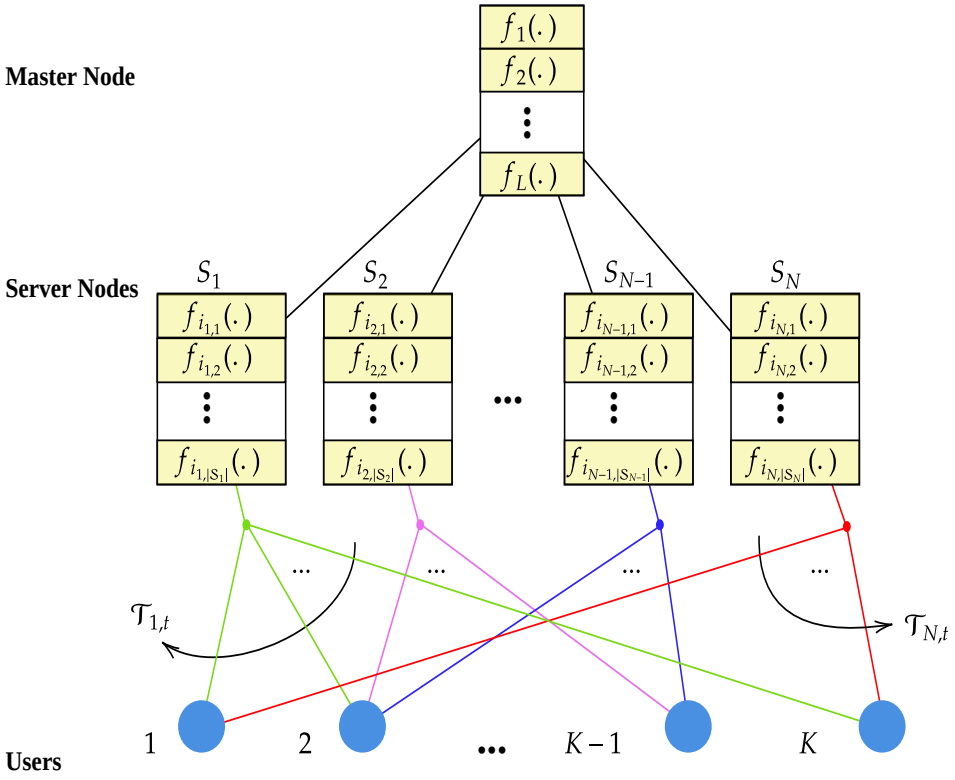


Figure 1.2: The K-user, N-server, T-shot Multi-User Linearly-Decomposable Distributed Computing Setting. Each server n computes the subfunctions in $\mathcal{S}_n = \{f_{i_{n,1}}(.), f_{i_{n,2}}(.), \ldots, f_{i_{n,|\mathcal{S}_n|}}(.)\}$ and communicates to K different users in $\mathcal{T}_{n,t}$.

In this thesis, we analyzed the problem by working on two mathematical "fields". The first is the finite field, as it is much more common in the information theory community to model many networked information systems in a finite field such as what has been done in [17], [22]. But also many systems are investigated in real field domain, such as [48]–[51]. In this thesis, we also investigated this problem in the real domain which will indeed constitute a substantial deviation from the finite field case. The focus on the real (or complex) domain is essential due to the impracticality of computing a real-valued problem over a finite field (after discretization). This is primarily because discretization can result in significant precision costs and accuracy reductions. Moreover, finite field computations are notably slower than floating-point operations [52]. Therefore, we will analyze real-valued functions over L real-valued datasets (or equivalently, with L component/basis subfunctions), alongside N computing servers and K users, each requesting their function.

Then, once we have agreed on the system model, we begin our mathematical abstraction deduction procedures. We will formulate the problem in Chapters 2, 4 and 5, for both single-shot and multi-shot cases. In both finite and real-valued fields, we observe that the problem formulation reduces to a matrix factorization problem when one wants to approximate a demand (jobs) matrix $\mathbf{F} \in \mathbb{F}^{K\times L}$ by the multiplication of two communication $\mathbf{D} \in \mathbb{F}^{K\times N}$ matrix and computing matrix matrices $\mathbf{E} \in \mathbb{F}^{N\times L}$. The approximation error has to be zero in the finite field scenario. We have discovered, in any multi-user linearly-decomposable problem that the more sparse $\mathbf{D}$ matrix, one can achieve a feasible scheme with less communication cost and symmetrically, the more sparse $\mathbf{E}$ matrix, the less computation cost can be achieved.

To characterize the fundamental limits of any multi-user linearly-decomposable problem, in Chapter 2, we tackled this matrix-factorization problem by first focusing on reducing the computation cost (per subfunction), defined as the maximum number of servers that a subfunction has been assigned to. We interestingly, discover that if we re-write the problem as $\mathbf{F}(:,\ell) = \mathbf{D}\mathbf{E}(:,\ell)$, $\ell \in [L]$, we can view the problem as a syndrome decoding algorithm by noting the similarities, then we prove that the maximum number of non-zero elements in each column of $\mathbf{E}$, which is related to the computation cost, can be bounded by the covering radius of $\mathcal{C}_{\mathbf{D}}$, which represents a linear code that its parity-check matrix is $\mathbf{D}$. Via this intuition, we will introduce a new type of code called partial-covering codes and show their existence (both constructively and statistically) which is crucial for any scheme that intends to be optimal from the computation cost, perspective. Moreover, via an algebraic converse, we characterize a converse bound for the computation cost. Then by proving the existence of the Low-Density Parity-Check of partial covering codes, we have been able to offer an achievable scheme with reduced computation and

communication costs. At the same time, the computation cost can be optimal under some additional assumptions on L number of subfunctions and $\mathbf{F}$ matrix. The same method was applied to the multi-shot case and parallel results in terms of computation and communication costs have been achieved.

In Chapter 3, we begin with the same problem formulation but, we mainly focus on the computational delay, defined as the maximum number of subfunctions assigned to a server and the cumulative computational cost, defined as the total number of computations done by all of the servers, which can be lower and upper-bounded as a function of packing radius and packing density of $\mathcal{C}_{\mathbf{H}}$. We interestingly see that if $\mathbf{D}$ be a parity-check matrix of a perfect code, the computational delay and computation cost are optimal for a generic case of the problem.

Our method in Chapter 4, completely deviates from Chapters 2 and 3, since the requested functions and each component function, is defined on real fields, thus the computation and communication matrices have to be defined in real numbers. By rearranging the problem, to a vectorized equivalent of the matrix multiplication, we related the multi-user linearly-separable problem to the compressed sensing literature and offered two solutions to construct a scheme with reduced cumulative computation cost[1].

In Chapters 5 and 6, we elaborate on a new method called Tesselated Distributed Computing which is based on a recently, discovered approach of [53] to tackle the problem of Fixed-Support Matrix Factorization. In the above chapters, our method towards real-valued multi-user linearly-decomposable distributed computing is to reduce both computational delay and the communication cost per server[2] —which is defined as the number of transmissions from a server to the users— simultaneously by partitioning $\mathbf{F}$ submatrices and then performing complete-SVD for the lossless case or truncated SVD, for the case. This approach in Chapter 5, results in characterizing an achievable scheme and converse bound on users per server (will be referred to as rate R) in terms of computation and computation cost, which are optimal in general cases has been derived, which results to the characterization of a feasibility region for general valued K, N, L and computation and communication cost. In the case investigated in Chapter 6, we will offer an achievable scheme with an optimal average error by accumulating the approximation error of each sub-matrix (Tile) and assuming some basic statistical assumptions on the demand matrix $\mathbf{F}$ [3], we used the Marchenko–Pastur law, to characterize the overall average reconstruction error of our problem.

[1] which is the total number of computations done by all of the servers.

[2] The maximum number of transmissions per server.

[3] For instance we have assumed that the element of $\mathbf{F}$, are *i.i.d* distributed with zero-mean and unit variance.

In Chapter 7, we thoroughly examine and conclude, providing further elaboration on future prospects and open problems.

1.3 Main Contributions

The main contributions of this thesis is as follows:

- *Connection to the problem of matrix factorization into sparse components or sparse matrix factorization:* First, when exploring our distributed computing problem, one can see that the lossless feasibility conditions that ensure that each user recovers its desired function, constitute a (preferably sparse) matrix factorization problem of the form

$$\mathbf{DE} = \mathbf{F}$$

  where the problem is over some field $\mathbb{F}$, and where any potential sparsity of $\mathbf{D}$ and $\mathbf{E}$ translates to savings in communication and computation costs respectively. One can find a similar problem when analysing the lossy version of this problem, which can be translated to the optimisation problem, as follows

$$\text{Minimize} \quad \|\mathbf{DE} - \mathbf{F}\|_F \tag{1.2}$$
$$\text{subject to } \|\mathbf{D}\|_0/KN \leq \delta, \|\mathbf{E}\|_0/NL \leq \gamma \tag{1.3}$$

  where $\delta, \gamma \in (0, 1]$, are normalized communication and computation costs of the system. $\mathbf{F}$ is given and $\mathbf{D}$ and $\mathbf{E}$ are variables that have to be achieved by an optimization algorithm (Chapter 2, Chapter 5).

- *Connection to coding theory and syndrome decoding:* On the way to resolve this problem in a manner that yields non-trivial sparse factors, we notice that — if for example, we were to fix the above matrix $\mathbf{D}$, and associate this to the parity-check matrix of some linear code — then for each column $\mathbf{E}_\ell$ of $\mathbf{E}$ and associated column $\mathbf{F}_\ell$ of $\mathbf{F}$, the corresponding equation $\mathbf{D} \cdot \mathbf{E}_\ell = \mathbf{F}_\ell$ would tells us that the desired sparse $\mathbf{E}_\ell$ can be the lowest-weight coset leader whose syndrome is equal to $\mathbf{F}_\ell$. Hence, under this analogy, the columns of $\mathbf{E}$ are associated to error vectors, the columns of $\mathbf{F}$ to the corresponding syndromes, and $\mathbf{D}$ is assigned the role of a parity-check matrix (Chapter 2).

- *Connection to covering codes and the new class of partial covering codes:* The above connection with syndromes, in turn brings about the concept

of covering codes that refer to codes with good covering properties, which in turn entail low weight $\mathbf{E}_\ell$, which is what we need. In (error-control) coding theory though — which generally considers that any error vector is possible — such covering codes consider a full space of possible syndromes, i.e., consider the case where any appropriately-dimensioned vector can indeed be a syndrome. To account for the fact that $\mathbf{F}$ corresponds to a *restricted* set of syndromes (only those that correspond to the columns of our $\mathbf{F}$), we here consider a new class of *partial covering codes*, the analysis of which is part of this work. This connection is articulated through the following theorem in Chapter 2.

Theorem 1: For the setting of distributed-computing with K users, N servers, and L subfunctions, a solution to the linearly separable function computation problem $\mathbf{DE} = \mathbf{F}$ with normalized computation cost per subfunction [4] γ_f exists if and only if $\mathbf{D}$ is the parity check matrix to a $(\gamma_f, \mathcal{X})$-partial covering code $\mathcal{C}_\mathbf{D}$ for some existing set

$$\mathcal{X} \supset \mathcal{X}_{\mathbf{F},\mathbf{D}} \triangleq \{\mathbf{x} \in \mathbb{F}^N | \mathbf{Dx} = \mathbf{F}(:, \ell), \text{ for some } \ell \in [L]\}.$$

With such $\mathbf{D}$ in place, each $\mathbf{E}(:, \ell)$ is the output of the minimum-distance syndrome decoder of $\mathcal{C}_\mathbf{D}$ for syndrome $\mathbf{F}(:, \ell)$.

- *Characterizing a bound on the optimal computation cost*: The contribution is summarized via the following Theorem in Chapter 2:

  Theorem 2: For the setting of distributed-computing of linearly-decomposable functions, with K users, N servers and any number of L subfunctions, the optimal computation cost per subfunction is bounded as

  $$\gamma_f \in (H_q^{-1}(\frac{\log_q(L)}{N}), H_q^{-1}(\frac{K}{N})).$$

  where K/N and $\log_q(L)/N$ are fixed and N goes to infinity.

- *Connection with codes having low-density parity-check matrices:* The above effort yields a sparse $\mathbf{E}$. Our effort is concluded when the aforementioned exploration of covering codes and partial covering codes (which yielded a sparse $\mathbf{E}$), is extended to involve analysis of codes with a sparse $\mathbf{D}$ as well. As the result of this abstract connection we have managed to bound the optimal computation and communication cost as follows in Chapter 2:

[4] which the maximum number of a subfunction assigned to all server. The notion will be defined precisely in Chapter 2.

Theorem 3: For the setting of distributed-computing of linearly-decomposable functions, with K users, N servers and L subfunctions, the optimal computation cost per subfunction is bounded as

$$\gamma_f \in (H_q^{-1}(\frac{\log_q(L)}{N}), H_q^{-1}(\frac{K}{N}))$$

and for any achievable computation cost $\gamma_f \leq \min\{\frac{\sqrt{5}-1}{2}, 1-\frac{1}{q}\}$, then the corresponding achievable communication cost takes the form

$$\delta_c \doteq \frac{\sqrt{\log_q(N)}}{N}. \tag{1.4}$$

where K/N and $\log_q(L)/N$ are fixed and N goes to infinity.

- *Extending the one-shot scenario:* Our framework allows us to address but also extend the one-shot scenario which is the scenario of choice in various works (see for example [22]) and which, in our case, asks that each server can send only one linear combination to one set of users. We extend this model to the practical and realistic scenario where, for a fixed subset of subfunctions/files $\{f(.)\}$ computed locally at each server, the server can communicate linear combinations to various sets of users (Chapter 2, Chapter 6 and Chapter 5). As a result of this extension, we managed to prove:

  Theorem 4: For the setting of distributed-computing of linearly-decomposable functions, with K users, N servers, L sub-functions and T shots, the optimal computation cost per subfunciton γ_f is upper bounded by

  $$\gamma_f \leq TH_q^{-1}(\frac{K}{NT}).$$

  where K/NT and T is fixed and N goes to infinity.

- *Connection with perfect codes, packing radius, and packing density:* To the best of our understanding, this is the first time that perfect codes (and the closely related quasi-perfect codes) have been associated with distributed computing and the equivalent problem of matrix factorization. We derived novel bounds on the cumulative computational cost[5] Γ as well as on the computational delay[6] Λ of a multi-user linearly-decomposable system to capture the importance of the packing density

[5]Which is defined as the total number of subfunctions being processed by all the servers (cf. chapter 2).

[6]which is the maximum number of subfunctions being processed by a server.

as well as the packing and covering radius of a code whose parity-check matrix is our communication-and-computing matrix $\mathbf{D}$ (Chapter 3). The most important results are summarized as follows,

Theorem 6: The optimal computational delay Λ of the (K, N) multi-user linearly decomposable problem implemented based on the decomposition $\mathbf{DE} = \mathbf{F}$, is bounded as

$$\Lambda \leq \min\{L, \sum_{i=1}^{\tau} \binom{N-1}{i-1}(q-1)^i + (1-\mu_\tau)q^K\}$$

where τ and μ_τ are respectively the packing radius and the corresponding packing density of $\mathcal{C}_{\mathbf{D}}$.

Theorem 7: The optimal cumulative computation cost Γ of the (K, N) multi-user linearly decomposable problem implemented based on the decomposition $\mathbf{DE} = \mathbf{F}$, is bounded as

$$\Gamma \leq \min\{NL, \sum_{i=1}^{\tau} \binom{N}{i}(q-1)^i i + (1-\mu_\tau)q^K \rho\}$$

where τ, ρ and μ_τ are respectively the packing radius, covering radius, and packing density of $\mathcal{C}_{\mathbf{D}}$.

Proposition 4: The optimal computational delay Λ and cumulative computation cost Γ of the (K, N) multi-user linearly decomposable problem with maximal basis, are lower bounded as

$$\Lambda \geq \sum_{i=1}^{\tau} \binom{N-1}{i-1}(q-1)^i, \quad \Gamma \geq \sum_{i=1}^{\tau} \binom{N}{i}(q-1)^i i$$

where τ is the packing radius of $\mathcal{C}_{\mathbf{D}}$. Regarding optimality, we have the following proposition.

Proposition 5: The optimal computational costs Λ and Γ for the cases (K, N) for which a perfect code exists, take the form

$$\Lambda = \sum_{i=1}^{\tau} \binom{N-1}{i-1}(q-1)^i, \quad \Gamma = \sum_{i=1}^{\tau} \binom{N}{i}(q-1)^i i$$

where τ is the packing radius of the used perfect code $\mathcal{C}_{\mathbf{D}}$.

- *Connection with compressed sensing and utilizing its techniques to bound the normalized cumulative computation cost:* In this thesis for the first time, we established a connection between distributed computing and

compressed sensing. We proved that there exists an achievable scheme whose normalized cumulative computational cost is bounded above as $\gamma_c \leq \frac{K}{N}$. This is a probabilistic scheme, where $\mathbf{D}$ is chosen from the Gaussian ensemble, and where the corresponding sparsity of $\mathbf{E}$ is the outcome of a randomized process. Then we propose ℓ_0-minimization, which takes as input $\mathbf{D}$ and $\mathbf{F}$ to yield a sparse $\mathbf{E}$. This minimization though is generally intractable, and for this reason, we draw from the rich literature of compressed sensing to suggest a more practical approach where we show (Theorem 8) that as long as there exists a scheme whose computational cost is bounded by $\gamma_c \leq -r\frac{K}{N}W_{-1}^{-1}(-\frac{2K}{eNr})$ (where $W_{-1}(\cdot)$ is the Lambert function and r is a parameter that calibrates the communication between servers and users) we can in fact employ a tractable basis pursuit ℓ_1-minimization to derive such scheme. The important results which can be found in Chapter 4, are as follows:

Proposition 7 For the multi-user linearly-decomposable distributed computing problem, with K users, N servers and L datasets, employing a random Gaussian matrix $\mathbf{D}$, guarantees that with probability 1, there exists a scheme with bounded normalized cumulative computation cost $\gamma_c \leq K/N$, which serves as an upper bound the ℓ_0-minimal cost.

Theorem 8 For the multi-user linearly-decomposable distributed computing problem, with K users, N servers and L datasets, if a scheme exists with a (κ, β) sub-Gaussian random matrix $\mathbf{D}$ (cf. Lemma 13) for which ℓ_0-minimisation would yield

$$\gamma_c \leq -\frac{1}{r}\frac{K}{N}W_{-1}^{-1}(-\frac{2K}{erN}), \quad 0 \leq K/N \leq 12(2\beta+\kappa)/\kappa^2,$$

where W_{-1} is the first branch Lambert function. Then the corresponding (and unique) $\mathbf{E}$ can be found via basis pursuit ℓ_1-minimization with probability at least $1 - 2e^{-\frac{KL}{r}}$, where $r = 12(4\beta + 2\kappa)/\kappa^2$.

- *Connection to the problem of Fixed-Support Matrix Factorization and Tessellation Theory:* Recently, the work in [53] explored the problem of *Fixed Support (sparse) Matrix Factorization* (FSMF) which intends to approximate a matrix, with two or more factors having a fixed support, priory. We showed that our problem can be translated to a variant of the Fixed-Support Matrix Factorization problem where some families of supports of the factor matrices, relate to some valid solutions with a certain computational delay and communication cost per server. Coming up with a novel approach, we saw that our problem was related to the tessellation theory that concerns covering a rectangular region with shifts and translation of smaller rectangles (Chapter 5).

- *Characterizing the lossless capacity of tessellated distributed computing:* We first consider the case where each user has to and after employing an achievable scheme using novel concepts and algorithms introduced in [53] and a converse using combinatorial tilling arguments [54]. A general lower and upper bound on the number of optimal servers has been characterized via the following theorem in Chapter 5:

  Theorem 9 The Optimal Achievable rate of a lossless K, N, T, Γ, Δ distributed computing setting takes the form $C = K/N_{\text{opt}}$, where

  $$\begin{aligned}
  &\lceil \frac{\min(\Delta, \Gamma)}{T} \rceil \lfloor \frac{K}{\Delta} \rfloor \lfloor \frac{L}{\Gamma} \rfloor + \lceil \frac{\min(\text{mod}(K, \Delta), \Gamma)}{T} \rceil \lfloor \frac{L}{\Gamma} \rfloor \\
  &+ \lceil \frac{\min(\quad \text{mod}\ (L, \Gamma), \Delta)}{T} \rceil \lfloor \frac{K}{\Delta} \rfloor + \lceil \frac{\min(\quad \text{mod}\ (K, \Delta), \quad \text{mod}\ (L, \Gamma))}{T} \rceil \\
  &\geq N_{opt} \\
  &\geq \frac{KL}{T \max(\Gamma, \Delta)}.
  \end{aligned}$$

  The achievable scheme's performance is optimal, for the general single-shot case where $(\Gamma \geq \Delta \,\&\, \Gamma | L \,\&\, T | \Delta)$ or $(\Delta \geq \Gamma \,\&\, \Delta | K \,\&\, T | \Gamma)$ which is composed of all the cases where $\Gamma | L \,\&\, \Delta | K \,\&\, T | \min(\Delta, \Gamma)$, also the case where $T \geq \min(\Delta, \Gamma)$ while for the broad case where $\delta^{-1}, \gamma^{-1} \in \mathbb{N}$, the system capacity takes the form

  $$\begin{cases} T \max(\zeta, \gamma), & \text{if } T \mid L \ \min(\zeta, \gamma) \\ L\zeta\gamma, & \text{if } T > L \min(\zeta, \gamma). \end{cases}$$

  revealing that for the first case, the optimal communication-vs-computation points (γ, δ), are $(\frac{K}{NT}, \frac{T}{K})$ and $(\frac{T}{N}, \frac{L}{NT})$, while for the other case the tradeoff takes the form

  $$\gamma\delta = \frac{1}{N}.$$

- *Characterizing the minimum value of error with given conditions on the computation and communication cost in a statistical setting:* Subsequently, for the lossy case, we will consider the large-N scaling regime with an average error guarantee ϵ, averaged over matrices $\mathbf{F}$ and over the subfunctions' outputs. Employing a similar achievable scheme as in the error-free case, paired with a standard truncated-SVD low-rank approximation approach, and under the assumption that the elements of $\mathbf{F}$ and the output of subfunctions are *i.i.d* with zero mean and unit variance, we can bound the average optimal error by the expression

  $$\Phi_{\text{MP},\lambda}(t, r) \triangleq \int_r^t x f_{\text{MP},\lambda}(x) dx$$

where $\Phi_{\mathrm{MP},\lambda}(t,r)$ is the incomplete first moment of the standard Marchenko-Pastur distribution with parameters $\lambda = \frac{\delta K}{\gamma L} = \frac{\Delta}{\Gamma}, r = (1-\sqrt{\lambda})^2$, where t is the solution to $F_{MP,\lambda}(t) = 1 - T\frac{\gamma N}{K}$, and where $F_{MP,\lambda}(.), f_{MP,\lambda}(.)$ are the CDF and PDF of the same distribution. The scheme that provides the above bound, employs tiles after consideration that the size and shape of the tiles, alters the statistics of the SVD approximations of the parts of the matrices that each tile corresponds to. As it turns out, this scheme and the above performance, is optimal over all choices of $\mathbf{D}$ and $\mathbf{E}$ whose supports $\mathcal{I}, \mathcal{J}$ are disjoint and similar in size (Chapter 6).

Chapter 2

Multi-User Linearly-Decomposable in Finite Fields

2.1 Introduction

As has been mentioned in Chapter 1, distributed computing plays an ever-increasing role in speeding up non-linear and computationally hard computing tasks. As the complexity of these tasks increases, research seeks novel parallel processing techniques to efficiently offload computations to groups of distributed servers, under various frameworks such as MapReduce [5] and Spark [55]. Distributed computing naturally entails several challenges that involve accuracy [56]–[58], scalability [49], [59]–[62], privacy and security [48], [63]–[74], as well as latency and straggler mitigation [18], [22], [24], [32], [33], [36], [75], [76].

This aforementioned effort to efficiently distribute computation load across multiple servers, is intimately intertwined with the concept of communication complexity which refers to the amount of communication required to solve a computation problem when the desired task is distributed among two or more parties [77]. This celebrated computation-vs-communication relationship has been studied in a variety of different forms and scenarios [8], [21], [32], [45], [46], [75], [78]–[84] for various types of problems.

Preliminary description of setting This same relationship between computation and communication costs, is the topic of interest in this chapter for the very broad and practical setting of multi-user, multi-server computation of linearly-decomposable functions. In particular, our setting here considers

a master node that manages N server nodes that must contribute in a distributed manner to the computation of the desired functions of K different users. Under the linearly-decomposable assumption (cf. [22]), we consider that user $k \in \{1, 2, \ldots, K\}$ demands a function $F_k(.)$ that each such requested function takes the basic form

$$F_k(.) = \sum_{\ell=1}^{L} f_{k,\ell} f_\ell(.) = \sum_{\ell=1}^{L} f_{k,\ell} W_\ell \tag{2.1}$$

where in the above, $W_\ell = f_\ell(.)$ denotes the computed output of a subfunction, and where $f_{k,\ell}$ are the combining coefficients which belong, together with the entries of W_ℓ, in some finite field[1]. Upon notification of the users' requests — where these requests are jointly described by the $K \times L$ matrix $\mathbf{F}$ that contains the different coefficients $f_{k,\ell}$ — the master instructs the servers to compute some of the subfunctions $f_\ell(.)$. Each server may naturally compute a different number of functions. Upon completing its computations, each server communicates linear combinations of its locally computed outputs (files) to carefully selected subsets of users. Each user can then only linearly combine what it receives from the servers that have transmitted to it, and the goal is for each user to recover its desired function. The problem is completed when every user k retrieves its desired $F_k(.)$.

We note that there is a clear differentiation between the server nodes that are asked to compute hard (generally non-linear) component functions (subfunctions), and the users that can only linearly combine their received outputs. Generating the so-called output files $W_\ell = f_\ell(.), \ell \in \{1, 2, \ldots, L\}$, can be the result of a computationally intensive task that may for example relate to training a deep learning model on a subfunction, or it can relate to the distributed gradient coding problem [23]–[25], [38], the distributed linear-transform computation problem [26], [45], or even the distributed matrix multiplication and the distributed multivariate polynomial computation problems [18], [27]–[29], [31]–[33], [35], [36], [46].

Brief summary of the basic ingredients of the problem Our setting brings to the fore the following crucial questions.

- How many and which servers must compute each subfunction $f_\ell(.)$?

[1]The setting nicely includes the case where each F_k itself is a linear combination of some linearly separable functions, i.e., where F_k can itself be written as $F_k(D_1, \ldots, D_L) = \sum_{\ell=1}^{L} \sum_{i=1}^{M} f_{k,\ell,i} f_{\ell,i}(.) = \sum_{\ell=1}^{L} \sum_{i=1}^{M} f_{k,\ell,i} W_{\ell,i}$, corresponding to some set of basis subfunctions $f_{\ell,i}(.)$. For simplicity we will henceforth refer to the model in (2.1).

  - This decision defines the computation cost: the more the servers that compute a subfunction, the higher the computation cost. The extreme centralized scenario where each active server would compute all L sub-functions, would imply a maximal computation cost, but a minimal communication cost, equal to (as we can see) one transmission received per user. The other extreme scenario (for the case of $L = N$) would imply a minimal computation cost of 1 subfunction per server, but a maximal communication cost of N shots received per user.

- What linear combinations of its computed outputs must each server generate?

  - These linear combination coefficients in question, define an $N \times L$ matrix $\mathbf{E}$ that describes which servers compute each subfunction, and how each server combines its computed outputs in order to transmit them. This matrix must be designed in consideration of the requested functions, which are themselves described by the aforementioned $K \times L$ matrix $\mathbf{F}$.
  - The number of non-zero elements in $\mathbf{E}$ reflects the computation cost on the collective of servers.

- What fraction of the servers must each user get data from, and from which servers?

  - This defines the communication cost. The more data each user gets, the higher the cost.

- How must each user combine (linearly decode) the computed outputs arriving from the servers?

  - This step is determined by a $K \times N$ communication and decoding matrix $\mathbf{D}$ that must be carefully designed. The number of non-zero elements of $\mathbf{D}$ reflects our communication cost. Having a non-sparse $\mathbf{D}$, implies the need to activate a substantial fraction of the existing communication links.

- How sparse can $\mathbf{D}$ and $\mathbf{E}$ be so that each user recovers their desired function?

  - This defines the overall costs of computation and communication. As one might expect, the larger the number L of possible subfuntions, the higher the worst-case costs. Having a larger L

allows the computing service to provide more refined computations, conceivably though at a higher cost.

To answer these questions, we take a novel approach that employs coding theory. The general idea behind our approach is described as follows.

Brief summary of the new connection to sparse matrix factorization and covering codes

- *Connection to the problem of matrix factorization into sparse components:* First, when exploring our distributed computing problem, one can see that the feasibility conditions that ensure that each user recovers its desired function, constitute in fact a (preferably sparse) matrix factorization problem of the form

$$\mathbf{DE} = \mathbf{F} \tag{2.2}$$

  where the problem is over some q-sized finite field $\mathbb{F}$, and where any potential sparsity of $\mathbf{D}$ and $\mathbf{E}$ translates to savings in communication and computation costs respectively.

- *Connection to coding theory and syndrome decoding:* To then resolve this problem in a manner that yields non-trivial sparse factors, we notice that — if for example, we were to fix the above matrix $\mathbf{D}$, and associate this to the parity-check matrix of some linear code — then for each column $\mathbf{E}_\ell$ of $\mathbf{E}$ and associated column $\mathbf{F}_\ell$ of $\mathbf{F}$, the corresponding equation $\mathbf{D} \cdot \mathbf{E}_\ell = \mathbf{F}_\ell$ would tells us that the desired sparse $\mathbf{E}_\ell$ can be the lowest-weight coset leader whose syndrome is equal to $\mathbf{F}_\ell$. Hence, under this analogy, the columns of $\mathbf{E}$ are associated to error vectors, the columns of $\mathbf{F}$ to the corresponding syndromes, and $\mathbf{D}$ is assigned the role of a parity check matrix, and the question is of which code?

- *Connection to covering codes and the new class of partial covering codes:* The above connection with syndromes, in turn brings about the concept of covering codes that refer to codes with good covering properties, which in turn entail low weight $\mathbf{E}_\ell$, which is what we need. In (error-control) coding theory though — which generally considers that any error vector is possible — such covering codes consider a full space of possible syndromes, i.e. consider the case where any appropriately-dimensioned vector can indeed be a syndrome. To account for the fact that $\mathbf{F}$ corresponds to a *restricted* set of syndromes (only those that correspond to the columns of our $\mathbf{F}$), we here consider a new class of *partial covering codes*, the analysis of which is part of this chapter.

- *Connection with codes having low-density parity-check matrices:* The above effort yields a sparse $\mathbf{E}$. Our effort is concluded when the aforementioned exploration of covering codes and partial covering codes (which yielded a sparse $\mathbf{E}$), is extended to involve analysis of codes with a sparse $\mathbf{D}$ as well.

- *Extending the one-shot scenario:* Our framework allows us to address but also extend the one-shot scenario which is the scenario of choice in various works (see for example [22]) and which, in our case, asks that each server can send only one linear combination to one set of users. We extend this model to the practical and realistic scenario where, for a fixed subset of subfunctions/files $\{f_\ell(.)\}$ computed locally at each server, the server can communicate linear combinations to various sets of users.

Highlights of contributions Our focus is on establishing the normalized computation[2] cost $\gamma_f = \frac{1}{N} \max_{l \in \{1,\dots,L\}} \omega(\mathbf{E}(:,l))$, and the normalized cumulative communication cost $\Delta_c = \omega(\mathbf{D})/KN$. In our setting, $\gamma_f \in (0,1]$ represents the maximum fraction of all servers that must compute any one subfunction, while $\delta_c \in (0,1]$ represents the average fraction of servers that each user gets data from, which in turn simply implies an average number of $\Delta_c = \delta_c N$ 'symbols' received by each user.

We first consider the one-shot case. We proceed to highlight some of the derived results, whose exact statement can be found in the following sections.

- Theorem 1 makes the connection between coding theory and our distributed computing problem, by showing that a (γ_f, δ_c)-feasible distributed computing scheme exists if and only if the decoding matrix $\mathbf{D}$ has a degree of sparsity δ_c and is the parity check matrix of an N-length code $\mathcal{C} \subset \mathbb{F}^N$ over a field $\mathbb{F}$ where this code has minimum normalized distance from each vector $\{\mathbf{x} \in \mathbb{F}^N | \mathbf{D}\mathbf{x} = \mathbf{F}(:,\ell), \ell \in \{1,\dots,L\}\}$ that is at most $\gamma_f N$. This brings to the fore the concept of covering and partial covering codes, where covering codes are codes that guarantee a minimum distance to each vector of the entire vector space, while partial covering codes must guarantee a minimum distance to only a specific subset of the entire space. Establishing the properties of such codes is key to our problem.

[2]Both communication and computation costs will be defined in more detail later on. Also, in the following, $\omega(\cdot)$ represents the well-known Hamming weight of the argument vector or matrix.

- Theorem 2 shows that in the limit of large N, the optimal computation cost per server is in the range $\gamma_f \in (H_q^{-1}(\frac{\log_q(L)}{N}), H_q^{-1}(K/N))$, where H_q is the entropy function over our field of size q. This theorem reveals the role of what one might refer to as the *functional rate* $R_f = \log_q(L)/N$. The higher this rate, the more 'involved' is the space of functions we can compute. In this sense — given that, from the above, $\frac{log_q(L)}{N} \leq H_q(\gamma_f)$ — the expression $H_q(\gamma_f)$ plays the role of an upper bound on what one might call the *functional capacity* of the system.

- Extending the famous covering codes theorem of Blinovskii from [85], we established our bounds on partial covering codes to the setting of codes with low density parity check matrices, revealing that any aforementioned achievable computation cost γ_f, can be achieved with normalized cumulative communication cost that vanishes[3] as $\delta_c \doteq \sqrt{\log_q(N)}/N$. This latter cost will be unboundedly lower than in the uncoded approach of resource-sharing between the two extreme regimes discussed previously in the introduction (See Figure 2.4 in Section 2.4.4). As a consequence, we can talk of an unbounded coding gain in our distributed computing problem.

- We also consider the multi-shot scenario where, for the same fixed subset of subtasks/files $\{f_\ell(.)\}$ computed locally at each server, now the server can communicate different linear combinations to different sets of users. This ability offers a certain degree of refinement that the single-shot scenario may lack. This is exploited, and Theorem 4 reveals a range of parameters for which the multi-shot approach provides computation savings over the single-shot scenario. Interestingly, these computational savings are shown to be unbounded.

2.2 System Model

We consider the multi-user linearly-decomposable distributed computation setting (cf. Fig. 2.1), which consists of K users/clients, N active (non-idle) servers, and a master node that coordinates servers and users. A main characteristic of this setting is that the tasks performed at the servers, substantially outweigh in computation cost of the linear operations performed at the different users. Another defining characteristic is that the cost of having the servers communicate to the users is indeed non-trivial. We consider the

[3]We will henceforth use $\doteq$ to denote asymptotic optimality. This will be clarified later on.

setting where each server can use T consecutive time slots to communicate different messages to different subsets of users, where in particular, during time-slot (shot) $t \in [T]$, server n communicates to some arbitrary user-set $\mathcal{T}_{n,t} \subset [K]$, via a dedicated broadcast channel.

In our setting, each user asks for a (generally non-linear) function from a space of linearly separable functions, where each such function can takes several subfunctions as input. Each desired function can be decomposed into a different linear combination of individual (again generally non-linear, and computationally hard) sub-functions $f_\ell(.)$. Consequently the demanded function $F_k(.)$ of each user $k \in [K]$, and it takes the general linearly-decomposable form

$$F_k(.) \triangleq f_{k,1}f_1(.) + f_{k,2}f_2(.) + \ldots + f_{k,L}f_L(.), \tag{2.3}$$

$$= f_{k,1}W_1 + f_{k,2}W_2 + \ldots + f_{k,L}W_L, \;\; k \in [K] \tag{2.4}$$

where, as previously discussed, $W_\ell = f_\ell(.) \in \mathbb{F}$, $\ell \in [L]$ is a so-called 'file' output, and $f_{k,\ell} \in \mathbb{F}$, $k \in [K], \ell \in [L]$ are the linear combination coefficients. As also mentioned before, F_k itself can be a linear combination of some linearly separable functions.

2.2.1 Phases

The model involves three phases, with the first being the *demand phase*, then the *assignment and computation phase*, and then the *transmission and decoding phase*. In the demand phase, each user $k \in [K]$ sends the information of its desired function $F_k(.)$ to the master node, who then deduces the linearly-decomposable decomposition of this function according to (2.4). Then based on these K desired functions, during the assignment and computation phase, the master assigns some of the subfunctions to each server, who then proceeds to compute these and produce the corresponding files $W_\ell = f_\ell(.)$. In particular, each subfunction $f_\ell(.)$ will be assigned to the servers belonging to some carefully chosen server-set $\mathcal{W}_\ell \subset [N]$.

During the transmission phase, each server $n \in [N]$ broadcasts during time slots $t = 1, 2, \ldots, T$, different linear combinations of the locally computed output files, to different subsets of users $\mathcal{T}_{n,t}$. In particular, during time slot t, each server n transmits

$$z_{n,t} \triangleq \sum_{\ell \in [L]} e_{n,\ell,t} W_\ell, \;\; n \in [N], t \in [T] \tag{2.5}$$

where the so-called encoding coefficients $e_{n,\ell,t} \in \mathbb{F}$ are determined by the master. Finally during the decoding part, each user k linearly combines the

received signals as follows

$$F'_k \triangleq \sum_{n\in[N],t\in[T]} d_{k,n,t} z_{n,t} \tag{2.6}$$

for some decoding coefficients $d_{k,n,t} \in \mathbb{F}, n \in [N], t \in [T]$, determined again by the master node. Naturally $d_{k,n,t} = 0, \forall k \notin \mathcal{T}_{n,t}$. Decoding is successful when $F'_k = F_k$ for all $k \in [K]$.

2.2.2 Computation and Communication Costs

Remembering that $|\mathcal{W}_\ell|$ indicates the number of servers that compute a subfunction $W_\ell = f_\ell(.)$, $\ell \in [L]$, our *normalized per subfunction computation cost* metric takes the form

$$\gamma_f \triangleq \frac{\max\limits_{\ell\in[L]} |\mathcal{W}_\ell|}{N} \tag{2.7}$$

and represents the maximum fraction of all servers that must compute any subfunction.

We defined the computational cost as (7) to have an emphasis on the number of replication of various subfunction computation over the servers. Note that the best possible conceivable parallelization is for the case where only one subfunction is assigned to every active server. In this case, we have $\gamma_f = 1/N$ and we observe that there is no replication for any of the subfunctions and the system is fully parallelized. On the other hand in the extreme case, there is a scheme where each server has to store all of the subfunctions, in this particular case, the master node copied each subfunction N times, so we have the full replication and $\gamma_f = 1$. Note that since we did not impose any assumption on the computational power of each server nor the computational complexity of computing each sub-function, we can not define the computation cost as the maximum number of subfunctions assigned to a server which is also not the case for many applications. Other variants of the computation cost will be analyzed in the incoming chapters.

We also formally define the *normalized cumulative communication cost* as

$$\delta_c \triangleq \frac{\sum_{t=1}^{T}\sum_{n=1}^{N} |\mathcal{T}_{n,t}|}{KN} \tag{2.8}$$

to represent the average fraction of servers that each user gets data from[4],[5].

[4]We here clarify that our setting implies that any link can be exploited, and our metric simply captures how many of these links are engaged when communicating. Reducing the communication cost implies activating fewer of these links, leaving the rest to be used for other responsibilities of the computing network.

[5]The observant reader may notice the computational cost being a worst-case cost, unlike

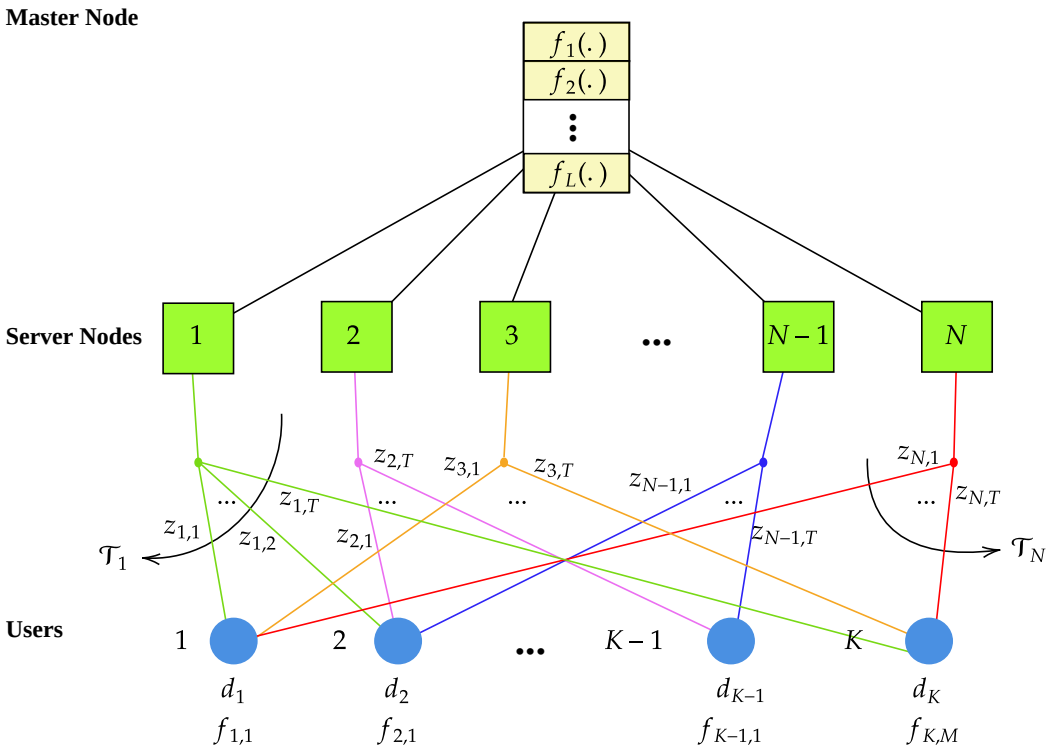


Figure 2.1: The figure represents the K-user, N-server, linearly separable computation setting. In this problem after each user informs the master of its desired function $F_k(.)$, each component subfunction $W_\ell = f_\ell(.)$ is computed at each server in $\mathcal{W}_\ell \subset [N]$. During slot t, each server n broadcasts a linear combination $z_{n,t}$ (of the locally available computed files) to all users in $\mathcal{T}_{n,t}$. This combination is defined by the coefficients $e_{n,\ell,t}$. Finally, to decode, each user $k \in [K]$ linearly combines (based on decoding vectors $\mathbf{d}_k$) all the received signals from all the slots and servers it has received from. Decoding must produce for each user its desired function $F_k(.)$.

Hence in our setting,

$$\Delta_c \triangleq \delta_c N \tag{2.9}$$

represents the average number of transmitted 'symbols' received by each user. We wish to provide schemes that correctly compute the desired functions, at reduced computation and communication costs.

2.3 Problem Formulation: One-Shot Setting

In this single-shot setting of $T = 1$, we will remove the use of the index t. Thus the transmitted value from (2.5) will take the form

$$z_n = \sum_{\ell \in [L]} e_{n,\ell} W_\ell, \;\; n \in [N] \tag{2.10}$$

where $e_{n,\ell} \in \mathbb{F}$ will denote the corresponding encoding coefficients, and where each such transmitted value at server n will now be destined for the users in set $\mathcal{T}_n$. Similarly, the decoding value at each user k (cf. (2.6)) will take the form $F'_k \triangleq \sum_{n \in [N]} d_{k,n} z_n$, where now $d_{k,n}, n \in [N]$, are the decoding coefficients. The desired functions $F_k(.)$ (cf. (2.4)), their linear decomposition coefficients $f_{k,\ell}$ (cf. (2.4)), and the decoded functions $F'_k(.)$ in (2.6), remain as previously described. With the above in place, we will use

$$\mathbf{f} \triangleq [F_1, F_2, \ldots, F_K]^\intercal \tag{2.11}$$

$$\mathbf{f}_k \triangleq [f_{k,1}, f_{k,2}, \ldots, f_{k,L}]^\intercal, \; k \in [K] \tag{2.12}$$

$$\mathbf{w} \triangleq [W_1, W_2, \ldots, W_L]^\intercal \tag{2.13}$$

where $\mathbf{f}$ represents the vector of the output demanded functions (cf. (2.4)), $\mathbf{f}_k$ the vector of function coefficients for user k (cf. (2.4)), and $\mathbf{w}$ the vector of output files. We also have

$$\mathbf{e}_n \triangleq [e_{n,1}, e_{n,2}, \ldots, e_{n,L}]^\intercal, \; n \in [N] \tag{2.14}$$

$$\mathbf{z} \triangleq [z_1, z_2, \ldots, z_N]^\intercal \tag{2.15}$$

respectively representing the encoding vector at server n, and the overall transmitted vector across all the servers (cf. (2.10)). Furthermore, we have

$$\mathbf{d}_k \triangleq [d_{k,1}, d_{k,2}, \ldots, d_{k,N}]^\intercal, \; k \in [K] \tag{2.16}$$

the communication cost which refers to the average case. This choice is essential in making the connection to coding theory. This same choice though has an advantage; it allows us to better capture the effect of having some subfunctions that are much harder to compute than others.

$$\mathbf{f}' \triangleq [F'_1, F'_2, \ldots, F'_K]^\intercal \tag{2.17}$$

respectively representing the decoding vector at user k, and the vector of the decoded functions across all the users. In addition, we have

$$\mathbf{F} \triangleq [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_K]^\intercal \in \mathbb{F}^{K \times L} \tag{2.18}$$

$$\mathbf{E} \triangleq [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N]^\intercal \in \mathbb{F}^{N \times L} \tag{2.19}$$

$$\mathbf{D} \triangleq [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_K]^\intercal \in \mathbb{F}^{K \times N} \tag{2.20}$$

where $\mathbf{F}$ represents the $K \times L$ matrix of all function coefficients across all the users, where $\mathbf{E}$ represents the $N \times L$ *computing and encoding matrix* across all the servers, and where $\mathbf{D}$ represents the $K \times N$ *decoding matrix* across all the users.

Directly from (2.4), we have that

$$\mathbf{f} = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_K]^\intercal \mathbf{w} \tag{2.21}$$

and from (2.5) we have the overall transmitted vector taking the form

$$\mathbf{z} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N]^\intercal \mathbf{w} = \mathbf{E}\mathbf{w}. \tag{2.22}$$

Furthermore, directly from (2.6) we have that

$$F'_k = \mathbf{d}_k^\intercal \mathbf{z} \tag{2.23}$$

and thus we have

$$\mathbf{f}' = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_K]^\intercal \mathbf{z} = \mathbf{D}\mathbf{z}. \tag{2.24}$$

Recall that we must guarantee that

$$\mathbf{f}' = \mathbf{f}. \tag{2.25}$$

After substituting (2.21), (2.22) and (2.24) into (2.25), we see that the above feasibility condition in (2.25) is satisfied iff

$$\mathbf{D}\mathbf{E}\mathbf{w} = \mathbf{F}\mathbf{w}. \tag{2.26}$$

For this to hold for any $\mathbf{w}$, we must thus guarantee

$$\mathbf{D}\mathbf{E} = \mathbf{F}. \tag{2.27}$$

At this point, since $\mathcal{W}_\ell = \sup(\mathbf{E}(:, \{\ell\})^\intercal)$, and since $|\mathcal{W}_\ell| = \omega(\mathbf{E}(:, \{\ell\}))$, we have that

$$\max_{\ell \in [L]} \omega(\mathbf{E}(:, \ell)) = \max_{\ell \in [L]} |\mathcal{W}_\ell| \tag{2.28}$$

which simply tells us that our computation cost γ_f from (2.7) takes the form

$$\gamma_f = \frac{1}{N} \max_{\ell \in [L]} \omega(\mathbf{E}(:, \ell)). \tag{2.29}$$

Similarly, directly from (2.6) and (2.9), we see that

$$\delta_c = \frac{\omega(\mathbf{D})}{KN} \tag{2.30}$$

which simply says (cf. (2.9)) that

$$\Delta_c = \frac{\omega(\mathbf{D})}{K}. \tag{2.31}$$

It is now clear that decomposing $\mathbf{F}$ into the product of two relatively sparse matrices $\mathbf{D}$ and $\mathbf{E}$, implies reduced communication and computation costs respectively.

We here provide a simple example to help clarify the setting and the notation.

2.3.1 Simple Example

As illustrated in Figure 2.2, we consider the example of a system with a master node, $N = 8$ servers, $K = 4$ users, $L = 6$ subfunctions, and a field of size $q = 7$.

Let us assume that the users ask for the following functions:

$$F_1 = 2f_1(D_1) + 4f_2(D_2) + 4f_3(D_3) + 5f_4(D_4) + 5f_5(D_5) = \mathbf{f}_1^\intercal \mathbf{w}, \tag{2.32}$$
$$F_2 = 3f_1(D_1) + 4f_2(D_2) + 5f_3(D_3) + 2f_4(D_4) + 6f_5(D_5) \tag{2.33}$$
$$+ 6f_6(D_6) = \mathbf{f}_2^\intercal \mathbf{w}, \tag{2.34}$$
$$F_3 = 2f_1(D_1) + 4f_2(D_2) + 6f_3(D_3) + 5f_4(D_4) + 2f_5(D_5) = \mathbf{f}_3^\intercal \mathbf{w}, \tag{2.35}$$
$$F_4 = 3f_1(D_1) + 5f_2(D_2) + 2f_4(D_4) + 3f_5(D_5) + f_6(D_6) = \mathbf{f}_4^\intercal \mathbf{w} \tag{2.36}$$

where $F_k, \mathbf{f}_k, \ k \in [4]$, and $\mathbf{w}$, are respectively defined in (2.4), (2.13) and (2.12). Consequently from (2.18), our function matrix takes the form

$$\mathbf{F} = \begin{bmatrix} 2 & 4 & 4 & 5 & 5 & 0 \\ 3 & 4 & 5 & 2 & 6 & 6 \\ 2 & 4 & 6 & 5 & 2 & 0 \\ 3 & 5 & 0 & 2 & 3 & 1 \end{bmatrix}. \tag{2.37}$$

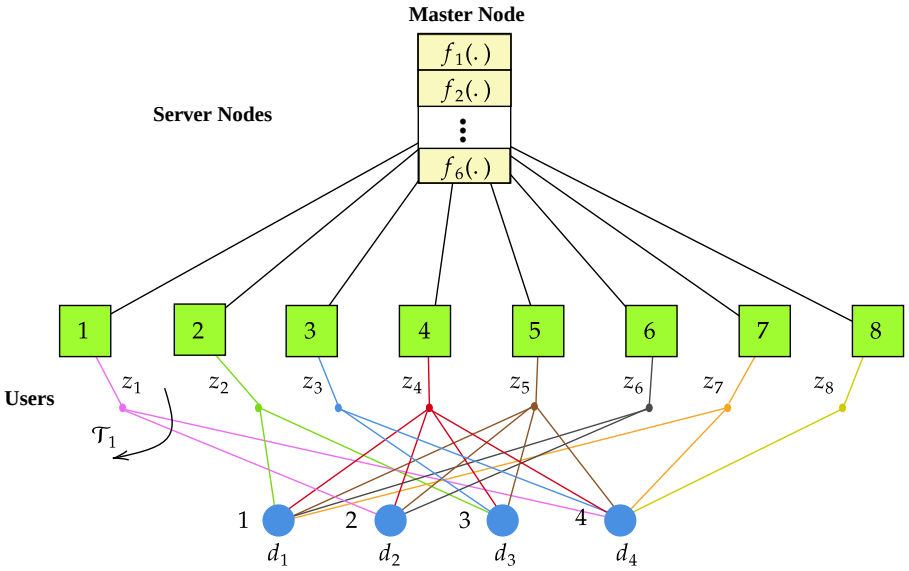


Figure 2.2: Multi-user distributed computing setting with 8 servers, 4 users, and 6 subfunctions.

In the assignment phase, the master allocates the computation of $f_1(D_1)$, $f_2(D_2), \ldots, f_6(D_6)$ to the 8 servers according to

$$\mathcal{W}_1 = \{1,2,3,5,8\},\ \mathcal{W}_2 = \{1,2,3,4,6,7\}, \tag{2.38}$$
$$\mathcal{W}_3 = \{1,2,3\},\ \mathcal{W}_4 = \{1,4,5,7\} \tag{2.39}$$
$$\mathcal{W}_5 = \{1,2,4,5,6,8\},\ \mathcal{W}_6 = \{3,4,5,6,7,8\} \tag{2.40}$$

so that for example subfunction $f_3(D_3)$ is assigned to servers $\{1,2,3\}$, while we can also see that for example server 2 has to compute $W_1 = f_1(.), W_2 = f_2(), W_3 = f_3()$, and $W_5 = f_5(.)$. A quick inspection shows that the normalized computation cost (cf. (2.7)) is equal to

$$\gamma_f = \frac{\max_{\ell\in[6]}|\mathcal{W}_\ell|}{8} = 6/8. \tag{2.41}$$

After computing their designated output files, each server n transmits z_n as follows

$$z_1 = 2W_1 + 6W_2 + 3W_3 + W_4 + 2W_5, \tag{2.42}$$
$$z_2 = 4W_1 + 5W_2 + 2W_3 + 3W_5, \tag{2.43}$$
$$z_3 = W_1 + 2W_2 + W_3 + 2W_6, \tag{2.44}$$

$$z_4 = W_2 + 2W_4 + 4W_5 + W_6, \tag{2.45}$$
$$z_5 = 2W_1 + W_4 + 3W_5 + 2W_6, \tag{2.46}$$
$$z_6 = 2W_2 + 5W_5 + 3W_6 \tag{2.47}$$
$$z_7 = W_2 + 2W_4 + 4W_6, \tag{2.48}$$
$$z_8 = 2W_1 + 4W_5 + 5W_6 \tag{2.49}$$

corresponding to a computing and encoding matrix (cf. (2.22)) of the form

$$\mathbf{E} = \begin{bmatrix} 2 & 6 & 3 & 1 & 2 & 0 \\ 4 & 5 & 2 & 0 & 3 & 0 \\ 1 & 2 & 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 & 4 & 1 \\ 2 & 0 & 0 & 1 & 3 & 2 \\ 0 & 2 & 0 & 0 & 5 & 3 \\ 0 & 1 & 0 & 2 & 0 & 4 \\ 2 & 0 & 0 & 0 & 4 & 5 \end{bmatrix}. \tag{2.50}$$

We can quickly verify (cf. (2.41)) that indeed $\max_{\ell \in [6]} \omega(\mathbf{E}(:, \ell))/8 = 6/8 = \gamma_f$.

Subsequently, the master asks each server n to send its generated z_n to its designated receiving users, where for each server, these user-sets are:

$$\mathcal{T}_1 = \{2, 4\},\ \mathcal{T}_2 = \{1, 3\},\ \mathcal{T}_3 = \{3, 4\},\ \mathcal{T}_4 = \{1, 2, 3, 4\}, \tag{2.51}$$
$$\mathcal{T}_5 = \{1, 2, 3, 4\},\ \mathcal{T}_6 = \{1, 2\}, \mathcal{T}_7 = \{1, 4\}, \mathcal{T}_8 = \{4\} \tag{2.52}$$

so now, for example, server 2 will broadcast z_2 to users 1 and 3. A quick inspection also shows that users 1 and 4 will receive 5 different symbols, whereas users 2 and 3 will receive 4 symbols each. The above corresponds to a normalized cumulative communication cost (cf. (2.9)) equal to

$$\delta_c = \frac{\sum_{n=1}^{8} |\mathcal{T}_n|}{4 \cdot 8} = (5 + 4 + 4 + 6)/32 = 19/32 \tag{2.53}$$

corresponding to an average of $\Delta_c = \frac{19}{4}$ symbols received per user.

To decode, each user $k \in [4]$ computes the linear combination F'_k as

$$\begin{aligned} F'_1 &= 2z_2 + 3z_4 + 4z_5 + 2z_6 + z_7, \\ F'_2 &= 4z_1 + 2z_4 + z_5 + 3z_6, \\ F'_3 &= 4z_2 + 5z_3 + 2z_4 + z_5, \\ F'_4 &= 4z_1 + 2z_3 + z_4 + 2z_5 + 4z_7 + 5z_8 \end{aligned} \tag{2.54}$$

adhering to a decoding matrix of the form

$$\mathbf{D} = \begin{bmatrix} 0 & 2 & 0 & 3 & 4 & 2 & 1 & 0 \\ 4 & 0 & 0 & 2 & 1 & 3 & 0 & 0 \\ 0 & 4 & 5 & 2 & 1 & 0 & 0 & 0 \\ 4 & 0 & 2 & 1 & 2 & 0 & 4 & 5 \end{bmatrix}. \tag{2.55}$$

A quick verification[6] reveals the correctness of decoding, and that indeed $F'_k = F_k$ for all $k = 1, 2, 3, 4$. For example, for the first user, we see that $F'_1 = 2z_2+3z_4+4z_5+2z_6+z_7 = 2(4W_1+5W_2+2W_3+3W_5)+3(W_2+2W_4+4W_5+W_6)+4(2W_1+W_4+3W_5+2W_6)+2(2W_2+5W_5+3W_6)+(W_2+2W_4+4W_6) = 2W_1 + 4W_2 + 4W_3 + 5W_4 + 5W_5 + 0W_6$ which indeed matches F_1. In this example, each user recovers their desired function, with a corresponding normalized per subfunction computation cost $\gamma_f = 3/4$ and a normalized cumulative communication cost $\delta_c = 19/32$. This has just been an example to illustrate the setting. The effort to find a solution with reduced computation and communication costs, follows in the subsequent section.

2.4 Computation and Communication Costs for the Single-Shot Setting

In this section we present the results for the one-shot setting. We first rigorously establish the bridge between our problem, coding theory, covering and partial covering codes. The main results — focusing first on the computational aspects — are presented in Section 2.4.2 which derives bounds on the optimal computation cost in the large N setting. With these results in place, the subsequent Section 2.4.3 extends our consideration to the communication cost as well. Finally, Section 2.4.4 offers some intuition on the results of this current section.

We briefly recall (cf. [86]) that an n-length code $\mathcal{C} \subset \mathbb{F}^n$ is called a ρ-covering code if it satisfies

$$d(\mathbf{x}, \mathcal{C}) \leq \rho n, \ \ \forall \mathbf{x} \in \mathbb{F}^n \tag{2.56}$$

for some $\rho \in (0, 1)$ which is referred to as the normalized covering radius.

2.4.1 Establishing a Relationship to Covering Codes and Partial Covering Codes

We will first seek to decompose $\mathbf{F}$ into $\mathbf{F} = \mathbf{DE}$ under a constrained computation cost γ_f which will generally imply a sparsity constraint on $\mathbf{E}$. For $\mathbf{E}_\ell \triangleq \mathbf{E}(:, \ell)$ and $\mathbf{F}_\ell \triangleq \mathbf{F}(:, \ell)$ denoting the ℓth column of $\mathbf{E}$ and $\mathbf{F}$ respectively, we can rewrite our decomposition as

$$\mathbf{DE}_\ell = \mathbf{F}_\ell, \ \ \forall \ell \in [L]. \tag{2.57}$$

[6]Let us recall that each decoded symbol takes the form $F'_k = \mathbf{d}_k^\intercal \mathbf{z}$ where $\mathbf{d}_k^\intercal$ is the kth row of $\mathbf{D}$, and where $\mathbf{z} = [z_1 \ z_2 \ \cdots \ z_N]^T$.

As suggested before, if we viewed $\mathbf{D} \in \mathbb{F}^{K\times N}$ as a parity check matrix $\mathbf{H}_{\mathcal{C}} = \mathbf{D}$ of a code $\mathcal{C} \subset \mathbb{F}^N$, then we could view $\mathbf{E}_\ell \in \mathbb{F}^N$ as an arbitrary error pattern, and $\mathbf{F}_\ell \in \mathbb{F}^K$ as the corresponding syndrome. Since we wish to sparsify $\mathbf{E}_\ell$, we are interested in having $\mathbf{E}_\ell$ be the minimum-weight coset leader whose syndrome is $\mathbf{F}_\ell$. This is simply the output of the minimum-distance syndrome decoder[7]. To get a first handle on the weights of $\mathbf{E}_\ell$, we can refer to the theory of covering codes which bounds the weights of coset leaders, where these weights are bounded by the code's covering radius $\rho(\mathcal{C})N$, for some normalized radius $\rho(\mathcal{C}) \in (0, 1)$. Since the covering radius ρN upper bounds the weights of the coset leaders[8], it upper bounds our computation cost. A covering radius $\gamma_f N$ would reflect our computation constraint γ_f.

To capture some of the coding-theoretic properties, we will transition to the traditional coding-theoretic notation which speaks of an n-length code $\mathcal{C} \subset \mathbb{F}^n$ of rate k/n, where for us $n = N$ and $k = N - K$. The parity check matrix $\mathbf{H}_{\mathcal{C}} \in \mathbb{F}^{(n-k)\times n}$ will generally be associated to our decoding matrix $\mathbf{D} \in \mathbb{F}^{K\times N}$, the received (or error) vectors $\mathbf{x} \in \mathbb{F}^n$ will be associated to the encoding vectors $\mathbf{E}_\ell \in \mathbb{F}^N$, and its syndrome $\mathbf{s}_{\mathbf{x}} \in \mathbb{F}^{n-k}$ (or just $\mathbf{s}$, depending on the occasion) will be associated to $\mathbf{F}_\ell \in \mathbb{F}^K$. Please recall that when we write $\mathcal{C}_{\mathbf{D}}$ (or $\mathcal{C}_{\mathbf{H}}$), we will refer to the code whose parity check matrix is $\mathbf{D}$ (or $\mathbf{H}$).

As a first step, we extend the concept of covering codes to the following class.

Definition 1. For some $\rho \in (0, 1]$, we say that a set $\mathcal{X} \subseteq \mathbb{F}^n$ is ρ-covered by a code $\mathcal{C} \subseteq \mathbb{F}^n$ iff

$$d(\mathbf{x}, \mathcal{C}) \leq \rho n, \ \ \forall \mathbf{x} \in \mathcal{X} \tag{2.58}$$

in which case we say that $\mathcal{C}$ is a $(\rho, \mathcal{X})$-partial covering code.

Naturally when $\mathcal{X} = \mathbb{F}^n$, such a $(\rho, \mathcal{X})$-partial covering code is simply the traditional covering code. We are now able to link partial covering codes to our distributed computing problem.

Theorem 1. *For the setting of distributed-computing with K users, N servers and L subfunctions, a solution to the linearly separable function computation problem $\mathbf{DE} = \mathbf{F}$ with normalized per subfunction computation cost γ_f exists*

[7]Naturally our viewing $\mathbf{D}$ as a parity check matrix, does not limit the scope of options in choosing $\mathbf{D}$. Similarly, associating $\mathbf{E}_\ell$ the role of an error pattern, or a minimum-weight coset leader, is again not a limiting association.

[8]Let us recall (cf. [87]) that the preferred coset leaders are the minimum-weight vectors in each row of the standard array.

if and only if $\mathbf{D}$ *is the parity check matrix to a* $(\gamma_f, \mathcal{X})$*-partial covering code* $\mathcal{C}_{\mathbf{D}}$ *for some existing set*

$$\mathcal{X} \supset \mathcal{X}_{\mathbf{F},\mathbf{D}} \triangleq \{\mathbf{x} \in \mathbb{F}^N | \mathbf{D}\mathbf{x} = \mathbf{F}(:,\ell), \text{ for some } \ell \in [L]\}. \tag{2.59}$$

With such $\mathbf{D}$ *in place, each* $\mathbf{E}(:,\ell)$ *is the output of the minimum-distance syndrome decoder of* $\mathcal{C}_{\mathbf{D}}$ *for syndrome* $\mathbf{F}(:,\ell)$[9]*.*

Proof. To first prove that the computation constraint $\gamma_f = \rho$ indeed requires $\mathbf{D}$ to correspond to a partial covering code that covers $\mathcal{X}$, let us assume that $\mathbf{D}$ does not have this property, and that there exists an $\mathbf{x} \in \mathcal{X}$ such that $d(\mathbf{x}, \mathcal{C}_{\mathbf{D}}) > \rho n$. Let $\mathbf{c}_{\min}$ be the closest codeword to $\mathbf{x}$ in the sense that $d(\mathbf{x}, \mathbf{c}_{\min}) = d(\mathbf{x}, \mathcal{C}_{\mathbf{D}})$. Now let $\mathbf{e}_{\min} = \mathbf{x} - \mathbf{c}_{\min}$, and note, directly from the above assumption, that $\omega(\mathbf{e}_{\min}) > \rho n$. Naturally $\mathbf{D}\mathbf{x} = \mathbf{D}(\mathbf{e}_{\min} + \mathbf{c}_{min}) = \mathbf{D}\mathbf{e}_{\min}$ by virtue of the fact that $\mathbf{D}$ is the parity check matrix of $\mathcal{C}_{\mathbf{D}}$. Since $\mathbf{x} \in \mathcal{X}$, we know that $\exists\, \ell \in [L]$ such that $\mathbf{D}\mathbf{x} = \mathbf{F}(:,\ell)$, which directly means that $\exists\, \ell \in [L]$ such that $\mathbf{D}\mathbf{e}_{\min} = \mathbf{F}(:,\ell)$. This $\mathbf{e}_{\min}$ is the coset leader associated to syndrome $\mathbf{F}(:,\ell)$.

