Sparse-Graph Codes for the Big Data Era: Exploiting Sparsity to Speed Up Computation

by

Orhan Ocal

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Kannan Ramchandran, Chair
Professor Laurent El Ghaoui
Professor Bruno Olshausen

Spring 2020

Sparse-Graph Codes for the Big Data Era: Exploiting Sparsity to Speed Up Computation

Abstract

Sparse-Graph Codes for the Big Data Era: Exploiting Sparsity to Speed Up Computation

by

Orhan Ocal

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Kannan Ramchandran, Chair

We are witnessing an unprecedented growth in the amount of data being sensed, collected and processed to drive modern applications across many domains. This growth has challenged classical algorithms to scale up to big-data regimes. Because time is critically important in settings such as control and decision-making, interventional medical imaging, and security, speeding up widely-used routines has the potential to open up the application space.

In this thesis, we focus on six problems of fundamental importance, and develop fast algorithms for solving them whenever there is an underlying structure of *sparsity*, which typically holds in most natural settings. Our algorithms attain their speed by being sample and computationally efficient whenever there is sparsity. In particular, the problems we study and a summary of our results are as follows. (1) We propose a novel architecture for sampling continuous-time signals that have band-limited and sparse spectra at their minimum sampling rate, which can be much lower than the Nyquist rate. (2) We improve the noise robustness of divide-and-conquer-based sparse Discrete Fourier Transform algorithms by developing a novel method for fast and sample-efficient frequency identification. (3) We develop a fast algorithm for recovering sparse wavelet coefficients of a signal from its Fourier domain samples. This setting arises in Magnetic Resonance Imaging, and our approach provides a theoretical framework for fast imaging. (4) We propose a sample and computationally efficient algorithm for recovering sparse pseudo-Boolean functions with application to learning a DNA repair outcomes for CRISPR-Cas9 experiments. (5) We propose a computationally efficient method for coded computation to reduce the effect of slow computational nodes in distributed matrix multiplication. (6) We propose a computationally efficient algorithm for large-scale matrix multiplication whenever the output is sparse with application to database search.

Our approach in tackling these diverse problems is based on a unifying divide-and-conquer strategy, where by leveraging ideas from *modern coding theory* (in particular, Low Density Parity Check Codes) which have previously been studied in the context of communications, we create structures that enable simple and fast *peeling*-based recovery.

To my parents.

Contents

List of Figures

List of Tables

Acknowledgments

This thesis would not have been possible without the help of many people over the years, and I would like to acknowledge those whose support was integral for me in my journey. This list is nowhere comprehensive, and there are many whose presence I appreciated greatly but did not get a chance to thank here.

First and foremost, I would like to express my heartfelt gratitude to my advisor Kannan Ramchandran. He was a great source of support, inspiration, and motivation for me throughout my journey. He was always available for having stimulating discussions on research, brainstorming about new directions to pursue, open to exploring any area that I was attracted to, and he helped me keep the big picture in mind in every step I took. I also benefited a lot from the principles he taught me to follow to present my work and ideas in an accessible yet precise way in speech, writing or while doing a presentation. His enthusiasm for research and devotion to his work are forever going to be an inspiration for me in my career and personal life.

I would also like to acknowledge the contributions of amazing faculty who inspired me in my studies over the years. I learned a lot from Anant Sahai during my semester of being a teaching assistant for EE16A where he never compromised from targeting for perfection in teaching. I would also like to thank Laurent El Ghaoui, Bruno Olshausen, and Borivoje Nikolic for being on my thesis committee and qualification exam, and giving me valuable feedback and support. I was very lucky to take classes and learn from many amazing faculty – Laurent El Ghaoui, Thomas Courtade, Venkat Anantharam, Michael Mahoney, Laura Waller, Ben Recht, John Canny, Bin Yu, and Tsu-Jae King Liu – which provided me with the confidence and skills to approach new problems. I would also like to thank my masters advisor Martin Vetterli at EPFL, who believed in me and gave me the opportunity to work in his group, and fortified my interest in academic research by being an amazing advisor and person alike. I would also like to thank Emre Telatar and Ivan Dokmanic for their invaluable guidance during my masters studies and beyond.

Over my graduate studies, I had the privilege to work with extraordinarily talented people on research projects, and this thesis would not have been possible without their integral contributions. I would like to thank my collaborators Kabir Chandrasekher, Tavor Baharav, Vipul Gupta, Raaz Dwivedi, Kamil Nar, Frank Ong, Amirali Aghazadeh, Xiao Li, Kangwook Lee, Angie Wang and Swanand Kadhe. It was a pleasure to work with them on research projects, and I have learned a lot from them through our discussions. I would also like to thank my mentors and collaborators Oguz Elibol, Gokce Keskin, Cory Stephenson, and Anil Thomas from my summer internship at Intel AI Lab for their support and guidance.

PhD, by its nature, requires focusing on specific problems by yourself, and that can make one feel lonely at times. However, I was so lucky to have amazing friends who were always there to be a part of this journey and find support together. I would like to thank Fanny Yang for the great conversations, outdoor trips, and the amazing music that she shared with us through her violin, Raaz Dwivedi for sharing his statistics knowledge with me and being the best teammate one can ever have: whenever we were in the same class, I knew we would form another team RaHan for the project, and Yuting Wei for lunch, coffee and concert breaks, and her aesthetic additions to our lab space. The members of BLISS/Wi-Fo Lab., Vipul Gupta, Soham Phade, Dong Yin, Vasuki Swamy, Vidya Muthukumar, Ashwin Pananjady, Payam Delgosha, Sang Min Han, Avishek Ghosh, Jichan Chung, Kuan-Yun Lee, Efe Aras, Banghua Zhu, Nihar Shah, Yaoqing Yang, Ramtin Pedarsani, Dimitris Papailiopoulos, Rashmi Vinayak, Gireeja Ranade, Vijay Kamble, Giulia Fanti, Varun Jog, Laura Brink, Ilan Shomorony, and Reinhard Heckel to name a few, were great sources of support, knowledge, and friendship. I would also like to thank my friends Kamil Nar, Tugce Gurek, Erden Yildizdag, Sophie Pham and Vivian Zhang for being pillars of support in school and life in general, and Eren Sasoglu, who has been an academic and career mentor, and an invaluable friend since my masters studies. I would also like to thank my friends from home, Efe Yigitbasi, Murat Gursu, Erdem Altay, Tanju Yasar, and Bora Durmaz for always being there for sharing the happy moments and providing selfless help and support whenever I needed.

I would also like to have a special shout out to the friendly and extremely helpful staff of EECS at Berkeley. In particular, Shirley Salanio has made my (and everybody's) life at EECS better; whenever I needed help, I knew I could get timely support Shirley that would solve any bureaucratic issue I had, and also hear her nice words, which would brighten any day. I would also like to thank Ming Wong from the Engineering Support Group; I have met him when I was an assistant for EE16A, and since then, whenever I needed help with technical issues he went out of his way to help.

One of the opportunities I am extremely grateful about was being able to take classes from the physical education department at Berkeley. I would like to express my special thanks to Toni Mar and Lucy Yoo for their valuable teachings on how to keep a healthy life lifestyle, and providing us the opportunities to keep our body and mind healthy.

Last but not least, I would like to thank my parents Ruhsar Ocal and Erol Ocal, to whom this thesis is dedicated to. They have always made me feel loved and supported, and placed my well-being above anything else. With their contagious love and respect for education and science, they provided me the integral support and motivation for me to pursue my dreams. I owe all my achievements to them and their unconditional love and support, and I will forever be grateful to them.

Chapter 1

Introduction

We are witnessing an unprecedented growth in the amount of data being sensed, collected, and processed to drive modern applications across a multitude of domains. This growth has been, in part, fueled by recent advancements in machine learning and decision-making, where the approach has shifted to using complex models which require large amounts of data for training. This has challenged classical algorithms to scale up to big-data regimes, rendering them too resource-hungry and slow. Because time is critically important in settings such as control and decision-making, interventional medical imaging, and security, speeding up widely-used routines has the potential to open up the application space. In this thesis, we focus on six problems of interest, and develop fast algorithms for solving them whenever there is sparsity, which appears naturally in a wide range of real-world problems.

First, sparsity can be in the data representation. For example, in Magnetic Resonance Imaging (MRI), the images have a sparse representation in the wavelet domain. This structure of sparsity has already been exploited to compress data after acquisition, as is done in JPEG to compress images. However, this pipeline of first acquiring the data and then compressing it can be too costly. To combat high acquisition costs, a new paradigm for data acquisition called *compressed sensing* was developed by researchers. In this paradigm, data is subsampled at the acquisition time to compress the data at its source. Then, an appropriate recovery algorithm is used to reconstruct the object of interest based on the acquired samples. This paradigm enabled faster acquisition, and led to important advancements in real-world problems, such as reducing the time a patient spends in the MRI machine. The now well-known recovery algorithms in compressed sensing literature work well when the ambient dimension is relatively small and the bottleneck is the data acquisition. However, for large ambient dimensions, these recovery algorithms are too costly because their computational complexities are at least linear in the ambient dimension even though the information might reside in a significantly smaller subspace. This dependence of computational complexity in the ambient dimension renders the well-known recovery algorithms in compressed sensing literature (such as LASSO [86] or OMP [87]) impractical. In this thesis, we propose frameworks that are fast at *both* acquisition and recovery. In particular, the developed algorithms have sample complexity and computational complexity that scale *sublinearly* in the ambient dimension whenever the degrees of freedom (sparsity level) also scale sublinearly.

Another setting we study is where there are sparse nonidealities in a system. Concretely, we will consider distributed computing where a large computation is divided into smaller chunks and the workload is distributed across multiple workers to speed up computation. In this modality there can be slow worker nodes, called stragglers, that affect the overall runtime of the computation. Sparsity in this setting means that the number of straggler nodes is small compared to the total number of workers. In this setting, the user can introduce controlled redundancy to combat against these straggling nodes. For this setting, we develop a fast algorithm to combat against stragglers.

Our approach in tackling these seemingly diverse problems is based on a unifying divide-and-conquer strategy, where by leveraging ideas from coding theory (Low Density Parity Check Codes and Product Codes) which have been previously studied in the context of communications, we create structures that enable simple and fast *peeling*-based recovery. As shown in Figure 1.1, sparse-graph codes (to be described in Section 1.1) sit at the center like an engine that "powers" the structure in tackling our diverse problems of interest. The problems we consider and our contributions therein are as follows.

In Chapter 2, we study the problem of *spectrum-blind* sampling, that is, sampling signals with sparse spectrum whose frequency support is unknown. The minimum sampling rate for this class of signals has been established as twice the measure of its frequency support; however, constructive sampling schemes that achieve this minimum rate are not known to exist. We propose a novel sampling framework by leveraging a mix of tools from *modern coding theory*, which has been largely untapped in the field of sampling. We make interesting connections between the fundamental problem of spectrum-blind sampling, and that of designing erasure-correcting codes based on sparse-graph codes, which have both theoretically and practically revolutionized the design of modern communication systems. The key idea is to cleverly exploit, rather than avoid, the resulting aliasing artifact induced by subsampling, which introduces linear mixing of spectral components in the form of parity constraints for *sparse-graph codes*. We show that the signal reconstruction under this sampling scheme is equivalent to the peeling decoding of *sparse-graph codes* for reliable transmission on an erasure channel. We further show that the achievable *sampling rate* is determined by the *rate* of the sparse-graph code used in the filter bank. As a result, based on insights derived from the design of capacity-achieving sparse-graph codes, we *simultaneously* approach the minimum sampling rate for spectrum-blind sampling and low computational complexity based on fast peeling-based decoding with operations per unit of time scaling *linearly* with the sampling rate.

In Chapter 3, we focus on the Discrete Fourier Transform (DFT) and take steps towards improving the practical impact of a class of divide-and-conquer-based fast sparse DFT algorithms. The class of algorithms we consider computes a K-sparse DFT by dividing the problem into recovering K-many *single tones* and then combining their results. We improve the noise robustness of such algorithms by developing a novel sample and computationally efficient method for single tone identification from noisy measurements. This is a step towards making the algorithm more impactful in low SNR applications such as medical imaging and radio astronomy. Besides having strong theoretical guarantees, these algorithms also have strong practical performance, and our collaboration resulted in a hardware implementation.

In Chapter 4, we introduce a new ensemble of random bipartite graphs, which we term the

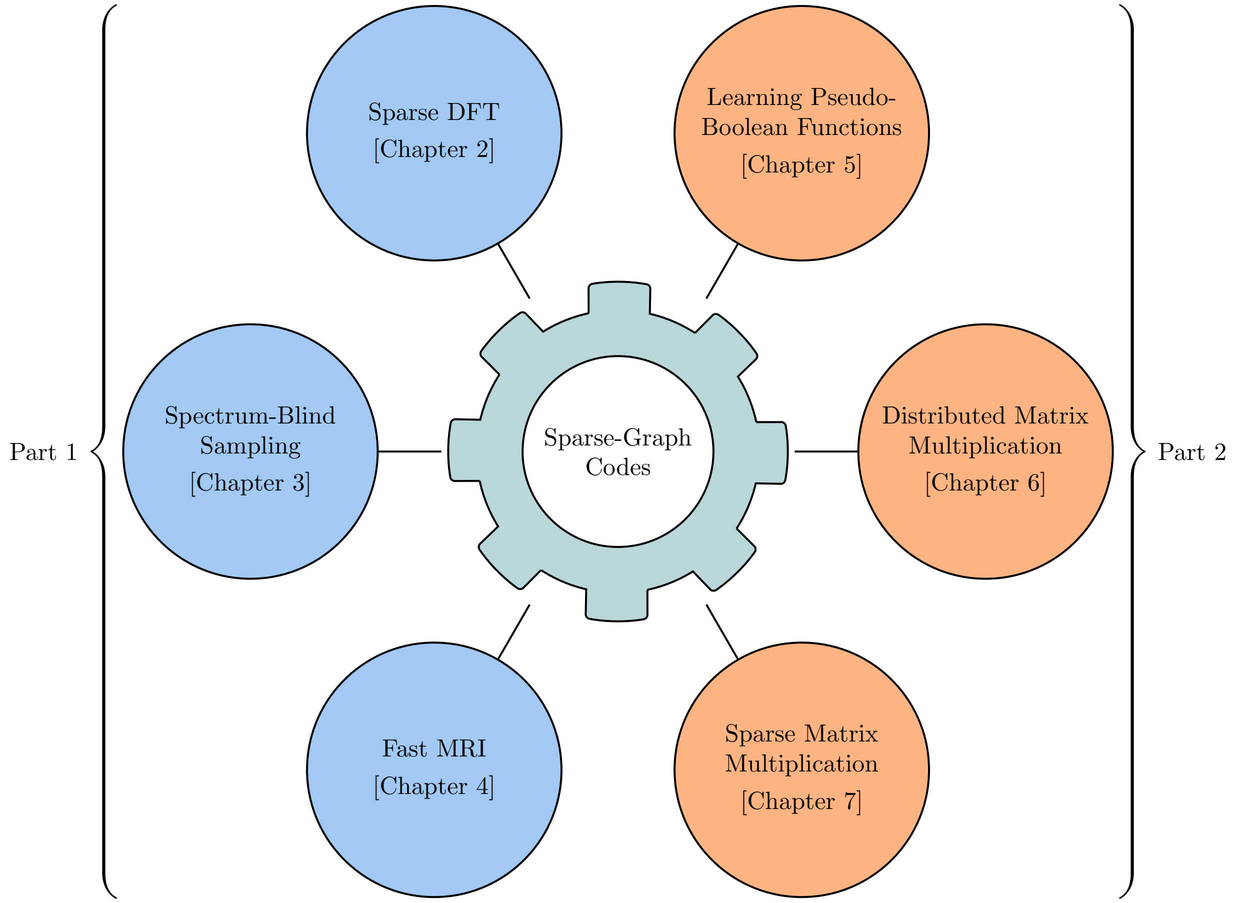


Figure 1.1: In this dissertation, we focus on six problems. Underlying our solutions to these problems is the use of sparse-graph codes as a divide-and-conquer engine to devise fast peeling-based algorithms.

'smearing ensemble', where each left node is connected to some number of consecutive right nodes. Such graphs arise naturally in the recovery of sparse wavelet coefficients when signal acquisition is in the Fourier domain, such as in MRI. Graphs from this ensemble exhibit small, structured cycles with high probability, rendering current techniques for determining iterative decoding thresholds inapplicable. We develop a theoretical platform to analyze and evaluate the effects of smearing-based structure. Despite the existence of these small cycles, we derive exact density evolution recurrences for iterative decoding on graphs with smear-length two. Further, we give lower bounds on the performance of a much larger class from the smearing ensemble, and provide numerical experiments showing tight agreement between empirical thresholds and those determined by our bounds. Finally, we describe a system architecture to recover sparse wavelet representations in the MRI setting that is sample and computationally efficient.

In Chapter 5, we present a sample and computationally efficient method for recovering low-degree pseudo-Boolean functions. Pseudo-Boolean functions are functions whose input variables are binary and the output is real-valued. Such functions show up in many applications in computational biology and computer science to name a few. Pseudo-Boolean functions lend themselves to a spectral representation, which is closely related to the Walsh-Hadamard Transform (WHT) from signal processing. In many problems of interest, the coefficients of the spectral representation are active only on low-degree terms, which means that the cardinality of the interactions between the features is small. Our method is based on evaluating the input pseudo-Boolean function at points given by the codewords of a codebook, and then performing a WHT on the resulting signal. Codes having high rates and good minimum distance properties yield sets of evaluations points whose size is close to the number of low-degree coefficients. In particular perfect codes, such as Hamming Codes or the Golay Code, enable efficient recovery with a minimal number of evaluations of the function. We use our method for learning the landscape of DNA repair outcomes after CRISPR-Cas9 double-strand breaks. Recent studies have shown that after a double-strand break, the DNA repair outcomes are consistent across experimental replicates, and that they are a function of the sequence around the cut site. Learning a model for repair outcomes is of interest as it can be used for designing gene knock out experiments. We represent the DNA sequence in a window around the cut site as a binary sequence, and learn the function mapping the sequence to the repair outcomes. As nucleotides in DNA can take on 4 different bases (A,T,C,G), a window length of n nucleotides has 4^n possible base sequences, which poses a high dimensional problem. However, we observe that WHT-sparse models have significant predictive power, and we use our method for learning it. Furthermore, the WHT coefficients provides interpretable insights on the DNA repair process.

In Chapter 6, we present a novel coded matrix-matrix multiplication scheme to combat stragglers, that is, slow worker nodes, in distributed computing. Distributed computing allows for large-scale computation and machine learning tasks by enabling parallel computing at massive scale. A critical challenge to speeding up distributed computing comes from *stragglers*, a crippling bottleneck to system performance. Recently, coding theory has offered an attractive paradigm dubbed as coded computation for addressing this challenge through the judicious introduction of redundant computing to combat stragglers. However, most existing approaches have limited applicability if the system scales to hundreds or thousands of workers, as is the trend in computing platforms. At these scales, previously proposed algorithms are too expensive due to their hidden cost, such as computing and communication costs associated with the encoding/decoding procedures. On the other hand, our novel coded matrix-matrix multiplication scheme based on product codes allows for order-optimal computation/communication costs for the encoding/decoding procedures while achieving near-optimal compute time.

In Chapter 7, we design a framework for speeding up large-scale matrix multiplication when the output is sparse. Leveraging the sparsity of the output matrix in a matrix multiplication operation has applications ranging from the recovery of sparse covariance matrices to fast string search across a database (where the rows of the first matrix represent the database of strings and the columns of the second matrix matrix represent the query strings to be searched in the database) to sketching sparse matrices. Our solution is based on a novel use of product codes. When multiplying two matrices of sizes $n \times d$ and $d \times n$ where the output matrix is (exactly) K-sparse with support uniformly distributed, our algorithm requires $\max(O(dK), O(dn))$ computations to perform the matrix-matrix multiplication with high probability.

1.1 Sparse-Graph Codes

We start with a brief introduction to sparse-graph codes, which have been revolutionary in the communications literature for doing error correction. Assume, as the sender, we have a sequence of n symbols $x_1, \dots, x_n$ that we want to send over an *erasure channel* which erases each symbol with a constant probability p_e. Hence, at the receiver side, some symbols will have possibly been erased (on average np_e symbols are erased). If we want the receiver to be able to recover all the symbols with high probability, then we need to introduce some redundancy to the message we send so that the erasures can be recovered at the receiver. The level of redundancy needed for reliable communication is characterized by information theory founded by the seminal work of Claude E. Shannon [79], and for the erasure channel, we need a redundancy rate of p_e. Coding theory is the study of how to introduce controlled redundancy so that reliable communication is possible over noisy channels. Sparse-graph codes provide a way to introduce this redundancy. They are fast for encoding and decoding the message, and simultaneously enable the amount of introduced redundancy to be arbitrarily close to the optimal.

The most relevant way of introducing sparse-graph codes for our work is through their graphical representation. Sparse-graph codes represent message symbols as *variable nodes* and some linear constraints on these variable nodes as *check nodes*. Traditionally, the variable nodes are represented as circles on the left-hand side of a bipartite graph, and the check nodes are represented as squares on the right-hand side of that graph. If a variable node appears in the linear constraint related to a check node, then that variable node and the check node are connected in the graph. These graphs are called Tanner Graphs, and an example is provided in Figure 1.2. The "sparsity" in sparse-graph codes denotes that the number of edges connecting the variable nodes to the check nodes is linear in the number of nodes.

The property that we are going to use extensively in our works is that erasures can be decoded on a sparse-graph code in linear time through *peeling*. The idea behind peeling is similar to the back-substitution method for solving linear systems involving triangular matrices. In that method, we recover the value of an unknown by reading off its value from an equation that involves that single coefficient, and then peel off its contributions by substituting it back into the equations it shows up.

We explain the peeling process in sparse-graph codes with reference to Figure 1.3 where an example is provided. First, variable nodes corresponding to non-erased symbols are peeled off the graph by subtracting their contributions from the check nodes. We call this resulting graph the *pruned* graph. We categorize the check nodes into three classes:

- **zeroton** check nodes are connected to no unresolved variable nodes,
- **singleton** check nodes are connected to a *single* unresolved variable node,
- **multiton** check nodes are connected to *multiple* unresolved variable nodes.

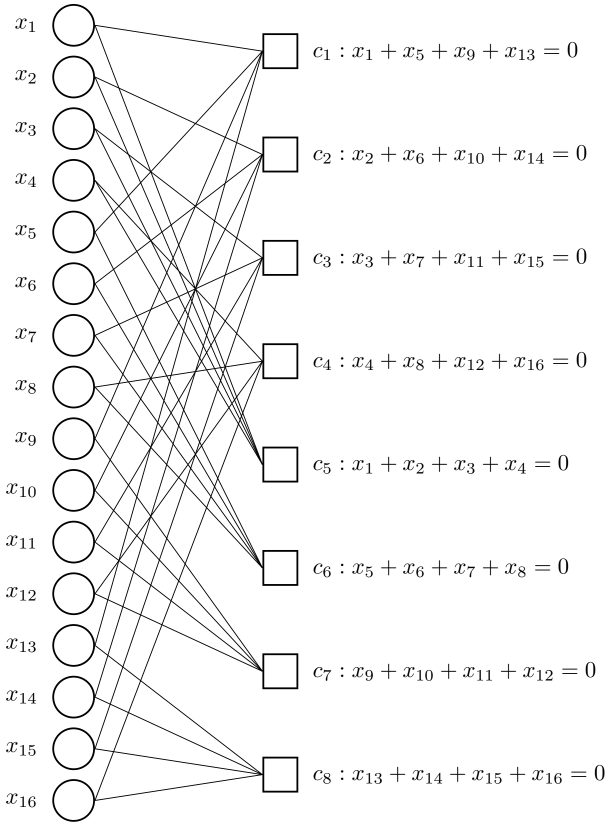
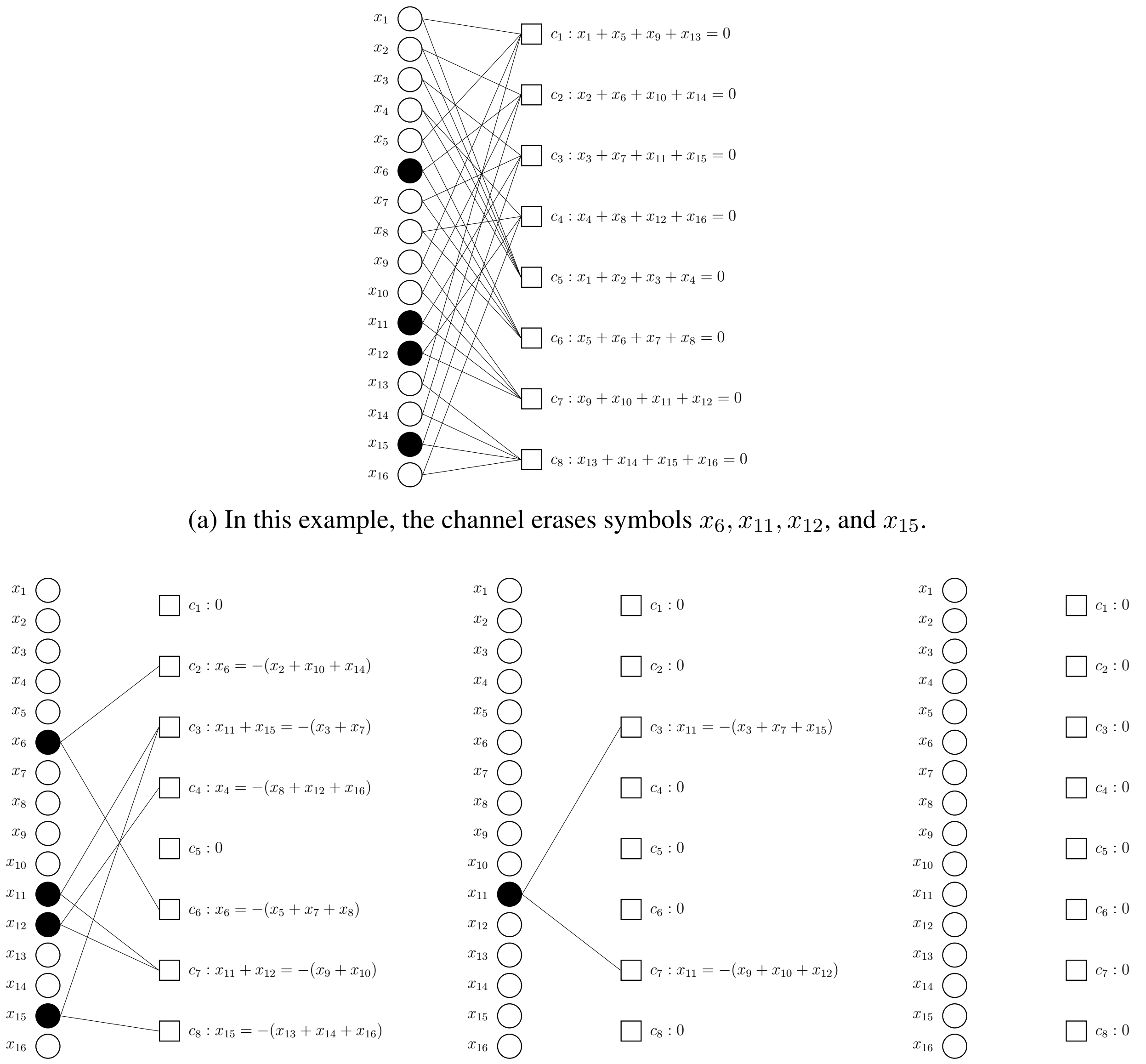

x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
x_9
x_{10}
x_{11}
x_{12}
x_{13}
x_{14}
x_{15}
x_{16}
$c_1 : x_1 + x_5 + x_9 + x_{13} = 0$
$c_2 : x_2 + x_6 + x_{10} + x_{14} = 0$
$c_3 : x_3 + x_7 + x_{11} + x_{15} = 0$
$c_4 : x_4 + x_8 + x_{12} + x_{16} = 0$
$c_5 : x_1 + x_2 + x_3 + x_4 = 0$
$c_6 : x_5 + x_6 + x_7 + x_8 = 0$
$c_7 : x_9 + x_{10} + x_{11} + x_{12} = 0$
$c_8 : x_{13} + x_{14} + x_{15} + x_{16} = 0$

Figure 1.2: Tanner Graph representation of a sparse-graph code. The symbols of the codeword are shown on the left as variable nodes, and the check nodes corresponding to linear constraints on the symbols are shown on the right. If a variable node appears in the linear equation corresponding to a check node, that variable node and the check node are connected by an edge.

At each round of peeling, first, singleton check nodes resolve the variable node connected to them. Then, the contributions of these resolved variable nodes are removed from the graph. This induces a new graph with fewer unresolved variable nodes, and the peeling process is iterated. The procedure stops when there are no singleton check nodes left. If all check nodes are zerotons, we have recovered all erased symbols.

It is important to characterize when this procedure recovers all erased symbols. The theoretical analysis for decoding a sparse-graph code through peeling is deferred to Appendix A, but we give, at a high-level, the results that will be useful for designing our algorithms. Let r be the number of check nodes. Asymptotically in the number of message symbols n, we can design sparse-graph codes while keeping r/n arbitrarily close to p_e and recover the erasure pattern with high probability. This means that we can design codes that are arbitrarily close to optimal. Since check nodes correspond to linear constraints, each check node takes away one degree of freedom from the message sequence $x_1, \ldots, x_n$. Hence, (if these linear constraints are independent) the aggregate redundancy rate is equal to r/n, which can be made arbitrarily close to p_e while enabling recovery of the erasures with


x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
x_9
x_{10}
x_{11}
x_{12}
x_{13}
x_{14}
x_{15}
x_{16}
$c_1 : x_1 + x_5 + x_9 + x_{13} = 0$
$c_2 : x_2 + x_6 + x_{10} + x_{14} = 0$
$c_3 : x_3 + x_7 + x_{11} + x_{15} = 0$
$c_4 : x_4 + x_8 + x_{12} + x_{16} = 0$
$c_5 : x_1 + x_2 + x_3 + x_4 = 0$
$c_6 : x_5 + x_6 + x_7 + x_8 = 0$
$c_7 : x_9 + x_{10} + x_{11} + x_{12} = 0$
$c_8 : x_{13} + x_{14} + x_{15} + x_{16} = 0$
$c_1 : 0$
$c_2 : x_6 = -(x_2 + x_{10} + x_{14})$
$c_3 : x_{11} + x_{15} = -(x_3 + x_7)$
$c_4 : x_4 = -(x_8 + x_{12} + x_{16})$
$c_5 : 0$
$c_6 : x_6 = -(x_5 + x_7 + x_8)$
$c_7 : x_{11} + x_{12} = -(x_9 + x_{10})$
$c_8 : x_{15} = -(x_{13} + x_{14} + x_{16})$
$c_1 : 0$
$c_2 : 0$
$c_3 : x_{11} = -(x_3 + x_7 + x_{15})$
$c_4 : 0$
$c_5 : 0$
$c_6 : 0$
$c_7 : x_{11} = -(x_9 + x_{10} + x_{12})$
$c_8 : 0$
$c_1 : 0$
$c_2 : 0$
$c_3 : 0$
$c_4 : 0$
$c_5 : 0$
$c_6 : 0$
$c_7 : 0$
$c_8 : 0$

(a) In this example, the channel erases symbols x_6, x_{11}, x_{12}, and x_{15}.

(b) First, the contributions of non-erased symbols are subtracted from the check nodes to yield a pruned graph. In the pruned graph, we see that c_2, $c6$, and c_8 are singleton check nodes, and they recover x_6, x_{12}, and x_{15}. The contributions of these symbols are then removed from the check nodes they connect to.

(c) After the first round of peeling, we then see that c_3, and $c7$ are singleton check nodes, and they recover x_{11}. The contributions of this symbol is then removed from the check nodes it connects to.

(d) After 2 rounds of peeling, all the erasures are recovered.

Figure 1.3: Decoding a sparse-graph code on an erasure channel. The unknown symbols (erasures) are marked with filled variable nodes.

high probability. As a redundancy rate of p_e is necessary for reliable communication on an erasure channel with erasure probability p_e, we are arbitrarily close to optimal.

In our work, we are going to use sparse-graph codes not for communications but for big-data problems. In doing so, it will be easier to conceptualize our algorithms by starting from the pruned graph. Figure 1.4 summarizes the process starting from the pruned graph abstracting away the application.

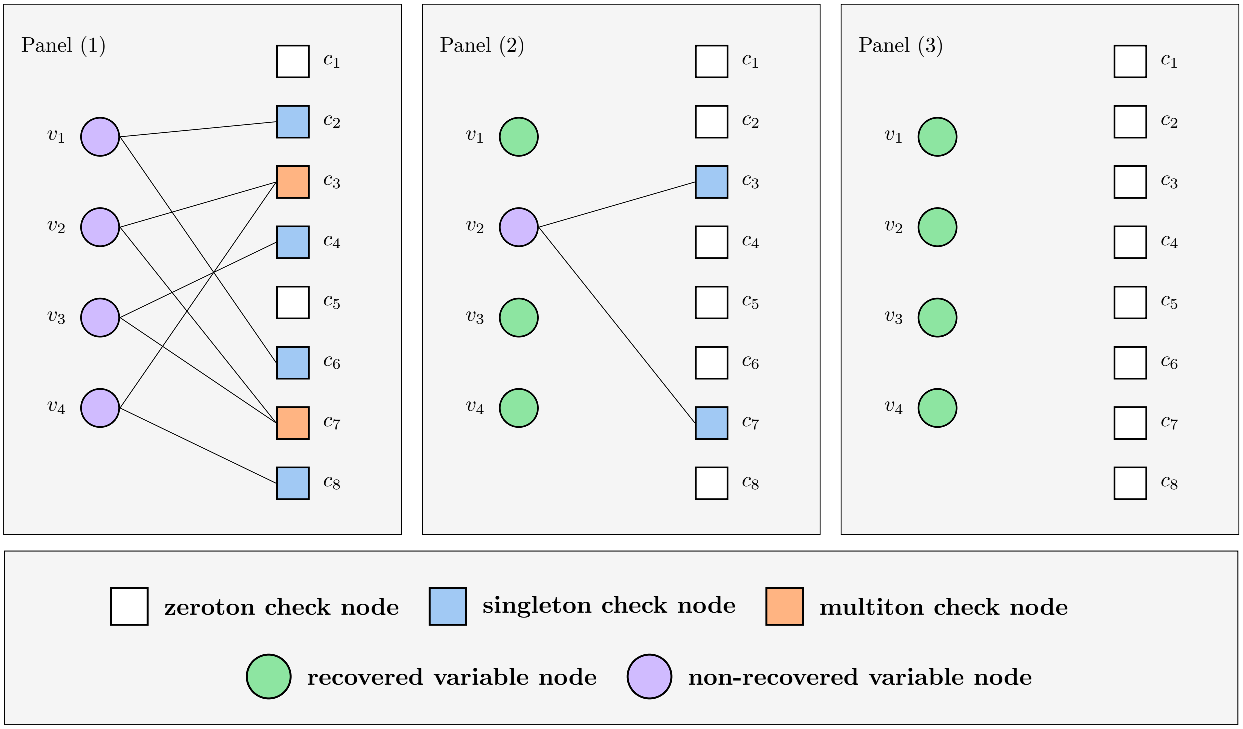


Figure 1.4: Decoding on a sparse-graph code through peeling. At each round, messages are passed between the check nodes and the variable nodes. First, singleton check nodes recover the variable nodes that are connected to them, and then, the recovered variable nodes remove edges connected to them from the connectivity graph. The peeling stops when there are no singleton check nodes remaining. Here, panel (1) shows the connectivity between the variable nodes and the check nodes. We see that c_2, c_4, c_6, and c_8 are singleton check nodes, and they recover variable nodes v_1, v_3, and v_4. After the first round of peeling, the connections from these variable nodes are removed from the graph to yield the connectivity in panel (2). In the second round, we see that c_3 and c_7 are singleton check nodes, and they recover the variable node v_2. In this example, in two rounds of peeling, all the variable nodes are recovered as can be seen in panel (4).

Part I

Classic Problems of Interest

In this part of the thesis, we study speeding up “classical” problem settings:

- **Chapter 2:** Sampling signals with sparse spectrum below the Nyquist Rate
- **Chapter 3:** Improving the noise robustness of divide-and-conquer-based sparse DFT algorithms
- **Chapter 4:** Fast MRI and theory for fast signal recovery with wavelet-domain sparsity

Chapter 2

Minimum-Rate Spectrum-Blind Sampling

2.1 Introduction

In the past few decades, we have witnessed the power of sampling theory ushering into the digital age. This has been enabled by a diverse set of tools from mathematics and signal processing, which allow us to capture continuous-time signals through discrete-time representations without loss of information. In particular, the celebrated sampling theorem, also known as the Shannon-Nyquist sampling theorem [80], has provided the foundation for digital signal processing through analog-to-digital conversion (ADC) over the last century. The basic theorem states that any real-valued bandlimited signal can be reconstructed from its point-wise samples taken at or above the Nyquist rate that is twice the maximum frequency component of the underlying signal. With a deeper understanding of the concept of signal spaces, advances in signal processing and harmonic analysis have contributed significantly to the field of sampling theory, leading to sophisticated extensions of the basic Nyquist sampling framework [88, 91, 18].

The problem of *spectrum-blind* sampling has attracted considerable recent interest, mostly because of its promise of offering significant reduction in sampling rate over the Nyquist rate when the signal spectrum is sparse but its frequency support is *not known a priori*. This important class of bandlimited signals is commonly referred to as *multiband signals*, which consist of a union of (arbitrary) continuous spectral bands spread across a wide spectrum. There exists a rich body of research related to the theory and algorithms for sampling this class of signals. As shown in the seminal work by Landau in 1960's [43], any bandlimited signal with an *arbitrary but known* frequency support can be recovered from samples taken at a rate equal to the measure of its frequency support (that is, the sum of existing spectral bands), also known as the *Landau rate*. Interestingly, it was later pointed out by the authors of [53] that *spectrum-blind* sampling of multiband signals requires a minimum sampling rate of *twice the Landau rate*, establishing that the price of not knowing the signal spectrum support is a factor of 2 over the Landau rate. This result guarantees that given samples of a multiband signal taken at twice the Landau rate, there is a unique multiband signal that generates those samples. However, the result does not provide a constructive

This chapter is based on joint work with Xiao Li [62].

method for recovering that unique multiband signal.

In order to constructively approach this minimum rate, the authors of [24, 10] showed that using multicoset sampling, the signal can be reconstructed from the discrete-time Fourier transform (DTFT) of the samples if the cosets are chosen carefully. Another approach was proposed by the authors of [58, 57] where the problem of spectrum-blind sampling was transformed to a compressed sensing problem through modulation by random periodic functions and low pass filtering. However, the proposed scheme is not tight as it incurs an extra logarithmic factor away from the minimum rate, and the computational complexity depends on the total bandwidth of the signal.

In this chapter, we study this open problem (and sampling theory in general) from a radically different angle based on *coding theory*, which at the surface seems unrelated to the problem of interest, and has been largely unexplored to-date in *sampling theory*. This is partly due to the disparity between the two fields with respect to problem statements, goals, and concepts, which have had little overlap in the past. Here, our prime goal is to first shed some new light on general sampling designs from a coding-theoretic perspective, and further pinpoint the unexplored benefits of sub-Nyquist sampling strategies based on *sparse-graph codes*, which have theoretically and practically revolutionized the design of modern communication systems. This allows us to derive a novel sampling framework that reflects interesting connections between the fundamental problem of *minimum-rate spectrum-blind sampling*, and the well-studied problem of designing *capacity-achieving* sparse-graph codes, such as Low Density Parity Check (LDPC) codes [26, 74].

2.1.1 Our Contributions

The key contribution of our framework is the coding-theoretic design of a multichannel *sparse-graph-coded* filter bank sampling architecture, which approaches the minimum spectrum-blind-sampling rate constructively with fast reconstruction. By treating the underlying signal as a *union of multiple disjoint spectral bands* such that the *frequency support occupies only a few slices*, the main idea is to design each sampling channel in the filter bank such that undersampling the filtered output leads to a certain aliasing pattern on all the spectral bands. This is similar in spirit to inducing *parity check* constraints for the sparse spectral components of the signal residing at each existing frequency band. Therefore, the reconstruction is equivalent to decoding the sparse spectral slices from the parity check constraints.