Since though $\mathbf{D}\mathbf{E} = \mathbf{F}$, we also have that $\mathbf{D}\mathbf{E}(:,\ell) = \mathbf{F}(:,\ell)$. Since $\mathbf{E}(:,\ell)$ and $\mathbf{e}_{\min}$ are in the same coset (of the same syndrome $\mathbf{F}(:,\ell)$), and since $\mathbf{e}_{\min}$ is the minimum-weight coset leader, we can conclude that $\omega(\mathbf{E}(:,\ell)) \geq \mathbf{e}_{\min}$. Thus the assumption that $\omega(\mathbf{e}_{\min}) > \rho n$ implies that $\omega(\mathbf{E}(:,\ell)) > \rho n$ which contradicts the computation-cost requirement that $\omega(\mathbf{E}(:,\ell)) \leq \rho n$ from (2.29). Thus if $\mathbf{D}$ does not correspond to a partial covering code (with $\rho = \gamma_f$) that covers $\mathcal{X}_{\mathbf{F},\mathbf{D}}$, the complexity constraint is violated.

On the other hand, recalling that $\mathcal{C}_{\mathbf{D}}$ is a partial covering code for $\mathcal{X}$, we get that for any $\mathbf{x} \in \mathcal{X}$ then $d(\mathbf{x}, \mathcal{C}_{\mathbf{D}}) \leq \rho n$. For the same $\mathbf{x} \in \mathcal{X}$, let $\mathbf{c}_{\min}$ be again its closest codeword, and let $\mathbf{e}_{\min} = \mathbf{x} - \mathbf{c}_{\min}$, where again by definition of the partial covering code, $\omega(\mathbf{e}_{\min}) \leq \rho n$. Since, like before, $\mathbf{D}\mathbf{e}_{\min} = \mathbf{F}(:,\ell)$ for some $\ell \in [L]$, then we simply set $\mathbf{E}(:,\ell) = \mathbf{e}_{\min}$ whose weight is indeed sufficiently low to guarantee the computation constraint. We recall that for each $\mathbf{F}(:,\ell)$, this coset leader $\mathbf{E}(:,\ell) = \mathbf{e}_{\min}$ can be found by using the minimum-distance syndrome decoder. □

Intuitively, a smaller $\mathcal{X}$ could potentially — depending on $\mathcal{X}$ and the code — be covered in the presence of a smaller covering radius. Now that we have established the connection with partial covering codes, we proceed to present computation bounds. The following result, as well as all subsequent results, assumes large N.

[9]Definition of $\mathcal{X}_{\mathbf{F},\mathbf{D}}$ also can be expressed as $\mathbf{x} \in \mathcal{X}_{\mathbf{F},\mathbf{D}} \iff \exists \ell \in [L] : \mathbf{D}\mathbf{x} = \mathbf{F}(:,\ell)$

2.4.2 Bounds on the Optimal Computation Cost

The following theorem bounds the optimal computation cost of the multi-user linearly-decomposable computation setting.

Theorem 2. *For the setting of distributed-computing of linearly-decomposable functions, with K users, N servers and any number of L subfunctions, the optimal computation cost is bounded as*

$$\gamma_f \in (H_q^{-1}(\frac{\log_q(L)}{N}), H_q^{-1}(\frac{K}{N})). \tag{2.60}$$

where K/N and $\log_q(L)/N$ are fixed and N goes to infinity.

Proof. The proof of the converse (lower bound in (2.60)) employs sphere-covering arguments, and can be found in Appendix 2.8. The proof of achievability follows from covering- and partial covering-code arguments, and can be found in Appendix 2.9. □

Remark 1. The two bounds meet when $L = q^K$.

Theorem 2 suggests a range of computation costs. In the next corollary, we will describe the conditions under which a reduced normalized per subfunction computation cost, strictly inside this range, can be achieved. This reduced cost will relate to (our ability to choose) a set $\mathcal{X} \subset \mathbb{F}^N$. As we will see, a smaller $\mathcal{X}$ will imply a smaller γ_f. To understand the connection between our problem and this set $\mathcal{X}$, and thus to better understand the following theorem whose proof will be presented in Appendix 2.10, we provide the following sketch of some crucial elements in the proof of Theorem 2. In particular, we will here sketch an algorithm that iterates in order to converge to the aforementioned $\mathcal{X}$, and then to the corresponding decoding matrix $\mathbf{D}$, that will eventually provide reduced normalized complexity γ_f. Before describing the algorithm, it is worth noting that a crucial ingredient can be found in Lemma 1 (see Appendix 2.11), which modifies the approach in [88] in order for us to design — for any set $\mathcal{X}' \in \mathbb{F}^N$ — a $(\rho, \mathcal{X}')$-partial covering code for some $\rho = H_q^{-1}(\frac{K}{N} - (1 - \frac{\log_q(|\mathcal{X}'|)}{N})$.

With this in place, the algorithm starts by picking an initial set $\mathcal{X}_0 \in \mathbb{F}^N, |\mathcal{X}_0| = Lq^{N-K}$, and then applies Lemma 1 to construct a $(\rho_0, \mathcal{X}_0)$-partial covering code, $\mathcal{C}_0$, where $\rho_0 = H_q^{-1}(\frac{K}{N} - (1 - \frac{\log_q(|\mathcal{X}_0|)}{N})$. With this code $\mathcal{C}_0$ in place, we create — as a function of $\mathcal{C}_0$ — the set $\mathcal{X}_{\mathbf{F},\mathbf{D},0}$ as defined in (2.59) where $\mathbf{D} = \mathbf{H}_{\mathcal{C}_0}$, and then we check if $\mathcal{X}_0 \supseteq \mathcal{X}_{\mathbf{F},\mathbf{D},0}$. If so, then the algorithm terminates, else it goes to the next iteration which starts by picking a new larger set $\mathcal{X}_1 \in \mathbb{F}^N, |\mathcal{X}_1| = Lq^{N-K} + 1$, then uses Lemma 1 to create a new

$(\rho_1, \mathcal{X}_1)$-partial covering code for $\rho_1 = H_q^{-1}(\frac{K}{N} - (1 - \frac{\log_q(|\mathcal{X}_1|)}{N})$, and then compares if $\mathcal{X}_1 \supseteq \mathcal{X}_{\mathbf{F},\mathbf{D},1}$. This procedure terminates during some round m where this terminating round is the first round for which the chosen set $\mathcal{X}_m$ (now of cardinality $|\mathcal{X}_m| = Lq^{N-K} + m$) and the corresponding $(\rho_m, \mathcal{X}_m)$-partial covering code with $\rho_m = H_q^{-1}(\frac{K}{N} - (1 - \frac{\log_q(|\mathcal{X}_m|)}{N})$, yield $\mathcal{X}_m \supseteq \mathcal{X}_{\mathbf{F},\mathbf{D},m}$.

In the following corollary, the mentioned $\mathcal{X}$ refers to the terminating[10] $\mathcal{X}_m$, and the decoding matrix $\mathbf{D}$ will be the parity-check matrix of the aforementioned $(\rho_m, \mathcal{X}_m)$-partial covering code that covers the terminating $\mathcal{X} = \mathcal{X}_m$, while the normalized per subfunction computation cost in the theorem will take the form $\gamma_f = \rho = \rho_m$.

With the above in place, the following speaks of a set $\mathcal{X}$ that is ρN-covered by a code $\mathcal{C}_\mathbf{D}$ that generates — as described in (2.59) — its set $\mathcal{X}_{\mathbf{F},\mathbf{D}}$.

Corollary 1. *In the multi-user linearly separable computing problem* $\mathbf{DE} = \mathbf{F}$, *if there exists a set*

$$\mathcal{X} \supset \mathcal{X}_{\mathbf{F},\mathbf{D}} \triangleq \{\mathbf{x} \in \mathbb{F}^N | \mathbf{D}\mathbf{x} = \mathbf{F}(:, \ell), \text{ for some } \ell \in [L]\}$$

that is ρN-covered by a code $\mathcal{C}_\mathbf{D}$ for $\rho = H_q^{-1}(\frac{K}{N} - (1 - \frac{\log_q(|\mathcal{X}|)}{N}))$, then the computation cost

$$\gamma_f = H_q^{-1}(\frac{K}{N} - (1 - \frac{\log_q(|\mathcal{X}|)}{N}))$$

is achievable. If $\mathcal{X} = \mathcal{X}_{\mathbf{F},\mathbf{D}}$, then $\gamma_f = H_q^{-1}(\frac{\log_q(L)}{N})$ is achievable and optimal.

Proof. The proof can be found in Appendix 2.10. □

As suggested before, the above reflects that covering a smaller $\mathcal{X}$ could entail a smaller covering radius and thus a smaller computation cost.

2.4.3 Jointly Considering Computation and Communication Costs

The following theorem combines computation and communication considerations. Theorem 3 builds on Theorem 1, where now we recall that any chosen decoding matrix $\mathbf{D}$ will automatically yield a normalized cumulative communication cost $\delta_c = \frac{\omega(\mathbf{D})}{KN}$ corresponding to $\Delta_c = \delta_c N = \frac{\omega(\mathbf{D})}{K}$. The following bounds this communication cost.

[10]Note that in the worst case this termination will happen when $\mathcal{X}_m = \mathbb{F}^N$, in which case the output code will be a covering code.

Theorem 3. *For the setting of distributed-computing of linearly-decomposable functions, with K users, N servers and L subfunctions, the optimal computation cost is bounded as*

$$\gamma_f \in (H_q^{-1}(\frac{\log_q(L)}{N}), H_q^{-1}(\frac{K}{N})) \tag{2.61}$$

and for any achievable computation cost $\gamma_f \leq \min\{\frac{\sqrt{5}-1}{2}, 1-\frac{1}{q}\}$, then the corresponding achievable communication cost takes the form

$$\delta_c \doteq \frac{\sqrt{\log_q(N)}}{N}. \tag{2.62}$$

where K/N and $\log_q(L)/N$ are fixed and N goes to infinity.

Proof. The proof can be found in Appendix 2.12. □

We here offer a quick sketch of the proof of the above theorem. The proof first employs a modified version of the famous result by Blinovskii in [85] which proved that, as n goes to infinity, almost all random linear codes $\mathcal{C}(k,n)$ are covering codes, as long as the normalized covering radius satisfies $\rho \geq H_q^{-1}(\frac{n-k}{n})$. This modification of Blinovskii's theorem is presented in Theorem 5, whose proof is found in Appendix 2.12. With this modification in place, we prove that almost all (k,n) random linear codes with

$$\rho = H_q^{-1}(\frac{\log_q(|\mathcal{X}|)-k}{n}) \tag{2.63}$$

are $(\rho,\mathcal{X})$-partial covering codes, each for their own set $\mathcal{X} \in \mathbb{F}^n$. This is again in Theorem 5. With this theorem in place, we then employ a concatenation argument (which can be found in the proof of Theorem 3 in Appendix 2.12), to build a sparse parity-check matrix $\mathbf{H}$ of a partial covering code, which — by virtue of the connection made in Theorem 1 — allows us to complete the proof of Theorem 3.

To show that sparse parity check codes can indeed offer reduced computation costs, we had to show that sparse codes can indeed offer good partial covering properties. To do that, we followed some of the steps described below. In particular, we designed an algorithm that begins with constructing a sparse parity check code that can cover, for a given radius ρ_0, a minimum necessary cardinality set $\mathcal{X}_0$, where this minimum cardinality of $|\mathcal{X}_0| = Lq^{N-K}$ is imposed on us by $\mathbf{F}$. The parity-check matrix of this first code is $\mathbf{H}_0$. Then following the steps in the proof of Theorem 1, we set $\mathbf{D} = \mathbf{H}_0$ and check if $\mathcal{X}_0 \supseteq \mathcal{X}_{\mathbf{F},\mathbf{D}}$ holds. If it indeed holds, the algorithm outputs $\mathbf{D}$ and $\mathcal{X}_0$,

and the corresponding cost is $\gamma_f = \rho_0$, where this ρ value is derived from (2.63) by setting $\mathcal{X} = \mathcal{X}_0$. Otherwise the algorithm constructs another sparse partial covering code with a new parity check matrix $\mathbf{H}_1$, now covering a set $\mathcal{X}_1$ with cardinality $|\mathcal{X}_1| = Lq^{N-K} + 1$, and then checks again the same inclusion condition as above. The procedure continues until it terminates, with some covered set $\mathcal{X}_m$ of cardinality $|\mathcal{X}_m| = Lq^{N-K} + m$. As before, reaching $\mathcal{X}_m = \mathbb{F}^N$ will terminate the algorithm (if it has not terminated before that). In the proposition below, the set $\mathcal{X}$ is exactly our terminating set $\mathcal{X}_m$ we referred to above.

Proposition 1. *After adopting the achievable scheme proposed in Theorem 3 together with adopting the corresponding conditions on ρ, δ_c and its corresponding $\mathbf{D}$ that was designed as a function of $\mathbf{F}$, then if there exists a subset $\mathcal{X} \supseteq \mathcal{X}_{\mathbf{F},\mathbf{D}}, \mathcal{X} \subseteq \mathbb{F}^N$, that is ρN-covered by $\mathcal{C}_{\mathbf{D}}$ for some $\rho = H_q^{-1}(\frac{K}{N} - (1 - \frac{\log_q(|\mathcal{X}|)}{N}))$, we can conclude that the computation cost $\gamma_f = H_q^{-1}(\frac{K}{N} - (1 - \frac{\log_q(|\mathcal{X}|)}{N}))$ is achievable. If $\mathcal{X} = \mathcal{X}_{\mathbf{F},\mathbf{D}}$, then the computation cost converges to the optimal $H_q^{-1}(\frac{\log_q(L)}{N})$. The above remains in place for any $\mathbf{D}$ which yields communication cost no less than $\Delta_c = O(\sqrt{\log_q(N)})$.*

Proof. The proof can be found in Appendix 2.14. □

2.4.4 Discussing the Results of the Current Section

Theorem 3 reveals that the optimal computation cost lies in the region $\gamma_f \in (H_q^{-1}(\frac{\log_q(L)}{N}), H_q^{-1}(\frac{K}{N}))$, and that this cost can be achieved with communication cost that vanishes as $\delta_c \doteq \sqrt{\log_q(N)}/N$. To get a better sense of the improvements that come from our coded approach, let us compare this to the uncoded case. Looking at Figure 2.3, this uncoded performance is described by (the line connecting) point 1 and point 2. Point 1, located at $(\gamma_f = 1/N, \delta_c = 1)$, corresponds to the fully parallelized scenario where each server must compute just one subfunction[11], but which in turn implies that each server must communicate to all K users. This scenario corresponds to the decomposition $\mathbf{DI} = \mathbf{F}$ where we maximally[12] sparsify $\mathbf{E}$ by setting it equal to $\mathbf{E} = \mathbf{I}_{N\times N}$.

[11] Due to the single-shot assumption, this corresponds to having $N(q-1) = L$. This matches the converse — in our large N setting — because after writing $L = N(q-1) = \binom{N}{1}(q-1) \simeq q^{NH_q(1/N)} = q^{NH_q(\gamma_f)}$, we see that $H_q^{-1}(\frac{\log_q(L)}{N}) = \gamma_f = \frac{1}{N}$.

[12] Note that the stated $\delta_c = 1$ accounts for the worst-case scenario where $\mathbf{F}$ contains no zero elements.

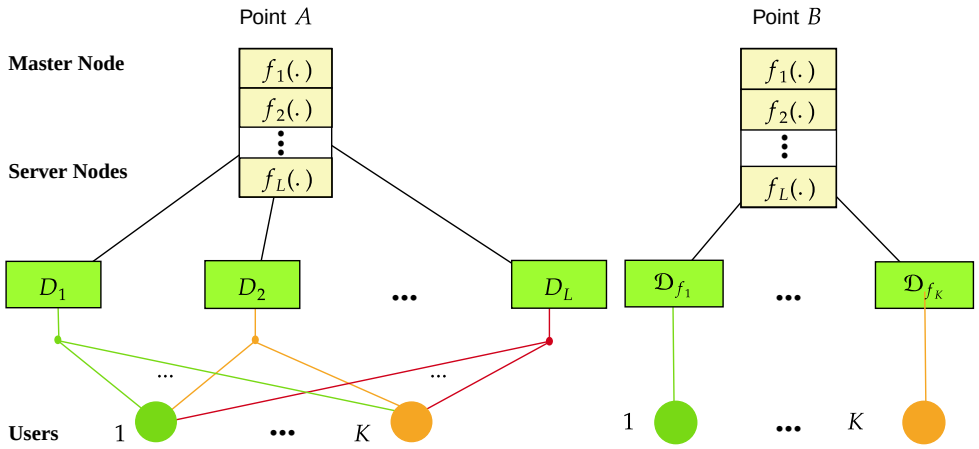


Figure 2.3: (Left. Fully parallelized): Uncoded scheme for point 1 corresponding to $(\gamma_f = 1/N, \delta_c = 1)$. Each of the $N(q-1) = L$ servers, computes one subfunction, but must send to all K users. (Right. Fully centralized): Uncoded scheme for point 2 corresponding to $(\gamma_f = 1, \delta_c = 1/K)$. K activated servers, each computing L subfunctions, and each transmitting to a single user.

On the other hand, point 2, located at $(\gamma_f = 1, \delta_c = 1/N)$, corresponds to the fully centralized scenario where each of the K activated servers[13] is asked to compute all L subfunctions, but where now each server need only transmit to a single user. Point 5 is a trivial converse.

From Theorem 3, we now know that point 3 at $(\gamma_f = H_q^{-1}(\frac{K}{N}), \delta_c \doteq \sqrt{\log_q(N)})/N)$ is a guaranteed achievable point, and so is any point inside the triangle defined by points $1, 2, 3$. Any point inside the region defined by points $1, 4, 2, 3$, is conditionally achievable in accordance to Theorem 1, and in particular in accordance to Corollary 1. The converse also tells us that no point to the left of point 4, i.e., no point with $\gamma_f < H_q^{-1}(\frac{\log_q(L)}{N})$, can be achieved. Finally, the points inside the triangle defined by corner points $5, 2, 4$ could conceivably be achievable under additional techniques that manage to further increase the sparsity of $\mathbf{D}$.

[13]This number of activated users is again a consequence of the single-shot assumption.

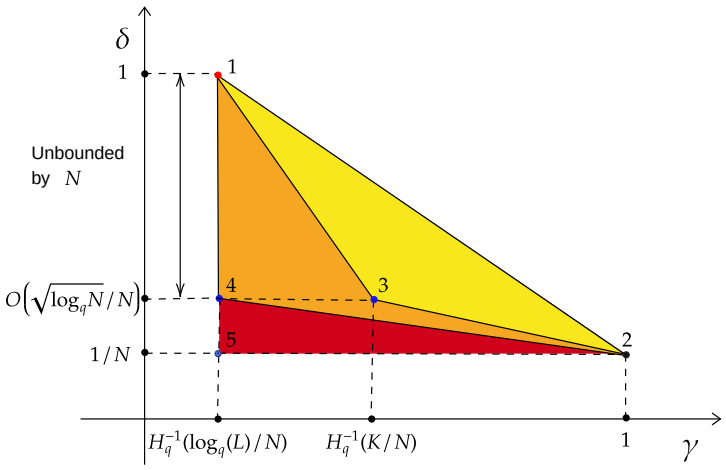


Figure 2.4: The figure summarizes the results of Theorem 3. Recall that while N is asymptotically large, both K/N and $\log_q(L)/N$ are fixed.

2.5 Distributed Computing of Linearly-Decomposable Functions with Multi-Shot Communications ($T > 1$)

In this section we present our results for the multi-shot setting where each server is able to broadcast T consecutive transmissions to T potentially different subsets of users. This is mainly motivated by the fact that having $T > 1$, naturally allows us to employ fewer servers, but it is also motivated — as we will discuss later on — by an additional coding flexibility and refinement that multiple transmissions can provide. We briefly note that we assume as before that K and N are sufficiently large.

2.5.1 Problem Formulation

The notation of the parameters that characterize the system will now generally follow directly from Section 2.3, sometimes after clarifying the corresponding time-slot t of interest. For example, as before we will have

$$\mathbf{f} \triangleq [F_1, F_2, \ldots, F_K]^\intercal, \tag{2.64}$$

$$\mathbf{f}_k \triangleq [f_{k,1}, f_{k,2}, \ldots, f_{k,L}]^\intercal, \ k \in [K], \tag{2.65}$$

$$\mathbf{w} \triangleq [W_1, W_2, \ldots, W_L]^\intercal, \tag{2.66}$$

$$\mathbf{f}' \triangleq [F'_1, F'_2, \ldots, F'_K]^\intercal, \tag{2.67}$$

$$\mathbf{F} \triangleq [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_K]^\intercal. \tag{2.68}$$

On the other hand, the notation for the encoding coefficients and the corresponding transmitted symbols during slot t, will now take the slightly modified form

$$\mathbf{e}_{n,t} \triangleq [e_{n,1,t}, e_{n,2,t}, \ldots, e_{n,L,t}]^\intercal,\ n \in [N],\ t \in [T], \tag{2.69}$$

$$\mathbf{z}_t \triangleq [z_{1,t}, z_{2,t}, \ldots, z_{N,t}]^\intercal,\ t \in [T], \tag{2.70}$$

$$\mathbf{z} \triangleq [\mathbf{z}_1^\intercal, \mathbf{z}_2^\intercal, \ldots, \mathbf{z}_T^\intercal]^\intercal \tag{2.71}$$

with the corresponding modified

$$\mathbf{E}_t \triangleq [\mathbf{e}_{1,t}, \mathbf{e}_{2,t}, \ldots, \mathbf{e}_{N,t}]^\intercal,\ t \in [T] \tag{2.72}$$

while the corresponding decoding coefficients will now take the form

$$\mathbf{d}_{k,t} \triangleq [d_{k,1,t}, d_{k,2,t}, \ldots, d_{k,N,t}]^\intercal,\ k \in [K], t \in [T], \tag{2.73}$$

$$\mathbf{d}_k \triangleq [\mathbf{d}_{k,1}^\intercal, \mathbf{d}_{k,2}^\intercal, \ldots, \mathbf{d}_{k,T}^\intercal]^\intercal,\ k \in [K]. \tag{2.74}$$

We note that the decoding coefficients are decided as a function of all received signals throughout all T transmissions.

As before, (cf. (2.4)), we have that

$$\mathbf{f} = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_K]^\intercal \mathbf{w} \tag{2.75}$$

and now we use

$$\mathbf{z}_t = \mathbf{E}_t \mathbf{w} = [\mathbf{e}_{1,t}, \mathbf{e}_{2,t}, \ldots, \mathbf{e}_{N,t}]^\intercal \mathbf{w} \tag{2.76}$$

to denote the t-th slot transmission vector across all servers. The set of all transmissions now takes the form

$$\mathbf{z} = \mathbf{E}\mathbf{w} \tag{2.77}$$

where now the computing and encoding matrix takes the form

$$\mathbf{E} \triangleq [\mathbf{E}_1^\intercal, \mathbf{E}_2^\intercal, \ldots, \mathbf{E}_T^\intercal]^\intercal \in \mathbb{F}^{NT \times L}. \tag{2.78}$$

Upon decoding, each user k generates

$$F'_k = \mathbf{d}_k^T \mathbf{z} \tag{2.79}$$

and the cumulative set of all decoded elements across the users takes the form

$$\mathbf{f}' = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_K]^\intercal \mathbf{z}. \tag{2.80}$$

Now we note that our decoding matrix

$$\mathbf{D} \triangleq [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_K]^\intercal \in \mathbb{F}^{K \times NT} \tag{2.81}$$

is of dimension $K \times NT$. Naturally, correct decoding requires

$$\mathbf{f} = \mathbf{f}' \tag{2.82}$$

and after substituting (2.75), (2.76), (2.80) into (2.82), we can conclude as before that computing succeeds if and only if

$$\mathbf{DE} = \mathbf{F}. \tag{2.83}$$

The problem remains similar to the one in the single-shot scenario, except that now our communication and decoding matrix $\mathbf{D} \in \mathbb{F}^{K \times NT}$ and computing and encoding matrix $\mathbf{E} \in \mathbb{F}^{NT \times L}$ are bigger[14] and can have a certain restrictive structure.

Again similar to before, each server $n \in [N]$ is asked to compute all the subfunctions in $\cup_{t=1}^{T} \sup(\mathbf{e}_{n,t})$, and thus equivalently the set of servers $\mathcal{W}_\ell$ that must compute subfunction $f_\ell(.)$, takes the form

$$\mathcal{W}_\ell = \cup_{t=1}^{T} \sup(\mathbf{E}([(t-1)N + 1 : tN], \{\ell\})^\intercal), \forall \ell \in [L],\ \forall t \in [T]. \tag{2.84}$$

The following theorem provides an achievable upper bound on the computation cost γ_f of our distributed computing setting for the multi-shot scenario.

Theorem 4. *For the setting of distributed-computing of linearly-decomposable functions, with K users, N servers, L sub-functions and T shots, the optimal computation cost γ_f is upper bounded by*

$$\gamma_f \leq T H_q^{-1}(\frac{K}{NT}). \tag{2.85}$$

where K/NT and T is fixed and N goes to infinity.

Proof. We first note that, directly from (2.84) and the union bound, we have that

$$\max_{\ell \in [L]} \omega(\mathbf{E}(:, \ell)) \geq \max_{\ell \in [L]} |\mathcal{W}_\ell| \tag{2.86}$$

[14]The size of $\mathbf{F} \in \mathbb{F}^{K \times L}$ remains the same, and thus again we have $L \leq q^K$.

and thus our normalized per subfunction computation cost will be upper bounded as

$$\gamma_f \leq \max_{\ell \in [L]} \omega(\mathbf{E}(:, \ell))/N.$$

To bound $\max_{\ell \in [L]} \omega(\mathbf{E}(:, \ell))$, we apply covering code arguments as in the single-shot case, after though accounting for the dimensionality change from having larger matrices. In particular, this means that now the corresponding covering code $\mathcal{C}(n, k)$ will have $n = NT$ and again $K = n - k$ (now we only ask that $NT \geq K$). To account for this increase in n, we note that while the computation cost must still be normalized by the same number of servers N, when considering our covering code[15], we must consider a radius $\frac{\gamma_f}{T} n = \frac{\gamma_f}{T} NT = \gamma_f N$ to guarantee our computation constraint. In other words, the ρ-covering codes that will guarantee the computation constraint, will be for $\rho = \gamma_f / T$. Consequently, combined with the aforementioned union bound, we now see that ρT serves as an achievable upper bound on γ_f. The rest follows directly from the proof of the corresponding theorem in the single-shot scenario. □

Remark 2. Note the here we use the same asymptotic achievable sphere-covering bound used for the single-shot setting which required fixation of K/N while N to be sufficiently large, here we just instead require fixation of K/NT and T while at the same time N has to be sufficiently large. Therefore K in both setting also has to be sufficiently large.

The following two propositions help us make sense of the computational effect of having $T > 1$.

Proposition 2. *In the distributed computing setting of interest in the limit of large T, the normalized per subfunction computation cost γ_f vanishes to zero.*

Proof. The proof is direct once we prove that for any fixed c, then

$$\lim_{T \to \infty} T H_q^{-1}(c/T) = 0. \tag{2.87}$$

This property will be proved in Appendix 2.16.1. □

In a system with an unchanged number of users and servers, the above reveals the notable (unbounded) computational advantage of allowing a

[15] Let us quickly recall that in the previous single-shot scenario, a covering code with covering radius $\rho n = \rho N$ implied a computation cost of $\gamma_f N = \rho N$ and thus a normalized per subfunction computation cost of $\gamma_f = \rho$.

large number T of distinct transmissions per server. This advantage of the multi-shot approach must be seen in light of the fact that in the single-shot approach, the computation cost γ_f was always bounded below by a fixed $\gamma_f \geq H_q^{-1}(\frac{\log_q(L)}{N})$, irrespective of the communication cost. Consequently we can deduce that the computational gains that we see in the regime of larger T, are — at least partly — a result of the increased refinement in transmission that a larger T allows, and it should not be solely attributed to an increased communication cost.

The following proposition discusses the non-asymptotic computational effect of increasing T beyond 1. Recall that our results hold for sufficiently large K and N.

Proposition 3. *For $q = 2$, then γ_f monotonically decreases in T, while for $q > 2$ then γ_f monotonically decreases in T after any $T \geq \lceil \frac{K}{NH_q^{-1}(1/q)} \rceil$.*

Proof. The proof is based on the fact that the derivative of $f = TH_q^{-1}(c/T), 0 \leq c/T \leq 1 - 1/q$, with respect to T, satisfies

$$\frac{\partial f}{\partial T} = \frac{H_q(f/T)}{\log_q\left(\frac{f/T}{1-f/T}(q-1)\right)} + f/T. \tag{2.88}$$

This is proved in Appendix 2.16.2. From the above, and after observing that $\frac{\partial f}{\partial T} \leq 0$ where $0 \leq H_q^{-1}(K/NT) = f/T \leq 1/q$, we can conclude that since $0 \leq H_q^{-1}(K/NT) = f/T \leq 1/2 = 1 - 1/q$, then for $q = 2$, increasing T always strictly reduces γ_f. On the other hand, when $q > 2$, this reduction happens — as we see above — when $T \geq \lceil T_0 \rceil$ for some real T_0 for which $H_q(K/NT_0) = 1/q$. □

2.6 Conclusions

In this chapter we have introduced a new multi-user distributed-computation setting for computing from the broad class of linearly-decomposable functions.

Our work revealed the link between distributed computing and the problem of factorizing a 'functions' matrix $\mathbf{F}$ into a product of two preferably sparse matrices, these being the computing and encoding matrix $\mathbf{E}$ and the decoding matrix $\mathbf{D}$. The work then made the new connection to the area of covering codes, revealing for the first time the importance of these codes in distributed computing problems, as well as in sparse matrix factorization over finite fields. Furthermore, this chapter here brought to the fore the concept of partial covering codes, and the need for codes that cover well smaller subsets of the ambient vector space. For this new class of codes — which constitute a

generalization of covering codes — we have provided some extensions and generalizations of well-studied results in the literature.

Our two metrics — γ_f, representing the maximum fraction of all servers that must compute any subfunction, and δ_c, representing the average fraction of servers that each user gets data from — capture the computation and communication costs, which are often at the very core of distributed computing problems. The observant reader might notice that the creation of $\mathbf{E}$ entails a complexity equal to that of syndrome decoding. Our results hold unchanged when we consider — as suggested before — that the computational cost of evaluating the various subfunctions, far exceeds all other costs. What the results reveal is that in the large N regime, the optimal computation cost lies in the region $\gamma_f \in (H_q^{-1}(\frac{\log_q(L)}{N}), H_q^{-1}(\frac{K}{N}))$, and that this entails the use of a vanishingly small fraction $\delta_c \doteq \sqrt{\log_q(N)}/N$ of all communication resources. What we show is that our coded approach yields unbounded gains over the uncoded scenario, in the sense that the ratio $\frac{\delta_{c_{un}}(\gamma_f)}{\delta_c(\gamma_f)}$ between the uncoded and coded communication costs, is unbounded.

We have also studied the multi-shot setting, where we have explored the gains over the single-shot approach. What we now know is that the gains from increasing T, are unbounded (and strictly increasing) in the regime of large T, whereas in the regime of finite T, the gains are strictly increasing after some threshold value of T. We are thus able to conclude, as suggested before, that computation reductions due to larger T, are — at least partly — a result of the increased refinement in transmission that a larger T allows, and that these gains should not be interpreted as being purely the result of an increased communication load.

Our work naturally relates partly to the recent results in [22] that considered the single-user linearly-separable distributed computing scenario, where a single user may request multiple linearly-separable functions. In this setting in [22], as well as in the extended works in [42] and [89], a key ingredient is the presence of straggling servers, while another key ingredient is that the subfunction-assignment is fixed and oblivious of the actual functions requested by the user. In this context, the coefficients of the functions are assumed to be distributed uniformly and *i.i.d*, and the decodability is probabilistic. There is also an interesting connection (cf. [90], [91]) between compressed sensing and coding theory. Naturally this connection entails no link to covering codes, as the problem of compressed sensing relates to decodability and is very different from the existence problem that we are faced with.

As suggested above, our setting can apply to a broad range of 'well-behaved' functions, and thus can enjoy several use cases, some of which are suggested in our introduction (see also [22] for additional motivation of the linearly

separable function computation problem). When considering problems over the real numbers, we may consider a very large q. An additional new scenario that our work can extend is the so-called hierarchical or tree-like scenario introduced in [80], [92] whose purpose is to ameliorate bandwidth limitations and straggler effects in distributed gradient coding [18]. In this hierarchical setting, each user[16] is connected to a group of servers in a hierarchical manner[17] that allows for a hierarchical aggregation of the sub-gradients. Our approach can extend the hierarchical model by allowing the users to connect to any subset of servers, as well as by allowing them to deviate from the single-shot assumption. Finally as one might expect, our analysis also applies to the transposed computing problem corresponding to $\mathbf{E}^\intercal\mathbf{D}^\intercal = \mathbf{F}^\intercal$, on that case the provided bound will be on the communication cost per user where K the number of users has to be significantly larger than L, the number of files while L/N (the ratio between the number of files and servers) remains fixed since in the above case $\mathbf{E}^\intercal$ would be seen as the parity-check matrix of a partial covering code, not $\mathbf{D}$. We can see that this scenario investigates a less practical scenario where K, the number of users is much bigger than the number of sub-functions.

Also as a suggestion for future work towards a more practical setting, we can conceive of a setting where the size of the transmitted signals sent by different servers $\mathbf{z}_n$, $n \in [N]$, in the single shot scenario differs from each other. More precisely, in this chapter we assumed that the output of each sub-function, so-called file-output $f_\ell(.)$ is just a member of $\mathbf{GF}(q)$ (Cf. (2.4)) so that the model captures the most general and simple instance of the multi-user linearly separable distributed computing problem, because of that we see the transmitted signals $\mathbf{z}_n, n \in [N]$ can be any member of $\mathbf{GF}(q)$ since the signal is just a linear combination of the files and mathematically $\mathbf{z}_n$ can also be any element of $\mathbf{GF}(q)$, therefore in this system model, there is no difference between any of the transmitted signals and the communication cost is simply the total number of activated links.

If we are to analyse the system model where there might exist two transmitted signals $\mathbf{z}_1$ and $\mathbf{z}_2$, with two different sizes then we have to also define a probability measure on each of the output files. For instance, we might investigate the case where W_1 has a non-uniform distribution, or apriori the master knows that $W_1 \neq 0$ while other files have a uniform distribution. In this case, if the master node allocated D_1 to the server 1 but not to the server 2, then $\mathbf{z}_1$ consists of W_1 and other files and $\mathbf{z}_2$ does not contain W_1

[16] In [80], these users are referred to as master nodes.

[17] In particular, each user computes a linearly separable function based on its locally available data, and then sends this to the 'Aggregator' that finally computes the gradient.

in their linear decomposition, then $\mathbf{z}_1$ has a non-uniform distributed while $\mathbf{z}_2$ has a uniform distribution, which makes $H(\mathbf{z}_1) < H(\mathbf{z}_2)$, where H is an entropy defined on random variables $\mathbf{z}_1$ and $\mathbf{z}_2$. We see that this results in an interesting problem where the communication cost also has to be dependent on the output files distribution, which might be dependent on some kind of weighed sparsity criteria of both $\mathbf{D}$ and $\mathbf{E}$.

Another further worthwhile path to investigate this problem is to extend the results to the case where the output files, decoding and encoding procedures has real value which is more applicable for the at hand computational distributed systems. In fact, as we mention above in this chapter we have established a bridge between multi-user linearly separable distributed computing and decoding of linear codes. On the other hand, we now have the enriched literature of compressed sensing, initiated by [93] entitled "Decoding by Linear Programming" which describes a scenario where a decoder receives a real valued noisy signal and its desire is to decode a message, the exact same intention in the error-correcting literature. Therefore one can readily use the results in the compressed sensing literature to drive and explore the same fundamental limits driven in this chapter for the real-valued version of this problem.

Additional considerations that involve stragglers, channel unevenness or computational heterogeneity, are all interesting research directions. In the next Chapters, on the basis of a similar problem formulation, we will investigate the same problem from the lens of perfect codes and its impact on the computation and communication costs concepts of the system. In Chapters 4, 5 and 6, we will investigate the real-valued variant of this problem and in the last chapter we will discuss its results and implications.

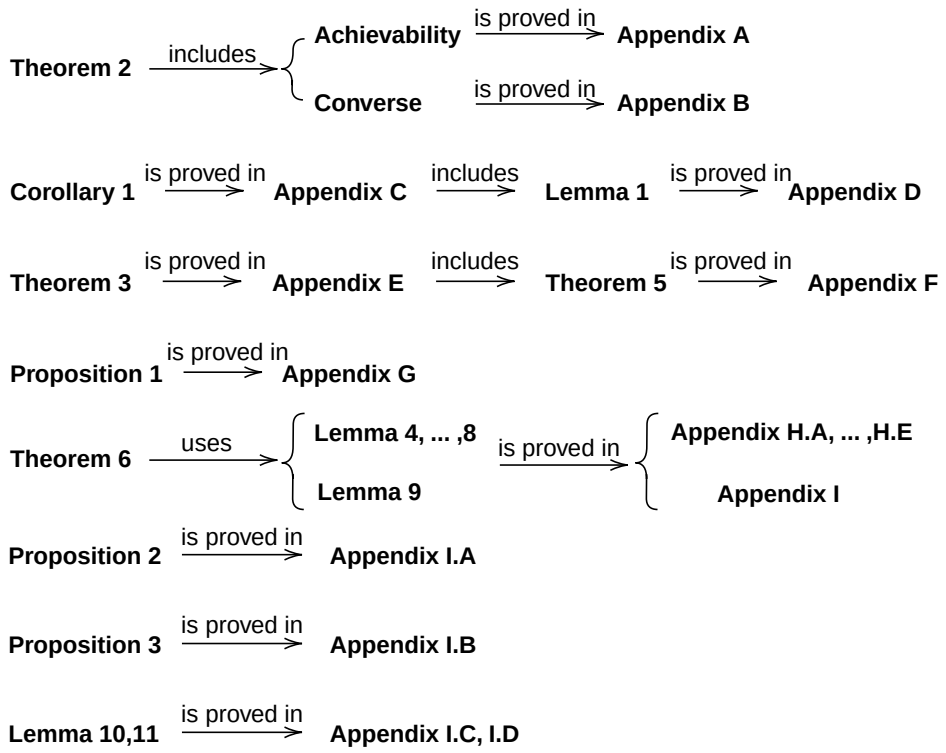


Figure 2.5: Map of lemmas, theorems and appendices.

2.7 Appendices

2.8 Proof of Converse in Theorem 2

To prove the converse in (2.60), we modify the sphere-covering bound for the case of partial covering codes. We wish to show that for a set $\mathcal{X}$ that satisfies $\mathcal{X} \subseteq \mathbb{F}_q^n$, $|\mathcal{X}| = q^k L$, $k \in \mathbb{N}$, a $(\rho, \mathcal{X})$-partial covering code $\mathcal{C}(k, n)$ has to satisfy

$$\log_q(L) \leq \log_q(V_q(n, \rho)). \tag{2.89}$$

This is easy to show because having q^k codewords directly means that the maximum number of points they can jointly ρn-cover is equal to $q^k V_q(n, \rho)$. This in turn implies that

$$Lq^k \leq V_q(n, \rho) q^k \tag{2.90}$$

which yields (2.89) after taking the logarithm on both sides of the inequality.

Now letting the above $\mathcal{X}$ be the $\mathcal{X}$ found in Theorem 1, we note that if $|\mathcal{X}| = Lq^k$ then $\mathcal{X} = \mathcal{X}_F$. Then by substituting $N = n, K = n - k$, we see that $\log_q(L) \leq \log_q(V_q(N, \rho))$. Since $q^{NH_q(\rho)-o(N)} \leq V_q(N, \rho) \leq q^{NH_q(\rho)}$, we can conclude that $\log_q(L) \leq NH_q(\rho)$ and thus that $H_q^{-1}(\frac{\log_q(L)}{N}) \leq \rho$, which concludes the proof. □

2.9 Proof of Achievability in Theorem 2

Directly from [88], we know that there exists at least one ρ-covering code $\mathcal{C}_{\mathcal{X}}(k, n)$ that satisfies

$$n - k \geq \log_q(V_q(n, \rho)) - 2\log_2(n) + \log_q(n) - O(1). \tag{2.91}$$

Then applying Theorem 1 with $\mathbf{D} = \mathbf{H}_{\mathcal{C}}, N = n, K = n - k$ and $\mathcal{X} = \mathbb{F}^n$, allows us to conclude that there exists a feasible scheme for the distributed computing problem, with computation cost $\gamma_f = \rho$, that satisfies

$$K/N \geq \log_q(V_q(N, \rho))/N - 2\log_2(N)/N + \log_q(N)/N - O(1)/N. \tag{2.92}$$

Combining this with the fact that $q^{NH_q(\rho)-o(N)} \leq V_q(N, \rho) \leq q^{NH_q(\rho)}$, yields

$$K/N \geq H_q(\rho) - \epsilon(N) \tag{2.93}$$

which tells us that $\rho \leq H_q^{-1}(K/N + \epsilon(N))$, which in turn proves the result in the limit of large N. □

2.10 Proof of Corollary 1

We first start with the following lemma which proves the existence of a $(\rho, \mathcal{X})$-partial covering linear code $\mathcal{C}$, for a properly-sized set $\mathcal{X} \subseteq \mathbb{F}^n$ that encloses $\mathcal{B}_q(0, \rho)$. Before proceeding with the lemma, we note that the lemma is an outcome of involving a linear greedy algorithm. Let us also briefly recall from Theorem 2 and its proof in Appendix 2.8, that $\log_q(L) \leq \log_q(V_q(n, \rho))$.

Lemma 1. Let $\mathcal{X} \subseteq \mathbb{F}_q^n$ be a set of size $|\mathcal{X}| = L'q^k$ that satisfies $\mathcal{X} \supseteq \mathcal{B}_q(0, \rho)$. Then as long as

$$\log_q(L') \geq \log_q(V_q(n, \rho)) - 2\log_2(n) + \log_q(n) - O(1) \tag{2.94}$$

there exists a $(\rho, \mathcal{X})$-partial covering code.

Proof. The proof is found in Appendix 2.11. □

With this lemma in place, let us define $\mathcal{A}_m \triangleq \{\mathcal{X} \subseteq \mathbb{F}^n \mid |\mathcal{X}| = m, \mathcal{X} \supseteq \mathcal{B}_q(0,\rho)\}$ to be the family of all subsets of $\mathbb{F}^n$ which have cardinality m and which enclose $\mathcal{B}_q(0,\rho)$. Consider the following algorithm.

1. Assign $m = Lq^{N-K}$.
2. For each $\mathcal{X}$ in $\mathcal{A}_m$, find a $(\rho,\mathcal{X})$-partial covering code $\mathcal{C}_\mathcal{X}$ via (the algorithm corresponding to) Lemma 1.
3. For each $\mathcal{X}$ in $\mathcal{A}_m$, set $\mathbf{D} = \mathbf{H}_{\mathcal{C}_\mathcal{X}}$, and create $\mathcal{X}_{\mathbf{F},\mathbf{D}} = \{\mathbf{x} \in \mathbb{F}^N | \mathbf{D}\mathbf{x} = \mathbf{F}(:,\ell),\ \text{for some } \ell \in [L]\}$.
4. If there exists an $\mathcal{X}$ in $\mathcal{A}_m$, for which $\mathcal{X} \subseteq \mathcal{X}_{\mathbf{F},\mathbf{D}}$, then output this $\mathcal{X}$ and its corresponding $\mathbf{D} = \mathbf{H}_{\mathcal{C}_\mathcal{X}}$ from the above step.
5. If there exists no $\mathcal{X}$ in $\mathcal{A}_m$ for which $\mathcal{X} \subseteq \mathcal{X}_{\mathbf{F},\mathbf{D}}$, then increase m by one and go back to step 2.

Let us continue now by supposing that the scheme terminates, outputting $\mathbf{D}$ and $\mathcal{X}$ at the fourth step, before m reaches $m = q^N$. Lemma 1, which guarantees (cf. (2.94)) that

$$\log_q(|\mathcal{X}|q^{-k}) \geq \log_q(V_q(n,\rho)) - 2\log_2(n) + \log_q(n) - O(1) \tag{2.95}$$

also guarantees that

$$\frac{\log_q(|\mathcal{X}|) - (N-K)}{N} \tag{2.96}$$

$$\geq \log_q(V_q(N,\rho))/N - 2\log_2(N)/N + \log_q(N)/N - O(1)/N \tag{2.97}$$

where this last inequality holds after setting $N = n, K = n-k$, and after we divide both sides of (2.95) by N, and then apply Theorem 1 after recalling that $\mathcal{X}$ is indeed ρn-covered by $\mathcal{C}_\mathbf{D}$. Applying that $q^{NH_q(\rho)-o(N)} \leq V_q(N,\rho) \leq q^{NH_q(\rho)}$ into (2.96), gives

$$(\frac{K}{N} - 1 + \frac{\log_q(|\mathcal{X}|)}{N}) \geq H_q(\rho) - \epsilon(N) \tag{2.98}$$

telling us that the algorithm yields a scheme with computation cost[18]

$$\gamma_f = \rho \leq H_q^{-1}(K/N - 1 + \log_q(|\mathcal{X}|/N) + \epsilon(N)) \tag{2.99}$$

[18]Here it is worth elaborating on a fine point regarding our metric. As the reader may recall, γ_f describes the fraction of *active* (non-idle) servers that compute any subfunction. Then the observant reader may wonder if our proposed scheme indeed activates all existing

which matches the stated result in the regime of large N. Note that when $\mathcal{X} = \mathcal{X}_{\mathbf{F},\mathbf{D}}$, then naturally $|\mathcal{X}| = Lq^{N-K}$ which, directly from (2.99), yields $\gamma_f = \rho = H_q^{-1}(\log_q(L)/N + \epsilon(N))$. At the other extreme, when the algorithm terminates at the very end when $\mathcal{X} = \mathbb{F}^N$, then the corresponding code will be the standard ρ-covering code (see Appendix 2.9), and the computation cost will correspond to $\gamma_f = H_q^{-1}(K/N)$. □

2.11 Proof of Lemma 1

We here start by employing the recursive construction approach of Cohen and Frankl in [88]. This recursive approach builds an $(n, j+1)$ code $\mathcal{C}_{j+1}$ from a previous (n, j) code $\mathcal{C}_j$, by carefully adding a vector $\mathbf{x}$ on the basis of $\mathcal{C}_j$, so that now the new basis span is bigger. Our aim will be to recursively construct ever bigger codes that cover an ever increasing portion of our set $\mathcal{X}$.

Let us start by setting $\mathcal{C}_0 = \{\mathbf{0}\}$. Let us then make the assumption that the aforementioned integer L' in Lemma 1, takes the form

$$L' = q^{n-k'} \tag{2.100}$$

for some real $k' \geq k$. Let $Q(\mathcal{C})$ denote the set of points in $\mathcal{X}$ that are not ρn-covered by $\mathcal{C}$, and let

$$q(\mathcal{C}) \triangleq \frac{|Q(\mathcal{C})|}{q^{n+k-k'}} \tag{2.101}$$

where naturally

$$|Q(\mathcal{C}_0)| = q^{n+k-k'} - V_q(n, \rho) \tag{2.102}$$

servers. This corresponds to having a scheme with an $\mathbf{E}$ matrix that has no all-zero rows. In the (rare) degenerate scenario where a row of $\mathbf{E}$ may contain only zeros, then our derived computation cost γ_f would — by definition — have to be recalculated (to account for having idle servers) and would be higher than stated here. To account for this degenerate case, we add a small step in our algorithm which reduces the recorded computation cost by guaranteeing that all servers are active. This step simply says that if a row in $\mathbf{E}$ contains only zeros, then this row is substituted by an arbitrary non-zero row (let's say, the first row) of $\mathbf{E}$, except that, if that (first) row contains a non-zero element in the position $\ell_{\max} = \arg\max\, \omega(\mathbf{E}(:, \ell))$, then this element is substituted by a zero. Then the two servers (the first server and the previously idle server) will split their communication load, except that the server corresponding to the originally all-zero row, will not send any linear combination that involves $w_{\ell_{\max}}$. This small modification guarantees that whatever γ_f we declare here as being achievable, is indeed achievable even in degenerate scenarios. Finally, this degenerate scenario does not affect the algebraic converse.

and

$$q(\mathcal{C}_0) = 1 - V_q(n,\rho)q^{-(n+k-k')}. \tag{2.103}$$

To proceed, we need the following lemma from [88].

Lemma 2 ([88])**.** Let $\mathcal{Y} \subseteq \mathbb{F}^n, \mathcal{Z} \subset \mathbb{F}^n$, and consider $\mathcal{Y}+\mathbf{x} = \{\mathbf{y}+\mathbf{x} : \mathbf{y} \in \mathcal{Y}\}$ for some $\mathbf{x} \in \mathbb{F}$. Then

$$\mathbb{E}(|(\mathcal{Y}+\mathbf{x}) \cap \mathcal{Z}|) = q^{-n}|\mathcal{Y}||\mathcal{Z}| \tag{2.104}$$

where the average is taken, with uniform probability, over all $\mathbf{x} \in \mathbb{F}^n$.

Now, we develop the proof in two parts.

1. **Binary Case:** The proof for $q = 2$ where $k = k'$ (corresponding to the singular case of maximal $L = 2^K$) has been presented in [94] and [88] in two different ways. We will modify the latter approach to establish our claim for any $k' \geq k$ (which will allow us to also handle L values that are smaller than 2^K). First let us easily deduce from Lemma 2 that there exists an $\mathbf{x} \in \mathbb{F}^n$ for which $|(\mathcal{Y}+\mathbf{x}) \cap \mathcal{Z}| \leq \frac{|\mathcal{Y}||\mathcal{Z}|}{q^n}$. Now let us set $\mathcal{Y} = \mathcal{Z} = Q(\mathcal{C}_j)$, and let us append a vector $\mathbf{x}$ to the generator matrix of $\mathcal{C}_j$ to create $\mathcal{C}_{j+1}$, where $\mathbf{x}$ is chosen to minimize $|\mathcal{Q}(\mathcal{C}_{j+1})|$. Now we can directly verify that

$$\begin{aligned} |\mathcal{Q}(\mathcal{C}_{j+1})| &= |\mathcal{Q}(\mathcal{C}_j) \cap \mathcal{Q}(\mathcal{C}_j+\mathbf{x})| = |\mathcal{Q}(\mathcal{C}_j) \cap (\mathcal{Q}(\mathcal{C}_j)+\mathbf{x})| && (2.105)\\ &\leq |\mathcal{Q}(\mathcal{C}_j)|^2/2^n && (2.106)\end{aligned}$$

which implies that

$$q(\mathcal{C}_{j+1}) \leq q(\mathcal{C}_j)^2 2^{k-k'} \leq q(\mathcal{C}_j)^2 \tag{2.107}$$

where the latter inequality holds because $k' \geq k$. Combining (2.103) and (2.107), gives

$$q(\mathcal{C}_k) \leq q(\mathcal{C}_0)^{2^k} \leq (1 - V_q(n,\rho)2^{-(n-k'+k)})^{2^k} \tag{2.108}$$

where the latter inequality again holds due to the fact that $k' \geq k$. Now let us continue this recursion until k is such that

$$2^k = \lceil (n-k'+k)2^{(n-k'+k)} \ln(2)/V_2(n,\rho) \rceil \tag{2.109}$$

in which case — given that $(1-\frac{1}{x})^x \leq e^{-1}$, $\forall x \geq 1$ — we get that

$$q(\mathcal{C}_k) < 2^{-(n+k-k')} \tag{2.110}$$

which automatically yields that $Q(\mathcal{C}_k) = 0$. This, again with the choice of k in (2.109), tells us that for a set $\mathcal{X}$ that satisfies $\mathcal{B}_q(0,\rho) \subseteq \mathcal{X} \subseteq \mathbb{F}_q^n$, $|\mathcal{X}| = Lq^k$, then indeed there exists a $(\rho, \mathcal{X})$-partial covering code $\mathcal{C}(n,k)$ satisfying

$$0 \leq \log_q(L/V_q(n,\rho)) \tag{2.111}$$

$$+ 2\log_2(\log_q(|\mathcal{X}|)) - \log_q(\log_q(|\mathcal{X}|)) + O(1). \tag{2.112}$$

This conclusion can be considered as a tighter version of Lemma 1. After a few very basic algebraic manipulations we get the proof of Lemma 1, for the binary case of $q = 2$.

2. **Non-Binary Case:** Considering first an arbitrary $\mathcal{Z} \subset \mathbb{F}^n$, we have that

$$\mathbb{E}(1 - (q^{-n+k'-k}|(\mathcal{Z}+\mathbf{x}) \cup \mathcal{Z}|)) \tag{2.113}$$

$$= \mathbb{E}(1 - q^{-n+k'-k}((|(\mathcal{Z}+\mathbf{x})| + |\mathcal{Z}|) - |(\mathcal{Z}+\mathbf{x}) \cap \mathcal{Z}|)) \tag{2.114}$$

$$= \mathbb{E}(1 - 2q^{-n+k'-k}|\mathcal{Z}| + q^{-n+k'-k}|(\mathcal{Z}+\mathbf{x}) \cap \mathcal{Z}|) \tag{2.115}$$

$$\overset{(a)}{=} 1 - 2q^{-n+k'-k}|\mathcal{Z}| + q^{-2n+k'-k}|\mathcal{Z}|^2 \tag{2.116}$$

$$\overset{(b)}{\leq} 1 - 2q^{-(n-k'+k)}|\mathcal{Z}| + q^{-2(n-k'+k)}|\mathcal{Z}|^2 \tag{2.117}$$

$$= (1 - \frac{|\mathcal{Z}|}{q^{(n-k'+k)}})^2 \tag{2.118}$$

where (a) is directly from Lemma 2, and where (b) holds since $k' \geq k$. Similarly to the binary case, we begin with $\mathcal{C}_0 = \{\mathbf{0}\}$, and again recursively extend as

$$\mathcal{C}_{j+1} = < \mathcal{C}_j; \mathbf{x} > \tag{2.119}$$

where $\mathbf{x}$ is chosen so that $|\mathcal{Z}|$ is maximized. We do so, after again setting $\mathcal{Z} = Q(\mathcal{C}_j)$.

At this point, from (2.118) we have that

$$q(\mathcal{C}_{j+1}) \leq q(\mathcal{C}_j)^2. \tag{2.120}$$

We now consider the following lemma from [88].

Lemma 3. ([88, Lemma 2]) For any fixed $\mathcal{Z} \subseteq \mathcal{X} \subset \mathbb{F}^n$ where $|\mathcal{Z}|q^{-(n-k'+k)} = \epsilon < (q(n-k'+k))^{-1}$, then

$$\mathbb{E}_{\mathbf{x}\in\mathbb{F}^n}(1 - q^{-(n-k'+k)}|\cup_{\alpha\in\mathbb{F}_q} \mathcal{Z} + \alpha\mathbf{x}|) \leq (1-\epsilon)^{q(1-(2(n-k'+k))^{-1})}. \tag{2.121}$$

Continuing from $\mathcal{Z} = \mathcal{X} \cap (\cup_{\mathbf{c}\in\mathcal{C}_j} \mathcal{B}_q(\mathbf{c}, \rho))$, where

$$|\mathcal{Z}| < \frac{1}{n} q^{(n-k'+k-1)}, \qquad q(\mathcal{C}_{j+1}) \leq q(\mathcal{C}_j)^{q(1-(2(n-k'+k)^{-1}))}$$

we have that

$$q(\mathcal{C}_{j+1}) \leq (1 - q^{n-k'+k} V_q(n, \rho))^{(q(1-(2(n-k'+k))^{-1}))^j} \tag{2.122}$$

$$\leq (1 - q^{n+k-k'} V_q(n, \rho))^{e^{-0.5} q^j} \tag{2.123}$$

since $(1 - (2(n - k' + k))^{-1}) \geq (1 - (2(n - k' + k))^{-1})^{n-k'+k-1} \geq e^{-0.5}$. For

$$j_1 \triangleq \arg\min_j \{q(\mathcal{C}_j) \leq 1 - (q(n + k - k'))^{-1}\} \tag{2.124}$$

we see that

$$j_1 \leq n - \log_q(q^{k'-k} V_q(n, \rho)) - \log_q(n + k - k') + O(1) \tag{2.125}$$

where the inequality holds by first observing that Lemma 3 yields

$$1 - (q(n - k' + k))^{-1} \leq q(\mathcal{C}_j) \leq (1 - q^{(n-k'+k)V_q(n,\rho)})^{q^{j_1-1}} e^{-1/2} \tag{2.126}$$

and then by comparing the upper and lower bounds in (2.126).

We now have an (n, j_1) code $\mathcal{C}$ and we have (2.120). We are now looking for the minimum number j_2 of generators $\mathbf{x}$ that have to be appended to the generator of $\mathcal{C}$ in order to get a $(n, j_1 + j_2)$ code with $q(\mathcal{C}_{j_1+j_2}) \leq q^{-(n-k'+k)}$. We note that $q(\mathcal{C}_{j_1}) \leq 1 - (q(n - k' + k))^{-1}$, so by (2.126) we only need to ensure that $(1 - (q(n - k' + k))^{-1})^{2^{j_2}} \leq q^{-(n-k'+k)}$, which can be achieved by using

$$j_2 = 2 \log_2(n - k' + k) + O(1). \tag{2.127}$$

Hence for $k = j_1 + j_2$, there indeed exist (n, k) codes with normalized covering radius no bigger than ρ. Applying (2.125), (2.127), (2.100), and the fact that $|\mathcal{X}| = Lq^k$, proves (2.94) and thus proves Lemma 1.

□

2.12 Proof of Theorem 3

We quickly note that the converse (lower bound on γ_f) holds directly from the converse arguments in Theorem 2.

Let us start with the following definition.

Definition 2. Let $\rho \in (0, 1 - \frac{1}{q}]$, and let $\tau \in (0, 1]$. A code $\mathcal{C} \subseteq \mathbb{F}^n$ is said to be a (ρ, τ)-partial covering code if there exists a set $\mathcal{X} \subseteq \mathbb{F}^n$, with $\frac{1}{n}\log_q(|\mathcal{X}|) = 1 - \tau$, that is ρ-covered by $\mathcal{C}$.

We now present a theorem that extends the famous Theorem of Blinovskii in [85], which proved that almost all linear codes satisfy the sphere-covering bound. We recall that $\mathcal{C}_{k,n}$ denotes the ensemble of all linear codes generated by all possible $k \times n$ matrices in $\mathbb{F}^{k\times n}$.

Theorem 5. *Let $\rho \in (0, 1 - \frac{1}{q}]$. Then there exists an infinite sequence k_n that satisfies*

$$\frac{k_n}{n} \leq 1 - \tau - H_q(\rho) + O(n^{-1}\log_q(n)) \tag{2.128}$$

for $\tau \in [0, 1 - H_q(\rho) - \frac{k}{n}]$ so that the fraction of codes $\mathcal{C}_n \in \mathcal{C}_{k_n,n}$ that are (ρ, τ)-partial covering, tends to 1 as n grows to infinity. Thus in the limit of large n, almost all codes of rate less than $1 - \tau - H(\rho)$ will be (ρ, τ)-partial covering.

Proof. The proof can be found in Appendix 2.13. □

Now let us design such covering codes. In the following we will consider the set of codes in $\mathcal{C}_{k_n,n}$ that are (ρ, τ)-partial covering, for the claimed sequence k_n of Theorem 5, and for some real τ. We will also consider $g(n)$ to be the fraction of such (ρ, τ)-partial covering codes among all codes in $\mathcal{C}_{k_n,n}$. The scheme design is defined by the following steps.

1. Assign $m = L$.
2. Set $\tau = \frac{K - \log_q(m)}{N}$.
3. Noticing that the value

$$m_n \triangleq g(n)q^{k_n n} \tag{2.129}$$

serves as a lower bound on the number of (ρ, τ)-partial covering codes in the ensemble $\mathcal{C}_{k_n,n}$, we now create $\mathcal{B} \triangleq \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{m_n}\}$ to be the set of the first m_n such codes.

Now let

$$\mathbf{D}_n \triangleq \begin{bmatrix} \mathbf{H}_{\mathcal{C}_1} & & & \\ & \mathbf{H}_{\mathcal{C}_2} & & \\ & & \ddots & \\ & & & \mathbf{H}_{\mathcal{C}_{m_n}} \end{bmatrix} \tag{2.130}$$

and accordingly set $K = m_n(n - k_n)$, and $N = m_n n$.

Now design $\mathcal{C}_{\mathcal{W}_n} = [\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{m_n}]$, and then create the set

$$\mathcal{X}_{\mathbf{F},\mathbf{D}} \triangleq \{\mathbf{x} \in \mathbb{F}^N | \mathbf{D}\mathbf{x} = \mathbf{F}(:, \ell), \text{for some } \ell \in [L]\}. \tag{2.131}$$

Then create the set

$$\mathcal{X} \triangleq \{\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{m_n}] \mid \mathbf{x}_i \in \mathcal{X}_i\} \tag{2.132}$$

where $\mathcal{X}_i, i \in [m_n]$, is the set of all n-length vectors that are ρn-covered by $\mathcal{C}_i$. Then note that

$$|\mathcal{X}_i| \geq q^{n(1-\tau)}, \forall i \in [m_n] \tag{2.133}$$

because of Definition 2. We now note that for any $\mathbf{x} \in \mathcal{X}$, it is the case that

$$\mathbf{d}(\mathbf{x}, \mathcal{C})/N = \sum_{i=1}^{m_n} \mathbf{d}(\mathbf{x}_i, \mathcal{C}_i)/N \leq \sum_{i=1}^{m_n} \frac{\rho n}{m_n n} = \sum_{i=1}^{m_n} \rho \frac{1}{m_n} = \rho \tag{2.134}$$

which means that $\mathcal{C}_{\mathbf{D}_n}$ is also a $(\rho, \mathcal{X})$-partial covering code. Now if $\mathcal{X} \not\supseteq \mathcal{X}_{\mathbf{F},\mathbf{D}}$, then m has to be increased by one, and the procedure starts again from Step 2.

4. Let us define $k'_n \triangleq n - k_n$. From (2.128), we know that

$$\frac{k'_n}{n} \geq \tau + H_q(\rho) - O(n^{-1} \log_q(n)). \tag{2.135}$$

We now see that $R \triangleq \frac{K}{N} = \frac{k'_n}{n}$ since $K = k'_n m_n, N = n m_n$. Thus, directly from the above, we have that

$$K/N = R = H_q(\rho) + \tau - \epsilon(N). \tag{2.136}$$

We note that as n (and thus N) goes to infinity, the term $O(n^{-1} \log_q(n))$ vanishes, and thus from the above we have that

$$\rho = H_q^{-1}(\frac{\log_q(m)}{N} + \epsilon(N)). \tag{2.137}$$

We also have that

$$\frac{\omega(\mathbf{D}_n)}{K} \overset{(a)}{\leq} \frac{m_n n k'_n}{m_n k'_n} = n \tag{2.138}$$

where (a) holds since $\omega(\mathbf{D}_n) = m_n k_n n$ is the maximum number of nonzero elements that $\mathbf{D}$ can have, due to the block-diagonal design.