To achieve minimum-rate spectrum-blind sampling, we design the sampling filters to induce “good” parity checks that emulate capacity-achieving *sparse-graph codes*. This allows fast decoding through a simple decoder using the minimum number of parity checks (which corresponds to the minimum number of sampling channels). As a result, we can simultaneously achieve the minimum sampling rate and reduce the reconstruction complexity in the spirit of peeling decoding for LDPC codes used in erasure channels. Exploiting the powerful analysis tools in modern coding theory [76] (e.g., *density evolution*), our results show that any multiband signal can be sampled asymptotically at the minimum rate in a probabilistic setting, where the probability of reconstruction failure goes to zero. To the best of our knowledge, this is the first theoretical result that achieves the minimum rate for *spectrum-blind* sampling, and meanwhile, admits a constructive scheme for reconstructions at such rate.

2.1.2 Organization and Notation

This chapter is organized as follows. In Section 2.2, we state the problem and describe the key underlying ideas behind our results. In Section 2.3, we present a multiband filter bank sampling architecture, and in Section 2.4, we describe the design of sparse-graph code filter banks that enable minimum-rate sampling of wideband signals, and describe an implementation for realizing such filters. We then derive an extension of our method for the discrete-time setting in Section 2.5. We provide simulation results in Section 2.6, and present concluding remarks in Section 2.7. In order to improve readability, the technical proofs of our main results are deferred to Appendix 2.A.

Functions denoted by upper case letters are in the Fourier domain, e.g., $X(f)$ denotes the Fourier transform of $x(t)$ defined as

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}\, \mathrm{d}t. \tag{2.1}$$

Whenever it is convenient, we denote Fourier transform pairs as $x(t) \overset{\mathcal{F}}{\longleftrightarrow} X(f)$. The complex conjugate of a number $z \in \mathbb{C}$ is denoted as z^*. Lower case bold letters denote vectors, e.g., $\mathbf{x}$, and upper case bold letters denote matrices, e.g., $\mathbf{A}$. There will be additional definitions introduced as needed; for a quick lookup, Table 2.1 gives a brief description of the upcoming definitions.

Table 2.1: Glossary of important notation for Chapter 2.

Notation	Definition
$x(t)$:	input signal
$f_{\max}$:	maximum bandwidth of the input
f_L :	Lebesgue measure of the frequency support of input signal
N :	number of frequency bands divided within $[0, f_{\max})$
B :	width of each band, $B = f_{\max}/N$
$x_k(t)$:	signal at band k moved to baseband $[0, B)$
$y_i(t)$:	output of the multiband filter i
$y_i[n]$:	subsampled output of the multiband filter i
$h_i(t)$:	multiband filter i

2.2 Design Objective and Philosophy

We first give a precise definition of our signals of interest.

Definition 1. By $\mathrm{BandLimited}(f_{\max}, f_L)$ we denote signals $x(t) \in \mathbb{C}$ whose frequency spectrum is zero outside[1] $[0, f_{\max})$, i.e.,

$$X(f) = 0 \quad \text{if} \quad f < 0 \text{ or } f \geq f_{\max}, \tag{2.2}$$

and its frequency support $\mathcal{F}$ has Lebesgue measure $|\mathcal{F}| = f_L$ and is of the form of the union of a finite number of intervals with unknown locations.

An example signal from $\mathrm{BandLimited}(f_{\max}, f_L)$ is given in Figure 2.1.

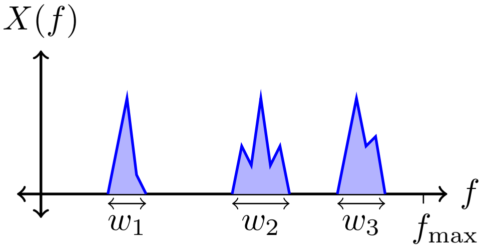


Figure 2.1: Illustration of a signal from $\mathrm{BandLimited}(f_{\max}, f_L)$. In this example, the Lebesgue measure of the frequency support is $|\mathcal{F}| = w_1 + w_2 + w_3 = f_L$.

Corollary 1. *The minimum rate for sampling signals in* $\mathrm{BandLimited}(f_{\max}, f_L)$ *so that lossless recovery is possible is* $\min(2f_L, f_{\max})$ *[53].*

2.2.1 Sampling Framework and Design Objective

Our sampling framework is summarized in Figure 2.2. An input signal given by Definition 1 is passed through a filter bank consisting of M filters with frequency response $H_i(f)$ for $i = 1, \ldots, M$. The filter bank outputs are sampled regularly at B Hz to produce M streams of samples $\{y_i[n]\}_{n \in \mathbb{Z}}$ for $i \in \{1, \ldots, M\}$, which are then fed as input to a (non-linear) reconstruction scheme.

Our goal is to design a spectrum-blind, lossless sampling and reconstruction scheme that achieves a sampling rate approaching the minimum rate, and that perfectly recovers the signal (with probability approaching 1 asymptotically in M). Given a certain reconstructed signal $\widehat{x}(t)$, the performance of our design is characterized by the triplet $(f_s, T, \mathbb{P}_F)$, where f_s is the aggregate sampling rate, T is the reconstruction complexity in terms of arithmetic operations, and $\mathbb{P}_F$ is the failure probability defined as

$$\mathbb{P}_F = \Pr\left(\widehat{x}(t) \neq x(t)\right). \tag{2.3}$$

The probability $\mathbb{P}_F$ is evaluated with respect to the randomness associated with the sampling frontend design. In other words, for any given sparse wideband signal $x(t)$, our design generates a

[1] Note that the choice of the lower limit to be 0 is without loss of generality because any given signal with known frequency limits $[f_{\min}, f_{\min} + f_{\max})$ can be modulated to the band $[0, f_{\max})$ by a multiplication with $e^{-j2\pi f_{\min} t}$.

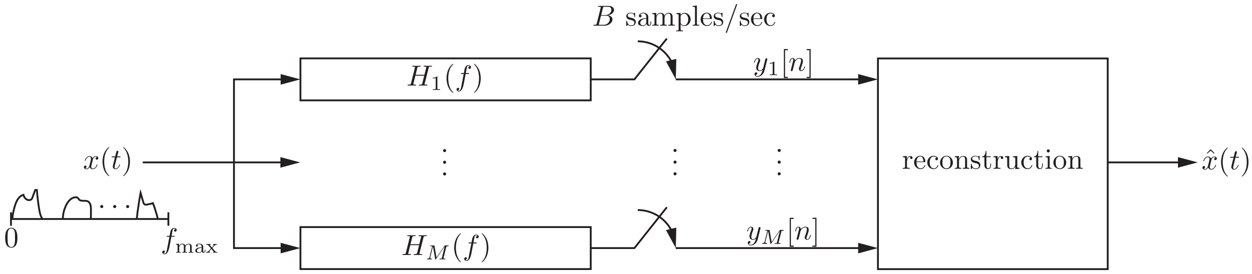


Figure 2.2: Proposed framework for spectrum-blind sampling and reconstruction of bandlimited multiband signals. The framework consists of a sampling frontend and a reconstruction backend. The frontend takes sub-Nyquist-rate samples of a filter bank of carefully designed sparse-graph-code based multiband filters (see Section 2.4 for details). The backend performs decoding by fast peeling on a sparse-graph induced by the filter bank structure (see Section 2.2.2 for details).

random sampling frontend (from a specific random ensemble[2]) and produces an estimate $\widehat{x}(t)$ where probability $1 - \mathbb{P}_F$ approaches 1 asymptotically in M. Some questions of interest are: (1) How to design the filter response $H_i(f)$ and how to choose the sampling rate B? (2) How many filters (i.e., M) are necessary? How are the design parameters B, M related to the minimum rate? (3) How to reconstruct the signal given the filter bank output samples $\{y_i[n]\}_{n\in\mathbb{Z}}$ for $i \in \{1, \dots, M\}$? Before answering the above questions in details, we first explain our design philosophy using the cartoon illustration in Figure 2.3.

2.2.2 Design Philosophy

We explain our design philosophy based on the example provided in Figure 2.3, where a signal from $\mathrm{BandLimited}(4B, 3B)$ is considered. For simplicity, the frequency spectrum is sliced into 4 intervals of equal width B. In this example, the input in Figure 2.3 is supported on 3 out of the 4 slices, over bands 0, 2 and 3 color-coded in *red, green* and *blue* respectively[3].

To sample this signal from $\mathrm{BandLimited}(4B, 3B)$, we use a filter bank with $M = 3$ sampling channels, where each sampling filter has a frequency response $H_i(f)$ taking a *multiband* structure. In this example, each multiband filter has unity gains, $H_i(f) = 1$, in some bands and $H_i(f) = 0$ elsewhere. For example, in Figure 2.3, the first filter passes bands 0 and 2, the second filter passes bands 1 and 2, and the third filter passes bands 0, 1 and 3. Then, conforming with the multiband structure of the sampling filter, the associated filter output is also (sub)sampled at rate B. Clearly, sampling at rate B results in *aliasing* of the output spectrum, or namely the discrete-time Fourier transform (DTFT) of output samples $\{y_i[n]\}_{n\in\mathbb{Z}}$ for $i \in \{1, \dots, M\}$, because the rate is below the Nyquist rate $4B$. However, due to the variations of the passbands across the sampling filters, the aliasing pattern observed at each channel is different; this is depicted in Figure 2.3 as mixtures of different colors. That is to say, each output stream of samples is a linear combination of signal

[2] In the compressed sensing literature, this is what is known as the 'for-each' guarantee [27] in contrast to the 'for-all' guarantee, where, in the latter, a single measurement matrix (once generated) is used for all signals.

[3] The 'shape' of the spectral content in each band can be arbitrary. We use rectangles for simplicity of illustration.

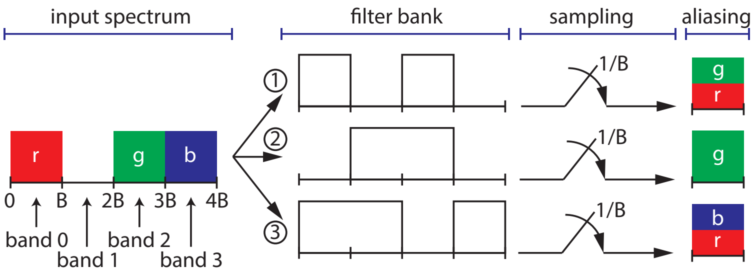


Figure 2.3: The illustration of the aliased spectrum at the outputs of the sampling channels. Each multiband sampling filter passes a controlled subset of bands of the signal, and subsampling introduces aliasing of these components, that is, introduces a linear combination of the signals residing on the passed bands.

components in a controlled selection of bands. For example, stream 1 is a linear combination of signals in bands 0 and 2 (red and green), and stream 2 is equal to the signal in band 2 (green) since there is no signal in band 1.

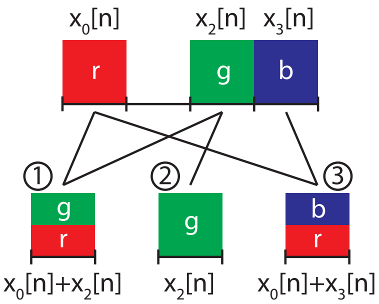


Figure 2.4: Spectral aliasing induces *parity check* constraints for each sampling instant. The reconstruction is equivalent to finding the sample values at the bands that satisfy these parity check constraints.

It will become clear in Section 2.4 that the design of these filters is guided by coding patterns from *sparse-graph codes* to create careful linear mixing of the spectral components. For illustration, we highlight such mixing in Figure 2.4 in the form of sparse-graph codes, where each color mixture corresponds to a *parity check* constraint of the codeword for each sampling snapshot n. This sparse bipartite graph structure helps decode the signal at each frequency band through peeling the edges off the graph. If the decoder knows that a certain stream of samples is representative of a single color, it can peel off its contribution from other streams of samples, resulting in new streams of samples. This is similar in spirit to the 'peeling decoder' in sparse-graph codes, and decodes the input by iteratively unmixing the aliased spectral components.

Next, we describe how to reconstruct the input through peeling with reference to Figure 2.5. The aliasing structure resulting from the subsampling of multiband filter outputs can be rendered as a bipartite graph shown in the 'aliasing structure' box in Figure 2.5. For the sake of simplicity, let us first assume that we have a mechanism that informs the decoder if an output stream consists of

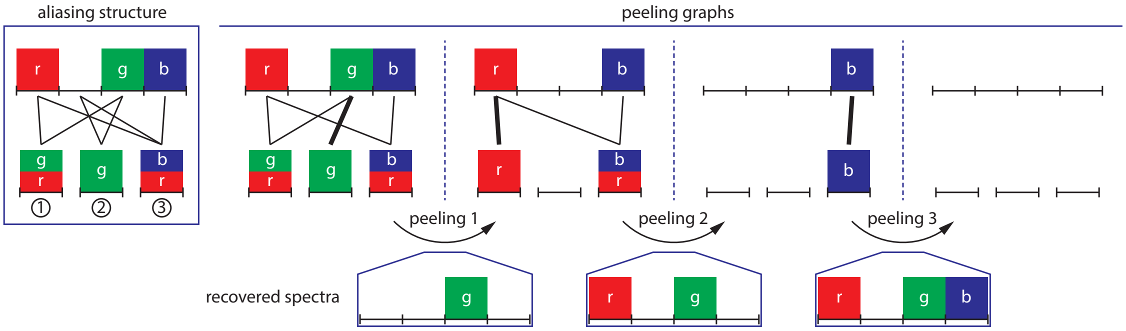


Figure 2.5: Illustration depicting a three stage peeling. The decoder detects streams composed of a single color and peels this color off from the streams that it appears.

spectral components coming from a single band or not, and if so, which band it is coming from (we will later reveal how to realize this mechanism). Given this mechanism, we perform the following:

- Stream ② is composed of a single component colored by green, so we recover the signal residing in the green component;
- Because the green component contributes to streams ① and ②, we peel it off from them (peeling 1 in the figure).
- Then stream ① is made out of a single component marked with red, so we recover the signal residing in the red component;
- Because the red component contributes to streams ① and ③, we peel it off from them (peeling 2 in the figure);
- Then stream ③ is composed of a single component colored by blue, so we recover the signal residing in the blue component;
- Because the blue component contributes to stream ③, we peel it off from it (peeling 3 in the figure);
- After these peeling rounds, the signal is completely recovered.

2.3 Spectrum-Blind Sampling

We describe our approach through an example.

2.3.1 A Simple Example

We now give a mathematically detailed account for Figure 2.3 and Figure 2.5. Let the input signal be

$$x(t) = \sum_{k=\{0,2,3\}} e^{jB\pi(2k+1)t} \sum_{n=-\infty}^{\infty} (-1)^n x_k[n] \operatorname{sinc}(Bt - n),$$

where $x_k[n] \in \mathbb{C}$, and

$$\operatorname{sinc}(t) = \begin{cases} \sin(t)/t & \text{if } t \neq 0 \\ 1 & \text{if } t = 0. \end{cases}$$

Because we have

$$\operatorname{sinc}(Bt) \overset{\mathcal{F}}{\longleftrightarrow} \operatorname{box}_B(f) := \begin{cases} 1/B & \text{if } |f| \leq B/2 \\ 0 & \text{otherwise,} \end{cases}$$

the spectrum of $x(t)$ has the support shown in Figure 2.3 and Figure 2.5. Let filters $h_1(t)$, $h_2(t)$ and $h_3(t)$ denote the multiband filters shown in Figure 2.3, i.e, their frequency responses are

$$\begin{aligned} H_1(f) &= \operatorname{box}_B(f - B/2) + \operatorname{box}_B(f - 5B/2), \\ H_2(f) &= \operatorname{box}_B(f - 3B/2) + \operatorname{box}_B(f - 5B/2), \\ H_3(f) &= \operatorname{box}_B(f - B/2) + \operatorname{box}_B(f - 3B/2) + \operatorname{box}_B(f - 7B/2). \end{aligned} \tag{2.4}$$

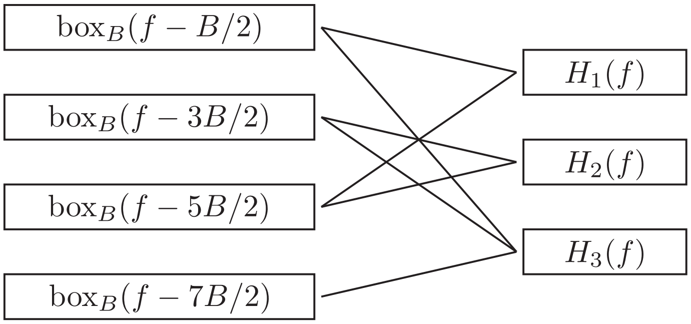

$\operatorname{box}_B(f - B/2)$
$\operatorname{box}_B(f - 3B/2)$
$\operatorname{box}_B(f - 5B/2)$
$\operatorname{box}_B(f - 7B/2)$
$H_1(f)$
$H_2(f)$
$H_3(f)$

Figure 2.6: The design of the multiband sampling filters is guided by a sparse-graph. The design methodology is described in detail in Section 2.4.2.

The design of the multiband filters is guided by sparse-graph codes, which are explained in more detail in Section 2.4. In particular, the filters in (2.4) are designed from the graph shown in Figure 2.6. Note that each multiband filter passes signal components supported on a controlled subset of bands. Furthermore, when $f_L \leq f_{\max}$, the signal is supported on a subset of its possible bandwidth. Hence, the output of each multiband filter will have components at frequencies where its passband and the input's frequency support intersect. This introduces a sparse relation between filter outputs and the components of the input signal at the frequency bands. In other words, each filter output is a linear combination of a small number of components of signal at the frequency bands. This relation results in a sparse-graph structure with respect to input and the measurements.

To see the sparse-graph structure for our example, define signals at the outputs of the filters, $y_i(t)$, and their samples, $y_i[n]$, as

$$y_i(t) := x(t) * h_i(t), \tag{2.5}$$

$$y_i[n] := y_i(n/B). \tag{2.6}$$

it follows that $y_1(t) = x(t) * h_1(t) = \sum_{n=-\infty}^{\infty} \left((-1)^n x_0[n] e^{jB\pi t} + (-1)^n x_2[n] e^{jB\pi 5t}\right) \operatorname{sinc}(Bt - l)$, and

$$y_1[n] = y_1(n/B) = x_0[n] + x_2[n],$$

likewise,

$$\begin{aligned} y_2[n] &= x_2[n], \\ y_3[n] &= x_0[n] + x_3[n]. \end{aligned}$$

For each sampling instant n, the samples $\{y_i[n]\}_{i=1,2,3}$ can be viewed as a set of constraints that input samples need to satisfy, and the resulting sparse-graph structure can be seen in Figure 2.4. With regard to the connectivity of the sparse-graph, we categorize the streams into the following types:

1. **Zeroton:** a stream is a zeroton if it does not involve any contribution from the input signal.
2. **Singleton:** a stream is a singleton if it involves contribution coming from a single band of the input. More specifically, we refer to the band k of the non-zero contribution and the sample measurement $x_k[n]$ as the 'band-sample' pair for that singleton.
3. **Multiton:** a stream is a multiton if its value is the superposition of contributions from more than one band.

To help describe the decoding algorithm, let us first assume that there exists an 'oracle' that informs the decoder which streams are singletons and provides the 'band-sample' pair for such streams (next paragraph describes how to practically implement an equivalent mechanism). In this example, in the first peeling round, the oracle informs the decoder that band 2 is a singleton with value $x_2[n]$, which we compactly write as $(2, x_2[n])$, Then the decoder subtracts this contribution from all measurements that have contribution from band 3, in this example, the stream 1, forming a new singleton. Hence, the peeling on this example works as follows:

1. oracle informs decoder $(2, x_2[n])$, decoder peels $x_2[n]$ from stream 1 to get residual $y_1[n] - x_2[n] = x_0[n]$,
2. oracle informs decoder $(0, x_0[n])$, decoder peels $x_0[n]$ from stream 3 to get residual $y_3[n] - x_0[n] = x_3[n]$,
3. oracle informs decoder $(3, x_3[n])$, and the whole signal is recovered.

The above discussion assumed an oracle. We can realize an equivalent mechanism to the oracle by introducing a second set of filters defined as

$$\begin{aligned} H_{c,1}(f) &= \operatorname{box}_B(f - B/2) + e^{j2\frac{2\pi}{4}} \operatorname{box}_B(f - 5B/2), \\ H_{c,2}(f) &= e^{j\frac{2\pi}{4}} \operatorname{box}_B(f - 3B/2) + e^{j2\frac{2\pi}{4}} \operatorname{box}_B(f - 5B/2), \\ H_{c,3}(f) &= \operatorname{box}_B(f - B/2) + e^{j\frac{2\pi}{4}} \operatorname{box}_B(f - 3B/2) + e^{j3\frac{2\pi}{4}} \operatorname{box}_B(f - 7B/2), \end{aligned}$$

where we used the subscript 'c' to stand for 'coded'. Then, similarly to (2.5) and (2.6), defining the signals,

$$y_{c,i}(t) := x(t) * h_{c,i}(t), \tag{2.7}$$
$$y_{c,i}[n] := y_{c,i}(n/B), \tag{2.8}$$

we have

$$\begin{aligned} y_{c,1}[n] &= x_0[n] + e^{j2\frac{2\pi}{4}} x_2[n], \\ y_{c,2}[n] &= e^{j2\frac{2\pi}{4}} x_2[n], \\ y_{c,3}[n] &= x_0[n] + e^{j3\frac{2\pi}{4}} x_3[n]. \end{aligned}$$

Now with these measurements, one can effectively determine if a stream is coming from a single band with a ratio test

$$\frac{\angle(y_{c,i}[n]/y_i[n])}{2\pi/4}, \tag{2.9}$$

where $\angle(\cdot)$ returns the argument of its input. If the ratio (2.9) is an integer, it indicates that the sample $y_i[n]$ comes from a single band, and the value of the ratio gives the band index. On the other hand, if $y_i[n]$ is a superposition of signals coming from multiple bands, then under conditions that are going to be stated later on, the ratio will not be an integer. As a last case, if the sample has no contribution from any support bands then the measurement $y_i[n]$ will be zero. Hence, by making use of the two filter banks, we have a way to detect zerotons, singletons, multitons, and perform the operations of the oracle.

This simple example shows how the problem of sampling and reconstructing sparse wideband signals can be cast as an instance of sparse-graph decoding. The sparse bipartite graph in this example only shows the idea of peeling decoding, but does not guarantee successful sampling and reconstruction for an arbitrary signal. Furthermore, this example also suggests that it is possible to obtain the band-sample pair of any singleton without the help of an 'oracle' by adding more sampling filters. In the following, we present our general sampling architecture, and then later discuss in detail how to design the sampling filters for the proposed architecture.

2.3.2 Sampling Architecture and Observation Model

Choose $N \in \mathbb{Z}^{++}$ and divide the interval $[0, f_{\max})$ into N intervals of equal width $B = f_{\max}/N$. We can write any signal bandlimited to $[0, f_{\max})$ as a sum

$$x(t) = \sum_{k=0}^{N-1} x_k(t) e^{j2\pi kB}, \tag{2.10}$$

where each $x_k(t) \in \mathbb{C}$ is bandlimited to the frequency interval $[0, B)$. Because this is a universal representation, the signal $x(t)$ can be reconstructed if each $x_k(t)$ is recovered. As each $x_k(t)$ is bandlimited to $[0, B)$, it can be recovered from its samples taken at rate B

$$x_k[n] = x_k(n/B). \tag{2.11}$$

In the case where the signal is from $\mathrm{BandLimited}(f_{\max}, f_L)$ with $f_L \ll f_{\max}$, the number of active components in (2.10) is significantly less than N whenever the frequency spectrum is sliced finely. More precisely, by assumption of a finite union of intervals, there exists a lower bound $B_0 > 0$ for the smallest consecutive interval of the signal spectrum. Using this fact, we see that only K substreams are active, i.e., $x_k[n] \neq 0$, where $f_L/B \leq K \leq f_L/B + f_L/B_0$. Note that, as the slicing becomes finer, namely, as N grows, we have the ratio of the active components of $\mathbf{x}[n]$ to N approach the frequency occupancy ratio, that is,

$$\lim_{N\to\infty} \frac{|\mathrm{supp}\,(\mathbf{x}[n])|}{N} = \frac{f_L}{f_{\max}}. \tag{2.12}$$

To leverage sparsity in the number of active components in order to reduce the sampling rate, define M multiband filters, $h_i(t)$, where each filter has frequency response a weighted combination of ideal bandpass filters, generalizing the example given in Section 2.3.1:

$$H_i(f) = \sum_{k=0}^{N-1} c_{i,k}\,\mathrm{box}_B(f - kB - B/2).$$

Under this design, M channels sampled at rate B result in an aggregate sampling rate of $f_s = MB$. We want to design the filter bank such that as the number of slicing N grows, f_s approaches the minimum rate $2f_L$. The resulting M samples output from the filter bank at each sampling instant $n \in \mathbb{Z}$ can be described as

$$y_i[n] = \sum_{k=0}^{N-1} c_{i,k} x_k[n]. \tag{2.13}$$

The multiple measurements obtained through the use of M different $h_i(t)$ can be represented, for each sampling instant n, by a linear equation

$$\mathbf{y}[n] = \mathbf{C}\mathbf{x}[n], \tag{2.14}$$

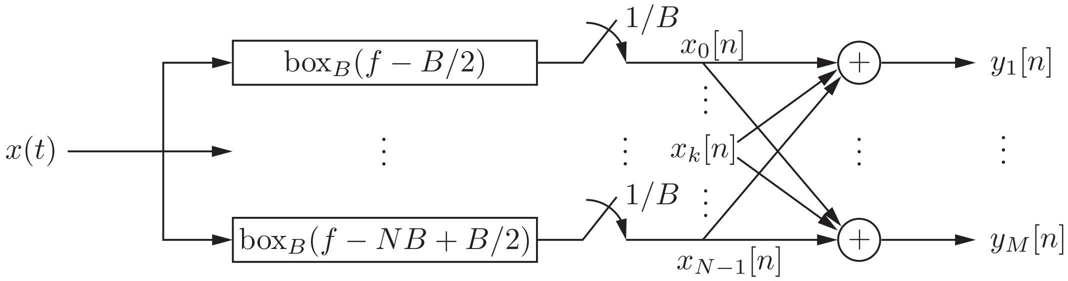


Figure 2.7: Illustration representing the mathematical relation between the input signal and the output samples.

where $\mathbf{C} \in \mathbb{C}^{M \times N}$. The problem then boils down to designing the matrix $\mathbf{C}$ that will enable recovery of $\mathbf{x}[n]$ from $\mathbf{y}[n]$. When the frequency support is sparse, the vector $\mathbf{x}[n]$ is going to be sparse for each n. This sparse recovery problem has been studied extensively in compressed sensing where there is a prolific body of literature with various measurement scaling results for M [18]. A typical construction is to choose $\mathbf{C}$ from some random matrix ensemble. For example, the authors of [58] proposed using such randomized constructions that require oversampling with a logarithmic factor $\log(N)$ with respect to the number of slices, which does not achieve the minimum rate. Another approach is to design $\mathbf{C}$ with a Vandermonde structure, where by using Prony's method, one can recover the signal using samples twice the number of non-zero entries in $\mathbf{x}[n]$ [33]. However, Prony's method requires solving for the roots of a polynomial. As the number of slices increases, the numerical precision for the roots needs to scale with N, increasing the computational complexity and the numerical instability. To approach the minimum sampling rate together with low computational complexity, we use the framework developed in [52] based on sparse-graph codes, which circumvents the above issues.

2.4 Filter Bank Design

The compressed sensing formulation in (2.14) captures the filter bank architecture by the $M \times N$ coefficient matrix $\mathbf{C}$, and the goal is to recover the baseband samples $\mathbf{x}[n]$ in each band from the filter bank outputs $\mathbf{y}[n]$. More specifically, each row of the matrix $\mathbf{C}$ characterizes the frequency response of the sampling filter in each channel of the filter bank. In the following, we discuss how to design the frequency response of the filter bank to achieve minimum rate sampling and fast reconstruction.

2.4.1 Filter Bank Response

Here we specify our filter bank frequency response in terms of its *magnitude response* over N bands and *phase response* over N bands respectively. In our sampling framework, the *magnitude response* is sparse-graph encoded to introduce peeling-friendly aliasing artifacts, and the *phase response* is appropriately chosen for the purpose of identifying the original band location of each baseband signal samples.

We specify R *magnitude responses* for the sampling filters through an $R \times N$ magnitude response matrix $\mathbf{H}$, and further P associated *phase responses* for each *magnitude response* through a $P \times N$ phase response matrix $\mathbf{S}$ such that $M = RP$ for some positive integers R and P. Given an $R \times N$ *magnitude response matrix* $\mathbf{H}$, and a $P \times N$ *phase response matrix* $\mathbf{S}$, the $M \times N$ filter bank response matrix $\mathbf{C}$ is given by

$$\mathbf{C} = \mathbf{H} \boxtimes \mathbf{S}, \tag{2.15}$$

where $\boxtimes$ is a row-tensor product. For example, let the magnitude response matrix $\mathbf{H}$ be a sparse matrix consisting of $\{0, 1\}$ and the phase response matrix $\mathbf{S}$ be chosen as the first two rows of a

DFT matrix as in the example in Section 2.3.1 as

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega_7 & \omega_7^2 & \omega_7^3 & \omega_7^4 & \omega_7^5 & \omega_7^6 \end{bmatrix}, \tag{2.16}$$

where $\omega_7 = e^{\mathrm{i}\frac{2\pi}{7}}$. Then the row-tensor product is given by

$$\mathbf{C} = \mathbf{H} \boxtimes \mathbf{S} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & \omega_7 & 0 & \omega_7^3 & 0 & \omega_7^5 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & \omega_7 & 0 & \omega_7^3 & 0 & 0 & \omega_7^6 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & \omega_7^3 & \omega_7^4 & \omega_7^5 & \omega_7^6 \end{bmatrix}. \tag{2.17}$$

In this example, the filter bank has $M = 6$ channels, where there are 3 magnitude response patterns where each pattern is further associated with 2 phase response patterns. Next we formally define our filter bank ensemble using this construction.

Definition 2 (Filter Bank Ensemble $\mathcal{Q}(N, B, M)$)**.** Let the sampling frontend be an instance from the ensemble of M-channel filter banks. Given an $R \times N$ *magnitude response matrix* $\mathbf{H}$ and a $P \times N$ *phase response matrix* $\mathbf{S}$, the $M \times N$ filter bank coefficient matrix $\mathbf{C}$ associated with the filter bank ensemble $\mathcal{Q}(N, B, M)$ is given by

$$\mathbf{C} = \mathbf{H} \boxtimes \mathbf{S}, \tag{2.18}$$

where the $R \times N$ magnitude response matrix $\mathbf{H} = [H_{r,n}]_{R\times N}$ is chosen as the adjacency matrix of a bipartite graph $\mathcal{G}$ consisting of N left nodes $V_1 := [N]$ and R right nodes $V_2 := [R]$ with an edge set $\mathcal{E} := V_1 \times V_2$, and the $P \times N$ phase response matrix is chosen as the first two rows of an N-point DFT matrix:

$$\mathbf{S} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_N & \omega_N^2 & \omega_N^3 & \omega_N^4 & \cdots & \omega_N^{N-1} \end{bmatrix}, \tag{2.19}$$

where $\omega_N = e^{\mathrm{i}\frac{2\pi}{N}}$.

Corollary 2. *The measurement* $\mathbf{y}[n] = \mathbf{C}\mathbf{x}[n]$ *is divided into* R *bins as* $\mathbf{y}[n] = [\mathbf{y}_1^\top[n], \ldots, \mathbf{y}_R^\top[n]]^\top$ *with*

$$\mathbf{y}_r[n] = \mathbf{S}\mathbf{z}_r[n], \quad \textit{for } r = 1, \ldots, R, \tag{2.20}$$

where $\mathbf{z}_r[n]$ *is a variant of the vector* $\mathbf{x}[n]$ *sparsified by the connectivity of the right node* r *in graph* $\mathcal{G}$*. Specifically, the support of* $\mathbf{z}_r[n]$ *satisfies*

$$\operatorname{supp}(\mathbf{z}_r[n]) = \operatorname{supp}(\mathbf{x}[n]) \cap \mathcal{N}(r), \tag{2.21}$$

where $N(r)$ *is the left neighborhood of right node* r*.*

Proof. The proof is straightforward and is omitted. □

Next, we discuss the specific constructions for $\mathbf{H}$ and $\mathbf{S}$ in our sampling filter design. We first present the design of the magnitude response matrix $\mathbf{H}$ for blind sampling with the help of a singleton oracle during reconstruction, and then discuss how to use the phase response matrix $\mathbf{S}$ to complete our sampling framework without using the singleton oracle.

2.4.2 Oracle-Based Reconstruction with Blind Sampling

In this section, we describe the design of the magnitude response of the filter bank using a constant phase response $\mathbf{S} = \mathbf{1}_{1\times N}$ such that $\mathbf{C} = \mathbf{H}$. In this setup, we assume that the sampling is done through the filter bank blindly (without knowing the spectral support) and yet the reconstruction benefits from the singleton oracle, which informs the reconstruction process of the band location and sample value of a singleton bin. To define $\mathbf{H}$, we define the *regular graph ensemble* and *irregular graph ensemble* for designing the magnitude response $\mathbf{H}$.

Definition 3 (Regular graph ensemble)**.** Given N left nodes and R right nodes and $d \in \mathbb{Z}^+$ with $d \geq 2$, regular graph ensemble $\mathcal{G}_{\text{reg}}^N(R, d)$ is the set of all graphs where each left node is connected to d right nodes.

Theorem 1. *Given the filter bank ensemble $\mathcal{Q}(N, B, M)$ designed by regular graph ensemble, if there exists a singleton oracle in reconstruction, any signal $x(t) \in \text{BandLimited}(f_{\max}, f_L)$ can be sampled at rate $f_s = 1.23 f_L$ and perfectly reconstructed with probability at least $1 - O(B/f_L)$ by performing $O(f_L)$ arithmetic operations per unit time.*

Proof. The proof is postponed to Appendix 2.A. □

This theorem states that, assuming a singleton oracle in reconstruction, a multiband signal can be sampled a constant 1.23 factor away from the Landau rate. To lower this constant factor, the filter bank can be constructed based on the following graph ensemble.

Definition 4 (Irregular graph ensemble)**.** Given N left nodes and R right nodes and $D \in \mathbb{Z}^+$ with $D \geq 2$, the edge set in the irregular graph ensemble $\mathcal{G}_{\text{irreg}}^N(R, D)$ is characterized by the degree sequence

$$\lambda_j = \frac{1}{L(D)(j-1)}, \quad \text{for } j = 2, \ldots, D+1, \tag{2.22}$$

where λ_j denotes the fraction of edges that connect to a left node with degree j and $L(D) = \sum_{j=1}^{D} 1/j$ is chosen for normalization, i.e., to have $\sum_{j\geq 2} \lambda_j = 1$.

Theorem 2. *Given the filter bank ensemble $\mathcal{Q}(N, B, M)$ designed by irregular graph ensemble, if there exists a singleton oracle in reconstruction, any signal $x(t) \in \text{BandLimited}(f_{\max}, f_L)$ can be sampled at rate $f_s = (1+\epsilon) f_L$ for any $\epsilon > 0$ and perfectly reconstructed with probability at least $1 - O(B/f_L)$ by performing $O(f_L)$ arithmetic operations per unit time.*

Proof. The proof is postponed to Appendix 2.A. □

Filter bank designed through irregular graph ensemble allows sampling at rate $f_s = (1+\epsilon)f_L$ for any $\epsilon > 0$, while the probability of perfect reconstruction is at least $1 - O(B/f_L)$. However, design through irregular graph ensemble needs asymptotics in N for performance. Hence, for empirical validations, we will use the regular graph ensemble.

2.4.3 Blind Reconstruction with Blind Sampling

We now discuss how to use the phase response of the filter bank,

$$\mathbf{S} := \begin{bmatrix} 1 & 1 & \cdots & 1 & 1 \\ 1 & \omega_N & \cdots & \omega_N^{N-2} & \omega_N^{N-1} \end{bmatrix}, \tag{2.23}$$

where ω_N is the N-th root of unity as before, to get rid of the need of an oracle in reconstruction. By using this choice of $\mathbf{S}$, for each filter with a magnitude response specified in $\mathbf{H}$ and a constant phase response, we get an extra auxiliary filter whose magnitude response is identical but with a piecewise linear phase response over the N bands. This introduces a factor of $P = 2$ in the number of sampling channels such that the resulting aggregate sampling rate f_s is twice that of the oracle-based reconstruction with blind sampling.

Using the sample outputs $\mathbf{y}_r[n] = [y_r[n], y_{c,r}[n]]^\top$ for each pair of the filters with the same magnitude response $r = 1, \dots, R$, we perform the following tests to reliably identify the singleton bins and obtain the correct band-sample pair for any singleton:

- **Zeroton Test**: since there is no noise, it is clear that the bin is a zeroton if $\|\mathbf{y}_r[n]\|^2 = 0$.
- **Multiton Test**: The measurement bin is a multiton as long as $|y_{c,r}[n]| \neq |y_r[n]|$ and/or $\angle\left(y_{c,r}[n]/y_r[n]\right) \neq 0 \mod 2\pi/N$. The multiton test fails when the relative phase is a multiple of $2\pi/N$;
- **Singleton Test**: After the zeroton and multiton tests, if $|y_{c,r}[n]| = |y_r[n]|$ and

  $$\frac{y_{c,r}[n]}{y_r[n]} = e^{\jmath\frac{2\pi\ell}{N}}, \quad \text{for some } \ell \in [N], \tag{2.24}$$

  the measurement bin is detected as a singleton with the band-sample pair:

  $$\widehat{k}_r[n] = \frac{N}{2\pi}\angle\left(\frac{y_{c,r}[n]}{y_r[n]}\right), \quad \widehat{x}[\widehat{k}_r[n]] = y_r[n]. \tag{2.25}$$

  This gives us the band-sample pair of the singleton for peeling.

By performing these tests on all the outputs of the sampling channels, the baseband samples from each band can be reconstructed via peeling in the same manner as the oracle-based reconstruction scheme.

2.4.4 Implementation of the Filters with Modulators

The filter banks described above can be implemented as follows. Let $p_i(t) \in \mathbb{R}$, $i = 1, \ldots, M$ denote periodic signals with period $T_p = N/f_{\max} = 1/B$, that is,

$$p_i(t) = p_i(t + T_p).$$

Let $h(t)$ be an ideal one-sided lowpass filter, with frequency response

$$h(t) \overset{\mathcal{F}}{\longleftrightarrow} H(f) = \begin{cases} 1, & \text{if } 0 \leq f < B, \\ 0, & \text{otherwise.} \end{cases}$$

We define signals $y_i(t)$ for $i = 1, \ldots, M$, as

$$y_i(t) := (x(t) \times p_i(t)) * h(t). \tag{2.26}$$

Note that because of the convolution with $h(t)$, $y_i(t)$ is bandlimited to $[0, B)$ and can be recovered from its samples taken at rate B,

$$y_i[n] = y_i(n/B) = y_i(nT_p).$$

Because $p_i(t)$ is periodic with $T_p = 1/B$, it can be written as a Fourier series

$$p_i(t) = \sum_{k=-\infty}^{\infty} c_{i,k} e^{-j2\pi Bkt}, \tag{2.27}$$

where $c_{i,k}$ are Fourier series coefficients of the signal $p_i(t)$,

$$c_{i,k} = \frac{1}{T_p} \int_{t=0}^{T_p} p_i(t) e^{j2\pi Bkt}. \tag{2.28}$$

Using the Fourier series expansion for $p_i(t)$, we get

$$x(t) \times p_i(t) \overset{\mathcal{F}}{\longleftrightarrow} \sum_{k=-\infty}^{\infty} c_{i,k} X(f + kB).$$

Hence, the spectrum of $y_i(t)$ satisfies

$$Y_i(f) = \begin{cases} \sum_{k=-\infty}^{\infty} c_{i,k} X(f + kB), & \text{if } 0 \leq f < B, \\ 0, & \text{otherwise.} \end{cases}$$

Furthermore, because $X(f)$ is zero outside $[0, f_{\max})$, we can have contribution only for $k = 0, \ldots, N-1$. Hence, we have

$$Y_i(f) = \begin{cases} \sum_{k=0}^{N-1} c_{i,k} X(f + kB), & \text{if } 0 \leq f < B, \\ 0, & \text{otherwise.} \end{cases}$$

To simplify the notation, define

$$X_k(f) = \begin{cases} X(f+kB), & \text{if } 0 \le f \le B, \\ 0, & \text{otherwise.} \end{cases}$$

Hence, we can write

$$Y_i(f) = \sum_{k=0}^{N-1} c_{i,k} X_k(f), \tag{2.29}$$

which yields the same relation as (2.13).