After taking the logarithm on both sides of the above, and since $N = m_n n$ and $k_n = (1-R)n$, and after considering (2.129), we have that

$$\log_q(n) + n^2(1-R) + \log_q(g(n)) = \log_q(N) \tag{2.139}$$

and thus we have that $n^2(1-R) \leq \log_q(N)$ and $n \leq \sqrt{\frac{\log_q(N)}{(1-R)}}$. Combining this with (2.138) and Theorem 1, we have that

$$\Delta_c \leq \sqrt{\frac{\log_q(N)}{(1-R)}} \tag{2.140}$$

where, as mentioned before, R is constant.

We can also see that the above design terminates, since reaching $m = q^K$ implies that $\tau = 0$. Then we will have $\mathcal{X}_i = \mathbb{F}^n$ since $|\mathcal{X}_i| = q^n$ from Definition 2. Therefore from (2.132), we will have that $\mathcal{X} = \mathbb{F}^N = \mathbb{F}^{m_n n}$, which means that $\mathcal{C}_{\mathcal{D}_n}(N, N-K)$ is a ρ-covering code, and that $\mathcal{X} \supseteq \mathcal{X}_{\mathbf{F},\mathbf{D}}$, and thus the scheme would terminate at Step 4 with $\gamma_f = \rho = H_q^{-1}(\frac{K}{N} + \epsilon(N))$ from (2.137), and with communication cost as shown in (2.140). □

2.13 Proof of Theorem 5

Before offering the formal proof, we provide a quick sketch of the proof to help the reader place the different steps in context.

First we consider the ensemble[19] of codes $\mathcal{C}_{k^*,n}$, and we prove that with a consistent enumeration of codewords, each nonzero point in $\mathbb{F}^n$ has the same chance to be a codeword of a certain index, as we move across the code ensemble.

Second, we pick a code $\mathcal{C} \in \mathcal{C}_{k^*,n}$ at random, and fix it. Then, based on this code, and for a specific choice of τ (to be described later on), we introduce a random so-called 'covered set' $\mathcal{X}_\mathcal{C}$ of size $2^{n(1-\tau)}$ that includes code $\mathcal{C}$.

Then we will see that every point in $\mathbb{F}^n \backslash \mathcal{B}(\mathbf{0}, \rho)$ has an equal probability — as we go through the choices of $\mathcal{C} \in \mathcal{C}_{k^*,n}$ — of belonging to this subset. To analyze the ρ-coverage of points inside $\mathcal{X}_\mathcal{C}$, we derive $\mathbb{P}(\mathbf{c}_i = \mathbf{x} | \mathbf{x} \in \mathcal{X}_\mathcal{C})$, where $\mathbf{c}_i$ describes the codeword indexed by a fixed i, as we move across the codes (and the corresponding generator matrices) in the ensemble.

[19] The details about the choice of k^* will be described later on.

Toward showing that $\mathcal{X}_{\mathcal{C}}$ is covered by $\mathcal{C}$, we first note that $\mathcal{B}(\mathbf{0},\rho)$ is covered since $\mathbf{0}\in\mathcal{C}$. To prove that the remaining part, $\mathcal{X}_{\mathcal{C}}\backslash\mathcal{B}(\mathbf{0},\rho)$, is also covered by $\mathcal{C}$, we prove that if codes in the ensemble are sufficiently large, then there is, for almost all codes $\mathcal{C}$, a large number (polynomial in n) of codewords that covers each specific point in $\mathbf{x}\in\mathcal{X}_{\mathcal{C}}$. With this in place, we will be able to conclude that almost all codes come close to being (ρ,τ)-partial covering.

Finally, we utilize a linear greedy algorithm and successive appending of a very small number of $\lfloor\log_q n(1-\tau)\rfloor$ carefully selected vectors (cf. Lemma 2) to each of these almost (ρ,τ)-partial covering codes, to render them fully (ρ,τ)-partial covering codes.

We proceed with the formal proof.

Let $k^*\triangleq k-\lceil\log_q(n(1-\tau))\rceil$, $k,n\in\mathbb{N},0\leq\tau\leq 1$ and let $\mathcal{C}_{k^*,n}$ be the ensemble of codes generated by $k^*\times n$ generator matrices whose elements are chosen randomly and independently with probability $\frac{1}{q}$ from $\mathbb{F}_q$. Naturally, any fixed non-zero linear combination of rows of the generator matrix, will generate — as we move across the ensemble of generator matrices — all possible q^n vectors in $\mathbb{F}^n$. The zero codeword corresponds to the void linear combination of rows, and is present in all generated codes. Also let us assume a consistent enumeration of the codewords, in the sense that a word's index is defined by the linear combination of rows of the generator matrix, that generate that codeword, in each code. For example, the word indexed by 5, will vary in value across the different codes, but it will always be defined as the output of a specific (the fifth) linear combination of the corresponding generator matrix. The first codeword in all codes will be the zero word. We proceed with the following lemma.

Lemma 4. For any fixed $i\in[2:2^{k^*}]$, and for any fixed $\mathbf{x}\in\mathbb{F}^n$, then

$$\mathbb{P}(\mathbf{c}_i=\mathbf{x})=q^{-n} \tag{2.141}$$

where the probability is over all codes $\mathcal{C}\in\mathcal{C}_{k^*,n}$.

Proof. The proof is presented in Appendix 2.15.1. □

Let us set $\tau\in[0,1-H_q(\rho)-k^*/n]$, and let us note that for sufficiently[20] large n, we can guarantee that

$$q^{n(1-\tau)}\geq V_q(n,\rho)+q^{k^*}. \tag{2.142}$$

[20]We quickly remind the reader that our results here will hold for sufficiently large n.

Let us now go over the ensemble of codes $\mathcal{C} \in \mathcal{C}_{k^*,n}$, and for each code, let us create the covered set $\mathcal{X}_{\mathcal{C}}$ such that

$$|\mathcal{X}_{\mathcal{C}}| = q^{n(1-\tau)} \tag{2.143}$$

$$\mathcal{C} \cup \mathcal{B}(\mathbf{0}, \rho) \subseteq \mathcal{X}_C \subseteq \mathbb{F}^n. \tag{2.144}$$

We can see that (2.142) is a necessary condition for the above to happen. The procedure for designing $\mathcal{X}_{\mathcal{C}}$, simply starts by taking the union $\mathcal{C} \cup \mathcal{B}(\mathbf{0}, \rho)$, and then proceeds by appending on this union, a sufficiently large number of vectors, chosen uniformly and independently at random from $\mathbb{F}^n \backslash \mathcal{C} \cup \mathcal{B}(0, \rho)$. The following lemma simply says that every point $\mathbf{x} \in \mathbb{F}^n \backslash \mathcal{B}(\mathbf{0}, \rho)$ has an equal probability — as we go through the choices of $\mathcal{C} \in \mathcal{C}_{k^*,n}$ — of belonging to this subset $\mathcal{X}_{\mathcal{C}}$.

Lemma 5. For any fixed $\mathbf{x} \in \mathbb{F}^n$, then

$$\mathbb{P}(\mathbf{x} \in \mathcal{X}_{\mathbf{C}}) = \begin{cases} 1 & \omega(\mathbf{x}) \leq \rho n \\ \frac{q^{n(1-\tau)} - V(\rho,n)}{q^n - V(\rho,n)} & \omega(\mathbf{x}) > \rho n. \end{cases} \tag{2.145}$$

Proof. The proof is presented in Appendix 2.15.2. □

With the above lemma in place, we will now calculate the following conditional probability. The following asks us to first pick and fix a vector $\mathbf{x} \in \mathbb{F}^n$, and then pick an index $i \in [1 : 2^{k^*}]$. Recall — from the above discussion on the consistent enumeration of codewords — that this index will define a codeword $\mathbf{c}_i$, which changes as we go across all the codes $\mathcal{C}$ in the ensemble $\mathcal{C}_{k^*,n}$. The following conditional probability is again calculated over the code ensemble.

Lemma 6. Pick any vector $\mathbf{x} \in \mathbb{F}^n$ and any index $i \in [1 : 2^{k^*}]$. Then

$$\mathbb{P}(\mathbf{c}_i = \mathbf{x} | \mathbf{x} \in \mathcal{X}_{\mathcal{C}}) = \begin{cases} 0 & i = 1, \mathbf{x} \neq 0 \\ 1 & i = 1, \mathbf{x} = 0 \\ q^{-(n)} & i \in [2 : K^*], \mathbf{x} \neq 0, \omega(\mathbf{x}) \leq \rho n \\ q^{-n(1-\tau)}\zeta(n) & i \in [2 : K^*] \text{ and } \omega(\mathbf{x}) > \rho n \end{cases} \tag{2.146}$$

where the term $\zeta(n)$ converges to 1 as n approaches to infinity.

Proof. The proof is presented in Appendix 2.15.3. □

Let us now discuss the coverage of any vector $\mathbf{x} \in \mathcal{X}_{\mathcal{C}}$, as we go along the ensemble $\mathcal{C} \in \mathcal{C}_{k^*,n}$. First of all, it is clear that any $\mathbf{x} \in \mathcal{B}(0, \rho)$ is both ρ-covered by the code $\mathcal{C}$ (because $\mathbf{0} \in \mathcal{C}$) as well as is included in $\mathcal{X}_{\mathcal{C}}$ (because

$\mathbf{x} \in \mathcal{B}(0,\rho) \subset \mathcal{X}_{\mathcal{C}}$). Thus for each code $\mathcal{C} \in \mathcal{C}_{k^*,n}$, for the purposes of the current proof, we can focus on the set

$$\mathcal{X}'_{\mathcal{C}} \triangleq \mathcal{X}_{\mathcal{C}} \backslash \mathcal{B}(\mathbf{0},\rho). \tag{2.147}$$

For every $\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}$, let us define the random variable $\eta_{\mathbf{x},i}$ which takes the value 1 if $\mathbf{c}_i$ ρn-covers $\mathbf{x}$, and which takes the value 0 otherwise. Thus

$$\eta_{\mathbf{x}} \triangleq \sum_{i=1}^{2^{k^*}} \eta_{\mathbf{x},i} \tag{2.148}$$

describes the number of codewords that cover $\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}$. For any fixed $\mathbf{x} \in \mathbb{F}^n$, the following lemma describes the conditional average $\eta_{\mathbf{x}}$, where again the average is taken over the code ensemble.

Lemma 7. For any fixed $\mathbf{x} \in \mathbb{F}^n$, then

$$\mathbb{E}(\eta_{\mathbf{x}}|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}) = |\{0\} \cap \mathcal{B}(\mathbf{x},\rho)| \times 1 \tag{2.149}$$

$$+ (q^{k^*}-1)[|(\mathcal{B}(0,\rho)\backslash\{0\}) \cap \mathcal{B}(\mathbf{x},\rho)|q^{-n} \tag{2.150}$$

$$+ |\mathcal{B}(\mathbf{x},\rho)\backslash\mathcal{B}(0,\rho)|q^{-n(1-\tau)}\zeta(n)] \tag{2.151}$$

where again $\zeta(n) \to 1$ as n increases.

Proof. The proof is in Appendix 2.15.4, and it involves an extension of Blinovskii's Theorem [85], from evaluating $\mathbb{E}(\eta_{\mathbf{x}})$ to evaluating the conditional $\mathbb{E}(\eta_{\mathbf{x}}|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}})$. □

Before proceeding, we need the following lemma which is an extension of a related lemma found in [85]. The following considers as before the set $\mathcal{X}'_{\mathcal{C}} \triangleq \mathcal{X}_{\mathcal{C}} \backslash \mathcal{B}(\mathbf{0},\rho)$, and considers again the variance and expectation, over the aforementioned code ensemble.

Lemma 8. For any fixed $\mathbf{x} \in \mathbb{F}^n$, then

$$\frac{Var(\eta_{\mathbf{x}}|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}})}{\mathbb{E}(\eta_{\mathbf{x}}|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}})q^2} \leq 1. \tag{2.152}$$

Proof. The proof is presented in Appendix 2.15.5. □

Combining (2.152) with Chebyshev's inequality, gives

$$\mathbb{P}(|\eta_{\mathbf{x}} - \mathbb{E}(\eta_{\mathbf{x}}|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}})| > q^{\epsilon+1}\sqrt{\mathbb{E}(\eta_{\mathbf{x}}|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}})}\Big|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}) \tag{2.153}$$

$$< \frac{Var(\eta_{\mathbf{x}}|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}})}{q^{2\epsilon+2}\mathbb{E}(\eta_{\mathbf{x}}|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}})} \leq q^{-2\epsilon}. \tag{2.154}$$

Our aim is to show that, for any $\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}$, — under some conditions on k^* and ϵ — $\eta_{\mathbf{x}}$ will be, with high probability, bigger than 0 which in turn implies that any $\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}$ will, with high probability, be covered by $\mathcal{C}$. To see this, we continue from (2.154). We first see from (2.151) that $\mathbb{E}(\eta_{\mathbf{x}}|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}) > 0$. Hence whenever we have $\eta_{\mathbf{x}} > \mathbb{E}(\eta_{\mathbf{x}}|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}})$, we also have that $\eta_{\mathbf{x}} > 0$. Let us now focus on the remaining scenario where $\eta_{\mathbf{x}} \leq \mathbb{E}(\eta_{\mathbf{x}}|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}})$. In this case, from (2.154), we have that

$$\mathbb{P}\Big(\eta_{\mathbf{x}} \geq \mathbb{E}(\eta_{\mathbf{x}}|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}) - q^{\epsilon+1}\sqrt{\mathbb{E}(\eta_{\mathbf{x}} \mid \mathbf{x} \in \mathcal{X}'_{\mathcal{C}})}\Big|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}})) \geq 1 - q^{-2\epsilon}. \quad (2.155)$$

We also have that

$$\mathbb{P}\Big(\eta_{\mathbf{x}} > 0 \mid \mathbf{x} \in \mathcal{X}'_{\mathcal{C}}\Big) \quad (2.156)$$

$$\geq \mathbb{P}\Big(\eta_{\mathbf{x}} \geq \mathbb{E}(\eta_{\mathbf{x}}|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}) - q^{\epsilon+1}\sqrt{\mathbb{E}(\eta_{\mathbf{x}} \mid \mathbf{x} \in \mathcal{X}'_{\mathcal{C}})} \mid \mathbf{x} \in \mathcal{X}'_{\mathcal{C}}\Big) \quad (2.157)$$

under the assumption that

$$\beta(\epsilon) \triangleq \mathbb{E}(\eta_{\mathbf{x}}|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}) - q^{\epsilon+1}\sqrt{\mathbb{E}(\eta_{\mathbf{x}}|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}})} > 0. \quad (2.158)$$

This assumption will be guaranteed — as we will see later on — by a proper choice of k^* and ϵ.

Now combining (2.155) with (2.157), we will show that

$$\mathbb{P}\Big(\eta_{\mathbf{x}} \geq \mathbb{E}(\eta_{\mathbf{x}}|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}) - q^{\epsilon+1}\sqrt{\mathbb{E}(\eta_{\mathbf{x}} \mid \mathbf{x} \in \mathcal{X}'_{\mathcal{C}})} \mid \mathbf{x} \in \mathcal{X}'_{\mathcal{C}}\Big) \to 1 \quad (2.159)$$

as n grows to infinity.

To guarantee that $\beta(\epsilon) > 0$, we must guarantee that

$$\mathbb{E}(\eta_{\mathbf{x}}|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}) > q^{2\epsilon+2}. \quad (2.160)$$

To do this, given Lemma 7, we must prove that

$$|\{0\} \cap \mathcal{B}(\mathbf{x}, \rho)| \times 1 + (q^{k^*} - 1)[|(\mathcal{B}(0, \rho) \backslash \{0\}) \cap \mathcal{B}(\mathbf{x}, \rho)|q^{-n} \quad (2.161)$$

$$+ |\mathcal{B}(\mathbf{x}, \rho) \backslash \mathcal{B}(0, \rho)|q^{-n(1-\tau)}\zeta(n)] > q^{2\epsilon+2} \quad (2.162)$$

again for some properly chosen k^* and ϵ. The following applies toward this effort.

Lemma 9. For any $\mathcal{C} \in \mathcal{C}_{k^*,n}$, any $\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}$, and any $\rho \in (0, \min\{1-1/q, \frac{\sqrt{5}-1}{2}\}]$, then

$$|\mathcal{B}(\mathbf{x}, \rho) \backslash \mathcal{B}(0, \rho)| > q^{nH_q(\rho)-o(n)}. \quad (2.163)$$

Proof. The proof is in Appendix 2.16. □

We now combine (2.151) and (2.163) to get

$$E(\eta_{\mathbf{x}}|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}) > (q^{k^*} - 1)q^{nH_q(\rho)-o(n)-n(1-\tau)}\zeta(n) \tag{2.164}$$

and we also choose k^*, ϵ, to guarantee (cf. (2.160)) that the inequality

$$(q^{k^*} - 1)q^{nH_q(\rho)-o(n)-n(1-\tau)}\zeta(n) \geq q^{2\epsilon+2} \tag{2.165}$$

holds for large n. Thus with (2.164) and (2.165) in place — something that will indeed be validated by the end of the proof (cf. (2.187)) — we can guarantee (2.160). Thus we know that

$$\mathbb{P}(\eta_{\mathbf{x}} < n^{\alpha}\Big|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}) < q^{-2\epsilon}. \tag{2.166}$$

Following the approach in [85], we consider points in $\mathcal{X}'_{\mathcal{C}}$ that are called 'partial-remote points', which are the points that are ρn-covered by fewer than $n^{\alpha}, \alpha > 1$ codewords. Now let $Q_0(\mathcal{X}'_{\mathcal{C}}) \subset \mathcal{X}'_{\mathcal{C}}$ be the set of partial remote points in $\mathcal{X}'_{\mathcal{C}}$, and let

$$q_0(\mathcal{X}'_{\mathcal{C}}) \triangleq \frac{|\mathcal{Q}_0(\mathcal{X}'_{\mathcal{C}})|}{q^{n(1-\tau)} - V_q(n,\rho)}. \tag{2.167}$$

Now applying (2.166), gives

$$\sum_{\mathbf{x}\in\mathcal{X}'_{\mathcal{C}}} \mathbb{P}(\eta_{\mathbf{x}} < n^{\alpha}\Big|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}) \leq (q^{n(1-\tau)} - V_q(n,\rho))q^{-2\epsilon} \tag{2.168}$$

and thus we see that

$$\sum_{\mathbf{x}\in\mathcal{X}'_{\mathcal{C}}} \mathbb{P}(\eta_{\mathbf{x}} < n^{\alpha}\Big|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}) \overset{(a)}{=} \sum_{\mathbf{x}\in\mathcal{X}'_{\mathcal{C}}} \mathbb{E}[\mathbb{1}(\eta_{\mathbf{x}} < n^{\alpha}\Big|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}})] \tag{2.169}$$

$$\overset{(b)}{=} \mathbb{E}[\sum_{\mathbf{x}\in\mathcal{X}'_{\mathcal{C}}} \mathbb{1}(\eta_{\mathbf{x}} < n^{\alpha}\Big|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}})] \tag{2.170}$$

$$\overset{(c)}{=} \mathbb{E}[|\mathcal{Q}_0(\mathcal{X}'_{\mathcal{C}})|] \tag{2.171}$$

where now the expectation in (a) is over the codes in the ensemble $\mathcal{C}_{k^*,n}$, where (b) results from interchanging the expectation with the summation, and where (c) is by definition of $\mathcal{Q}_0$.

Now combining (2.167), (2.168) and (2.171), we have

$$\mathbb{E}(q_0) \leq q^{-2\epsilon} \tag{2.172}$$

which bounds the average (over the code ensemble) number of partial-remote points in $\mathcal{X}'_{\mathcal{C}}$. Then Markov's inequality directly tells us that

$$\mathbb{P}(q_0 > q^{\epsilon}\mathbb{E}(q_0)) < q^{-\epsilon} \tag{2.173}$$

which means that the expression $q_0 \leq q^{\epsilon}\mathbb{E}(q_0)$ holds for a proportion greater than $1 - q^{-\epsilon}$ of all codes.

Now, in the footsteps of [86], we apply a procedure that successively appends cosets to an initial code $\mathcal{C}' \in \mathcal{C}_{k^*,n}$ that belongs in this above family of codes that indeed satisfies $q_0 \leq q^{\epsilon}\mathbb{E}(q_0)$. Let us quickly remember that the optimal successive appending linear greedy method resulting from Lemma 2 and enclosed in Appendix 2.11, allowed us to prove (2.114)–(2.118) which yielded

$$q(\mathcal{C}_{j+1}) \leq q(\mathcal{C}_j)^2 \tag{2.174}$$

where $\mathcal{C}_{j+1} =< \mathcal{C}_j; \mathbf{x} >$, and where $q(\mathcal{C}_j)$ represented the number of remote points of the code $\mathcal{C}_j$, normalized by $q^{n(1-\tau)} - V_q(n, \rho)$.

With the above in mind, let us now set this first initializing code $\mathcal{C}_0$ to be equal to $\mathcal{C}_0 = \mathcal{C}'$, where $\mathcal{C}'$ is one of the aforementioned 'good' codes that satisfy

$$q_0 \leq q^{\epsilon}\mathbb{E}(q_0). \tag{2.175}$$

Then we will design $\mathcal{C}_1 =< \mathcal{C}_0; \mathbf{x} >$ where $\mathbf{x}$ is a guaranteed-to-exist vector (cf. (2.174)) that increases the span of $\mathcal{C}_0$.

Now we calculate the same quantity we calculated in (2.167), but we do so for $\mathcal{X}'_{\mathcal{C}_1}$. In other words, we calculate

$$q_1(\mathcal{X}'_{\mathcal{C}_1}) \triangleq \frac{|\mathcal{Q}_0(\mathcal{X}'_{\mathcal{C}_1})|}{q^{n(1-\tau)} - V_q(n, \rho)} \tag{2.176}$$

where similar to before, now q_1 represents the average normalized remote points of the code $\mathcal{C}_1$ with respect to its associated $\mathcal{X}_{\mathcal{C}_1}$. We can now see that directly from (2.174), we have that

$$\mathbb{E}(q_1) \leq q_0^2 \tag{2.177}$$

where the above average is taken over all $\mathcal{C}_1$ codes, meaning over all codes that can take the role of our aforementioned $\mathcal{C}_1$, going over all possible initializing codes $\mathcal{C}_0 = \mathcal{C}'$, and over all possible base-expanding vectors $\mathbf{x}$. Now we apply again Markov's inequality, this time over the expanded codes $\mathcal{C}_1$, to get

$$\mathbb{P}(q_1 > q^{\lambda}\mathbb{E}(q_1)) < q^{-\lambda},\ \lambda \in \mathbb{R} \tag{2.178}$$

which tells us — as before — that the proportion of codes $\mathcal{C}_1$ that satisfy

$$q_1 \leq q^{\lambda-2\epsilon} \tag{2.179}$$

is at least $1-q^{-\lambda}$. This proportion of codes that achieve (2.179), is over all generated $\mathcal{C}_1$ that were built over all 'good' $\mathcal{C}_0 = \mathcal{C}'$ that already satisfied (2.175). Thus we now know (cf. (2.173)) that the proportion of codes $\mathcal{C}_1$ — among all codes in the entire ensemble $\mathcal{C}_{k^*+1,n}$ — that satisfy (2.179), is equal to $(1-q^{-\epsilon})(1-q^{-\lambda})$.

We now go from our step 1, to an arbitrary step i, and following the same logic as before, we conclude that the proportion of codes $\mathcal{C}_i$ — where this proportion is among all codes in the entire ensemble $\mathcal{C}_{k^*+i,n}$ — that satisfy

$$q_i < q^{-2^i(\epsilon-\lambda)-\lambda} \tag{2.180}$$

is equal to $(1-q^{-\epsilon})(1-q^{-\lambda})^i$.

Now let us go to some step m which will allow us to terminate. We explain when will this termination happen. Consider, for this step $i=m$, as before, the quantity

$$q_m(\mathcal{X}'_{\mathcal{C}_m}) \triangleq \frac{|\mathcal{Q}_0(\mathcal{X}'_{\mathcal{C}_m})|}{q^{n(1-\tau)} - V_q(n,\rho)} \tag{2.181}$$

where similar to before, now q_m represents the average normalized remote points of the code $\mathcal{C}_m$ with respect to its associated $\mathcal{X}_{\mathcal{C}_m}$. We want the corresponding $\mathcal{Q}_m$ (cf. (2.167)) to be empty. This will be guaranteed when m is such that

$$q_m < q^{-n(1-\tau)} \tag{2.182}$$

where the above guarantee can be provided given that the cardinality of a set is a non-negative integer.

This will be achieved by setting $m = \lceil \log_2(n(1-\tau)) \rceil$ and $\lambda = \epsilon - 1$. This can be indeed verified by considering (2.180) after setting $i = m, \lambda = \epsilon - 1$. In conclusion, the proportion of 'good' codes (among the entire ensemble) designed at this stage m, is no less than

$$(1-q^{-\epsilon})(1-q^{-\epsilon+1})^{\lceil \log_2 n(1-\tau) \rceil} \tag{2.183}$$

and for each such code $\mathcal{C}$, every point $\mathbf{x} \in \mathcal{X}_{\mathcal{C}}$ will be ρn-covered by at least n^α codewords of that same code.

With the above in place, let us return to (2.158) where we wish to guarantee that $\beta(\epsilon) > 0$. Toward this, let us consider (2.183) and in this equation, let us set $\epsilon = 2\log_q \log_2(n(1-\tau))$.

Let us now prove that there exists an $\alpha > 1$ such that

$$\mathbb{E}(\eta_{\mathbf{x}}|\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}) \geq (n(1-\tau))^{\alpha} = (q^{k^*} - 1)q^{nH_q(\rho)-o(n)}q^{-(n-n\tau)}. \tag{2.184}$$

This can be readily shown (cf. (2.164)) by noting that, for large n, then the expression

$$\exists\, \alpha > 1 : (n - n\tau)^{\alpha} = (q^{k^*} - 1)q^{nH_q(\rho)-o(n)}q^{-(n-n\tau)} \tag{2.185}$$

holds. With (2.184) in place, we take the logarithm on both sides of the above, and after dividing by n, we get

$$\frac{k^*}{n} = 1 - \tau - H_q(\rho) + \frac{\alpha \log_q(n - n\tau) + o(n)}{n}. \tag{2.186}$$

Let us now recall that we had conditionally accepted (2.165), by saying that (2.165) holds for some properly chosen ϵ and k^*. We will use the aforementioned $\epsilon = 2\log_q \log_2(n(1-\tau))$ and the k^* from (2.186). Let us apply these values in the LHS of (2.165), and note, after employing (2.185), that for these values in place, it holds that

$$(q^{k^*} - 1)q^{nH_q(\rho)-o(n)-n(1-\tau)}\zeta(n) = (n - n\tau)^{\alpha}\zeta(n).$$

Let us now note that for sufficiently large n then

$$(n - n\tau)^{\alpha}\zeta(n) \geq q^2 \log_2(n(1-\tau))^4 \tag{2.187}$$

simply because the RHS is logarithmic in n. At the same time though, we also note that $q^2 \log_2(n(1-\tau))^4 = q^{2(2\log_q(\log_2(n(1-\tau))))+2}$ and then, by applying the chosen ϵ, we get that $q^2 \log_2(n(1-\tau))^4 = q^{2\epsilon+2}$. Thus we now know that $(n - n\tau)^{\alpha}\zeta(n) \geq q^{2\epsilon+2}$ which, after applying (2.187), gives that

$$(q^{k^*} - 1)q^{nH_q(\rho)-o(n)-n(1-\tau)}\zeta(n) \geq q^{2\epsilon+2}$$

which is exactly (2.165). Thus (2.165) is validated, and consequently, directly, we can also guarantee (2.160), which in turn guarantees $\beta(\epsilon) \geq 0$, which in turn proves that (2.157) indeed holds.

Following the logic immediately before (2.157), and with (2.157) now in place, we can conclude that for any $\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}$, — given our chosen k^* and ϵ — $\eta_{\mathbf{x}}$ will be, with high probability, bigger than 0 which in turn implies that any $\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}$ will, with high probability, be covered by $\mathcal{C}$. As it can be seen, the proportion of partial-covering codes (2.183) approaches 1, as n increases.

The only thing that remains now to be verified is the rate k_n/n of these codes, which was declared in the theorem to be as in (2.128). To verify that

indeed this is our rate, we recall that we have started with a code from $\mathcal{C}_{k^*,n}$ and that we have performed the appending procedure m times, where we chose $m = \log_2(n - n\tau)$. This means that the current message length becomes

$$k_n = k^* + \log_2(n(1-\tau)).$$

Now directly adding m/n on both sides of the equation in (2.186), we have that

$$\frac{k_n}{n} = \frac{k^* + \log_2(n - n\tau)}{n} \tag{2.188}$$

$$= 1 - \tau - H(\rho) + \frac{(\alpha+1)\log_2(n-n\tau) + o(n)}{n} \tag{2.189}$$

which simply says that $\frac{k_n}{n} \leq 1 - \tau - H_q(\rho) + O(n^{-1}\log_q(n))$ as in (2.128). This concludes the proof of the theorem. □

2.14 Proof of Proposition 1

Referring to the proof of Theorem 3 in Appendix 2.12, let us suppose that $L \leq m < q^K$ and that $\mathcal{X} \supseteq \mathcal{X}_{\mathbf{F},\mathbf{D}}$. From (2.132) we see that

$$|\mathcal{X}| \overset{(a)}{=} \Pi_{i=1}^{m_n} |\mathcal{X}_i| \tag{2.190}$$

$$\overset{(b)}{\geq} q^{nm_n(1-\tau)} \tag{2.191}$$

$$\overset{(c)}{=} q^{N(1-\tau)} \tag{2.192}$$

where (a) comes from the definition of $\mathcal{X}$ (cf. (2.132)), where (b) holds due to (2.133), and where (c) holds since $N = nm_n$. Then from (2.136) and (2.192), we can conclude that

$$\rho = H_q^{-1}(\frac{K}{N} - \tau + \epsilon(N)) \tag{2.193}$$

$$\leq H_q^{-1}(\frac{K}{N} - (1 - \frac{\log_q(|\mathcal{X}|)}{N}) + \epsilon(N)). \tag{2.194}$$

Setting $m = L$ gives $\rho = H_q^{-1}(\frac{\log_q(L)}{N} + \epsilon(N))$ (cf. (2.137)). The communication cost is as described in (2.140). □

2.15 Various Proofs

2.15.1 Proof of Lemma 4

For a fixed index $i \neq 0$, there is a fixed information vector $\mathbf{d}_i \in \mathbb{F}^{k^*} \backslash \mathbf{0}$ that generates — as we move across the generator matrices $\mathbf{G}$ in the ensemble of codes $\mathcal{C}_{k,n}$ — the codewords $\mathbf{c}_i$ that take the form

$$\mathbf{c}_i = \mathbf{d}_i \mathbf{G} = \sum_{j=1}^{n} d_i(j,1)\mathbf{G}(j,:). \tag{2.195}$$

Given that the elements of $\mathbf{G} \in \mathbb{F}^{k^* \times n}$ are chosen uniformly and independently at random from $\mathbb{F}$, directly implies that the same holds for the elements of $\mathbf{c}_i$, since in the above linear combination, the elements of $\mathbf{d}_i$ are fixed, and naturally because the operations are over a *finite* field.

□

2.15.2 Proof of Lemma 5

We can first see that whenever $\omega(\mathbf{x}) \leq \rho n$, then (2.145) automatically holds simply because such $\mathbf{x}$ must belong in $\mathcal{B}(0,\rho)$ which in turn is a subset of $\mathcal{X}_{\mathcal{C}}$.

Let us now consider the case of $\omega(\mathbf{x}) > \rho n, \mathbf{x} \in \mathbb{F}^n$. Let us also recall the element selection process[21] that was described right underneath equation (2.144). Let $\mathcal{S} \subset \mathbb{F}^n$ be the set of all vectors $\mathbf{x}$ that are not codewords but are selected randomly in the aforementioned process. At this point, we can see that

$$\mathbb{P}(\mathbf{x} \in \mathcal{X}_{\mathcal{C}}) = \mathbb{P}(\mathbf{x} \in \mathcal{C}) + \mathbb{P}(\mathbf{x} \notin \mathcal{C})\mathbb{P}(\mathbf{x} \in \mathcal{S} \mid \mathbf{x} \notin \mathcal{C}) \tag{2.196}$$

where $\mathcal{C}$ is the code that has been chosen uniformly at random from $\mathcal{C}_{k^*,n}$. Consider a vector $\mathbf{y} \in \mathbb{F}^n$ with $\omega(\mathbf{y}) \geq \rho n$. We clearly see that $\mathbb{P}(\mathbf{x} \in \mathcal{C}) = \mathbb{P}(\mathbf{y} \in \mathcal{C})$, and we also see that $\mathbb{P}(\mathbf{x} \in \mathcal{S} \mid \mathbf{x} \notin \mathcal{C}) = \mathbb{P}(\mathbf{y} \in \mathcal{S} \mid \mathbf{y} \notin \mathcal{C})$ as a direct outcome of the aforementioned vector selection process, and of the fact that $\omega(\mathbf{x}) > \rho n, \omega(\mathbf{y}) > \rho n$, which yields

$$\mathbb{P}(\mathbf{x} \in \mathcal{X}_{\mathcal{C}}) = \mathbb{P}(\mathbf{y} \in \mathcal{X}_{\mathcal{C}}). \tag{2.197}$$

Now let us note that

$$\sum_{\mathbf{y} \in \mathbb{F}^n \backslash \mathcal{B}(0,\rho)} \mathbb{P}(\mathbf{y} \in \mathcal{X}_{\mathcal{C}}) = \sum_{\mathbf{y} \in \mathbb{F}^n \backslash \mathcal{B}(0,\rho)} \mathbb{E}[\mathbb{1}(\mathbf{y} \in \mathcal{X}_{\mathcal{C}})] \tag{2.198}$$

$$\overset{(a)}{=} \mathbb{E}[\sum_{\mathbf{y} \in \mathbb{F}^n \backslash \mathcal{B}(0,\rho)} \mathbb{1}(\mathbf{y} \in \mathcal{X}_{\mathcal{C}})] \tag{2.199}$$

[21] We recall that the procedure for designing $\mathcal{X}_{\mathcal{C}}$, simply starts by taking the union $\mathcal{C} \cup \mathcal{B}(\mathbf{0},\rho)$, and then appending on this union, a sufficiently large number of vectors, chosen uniformly and independently at random from $\mathbb{F}^n \backslash (\mathcal{C} \cup \mathcal{B}(0,\rho))$.

$$\stackrel{(b)}{=} \mathbb{E}[q^{n(1-\tau)} - V_q(n,\rho)] = q^{n(1-\tau)} - V_q(n,\rho) \quad (2.200)$$

where the average is over the codes in the ensemble $\mathcal{C}_{k^*,n}$ and over the randomness in constructing $\mathcal{X}_\mathcal{C}$ once $\mathcal{C} \in \mathcal{C}_{k^*,n}$ is picked. In the above, (a) follows by interchanging summation and expectation, and (b) holds since for every occurrence of $\mathcal{C} \in \mathcal{C}_{k^*,n}$, there exist $q^{n(1-\tau)} - V_q(n,\rho)$ elements of $\mathbb{F}\backslash\mathcal{B}(0,\rho)$ that are in $\mathcal{X}_\mathcal{C}$. Finally, (2.197) and (2.200) jointly imply that

$$(q^n - V_q(n,\rho))\mathbb{P}(\mathbf{x} \in \mathcal{X}_\mathcal{C}) \quad (2.201)$$
$$= \sum_{\mathbf{y}\in\mathbb{F}^n\backslash\mathcal{B}(0,\rho)} \mathbb{P}(\mathbf{y} \in \mathcal{X}_\mathcal{C}) = q^{n(1-\tau)} - V_q(n,\rho) \quad (2.202)$$

which completes the proof. □

2.15.3 Proof of Lemma 6

For any $i \neq 1$ and $\mathbf{x} \in \mathbb{F}^n, \mathbf{x} \neq \mathbf{0}$, then

$$\mathbb{P}[\mathbf{c}_i = \mathbf{x}|\mathbf{x} \in \mathcal{X}_\mathcal{C}]\mathbb{P}[\mathbf{x} \in \mathcal{X}_\mathcal{C}] \quad (2.203)$$
$$\stackrel{(a)}{=} \mathbb{P}[[\mathbf{c}_i = \mathbf{x}] \cap [\mathbf{x} \in \mathcal{X}_\mathcal{C}]] \quad (2.204)$$
$$\stackrel{(b)}{=} \mathbb{P}[[\mathbf{c}_i = \mathbf{x}] \ \cap \ [\mathbf{c}_i \in \mathcal{X}_\mathcal{C}]] \quad (2.205)$$
$$\stackrel{(c)}{=} \mathbb{P}[\mathbf{c}_i = \mathbf{x}] \quad (2.206)$$
$$\stackrel{(d)}{=} q^{-n} \quad (2.207)$$

where (a) is directly from the definition of conditional probability [95], (b) is true since the LHS requirement that $\mathbf{x} = \mathbf{c}_i$ is maintained in the RHS, (c) is true since $\mathcal{C} \subset \mathcal{X}_\mathcal{C}$, and (d) is from (2.141) in Lemma 4.

Thus for $i \neq 1$ and $\mathbf{x} \neq 0$, we have that

$$\mathbb{P}(\mathbf{c}_i = \mathbf{x}|\mathbf{x} \in \mathcal{X}_\mathcal{C}) = q^{-n(1-\tau)}\frac{1 - q^{-n}V(\rho,n)}{1 - q^{-(n-n\tau)}V(\rho,n)} \quad (2.208)$$
$$= q^{-(n-n\tau)}\zeta(n), \ i \in [2 : K^*], \ \omega(\mathbf{x}) > \rho n \quad (2.209)$$

and the proof is concluded by noting that in the limit of large n, the expression

$$\zeta(n) \triangleq \frac{1 - q^{-n}V(\rho,n)}{1 - q^{-(n-n\tau)}V(\rho,n)}$$

converges to 1. □

2.15.4 Proof of Lemma 7

From the definition in (2.148), let us recall that

$$\eta_{\mathbf{x}} \triangleq \sum_{i=1}^{q^{k^*}} \eta_{\mathbf{x},i} \tag{2.210}$$

describes the number of codewords that cover $\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}$. Consider a Hamming ball of radius ρ centered around $\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}$.

- Considering (2.146), we know that if $|\{0\} \cap \mathcal{B}(\mathbf{x}, \rho)| = 1$, then with probability one, $\mathbf{0}$ covers $\mathbf{x}$. Hence now the assumption that $\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}$, contradicts the above, and thus we can conclude that $|\{0\} \cap \mathcal{B}(\mathbf{x}, \rho)| = 0$.

- Now consider some vector $\mathbf{x}'$ in $(\mathcal{B}(0, \rho) \backslash \{0\}) \cap \mathcal{B}(\mathbf{x}, \rho)$, i.e., some vector that covers our aforementioned $\mathbf{x} \in \mathcal{X}'_{\mathcal{C}}$. We are interested in the probability $\mathbb{P}(\mathbf{c}_i = \mathbf{x}' | \mathbf{x}' \in \mathcal{X}'_{\mathcal{C}})$ which is the probability that $\mathbf{x}'$ is equal to $\mathbf{c}_i$, for our fixed $i, i \neq 1$. From (2.146), we know that this probability is equal to q^{-n}. Now going over all $q^{k^*} - 1$ codewords of interest, we can conclude that our current case of interest, contributes to the sum $\eta_{\mathbf{x}}$, by an amount equal to $(q^{k^*} - 1)[|(\mathcal{B}(0, \rho) \backslash \{0\}) \cap \mathcal{B}(\mathbf{x}, \rho)| q^{-n}$.

- Now let us consider the dominant case where $\mathbf{x}'$ in $\mathcal{B}(\mathbf{x}, \rho) \backslash \mathcal{B}(0, \rho)$. In this case, directly from (2.146), the aforementioned probability $\mathbb{P}(\mathbf{c}_i = \mathbf{x}' | \mathbf{x}' \in \mathcal{X}'_{\mathcal{C}})$ takes the form $q^{-n(1-\tau)} \zeta(n)$, and thus — similarly to above — yields a contribution to the sum $\eta_{\mathbf{x}}$ by an amount equal to

$$(q^{k^*} - 1)[|\mathcal{B}(\mathbf{x}, \rho) \backslash \mathcal{B}(0, \rho)| q^{-n(1-\tau)} \zeta(n).$$

□

2.15.5 Proof of Lemma 8

We prove the lemma in two steps.

In the first step, after defining $\overline{\eta} \triangleq \mathbb{E}(\eta_{\mathbf{x},i} | \mathbf{x} \in \mathcal{X}'_{\mathcal{C}})$ and $\overline{\eta^{(2)}} \triangleq \mathbb{E}(\eta_{\mathbf{x},i} \eta_{\mathbf{x},j} | \mathbf{x} \in \mathcal{X}'_{\mathcal{C}})$ for any $i, j \in [q^{k^*}]$, we can see that

$$Var(\eta_{\mathbf{x}} | \mathbf{x} \in \mathcal{X}_{\mathcal{C}}) \leq (q^{k^*} - 1)(q - 1)\overline{\eta}(1 - \frac{q-2}{q-1}\overline{\eta}) \tag{2.211}$$

since

$$Var(\eta_{\mathbf{x}} | \mathbf{x} \in \mathcal{X}_{\mathcal{C}}) = \mathbb{E}((\sum_{i=1}^{q^{k^*}} \eta_{\mathbf{x},i})^2 | \mathbf{x} \in \mathcal{X}'_{\mathcal{C}}) - \mathbb{E}^2(\sum_{i=1}^{q^{k^*}} \eta_{\mathbf{x},i} | \mathbf{x} \in \mathcal{X}'_{\mathcal{C}}) \tag{2.212}$$

$$= \mathbb{E}(\sum_{i=1}^{q^{k^*}} \eta_{\mathbf{x},i}^2 + \sum_{p.i=1,i\neq p}^{q^{k^*}} \eta_{\mathbf{x},i}\eta_{\mathbf{x},p} | \mathbf{x} \in \mathcal{X}_{\mathcal{C}}') \tag{2.213}$$

$$- \mathbb{E}^2(\sum_{i=1}^{q^{k^*}} \eta_{\mathbf{x},i} | \mathbf{x} \in \mathcal{X}_{\mathcal{C}}') \tag{2.214}$$

$$= \sum_{i=1}^{q^{k^*}} \mathbb{E}(\eta_{\mathbf{x},i} | \mathbf{x} \in \mathcal{X}_{\mathcal{C}}') \tag{2.215}$$

$$+ \sum_{p,i=1,i\neq p}^{q^{k^*}} \mathbb{E}(\eta_{\mathbf{x},i}\eta_{\mathbf{x},p} | \mathbf{x} \in \mathcal{X}_{\mathcal{C}}') - \mathbb{E}^2(\sum_{i=1}^{q^{k^*}} \eta_{\mathbf{x},i} | \mathbf{x} \in \mathcal{X}_{\mathcal{C}}') \tag{2.216}$$

$$= q^{k^*}\overline{\eta} + q^{k^*}(q^{k^*}-1)\overline{\eta^{(2)}} \tag{2.217}$$

$$- q^{2k^*}\overline{\eta}. \tag{2.218}$$

Combining now the above with (2.211), allows us to modify the main claim as

$$q^{k^*}\overline{\eta} + q^{k^*}(q^{k^*}-1)\overline{\eta^{(2)}} - q^{2k^*}\overline{\eta} - (q^{k^*}-1)(q-1)\overline{\eta}(1 - \frac{q-2}{q-1}\overline{\eta}) \leq 0 \tag{2.219}$$

so we now need to prove (2.219). To prove this, we focus on the LHS and show that

$$q^{k^*}\overline{\eta} + q^{k^*}(q^{k^*}-1)\overline{\eta^{(2)}} - q^{2k^*}\overline{\eta} - (q^{k^*}-1)(q-1)\overline{\eta}(1 - \frac{q-2}{q-1}\overline{\eta}) \tag{2.220}$$

$$\overset{(a)}{\leq} q^{k^*}\overline{\eta} + q^{k^*}(q^{k^*}-1)\overline{\eta}^2 - q^{2k^*}\overline{\eta} - (q^{k^*}-1)(q-1)\overline{\eta}(1 - \frac{q-2}{q-1})\overline{\eta} \tag{2.221}$$

$$\overset{(b)}{\leq} -(q^{k^*+1} - 2q^{k^*} - q + 1)\overline{\eta} + (q^{k^*+1} - 3q^{k^*} - q + 2)\overline{\eta}^2 \tag{2.222}$$

$$\overset{(c)}{\leq} 0 \tag{2.223}$$

where (a) holds because $\overline{\eta^{(2)}} \leq \overline{\eta}^2$, (b) follows after simple rearranging of terms, and (c) holds because $0 \leq \overline{\eta} \leq 1$ and because $q^{k^*+1} - 2q^{k^*} - q + 1 > q^{k^*+1} - 3q^{k^*} - q + 2$ for any $q^{k^*} \geq 1$.

In the second step, we will prove that

$$\frac{(q^{k^*}-1)(q-1)\overline{\eta}(1 - \frac{q-2}{q-1}\overline{\eta})}{\mathbb{E}(\eta_{\mathbf{x}} | \mathbf{x} \in \mathcal{X}_{\mathcal{C}}')q^2} \leq 1. \tag{2.224}$$

To see this, we note that

$$\frac{(q^{k^*}-1)(q-1)\overline{\eta}(1 - \frac{q-2}{q-1}\overline{\eta})}{\mathbb{E}(\eta_{\mathbf{x}} | \mathbf{x} \in \mathcal{X}_{\mathcal{C}}')q^2} \overset{(a)}{=} \frac{(q^{k^*}-1)(q-1)(1 - \frac{q-2}{q-1}\overline{\eta})}{q^{k^*}q^2} \tag{2.225}$$

$$\stackrel{(b)}{=} (\frac{q^{k^*}-1}{q^{k^*}})(\frac{q-1}{q^2})(1-\frac{q-2}{q-1}\overline{\eta}) \tag{2.226}$$

$$\stackrel{(c)}{\leq} 1 \tag{2.227}$$

where (a) holds because $\mathbb{E}(\eta_{\mathbf{x}}) = \sum_{i=1}^{q^{k^*}} \mathbb{E}(\eta_{\mathbf{x},i}) = q^{k^*}\overline{\eta}$, where (b) holds by rearranging terms, and where (c) holds because each multiplicative element in the RHS of (b) is non-negative and less than 1.

Now combining the two steps by bringing together (2.211) with (2.224), yields the desired (2.152). □

2.16 Proof of Lemma 9

Let us define

$$\mathcal{I}_{(\omega(\mathbf{x}),\rho)} \triangleq |\mathcal{B}(\mathbf{x},\rho) \cap \mathcal{B}(0,\rho)| \tag{2.228}$$

and let us note that

$$\rho n = \underset{\omega(\mathbf{x}):\mathbf{x}\in\mathcal{X}_{\mathcal{C}}\backslash\mathcal{B}(0,\rho)}{\arg\max} |\mathcal{I}_{(\omega(\mathbf{x}),\rho)}| \tag{2.229}$$

since the distance between $\mathbf{x} \in \mathcal{X}_{\mathcal{C}}\backslash\mathcal{B}(0,\rho)$ and $\mathbf{0}$ is minimized when $\omega(\mathbf{x}) = \rho n$. We also know that

$$|\mathcal{B}(\mathbf{x},\rho)\backslash\mathcal{B}(0,\rho)| = Vol(n,\rho) - |\mathcal{I}(\rho,\rho)|. \tag{2.230}$$

Let us focus on the case where $q = 2$ and $0 \leq \rho \leq \frac{1}{2}$, where from [96] we have that

$$|\mathcal{I}(\rho,\rho)| = \sum_{i=0}^{\lfloor\frac{n\rho}{2}\rfloor}\sum_{j=0}^{i}\binom{n\rho}{i}\binom{n-n\rho}{j} + \sum_{i=\lfloor\frac{n\rho}{2}\rfloor+1}^{n\rho}\sum_{j=0}^{n\rho-i}\binom{n\rho}{i}\binom{n-n\rho}{j}. \tag{2.231}$$

We know that $n\rho \leq n - n\rho$ and that

$$V_q(n,\rho) = \sum_{i=0}^{\rho n}\binom{n}{i} = \sum_{i=0}^{\lfloor\frac{n\rho}{2}\rfloor}\sum_{j=0}^{n\rho-i}\binom{n\rho}{i}\binom{n-n\rho}{j} \tag{2.232}$$

$$+ \sum_{i=\lfloor\frac{n\rho}{2}\rfloor+1}^{n\rho}\sum_{j=0}^{n\rho-i}\binom{n\rho}{i}\binom{n-n\rho}{j} \tag{2.233}$$

and thus after substituting (2.231) and (2.233) into (2.230), we conclude that

$$|\mathcal{B}(\mathbf{x},\rho)\backslash\mathcal{B}(0,\rho)| = \sum_{i=0}^{\lfloor\frac{n\rho}{2}\rfloor}\sum_{j=i+1}^{n\rho-i}\binom{n\rho}{i}\binom{n-n\rho}{j}. \tag{2.234}$$

Now considering that $0 < \rho \leq \frac{1}{2}$, we have

$$\binom{n}{n\rho} = \sum_{i=0}^{\lfloor \frac{n\rho}{2} \rfloor} \binom{n\rho}{i}\binom{n-n\rho}{n\rho-i} + \sum_{i=\lfloor \frac{n\rho}{2} \rfloor+1}^{n\rho} \binom{n\rho}{i}\binom{n-n\rho}{n\rho-i} \tag{2.235}$$

$$= \sum_{i=0}^{\lfloor \frac{n\rho}{2} \rfloor} \binom{n\rho}{i}\binom{n-n\rho}{n\rho-i} + \sum_{i=\lfloor \frac{n\rho}{2} \rfloor+1}^{n\rho} \binom{n\rho}{n\rho-i}\binom{n-n\rho}{n\rho-i} \tag{2.236}$$

$$= \sum_{i=0}^{\lfloor \frac{n\rho}{2} \rfloor} \binom{n\rho}{i}\binom{n-n\rho}{n\rho-i} + \sum_{i=0}^{n\rho-\lfloor \frac{n\rho}{2} \rfloor-1} \binom{n\rho}{i}\binom{n-n\rho}{i}. \tag{2.237}$$

Now let us note that

$$|\mathcal{B}(\mathbf{x},\rho)\backslash\mathcal{B}(0,\rho)| \overset{(a)}{=} \sum_{i=0}^{\lfloor \frac{n\rho}{2} \rfloor} \sum_{j=i+1}^{n\rho-i} \binom{n\rho}{i}\binom{n-n\rho}{j} \tag{2.238}$$

$$\overset{(b)}{=} \sum_{i=0}^{\lfloor \frac{n\rho}{2} \rfloor} \sum_{j=i+2}^{n\rho-i-1} \binom{n\rho}{i}\binom{n-n\rho}{j} \tag{2.239}$$

$$+ \sum_{i=0}^{\lfloor \frac{n\rho}{2} \rfloor} \binom{n\rho}{i}\binom{n-n\rho}{n\rho-i}(q-1)^{n\rho} \tag{2.240}$$

$$+ \sum_{i=0}^{\lfloor \frac{n\rho}{2} \rfloor - \mathbb{1}[\lfloor \frac{n\rho}{2} \rfloor = \frac{n\rho}{2}]} \binom{n\rho}{i}\binom{n-n\rho}{i+1} \tag{2.241}$$

$$\overset{(c)}{\geq} \sum_{i=0}^{\lfloor \frac{n\rho}{2} \rfloor} \binom{n\rho}{i}\binom{n-n\rho}{n\rho-i} + \sum_{i=0}^{n\rho-\lfloor \frac{n\rho}{2} \rfloor-1} \binom{n\rho}{i}\binom{n-n\rho}{i} \tag{2.242}$$

$$\overset{(d)}{=} \binom{n}{n\rho} \overset{(e)}{\geq} 2^{nH(\rho)-o(n)} \tag{2.243}$$

where (a) holds from (2.234), (b) follows after expanding the inner summation, (c) holds because the first summation of the RHS in (b) is non-negative, the second summation is present on the RHS of (c) after considering that $q = 2$, and because the third summation of the RHS in (b) is present on the RHS of (c) after considering Lemma 11 found in Appendix 2.16.4. Furthermore (d) follows from (2.237), and (e) follows from the Stirling inequality that applies because $0 < \rho \leq \frac{1}{2}$. The proof is concluded for the binary case of $q = 2$.

Let us now consider the more involved case of $q > 2$. For this we will need the following lemma, whose proof is found in Appendix 2.16.3.

Lemma 10. For $\mathcal{I}_{(\omega(\mathbf{x}),\rho)} \triangleq |\mathcal{B}(\mathbf{x},\rho) \cap \mathcal{B}(0,\rho)|$ (cf. (2.228)), then

$$|\mathcal{I}(\rho,\rho)| = \sum_{i+j=\rho n,\, i\leq j} \binom{n-\rho n}{i}\binom{\rho n}{j}(q-1)^{\rho n} + \theta(q) \tag{2.244}$$

where we can guarantee that $\theta(q) \leq V_q(n, \rho - 1/n)$.

Proof. See Appendix 2.16.3. □

With this lemma in place, we now note that

$$|\mathcal{B}(\mathbf{x}, \rho) \backslash \mathcal{B}(0, \rho)| \overset{(a)}{=} V_q(n, \rho) - |\mathcal{I}(\rho, \rho)| \tag{2.245}$$

$$\overset{(b)}{=} V_q(n, \rho - 1/n) + \binom{n - \rho n}{\rho n} \tag{2.246}$$

$$- \sum_{i+j=\rho n,\, i \leq j} \binom{n - \rho n}{i} \binom{\rho n}{j} (q-1)^{\rho n} - \theta(q) \tag{2.247}$$

$$\overset{(c)}{\geq} \sum_{i+j=\rho n,\, i > j} \binom{n - \rho n}{i} \binom{\rho n}{j} (q-1)^{\rho n} \tag{2.248}$$

$$\overset{(d)}{=} \sum_{j=\min\{0, 2\rho n - n\}}^{\lfloor \frac{\rho n}{2} \rfloor - 1} \binom{n - \rho n}{\rho n - j} \binom{\rho n}{j} (q-1)^{\rho n} \tag{2.249}$$

where (a) follows from a basic set cardinality rule, (b) follows from (2.244), (c) follows from Lemma (10) which tells us that $\theta(q) \leq V_q(n, \rho - 1/n)$, and from the fact that $\binom{n-\rho n}{\rho n} = \sum_{i+j=\rho n} \binom{n-\rho n}{i}\binom{\rho n}{j}(q-1)^{\rho n}$, and (d) holds since $j \geq 0$ and at the same time $i \leq n - \rho n$ which gives $j \geq 2\rho n - n$.

Now from Stirling's bound, we know that for any $j \in [\min\{0, 2\rho n - n\}, \lfloor \frac{\rho n}{2} \rfloor - 1]$, the following holds

$$\binom{n - \rho n}{\rho n - j} \binom{\rho n}{j} \geq \sqrt{\frac{n - \rho n}{8(\rho n - j)(n - 2\rho n + j)}} 2^{nH((\rho n - j)/(n - \rho n))} \tag{2.250}$$

$$\times \sqrt{\frac{\rho n}{8(j)(\rho n - j)}} 2^{nH(j/\rho n)} \tag{2.251}$$

$$= 2^{n[H((\rho n - j)/\rho n) + H((\rho n - j)/(n - \rho n))] - o(n)}. \tag{2.252}$$

After defining

$$\kappa \triangleq \frac{\rho n - j}{\rho n} \tag{2.253}$$

the exponent in (2.252) takes the form

$$n[H(\kappa) + H(\kappa \frac{\rho}{1 - \rho})] - o(n). \tag{2.254}$$

With this exponent in place, let us consider a large n and let us assume without loss of generality[22] that $n\rho^2 \in \mathbb{N}$. For the case where $0 < \rho \leq \frac{1}{2}$, let

[22]This assumption, along with the assumption that $n\rho$ is an integer, has no impact on the result, because any non-integer residual will vanish in importance as n increases.

us set j such that $j = n\rho^2 \leq \lfloor \frac{n\rho}{2} \rfloor - 1$, in which case we get $\kappa = 1 - \rho$. On the other hand, for the case where $\frac{1}{2} < \rho \leq -1/2 + \sqrt{5}/2$, let us set j such that $2\rho n - n \leq j = n\rho(1-\rho) \leq \lfloor \frac{n\rho}{2} \rfloor - 1$, in which case $\kappa = \rho$. In each case, κ is plugged in (2.254), and after utilizing (2.252), we see that

$$\sum_{j=\min\{0,2\rho n-n\}}^{\lfloor \frac{\rho n}{2} \rfloor - 1} \binom{n-\rho n}{\rho n - j} \binom{\rho n}{j} (q-1)^{\rho n} \geq q^{nH_q(\rho) - o(n)} \tag{2.255}$$

which holds for the range $0 < \rho \leq -1/2 + \sqrt{5}/2$ which is the union of the above two regions in ρ. □

2.16.1 Proof of Proposition 2

For $h(x) \triangleq -x \log_q(x)$, we know that

$$h(x) \leq H_q(x),\ 0 \leq x \leq 1 - 1/q \tag{2.256}$$

and thus that

$$H_q^{-1}(x) \leq h^{-1}(x). \tag{2.257}$$

We also know that if $y = x \ln(x)$, then $x = e^{W(y)}$ where $W(.)$ is the Lambert function. Also note that since $h(x) = -\log_q(e) x \ln(x)$, we have that

$$h^{-1}(x) = e^{W(-\ln(q)x)}. \tag{2.258}$$

Furthermore, for $c > 0$ being a positive real number, we have that

$$\lim_{T\to\infty} T e^{W(-c/T)} \overset{(a)}{=} \lim_{T\to\infty} \frac{e^{W(-c/T)}}{1/T} \tag{2.259}$$

$$\overset{(b)}{=} \lim_{T\to\infty} \frac{e^{W(-c/T)} \frac{1}{-c/T + e^{W(-c/T)}} cT^{-2}}{-T^{-2}} \tag{2.260}$$

$$\overset{(c)}{=} \lim_{T\to\infty} \frac{ce^{W(-c/T)}}{c/T - e^{W(-c/T)}} \tag{2.261}$$

$$\overset{(d)}{=} \lim_{T\to\infty} \frac{cTe^{W(-c/T)}}{c - Te^{W(-c/T)}} \tag{2.262}$$

$$\overset{(e)}{=} \frac{\lim_{T\to\infty} cTe^{W(-c/T)}}{\lim_{T\to\infty} c - Te^{W(-c/T)}} \tag{2.263}$$

where (a), (c) and (d) follow by basic algebraic rearranging, (b) follows from L'Hopital's rule, and (e) follows from the Algebraic Limit Theorem. The above implies that

$$\lim_{T\to\infty} T e^{W(-c/T)} = 0 \tag{2.264}$$

which allows us to conclude that

$$\lim_{T\to\infty} TH_q^{-1}((c/T) \overset{(a)}{\leq} \lim_{T\to\infty} Th^{-1}(c/T) \tag{2.265}$$

$$\overset{(b)}{\leq} \lim_{T\to\infty} Te^{W(-\ln(q)c/T)} \tag{2.266}$$

$$\overset{(c)}{=} 0 \tag{2.267}$$

where (a) follows from (2.257), (b) from (2.258), and (c) from (2.264). Thus the proof is concluded. □

2.16.2 Proof of Proposition 3

Starting from

$$c/T = H_q(f/T) \tag{2.268}$$

we take the derivative with respect to T on both sides, to get

$$c = \log_q(\frac{f/T}{1-f/T}(q-1))(\frac{\partial f}{\partial T}T - f). \tag{2.269}$$

After applying (2.268) into (2.269), and after some basic algebraic rearranging, the proof is concluded. □

2.16.3 Proof of Lemma 10

Let us consider two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}^n$, where $\omega(\mathbf{a}) = \rho n$ and where $\mathbf{b} \in \mathcal{B}(\mathbf{0}, \rho) \cap \mathcal{B}(\mathbf{a}, \rho)$. Let

$$\mathcal{B} \triangleq \{\mathbf{b}(i), \in [n] \ : \ \mathbf{a}(i) = 0 \ \& \ \mathbf{b}(i) \neq 0\} \tag{2.270}$$

$$\mathcal{C} \triangleq \{\mathbf{b}(i), \in [n] \ : \ \mathbf{a}(i) \neq 0 \ \& \ \mathbf{b}(i) = \mathbf{a}(i)\} \tag{2.271}$$

and let $x \triangleq |\mathcal{B}|, y \triangleq |\mathcal{C}|$. Let $b_j, j \in [x]$ denote the jth element of $\mathcal{B}$, and let $c_j, j \in [y]$ denote the jth element of $\mathcal{C}$. Note that ordering does not matter. Let $a_j, j \in [\rho n]$ be the non-zero elements of $\mathbf{a}$, and without loss of generality, let

$$\mathbf{a} = [0, \ldots, 0, a_1, a_2, \ldots, a_y, \ldots, a_{\rho n}] \tag{2.272}$$

as well as let

$$\mathbf{b} = [b_1, b_2, \ldots, b_x, 0, 0, \ldots, 0, c_1, c_2, \ldots, c_y, 0, \ldots, 0]. \tag{2.273}$$

Since $\omega(\mathbf{b}) \leq \rho n$ and $d(\mathbf{a}, \mathbf{b}) \leq \rho n$, we have that

$$x + y \leq \rho n \tag{2.274}$$
$$x \leq y \tag{2.275}$$

and we have that for any fixed pair $(x, y) \in [\rho n]^2$, there are $\binom{n-\rho n}{x}\binom{\rho n}{y}(q-1)^{x+y}$ such points (that satisfy that given (x, y)) in $\mathcal{B}(\mathbf{0}, \rho) \cap \mathcal{B}(\mathbf{a}, \rho)$. Therefore, accumulating over all (x, y) for which $x + y = \rho n$, the overall intersection gains cardinality $|\mathcal{B}(\mathbf{0}, \rho) \cap \mathcal{B}(\mathbf{a}, \rho)| = \sum_{i+j=\rho n,\, i \leq j} \binom{n-\rho n}{i}\binom{\rho n}{j}(q-1)^{\rho n}$, while when $x + y < \rho n$ — due to (2.275) — this same accumulated intersection has at most $V_q(n, \rho - 1/n)$ points. This concludes the proof.

□

2.16.4 Statement and Proof of Lemma 11

Lemma 11. If $0 < \rho \leq 1/2$ and $\rho n \in \mathbb{N}$, then

$$\sum_{i=0}^{\lfloor \frac{n\rho}{2} \rfloor - \mathbb{1}[\lfloor \frac{n\rho}{2} \rfloor = \frac{n\rho}{2}]} \binom{n\rho}{i}\binom{n-n\rho}{i+1} \geq \sum_{i=0}^{n\rho - \lfloor \frac{n\rho}{2} \rfloor - 1} \binom{n\rho}{i}\binom{n-n\rho}{i}. \tag{2.276}$$

Proof. We solve the problem by considering the following cases.

Case 1: ($2 \mid \rho n$). We note that if $2|\rho n$, then $\lfloor \frac{n\rho}{2} \rfloor = \frac{n\rho}{2}$, which gives

$$\sum_{i=0}^{\frac{n\rho}{2}-1} \binom{n\rho}{i}\binom{n-n\rho}{i+1} \geq \sum_{i=0}^{\frac{n\rho}{2}-1} \binom{n\rho}{i}\binom{n-n\rho}{i}. \tag{2.277}$$

Since $0 < \rho \leq \frac{1}{2}$, we have that $i + 1 \leq (n - n\rho)/2,\ \forall i \in [0 : n\rho/2 - 1]$, which gives

$$\binom{n-n\rho}{i+1} \geq \binom{n-n\rho}{i} \tag{2.278}$$

to conclude the proof for this case.

Case 2: ($2 \nmid \rho n$). Here we note that having $2 \nmid \rho n$, implies $\lfloor \frac{n\rho}{2} \rfloor = \frac{n\rho}{2} - 0.5$ which gives

$$\sum_{i=0}^{\frac{n\rho}{2}-0.5} \binom{n\rho}{i}\binom{n-n\rho}{i+1} \geq \sum_{i=0}^{\frac{n\rho}{2}-0.5} \binom{n\rho}{i}\binom{n-n\rho}{i}. \tag{2.279}$$

For this case, we consider the following two subcases.