2.5 Spectrum-Blind Sampling of Discrete-Time Signals

Here we consider the case for discrete-time complex signals $x[n] \in \mathbb{C}$. Assume that the discrete-time Fourier transform (DTFT) of $x[n]$

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}.$$

is supported on a union of a finite number of intervals. Note that the DTFT is periodic in ω with period $= 2\pi$, hence we can limit our discussion to region $[0, 2\pi)$.

Similar in flavor to the continuous-time case, let us divide the frequency range $[0, 2\pi)$ into $N \in \mathbb{Z}^{++}$ intervals of width $B = 2\pi/N$. We denote the baseband signals of interest

$$X_k(e^{j\omega}) := \begin{cases} X\left(e^{j(\omega+kB)}\right), & \text{if } 2\pi\ell \le \omega \le 2\pi\ell + B \text{ for some } \ell \in \mathbb{Z}, \\ 0, & \text{otherwise.} \end{cases}$$

In the discrete-time setting, this corresponds to

$$x_k[n] = (x[n] e^{-jkBn}) * h[n],$$

where $h[n]$ is a low-pass filter with frequency spectrum

$$H(e^{j\omega}) = \begin{cases} 1, & \text{if } 0 \le \omega \le B, \\ 0, & \text{otherwise.} \end{cases}$$

Like in the continuous-time case, say we take measurements of the signal x by first modulating and then filtering as

$$y_i[n] = (x[n] \times p_i[n]) * h[n],$$

where the modulating signal is given by

$$p_i[n] = \sum_{k=0}^{N-1} c_{i,k} e^{-jkBn}.$$

Then, the discrete-time Fourier transform of the signal y satisfies

$$Y_i(e^{j\omega}) = \sum_{k=0}^{N-1} c_{i,k} X_k(e^{j\omega}).$$

Now, since each $y_i[n]$ is bandlimited to $[0, 2\pi/N)$ we can downsample each $y_i[n]$ by N without aliasing. When the support of the frequency spectrum of the input signal has Lebesgue measure f_L, the discussions about the sparsity of the measurements in the previous section hold true. Hence, we state the following theorem.

Theorem 3. *Any discrete-time signal $x[n]$ whose frequency support is a union of finite number of intervals with total Lebesgue measure f_L, using sparse-graph code filter bank, can be downsampled to rate $2(1+\epsilon)f_L$, for any $\epsilon > 0$, with the probability of perfect reconstruction arbitrarily close to one. Furthermore, the time averaged computational complexity of the recovery algorithm is $O(f_L)$.*

Proof. Follows the same steps as the continuous time case with band limits replaced by $[0, 2\pi)$. □

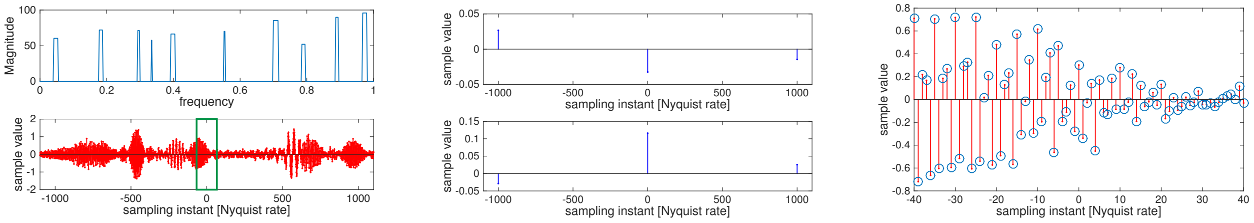
2.6 Empirical Validation

In this section we present numerical experiments validating the performance of our *spectrum-blind sampling and reconstruction* framework based on sparse-graph-coded filter banks.

For the first experiment, we generate a random signal from $\text{BandLimited}(1, 0.1)$ having a piecewise constant spectrum on 10 disjoint frequency bands occupying a total of $1/10$ of the interval $[0, 1)$. The frequency spectrum and the Nyquist rate samples of the input signal is shown in Figure 2.8a. We choose number disjoint frequency slices $N = 1000$ and use $M = 284$ channels designed through the regular graph ensemble. The sampling rate at the output of each filter is $1/N$. Samples from a choice of two channels of the filter bank are shown in Figure 2.8b. The aggregate sampling rate from this setting is equal to $f_s = f_{\max} M/N = 0.284$.

Figure 2.8c shows the Nyquist samples from the original signal together with the reconstruction from samples obtained by the sparse-graph-coded filter bank. As can be seen, Nyquist rate samples, hence the whole time domain signal, can be recovered *spectrum-blind* from samples taken at rate 0.284. Our result in Theorem 1 implies a rate of 0.246 for this signal model. The difference is due to finite slicing of the spectrum; as we increase N the empirical aggregate sampling rate f_s approaches the theoretical value.

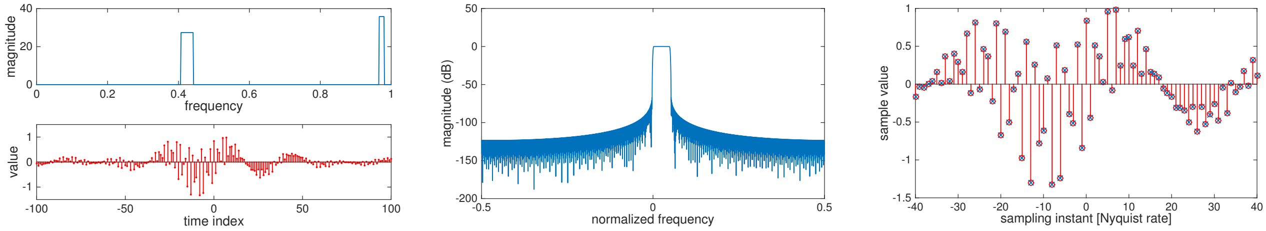
The second simulation shows spectrum-blind sampling and reconstruction of discrete time signals using finite length practical filters. The input signal has components in two frequency bands occupying a total of 5% of the spectrum as shown in Figure 2.9a. For frequency domain slicing,



(a) (b) (c)

Figure 2.8: Simulation with ideal filters. (a) Input signal's magnitude spectrum (top) and the real part of Nyquist rate samples (bottom). (b) Output samples of two channels of the sparse-graph-coded filter bank; note that the filter bank outputs are sampled at rate 10^{-3}. (c) Nyquist rate samples corresponding the region marked by the green box in (a); red lines correspond to the Nyquist rate samples of the input signal, and the blue circles correspond to the reconstruction using our proposed sampling framework.

we choose $N = 20$ and use a low-pass filter of length 801 whose frequency response is shown in Figure 2.9b. As can be seen in Figure 2.9c, the input signal can be reconstructed from the samples obtained using the proposed framework. The aggregate sampling rate achieved with this simulation setting is $f_s = 0.4$.



(a) (b) (c)

Figure 2.9: Simulation with non-ideal filters. (a) Input signal's magnitude spectrum (top) and the real part of the signal (bottom). (b) Frequency response of the low pass filter of the sparse-graph-coded filter bank. (c) Reconstruction overlaid on top of the input signal; red lines correspond to the input signal, and the blue circles correspond to the reconstruction using our proposed sampling framework.

2.7 Conclusions and Future Directions

In this chapter, we studied the problem of *spectrum-blind* sampling of continuous-time signals with sparse frequency spectra. The minimum sampling rate for this class of signals has been established as twice the measure of their frequency support; however, constructive schemes that achieve this minimum rate were not known. We proposed a sampling framework based on ideas from *modern coding theory* that approaches this minimum rate. Furthermore, we made interesting connections between the problem of spectrum-blind sampling, and that of designing erasure-correcting codes based on sparse-graph codes. We showed that the signal reconstruction under our sampling scheme is equivalent to the peeling decoding of *sparse-graph codes* for reliable transmission on an erasure channel, and that the achievable *sampling rate* is determined by the *rate* of the sparse-graph code used in the filter bank. As a result, our method *simultaneously* approaches the minimum sampling rate for spectrum-blind sampling and has low computational complexity based on fast peeling-based decoding with operations per unit of time scaling *linearly* with the sampling rate.

Our framework has potential applications in communication settings where a receiver needs to monitor transmissions at unknown locations over a large frequency band. For example, in frequency-hopping spread spectrum method, communication signals are transmitted while rapidly changing the carrier frequency over a large spectral band [29]. If the frequency-hopping pattern is unknown, receiver of the signal needs to sample at the Nyquist rate corresponding to the whole frequency range, which can be prohibitive large in practice. However, because transmitter hops between carrier frequencies, the signal occupies a sparse spectrum at any given time window. Our method can be used to design receiver architectures that can recover the signal using practical sampling rates without knowing the frequency-hopping pattern. Another application area is cognitive radio, where secondary users can communicate on unoccupied bands when primary users of these bands are inactive. Our framework can be used to detect unoccupied bands over a large range without requiring prohibitively large sampling rates. Tailoring our method for these applications are future research directions.

Appendix

2.A Proofs

Proof of Theorem 1. Choose $N \in \mathbb{Z}^{++}$ and divide the frequency interval $[0, f_{\max})$ to N bins of equal width, $B = f_{\max}/N$. The signal is assumed to have a finite number of disjoint bands, say that it has K_0. Let K denote the number of bins that overlap with a frequency support, K satisfies

$$\frac{f_L}{B} \leq K \leq \frac{f_L}{B} + K_0. \tag{2.30}$$

Because the vector $\mathbf{x}(t)$ is made up of the signal components at the bands, the sparsity of $\mathbf{x}(t)$ is also upper bounded by $f_L/B + K_0$. Choose

$$R = 1.23\left(\frac{f_L}{B} + K_0\right).$$

From Theorem 11 in Appendix A, it follows that the oracle-based peeling reconstruction succeeds with probability

$$\mathbb{P}_s \geq 1 - O\left(\frac{1}{K}\right) \geq 1 - O\left(\frac{B}{f_L}\right) = 1 - O\left(\frac{f_{\max}}{f_L} \cdot \frac{1}{N}\right). \tag{2.31}$$

Hence, as the slicing in the spectral domain becomes finer with increasing N, the probability of success approaches 1 asymptotically. We sample at rate B at each channel, so the resulting aggregate sampling rate is

$$\begin{aligned} f_s = RB &= 1.23\left(\frac{f_L}{B}+K_0\right)B \\ &= 1.23 f_L + 1.23\frac{K_0}{N}. \end{aligned} \tag{2.32}$$

Hence, as the number of frequency slices N increases, the sampling rate approaches $1.23f_L$. Furthermore, this scheme requires $O(K)$ arithmetic operations at rate B because there are $O(K)$ edges in the graph that needs to be processed, each is processed in constant time, and this processing needs to be done at rate B. As $K = O(f_L/B)$, this results in $O(f_L)$ operations per unit time. □

Proof of Theorem 2. Choose $N \in \mathbb{Z}^{++}$ and divide the frequency interval $[0, f_{\max})$ to N bins of equal width, $B = f_{\max}/N$. By assumption of finite union of intervals, there exists a lower bound $B_0 > 0$ for the smallest support interval, and the number of disjoint bands in the signal is upper bounded by $\lfloor f_L/B_0 \rfloor$. The number of bins that overlap with a frequency support k satisfies

$$\frac{f_L}{B} \leq \left\lceil \frac{f_L}{B} \right\rceil \leq K \leq \left\lceil \frac{B_0}{B} \right\rceil \left\lfloor \frac{f_L}{B_0} \right\rfloor \leq \frac{f_L}{B} + \frac{f_L}{B_0}, \tag{2.33}$$

which means that the sparsity of $\mathbf{x}(t)$ is also bounded the same way. Choose $\epsilon' > 0$ and denote $R = (1+\epsilon')(f_L/B + f_L/B_0)$. From Theorem 5 in [96], the oracle-based peeling reconstruction succeeds with probability

$$\mathbb{P}_s \geq 1 - O\left(\frac{1}{K}\right) \geq 1 - O\left(\frac{B}{f_L}\right) = 1 - O\left(\frac{f_{\max}}{f_L}\cdot\frac{1}{N}\right). \tag{2.34}$$

Hence, as the spectral slicing becomes finer as N increases, the probability of success approaches 1. We sample at rate B at each channel, so the resulting sampling rate is

$$\begin{aligned} f_s = RB &= (1+\epsilon')\left(\frac{f_L}{B}+\frac{f_L}{B_0}\right)B \\ &= (1+\epsilon')f_L + (1+\epsilon')\frac{f_L}{B_0}\frac{f_{\max}}{N}. \end{aligned}$$

Given an arbitrarily small $\epsilon > 0$, the sampling rate f_s can be made smaller than $(1+\epsilon)f_L$ by appropriate choices of $\epsilon'$and N (i.e., B), while satisfying any given success probability. This scheme requires only $O(f_L/B)$ arithmetic operations at rate B, which results in f_L operations per unit time. □

2.B Special Case of Real-Value Signals

Assume now that signal is real-valued, that is, $x(t) \in \mathbb{R}$ for all t. Note that this implies that the spectrum of x is conjugate symmetric, that is, $X(f) = X^*(-f)$. Hence, instead of a one-sided spectrum, we now assume that the signal has a two-sided spectrum bandlimited to $f_{\max}/2$

on the positive and the negative side, i.e., $X(f) = 0$ for $|f| > f_{\max}/2$. For partitioning the frequency domain, we choose $L_0 \in \mathbb{Z}^{++}$ and divide the frequency interval $(-f_{\max}/2, f_{\max}/2)$ into $N = 2L_0 + 1$ equal bands. Furthermore, we limit the choice of the periodic modulators to be real-valued, that is, $p_i(t) \in \mathbb{R}$ for all t. Because $p_i(t) \in \mathbb{R}$, it follows from (2.28) that we have

$$c_{i,k} = c^*_{i,-k}. \tag{2.35}$$

We define signals $y_i(t)$ for $i = 1, \ldots, M$ as in (2.29); because $X(f) = X^*(-f)$, we have $X(f - kB) = X^*(-f - (-k)B)$, hence in (2.29) we have

$$X_k(f) = X^*_{-k}(-f). \tag{2.36}$$

Writing (2.29) in time domain, and using (2.35) and (2.36) we have

$$y_i(t) = c_{i,0}x_0(t) + 2\sum_{k=1}^{L_0} \Re(c_{i,k}x_k(t)),$$

where $\Re(\cdot)$ denotes the real part of its argument. In vector form, measurements are

$$y_i(t) = \begin{pmatrix} c^*_{i,L_0} & \cdots & c_{i,L_0} \end{pmatrix} \begin{pmatrix} x^*_{L_0}(t) \\ \vdots \\ x_{L_0}(t) \end{pmatrix}.$$

The multiple measurements through the use of M different $p_i(t)$ can be represented for each time t by the linear equation

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t), \tag{2.37}$$

where $\mathbf{C} \in \mathbb{C}^{M \times 2L_0+1}$. Note that, as B asymptotically approaches 0, for each time instant t, the sparsity of $\mathbf{x}(t)$ is going to approach Lebesgue measure of the frequency support of $x(t)$ divided by B.

2.B.1 Constructing Sparse-Graph Codes for Real-Valued Signals

Because we study the case where the periodic modulators $p_i(t)$ are real in this section, we have constraints in the design of the matrix $\mathbf{C}$, namely it is required that $c_{i,k} = c^*_{i,-k}$. Although this seems to constrain the design of $\mathbf{C}$, because the signal support comes in conjugate pairs, we actually only need to design the sparse-graph code for the positive part of the spectrum.

As for the design of support detection tensor, because the signal support comes in conjugate pairs, if $c_{i,k}x_k(t) \neq 0$ for $k > 0$, we need to recover two entries of $\mathbf{x}(t)$. Hence, we need a new support detection algorithm tailored for this setting. By leveraging the conjugate pair nature of the support, we will be able to design a support detection tensor that is similar to the Fourier-friendly bin detection algorithm proposed by the authors of [52].

In designing the support detector, we are going to use the property of Fourier series expansion that states shifts in time corresponds to multiplication by complex exponentials of Fourier coefficients. More formally, we have

$$p_i\left(t + l\frac{T_p}{2L_0+1}\right) = \sum_{k=-L_0}^{L_0}\left[c_{i,k}\exp\left(j\frac{2\pi}{2L_0+1}lk\right)\right]\exp(j2\pi Bkt). \tag{2.38}$$

To simplify the notation, we define $\omega_0 := \frac{2\pi}{2L_0+1}$. The rest of the text is going to talk about how we can recover the support and the signal from measurements with a total of three delays in each $p_i(t)$.

Say we have the measurement

$$y_{i,l}(t) = \left(x(t) \times p_i\left(t + l\frac{T_p}{2L_0+1}\right)\right) * h(t), \tag{2.39}$$

combining (2.38) and (2.39), and substituting in ω_0, we get

$$y_{i,l}(t) = \sum_{k=-L_0}^{L_0} c_{i,k}e^{j\omega_0 lk}x_k(t).$$

Now, say we obtain three measurements $y_{i,0}$, $y_{i,1}$ and $y_{i,2}$, and there is only one $k > 0$ such that $c_{i,k}x_k(t) \neq 0$ (the case of $k = 0$ has a simpler relation that is explained after the discussion of $k > 0$). Then we have

$$\begin{aligned} y_{i,0}(t) &= c_{i,k}^* x_k^*(t) + c_{i,k}x_k(t), \\ y_{i,1}(t) &= e^{-j\omega_0 k}c_{i,k}^* x_k^*(t) + e^{j\omega_0 k}c_{i,k}x_k(t), \\ y_{i,2}(t) &= e^{-j2\omega_0 k}c_{i,k}^* x_k^*(t) + e^{j2\omega_0 k}c_{i,k}x_k(t). \end{aligned}$$

Defining

$$re^{j\theta} := 2c_{i,k}x_k(t), \tag{2.40}$$

and dropping the time index t and subscript i for brevity, we can write

$$\begin{aligned} y_0 &= r\cos(\theta), \\ y_1 &= r\cos(\theta + \omega_0 k), \\ y_2 &= r\cos(\theta + 2\omega_0 k). \end{aligned}$$

To find the support, we need to recover k, and to find the signal we need to recover r and θ. We have, three equations and three unknowns, and show how to solve this set of equations. Define

$$\begin{aligned} a &:= \cos(\omega_0 k), \\ b &:= \sin(\omega_0 k). \end{aligned}$$

By trigonometric identities, we can write

$$\begin{aligned}
y_0 &= r\cos(\theta), \\
y_1 &= r\cos(\theta)\cos(\omega_0 k) - r\sin(\theta)\sin(\omega_0 k) \\
&= a y_0 - b r\sin(\theta), \\
y_2 &= r\cos(\theta)\cos(2\omega_0 k) - r\sin(\theta)\sin(2\omega_0 k) \\
&= y_0(2a^2-1) - 2abr\sin(\theta),
\end{aligned}$$

which gives

$$a = \frac{y_0 + y_2}{2y_1}.$$

Hence, we can find

$$k = \cos^{-1}(a)/\omega_0. \tag{2.41}$$

If the resulting $k \in \mathbb{Z}^+$ then we have found a singleton/doubleton. It then follows that $b = \sin(k\omega_0)$. From the set of equations we also have

$$r\sin(\theta) = \frac{a y_0 - y_1}{b},$$

which gives

$$\theta = \tan^{-1}(r\sin(\theta), y_0),$$

where $\tan^{-1}(u,v)$ is the four quadrant inverse tangent function satisfying $r\sin(\tan^{-1}(u,v)) = u$ and $r\cos(\tan^{-1}(u,v)) = v$. We can also find

$$r = y_0/\cos(\theta).$$

Then, from these equations, for $k > 0$, it follows that

$$\begin{aligned}
x_{i,k}(t) &= \frac{re^{j\theta}}{2c_{i,k}}, \\
x_{i,-k}(t) &= \frac{re^{-j\theta}}{2c_{i,k}^*}.
\end{aligned}$$

Now we treat the special case of $k = 0$. In that case, in (2.41) we get $k = 0$, and $\theta = 0$ since $z_{i,0}(t)$ is real. Hence, we have

$$x_{i,0}(t) = y_1(t).$$

The pseudocode for support and signal detection is given in Algorithm 1.

Note that once the support of the vector $\mathbf{x}(t)$ is recovered, the extra channel for support detection can be turned. Hence, we state the following theorem.

Algorithm 1: Support and signal detection for real valued signals

```
Input  : ${y_i}_{i=0,1,2}$, ${c_k}_{k=0,...,L_0}$, $\omega_0$
Output : Support $k$ and amplitude $x$ if singleton, otherwise declare not a singleton
$a \leftarrow (y_0 + y_2)/(2y_1)$
$k' \leftarrow \cos^{-1}(a)/\omega_0$
if $k' = 0$ then
    $k \leftarrow k'$
    $x \leftarrow y_0$
else if $k' \in \mathbb{Z}^{++}$ then
    $k \leftarrow k'$
    $\theta \leftarrow \tan^{-1}\left((ay_0 - y_1)/\sin(k\omega_0), y_0\right)$
    $r \leftarrow y_0/\cos(\theta)$
    $x \leftarrow re^{j\theta}/(2c_k)$
else
    declare not a singleton
end
```

Theorem 4. *Any continuous-time real valued signal $x(t)$ that is bandlimited within a known frequency interval, and whose frequency support is a union of intervals with total Lebesgue measure f_L, using sparse-graph code filter bank, can be sampled at rate $1.5(1+\epsilon)f_L$ while the probability of perfect reconstruction can be made arbitrarily close to one. Furthermore, the coded filter bank can be implemented with real signals, and time averaged computational complexity of the recovery algorithm is $O(f_L)$.*

Proof. The proof follows similar steps to Theorem 1. Hence, we are going to highlight the main difference. Note that the sparse-graph code remains the same for one sided spectrum. Assume, for simplifying the proof, that the signal does not have support on an interval near the zero-frequency. The Lebesgue measure of the support on positive frequencies is $f_L/2$ since the frequency components come in complex conjugate pairs. Then we have

$$\frac{f_L}{2B} \leq k \leq \frac{f_L}{2B} + \frac{f_L}{2B_0}.$$

Hence, in the limit as B tends to 0, we have sampling rate for the 'oracle', i.e., $(1+\epsilon)kB$, go to $(1+\epsilon)f_L/2$. We can replace 'oracle' by using 3 channels at each multiband filter. Hence, the sampling rate can be made $1.5(1+\epsilon)f_L$. □

Chapter 3

Sparse Discrete Fourier Transform: Enabling Low SNR Regimes

3.1 Introduction

In Chapter 2 we focused on sampling continuous-time signals and presented extensions for the discrete-time case. In both of these settings, the signal of interest was of infinite length. In this chapter, we focus our attention to finite-length discrete-time signals which have widespread practical applications as they lend themselves to analysis on computers. In this chapter, we focus on the Discrete Fourier Transform (DFT) which has been critically important in science and engineering, and is used in applications related to audio, video, and communications to name a few.

The Fast Fourier Transform (FFT) is a revolutionary algorithm to compute the DFT of an arbitrary N-length signal using $O(N \log N)$ operations [14]. It is widely accepted that this scaling of the computational complexity is necessary for obtaining the DFT of arbitrary signals [66]. However, in many applications of interest, such as audio, video, and spectroscopy to name a few, the DFT has a sparse structure, which can be used to cut down sample and computational complexity over that of FFT's. Recently, the authors [70, 69] proposed the algorithms Fast Fourier Aliasing-Based Sparse Transform (FFAST) and Robust-FFAST (R-FFAST) for computing an N-length K-sparse DFT in sample and computational complexity that scale sublinearly in the ambient dimension N whenever the sparsity K also scales sublinearly in N. At a high level, these algorithms work by first subsampling the signal judiciously and then performing smaller-sized FFTs using the obtained samples. The subsampling reduces the sample complexity, and by performing FFTs on the subsampled streams, the computational complexity is also reduced. Through a careful design of the subsampling patterns, the recovery of K-sparse DFT is transformed into recovering K-many *single frequencies* and combining their results to get the DFT of the input signal. Besides having strong theoretical guarantees, these algorithms also work well in practice, and they are implemented in software[1] and hardware [94, 93]. However, one of the limitations of these algorithms is that they

This chapter is based on joint work with Raaz Dwivedi.

[1] Repository can be found on Github at `https://github.com/UCBASiCS`

require a sufficiently-high signal to noise ratio (SNR) in order to attain a sublinear computational complexity [69]. This can be a bottleneck in applications requiring operating in low SNR, such as medical imaging, radio astronomy, and wideband spectrum sensing

In this chapter, we propose a method to improve the SNR range of operation of R-FFAST [69]. Specifically, we propose a sample and computationally efficient method for single frequency identification from noisy measurements. This is a subproblem in R-FFAST where the recovery of a K-sparse DFT (signal is a superposition of K frequencies) is divided to recovering K-many 1-sparse DFTs (signal is a single frequency). By providing stronger guarantees for the low SNR regime, our method enhances the robustness guarantees of R-FFAST. This is a step towards making the algorithm more impactful in low SNR applications.

The organization of this chapter is as follows. In Section 3.2 we cover the necessary background on FFAST and R-FFAST. In Section 3.4 we describe the details of our novel method for single tone identification, and give guarantees on its sample and computational complexity in Section 3.5. We then present numerical experiments in Section 3.6, comparing our method with a baseline of the successive estimation algorithm based on Kay's method [38] which is the method proposed in the original R-FFAST paper [69], and conclude with Section 3.7.

3.2 Background on Aliasing-Based Sparse DFT

Let $x \in \mathbb{C}^N$ be an N-length signal. Any such x can be represented as a sum of N complex exponentials

$$x[n] = \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N}kn}, \tag{3.1}$$

where the amplitudes $X[k] \in \mathbb{C}$ for $k \in \{0, \ldots, N-1\}$. These amplitudes are called the DFT coefficients of X, and they can be calculated as

$$X[k] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn}. \tag{3.2}$$

The operation in equation (3.2) is called the DFT, and the one in equation (3.1) is called the inverse DFT[2].

We are going to give a brief introduction to FFAST and R-FFAST through a running example. For the example, assume that the signal length is $N = 3 \times 4 = 12$, and let $x \in \mathbb{C}^{12}$ such that

$$x[n] = e^{j\frac{2\pi}{12}n} + e^{j\frac{2\pi}{12}4n} + e^{j\frac{2\pi}{12}5n}, \tag{3.3}$$

that is, x is made up of a superposition of 3 tones at locations 1, 4, and 5 over a grid of size 12. Note that because x is made up of a superposition of 3 tones, the DFT of x is $K = 3$ sparse. Let $X \in \mathbb{C}^{12}$ be the DFT of x; then, X is equal to 0 except for $X[1] = 1$, $X[4] = 1$, $X[5] = 1$.

[2] Note that the scaling of $1/N$ is usually included in the inverse DFT equation. We included the scaling in the DFT equation because it helps us simplify the notation in this chapter.

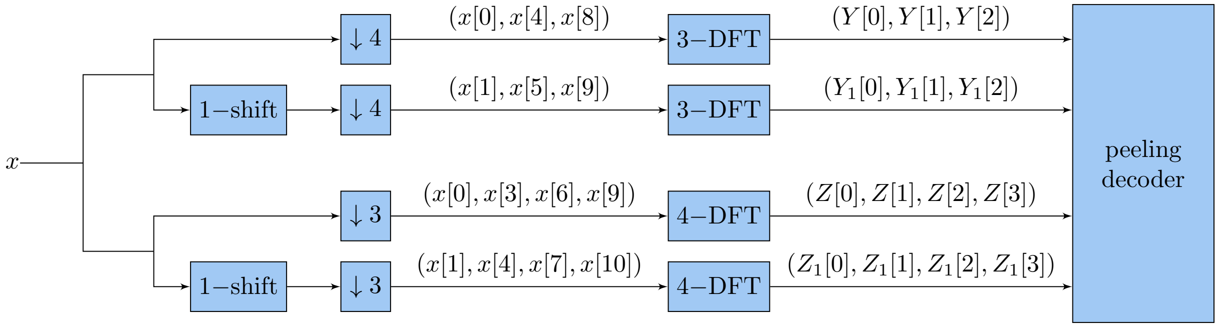

x
1−shift
1−shift
↓ 4
↓ 4
↓ 3
↓ 3
$(x[0], x[4], x[8])$
$(x[1], x[5], x[9])$
$(x[0], x[3], x[6], x[9])$
$(x[1], x[4], x[7], x[10])$
3−DFT
3−DFT
4−DFT
4−DFT
$(Y[0], Y[1], Y[2])$
$(Y_1[0], Y_1[1], Y_1[2])$
$(Z[0], Z[1], Z[2], Z[3])$
$(Z_1[0], Z_1[1], Z_1[2], Z_1[3])$
peeling decoder

Figure 3.1: Architecture of FFAST. First, the signal is shifted and subsampled at coprime factors. Smaller sized DFT (compared to the original signal length) is then performed on each subsampling output. The outputs of these DFTs are then fed into the peeling decoder. This peeling decoder then outputs the DFT of the original signal x. The example here is given for a signal length of $3 \times 4 = 12$.

As shown in Figure 3.1, FFAST works by first subsampling the signal x by coprime subsampling factors. Since the signal length is $N = 3 \times 4$, the coprime factors that evenly divide N are 3 and 4, and FFAST subsamples the input by these two factors. The subsampling operations are repeated after circularly shifting x by 1. These shifts are going to be used at the *check nodes* for location identification (as will be described below).

We will make use of two properties of DFT. First one states that circularly shifting the input signal equals to modulation in the DFT domain, that is,

$$x[(n+r)_N] \xleftrightarrow{\text{DFT}} X[k]e^{j\frac{2\pi}{N}kr}, \tag{3.4}$$

where $(n+r)_N := n + r \mod N$. The second property is the aliasing property, which states that subsampling the input signal creates linear mixing of the DFT coefficients. In particular, for a signal $x \in \mathbb{C}^N$ of length N, where N is a product of two factors $N = N_1 N_2$, let $y[n] = x[N_1 n]$ be the signal obtained by subsampling x with at rate N_1. Then, the DFT Y of y satisfies

$$y[n] = x[N_1 n] \xleftrightarrow{\text{DFT}} Y[k] = \sum_{\ell=0}^{N_1-1} X[k + N_2\ell]. \tag{3.5}$$

Combining these two properties, for the streams given in Figure 3.1, we get:

$$\begin{array}{ll} Y[0] = 0, & Y_1[0] = 0, \\ Y[1] = X[1] + X[4], & Y_1[1] = X[1]e^{j\frac{2\pi}{12}} + X[4]e^{j\frac{2\pi}{12}4} \\ Y[2] = X[5], & Y_1[2] = X[5]e^{j\frac{2\pi}{12}5}, \end{array} \tag{3.6}$$

and

$$\begin{array}{ll} Z[0] = X[4], & Z_1[0] = X[4]e^{j\frac{2\pi}{12}4}, \\ Z[1] = X[1] + X[5], & Z_1[1] = X[1]e^{j\frac{2\pi}{12}} + X[5]e^{j\frac{2\pi}{12}5} \\ Z[2] = 0, & Z_1[2] = 0, \\ Z[3] = 0, & Z_1[3] = 0. \end{array} \tag{3.7}$$

The vectors Y, Y_1, Z, and Z_1 are then laid out on a graph as shown in Figure 3.2 to make up the *variable nodes* and *check nodes*. As can be seen in the figure, each index of (Y, Y_1) pair and (Z, Z_1) pair make up one check node. The result is a sparse-graph where each variable node (left node) is connected to 2 check nodes (right node).

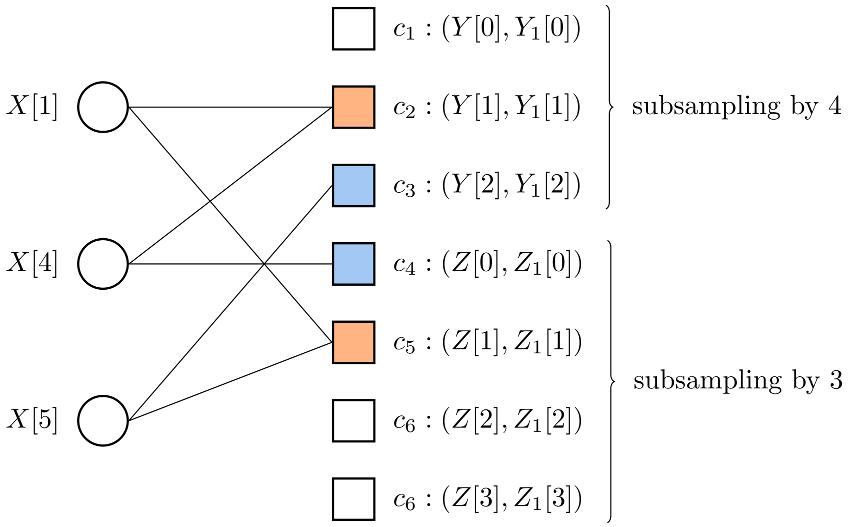


Figure 3.2: Aliasing of the DFT coefficients induces a sparse-graph code structure in FFAST. The figure depicts the graph for a signal with three non-zero DFT coefficients: $X[1]$, $X[4]$, and $X[5]$. The variable nodes (left nodes) correspond to the non-zero DFT coefficients of the signal, and the check nodes (right nodes) correspond to the grouping of equations in (3.6) and (3.7). An edge is drawn between a variable node and a check node if the DFT coefficient corresponding to that variable node appears in the equations corresponding to that check node.

In order for the peeling decoder introduced in Section 1.1 to work, we need a mechanism that identifies singleton check nodes, and recovers the location and the value of the variable node connected to that check node for peeling. This mechanism can be achieved as follows; if $|Y[i]| = |Y_1[i]|$, and we have $(2\pi/N)\angle(Y_1[i]/Y[i]) = \widehat{k}$ an integer, then the check node i is a singleton, and it recovers the location at $\widehat{k}$ and value $Y[i]$. This information is than peeled off from the graph as was done in the example shown in Figure 1.4 for decoding. This concludes the coverage of FFAST in noiseless setting.

The noisy setting we consider is where the signal is

$$x[n] = e^{j\frac{2\pi}{12}n} + e^{j\frac{2\pi}{12}4n} + e^{j\frac{2\pi}{12}5n} + w[n], \tag{3.8}$$

with noise $w[n] \overset{\text{i.i.d.}}{\sim} \mathcal{N}_c(0, \sigma^2)$, which is independent and identically distributed as centrally-symmetric complex Gaussian with mean 0 and variance σ^2. In this setting, the mechanism designed for the noiseless case described above will not work as $|Y[i]| = |Y_1[i]|$ and $(2\pi/N)\angle(Y_1[i]/Y[i]) = \widehat{k}$ being an integer both happen with zero probability. However, we can design a method that utilizes more shifts applied to the input signal as shown in Figure 3.3. As can be seen from the relation in (3.4), a circular shift of r corresponds to multiplication of the DFT coefficient at location k by a complex exponential $e^{j\frac{2\pi}{N}k}$. Using a shift sequence of $(r_1, r_2, \ldots, r_D)$, we have the following set of relations:

$$\begin{aligned} x[(n+r_1)_N] &\xleftrightarrow{\text{DFT}} X[k]e^{j\frac{2\pi}{N}kr_1}, \\ x[(n+r_2)_N] &\xleftrightarrow{\text{DFT}} X[k]e^{j\frac{2\pi}{N}kr_2}, \\ &\vdots \\ x[(n+r_D)_N] &\xleftrightarrow{\text{DFT}} X[k]e^{j\frac{2\pi}{N}kr_D}. \end{aligned} \tag{3.9}$$

Looking at the right-hand side of these relations, we have a list of samples equal to

$$\left(X[k]e^{j\frac{2\pi}{N}kr_1}, X[k]e^{j\frac{2\pi}{N}kr_2}, \ldots, X[k]e^{j\frac{2\pi}{N}kr_D}\right).$$

Note that this is equal to sampling a single tone $\tilde{x}[n] = X[k]e^{j\frac{2\pi}{N}kn}$ at the sampling points n corresponding to the shifts $\{r_1, r_2, \ldots, r_D\}$. Note that the frequency location is equal to k, and the amplitude is equal to $X[k]$. Hence, identifying the frequency and amplitude recovers the location k and amplitude $X[k]$ respectively to be used in the peeling decoder.

This puts single frequency identification from its noisy measurements at the core of the peeling process. One method to do single frequency identification is through using the maximum likelihood method. However, this requires going over all possible frequencies, and calculating the likelihood for each. Hence, the computational complexity of that algorithm scales linearly with the ambient dimension of the problem. In R-FFAST [69], the authors propose a method that uses Kay's tone identification algorithm [38] for building a computationally efficient algorithm which has a computational complexity that scales as $O(\log^{4/3}(N))$. However, for this to hold, it requires a sufficiently high signal to noise ratio (SNR). This limits the applicability of R-FFAST to problems that has low SNR.

In the rest of the this chapter, we introduce a novel method, dubbed maximum-likelihood-based successive digit recovery (MSDR), for identifying a single tone whose frequency lies on a regular grid, from a set of noisy measurements. For any targeted error probability, we establish that the sample and computational complexities of our algorithm scale logarithmically with the grid size. Moreover, our guarantees apply for *any* given SNR. Compared to the maximum likelihood method, our method has an exponentially better dependence on the grid size for computational complexity, at the expense of a worse sample complexity by a doubly-logarithmic factor. Our proposed method is a robust building block for many applications involving the sparse DFT, e.g., MRI, radio astronomy, optical imaging, wideband sparse spectrum sensing, via R-FFAST algorithm, whose robustness guarantees get significantly enhanced.

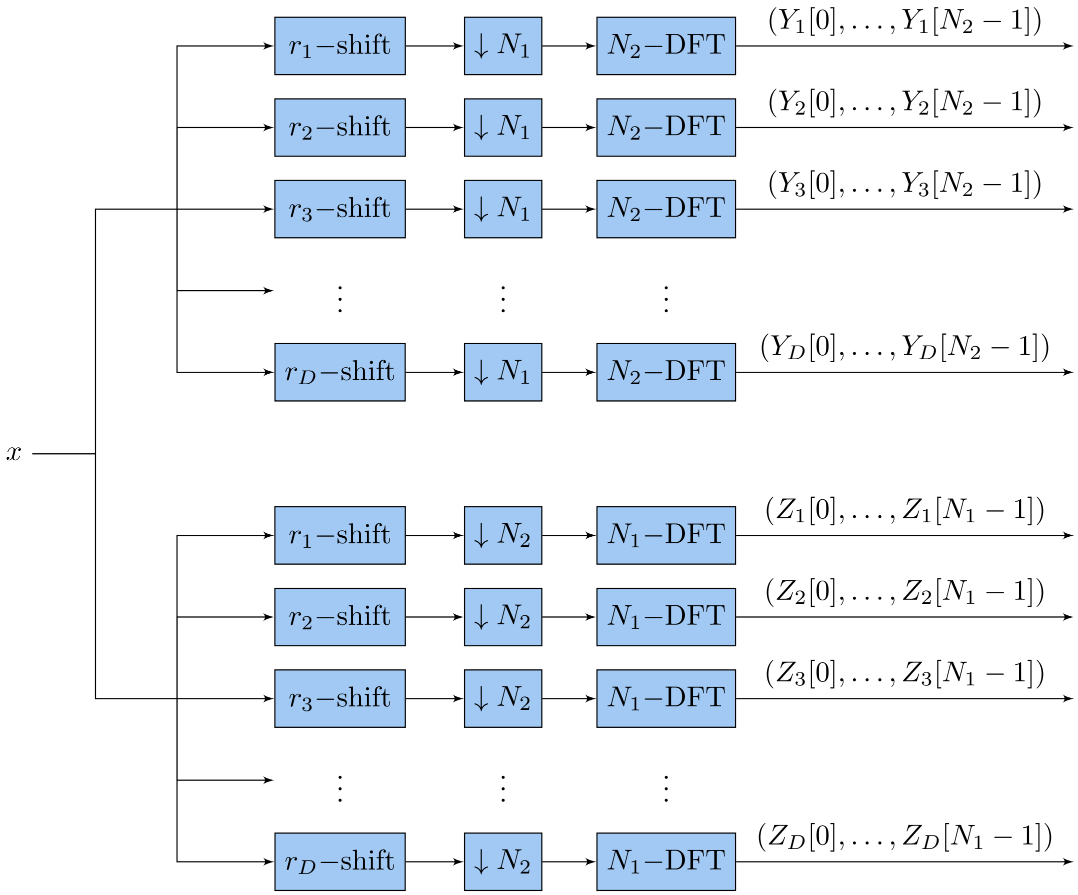

x
r_1-shift
r_2-shift
r_3-shift
r_D-shift
$\downarrow N_1$
$\downarrow N_2$
N_2-DFT
N_1-DFT
$(Y_1[0],\ldots,Y_1[N_2-1])$
$(Y_2[0],\ldots,Y_2[N_2-1])$
$(Y_3[0],\ldots,Y_3[N_2-1])$
$(Y_D[0],\ldots,Y_D[N_2-1])$
$(Z_1[0],\ldots,Z_1[N_1-1])$
$(Z_2[0],\ldots,Z_2[N_1-1])$
$(Z_3[0],\ldots,Z_3[N_1-1])$
$(Z_D[0],\ldots,Z_D[N_1-1])$

Figure 3.3: Schematic of shifts and downsampling of R-FFAST in the noisy setting. Here, D is the number of shifts, and the signal length is $N = N_1 N_2$. Compared to the noiseless case shown in Figure 3.1, more shifts are used for combating against noise. The shifts $\{r_1, r_2, \ldots, r_D\}$ induce a signal equivalent to sampling of a single tone at singleton check nodes (see equation 3.9).

3.3 Single Tone Identification

Single tone identification is an important problem in signal processing that arises in a wide variety of applications including direction of arrival estimation, radar, and sparse DFT. Prior works have extensively studied the continuous-valued frequency setting and provided several notable sample and computationally efficient methods [38, 25, 55]. These methods have a performance close to the maximum likelihood method in terms of sample complexity, and often achieve a low computational complexity when the signal-to-noise ratio (SNR) is high enough.

In this work, we consider a setting where we have partial knowledge about the frequency of the tone; in particular, the frequency lies on a regular grid known apriori, but with its precise location unknown. Such a setting arises in many modern applications of interest including sparse-DFT algorithms. One class of these algorithms work with a divide-and-conquer strategy. More concretely, in order to recover the DFT of a signal composed of a small number of tones, they divide the problem into multiple simpler problems of identifying a single tone [70, 69]. This 'divide' step is achieved via a judicious sub-sampling of the signal followed by computations of smaller-sized Fourier transforms. The simpler single-tone identification problems are then solved by using the well-known continuous frequency identification methods, along with a scheme to round to the closest grid point. Such a combined strategy does not fully leverage the discrete nature of the problem. Hence, the obtained results are often not optimal especially in the low SNR regime. One method that leverages the discrete nature is the maximum likelihood (ML) method. Given a set of noisy samples from the signal, ML method maximizes the inner product of the measurements with the signatures corresponding to all possible locations on the grid. Given a regular grid of size N and a fixed error probability, the ML method requires $O(\log(N))$ samples; however, the computational complexity scales as $O(N \log(N))$ which is superlinear [68].

We present a method that attains a near-ML performance in terms of sample complexity and achieves a logarithmic-time complexity. More precisely, our algorithm has $O(\log(N)\log(\log(N)))$ sample and computational complexity for every SNR point. By having a logarithmic-time complexity at every SNR point, our method provides a *robust* building block for many applications involving the sparse discrete Fourier transform like MRI, radio astronomy, optical imaging, wideband sparse spectrum sensing to name a few, via the recently proposed sublinear-time R-FFAST algorithm [69], whose robustness guarantees get significantly enhanced.

3.4 Proposed algorithm

We begin with a description of our method assuming access to an infinite sequence of the signal. We describe how to handle a practical setting of having access to the signal on a limited interval in Section 3.4.2.

Consider the case of $N = p^n$ where p is a prime and $n \in \mathbb{N}$. We assume access to the following signal:

$$y[m] = Ae^{j\frac{2\pi}{N}km} + w[m], \tag{3.10}$$

where k is an integer in $[0, N)$ and denotes the *true* frequency location, $w[m] \overset{\text{i.i.d.}}{\sim} \mathcal{N}_c(0, \sigma^2)$ denotes the noise with a centrally-symmetric complex Gaussian distribution with mean 0 and variance σ^2. Our goal is to determine the frequency location k in a sample and computationally efficient way.

Our method makes use of the following fact that; since $N = p^n$, we can represent k as

$$k = k_0 + k_1 p + k_2 p^2 + \cdots + k_{n-1} p^{n-1}, \tag{3.11}$$

where k_i are integers in $[0, p)$ for all $i \in \{0, \ldots, n-1\}$. Recovering k is equivalent to recovering all of its coefficients k_i (which we refer to as a digit). The main idea behind our proposed method is to recover k_i progressively by performing carefully-designed sub-sampling (see Figure 3.4 for an example).

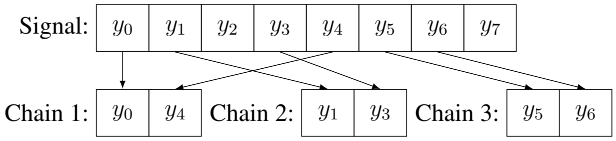


Figure 3.4: Our algorithm recovers digits k_i given in equation (3.11) of the frequency location from the least significant one to the most significant. In order to achieve this, we construct chains of samples from the signal. The example in the figure corresponds to a setting where $p = 2$, $n = 3$.

For recovering k_0, consider sub-sampling the signal with the interval between successive samples[3] equal to p^{n-1}. Noting that $N = p^n$, the resulting signal is given by

$$\begin{aligned} y[p^{n-1}m] &= Ae^{j\frac{2\pi}{N}kp^{n-1}m} + w[p^{n-1}m] \\ &= Ae^{j\frac{2\pi}{p}(\sum_{i=0}^{n-1} k_i p^i)m} + w[p^{n-1}m] \\ &= Ae^{j\frac{2\pi}{p}k_0 m} + w[p^{n-1}m], \end{aligned} \tag{3.12}$$

which is a tone with frequency $2\pi k_0/p$ radians per sample. We can find k_0 by comparing p-many candidate frequency locations $\{0, \ldots, p-1\}$. In contrast, finding k would require us to compare p^n-many frequencies. Thus, by sub-sampling the signal by a carefully-chosen period, we obtain a simpler problem of recovering a single digit of k. We choose an integer ℓ and sample the resulting signal at $m \in \{0, \ldots, \ell p - 1\}$, and let the resulting signal be $\mathbf{y}$, a vector of size ℓp. The maximum likelihood estimate of the location k_0 is then found by

$$\widehat{k}_0 = \underset{c \in \{0,\ldots,p-1\}}{\operatorname{argmax}} \ |\langle \mathbf{y}, \mathbf{u_c} \rangle|^2 , \tag{3.13}$$

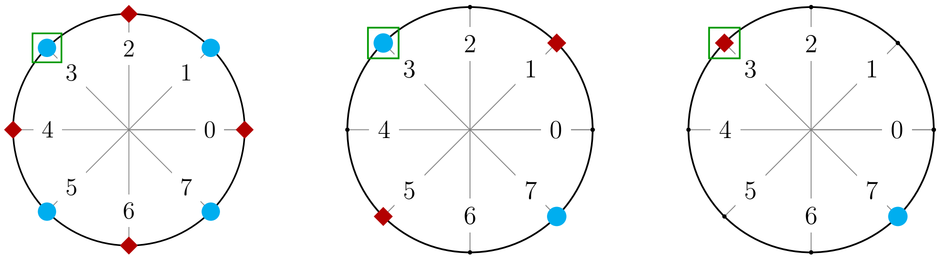
where $\mathbf{u_c} \in \mathbb{C}^{\ell p}$ with $u_c[m] = \exp(j(2\pi/p)cm)$.

Assume that we have recovered the location k_0 correctly. We can then remove its contribution from the original signal to obtain a new signal:

$$\begin{aligned} y^{(1)}[m] &= y[m]e^{-j\frac{2\pi}{p}k_0 m} \\ &= e^{j\frac{2\pi}{p^{n-1}}(k_1 + \cdots + k_{n-1}p^{n-2})m} + w[m]e^{-j\frac{2\pi}{p}k_0 m}. \end{aligned}$$

The new signal $y^{(1)}$ can be associated with a new grid corresponding to $N^{(1)} = p^{n-1}$, and its frequency is located at $k^{(1)} = k_1 + k_2 p + \cdots + k_{n-1}p^{n-2}$. Note that the resulting noise is still independent and distributed as $\mathcal{N}_c(0, \sigma^2)$ since it was centrally-symmetric. Thus, we now have a new problem of finding a tone sitting on a frequency grid of size $N^{(1)} = p^{n-1}$. We can recover the next least significant digit of k by using a sampling period of p^{n-2}. The sub-sampling and digit recovery procedure is then repeated a total of n times to recover k_i for $i = [0, n)$ to obtain k as in equation (3.11). Thus our algorithm is an n-stage method, where at stage s it recovers the digit k_s corresponding to the unknown frequency k. See Figure 3.5 for a visual illustration and Algorithm 2 for a pseudo-code.

[3] We will refer to this as the sampling period.



(a) Stage 1 (b) Stage 2 (b) Stage 3

Figure 3.5: The figure depicts an example for $N = 8$, and recovering the discrete frequency at location $k = 3$. Each stage determines if the frequency of the signal belongs to the red diamond or blue circle sets. In the first stage, frequencies that are $\pi/2$ apart belong to the same set, in the second stage frequencies that are π apart belong to the same set, and in the third stage each set contains a single frequency.

Algorithm 2: Pseudocode for the proposed method (MSDR) when recovering a single digit from the signal at each stage of the algorithm.

```
Input: Signal $y[m]$, $p$, $n$, $\ell$
Output: $\hat{k}$ (the estimate of the frequency index)
/* initialize the estimate */
$\hat{k} \leftarrow 0$
/* go over each digit */
for $s = 0, \dots, n-1$ do
    /* remove the estimate so far */
    $y^{(s)}[m] \leftarrow y[m]e^{-i2\pi\hat{k}/n^p}$
    /* subsample signal with period $p^{n-1-s}$ */
    $\mathbf{y}^{(s)} \leftarrow [y^{(s)}[0], \dots, y^{(s)}[(\ell p - 1)p^{n-1-s}]]$
    /* new digit */
    $\hat{k}_s \leftarrow \operatorname*{argmax}_{c\in\{0,\dots,1-p\}} \left|\langle \mathbf{y}^{(S)}, \mathbf{u_c}\rangle\right|^2$
    /* update the estimate */
    $\hat{k} \leftarrow \hat{k} + \hat{k}_s p^s$
```

3.4.1 Extension to a general grid

This method can be extended to any positive integer N as it admits a prime factorization of the form $N = \prod_{i=1}^{L} p_i^{n_i}$ for some L, where each p_i is a unique prime. In that setting, sub-sampling the signal with sampling period $\overline{N}_i = N/p_i^{n_i}$, yields a signal with frequency $2\pi k/N_i$ where $N_i = p_i^{n_i}$. From the resulting signal of this sub-sampling, we recover k modulo N_i via MSDR. Doing this procedure for each $i = 1, \dots, L$ yields the estimate for k modulo N_i for each i. As N_i are coprime and their product is equal to N, by Chinese Remainder Theorem [34], we can recover k from its remainders modulo N_i via several methods. For instance, letting $k^{(i)}$ denote the remainders of k modulo $p_i^{n_i}$, then Euler's theorem [34] implies that

$$k = \sum_{i=1}^{L} k^{(i)} \bar{N}_i^{(p_i-1)p_i^{n_i-1}} \pmod{N}. \tag{3.14}$$

We focus our discussion only on the case $L = 1$ as our results can be extended to more general case of $L > 1$ using this observation.

3.4.2 Limited Interval Setting

We now consider the case when we have access to a limited interval to take measurements from the signal. The setting where the samples can be taken only in $[0, N)$ is of particular interest as it corresponds to performing a 1-sparse DFT of an N-length signal using its noisy measurements. In this scenario, we cannot obtain an arbitrary number of samples for large sampling periods. For example, for $N = p^n$, when using a sampling period of p^{n-1} for recovering k_0, we obtain only p samples that are corrupted by independent noise as the next sample would be same as the first one when the samples are taken modulo N.

For such a setting, in order to increase the number of samples taken at each phase of the algorithm, we recover multiple digits in the decomposition (3.11). Denote the number of digits recovered at each phase by d; then, the largest sampling period equals to p^{n-d}, and it enables obtaining $p^n/p^{n-d} = p^d$ samples. The equation (3.13) is then adapted to search over p^d possible locations, and the number of phases of the algorithm is $\lceil n/d \rceil$.

This change in the recovery algorithm can be viewed as making the decomposition (3.11) with respect to p^d instead of p. While this allows us to increase the number of samples, it also increases the size of the search space as the number of competing locations in equation (3.13) increases to p^d. Nonetheless, the overall computational complexity still stays logarithmic in N for fixed SNR as shown in Section 3.5.1.

3.5 Analysis of the algorithm

In this section, we analyze the sample and computational complexity of our proposed method for the infinite sequence and the limited interval settings. We use the short-hand notation

$$\tilde{\sigma}^2 := \frac{1}{\text{SNR}} = \frac{\sigma^2}{|A|^2} \tag{3.15}$$

for the inverse of the signal-to-noise ratio (SNR).

3.5.1 Sample complexity

We start with the infinite sequence case. At the first stage of our procedure, we estimate the digit $k_0 \in [0, p)$ (cf., representation in (3.11)) from the noisy signal y given by equation (3.10). We assume access to ℓ periods (the value to be made precise in Theorem 5) of the signal so that we have access to $M = \ell p$ number of measurements after sub-sampling with period p^{n-1}. Then the maximum likelihood estimate $\widehat{k}_0$ in (3.13) for the digit k_0 satisfies the following high probability bound.

Theorem 5. *For any fixed $\delta \in (0, e^{-1})$, if the sample size satisfies $M \geq 7\tilde{\sigma}^2 \log(p/\delta)$ then $\mathbb{P}(\widehat{k}_0 \neq k_0) \leq \delta$.*

Theorem 5 establishes that each coefficient of k can be estimated with number of samples scaling inversely with $\mathrm{SNR} = 1/\tilde{\sigma}^2$ and logarithmically with the inverse of the error probability. We stated our results for $\delta \in (0, e^{-1})$ because the regime of small error probability is of most interest. Larger δ can be dealt with a change in the constant factor. Here the probability is taken over the randomness in the noise. Note that our guarantee does not impose any restriction on the SNR.

Proof of Theorem 5. When $M = \ell p$ samples are used, the inner product in equation (3.13) is distributed as $Z_1 \sim \mathcal{N}_c(A, \sigma^2/M)$ for the correct location. For a wrong location, that inner product is distributed as $Z_2 \sim \mathcal{N}_c(0, \sigma^2/M)$. The location is classified wrongly with probability $P(|Z_1|^2 \leq |Z_2|^2)$. Since the noise is centrally symmetric we can consider the mean of Z_1 to be $|A|$ without loss of generality. We have $P(|Z_1|^2 \leq |Z_2|^2) = P(Z_{1,r}^2 + Z_{1,i}^2 - Z_{2,r}^2 - Z_{2,i}^2 \leq 0)$, where $Z_{1,r} \sim N(|A|, \sigma^2/(2M))$, and $Z_{1,i}$, $Z_{2,r}$, and $Z_{2,i}$ are distributed as $\mathcal{N}(0, \sigma^2/(2M))$. Let $Z = (Z_{1,r}^2 + Z_{1,i}^2 - Z_{2,r}^2 - Z_{2,i}^2)/(\sigma^2/(2M))$. Since Z is a sum of sub-exponential random variables, it is a sub-exponential random variable [92]. It has sub-exponential parameters $(\sqrt{12 + 12M/\tilde{\sigma}^2}, 6)$, and $\mathbb{E}[Z] = 2M/\tilde{\sigma}^2$, where $\tilde{\sigma}^2 = \sigma^2/|A|^2$. It then follows that

$$P(Z < 0) \leq \exp\left(-\frac{4M^2/\tilde{\sigma}^4}{2(12 + 12M/\tilde{\sigma}^2)}\right), \tag{3.16}$$

from the properties of sub-exponential random variables [92]. Whenever $M \geq 7\tilde{\sigma}^2 \log(p/\delta)$, the right hand side of (3.16) is less than or equal to δ/p. Union bounding over the $p - 1$ competing locations gives the desired result. □

Theorem 5 can be applied to bound the probability of error for estimating the digit k_{i+1}, given that the prior digits $k_0, \ldots, k_i$ have been estimated correctly. Let $\widehat{k}_{\mathrm{MSDR}}$ denote the estimate of k returned by MSDR method. Then, applying the standard union bound we obtain the following result, which shows that we need $O(n \log n)$ samples to correctly recover the frequency using the MSDR method.

Corollary 3. *For any fixed $\delta \in (0, e^{-1})$, if the sample size satisfies $M \geq 7\tilde{\sigma}^2 n \log(np/\delta)$, then $\mathbb{P}(\widehat{k}_{MSDR} \neq k) \leq \delta$.*

When the samples can be taken only in the limited interval of $[0, N)$, we increase the number of samples implicitly by recovering multiple digits at each phase (refer to Section 3.4.2 for the discussion of this case). The following proposition gives the number of digits to be recovered to get an error accuracy.

Theorem 6. *For any fixed $\delta \in (0, e^{-1})$, the probability of error in recovering the first d digits of k is less than or equal to δ whenever d satisfies $p^d \geq 7\tilde{\sigma}^2 \max\left(2\log\left(7\tilde{\sigma}^2/\delta\right), \log\left(p/\delta\right)\right)$ samples are used.*

The proof of this result is similar to that of Theorem 5 with modifications to the bound in (3.16), and is thereby omitted.

Given the analysis of one stage, we can (in a manner analogous to Corollary 3) conclude the following result.

Corollary 4. *For any fixed $\delta \in (0, e^{-1})$, if the sample size satisfies*

$$M \geq 7\tilde{\sigma}^2 \frac{n}{d} \max\left(2\log\left(\frac{7\tilde{\sigma}^2 n}{d\delta}\right), \log\left(\frac{np}{d\delta}\right)\right), \tag{3.17}$$

then we have $\mathbb{P}(\widehat{k}_{MSDR} \neq k) \leq \delta$.

Corollary 4 follows immediately from an application of Theorem 6 along with the union bound.

The sample complexity in Corollary 4 is a decreasing function of d, the number of digits recovered at each phase of the algorithm. It would be minimum if we estimate all n digits in a single phase. However, increasing d results in an increase in the computational complexity of the algorithm. Thus an optimal choice of d depends on a precise characterization of the trade off between the sample and computational complexity. We now turn our attention to the latter of the two.

3.5.2 Computational complexity

For detecting each digit (or multiple digits at a time) of the frequency location k in equation (3.11), we need to compute inner products as in (3.13). Since these inner-products actually correspond to calculating the DFT coefficients of the small-sized signal, they can be computed using FFT. To estimate d-coefficients from the decomposition (3.11) in a single stage, the length of the input is ℓp^d, hence the FFT can be computed using $O(\ell p^{d+1} d)$ operations using radix-p FFT. After finding these d digits, we need to remove the contribution of the frequency recovered so far in order to obtain the next set of samples. This step requires $O(\ell p^d)$ operations for a set of d-digits thereby not changing the order of overall computational complexity. In total we have n coefficients to recover so we repeat these computations $\lceil n/d \rceil$ times, thereby giving a computational complexity of $O(\ell p^{d+1} n)$.

For the case of infinite length sequence, we choose $d = 1$, and $n\ell p = O(n\tilde{\sigma}^2 \log(np/\delta))$. Thus the overall computational complexity for the MSDR method turns out to be $O(np\tilde{\sigma}^2 \log(np/\delta))$.

For the case of access to a limited interval from the signal, we choose $\ell = 1$, and minimum value of p^d as given by Theorem 3.5. For the low SNR regime (where the complexity is dominated by the terms with $\tilde{\sigma}^2$), we get $n\ell p^d = O(n\tilde{\sigma}^2 \log(n\tilde{\sigma}^2/\delta))$; thus the overall computational complexity in this setting becomes $O(np\tilde{\sigma}^2 \log(n\tilde{\sigma}^2/\delta))$.

3.5.3 Applications to Sparse DFT

The authors of R-FFAST [69] propose an algorithm that calculates a sparse N-length DFT in sublinear (in N) sample and computational complexity whenever the number of non-zero DFT coefficients K scales sublinearly as $K = O(N^{\alpha})$ for some $0 < \alpha < 1$. The R-FFAST algorithm has a sublinear computational complexity only for *sufficiently high* SNR. This restriction is caused due to the utilization of Kay's method [38] in their algorithm, which in turn requires a sufficiently high SNR as one of its assumptions. Thus, our method is well-suited as an efficient alternative in the low-SNR regime, in particular to obtain sublinear computational complexity for any non-zero SNR. In order to use our algorithm as a part of R-FFAST, we need to target an error probability of $\delta = O(1/N^2)$. This choice allows operation at every SNR point and results in number of samples scaling as $M = O(\log(N)^2)$, and computational complexity scaling as $O(\log(N)^3)$ for the modified R-FFAST algorithm based on our method.

3.6 Numerical experiments

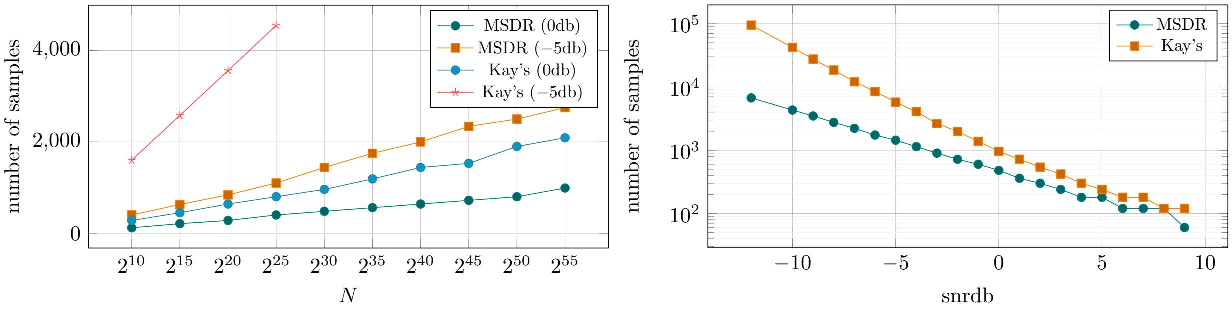


Figure 3.6: (Left) The number of samples required against the signal length (in log scale) for an error probability of 1% at 0dB SNR. (Right) The number of samples required for reaching an error probability of 1% a function of SNR for a fixed $N = 2^{30}$.

We present numerical experiments showing comparisons between our method and the successive estimation method based on Kay's estimator [69]. Note that the latter method has better sample complexity compared to Kay's estimator for identifying frequencies on a grid [69], so it is an upper bound on the performance of using Kay's method for our problem.

In the left panel of Figure 3.6, we plot the number of samples required to achieve an error probability of 1% at 0dB SNR and -5dB SNR as a function of the grid size N. The plot shows that our method requires fewer samples for each N to achieve a targeted error rate, and that the gap grows as SNR decreases. To better understand the difference in the two methods as the SNR changes, in the right panel in Figure 3.6 we plot the number of samples for reaching an error probability of 1% a function of SNR for a fixed $N = 2^{30}$. It is evident that the gap between the samples required by the algorithms grow as SNR decreases, providing evidence that MSDR is a more sample efficient alternative to Kay's method.

3.7 Conclusions and Future Directions

In this chapter, we presented a novel stagewise method (MSDR) for identifying a single tone on a grid from its noisy measurements in a sample and computationally efficient way. The sample complexity of our algorithm is close to that of the maximum likelihood (ML) method (worse only by a doubly-logarithmic factor on the grid size), while the computational complexity is logarithmic in the number of grid points which is exponentially better than that of the ML method. Our proposed method is a robust building block for many applications involving the sparse discrete Fourier transform, e.g., MRI, radio astronomy, optical imaging, wideband sparse spectrum sensing, via the recently proposed sublinear-time FFAST algorithm [69]. Our hardware implementation [93] of R-FFAST [69] uses the single tone identification algorithm based on Kay's method. Implementing the new single tone identification algorithm in hardware could improve the SNR performance and open up the application areas.

Chapter 4

A Theoretical Framework for Fast MRI

4.1 Introduction

Magnetic resonance imaging (MRI) is a noninvasive medical imaging modality of widespread use for diagnostic studies of the nervous system and the musculoskeletal system [60]. In this modality, the MRI machine acquires samples from the Fourier transform of the region to be imaged. At the high level, an inverse Fourier transform is then applied to the samples to recover the desired image. The total time a patient spends in the MRI machine is directly related to the number of Fourier-domain samples acquired. By subsampling in the Fourier domain, the total time can be cut down. However, as discussed in chapters 2 and 3, this subsampling introduces *aliasing* artifacts. To combat these artifacts, the authors of [54] propose a framework called Compressed Sensing MRI which cuts down the number of samples acquired and still enables the recovery of high-quality images by leveraging the underlying sparsity of MR images. In their framework, the Fourier domain is subsampled at random locations, and a convex optimization problem is solved to recover the MR image. Although, this cuts down the acquisition time, solving the optimization problem is costly, and requires computations that scale super-linearly with the size of the image. The image-recovery time could be a bottleneck in some applications, in particular, in interventional settings.

In order to get fast acquisition and fast image-recovery time, we can use robust FFAST [69, 63] described in Chapter 3. However, in many MR images, the sparsity is in the wavelet domain instead of the canonical domain (except for angiograms) [54]. It turns out that when the sparsity is in a wavelet domain instead of the canonical basis, the induced sparse-graph in FFAST will contain cycles. Standard techniques based on density evolution equations for analyzing the performance of decoding a sparse-graph code through peeling (see Appendix A.1) does not work whenever there are cycles in the graph. In this chapter, we develop novel techniques for this analysis, and show that the algorithm described in Chapter 3 can be adapted to handle the case where sparsity is in a wavelet domain.

We will now abstract away from the MRI problem and consider an intriguing balls-and-bins game (the interesting connection between this game and MRI will be made in Section 4.3). Assume

This chapter is based on joint work with Kabir Chandrasekher [11, 12].

there are n distinct colors, d balls of each color and M bins. You know beforehand that only $K \ll n$ of the colors, which are selected uniformly at random from the n possible colors, will be 'active', but you do not know which ones they are. You have to throw all the (dn) balls into the (M) bins. The rules of the game are as follows:

R1) For each color c, you choose a subset (possibly by a randomized strategy) $B_c \subset \{0, \ldots, M-1\}$ of size d. Then, the system throws the balls of that color c into bins $\{b + b_c : b \in B_c\}$ modulo M where b_c is sampled uniformly at random from $\{0, \ldots, M-1\}$[1].

R2) If a bin contains a single active ball, then all d balls having the same color as that ball can be removed.

R3) The process continues iteratively until either (a) all active balls have been removed or (b) there is no bin having a single active ball.

The goal of the game is to remove all active balls using the minimum number of bins. We focus on the regime in which $(n, K, M) \to \infty$, $d = O(1)$ and ask the following questions:

1. What is the optimal method of dispatching the d balls? That is, what is the optimal strategy for designing the subsets in R1?

2. Given (n, K, d), what is the minimum number of bins (M) necessary?

While this is an intriguing game in its own right, more importantly, it has connections to the design of sparse-graph codes and peeling decoding. Surprisingly, and more relevant here, it is also intimately related to the recovery of sparse wavelet representations from Fourier domain samples (see Section 4.3).

To illustrate, suppose that $d = 3$, then the best known strategy is to throw each ball at a bin selected uniformly at random. It has been demonstrated that we need asymptotically $M \simeq 1.23K$ bins as K grows. This can be shown through density evolution methods (see Appendix A.1, introduced by Richardson and Urbanke in [74], which have proven powerful in analyzing the performance of sparse-graph codes. Now suppose that $d = 6$. The natural strategy is to again throw each ball at a bin selected uniformly at random. Surprisingly, this strategy is not optimal. To see this, we give a brief introduction to the *smearing ensemble*. Consider a g dimensional vector $s = [s_1, s_2, \ldots, s_g]$ where $\sum_{i=1}^{g} s_i = d$. The ensemble is such that g bins are selected at random, and for the ith bin, the immediately following $s_i - 1$ bins are deterministically selected[2]; this is what we term the smearing ensemble. Figure 4.1 shows an example illustration for $s = [2, 2]$, and we formally define the smearing ensemble in Section 4.2. Examining Table 4.1, one can see that many simple smearing strategies outperform the fully random ensemble (corresponding to the column labeled as $1, 1, 1, 1, 1, 1$ in the table). Note that, we do not claim to design an optimal strategy for this game; rather, we provide a theoretical platform to analyze these structure-exploiting policies.

[1] Note that the only effect of this is to randomly offset the bins for each color.

[2] For example, if $s_i = 2$ (we henceforth refer to this as the smear-length), then bins b_i and $b_i + 1$ (modulo M) are selected, where b_i is uniformly selected on $\{0, 1, \ldots, M-1\}$. See Figure 4.1 for an illustration.

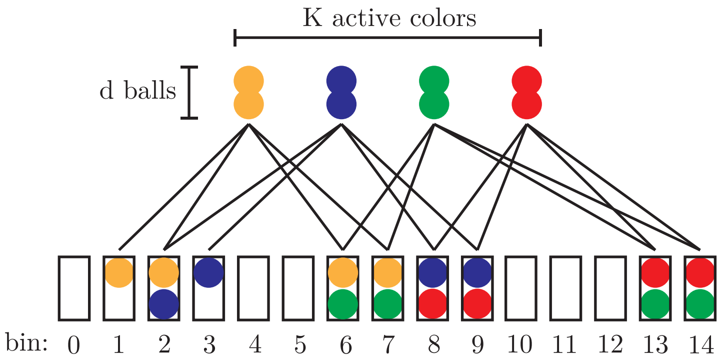


Figure 4.1: An example graph showing active balls and bins that can result from rule R1. Here $K = 4$, $M = 15$ and $d = 4$. Each color is partitioned into $g = 2$ groups and the number of balls per group is 2. The game proceeds by following the rules R2-R3. We want to find the minimum number of bins necessary to recover all the active colors.

Our balls-and-bins game is motivated by an extension of the recently proposed FFAST (Fast Fourier Aliasing-based Sparse Transform) algorithm [67] to the case where sparsity is with respect to some wavelet basis. In this setting, the balls correspond to the wavelet coefficients and the bins correspond to samples. Wavelets are universally recognized to be an efficient sparse representation for the class of piecewise smooth signals having a relatively small number of discontinuities, a very good model for many real-world signals such as natural images [90]. In particular, we note that in MRI, images are observed to be sparse with respect to appropriately chosen wavelet bases, and acquisition is in the Fourier domain [54]. The computational bottleneck in recovering these images has been observed to be the computation of multiple large Fourier transforms. Our extension of the FFAST algorithm targets this problem. The details of this extension can be found in Section 4.3.

Table 4.1: Thresholds (M/K) for graphs of various smearing for $d = 6$. Note that this table contains a subset of all possible strategies.

Regime	1,1,1,1,1,1	1,1,1,1,2	1,1,1,3	1,1,4	1,1,2,2	1,2,3	2,2,2
M/K	1.570	1.533	1.489	1.518	1.533	1.542	1.547

4.1.1 Related Work

Density evolution methods have proven powerful in analyzing the performance of sparse-graph codes and their extensions [74, 76]. Unfortunately, these methods apply only for sparse random graphs that are locally cycle-free. This is not the case for all ball-throwing strategies in the game we outlined above (for example the $[2, 2, 2]$ scheme). Recently the authors of [19, 6] have introduced approximate message passing (AMP) techniques to extend the message passing paradigm to the case when the underlying factor graph is dense. Although AMP has been successfully applied to many problem domains, e.g., [84, 56], it imposes a dense structure on the factor graph. Additionally, the authors of [40, 41] have been able to show the benefit of structure in convolutional LDPC codes through the spatial coupling effect. However, if the bipartite graph is sparse, but contains small, structured cycles, it may not be necessary to invoke such methods.

4.1.2 Main Results and Organization

The main results presented in this chapter are the derivation of exact thresholds for random graphs with smear-length 2, and bounds for higher smear-length which are empirically shown to track the actual performance of the peeling decoder well. We additionally present the details for an application to fast recovery of sparse wavelet representations of signals when acquisition is in the Fourier domain. In particular, given an n-dimensional signal that is K-sparse[3] with respect to the 1-stage Haar wavelet, our analysis shows that $2.63K$ Fourier domain samples are needed to recover the signal in $O(K \log K)$ time.

The rest of this chapter is organized as follows. In Section 4.2.1, we derive exact thresholds for the specific case of 2-smearing. In Section 4.2.2, we derive lower bounds for the probability of recovery in the case of arbitrary smearing. We outline the connection between the ball coloring game and the recovery of sparse wavelet representations in Section 4.3. We conclude with Section 4.4 by summarizing some interesting open problems and conjectures that have resulted from this work.

4.2 Main Results

In this section, we introduce a new random graph ensemble, termed the ‘smearing ensemble’, and present a method for deriving the density evolution recursions for graphs from this ensemble. The derivations are done for a smear-length of 2, and we give lower bounds for smear-length L. We now formally define the smearing ensemble:

Definition 5. Let $\mathcal{G}(K, M, s)$ denote the ‘smearing graph ensemble’ with K left nodes and M right nodes with connectivity characterized by s. Each left node selects g right nodes, where g is the length of the vector s. At the ith iteration ($0 \le i < g$), edges are put between the left node and right nodes $\{b_i, b_{i+1}, \dots, b_{s_i - 1}\}$ modulo M where b_i is selected uniformly at random from $\{0, 1, \dots, M-1\}$.

Henceforth, we are going to use the terms ‘left node’ and ‘ball’ interchangeably as well as ‘right node’ and ‘bin’ interchangeably. We additionally refer to balls thrown to the same set of bins as a stream. Now, let $\lambda := gK/M$. If K balls are thrown into the M bins randomly, the degree (that is the number of balls in) of each stream will be distributed like Poisson(λ) by the Poisson approximation to the binomial distribution (see Appendix A.1). We now carefully derive the density evolution recurrence for smearing ensembles with the maximum smearing length of 2.

[3] That is, an n-dimensional signal with exactly K non-zero entries.

4.2.1 Density Evolution on Graphs with Smearing Length of Two

In our ball-coloring game, we noted that threshold for the setting $[2, 2, 2]$ outperforms that for the setting $[1, 1, 1, 1, 1, 1]$. In this section, we give exact analysis for these thresholds[4]. First, we define our notation in Table 4.2.

Table 4.2: Notation for density evolution on a smearing graph with smear-length 2.

Notation	Definition
x_t :	Probability that a random ball is *not* removed by iteration t
q_t :	Probability that none of the bins in a smeared pair is resolved by iteration t
d_t :	Probability that *all* balls in the same stream as the reference ball are removed by time t
s_t :	Probability that *all* balls in a stream which intersects, but does not fully overlap, with the reference stream are removed by time t

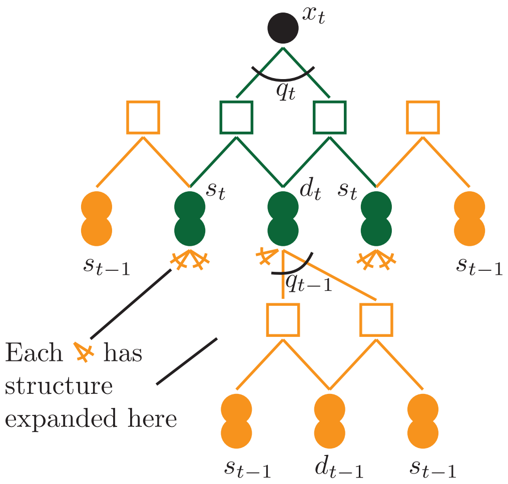


Figure 4.2: Depth 2 neighborhood of a ball in a graph with $[2, 2, 2]$ smearing (see Definition 5). The bins and balls are grouped by their colors with respect to distance from the root.

Unlike as in conventional density evolution methods in the LDPC literature which are based on an edge perspective, we take a node perspective here because dependencies between edges in the smearing setting complicate the analysis. We refer the interested reader to [75] for a discussion of the differences between node perspective and edge perspective. Our goal is to derive recurrences

[4] Although thresholds can be derived for the other elements in the table, we give the $[2, 2, 2]$ derivation for clarity.

for each of the quantities defined in Table 4.2. In this case, the equations for x_t and q_t are clear:

$$x_t = q_t^3, \tag{4.1}$$

$$q_t = 1 - d_t(1-(1-s_t)^2). \tag{4.2}$$

To see these, we recall the dynamics of the peeling decoder: a ball is removed as soon as any of its neighbors are removed, whereas a bin is removed only if all balls contained in it are removed. Thus, x_t only occurs when none of the pairs of smeared bins to which it is connected are removed. On the other hand, for a bin to be removed, *all* of its connected nodes must be removed. Thus, every ball in the same stream as the reference ball must be removed, which happens with probability d_t. Additionally, at least one of the two streams that do not fully overlap with the reference ball must be removed, which happens with probability $(1-(1-s_t)^2)$. Now, note that in Figure 4.2, the recurring structures are those with probabilities s_t and d_t, so we are done upon calculating these quantities.

We first calculate d_t. Let D_1 denote the number of balls in this stream other than the reference ball we are looking at. It follows that D_1 is distributed as Binomial$(3(K-1), 1/M)$, which can be approximated well for large K and M by Poisson(λ), where $\lambda = 3K/M$. It follows that:

$$\begin{aligned} d_t &= \sum_{i=0}^{\infty} P(D_1 = i)(1-q_{t-1}^2)^i \\ &= \sum_{i=0}^{\infty} e^{-\lambda}\frac{\lambda^i}{i!}(1-q_{t-1}^2)^i = e^{-\lambda q_{t-1}^2}. \end{aligned} \tag{4.3}$$

Now, we tackle s_t. Note that intuitively, $s_t \geq d_t$. This is because each ball in a stream tracked by s_t gets the same independent help from 2 bins as d_t. However, there is additionally a shared bin between all these balls. This shared bin is able to aid in the removal of the stream tracked by s_t when exactly one ball is left in the stream, and the bin has no contributions from elsewhere. The incorporation of this help is the key ingredient in using the structure to help characterize the decoding thresholds. Letting D_2 be the number of balls in this stream, we precisely characterize this as follows:

$$\begin{aligned} s_t &= \sum_{i=0}^{\infty} P(D_2 = i)\big[(1-q_{t-1}^2)^i + i s_{t-1} q_{t-1}^2 (1-q_{t-2}^2)^{i-1}\big] \\ &= e^{-\lambda q_{t-1}^2} + \lambda s_{t-1} q_{t-1}^2 e^{-\lambda q_{t-2}^2}. \end{aligned} \tag{4.4}$$

Note that the first term in this recursion is exactly d_t. The second term describes the help received from the shared bin. In order to properly characterize this term, it is necessary to introduce the notion of *memory*: the shared bin can help if it has all contributions removed except for one by time t. Unlike when the neighborhood is cycle-free and the branches of the computation tree become independent [30], when cycles are introduced, there is dependence between the branches. The introduced memory captures exactly this dependence. In Figure 4.3 we plot the evolution of x_t given in equation (4.1) obtained using numerical experiments against the predictions by our theory, and we can see that they are in very close agreement.

We summarize the results of this section with the following lemma:

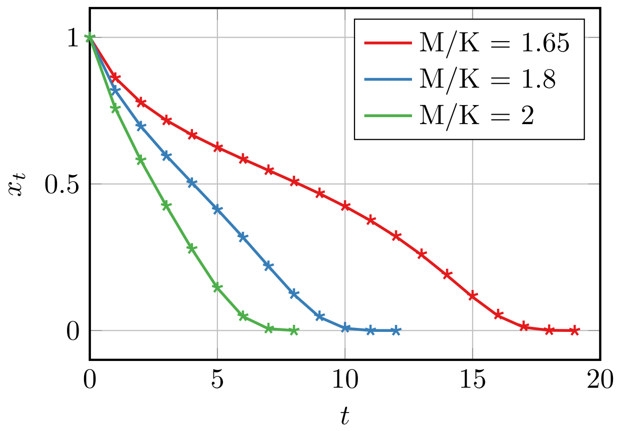


Figure 4.3: Evolution of x_t as a function of the number of peeling rounds t for different values of M/K. The markers are obtained by simulations of peeling rounds and the solid lines are obtained through the density evolution equations. We see that our equations capture the behavior of the peeling decoder well.