Case 2a: (n is odd). We first note that if n is an odd number, then $n - n\rho$ is an even number. Also since $\rho n \in \mathbb{N}$, $0 < \rho < 1/2$ and $n\rho \leq \frac{n-1}{2}$,

then $2(\frac{n\rho}{2} + 0.5) \leq n - n\rho$. Thus as before we can say that $\forall i \in [0 : n\rho/2 - 0.5], i + 1 \leq (n - n\rho)/2$, which gives

$$\binom{n - n\rho}{i + 1} \geq \binom{n - n\rho}{i} \tag{2.280}$$

which in turn concludes the proof for this case.

Case 2b: (n is even). If n is even then $n - n\rho$ is odd, and thus we can again say that having $0 < \rho \leq 1/2$ gives $i + 1 \leq (n - n\rho + 1)/2,\ \forall i \in [0 : n\rho/2 - 0.5]$, which gives

$$\binom{n - n\rho}{i + 1} \geq \binom{n - n\rho}{i} \tag{2.281}$$

which in turn completes the proof for this final case also. □

Chapter 3

Perfect Codes for Multi-User Linearly-Decomposable

3.1 Introduction

Motivated by need to efficiently parallelize multiple computational tasks, which entails a master node that acts as a total trusted authority in managing N server nodes, serving K users that each demand their function to be computed. Under the linearly-decomposable assumption, where each function is a linear combination of L basis subfunctions, it was shown in Chapter 2 that the multi-user distributed computing problem is mathematically equivalent to a sparse matrix factorization problem of the form $\mathbf{F} = \mathbf{DE}$, where $\mathbf{F} \in \mathbf{GF}(q)^{K\times L}$ describes the linear coefficients of the demands of the users, where the so-called computation and encoding matrix $\mathbf{E} \in \mathbf{GF}(q)^{N\times L}$ describes which subfunctions each server evaluates and how each server combines the subfunctions' outputs, while the so-called communication and decoding matrix $\mathbf{D} \in \mathbf{GF}(q)^{K\times N}$ describes the server-to-user activated connectivity and the manner with which each user combines its received data.

This chapter studies the above scenario by jointly considering both the cumulative computational cost across the servers, as well as (under a uniformity assumption in the computational delay of evaluating each subfunction) the worst-case computational load which, as we will discuss later, captures the concept of computational delay. Our main contribution is to connect the above two metrics of our distributed computing problem, to the coding-theoretic concepts of the covering radius and the packing radius respectively. Then, in terms of the construction of schemes that efficiently resolve our distributed computing problem, our work provides a never-before-seen connection between distributed computing and the powerful structure of perfect

codes. Deviating from the approach of Chapter 2 which uses covering codes to reduce the computation cost in asymptotic settings, we here show how perfect codes — which optimize both the covering radius and packing density of codes — yield an improved solution to our distributed computing problem, both in terms of cumulative as well as worst-case costs, and do so for finite dimensions. To the best of our understanding, this is the first time that perfect codes (and the closely related quasi-perfect codes) have been associated with distributed computing and the equivalent problem of matrix factorization. The derived novel bounds on the cumulative computational cost as well as on the computational delay of a multi-user linearly-decomposable system capture the importance of the packing density as well as the packing and covering radius (defined in [97]) of a code whose parity-check matrix is our communication-and-computing matrix $\mathbf{D}$. Section 3.2 introduces the system model, Section 3.3 formulates the problem, Section 3.4 presents the main results, and finally, Section 3.5 concludes.

3.2 System Model

We consider the same system Model as of Chapter 2 and the same problem formulation. But the cost definitions are different and more realistic, as follows

3.2.1 Cumulative Computation Cost and Computational Delay

Recalling that $|\mathcal{W}_n|$ indicates the number of subfunctions that server n computes, we consider the *Cumulative Computation Cost* to take the form

$$\Gamma_c \triangleq \sum_{n=1}^{N} |\mathcal{W}_n| \tag{3.1}$$

representing the cumulative number of sub-function computations across all servers.

Furthermore, assuming that the jobs at each server are computed sequentially — and under a uniformity assumption that evaluating each file $W_l, l \in [L]$ from the subfunction $f_l(.)$ requires a normalized unit of time — we may consider a server's computational delay $T_n = |\mathcal{W}_n|$ and thus, under some basic synchronization assumptions, we may consider an overall computational delay that takes the form

$$\Lambda \triangleq \max_{n\in[N]} |\mathcal{W}_n|. \tag{3.2}$$

We wish to provide schemes that correctly compute the desired functions, with reduced costs Γ and Λ.

3.3 Problem Formulation: One-Shot Setting

The problem formulation is similar to Chapter 2's one-shot setting problem formulation, to reach the equation:

$$\mathbf{DE} = \mathbf{F}. \tag{3.3}$$

At this point, we note that $\mathcal{W}_n = \text{supp}(\mathbf{E}(n,:))$ and $|\mathcal{W}_n| = \omega(\mathbf{E}(n,:))$, which gives

$$\Lambda = \max_{n\in[N]} |\mathcal{W}_n| = \max_{n\in[N]} \omega(\mathbf{E}(n,:)) \tag{3.4}$$

revealing how the computational delay Λ of our system is captured by the maximum number of non-zero elements of any row of $\mathbf{E}$. This is one of the two sparsity constraints that interest us. The other sparsity constraint can be seen by recalling that $|\mathcal{W}_n| = \omega(\mathbf{E}(n,:))$ which gives

$$\Gamma = \sum_{n=1}^{N} |\mathcal{W}_n| = \omega(\mathbf{E}) \tag{3.5}$$

representing the total number of non-zero elements of $\mathbf{E}$, and which relates naturally to the cumulative computational cost across all servers. Thus, being able to decompose $\mathbf{F}$ as $\mathbf{F} = \mathbf{DE}$ where $\mathbf{E}$ corresponds to a reduced Λ and Γ, allows us indeed to reduce the two computation costs. To do this, we will turn to perfect codes, as well as to quasi-perfect codes.

Note that whenever we say the multi-user linearly-decomposable is implemented based on the decomposition $\mathbf{DE} = \mathbf{F}$, it means that based on the dimensionality K, N, we pick a code and choose its parity-check matrix as our $\mathbf{D}$, and then perform a syndrome decoding algorithm by regarding each column of $\mathbf{F}$ as a syndrome and each column of $\mathbf{E}$ as its corresponding error vector.

Before doing so, let us here provide a simple example to help clarify the setting and the notation.

3.3.1 Example

We consider the example of a system with a master node, $N = 11$ servers, $K = 5$ users, $L = 12$ subfunctions, and a field of size $q = 3$. In our example,

the jobs are defined (cf. (2.18)) by a demand matrix that here takes the form

$$\mathbf{F} = \begin{bmatrix} 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 & 2 & 2 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 & 1 & 0 & 2 & 1 & 2 & 0 & 1 & 1 \\ 0 & 2 & 0 & 2 & 0 & 1 & 2 & 1 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 1 & 2 & 2 & 0 & 1 & 1 & 1 & 2 \end{bmatrix}.$$

In the computation and encoding phase, the master allocates the computation of the different subfunctions $f_1(.), f_2(.), \ldots, f_{12}(.)$ across the 11 servers according to

$$\begin{aligned} &\mathcal{S}_1 = \{1, 8, 10\},\ \mathcal{S}_2 = \{2, 6\},\ \mathcal{S}_3 = \{5, 9\}, \\ &\mathcal{S}_4 = \{4\},\ \mathcal{S}_5 = \{7, 11\},\ \mathcal{S}_6 = \{1, 6, 12\},\ \mathcal{S}_7 = \{3\}, \\ &\mathcal{S}_8 = \{2\},\ \mathcal{S}_9 = \{3, 10\},\ \mathcal{S}_{10} = \{7, 12\},\ \mathcal{S}_{11} = \{5, 8\} \end{aligned}$$

forcing the two costs to be $\Gamma = \sum_{n=1}^{N} |\mathcal{W}_n| = 21$ and $\Lambda = \max_{n\in[N]} |\mathcal{W}_n| = 3$. After computing their designated output files, each server n transmits z_n corresponding to a computation and encoding matrix (cf. (2.22)) which here takes the value

$$\mathbf{E} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and which indeed abides by the constraint $\max_{n\in[N]} \omega(\mathbf{E}(n,:)) = \Lambda = 3$ (cf. (3.4)) and the constraint $\omega(\mathbf{E}) = \Gamma = 21$ (cf. (3.5)).

Subsequently, the master asks each server n to send its generated z_n to its designated receiving users, where for each server these user-sets are

$$\begin{aligned} &\mathcal{T}_1 = \{1, 2, \ldots, 5\},\ \mathcal{T}_2 = \{1, 2, \ldots, 4\}, \\ &\mathcal{T}_3 = \{1, 2, 3, 5\},\ \mathcal{T}_4 = \{1, 2, 4, 5\}, \\ &\mathcal{T}_5 = \{1, 3, 4, 5\},\ \mathcal{T}_6 = \{2, 3, 4, 5\},\ \mathcal{T}_7 = \{1\}, \mathcal{T}_8 = \{2\}, \end{aligned}$$

$$\mathcal{T}_9 = \{3\},\ \mathcal{T}_{10} = \{4\}, \mathcal{T}_{11} = \{5\}$$

which simply says that, for example, server 2 will broadcast z_2 to users $1, 2, 3, 4$. Subsequently, the decoding procedure is executed, adhering to a decoding matrix which takes the value

$$\mathbf{D} = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 0 & 2 & 0 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 1 & 2 & 0 & 0 & 1 & 0 & 0 \\ 1 & 2 & 0 & 1 & 2 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 2 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.6}$$

In the next section, we will show that the decoding matrix $\mathbf{D}$, is drawn from a class of perfect codes whose properties — as we show below — allow for an improved performance.

3.4 Main Results

We proceed with the main results of our work.

Theorem 6. *The optimal computational delay Λ of the (K, N) multi-user linearly decomposable problem implemented based on the decomposition $\mathbf{DE} = \mathbf{F}$, is bounded as*

$$\Lambda \leq \min\{L, \sum_{i=1}^{\tau} \binom{N-1}{i-1}(q-1)^i + (1-\mu_\tau)q^K\} \tag{3.7}$$

where τ and μ_τ are respectively the packing radius and the corresponding packing density of $\mathcal{C}_{\mathbf{D}}$.

Proof. Let us rewrite (3.3) as

$$\mathbf{DE}(:, l) = \mathbf{F}(:, l),\ \forall l \in [L] \tag{3.8}$$

and let us note that given a certain $\mathbf{D}$, we ask that for every column $\mathbf{F}(:, l) \in \mathbb{F}^K$, the resulting $\mathbf{E}(:, l)$ has a reduced number of non-zero elements. To achieve this, after regarding $\mathbf{D}$ to be a parity-check matrix of a linear code, we employ a syndrome decoder, rendering the 'error pattern' $\mathbf{E}(:, l)$ to be a coset leader associated to syndrome $\mathbf{F}(:, l)$. Now let's define the following set

$$\mathcal{S} \triangleq \{l : \omega(\mathbf{E}(:, l)) \leq \tau\} \tag{3.9}$$

which represents the syndromes $\mathbf{F}(:, l)$ for which the weight of their associated coset leader is below the packing radius τ. From the basic error-correcting

argument, we know that all the error patterns having no more than than τ non-zero elements are indeed present in $\mathcal{S}$, and thus we have that $|\mathcal{S}| \leq \sum_{i=1}^{\tau} \binom{N}{i}(q-1)^i$. Let us now focus on a row subvector $\mathbf{E}(n, \mathcal{S})$, and let us count the number of its non-zero elements. Since we know that for all $l \in \mathcal{S}$ it is indeed the case that $\omega(\mathbf{E}(:, l)) \leq \tau$, and since the collection of all such columns $\mathbf{E}(:, l)$ contains all possible error patterns of weight up to τ, we can conclude that

$$\omega(\mathbf{E}(n, \mathcal{S})) \leq \sum_{i=1}^{\tau} \binom{N-1}{i-1}(q-1)^i, \quad \forall n \in [N] \tag{3.10}$$

because i) each element $\mathbf{E}(n, l) \neq 0, l \in \mathcal{S}$ can take any one of $(q-1)$ possible values, because ii) such a column vector $\mathbf{E}(:, l)$ (where again $l \in \mathcal{S}$) has at most τ such non-zero elements, and because iii) there exist at most $\binom{N-1}{i-1}(q-1)^{i-1}$ vectors $\mathbf{E}(:, l) \neq 0, l \in \mathcal{S}$ whose nth entry $\mathbf{E}(n, l)$ is non-zero and whose total weight is $i \leq \tau$. Summing across all $i = 1, 2, ..., \tau$ yields the first term in our bound in (3.10). To conclude the proof, we proceed to count the number of syndromes (columns $\mathbf{F}(:, l)$ of $\mathbf{F}$) whose coset leaders (referring to the corresponding column $\mathbf{E}(:, l)$ of $\mathbf{E}$) have weight strictly bigger than τ. To do so, we first note that the number of points that are not covered by the ball $\mathcal{B}(\mathbf{c}, \tau)$, $\mathbf{c} \in \mathcal{C}_{\mathbf{D}}$, is $(1-\mu_\tau)q^N$. Recalling that each coset corresponds to q^{N-K} points (vectors) in $\mathbf{GF}(q)^N$, we can conclude that the number of cosets in question is $(1-\mu_\tau)q^K$. These are the cosets corresponding to syndromes with indices from $\mathcal{S}' \triangleq \{l : \omega(\mathbf{E}(:, l)) > \tau\}$. Thus, for the worst-case scenario of interest, we can consider that for all l for which $\omega(\mathbf{E}(:, l)) > \tau$, it is the case that we will encounter a non-zero $\mathbf{E}(n, l)$ for any $n \in [N]$, which is reflected in the addition of the second term $(1-\mu_\tau)q^K$ in our bound. Naturally, L is a trivial upper bound on Λ. This concludes the proof of the theorem. □

We now proceed with the second result, this time regarding the cumulative computation cost.

Theorem 7. *The optimal cumulative computation cost Γ_c of the (K, N) multi-user linearly decomposable problem implemented based on the decomposition $\mathbf{DE} = \mathbf{F}$, is bounded as*

$$\Gamma_c \leq \min\{NL, \sum_{i=1}^{\tau} \binom{N}{i}(q-1)^i i + (1-\mu_\tau)q^K \rho\} \tag{3.11}$$

where τ, ρ and μ_τ are respectively the packing radius, covering radius, and packing density of $\mathcal{C}_{\mathbf{D}}$.

Proof. The proof follows the steps of the proof of Theorem 6, up until the definition in (3.9), where similarly we now know that all the error patterns having no more than τ non-zero elements are present in the set of all possible $\mathbf{F}(:,l),\ l \in \mathcal{S}$, and thus we know that the sum of the Hamming weights of the coset leaders corresponding to the syndromes $\mathbf{F}(:,l),\ l \in \mathcal{S}$, takes the form $\sum_{i=1}^{\tau} \binom{N}{i} i(q-1)^i$ which matches the first term of our bound. Regarding the second term, we know that for any $l \notin \mathcal{S}$, the number of non-zero elements in $\mathbf{E}(:,l)$ is — by definition of the covering radius — no greater than ρ. We also see, following the ball arguments in the proof of the previous theorem, that the set $\mathcal{S}' \triangleq \{l : \omega(\mathbf{E}(:,l)) > \tau\}$ has size $|\mathcal{S}'| = (1-\mu_\tau)q^K$. Hence, we can now conclude that the sum of the weights of the coset leaders of the vectors (seen as syndromes) in $\mathcal{S}'$, is upper bounded by $\rho|\mathcal{S}'|$. □

3.4.1 The Connection to Perfect and Quasi-Perfect Codes

At this point, we note that the above bound on Γ_c is reduced when the covering radius ρ is reduced, while the bound on Λ is reduced when, for any given τ, the packing density μ_τ is increased. For this reason we are looking for codes (whose parity check matrix will be used as the decoding matrix $\mathbf{D}$) that indeed minimize ρ and increase μ_τ for a fixed τ. This special and rare property sought in $\mathcal{C}_{\mathbf{D}}$ is indeed attributed to the well-known class of perfect codes [97].

It is though known (cf. [98]) that there exist few such perfect codes, for a few select dimensions. Thus, for other dimensions, we will resort to the use of quasi-perfect codes which indeed ensure a similar performance, by guaranteeing optimal ρ and near-optimal μ_τ. We elaborate on this later on.

Remark 3. Looking back to our previous example corresponding to $N = 11, K = 5, L = 12$ and $q = 3$, the distributed computing solutions employed an $\mathbf{E}$ matrix that resulted from an optimal syndrome decoding process of a code whose parity check matrix $\mathbf{D}$ is indeed that of a ternary Golay code, which is a known perfect code.

3.4.2 The Special Case of Maximal Basis Set

We here consider also the case where $L = q^K$. This particular case of having a maximum number of basis subfunctions presents interesting advantages. In essence, under the same assumptions as before, we are now able to set the decoding matrix $\mathbf{D}$ once, well before the desired functions are declared. This allows us to have a fixed network connectivity, where — under the assumption

of knowing in advance K, N — we can fix $\mathbf{D}$, and we can then account for each $\mathbf{F}$ simply by altering $\mathbf{E}$. For this setting, we have the following propositions.

Proposition 4. *The optimal computational delay Λ and cumulative computation cost Γ_c of the (K, N) multi-user linearly decomposable problem with maximal basis, are lower bounded as*

$$\Lambda \geq \sum_{i=1}^{\tau} \binom{N-1}{i-1}(q-1)^i, \quad \Gamma_c \geq \sum_{i=1}^{\tau} \binom{N}{i}(q-1)^i i \tag{3.12}$$

where τ is the packing radius of $\mathcal{C}_{\mathbf{D}}$.

Proof. Regarding the bound on Λ, we follow all the steps of the proof in Theorem 6, with the only differences being firstly that the used inequality $|\mathcal{S}| \leq \sum_{i=1}^{\tau} \binom{N}{i}(q-1)^i$ is now automatically forced into an equality, and secondly that we consider, by choice, a lower bound $|\mathcal{S}'| \geq 0$. Exactly the same approach applies for the bound on Γ_c. □

Regarding optimality, we have the following proposition.

Proposition 5. *The optimal computational costs Λ and Γ_c for the cases (K, N) for which a perfect code exists, take the form*

$$\Lambda = \sum_{i=1}^{\tau} \binom{N-1}{i-1}(q-1)^i, \quad \Gamma_c = \sum_{i=1}^{\tau} \binom{N}{i}(q-1)^i i \tag{3.13}$$

where τ is the packing radius of the used perfect code $\mathcal{C}_{\mathbf{D}}$.

Proof. For the case where (K, N) accepts a perfect code, we know (cf. [97]) that $\mu_\tau = 1$ for which the upper bounds in (3.7) and (3.11) match the corresponding upper bounds in (3.12). □

We also have the following proposition for a much broader range of dimensionalities.

Proposition 6. *For all cases (K, N) for which there exists a quasi-perfect code $\mathcal{C}_{\mathbf{D}}$, the optimal performance is upper bounded as $\Lambda < \sum_{i=1}^{\tau} \binom{N-1}{i-1}(q-1)^i + \binom{N}{\tau+1}(q-1)^{\tau+1}$ and $\Gamma_c < \sum_{i=1}^{\tau+1} \binom{N}{i}(q-1)^i i$, where τ is the packing radius of the corresponding quasi-perfect code $\mathcal{C}_{\mathbf{D}}$.*

Proof. The proof is direct by noting that quasi-perfect codes have $\mu_\tau > 1 - \binom{N}{\tau+1}(q-1)^{\tau+1} q^{-K}$ and $\tau = \rho - 1$ [97]. □

Remark 4. Given the non-existence results in [98], it is in fact not difficult to show that quasi-perfect codes minimize the gaps between the upper bounds in (3.7) and (3.11) and the corresponding lower bounds in (3.12).

3.5 Conclusion

We have explored the computational cost of the multi-user distributed computing setting of linearly decomposable functions, which nicely captures various problems such as the distributed gradient coding problem [23], the distributed linear transform problem [26], the distributed matrix multiplication problem, and the distributed multivariate polynomial computation problems [33], [36], among others. The work established various upper and lower bounds on the computational delay Λ and the cumulative computation cost $\Gamma_c \in [0, NL]$, and revealed new connections to the packing and covering capabilities of codes thus revealing for the first time powerful connections with the class of perfect codes.

Chapter 4

Real-Valued Multi-User Linearly-Decomposable Distributed Computing

4.1 Introduction

Focusing on functions over finite fields, Chapter 2 proposed the *Multi-User Linearly-Decomposable Distributed-Computing framework*, which allows for distributed computation of functions that adhere to the very broad linearly separable format, which in turn captures various classes of linear and non-linear functions of practical interest[1]. Such functions have the form

$$F(.) = \sum_{l=1}^{L} f_l g_l(.)$$

$W_l = g_l(.)$ are the computed outputs of basis subfunctions $g_l(.)$, and f_l are scalar coefficients. In the multi-user (K users and N servers) setting where each user asks for its own function, the work in Chapter 2 and Chapter 3 transformed the distributed computing problem into a simple (preferably sparse) matrix factorization problem over finite fields, and then proceeded to make the direct connection between distributed computing, matrix factorization, and multiple coding-theoretic approaches. In particular, for a $K \times L$ *demand matrix* $\mathbf{F}$ where each row describes the coefficients that define the function requested by a user, the problem was transformed into the factorization problem $\mathbf{F} = \mathbf{DE}$, where $\mathbf{D}$ and $\mathbf{E}$ are the communication & decoding and computing & encoding matrices respectively. $\mathbf{E}$ dictated which server should compute which subfunction and then how each server should combine

[1]For more information on this, please see [99], [100].

the computed outputs before transmitting, while the $K \times N$ decoding matrix $\mathbf{D}$ dictated which user should each server communicate to and how each user should combine the various received signals. Then, a solution was proposed that derived from the powerful class of covering codes, and from a new class of so-called *partial covering codes*. In particular, the parity-check matrix from such codes played the role of $\mathbf{D}$, while then, after considering the columns of $\mathbf{F}$ as syndromes, the columns of $\mathbf{E}$ were identified as the coset leaders with minimum weight, thus guaranteeing the sparsest $\mathbf{E}$ and thus the least computational cost. For example, when $\mathbf{D}$ is derived from the basic class of covering codes over a q-ary field, then the corresponding normalized per subfunction computational cost $\gamma_c \in [0, 1]$ — describing the fraction of all subfunctions each server had to compute — took the form $\gamma_c = H_q^{-1}(K/N)$ where $H_q^{-1}(\cdot)$ is the functional inverse of the entropy function.

We are here though interested in computing functions directly over the reals, which will indeed constitute a substantial deviation from the finite field case. This emphasis on the real (or complex) domain is necessitated by the fact that computing a real-valued problem over a finite field (after discretization) may not be as practical as computing it directly over the reals, mainly because discretization may entail large precision costs and accuracy losses, as well as because finite field computations are notoriously slower than floating point operations. For that, we will consider real-valued functions over L real-valued subfunctions (or equivalently, with L component/basis subfunctions), and N computing servers and K users each demanding their own function. As we are now working in the field of real numbers, the coding theoretic approach in chapters 2 and 3 does not directly apply, and thus a new approach is required.

Here, our approach is based on establishing, for the first time, a connection between distributed computing and compressed sensing. As a first step, we show (Proposition 7) that there exists an achievable scheme whose normalized cumulative computational cost is bounded above as $\gamma_c \leq \frac{K}{N}$. This is a probabilistic scheme, where $\mathbf{D}$ is chosen from the Gaussian ensemble, and where the corresponding sparsity of $\mathbf{E}$ is the outcome of a randomized process. Then we propose ℓ_0-minimization, which takes as input $\mathbf{D}$ and $\mathbf{F}$ to yield a sparse $\mathbf{E}$. This minimization though is generally intractable, and for this reason, we draw from the rich literature of compressed sensing to suggest a more practical approach where we show (Theorem 8) that as long as there exists a scheme whose computational cost is bounded by $\gamma_c \leq -r\frac{K}{N}W_{-1}^{-1}(-\frac{2K}{eNr})$ (where $W_{-1}(\cdot)$ is the Lambert function and r is a parameter that calibrates the communication between servers and users) we can in fact employ a tractable basis pursuit ℓ_1-minimization to derive such scheme.

4.2 System Model and Problem Formulation

We consider the multi-user linearly-decomposable distributed computation setting, which consists of K users/clients, N active servers, and a master node that coordinates servers and users. The tasks performed on each server may entail substantial computational complexity as well as time constraints. We consider a setting where each server n can communicate in a single shot (a single time-slot) to some arbitrary user-set $\mathcal{T}_n \subset [K]$, via a dedicated broadcast channel.

In our setting, each user asks for a (generally non-linear) function from a space of linearly-decomposable functions and is of the form of a linear combination of individual subfunctions $g_l(.) \in \mathbb{R}$. Thus, the function $F_k(.) \in \mathbb{R}$, demanded by user $k \in [K]$, is a real-valued function of the form

$$F_k(.) \triangleq f_{k,1}g_1(.) \ldots + f_{k,L}g_L(.) \tag{4.1}$$

$$= f_{k,1}W_1 + \ldots + f_{k,L}W_L \tag{4.2}$$

where $W_l = g_l(.) \in \mathbb{R}$, $l \in [L]$ is a so-called 'file' output, and $f_{k,l} \in \mathbb{R}$, $k \in [K], l \in [L]$ are the linear combination coefficients that define each desired function.

4.2.1 Phases of the Process

The model involves three phases, with the first being the *demand phase*, the second being the *assignment and computation phase*, and the final one the *transmission and decoding phase*. In the demand phase, each user $k \in [K]$ requests $F_k(\cdot)$ from the master node, who then deduces the decomposition as in (4.2). Then, based on these K desired functions, during the assignment and computation phase, the master assigns some of the subfunctions to each server n, which then proceeds to compute these and produce the corresponding files $W_l = f_l(.)$ for all the subfunctions $f_l(.), l \in \mathcal{W}_n$ it is responsible for.

During the transmission phase, each server $n \in [N]$ broadcasts

$$z_n \triangleq \sum_{l \in [L]} e_{n,l} W_l, \;\; n \in [N] \tag{4.3}$$

in a single shot its own linear combination of the locally computed output files, and does so to its own particular subset of users $\mathcal{T}_n$. The above is defined by the so-called *encoding coefficients* $e_{n,l} \in \mathbb{R}$ which are determined by the master.

Finally, during the decoding phase, each user k linearly combines the received signals as follows

$$F'_k \triangleq \sum_{n \in [N]} d_{k,n} z_n \tag{4.4}$$

for some decoding coefficients $d_{k,n} \in \mathbb{R}, n \in [N]$, determined again by the master node. Naturally $d_{k,n} = 0, \forall k \notin \mathcal{T}_n$. In the end, we say the exact decoding is successful when $F'_k = F_k$ for all $k \in [K]$.

4.2.2 Problem Formulation

Similarly to the finite field case, also here, to formulate the problem we use $\mathbf{f} \triangleq [F_1, F_2, \ldots, F_K]^\intercal$, $\mathbf{f}_k \triangleq [f_{k,1}, f_{k,2}, \ldots, f_{k,L}]^\intercal$, $k \in [K], \mathbf{w} \triangleq [W_1, W_2, \ldots, W_L]^\intercal$ where $\mathbf{f}$ represents the vector of the demanded functions outputs (cf. (4.2)), $\mathbf{f}_k$ the vector of function coefficients for user k (cf. (4.2)), and $\mathbf{w}$ the vector of output files combined over all subfunctions. We also have $\mathbf{e}_n \triangleq [e_{n,1}, e_{n,2}, \ldots, e_{n,L}]^\intercal$, $n \in [N]\mathbf{z} \triangleq [z_1, z_2, \ldots, z_N]^\intercal$, respectively representing the encoding vector at server n, and the overall transmitted vector across all the servers (cf. (4.3)). Furthermore, we have $\mathbf{d}_k \triangleq [d_{k,1}, d_{k,2}, \ldots, d_{k,N}]^\intercal$, $k \in [K]\mathbf{f}' \triangleq [F'_1, F'_2, \ldots, F'_K]^\intercal$

respectively representing the decoding vector at user k, and the vector of the decoded functions across all the users. In addition, we have $\mathbf{F} \triangleq [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_K]^\intercal \in \mathbb{R}^{K \times L}, \mathbf{E} \triangleq [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N]^\intercal \in \mathbb{R}^{N \times L}, \mathbf{D} \triangleq [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_K]^\intercal \in \mathbb{R}^{K \times N}$

where $\mathbf{F}$ represents the $K \times L$ so-called *jobs matrix* of all function coefficients across all the users, where $\mathbf{E}$ represents the $N \times L$ *computing and encoding matrix* across all servers, and where $\mathbf{D}$ represents the $K \times N$ *communication and decoding matrix* across all the users.

Directly from (4.2), we have that

$$\mathbf{f} = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_K]^\intercal \mathbf{w} \tag{4.5}$$

and from (4.3) we have the overall transmitted vector taking the form

$$\mathbf{z} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N]^\intercal \mathbf{w} = \mathbf{E}\mathbf{w}. \tag{4.6}$$

Furthermore, directly from (4.4) we have that

$$F'_k = \mathbf{d}_k^T \mathbf{z} \tag{4.7}$$

and thus we have

$$\mathbf{f}' = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_K]^\intercal \mathbf{z} = \mathbf{D}\mathbf{z}. \tag{4.8}$$

Recall that we must guarantee

$$\mathbf{f}' = \mathbf{f}. \tag{4.9}$$

After substituting (4.5), (4.6) and (4.8) into (4.9), we see that the above feasibility condition in (4.9) is satisfied if and only if

$$\mathbf{DEw} = \mathbf{Fw}. \tag{4.10}$$

For this to hold for any $\mathbf{w}$, we must thus have

$$\mathbf{DE} = \mathbf{F}. \tag{4.11}$$

Note that throughout the whole paper, we assume that $K \leq N$.

4.2.3 Computational Cost

Recalling quickly that each server n computes the subfunctions whose index are in $\mathcal{W}_n$, and since $\mathcal{W}_n = \text{supp}(\mathbf{E}(n,:))$, then the *normalized cumulative computation cost* in our case naturally takes the form

$$\gamma_c \triangleq \frac{\sum_{n=1}^{N} |\mathcal{W}_n|}{NL} = \frac{\|\mathbf{E}\|_0}{NL}. \tag{4.12}$$

As one can see, γ_c simply describes the average fraction of subfunctions that must be computed by each server, which is also the fraction of non-zero elements in $\mathbf{E}$. It is now clear that decomposing $\mathbf{F}$ into the product of two matrices $\mathbf{D}$ and sparse $\mathbf{E}$, implies reduced cumulative computation cost which results in reduced delay. In particular, the fewer number of nonzero elements in a row of $\mathbf{E}$ means less delay in finishing up a task for a server.

4.3 Results

In this section, we first give a basic probabilistic scheme for subtask assignment, based on employing a Gaussian[2] random matrix $\mathbf{D}$, where the scheme employs a simple zero-forcing approach that solves a determined linear system. Albeit basic, this will allow us to upper bound the optimal normalized cumulative computation cost — which a generally intractable ℓ_0-minimization would give — as $\gamma_c \leq \frac{K}{N}$.

Proposition 7. *For the multi-user linearly-decomposable distributed computing problem, with K users, N servers and L subfunctions, employing a random Gaussian $\mathbf{D}$, guarantees that with probability 1, there exists a scheme with bounded normalized cumulative computation cost $\gamma_c \leq K/N$, which serves as an upper bound the ℓ_0-minimal cost.*

[2]This implies that each entry of $\mathbf{D}$ is independently and identically picked from a Gaussian distribution.

Proof. From (4.11), we have that $\mathbf{F}(:,l) = \mathbf{DE}(:,l), \mathbf{F}(:,l) \in \mathbb{R}^{K\times 1}$, $\forall l \in [L]$ where for each l, in the context of the ℓ_0-minimization in (4.15), we have $\mathbf{y} = \mathbf{F}(:,l)$, $\mathbf{D} = \mathbf{A}$ and $\mathbf{E}(:,l) = \mathbf{z}$, where again we have an underdetermined system of equations.

Consider choosing L arbitrary random subsets $\mathcal{S}_l \subset [N]$, $l \in [L]$ where $|\mathcal{S}_l| = K$. Now for each $l \in [L]$, we focus on the $l-th$ column $\mathbf{E}([N], l)$ of $\mathbf{E}$ and set the elements $\mathbf{E}([N]\backslash\mathcal{S}_l, l) = 0$, i.e., from column l, only the elements indexed by $\mathcal{S}_l$ remain non-zero. Now since $\mathbf{F}(:,l) = \mathbf{D}([K], [N]\backslash\mathcal{S}_l)\mathbf{E}([N]\backslash\mathcal{S}_l, l) + \mathbf{D}([K], \mathcal{S}_l)\mathbf{E}(\mathcal{S}_l, l)$, we get $\mathbf{F}(:,l) = \mathbf{D}([K], \mathcal{S}_l)\mathbf{E}(\mathcal{S}_l, l)$, which is a determined system of equations that allows us to determine $\mathbf{E}(\mathcal{S}_l, l)$. In the above, $\mathbf{D}([K], \mathcal{S}_l)$ is the corresponding $K \times K$ submatrix of $\mathbf{D}$. Given the above, each such $\mathbf{D}([K], \mathcal{S}_l)$ is a $K \times K$ Gaussian sub-matrix, which is naturally nonsingular with probability one. Thus, the determined system of equations always has a unique solution for $\mathbf{E}(\mathcal{S}_l, l), \forall l \in [L]$, therefore the scheme works for all $\mathbf{F}(:,l)$, $l \in [L]$. This scheme guarantees that $\|\mathbf{E}(:,l)\|_0 \leq K$, then $\|\mathbf{E}\|_0 \leq KL$, and thus guarantees that $\gamma_c \leq \frac{K}{N}$. Better performance can be achieved by employing ℓ_0-minimization as in (4.15). □

As we will discuss in Section 4.4, ℓ_0-minimization is known to be NP-hard [101], hence intractable. To offer a practical solution, the following theorem utilizes results from compressed sensing to describe a range of practical solutions, which now use ℓ_1-minimisation in order to find — as we will clarify later on — sparse (and unique) computing and encoding matrices $\mathbf{E}$.

Theorem 8. *For the multi-user linearly-decomposable distributed computing problem, with K users, N servers and L subfunctions, if a scheme exists with a (κ, β) sub-Gaussian random matrix $\mathbf{D}$ (cf. Lemma 13) for which ℓ_0-minimisation would yield*

$$\gamma_c \leq -\frac{1}{r}\frac{K}{N}W_{-1}^{-1}(-\frac{2K}{erN}), \quad 0 \leq K/N \leq 12(2\beta+\kappa)/\kappa^2, \tag{4.13}$$

then the corresponding (and unique) $\mathbf{E}$ can be found via basis pursuit ℓ_1-minimization with probability at least $1 - 2e^{-\frac{KL}{r}}$, where $r = 12(4\beta + 2\kappa)/\kappa^2$.

Proof. The proof is provided in the following section, which starts with a brief primer on compressed sensing. □

4.4 Proof of Theorem 8

Before proceeding with the proof, we quickly describe some basic properties of compressed sensing.

4.4.1 Brief Primer on Compressed sensing

We provide here a brief introduction of the compressed sensing results [101]–[108], which will be employed in our distributed computing problem. We will utilize notation common to the compressed sensing literature, and the link to the computing parameters will be clarified in the next subsection.

As described in [105], compressed sensing seeks to recover a sparse vector $\mathbf{x} \in \mathbb{R}^p$ from a few underdetermined linear measurements of the form:

$$\mathbf{y} = \mathbf{A}\mathbf{x} \in \mathbb{R}^m \tag{4.14}$$

where $\mathbf{A} \in \mathbb{R}^{m \times p}, m, p \in \mathbb{N}$ is the so-called measurement matrix, and $\mathbf{y} = [y_1, ..., y_m]^\intercal$ is the measurement vector. In our case, as we will see later on, $\mathbf{y}$ will be associated to our computing and encoding matrix $\mathbf{E}$, then $\mathbf{A}$ to the communication and decoding matrix $\mathbf{D}$, and $\mathbf{x}$ will be associated to the jobs matrix $\mathbf{F}$. The general approach is to recover the sparsest solution via a basic but computationally intractable ℓ_0-minimization that takes the form

$$\min_{\mathbf{z} \in \mathbb{R}^p} \|\mathbf{z}\|_0 := \sum_{i=1}^{p} 1_{|z_i| \neq 0} \text{ subject to } \mathbf{y} = \mathbf{A}\mathbf{z} \tag{4.15}$$

where $1_{|z_i| \neq 0}$ denotes the indicator function. This same optimization will lead to the sparsest solution for $\mathbf{E}$ and thus will yield the smallest possible γ_c. To the best of our knowledge, there are no results that enables us to bound the weight of this sparsest solution, and for that we will use a basic constructive approach to bound γ_c.

The NP-hard nature of the optimization problem in (4.15) has led to the consideration of an ℓ_1-norm minimization approach, also known as *basis pursuit*, which is considered as the closest convex tractable alternative for (4.15), and which is given by

$$\min_{\mathbf{z} \in \mathbb{R}^p} \|\mathbf{z}\|_1 := \sum_{i=1}^{p} |z_i| \tag{4.16}$$

$$\text{s.t. } \mathbf{y} = \mathbf{A}\mathbf{z}. \tag{4.17}$$

It is well established in compressed sensing that the estimate $\widehat{\mathbf{x}} \in \mathbb{R}^p$ obtained by solving (4.16), achieves the desired unique solution $\mathbf{x}$ as long as some conditions are satisfied[3]. These conditions are closely related to certain properties of the measurement matrix, with one such condition being the well known restricted isometry property (RIP) [93], which dictates how well

[3]For our computing problem. these will be conditions in the form of an upper bound on γ_c.

ℓ_1-norm optimization algorithms, such as basis pursuit [109], can perform. This is captured in the following result, found in [101, Theorem 6.2], which is reproduced here.

Lemma 12. For a matrix $\mathbf{A} \in \mathbb{R}^{m\times p}$ and for

$$\delta_s(\mathbf{A}) \triangleq \max_{\mathcal{S}\subset[p],|\mathcal{S}|\leq s} \|\mathbf{A}_{\mathcal{S}}^{*}\mathbf{A}_{\mathcal{S}} - \mathbf{I}_{m\times m}\|_{2\to 2}^{2} \tag{4.18}$$

being the sth restricted isometry constant, and if $\delta_{2s}(\mathbf{A}) < \frac{1}{3}$, then every s-sparse vector $\mathbf{x} \in \mathbb{R}^p$ is the unique solution of

$$\underset{\mathbf{z}\in\mathbb{R}^p}{\text{minimize}} \|\mathbf{z}\|_1 \quad \text{subject to} \quad \mathbf{A}\mathbf{z} = \mathbf{A}\mathbf{x}. \tag{4.19}$$

In particular, the above Lemma shows that having a measurement vector $\mathbf{y} \in \mathbb{R}^m$, where we know apriori that it is a result of a linear system $\mathbf{A}\mathbf{x}$, $\mathbf{x} \in \mathbb{R}^p, m \leq p$, induced by a unique and s-sparse vector $\mathbf{x}$ if $\delta_{2s}(\mathbf{A}) < \frac{1}{3}$, then via having a ℓ_1-minimizer, we can find $\mathbf{z}$ as a solution to $\mathbf{A}\mathbf{z}$, which it has minimum $\|\mathbf{z}\|_1$, then the above Lemma guarantees that the solution of this minimization is equal to $\mathbf{x}$. The above lemma shows that having $\delta_{2s}(\mathbf{A}) < 1/3$ is sufficient to guarantee the exact recovery of all unique s-sparse vectors via ℓ_1-minimization. It basically states that if $\mathbf{A}$ behaves relatively similar to orthonormal matrices when operating on sparse vectors, then ℓ_1-minimization will act as an ℓ_0-minimization.

In our problem here, it is important that we pick $\mathbf{D}$ such that $\mathbf{A}$ abides by the above property. The following two results tell us directly how to do that. The first, below, is directly adapted from [101, Theorem 9.2].

Lemma 13. Let $\mathbf{A} \in \mathbb{R}^{m\times p}$ be an i.i.d. (zero mean, unit variance) sub-Gaussian random matrix with parameters β and κ such that

$$\mathbb{P}(|A_{i,j}| \geq t) \leq \beta \mathrm{e}^{-\kappa t^2}, \forall t > 0. \tag{4.20}$$

Then, $\delta_{2s}(\frac{\mathbf{A}}{\sqrt{m}}) \leq \delta$ is satisfied with probability at least $1 - 2\mathrm{e}^{-\frac{\delta^2 m}{2c}}$ for any δ such that

$$m \geq 2c\delta^{-2} s \ln(\frac{ep}{2s}) \tag{4.21}$$

where

$$c = \frac{2(4\beta + 2k)}{3k^2}. \tag{4.22}$$

Combining Lemma 13 and Lemma 12 after setting $\delta = 1/3$, implies that the uniform recovery of all s-sparse vectors is possible with high probability via the ℓ_1-minimization in (4.16) as long as the number of measurements satisfies $m \geq (12(4\beta + 2\kappa)/\kappa^2) s \ln(\frac{ep}{2s})$.

4.4.2 The Exact Description of Proof of Theorem 8

Directly from (4.11), we have

$$\text{vec}(\mathbf{F}) = (\mathbf{D} \otimes \mathbf{I}_{L\times L}) \times \text{vec}(\mathbf{E}), \tag{4.23}$$

which matches the compressed sensing setting $\mathbf{y} = \mathbf{A}\mathbf{x}$ in (4.14) when considering $\mathbf{y} = \text{vec}(\mathbf{F})$, $\mathbf{A} = \mathbf{D} \otimes I_{L\times L}$, and $\mathbf{x} = \text{vec}(\mathbf{E})$, where now $m = KL$ and $p = NL$.

Furthermore, let us also note that directly from [110], we have

$$\delta_s(\mathbf{D} \otimes \mathbf{I}_{L\times L}) \leq \delta_s(\mathbf{D}). \tag{4.24}$$

With the elements of $\mathbf{D}$ being chosen independently from a zero-mean, unit-variance sub-Gaussian distribution with parameters β, κ (cf. (4.20)), we can now employ Lemma 12 and (4.24), together with Lemma 13 after setting $\delta_{2s} < \delta = 1/3$, to conclude that the exact recovery threshold for a unique $\mathbf{E}$ matrix via ℓ_1-minimization driven by basis pursuit, takes the form

$$KL \geq r \left\|\mathbf{E}\right\|_0 \ln(\frac{eNL}{2r \left\|\mathbf{E}\right\|_0}) \tag{4.25}$$

where $r = 12(4\beta + 2\kappa)/\kappa^2$. Then after normalizing both sides by NL and applying (4.12), we get

$$\frac{K}{N} \geq r\gamma_c \ln(\frac{e}{2r\gamma_c}). \tag{4.26}$$

Let us now define $f(x) \triangleq -r^{-1}xW_{-1}^{-1}(-\frac{2x}{e})$ and evaluate it on the both sides of (4.26), at $x_1 = K/N$ and $x_2 = r\gamma_c \ln(\frac{e}{2r\gamma_c})$ since $f(x)$ is a monotonically increasing function on $0 \leq x \leq r/2$ and $r/2 \geq x_1 \geq x_2 \geq 0$, we have $f(x_1) \geq f(x_2)$, in the view of the fact that its inverse[4] function is $f^{-1}(x) = rx \ln(\frac{e}{2rx})$, we can retrieve the claim.

4.5 Discussion and Conclusion

In the context of our distributed computing problem, it is interesting to observe some of the similarities that exist between the real case (which employed compressed sensing techniques) and the finite-field case in chapter 2,

[4]To see this, for $g(x) \triangleq rx \ln(\frac{e}{2rx})$ and $f(x) \triangleq -r^{-1}xW_{-1}^{-1}(-2x/e)$, we see that $g^{-1}(x) = f(x)$ simply because $f(g(x)) = -r^{-1}g(x)W_{-1}^{-1}(-\frac{2g(x)}{e}) = -x\ln(\frac{e}{2rx})W_{-1}^{-1} = -x\ln(e/2rx)W_{-1}^{-1}(\frac{2rx}{e}\ln(2rx/e)) = -x\ln(e/2rx)/\ln(2rx/e) = x$.

which employed the structure of covering codes whose covering radius was a measure of the sparsity of the solution for $\mathbf{E}$. For example, in the extreme case of $L = q^K$, the work in chapter 2 revealed[5] the optimal normalized computational cost to be of the form of $\gamma_c \simeq H_q^{-1}(K/N)$, which almost matches $\gamma_c \simeq K/N$ in the limit of large q, derived in this paper.

It is also worth noting that our result in Theorem 8 automatically accepts an additional uniqueness property — on the sparsest solution $\mathbf{x}$ in (4.14) — which is in fact not needed in our distributed computing problem. It would be interesting to explore further improvements in the computational costs, upon the removal of this uniqueness condition. Finally, one can imagine that further improvements in the distributed computing problem could also benefit from the deep connections revealed in [105] between compressed sensing and error correction.

[5]Over a q-ary alphabet, $H_q(x)$ denotes the entropy function, which takes the form $H_q(x) \triangleq x \log_q(q-1) - x \log_q(x) - (1-x) \log_q(1-x)$ for all $0 < x < 1 - 1/q$.

Chapter 5

Lossless Tessellated Distributed Computing

5.1 introduction

As signified in the previous chapters, the celebrated computation-vs-communication relationship stands at the very core of distributed computing as a fundamental principle with profound ramifications. This principle appears as a limiting factor in a variety of distributed computing scenarios [8], [21], [45], [46], [75], [78]–[84], [111] where indeed communication and computation are often the two intertwined bottlenecks that most heavily define the overall performance.

Another important factor pertains to computational accuracy and the ability to recover desired functions with reduced error or distortion. There is indeed a variety of techniques dedicated to increasing accuracy (cf. [56]–[58], [112]–[126]), such as for example the *sketching technique* [113], [116], [126] which utilized a randomized linear algebraic approach to compute an approximation of the multiplication of two massive matrices (often by approximating input matrices by multiplying them with a random matrix having certain properties), as well as successive approximation coding techniques (cf. [121]) which can tradeoff accuracy and speed, allowing for better approximations and increased accuracy over time.

This triptych between accuracy, communication costs and computation costs, lies at the center of distributed computing. We here explore this triptych for a pertinent setting of multi-user distributed computing.

5.1.1 Multi-User Linearly-Decomposable Distributed Computing

We focus on the very broad and arguably practical setting of *multi-user distributed computation of linearly-decomposable real functions*, which captures several classes of computing problems that include distributed gradient coding problems [23]–[25], [38], the distributed linear-transform computation problem [26], [45], the distributed matrix multiplication or the distributed multivariate polynomial computation problems [18], [27]–[29], [31]–[33], [35], [36], [46], as well as the distributed computing problem of training large-scale machine learning algorithms and deep neural networks with massive data [8]. These constitute a broad collection of problems where both computation and communication costs are crucial [43], [44].

Our setting, as depicted in Fig. 5.1, initially considers a *master node* that coordinates, in three phases, a set of N distributed *servers* that compute functions requested by the K *users*. During the initial *demand* phase, each user $k \in \{1, 2, \ldots, K\}$ independently requests the computed output of a single real function $F_k(.)$. Under the real-valued[1] linear decomposability assumption[2], these functions take the basic form

$$F_k(.) = \sum_{\ell=1}^{L} f_{k,\ell} f_\ell(.) = \sum_{\ell=1}^{L} f_{k,\ell} W_\ell \tag{5.1}$$

where $f_\ell(\cdot)$ denotes a *(basis or component) subfunction*, where $f_{k,\ell}$ denotes a real-valued combining coefficient, and where $W_\ell = f_\ell(x), x \in \mathcal{D}$, denotes the real-valued *output file* of $f_\ell(\cdot)$ for an input x from any domain set $\mathcal{D}$.

Subsequently, during the *computing phase*, the master assigns to each server $n \in [N]$, a set of subfunctions $\mathcal{S}_n \subseteq [L]$ to compute locally[3] in order to generate the corresponding W_ℓ, and then during the *communication phase*, server n forms signals

$$z_{n,t} \triangleq \sum_{\ell \in [L]} e_{n,\ell,t} W_\ell, \;\; n \in [N], \; t \in [T] \tag{5.2}$$

[1]The real-valued exposition entails a variety of advantages over finite-field approaches [22], [42], [89], [127], such as accuracy advantages stemming from using real-valued fixed point data representations, as well as advantages regarding computation overflows, quantization errors and scalability barriers [48]–[50].

[2]This naturally incorporates linearly separable functions (see for example [22]) where each $F_k(.)$, taking L subfunctions as input, can be written as a linear combination of L *univariate* subfunctions. In our work, these subfunctions need not be univariate.

[3]We will interchangeably use $\mathcal{S}_n$ to describe sets of indices (of subfunctions), as well as the subfunctions themselves.

as dictated by the *encoding coefficients* $e_{n,\ell,t} \in \mathbb{R}, n \in [N], t \in [T], \ell \in [L]$, and proceeds to transmit $z_{n,t}$ during time-slot $t = 1, 2, \ldots, T$ to a subset of users $\mathcal{T}_{n,t} \subseteq [K]$, via a dedicated error-free broadcast channel. Finally, during the decoding part of the last phase, each user k linearly combines its received signals to get

$$F'_k \triangleq \sum_{n\in[N],t\in[T]} d_{k,n,t} z_{n,t} \tag{5.3}$$

as dictated by the *decoding coefficients* $d_{k,n,t} \in \mathbb{R}, n \in [N], t \in [T], k \in [K]$. Naturally $d_{k,n,t} = 0, \forall k \notin \mathcal{T}_{n,t}$, simply because user $k \in [K]$ does not receive any symbol from server $n \in [N]$ during time $t \in [T]$. Note that both encoding and decoding coefficients are determined by the master node after the demand phase, and thus are dependent on the functions requested, but remain independent of the instance of the *input* to the requested functions.

In case of error, we consider

$$\mathcal{E} = \sum_{k=1}^{K} |F'_k - F_k|^2, \ \mathcal{E} \in \mathbb{R}, \quad \forall k \in [K] \tag{5.4}$$

to be the Euclidean distortion during function retrieval. For $\mathcal{T}_n = \cup_{t=1}^{T} \mathcal{T}_{n,t}$, we consider the computation and communication costs

$$\Gamma \triangleq \max_{n\in[N]} |\mathcal{S}_n|, \quad \Delta \triangleq \max_{n\in[N]} |\mathcal{T}_n| \tag{5.5}$$

respectively representing the maximum number of subfunctions to be locally computed at any server[4], and the number of users that a server can communicate to. After normalization, we here consider the normalized costs

$$\epsilon \triangleq \frac{\mathcal{E}}{KL}, \quad \gamma \triangleq \frac{\Gamma}{L}, \quad \delta \triangleq \frac{\Delta}{K}. \tag{5.6}$$

The three parameters are bounded[5] between 0 and 1.

[4]Our focus directly on the cost of computing the component subfunctions $f_\ell(\cdot)$ stems from the point of view that these functions typically capture computationally intensive (and generally non-linear) tasks which would dominate, in terms of load, the remaining easier linear manipulations at the servers and users during encoding and decoding.

[5]In brief, γ is the fraction of subfunctions that must be computed locally, and δ is the fraction of available links to be activated. Having $\gamma = 1$ corresponds to the centralized scenario of having to locally calculate all subfunctions, while $\delta = 1$ matches an extreme parallelized scenario that activates all available communication links. We will discuss later in Section 6.1 the justification of having $\epsilon \leq 1$.

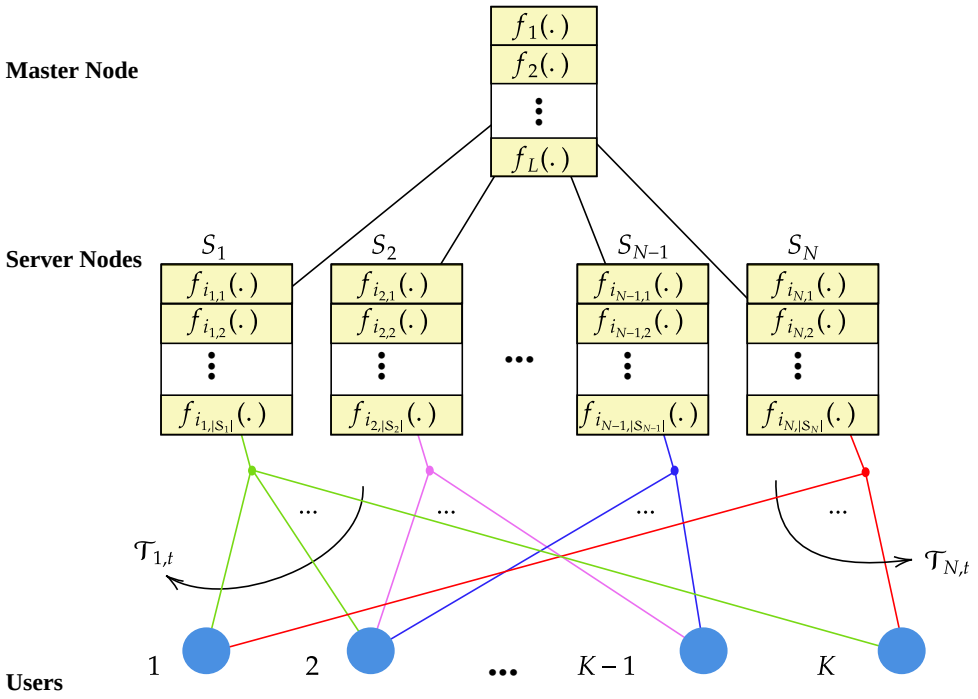


Figure 5.1: The K-user, N-server, T-shot setting. Each server n computes the subfunctions in $\mathcal{S}_n = \{f_{i_{n,1}}(.), f_{i_{n,2}}(.), \ldots, f_{i_{n,|\mathcal{S}_n|}}(.)\}$ and communicates to users in $\mathcal{T}_{n,t}$, under computational constraint $|\mathcal{S}_n| \leq \Gamma \leq L$ and communication constraint $|\mathcal{T}_n| \leq \Delta \leq K$, yielding a system with normalized constraints $\gamma = \frac{\Gamma}{L}, \delta = \frac{\Delta}{K}$ and with an error constraint $\epsilon = \frac{\mathcal{E}}{KL}$, where $\gamma, \delta, \epsilon \in [0, 1]$.

Another two parameters of interest are

$$\zeta \triangleq \frac{\Delta}{L}, \quad \kappa \triangleq \frac{K}{L} \tag{5.7}$$

where ζ normalizes the number of activated communication links by the number of subfunctions, while κ reflects the dimensionality ('fatness') of matrix $\mathbf{F}$ and thus will define the statistical behavior of the singular values of $\mathbf{F}$ which will be crucial in the lossy case where we allow for function reconstruction error.

In a system defined by K, N and L, our goal is to find schemes that can recover any set of desired functions, with the best possible decoding refinement ϵ, and the smallest possible computation and communication loads γ, δ. To do so, we must carefully decide which subfunctions each server computes, and which combinations of computed outputs each server sends to which users. Having to serve many users with fewer servers naturally places a burden on the system (suggesting higher γ, δ, ϵ), bringing to the fore the concept of the *system rate*

$$R \triangleq \frac{K}{N} \tag{5.8}$$

and the corresponding *system capacity* C representing the supremum of all rates.

5.1.2 Connection to Sparse Matrix Factorization, and Related Works

Toward analysing our distributed computing problem, we can see from (2.1) that the desired functions are fully represented by a matrix $\mathbf{F} \in \mathbb{R}^{K \times L}$ of the aforementioned coefficients $f_{k,\ell}$. With $\mathbf{F}$ in place, we must decide on the computation-assignment and communication (encoding and decoding) protocol. As we have seen in [127], [128], for the error-free case of $\epsilon = 0$, this task is equivalent — directly from (5.2),(5.3) and (5.4) — to solving a (sparse) matrix factorization problem of the form

$$\mathbf{DE} = \mathbf{F} \tag{5.9}$$

where, as we will specify later on, the $NT \times L$ *computing matrix* $\mathbf{E}$ holds the coefficients $e_{n,\ell,t}$ from (5.2), while the $K \times NT$ *communication matrix* $\mathbf{D}$ holds the decoding coefficients $d_{k,n,t}$ from (5.3). As one can suspect, a sparser $\mathbf{E}$ reflects a lower γ and a sparser $\mathbf{D}$ a lower δ, at the cost though of a potentially higher ϵ. Focusing on functions over finite fields, the work in [127],

after making the connection between distributed computing and the above factorization problem, employed from coding theory the class of covering codes and a new class of *partial covering codes*, in order to derive bounds on the optimal communication and computation costs for the *error-free case*. In brief, by choosing $\mathbf{D}$ to be the sparse parity-check matrix of a (shown to exist) sparse partial covering code, each column of $\mathbf{E}$ was subsequently produced to be the coset leader from syndrome decoding (with the syndrome being) the corresponding column[6] of $\mathbf{F}$. This allowed for reduced communication and computation costs, where for example in the single shot scenario with a q-ary finite field, the (somewhat different from here) normalized optimal computation cost $\gamma \in (0, 1]$ was bounded as a function of the q-ary entropy function H_q to be in the range $\gamma \in [H_q^{-1}(\frac{\log_q(L)}{N}), H_q^{-1}(\frac{K}{N})]$.

A first exposition of the *real-valued* variant of our computing problem, again for the error-free case of $\epsilon = 0$, can be found in [129], which reformulated the equivalent sparse matrix factorization problem $\mathbf{DE} = \mathbf{F}$ into the well-known compressed sensing problem $\mathbf{Ax} = \mathbf{y}$ which seeks to efficiently identify unique sparse solutions to an under-determined system of equations[7]. This reformulation allowed for conditional bounds on γ of the form $\gamma \leq -\frac{1}{r}\frac{K}{N}W_1^{-1}(\frac{-2K}{erN})$, where though these bounds[8] remained loose and conditional, for two main reasons. The first reason stems from the fact that the focus of the compressed sensing machinery is mainly on the search efficiency and uniqueness of the sparse solutions, rather than on the level of sparsity itself[9]. The second reason is that, while in our computing setting our communication matrix $\mathbf{D}$ must be a function of $\mathbf{F}$, compressed sensing places its focus on designing $\mathbf{A}$ in a manner that is oblivious to the instance of $\mathbf{y}$ (which corresponds to our $\mathbf{F}$). These mismatches are part of what our work

[6]Thus for example, the first column of $\mathbf{E}$ is the coset leader to the coset corresponding to the syndrome described by the first column of $\mathbf{F}$ and by the code whose parity check matrix is $\mathbf{D}$.

[7]This reformulation identifies the observed vector $\mathbf{y}$ with the vectorized $\mathbf{F}$, the sparse solution $\mathbf{x}$ with the vectorized $\mathbf{E}$, and the alphabet matrix $\mathbf{A}$ with the Kronecker product of the communication matrix $\mathbf{D}$ with an identity matrix.

[8]Here $W_1(.)$ is the first branch of the Lambert function, while r calibrates the statistical distribution of $\mathbf{D}$.

[9]Search efficiency and uniqueness are not fundamental to our distributed computing problem. For example, what a compressed sensing exposition of our problem effectively shows is that, under the assumption that the sparsest solution for $\mathbf{E}$ has sparsity-level not more than the above $\gamma = -\frac{1}{r}\frac{K}{N}W_1^{-1}(\frac{-2K}{erN})$, and under the additional assumption that this solution is unique, then — with high probability, in the limit of large N — there is an l_1-minimization approach that will efficiently find this sparsest unique solution. For us, the efficiency of identifying $\mathbf{E}$ is of secondary importance, and the possibility of having another equally sparse $\mathbf{E}$ is not an issue.

here addresses, allowing us to directly explore the fundamental principles of our computing problem.

5.1.3 New Connection Between Distributed Computing, Fixed Support Matrix Factorization, and Tessellations

As we will see almost directly from (5.2), (5.3) and (5.4), (5.9), and also from Lemma 23, solving our distributed computing problem will be equivalent to solving the approximate matrix factorization problem

$$\hat{\mathcal{E}} = \min_{\mathbf{D},\mathbf{E}} \|\mathbf{DE} - \mathbf{F}\|_F^2 \tag{5.10}$$

under dimensionality constraints posed by K, NT, L, and under sparsity constraints on $\mathbf{D}$ and $\mathbf{E}$ posed by δ and γ respectively. These sparsity constraints will be described in detail later on.

This problem encompasses the problem of compressed sensing, and it is known to be NP hard [130]. In general, finding the optimal solution $(\hat{\mathbf{D}}, \hat{\mathbf{E}}) = \arg\min_{\mathbf{D},\mathbf{E}} \|\mathbf{DE} - \mathbf{F}\|_F^2$ to (5.10), under the aforementioned dimensionality and sparsity constraints, requires an infeasible coverage of the entire space of solutions. Otherwise, establishing optimality of an algorithmic solution, generally requires establishing uniqueness of that solution, which is challenging [131], [132]. Furthermore, to date, little is known in terms of clear guarantees on the optimal error performance $\hat{\mathcal{E}}$, for any given $\mathbf{F}$ and any given dimensionality and sparsity constraints on $\mathbf{D}, \mathbf{E}$.

Recently, the work in [53] explored the problem of *Fixed Support (sparse) Matrix Factorization* (FSMF), which — under the equivalent dimensionality and sparsity constraints of the unbounded problem of (5.10) — seeks to find

$$\begin{aligned} \hat{\mathcal{E}}_{\mathcal{I},\mathcal{J}} &= \min_{\mathbf{D},\mathbf{E}} \|\mathbf{DE} - \mathbf{F}\|_F^2 \\ &\text{Subject to: } \operatorname{supp}(\mathbf{D}) \subseteq \mathcal{I}, \operatorname{supp}(\mathbf{E}) \subseteq \mathcal{J} \end{aligned} \tag{5.11}$$

where $\mathcal{I} \subseteq [K] \times [NT]$ and $\mathcal{J} \subseteq [NT] \times [L]$ respectively define the support constraint $\operatorname{supp}(\mathbf{D})$ and $\operatorname{supp}(\mathbf{E})$ of $\mathbf{D}$ and $\mathbf{E}$, where such support constraint entails that $\mathbf{D}(i,j) = 0,\ \forall (i,j) \notin \mathcal{I}$ and $\mathbf{E}(i,j) = 0,\ \forall (i,j) \notin \mathcal{J}$. FSMF remains a broad[10] and challenging problem, partly because, as argued in [53],

[10]For connections between the FSMF problem with *Low-rank matrix approximation* [133], *LU decomposition*[134], *Butterfly structure and fast transforms*[135], Hierarchical $\mathcal{H}-$matrices [136] and *matrix completion* [137], the reader may read [53].

it does not directly accept the existing algorithms from the unconstrained problem in (5.10).

After showing that ill-conditioned supports may lead certain algorithms [53] to converge to local minima (referred to as 'spurious local valleys'), the same work in [53] revealed that for some specific $\mathcal{I}, \mathcal{J}$, some algorithms can provably converge to the corresponding $\hat{\mathcal{E}}_{\mathcal{I},\mathcal{J}}$ which is shown to be unique but, clearly, optimal only *within the space of* $\mathbf{D}, \mathbf{E}$ *defined by the specific* support $\mathcal{I}, \mathcal{J}$. The work in [53] placed some of its focus on a particular class of "disjoint" supports, corresponding to the class of those supports $\mathcal{I}, \mathcal{J}$ that (as we will clarify later on) map onto disjoint regions of $\mathbf{F}$. The finding in [53] is that such "disjoint" $\mathcal{I}, \mathcal{J}$ render (5.11) tractable.

Naturally, depending on $\mathcal{I}, \mathcal{J}$, even such optimal "support-limited" solutions in (5.11) can have unbounded gaps $\hat{\mathcal{E}}_{\mathcal{I},\mathcal{J}} - \hat{\mathcal{E}}$ to the global optimal $\hat{\mathcal{E}}$ from (5.10), as we simply do not know how badly the performance deteriorates by limiting the search within the specific fixed-support set of matrices.

In summary, to date, in terms of explicit solutions to the matrix factorization problem in (5.10), little is known in terms of designing good supports, while in terms of optimality guarantees, these are restricted to within the specific problem in (5.11) where the search is for a given specific support. To date, little is known about explicitly characterizing a desired error performance $\hat{\mathcal{E}}$ under any desired sparsity constraints.

Our contribution to the sparse matrix factorization problem We begin by clarifying that the progress that we have made in this domain is limited to the sparse matrix factorization problem where the sparsity constraints are on the columns of $\mathbf{D}$ and the rows of $\mathbf{E}$. To be clear, we are placing a constraint that no column of $\mathbf{D}$ has a fraction of non-zero elements that exceeds δ, and no row of $\mathbf{E}$ has a fraction of non-zero elements that exceeds γ. With this in place, in terms of designs, for the lossless case of $\epsilon = 0$, our contribution is to identify supports that are optimal over a very large class of $\mathbf{D}, \mathbf{E}$, and to explicitly identify the conditional optimal $(\hat{\mathbf{D}}, \hat{\mathbf{E}})$ of the problem in (5.10) restricted to the aforementioned very large class of $\mathbf{D}, \mathbf{E}$. Under our constraints, these are the sparsest supports that guarantee lossless reconstruction of any $\mathbf{F}$. On the other hand, when $\mathcal{E} > 0$, we can identify — under some additional uniformity assumptions — the best possible support among all conceivable disjoint supports, in that no other disjoint support can be sparser. Hence, in terms of achievable schemes for $\mathbf{D}, \mathbf{E}$, our contribution is to first make the connection between the approach in [53] and the combinatorics analysis of tiling [54] which guarantees that our schemes satisfy a crucial covering condition, and to then explicitly design easy to

represent SVD-based schemes for $\mathbf{D}, \mathbf{E}$ for any $\mathbf{F}$.

In terms of guarantees on performance, and under our disjoint assumption, we provide clear expressions on the minimum possible sparsity γ, δ that guarantees an $\epsilon = 0$. Now for the lossy case of $\epsilon > 0$, we are providing upper bounds on the optimal ϵ under any given sparsity constraint (or equivalently, lower bounds on the sparsity and dimensionality constraints, for any given $\epsilon > 0$). These bounds on the optimal ϵ are provided after we employ the asymptotic setting of scaling K, N, L, after we accept some uniformity assumptions on the weights of the supports of $\mathbf{D}$ and $\mathbf{E}$ (reflecting a uniformity on the load of the servers), and they are provided in the stochastic sense by averaging over the ensemble of possible $\mathbf{F}$, under some basic statistical assumptions. These, to the best of our knowledge, are the first explicit characterizations that identify or bound the optimal performance of the sparse matrix factorization problem, again under our assumptions.

Our ability to handle the stochastic problem, particularly benefits from being able to reduce the overall factorization problem into a sequence of combinatorially-designed[11], SVD-resolved low-rank matrix approximation problems ([53]), whose simplicity opens up to the benefits from existing powerful results from random matrix theory [138]. This same problem of using tessellations to allow for matrix decomposition with statistically good approximations, is a very interesting problem because as we will see, the shape of the tiles affects the goodness of the approximation that each tile offers to particular parts of the matrix $\mathbf{F}$. This is an interesting connection which, to the best of our knowledge, appears here for the first time.

Summary of our contributions on the problem of multi-user distributed computing of linearly-decomposable functions Having made the connection between matrix factorization and distributed computing, we here identify the FSMF problem to be key in the resolution of our real-valued multi-user distributed computing problem, for which we provide the following results.

We first consider the lossless case of $\mathcal{E} = 0$, and after we design an achievable scheme using novel concepts and algorithms introduced in [53] and a converse using combinatorial tiling arguments [54], Theorem 9 establishes the exact system capacity $C = \frac{K}{N_{opt}}$ where

$$N_{opt} = \frac{\min(\Delta, \Gamma)}{T} \lfloor \frac{K}{\Delta} \rfloor \lfloor \frac{L}{\Gamma} \rfloor + \frac{\min(\mathrm{mod}(K, \Delta), \lfloor \frac{L}{\Gamma} \rfloor)}{T} \lfloor \frac{L}{\Gamma} \rfloor \tag{5.12}$$

[11] As we note here, the approach of [53] is a special case of tiling [54], focusing on disjoint tiles.

$$+\frac{\min(\mathrm{mod}(L,\Gamma),\lfloor\frac{K}{\Delta}\rfloor)}{T}\lfloor\frac{K}{\Delta}\rfloor+\frac{\min(\mathrm{mod}(K,\Delta),\mathrm{mod}(L,\Gamma))}{T} \quad (5.13)$$

where this exact optimality is established for the case of $T \geq \min(\Delta,\Gamma)$ as well as the case of $\Delta \geq \Gamma, \Delta|K, T|\Gamma$ as well as the case of $\Gamma \geq \Delta, \Gamma|K, T|\Delta$, or $\Delta|K, \Gamma|L, T|\min(\Delta,\Gamma)$. For the remaining cases (corresponding to a subset of the cases where $T < \min(\Delta,\Gamma)$), our achievable scheme is shown to suffer from only a constant gap to the optimal. In terms of design, many of the above cases are of particular interest because the tessellation patterns that we must construct, must accommodate for tiles of various sizes and shapes. In terms of insight, we can highlight for example the simplified case of (5.40), which applies to the relatively broad setting of $\delta^{-1}, \gamma^{-1} \in \mathbb{N}$ (corresponding to having $\Delta|K, \Gamma|L$), where the capacity now takes the insightful form

$$C = \begin{cases} T\max(\zeta,\gamma), & \text{if } T \mid L\min(\zeta,\gamma) \\ L\zeta\gamma, & \text{if } T > L\min(\zeta,\gamma) \end{cases} \quad (5.14)$$

revealing that for the first case, the optimal communication-vs-computation points (γ,δ), are $(\frac{K}{NT},\frac{T}{K})$ and $(\frac{T}{N},\frac{L}{NT})$, while for the other case the tradeoff takes the form

$$\gamma\delta = \frac{1}{N}.$$

5.1.4 Chapter Organization

The rest of the chapter is organized as follows. Section 5.2 formulates the system model for the setting of multi-user distributed computing of linearly-decomposable functions. Section 5.3 addresses the error-free case, providing schemes and converses that lead to Theorem 9. Subsequently, Section 6.1 addresses the lossy-computation case in the asymptotic setting, providing schemes and converses that lead to Theorem 10. In Section 5.4, we discuss some of the results, before proceeding with various appendix sections that host some of our proofs as well as a small primer on matrix approximation.

5.2 Problem Formulation

We here describe in detail the main parameters of our model, making the clear link between our distributed computing problem and sparse matrix factorization, defining matrices $\mathbf{D}, \mathbf{E}, \mathbf{F}$ and the various metrics, rigorously making the link between the distributed computing problem and the factorization in (5.9) (see also (5.32)) for the error-free case, as well as rigorously presenting the

equivalence of our lossy distributed computing problem to the approximate matrix factorization problem corresponding to (5.10). Let us consider

$$\mathbf{f} \triangleq [F_1, F_2, \ldots, F_K]^\intercal \in \mathbb{R}^K, \tag{5.15}$$

$$\mathbf{f}_k \triangleq [f_{k,1}, f_{k,2}, \ldots, f_{k,L}]^\intercal \in \mathbb{R}^L,\ k \in [K], \tag{5.16}$$

$$\mathbf{w} \triangleq [W_1, W_2, \ldots, W_L]^\intercal \in \mathbb{R}^L \tag{5.17}$$

where $\mathbf{f}$ represents the vector of desired function outputs F_k from (5.1), where $\mathbf{f}_k$ represents the vector of function coefficients $f_{k,\ell}$ from (5.1) for the function requested by user k, and where $\mathbf{w}$ denotes the vector of output files $W_\ell = f_\ell(\cdot)$ again from (5.1). Then recalling the encoding coefficients $e_{n,\ell,t}$ and transmitted signals $z_{n,t}$ from (5.2), as well as the decoding coefficients $d_{k,n,t}$ and decoded functions F'_k from (5.3), we have

$$\mathbf{e}_{n,t} \triangleq [e_{n,1,t}, e_{n,2,t}, \ldots, e_{n,L,t}]^\intercal \in \mathbb{R}^L,\ n \in [N],\ t \in [T], \tag{5.18}$$

$$\mathbf{z}_n \triangleq [z_{n,1}, z_{n,2}, \ldots, z_{n,T}]^\intercal \in \mathbb{R}^T,\ n \in [N], \tag{5.19}$$

$$\mathbf{E}_n \triangleq [\mathbf{e}_{n,1}, \mathbf{e}_{n,2}, \ldots, \mathbf{e}_{n,T}]^\intercal \in \mathbb{R}^{T\times L},\ n \in [N], \tag{5.20}$$

$$\mathbf{d}_{k,n} \triangleq [d_{k,n,1}, d_{k,n,2}, \ldots, d_{k,n,T}]^\intercal \in \mathbb{R}^T,\ k \in [K], n \in [N], \tag{5.21}$$

$$\mathbf{d}_k \triangleq [\mathbf{d}_{k,1}^\intercal, \mathbf{d}_{k,2}^\intercal, \ldots, \mathbf{d}_{k,N}^\intercal]^\intercal \in \mathbb{R}^{N\times T},\ k \in [K] \tag{5.22}$$

and thus from (5.15), the output vector taking the form

$$\mathbf{f} = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_K]^\intercal \mathbf{w} \tag{5.23}$$

as well as the transmitted vector by server n taking the form

$$\mathbf{z}_n = \mathbf{E}_n \mathbf{w} = [\mathbf{e}_{n,1}, \mathbf{e}_{n,2}, \ldots, \mathbf{e}_{n,T}]^\intercal \mathbf{w}. \tag{5.24}$$

This allows us to form the matrices

$$\mathbf{F} \triangleq [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_K]^\intercal \in \mathbb{R}^{K\times L}, \tag{5.25}$$

$$\mathbf{E} \triangleq [\mathbf{E}_1^\intercal, \mathbf{E}_2^\intercal, \ldots, \mathbf{E}_N^\intercal]^\intercal \in \mathbb{R}^{NT\times L}, \tag{5.26}$$

$$\mathbf{D} \triangleq [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_K]^\intercal \in \mathbb{R}^{K\times NT} \tag{5.27}$$

where $\mathbf{F}$ represents the $K \times L$ matrix of all function coefficients across all the users, where $\mathbf{E}$ represents the aforementioned $NT \times L$ *computing and encoding matrix* capturing the computing and linear-encoding tasks of servers in each shot, and where $\mathbf{D}$ represents the $K \times NT$ *communication and decoding matrix* capturing the communication protocol and the linear decoding task done by each user. Thus, we clarify here, that all presented results, schemes and

converses, assume our multi-user linearly-decomposable distributed computing problem, and assume, in their entirety, linear encoding at the servers, and decoding at the receivers.

To see the transition to the matrix factorization problem, we first note that from (5.2) and (5.23) we have that the overall transmitted vector $\mathbf{z} \triangleq [\mathbf{z}_1^\intercal, \mathbf{z}_2^\intercal, \ldots, \mathbf{z}_N^\intercal]^\intercal \in \mathbb{R}^{N \times T}$ takes the form

$$\mathbf{z} = [\mathbf{E}_1^\intercal, \mathbf{E}_2^\intercal, \ldots, \mathbf{E}_N^\intercal]^\intercal \mathbf{w} = \mathbf{E}\mathbf{w} \tag{5.28}$$

and then that given the decoding from (5.3), each retrieved function takes the form

$$F_k' = \mathbf{d}_k^\intercal \mathbf{z} \tag{5.29}$$

thus resulting in the vector of all retrieved functions taking the form $\mathbf{f}' = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_K]^\intercal \mathbf{z}$. Our aim is to minimize the recovery error

$$\mathcal{E} \triangleq \|\mathbf{f}' - \mathbf{f}\|^2 \tag{5.30}$$

solving the following problem

$$\underset{\mathbf{f}'}{\text{minimize}}\, \|\mathbf{f}' - \mathbf{f}\|_2 = \underset{\mathbf{D},\mathbf{E}}{\text{minimize}}\, \|\mathbf{D}\mathbf{E}\mathbf{w} - \mathbf{F}\mathbf{w}\|_2 = \underset{\mathbf{D},\mathbf{E}}{\text{minimize}} \|(\mathbf{D}\mathbf{E} - \mathbf{F})\mathbf{w}\|_2. \tag{5.31}$$

Directly from the above, we see that for the error-free case of $\mathcal{E} = 0$, resolving our distributed computing problem requires that $\mathbf{F}$ be decomposed as

$$\mathbf{F} = \mathbf{D}\mathbf{E} \tag{5.32}$$

as seen in (5.9).

For the lossy case, directly from upcoming Lemma 23, and under the assumption of the independence of $\mathbf{w}$ from $\mathbf{D}, \mathbf{E}, \mathbf{F}$, as well as from the fact that $\mathbf{D}, \mathbf{E}$ are deterministic functions of $\mathbf{F}$, and from the fact that the elements of $\mathbf{w}$ are *i.i.d* with unit variance, we have that

$$\epsilon \triangleq \frac{\mathbb{E}_{\mathbf{F},\mathbf{w}}\{\mathcal{E}\}}{KL} = \frac{\mathbb{E}_{\mathbf{F},\mathbf{w}}\{\|(\mathbf{D}\mathbf{E} - \mathbf{F})\mathbf{w}\|_2^2\}}{KL} = \frac{\mathbb{E}_{\mathbf{F}}\{\|\mathbf{D}\mathbf{E} - \mathbf{F}\|_F^2\}}{KL} \tag{5.33}$$

which holds for any scheme that deterministically derives $\mathbf{D}, \mathbf{E}$. This brings to the fore the aforementioned (cf. (5.10)) minimization $\hat{\mathcal{E}} = \min\limits_{\mathbf{D},\mathbf{E}} \|\mathbf{D}\mathbf{E} - \mathbf{F}\|_F^2$, and our aim will be to bound

$$\hat{\epsilon} \triangleq \frac{\mathbb{E}_{\mathbf{F},\mathbf{w}}\{\min\limits_{\mathbf{D},\mathbf{E}} \|\mathbf{D}\mathbf{E} - \mathbf{F}\|_F^2\}}{KL} \tag{5.34}$$

which matches the function reconstruction error

$$\hat{\epsilon} = \frac{\mathbb{E}_{\mathbf{F},\mathbf{w}}\{\min\limits_{\mathbf{D},\mathbf{E}} \sum_{k=1}^{K} |F_k - F_k'|^2\}}{KL}. \tag{5.35}$$

In terms of the corresponding connection to the sparsity of $\mathbf{D}$ and $\mathbf{E}$, we recall from (5.5) our metrics $\Gamma \triangleq \max_{n\in[N]} |\mathcal{S}_n|$ and $\Delta \triangleq \max_{n\in[N]} |\mathcal{T}_n|$, which directly from (5.22)–(5.24) and from (5.25)–(5.27), imply that

$$\max_{n\in[N]} |\cup_{t=1}^{T} \text{supp}(\mathbf{D}(:,(n-1)T+t))| \leq \Delta \tag{5.36}$$

and that

$$\max_{n\in[N]} |\cup_{t=1}^{T} \text{supp}(\mathbf{E}((n-1)T+t,:))| \leq \Gamma \tag{5.37}$$

and thus we see how the normalized costs $\delta = \frac{\Delta}{K}, \gamma = \frac{\Gamma}{L}$ from (5.6) form the upper bound on the fraction of non-zero elements of the columns of $\mathbf{D}$ and rows of $\mathbf{E}$, respectively.

Finally, from (5.8) we recall the system rate $R = \frac{K}{N}$, the corresponding system capacity C representing the supremum of all rates for error-free function reconstruction, and the two parameters of interest $\zeta = \frac{\Delta}{L}$ and $\kappa = \frac{K}{L}$.

Some additional definitions and assumptions on the stochastic aspects of our problem, will be described in Section 6.1.

5.3 Lossless Distributed Computing of Linearly-Decomposable Functions

We proceed with the main results for the error-free (lossless) case. We recall the distributed computing setting, which involves K users, N servers, T communication slots, computational and communication costs $\Gamma \leq L,\ \Delta \leq K$ respectively, normalized costs $\gamma, \delta \in [0,1]$, a communication-related parameter $\zeta = \frac{\Delta}{L}$, and a system capacity corresponding to the maximum ratio K/N that achieves lossless reconstruction of the functions. We first present the result without any restriction on the dimensions, while we recall that the optimality requires that each submatrix of $\mathbf{F}$ be fullrank, which is a condition that is readily justified in our real-function setting of interest here. We apply a soft assumption on the dimensionality, to offer crisp expressions. All the results hold under the assumption that $NT \geq L$ and $NT \geq K$. The following main result will hold under the basic *disjoint support assumption* on the matrices $\mathbf{D}, \mathbf{E}$, where this assumption will be clarified in Definition 3 right after the theorem. The proofs will follow from an achievable scheme and a converse that will be presented later on.

Theorem 9. *The optimal achievable rate of the lossless K, N, T, Γ, Δ distributed computing setting takes the form $C = K/N_{opt}$, where*

$$\lceil \frac{\min(\Delta,\Gamma)}{T} \rceil \lfloor \frac{K}{\Delta} \rfloor \lfloor \frac{L}{\Gamma} \rfloor + \lceil \frac{\min(\mathrm{mod}(K,\Delta),\Gamma)}{T} \rceil \lfloor \frac{L}{\Gamma} \rfloor$$
$$+ \lceil \frac{\min(\mathrm{mod}(L,\Gamma),\Delta)}{T} \rceil \lfloor \frac{K}{\Delta} \rfloor + \lceil \frac{\min(\mathrm{mod}(K,\Delta),\mathrm{mod}(L,\Gamma))}{T} \rceil \tag{5.38}$$
$$\geq N_{opt} \geq \frac{KL}{T \max(\Gamma,\Delta)} \tag{5.39}$$

and the bounds exactly meet (and thus the achievable scheme is exactly optimal) for $T \geq \min(\Delta,\Gamma)$ as well as for $\Delta \geq \Gamma, \Delta | K, T | \Gamma$ as well as for $\Gamma \geq \Delta, \Gamma | K, T | \Delta$. Else the achievable capacity is within a constant gap from the optimal. Finally, when $\delta^{-1}, \gamma^{-1} \in \mathbb{N}$ (corresponding to the case of $\Delta | K, \Gamma | L, T | \min(\Delta,\Gamma)$), the capacity takes the form

$$C = \begin{cases} T \max(\zeta,\gamma), & \text{if } T \mid L \min(\zeta,\gamma), \\ L\zeta\gamma, & \text{if } T > L\min(\zeta,\gamma). \end{cases} \tag{5.40}$$

Proof. The proof can be found in the appendix Section 5.7 which describes the achievability of the corresponding decomposition $\mathbf{DE} = \mathbf{F}$, and how this is translated to our distributed computing setting with the corresponding communication and computation cost constraints. The converse can be found in the appendix Section 5.8, and so is the proof of exact optimality and order optimality of the achievable scheme. □

The above assumption on optimality is clarified in the following definition.

Definition 3. [Disjoint Support Assumption] We say that two matrices $\mathbf{D} \in \mathbb{R}^{K \times NT}, \mathbf{E} \in \mathbb{R}^{NT \times L}$, accept the *disjoint support assumption* if and only if for any two columns $\mathbf{D}(:,i), \mathbf{D}(:,i'), i, i' \in [NT]$ of $\mathbf{D}$ and the respective two rows $\mathbf{E}(i,:), \mathbf{E}(i',:)$ of $\mathbf{E}$, then $\mathrm{supp}(\mathbf{D}(:,i)\mathbf{E}(i,:)) = \mathrm{supp}(\mathbf{D}(:,i')\mathbf{E}(i',:))$ or $\mathrm{supp}(\mathbf{D}(:,i)\mathbf{E}(i,:)) \cap \mathrm{supp}(\mathbf{D}(:,i'), \mathbf{E}(i',:)) = \emptyset$.

Let us introduce an illustrative example that pertains to the simpler single-shot scenario.

Example 1. Let us see a basic example of a multi-user linearly-decomposable problem. Here, we have N servers tasked with computing functions for $K = 6$ users. These functions are linear combinations of $L = 10$ subfunctions. We will focus on the simplified scenario with $T = 1$ (single-shot communication)[12].

[12]In the single shot scenario, a server broadcasts a single linear combination of the output files to its connected users. For $T > 1$, a server can broadcast multiple linear combinations of the output files, not necessarily to the same set of users. The communication cost Δ measures the union of all the activated links throughout the T shots.

Our budget allows for a maximum of $\Delta = 3$ communication links per server, equivalent to $\delta = \frac{\Delta}{K} = \frac{3}{6}$, and a per-server computational cost of $\Gamma = 5$ (as defined in equation (5.5)) computed subfunctions, corresponding to $\gamma = \frac{\Gamma}{L} = \frac{5}{10}$. With this in place, we seek the minimum number of servers needed to guarantee lossless reconstruction ($\mathcal{E} = 0$) of the desired functions at the users, and for this we draw from (5.40) to conclude that we need $N = 12$ servers. To tackle this challenge of reconstruction, we need to construct two key matrices based on $\mathbf{F}$:

1. The $(NT \times L) = (12 \times 10)$ computing-and-encoding matrix $\mathbf{E}$, which specifies which subfunctions each server computes. Each row of $\mathbf{E}$ should have at most $\Gamma = 5$ non-zero elements.

2. The $(K \times NT) = (6 \times 12)$ communication-and-decoding matrix $\mathbf{D}$, which determines where each server communicates to and how each user collects data from different servers. Each column of $\mathbf{D}$ should have at most $\Delta = 3$ non-zero elements.

These matrices originate from the decomposition $\mathbf{F} = \mathbf{DE}$. Recalling from equation (5.25) that the matrix $\mathbf{F}$ representing the requested functions, is of size $K \times L = 6 \times 10$, and considering $\Gamma = 5$, which corresponds to $\gamma = 1/2$, the solution is as follows:

1. Initially we partition $\mathbf{F}$ into $\frac{K}{\Delta}\frac{L}{\Gamma} = 2 \cdot 2 = 4$ disjoint 3×5 submatrices, which we will call 'tiles', $\mathbf{S}_j \in \mathbb{R}^{\Delta \times \Gamma} = \mathbb{R}^{3\times 5}$, where j ranges from 1 to 4, as illustrated in Figure 5.2.

2. Then using the standard matrix decomposition form (see later on in (5.46)), we SVD-decompose each $\mathbf{S}_j$ into $\mathbf{S}_j = \mathbf{L}_j\mathbf{R}_j$, where $\mathbf{L}_j \in \mathbb{R}^{3\times 3}, \mathbf{R}_j \in \mathbb{R}^{3\times 5}$ for all $j \in [4]$, noting that such full decomposition is possible since the maximum rank of each $\mathbf{S}_j$ is $\min(\Delta, \Gamma) = 3$.

3. Then we construct $\mathbf{D} \in \mathbb{R}^{6\times 12}$ and $\mathbf{E} \in \mathbb{R}^{12\times 10}$ by tiling them with $\mathbf{L}_j$ and $\mathbf{R}_j$ respectively, in the manner clearly illustrated in Figure 5.2. Thus, for example, the upper left 3×3 submatrix of $\mathbf{D}$ is equal to $\mathbf{L}_1$, the 3×3 tile to the right of that is zero, while the lower left corner of $\mathbf{E}$ is equal to $\mathbf{R}_4$.

□

The above example offers a glimpse, albeit partly illustrative, of the general principle behind creating our achievable scheme. In brief, for the simpler case corresponding to (5.40), we begin by splitting our $K \times L$ matrix $\mathbf{F}$, into $\gamma^{-1}\delta^{-1}$ submatrices of size $\Delta \times \Gamma$, and we SVD-decompose (using

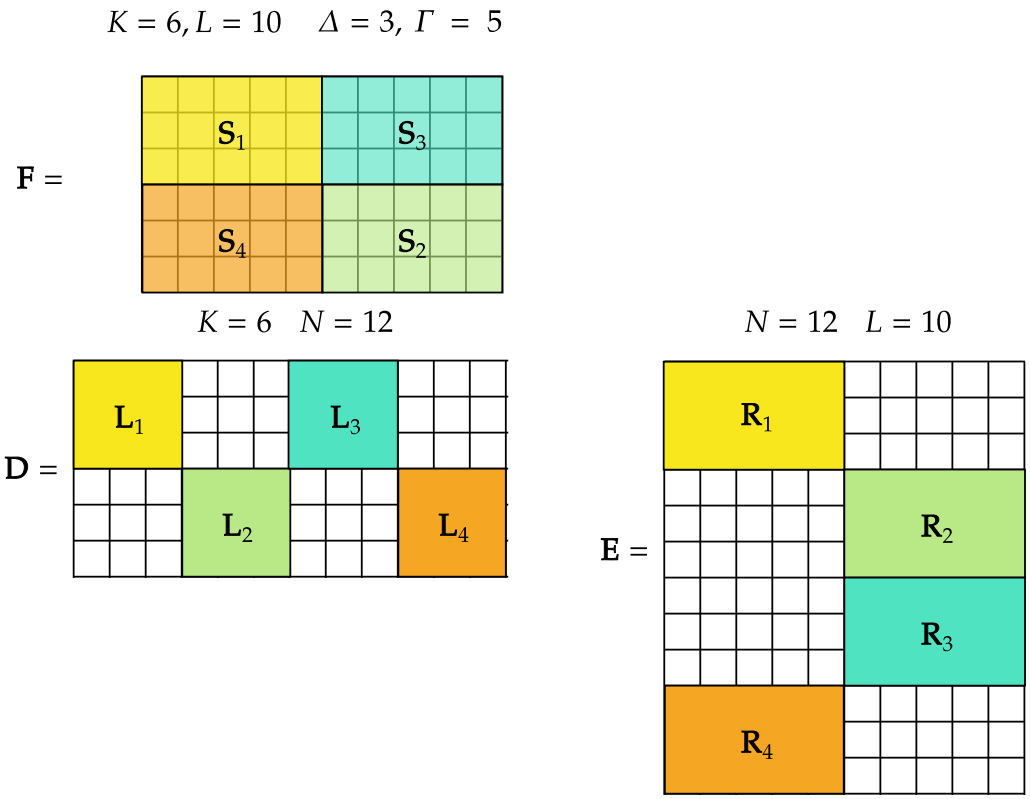


Figure 5.2: Corresponding to Example 1, this figure illustrates the partitioning of $\mathbf{F}$ into 4 tiles of size $(\Delta \times \Gamma) = (3 \times 5)$, and then the sparse tiling of $\mathbf{D}$ and $\mathbf{E}$ with tiles $\mathbf{L}_j$ and $\mathbf{R}_j$ respectively, resulting in the full tiling of $\mathbf{F} = \mathbf{DE}$ which is covered by the four $\mathbf{S}_j = \mathbf{L}_j\mathbf{R}_j, j \in [4]$ (see Figure 5.2), guaranteeing sparsity $\delta = \gamma = \frac{1}{2}$ for $\mathbf{D}$ and $\mathbf{E}$ respectively, thus satisfying the per-server communication and computing constraints, while yielding lossless reconstruction of the functions.

the matrix decomposition form) each submatrix into the $\mathbf{L}_j, j \in [4]$ part that becomes a tile of $\mathbf{D}$, and into the $\mathbf{R}_j, j \in [4]$ part to becomes a tile of $\mathbf{E}$. The tile-placement must respect the sparsity constraints from Γ, Δ and must yield $\mathbf{DE} = \mathbf{F}$. Regarding the required number of servers, a quick rule of thumb (at least for the case of $T = 1$) is that N is simply the number of submatrices, multiplied with the rank of each submatrix[13]. In our example, we had 4 submatrices, each of rank 3, thus we employed 12 servers, which later turns out to be optimal.

On the other hand, once we reduce the computation and communication capabilities of each server, more servers may be required. This is illustrated in the following example.

Example 2. For the same setting as in Example 1, again for $K = 6, L = 10, T = 1$, and again for $\Delta = 3$ (corresponding to $\delta = \frac{1}{2}$), if we wish to substantially reduce the load on each server to only having to compute $\Gamma = 2$ subfunctions (corresponding now to $\gamma = \frac{1}{5}$), then the minimum number of servers N is now 20, and the corresponding tessellation pattern is presented in Figure 5.3, where we see 10 submatrices of rank 2. □

At the same time, some reductions in γ, δ may come for free. As it turns out, there can be multiple tessellation patterns resulting in the same value of N, but depending on the size and placement of the tiles, such patterns could correspond to different Γ and Δ. To illustrate, consider the following example.

Example 3. In a lossless computing setting similar to that in Example 1, we consider again $N = 12$ servers, $K = 6$ users, $L = 10$ subfunctions and $T = 1$. As in Example 1, we maintain a computational cost of $\Gamma = 5$, but now we see that the tessellation pattern in Figure 5.4 allows for a reduced $\Delta = 2$ corresponding to $\delta = 1/3$. □

Such reductions in the communication and computation costs can continue, up to a point, and then again more servers may be required to provide lossless function reconstruction, as suggested by Example 2. The following corollary addresses this aspect, by providing the optimal communication-vs-computation tradeoff for a broad setting. We recall our usual conditions that $NT \geq L$ and $NT \geq K$.

Corollary 2. *In the $(K, N, T, \gamma, \delta)$ lossless distributed computing setting with $\delta^{-1}, \gamma^{-1} \in \mathbb{N}$, the optimal communication-vs-computation relationship takes*

[13]Having a larger T augments the span of the transmitted signals, thus allowing for fewer required servers.

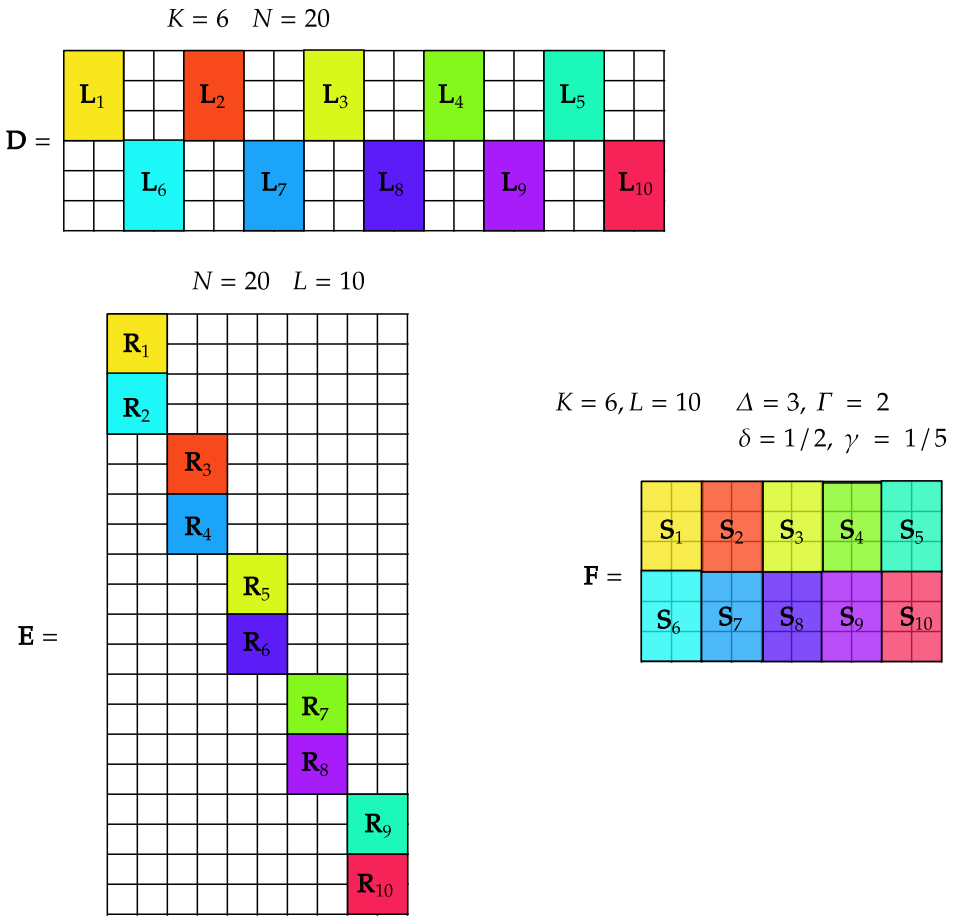


Figure 5.3: A problem setting with the same $K = 6, L = 10,\ \Delta = 3$ and $\mathcal{E} = 0$ as the Example 5.2, but a smaller computation cost $\Gamma = 2$ corresponding to $\gamma = 1/5$. The number of servers used now for zero-error function recovery increases from 12 to 20.

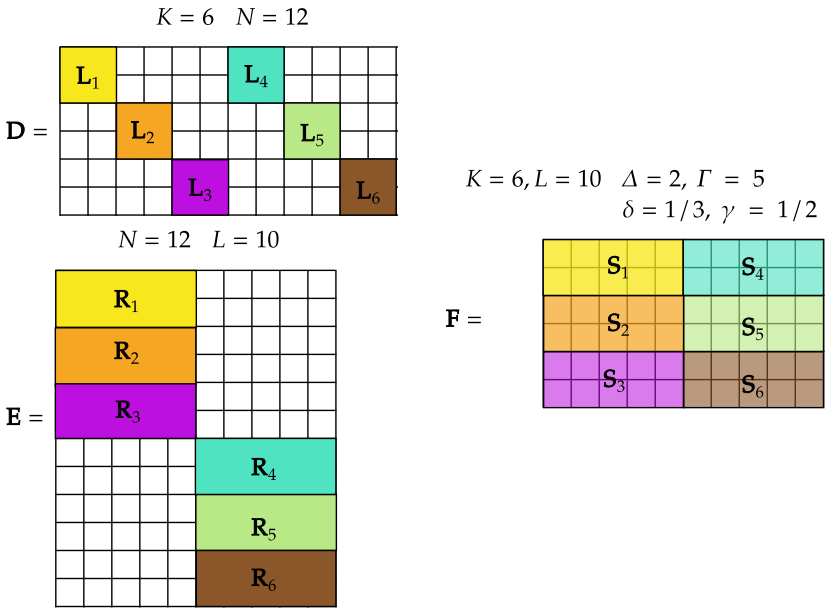


Figure 5.4: Pertaining to Example 3 with $K = 6, L = 10, T = 1, \Gamma = 5$ and an optimal number of $N = 12$ servers, the new tessellation pattern allows for a reduced $\Delta = 2$ reflecting a reduction from $\delta = 1/2$ to $\delta = 1/3$.

the form

$$\gamma\delta = \frac{1}{N}, \quad \textit{for } T > L\min(\zeta, \gamma) \tag{5.41}$$

whereas when $T \mid L\min(\zeta, \gamma)$, *the* (γ, δ) *operating points*

$$(\frac{K}{NT}, \frac{T}{K}) \quad \textit{and } (\frac{T}{L}, \frac{L}{NT}) \tag{5.42}$$

are again optimal.

Proof. The proof is direct from (5.40). □

The results of the above corollary are illustrated in Figure 5.5.

Corollary 3. *For the case where* $T < \max(\Delta, \Gamma)$, *the achievable rate* R *in* (5.38) *is at most a multiplicative factor of* 8 *from the optimal.*

Proof. The proof can be found in the Appendix 5.8. □

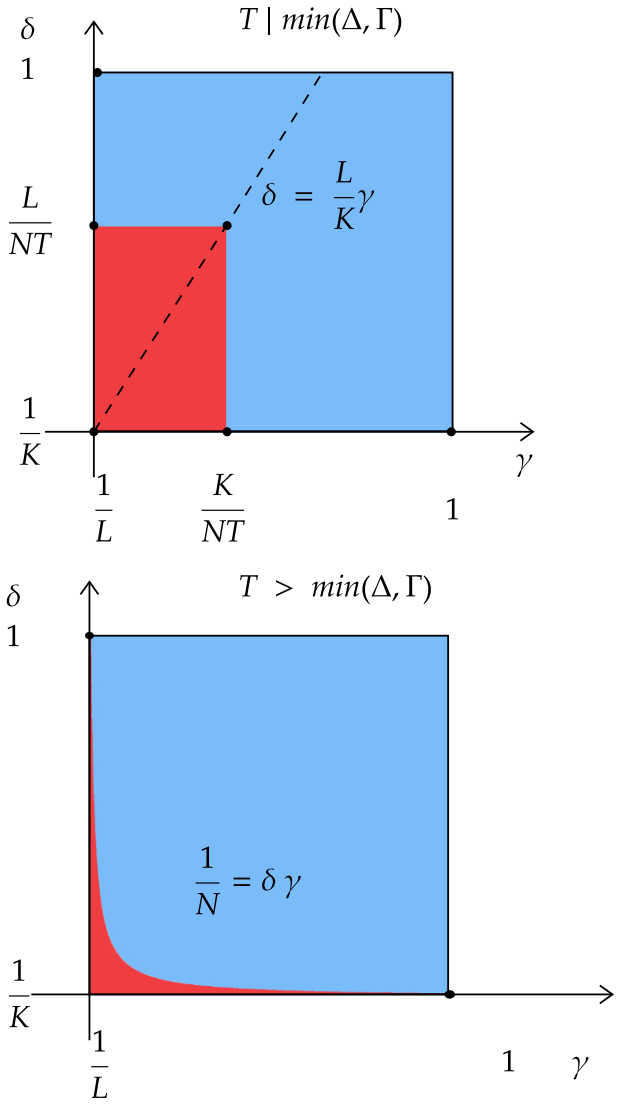


Figure 5.5: On the right we see the optimal performance for $T \geq \min(\Delta, \Gamma)$, which contrasts the blue achievable region with the red provably non-achievable region. On the left, we illustrate for the simple single-shot case, the two optimal points $(\gamma = \frac{K}{N}, \delta = \frac{1}{K})$ and $(\gamma = \frac{1}{L}, \delta = \frac{L}{N})$, which are compared to the operating points $A = (\gamma = 1, \delta = 1/K)$ and $B = (\gamma = 1/L, \delta = 1)$ of two conceivable baseline schemes. Point $A = (\frac{1}{L}, 1)$ is that of a baseline fully-centralized scheme where servers $n \in [K]$ are assigned all subfunctions (the rest are assigned no functions), while point $B = (1, \frac{1}{K})$ corresponds to a fully-parallelized baseline scheme where each server only computes one subfunction output and sends it, by necessity, to all users. The two points correspond to the trivial decompositions $\mathbf{F} = [\mathbf{I}_K \;\; \mathbf{0}_{(K,K-N)}] \cdot [\mathbf{F}^\intercal \;\; \mathbf{0}_{(L,N-K)}]^\intercal$ and $\mathbf{F} = [\mathbf{F} \;\; \mathbf{0}_{(K,N-L)}] \cdot [\mathbf{I}_L \;\; \mathbf{0}_{(L,N-L)}]$ respectively.

5.4 Conclusion

In this Chapter, we investigated the fundamental limits of multi-user distributed computing of real-valued linearly-decomposable functions. In addressing this problem, we have made clear connections to the problem of fixed support matrix factorization, tessellation their via various examplesy. We characterized the error-free system capacity $C = \frac{K}{N_{opt}}$ in Theorem 9 revealed the minimal computational and communication resources γ, δ, N required to accommodate a certain number of users and subfunctions. The same result yields a simple relationship between computational complexity and communication load, as this is described in Corollary 2, and the results are proven optimal over some broad schemes. The lossy variant of this problem in a statistical setting will be investigated in the next chapter. For future work, one can imagine cases where the servers have unbalanced computational and communication capabilities.

5.5 Appendices

5.6 Concepts Relating to the Design of the Schemes

Before describing our schemes and converses, we present in this section some basic properties and definitions that will be useful later on.

We recall that our goal will be to design the communication matrix $\mathbf{D}$ and the computing matrix $\mathbf{E}$ that yield $\mathbf{DE} = \mathbf{F}$ (or $\mathbf{DE} \approx \mathbf{F}$, depending on our case), under fixed N, T, and under a constraint of at most Δ non-zero elements in any column of $\mathbf{D}$ and at most Γ non-zero elements in any row of $\mathbf{E}$. To do so, we will need some basic concepts and definitions relating to the approach[14] in [53].

Definition 4. Given two support constraints $\mathbf{I} \in \{0,1\}^{K \times NT}$ and $\mathbf{J} \in \{0,1\}^{NT \times L}$ of two matrices $\mathbf{D} \in \mathbb{R}^{K \times NT}$ and $\mathbf{E} \in \mathbb{R}^{NT \times L}$ respectively, then for any $n \in [NT]$, we refer to $\mathbf{S}_n(\mathbf{I}, \mathbf{J}) \triangleq \mathbf{I}(:, n)\mathbf{J}(n, :) \in \mathbb{R}^{K \times L}$ as the nth *rank-one contribution support*[15] of $\mathbf{DE}$ [53].

[14]We quickly recall that for a matrix $\mathbf{A}$, then $\mathbf{A}(:, n)$ represents its nth column, $\mathbf{A}(n, :)$ its nth row, supp($\mathbf{A}$) the binary matrix indicating the support of $\mathbf{A}$, while for a vector $\mathbf{a}$, then supp($\mathbf{a}$) represents the set of indices of $\mathbf{a}$ with non-zero elements. Also, when we refer to a support constraint, this will be in the form of a binary matrix that indicates the support (the position of the allowed non-zero elements) of a matrix of interest.

[15]We have naturally assumed that $\mathbf{I}(:, n)$ and $\mathbf{J}(n, :)$ each have at least one non-zero

We note that when the supports are implied, we may shorten $\mathbf{S}_n(\mathbf{I}, \mathbf{J})$ to just $\mathbf{S}_n$. This will relate to the support of $\mathbf{DE}$. On this, we have the following lemma.

Lemma 14. For $\mathbf{I} \triangleq \text{supp}(\mathbf{D})$ and $\mathbf{J} \triangleq \text{supp}(\mathbf{E})$, then

$$\cup_{n=1}^{N} \mathbf{S}_n(\mathbf{I}, \mathbf{J}) \supseteq \text{supp}(\mathbf{DE}). \tag{5.43}$$

Proof. The above follows from Definition 4, from having $\cup_{n=1}^{N} \mathbf{S}_n(\mathbf{I}, \mathbf{J}) = \cup_{n=1}^{N} \mathbf{I}(:, n)\mathbf{J}(n, :)$, and from the fact that $\mathbf{DE} = \sum_{n=1}^{NT} \mathbf{D}(:, n)\mathbf{E}(n, :)$. □

We also need the following definition.

Definition 5. For $\mathbf{I} = \text{supp}(\mathbf{D}) \in \{0, 1\}^{K \times NT}$ and $\mathbf{J} = \text{supp}(\mathbf{E}) \in \{0, 1\}^{NT \times L}$, the *equivalence classes of rank-one supports* are defined by the equivalence relation $i \sim j$ on $[NT]$ which holds if and only if $\mathbf{S}_i = \mathbf{S}_j$. This relation allows us to define $\mathcal{C}$ to be the set of all equivalence classes [53].

The above splits the columns of $\mathbf{D}$ (and correspondingly the rows of $\mathbf{E}$) such that the equivalence $i \sim j$ holds if and only if $\mathbf{I}(:, i)\mathbf{J}(i, :) = \mathbf{I}(:, j)\mathbf{J}(j, :)$.

Definition 6. For two supports $\mathbf{I} \in \{0, 1\}^{K \times NT}, \mathbf{J} \in \{0, 1\}^{NT \times L}$ of $\mathbf{D}$ and $\mathbf{E}$ respectively, and for $\mathcal{C}$ being the collection of equivalence classes as in Definition 5, then each class $\mathcal{P} \in \mathcal{C}$ will have a *representative support* which we will denote as $\mathbf{S}_{\mathcal{P}}$.

Definition 7. For a representative support $\mathbf{S}_{\mathcal{P}}$ of a class $\mathcal{P} \in \mathcal{C}$ (as in Definition 6), and for some $n \in \mathcal{P}$, we define $\mathbf{c}_{\mathcal{P}} \triangleq \mathbf{I}(:, n)$ (resp. $\mathbf{r}_{\mathcal{P}} \triangleq \mathbf{J}(n, :)$) to be the corresponding *component column* (resp. *component row*) of $\mathbf{S}_{\mathcal{P}}$, and we define $\mathcal{C}_{\mathcal{P}} \triangleq \text{supp}(\mathbf{c}_{\mathcal{P}}) \subset [K]$ to be the set of indices of the non-zero elements in $\mathbf{c}_{\mathcal{P}}$, while we define $\mathcal{R}_{\mathcal{P}} \triangleq \text{supp}(\mathbf{r}_{\mathcal{P}}) \subset [L]$ to be the set of indices of the non-zero elements in $\mathbf{r}_{\mathcal{P}}$.

Remark 5. As we will see later on, the non-zero part of $\mathbf{S}_{\mathcal{P}}$ will define the position and size of *tile*[16] $\mathcal{P}$ of $\mathbf{DE}$. Furthermore, the non-zero part of $\mathbf{I}(:, \mathcal{P})$ and the non-zero part of $\mathbf{J}(\mathcal{P}, :)$ will define the so-called tiles of $\mathbf{D}$ and $\mathbf{E}$ respectively[17], and they will naturally map onto tile $\mathcal{P}$ of $\mathbf{DE}$. Also note that a tile $\mathcal{P}$ (which corresponds to the representative contribution support $\mathbf{S}_{\mathcal{P}}$, i.e., which corresponds to the entire class $\mathcal{P}$) should not be confused with the rank-one contribution support $\mathbf{S}_n$. A tile may entail multiple (specifically $|\mathcal{P}|$ such) rank-one contribution supports.

element.

[16]Note that $\mathbf{S}_n = \mathbf{S}_{\mathcal{P}}$ for any $n \in \mathcal{P}$.

[17]To help the reader with the notation, we remind here that $\mathbf{I}(:, \mathcal{P})$ simply refers to the columns of $\mathbf{I}$ labeled by the elements inside set $\mathcal{P}$. Similarly, $\mathbf{J}(\mathcal{P}, :)$ corresponds to the rows of $\mathbf{J}$ indexed by $\mathcal{P}$.

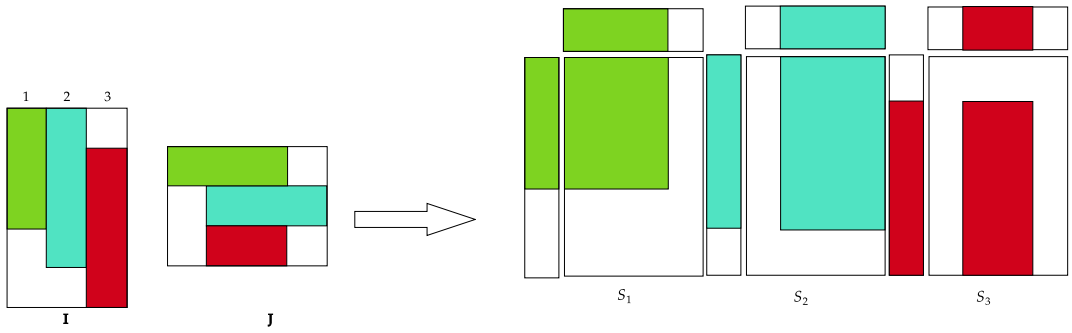


Figure 5.6: The figure on the left illustrates the support constraints $\mathbf{I}$ and $\mathbf{J}$ on $\mathbf{D}$ and $\mathbf{E}$ respectively. The constraints $\mathbf{I}(:,1)$ and $\mathbf{J}(1.:)$ on the columns and rows of $\mathbf{D}$ and $\mathbf{E}$ respectively are colored green, $\mathbf{I}(:,2)$ and $\mathbf{J}(2.:)$ are colored cyan and $\mathbf{I}(:,3)$ and $\mathbf{J}(3.:)$ are colored red. The product of a column with a row of the same color, yields the corresponding rank-one contribution support $\mathbf{S}_n(\mathbf{I},\mathbf{J}), n = 1,2,3$, as described in Definition 4, and as illustrated on the right side of the figure.

We proceed with further definitions.

Definition 8. For every subset $\mathcal{C}' \subseteq \mathcal{C}$ of equivalence classes, we define the *union of the representative supports* $\mathcal{S}_{\mathcal{C}'} \triangleq \cup_{\mathcal{P}\in\mathcal{C}'}\mathcal{S}_{\mathcal{P}}$ to be — as defined in Section 5.1.4 — the point-wise logical OR of the corresponding $\mathcal{S}_{\mathcal{P}}$.

In the above, $\mathcal{S}_{\mathcal{C}'}$ is simply the area of the product matrix $\mathbf{DE}$ covered by all the tiles $\mathcal{P}$ in $\mathcal{C}'$. Furthermore we have the following definition.

Definition 9 ([53])**.** The *maximum rank of a representative support of class* $\mathcal{P} \in \mathcal{C}$ takes the form

$$r_{\mathcal{P}} \triangleq \min(|\mathcal{C}_{\mathcal{P}}|, |\mathcal{R}_{\mathcal{P}}|). \tag{5.44}$$

As one can readily see, when $\mathbf{I} = \text{supp}(\mathbf{D})$ and $\mathbf{J} = \text{supp}(\mathbf{E})$, then the part of matrix $\mathbf{DE}$ covered by tile $\mathcal{S}_{\mathcal{P}}$, can have rank which is at most $r_{\mathcal{P}}$.

Remark 6. While in mathematics, tiling a matrix (or a surface area) corresponds to finding a way to 'cover' this matrix with tiles of specific sizes, here our task is somewhat different. Here our goal is to cover a product matrix ($\mathbf{DE}$ in this case) with tiles that are the result of a product of properly placed tiles of $\mathbf{D}$ and of $\mathbf{E}$, where these last tiles must adhere to the sparsity

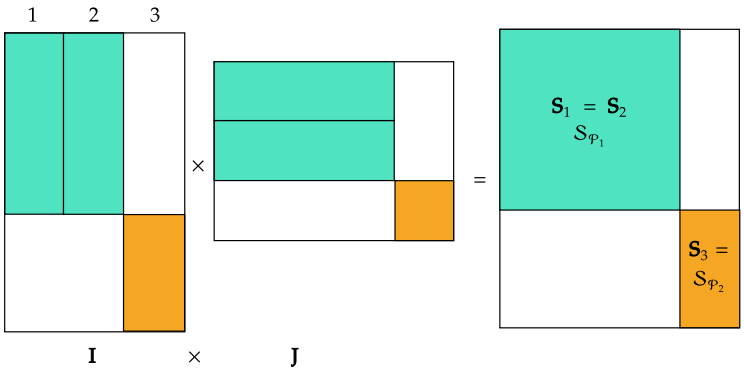


Figure 5.7: This figure illustrates three different rank-one contribution supports $\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3$, where the first two fall into the same equivalence class $\mathcal{S}_{\mathcal{P}_1} = \mathbf{S}_1 = \mathbf{S}_2$, while $\mathcal{S}_{\mathcal{P}_2} = \mathbf{S}_3$.

constraints of $\mathbf{D}$ and $\mathbf{E}$. Furthermore, the size of the non-zero part of each $\mathbf{S}_n$, $n \in \mathcal{P}$, together with the number of such equivalent elements in $\mathcal{P}$, will jointly determine the degree of the approximation of the corresponding submatrix of $\mathbf{F}$.

Before proceeding with the scheme, we here also give a very brief reminder on the basic concepts regarding SVD decompositions.

5.6.1 Brief Primer on Matrix Approximation

For an $m \times n$ matrix $\mathbf{A}$, for some $k \leq \min\{m, n\}$, the rank-k approximation of $\mathbf{A}$ takes the form[18]

$$\mathbf{A}_{m\times n} \simeq \mathbf{B}_{m\times k}\mathbf{C}_{k\times n} \tag{5.45}$$

where $\mathbf{B}_{m\times k}$ and $\mathbf{C}_{k\times n}$ are two full rank matrices of dimension $m \times k$ and $k \times n$ respectively. Such decomposition allows us to represent $\mathbf{A}$ with at most $k(m+n)$ elements, which can be substantially fewer than the mn elements required to represent $\mathbf{A}$.

[18]In the following, when needed we will be using matrix-subscripts to denote the dimensionality of matrices. For example, we will be using $\mathbf{B}_{m\times k}$ to emphasize that matrix $\mathbf{B}$ has dimensionality $m \times k$.

As we will further recall below, matrices can be represented by employing SVD decomposition. For a rank-r matrix $\mathbf{A}_{m\times n}$, this SVD takes the form

$$\mathbf{A} = \mathbf{U}_{m\times r}\mathbf{S}_{r\times r}\mathbf{V}^{\intercal}_{r\times n} = \Big[\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r\Big] \begin{bmatrix} \sigma_1 & & & 0 \\ & \sigma_2 & & \\ & & \ddots & \\ 0 & & & \sigma_r \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^{\intercal} \\ \mathbf{v}_2^{\intercal} \\ \vdots \\ \mathbf{v}_r^{\intercal} \end{bmatrix} = \sum_{i=1}^{r} \sigma_i u_i v_i^{\intercal} \tag{5.46}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, where $\mathbf{S}$ is diagonal with entries $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq 0$ being the singular values in descending order, where $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r$ are the columns of $\mathbf{U}_{m\times r}$, and where $\mathbf{v}_1^{\intercal}, \mathbf{v}_2^{\intercal}, \ldots, \mathbf{v}_r^{\intercal}$ are the rows of $\mathbf{V}_{n\times r}$.

Subsequently, the optimal rank-k approximation matrix $\mathbf{A}_k, k < r$ of $\mathbf{A}$, takes the form

$$\mathbf{A}_k = \mathbf{U}_{m\times k}(\mathbf{S}_k)_{k\times k}\mathbf{V}^{\intercal}_{k\times n} = \sum_{i=1}^{k} \sigma_i u_i v_i^{\intercal} = \mathbf{U}_{m\times k}\mathbf{U}^{\intercal}_{k\times m}\mathbf{A} \tag{5.47}$$

$$= (\sum_{i=1}^{k} u_i u_i^{\intercal})\mathbf{A} = \mathbf{A}\mathbf{V}_{n\times k}\mathbf{V}^{\intercal}_{k\times n} = \mathbf{A}(\sum_{i=1}^{k} \mathbf{v}_i \mathbf{v}_i^{\intercal}) \tag{5.48}$$

where $(\mathbf{S}_k)_{k\times k} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_k), \mathbf{U}_{m\times k} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k]$ and $\mathbf{V}^{\intercal}_{k\times n} = [\mathbf{v}_1^{\intercal}, \mathbf{v}_2^{\intercal}, \ldots, \mathbf{v}_k^{\intercal}]$. This approximation, also referred to as the *truncated SVD approximation*, is simply the projection of $\mathbf{A}$ onto the space spanned by the strongest k singular vectors of $\mathbf{A}$. Directly from the well-known Eckart-Young Theorem [133], this approximation is optimal, as it guarantees $\|\mathbf{A} - \mathbf{B}\|_F \geq \|\mathbf{A} - \mathbf{A}_k\|_F = \sqrt{\sigma_{k+1}^2 + \ldots + \sigma_r^2}$ for any rank-k matrix $\mathbf{B}$.

With the above in place, we proceed to describe the scheme for the lossless case.

5.7 Scheme for Lossless Reconstruction (Achievability Proof of Theorem 9)

We recall that we wish to design the decomposition $\mathbf{DE} = \mathbf{F}$, for fixed N, K, T, Δ, Γ, that will guarantee the reconstruction of all requested function outputs, with each of the N servers locally calculating up to Γ subfunctions, and each engaging in communication with at most Δ users over the entirety of T broadcast shots.

In what follows, we will explain the tile design, elaborating on the positioning of these tiles in $\mathbf{D}$ and $\mathbf{E}$, as well as defining how these tiles are filled

as a function of the SVD decomposition of carefully selected submatrices of $\mathbf{F}$. This is done both for the single-shot as well as the multi-shot scenarios. Our design naturally abides by the number of servers available. In particular, the position and size of the tiles of $\mathbf{D}$ and $\mathbf{E}$ will be crucial, because they will determine the rank of corresponding submatrices of $\mathbf{DE}$, where in particular the sum of these ranks must be bounded by NT. Before proceeding with the scheme, we also note that we have Examples 4 and 5 that illustrate the design of our scheme for the more challenging case of $\Delta \nmid K, \Gamma \nmid L, T = 1$ as well as of $\Delta | K, \Gamma | L, T > 1$.

5.7.1 Construction of $\mathbf{D}, \mathbf{E}$

The construction will involve the following steps: a) Sizing and positioning the tiles of $\mathbf{D}, \mathbf{E}$ and $\mathbf{DE}$, b) Filling the non-zero tiles in $\mathbf{DE}$ as a function of $\mathbf{F}$, and finally c) Filling the tiles in $\mathbf{D}$ and $\mathbf{E}$. We elaborate on these steps below.

First step: Sizing and positioning the tiles of $\mathbf{D}, \mathbf{E}$ and of $\mathbf{DE}$ We first partition the set of equivalent classes $\mathcal{C}$ (cf. Definition 5) into the following subsets of equivalence classes

$$\mathcal{C}_1 \triangleq \{\mathcal{P}_{i,j} \mid \mathbf{c}_{\mathcal{P}_{i,j}} = [\mathbf{0}^\intercal_{(i-1)\Delta}, \mathbf{1}^\intercal_{\Delta}, \mathbf{0}^\intercal_{K-i\Delta}]^\intercal, \mathbf{r}_{\mathcal{P}_{i,j}} = [\mathbf{0}^\intercal_{(j-1)\Gamma}, \mathbf{1}^\intercal_{\Gamma}, \mathbf{0}^\intercal_{L-j\Gamma}], \tag{5.49}$$

$$(i,j) \in [\lfloor \frac{K}{\Delta} \rfloor] \times [\lfloor \frac{L}{\Gamma} \rfloor]\}, \tag{5.50}$$

$$\mathcal{C}_2 \triangleq \{\mathcal{P}_{i,*} \mid \mathbf{c}_{\mathcal{P}_{i,*}} = [\mathbf{0}^\intercal_{(i-1)\Delta}, \mathbf{1}^\intercal_{\Delta}, \mathbf{0}^\intercal_{K-i\Delta}]^\intercal, \tag{5.51}$$

$$\mathbf{r}_{\mathcal{P}_{i,*}} = [\mathbf{0}^\intercal_{L-\mathrm{mod}(L,\Gamma)}, \mathbf{1}^\intercal_{\mathrm{mod}(L,\Gamma)}], i \in [\lfloor \frac{K}{\Delta} \rfloor]\}, \tag{5.52}$$

$$\mathcal{C}_3 \triangleq \{\mathcal{P}_{*,j} \mid \mathbf{c}_{\mathcal{P}_{*,j}} = [\mathbf{0}^\intercal_{K-\mathrm{mod}(K,\Delta)}, \mathbf{1}^\intercal_{\mathrm{mod}(K,\Delta)}]^\intercal, \tag{5.53}$$

$$\mathbf{r}_{\mathcal{P}_{*,j}} = [\mathbf{0}^\intercal_{(j-1)\Gamma}, \mathbf{1}^\intercal_{\Gamma}, \mathbf{0}^\intercal_{L-j\Gamma}], j \in [\lfloor \frac{L}{\Gamma} \rfloor]\}, \tag{5.54}$$

$$\mathcal{C}_4 \triangleq \{\mathcal{P} \mid \mathbf{c}_{\mathcal{P}} = [\mathbf{0}^\intercal_{K-\mathrm{mod}(K,\Delta)}, \mathbf{1}^\intercal_{\mathrm{mod}(K,\Delta)}]^\intercal, \tag{5.55}$$

$$\mathbf{r}^\intercal_{\mathcal{P}} = [\mathbf{0}^\intercal_{L-\mathrm{mod}(L,\Gamma)}, \mathbf{1}^\intercal_{\mathrm{mod}(L,\Gamma)}]\} \tag{5.56}$$

where $\mathbf{c}_{\mathcal{P}_{i,j}}, \mathbf{c}_{\mathcal{P}_{i,*}}, \mathbf{c}_{\mathcal{P}_{*,j}}, \mathbf{c}_{\mathcal{P}}$ are the corresponding component columns (as defined in Definition 6) of $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4$ respectively. Similarly $\mathbf{r}_{\mathcal{P}_{i,j}}, \mathbf{r}_{\mathcal{P}_{i,*}}, \mathbf{r}_{\mathcal{P}_{*,j}}, \mathbf{r}_{\mathcal{P}}$ are the corresponding component rows, from the same definition. These describe the exact position and size of each tile of $\mathbf{D}$, of $\mathbf{E}$, and thus automatically of each tile of $\mathbf{DE}$.

We now note that the number of classes in each subset $\mathcal{C}_i,\ i=1,2,3,4$, is equal to

$$|\mathcal{C}_1| = \lfloor\frac{K}{\Delta}\rfloor\lfloor\frac{L}{\Gamma}\rfloor,\ |\mathcal{C}_2| = \lfloor\frac{K}{\Delta}\rfloor,\ |\mathcal{C}_3| = \lfloor\frac{L}{\Gamma}\rfloor,\ |\mathcal{C}_4| = 1 \tag{5.57}$$

while we also note from (5.44) and Definition 9, that the maximum rank of each representative support (i.e. of each tile of **DE**) takes the form

$$r_{\mathcal{P}} = \begin{cases} \min(\Delta,\Gamma), & \mathcal{P}\in\mathcal{C}_1, \\ \min(\mathrm{mod}(K,\Delta),\Gamma), & \mathcal{P}\in\mathcal{C}_2, \\ \min(\mathrm{mod}(L,\Gamma),\Delta), & \mathcal{P}\in\mathcal{C}_3, \\ \min(\mathrm{mod}(K,\Delta),\mathrm{mod}(L,\Gamma)), & \mathcal{P}\in\mathcal{C}_4. \end{cases} \tag{5.58}$$

The above information will be essential in enumerating our equivalence classes and associating each such class to a collection of servers.

Second step: Filling the non-zero tiles in DE as a function of F Recall that we have a tile $\mathbf{S}_{\mathcal{P}}(\mathcal{R}_{\mathcal{P}},\mathcal{C}_{\mathcal{P}})$ corresponding to the non-zero elements of $\mathbf{S}_{\mathcal{P}}$. This tile is now empty, in the sense that the non-zero entries are all equal to 1. To fill this tile, we consider

$$\mathbf{F}_{\mathcal{P}} \triangleq (\mathbf{F}\odot\mathbf{S}_{\mathcal{P}})(\mathcal{R}_{\mathcal{P}},\mathcal{C}_{\mathcal{P}}),\ \forall\mathcal{P}\in\cup_{i=1}^{4}\mathcal{C}_i \tag{5.59}$$

where this filled tile, in our current lossless case, is simply the submatrix of $\mathbf{F}$ at the position defined by the non-zero elements of $\mathbf{S}_{\mathcal{P}}$. The schematic illustration in Figure 5.8 of Example 4, may help clarify the above.

Third step: Filling the tiles in D and E We now SVD-decompose each $\mathbf{F}_{\mathcal{P}}$, as

$$\mathbf{F}_{\mathcal{P}} = \mathbf{D}_{\mathcal{P}}\mathbf{E}_{\mathcal{P}} \tag{5.60}$$

where $\mathbf{D}_{\mathcal{P}}\in\mathbb{R}^{|\mathcal{R}_{\mathcal{P}}|\times r_{\mathcal{P}}}, \mathbf{E}_{\mathcal{P}}\in\mathbb{R}^{r_{\mathcal{P}}\times|\mathcal{C}_{\mathcal{P}}|}$. Going back to (5.46), our matrices $\mathbf{F}_{\mathcal{P}},\mathbf{D}_{\mathcal{P}},\mathbf{E}_{\mathcal{P}}$ are associated to $\mathbf{A}$, $\mathbf{U}\cdot\mathbf{S}$ and $\mathbf{V}$ respectively, and all correspond to complete SVD decompositions. For a visual representation of this second step, we provide Figure 5.8 of Example 4.

Let us now enumerate our classes (i.e., our tiles) as follows

$$\cup_{i=1}^{4}\mathcal{C}_i = \{\mathcal{P}_1,\mathcal{P}_2,\ldots,\mathcal{P}_m\},\ m\in\mathbb{N}. \tag{5.61}$$

Now let us position each tile of $\mathbf{D}$ as follows

$$\mathcal{R}_{\mathcal{P}_j},\ [\sum_{i=1}^{j-1}T\lceil\frac{r_{\mathcal{P}_i}}{T}\rceil+1 : \sum_{i=1}^{j}T\lceil\frac{r_{\mathcal{P}_i}}{T}\rceil],\ \forall\mathcal{P}_j\in\cup_{i=1}^{4}\mathcal{C}_i \tag{5.62}$$

and each tile of $\mathbf{E}$ as follows

$$[\sum_{i=1}^{j-1} T\lceil\frac{r_{\mathcal{P}_i}}{T}\rceil + 1 : \sum_{i=1}^{j} T\lceil\frac{r_{\mathcal{P}_i}}{T}\rceil],\ \mathcal{C}_{\mathcal{P}_j},\ \forall \mathcal{P}_j \in \cup_{i=1}^{4}\mathcal{C}_i \tag{5.63}$$

where the above describes the indices of columns and rows where each tile resides.