Lemma 1. *Consider a random graph from the ensemble $G(K, M, [2, 2, 2])$. Then, for $M \geq 1.547K$, recovery using the peeling decoder will succeed with high probability.*

Proof. The threshold follows from the density evolution derived above. It is important to note that these recurrences were derived using only high probability cycles and to be precise, it is necessary to show that the actual fraction of unidentified balls concentrates around the average, x_t. This is done with a slight tweak of the proof presented in Appendix A.2. □

The analogous density evolution equations can be derived exactly for the cases of smear-lengths greater than 2; however, the equations get complicated in that case. Instead, in the next section, we derive simple, but effective, lower bounds for L-smearing. We do this by using the following principles of generalization, inspired by the derivation above:

1. In a stage with L smearing, there will be $L-1$ steps of memory necessary in order to properly capture the smearing structure.

2. In a stage with L smearing, the number of recurring structures will be L.

3. Shared bins can be used through the introduction of memory in the recursion.

Following the above principles of generalization, we now derive lower bounds on the density evolution for the general smearing ensemble.

4.2.2 Lower Bound on L-smearing

For clarity, we will consider the ensembles with $s = [L, L, L]$. Along with x_t, q_t, d_t as described in Table 4.2, we define quantity $s_t^{(i)}$ as the probability that *all* nodes in the streams which *does*

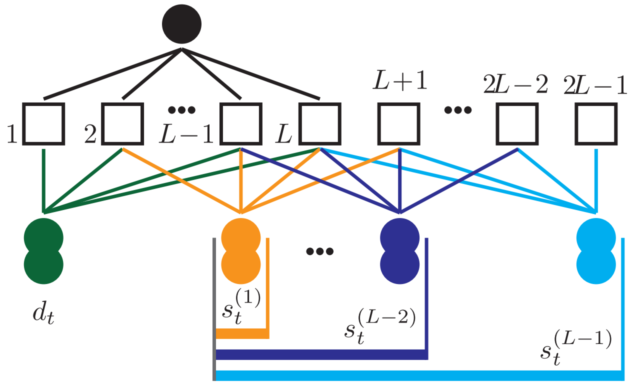


Figure 4.4: Depth 1 neighborhood of a ball under L-smearing. The quantities d_t and $\{s_t^{(i)}\}_{i=1}^{L-1}$ are the recurrent structures, where $s_t^{(i)}$ tracks the joint probability that all the streams which intersect with the reference stream in $\{j\}_{j=L-i}^{L-1}$ bins are removed at time t. The neighborhood is drawn for the right-hand side of the bins labeled from 1 to L only, and the symmetric left-hand side is omitted.

not intersect with the reference stream in j bins where $1 \leq j \leq i$ bins are removed at time t (see Figure 4.4 for an illustration of this notation).

As described in the principles of generalization, there will be L recursions, and up to $L-1$ memory. The probability d_t is unaffected as it depends only on the other stages. All the $s_t^{(i)}$, however, can have up to i memory (corresponding to the number of bins that do not overlap with the reference ball). Thus, we take an approach much like a first-order approximation of a Taylor series, and allow each $s_t^{(i)}$ to use only one step of memory. The following lemma characterizes the critical quantity q_t in terms of its component streams.

Lemma 2. *In a stage with L-smearing,*

$$1 - q_t = d_t \left[2s_t^{(L-1)} + \sum_{i=2}^{L-1} s_t^{(i-1)} s_t^{(L-i)} - \sum_{i=1}^{L-1} s_t^{(i)} s_t^{(L-i)} \right].$$

Proof. See Appendix 4.A □

The following lemma establishes monotonicity of $1 - q_t$ with respect to $d_t, s_t^{(i)}$. Using this fact, we can use plug in bounds on d_t and $s_t^{(i)}$ to give a bound on the performance of the peeling decoder on graphs from the smearing ensemble.

Lemma 3. *Let $f(d_t, s_t^{(1)}, \ldots s_t^{(L-1)}) = 1 - q_t$, then $f(d_t, s_t^{(1)}, \ldots s_t^{(L-1)})$ is non-decreasing in $(d_t, s_t^{(1)}, \ldots, s_t^{(L-1)})$.*

Proof. See Appendix 4.A. □

Given the characterization of q_t, it now remains to give lower bounds for $d_t, s_t^{(1)}, \ldots, s_t^{(L-1)}$. The following provides a set of bounds:

$$d_t = e^{-\lambda q_{t-1}^2}, \tag{4.5}$$

$$s_t^{(i)} \geq e^{-i\lambda q_{t-1}^2} + \lambda q_{t-1}^2 e^{-\lambda q_{t-2}^2} r_{t-1}^{(i)}, \tag{4.6}$$

for $i \in \{1, \ldots, L-1\}$, where

$$r_t^{(i)} := \sum_{j=1}^{i} \sum_{k=1}^{j} s_{t-1}^{(k)} e^{-(L-k-1)\lambda q_{t-2}^2} e^{-(k-1)\lambda q_{t-1}^2}. \tag{4.7}$$

There is a simple way to think about the problem to attain these bounds. Consider the bound given in equation (4.6) on $s_t^{(i)}$. This term tracks the joint probability that all the streams which intersect the reference stream in $L - j$ bins where $1 \leq j \leq i$ are removed at time t. In order for all of these streams to be removed, there are two cases:

1. Each stream was removed from another stage. This probability is tracked by the first term: $e^{-i\lambda q_{t-1}^2}$.

2. Exactly one ball remains among all the streams. This probability is tracked by the second term.

We focus on the second case. Suppose that the remaining ball is in the stream which intersects the reference stream in j bins. This implies that it is also contained in $L - j$ shared bins that do not intersect the reference stream. It can be removed by any of these bins, as long as it is the only contribution. This help is tracked by the summation in the second term.

Corollary 5. *The lower bounds given in equations* (4.5) *and* (4.6) *imply a lower bound on x_t, the probability that a random node is removed at time t.*

Proof. This follows directly from Lemma 2 and Lemma 3. □

We now give numerical experiments corroborating that the bounds in Corollary 5 capture the actual thresholds well. We experimentally find the thresholds for full recovery by sweeping λ, and compare them to the thresholds implied by the bounds. Figure 4.5 shows these for filters with different lengths.

4.3 Connections to Recovery of Sparse Wavelet Representations

In MRI, one acquires samples of the Fourier transform of an input signal of interest. MRI speed is directly related to the number of samples acquired. An inverse transform is then used to recover the original signal. Mathematically, let x be an n-length signal, and $X := F_n x$ be its Fourier transform,

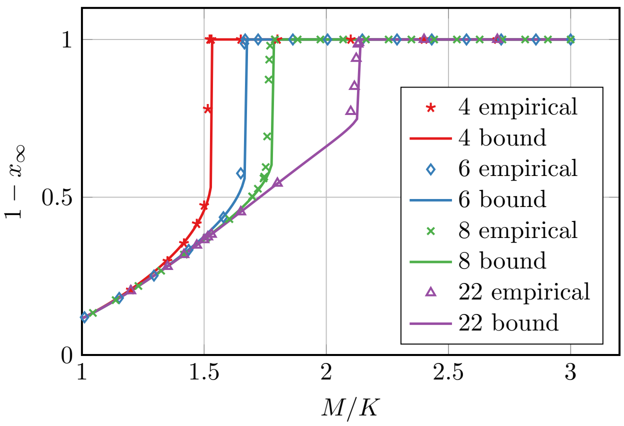


Figure 4.5: Lower bounds against empirical simulations with settings $s = [1, 1, L]$ for various values of L. The vertical axis denotes the probability that a random ball is removed when peeling stops (that is, $1 - x_t$ as $t \to \infty$), and the horizontal axis denotes the oversampling factor. We chose these ensembles for the plot as they are of particular interest for recovery of sparse wavelet representations (see Section 4.3 for the connection).

where F_n is the Fourier matrix of size $n \times n$. In MRI, the problem is to recover x from $\{X_i : i \in \mathcal{I}\}$ where the set $\mathcal{I}$ denotes the sampling locations, and this set is a design parameter.

We now present how the game of balls-and-bins and its analysis as described in Sections 3.3 and 4.2 relates to MRI. For ease of illustration we confine ourselves to the noiseless setting and exact sparsity, but these assumptions can be relaxed. If the signal x is K-sparse, one can use the FFAST algorithm to recover x from $O(K)$ samples with $O(K \log K)$ computations [63, 67]. However, the images of interest in MRI are generally not sparse, but they do have sparse wavelet representations [54]. That is, we can express $x = W_n^{-1}\alpha$, where W_n^{-1} is an appropriate wavelet, and α is sparse. Under this signal model, the problem of recovering α can be transformed into the problem of decoding on an erasure channel using a sparse-graph code. In particular, the graph for the code is drawn from a smearing ensemble with smearing length L a function of the length of the underlying wavelet filter.

Furthermore, assume that α is K-sparse and the length of x is of the form $n = f_1 f_2 f_3$ where f_1, f_2 and f_3 are co-prime. For $m \in \mathbb{Z}$ that divides n, let $D_{m,n}$ be the regular downsampling matrix from length n to m, that is, $D_{m,n} = [I_m \cdots I_m]$ with I_m is repeated n/m times. Let y_{f_ℓ} for $\ell \in \{1, 2, 3\}$ be the inverse Fourier transform of the downsampled X, that is, $y_{f_\ell} := F_{f_\ell}^{-1} D_{f_\ell,n} X$. Using the properties of Fourier Transform, it follows that $y_{f_\ell} = D_{f_\ell,n} x = D_{f_\ell,n} W_n^{-1}\alpha$.

Now, for simplicity, assume that W is a block transform with block size L (eg., for 1 stage Haar wavelets $L = 2$), and the support of α is chosen uniformly random over the subsets of size K. Using the relations between y_{f_1}, y_{f_2} and y_{f_3} and α, recovering α is equivalent to decoding on a random graph from $G(K, M = f_1 + f_2 + f_3, s = [L, L, L])$.

We can actually 'improve' the induced graph if a factor of the signal length has L as a factor. Say that L divides f_1, it follows that $A_{f_1,n} W_n^{-1} = [W_{f_1}^{-1} \cdots W_{f_1}^{-1}]$, where $W_{f_1}^{-1}$ is repeated n/f_1 times. It can be verified that $y'_{f_1} := W_{f_1} y_{f_1} = A_{f_1,n} a$, hence it aliases the wavelet coefficients regularly without smearing. The relation between y'_{f_1}, y_{f_2} and y_{f_3} and α then induces a graph from $G(K, M = f_1 + f_2 + f_2, s = [1, L, L])$, which gives raise to a better threshold.

To complete the equivalence to decoding a sparse-graph code on an erasure channel, we need a mechanism to check if there is a single component in a bin (a single color in a bin). This can be implemented by processing a shifted version of x (incurring an additional factor of 2 of oversampling). We end this section with the following lemma.

Lemma 4. *Consider a signal α with ambient dimension n and sparsity K, and access to samples from $F_n W_n^{-1}\alpha$. Then, the subsampling scheme described above along with the peeling decoder is able to exactly recover the sparse signal α using $2.63K$ samples and time complexity $O(K \log K)$.*

Proof. The threshold $2.63K$ follows from the density evolution derived in Section 4.2. For computational complexity, the creation of the bipartite graph is done in $O(K \log K)$ time. Decoding uses an iterative decoder with a constant number of iterations [76], and the number of bins iterated over is $O(K)$. Combining these two, we get $O(K \log K)$ for computational complexity. □

4.4 Conclusions and Future Directions

We have introduced a new random graph ensemble, termed the ‘smearing ensemble’ and developed a framework for deriving density evolution recurrences for random graphs from this ensemble. Recalling our balls-and-bins game, our results show that some amount of smearing can lead to a better strategy than the full random case. A fascinating open question arises here: what is the optimal ball-throwing strategy and what are the density evolution recurrences for such a strategy? Here, we have given the first steps in analyzing this problem rigorously. To do this, we have leveraged the existence of small, structured cycles and introduced the notion of memory into our density evolution. We believe there to be a deep connection between the introduction of memory in our recurrences and the introduction of the ‘Onsager’ term in the update equations of AMP [20]. We additionally believe the gains seen in spatially coupled ensembles [41] are intimately related to the structural gains of the smearing ensemble. An interesting open problem is to determine the nature of these connections. We have additionally shown the practical connection between the smearing ensemble and the recovery of a sparse wavelet representation of a signal whose samples are taken in the Fourier domain.

Appendix

4.A Proofs

Here, we provide proofs for the bounds on L-smearing.

Proof of Lemma 2. Consider the depth 1 neighborhood of a random node, as shown in Figure 4.4. Let A_i, for $i \in \{1, \ldots, L\}$, be the event that the check node labeled with i is resolved (that is, contains no other balls than the root) by time t. Then, we have

$$\begin{aligned}
1 - q_t &= \Pr\left(\bigcup_{i=1}^{L} A_i\right) \\
&= \Pr\left(A_1 \cup (A_2 \setminus A_1) \cup (A_3 \setminus (A_2 \cup A_1)) \cup \cdots \cup (A_L \setminus (A_{L-1} \cup \cdots \cup A_1))\right) \\
&\overset{(a)}{=} \Pr(A_1) + \Pr(A_2 \setminus A_1) + \Pr(A_3 \setminus (A_2 \cup A_1)) + \cdots + \Pr(A_L \setminus (A_{L-1} \cup \cdots \cup A_1)) \\
&\overset{(b)}{=} \Pr(A_1) + \Pr(A_2 \setminus A_1) + \Pr(A_3 \setminus A_2) + \cdots + \Pr(A_L \setminus A_{L-1}) \\
&= \sum_{i=1}^{L} \Pr(A_i) - \sum_{i=1}^{L-1} \Pr(A_i, A_{i+1}), \qquad (4.8)
\end{aligned}$$

where (a) follows because the events are disjoint, and (b) follows because $A_i \setminus \cup_{j=1}^{i-1} A_j = A_i \setminus A_{i-1}$ where empty union is defined to be the empty set. Now, we note that $\Pr(A_1) = \Pr(A_L) = d_t s_t^{(L-1)}$ and $\Pr(A_i) = d_t s_t^{(L-i)} s_t^{(i-1)}$ where $1 < i < L$. Additionally, we can see that $\Pr(A_i, A_{i+1}) = d_t s_t^{(i)} s_t^{(L-i)}$. Plugging these into equation (4.8) gives the result. □

Proof of Lemma 3. First, we note that

$$d_t \geq s_t^{(j)} \geq s_t^{(i)} \qquad (4.9)$$

if $i \geq j \geq 1$ by definition. Additionally, we can see that $f(0, 0, \ldots, 0) = 0$ and $f(1, 1, \ldots, 1) = 1$. Now, we have

$$\begin{aligned}
\frac{\partial f}{\partial d_t} &= 2s_t^{(L-1)} + \sum_{i=2}^{L-1} s_t^{(i-1)} s_t^{(L-i)} - \sum_{i=1}^{L-1} s_t^{(i)} s_t^{(L-i)} \\
&= s_t^{(L-1)} + \sum_{i=1}^{L-2} s_t^{(1)} (s_t^{(L-i-1)} - s_t^{(L-i)}) + s_t^{(L-1)} - s_t^{(L-1)} s_t^{(1)} \\
&\geq s_t^{(L-1)} + s_t^{(L-1)} (1 - s_t^{(1)}), \\
&\geq 0
\end{aligned}$$

where the first inequality follows by equation (4.9), and the last inequality follows since $s_t^{(1)} \leq 1$ and $s_t^{(L-1)} \geq 0$. Additionally, we can see that

$$\frac{\partial f}{\partial s_t^{(i)}} = 2d_t (s_t^{(L-i-1)} - s_t^{(L-i)}) \geq 0,$$

for $1 \leq i < L-1$, where the inequality follows from equation (4.9). The argument follows similarly for $\frac{\partial f}{\partial s_t^{(L-1)}}$. Thus, the result follows because all partial derivatives are non-negative. □

Part II

Contemporary Problems of Interest

In this part of the thesis, we study speeding up “contemporary” problem settings:

- **Chapter 5:** Learning DNA repair landscape in CRISPR-Cas9 experiments through sparse function recovery
- **Chapter 6:** Speeding up distributed computing by reducing the effect of slow compute nodes
- **Chapter 7:** Fast matrix multiplication with output sparsity

Chapter 5

Low-Degree Pseudo-Boolean Function Recovery

5.1 Introduction

The Walsh-Hadamard transform (WHT) is a well-known signal processing tool that has applications in areas such as image compression [72], multi-user communications via spreading sequences and spectroscopy [59], and more recently, in learning pseudo-Boolean functions and decision trees [39], and hypergraph sketching [50]. The WHT of an N-length signal, in general, can be calculated with $O(N \log N)$ operations via a recursive algorithm [47, 36] similar to the Fast Fourier Transform (FFT). Recent work has shown that when the number of significant WHT coefficients is much smaller than the ambient dimension N, the number of samples needed from the input signal and the computational complexity for recovering the significant WHT coefficients can be reduced [78, 49]. These algorithms have similar work flows to that of the sparse DFT algorithm described in Chapter 3. First, they subsample the input signal and take the WHT of the subsampled signal, which can be done using fewer operations. Then a peeling decoder recovers the WHT of the input signal. For classes of signals whose WHT has K-nonzero coefficients, where $K = N^{\alpha}$ for $\alpha \in (0, 1)$, the algorithms compute the WHT using $O(K \log(N))$ samples with $O\big(K \log^2(N)\big)$ operations. The algorithms assume that the support set is uniformly distributed over all subsets of cardinality K in the WHT domain, and the recovery guarantees are probabilistic[1], where the probability is calculated over the randomness of the support set.

In a variety of applications of interest, the significant WHT coefficients are at indices that have a small number of ones in their binary expansion. This is the case for recovering polynomials $f : \{-1, +1\}^n \to \mathbb{R}$ where the significant coefficients of the polynomial are for the monomials with small degrees. Such polynomials arise, for example, in decision trees with small depths [42] and as cut functions of hypergraphs where the hyperedges have small degrees [50]. Because the support of significant coefficients differs from being uniform, the algorithms [78, 49] cannot be used directly.

This chapter is based on joint work with Swanand Kadhe [61], and Amirali Aghazadeh [2].

[1] The guarantees are of the form $\Pr$ (recovery) $\to 1$ as $N \to \infty$.

In this chapter, we show that we can recover the WHT of signals whose non-zero WHT coefficients are at indices whose binary expansion have a small number of ones (that is, "degree" of the multinomial is small), in a sample and computationally efficient way. We do this by designing subsampling patterns carefully so that such non-zero coefficients do not alias together. The sampling patterns we use have interesting connections with coding theory: they are obtained by using parity check matrices corresponding to codes with good minimum distance properties (see Property 4 for a precise statement). In particular, we show that subsampling a signal at points obtained by using the parity check matrix of a perfect code (see Definition 6) with minimum distance $2t+1$, we are sample optimal in recovering WHT coefficients whose binary expansion have at most t ones.

We use a variant of our method for learning *fitness* landscapes for DNA repair outcomes after CRISPR-Cas9 double-strand breaks. After a double-stranded break, DNA is repaired by the cell with possible insertions and deletions of nucleotides. Fitness, in our problem, captures the propensity of repair outcomes to satisfy a criteria of interest, such as the probability of insertion of a certain base pair in the DNA during repair. Learning a model for repair outcomes is of significant interest in biology as it can be used for knocking out genes or treating genetic conditions through gene editing. Recent studies have shown that after a double-strand break, the DNA repair outcomes are a function of the DNA sequence around the cut site [65]. As nucleotides in the DNA can take on 4 different bases (A,T,C,G), a sequence of n nucleotides can have 4^n possible values, which poses a high-dimensional problem. First step in our approach is to represent the DNA sequence in a window around the cut site as a binary sequence, and study the function that maps this sequence to the fitness of DNA repair outcomes. We observe that this function is sparse in the WHT domain (see Section 5.5.3), and use our method for learning it in a sample and computationally efficient way. Furthermore, the learned model provides interpretable insights on the DNA repair process as the nonzero coefficients in the WHT capture which nucleotides around the cut site interact to influence the fitness of the repair outcomes.

5.1.1 Notation and the Organization of the Chapter

We use row vectors having elements in $\mathbb{F}_2$ to denote indices. For a binary string $\ell \in \mathbb{F}_2^n$, let ℓ_i denote its i-th coordinate, the Hamming weight of ℓ is defined as the number of ones in ℓ, that is, $\mathrm{w_H}(\ell) = |\{i : \ell_i \neq 0\}|$. We use $\mathcal{K}_t^n := \{\ell \in \mathbb{F}_2^n : \mathrm{w_H}(\ell) \leq t\}$ to denote the set of all binary strings of length n with Hamming weight less than or equal to t. For two strings $\ell \in \mathbb{F}_2^n$ and $m \in \mathbb{F}_2^n$, both of length n, we define $\langle \ell, m \rangle = \sum_{i=0}^{n-1} \ell_i m_i$ where the summation is over $\mathbb{F}_2$. We use $h(\cdot)$ to denote the binary entropy function $h(p) = -p\log_2(p) - (1-p)\log_2(1-p)$ for $p \in (0,1)$ and $h(p) = 0$ for $p \in \{0,1\}$.

The organization of this chapter is as follows. In Section 5.2 we cover background on WHT and coding theory that will be useful to explain our main concepts. In Section 5.3, we formulate the problem and explain our method. In Section 5.4, we look at the performance of various deterministic and randomized constructions for generating subsampling patterns. We apply a variant of our algorithm for learning the fitness landscape of DNA repair outcomes in CRISPR-Cas9 experiments in Section 5.5. In Section 5.6, we describe hypergraph sketching as another application of our algorithm, and conclude with Section 5.7.

5.2 Background

We first cover relevant background on WHT and then notions of coding theory that will be useful.

5.2.1 Walsh-Hadamard Transform

Let signal length be $N = 2^n$, for some integer $n \geq 0$. The Walsh-Hadamard transform (WHT) of a vector $x \in \mathbb{R}^N$ is the vector $X \in \mathbb{R}^N$ whose ℓth element equals to[2]

$$X_\ell = \frac{1}{N} \sum_{m \in \mathbb{F}_2^n} x_m (-1)^{\langle \ell, m \rangle}. \tag{5.1}$$

The inverse transformation is given by

$$x_m = \sum_{\ell \in \mathbb{F}_2^n} X_\ell (-1)^{\langle m, \ell \rangle}, \tag{5.2}$$

which has the same form with equation (5.1). Looking at the inverse transformation, we can interpret the WHT as follows. Consider the polynomial (multinomial) in n variables

$$f(z_0, \ldots, z_{n-1}) = \sum_{\ell \in \mathbb{F}_2^n} X_\ell \prod_{i : \ell_i = 1} z_i, \tag{5.3}$$

then x_m is equal to the evaluation of this polynomial at $z_i = (-1)^{m_i}$ for $i \in \{0, \ldots, n-1\}$. As can be seen, the WHT coefficient X_ℓ multiplies the monomial $\prod_{i:\ell_i=1} z_i$, which is a function of $\mathrm{w_H}(\ell)$ many variables $\{z_i : \ell_i = 1\}$, and we call $\mathrm{w_H}(\ell)$ the degree of the coefficient X_ℓ. For example, the pseudo-Boolean function

$$f(z_0, z_1, z_2, z_3, z_4) = 12 z_0 z_3 - 3 z_2 + 6 z_0 z_1 z_4, \tag{5.4}$$

has three terms with degrees 2, 1, and 3 and WHT coefficients 12, -3, and 6, respectively. Note that the pseudo-Boolean function in this example is also sparse since out of $2^5 = 32$ possible monomials[3] only three are active (that is, have non-zero coefficients).

In order to reduce down the number of samples we need from the input signal (or evaluations of the polynomial), we are going to subsample it. The following property relates the WHT of a subsampled signal to the WHT of the input signal.

Property 1 (Subsampling/Aliasing). *Let x be an $N = 2^n$ length vector. Given a full-rank matrix $\mathbf{H} \in \mathbb{F}_2^{b \times n}$, let y be the $B = 2^b$ length vector where $y_m = x_{m\mathbf{H}}$ for all $m \in \mathbb{F}_2^b$. Then, the WHT coefficients of y satisfy*

$$Y_k = \sum_{j \in \mathbb{F}_2^N : j\mathbf{H}^\top = \ell} X_j, \tag{5.5}$$

for all $\ell \in \mathbb{F}_2^B$, where X_j is the jth WHT coefficient of x.

[2] Note that the scaling of $1/N$ is usually included in the inverse WHT equation. We included the scaling in the WHT equation because it helps us simplify the notation in this chapter.

[3] The monomials are $\left\{\prod_{z \in \mathcal{Z}} z : \mathcal{Z} \in \mathcal{P}\left(\{z_0, z_1, z_2, z_3, z_4\}\right)\right\}$, where $\mathcal{P}\left(\cdot\right)$ denotes the powerset of its argument.

Property 1 says that if we subsample the input signal x at indices given by the range of the rows of the matrix $\mathbf{H}$, the jth index WHT coefficient aliases to the bin with index $j\mathbf{H}^\top$. Below is an example of the aliasing pattern induced by subsampling.

Example 1. Given $x \in \mathbb{R}^8$, let $y \in \mathbb{R}^4$ be equal to $y_m = x_{m\mathbf{H}}$ where $\mathbf{H} \in \mathbb{F}_2^{2\times 3}$ is

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}. \tag{5.6}$$

Then, the elements of y are: $y_{00} = x_{000}$, $y_{01} = x_{011}$, $y_{10} = x_{101}$, and $y_{11} = x_{110}$, and the WHT coefficients of y satisfy

$$\begin{aligned} Y_{00} &= X_{000} + X_{111}, \\ Y_{01} &= X_{100} + X_{011}, \\ Y_{10} &= X_{010} + X_{101}, \\ Y_{11} &= X_{110} + X_{001}, \end{aligned} \tag{5.7}$$

which follows from the Property 1. ■

There is another way to view these aliasing patterns. For the $b \times n$ matrix $\mathbf{H}$, there is a full rank $(n-b) \times n$ matrix $\mathbf{G} \in \mathbb{F}_2^{(n-b)\times n}$ satisfying $\mathbf{H}\mathbf{G}^\top = \mathbf{0}$, where $\mathbf{0}$ is the all zero matrix of appropriate size. Then, if the WHT coefficient at ℓ of the input signal aliases to a WHT coefficient of the sampled signal, then coefficients of the input signal at $\{\ell + u\mathbf{G} : u \in \mathbb{F}_2^{n-b}\}$ also aliases to that same WHT coefficient of the sampled signal. Note that because the rows of $\mathbf{H}$ and $\mathbf{G}$ have inner products equal to 0, sampling of the signal in the input space and the aliasing in the WHT domain happens in orthogonal dimensions. The following is an example for this equivalent view of aliasing, and Figure 5.1 shows examples of sampling and corresponding aliasing patterns that makes the geometry more evident.

Example 2. For the matrix $\mathbf{H}$ given in Example 1, the matrix $\mathbf{G}$ given by

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \tag{5.8}$$

satisfies $\mathbf{H}\mathbf{G}^\top = \mathbf{0}$. As can be seen in equation 5.7, the coefficients at locations ℓ and $\ell + (1,1,1)$ are aliased to the same coefficient in Y. ■

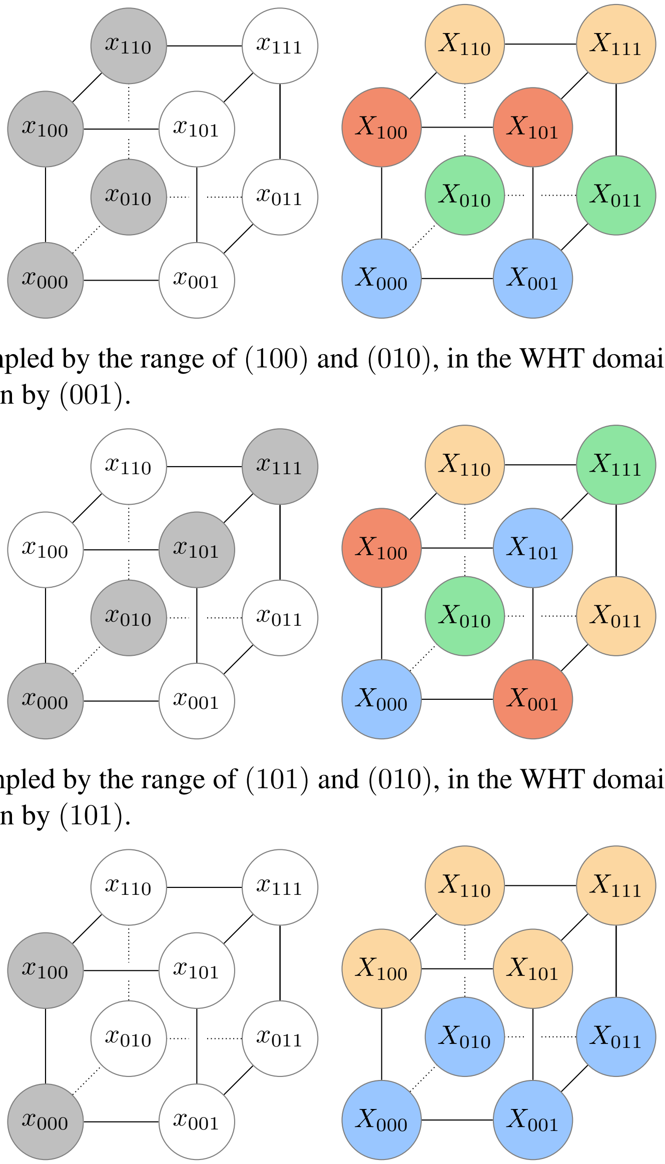
There is one more property of WHT that will be useful in deriving our algorithm.

Property 2 (Shifting/Modulation). *For $p \in \mathbb{F}_2^n$, $z_m = x_{m+p}$ has WHT coefficients*

$$Z_\ell = (-1)^{\langle \ell, p\rangle} X_\ell, \tag{5.9}$$

for all $\ell \in \mathbb{F}_2^B$, where X_ℓ is the ℓth WHT coefficient of x.

This property says that shifting the signal generates modulation patterns for WHT coefficients based on the coefficient's index and the shift.



(a) When the input is sampled by the range of (100) and (010), in the WHT domain, coefficients are aliased along the dimension given by (001).

(b) When the input is sampled by the range of (101) and (010), in the WHT domain, coefficients are aliased along the dimension given by (101).

(c) When the input is sampled by the range of (100), in the WHT domain, coefficients are aliased along the range of vectors (001) and (010).

Figure 5.1: Left images correspond to the input domain, and the right images correspond to the WHT domain. Gray shaded nodes denote the samples taken from the input signal. Nodes with the same color are aliased together in the WHT domain. The aliasing happens along vectors that are orthogonal to the vectors along which the samples of the input signal are chosen.

5.2.2 Relevant Coding Theory Background

We give a brief overview the relevant notions from linear codes that will be useful to explain our main concepts. A binary (n, k, d) linear code $\mathcal{C}$ consists of 2^k binary strings each of length n, and has minimum distance $d_{\text{min}} := \min_{\ell \neq m, \ell, m \in \mathcal{C}} \mathrm{w_H}(\ell + m)$, equal to d. Note that, in a linear code, minimum distance is equal to the minimum over the Hamming weights of the non-zero codewords of the code. The codewords of a linear code can be generated as the range space of the rows of an associated *generator matrix* $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ that has full rank. There is an associated *parity check matrix* $\mathbf{H} \in \mathbb{F}_2^{(n-k) \times n}$ for the code such that $\ell \in \mathcal{C}$ if and only if $\ell \mathbf{H}^\top = 0$. The minimum-weight property and the existence of a parity check matrix for linear codes yields the following property [8].

Property 3. *Let* $\mathbf{H}$ *be a parity check matrix of an* (n, k, d) *linear code; then,* $\ell \mathbf{H}^\top \neq 0$ *for every* $\ell \neq 0$ *with* $\mathrm{w_H}(\ell) < d$*.*

5.3 Low-Degree WHT by Sampling With Codes

Assume that we want to find the WHT coefficients of $x \in \mathbb{R}^N$, where $N = 2^n$, and we know that the WHT coefficients X_ℓ for $\mathrm{w_H}(\ell) > t$ are zero for a given $t < n$. Hence, the number of non-zero coefficients can be at most $|\mathcal{K}_t^n|$, which could be much less than N depending on t. In order to reduce sampling and computation cost in recovering the non-zero coefficients, let us consider first subsampling and then performing the WHT on the subsampled signal. Consider using a parity check matrix $\mathbf{H}$ of an (n, k, d) linear code where $d > 2t$ to get sampled signal y of length 2^{n-k} as $y_m = x_{m\mathbf{H}}$ for $m \in \{0, \dots, 2^{m-k} - 1\}$. WHT of y is then obtained using $O\big((n-k)2^{n-k}\big)$ operations. Every WHT coefficient X_ℓ with $\ell \in \mathcal{K}_t^n$ will appear at a different WHT coefficient of y by the following property.

Property 4. *Let* $\mathbf{H}$ *be a parity check matrix of an* (n, k, d) *linear code with minimum distance* $d > 2t$*. For* $\ell \in \mathbb{F}_2^n$ *and* $m \in \mathbb{F}_2^n$*, with* $\ell \neq m$*, we have* $\ell \mathbf{H}^\top \neq m \mathbf{H}^\top$ *whenever* $\mathrm{w_H}(\ell) \leq t$ *and* $\mathrm{w_H}(m) \leq t$*.*

This follows from Property 3; as the minimum distance is $d > 2t$, the code that has $\mathbf{H}$ as its parity check matrix has no codeword with weight less than or equal to $2t$ (except for the all zero codeword). Hence, $(\ell - m)\mathbf{H}^\top \neq 0$, as $\mathrm{w_H}(\ell - m) \leq 2t$ whenever $\mathrm{w_H}(\ell) \leq t$ and $\mathrm{w_H}(m) \leq t$. This property implies that if the minimum distance d of a code satisfies $d > 2t$, then every binary string with Hamming weight less than or equal to t is mapped to a different vector when multiplied by $\mathbf{H}^\top$ on the right.

Example 3. Matrix $\mathbf{H}$ given in Example 1 is a parity check matrix of an $(n = 3, k = 1, d = 3)$ code. We can validate that the minimum distance of the code is 3 because the columns of the parity check matrix $\mathbf{H}$ are the non-zero strings of length 2, so no two columns are equal to each other, but sum of all 3 columns equals to 0. As $d > 2t$ for $t = 1$, coefficients X_ℓ for $\ell \in \mathcal{K}_{t=1}^{n=3}$ alias to different coefficients in Y. This can be seen to be true from the equations in (5.7). ■

For $\ell \in \mathcal{K}_t^n$, in order to find the WHT coefficient X_ℓ, we read off the WHT coefficient $Y_{\ell\mathbf{H}^\top}$. Hence, we need a mapping between $\mathcal{K}_t^n$ and the strings $\{u : u = \ell\mathbf{H}^\top, \ell \in \mathcal{K}_t^n\}$. This mapping can be calculated in $O(|\mathcal{K}_t^n|\,(n-k))$ time, because given the value of $u\mathbf{H}^\top$ for some u, we can calculate $v\mathbf{H}^\top$ for v satisfying $\mathrm{w_H}(u+v) = 1$ in $O(n-k)$ time. Hence, we can read off all the aliased indices with $O(|\mathcal{K}_t^n|\,(n-k))$ operations.

Combining the complexity results for taking the WHT and reading off the coefficients, the overall computational complexity becomes $O((n-k)2^{n-k})$. The complexity is dominated by taking WHTs as $2^{n-k} \geq |\mathcal{K}_t^n|$ for an (n, k, d) code with $d > 2t$, which follows from the well-known Hamming bound [8].

Our algorithm to compute the WHT of an $N = 2^n$ length x, if its WHT coefficients X_ℓ for $\ell \notin \mathcal{K}_t^n$ for some t are zero, can be summarized as follows:

1. Choose an (n, k, d) linear code with $d > 2t$, and let $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ be its parity check matrix,
2. Subsample x to get $y_m = x_{m\mathbf{H}}$ for 2^{n-k} values of $m \in \mathbb{F}_2^{n-k}$,
3. Take the WHT of y to get Y,
4. Read off the WHT coefficient X_ℓ, where $\ell \in \mathcal{K}_t^n$, from $Y_{\ell\mathbf{H}^\top}$.

Remark 1. In deriving our algorithm and the results, we assumed that the support of the WHT coefficients are exactly on $\mathcal{K}_t^n$. The algorithm can be made robust to accommodate significant coefficients outside this set or sparsity within this set by assigning modulation patterns to the WHT coefficients of the input signal by introducing shifts (see Property 2) before sampling. This idea is similar to [78, 49], where the authors use these modulation patterns to recover the indices of the WHT coefficients of the input signal from WHT coefficients of the subsampled signal. In Section 5.5.2 of this chapter, we present a variation of our method for such settings.

In contrast to our algorithm, a naïve approach to computing the WHT coefficients in $\mathcal{K}_t^n$ would be by inverting a linear system. Because there are $|\mathcal{K}_t^n|$ many unknowns that can be non-zero, we can obtain $|\mathcal{K}_t^n|$ samples of the input signal to get an invertible linear system. Solving such a linear system requires $O(|\mathcal{K}_t^n|^2)$ operations in general. Our algorithm can be seen as obtaining samples at carefully designed locations, possibly more than the minimum amount necessary, so that the resulting linear system can be inverted efficiently.

It is useful to compare the computational complexity of our proposed method to that of the naïve approach. For that, we are going to use an existence result. Gilbert-Varshamov bound states that for every $0 \leq \delta < 1/2$, and $0 < \epsilon \leq 1 - h(\delta)$, there exists a linear code with rate $R \geq 1 - h(\delta) - \epsilon$, and relative distance δ [89]. Which means that for $\epsilon > 0$, $\delta \in (0, 1/4)$, and for sufficiently large n, there exists an $\mathbf{H}$ with size $n(h(2\delta)+\epsilon) \times n$ with minimum distance greater than $2\delta n$. This enables our algorithm to recover coefficients in $\mathcal{K}_{n\delta}^n$. The ratio of the computational complexity of the naïve algorithm to that of ours, order-wise, is

$$\frac{|\mathcal{K}_t^n|^2}{n(h(2\delta)+\epsilon)2^{n(h(2\delta)+\epsilon)}} \gtrsim \frac{2^{2nh(\delta)}}{n^3(h(2\delta)+\epsilon)2^{n(h(2\delta)+\epsilon)}} = O\big(2^{n(2h(\delta)-h(2\delta)-\epsilon)}\big)\,. \tag{5.10}$$

Since $2h(\delta) - h(2\delta) > 0$, for each $\delta \in (0, 1/4)$ fixed, we can choose an appropriate $\epsilon > 0$ and get savings in the computational complexity that is exponential in n.

Even though the result above is enough to show the existence of good codes for our algorithm, in some settings, we can get savings that are far better. Below we provide a concrete example.

5.3.1 An Illustrative Example

Say $n = 23$, and in the WHT domain, only coefficients whose indices have Hamming weight less than or equal to 3 are non-zero. By the polynomial interpretation of the WHT in equation (5.3), this corresponds to having non-zero coefficients at multinomial terms that have degree less than or equal to 3. Counting the number of such terms, we have

$$|\mathcal{K}_3^{23}| = \sum_{i=0}^{3} \binom{23}{i} = 2^{11} \tag{5.11}$$

many of them.

Now, choose a subsampling matrix $\mathbf{H} \in \mathbb{F}^{11\times 23}$. We know from Property 1 that a coefficient with index $j \in \mathbb{F}_2^{23}$ will be aliasing to the coefficient at index $\ell \in \mathbb{F}_2^{11}$ where $\ell = j\mathbf{H}^\top$. By the assumption on the signal, we know that the index set that contains non-zero coefficients are $\mathcal{K}_3^{23}$. From equation (5.11), it follows that $|\mathcal{K}_3^{23}| = 2^{11}$, and the number of bins is also equal to 2^{11}. An interesting question is how are the WHT coefficients of the input signal distributed over the 2^{11} WHT coefficients of the subsampled signal.

Now, choose $\mathbf{H}$ as a parity check matrix of the $(23, 12, 7)$-Golay code [28]. Since the minimum distance is 7, it means that every binary string of length 23 with weight up to and including 3 must map to a different vector when multiplied by $\mathbf{H}^\top$ (cf. Property 4). There are 2^{11} output vectors and there are 2^{11} input strings which map to a unique vector, it must be that each vector gets assigned exactly one such string. This is equivalent to each element of $\mathcal{K}_3^{23}$ mapping to a unique index when we subsample the input signal using this particular $\mathbf{H}$.