At this point, we note that for the single shot case of $T = 1$, the above yields

$$\mathbf{D}(\mathcal{R}_{\mathcal{P}_j}, [\sum_{i=1}^{j-1} r_{\mathcal{P}_i} + 1, \sum_{i=1}^{j} r_{\mathcal{P}_i}]) = \mathbf{D}_{\mathcal{P}_j} \tag{5.64}$$

and

$$\mathbf{E}([\sum_{i=1}^{j-1} r_{\mathcal{P}_i} + 1, \sum_{i=1}^{j} r_{\mathcal{P}_i}], \mathcal{C}_{\mathcal{P}_j}) = \mathbf{E}_{\mathcal{P}_j} \tag{5.65}$$

while naturally the remaining non-assigned elements of $\mathbf{D}$ and $\mathbf{E}$ are zero. This step can also be visualized in Figure 5.8 which illustrates this step as it applies to Example 4.

We recall from Section 2.3 (cf. (2.78), (2.81)), that each server $n \in [N]$ corresponds to a column and row index of $\mathbf{D}$ and $\mathbf{E}$ respectively. To make the connection between N and the parameters employed in our scheme, let us recall our tiles $\mathcal{P} \in \cup_{i\in[4]}\mathcal{C}_i$ and the corresponding $\mathbf{D}_{\mathcal{P}}$ as seen in (5.60), and let us recall, as described in (5.64), that all such $\mathbf{D}_{\mathcal{P}}$ occupy $\sum_{i=1}^{m} r_{\mathcal{P}_i}$ columns in total. Combining this information with the value of $r_{\mathcal{P}}$ in (5.58), yields that

$$\begin{aligned} N = \sum_{i\in[4]}\sum_{\mathcal{P}\in\mathcal{C}_i} r_{\mathcal{P}}|\mathcal{C}_i| &= \min(\Delta,\Gamma)\lfloor\frac{K}{\Delta}\rfloor\lfloor\frac{L}{\Gamma}\rfloor && (5.66)\\ &\quad + \min(\mathrm{mod}(K,\Delta),\Gamma)\lfloor\frac{L}{\Gamma}\rfloor \\ &\quad + \min(\mathrm{mod}(L,\Gamma),\Delta)\lfloor\frac{K}{\Delta}\rfloor && (5.67)\\ &\quad + \min(\mathrm{mod}(K,\Delta),\mathrm{mod}(L,\Gamma)) && (5.68) \end{aligned}$$

which proves Theorem 9 for the case of $T = 1$. An additional clarifying illustration for the single shot case can be found in Example 4.

For the case of $T > 1$, the difference is in the third step, where we now replace (5.64) and (5.65), where by accounting for (5.63), we see that the tiles of $\mathbf{D}$ and $\mathbf{E}$ respectively take the form

$$\mathbf{D}_{\mathcal{P}_j} = \mathbf{D}(\mathcal{R}_{\mathcal{P}_j}, [\sum_{i=1}^{j-1} T\lceil\frac{r_{\mathcal{P}_i}}{T}\rceil + 1 : \sum_{i=1}^{j} T\lceil\frac{r_{\mathcal{P}_i}}{T}\rceil]) \tag{5.69}$$

and

$$\mathbf{E}_{\mathcal{P}_j} = \mathbf{E}([\sum_{i=1}^{j-1} T\lceil\frac{r_{\mathcal{P}_i}}{T}\rceil + 1 : \sum_{i=1}^{j} T\lceil\frac{r_{\mathcal{P}_i}}{T}\rceil], \mathcal{C}_{\mathcal{P}_j}). \tag{5.70}$$

One aspect in our design that distinguishes the multi-shot from single-shot case, regards the association of tiles to servers. An additional level of complexity here has to do with what one might describe as an "accumulation of rank" at the different servers. As we have realized, there is an association between tiles and servers. In the case of $T = 1$, a single tile $\mathcal{P}_i$ of rank $r_{\mathcal{P}_i}$, could be associated with $r_{\mathcal{P}_i}$ servers, each contributing to a single rank. If the number of servers associated with a tile reached the rank of that tile, then that tile could be decomposed in a lossless manner. Now though, as one can imagine, having multiple shots (corresponding to $T > 1$) can allow a single server to span more than one dimension, i.e., to contribute to more than one rank inside that tile. For arbitrary T and $r_{\mathcal{P}_i}$ though, one can imagine the possibility of having underutilized servers. Imagine for example a tile with rank $r_{\mathcal{P}_i}$, which could fully utilize $\lfloor\frac{r_{\mathcal{P}_i}}{T}\rfloor$ servers, leaving the tile with an accumulated rank that is $\text{mod}(r_{\mathcal{P}_i}, T)$ smaller than its desired $r_{\mathcal{P}_i}$, and leaving us also with a server that is underutilized (in terms of accumulating rank) in the decomposition of the particular tile. One might consider as a remedy, the possibility of associating the underutilized server with an additional tile. This though, runs the risk of violating the communication and computation constraints, simply because this additional (second) tile of **DE** may correspond to a tile of **D** and **E**, whose union with the aforementioned (first) tile of **D** or **E** associated to this same underutilized server, may have a number of non-zero elements that exceeds the number allowed by the communication and computation constraints. The scheme that we present accounts for this, and provably maintains a small and bounded under-utilization of our servers compared to the optimal case.

With this in mind, we associate $\lceil\frac{r_{\mathcal{P}}}{T}\rceil$ servers to each equivalence class where, continuing as before, we can evaluate $r_{\mathcal{P}}$ using (5.58) for any $\mathcal{P} \in \mathcal{C}_j, j \in [4]$. This yields automatically

$$\begin{aligned} N = &\lceil\frac{\min(\Delta,\Gamma)}{T}\rceil\lfloor\frac{K}{\Delta}\rfloor\lfloor\frac{L}{\Gamma}\rfloor + \lceil\frac{\min(\text{mod}(K,\Delta),\Gamma)}{T}\rceil\lfloor\frac{L}{\Gamma}\rfloor \\ &+ \lceil\frac{\min(\text{mod}(L,\Gamma),\Delta)}{T}\rceil\lfloor\frac{K}{\Delta}\rfloor + \lceil\frac{\min(\text{mod}(K,\Delta),\text{mod}(L,\Gamma))}{T}\rceil. \end{aligned} \tag{5.71}$$

To complete the achievability proof of Theorem 9, we proceed to evaluate (5.40), which gives our capacity to be

$$C_0 \overset{(a)}{=} \frac{K}{N} \overset{(b)}{=} \frac{\Delta\Gamma}{\lceil\frac{\min(\Delta,\Gamma)}{T}\rceil L} \overset{(c)}{=} \begin{cases} (T/L)(\Delta\Gamma/\min(\Delta,\Gamma)), & \text{if } T \mid \min(\Delta,\Gamma), \\ (\Delta\Gamma)/L, & \text{if } T > \min(\Delta,\Gamma) \end{cases}$$

where (a) follows by definition, (b) follows from Theorem 9, and (c) follows after basic algebraic manipulations where we see that if $T|\min(\Delta,\Gamma)$ then $\lceil\frac{\min(\Delta,\Gamma)}{T}\rceil = \frac{\min(\Delta,\Gamma)}{T}$, while if $T > \min(\Delta,\Gamma)$ then $\lceil\frac{\min(\Delta,\Gamma)}{T}\rceil = 1$. The proof is then completed directly after applying (1.1) and (5.7) to evaluate the RHS of the last equality.

5.7.2 Examples

We now present two small examples, first for the single-shot and then for the multi-shot case, which can help us better understand the concepts of representative supports (tiles), and the tessellation pattern that yields $\mathbf{D}$ and $\mathbf{E}$.

Example 4. Consider a single-shot scenario with $K = 7$ users, $L = 11$ subfunctions (thus corresponding to a demand matrix $\mathbf{F} \in \mathbb{R}^{7\times 11}$), under the constraint $\Delta = 3$ and $\Gamma = 5$. Let us go through the design steps described above.

- *First step — Sizing and positioning the tiles of* $\mathbf{D}, \mathbf{E}$ *and of* $\mathbf{DE}$*:* The positions of the tiles are derived according to (5.50)–(5.56), yielding the corresponding tessellation pattern illustrated in Figure 5.8. As we can see, the pattern entails four tile families $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4$, of respective sizes $3\times 5, 3\times 1, 1\times 5$ and 1×1. Each family has the following number of tiles $|\mathcal{C}_1| = \lfloor\frac{K}{\Delta}\rfloor\lfloor\frac{L}{\Gamma}\rfloor = 2\times 2 = 4$, $|\mathcal{C}_2| = \lfloor\frac{K}{\Delta}\rfloor = 2$, $|\mathcal{C}_3| = \lfloor\frac{L}{\Gamma}\rfloor = 2$, $|\mathcal{C}_4| = 1$, and the tiles have a maximum rank (cf. Definition 9) equal to $r_{\mathcal{P}} = 3$ for $\mathcal{P} \in \mathcal{C}_1$, and $r_{\mathcal{P}} = 1$ for the rest. Figure 5.8 also illustrates how the designed tessellation pattern successfully covers $\mathbf{F}$, which — in the lossless case — is a necessary condition.

- *Second step — Filling the non-zero tiles in* $\mathbf{DE}$*:* The master node extracts the submatrices corresponding to each of the tiles as described in (5.59), and we have now matrices $\mathbf{F}_{\mathcal{P}} = (\mathbf{F}\odot\mathbf{S}_{\mathcal{P}})(\mathcal{R}_{\mathcal{P}}, \mathcal{C}_{\mathcal{P}})$, which tell us how the tiles of $\mathbf{F}$ are filled.

- *Third step — Filling the tiles in* $\mathbf{D}$ *and* $\mathbf{E}$*:* In this final step, the master node proceeds to perform complete SVD decompositions for all matrices $\mathbf{F}_{\mathcal{P}}$ above, where each SVD decomposition takes the form $\mathbf{F}_{\mathcal{P}} = \mathbf{D}_{\mathcal{P}}\mathbf{E}_{\mathcal{P}}$, thus yielding all $\mathbf{D}_{\mathcal{P}}$ and $\mathbf{E}_{\mathcal{P}}$, $\forall\mathcal{P} \in \mathcal{C}$, as described in (5.60). Note that there are four different types of SVD decompositions, depending on whether $\mathcal{P}$ comes from $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ or $\mathcal{C}_4$. These filled tiles are placed inside $\mathbf{D}$ and $\mathbf{E}$ respectively (as illustrated in Figure 5.9), in accordance to the positioning steps in (5.69) and (5.70).

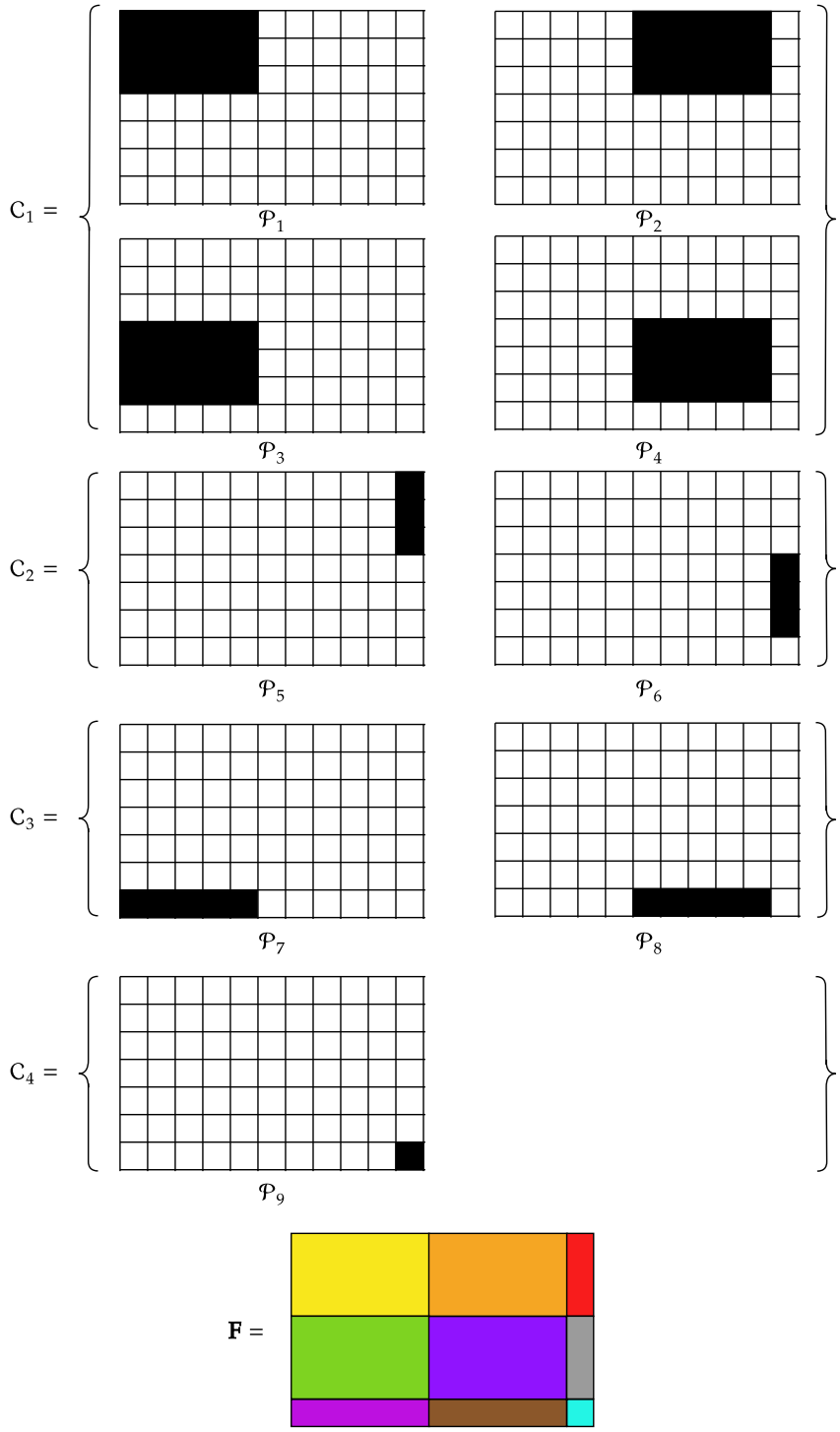


Figure 5.8: Corresponding to Example 4, the figure on the left represents in black the families of the equivalent classes (cf. (5.50)–(5.56)). The 9 equivalent classes (right) cover the entire **F**.

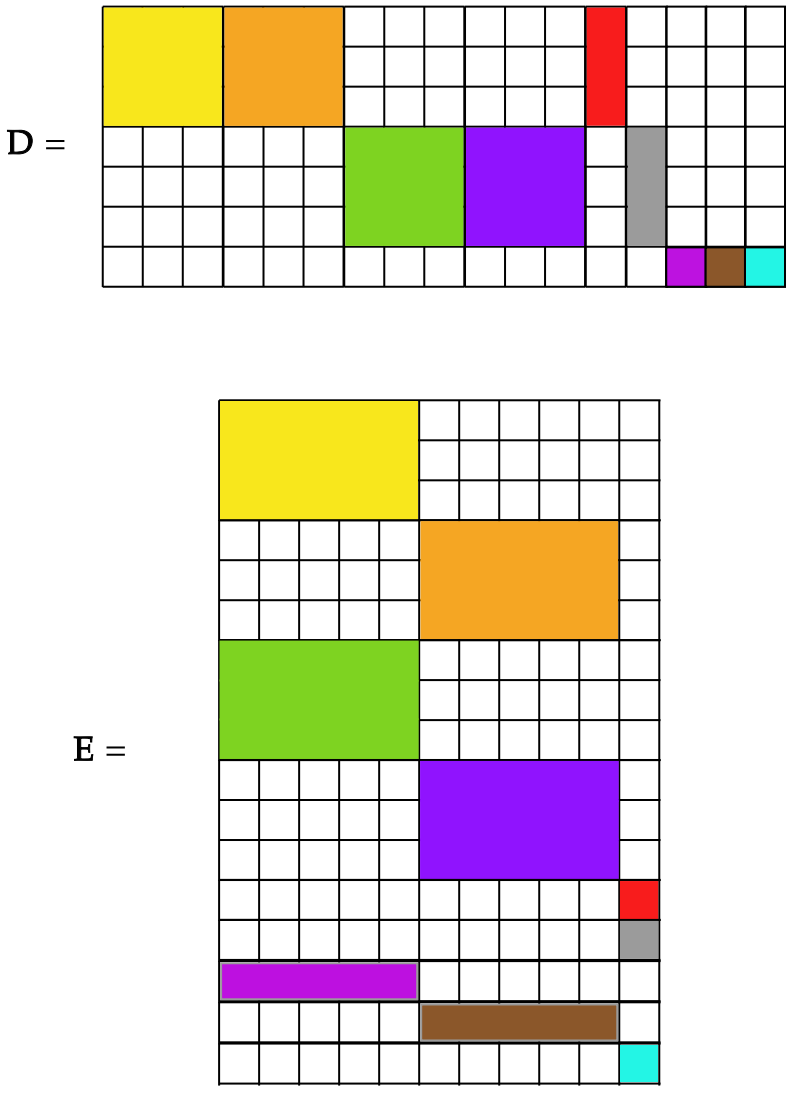


Figure 5.9: Creating our communication and computing matrices $\mathbf{D}, \mathbf{E}$ and applying the coordinates given in (5.62)–(5.65).

At this point, we calculate the number of used servers (cf. (4)) to be $N = 3 \times 4 + 1 + 1 \times 2 + 1 \times 2 = 17$.

□

We proceed with an additional example, now for the multi-shot case.

Example 5. We consider the same setting as in Example 1, with $K = 6$ users, $L = 10$ subfunctions, a communication cost $\Delta = 3$ and a computation cost of $\Gamma = 5$, but where now we consider each server to be able to communicate to $T = 2$ different sets of users, in two respective shots (one shot to one group of users, and another one to a potentially different group of users). Let us go through the steps described in our section above.

- *First step — Sizing and positioning the tiles of* $\mathbf{D}, \mathbf{E}$ *and of* $\mathbf{DE}$*:* The sizes and locations of the tiles of $\mathbf{D}, \mathbf{E}$ and $\mathbf{DE}$, are described by the tessellation pattern given directly from (5.50)–(5.56), as also illustrated on the left side of Figure 5.2. This pattern, in our simpler case here, entails tiles only from $\mathcal{C}_1$, where we find a total of $|\mathcal{C}_1| = \lfloor \frac{K}{\Delta} \rfloor \lfloor \frac{L}{\Gamma} \rfloor = 2 \times 2 = 4$ tiles, each of size 3×5, and each of maximum rank $r_{\mathcal{P}} = 3$ (cf. Definition 9). We again note that the pattern successfully covers $\mathbf{F}$.

- *Second step — Filling the non-zero tiles in* $\mathbf{DE}$*:* Exactly as in the single-shot case, again here the master node extracts the submatrices corresponding to each of the tiles as described in (5.59). This yields all the $\mathbf{F}_{\mathcal{P}}$ which are simply the filled tiles of $\mathbf{F}$.

- *Third step — Placing the filled cropped tiles* $\mathbf{D}_{\mathcal{P}}$ *and* $\mathbf{E}_{\mathcal{P}}$ *in* $\mathbf{D}$ *and* $\mathbf{E}$*:* Finally, the master SVD decomposes each $\mathbf{F}_{\mathcal{P}}$ as $\mathbf{F}_{\mathcal{P}} = \mathbf{D}_{\mathcal{P}}\mathbf{E}_{\mathcal{P}}$, which provides the required $\mathbf{D}_{\mathcal{P}}$ and $\mathbf{E}_{\mathcal{P}}$, $\forall \mathcal{P} \in \mathcal{C}$, as described in (5.60). Finally, these tiles are placed in $\mathbf{D}$ and $\mathbf{E}$ respectively, in direct accordance to the coordinates described in (5.62)–(5.65). This last part is illustrated in Figure 5.10.

For this setting, applying directly (5.71) tells us that we need $N = 8$ servers. We also observe that while the tiles of $\mathbf{F}$ remain the same as in the corresponding single shot case of Example1 (as also illustrated in Fig 5.2), indeed our tiles of $\mathbf{D}$ and $\mathbf{E}$ change[19], as illustrated in Figure 5.10. The same figure also

[19]We note here that if we had forced N down to $N = 6$ as in Example 1 (corresponding to the tiling in Figure 5.2), then some servers would violate the communication and computation cost constraints. For example, server 2 would have been forced to support a communication cost of 6 (6 links to different users), since the union of supports of the third and fourth column of $\mathbf{D}$ would have size 6. Similarly the corresponding $\mathbf{E}$ would entail a computation cost of 10 (10 subfunctions locally calculated), since the size of the union of the supports of the third and the fourth columns of $\mathbf{E}$ would have been 10.

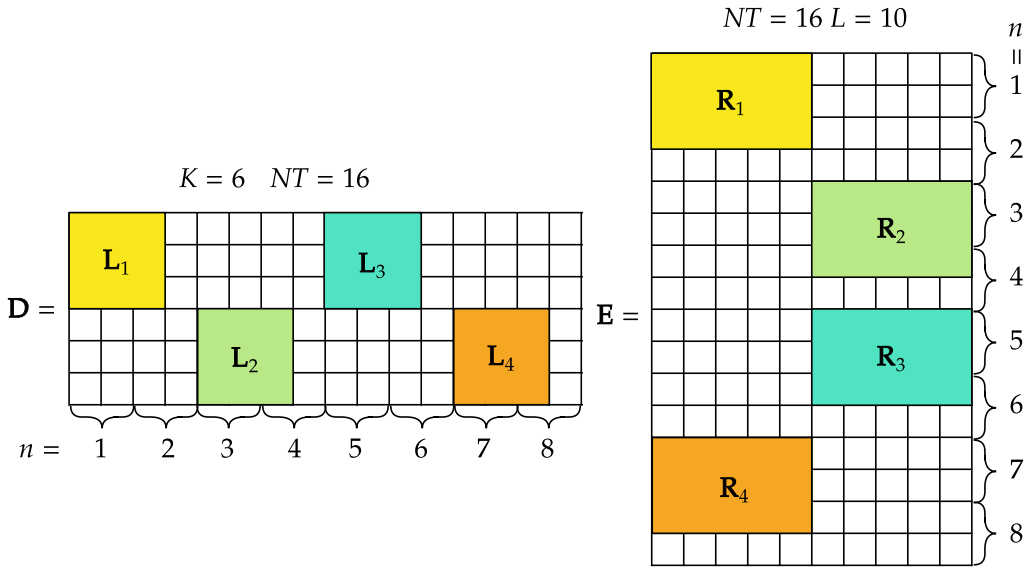


Figure 5.10: Corresponding to Example 5, this figure illustrates the tiling of $\mathbf{D}$ and $\mathbf{E}$ respectively with $\mathbf{D}_{\mathcal{P}} \in \mathbb{R}^{3\times 3}$ and $\mathbf{E}_{\mathcal{P}} \in \mathbb{R}^{3\times 5}$, for $\mathcal{P} \in \{P_1, P_2, P_3, P_4\}$. Guaranteeing $\mathbf{D}_{\mathcal{P}}\mathbf{E}_{\mathcal{P}} = \mathbf{F}_{\mathcal{P}} \in \mathbb{R}^{3\times 5}, \forall \mathcal{P} \in \{P_1, P_2, P_3, P_4\}$, in turn guarantees lossless function reconstruction. We can see that the union of the supports of $T = 2$ consecutive columns of $\mathbf{D}$ includes at most $\Delta = 3$ non-zero elements thus guaranteeing the communication constraint. We also see that the union of the supports of $T = 2$ consecutive rows of $\mathbf{E}$ includes at most $\Gamma = 5$ non-zero elements thus guaranteeing the computation constraint.

illustrates how each two consecutive columns of $\mathbf{D}$ and two consecutive rows of $\mathbf{E}$, correspond to a server. Note that although the rank of each tile is at most $r_{\mathcal{P}} = 3$, the number of servers associated with each tile is $\lceil \frac{r_{\mathcal{P}}}{T} \rceil = \lceil \frac{3}{2} \rceil = 2$. This reflects the fact that the second shot of the even-numbered servers remains unused, to avoid having a server be associated to two tiles, and thus to avoid violating the computation and communication constraints. □

5.8 Appendix:Proof of The Converse for Theorem 9

The converse that we provide here will prove that the scheme for the single shot case is exactly optimal when $\Gamma \geq \Delta, \Gamma | L, T | \Delta$ or $\Delta \geq \Gamma, \Delta | K, T | \Gamma$ and also the scheme for the multi-shot case is optimal when $T \geq \min(\Delta, \Gamma)$. We begin with three lemmas that will be useful later on. First, Lemma 15 will state the necessity of having a one-to-one correspondence between each rank-one contribution support[20] and each server, then Lemma 16 will state the necessity of having a tessellation pattern that covers the whole area of $\mathbf{F}$, and then Lemma 17 will elaborate more on size limits of each tile as a consequence of the communication and computation constraints. Lemma 18 then establishes the conceptual equivalence between Definition 3 and disjoint tiles. Lemma 19, lower bounds the number of tiles is each equivalence class by the number of subtiles (cf. Definition 19) corresponding to a tile. Subsequently, Lemma 20, gives the necessary number of subtiles, so that a covering scheme can be constructed, then via combining the last two of the aforementioned lemmas, we give a lower bound on the number of rank-one contribution supports for any covering scheme with proper sizes, and then using Lemma 15, we finalize our converse by giving a lower-bound on the number of servers.

When focusing on the general case of $T > 1$ in Section 5.8.2, we will substitute Lemma 15 with Lemma 21 to eventually show how each server's transmission (to one set of users) must correspond to a rank-one contribution support. Then using the same argument as in the single-shot case, we lower bound the necessary number of servers by the number $\frac{KL}{T \max(\Gamma, \Delta)}$. Then the lower bound is further tightened for the cases where $T \geq \min(\Delta, \Gamma)$, since then each tile corresponds to one server which will enable us, using Lemma 22, to obtain a new lower bound of $\lceil \frac{K}{\Delta} \rceil \lceil \frac{L}{\Gamma} \rceil$, which matches the value obtained by the achievable scheme.

5.8.1 Converse for The Single-Shot Case of $T = 1$

Before presenting the lemmas, let us recall that rank-one contribution supports were defined in Definition 4, that representative supports (i.e. tiles) were defined in Definition 6, as well as let us recall from the same definition that the collection of all classes is represented by $\mathcal{C}$.

The following lemma, while stating the obvious, will be useful in associating the number of servers to the number of rank-one contribution supports.

[20] Recall that the nth rank-one contribution support takes the form $\mathbf{S}_n(\mathbf{I}, \mathbf{J}) = \mathbf{I}(:, n)\mathbf{J}(n, :)$.

Lemma 15. For any $\mathbf{D}, \mathbf{E}$ with respective supports $\mathbf{I} = \text{supp}(\mathbf{D}) \in \{0,1\}^{K\times N}$ and $\mathbf{J} = \text{supp}(\mathbf{E}) \in \{0,1\}^{N\times L}$, there exists a one-to-one mapping between the server indices $n \in [N]$ and rank-one contribution supports $\mathbf{S}_n(\mathbf{I}, \mathbf{J})$.

Proof. The proof is direct by first recalling Definition 4 which, for any $n \in [N]$, says that $\mathbf{S}_n(\mathbf{I}, \mathbf{J}) = \mathbf{I}(:, n)\mathbf{J}(n, :)$, and then by recalling from (2.14),(2.16),(2.78) and (2.81) that, in the single-shot setting, each server $n \in \mathbb{N}$ corresponds to the nth column of $\mathbf{D}$ (itself corresponding to $\mathbf{I}(:, n)$) and the nth row of $\mathbf{E}$ (corresponding to $\mathbf{J}(n, :)$). □

We proceed with the next lemma, which simply says that every element of $\mathbf{F}$ must belong to at least one representative support (tile).

Lemma 16. In any lossless function reconstruction scheme corresponding to $\mathbf{DE} = \mathbf{F}$, for each $(i, j) \in [K] \times [L]$, there exists a class $\exists \mathcal{P} \in \mathcal{C}$ such that $(i, j) \in \mathcal{R}_{\mathcal{P}} \times \mathcal{C}_{\mathcal{P}}$.

Proof. The lemma aims to prove that there exists no element $\mathbf{F}(i, j)$ of $\mathbf{F}$ that has not been mapped to a tile $\mathbf{F}_{\mathcal{P}}$. We will prove that for each $(i, j) \in [K] \times [L]$ then $\exists \mathcal{P} \in \mathcal{C} \ : \ (i, j) \in \mathcal{R}_{\mathcal{P}} \times \mathcal{C}_{\mathcal{P}}$ and we will do so by contradiction. Let us thus assume that there exists $(i, j) \in [K] \times [L]$ such that $\nexists \mathcal{P} \in \mathcal{C} \ : \ (i, j) \in \mathcal{R}_{\mathcal{P}} \times \mathcal{C}_{\mathcal{P}}$, which would in turn imply — directly from (5.64), (5.65) — that $\mathbf{DE}(i, j) = 0$ as well as would imply the aforementioned fact that the non-assigned (by the process in (5.64), (5.65)) elements of $\mathbf{D}$ and $\mathbf{E}$, are zero. Now we see that

$$\begin{aligned}
\|(\mathbf{DE} - \mathbf{F})\mathbf{w}\|_2^2 &= \|\sum_{k=1}^{K}(\mathbf{DE} - \mathbf{F})(k, :)\mathbf{w}\|_2^2 \\
&= \sum_{k=1, k\neq i}^{K}\sum_{\ell=1, \ell\neq j}^{L}[(\mathbf{DE} - \mathbf{F})(k, \ell)\mathbf{w}(\ell)]^2 + (\mathbf{DE} - \mathbf{F})^2(i, j)\mathbf{w}^2(j) \\
&+ 2(\mathbf{DE} - \mathbf{F})(i, j)\mathbf{w}(j)\sum_{k=1, k\neq i}^{K}\sum_{\ell=1, \ell\neq j}^{L}(\mathbf{DE} - \mathbf{F})(k, \ell)\mathbf{w}(\ell) \\
&= \sum_{k=1, k\neq i}^{K}\sum_{\ell=1, \ell\neq j}^{L}[(\mathbf{DE} - \mathbf{F})(\ell, k)\mathbf{w}(\ell)]^2 \\
&+ \mathbf{F}^2(i, j)\mathbf{w}^2(j) - 2\mathbf{F}(i, j)\mathbf{w}(j)\sum_{k=1, k\neq i}^{K}\sum_{\ell=1, \ell\neq j}^{L}(\mathbf{DE} - \mathbf{F})(\ell, k)\mathbf{w}(\ell).
\end{aligned} \tag{5.72}$$

Let us now recall that lossless function reconstruction implies that $\|(\mathbf{DE} - \mathbf{F})\mathbf{w}\|_2^2 = 0$ for all $\mathbf{F} \in \mathbb{R}^{K\times L}$ and all $\mathbf{w} \in \mathbb{R}^L$. Under the special case of

$\mathbf{w}(\ell) = 0, \forall \ell \in [L]\backslash\{j\}$ and $\mathbf{w}(j)\mathbf{F}(i,j) \neq 0$, we see — directly from (5.72) — that $\|(\mathbf{DE}-\mathbf{F})\mathbf{w}\|_2^2 = \mathbf{F}^2(i,j)\mathbf{w}^2(j) \neq 0$, which contradicts the lossless assumption, thus concluding the proof of the lemma. □

The next lemma now limits the sizes of each tile.

Lemma 17. For any feasible scheme yielding $\mathbf{DE} = \mathbf{F}$, then each representative support $\mathcal{P} \in \mathcal{C}$ satisfies

$$0 < \|\mathbf{S}_{\mathcal{P}}(k,:)\|_0 \leq \Gamma, \forall k \in \mathcal{R}_{\mathcal{P}}\,, \quad 0 < \|\mathbf{S}_{\mathcal{P}}(:,l)\|_0 \leq \Delta, \forall l \in \mathcal{C}_{\mathcal{P}} \tag{5.73}$$

which means that each $\mathbf{S}_{\mathcal{P}}$ can have at most Γ non-zero elements in each row and Δ non-zero elements in each column.

Proof. We first recall from Definition 4 that $\text{supp}(\mathbf{D}) = \mathbf{I}, \text{supp}(\mathbf{E}) = \mathbf{J}$. We also recall that $\max_{n\in[N]} |\cup_{t=1}^{T} \text{supp}(\mathbf{D}(:,(n-1)T+t))| \leq \Delta$ and $\max_{n\in[N]} |\cup_{t=1}^{T}$ $\text{supp}(\mathbf{E}((n-1)T+t,:))| \leq \Gamma$ (cf. (5.36),(5.37)) must hold for any feasible scheme. Hence for all $n \in [NT]$, we have that $\|\mathbf{I}(:,n)\|_0 \leq \Delta, \|\mathbf{J}(n,:)\|_0 \leq \Gamma$, and consequently since $\mathbf{S}_n = \mathbf{I}(:,n)\mathbf{J}(n,:)$ (cf. Definition 4), we must have that $\|\mathbf{S}_n(k,:)\|_0 \leq \Gamma, \forall k \in [K]\,, \|\mathbf{S}_n(:,l)\|_0 \leq \Delta, \forall l \in [L]\,\forall n \in [N]$. Then from Definition 6, we see that for all $\mathcal{P} \in \mathcal{C}$, there exists an $n \in [N]$ such that $\mathbf{S}_n = \mathcal{S}_{\mathcal{P}}$. Note that the lower bound follows from Definition 7 where $\mathcal{R}_{\mathcal{P}}, \mathcal{C}_{\mathcal{P}}$ are defined. □

Continuing with the main proof, in order to relate the notion of tiles to the rank-one contribution supports, we need first to define the notion of sub-tiles, where each entry of a sub-tile is a matrix coordinate. We also recall that $\mathcal{R}_{\mathcal{P}}$ and $\mathcal{C}_{\mathcal{P}}$ are respectively the row and column indices of tile $\mathcal{P}$, as given in Definition 7, as well as note that we here regard $\mathcal{R}_{\mathcal{P}}$ and $\mathcal{C}_{\mathcal{P}}$ as arbitrarily ordered sets.

Definition 10. For each tile $\mathcal{P}$, the set of (at most) Δ horizontal sub-tiles takes the form

$$\mathcal{H}_{\mathcal{P},h_{\mathcal{P}}} \triangleq \{(\mathcal{R}_{\mathcal{P}}(h_{\mathcal{P}}), j) | j \in \mathcal{C}_{\mathcal{P}},\ h_{\mathcal{P}} \in [\Delta]\} \tag{5.74}$$

while the set of (at most) Γ vertical sub-tiles takes the form

$$\mathcal{V}_{\mathcal{P},v_{\mathcal{P}}} \triangleq \{(i, \mathcal{C}_{\mathcal{P}}(v_{\mathcal{P}})) | i \in \mathcal{R}_{\mathcal{P}},\ v_{\mathcal{P}} \in [\Gamma]\}. \tag{5.75}$$

In the above, $\mathcal{R}_{\mathcal{P}}(h_{\mathcal{P}})$ represents the $h_{\mathcal{P}}$-th element of $\mathcal{R}_{\mathcal{P}}$, and similarly $\mathcal{C}_{\mathcal{P}}(v_{\mathcal{P}})$ represents the $v_{\mathcal{P}}$-th element of $\mathcal{C}_{\mathcal{P}}$. We also note (cf. Lemma 17) that each horizontal (resp. vertical) sub-tile can have at most Γ (resp. Δ) elements. We also need the following function definition.

Definition 11. For $\mathcal{G}$ being the power set of all horizontal and vertical subtiles $\{\mathcal{H}_{\mathcal{P},h_\mathcal{P}}, \mathcal{V}_{\mathcal{P},v_\mathcal{P}}\}_{\mathcal{P}\in\mathcal{C},h_\mathcal{P}\in[\Delta],v_\mathcal{P}\in[\Gamma]}$, we define the function $\Phi(.) : \{0,1\}^{K\times L} \to \mathcal{G}$ as

$$\Phi(\mathbf{S}_\mathcal{P}) \triangleq \begin{cases} \{\mathcal{H}_{\mathcal{P},h_\mathcal{P}} | h_\mathcal{P} \in [\Delta]\}, & \text{if } |\mathcal{R}_\mathcal{P}| \leq |\mathcal{C}_\mathcal{P}|, \\ \{\mathcal{V}_{\mathcal{P},v_\mathcal{P}} | v_\mathcal{P} \in [\Gamma]\}, & \text{if } |\mathcal{R}_\mathcal{P}| > |\mathcal{C}_\mathcal{P}|. \end{cases} \tag{5.76}$$

We now proceed to bound the number of rank-one contribution supports, and we do so under our previously stated assumption of disjoint supports (cf. Definition 3), which is equivalent to disjoint tiles assumptions via the following Lemma,

Lemma 18. For two matrices $\mathbf{D}, \mathbf{E}$, the representative supports $\{\mathbf{S}_{\mathcal{P}_i}\}_{i=1}^m$ of $\mathbf{DE}$ are disjoint (i.e., $\mathbf{S}_{\mathcal{P}_i} \cap \mathbf{S}_{\mathcal{P}_j} = \mathbf{0},\ j \neq i$) if and only if $\mathbf{D}$ and $\mathbf{E}$ accept the disjoint support assumption of Definition 3.

Proof. Assuming that $\mathbf{D} \in \mathbb{R}^{K\times NT}, \mathbf{E} \in \mathbb{R}^{NT\times L}$ abide by the disjoint support assumption from Definition 3, then for all $i, i' \in [NT]$, we have that either $\text{Supp}(\mathbf{D}(:,i)\mathbf{E}(i,:)) = \text{Supp}(\mathbf{D}(:,i')\mathbf{E}(i',:))$ or that $\text{Supp}(\mathbf{D}(:,i)\mathbf{E}(i,:)) \cap \text{Supp}(\mathbf{D}(:,i')\mathbf{E}(i',:)) = \emptyset$. This in turn implies that for $\mathbf{I} = \text{Supp}(\mathbf{D}) \in \{0,1\}^{K\times NT}, \mathbf{J} = \text{Supp}(\mathbf{E}) \in \{0,1\}^{NT\times L}$, then either $\mathbf{I}(:,i)\mathbf{J}(i,:) = \mathbf{I}(:,j)\mathbf{J}(j,:)$ or $\mathbf{I}(:,i)\mathbf{J}(i,:) \cap \mathbf{I}(:,j)\mathbf{J}(j,:) = \mathbf{0}_{K\times L}$, which in turn yields the assumption in Definition 5 of disjoint representative support equivalence classes.

In reverse, if $\mathbf{DE}$ accepts the disjoint representative support assumption, and if $\mathcal{C} = \{\mathcal{P}_1, \ldots, \mathcal{P}_m\}$ is the collection of the equivalence classes, then $\forall \mathcal{P}, \mathcal{P}' \in \mathcal{C}, \mathcal{C}_\mathcal{P} = \mathcal{C}_{\mathcal{P}'}$ or $\mathcal{C}_\mathcal{P} \cap \mathcal{C}_{\mathcal{P}'} = \emptyset$ and similarly $\forall \mathcal{P}, \mathcal{P}' \in \mathcal{C}, \mathcal{R}_\mathcal{P} = \mathcal{R}_{\mathcal{P}'}$ or $\mathcal{R}_\mathcal{P} \cap \mathcal{R}_{\mathcal{P}'} = \emptyset$, which in turn implies that $\forall i, i' \in [NT]$ then $\text{supp}(\mathbf{D}(:,i)) = \text{supp}(\mathbf{D}(:,i'))$ or $\text{supp}(\mathbf{D}(:,i)) \cap \text{supp}(\mathbf{D}(:,i')) = \emptyset$, as well implies that $\forall i, i' \in [NT]$ then $\text{supp}(\mathbf{E}(i,:)) = \text{supp}(\mathbf{E}(i',:))$ or $\text{supp}(\mathbf{E}(i,:)) \cap \text{supp}(\mathbf{E}(i',:)) = \emptyset$. Consequently, if $\text{supp}(\mathbf{D}(:,i)) = \text{supp}(\mathbf{D}(:,i'))$ and $\text{supp}(\mathbf{E}(i,:)) = \text{supp}(\mathbf{E}(i,:))$ both hold, then $\text{supp}(\mathbf{D}(:,i)\mathbf{E}(i,:)) = \text{supp}(\mathbf{D}(:,i')\mathbf{E}(i',:))$ or other wise $\text{supp}(\mathbf{D}(:,i)\mathbf{E}(i,:)) \cap \text{supp}(\mathbf{D}(:,i')\mathbf{E}(i',:)) = \emptyset$, which in turn yields the assumption in Definition 3 that $\mathbf{D}$ and $\mathbf{E}$ satisfy the disjoint support assumption. □

We proceed with the next lemma.

Lemma 19. In any lossless function reconstruction scheme corresponding to $\mathbf{DE} = \mathbf{F}$, the number of rank-one contribution supports $|\mathcal{P}|$ in each class $\mathcal{P}$, satisfies $|\mathcal{P}| \geq |\Phi(\mathbf{S}_\mathcal{P})|$.

Proof. Recall from Definition 3 and Lemma 18 that in the context of lossless schemes, each representative support is disjoint. Then we can see that

$$|\mathcal{P}| \geq \min(\min(|\mathcal{R}_\mathcal{P}|, |\mathcal{P}|), \min(|\mathcal{C}_\mathcal{P}|, |\mathcal{P}|)) \tag{5.77}$$

$$\overset{(a)}{=} \min(\text{rank}(\mathbf{D}_{\mathcal{P}}), \text{rank}(\mathbf{E}_{\mathcal{P}})) \overset{(b)}{\geq} \text{rank}(\mathbf{F}_{\mathcal{P}}) \overset{(c)}{=} \min(|\mathcal{R}_{\mathcal{P}}|, |\mathcal{C}_{\mathcal{P}}|) = r_{\mathcal{P}} \tag{5.78}$$

where (a) follows from Definition 3 and Lemma 18 which tells us that in the context of lossless schemes then each representative support is disjoint which in turn tells us that (5.60) applies in which case we have $\mathbf{D}(\mathcal{R}_{\mathcal{P}}, \mathcal{P}) = \mathbf{D}_{\mathcal{P}}$ and $\mathbf{E}(\mathcal{P}, \mathcal{C}_{\mathcal{P}}) = \mathbf{E}_{\mathcal{P}}$. Subsequently (b) follows from the fact that $\mathbf{D}_{\mathcal{P}}\mathbf{E}_{\mathcal{P}} = \mathbf{F}_{\mathcal{P}}$, (c) follows from the dimensionality of $\mathbf{F}_{\mathcal{P}}$, and the last equality holds from the definition of $r_{\mathcal{P}}$.

□

We now proceed with a lemma that lower bounds the minimum number of horizontal or vertical sub-tiles needed[21] to cover $\mathbf{F} \in \mathbb{R}^{K \times L}$.

Lemma 20. For any single-shot lossless function reconstruction scheme, and for the corresponding $\mathbf{DE} = \mathbf{F}$ decomposition, the minimum number of sub-tiles needed to cover $\mathbf{F}$ is at least $\frac{KL}{\max(\Delta,\Gamma)}$.

Proof. Suppose first that $\Gamma \geq \Delta$ and consider a scheme that covers the entire $\mathbf{F}$, with m_1 horizontal sub-tiles and m_2 vertical sub-tiles. We wish to show that $m_1 + m_2 \geq \frac{KL}{\max(\Delta,\Gamma)}$. To see this, we first note that since there is no intersection between each of the tiles (cf. Definition 3 and Lemma 18), and since there is no intersection between each sub-tile inside a tile (Definition 11), then we can conclude that for any $\mathcal{P}, \mathcal{P}' \in \mathcal{C}$ and any $\mathcal{S} \in \Phi(\mathbf{S}_{\mathcal{P}}), \mathcal{S}' \in \Phi(\mathbf{S}_{\mathcal{P}'})$, we must have $\mathcal{S} \cap \mathcal{S}' = \emptyset$. This in turn implies that the sub-tiles can now cover at most $m_1\Gamma + m_2\Delta$ elements of $\mathbf{F}$, which in turn means that $m_1\Gamma + m_2\Delta \geq KL$, which means that $\frac{KL}{\Gamma} - m_2\frac{\Delta}{\Gamma} \leq m_1$, which means that $\frac{KL}{\Gamma} + (1 - \frac{\Delta}{\Gamma})m_2 \leq m_1 + m_2$. Since $\frac{\Delta}{\Gamma} \leq 1$, we have that $\frac{KL}{\Gamma} \leq \frac{KL}{\Gamma} + (1 - \frac{\Delta}{\Gamma})m_2$, which directly tells us that $m_1 + m_2 \geq \frac{KL}{\Gamma}$. This concludes the proof for the case of $\Gamma \geq \Delta$. The same process follows directly also for the case of $\Gamma \leq \Delta$, thus concluding the proof. □

At this point we can combine our results. We know from Definitions 10,11 and from Lemma 20 that $\sum_{\mathcal{P}\in\mathcal{C}} |\Phi(\mathbf{S}_{\mathcal{P}})| > \frac{KL}{\max(\Gamma,\Delta))}$, while we know from Lemma 11 that $\sum_{\mathcal{P}\in\mathcal{C}} |\mathcal{P}| \geq \sum_{\mathcal{P}\in\mathcal{C}} |\Phi(\mathbf{S}_{\mathcal{P}})|$. Now by recalling that $\sum_{\mathcal{P}\in\mathcal{C}} |\mathcal{P}|$ is the number of rank-one contribution supports, and by recalling from Lemma 15 that each rank-one contribution support corresponds to a server in any lossless scheme, we can use the lower bound on the number of sub-tiles in Lemma 20 as a lower bound on the number of servers, allowing us to thus conclude that $N_{opt} \geq \frac{KL}{\max(\Delta,\Gamma)}$, thus concluding the proof of our converse for the single-shot case.

[21] To "cover" in this context means that every element of $\mathbf{F}$ has to be in at least one representative support.

5.8.2 The General Multi-Shot Case of $T > 1$

We begin with our first lemma for this case.

Lemma 21. To guarantee lossless function reconstruction with constraints Δ, Γ, any T-shot scheme with computation and communication matrices $\mathbf{D}, \mathbf{E}$, must associate each server transmission (shot) to a unique rank-one contribution support $\mathbf{S}_n(\mathbf{I}, \mathbf{J}), n \in [NT]$, where $\mathbf{I} \in \{0,1\}^{K \times NT}$ and $\mathbf{J} \in \{0,1\}^{NT \times L}$ are the support constraints of $\mathbf{D}$ and $\mathbf{E}$ respectively as in Definition 4.

Proof. The proof follows directly the proof steps of Lemma 15, after noting that in our current multi-shot setting, each column of $\mathbf{D}$ and row of $\mathbf{E}$ correspond to a single transmission by a unique server. □

We use Lemma 21 to recall that NT is equal to the number of rank-one contribution supports. Then we apply the same proof steps found at the last paragraph of Appendix 5.8.1 to see that the number of rank-one contribution supports is no less than $\frac{KL}{\max(\Gamma,\Delta)}$, which in turn implies that $N \geq \frac{KL}{T\max(\Delta,\Gamma)}$. This holds for all T. The following lemma will allow us to provide a tighter bound, for the case of $T \geq \min(\Delta, \Gamma)$. After proving the following lemma, we will complete the proof of the converse for the multi-shot case. In the following, we recall that the term “cover" in our context means that every element of $\mathbf{F}$ has to be in at least one representative support.

Lemma 22. For lossless function reconstruction, the corresponding $\mathbf{DE} = \mathbf{F}$ decomposition needs at least $\lceil \frac{K}{\Delta} \rceil \lceil \frac{L}{\Gamma} \rceil$ representative supports (tiles) to cover the entire matrix $\mathbf{F} \in \mathbb{R}^{K \times L}$.

Proof. Let us first recall from Lemma 16 that (the tiles corresponding to) any lossless optimal scheme must cover $\mathbf{F}$. Let $\mathcal{C}$ be the collection of classes of an optimal scheme, and let $\mathbf{I} = \text{Supp}(\mathbf{D}), \mathbf{J} = \text{Supp}(\mathbf{E})$ and $\text{Supp}(\mathbf{F}) = \text{Supp}(\mathbf{DE}) = \mathbf{IJ}$.

Let us first consider the simplest instance where $\Delta | K, \Gamma | L$, in which case we first note that $\mathbf{F}$ has a total of KL elements, and that each tile can cover at most $\Delta\Gamma$ elements of $\mathbf{F}$ (cf. Lemma 17), which in turn means that the minimum number of disjoint covering tiles is simply $\frac{KL}{\Delta\Gamma}$ (cf. Lemma 3), which then completes the proof of the lemma for this instance. Subsequently, for the case of $\Delta \nmid K, \Gamma | L$, we first split the rows of $\mathbf{F}$ into an upper part $\mathcal{R}_1$ with $K_1 = \lfloor \frac{K}{\Delta} \rfloor \Delta$ rows, and a lower part $\mathcal{R}_2$ with $K_2 = \text{mod}(K, \Delta) = K - K_1$ rows. Here $\mathcal{R}_1$ and $\mathcal{R}_2$ are the corresponding row indices. Using the argument from the previous case of $\Delta | K, \Gamma | L$, we can see that the upper part $\mathbf{F}(\mathcal{R}_1, :)$ needs at least $\lfloor \frac{K}{\Delta} \rfloor \frac{L}{\Gamma}$ tiles to complete covering. For the lower submatrix $\mathbf{F}(\mathcal{R}_2, :)$,

we know that having $\lfloor \frac{K}{\Delta} \rfloor \frac{L}{\Gamma}$ tiles is not enough since we would only be able to cover $K_1 L = \lfloor \frac{K}{\Delta} \rfloor \Delta L$ elements, thus leaving L uncovered elements in each row in $\mathcal{R}_2$. Since each tile can cover at most Γ elements in each column (cf. Lemma 17), there has to be at least $\frac{L}{\Gamma}$ additional tiles to cover all of $\mathbf{F}$, which implies a total of $\lfloor \frac{K}{\Delta} \rfloor \frac{L}{\Gamma} + \frac{L}{\Gamma} = \lceil \frac{K}{\Delta} \rceil \frac{L}{\Gamma}$ tiles, which proves our claim for the case of $\Delta \nmid K, \Gamma | L$. The third case of $\Delta | K, \Gamma \nmid L$ is similar.

We now consider the more involved, general case of $K = q_1 \Delta + r_1, L = q_2 \Gamma + r_2$, where $q_1, q_2, r_1, r_2 \in \mathbb{N} \cup \{0\}, 0 < r_1 \leq \Delta, 0 < r_2 \leq \Gamma$. We first recall from Lemma 17 that each tile $\mathcal{P} \in \mathcal{C}$ can cover at most Γ elements of a row of $\mathbf{F}$, which in turn implies that $\|\mathbf{S}_{\mathcal{P}}(k,:) \cap \text{Supp}(\mathbf{F})(k',:)\|_0 \leq \Gamma, \ \forall k, k' \in [N], \forall \mathcal{P} \in \mathcal{C}$. Similarly, we also know that each tile can cover up to Δ elements in one column of $\mathbf{F}$, which in turn implies that $\|\mathbf{S}_{\mathcal{P}}(:,\ell) \cap \text{Supp}(\mathbf{F})(:,\ell')\|_0 \leq \Delta, \ \forall \ell, \ell' \in [L], \forall \mathcal{P} \in \mathcal{C}$.

We will employ a two-dimensional version of the pigeon-hole principle. In reference to this principle, our "pigeon" here will correspond to an element of $\mathbf{F}$, a "hole" will correspond to a tile, while now also each row and column of a tile (which corresponds to a horizontal and vertical sub-tile (cf.Definition 10)) will correspond to a 'sub-hole'. Let us recall from Definition 10 that each tile consists of Δ horizontal sub-tiles, and then recall from Lemma 17 that each such horizontal sub-tile (sub-hole) has up to Γ elements[22]. Similarly we recall that each tile consists of Γ vertical sub-tiles, each having at most Δ elements. For any scheme whose corresponding $\mathbf{D}, \mathbf{E}$ satisfy the disjoint support assumption (Definition 3 and Lemma 18), we now define a mapping function $\Xi_{\mathbf{D},\mathbf{E}} : [K] \times [L] \to \mathcal{C} \times \mathcal{C}_h \times \mathcal{C}_v$, where $\mathcal{C}, \mathcal{C}_h, \mathcal{C}_v$ are respectively the set of all tiles, horizontal sub-tiles, and vertical sub-tiles involved in the scheme. This function is here defined to take the form

$$\Xi_{\mathbf{D},\mathbf{E}}((i,j)) \triangleq \{(\mathcal{P}, h_{\mathcal{P}}, v_{\mathcal{P}})\}, (i,j) \in [K] \times [L] \tag{5.79}$$

where the input (i,j) is the location of the element in $\mathbf{F}$, and where the output consists of the tile $\mathcal{P}$ that covers (i,j), and the corresponding sub-tile index $h_{\mathcal{P}} \in [\Delta]$ of the horizontal sub-tile that covers (i,j), as well as sub-tile index $v_{\mathcal{P}} \in [\Gamma]$ of the vertical sub-tile again covering (i,j). For ease of reading, we recall here from Definition 10 that horizontal and vertical sub-tiles are defined as

$$\mathcal{H}_{\mathcal{P},h_{\mathcal{P}}} = \{(\mathcal{R}_{\mathcal{P}}(h_{\mathcal{P}}), j) | j \in \mathcal{C}_{\mathcal{P}}\} \tag{5.80}$$

$$\mathcal{V}_{\mathcal{P},v_{\mathcal{P}}} = \{(i, \mathcal{C}_{\mathcal{P}}(v_{\mathcal{P}})) | i \in \mathcal{R}_{\mathcal{P}}\} \tag{5.81}$$

where $\mathcal{R}_{\mathcal{P}}$ and $\mathcal{C}_{\mathcal{P}}$, are defined in Definition 7. We also recall that we regard $\mathcal{R}_{\mathcal{P}}$ and $\mathcal{C}_{\mathcal{P}}$ as ordered sets, with an arbitrary order, where $\mathcal{R}_{\mathcal{P}}(h_{\mathcal{P}})$ is the

[22]Recall that a tile, and by extension, a sub-tile, is a set of indices.

$h_{\mathcal{P}}$-th element of $\mathcal{R}_{\mathcal{P}}$ and where similarly $\mathcal{C}_{\mathcal{P}}(v_{\mathcal{P}})$ is the $v_{\mathcal{P}}$-th element of $\mathcal{C}_{\mathcal{P}}$. Now we claim that

$$\forall i \in [K] : |\{\mathcal{P}|\Xi_{\mathbf{D},\mathbf{E}}(i,j) = \{(\mathcal{P}, h_{\mathcal{P}}, v_{\mathcal{P}}), j \in [L]\}| \geq q_2 + 1 \tag{5.82}$$

which says that the intersection of each row of $\mathbf{F}$ with any specific tile, is at least $q_2 + 1$. This is obvious because if this intersection was less than $q_2 + 1$ then — as a consequence of the pigeon-hole principle, where again each element of $\mathbf{F}(i,:)$ is our "pigeon" and each horizontal sub-tile as a sub-hole — there would exist at least one sub-hole covering at least P_1 elements, where

$$P_1 = \lceil \frac{L}{q_2} \rceil = \lceil \frac{q_2\Gamma + r_2}{q_2} \rceil = \Gamma + 1 \tag{5.83}$$

which would though contradict Lemma 17 which guarantees that there can exist no tile covering more than Γ elements in a row, which simply translates to having no horizontal sub-tile with more than Γ elements.

With (5.82) in place, we now conclude that[23] in each row, at least $q_2 + 1$ horizontal sub-tiles reside. Recall that by Lemma 17, each horizontal sub-tile in a tile can cover at most Γ elements. Note also that by Definition 10, for any given row of $\mathbf{F}$, we cannot encounter two or more sub-tiles from the same tile (they must be from different tiles). In this context, for any given row i of $\mathbf{F}$, let $(i,1), (i,2), \cdots, (i,u_i)$ be the indices of the corresponding horizontal sub-tiles that reside in that row, from left to right. Recall from (5.82) that $u_i \geq q_2 + 1$.

We proceed to provide a similar enumeration, now though for vertical sub-tiles.

We first note from Lemma 17 that each column of $\mathbf{F}$ can intersect at most Δ *horizontal* sub-tiles. Similar to before, we can also see that

$$\forall j \in [L] : |\{\mathcal{P}|\Xi_{\mathbf{D},\mathbf{E}}(i,j) = \{(\mathcal{P}, h_{\mathcal{P}}, v_{\mathcal{P}})\}, i \in [K]\}| \geq q_1 + 1 \tag{5.84}$$

which says that the intersection of each column of $\mathbf{F}$ with any specific tile, entails at least $q_1 + 1$ tiles, because if this intersection was less than $q_1 + 1$ then — as a consequence of the pigeon-hole principle, where now each horizontal sub-tile is our "pigeon" and each tile is a hole — there would exist at least one sub-hole covering at least P_2 elements, where

$$P_2 = \lceil \frac{K}{q_1} \rceil = \lceil \frac{q_1\Delta + r_1}{q_1} \rceil = \Delta + 1 \tag{5.85}$$

[23]For the case where $q_2 = 0$, Lemma 16 simply guarantees that each row has to be at least in one tile.

which though would contradict Lemma 17 which now guarantees that there can exist no tile covering more than Δ elements in a column, which now simply translates to having no vertical sub-tile intersecting with more than Γ horizontal sub-tiles. We can now conclude that[24] each row of $\mathbf{F}$ intersects with at least q_2+1 tiles. After recalling from Definition 10 that two horizontal sub-tiles of the same tile have the same beginning and end[25], we can conclude that two horizontal sub-tiles (i', j') and (i'', j'') (where $j' \neq j''$), cannot be found in the same tile, because their respective column indices differ by at least one element. We will now only consider the parts of $\mathbf{F}$ that correspond to horizontal sub-tiles with indices restricted to the set $[K] \times [q_2 + 1]$. Note that this "pruning" is in line with our effort to lower bound the number of required tiles for covering $\mathbf{F}$. We proceed by combining the fact that each column of $\mathbf{F}$ intersects with at least $q_1 + 1$ tiles (cf. (5.84)), together with the aforementioned fact that the column coordinates of two horizontal sub-tiles of the same tile are identical. Thus we can finally adopt the tile indexing $(1, \hat{j}), (2, \hat{j}), \ldots, (v_{\hat{j}}, \hat{j})$ for all $\hat{j} \in [q_2 + 1]$. The fact that this indexing applies to any conceivable covering tessellation scheme, allows us to conclude that any covering scheme must entail at least $\sum_{\hat{j}=1}^{q_2+1} v_{\hat{j}}$ tiles. Noting now from (5.82), we have that $v_{\hat{j}} \geq q_1 + 1, \forall \hat{j} \in [q_2 + 1]$. This fact allows us to conclude that there must exist at least $(q_1 + 1)(q_2 + 1)$ tiles, which completes the proof after recalling that $K = q_1\Delta + r_1, L = q_2\Gamma + r_2$ which in turn says that $q_1 = \lceil \frac{K}{\Delta} \rceil, q_2 = \lceil \frac{L}{\Gamma} \rceil$. □

With the above lemma in place, focusing again on the particular case of $T \geq \min(\Delta, \Gamma)$, we can now tighten the bound on N, from $N \geq \frac{KL}{T \max(\Delta,\Gamma)}$ to $N \geq \lceil \frac{K}{\Delta} \rceil \lceil \frac{L}{\Gamma} \rceil$. We see this by first noting that in our case of $T \geq \min(\Delta, \Gamma)$ each tile corresponds to a server, and then by combining this with Lemma 21 which additionally tells us that each server corresponds to T distinct rank-one contribution supports. Furthermore, we also know from Definition 20 and Lemma 19 that $\min(\Delta, \Gamma) = r_{\mathcal{P}} = |\Phi(\mathbf{S}_{\mathcal{P}})|$. Finally, from the fact that no server can be associated with two different tiles[26], and directly from Definition 11, we can conclude that the optimal number of servers will be the minimum number of covering tiles, which was shown Lemma 22 to be equal to $\lceil \frac{K}{\Delta} \rceil \lceil \frac{L}{\Gamma} \rceil$. This concludes the proof that $N \geq \lceil \frac{K}{\Delta} \rceil \lceil \frac{L}{\Gamma} \rceil$ for $T \geq \min(\Delta, \Gamma)$. This concludes the converse for the multi-shot case.

[24]For the case where $q_1 = 0$. Lemma 16 simply guarantees that each column has to be at least in one tile.

[25]More rigorously, we can rephrase the above by saying 'after recalling from Definition 10 that $\forall h_{\mathcal{P}}, h'_{\mathcal{P}} \in [\Delta] : \{j|(i,j) \in \mathcal{H}_{\mathcal{P},h_{\mathcal{P}}}\} = \{j|(i,j) \in \mathcal{H}_{\mathcal{P},h'_{\mathcal{P}}}\}$'.

[26]This was shown in appendix Section 5.7.1, to be a necessary condition for guaranteeing the Γ and Δ constraints.

5.8.3 Proof of Corollary 3

Our aim here is to show that for $T < \min(\Delta, \Gamma)$, then the achievable N (and thus the achievable rate) in (5.38) is at most a multiplicative factor of 8 from the corresponding converse expression in (5.39). The proof can be derived from the following sequence of expressions

$$\begin{aligned}
\frac{N_{\text{upper}}}{N_{\text{Lower}}} &\leq \frac{(\min(\Delta,\Gamma)/T+1)(KL/(\Delta\Gamma)+L/\Gamma+K/\Delta)+\min(\Delta,\Gamma)/T)}{KL/(T\max(\Gamma,\Delta))} \\
&= 1+\frac{T\max(\Delta,\Gamma)}{\Delta\Gamma}+\frac{T\max(\Delta,\Gamma)\min(\Delta,\Gamma)}{TK\Gamma} \\
&+\frac{T\max(\Gamma,\Delta)L}{KL\Gamma}+\frac{T\max(\Delta,\Gamma)\min(\Delta,\Gamma)}{T\Delta L}+\frac{T\max(\Delta,\Gamma)}{L\Delta} \\
&+\frac{T\max(\Delta,\Gamma)\min(\Delta,\Gamma)}{TKL}+\frac{T\max(\Delta,\Gamma)}{KL} \\
&\leq 1+4\frac{T}{\min(\Delta,\Gamma)}+\delta+\gamma+\delta\gamma<8
\end{aligned}$$

where the final answer results by noting that $T < \min(\Delta, \Gamma), \Delta \leq K, \Gamma \leq L$.

Chapter 6

Lossy Tessellated Distributed Computing

6.1 Introduction

Another important factor of Tesselated Distributed Computing introduced in Chapter 5 pertains to computational accuracy and the ability to recover desired functions with reduced error or distortion. There is indeed a variety of techniques dedicated to increasing accuracy (cf. [56]–[58], [112]–[126]), such as for example the *sketching technique* [113], [116], [126] which utilized a randomized linear algebraic approach to compute an approximation of the multiplication of two massive matrices (often by approximating input matrices by multiplying them with a random matrix having certain properties), as well as successive approximation coding techniques (cf. [121]) which can tradeoff accuracy and speed, allowing for better approximations and increased accuracy over time.

In this chapter we investigate this triptych between accuracy and communication and computation costs, lies at the center of distributed computing. We here explore this triptych using as building blocks various tools such as truncated SVD (singular value decomposition), tessellations, and low-rank matrix approximation techniques[139].

We now examine the scenario of lossy function reconstruction, where it is possible for the reconstruction error to surpass zero. Our primary goal is to constrain and quantify the error that occurs when the available system resources, represented by γ, δ, N, are insufficient for achieving lossless reconstruction. There are instances where certain levels of reconstruction error can be accepted, prompting us to explore the potential benefits and savings associated with tolerating such error.

Let us recall that our aim is to approach, under the available γ, δ, N, T, the minimum (cf. (5.34),(5.35))

$$\hat{\epsilon} = \frac{1}{KL}\mathbb{E}_{\mathbf{F},\mathbf{w}}\{\min_{\mathbf{D},\mathbf{E}} \sum_{k=1}^{K} |F_k - F_k^{'}|^2\} = \frac{1}{KL}\mathbb{E}_{\mathbf{F},\mathbf{w}}\{\min_{\mathbf{D},\mathbf{E}} \|\mathbf{DE} - \mathbf{F}\|_F^2\}$$

which means that our distributed computing challenge ultimately reduces to the problem of sparse matrix factorization, as seen below

$$\min_{\mathbf{D},\mathbf{E}} \|\mathbf{DE} - \mathbf{F}\|_F^2. \tag{6.1}$$

Solving this problem optimally has been a persistent challenge, often resisting a straightforward characterization of the most efficient performance. Consequently, we will turn to scaling asymptotically with parameters of interest that grow with N, as well as adopt a statistical approach. Instead of offering guarantees for each individual matrix $\mathbf{F}$, we will provide assurances across the ensemble of matrices $\mathbf{F}$ under certain basic assumptions. Adopting the specific vector-wise metrics of sparsity γ, δ, our aim will be to bound the average optimal error $\hat{\epsilon} = \frac{1}{KL}\mathbb{E}_{\mathbf{F},\mathbf{w}}\{\min_{\mathbf{D},\mathbf{E}} \|\mathbf{DE} - \mathbf{F}\|_F^2\}$, under the assumptions that the entries of $\mathbf{w}$ from (2.13) and of $\mathbf{F}$, are i.i.d with zero mean and unit variance.