Combining the above ideas, the algorithm to recover the WHT if the non-zeros are over $\mathcal{K}_3^{23}$ defined as above is as follows:

1. Subsample x using a parity check matrix $\mathbf{H}$ of the $(23, 12, 7)$-Golay code, and get $y_m = x_{m\mathbf{H}}$, for 2^{11} values of m,

2. Take the WHT of y which is of length 2^{11},

3. Read off X_ℓ, for $\mathrm{w_H}(\ell) \leq 3$, from $Y_{\ell\mathbf{H}^\top}$.

5.4 Choice of Codes

In this section, we analyze the sample and computational complexity of recovering low-degree WHT coefficients by using codes from particular families.

5.4.1 Perfect Codes

The subsampling matrix used in the illustrative example in the previous section and in Example 1 are called *perfect codes*.

Definition 6 (Perfect codes)**.** A binary linear $(n, k, d = 2t + 1)$ code is perfect, if

$$\sum_{i=0}^{t} \binom{n}{i} = 2^{n-k}. \tag{5.12}$$

This definition says that the balls of radius t around the 2^k codewords perfectly cover the whole space of binary strings of length n. While recovering coefficients from $\mathcal{K}_t^n$, because $2^{n-k} = |\mathcal{K}_t^n|$ for a perfect code, the number of samples we take from the input signal is equal to the number of possible non-zero coefficients. This makes the sample complexity optimal, and the computational complexity $O(|\mathcal{K}_t^n| \log(|\mathcal{K}_t^n|)$. However, it is well-known that any nontrivial perfect code (over any finite field) must have the same parameters n, k, and d as one of the Hamming or Golay codes [8]. Hence, the application of such codes to recovering low-degree WHT coefficients is limited. Next, we demonstrate that several well-known families of binary codes can be used to cover a wider range of parameters.

5.4.2 Bose-Chaudhuri-Hocquenghem Codes

Bose-Chaudhuri-Hocquenghem (BCH) codes are a class of binary linear codes [31, 9]. Given integers $r \geq 3$ and $t < 2^{r-1}$, there exists a BCH code with [8]:

$$n = 2^r - 1, \tag{5.13}$$

$$n - k \leq rt, \tag{5.14}$$

$$d_{\text{min}} \geq 2t + 1. \tag{5.15}$$

Combining (5.13) and (5.14), we get $n - k \leq t \log_2(n + 1)$.

Say we want to recover the WHT coefficients at $\mathcal{K}_t^n$ by subsampling the input signal at points given by the parity check matrix of a BCH code, then we will take $2^{n-k} \leq (n + 1)^t$ samples of the signal. We have

$$|\mathcal{K}_t^n| \geq \binom{n}{t} \geq \left(\frac{n}{t}\right)^t. \tag{5.16}$$

Hence, the oversampling factor, that is, the ratio of the number of samples taken to the degrees of freedom in the signal, satisfies

$$\frac{2^{n-k}}{|\mathcal{K}_t^n|} \leq \frac{(n + 1)^t}{(n/t)^t} \leq \left(\frac{8t}{7}\right)^t, \tag{5.17}$$

where $8/7$ comes because $n \geq 7$ by the assumption $r \geq 3$. Hence, whenever $t = O(1)$, we get a constant oversampling factor. Even though we get a constant oversampling in this setting, the bound in (5.17) can be large. However, this upper bound is observed to be loose in many settings. Table 5.1 shows operating points of BCH codes along with resulting oversampling factors. We observe that they are significantly smaller than the bound in equation (5.17).

Table 5.1: Sample operating points for BCH codes (adapted from [13]).

n	k	t	$2^{n-k}/\left\|\mathcal{K}_t^n\right\|$
63	57	1	1.00
	51	2	2.04
	45	3	6.29
	39	4	26.33
	36	5	17.51
127	120	1	1.00
	113	2	2.02
	106	3	6.15
	99	4	25.15
	92	5	129.71

5.4.3 Random Linear Codes

Random linear codes satisfy the Gilbert-Varshamov bound with probability going to one exponentially fast in the blocklength [89]. Hence, we can use them to recover coefficients in $\mathcal{K}_{\delta n}^n$ for $\delta < 1/4$ with high probability. For this, choose $\mathbf{H} \in \mathbb{F}_2^{b\times n}$ where $b = n(h(2\delta) + \epsilon)$, at random uniformly over all binary parity check matrices of size $b \times n$. The probability of the code to have minimum distance less-than or equal to $2\delta n$ goes to zero exponentially fast in n. The oversampling factor becomes

$$\frac{2^b}{|\mathcal{K}_{\delta n}^n|} \leq (n+1)2^{n(h(2\delta)+\epsilon-h(\delta))} \leq (n+1)N^{h(2\delta)+\epsilon-h(\delta)}.$$

By the Gilbert-Varshamov bound, for every $0 < \epsilon$ and sufficiently large n, there exists a subsampling pattern giving an oversampling factor of $O(N^{h(2\delta)+\epsilon-h(\delta)})$ to recover WHT coefficients on $\mathcal{K}_{\delta n}^n$. Using a random linear code will be able to do this with probability going to 1 exponentially in n.

5.5 Learning Fitness Landscapes for DNA Repair Outcomes

In this section of the chapter we apply a variant of our algorithm (to be described in Section 5.5.2) for learning *fitness* landscapes for DNA repair outcomes after CRISPR-Cas9 double-strand breaks. After a double-stranded break, the DNA is repaired by the cell with possible insertions and deletions of nucleotides. Fitness, in our problem, captures the propensity of repair outcomes to satisfy a chosen criterion of interest, such as the probability of insertion of a certain base pair during repair.

Recent studies on site-specific double stranded breaks generated by the RNA-guided DNA endonuclease Cas9 have shown that DNA repair outcome following Cas9 cutting is nonrandom and consistent across experimental replicates, cell lines, and reagent delivery methods and highly a function of the DNA sequence around the cut site [65]. Follow-up studies employed machine learning models such as deep neural networks [81], decision trees [48], and logistic regression models [3] to train DNA repair models using a database of double stranded breaks generated in CRISPR-Cas9 experiments. These models take in the DNA sequence of the genome in a window surrounding the cut and predict key statistics of the repair outcomes such as the percentage of cells with certain frame shifts, insertions, and deletions.

While initial studies have shown the applicability of these machine learning models in designing gene knock-out [48] and knock-in [81] experiments, their applicability is limited due to the lack of a reliable, interpretable, and scalable machinery to train and analyze the DNA repair models.

Reliability. Currently, there is no clear mechanism to determine the number of site-specific CRISPR experiments required for training DNA repair models reliably. The number of experiments is typically defined by the total budget invested in the experiments rather than a principled scientific mechanism backed by theoretical reasoning. In addition, the location of cut sites (guide RNAs) on the genome are determined either randomly rather than experimental design procedures tailored for the repair models.

Interpretability. Our understanding of how the current DNA repair models operate is limited; for example, we do not have a clear mechanism to determine what features or combination of features enrich for key repair outcomes such as frame shifts, insertions, and deletions. Interpretable models would enable an understanding of the repair process as well as a set of design rules for gene knock-out and knock-in experiments.

Scalability. Models for DNA are slow in the inference time. Considering the growing interest in using CRISPR-Cas9 in different cell types, there is a critical need for more scalable methods at the inference time.

Here, we aim to address these problems by analyzing the landscape of the DNA repair process in the *spectral domain* by using our sparse WHT recovery algorithm. The key insight that enables our framework is that the DNA repair model, in its full generality, can be formulated by a pseudo-Boolean function $f(z_1, z_2, \ldots, z_n)$: $\{-1, 1\} \to \mathbb{R}$, where $z_i \in \{-1, 1\}$ are binary variables that encode the input DNA sequence in a window of length L surrounding the cut, and $f(\mathbf{z})$ is a real-valued outcome of interest, for example, percentage of insertions. This function admits a polynomial (multinomial) representation as in equation 5.3. From a biological perspective, the index of the non-zero coefficients in the WHT domain reveal high-order interactions between the input features that enrich (or deplete) certain DNA repair outcomes. The interaction order depends on the number of "1"s in the coefficient index (degree of the coefficient) as it corresponds to a multinomial term with that many variables (see Section 5.2 for this interpretation). Our goal in molecular biology is to efficiently estimate these coefficients from the underlying biological process. The main challenge is that the scale of the problem grows exponentially with n as there are 4^n different configurations for n nucleotides. The key observation that enables the recovery of the coefficients is that the DNA repair landscape is sparse in the WHT domain (see Figure 5.4). Therefore, the problem reduces to recovering the WHT coefficients (equivalently, the pseudo-Boolean function) when there is sparsity in the WHT domain. While methods proposed in compressed sensing literature, such as LASSO [86], OMP [87], or FISTA [7], can be used to recover a sparse signal in a *sample efficient* way, their *computational complexities* scale at least linearly with the ambient dimension N. This makes these well-known algorithms infeasible for use, as even for modest value of the window size, e.g., $n = 40$, the dimension of the problem N becomes too large to handle.

5.5.1 Contributions

The variant of our method discussed in this chapter provides an efficient algorithm that employs a divide-and-conquer strategy which breaks the problem of recovering a K-sparse signal into K-many smaller problems of recovering 1-sparse signal, and solve each 1-sparse problem efficiently, and combine their solutions to recover the original signal. The recovery algorithm is closely tied to decoding a sparse-graph-code by peeling, and we use techniques from the literature on Low Density Parity Check (LDPC) codes [76] and product codes [23] for designing and analyzing our algorithm. Our contributions for learning the fitness landscape of DNA repair outcomes can be summarized as follows.

- **Approach:** We analyze the fitness landscape of the DNA repair outcome after a CRISPR double strand break in the WHT domain. We show that the landscape is highly sparse, and the sparsity level depends on the type of the repair outcome.
- **Methodology:** Our method is the first machine-learning framework that leverages the sparsity of DNA repair landscape in the WHT spectral domain.
- **Interpretation:** By looking at which coefficients are active in the WHT domain, our method provides insights into molecular biology by extracting higher-order interactions among the input features.
- **Scalability:** Our method is scalable to high dimensions as computational complexity scales linearly with the inherent degrees of freedom of the signal (sparsity in the WHT domain) and has a week dependence on the ambient dimensions (polylogarithmic).

5.5.2 Learning Framework

In Section 5.3, we described a method for recovering *all* of the low-degree WHT coefficients by choosing an appropriate subsampling matrix. It turns out, many functions of interest, such as fitness landscape of DNA repair outcomes, the WHT is *sparse* on top of being low degree and can possibly have a small number active high-degree terms. Hence, we develop a variation of our algorithm where we can leverage this sparsity to reduce down the sample and computational complexity beyond that of described in Section 5.3 and handle higher-degree terms. To do this, we combine our subsampling strategies with peeling-based algorithms for sparse WHT [50, 78, 51], which are algorithms similar in flavor to the sparse DFT algorithm described in Chapter 3. These algorithms assumes that the support in the WHT domain is chosen uniformly at random over subsets of size K

of N elements. In particular, we have the following theorem for the SPRIGHT algorithm (SParse Robust Iterative Graph-based Hadamard Transform) from [51].

Theorem 7 (Adapted from Theorem 1 from [51])**.** *Suppose that $N = 2^n$ and assume that $K = N^\alpha$ for a fixed $\alpha \in (0, 1)$. Let $x \in \mathbb{R}^N$ be a vector, and $X \in \mathbb{R}^N$ be its WHT. Assume that X is K-sparse and its support is selected uniformly at random among all possible subsets of size K. Then, SPRIGHT algorithm has the following properties:*

1. *Sample complexity: SPRIGHT uses $O\big(K \log^2 N\big)$ samples of x.*
2. *Computational complexity: Total number of operations to successfully decode all nonzero WHT coefficients or declare a decoding failure is $O\big(K \log^3 N\big)$.*
3. *Success probability: Probability of recovering X completely approaches to 1 as N grows, where the probability is taken over randomness of the support of X.*

Similar to the sparse DFT architecture shown in Figure 3.1, SPRIGHT first subsamples the signal and then takes short WHTs. This creates linear mixing of WHT coefficients through Property 1 and Property 2. Then, the algorithm proceeds to recover the WHT coefficients of the original input through fast *peeling* based on the aliased measurements. Algorithm 3 provides a pseudocode for the overall procedure.

Algorithm 3: Sparse-WHT Learning Algorithm

```
Input  : Function $x_m$ for $m \in \mathbb{F}_2^n$, number of subsampling groups $C$, number of shifts per
         subsampling group $P$
Output : WHT support and coefficient values $\mathcal{S}$
$Y, H, \mathcal{P} \leftarrow$ Subsample$(x, C, P)$
$\mathcal{S} \leftarrow \varnothing$
/* go over each digit */
for $i \leftarrow 1$ to $I$ do
    /* go over subsampling groups */
    for $c \leftarrow 1$ to $C$ do
        /* go over check-nodes */
        for $\ell \in \mathbb{F}_2^b$ do
            (type, $\hat{\ell}, \hat{v}$) = ClassifyNode$(Y_\ell^{(c)}, \mathcal{P})$
            if type is singleton then
                $\mathcal{S} \leftarrow \mathcal{S} \cup (\hat{\ell}, \hat{v})$
                Peel$(\hat{\ell}, \hat{v}, H)$
            end
        end
    end
end
```

When the sparsity is not uniformly distributed over size K subsets of N, but over low-degree terms, the induced sparse-graph using the subsampling algorithm proposed in [51] does not produce the guarantees of Theorem 7, and its performance degrades significantly. The reason is that, under that subsampling scheme, most of the low-degree terms alias together at the same check nodes, and the peeling algorithm cannot progress because of the lack of singletons (see Section 1.1 and Figure 1.4 for the importance of singletons for the peeling decoder). Hence, to handle support over the low-degree terms, the subsampling matrices need to be designed guided by the ideas described in Section 5.4. Example 4 depicts a setting showing how the subsampling matrices affect peeling decoder when sparsity is distributed over the low-degree terms.

Example 4. Example 1 used the subsampling matrix

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix},$$

which had each WHT coefficient with degree less than or equal to 1 alias to a separate check node. If, instead, we subsample using a single row of this matrix, say the first row of $\mathbf{H}$

$$\mathbf{H}_1 = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix},$$

then $y_m^{(1)} = x_{m\mathbf{H}_1}$ has WHT equal to

$$\begin{aligned} Y_0^{(1)} &= X_{000} + X_{010} + X_{101} + X_{111}, \\ Y_1^{(1)} &= X_{001} + X_{100} + X_{011} + X_{110}. \end{aligned}$$

As can be seen, each bin has exactly 2 coefficients with degree less than or equal to 1. On the other hand, if we use a subsampling matrix chosen as one of the rows of the identity matrix of size 3, say

$$\mathbf{H}_2 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix},$$

then $y_m^{(2)} = x_{m\mathbf{H}_2}$ has WHT equal to

$$\begin{aligned} Y_0^{(2)} &= X_{000} + X_{001} + X_{010} + X_{011}, \\ Y_1^{(2)} &= X_{100} + X_{101} + X_{110} + X_{111}, \end{aligned}$$

which shows that 3 out of 4 of the WHT coefficients with degree less than or equal to 1 alias to check node 0. Whenever the support is uniformly distributed over the low-degree terms, the peeling decoder has a higher probability of failing to recover all of the variable nodes when $\mathbf{H}_2$ is used for subsampling compared to $\mathbf{H}_1$. For example, if sparsity if $K = 2$ with support distributed over the term of X with degree $t \leq 1$, the peeling decoder has $2/3$ chance recovering the full support when subsampling with $\mathbf{H}_1$ compared to $1/2$ with $\mathbf{H}_2$. Note that these probabilities are for this example only, and the gap between them grow with problem size (see the plot in Figure 5.2 for an example). ■

The example provides a small setting showing the importance of the choice of the subsampling matrices. Here, we show the results of a larger numerical experiment depicting how the performance gap grows with problem size. In the numerical experiment $n = 24$, hence $N \approx 16$ million, number of check nodes $M = 768$, sparsity is $K = 548$, and the support is uniform over the low-degree terms with $t \leq 5$. We use two subsampling schemes. First one is the 'canonical design' where we subsample the input signal using subsets of the rows of the identity matrix of size 24×24 (as is done in [51]), and the second one is the 'random design' where we subsample the input signal using random matrices. Let R be the fraction of variable nodes recovered in a run of the experiment. Figure 5.2 shows the empirical $\Pr(R \geq r)$ we obtain from 500 random experiments from this setting. As can be seen, the 'random design' allows for the recovery of all variable nodes with high probability, while the 'canonical design' fails to recover more than $r = 0.2$ fraction of the variable nodes.

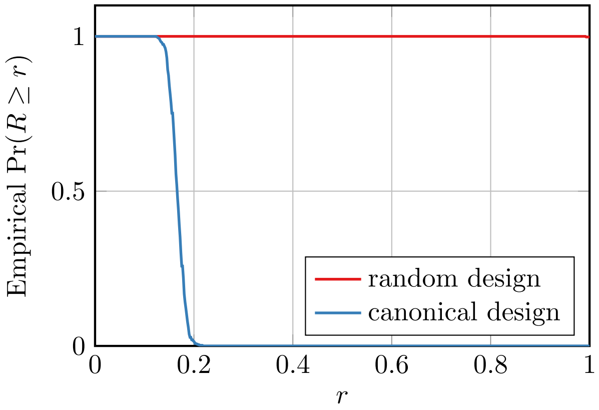


Figure 5.2: Recovery of variable nodes using subsampling matrices from 'random design' and 'canonical design'. The setting is where $n = 24$, $N \approx 16$ million, the number of check nodes $M = 768$, sparsity is $K = 548$, and the support of WHT coefficients is uniform over the low-degree terms with $t \leq 5$. The 'canonical design' subsamples the input signal using subsets of the rows of the identity matrix, and the 'random design' subsamples the input signal using random matrices. R denotes the fraction of variable nodes recovered. We can see that the 'random design' allows for the recovery of all variable nodes with high probability, while the 'canonical design' fails to recover more than $r = 0.2$ fraction of the variable nodes.

In light of the discussion above, we use random matrices for subsampling the input signal as in Algorithm 4 (in the appendix to this chapter). The peeling process works as described in Section 1.1 where for each recovered variable node (coefficient), we strip its contributions from the check nodes it corresponds to as shown in Algorithm 5 (in the appendix to this chapter). This assumes that we have a mechanism to detect singleton check nodes, and recover the location and value of the variable node connected to them. One way to do this is by using $O\big(\log^2 N\big)$ shifts for each subsampling group where $\log(N)$ shifts are to recover each digit of the location ℓ, and we take $O(\log N)$ samples for each location for noise averaging. The ℓth WHT coefficient of the signal obtained by shifting indices of x by p and then subsampling by $\mathbf{H}$ is equal to

$$U_{\mathbf{H},p}(\ell) := \sum_{j:j\mathbf{H}^{\top}=\ell} (-1)^{\langle j,p\rangle} X_j, \tag{5.18}$$

which follows from combining Property 1 and Property 2. Furthermore, let us define the ratio of a WHT coefficient obtained by using the same subsampling matrix but using two different delays p and q as

$$r_{\mathbf{H},p,q}(\ell) := \frac{U_{\mathbf{H},p+q}(\ell)}{U_{\mathbf{H},p}(\ell)}. \tag{5.19}$$

Assume that for a transform index ℓ, there is only one index j such that $\mathbf{H}^{\top} j = \ell$ and $X_j \neq 0$ in equation (5.18) (that is, the check node corresponding to it is a single-ton). Then, it follows that $U_{\mathbf{H},p}(\ell) = (-1)^{\langle j,p\rangle} X_j$. Using $q = e_i \in F^n$ (the vector with all indices equal to 0 except for the ith index which is equal to 1) in equation (5.19) then yields

$$r_{\mathbf{H},p,e_i}(\ell) = \frac{(-1)^{\langle j,p+e_i\rangle} X_j}{(-1)^{\langle j,p\rangle} X_j} = (-1)^{\langle j,e_i\rangle}. \tag{5.20}$$

Note that this value is in $\{-1,+1\}$ for all p if there is no noise. As the value of $\langle j, e_i\rangle$ is equal to the ith index of the location $j \in \mathbb{F}_2^n$, by using shifts $\{e_i\}_{i=0}^{n-1}$ going through all indices of j we can recover it. When there is noise, it can be shown that by taking $O(\log N)$ random shifts and using a threshold τ that depends on the noise level, the probability of detecting a single-ton and finding its location wrongly can be made polynomially small [51]. The pseudocode for processing check nodes is shown in Algorithm 6 in Appendix 5.A.

Next we show the results we get by applying this algorithm to learning fitness landscapes of DNA repair outcomes.

5.5.3 DNA Repair Landscapes

We will consider two DNA repair landscapes of scientific interest: insertion percentage and in-frame shift [4] percentage. In order to compute the fitness landscape, we first generate all the 4^L possible binary codewords $\mathbf{m}$ of length $n = 2L$. We then decode the binary code words into their corresponding DNA sequences through the following mapping: A:00, T:01, C:10, and G:11. With this encoding, we can represent a DNA sequence of length L using a binary code $\mathbf{m}$ of length $n = 2L$.

We then construct CRISPR experiments where we break the genome exactly in the middle of the DNA sequences constructed as described above. We repeat the same experiment for all the 4^L

[4] Amino acids in the DNA are represented by triplets where consecutive three nucleotides correspond to an amino acid in the synthesized protein. An indel (insertions or deletions) that is not divisible by three can significantly can change the grouping of the nucleotides, resulting in a completely different sequence of amino acids in the synthesized protein. By in-frame shift, we denote indel lengths divisible by three.

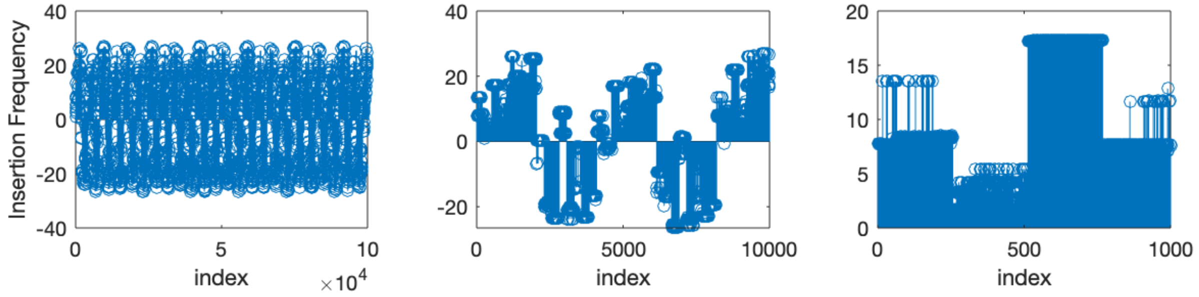


Figure 5.3: The fitness landscape of DNA repair outcomes in U2OS (human bone Osteosarcoma epithelial cells) in terms of the insertion percentage. Only the first $100\,000$, $10\,000$, and $1\,000$ coordinates of the landscape are illustrated for clarity. Insertion percentages are mean-subtracted. The landscape shows redundant structures in different resolutions.

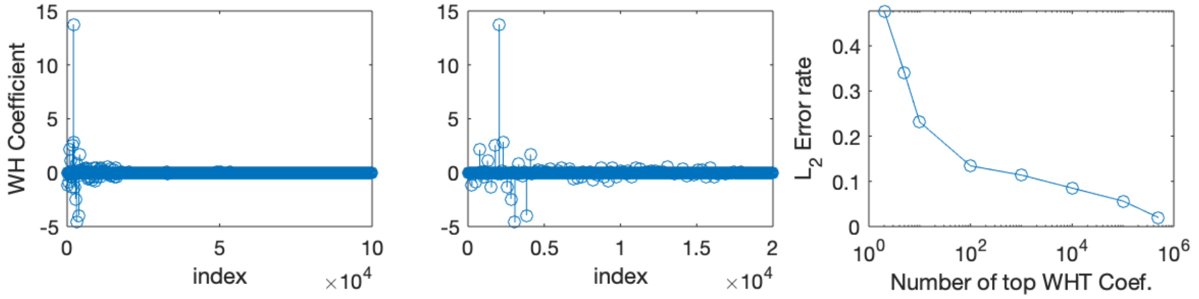


Figure 5.4: The WHT of the repair landscape for insertion percentage in U2OS. Only the first $100\,000$ and $20\,000$ coordinates of the landscape are illustrated for clarity. Right-most plot shows the ℓ_2 error in recovering the landscape using the top coefficients of the WHT transform.

generated DNA sequences and use a trained repair model (e.g., inDelphi [81] or SPROUT [48]) to find the repair outcome for all the generated DNA sequences. We then store the key summary statistics (outputs) of the DNA repair outcome that we want to learn, that is, in-frame shift percentage and insertion percentage. We represent each summary statistics in a 4^L-dimensional vector in the same order as they appear in the binary code $\mathbf{m}$; we call these vectors the *fitness landscape* of the DNA repair outcomes. These landscapes fully describe the DNA repair models in terms of the key statistics.

An example of the fitness landscape in terms of the insertion percentage is illustrated in Figure 5.3 for window length $L = 10$ in U2OS (human bone Osteosarcoma epithelial cells). The figure illustrates the mean subtracted insertion percentage of the first $100\,000$, $10\,000$, and $1\,000$ coordinates of the $4^{10} = 1\,048\,576$-dimensional landscape. As can be seen from the figure, the insertion percentage displays structure.

We applied the WHT on the resulting 4^{10}-dimensional landscape signal. The spectrum of insertion percentage and in-frame shift percentage are depicted in Figures 5.4 and 5.5 respectively. In order to measure how sparse the WHT transform is, we plot the recovery error of the inverse

WHT using only the top components of the WHT transform in terms of the ℓ_2 norm. Our key observation is that the WHT transform of the landscape is sparse; only the top-100 coefficients out of the total of 4^{10} coefficients ($< 0.01\%$) suffices to recover the insertion percentage landscape with around 10% error. The WHT of the in-frame shift landscape is more wide spread and shows coefficients that are periodically repeated along the spectrum. The spectrum is still sparse; however, it is more dense compared to the insertion percentage. More than $10\,000$ WHT coefficients out of the total of 4^{10} coefficients ($\approx 1\%$) are required to recover the landscape with around 10% error. This suggests that the information required to estimate the in-frame shift percentage is more widespread compared to insertion percentage, which is more localized to the nucleotides around the cut site. Our finding in Figures 5.4 and 5.5 also suggest a bound in terms of the number of samples required to approximate the function $x[\mathbf{m}]$.

Knowing that the landscape of the repair outcomes can be recovered using only a small fraction of the WHT coefficients, a natural question is what biological features the top WHT coefficients corresponds to. Each WHT coefficient $\alpha_{\mathcal{S}}$ corresponds to a monomial defined by $\mathcal{S}$. Table 5.2 shows the index of the top-5 WHT coefficients, their corresponding binary representation, monomial and a short interpretation of each monomial. Each code in the second column is the binary representation of the index written in $n = 2L = 20$ bits. The monomial is obtained based on the pattern of "1"s in the binary code. We can write the repair model $x[\mathbf{m}]$ in terms of the first five monomials as, $x[\mathbf{m}] \approx 13.76z_9 - 4.61z_9z_{10} - 3.96z_9z_{10}z_{11}z_{12} + 2.84z_9z_{12} + 2.56z_{10}z_{11}z_{12}$, where $z_i = (-1)^{m_i}$. Note that the cut site is in between z_{10} and z_{11} and is indicated by a straight line in Table 5.2.

Table 5.2: Top-5 WHT coefficients of the repair landscape for percentage of insertions in U2OS cells. WHT coefficients correspond to monomials defined by their binary expansions. Each monomial can be interpreted as a question regarding the type of the nucleotides around the cut site.

index	code	monomial	interpretation	coefficient
2048	00 00 00 00 10 | 00 00 00 00 00	z_9	*{C,G}^ {A,T}@-1?*	13.76
3072	00 00 00 00 11 | 00 00 00 00 00	z_9z_{10}	*-1 bp?*	-4.61
3840	00 00 00 00 11 | 11 00 00 00 00	$z_9z_{10}z_{11}z_{12}$	*-1 & +1 bp?*	-3.96
2304	00 00 00 00 10 | 01 00 00 00 00	z_9z_{12}	*{C,G}^{A,T}@-1 & {T,G}^{A,C}@+1?*	2.84
1792	00 00 00 00 01 | 11 00 00 00 00	$z_{10}z_{11}z_{12}$	*+1 bp? {T,G}^{A,C}@-1?*	2.56

To better understand the interpretation of each monomial, we describe the biological meaning of the first monomial z_9, the rest follows from the same intuition. The monomial $z_9 = (-1)^{m_9}$ only considers the 9th binary digit. Recall that we encode the nucleotides using the following encoding: A:00, T:01, C:10, and G:11. The monomial z_9 is related to the nucleotide next to the cut on the left hand side (we call it location -1), and has the value -1 if the nucleotide is a C or a G (since these are the nucleotides that their binary code begin with the digit 1), and has the value 1 if the nucleotide is an A or a T. Therefore, this monomial is only asking if the nucleotide at location -1 is C/G or A/T. We represent this question using the logical statement *{C,G}^{A,T}@-1?* where $\wedge$ is an OR operator and @ points to a location on the genome. This feature has been shown in the biology

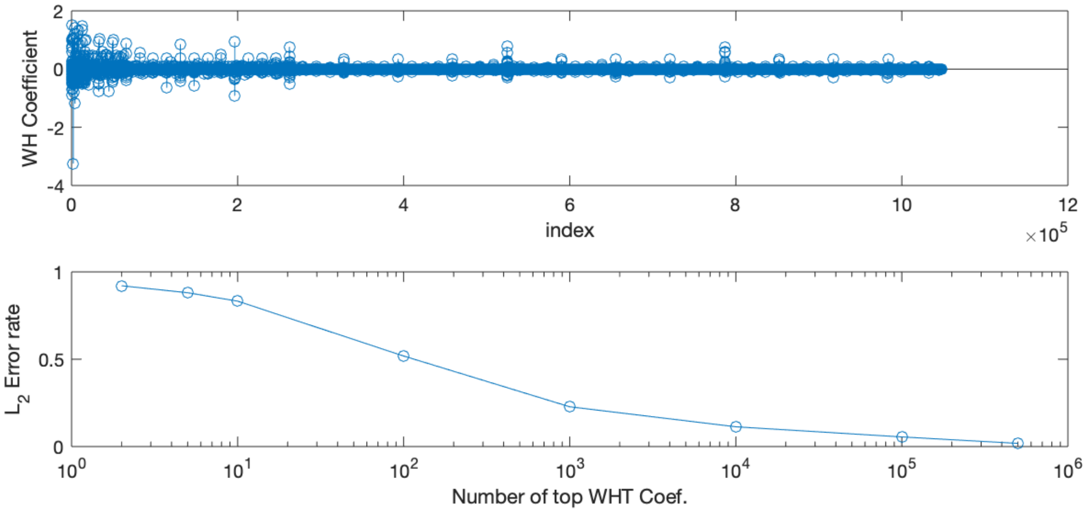


Figure 5.5: The WHT of the repair landscape for in-frame percentage in U2OS. The WHT of in-frame percentage is more dense than that of the insertion percentage. This suggests more number of samples is required to recover the landscape of in-frame percentage compared to insertion percentage.

Table 5.3: Top-5 WHT coefficients of the CRISPR fitness landscape when the output is set to the in-frame shift percentage of the repair outcome distribution in U2OS cells. WHT coefficients correspond to monomials defined by their binary expansions. The monomials with * sign indicates a microhomology feature, which are identical patterns repeating around the cut site.

index	code	monomial	interpretation	coefficient
2048	00 00 00 00 10 | 00 00 00 00 00	z_9	*{C,G}^{A,T}@-1?*	-3.24
256	00 00 00 00 00 | 01 00 00 00 00	z_{12}	*{T,G}^{A,C}@+1?*	1.50
12288	00 00 00 11 00 | 00 00 00 00 00	$z_7 z_8$	*-2 bp?*	1.48
4160	00 00 00 01 00 | 00 01 00 00 00	$^* z_8 z_{14}$	*{T,G}^{A,C}@-2 & +2 bp?*	1.42
12480	00 00 00 11 00 | 00 11 00 00 00	$^* z_7 z_8 z_{13} z_{14}$	*-2 & +2 bp?*	1.36

literature to have a significant correlation with the percentage of insertions [81, 48]. The rest of the monomials can be interpreted similarly. Note that all of the top-5 monomials in the WHT transform of the insertion percentage fitness landscape is related to the nucleotides that are adjacent to the cut site. This shows that the nucleotides around the cut are sufficient to describe the DNA repair landscape in terms of the insertion percentage.

We tabulate the top-5 WHT coefficients α_S of the repair landscape of in-frame shift percentage in Table 5.3. Similar to the landscape of insertion percentage, we see that the first two monomials correspond to nucleotides that are next to the cut site. The last two monomials, however, show a new pattern. These two monomials ask about symmetric patterns that occur around the cut site. These features resemble the microhomology patterns, i.e., repeating sequences around the cut site. Such higher order interactions between the features in the input space are very hard to capture using deep learning models.

5.5.4 Massive-scale Landscape Prediction

We now discuss the computational challenges in finding the full landscape of the repair outcomes when the sequence length goes beyond $L > 10$ and present the results of our algorithm. Testing a single DNA sequence using the inDelphi software [81] takes about 0.15 seconds on a 2.7 GHz Intel Core i5 with 8G of RAM at the inference time. Therefore, it takes around $4^{10} \times 0.15 = 157\,286$ seconds (close to 2 days) to obtain the full landscape of the repair outcome of each cell type *in silico* with $L = 10$ (assuming sequential processing of data). However, running the same inference problem for a longer sequence of length $L = 20$ (twice the length of out previous experiment) takes about 4^{10} times this (assuming sequential processing). This computation can be too costly in supercomputers today, let alone doing as experiments to obtain the landscape *in vitro*.

We do an analysis of DNA repair outcomes as a function of the window size around the cut site to find out for what values of L we can capture most of the variation in the repair outcomes. To do this, we consider a window considered around the cut size with varying length. The DNA sequence in the window is varied and the maximum range that the DNA repair outcome changes is monitored as a function of the window size (nucleotide on each side of the cut) and plotted in Figure 5.6. While knowing only the 3 nucleotides around the cut site (window of length 6) reduces the range of variation for insertion percentage below 5%, we need to know at least 20 nucleotides (a window of size 40) to reduce the range of variation for frame shift below 5%. This necessitates the need to develop a computational platform that scales to such large values of window lengths.

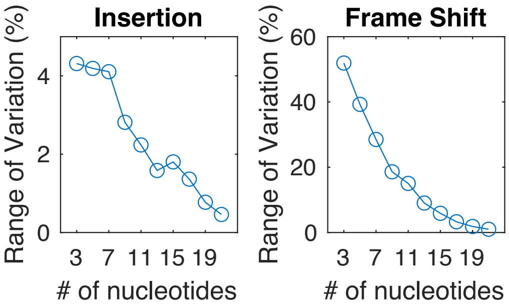
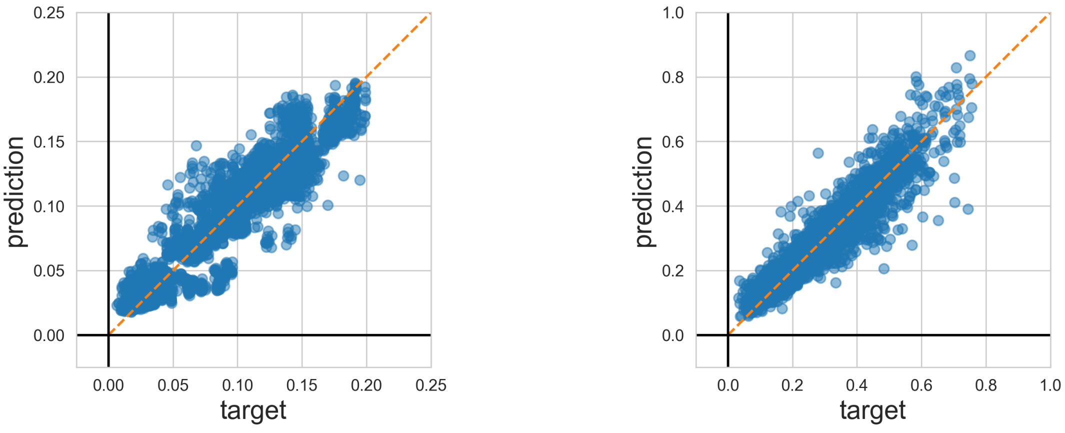


Figure 5.6: The variation of the DNA repair landscape as a function of the window size around the cut site (number of nucleotides from each side of the cut). For insertion percentage, a window of 3 nucleotides from each side is enough to capture the landscape with less than 5% variation. However, for in-frame shift, a window of 20 nucleotides from each side to get that level of precision.



(a) Insertion percentage, $R^2 = 0.90$. (b) Frameshift percentage, $R^2 = 0.88$.

Figure 5.7: The prediction results on $330\,000$ randomly chosen unseen test CRISPR experiments. The analysis has been done on the sparse insertion percentage landscape as well as the less sparse in-frame shift percentage landscape. Only $10\,000$ data points are plotted for more clarity. In both landscapes we predict the outcomes with very high accuracy.

We approximate the DNA repair landscape in terms of the insertion percentage and frame shifts in U2OS cells with a context window of size $L = 20$ using only about 3 million carefully designed guide RNAs (each corresponding to a sample). We design the input samples based on ideas from signal processing and coding theory. Note that, in general, the DNA sequences of all the guide RNAs suggested by our algorithm might not exactly appear on the human genome. In those cases, either a sufficiently close guide RNA can be selected or the recovery algorithm can be changed to accommodate for the deviation. Obviously, this will not be a problem when the repair outcomes are derived using a pretrained repair model.

Here, we evaluate the generalization performance of our method using $330\,000$ *unseen* samples from the DNA repair landscape. The results of our prediction algorithm can be seen in Figure 5.7. We predict the DNA repair outcome of the set of $330\,000$ unseen test guide RNAs with a very high accuracy ($R^2 \sim 0.9$) in both the easier insertion percentage and the harder in-frame shift percentage landscapes. Given the repair outcome of the 3 million designed guide RNAs (which takes about a week to acquire from inDelphi's software), our method requires only few seconds to recover the queried repair outcomes.

5.6 Hypergraph Sketching

Hypergraphs are a generalization of graphs to accommodate for edges with higher connectivities. A hypergraph $G = (V, E)$ is a collection of nodes $V = \{1, \ldots, n\}$ and a set E of hyperedges defined on these nodes. Each hyperedge $e \in E$ is a subsets of nodes V. The degree of a hyperedge is the cardinality of the set corresponding to that hyperedge. With this definition, a graph is a hypergraph where each edge has degree equal to 2.

The problem of hypergraph sketching is to recover a hypergraph from its cut values. A cut S on a hypergraph is a subset of its nodes $S \subseteq V$, and the cut value is equal to the number of hyperedges whose elements are split between S and $V \setminus S$, that is, $f_G(S) = |\{e : e \in E, S \cap e \neq \emptyset, S \cap e \neq e\}|$. Let us denote a cut by a vector $u \in \{-1, 1\}^n$ as $u_i = -1$ if $i \in S$, and $u_i = 1$ if $i \notin S$. Then we can represent the cut value function [50]

$$f_G(u) = \sum_{e \in E} \left(1 - \left(\prod_{i \in e} \frac{1 + u_i}{2} + \prod_{i \in e} \frac{1 - u_i}{2} \right) \right). \tag{5.21}$$

Letting $u_i = (-1)^{m_i}$, for $i \in \{1, \dots, n\}$, in the equation we get

$$x_m := f_G(u) = \sum_{\ell \in \mathbb{F}_2^n} X_\ell \cdot (-1)^{\langle \ell, m \rangle}, \tag{5.22}$$

which is in the form of a WHT relation. If the maximum degree of a hyperedge is t, then all the coefficients X_ℓ with $\mathrm{w_H}(\ell) > t$ will be 0, since in equation (5.21), each product will have at most t factors.