We proceed directly with the main result. We recall that $\Phi_{\mathrm{MP},\lambda}(t, r), F_{MP,\lambda}(.)$ and $f_{MP,\lambda}(.)$ are respectively the incomplete first moment, the CDF and PDF of the standard Marchenko-Pastur distribution. We also recall that we operate under the assumption that $NT \geq L, NT \geq K$.

Theorem 10. *In the limit of large N and constant $\delta, \gamma, \kappa, R$, the average optimal reconstruction error is bounded as*

$$\hat{\epsilon} \leq \Phi_{MP,\lambda}(t, r) = \int_r^t x f_{MP,\lambda}(x) dx \tag{6.2}$$

where $\lambda = \frac{\delta K}{\gamma L} = \frac{\Delta}{\Gamma}, r = (1 - \sqrt{\lambda})^2$, and where t is the solution to $F_{MP,\lambda}(t) = 1 - T\frac{\gamma N}{K}$. Furthermore, the bound is tight and the corresponding performance is optimal under the assumption that $\mathbf{D}$ and $\mathbf{E}$ satisfy the disjoint support assumption (cf. Definition 3).

Proof. The achievable part of the proof is based on the general scheme described in the appendix Section 6.3, while the statistical aspect reflects the properties of the Marchenko–Pastur distribution law, as described in the appendix Section 6.3. The converse utilizes basic arguments from tilling literature, conditioned to the requirement of having to cover all of $\mathbf{F}$, as described in the appendix Section 6.3. □

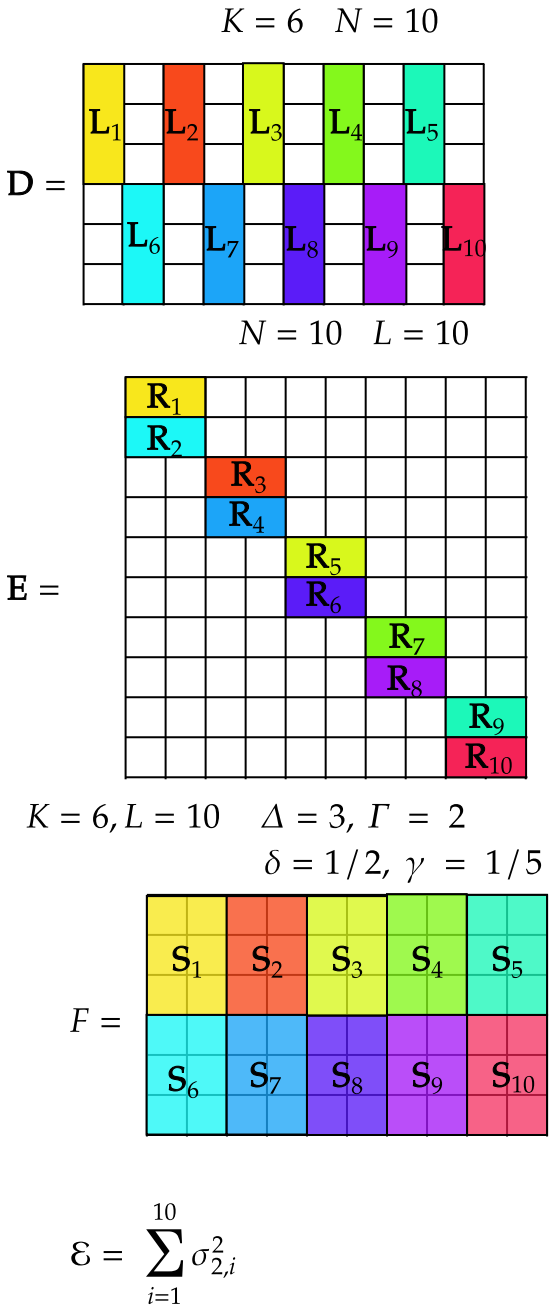


Figure 6.1: A problem setting with the same $K = 6, L = 10,\ \Delta = 3 \backslash \delta = 1/2, \Gamma = 2 \backslash \gamma = 1/5, N = 10$ and $\mathcal{E} = \sum_{i=1}^{10} \sigma_{2,i}^2$, where $\sigma_{2,i}$ as the second singular value of $\mathbf{S}_i$ in decreasing order (the least singular value). In this setting $\mathbf{L}_i\mathbf{R}_i$ is the best rank-1 approximation to the submatrix $\mathbf{S}_i$ given by the famous truncated SVD described in Subsection 5.6.1. Compared to the Example 5.2 and figure 5.3 settings, here the Computation cost $\Gamma = 2 \backslash \gamma = 1/5$ and less number of the required servers $N = 10$. The price paid to have this reduction in the number of servers is our tolerance for error in the recovery of the functions.

Definition 12. [Disjoint Balanced Support Assumption] We say that two matrices $\mathbf{D} \in \mathbb{R}^{K \times NT}, \mathbf{E} \in \mathbb{R}^{NT \times L}$ accept the *disjoint balanced support assumption* if an only if they accept the disjoint support assumption (cf. Definition 3) and additionally it holds that $\|\mathbf{D}(:,i)\|_0 = \|\mathbf{D}(:,j)\|_0$, $\|\mathbf{E}(i,:)\|_0 = \|\mathbf{E}(j,:)\|_0$, $\forall i,j \in [NT]$.

The uniformity assumption reflects a uniformity in the computational and communication capabilities across the servers.

Let us provide a small example of the scheme.

Example 6. We consider a multi-user distributed computing scenario which, as in Example 1, entails $K = 6$ users, $T = 1$ shots, $L = 10$ subfunctions, and a per-server computation and communication cost defined by $\Gamma = 5$ and $\Delta = 3$ respectively. The difference is that now the system only benefits from $N = 4$ servers. Recall from Example 1 that $\frac{KL}{\Delta\Gamma}\min(\Delta,\Gamma) = 12$ was the minimum number of servers (of the same computation and communication capabilities Γ, Δ) required to yield error-free reconstruction. Now, with a reduced $N = 4$, we expect to have erroneous reconstruction.

We describe how to construct $\mathbf{D} \in \mathbb{R}^{6\times 4}$ and $\mathbf{E} \in \mathbb{R}^{4\times 10}$, after receiving $\mathbf{F}$. Useful in this description will be Figure 6.2, which illustrates the tessellation pattern used to tile each non-zero submatrix of $\mathbf{D}$ and $\mathbf{E}$ with $\mathbf{L}_j$ and $\mathbf{R}_j$ respectively.

The scheme entails three main steps.

1. In the first step, we partition $\mathbf{F}$ into 4 submatrices $\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3, \mathbf{F}_4$ of size 3×5, as shown in Figure 5.2. We will approximate these respectively by submatrices $\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3, \mathbf{S}_4$, such that $\mathbf{S}_1$ will approximate the upper left 3×5 submatrix, $\mathbf{S}_2$ the lower right 3×5 submatrix, and so on.

2. As a second step, to create each $\mathbf{S}_j$, we start by SVD-decomposing each submatrix $\mathbf{F}_j$. As there are only 4 servers and four submatrices, we remove the two less dominant singular directions of each decomposition, preserving only each dominant singular direction. Based on this rank-1 SVD approximation (see Section 5.6.1 for more details), each $\mathbf{S}_j$ takes the form $\mathbf{S}_j = \mathbf{L}_j\mathbf{R}_j$ where now these dominant directions $\mathbf{L}_j \in \mathbb{R}^{3\times 1}$ and $\mathbf{R}_j \in \mathbb{R}^{1\times 5}, j \in [4]$, will serve as the tiles of our designed $\mathbf{D}$ and $\mathbf{E}$ respectively.

3. In the final third step, we decide where to place the tiles $\mathbf{L}_j, \mathbf{R}_j$ in $\mathbf{D}$ and $\mathbf{E}$ respectively. To do this, we employ the tessellation pattern in Figure 6.2. For example $\mathbf{L}_1$ and $\mathbf{L}_4$ are respectively the upper left and lower right 3×1 vectors in $\mathbf{D}$, while $\mathbf{R}_1$ and $\mathbf{R}_4$ are respectively the upper left and lower left 1×5 vectors in $\mathbf{E}$.

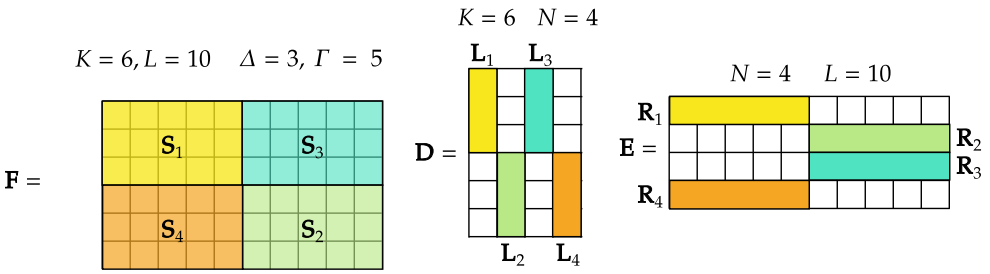


Figure 6.2: Corresponding to Example 6, this figure illustrates the tessellation pattern used to design $\mathbf{D}$ and $\mathbf{E}$ for a system with $K = 6$ users, $T = 1$ shots, $L = 10$ subfunctions, $\Gamma = 5$ and $\Delta = 3$, but with only $N = 4$ servers. The reduced number of servers forces SVD-based approximations which entail lossy function reconstruction.

To verify that the communication and computational costs at each server are not violated, we simply note that each column of $\mathbf{D}$ only has $\Delta = 3$ non-zero elements, and each row of $\mathbf{E}$ only has $\Gamma = 5$ non-zero elements. Finally, the overall approximation error for $\mathbf{F}$ takes the form

$$\sum_{j=1}^{4} \|\mathbf{F}_j - \mathbf{L}_j\mathbf{R}_j\|^2 = \sum_{j=1}^{4} \sqrt{\sigma_{2,j}^2 + \sigma_{3,j}^2} \tag{6.3}$$

where $\sigma_{i,j}, i \in [3], j \in [4]$ are the singular values of $\mathbf{F}_j$ in descending order. □

Remark 7. The above gives an example of the employed scheme for designing $\mathbf{D}$ and $\mathbf{E}$, and the expression in (6.3) gives an example of the corresponding error performance. The scheme is described for any dimension in Section 6.1, and the expression is simple and it takes the form

$$\mathcal{E}_{\mathbf{F}} = \sum_{j=1}^{n} \|\mathbf{F}_j - \mathbf{L}_j\mathbf{R}_j\|^2 = \sum_{j=1}^{n} \sqrt{\sigma_{q+1,j}^2 + \sigma_{q+2,j}^2 + \cdots} \tag{6.4}$$

where n describes the number of submatrices that we have divided $\mathbf{F}$ into, and where q simply represents the truncation depth of the SVD, where we keep only the q most dominant dimensions of each submatrix $\mathbf{F}_j$. In terms of guarantees, we clarify as follows. As is probably clear, an exact (non-truncated) SVD approach applies to the error free scenario of $\mathcal{E}_{\mathbf{F}} = 0$, for which we have indeed proven optimality under the assumption of full rank $\mathbf{F}$, where our optimality results tell us that $\mathcal{E}_{\mathbf{F}} = 0$ is reached with the least amount of resources, in terms of γ, δ, N, compared to any other conceivable scheme. In the error free case, the scheme is exactly optimal for any full-rank $\mathbf{F}$. On the other

hand, for the lossy case, we note that while the scheme applies and can be used for the finite dimensional case, we also recall that in terms of evaluating statistics in closed form and for providing optimality guarantees, we needed to revert to the asymptotic and stochastic setting, for which the scheme (and the corresponding averaging $\mathbb{E}[\mathcal{E}_{\mathbf{F}}]$ of (6.4), as rigorously described in (6.1)) is optimal under the assumption of disjoint supports. On the other hand, for the non-asymptotic and non-stochastic lossy regime (where guarantees are required for any specific finite-dimensional full-rank $\mathbf{F}$), we clarify that the general expression in (6.4) (seen more rigorously in (6.16)), offers a valid upper bound (directly from (5.34)) on the optimal error of our distributed computing problem, under an assumption of normalized outputs $\|\mathbf{w}\|^2 = 1$. Finally, in the asymptotic but non-stochastic setting (where indeed $\|\mathbf{w}\|^2 = 1$ is guaranteed with high probability, under commonly employed assumptions), the above general evaluation in (6.4) serves as an upper bound on the optimal error in the distributed computing problem, for any specific full-rank large $\mathbf{F}$, where naturally the bound holds irrespective of whether we accept or not the disjoint support assumption.

6.2 Discussion and Conclusion

In this chapter, we have investigated the lossy tesselated distributed computing. For this case, being presented with the challenging problem of approximate matrix factorization, we employed asymptotics and a stochastic metric, that allowed us to provide for the first time a clear bound on the optimal normalized reconstruction error as a function of the communication and computational resources. This bound is in fact tight under the assumption of disjoint supports, and the simple schemes that achieve this bound employ basic tilling techniques, together with truncated SVD decompositions[1].

In this work, we investigated the fundamental limits of multi-user distributed computing of real-valued linearly-decomposable functions. In addressing this problem, we have made clear connections to the problem of fixed support matrix factorization, tessellation theory, as well as have established an interesting connection between the problem of distributed computing, and the statistical properties of large matrices. Under a basic disjoint support assumption, the error-free system capacity $C = \frac{K}{N_{opt}}$ in Theorem 9 revealed the optimal computational and communication resources γ, δ, N required to

[1]The derived solutions in Theorem 9 and Theorem 10 entail at most $\lceil \frac{K}{\Delta} \rceil \lceil \frac{L}{\Gamma} \rceil$ SVD decompositions, and a corresponding additional (unaccounted for) complexity of $\lceil \frac{K}{\Delta} \rceil \lceil \frac{L}{\Gamma} \rceil O(\Delta \Gamma^2) = O(KL\Gamma) = O(KL^2)$. We believe that such complexity may often be dwarfed by the cost of evaluating the L subfunctions that may often be non-linear.

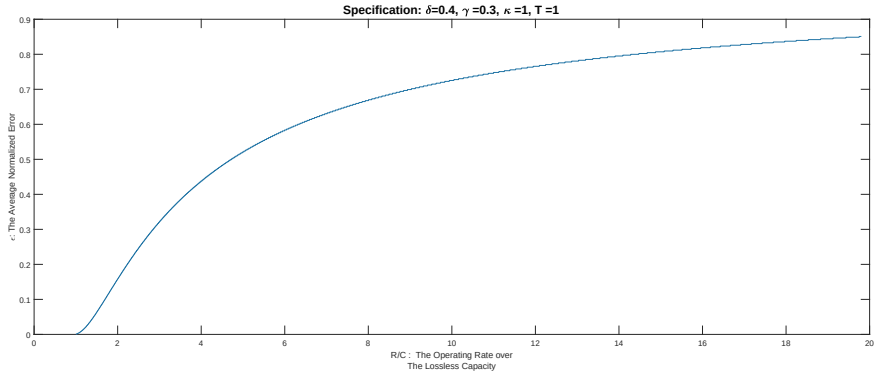


Figure 6.3: The y-axis represents the conditionally optimal average error ϵ derived in Theorem 10 for $\delta = 0.4, \gamma = 0.3, \kappa = 1$ and $T = 1$. The x-axis describes the ratio of the operating rate to the error-free or lossless system capacity, i.e. describes how many times higher is the system rate from the error-free system capacity.

accommodate a certain number of users and subfunctions. The same result yields a simple relationship between computational complexity and communication load, as this is described in Corollary 2. The derived performance is proven optimal over a sizeable class of schemes.

For the lossy case, after transitioning to the equivalent problem of approximate matrix factorization, we employed asymptotics and a stochastic metric, allowing us to provide a clear bound on the optimal normalized reconstruction error as a function of the matrix sparsity as it reflects our communication and computational resources. This bound is in fact tight under the assumption of uniform and disjoint supports, and it is an outcome of our schemes that employ tiling techniques, together with truncated SVD decompositions[2].

One of the interesting outcomes of the work is an analytical handle on how we can tradeoff our system rate (corresponding to our server resources or the clients we serve) with the function reconstruction error. Figure 6.3 offers some understanding on how lossy reconstruction is affected by having either too few servers or too many users. Starting from a lossless scenario where the operating ratio K/N matches the error-free system capacity (corresponding to the value of 1 on the x-axis), we see how the error increases as we either add more users or as we remove servers. For example, when the ratio R/C between

[2]The derived solutions in Theorem 9 and Theorem 10 entail at most $\lceil \frac{K}{\Delta} \rceil \lceil \frac{L}{\Gamma} \rceil$ SVD decompositions, and a corresponding additional (unaccounted for) complexity of $\lceil \frac{K}{\Delta} \rceil \lceil \frac{L}{\Gamma} \rceil O(\Delta\Gamma^2) = O(KL\Gamma) = O(KL^2)$. The proposed approach operates under the assumption that such costs are small compared to the costs of evaluating the L subfunctions that may often be non-linear.

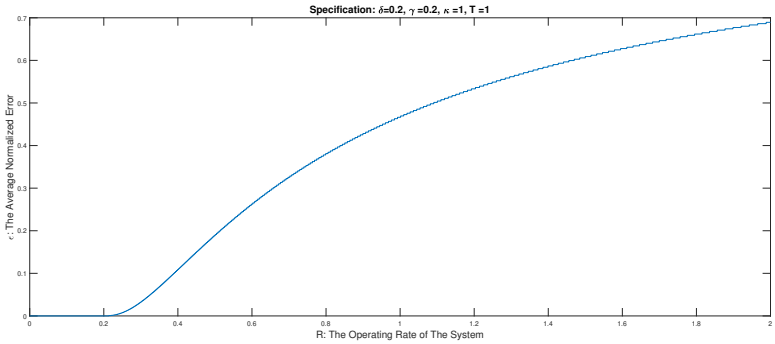


Figure 6.4: The y-axis plots ϵ from Theorem 10 for $\delta = 0.2, \gamma = 0.2, \kappa = 1$ and $T = 1$, while the x-axis represents the rate K/N.

the operating rate and the error-free capacity, is around 2 (after doubling the users or halving the servers) then we expect an error corresponding to $\epsilon = 0.15$. On the other hand, if we could accept $\epsilon = 0.3$, then — compared to the error-free scenario — we could triple the number of users or equivalently reduce our servers to about one third.

Similarly Figure 6.4, for the same setting, plots the error as a function of the operating rate (unnormalized), for a slightly different setting where now $\delta = 0.2, \gamma = 0.2, \kappa = 1, T = 1$. In this scenario, where the error-free capacity is merely $C_0 = 0.2$ (cf. (5.40)), an aggressive increase in the rate to $R = 2$ (2 times more users than servers, and $10 = \frac{R}{C_0}$ times more users than in the error-free case) yields a generally unmanageable $\epsilon = 0.68$, etc. Finally, Figure 6.5 reflects the effect of the per-server computation resources γ, the per-server communication resources δ, the effect of T and the effect of η (cf. (5.7)) where a higher η implies more communication resources per subfunction.

One interesting outcome of our work is that substantial computational and communication savings can be harvested with only a modest cost in function reconstruction error. This has to do with the nature of the singular value distribution of larger matrices (corresponding to many users and many basis subfunctions). Due to this nature, in principle, removing computational resources will indeed introduce error, but will do so in a manner that is initially slow. Such error power is initially proportional to the smallest singular value of a large $\mathbf{F}$, and then with more computational/communication savings, it becomes proportional to the sum of the few smallest singular values, and so on. Due to the statistical nature of larger matrices, this error accumulates slowly, thus considerable computational savings can be obtained, with a relatively modest function reconstruction error (See Figure 6.6).

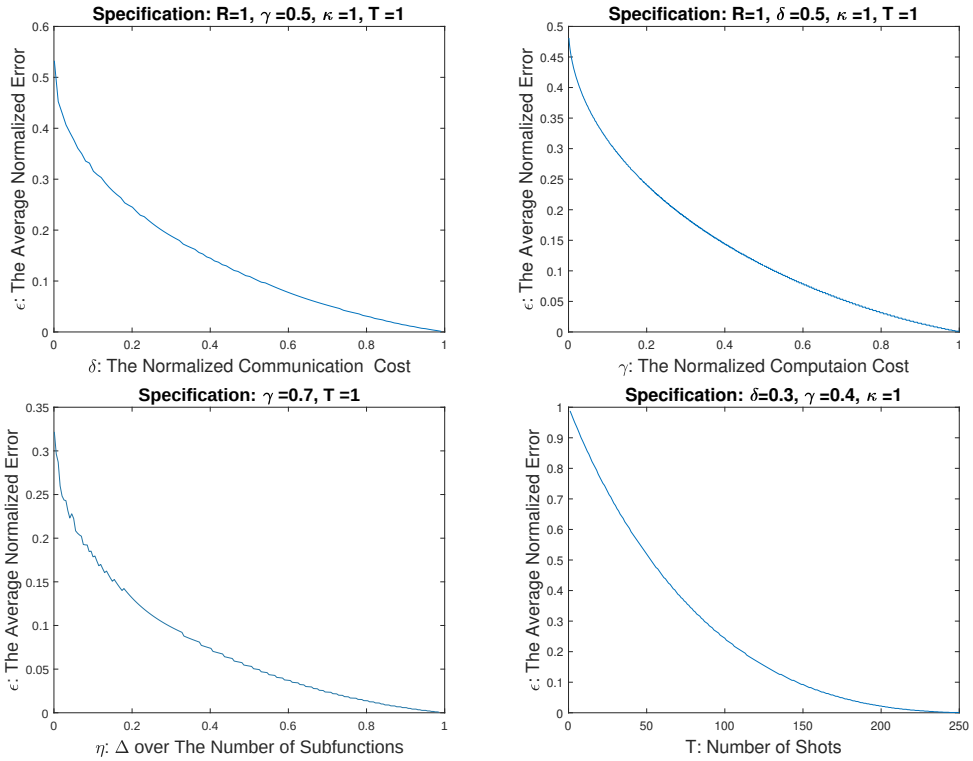


Figure 6.5: Plotting the error-effect of δ, γ, η, T (Theorem 10) in various settings. The y axis corresponds to ϵ (cf. (5.34)).

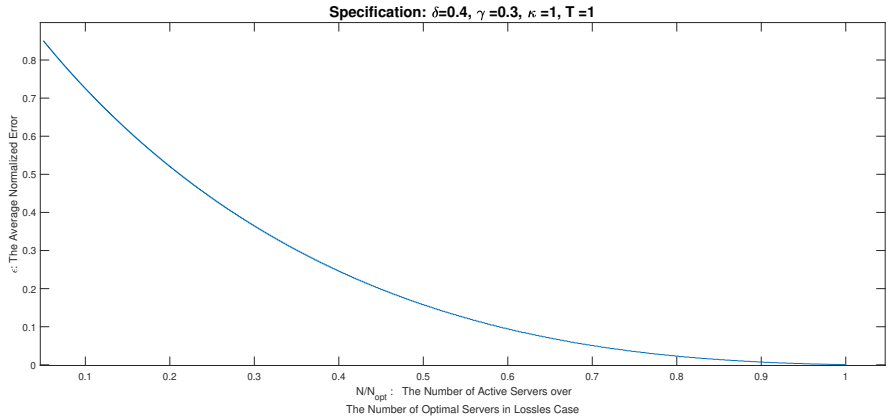


Figure 6.6: The y-axis represents the conditionally optimal average error ϵ derived in Theorem 10 for $\delta = 0.4, \gamma = 0.3, \kappa = 1$ and $T = 1$. The x-axis describes the number of active servers over the number of optimal serves in the lossless scheme.

For future work, we can consider various related scenarios such as the scenario where some subfunctions contribute more to the overall function than other subfunctions do. Another interesting direction could involve a pre-defined and fixed communication topology, which would then entail a support of $\mathbf{D}$ that is fixed and independent of $\mathbf{F}$, or similarly a scenario where each server can only compute a predefined subset of subfunctions, now bringing to the fore a support of $\mathbf{E}$ that is again fixed and independent of $\mathbf{F}$.

6.3 Appendix: Proof of The achievability and converse of Theorem 10

The proof incorporates an achievable and a converse argument. We first prove the following lemma used in (5.33), corresponding to the average normalized error[3] $\epsilon = \frac{\mathbb{E}_{\mathbf{F},\mathbf{w}}\{\mathcal{E}\}}{KL}$ of any scheme with communication matrix $\mathbf{D}$ and computing matrix $\mathbf{E}$. Recall that in our asymptotic setting, the parameter N scales to infinity, while the calibrating ratios $\delta, \gamma, \kappa, R$ remain constant. We also note that T here is a non-scaling constant, while we also recall that the elements of $\mathbf{w}$ are i.i.d, independent of $\mathbf{D}, \mathbf{E}, \mathbf{F}$, and have unit variance. We proceed with the first lemma:

Lemma 23. In the limit of large N, the average normalized error $\epsilon = \frac{\mathbb{E}_{\mathbf{F},\mathbf{w}}\{\mathcal{E}\}}{KL}$ of any lossy function reconstruction scheme corresponding to $\mathbf{DE} = \mathbf{F}$, under the assumptions of Theorem 10, takes the form

$$\epsilon = \frac{\mathbb{E}_{\mathbf{F}}\{\|\mathbf{DE} - \mathbf{F}\|_F^2\}}{KL}. \tag{6.5}$$

Proof. The proof starts with the definition of the average error from (5.33) where we defined this error to be $\epsilon \, \frac{\mathbb{E}_{\mathbf{F}}\{\|\mathbf{DE}-\mathbf{F}\|_F^2\}}{KL}$, then the following sequence of steps hold

$$\begin{aligned} \epsilon &= \frac{\mathbb{E}_{\mathbf{F},\mathbf{w}}\{[(\mathbf{DE} - \mathbf{F})\mathbf{w}]^\intercal[(\mathbf{DE} - \mathbf{F})\mathbf{w}]\}}{KL} && (6.6) \\ &= \frac{\mathbb{E}_{\mathbf{F},\mathbf{w}}\{\mathbf{w}^\intercal(\mathbf{DE} - \mathbf{F})^\intercal(\mathbf{DE} - \mathbf{F})\mathbf{w}\}}{KL} \\ &\overset{(a)}{=} \frac{\mathbb{E}_{\mathbf{F},\mathbf{w}}\{\mathrm{tr}(\mathbf{w}^\intercal(\mathbf{DE} - \mathbf{F})^\intercal(\mathbf{DE} - \mathbf{F})\mathbf{w})\}}{KL} && (6.7) \\ &\overset{(b)}{=} \frac{\mathbb{E}_{\mathbf{F},\mathbf{w}}\{\mathrm{tr}((\mathbf{DE} - \mathbf{F})^\intercal(\mathbf{DE} - \mathbf{F})\mathbf{w}\mathbf{w}^\intercal)\}}{KL} \end{aligned}$$

[3]We recall from (5.4) our instantaneous error $\mathcal{E} = \sum_{k=1}^{K} |F'_k - F_k|^2$, $\mathcal{E} \in \mathbb{R}$, $\forall k \in [K]$.

$$\overset{(c)}{=} \frac{\mathrm{tr}(\mathbb{E}_{\mathbf{F},\mathbf{w}}\{(\mathbf{DE}-\mathbf{F})^{\intercal}(\mathbf{DE}-\mathbf{F})\mathbf{w}\mathbf{w}^{\intercal}\})}{KL} \tag{6.8}$$

$$\overset{(d)}{=} \frac{\mathrm{tr}(\mathbb{E}_{\mathbf{F}}\{(\mathbf{DE}-\mathbf{F})^{\intercal}(\mathbf{DE}-\mathbf{F})\}\mathbb{E}_{\mathbf{w}}\{\mathbf{w}\mathbf{w}^{\intercal}\})}{KL}$$

$$\overset{(e)}{=} \frac{\mathrm{tr}(\mathbb{E}_{\mathbf{F}}\{(\mathbf{DE}-\mathbf{F})^{\intercal}(\mathbf{DE}-\mathbf{F})\})}{KL} \tag{6.9}$$

$$\overset{(f)}{=} \frac{\mathbb{E}_{\mathbf{F}}\{\mathrm{tr}((\mathbf{DE}-\mathbf{F})^{\intercal}(\mathbf{DE}-\mathbf{F}))\}}{KL} \tag{6.10}$$

$$\overset{(g)}{=} \frac{\mathbb{E}_{\mathbf{F}}\{\|\mathbf{DE}-\mathbf{F}\|_F^2\}}{KL} \tag{6.11}$$

where (a) holds since the trace argument is a scalar, (b) and (c) follow from the cyclic property of the trace operation and its linearity, (d) holds since $\mathbf{w}$ and $\mathbf{F}$ are independent, (e) holds since $\mathbb{E}_{\mathbf{w}}\{\mathbf{w}\mathbf{w}^{\intercal}\} = \mathbf{I}_L$, (f) follows from the interchangeability of the trace and expectation operations, and (g) holds since $\mathrm{tr}(\mathbf{A}^{\intercal}\mathbf{A}) = \sum_{i=1}^{I}\sum_{j=1}^{J}\mathbf{A}(i,j)^2 = \|\mathbf{A}\|_F^2$, $\forall \mathbf{A} \in \mathbb{R}^{I\times J}$. □

In the following, we provide the achievability part of the proof, where we present in this appendix, the lossy variant of the scheme of Theorem 9, while in this appendix we also derive the average of the normalized error. For the error analysis, in Lemma 24 (see also Remark 8) we adapt the well known Marchenko–Pastur Theorem to our setting, to then yield the proof of Lemma 25 that tells us how the approximation error corresponding to each tile is defined by the truncated first moment of the Marchenko–Pastur distribution. Subsequently, in Lemma 26, we describe the total error $\|\mathbf{DE}-\mathbf{F}\|_F^2$ as the sum of errors for each tile, reflecting the nature of our tile assignment from Theorem 9. The last step combines Lemmas 25 and 26 to derive the corresponding upper bound on the normalized error. Note that all the definitions in appendix 5.6 apply to our setting.

6.3.1 Scheme Design

Similar to Section 5.7.1 of this appendix, the construction of $\mathbf{D}, \mathbf{E}$ will involve the steps of: a) sizing and positioning the tiles of $\mathbf{D}$, of $\mathbf{E}$, and of $\mathbf{DE}$, b) filling the non-zero tiles in $\mathbf{DE}$ as a function of $\mathbf{F}$, and c) filling the tiles in $\mathbf{D}$ and $\mathbf{E}$. Crucial to our lossy-variant of our scheme, will be the rank $\alpha r_{\mathcal{P}} \in \mathbb{N}$ of each tile $\mathcal{P}$, where this rank will be defined by the desired error performance as we will see in this appendix. We proceed with the description of the steps.

Sizing and positioning the tiles of D, of E, and of DE Our first step applies the corresponding step in the achievable scheme of the appendix

Section 5.7.1, to the current setting of $\mathcal{C}_2 = \mathcal{C}_3 = \mathcal{C}_4 = \emptyset$, where this latter equality follows after noting that — in terms of the derived performance — $\Delta|K$, $\Gamma|L$ and $T|\min(\Delta, \Gamma)$ hold directly in our current setting of constant T and scaling Δ and Γ.

Filling the non-zero tiles in DE as a function of F We here first approximate each $\mathbf{F}_{\mathcal{P}}$ (cf. (5.59)) by a low-rank matrix $\mathbf{F}^{\alpha}_{\mathcal{P}}$, whose rank does not exceed $\alpha r_{\mathcal{P}}$, for some auxiliary variable $\alpha \in \mathbb{R}, 0 < \alpha \leq 1, \alpha r_{\mathcal{P}} \in \mathbb{N}$ [4]. To obtain the desired

$$\mathbf{F}^{\alpha}_{\mathcal{P}} \triangleq \underset{\mathbf{A}}{\operatorname{argmin}} \{ \|\mathbf{A} - \mathbf{F}_{\mathcal{P}}\|_F, \quad : \quad \operatorname{rank}(\mathbf{A}) \leq \alpha r_{\mathcal{P}} \} \tag{6.12}$$

we employ the well-known Echart-Young Theorem [139] to get

$$\mathbf{F}^{\alpha}_{\mathcal{P}} = \mathbf{D}^{\alpha}_{\mathcal{P}} \mathbf{E}^{\alpha}_{\mathcal{P}} \tag{6.13}$$

where $\mathbf{D}_{\mathcal{P}} \in \mathbb{R}^{|\mathcal{R}_{\mathcal{P}}| \times \alpha r_{\mathcal{P}}}, \mathbf{E}_{\mathcal{P}} \in \mathbb{R}^{\alpha r_{\mathcal{P}} \times |\mathcal{C}_{\mathcal{P}}|}$ are the results of the truncated SVD algorithm[5]. Naturally, since $\mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4 = \emptyset$, we have that $|\mathcal{R}_{\mathcal{P}}| = \Delta, |\mathcal{C}_{\mathcal{P}}| = \Gamma$ (cf.(5.58)).

Filling the tiles in D and E In this last step, after considering $\mathcal{C}_1 = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m\}$, then for each $j \in [m]$, we set

$$\mathbf{D}(\mathcal{R}_{\mathcal{P}_j}, [\sum_{i=1}^{j-1} \alpha r_{\mathcal{P}_i} + 1, \sum_{i=1}^{j} \alpha r_{\mathcal{P}_i}]) = \mathbf{D}_{\mathcal{P}_j} \tag{6.14}$$

and

$$\mathbf{E}([\sum_{i=1}^{j-1} \alpha r_{\mathcal{P}_i} + 1, \sum_{i=1}^{j} \alpha r_{\mathcal{P}_i}], \mathcal{C}_{\mathcal{P}_j}) = \mathbf{E}_{\mathcal{P}_j} \tag{6.15}$$

while the remaining non-assigned elements of $\mathbf{D}$ and $\mathbf{E}$ remain equal to zero.

[4] As we will see soon, the choice of α here will determine the degree of the approximation of $\mathbf{F}_{\mathcal{P}}$. As we will elaborate later on, this parameter will take the form $\alpha = T\frac{\max(\Delta,\Gamma)}{R}$, while guaranteeing that the rank $\alpha r_{\mathcal{P}}$ is an integer. Note that since $\Delta|K, \Gamma|L$, then $r_{\mathcal{P}} = \min(\Delta, \Gamma)$ (cf. (5.58)) which naturally scales with Δ and Γ.

[5] In particular, $\mathbf{F}^{\alpha}_{\mathcal{P}}$, $\mathbf{D}^{\alpha}_{\mathcal{P}}$ and $\mathbf{E}^{\alpha}$ are respectively associated to $\mathbf{A}_k$, $\mathbf{US}$ and $\mathbf{V}$ in (5.47) of appendix Section 5.6.

6.3.2 Normalized Error Analysis of the Designed Scheme

We proceed to evaluate in Lemma 25 the approximation error for each tile as a function of the truncated first moment of the Marchenko–Pastur distribution, and then to show in Lemma 26 that for our scheme, the total approximation error $\|\mathbf{DE}-\mathbf{F}\|_F^2$, for each instance of the problem, is equal to the sum of the approximation errors, where the sum is over all tiles. The proof for the achievability part of the error analysis is completed by properly combining the two aforementioned lemmas.

Directly from the Echart-Young Theorem, the truncated SVD solution yields the optimal approximation error for each tile, which takes the form

$$\|\mathbf{F}_{\mathcal{P}}^{\alpha}-\mathbf{F}_{\mathcal{P}}\|_F=\|\mathbf{D}_{\mathcal{P}}^{\alpha}\mathbf{E}_{\mathcal{P}}^{\alpha}-\mathbf{F}_{\mathcal{P}}\|_F=\sqrt{\sum_{i=\alpha r_{\mathcal{P}}+1}^{r_{\mathcal{P}}}\sigma_i^2(\mathbf{F}_{\mathcal{P}})} \tag{6.16}$$

where $\sigma_i(\mathbf{F}_{\mathcal{P}}), i\in[r_{\mathcal{P}}]$ are the singular values of $\mathbf{F}_{\mathcal{P}}$ (cf. (5.59)), in decreasing order.

Let us now recall that for each $\mathcal{P}\in\mathcal{C}_1$, then $\mathbf{F}_{\mathcal{P}}\in\mathbb{R}^{\Delta\times\Gamma}$ is a random matrix of *i.i.d* elements having zero mean and $\bar{\sigma}$ variance. Let $\lambda_1(\mathbf{Y}_{\mathcal{P}})\leq\lambda_2(\mathbf{Y}_{\mathcal{P}})\leq\ldots\leq\lambda_{\Delta}(\mathbf{Y}_{\mathcal{P}})$ be the ordered eigenvalues of the symmetric matrix

$$\mathbf{Y}_{\mathcal{P}}\triangleq\frac{1}{\Gamma}\mathbf{F}_{\mathcal{P}}\mathbf{F}_{\mathcal{P}}^{\intercal} \tag{6.17}$$

and consider

$$\mu_{\Delta}(x)=\frac{1}{\Delta}\#\{\lambda_j(\mathbf{Y}_{\mathcal{P}})\leq x\} \tag{6.18}$$

which will be associated below to the CDF $F_{MP,\lambda}=\mu_{\Delta}(x)$ of the Marchenko-Pastur distribution.

The following describes the well known Marchenko-Pastur law ([140]) as applied to our setting.

Lemma 24. [Marchenko-Pastur Law] Consider a random matrix $\mathbf{F}_{\mathcal{P}}\in\mathbb{R}^{\Delta\times\Gamma}$ having *i.i.d* elements with zero mean and variance $\bar{\sigma}$. Let $\lambda_1(\mathbf{Y}_{\mathcal{P}})\leq\lambda_2(\mathbf{Y}_{\mathcal{P}})\leq\ldots\leq\lambda_{\Delta}(\mathbf{Y}_{\mathcal{P}})$ be the ordered eigenvalues of $\mathbf{Y}_{\mathcal{P}}=\frac{1}{\Gamma}\mathbf{F}_{\mathcal{P}}\mathbf{F}_{\mathcal{P}}^{\intercal}$. Let $\Delta,\Gamma\to\infty$, and let $\frac{\Delta}{\Gamma}\to\lambda$ for some $\lambda>0$. Then μ_{Δ} (cf. (6.18)) converges in distribution to

$$\mu_{\Delta}(x)=\begin{cases}(1-\frac{1}{\lambda})\mathbf{1}_{(0\leq x)}+v(x), & \text{if } \lambda>1,\\ v(x), & \text{if } 0<\lambda\leq 1\end{cases} \tag{6.19}$$

where $v(x)$ is such that

$$d(v(x)) = \frac{1}{2\pi\bar{\sigma}^2}\frac{\sqrt{(\lambda_+ - x)(x-\lambda_-)}}{\lambda x}\mathbf{1}_{x\in[\lambda_-,\lambda_+]}dx \tag{6.20}$$

for any point $x \in \mathbb{R}$. This in turn means that the PDF of the Marchenko-Pastur distribution takes the form

$$f_{MP,\lambda}(x) = \frac{1}{2\pi\bar{\sigma}^2}\frac{\sqrt{(\lambda_+ - x)(x-\lambda_-)}}{\lambda x}\mathbf{1}_{x\in[\lambda_-,\lambda_+]} + \mathbf{1}_{(1<\lambda)}(1-\frac{1}{\lambda})\delta(x) \tag{6.21}$$

where $\lambda_\pm \triangleq \sigma^2(1\pm\sqrt{\lambda})^2$.

Remark 8. With the same notation as in Lemma 24, the CDF's explicit form is

$$F_{\mathrm{MP},\lambda}(x) = \begin{cases} \frac{\lambda-1}{\lambda}\mathbf{1}_{x\in[0,\lambda_-)} + \left(\frac{\lambda-1}{2\lambda} + F_{\mathrm{MP}}(x)\right)\mathbf{1}_{x\in[\lambda_-,\lambda_+)} + \mathbf{1}_{x\in[\lambda_+,+\infty)}, \\ \text{if } \lambda > 1, \\ F_{\mathrm{MP}}(x)\mathbf{1}_{x\in[\lambda_-,\lambda_+)} + \mathbf{1}_{[\lambda_+,\infty)}, \\ \text{if } 0 \le \lambda \le 1, \end{cases}$$

$$F_{\mathrm{MP}}(x) = \frac{1}{2\pi\lambda}\Big(\pi\lambda + \sigma^{-2}\sqrt{(\lambda_+ - x)(x-\lambda_-)} - (1+\lambda)\arctan(\frac{r(x)^2-1}{2r(x)}) + (1-\lambda)\arctan(\frac{\lambda_- r(x)^2 - \lambda_+}{2\sigma^2(1-\lambda)r(x)})\Big)$$

where $r(x) = \sqrt{\dfrac{\lambda_+ - x}{x-\lambda_-}}$.

Now utilizing Lemma 24, we provide an expression for the error contributed by any tile $\mathcal{P} \in \mathcal{C}_1$. The following lemma holds under the same assumptions as in Theorem 10 and Lemma 24.

Lemma 25. The average error attributed to each tile $\mathcal{P}$, described in (6.16), takes the form

$$\mathbb{E}_{\mathbf{F}}\{\|\mathbf{F}^\alpha_{\mathcal{P}} - \mathbf{F}_{\mathcal{P}}\|_F^2\} = \Gamma\Delta\int_{(1-\sqrt{\lambda})^2}^{F^{-1}_{MP,\lambda}(1-(\min(\Delta,\Gamma)/\Delta)\alpha)} x f_{MP,\lambda}(x)\,dx. \tag{6.22}$$

Proof. Addressing first the case where $\Delta \le \Gamma$, we see that

$$\mathbb{E}_{\mathbf{F}}\{\|\mathbf{F}^\alpha_{\mathcal{P}} - \mathbf{F}_{\mathcal{P}}\|_F^2\} \overset{(a)}{=} \mathbb{E}_{\mathbf{F}}\{\sum_{i=1}^{(1-\alpha)r_{\mathcal{P}}} \sigma^2_{i+\alpha r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}})\}$$
$$= \Gamma\mathbb{E}_{\mathbf{F}}\{\sum_{i=1}^{(1-\alpha)r_{\mathcal{P}}} \frac{\sigma^2_{i+\alpha r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}})}{\Gamma}\}$$

$$\overset{(b)}{=} \Gamma\mathbb{E}_{\mathbf{F}}\{\int_{-\infty}^{+\infty} \sum_{i=1}^{(1-\alpha)r_{\mathcal{P}}} \delta(x - \frac{\sigma^2_{i+\alpha r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}})}{\Gamma})\, x dx\}$$

$$\overset{(c)}{=} \Gamma\mathbb{E}_{\mathbf{F}}\{\sum_{i=1}^{(1-\alpha)r_{\mathcal{P}}} \int_{-\infty}^{+\infty} \delta(x - \frac{\sigma^2_{i+\alpha r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}})}{\Gamma})\, x dx\}$$

$$\overset{(d)}{=} \Gamma\mathbb{E}_{\mathbf{F}}\{\sum_{i=1}^{(1-\alpha)r_{\mathcal{P}}} \int_{-\infty}^{+\infty} \lim_{\delta x \to 0} \frac{\mathbf{1}(x < \sigma^2_{i+\alpha r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}})/\Gamma \leq x + \delta x)}{\delta x}\, x dx\}$$

$$\overset{(e)}{=} \Gamma \sum_{i=1}^{(1-\alpha)r_{\mathcal{P}}} \int_{-\infty}^{+\infty} \lim_{\delta x \to 0} \mathbb{E}_{\mathbf{F}}\{\frac{\mathbf{1}(\sigma^2_{i+\alpha r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}})/\Gamma \leq x + \delta x)}{\delta x}$$

$$- \frac{\mathbf{1}(\sigma^2_{i+\alpha r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}})/\Gamma \leq x)}{\delta x}\}\, x dx$$

$$\overset{(f)}{=} \Gamma\Delta \int_{-\infty}^{F^{-1}_{MP,\lambda}(1-\alpha)} \lim_{\delta x \to 0} \frac{F_{MP}(x + \delta x) - F_{MP,\lambda}(x)}{\delta x}\, x dx$$

$$\overset{(g)}{=} \Gamma\Delta \int_{(1-\sqrt{\lambda})^2}^{F^{-1}_{MP,\lambda}(1-\alpha)} x f_{MP,\lambda}(x)\, dx$$

where (a) holds because of (6.16), (b) is true because of the continuous delta Dirac function properties, (c) follows from the linearity of both the summation and integral operators, while (d) and (e) follow from the fact that $d(\mathbf{1}(x - \sigma^2_{i+\alpha r_{\mathcal{P}}}/\Gamma))/dx = \lim_{\delta x \to 0} \frac{\mathbf{1}(\sigma^2_{i+\alpha r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}})/\Gamma \leq x + \delta x) - \mathbf{1}(\sigma^2_{i+\alpha r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}})/\Gamma \leq x)}{\delta x}$. Then, to show (f), we first note that $\frac{1}{\Delta}\#\{\lambda_j(\mathbf{Y}_{\mathcal{P}}) = \frac{\sigma^2_j(\mathbf{F}_{\mathcal{P}})}{\Gamma} \leq x, j \in [r_{\mathcal{P}}]\} \to F_{MP,\lambda}(x)$ as Δ, Γ scale to infinity. Now, again in (f), to understand the change of limits in the integral, we note that given that $r_{\mathcal{P}} = \Delta \leq \Gamma$, we seek to find an x' such that $(1-\alpha)r_{\mathcal{P}} = (1-\alpha)\Delta = \Delta F_{MP}(x')$, which will allow x' to be the upper integration limit[6] of the integral in the RHS of equality (f), yielding $\sum_{i=1}^{(1-\alpha)r_{\mathcal{P}}} \mathbf{1}(\sigma^2_{i+\alpha r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}})/\Gamma \leq x) = \sum_{i=1}^{r_{\mathcal{P}}} \mathbf{1}(\sigma^2_i(\mathbf{F}_{\mathcal{P}})/\Gamma \leq x) = \Delta F_{MP,\lambda}(x)$ since $x \leq x'_1 = F^{-1}_{MP}(1-\alpha)$, thus yielding (f). Finally (g) is simply the result of the relationship between CDF and PDF, after considering the support of the Marchenko-Pastur distribution.

For the case where $\Delta > \Gamma$, we have that

$$\mathbb{E}_{\mathbf{F}}\{\|\mathbf{F}^{\alpha}_{\mathcal{P}} - \mathbf{F}_{\mathcal{P}}\|^2_F\} \overset{(a)}{=} \mathbb{E}_{\mathbf{F}}\{\sum_{i=1}^{(1-\alpha)r_{\mathcal{P}}} \sigma^2_{i+\alpha r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}})\}$$

$$= \Gamma\mathbb{E}_{\mathbf{F}}\{\sum_{i=1}^{(1-\alpha)r_{\mathcal{P}}} \frac{\sigma^2_{i+\alpha r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}})}{\Gamma}\}$$

[6] In essence, this step simply says that the first $(1-\alpha)r_{\mathcal{P}}$ eigenvalues (corresponding to the summation) correspond to the eigenvalues accounted for by $F^{-1}_{MP}(1-\alpha)$.

$$\stackrel{(b)}{=} \Gamma\mathbb{E}_{\mathbf{F}}\{\int_{-\infty}^{+\infty}\sum_{i=1}^{(1-\alpha)r_{\mathcal{P}}}\delta(x-\frac{\sigma^2_{i+\alpha r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}})}{\Gamma})\,xdx\}$$

$$\stackrel{(c)}{=} \Gamma\mathbb{E}_{\mathbf{F}}\{\sum_{i=1}^{(1-\alpha)r_{\mathcal{P}}}\int_{-\infty}^{+\infty}\delta(x-\frac{\sigma^2_{i+\alpha r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}})}{\Gamma})\,xdx\}$$

$$\stackrel{(d)}{=} \Gamma\mathbb{E}_{\mathbf{F}}\{\sum_{i=1}^{(1-\alpha)r_{\mathcal{P}}}\int_{-\infty}^{+\infty}\lim_{\delta x\to 0}\frac{\mathbf{1}(x<\sigma^2_{i+\alpha r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}})/\Gamma\leq x+\delta x)}{\delta x}\,xdx\}$$

$$\stackrel{(e)}{=} \Gamma\sum_{i=1}^{(1-\alpha)r_{\mathcal{P}}}\int_{-\infty}^{+\infty}\lim_{\delta x\to 0}\mathbb{E}_{\mathbf{F}}\{\frac{\mathbf{1}(\sigma^2_{i+\alpha r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}})/\Gamma\leq x+\delta x)}{\delta x}$$

$$-\frac{\mathbf{1}(\sigma^2_{i+\alpha r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}})/\Gamma\leq x)}{\delta x}\}\,xdx$$

$$\stackrel{(f)}{=} \Gamma\Delta\int_{-\infty}^{F^{-1}_{MP,\lambda}((\Gamma/\Delta)(1-\alpha)+(1-\frac{\Gamma}{\Delta}))}$$

$$\lim_{\delta x\to 0}\frac{F_{MP}(x+\delta x)-F_{MP,\lambda}(x)}{\delta x}\,xdx$$

$$\stackrel{(g)}{=} \Gamma\Delta\int_{(1-\sqrt{\lambda})^2}^{F^{-1}_{MP,\lambda}(1-\alpha\frac{\Gamma}{\Delta})}xf_{MP,\lambda}(x)\,dx$$

where firstly (a) through (e) and (g) follow as in the first case of $\Delta\leq\Gamma$. Then, regarding (f), we first note that $\frac{1}{\Delta}\#\{\lambda_j(\mathbf{Y}_{\mathcal{P}})=\frac{\sigma^2_j(\mathbf{F}_{\mathcal{P}})}{\Gamma}\leq x, j\in[r_{\mathcal{P}}]\}\to F_{MP,\lambda}(x)$ as Δ,Γ scale to infinity. We also note that $r_{\mathcal{P}}=\Gamma<\Delta$. Thus, if we find an x' such that $(1-\alpha)r_{\mathcal{P}}=(1-\alpha)\Gamma+(1-\frac{\Gamma}{\Delta})\Delta=\Delta F_{MP}(x')$, then x' can act as the upper limit of integration, thus yielding $\sum_{i=1}^{(1-\alpha)r_{\mathcal{P}}}\mathbf{1}(\sigma^2_{i+\alpha r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}})/\Gamma\leq x)=\sum_{i=1}^{r_{\mathcal{P}}}\mathbf{1}(\sigma^2_i(\mathbf{F}_{\mathcal{P}})/\Gamma\leq x)=F_{MP,\lambda}(x)$ because, as before, in the integration we have that $x\leq x'_1=F^{-1}_{MP}(\frac{\Gamma}{\Delta}(1-\alpha)+(1-\frac{\Gamma}{\Delta}))$. In essence, the change in the upper integration limit in (f) here (as compared to the same integration limit found the previous case of $\Delta\leq\Gamma$) accounts for the fact that the first $\Delta-\Gamma$ eigenvalues are now zero. This proves (f) and the entire lemma. □

We proceed with an additional lemma that bounds the approximation error of the design described in (6.14) and (6.15).

Lemma 26. The **D** and **E** design in (6.14) and (6.15), guarantees that

$$\|\mathbf{DE}-\mathbf{F}\|_F=\sqrt{\sum_{\mathcal{P}\in\mathcal{C}_1}\|\mathbf{F}^{\alpha}_{\mathcal{P}}-\mathbf{F}_{\mathcal{P}}\|^2_F}. \tag{6.23}$$

Proof. We first note that for

$$\mathbf{U}\triangleq\mathbf{DE}-\mathbf{F} \tag{6.24}$$

then following (6.14) and (6.15), we see that

$$\mathbf{U}(\mathcal{R}_{\mathcal{P}}, \mathcal{C}_{\mathcal{P}}) = \mathbf{D}^{\alpha}_{\mathcal{P}}\mathbf{E}^{\alpha}_{\mathcal{P}} - \mathbf{F}_{\mathcal{P}} = \mathbf{F}^{\alpha}_{\mathcal{P}} - \mathbf{F}_{\mathcal{P}} \tag{6.25}$$

where $\mathbf{F}_{\mathcal{P}}$ is defined in (5.59). Now we note that

$$\|\mathbf{U}\|_F = \sqrt{\sum_{(k,l)=(1,1)}^{(K,L)} \mathbf{U}^2(k,l)} \overset{(a)}{=} \sqrt{\sum_{\mathcal{P}\in\mathcal{C}_1} \sum_{(i,j)\in\mathcal{R}_{\mathcal{P}}\times\mathcal{C}_{\mathcal{P}}} \mathbf{U}^2(i,j)} \tag{6.26}$$

$$\overset{(b)}{=} \sqrt{\sum_{\mathcal{P}\in\mathcal{C}_1} \|\mathbf{U}(\mathcal{R}_{\mathcal{P}}, \mathcal{C}_{\mathcal{P}})\|_F^2} \tag{6.27}$$

$$\overset{(c)}{=} \sqrt{\sum_{\mathcal{P}\in\mathcal{C}_1} \|\mathbf{F}^{\alpha}_{\mathcal{P}} - \mathbf{F}_{\mathcal{P}}\|_F^2} \tag{6.28}$$

where (a) follows from the fact that the representative supports $\mathcal{P} \in \mathcal{C}_1$ partition matrix $\mathbf{F} \in \mathbb{R}^{K\times L}$[7], (b) follows from the definition of the Frobenius norm, while (c) follows as a direct consequence of (6.25). Combining (6.24) and (6.26) proves the lemma. □

We can now prove the theorem. For the case where $\Delta \leq \Gamma$, we have

$$\epsilon \overset{(a)}{=} \mathbb{E}_{\mathbf{F}}\{\|\mathbf{DE} - \mathbf{F}\|_F^2\}\frac{1}{KL} \tag{6.29}$$

$$\overset{(b)}{=} \mathbb{E}_{\mathbf{F}}\{\sum_{\mathcal{P}\in\mathcal{C}_1} \|\mathbf{F}^{\alpha}_{\mathcal{P}} - \mathbf{F}_{\mathcal{P}}\|_F^2\}\frac{1}{KL} \tag{6.30}$$

$$\overset{(c)}{=} \sum_{\mathcal{P}\in\mathcal{C}_1} \mathbb{E}_{\mathbf{F}}\{\|\mathbf{F}^{\alpha}_{\mathcal{P}} - \mathbf{F}_{\mathcal{P}}\|_F^2\}\frac{1}{KL} \tag{6.31}$$

$$\overset{(d)}{=} \frac{K}{\Delta}\frac{L}{\Gamma}\mathbb{E}_{\mathbf{F}}\{\|\mathbf{F}^{\alpha}_{\mathcal{P}} - \mathbf{F}_{\mathcal{P}}\|_F^2\}\frac{1}{KL} \tag{6.32}$$

$$\overset{(e)}{=} \int_{(1-\sqrt{\lambda})^2}^{F^{-1}_{MP,\lambda}(1-\alpha)} x f_{MP,\lambda}(x)\, dx \tag{6.33}$$

$$\overset{(f)}{=} \int_{(1-\sqrt{\frac{\delta\kappa}{\gamma}})^2}^{F^{-1}_{MP,\lambda}(1-T\frac{\gamma}{R})} x f_{MP,\lambda}(x)\, dx \tag{6.34}$$

where (a) follows from (6.5), (b) follows from (6.23), (c) results from the interchangeability of the statistical mean and the summation, (d) is true since the number of representative supports in $\mathcal{C}_1$ is $\frac{K}{\Delta}\frac{L}{\Gamma}$, (e) results from (6.22), and (f) results from the fact that $\alpha = T\frac{\gamma}{R}$, which follows from the fact that $r_{\mathcal{P}} = \min(\Delta, \Gamma) = \Delta$ and from the fact that $NT = \alpha\min(\Delta, \Gamma)\frac{K}{\Delta}\frac{L}{\Gamma} = \alpha\frac{KL}{\Gamma}$

[7]It means that the tiles are disjoint and cover the whole matrix $\mathbf{F}$.

(cf. (6.13),(6.14),(6.15)), which in turn says that $R = \frac{K}{N} = T\frac{\Gamma}{\alpha L} = T\frac{\gamma}{\alpha}$. The proof of (f) is concluded by noting that $\lambda = \frac{\Delta}{\Gamma} = \frac{\delta\kappa}{\gamma}$ (cf. (5.7)).

Similarly, for the case of $\Delta > \Gamma$, we have

$$\epsilon \overset{(a')}{=} \frac{K}{\Delta}\frac{L}{\Gamma}\mathbb{E}_{\mathbf{F}}\{\|\mathbf{F}_{\mathcal{P}}^{\alpha} - \mathbf{F}_{\mathcal{P}}\|_F^2\}\frac{1}{KL} \tag{6.35}$$

$$\overset{(b')}{=} \int_{(1-\sqrt{\lambda})^2}^{F_{MP,\lambda}^{-1}(1-\Gamma/\Delta\alpha)} x f_{MP,\lambda}(x)\, dx \tag{6.36}$$

$$\overset{(c')}{=} \int_{(1-\sqrt{\frac{\delta\kappa}{\gamma}})^2}^{F_{MP,\lambda}^{-1}(1-T\frac{\gamma}{R})} x f_{MP,\lambda}(x)\, dx \tag{6.37}$$

where (a') follows from equalities $(a), (b), (c), (d)$ corresponding to the above case of $\Delta \leq \Gamma$, where (b') results from (6.22), and where (c') results after substituting $\gamma/\kappa\delta = \Gamma/\Delta, \alpha = T\frac{\kappa\delta}{R}$, which follows after noting that $r_{\mathcal{P}} = \min(\Delta, \Gamma) = \Gamma$ and that $NT = \alpha\min(\Delta, \Gamma)\frac{KL}{\Delta\Gamma} = \alpha\frac{KL}{\Delta}$ (cf. (6.13),(6.14),(6.15)), which in turn says that $R = \frac{T\Delta}{\alpha L} = T\frac{\kappa\delta}{\alpha}$. The proof of (f) and of the entire lemma is concluded after recalling our asymptotic setting and also that $T|\min(\Delta, \Gamma)$ and that $\lambda = \frac{\delta\kappa}{\gamma}$ (cf. (5.7))..

6.3.3 Converse and Proof of Optimality

We will here provide a converse on the average normalized error. The converse will then allow us to show that under the disjoint balanced support assumption of Definition 12, the scheme proposed in this appendix is asymptotically optimal. First, Lemma 27 will offer a lower bound on the approximation error attributed to each tile, and then Lemma 28 proves that any scheme satisfying Definition 12, has to have disjoint and tiles with the same $|\mathcal{R}_{\mathcal{P}}|$ and $|\mathcal{C}_{\mathcal{P}}|$. Subsequently, Lemma 29 will establish the covering requirement for any optimal scheme. Combining the above will then yield the proof.

Our goal is to prove that in the limit of large N and constant $\delta, \gamma, \kappa, R$, and under the disjoint balanced support assumption, the average optimal reconstruction error is bounded as

$$\hat{\epsilon} \geq \Phi_{\mathrm{MP},\lambda}(t, r) = \int_r^t x f_{\mathrm{MP},\lambda}(x) dx \tag{6.38}$$

where $\lambda = \frac{\delta K}{\gamma L} = \frac{\Delta}{\Gamma}, r = (1 - \sqrt{\lambda})^2$, and where t is the solution to $F_{MP,\lambda}(t) = 1 - T\frac{\gamma N}{K}$. To achieve this, we begin with the following lemmas.

Lemma 27. Consider approximating any $\mathbf{F}$ by a product **DE** with limited Δ, Γ, and consider a disjoint set of m tiles $\mathcal{C} = \{\mathcal{P}_i\}_{i=1}^m$ (corresponding to

an arbitrary set of disjoint submatrices $\{\mathbf{F}_{\mathcal{P}_i}\}_{i=1}^{m}$ of $\mathbf{F}$) each allocated $\alpha_i r_{\mathcal{P}_i}$ contribution supports where $\alpha_i r_{\mathcal{P}_i} \in \mathbb{N}, \alpha_i \in \mathbb{R}, \alpha_i \leq 1$. Then

$$\sum_{i=1}^{m} \sum_{j=(1-\alpha_i)r_{\mathcal{P}_i}+1}^{r_{\mathcal{P}_i}} \sigma_j^2(\mathbf{F}_{\mathcal{P}_i}) \leq \|\mathbf{DE} - \mathbf{F}\|_F^2 \tag{6.39}$$

where $\sigma_j(\mathbf{F}_{\mathcal{P}_i}), j \in [r_{\mathcal{P}_i}], i \in [m]$ are the singular values of each $\mathbf{F}_{\mathcal{P}_i}$, in descending order.

Proof. The above can be seen by noting that

$$\sum_{i=1}^{m} \sum_{j=(1-\alpha_i)r_{\mathcal{P}_i}+1}^{r_{\mathcal{P}_i}} \sigma_j^2(\mathbf{F}_{\mathcal{P}_i}) \overset{(a)}{\leq} \sum_{i=1}^{m} \|\mathbf{D}_{\mathcal{P}_i}\mathbf{E}_{\mathcal{P}_i} - \mathbf{F}_{\mathcal{P}_i}\|_F^2 \tag{6.40}$$

$$\overset{(b)}{\leq} \|\mathbf{DE} - \mathbf{F}\|_F^2 \tag{6.41}$$

where, for $\mathbf{D}_{\mathcal{P}_i}\mathbf{E}_{\mathcal{P}_i}$ being a rank-$\alpha r_{\mathcal{P}_i}$ approximation of $\mathbf{F}_{\mathcal{P}_i}$, then (a) follows directly from the Eckart-Young Theorem that guarantees that $\sum_{j=(1-\alpha_i)r_{\mathcal{P}_i}+1}^{r_{\mathcal{P}_i}} \sigma_j^2(\mathbf{F}_{\mathcal{P}_i}) \leq \|\mathbf{D}_{\mathcal{P}_i}\mathbf{E}_{\mathcal{P}_i} - \mathbf{F}_{\mathcal{P}_i}\|_F^2$, while (b) follows after considering (5.59), after considering that the representative supports are disjoint, that the rank of $\mathbf{D}_{\mathcal{P}_i}\mathbf{E}_{\mathcal{P}_i}$ is $\alpha_i r_{\mathcal{P}_i}$, and after considering that parts of $\mathbf{F}$ may remain uncovered by the tiles. □

We proceed with the next lemma.

Lemma 28. For two matrices $\mathbf{D}, \mathbf{E}$, the representative supports $\{\mathbf{S}_{\mathcal{P}_i}\}_{i=1}^{m}$ of $\mathbf{DE}$ are disjoint (i.e., $\mathbf{S}_{\mathcal{P}_i} \cap \mathbf{S}_{\mathcal{P}_j} = \mathbf{0},\ j \neq i$) and $|\mathcal{R}_{\mathcal{P}_i}| = |\mathcal{R}_{\mathcal{P}_j}|, |\mathcal{C}_{\mathcal{P}_i}| = |\mathcal{C}_{\mathcal{P}_j}|\ \forall i, j \in [m]$ if and only if $\mathbf{D}$ and $\mathbf{E}$ accept the disjoint balanced support assumption of Definition 12.

Proof. The proof follows from the same proof of Lemma 3, after additionally now noting that for any class $\mathcal{P}_i \in \mathcal{C}$ then $|\mathcal{R}_{\mathcal{P}_i}| = \|\mathbf{I}(:, i)\|_0$ for some $i \in [NT]$, and after noting that due to the disjoint balanced support assumption of Definition 12 we have that $\|\mathbf{D}(:, i)\|_0 = \|\text{Supp}(\mathbf{D}(:, i))\|_0 = \|\mathbf{D}(:, j)\|_0 = \|\text{Supp}(\mathbf{D}(:, j))\|_0, \forall i \neq j, i, j \in [NT]$, which in turn implies that $|\mathcal{R}_{\mathcal{P}_i}| = |\mathcal{R}_{\mathcal{P}_j}|, \forall i, j \in [m]$. Similar considerations to $\mathcal{R}_{\mathcal{P}}$, also apply when focusing on $\mathcal{C}_{\mathcal{P}}$. □

We also have the following lemma.