Table 5.4: Empirical rate of full recovery of a randomly-selected hypergraph with 1000 nodes and hyperedge degree 5 as a function of the number of hyperedges.

	number of hyperedges							
b	1	2	3	4	5	6	7	8
14	100	99.0	94.0	93.0	89.0	78.5	75.5	66.0
15	100	99.0	97.5	95.5	92.5	88.5	83.0	81.5
16	100	99.0	99.5	97.5	95.5	94.0	94.0	90.8
17	100	100	100	98.0	97.0	97.5	97.5	97.2
18	100	100	100	99.5	99.2	99.4	99.0	98.5

If the non-zero coefficients of the cut value function are recovered, the hypergraph is also recovered. We can use the parity check matrix of an $(n, k, 2t+1)$ code to recover the coefficients. When the number of hyperedges in the graph is small, there will further be sparsity over the coefficients in $\mathcal{K}_t^n$, and we run simulations to test our algorithm's generalization performance to this sparse setting. In the simulations $n = 1000$ and $t = 5$, and we sweep the number of hyperedges from 1 to 8, and select hyperedges uniformly at random. We draw $\mathbf{H}^{b \times n}$ from random linear code design and empirically evaluate the success rate of recovering the full hypergraph. Table 5.4 shows the probability of success as a function of b and number of hyperedges. As can be seen, our algorithm can leverage further sparsity on the low-degree set.

5.7 Conclusions and Future Directions

We presented a sample and computationally-efficient algorithm to recover pseudo-Boolean functions whose non-zero coefficients are over monomials having small degree. This problem is equivalent to recovering the WHT of a signal whose WHT coefficients are non-zero only at indices whose binary expansion has small number of ones. The algorithm we proposed first evaluates the input pseudo-Boolean function at points given by the codewords of a codebook. These codebooks are related to codes with good minimum distance properties. We then perform a WHT on the resulting signal, and read off the coefficients of the input pseudo-Boolean function from the resulting WHT coefficients. Further, we incorporated our subsampling methods for recovering low-degree WHT coefficients in peeling-based sparse WHT algorithm of [51], and showed that it improves the recovery performance when the support is distributed mostly on the low-degree terms. We applied this variant of the algorithm to learning fitness landscapes for DNA repair outcomes, and showed that we get interpretable models and fast inference. Future research directions include exploring other application areas that lend themselves to pseudo-Boolean functions such as learning epistasis (interactions of genes) in genotype-phenotype maps [77, 64].

Appendix

5.A Pseudocodes

In this section, we provide pseudocodes for the subroutines called in Algorithm 3 for learning fitness landscapes for DNA repair outcomes. Subsampling is given in Algorithm 4, peeling is described in Algorithm 5, and check node classification is provided in Algorithm 6.

Algorithm 4: Subsample (Subsampling and WHT)

```
Input  : Function $x_m$ for $m \in \mathbb{F}_2^n$, number of subsampling groups $C$, number of shifts per
         subsampling group $P$
Output : Observations $\{Y^{(c)}\}_{c=1}^C$, subsampling matrices $\{\mathbf{H}_c\}_{c=1}^C$, shifts $\{\mathcal{P}_c\}_{c=1}^C$
/* go over groups */
for $c = 1$ to $C$ do
    /* generate subsampling matrices */
    $\mathbf{H}_c \leftarrow \text{Uniform}(\mathbb{F}_2^{b \times n})$
    /* generate shifts */
    $\mathcal{P}_c \leftarrow \{\text{Uniform}(\mathbb{F}_2^n)\}_{i=1}^P$
    /* go over shifts */
    for $p \in \mathcal{P}_i$ do
        /* subsample the signal */
        $y_m^{(c,p)} \leftarrow x_{m\mathbf{H}_c + p}$
        /* calculate $2^b$-length WHT */
        $Y_\ell^{(c,p)} \leftarrow \frac{1}{B} \sum_{m \in \mathbb{F}_2^b} y_m^{(c,p)} (-1)^{\langle m, \ell \rangle}$
    end
end
```

Algorithm 5: Peel (Single step of peeling)

```
Input  : Location $\ell$, value $v$ to peel from the measurements, subsampling matrices $\{\mathbf{H}_c\}_{c=1}^C$
/* go over subsampling groups */
for $c = 1$ to $C$ do
    /* the coefficient $\ell$ aliases to $\ell_c$ in subsampling group $c$
       (see Property 1) */
    $\ell_c \leftarrow \ell \mathbf{H}_c^\top$
    /* go over shifts */
    for $p \in \mathcal{P}_c$ do
        $Y_{\ell_c}^{(c,p)} \leftarrow Y_{\ell_c}^{(c,p)} - v(-1)^{\langle p, \ell \rangle}$
    end
end
```

Algorithm 6: ClassifyNode (Classification of check-nodes)

```
Input   : Measurements $Y^{(c)}[\ell]$, shifts $\mathcal{P}$, threshold $\tau$
if $\|Y^{(c)}[\ell]\| < \tau$ then
    declare zeroton
end
$\hat{\ell} \leftarrow$ all zeros of length $n$
/* go over each bit */
for $i = 1$ to $n$ do
    $\hat{s}_i \leftarrow 0$
    /* go over shifts */
    for $p \in \mathcal{P}_c$ do
        $\hat{s}_i \leftarrow \hat{s}_i + \mathrm{sign}(Y^{(c,p)})\,\mathrm{sign}(Y^{(c,p+e_i)})$
    end
    $\hat{\ell}_i \leftarrow (1 - \mathrm{sign}(\hat{s}_i))/2$
end
$\hat{v} \leftarrow \frac{1}{P} \sum_{p \in \mathcal{P}_c)} Y^{(c,p)} (-1)^{\langle \hat{\ell}, p \rangle}$
$r_p \leftarrow Y^{(c,p)} - \hat{v}(-1)^{\langle \hat{\ell}, p \rangle}$
if $\|r\| < \tau$ then
    declare singleton
else
    declare multiton
end
```

Chapter 6

Speeding Up Distributed Computing: Straggler-Resilient Matrix Multiplication

6.1 Introduction

In the previous chapters, we designed algorithms that leveraged the underlying structure of the input to speed up computations. On the other hand, in this chapter, we will study the setting where data is divided among multiple machines, as in distributed computing, to reduce the computational burden and scale up computations to big-data regimes. Distributed computing is becoming the main mode of computation for modern large-scale machine learning and data analytics jobs. Dividing the workload among multiple workers can help speed up computation as each worker node works on a problem of smaller size, and the results of these smaller problems are then combined to obtain the target. However, an important problem in distributed computing systems is that there are compute nodes that are slowed down due to unpredictable factors impacting their compute and communication time [16]. Such worker nodes are called *stragglers*. At the same time, the scales of distributed computing systems, that is, the number of worker nodes in the system, are growing, (e.g., using AWS Lambda [37]). The straggler problem is even more evident in such large-scale computing systems. For example, in [37], about 5% to 10% of 3000 distributed workers are observed to be stragglers.

Straggling compute nodes severely impact runtime, as the total execution time depends on the slowest worker, if resiliency to straggling nodes is not built in to the computation. The authors of [44] apply erasure codes to distributed matrix multiplication algorithms, and propose a new framework called *coded computation* to combat stragglers, significantly speeding up distributed computing algorithms. Subsequently, coded computation techniques have been shown to speed up a variety of computing tasks such as large-scale matrix multiplications [46], distributed gradient computing, convolutions [21], Fourier transforms [99], and personalized PageRank [97] to name a few.

The typical workflow of coded computation schemes is illustrated in Figure 6.1. In this chapter,

This chapter is based on joint work with Tavor Baharav and Kangwook Lee [4].

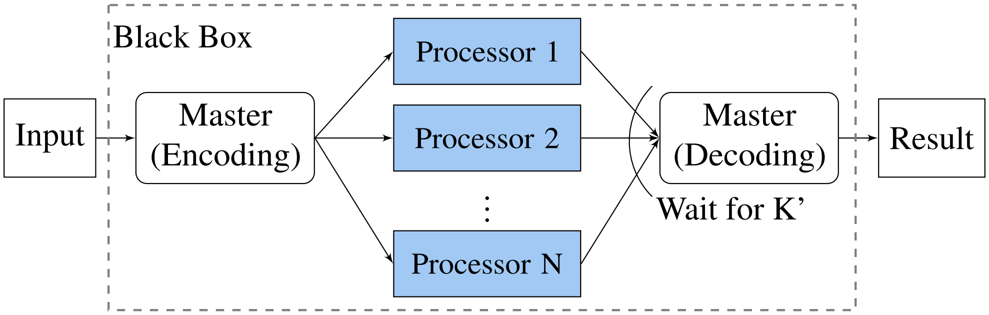


Figure 6.1: System architecture for distributed computing.

we assume a standard master-worker model but one may also consider a decentralized (or masterless) setting. First, unless the input data is already stored as encoded across the workers, the master node needs to *encode* the input data and share the encoded data with the distributed workers. We denote the total encoding time as the sum of 1) the time to encode data (T_{enc}) and 2) the time to transmit the encoded data from the master to the workers ($T_{\text{enc,comm}}$). Upon receiving the input data, the distributed workers start performing their assigned tasks. Upon completion of the tasks, the results of the computations need to be transmitted from each worker to the master. We denote the compute time (including the time to communicate the task result to the master) of each worker by T_i. Once a *decodable* set of task results are collected, the master runs the decoding algorithm to obtain the final output, and the time to run this stage is called the decoding time T_{dec}. Hence, the total execution time of a coded computing algorithm can be expressed as

$$T_{\text{total}} = \underbrace{T_{\text{enc}} + T_{\text{enc,comm}}}_{\text{Encoding}} + \underbrace{\min_{S \in \mathcal{S}} \max_{i \in S} T_i}_{\text{Compute } (T_{\text{comp}})} + \underbrace{T_{\text{dec}}}_{\text{Decoding}}, \tag{6.1}$$

where $\mathcal{S}$ is the set of all decodable sets of task indices [44].

Most existing works have focused on optimizing the *recovery threshold* of the algorithm [44, 85, 45]. Define the total time the distributed computation algorithm spends at the worker nodes by $T_{\text{comp}} = \min_{S \in \mathcal{S}} \max_{i \in S} T_i$; then, optimizing the recovery threshold is equivalent to minimizing the compute time T_{comp} only. This assumes that the encoding/decoding times are negligible in (6.1) compared to T_{comp}. When the number of worker nodes is relatively small, the encoding/decoding costs can be safely ignored. However, the encoding/decoding costs of coded computation schemes scale with the number of worker nodes, and this assumption does not hold for larger systems, e.g., systems that make use of tens of thousands workers [37].

This increase in the size of distributed computing platforms and the resulting need for balancing the encoding/decoding costs together with the computation time forms the precise motivation of our work. Our goal is to design a coded computation scheme that not only achieves low compute time but also allows for efficient encoding and decoding, hence a lower total execution time in (6.1).

6.1.1 Problem Formulation

We consider the matrix-matrix multiplication task [46], where we compute $A^\top B$ for $A \in \mathbb{R}^{s\times\ell}, B \in \mathbb{R}^{s\times t}$. We focus on the case where s is large, and $s \geq \ell$ and $s \geq t$, as otherwise the resulting matrix is rank-deficient, and there may be more efficient ways of computing the product. While our results and algorithm can easily be generalized to the case where $\ell \neq t$, we will focus on the setting where $\ell = t$ for ease of presentation. We divide A into m equal-sized chunks, $A_i \in \mathbb{R}^{s\times\ell/m}$ for $i \in \{0, \ldots, m-1\}$, Likewise, we divide B into h chunks, $B_j \in \mathbb{R}^{s\times\ell/h}$ for $j \in \{0, \ldots, h-1\}$. Then, the worker node i is assigned to perform a single matrix-matrix multiplication $\tilde{A}_i^\top \tilde{B}_i$, where $\tilde{A}_i$ and $\tilde{B}_i$ are input encoded matrices. The goal is to minimize T_{total} in (6.1) by carefully designing a code.

One interesting aspect of this problem that prevents us from using an arbitrary code are the constraints induced by the matrix-matrix multiplication. We desire the worker nodes to perform operations of the form $(\sum_{i\in\mathcal{S}_A} A_i)^\top(\sum_{j\in\mathcal{S}_B} B_j)$, that is, we want the task to factor. The reason is the following. For example, if a worker node needs to compute $A_0^\top B_0 + A_0^\top B_1$, it would be able to do this as $A_0^\top(B_0 + B_1)$. The summation $(B_0 + B_1)$ requires $O(s\ell/h)$ operations, and the multiplication requires $O(s\ell^2/(hm))$ operations. Note that the computational bottleneck is in the multiplication operation. However, if it wanted to compute $A_0^\top B_0 + A_1^\top B_1$, it would not be able to factor this and solve it quickly. The multiplication operations, in total, would require double the time. Furthermore, the runtime difference between the two operations increase with the number of terms that need to be summed. Thus, we cannot use an arbitrary code. We can formalize this constraint imposed by matrix-matrix multiplication as the requirement that if we represent our code in systematic form, then the rows of the generator matrix of our code must be of the form vec$(uv^\top)$ for vectors u, v, corresponding to the chunks of A, B selected[1].

6.1.2 Our Contributions

We propose a novel coded computation scheme based on high-dimensional product codes for matrix-matrix multiplication. We form this connection between matrix-matrix multiplication and decoding a high-dimensional product code by adding carefully designed parity chunks on the input matrices. We show that our proposed scheme can achieve near-optimal compute time performance as well as order-optimal encoding/decoding costs. This enables applicability to massive-scale distributed computing where encoding/decoding costs make prior solutions that only optimize over the recovery threshold impractical.

6.1.3 Related Works

In [46], the authors propose a 2-dimensional product code based scheme for straggler resilient matrix-matrix multiplication. The authors apply an (n, k, d_{min}) code to each matrix, generating

[1] In Chapter 7, we will describe a method for fast matrix-matrix multiplication whenever the output matrix is sparsity. In that problem, the same constraint will appear while inducing a code on the output matrix.

$n-k$ new *parity* chunks for each matrix. Then, worker (i,j) is assigned the task of computing $A_i^\top B_j$. While this scheme has linear encoding and decoding costs, it exhibits suboptimal T_{comp}.

Another scheme proposed for matrix-matrix multiplication is Polynomial Codes [100], where a Reed-Solomon (RS) code is applied to the resulting matrix chunks. The authors show that T_{comp} is optimal under the polynomial codes. This holds even when the straggling machines are chosen adversarially thanks to the RS codes being maximum distance separable (MDS)[2]. However, as the number of workers increases, the encoding/decoding costs are much higher than those of product codes. RS based schemes rely on polynomial interpolation for decoding, and as the problem size scales, this can become prohibitive[3]. Furthermore, polynomial interpolation is notoriously numerically unstable over the reals [22]. In order to improve numerical stability, the matrix must be quantized and converted into a finite field, which incurs additional complexity.

6.2 High-Dimensional Product Coded Computation

In this section, we propose the d-dimensional product coded matrix multiplication algorithm, generalizing the 2D algorithm proposed in [46]. Under the metric of end-to-end latency of (6.1), current MDS codes as in [100] exhibit undesirably long encoding/decoding times. On the other side of the spectrum, 2D product codes require more processors than optimal. The d-dimensional product codes fall between these two, exhibiting order-optimal encoding and decoding times while tolerating a constant fraction fewer stragglers than optimal (the constant factors are exactly characterized in Table 6.2).

Traditionally, product codes are thought of as 2D constructions, where one arranges the data in a $k \times k$ square, and applies a systematic linear (n,k) component code to every row and column, yielding an $n \times n$ square. We use the natural extension of this to higher dimensions, the d-dimensional product code as described more in depth in [5].

We describe the scenario of encoding a distributed matrix multiplication job via a d-dimensional product code. We proceed by partitioning $A \in \mathbb{R}^{s\times\ell}$ into $\sqrt{K} = (n-r)^{d/2}$ chunks, of size $A_i \in \mathbb{R}^{s\times\ell/\sqrt{K}}$, and $B \in \mathbb{R}^{s\times\ell}$ into $\sqrt{K} = (n-r)^{d/2}$ chunks, of size $B_i \in \mathbb{R}^{s\times\ell/\sqrt{K}}$, to be computed over $N = n^d$ processors. In the rest of the section, as we are going to describe the different phases of our algorithm in reference to Figure 6.2 as a toy example, and Table 6.1 summarizes the notation we will be using.

6.2.1 Encoding

Our encoding rests on the inherent tensored structure of a d-dimensional product code, as can be seen through the tensored structure of its encoding matrix. For a d-dimensional product code with

[2] An MDS code has parameters $(n, k, d_{\text{min}} = n-k+1)$. Hence, any subset of k erasures can be resolved.

[3] All the off-the-shelf RS erasure coding libraries we found are capable of operating only up to a block length of 256, in $GF(2^8)$. This is because RS implementations require storing 'log' and 'exp' tables for each element of $GF(2^8)$ to operate quickly.

Table 6.1: Glossary of important notation for High-Dimensional Product Codes.

Notation	Description
$(n, k, r+1)$	component code
d	dimension of product code
$N = n^d$	total number of workers
$K = k^d$	number of 'data' workers
$\ell \times s$	dimensions of input matrices A, B

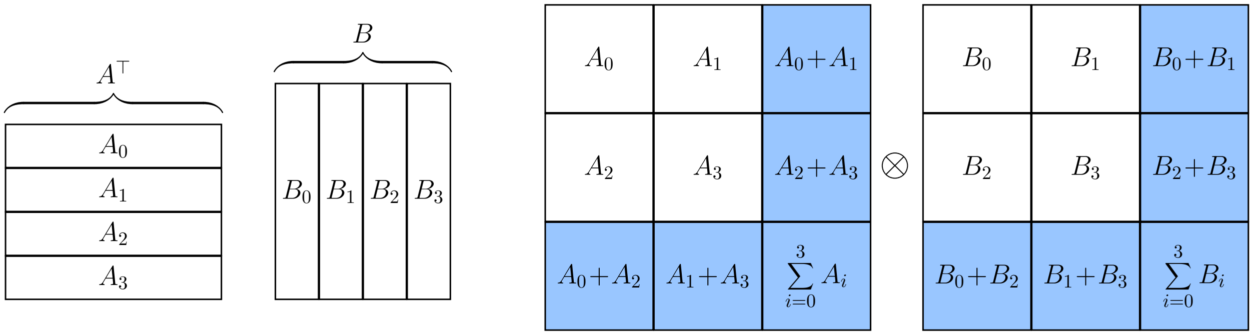


Figure 6.2: Toy example for a product code with $n = 3$, $d = 4$, $r = 1$, yielding $N = 3^4$. The 4D product code is equivalent to the two 2D product codes tensored together. Parity chunks are shaded in blue.

an $(n, k, r+1)$ component code, the overall generator matrix is simply the tensor product[4] of d generator matrices for its $(n, k, r+1)$ component code (see Figure 6.3). We make use of this inherent tensor structure by partitioning the tensors into 2 groups of $d/2$ matrices, and spreading A and B over $d/2$ dimensions each[5]. We achieve this by computing parities for the matrices A, B independently after we have divided them into chunks. We do this by arranging the $(n-r)^{d/2}$ matrix chunks of each matrix in a $d/2$ dimensional hypercube with side length $k = n - r$. We then encode these matrix chunks using a $d/2$-dimensional product code with component code $(n, k, r+1)$, yielding a $d/2$-dimensional hypercube with side length n of encoded chunks of A. For example, in the toy example in Figure 6.2 we have $d = 4$, so we divide each matrix along $4/2 = 2$ dimensions. Since each component code is $(n, k) = (3, 1)$, we have 2 data chunks and 1 parity chunk on each dimension.

6.2.2 Communication

As can be seen from Figure 6.2, there is a regular, easily exploitable structure to the matrix chunks that processors require. Due to the tensored structure of our encoding scheme, we need to send

[4] We refer to a Kronecker product on the blocks of the matrices A and B.

[5] This assumes d is even. For odd d, we spread one matrix over $\lceil d/2 \rceil$ dimensions, and the other one over $\lfloor d/2 \rfloor$.

$$G = \underbrace{\overbrace{\begin{bmatrix}1 & \\ & 1 \\ 1 & 1\end{bmatrix}}^{n-1} \otimes \ldots \otimes \overbrace{\begin{bmatrix}1 & \\ & 1 \\ 1 & 1\end{bmatrix}}^{n-1}}_{d}$$

Figure 6.3: Generator matrix of the d-dimensional product code with $(n, k, r+1) = (3, 2, 2)$.

each chunk of A to $n^{d/2}$ workers. In our toy example, the worker indexed by (i, j, k, ℓ) where each $i, j, k, l \in \{0, 1, 2\}$ needs to receive the (i, j)th chunk of A and (k, ℓ)th chunk of B. In particular, worker $(0, 0, 0, 0)$ receives A_0 and B_0, and worker $(0, 0, 0, 2)$ receives A_0 and $B_0 + B_1$.

6.2.3 Computation

We are able to induce a d-dimensional product code on the results of the worker nodes through a tensoring of coded matrix chunks of A and B. This is done by taking the $d/2$ dimensional hypercube of coded chunks of A and taking its tensor product with the $d/2$ dimensional hypercube of coded chunks of B. In our toy example depicted in Figure 6.2, each worker (i, j, k, ℓ) where each $i, j, k, l \in \{0, 1, 2\}$ multiplies the matrices it receives; worker $(0, 0, 0, 0)$ performs $A_0^\top B_0$, and worker $(0, 0, 0, 2)$ performs $A_0^\top(B_0 + B_1)$. This induces the desired d-dimensional product code structure.

6.2.4 Decoding

For decoding, we employ a simple *peeling* decoder. Due to the induced d-dimensional product code, we are able to arrange the results of the worker nodes in a d-dimensional hypercube with side length n, as suggested by the tensor structure (the example in Figure 6.2 would create a 4-dimensional hypercube). Since each 'row' is a codeword for the $(n, k, r+1)$ component code, we can iterate over all dn^{d-1} rows and resolve the erasures (where each erasure corresponds to a straggling node) if there are r or fewer erasures in a given row. Resolving an erasure in one row may lead to another row becoming resolvable, and so if there are few enough erasures (formally characterized in Theorem 8), we can iterate over the rows of the d-dimensional product code and resolve $(n, k, r+1)$ component codes until the results of all straggling worker nodes have been recovered.

6.2.5 Choosing d, r

The choice of d and r (row component codes) make a significant difference in performance and setting. We can formulate the choice of these parameters as an optimization problem for a given number of workers N. Assume that the master node can perform α floating point operations per second (FLOPS) and transmit β bits per second (BPS). Given the task runtime distribution, we can choose d and r to minimize the end-to-end latency in equation 6.1 by solving:

$$\underset{d,r}{\text{minimize}} \quad \alpha^{-1}T_{\text{enc}} + \beta^{-1}T_{\text{enc,comm}} + \mathbb{E}T_{\text{comp}} + \alpha^{-1}T_{\text{dec}}. \tag{6.2}$$

Depending on the parameters α and β, the optimum d might not correspond to the optimizer of the recovery threshold (as it would minimize $\mathbb{E}\left[T_{\text{comp}}\right]$ only).

6.3 Analysis

Our analysis utilizes the isomorphism between d-dimensional product codes and a class of d-left regular Low Density Parity Check (LDPC) codes presented in [5]. A high level intuition for this isomorphism is that the randomness of the erasure channel (nature's random choice of straggling processors) allows us to leverage LDPC analysis techniques on the residual graphs.

6.3.1 Compute Time

We can derive the density evolution update equation, and substitute in d and r to numerically compute the maximum number of unknown variables that the residual graph can peel.

Theorem 8. *In a d-dimensional product coded matrix multiplication scheme with $(n, k, r+1)$ component codes, a set of tasks S with $|S| \geq \left(N - \frac{N-K}{\eta(d,r)}\right)(1+\epsilon)$ drawn uniformly at random from the set of all N tasks is a decodable set with high probability for arbitrarily small $\epsilon > 0$ and $\eta(d, r)$ as given in Table 6.2.*

Proof. Through density evolution analysis, one can determine the erasure correcting capability of these d-dimensional product codes. This result directly follows from [5]. □

In order to calculate the expected runtime of a computation scheme we first assume the existence of a *mother runtime distribution* $F(t)$ [44]. We assume that running an algorithm using a single machine takes a random amount of time T, that is a positive-valued, continuous random variable distributed according to F, that is, $\Pr(T \leq t) = F(t)$. Furthermore, when the algorithm is distributed into K data subtasks, the runtime distribution for each subtask is assumed to be a scaled distribution of the mother distribution as $\Pr(T_i \leq t) = F(Kt)$ for $1 \leq i \leq N$. We assume that the computing times of these N tasks are independent of one another. For convenience, let

$$K' = \left(N - \frac{N-K}{\eta(d,r)}\right),$$

be the effective number of completed tasks a d-dimensional product code requires before it becomes decodable. Then, under these assumptions, $T_{\text{comp}} = X_{(K')}$, the K' order statistic of N i.i.d. random variable $X_i \sim f(x)$.

Empirically, a shifted exponential distribution which is the sum of a constant $c \geq 0$ and an exponential random variable with rate $\mu \geq 0$, i.e., $F(t) = 1 - e^{-\mu(t-c)}, \forall t \geq c$, has been found to be an accurate and tractable model for processor run time [44]. For $\{X_i\}_{i=1}^N$ distributed as i.i.d. shifted exponentials,

$$\mathbb{E}\left[X_{(m)}\right] = \frac{1}{K}\left(c + \frac{1}{\mu}\ln\left(\frac{N}{N-m}\right)\right).$$

Specializing to the case of a d-dimensional product code, we get

$$\mathbb{E}\left[T_{\text{comp}}\right] = \mathbb{E}\left[X_{(K')}\right] = \frac{1}{K}\left(c + \frac{1}{\mu}\ln\left(\frac{N}{(N-K)/\eta(d,r)}\right)\right) = O\left(\frac{1}{N}\left(c + \frac{1}{\mu}\ln(n)\right)\right).$$

In the case of a shifted exponential processor latency model, we can see that a d-dimensional product code has order optimal compute time. The bigger picture idea behind this is that the K'th order statistic will be very close in expectation to the Kth order statistic, while the Nth order statistic (max) will be much higher.

Table 6.2: Recovery thresholds, $\eta(d, r)$, for product codes as a function of the dimension of the code, d, and the error correcting capability of the component code, r.

d \ r	1	2	3	4	5	6
3	1.2218	1.2880	1.3797	1.4564	1.5202	1.5741
4	1.2949	1.4998	1.6568	1.7781	1.8760	1.9575
5	1.4250	1.7275	1.9409	2.1031	2.2327	2.3406
6	1.5697	1.9577	2.2244	2.4256	2.5864	2.7199
7	1.7189	2.1869	2.5051	2.7446	2.9361	3.0953

6.3.2 Encoding and Decoding costs

While achieving near-optimal compute time as stated in Theorem 1, our scheme also achieves order-optimal encoding (both T_{enc} and $T_{\text{enc,comm}}$) and decoding complexities.

Lemma 5. *T_{enc} for a d-dimensional product coded scheme is $O(\ell s)$, which is order-optimal.*

Proof. To encode the d-dimensional product code, we simply need to encode the matrix chunks of A, B in $d/2$-dimensional product codes, as explained in Sec. 6.2.1. Since this is a $d/2$-dimensional product code itself, it can be encoded in time linear with respect to the number of chunks: $O(n^{d/2})$. Since matrix chunks are of size $\ell \times \frac{s}{(n-r)^{d/2}}$, encoding of A is $n^{d/2}\frac{\ell s}{(n-r)^{d/2}} = O(\ell s)$, as d and r are constants, and $n \to \infty$. The lower bound comes from the requirement that any scheme needs to read every element of A and B in order to prepare the chunks for communication. □

Lemma 6. *Communication cost, $T_{enc,comm}$, is $O(\ell s\gamma(\sqrt{N}))$, where $\gamma(m)$ is the multicast cost of communicating the same message to m workers. This cost is order-optimal.*

Proof. Consider an arbitrary systematic coding scheme that divides A, B into m chunks each. It needs at least m^2 workers to return to finish computing, so $N \geq m^2$. We proceed by considering its communication cost for A. Assume that this scheme sends chunks of the same size to each worker, and that there are p, $m \leq p \leq N$, coded chunks of A. Then, the cost is $\frac{\ell s}{m} \times (\sum_{i=1}^{p} \gamma(c_i))$, where c_i is the number of times we send chunk i. Since the code is systematic, $c_i \geq m$ for $i = 1, ..., m$. Thus, the cost is lower bounded by $\frac{\ell s}{m} \times (m\gamma(m))$, as $\gamma(\cdot)$ is monotone non-decreasing, and positive. Because $m^2 = \theta(N)$ in our scheme, the cost can be lower bounded by $\Omega(\ell s \times \gamma(\sqrt{N}))$.

Due to the tensored structure of our encoding scheme, we send the same chunk of A to $n^{d/2}$ workers. Communication cost is thus $O\big((n^{d/2}\gamma(n^{d/2}))(\ell s)/((n-r)^{d/2})\big) = O(\ell s \times \gamma(\sqrt{N}))$. For instance, if $\gamma(m) = \log(m)$, then our communication cost is $O\big((n^{d/2}\log(n^{d/2}))(\ell s)/((n-r)^{d/2})\big) = O(\ell s \log(n))$. □

Lemma 7. *For a d-dimensional product code, decoding cost, T_{dec} is $O(\ell^2)$, which is order-optimal.*

Proof. The decoding time of our scheme is linear in the size of the output matrix, ℓ^2. After K' workers have arrived the residual graph will be resolvable with high probability (c.f., Theorem 8). Because each straggling matrix chunk is of size $\frac{\ell^2}{K}$, and there are n chunks per row, $\frac{n}{(n-r)^d}\ell^2$ work is required to resolve a row. Because we have dn^{d-1} rows, our decoding time can be upper bounded by the time it takes to resolve them all, which gives $\frac{dn^d}{(n-r)^d}\ell^2 = O(\ell^2)$. This is order-optimal, as any scheme will need to read enough workers' results in order to reconstruct the output $\ell \times \ell$ matrix. □

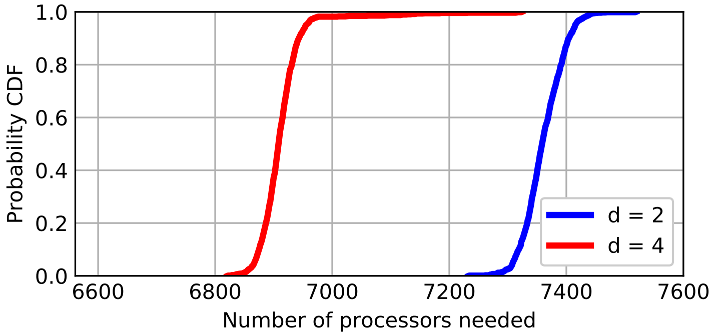


Figure 6.4: Simulation results for d-Dimensional Product Codes with $N = 10^4 = 100^2$, $k = 9^4 = 81^2$. Experiments are done using a $(100, 81, 20)$ component code for $d = 2$, and a $(10, 9, 2)$ component code for $d = 4$.

6.3.3 Comparisons

Reed-Solomon

We assume that a systematic RS code is used. For the parities, encoding is done by creating a Vandermonde structure, which entails a weighted linear combination of the chunks of A, of the

Table 6.3: Comparisons (orderwise) of encoding, communication, computation and decoding time for uncoded computation and coded computation using d-Dimensional Product Codes, and Systemmatic Reed-Solomon Codes.

	d-Dim. Product Code	Systematic Reed-Solomon	Uncoded
T_{enc}	$d\ell s$	$d\ell s n^{d-1}$	ℓs
T_{comm}	$\ell s\gamma(n^{d/2})$	$\ell s(\gamma(k^{1/2}) + dn^{d/2-1})$	$\ell s\gamma(\sqrt{N})$
T_{comp}	$\frac{1}{N}(c + \frac{1}{\mu}\log(n))$	$\frac{1}{N}(c + \frac{1}{\mu}\log(n))$	$\frac{1}{N}(c + \frac{1}{\mu}\log(N))$
T_{dec}	$d\ell^2$	$\ell^2 d^2 n^{d-2}$	ℓ^2

form $\sum_{i=1}^{\sqrt{K}} \alpha^i A_i$, for a properly chosen factor α. However, this work is unique for each parity, and involves using the entire matrix A. Thus, the total work of encoding will be $\ell s(N-K)$.

For communication, a systematic RS code will be able to broadcast the $\sqrt{K}$ data chunks of each matrix to the data workers; each chunk is $\ell \times \frac{s}{\sqrt{K}}$, and needs to be used by $\sqrt{K}$ workers, yielding a cost of $\frac{\ell s}{\sqrt{K}}(\sqrt{K} \cdot \gamma(\sqrt{K})) = \ell s \cdot \gamma(\sqrt{K})$. For the parity workers, each one uses a uniquely computed chunk of A, meaning that multicast gains cannot be utilized. The cost for sending the parity chunks is $\frac{\ell s}{\sqrt{K}} \cdot (N-K)$, giving a total communication cost of $\ell s(\gamma(\sqrt{K}) + \frac{N-K}{\sqrt{K}})$.

For decoding, a systematic RS code will need to interpolate a polynomial for each one of the $\frac{\ell^2}{K}$ entries of an output chunk. In each of these interpolations, one needs to solve a Vandermonde system of size $(N-K) \times (N-K)$, which is usually accepted to cost $O((N-K)^2)$ [17]. This results in a decoding cost of $\ell^2 \frac{(N-K)^2}{K} = O(\ell^2 d^2 n^{d-2})$.

Uncoded

Encoding is simply splitting the matrices into $\sqrt{N}$ chunks each, which requires a single pass over the data. Each processor receives $A_i, B_j \in \mathbb{R}^{s \times \ell/\sqrt{N}}$, and thus one broadcasts each A_i (similarly B_i) chunk to $\sqrt{N}$ processors. Thus, the overall communication cost is $O(\ell s\gamma(\sqrt{N}))$. The computation latency is $\mathbb{E}X_{(N)}$, and so the expected latency under a shifted exponential model is $O(c + \mu\ln(N))$.

6.4 Simulation Results

One way of comparing different schemes is by comparing their synthetic wall clock times; for a given matrix size, the number of worker nodes N and the number of straggling nodes K, we can calculate the number of flops required for the encoding and decoding procedures. We can then compute the expected worker latency for a given model (shifted exponential), and then express the wall clock time as $\alpha(T_{\text{enc}} + T_{\text{dec}}) + \mathbb{E}T_{\text{comp}}$. We sweep α over a wide range of values. We can see from Figure 6.5 that when encoding and decoding costs are negligible (when α is small), RS based coded computation method has lower latency. However, as the encoding/decoding costs factor in (as α increases), product codes achieve better latency. Furthermore, $d = 4$ product codes perform

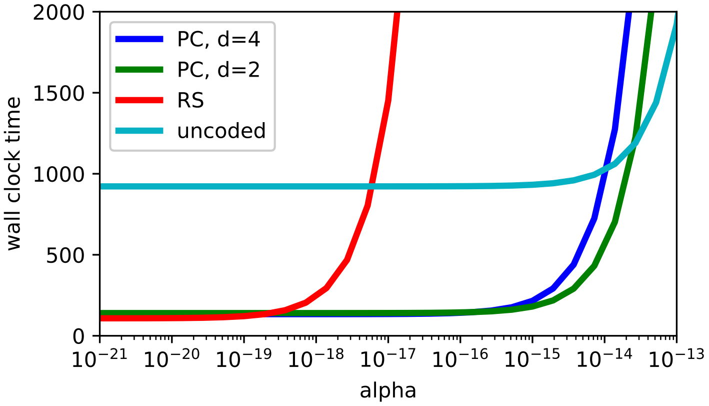


Figure 6.5: Synthetic wall clock times of coded computation methods for $N = 10^4, k = 9^4, r = s = 10^8$. Time = $\alpha \cdot (T_{\text{enc}} + T_{\text{dec}}) + \mathbb{E}T_{\text{comp}}$. PC denotes product codes, and RS denotes Reed-Solomon codes.

better compared to $d = 2$ due to their superior threshold. However, once encoding/decoding costs become dominant, $d = 2$ attains better latency to its marginally lower encoding/decoding costs. As α increases further, latency introduced due to stragglers become less of a bottleneck compared to the encoding/decoding costs, and the uncoded scheme becomes the fastest.

6.5 Conclusions and Future Directions

We formulated a new model for analyzing the performance of coded computation schemes which encapsulates the desire to minimize end-to-end latency, and presented a novel coded matrix-matrix multiplication scheme that has order-optimal computation/communication costs for the encoding/decoding procedures and achieves near-optimal compute time. Our scheme is based on the use of high-dimensional product codes, and allows for efficient encoding/decoding and low communication cost. Because product codes can be decoded by operating on single ‘rows’ and ‘columns’ in an iterative way, our algorithm is a suitable candidate to have an impact in master-less computation frameworks. In particular, in a platform having shared memory with fast input/output where workers write their results, separate workers assigned to the task of decoding can do so in a streamlined fashion. This master-less architecture is an exciting new computing paradigm which Product Codes are well suited for, and tailoring our algorithm to master-less setting is an interesting future research direction.

Chapter 7

Fast Matrix-Matrix Multiplication with Output Sparsity

7.1 Introduction

We are witnessing an unprecedented growth in the amount of data being collected, generated and processed to drive machine learning and decision-making tasks across a multitude of domains. This has challenged classical algorithms to scale up to big-data regimes, driving architectures to either leverage the underlying structure of the data to reduce the computational burden, or to divide the data among multiple machines, or do some combination of both. Motivated by this observation, in this work, we focus on reducing the computational burden of large-scale matrix-matrix multiplication when the output matrix has a particular structure, that is, it is sparse or approximately sparse.

Matrix-matrix multiplication is a fundamental linear algebra primitive that is the workhorse of many algorithms in machine learning, signal processing, and scientific computing; so a speed-up therein has potential to impact a multitude of applications. Leveraging the sparsity of the output matrix in a matrix-matrix multiplication operation has applications ranging from the recovery of sparse covariance matrices [95, 71] to fast string search across a database (where the rows of the first matrix represent the database of strings and the columns of the second matrix represent the query strings to be searched in the database) [35] to sketching sparse matrices [95, 15].

For the rest of this chapter, we abstract away the use-case scenarios and focus on the base problem of performing large-scale matrix-matrix multiplication efficiently whenever the resulting matrix is known to be sparse. We show that product codes, which have been studied for channel coding in communications [23, 73], can be used to accomplish this.

Our algorithm draws inspiration from the previous chapter of the thesis and [4], where we introduced controlled redundancy in large-scale distributed matrix-matrix multiplication to combat against slow computational nodes (stragglers) by inducing a product-code structure. Hence, the use of codes in the previous chapter is like their use in *channel coding*, where controlled redundancy is introduced through codes to combat against nonidealities in a communication channel. There is a *dual* use of codes, that is, for *source coding*, where codes are used for compression. In this chapter, we use product codes in their dual *source coding* form to perform compressed calculations in matrix-matrix multiplication to reduce computational complexity. To the best of our knowledge, this use case is new, and it is interesting to see that the product codes can be used in a variety of problems that relate to matrix multiplication.

The high-level summary of our method is as follows. We first sketch the input matrices to obtain matrices of smaller sizes, and then perform multiplications between them to create a set of *check nodes* (see Section 7.3 for details). Because multiplications are performed over smaller-sized matrices, we get a lower computational complexity for the creation of these check nodes compared to multiplying the original input matrices. By carefully designing these sketches to induce a product code structure on the check nodes (see Section 7.4 for details) and leveraging the underlying sparsity in the output matrix, we then recover the matrix-matrix multiplication output based on the check nodes through *peeling*.

7.1.1 Notation and the Organization of the Chapter

In Section 7.2 of this chapter, we present the problem formulation and summarize our main results. In Section 7.3, we describe the sketched matrix-matrix multiplication framework that we are going to use in our algorithm. We present our algorithm for sparse-output matrix-matrix multiplication in Section 7.4. In Section 7.5, we present numerical simulations corroborating our claims, and then conclude with Section 3.7.

For a positive integer n we use $[n]$ to denote the set $\{0, \dots, n-1\}$. For $i \in [n]$, we denote the ith column of a matrix $\mathbf{A}$ by the column vector $\mathbf{a}_i$. The entry at ith row and jth column of a matrix $\mathbf{C}$ is denoted by $\mathbf{C}_{i,j}$. We refer to the non-zero entries of a matrix as its support.

7.2 Problem formulation and our contributions

Given matrices $\mathbf{A} \in \mathbb{R}^{d \times n}$ and $\mathbf{B} \in \mathbb{R}^{d \times n}$ such that their product $\mathbf{C} = \mathbf{A}^\top \mathbf{B}$ is K-sparse, our goal is to design an algorithm that reduces the number of operations below the naïve $O(n^2 d)$ to find the support[1]. We propose a novel algorithm for doing so, which works by transforming the problem of finding the support of the output matrix to decoding a product code on an erasure channel through *peeling*. Our algorithm first obtains carefully-designed *sketches* of the input matrices, and then calculates the products between them. The sketches are designed so that the resulting multiplication outputs have a product code structure, from which the matrix $\mathbf{C}$ can be decoded. For theoretical results, we consider the following simplifying assumption.