Lemma 29. In the limit of large N and fixed R, $\frac{K}{L}$, and under the assumption of demand matrices $\mathbf{F} \in \mathbb{R}^{K \times L}$ drawn with *i.i.d* entries having zero mean and unit variance, then any optimal scheme that satisfies the disjoint balanced

support assumption of Definition 12 having fixed $\frac{|\mathcal{R}_{\mathcal{P}_i}|}{K}, \frac{|\mathcal{C}_{\mathcal{P}_j}|}{L}, \forall \mathcal{P}_i \in \mathcal{C} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m\}$, must guarantee that the union of all representative supports covers $\mathbf{F}$.

Proof. We prove this by contradiction. Suppose that there exists an optimal scheme whose union of representative supports (union of tiles) does not cover $\mathbf{F}$. Since the scheme satisfies Definition 12, by Lemma 18, we can let $|\mathcal{R}_{\mathcal{P}_i}| = |\mathcal{R}_{\mathcal{P}_j}| = X, \forall i, j \in [m]$ and $|\mathcal{C}_{\mathcal{P}_i}| = |\mathcal{C}_{\mathcal{P}_j}| = Y, \forall i, j \in [m]$. We also see due to the fact that the scheme does not cover $\mathbf{F}$, then

$$[K] \times [L] \backslash \cup_{\mathcal{P} \in \mathcal{C}} \mathcal{R}_{\mathcal{P}} \times \mathcal{C}_{\mathcal{P}} \neq \emptyset. \tag{6.42}$$

Furthermore we see that

$$\begin{aligned}
&[K] \times [L] \backslash \cup_{\mathcal{P} \in \mathcal{C}} \mathcal{R}_{\mathcal{P}} \times \mathcal{C}_{\mathcal{P}} && (6.43)\\
&= \cap_{\mathcal{P} \in \mathcal{C}} ([K] \times [L] \backslash \mathcal{R}_{\mathcal{P}} \times \mathcal{C}_{\mathcal{P}}) && (6.44)\\
&= \cap_{\mathcal{P} \in \mathcal{C}} ((([K] \backslash \mathcal{R}_{\mathcal{P}}) \times [L]) \cup ([K] \times [L] \backslash \mathcal{C}_{\mathcal{P}})) && (6.45)\\
&= \cup_{\mathcal{A} \subseteq \mathcal{C}} ((([K] \backslash \cup_{i \in \mathcal{A}} \mathcal{R}_{\mathcal{P}_i}) \times [L]) \cap ([K] \times [L] \backslash \cup_{i \in (\mathcal{C} \backslash \mathcal{A})} \mathcal{C}_{\mathcal{P}_I})) && (6.46)\\
&= \cup_{\mathcal{A} \subseteq \mathcal{C}} ([K] \backslash \cup_{i \in \mathcal{A}} \mathcal{R}_{\mathcal{P}_i} \times [L] \backslash \cup_{i \in (\mathcal{C} \backslash \mathcal{A})} \mathcal{C}_{\mathcal{P}_i}) \neq \emptyset && (6.47)
\end{aligned}$$

and thus there exists a subset $\mathcal{A} \subseteq \mathcal{C}$ such that $[K] \backslash \cup_{i \in \mathcal{A}} \mathcal{R}_{\mathcal{P}_i} \times [L] \backslash \cup_{i \in (\mathcal{C} \backslash \mathcal{A})} \mathcal{C}_{\mathcal{P}_i} \neq \emptyset$, which in turn says that $[K] \backslash \cup_{i \in \mathcal{A}} \mathcal{R}_{\mathcal{P}_i} \neq \emptyset$ and $[L] \backslash \cup_{i \in (\mathcal{C} \backslash \mathcal{A})} \mathcal{C}_{\mathcal{P}_i} \neq \emptyset$. Since $|[K] \backslash \cup_{i \in \mathcal{A}} \mathcal{R}_{\mathcal{P}_i}| = K - |\cup_{i \in \mathcal{A}} \mathcal{R}_{\mathcal{P}_i}|$, then from Definition 12 and Lemma 3, we see that either $\mathcal{R}_{\mathcal{P}_i} = \mathcal{R}_{\mathcal{P}_j}, \forall i, j \in \mathcal{A}$ or that $\mathcal{R}_{\mathcal{P}_i} \cap \mathcal{R}_{\mathcal{P}_j} = \emptyset, \forall i \neq j, i, j \in \mathcal{A}$. We also recall that $|\mathcal{R}_{\mathcal{P}_i}| = X, \forall i \in [m]$, and thus that X divides $|\cup_{i \in \mathcal{A}} \mathcal{R}_{\mathcal{P}_i}|$. Since also X divides K, we can see that X divides $|[K] \backslash \cup_{i \in \mathcal{A}} \mathcal{R}_{\mathcal{P}_i}|$. A similar argument can be made to show that Y divides $|[L] \backslash \cup_{i \in (\mathcal{C} \backslash \mathcal{A})} \mathcal{C}_{\mathcal{P}_i}|$.

Now, to show that this assumption results in a contradiction, we construct another scheme for the same set of parameters and analyse its performance. We first recall that $NT = \sum_{i=1}^{m} \operatorname{rank}(\mathbf{F}_{\mathcal{P}}^{\alpha}) = \sum_{i=1}^{m} \alpha_i r_{\mathcal{P}_i}$, i.e., that the sum of the ranks of the matrices $\mathbf{F}_{\mathcal{P}}^{\alpha}$ (cf. (6.16)) across $\mathcal{P} \in \mathcal{C}$ is equal to NT. Assuming, without loss of generality, that $X \leq Y$, yields $r_{\mathcal{P}} = X, \forall \mathcal{P} \in \mathcal{C}$ which in turn yields

$$NT = \sum_{i=1}^{m} \alpha_i r_{\mathcal{P}_i} = \sum_{i=1}^{m} \alpha_i X. \tag{6.48}$$

Now, consider another scheme with the same positioning of tiles, though now an additional tile $\mathcal{P}_{m+1}$, with a set of row indices $\mathcal{R}_{\mathcal{P}_{m+1}} \subseteq \cup_{i \in \mathcal{A}} \mathcal{R}_{\mathcal{P}_i}, |\mathcal{R}_{\mathcal{P}}| = X$ and column indices $\mathcal{C}_{\mathcal{P}_{m+1}} \subseteq [L] \backslash \cup_{i \in (\mathcal{C} \backslash \mathcal{A})} \mathcal{C}_{\mathcal{P}_i}, |\mathcal{C}_{\mathcal{P}_{m+1}}| = Y$. Let the now additionally introduced tiles be placed inside the previously "uncovered" area of the previous scheme, which means that $\mathcal{R}_{\mathcal{P}_{m+1}} \times \mathcal{C}_{\mathcal{P}_{m+1}} \subseteq [K] \times [L] \backslash \cup_{\mathcal{P} \in \mathcal{C}} \mathcal{R}_{\mathcal{P}} \times \mathcal{C}_{\mathcal{P}}$.

Let us now choose $\beta < \alpha_1, \beta r_{\mathcal{P}_1} \in \mathbb{N}$ and then let $\mathbf{F}_{\mathcal{P}_1}$ be now estimated by a matrix $\mathbf{F}_{\mathcal{P}_1}^{\alpha_1-\beta}$ of rank $(\alpha_1 - \beta)X$ (rather than by matrix $\mathbf{F}_{\mathcal{P}_1}^{\alpha_1}$ of rank $\alpha_1 X$.). We now recall that the error of the first tile of the optimal scheme is

$$\|\mathbf{F}_{\mathcal{P}_1}^{\alpha_1} - \mathbf{F}_{\mathcal{P}_1}\|_F = \sum_{i=1}^{(1-\alpha_1)r_{\mathcal{P}}} \sigma^2_{i+\alpha_1 r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}}) \tag{6.49}$$

which we know to be optimal from the Eckart-Young Theorem. Now we see that the error corresponding to the first tile, for the new scheme — which again employs the optimal truncated SVD method — takes the form

$$\|\mathbf{F}_{\mathcal{P}_1}^{\alpha_1-\beta} - \mathbf{F}_{\mathcal{P}_1}\|_F = \sum_{i=1}^{(1-\alpha_1+\beta)r_{\mathcal{P}}} \sigma^2_{i+(\alpha_1-\beta)r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}}). \tag{6.50}$$

For the same new scheme, the last used $\mathbf{F}_{\mathcal{P}_{m+1}}$ is being approximated by matrix $\mathbf{F}_{\mathcal{P}_{m+1}}^{\beta}$, which has rank $\beta r_{\mathcal{P}_{m+1}} = \beta X$, yielding a last-tile error of the form

$$\|\mathbf{F}_{\mathcal{P}_{m+1}}^{\beta} - \mathbf{F}_{\mathcal{P}_{m+1}}\|_F = \sum_{i=1}^{(1-\beta)r_{\mathcal{P}}} \sigma^2_{i+\beta r_{\mathcal{P}}}(\mathbf{F}_{\mathcal{P}}). \tag{6.51}$$

We also note that both the original and the new scheme incur $NT = (\alpha_1 - \beta)r_{\mathcal{P}_1} + \sum_{i=2}^{m} \alpha_i r_{\mathcal{P}_i} + \beta r_{\mathcal{P}_{m+1}} = ((\alpha_1 - \beta) + \sum_{i=2}^{m} \alpha_i + \beta)X = \sum_{i=1}^{m} \alpha_i X$ which comes from (6.48).

Now we can see that the average normalized error of the scheme can be thus expressed as

$$\epsilon_{opt} = \frac{\mathbb{E}_F\{\sum_{i=1}^{(1-\alpha_1)X} \sigma_{i+\alpha_1 X}(\mathbf{F}_{\mathcal{P}_i})\}}{KL} + \epsilon_{\{2,\dots,m\}} \tag{6.52}$$

$$+ \frac{\mathbb{E}_{\mathbf{F}}\{\sum_{(i,j)\in[K]\times[L]\setminus\cup_{\mathcal{P}\in\mathcal{C}}\mathcal{R}_{\mathcal{P}}\times\mathcal{C}_{\mathcal{P}}} \mathbf{F}(i,j)^2\}}{KL} \tag{6.53}$$

where the first fraction describes the error due to the first tile (cf. (6.49)), the second term $\epsilon_{\{2,\dots,m\}}$ is the average normalized error of the second to last tiles, while the last term represents the Frobenius norm for the part of the matrix that represents the "hole" left uncovered. Then the average normalized error of the constructed scheme can be expressed as

$$\epsilon_c = \frac{\mathbb{E}_F\{\sum_{i=1}^{(1-\alpha_1+\beta)X} \sigma_{i+(\alpha_1-\beta)X}(\mathbf{F}_{\mathcal{P}_i})\}}{KL} + \epsilon_{\{2,\dots,m\}}$$

$$+ \frac{\mathbb{E}_{\mathbf{F}}\{\sum_{(i,j)\in[K]\times[L]\setminus\cup_{\mathcal{P}\in\mathcal{C}}\mathcal{R}_{\mathcal{P}}\times\mathcal{C}_{\mathcal{P}}} \mathbf{F}(i,j)^2 - \sum_{i=1}^{\beta X} \sigma_i(\mathbf{F}_{\mathcal{P}_{m+1}})\}}{KL} \tag{6.54}$$

where the first term corresponds to the first tile (cf. (6.50)), the second term $\epsilon_{\{2,\dots,m\}}$ is again the average normalized error corresponding to the second to mth tiles, while the last error term is attributed to tile $m+1$ and to the still uncovered area. This last expression is simplified after considering the summation of the average Frobenious norm of the uncovered area that yields error $\mathbb{E}_{\mathbf{F}}\{\sum_{(i,j)\in[K]\times[L]\setminus\cup_{\mathcal{P}\in\mathcal{C}}\mathcal{R}_{\mathcal{P}}\times\mathcal{C}_{\mathcal{P}}}\mathbf{F}(i,j)^2 - \|\mathbf{F}_{\mathcal{P}_{m+1}}\|_{\mathbf{F}}\}$, as well as after considering that the error $\mathbb{E}_{\mathbf{F}}\{\|\mathbf{F}^{\beta}_{\mathcal{P}_{m+1}}-\mathbf{F}_{\mathcal{P}_{m+1}}\|_F\}$ of the $(m+1)$th tile takes the form in (6.51), and finally after noting that $\|\mathbf{F}^{\beta}_{\mathcal{P}_{m+1}}-\mathbf{F}_{\mathcal{P}_{m+1}}\|_F - \|\mathbf{F}_{\mathcal{P}_{m+1}}\|_{\mathbf{F}} = -\sum_{i=1}^{\beta X}\sigma_i(\mathbf{F}_{\mathcal{P}_{m+1}})$. Thus we can conclude that the normalized error attributed to tile $m+1$ and to the still uncovered area, takes the form of the numerator of (6.54).

Thus we can now see that

$$\epsilon_{opt}-\epsilon_c = \frac{XY}{KL}\Big(-\int_{F^{-1}_{MP,\lambda}(1-\alpha_1)}^{F^{-1}_{MP,\lambda}(1-\alpha_1+\beta)} x f_{MP,\lambda}(x)\,dx \tag{6.55}$$

$$+\int_{F^{-1}_{MP,\lambda}(1-\beta)}^{F^{-1}_{MP,\lambda}(1)} x f_{MP,\lambda}(x)\,dx\Big),\ \lambda=\frac{X}{Y} \tag{6.56}$$

which follows after considering (6.53), (6.54), as well as after following the corresponding steps found in the proof of Lemma 25. We can now also note that

$$\begin{aligned}
&\frac{d}{dt}\int_{F^{-1}_{MP,\lambda}(t-\beta)}^{F^{-1}_{MP,\lambda}(t)} x f_{MP,\lambda}(x)\,dx\\
&=\frac{1}{f_{MP,\lambda}(F^{-1}_{MP,\lambda}(t))}F^{-1}_{MP,\lambda}(t)f_{MP,\lambda}(F^{-1}_{MP,\lambda}(t))\\
&-\frac{1}{f_{MP,\lambda}(F^{-1}_{MP,\lambda}(t-\beta))}F^{-1}_{MP,\lambda}(t-\beta)f_{MP,\lambda}(F^{-1}_{MP,\lambda}(t-\beta))\\
&=F^{-1}_{MP,\lambda}(t)-F^{-1}_{MP,\lambda}(t-\beta)>0
\end{aligned}$$

where the last inequality is due to the fact that $F^{-1}_{MP,\lambda}(t)$ is a strictly monotonically increasing function of t, which means that the function $\int_{F^{-1}_{MP,\lambda}(t-\beta)}^{F^{-1}_{MP,\lambda}(t)} x f_{MP,\lambda}(x)dx$ for all $\beta\le t\le 1$ is a strictly monotonically increasing function of t, which now yields $\int_{F^{-1}_{MP,\lambda}(1-\beta)}^{F^{-1}_{MP,\lambda}(1)} x f_{MP,\lambda}(x)\,dx > \int_{F^{-1}_{MP,\lambda}(1-\alpha_1)}^{F^{-1}_{MP,\lambda}(1-\alpha_1+\beta)} x f_{MP,\lambda}(x)\,dx$, and thus — after considering (6.56) — yields $\epsilon_{opt}-\epsilon_c>0$, which contradicts our initial optimality assumption. By showing that there can always be a scheme that offers strictly lower error than the best scheme that fails to fully cover $\mathbf{F}$, we prove that optimal schemes must cover $\mathbf{F}$. □

Continuing with our main proof, we recall that $\mathbf{F}$ is a $K \times L$ real random matrix whose entries are *i.i.d* with zero mean and unit variance. In this context, and in the limit of large N, we recall Lemma 18 and Definition 12, and after seeing that $|\mathcal{R}_{\mathcal{P}}| = X, |\mathcal{C}_{\mathcal{P}}| = Y, \forall \mathcal{P} \in \mathcal{C}$, we first consider the case of $X \leq Y$, where we have $r = \frac{X}{Y}$ and $\sum_{i=1}^{m} \alpha_i X = NT$, and proceed to see that

$$
\begin{aligned}
&\mathbb{E}_{\mathbf{F}}\{\sum_{i=1}^{m} \sum_{j=(1-\alpha_i)r_{\mathcal{P}_i}+1}^{r_{\mathcal{P}_i}} \sigma_j^2(\mathbf{F}_{\mathcal{P}_i})\} \\
&\overset{(a)}{=} Y\mathbb{E}_{\mathbf{F}}\Big\{\frac{\sum_{i=1}^{m}\sum_{j=1}^{\alpha_i X}\sigma^2_{j+(1-\alpha_i)X}(\mathbf{F}_{\mathcal{P}_i})}{Y}\Big\} \\
&\overset{(b)}{=} Y\mathbb{E}_{\mathbf{F}}\Big\{\sum_{i=1}^{m}\sum_{j=1}^{\alpha_i X}\int_{-\infty}^{\infty}\delta(x-\frac{\sigma^2_{j+(1-\alpha_i)X}(\mathbf{F}_{\mathcal{P}_i})}{Y})xdx\Big\} \\
&\overset{(c)}{=} Y\mathbb{E}_{\mathbf{F}}\Big\{\sum_{i=1}^{m}\sum_{j=1}^{\alpha_i X}\int_{-\infty}^{\infty}(1/\delta x)\mathbf{1}(\frac{\sigma^2_{j+(1-\alpha_i)X}(\mathbf{F}_{\mathcal{P}_i})}{Y} \leq x+\delta x) \\
&- \mathbf{1}(\frac{\sigma^2_{j+(1-\alpha_i)X}(\mathbf{F}_{\mathcal{P}_i})}{Y} \leq x)xdx\Big\} \\
&\overset{(d)}{=} Y\sum_{i=1}^{m}\mathbb{E}_{\mathbf{F}}\Big\{\sum_{j=1}^{\alpha_i X}\int_{-\infty}^{\infty}(1/\delta x)\mathbf{1}(\frac{\sigma^2_{j+(1-\alpha_i)X}(\mathbf{F}_{\mathcal{P}_i})}{Y} \leq x+\delta x) \\
&- \mathbf{1}(\frac{\sigma^2_{j+(1-\alpha_i)X}(\mathbf{F}_{\mathcal{P}_i})}{Y} \leq x)xdx\Big\} \\
&\overset{(e)}{=} YX\sum_{i=1}^{m}\int_{(1-\sqrt{r})^2}^{F^{-1}_{MP,\lambda}(1-\alpha_i)}(1/\delta x)[F_{MP}(x+\delta x)-F_{MP,\lambda}(x)]xdx \\
&\overset{(f)}{=} YX\sum_{i=1}^{m}\int_{(1-\sqrt{r})^2}^{F^{-1}_{MP,\lambda}(1-\alpha_i)} f_{MP}(x)xdx \\
&\overset{(g)}{\geq} YXm\int_{(1-\sqrt{r})^2}^{F^{-1}_{MP,\lambda}(1-\frac{NT}{mX})} f_{MP}(x)xdx \\
&\overset{(h)}{=} KL\int_{(1-\sqrt{r})^2}^{F^{-1}_{MP,\lambda}(1-\frac{NTY}{KL})} f_{MP}(x)xdx \\
&\overset{(i)}{\geq} KL\int_{(1-\sqrt{\frac{\Delta}{\Gamma}})^2}^{F^{-1}_{MP,\lambda}(1-\frac{NT\Gamma}{KL})} f_{MP}(x)xdx
\end{aligned}
$$

where (a) follows since Y is a constant, (b) follows from basic properties of the delta Dirac function, (c) follows from the fact that the delta Dirac function is a derivative of the step function, (d) follows from the fact that expectation and sum are interchangeable functions, (e) follows from the same reasoning that allowed equality (f) for the case $\Delta \leq \Gamma$ of the proof of Lemma 25, and (f)

here follows from the fact that f_{MP} is the derivative of F_{MP}. Furthermore, (g) follows by first setting $g(\alpha_i) \triangleq \int_{(1-\sqrt{r})^2}^{F^{-1}_{MP,\lambda}(1-\alpha_i)} f_{MP}(x)$, and then by recalling that $\sum_{i=1}^{m} \alpha_i = \frac{NT}{X}$, and subsequently by noting that $g(\alpha_i)$ is a monotonic decreasing convex function of $\alpha_i, \forall 0 \leq \alpha_i \leq 1$, which in turn implies that its Lagrange optimizer function will be $\mathcal{L}(\alpha_1, \alpha_2, \ldots, \alpha_m) = Y \sum_{i=1}^{m} g(\alpha_i) - \lambda(\sum_{i=1}^{m} \alpha_i - \frac{NT}{X})$, where to ensure $\frac{\partial \mathcal{L}}{\partial \alpha_i} = 0, \forall i \in [m]$ we have $\frac{\partial g(\alpha_i)}{\partial \alpha_i} = F^{-1}_{MP,\lambda}(1 - \frac{NT}{mX}) = \lambda, \forall \alpha_i$, which in turn means that $\alpha_1 = \alpha_2 = \ldots = \alpha_m = \alpha$ which, combined with the optimization constraints, yields that $\alpha = \frac{NT}{mX}$. Then (h) results by noting that $m = \frac{KL}{XY}$, which itself holds since, by Lemma 29, we have that for any asymptotic scheme with bounded normalized error, the representative supports or tiles must cover the whole matrix $\mathbf{F}$. Finally (i) follows from the fact that $X \leq \Delta, Y \leq \Gamma$ and after seeing that the derivative of the RHS side of (h) with respect X is $\frac{KL}{X}((-2(1 - \sqrt{\frac{X}{Y}}))(\frac{1}{Y})(-\frac{1}{2})\frac{X}{Y}^{-1/2} f_{MP}((1 - \sqrt{X/Y})^2)(1-\sqrt{\frac{X}{Y}})^2) - \frac{KL}{X^2} \int_{(1-\sqrt{\frac{X}{Y}})^2}^{F^{-1}_{MP}(1-\frac{NT}{KL}Y)} x f_{MP,\lambda}(x) dx$, as well as by noting that since $f_{MP}((1 - \sqrt{X/Y})^2) = 0$, the derivative of the RHS side of (h) is always negative, which in turn means that the RHS side of (h) is a monotonically decreasing function of X which in turn yields that $X = \Delta$ gives the optimal solution. This is then combined with the fact that the derivative of the RHS side of (h) with respect regarding Y is $-\frac{NT}{KL} \frac{1}{f_{MP}(F^{-1}(1-\frac{NTY}{KL}))} F^{-1}_{MP,\lambda}(1 - \frac{NTY}{KL})$ $f_{MP}(F^{-1}_{MP,\lambda}(1 - \frac{NTY}{KL})) - 2(1-\sqrt{\frac{X}{Y}})(-\frac{1}{2})(\frac{X}{Y})^{-1/2} XY^{-2} = -\frac{NT}{KL} F^{-1}_{MP}(1-\frac{NTY}{KL}) -$ $(1 - \sqrt{X/Y})(\frac{X}{Y})^{-1/2} XY^{-2}$. Finally, since $X \leq Y$, this derivative is also negative, which in turn means that $Y = \Gamma$, with the same reasoning as the case where $X = \Delta$. This proves that

$$\mathbb{E}_{\mathbf{F}}\{\|\mathbf{DE} - \mathbf{F}\|_F^2\} \geq KL \int_{(1-\sqrt{\frac{\delta\kappa}{\gamma}})^2}^{F^{-1}_{MP,\lambda}(1-T\frac{\gamma}{R})} x f_{MP,\lambda}(x)\, dx \tag{6.57}$$

for the case of $X \leq Y$. As a result, this proves the lower bound in (6.38), and thus the entire lower bound (converse) of Theorem 10, over all schemes for which $|\mathcal{R}_{\mathcal{P}}| \geq |\mathcal{C}_{\mathcal{P}}|, \forall \mathcal{P} \in \mathcal{C}$.

Now for the case of $Y \leq X$, similarly we have $r = \frac{X}{Y}$, and now with $\sum_{i=1}^{m} \alpha_i Y = NT$, we see that

$$\begin{aligned}
&\mathbb{E}_{\mathbf{F}}\{\sum_{i=1}^{m} \sum_{j=(1-\alpha_i)r_{\mathcal{P}_i}+1}^{r_{\mathcal{P}_i}} \sigma_j^2(\mathbf{F}_{\mathcal{P}_i})\} \\
&\overset{(a,b,c,d)}{=} Y \sum_{i=1}^{m} \mathbb{E}_{\mathbf{F}}\Big\{ \sum_{j=1}^{\alpha_i Y} \int_{-\infty}^{\infty} (1/\delta x) \mathbf{1}\Big(\frac{\sigma^2_{j+(1-\alpha_i)X}(\mathbf{F}_{\mathcal{P}_i})}{Y} \leq x + \delta x\Big)
\end{aligned}$$

$$
\begin{aligned}
&- \mathbf{1}(\frac{\sigma^2_{j+(1-\alpha_i)X}(\mathbf{F}_{\mathcal{P}_i})}{Y} \leq x)xdx\Big\} \\
&\overset{(e)}{=} YX\sum_{i=1}^{m}\int_{(1-\sqrt{r})^2}^{F^{-1}_{MP,\lambda}(1-\frac{Y}{X}\alpha_i)}(1/\delta x)[F_{MP}(x+\delta x)-F_{MP,\lambda}(x)]xdx \\
&\overset{(f)}{=} YX\sum_{i=1}^{m}\int_{(1-\sqrt{r})^2}^{F^{-1}_{MP,\lambda}(1-\frac{Y}{X}\alpha_i)} f_{MP}(x)xdx \\
&\overset{(g)}{\geq} YXm\int_{(1-\sqrt{r})^2}^{F^{-1}_{MP,\lambda}(1-\frac{NT}{mX})} f_{MP}(x)xdx \\
&\overset{(h)}{=} KL\int_{(1-\sqrt{r})^2}^{F^{-1}_{MP,\lambda}(1-\frac{NTY}{KL}))} f_{MP}(x)xdx \\
&\overset{(i)}{\geq} KL\int_{(1-\sqrt{\frac{\Delta}{\Gamma}})^2}^{F^{-1}_{MP,\lambda}(1-\frac{NT\Gamma}{KL})} f_{MP}(x)xdx
\end{aligned}
$$

where the first equality is proven the same way as the equalities (a, b, c, d) in the previous case of $X \leq Y$, and where here (e) is proven similar to equality (f) of the case $\Delta > \Gamma$ in the proof of Lemma 25. Furthermore, here, (f) follows from the fact that f_{MP} is the derivative of F_{MP}. To establish (g), we first consider $g(\alpha_i) \triangleq \int_{(1-\sqrt{r})^2}^{F^{-1}_{MP,\lambda}(1-\alpha_i)} xf_{MP}(x)dx$ subject to $\sum_{i=1}^{m}\alpha_i = \frac{NT}{Y}$, and then note that $g(\alpha_i)$ is a monotonic decreasing convex function of α_i, in the range $\alpha_i \in \mathbb{R}$, thus allowing us to conclude that its Lagrange optimizer function will be $\mathcal{L}(\alpha_1, \alpha_2, \ldots, \alpha_m) = Y\sum_{i=1}^{m} g(\alpha_i) - \lambda(\sum_{i=1}^{m}\alpha_i - \frac{NT}{Y})$ where to ensure $\frac{\partial \mathcal{L}}{\partial \alpha_i} = 0, \forall i \in [m]$, we have $\frac{\partial g(\alpha_i)}{\partial \alpha_i} = F^{-1}_{MP,\lambda}(1-\frac{NT}{m}) = \lambda, \forall \alpha_i$, which means that $\alpha_1 = \alpha_2 = \ldots = \alpha_m = \alpha$, which is then combined with the optimization constraints to yield that $\alpha = \frac{NT}{mY}$. Subsequently, (h) results after noting that $m = \frac{KL}{XY}$ after recalling Lemma 29 which tells us that for any asymptotic scheme with normalized error, the representative supports must cover the entire $\mathbf{F}$. Finally, (i) follows directly as inequality (i) of the case of $X \leq Y$. As a result, this proves the lower bound in (6.38), and thus the entire lower bound (converse) of Theorem 10, over all schemes for which $|\mathcal{R}_{\mathcal{P}}| \leq |\mathcal{C}_{\mathcal{P}}|, \forall \mathcal{P} \in \mathcal{C}$.

Chapter 7

Conclusion, Open Problems and Future Works

In this thesis, we have introduced a new multi-user distributed-computation setting for computing from the broad class of linearly-decomposable functions from modeling the setting on real numbers and natural numbers. Our research unveiled the relation between distributed computing and the challenge of decomposing a 'functions' matrix $\mathbf{F}$ into two ideally sparse matrices: the encoding matrix $\mathbf{E}$ and the decoding matrix $\mathbf{D}$. Furthermore, we established a novel connection to covering codes, highlighting their significance in distributed computing quandaries and sparse matrix factorization within finite fields. Additionally, this thesis introduced the notion of partial covering codes and emphasized the necessity for codes proficient in covering smaller subsets of the overarching vector space. Expanding upon this new category of codes, which serves as an extension of covering codes, we presented several enhancements and broader implications of established findings in the literature

Furthermore, our investigation delved into the multi-shot setting, after we thoroughly examined the advantages of the single-shot approach. Our analysis happens on both real numbers and finite field scenarios, revealing that the benefits derived from augmenting T exhibit an unbounded and strictly increasing gain in the realm of large T. Conversely, within the domain of finite T, these benefits exhibit a strictly increasing pattern beyond a certain threshold value of T. Consequently, we can infer, as previously suggested, that the reductions in computation attributable to larger T stem, at least partially, from the heightened precision in transmission facilitated by a larger T, rather than solely from an increased communication cost. Also, in the real numbers scenarios of lossy tessellated distributed computing, we observed a decrease in the normalized reconstruction error as a consequence of the

increase in the number of shots (cf. Figure 6.5).

Our introduced normalized computation metrics consist of, the computation cost per server constraint γ_f introduced in Chapter 2, representing the normalized maximum fraction of all servers that must compute any subfunction, $\gamma_c = \frac{\Gamma_c}{NL}$ introduced in Chapters 3 and 4, representing the normalized number of cumulative computation costs done by all the servers overall subfunctions and $\gamma = \frac{\Lambda}{L} = \frac{\Gamma}{L}$, introduced and used in Chapters 3, 5 and 6, representing the maximum number of subfunctions assigned to a server and modelling the delay of the synchronous distributed system, has the following relationships:

$$\gamma_c \leq \gamma_f \tag{7.1}$$

$$\gamma_c \leq \gamma \tag{7.2}$$

note that in Theorem 2, we proved that $\gamma_f \leq H_q^{-1}(\frac{K}{N})$ in the single-shot setting, where the formulation has been done in finite fields. Using (7.2), we see that $\frac{\Gamma_c}{NL}$, defined in Chapter 3, can also be upper bounded by $H_q^{-1}(\frac{K}{N})$. Note that if q is sufficiently large, we have that $H_q^{-1}(\frac{K}{N}) \simeq \frac{K}{N}$, therefore we have $\gamma_f, \gamma_c \leq \frac{K}{N}$. On the other hand, using a completely different approach in Chapter 4, in Proposition 7, we derived that $\gamma_c \leq \frac{K}{N}$, also via another distinct approach in Chapter 5, Theorem 9, we have that $C = \gamma$ when $\zeta < \gamma$, which means that $\frac{K}{N_{opt}} = \gamma$. It is not obvious why this observation occurred and what logic is behind it. Why the results in the finite field modelling coincided with the results in the real field approaches when the size of the field q approached infinity? The answer to this question may pave the way for researchers in the field to be able to transform any result in the finite field into the real numbers field.

In terms of communication cost, we have introduced two new metrics the first one is δ_c, representing the average fraction of servers that each user gets data from, and δ, representing the maximum number of transmissions from a server to the other users. The same relation as of (7.2), holds for the defined communication costs $\delta_c \leq \delta$.

Furthermore, our setting can also apply to a broad range of 'well-behaved' functions and thus can enjoy several use cases, some of which are suggested in our introduction (see also [22] for additional motivation of the linearly separable function computation problem).

Our work naturally relates to the recent results in [22] that considered the single-user linearly-separable distributed computing scenario, where a single user may request multiple linearly-separable functions. In this setting in [22], as well as in the extended works in [42] and [89], a key ingredient is the presence of straggling servers, while another key ingredient is that the

subfunction-assignment is fixed and oblivious of the actual functions requested by the user.

Our analysis also applies to the transposed computing problem corresponding to $\mathbf{E}^\intercal\mathbf{D}^\intercal = \mathbf{F}^\intercal$. In that scenario, there will be a dual analysis by replacing the computation and communication cost concepts, with each other in each of the Chapters and swapping the K, the number of users with L the number of subfiles. One direction for future investigations would to be delve more into the results of this dual of the problem.

This thesis also established various upper and lower bounds on the computational delay Γ and the cumulative computation cost $\Gamma_c \in [0, NL]$, and revealed new connections to the packing and covering capabilities of codes thus revealing for the first time powerful connections with the class of perfect codes.

Also, relating our distributed computing to compressed sensing and driving new results in terms of cumulative computation cost, shows us how a compressed sensing approach and mathematical single-processing techniques can be useful in characterizing the fundamental limits of multi-user distributed systems. One of the approaches was to look at the problem from the perspective of the fixed-support matrix factorization problem raised from the sparse matrix decomposition literature, which leads to the results of the Tesselated Distributed Computing, described in Chapters 5 and 6.

The introduction of tesselated distributed computing, discussed in both Chapters 5 and 6, marks a significant stride in linking the multi-user linearly-decomposable problem with various realms, including sparse matrix factorization, fixed-support matrix factorization, tiling literature, distributed computing challenges, and the statistical landscape of large matrices. In tackling the lossy scenario, which presents the intricate problem of approximate matrix factorization, our approach harnessed asymptotic methods and stochastic metrics. These tools allowed us to delineate, for the first time, a precise boundary on the optimal normalized reconstruction error vis-à-vis communication and computational resources. Remarkably, this boundary holds firm under the assumption of disjoint supports, with straightforward schemes achieving this pinnacle through elemental tilling techniques in conjunction with truncated SVD decompositions.

7.1 Future Works and Open Problems

As suggested above, our setting can apply to a broad range of 'well-behaved' functions and thus can enjoy several use cases and theoretical expansion. We can name and describe the following direction as the possible future works as

follows:

- **Hierarchical Scenarios:** An additional new direction that our work can extend is the so-called hierarchical or tree-like scenario introduced in [80], [92] whose purpose is to ameliorate bandwidth limitations and straggler effects in distributed gradient coding [18]. In this hierarchical setting, each user[1] is connected to a group of servers in a hierarchical manner. In particular, each user computes a linearly separable function based on its locally available data and then sends this to the 'Aggregator' that finally computes the gradient. that allows for a hierarchical aggregation of the sub-gradients. Our approach can extend the hierarchical model by allowing the users to connect to any subset of servers, as well as by allowing them to deviate from the single-shot assumption.

- **Transpose Variant:** Our analysis also applies to the transposed computing problem corresponding to $\mathbf{E}^{\intercal}\mathbf{D}^{\intercal} = \mathbf{F}^{\intercal}$, on that case the provided bound will be on the communication cost per user where K the number of users has to be significantly larger than L, the number of files while L/N (the ratio between the number of files and servers) remains fixed since in the above case $\mathbf{E}^{\intercal}$ would be seen as the parity-check matrix of a partial covering code, not $\mathbf{D}$. We can see that this scenario investigates a less practical scenario where K, the number of users is much bigger than the number of subfunctions.

- **Different Entropy for Transmitted Symbols:** we can conceive of a setting where the size of the transmitted signals sent by different servers $\mathbf{z}_n$, $n \in [N]$, in the single shot scenario differs from each other. More precisely, in this chapter we assumed that the output of each subfunction, so-called file-output $f_\ell(.)$ is just a member of $\mathbf{GF}(q)$ (Cf. (2.4)) so that the model captures the most general and simple instance of the multi-user linearly separable distributed computing problem, because of that we see the transmitted signals $\mathbf{z}_n, n \in [N]$ can be any member of $\mathbf{GF}(q)$ since the signal is just a linear combination of the files and mathematically $\mathbf{z}_n$ can also be any element of $\mathbf{GF}(q)$, therefore in this system model, there is no difference between any of the transmitted signals and the communication cost is simply the total number of activated links. If we were to analyse the system model where there might exist two transmitted signals $\mathbf{z}_1$ and $\mathbf{z}_2$, with two different sizes then we have to also define a probability measure on each of the output files. For instance, we might investigate the case where

[1] In [80], these users are referred to as master nodes.

W_1 has a non-uniform distribution, or apriori the master knows that $W_1 \neq 0$ while other files have a uniform distribution. In this case, if the master node allocated D_1 to the server 1 but not to the server 2, then $\mathbf{z}_1$ consists of W_1 and other files and $\mathbf{z}_2$ does not contain W_1 in their linear decomposition, then $\mathbf{z}_1$ has a non-uniform distributed while $\mathbf{z}_2$ has a uniform distribution, which makes $H(\mathbf{z}_1) < H(\mathbf{z}_2)$, where H is an entropy defined on random variables $\mathbf{z}_1$ and $\mathbf{z}_2$. We see that this results in an interesting problem where the communication cost also has to be dependent on the output files distribution, which might be dependent on some kind of weighed sparsity criteria of both $\mathbf{D}$ and $\mathbf{E}$.

- **Achieving a Partial Covering Code for a Specific Set of Syndromes:** An interesting open problem is to design an algorithm so that for a given set of syndromes $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_L\}$ where $\mathbf{s} \in \mathbf{GF}(q)^{n-k}$, design an optimal partial covering code $\mathcal{C}(k, n)$, which means to design a code that covers the points in $\mathcal{X} = \{\mathbf{x} | \mathbf{H}_{\mathcal{C}}\mathbf{x} = \mathbf{s}, \mathbf{s} \in \mathcal{S}\}$ with partial covering radius $\rho_{\mathcal{X}} \simeq H_q^{-1}(\frac{\log_q(L)}{n})$.

- **Partial Perfect Codes:** As one may inspect, the code construction of Theorems 6 and7 in Chapter 3, is only optimal for the computational delay and cumulative computation cost, where $L = q^K$ and $\mathbf{F}$ columns are all different, which is regarded a the syndromes of a perfect code. What will be the result when $\mathbf{F}$ does not include all the possible variants of syndromes?

- **Eliminate The Uniqueness Assumption Property of The Sparsest Solution in The Compressed Sensing Literature** Recall that result in Theorem 8 automatically accepts an additional uniqueness property — on the sparsest solution $\mathbf{x}$ in (4.14) — which is in fact not needed in our distributed computing problem. It would be interesting to explore further improvements in the computational costs, upon the removal of this uniqueness condition. One can imagine that further improvements in the distributed computing problem could also benefit from the deep connections revealed in [105] between compressed sensing and error correction.

- **Approximation Theory:** For future work, we can consider the scenario where the various subfunctions contribute more to the reconstruction of the overall function than other subfunctions do. This can connect the problem to the approximation theory.

- **Predefined Communication Topology or Computation Assignment of Subfunctions** Another interesting direction could involve

a pre-defined and fixed communication topology corresponding to a support of $\mathbf{D}$ that is fixed and independent of $\mathbf{F}$, or similarly a scenario where each server can only compute a predefined subset of subfunctions, itself corresponding to a support of $\mathbf{E}$ that is fixed and independent of $\mathbf{F}$.

- **Optimally of The Lossless Tessellated Distributed Computing:** Another important direction is to prove optimality of the presented tessellated distributed computing scheme concerning all $K, L, N, T, \Delta, \Gamma \in \mathbb{N}$, or to offer another optimal scheme.

- **Analysis of Access Complexity:** Consider the evolution of research surrounding distributed storage systems, in addition to the optimization of repair bandwidth or storage within such systems, there's a growing concern over access complexity, as the time required to upload data can become a bottleneck. The exploration of optimal-access MDS codes began with [142], while a similar concept was introduced in [143] with locally repairable codes to minimize the number of nodes needing access. A trade-off between access complexity and storage amount was suggested for PIR in [144]. In the problem of multi-user linearly separable distributed computing, one might imagine the case where each server, is required to have some chunks of a big dataset being stored on the server. This requires the master node to access and upload a subset of datasets in the assignment phase. For example, if subfunctions 1, 2 and 4 are assigned to some servers, then the master node has to access 3 times server number 1 to upload the datasets while if datasets number 1, 2 and number 3, 4 are always jointly being uploaded by the master node, then the access complexity reduces to 2, since the master node in one access uploads datasets 1, 2 and in another access uploads datasets 3,4. In the problem formulation of our problem in Chapter 2, one might imagine modelling the access complexity via introducing the new concept of generalized covering radius [145] into the problem.

- **Quantum Version of The Multi-User Linearly Separable Distributed Computing:** Recently, the capacity of classical summation over a quantum MAC with arbitrarily distributed inputs and entanglements [146], has been investigated and discovered. The problem investigates a setting where a user is interested in receiving a summation of messages that are distributed arbitrarily across a number of servers. There are some independent entangled quantum systems that are distributed arbitrarily between servers. The communication capacity of the system is derived in [146] using N-sum box abstraction of [147].

One might imagine a similar setting, where each message is an output of some subfunction, where each subfunction's computation requires a substantial computational resource. Investigating the computation and communication trade-off of such a setting is an interesting direction for future investigations.

- **Utility of Dense Codes and Expander Codes:** In Chapter 3, we have established an upper bound on the computational delay Λ and the cumulative computation cost Γ based on the packing radius and packing density of the codes (cf. Theorem 6 and Theorem 7). One future direction is to tighten the proposed upper bounds by employing dense codes. The codes that have good parking radius and high parking density. One of the important instances of these codes is Expander codes introduced in [148] which enjoys having a Low-density parity-check matrix. This would also translate to having less communication cost in our setting when its parity-check matrix is utilized as the decoding matrix $\mathbf{D}$ in the proposed scheme. It would be an interesting direction to investigate the communication and computation cost trade-off when Expander codes are being employed.

- **other:** Additional considerations that involve stragglers, channel unevenness or computational non-heterogeneity of the servers, are all interesting research directions.

Bibliography

[1] X. Sun, Y. He, D. Wu, and J. Z. Huang, "Survey of distributed computing frameworks for supporting big data analysis," *Big Data Mining and Analytics*, vol. 6, no. 2, pp. 154–169, 2023.

[2] V. K. Prasanna, S. Iyengar, P. Spirakis, and M. Welsh, *Distributed Computing in Sensor Systems: First IEEE International Conference, DCOSS 2005, Marina Del Rey, CA, USA, June 30-July 1, 2005, Proceedings*. Springer, 2005, vol. 3560.

[3] J. C. Corbett, J. Dean, M. Epstein, *et al.*, "Spanner: Google's globally distributed database," *ACM Transactions on Computer Systems (TOCS)*, vol. 31, no. 3, pp. 1–22, 2013.

[4] M. McCool, A. D. Robison, and J. Reinders, "Structured parallel programming. patterns for efficient computation.," *Journal of Computer Science & Technology*, vol. 15, no. 1, pp. 43–45, 2015.

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

[6] F. Ahmad, S. T. Chakradhar, A. Raghunathan, and T. Vijaykumar, "Tarazu: Optimizing mapreduce on heterogeneous clusters," *ACM SIGARCH Computer Architecture News*, vol. 40, no. 1, pp. 61–74, 2012.

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

[8] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *ACM Computing Surveys (CSUR)*, vol. 53, no. 2, pp. 1–33, 2020.

[9] D. Vohra, "Practical hadoop ecosystem," *Chapter in Apache Parquet*, 2016.

[10] Z. Zhang, L. Cherkasova, and B. T. Loo, "Performance modeling of mapreduce jobs in heterogeneous cloud environments," in *2013 IEEE Sixth International Conference on Cloud Computing*, IEEE, 2013, pp. 839–846.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[13] R. K. Maity, A. S. Rawa, and A. Mazumdar, "Robust gradient descent via moment encoding and ldpc codes," in *2019 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2019, pp. 2734–2738.

[14] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, 2014.

[15] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "A unified coding framework for distributed computing with straggling servers," in *2016 IEEE Globecom Workshops (GC Wkshps)*, IEEE, 2016, pp. 1–6.

[16] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 109–128, 2018.

[17] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded mapreduce," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, 2015, pp. 964–971.

[18] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2017.

[19] J. S. Ng, W. Y. B. Lim, N. C. Luong, *et al.*, "A comprehensive survey on coded distributed computing: Fundamentals, challenges, and networking applications," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1800–1837, 2021.

[20] S. Li, S. Avestimehr, *et al.*, "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.

[21] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 109–128, 2017.

[22] K. Wan, H. Sun, M. Ji, and G. Caire, "Distributed linearly separable computation," *IEEE Transactions on Information Theory*, vol. 68, no. 2, pp. 1259–1278, 2022.

[23] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *International Conference on Machine Learning*, PMLR, 2017, pp. 3368–3376.

[24] N. Raviv, I. Tamo, R. Tandon, and A. G. Dimakis, "Gradient coding from cyclic mds codes and expander graphs," *IEEE Transactions on Information Theory*, vol. 66, no. 12, pp. 7475–7489, 2020.

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

[26] S. Dutta, V. Cadambe, and P. Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," *Advances In Neural Information Processing Systems*, vol. 29, 2016.

[27] A. Ramamoorthy, L. Tang, and P. O. Vontobel, "Universally decodable matrices for distributed matrix-vector multiplication," in *2019 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2019, pp. 1777–1781.

[28] A. B. Das and A. Ramamoorthy, "Distributed matrix-vector multiplication: A convolutional coding approach," in *2019 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2019, pp. 3022–3026.

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

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

[31] S. Wang, J. Liu, and N. Shroff, "Coded sparse matrix multiplication," in *International Conference on Machine Learning*, PMLR, 2018, pp. 5152–5160.

[32] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[33] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1920–1933, 2020.

[34] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 278–301, 2020.

[35] A. Ramamoorthy, A. B. Das, and L. Tang, "Straggler-resistant distributed matrix computation via coding theory: Removing a bottleneck in large-scale data processing," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 136–145, 2020.

[36] Z. Jia and S. A. Jafar, "Cross subspace alignment codes for coded distributed batch computation," *IEEE Transactions on Information Theory*, vol. 67, no. 5, pp. 2821–2846, 2021.

[37] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security, and privacy," in *The 22nd International Conference on Artificial Intelligence and Statistics*, PMLR, 2019, pp. 1215–1225.

[38] M. Ye and E. Abbe, "Communication-computation efficient gradient coding," in *International Conference on Machine Learning*, PMLR, 2018, pp. 5610–5619.

[39] W. Huang, K. Wan, H. Sun, M. Ji, R. C. Qiu, and G. Caire, "Fundamental limits of distributed linearly separable computation under cyclic assignment," in *2023 IEEE International Symposium on Information Theory (ISIT)*, 2023, pp. 2296–2301.

[40] H. Chen, M. Cheng, Z. Huang, and Y. Wu, "On decentralized linearly separable computation with the minimum computation cost," *arXiv preprint arXiv:2401.16181*, 2024.

[41] K. Wan, H. Sun, M. Ji, and G. Caire, "On secure distributed linearly separable computation," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 3, pp. 912–926, 2022.

[42] K. Wan, H. Sun, M. Ji, and G. Caire, "On the tradeoff between computation and communication costs for distributed linearly separable computation," *IEEE Transactions on Communications*, vol. 69, no. 11, pp. 7390–7405, 2021.

[43] M. Zinkevich, M. Weimer, L. Li, and A. Smola, "Parallelized stochastic gradient descent," in *Advances in Neural Information Processing Systems*, vol. 23, 2010.

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

[45] S. Wang, J. Liu, N. Shroff, and P. Yang, "Fundamental limits of coded linear transform," *arXiv preprint arXiv:1804.09791*, 2018.

[46] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 278–301, 2019.

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

[48] M. Soleymani, H. Mahdavifar, and A. S. Avestimehr, "Privacy-preserving distributed learning in the analog domain," *arXiv preprint arXiv:2007.08803*, 2020.

[49] M. Soleymani, H. Mahdavifar, and A. S. Avestimehr, "Analog lagrange coded computing," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 283–295, 2021.

[50] O. Makkonen and C. Hollanti, "Analog secure distributed matrix multiplication over complex numbers," in *2022 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2022, pp. 1211–1216.

[51] K. Tjell and R. Wisniewski, "Privacy in distributed computations based on real number secret sharing," *arXiv preprint arXiv:2107.00911*, 2021.

[52] M. Soleymani, H. Mahdavifar, and A. S. Avestimehr, "Analog lagrange coded computing," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 283–295, 2021.

[53] Q.-T. Le, E. Riccietti, and R. Gribonval, "Spurious valleys, np-hardness, and tractability of sparse matrix factorization with fixed support," *SIAM Journal on Matrix Analysis and Applications*, vol. 44, no. 2, pp. 503–529, 2023.

[54] F. Ardila and R. P. Stanley, "Tilings," *The Mathematical Intelligencer*, vol. 32, no. 4, pp. 32–43, 2010.

[55] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10)*, 2010.

[56] T. Jahani-Nezhad and M. A. Maddah-Ali, "Codedsketch: A coding scheme for distributed computation of approximated matrix multiplication," *IEEE Transactions on Information Theory*, vol. 67, no. 6, pp. 4185–4196, 2021.

[57] J. Wang, Z. Jia, and S. A. Jafar, "Price of precision in coded distributed matrix multiplication: A dimensional analysis," in *2021 IEEE Information Theory Workshop (ITW)*, IEEE, 2021, pp. 1–6.

[58] E. Ozfatura, S. Ulukus, and D. Gündüz, "Coded distributed computing with partial recovery," *IEEE Transactions on Information Theory*, vol. 68, no. 3, pp. 1945–1959, 2022.

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

[60] F. Haddadpour, M. M. Kamani, M. Mahdavi, and V. Cadambe, "Trading redundancy for communication: Speeding up distributed sgd for non-convex optimization," in *International Conference on Machine Learning*, PMLR, 2019, pp. 2545–2554.

[61] C.-S. Yang, R. Pedarsani, and A. S. Avestimehr, "Coded computing in unknown environment via online learning," in *2020 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2020, pp. 185–190.

[62] N. Charalambides, H. Mahdavifar, and A. O. Hero, "Numerically stable binary gradient coding," in *2020 IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 2622–2627.

[63] H. Sun and S. A. Jafar, "The capacity of private computation," *IEEE Transactions on Information Theory*, vol. 65, no. 6, pp. 3880–3897, 2018.

[64] M. Soleymani and H. Mahdavifar, "Distributed multi-user secret sharing," *IEEE Transactions on Information Theory*, vol. 67, no. 1, pp. 164–178, 2020.

[65] A. Khalesi, M. Mirmohseni, and M. A. Maddah-Ali, "The capacity region of distributed multi-user secret sharing," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 3, pp. 1057–1071, 2021.

[66] M. Soleymani, R. E. Ali, H. Mahdavifar, and A. S. Avestimehr, "List-decodable coded computing: Breaking the adversarial toleration barrier," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 3, pp. 867–878, 2021.

[67] C. Hofmeister, R. Bitar, M. Xhemrishi, and A. Wachter-Zeh, "Secure private and adaptive matrix multiplication beyond the singleton bound," *IEEE Journal on Selected Areas in Information Theory*, vol. 3, no. 2, pp. 275–285, 2022.

[68] C. Hofmeister, R. Bitar, M. Xhemrishi, and A. Wachter-Zeh, "Secure private and adaptive matrix multiplication beyond the singleton bound," *IEEE Journal on Selected Areas in Information Theory*, vol. 3, no. 2, pp. 275–285, 2022.

[69] H. Akbari-Nodehi and M. A. Maddah-Ali, "Secure coded multi-party computation for massive matrix operations," *IEEE Transactions on Information Theory*, vol. 67, no. 4, pp. 2379–2398, 2021.

[70] Z. Jia and S. A. Jafar, "On the capacity of secure distributed batch matrix multiplication," *IEEE Transactions on Information Theory*, vol. 67, no. 11, pp. 7420–7437, 2021.

[71] Z. Chen, Z. Jia, Z. Wang, and S. A. Jafar, "Gcsa codes with noise alignment for secure coded multi-party batch matrix multiplication," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 306–316, 2021.

[72] C.-S. Yang and A. S. Avestimehr, "Coded computing for secure boolean computations," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 326–337, 2021.

[73] Q. Yu and A. S. Avestimehr, "Coded computing for resilient, secure, and privacy-preserving distributed matrix multiplication," *IEEE Transactions on Communications*, vol. 69, no. 1, pp. 59–72, 2020.

[74] M. Xhemrishi, R. Bitar, and A. Wachter-Zeh, "Distributed matrix-vector multiplication with sparsity and privacy guarantees," in *2022 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2022, pp. 1028–1033.

[75] M. Egger, R. Bitar, A. Wachter-Zeh, and D. Gündüz, "Efficient distributed machine learning via combinatorial multi-armed bandits," in *2022 IEEE International Symposium on Information Theory (ISIT)*, 2022, pp. 1653–1658.

[76] A. Behrouzi-Far and E. Soljanin, "Efficient replication for straggler mitigation in distributed computing," *arXiv preprint arXiv:2006.02318*, 2020.

[77] A. C.-C. Yao, "Communication complexity and its applications," in *International Workshop on Frontiers in Algorithmics*, Springer, 2009, pp. 2–2.

[78] S. Ulukus, S. Avestimehr, M. Gastpar, S. Jafar, R. Tandon, and C. Tian, "Private retrieval, computing and learning: Recent progress and future challenges," *IEEE Journal on Selected Areas in Communications*, 2022.

[79] K. Wan, H. Sun, M. Ji, D. Tuninetti, and G. Caire, "Cache-aided matrix multiplication retrieval," *IEEE Transactions on Information Theory*, vol. 68, no. 7, pp. 4301–4319, 2022.

[80] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Codedreduce: A fast and robust framework for gradient aggregation in distributed learning," *IEEE/ACM Transactions on Networking*, 2021.

[81] N. Woolsey, R.-R. Chen, and M. Ji, "A new combinatorial coded design for heterogeneous distributed computing," *IEEE Transactions on Communications*, vol. 69, no. 9, pp. 5672–5685, 2021.

[82] N. Woolsey, R.-R. Chen, and M. Ji, "Coded elastic computing on machines with heterogeneous storage and computation speed," *IEEE Transactions on Communications*, vol. 69, no. 5, pp. 2894–2908, 2021.

[83] N. Woolsey, J. Kliewer, R.-R. Chen, and M. Ji, "A practical algorithm design and evaluation for heterogeneous elastic computing with stragglers," in *2021 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2021, pp. 1–6.

[84] M. Chen, D. Gündüz, K. Huang, *et al.*, "Distributed learning in wireless networks: Recent progress and future challenges," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3579–3605, 2021.

[85] V. M. Blinovskii, "Lower asymptotic bound on the number of linear code words in a sphere of given radius in (f_q)ˆn," *Problemy Peredachi Informatsii*, vol. 23, no. 2, pp. 50–53, 1987.

[86] G. Cohen, I. Honkala, S. Litsyn, and A. Lobstein, *Covering codes*. Elsevier, 1997.

[87] R. M. Roth, "Introduction to coding theory," *IET Communications*, vol. 47, 2006.

[88] G. Cohen and P. Frankl, "Good coverings of hamming spaces with spheres," *Discrete Mathematics*, vol. 56, no. 2-3, pp. 125–131, 1985.

[89] K. Wan, H. Sun, M. Ji, and G. Caire, "On secure distributed linearly separable computation," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 3, pp. 912–926, 2022.

[90] A. K. Das and S. Vishwanath, "On finite alphabet compressive sensing," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2013, pp. 5890–5894.

[91] A. G. Dimakis, R. Smarandache, and P. O. Vontobel, "Ldpc codes for compressed sensing," *IEEE Transactions on Information Theory*, vol. 58, no. 5, pp. 3093–3114, 2012.

[92] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Tree gradient coding," in *2019 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2019, pp. 2808–2812.

[93] E. Candes and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[94] G. Cohen, "A nonconstructive upper bound on covering radius," *IEEE Transactions on Information Theory*, vol. 29, no. 3, pp. 352–353, 1983.

[95] A. N. Kolmogoroff, *Foundations of the Theory of Probability*. Chelsea Publishing Company, 1956.

[96] H. E. Danoyan, "On some properties of intersection and union of spheres in hamming metric," *Mathematical Problems of Computer Science*, vol. 39, pp. 119–124, 2013.

[97] T. Etzion, *Perfect Codes and Related Structures*. WORLD SCIENTIFIC, 2022. eprint: `https://www.worldscientific.com/doi/pdf/10.1142/12823`.

[98] A. Tietäväinen, "On the nonexistence of perfect codes over finite fields," *SIAM Journal on Applied Mathematics*, vol. 24, no. 1, pp. 88–96, 1973.

[99] A. Mallick, S. Smith, and G. Joshi, "Rateless codes for distributed non-linear computations," in *2021 11th International Symposium on Topics in Coding (ISTC)*, IEEE, 2021, pp. 1–5.

[100] A. Mallick and G. Joshi, "Rateless sum-recovery codes for distributed non-linear computations," in *2022 IEEE Information Theory Workshop (ITW)*, IEEE, 2022, pp. 160–165.

[101] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*. 2013.

[102] E. J. Candes and Y. Plan, "A probabilistic and ripless theory of compressed sensing," *IEEE Transactions on Information Theory*, vol. 57, no. 11, pp. 7235–7254, 2011.

[103] E. J. Candes, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 59, no. 8, pp. 1207–1223, 2006.

[104] E. J. Candes, "The restricted isometry property and its implications for compressed sensing," *Comptes rendus mathematique*, vol. 346, no. 9-10, pp. 589–592, 2008.

[105] E. J. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?" *IEEE transactions on information theory*, vol. 52, no. 12, pp. 5406–5425, 2006.

[106] T. T. Cai, L. Wang, and G. Xu, "New bounds for restricted isometry constants," *IEEE Transactions on Information Theory*, vol. 56, no. 9, pp. 4388–4394, 2010.

[107] T. T. Cai, L. Wang, and G. Xu, "Shifting inequality and recovery of sparse signals," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1300–1308, 2010.

[108] S. Daei, F. Haddadi, and A. Amini, "Living near the edge: A lower-bound on the phase transition of total variation minimization," *IEEE Transactions on Information Theory*, vol. 66, no. 5, pp. 3261–3267, 2019.

[109] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM review*, vol. 43, no. 1, pp. 129–159, 2001.

[110] S. Jokar and V. Mehrmann, "Sparse solutions to underdetermined kronecker product systems," *Linear Algebra and its Applications*, vol. 431, no. 12, pp. 2437–2447, 2009.

[111] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[112] D. Malak and M. Médard, "A distributed computationally aware quantizer design via hyper binning," *IEEE Transactions on Signal Processing*, vol. 71, pp. 76–91, 2023.

[113] D. P. Woodruff *et al.*, "Sketching as a tool for numerical linear algebra," *Foundations and Trends® in Theoretical Computer Science*, vol. 10, no. 1–2, pp. 1–157, 2014.

[114] W.-T. Chang and R. Tandon, "Random sampling for distributed coded matrix multiplication," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 8187–8191.

[115] N. Charalambides, M. Pilanci, and A. O. Hero, "Approximate weighted c r coded matrix multiplication," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 5095–5099.

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

[117] N. S. Ferdinand and S. C. Draper, "Anytime coding for distributed computation," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2016, pp. 954–960.

[118] J. Zhu, Y. Pu, V. Gupta, C. Tomlin, and K. Ramchandran, "A sequential approximation framework for coded distributed optimization," in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2017, pp. 1240–1247.

[119] T. Jahani-Nezhad and M. A. Maddah-Ali, "Berrut approximated coded computing: Straggler resistance beyond polynomial computing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 111–122, 2023.

[120] H. Jeong, A. Devulapalli, V. R. Cadambe, and F. P. Calmon, "ϵ-approximate coded matrix multiplication is nearly twice as efficient as exact multiplication," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 3, pp. 845–854, 2021.

[121] S. Kiani and S. C. Draper, "Successive approximation coding for distributed matrix multiplication," *IEEE Journal on Selected Areas in Information Theory*, vol. 3, no. 2, pp. 286–305, 2022.

[122] R. Ji, A. Heidarzadeh, and K. R. Narayanan, "Sparse random khatri-rao product codes for distributed matrix multiplication," in *2022 IEEE Information Theory Workshop (ITW)*, 2022, pp. 416–421.

[123] M. Rudow, N. Charalambides, A. O. Hero, and K. Rashmi, "Compression-informed coded computing," in *2023 IEEE International Symposium on Information Theory (ISIT)*, 2023, pp. 2177–2182.

[124] N. Agrawal, Y. Qiu, M. Frey, *et al.*, "A learning-based approach to approximate coded computation," in *2022 IEEE Information Theory Workshop (ITW)*, 2022, pp. 600–605.

[125] T. Jahani-Nezhad and M. A. Maddah-Ali, "Optimal communication-computation trade-off in heterogeneous gradient coding," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 3, pp. 1002–1011, 2021.

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

[127] A. Khalesi and P. Elia, "Multi-user linearly-separable distributed computing," *IEEE Transactions on Information Theory*, vol. 69, no. 10, pp. 6314–6339, 2023.

[128] A. Khalesi and P. Elia, "Multi-user linearly separable computation: A coding theoretic approach," in *2022 IEEE Information Theory Workshop (ITW)*, 2022, pp. 428–433.

[129] A. Khalesi, S. Daei, M. Kountouris, and P. Elia, "Multi-user distributed computing via compressed sensing," in *2023 IEEE Information Theory Workshop (ITW)*, 2023, pp. 509–514.

[130] R. Gribonval and K. Schnass, "Dictionary identification—sparse matrix-factorization via $l-1$-minimization," *IEEE Transactions on Information Theory*, vol. 56, no. 7, pp. 3523–3539, 2010.

[131] L. Zheng, E. Riccietti, and R. Gribonval, "Identifiability in two-layer sparse matrix factorization," *arXiv preprint arXiv:2110.01235*, 2021.

[132] L. Zheng, E. Riccietti, and R. Gribonval, "Efficient identification of butterfly sparse matrix factorizations," *SIAM Journal on Mathematics of Data Science*, vol. 5, no. 1, pp. 22–49, 2023.

[133] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.

[134] C. F. Van Loan and G. Golub, "Matrix computations (johns hopkins studies in mathematical sciences)," *Matrix Computations*, vol. 5, 1996.

[135] T. Dao, A. Gu, M. Eichhorn, A. Rudra, and C. Ré, "Learning fast algorithms for linear transforms using butterfly factorizations," in *International conference on machine learning*, PMLR, 2019, pp. 1517–1527.

[136] W. Hackbusch, "A sparse matrix arithmetic based on-matrices. part i: Introduction to matrices," *Computing*, vol. 62, no. 2, pp. 89–108, 1999.

[137] C. R. Johnson, "Matrix completion problems: A survey," in *Matrix theory and applications*, vol. 40, 1990, pp. 171–198.

[138] T. Tao, *Topics in random matrix theory*. American Mathematical Soc., 2012, vol. 132.

[139] N. Kishore Kumar and J. Schneider, "Literature survey on low rank approximation of matrices," *Linear and Multilinear Algebra*, vol. 65, no. 11, pp. 2212–2244, 2017.

[140] V. A. Marchenko and L. A. Pastur, "Distribution of eigenvalues for some sets of random matrices," *Matematicheskii Sbornik*, vol. 114, no. 4, pp. 507–536, 1967.

[141] A. Khalesi and P. Elia, "Tessellated distributed computing," *arXiv preprint arXiv:2404.14203*, 2024.

[142] I. Tamo, Z. Wang, and J. Bruck, "Access versus bandwidth in codes for storage," *IEEE Transactions on Information Theory*, vol. 60, no. 4, pp. 2028–2037, 2014.

[143] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *IEEE Transactions on Information Theory*, vol. 58, no. 11, pp. 6925–6934, 2012.

[144] Y. Zhang, E. Yaakobi, T. Etzion, and M. Schwartz, "On the access complexity of pir schemes," in *2019 IEEE International Symposium on Information Theory (ISIT)*, 2019, pp. 2134–2138.

[145] D. Elimelech, M. Firer, and M. Schwartz, "The generalized covering radii of linear codes," *IEEE Transactions on Information Theory*, vol. 67, no. 12, pp. 8070–8085, 2021.

[146] Y. Yao and S. A. Jafar, "The capacity of classical summation over a quantum mac with arbitrarily distributed inputs and entanglements," *arXiv preprint arXiv:2305.03122*, 2023.

[147] M. Allaix, Y. Lu, Y. Yao, T. Pllaha, C. Hollanti, and S. Jafar, "N-sum box: An abstraction for linear computation over many-to-one quantum networks," in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, 2023, pp. 5457–5462.

[148] M. Sipser and D. Spielman, "Expander codes," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1710–1722, 1996.