Assumption 1. *The K non-zero entries of the output matrix $\mathbf{A}^\top \mathbf{B}$ are uniformly distributed in the set of n^2 entries.*

Our main results are as follows.

[1] In general, our method allows for matrices to be of size $d \times n$ and $d \times m$ where $m \neq n$. For simplicity we consider $n = m$.

Theorem 9. *Under Assumption 1, for $K = O(n^{\delta})$ where $\delta \in (0,2)$, using $\max(O(dK), O(dn))$ computations, the non-zero elements of $\mathbf{A}^{\top}\mathbf{B}$ can be recovered with probability $1 - O(1/K)$.*

Our algorithm can be extended to handle an approximately-sparse setting as the following theorem states.

Theorem 10. *Under Assumption 1, let $K = O(n^{\delta})$ for $0 < \delta < 2$. Let $\mathcal{S}$ denote the support, and assume that $(\mathbf{A}^{\top}\mathbf{B})_{i,j}$ in $\{-1, 1\}$ if $(i,j) \in \mathcal{S}$ and $\left|(\mathbf{A}^{\top}\mathbf{B})_{i,j}\right| \leq O(n^{-1})$ otherwise. Then, we can recover $\mathcal{S}$ and the coefficients of $\mathbf{A}^{\top}\mathbf{B}$ on $\mathcal{S}$ using $\max(O(dK\log^2(n)), O(dn\log^2(n)))$ computations with probability $1 - O(1/K)$.*

7.3 Sketched Measurements and Related Work

Given $\mathbf{A}$ and $\mathbf{B}$, consider measurements of the form

$$f_{\mathbf{A},\mathbf{B}}(\mathbf{u}, \mathbf{v}) = \mathbf{u}^{\top}\mathbf{A}^{\top}\mathbf{B}\mathbf{v} = \sum_{i=1}^{n}\sum_{j=1}^{m} u_i v_j (\mathbf{a}_i^{\top}\mathbf{b}_j). \tag{7.1}$$

Note that these measurements can be calculated by first *sketching* the input matrices by calculating $\mathbf{Au}$ and $\mathbf{Bv}$, and then performing the inner product $\langle \mathbf{Au} \rangle \mathbf{Bv}$. We refer to the vector inputs to f as *sketching vectors*, and the output of f as *measurements*. We will group the measurements $f_{\mathbf{A},\mathbf{B}}(\mathbf{u}, \mathbf{v})$ for which the support of $\mathbf{vu}^{\top}$ is the same, and refer to each group of measurements as a *check node*, and we will refer to the entries of the output matrix as *variable nodes*[2]. We seek an algorithm that obtains measurements in the form of (7.1) and recovers the support of $\mathbf{C} = \mathbf{A}^{\top}\mathbf{B}$.

Note that the measurements in (7.1) are linear in $\mathbf{vu}^{\top}$. Hence, our model can be viewed as taking linear measurements of the output matrix $\mathbf{C}$. Recently, the authors of [96] showed that one can design a system of linear measurements for a *vector* such that its support can be recovered with sublinear measurement and computational complexity in the ambient dimensions whenever the sparsity of the vector also scales sublinearly in the ambient dimensions. The algorithm therein works by dividing the problem of recovering the support of a K-sparse vector to recovering the support of K many 1-sparse vectors, and then combining their solutions to get the support of the input vector. Although our measurement system is linear, it is linear in $\mathbf{vu}^{\top}$, and because our freedom is through designing $\mathbf{v}$ and $\mathbf{u}$, we are constrained in the measurements we can generate. These constraints make the methods described in [96] not applicable to our problem. However, our approach is inspired by their work as we will still divide the problem into simpler sub-problems and combine their results to recover the support of the output matrix. Despite the constraint that our measurements are linear in $\mathbf{vu}^{\top}$, by carefully designing the sketching vectors $\mathbf{v}$ and $\mathbf{u}$, we are able induce a product code structure and recover the non-zero entries of the output matrix efficiently.

[2] The collection of measurements $f_{\mathbf{A},\mathbf{B}}(\mathbf{u}, \mathbf{v})$ where the support of $\mathbf{vu}^{\top}$ is the same will comprise the syndrome of the signal residing on that support in the output matrix. This connection will be made clear in Section 7.4.

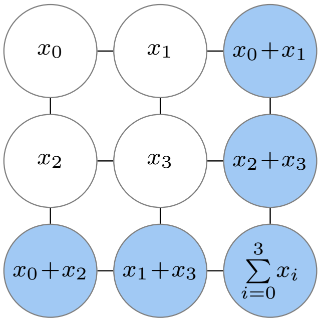


Figure 7.1: An example product code with $k = 2$ data symbols in each dimension. Parity symbols are colored in blue.

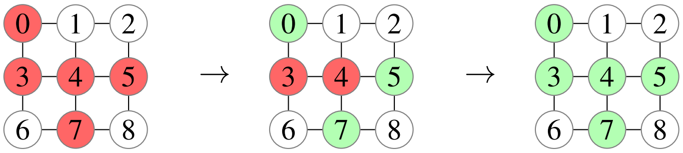


Figure 7.2: Decoding a 3×3 two-dimensional product code through peeling. The unresolved entries are shown in red, zero entries are shown in white, and recovered entries are shown in green. In the first round of peeling, entries labeled with 1, 5 and 6 singletons (single red circle in a row or a column), and are recovered. In the second round, entires 3 and 4 become singletons and are recovered.

7.4 High-dimensional product coded matrix-matrix multiplication

In standard 2-dimensional product codes [23] one first selects a linear component code $\mathcal{C}$. For simplicity, we will consider product codes whose component code $\mathcal{C}$ is the $(k+1, k)$ parity-check code. The resulting product code is achieved by arranging the k^2 data bits in a $k \times k$ square, and encoding to a $(k+1) \times (k+1)$ square. This is done by encoding each of the k data rows with $\mathcal{C}$, determining the parity symbol for each row, and then encoding all $k+1$ columns with $\mathcal{C}$ to determine the parity symbol of each column (see Figure 7.1 for an illustration). Product codes allow for the use of a *peeling* decoder in recovering erasures, which yields simple, linear-time decoding [1]. This is because if there is a single erasure in a row (or a column), then it can be recovered as each row (or a column) is a codeword of $\mathcal{C}$, which is the $(k+1, k)$ parity-check code. This may in turn allow us to decode other erasures as we have resolved some unknown symbols, and continue this iterative peeling process (see Figure 7.2 for an example peeling process). High-dimensional product codes extend this 2-dimensional framework to higher dimensions [4].

In this work we use product codes to perform *compression* by obtaining a *sketch* of the output matrix $\mathbf{C}$ from which the matrix $\mathbf{C}$ can be recovered. The sketches are composed of syndromes at check nodes, and these check nodes represent a product code on the elements of the matrix $\mathbf{C}$. Before describing the general product-coded matrix-matrix multiplication framework, we give two

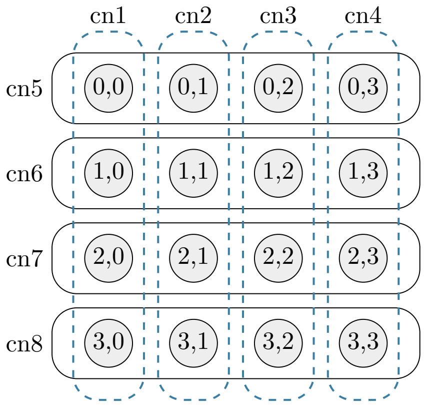


Figure 7.3: Illustration of check nodes for a 4×4 matrix. Elements of the matrix are shown as circles, and there are 8 check nodes represented by rounded rectangles with labels cn1 to cn8. Each check node consists of a set of linear combinations of the elements that they encircle. These linear combinations are designed so that if there is a single non-zero entry connected to a check node, we can resolve its location and value (see Sections 7.4.1 and 7.4.2.) The Tanner graph representation of the resulting code is shown in Figure 7.4

examples that illustrate how product codes can be constructed for matrix-matrix multiplication. The first example corresponds to a 2-dimensional product code.

Example 5. Let $\mathbf{A}$ and $\mathbf{B}$ be two $d \times 4$ matrices for some $d \in Z_{++}$. The resulting matrix $\mathbf{C} = \mathbf{A}^\top \mathbf{B}$ is of size 4×4. Entries of $\mathbf{C}$ can be indexed by a tuple (i, j) where the indices correspond to rows and columns of $\mathbf{C}$ respectively. As $\mathbf{C}_{i,j} = \mathbf{a}_i^\top \mathbf{b}_j$, the first index corresponds to the column index of $\mathbf{A}$, and the second index corresponds to the column index of $\mathbf{B}$ which get multiplied together to produce $\mathbf{C}_{ij}$. For inducing a product code structure, we calculate a set of linear combinations over the rows and the columns of $\mathbf{C}$. Each set of linear combinations is designed so that if there is a single non-zero element in a row or a column of $\mathbf{C}$, we can recover its position and value. The systematic way of designing these weighted measurements are going to be explained in sections 7.4.1 and 7.4.2.

Consider a linear combination of the elements in column 0 of $\mathbf{C}$. This can be achieved by choosing, in equation (7.1), $\mathbf{u} = (w_0, w_1, w_2, w_3)^\top$, and $\mathbf{v} = (1, 0, 0, 0)^\top$, where $\{w_0, w_1, w_2, w_3\}$ are the weights for the linear combination. Likewise, linear combinations of the elements in any row or column can be obtained by using an appropriate choice of vectors $\mathbf{u}$ and $\mathbf{v}$ in (7.1). Going over each row and column of the output matrix yields a total of 8 set of measurements. The support of these measurement sets are depicted in Figure 7.3. Through this set of measurements, we create 2-dimensional product code whose Tanner graph is shown in Figure 7.4. ■

Although the structure of a 2-dimensional product code yields an example with intuitive explanation because the matrices we are multiplying are also 2-dimensional structures, the theoretical guarantees and the practical performance are inferior to 3-dimensional product codes. This is because they suffer from an error floor when the component codes are single erasure correcting [70].

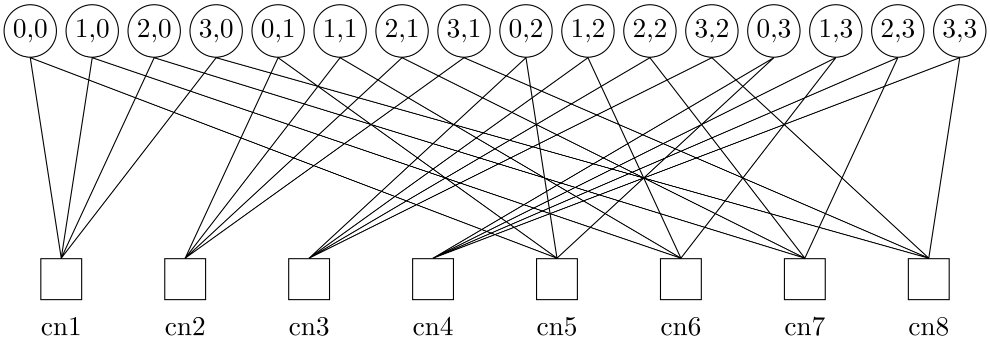


Figure 7.4: Tanner graph representation for the product code in Figure 7.3. Variable nodes (entries of the matrix) are shown as circles and the check nodes are represented by squares.

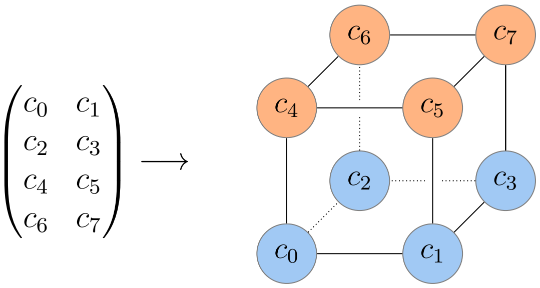


Figure 7.5: A 3-dimensional product code for a 4×2 output matrix. The 8 entries of the matrix are laid out on a 3 dimensional cube as shown. Sketched measurements correspond to weighted sum of elements along chosen (possibly multiple) dimensions. Here, we show two check nodes, each shaded a different color, and they are over elements at a fixed 'height.'

Increasing the dimension of the code to 3 or more removes this error floor. The following is an example of how one can create a 3-dimensional product code structure.

Example 6. Given $\mathbf{A}$ of size $d \times 4$ and $\mathbf{B}$ of size $d \times 2$, the output matrix $\mathbf{C} = \mathbf{A}^\top \mathbf{B}$ is of size 4×2 and has 8 elements. Index the entries of the resulting matrix using 3 bits as (i_1, i_0, j_0), where the first 2 bits correspond to the column index of $\mathbf{A}$, and the last bit corresponds to the column index of $\mathbf{B}$. Then, we choose a 3-way partitioning of the indices as $i' = (j_0)$, $j' = (i_0)$, and $k' = (i_1)$, and interpret each new index as corresponding to an axis of a cube. This corresponds to placing the 8 entries of the matrix on the vertices of a 3-dimensional cube as shown in Figure 7.5. The sketched measurements correspond to weighted sums of elements along each dimension of this resulting cube. ■

For the rest of this chapter, we will consider 3-dimensional product codes. The extensions to other dimensions can be performed by partitioning the indices accordingly. For simplifying the notation assume that $n = m = p^3$ where p is a non-negative integer[3]. The resulting matrix

[3] This is to simplify the explanation of our approach as this particular choice results in a 3-dimensional product

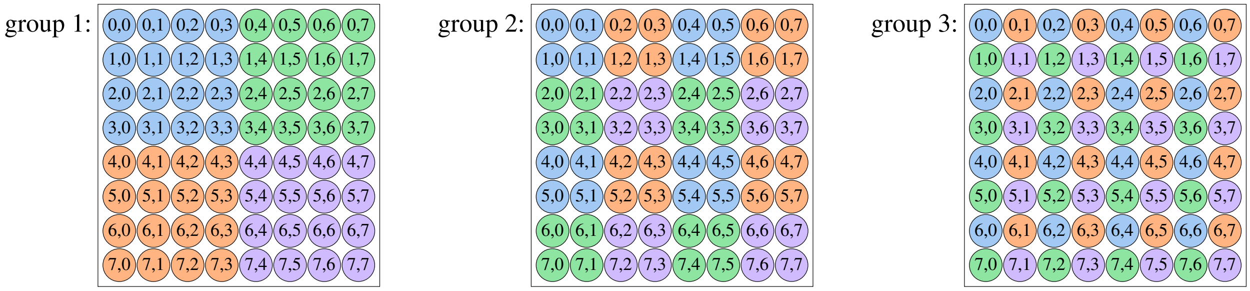


Figure 7.6: Illustration of a 3-dimensional product code on an 8×8 output matrix. There are 4 check nodes in each group (12 check nodes in total). The entries shaded with the same color in a group are connected to the same check node in that group; and each entry of the matrix is connected to 3 check nodes (1 in each group). Observe that each check node is connected to 16 entries. Furthermore, sketched measurements going to a check node can be obtained by first taking a linear combination of 4 columns of $\mathbf{A}$ and 4 columns of $\mathbf{B}$ and then taking an inner product between these two sketches. Given the information that an entry is connected to a check node in a block, it is equally likely to go to any check node in the other two groups.

$\mathbf{C} = \mathbf{A}^\top \mathbf{B}$ is of size $p^3 \times p^3$ having a total p^6 elements. We can assign an indexing to the elements of $\mathbf{C}$ so that the elements lie on a 3-dimensional cube of size $p^2 \times p^2 \times p^2$. For this, let $i \in [p^3]$ denote the row and $j \in [p^3]$ denote the column of $\mathbf{C}$. Represent the indices as $i = i_2 p^2 + i_1 p + i_0$ and $j = j_2 p^2 + j_1 p + j_0$ where each of the coefficients $\{i_0, i_1, i_2, j_0, j_1, j_2\}$ is in $[p]$. Then, map the indices (i, j) to on 3-dimensions (i', j', k') as

$$\begin{aligned}(i, j) &= (i_2 p^2 + i_1 p + i_0, j_2 p^2 + j_1 p + j_0) \\ &\mapsto (j_0 p + i_0, j_1 p + i_1, j_2 p + i_2) = (i', j', k').\end{aligned} \tag{7.2}$$

Note that each component of the new indices has equal number of contributions from original indices (i, j). As a result, check nodes in the product code can be obtained by first taking linear combinations of equal number of columns of $\mathbf{A}$ and $\mathbf{B}$, and then taking their inner products.

The number of check nodes necessary to recover $\mathbf{C}$ depends on the sparsity level K. Below, we describe the construction for the setting where $K = O(n^{2/3})$. In that case, we build 3 sets of p^2 checks nodes. As $p = n^{1/3}$, this yields a total of $3n^{2/3}$ check nodes. We then connect a check node for the elements of the output matrix whose indices are equal to each other in one dimension of (i', j', k'). For example, one of the check nodes is connected to indices $\{(i', j', k') : i' = 0\}$. Figure 7.6 shows the connectivity of the check nodes and the elements of the matrix $\mathbf{C}$.

Since the sparsity is uniformly distributed, each non-zero entry of $\mathbf{C}$ is connected to a check node chosen uniformly at random in each dimension. This gives a left-regular Low-Density-Parity-Check (LDPC) code construction where the variable nodes have degree 3, and the degrees of the check nodes follow a Poisson distribution [70]. The proof for recovering the support of $\mathbf{C}$ follows similar lines to the proof of the algorithm in [70]. Here, we outline the high level ideas for completeness.

code with length p^2 in each dimension. High-dimensional product codes can be built in higher dimensions and with different lengths in each dimension.

In peeling, we recover a variable node (non-zero coefficient of the matrix) if it is connected to a check node with degree 1, and remove the outgoing edges from that variable node. The density evolution equation for the expectation that a randomly chosen edge in the Tanner graph is not removed after ℓ rounds of peeling is given by the recursive equation $p_\ell = \left(1 - e^{-dp_{\ell-1}/(M/K)}\right)^{d-1}$, where $p_0 = 1$, and M is the total number of check nodes (see the derivation of equation A.10 in Appendix A.1). This equation assumes that the depth ℓ neighborhood of the chosen edge is a tree. We can show similarly to [70] that the depth ℓ neighborhood of a randomly chosen edge is a tree with high probability for any fixed ℓ. On average, an arbitrarily large fraction of edges are removed if p_ℓ goes to zero as $\ell \to \infty$. For p_ℓ to go to zero, M/K needs to be greater than a threshold for each fixed d. These thresholds are found through numerical methods (e.g., bisection) using the equation for p_ℓ. Table 7.1 shows thresholds for various code dimensions. Then, one can use a martingale-based argument to show that the fraction of non-recovered components concentrates around it's mean [74]. This guarantees recovery of an arbitrarily-large fraction of significant components. An expander-graph argument is then used to show that peeling continues to recover all components [70].

Table 7.1: Thresholds for recovery (see Appendix A for a derivation of these thresholds).

code dimensions	3	4	5	6	7
threshold M/K	1.2218	1.2949	1.4250	1.5697	1.7189

7.4.1 Recovering the Values and Locations in the Exactly-Sparse Setting

In the exactly-sparse setting, for each check node, we take two measurements that use complex exponentials. Define an $n \times n$ modulation matrix as $\mathbf{W}_n(x) := \mathrm{diag}\left((1, x, \ldots, x^{n-1})^\top\right)$, where $\mathrm{diag}(\cdot)$ creates a diagonal matrix out of its vector input, and consider measurements of the form

$$f_{\mathbf{A},\mathbf{B}}(\mathbf{u}\mathbf{W}_n(\omega_{n^2}^n), \mathbf{v}\mathbf{W}_n(\omega_{n^2})) = \sum_{i=1}^{n} \sum_{j=1}^{n} u_i v_j (\mathbf{a}_i^\top \mathbf{b}_j) \omega_{n^2}^{in+j},$$

where $\omega_{n^2} = \exp(\imath 2\pi / n^2)$ with $\imath$ denoting the unit complex number. If the check node corresponding to the vectors $\mathbf{u}$ and $\mathbf{v}$ picks up a single non-zero element of $\mathbf{C}$ (that is, if the check node is a *singleton*) we can find its location through a ratio test. Let $(i^\star, j^\star)$ denote the index of the non-zero element, then

$$\begin{aligned}\frac{f_{\mathbf{A},\mathbf{B}}(\mathbf{u}\mathbf{W}_n(\omega_{n^2}^n), \mathbf{v}\mathbf{W}_n(\omega_{n^2}))}{f_{\mathbf{A},\mathbf{B}}(\mathbf{u}, \mathbf{v})} &= \frac{u_{i^\star} v_{j^\star} (\mathbf{a}_{i^\star}^\top \mathbf{b}_{j^\star}) \omega_{n^2}^{i^\star n + j^\star}}{u_{i^\star} v_{j^\star} (\mathbf{a}_{i^\star}^\top \mathbf{b}_{j^\star})} \\ &= \omega_{n^2}^{i^\star n + j^\star}.\end{aligned}$$

As the result is unique for every $i^\star \in [n]$ and $j^\star \in [n]$ we can determine the index $(i^\star, j^\star)$. If this ratio is not an integer power of ω_{n^2}, it means that the measurement gets contribution from multiple non-zero elements of $\mathbf{C}$. After finding the location, the value of $\mathbf{a}_{i_0}^\top \mathbf{b}_{j_0}$ can be found as $\mathbf{C}_{i^\star, j^\star} = \mathbf{a}_{i^\star}^\top \mathbf{b}_{j^\star} = f_{\mathbf{A},\mathbf{B}}(\mathbf{u}, \mathbf{v}) / (u_{i^\star} v_{j^\star})$.

7.4.2 Recovering the Values and Locations in the Approximately-Sparse Setting

Assume that all significant components of the matrix $\mathbf{C} = \mathbf{A}^\top \mathbf{B}$ are in $\{-1, 1\}$, and that the values of all non-significant entries are smaller than $\epsilon = O(n^{-1})$ in magnitude[4]. In this setting, we construct $O(\log^2(n))$ linear combinations at each check node to detect the location and recover the value. First, we assign each row and each column a unique signature string. Let $Q = \lceil \log(n) \rceil$, and $\mathbf{b}^{(i)} \in \{0, 1\}^Q$ be the binary expansion of $i \in [n]$, that is, $i = \sum_{q=0}^{Q-1} b_q^{(i)} 2^q$. Define the matrix $S \in \{-1, 1\}^{n \times Q}$ as $\mathbf{S}_{i,q} = (-1)^{b_q^{(i)}}$, for all $i \in [n]$ and all $q \in [Q]$. The signature assigned to column or row $i \in [n]$ is then the ith row of $\mathbf{S}$. Note that this assigns each column and row a unique signature. We design auxiliary measurements at each check node that create these signatures for each row and column.

Consider a choice of $\mathbf{u}$ and $\mathbf{v}$ that defines which indices of the matrix $\mathbf{C}$ is included in a check node. Choose $P = O(\log(n))$ random vectors $\{\dot{\mathbf{u}}_p\}_{p \in [P]}$ and $\{\dot{\mathbf{v}}_p\}_{p \in [P]}$ where all have values drawn i.i.d. from Rademacher$(1/2)$. We will first find the row index and then the column index of the singleton. To find the qth entry of the signature of the row index, consider the measurements

$$\begin{aligned} y_0 &= f_{\mathbf{A},\mathbf{B}}(\mathbf{u} \odot \dot{\mathbf{u}}_p, \mathbf{v} \odot \dot{\mathbf{v}}_p), \\ y_1 &= f_{\mathbf{A},\mathbf{B}}(\mathbf{u} \odot \dot{\mathbf{u}}_p \odot \mathbf{s}_q, \mathbf{v} \odot \dot{\mathbf{v}}_p). \end{aligned} \tag{7.3}$$

If the measurement has only components at ith row we have $\operatorname{sign}(y_0) \operatorname{sign}(y_1)$ equal to $S_{i,q}$ in the exactly-sparse setting. In the approximately-sparse setting, this is true with a fixed probability. Thus repeating the procedure $P = \log(n)$ times gives an error probability that decays polynomially in n. The proof, which we omit, is based on standard concentration results [32] and is similar to [51]. In total, we need $PQ = O(\log^2(n))$ measurements to resolve singleton check nodes.

7.4.3 Computational complexity

From the previous section (and Table 7.1), we see that $O(K)$ check nodes are enough for decoding the product code. First, assume that the sketches of the matrices are obtained. Because each check node is obtained by taking the inner product of two vectors of length d, the complexity for calculating a measurement in a check node is $O(dK)$. In the exactly-sparse setting, each check node consists of 2 measurements, so overall complexity stays $O(dK)$. In the approximately-sparse setting, there are $\log^2(n)$ measurements for each check node, resulting in a computational complexity of $O(dK \log^2(n))$. For calculating the sketches of matrices, in each dimension of the product code, each entry of $\mathbf{A}$ and $\mathbf{B}$ is used once to create the check nodes. Hence, for a single measurement in a check node, we use $O(dn)$ operations. Because there are 2 measurements for each check node in the exactly-sparse setting, we get $O(dn)$ computations for calculating the sketches. In the approximately-sparse setting, there are $\log^2(n)$ measurements for each check node, so the overall complexity for sketching becomes $O(dn \log^2(n))$. Combining these, we get a computational complexity of

[4] Our algorithm can be extended to the setting where the significant components take arbitrary values using ideas from [96, 98] by increasing the computational complexity of our algorithm by a factor of $\log(K)$.

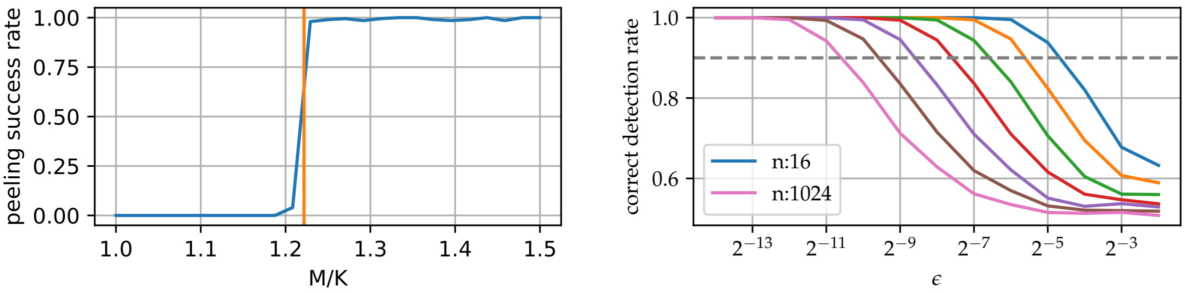


Figure 7.7: (Left) Fraction of complete recovery of the output matrix as a function of the ratio of the number of check nodes, M, to output sparsity, K. The plot assumes that we can recover the location and values from singleton check nodes. (Right) Probability of identifying an element of the signature of a singleton from measurements of the form given by (7.3) in the approximately-sparse setting as a function of the bound ϵ on the magnitude of non-significant entries for varying n.

$\max(O(dK), O(dn))$ for the exactly-sparse setting, and $\max(O(dK\log^2(n)), O(dn\log^2(n)))$ for the approximately-sparse setting.

7.5 Numerical Experiments

We present two numerical experiments corroborating our claims in Figure 7.7. The plot on the left shows the probability of complete recovery by the product code through peeling assuming we can recover entries from singleton check nodes. In the simulations $n = 10^3$ and the number of check nodes $M = 30,000$, and the output sparsity is around $K = O(n^4/3)$. Our theory states that asymptotically $M/K \approx 1.23$ measurements is enough for the recovery of all entries. As can be seen from the plot, the performance is close to the theoretical prediction. The plot on the right considers recovering the location from a singleton check node. It shows the probability of detecting the sign correctly using a pair of measurements given in (7.3) as a function of the bound ϵ on the magnitude on the non-significant entries of $\mathbf{C}$ for different values of n. Lines going from right to left depicts a doubling in n, and the dashed line is an arbitrarily chosen constant probability line (0.9 in this figure). We see that for keeping the probability of error the same when doubling n (that is, going to left from one plot to the other), one needs to halve ϵ, which means that if $\epsilon = O(n^{-1})$ the probability of correct detection stays the same. By having a constant error probability, we can repeat the process $O(\log n)$ times to get polynomially decaying error probability for recovering each entry of the signature of a non-zero location.

7.6 Conclusions and Future Directions

We have presented a fast algorithm for large-scale matrix-matrix multiplication when the output matrix is known to be sparse. Our algorithm is based on transforming the problem of support recovery of a matrix product to decoding a product code on an erasure channel. Through this formulation, we utilize a peeling-based algorithm to recover the support of the matrix product. Our problem setting arises in a variety of problems ranging from the recovery of sparse covariance matrices to fast string search across a database (where the rows of the first matrix represent the database of strings and the columns of the second matrix represent the query strings to be searched in the database) to sketching sparse matrices. Tailoring our method to these applications is a part of future research.

Chapter 8

Conclusions and Future Research Directions

In this thesis, we have studied six problems of practical and theoretical interest, and developed fast algorithms for solving them whenever there is some underlying sparse structure. This sparse structure can be that (1) in a signal processing application, the input signal is representable with a small number of non-zero coefficients in some basis, (2) in a distributed computing setting, the number of straggling machines is small, or (3) in performing matrix-matrix multiplication, the output matrix is sparse. The algorithms we developed attained their speed by being sample and computationally efficient through leveraging the structure of sparsity.

Our approach in tackling the problems was based on a unifying divide-and-conquer strategy where by leveraging ideas from Low Density Parity Check (LDPC) Codes and Product Codes which has previously been studied in the context of communications, we created structures that enable simple and fast *peeling*-based recoveries. In particular, the problems we considered and a summary of our results therein were:

- We proposed a novel architecture for sampling a continuous-time signal having a band-limited and sparse spectrum at its minimum sampling rate. This minimum rate is equal to twice the occupancy of its spectrum, which can be much lower than the rate given by the Shannon-Nyquist sampling theorem.

- We improved the noise robustness of peeling-based sparse DFT algorithm R-FFAST by developing a sample and computationally efficient single-tone identification algorithm. This helps pave the way for applications that require operating at low signal to noise ratios.

- We developed a fast algorithm for recovering sparse wavelet coefficients of a signal based on samples from its Fourier Transform. We analyzed our algorithm by deriving novel density evolution equations that takes into account cycles on the code graph. Using our analysis, we showed that the sample complexity of our algorithms is linear in the number of non-zero wavelet coefficients of the signal and the computational complexity requires only a term that depends logarithmically on the sparsity on top of the linear term. This setting arises in MRI, and our approach provides a theoretical framework for fast MRI.

- We proposed a sample and computationally efficient algorithm for recovering sparse pseudo-Boolean functions. We showed that there is an interesting connection between recovering low-degree pseudo-Boolean functions and the minimum distance properties of codes from communications literature. We applied a variant of our algorithm to learn fitness landscapes of DNA repair outcomes after CRISPR-Cas9 induced breaks.

- We proposed a coded computation method to speed up distributed matrix-matrix multiplication by reducing the effects of straggler nodes (slow worker nodes). Our method uses a novel code construction based on Product Codes in its channel coding form. Our code construction attains a good balance between straggler resiliency and having computationally efficient encoding/decoding at the master node.

- We propose a computationally efficient algorithm for large-scale matrix-matrix multiplication whenever the output matrix is sparse. Our method uses product codes in its dual compression form to obtain orderwise-optimal number of sketches from the output matrix, and then recovers it by fast peeling-based decoder.

Future research directions can be related to the overall unifying approach or problem-specific. First we provide directions about the overall approach. Unifying our algorithms is a divide-and-conquer strategy. We divide a hard problem into smaller easier problems using a sparse-graph code, such as an LDPC code or a product code, then recover the solution to the original problem by peeling, that is, by recovering solutions to the easier problems iteratively. This divide-and-conquer strategy is designed to work in a non-adversarial setting, and variations of our algorithms tailored for an adversarial setting might be of practical and theoretical interest. Another property of our algorithms is that while combining the solutions to the easier problems, we make local decisions. Alternatively, convex-optimization based algorithms for sparse-signal recovery, such as LASSO [86], make global computations. This results in higher robustness to non-idealities like noise or approximate sparsity. However, as a result, the complexity of such algorithms also scale super-linearly with the ambient dimension. Hybrid approaches that combine local and global operations, and hence, trading computational complexity with robustness against non-idealities are interesting to pursue.

Next, we provide problem-specific future directions.

In Chapter 2, we studied recovering band limited signals whose frequency occupation is exactly sparse. Many continuous-time signals, such as signals which have finite duration in time, are not bandlimited. However, such signals, if they have finite energy, have a finite bandwidth where most of the energy of the signal is concentrated [82, 83]. One future research direction can be to develop algorithms that recover arbitrarily large fraction of the signal energy for non-bandlimited signals or signals that has approximately-sparse spectra.

In Chapter 3, we presented an algorithm for single tone identification for improving the noise robustness of R-FFAST [69] algorithm for sparse DFT calculations. Our hardware implementation [93] of R-FFAST [69] uses the single tone identification algorithm based on Kay's method. Implementing the new single tone identification algorithm in hardware could improve the SNR performance and open up the application areas.

In Chapter 4, our results show that a strategy having some non-random components in the graph design can lead to better performance than the fully random case. An interesting open direction is the pursuit of optimal ball-throwing strategy. In our work, we have given initial steps in analyzing this problem rigorously. We additionally believe the gains seen in spatially coupled ensembles [41] can be related to the structural gains of the smearing ensemble; an interesting open problem is to study the connections.

In Chapter 5, we have shown that codes that have minimum distance greater than $2t$ enables recovery of all coefficients corresponding to multinomial terms of degree up to and including t. However, what is the optimal choice of the code if there is sparsity on the low-degree terms, and that we do not need to recover all of them but only the non-zero ones? If we use the codewords corresponding to a code with minimum distance less than or equal to $2t$, there will be coefficients of monomials with degree less than or equal to t which alias together in the same bins. As each coefficient is equally likely to be non-zero under the uniform-support assumption, intuition suggests that codes that minimize the maximum number of aliasing coefficients should perform well. An interesting future direction is to provide theoretical results and practical deterministic code constructions for this setting.

In Chapter 6, as we have utilized product codes which can be decoded by operating on single 'rows' and 'columns' in an iterative way, our algorithm is a suitable candidate to have an impact in master-less computation frameworks. In particular, in a platform having shared memory with fast input/output where workers write their results, separate workers assigned to the task of decoding can do so in a streamlined fashion. This master-less architecture is an exciting new computing paradigm which Product Codes are well suited for.

In Chapter 7, our algorithm for sparse-output matrix multiplication also uses product codes, and it can be combined with the algorithm from Chapter 6 for being straggler-resilient in a distributed computing setting. Furthermore, sparse-output matrix multiplication arises in a variety of problems ranging from the recovery of sparse covariance matrices to fast string search across a database (where the rows of the first matrix represent the database of strings and the columns of the second matrix represent the query strings to be searched in the database) to sketching sparse matrices. Tailoring our method to these application areas is another possible research direction.

Appendix A

Analysis of the Peeling Decoder

Here, we cover the theoretical analysis of decoding a sparse-graph code through peeling. The outline of the steps is as follows. First, a *graph ensemble* on which we perform peeling is defined, and the analysis is made for that ensemble. While decoding, at each step of peeling, we remove edges that are connected to singleton check nodes from the graph. Hence, we need to make sure that there are singleton check nodes until all the variable nodes are resolved. Let p_ℓ denote the fraction of edges that are remaining in the graph after a fixed ℓ rounds of peeling. We can derive *density evolution equations* that track the evolution of p_ℓ through the iterations of peeling (Section A.1). These density evolution equations are derived assuming that the depth ℓ neighborhood of an edge is *cycle-free*. Through counting arguments, the depth ℓ neighborhood of a randomly chosen edge is shown to be cycle-free with high probability for any fixed ℓ (Lemma 10). If p_ℓ goes arbitrarily close to 0 as ℓ increases, we can remove an arbitrarily large fraction of edges on average over the ensemble of graphs. Then, one can use a martingale-based argument to show that the fraction of non-recovered components concentrates around this average (Lemma 10). This guarantees recovery of an arbitrarily-large fraction of significant components with high probability. Then, an expander-graph argument is then used to show that peeling continues to remove all edges (Section A.3).

We have the following graph ensemble similar to the Definition 3. The slight changes in the notation is to have this appendix abstracted away from the application.

Definition 7. Given K left nodes and M right nodes and an integer d with $d \geq 2$, regular graph ensemble $C(K, M, d)$ is the set of all graphs where each left node is connected to d right nodes.

Figure A.1 shows an example graph from $C(K, M, d)$ with $K = 4$, $M = 8$, and $d = 2$. We have the following theorem adapted from Theorem 9 in [70].

Theorem 11. *The peeling decoder over a random graph from the ensemble $C(K, M, d)$ with $d \geq 3$, and $M = \eta K$ where η is given in Table A.1*

(a) *recovers all the variable nodes with probability at least $1 - O(1/K)$,*

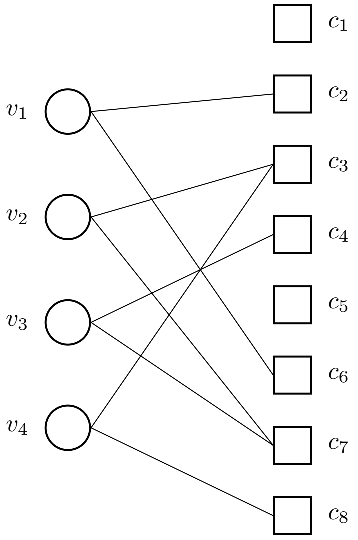


Figure A.1: An example graph from $C(K, M, d)$ with $K = 4$ left nodes (variable nodes), $M = 8$ right nodes (check nodes), and left node degree $d = 2$.

(b) recovers all but a vanishingly small fraction of variable nodes with probability at least $1 - \exp\bigl(-\gamma\epsilon^2 K^{1/(2\ell+1)}\bigr)$ *for some constants* γ*,* $\epsilon > 0$*,* $\ell > 0$ *.*

Proof. In Lemma 8 we show the result in (b), combining it with the result in Section A.3 we get the result in (a). □

Table A.1: Thresholds for density evolution on $C(K, M, d)$ as a function of the degree d and $\eta = M/K$. The calculation of these thresholds are explained in Section A.1.

d	3	4	5	6	7
η	1.2218	1.2949	1.4250	1.5697	1.7189

A.1 Density Evolution for Cycle-Free Case

We derive density evolution equations that track the average behavior of the peeling decoder. For this, let

$$\lambda(x) := \sum_{i=1}^{\infty} \lambda_i x^{i-1}, \tag{A.1}$$

where λ_i denotes the probability that an edge of the graph is connected to a left node of degree i. Likewise, let

$$\rho(x) := \sum_{i=1}^{\infty} \rho_i x^{i-1}, \tag{A.2}$$

where ρ_i denotes the probability that an edge of the graph is connected to a right node of degree i. Because the graphs in $C(K, M, d)$ are left regular with degree d, we have

$$\lambda(x) = x^{d-1}. \tag{A.3}$$

Next, is to calculate ρ. Let D_c be the degree of a check node c, then the expected number of right nodes with degree j is

$$\sum_{c=1}^{M} \Pr\left(D_c = j\right) = M\Pr\left(D_1 = j\right), \tag{A.4}$$

by symmetry. Then p_j can be calculated as

$$\rho_j = \frac{\Pr\left(D_1 = j\right) Mj}{Kd} = \frac{j\eta}{d}\Pr\left(D_1 = j\right), \tag{A.5}$$

where we used $M = \eta K$. According to $C(K, M, d)$, D_1 follows the binomial distribution $B(Kd/\eta, d)$, and it can be approximated well by a Poisson distribution

$$\Pr\left(D_1 = j\right) \approx \frac{(d/\eta)^j e^{d/\eta}}{j!}. \tag{A.6}$$

As a result, ρ_j, the fraction of edges connected to a right node having degree j is

$$\rho_j = \frac{(d/\eta)^{j-1} e^{d/\eta}}{(j-1)!}. \tag{A.7}$$

This gives

$$\rho(x) = e^{-(d/\eta)(1-x)}. \tag{A.8}$$

For deriving the density evolution equations we introduce the concept of directed neighborhood (see Figure A.2). A directed edge $\vec{e}$ means an ordered pair (c, v) or (v, c) where c is a check node and v is a variable node in the graph. A path in the graph is a sequence of directed edges $\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_t$ such that for every $i \in \{1, \ldots, t-1\}$ if we have $\vec{e}_i = (u_i, u'_i)$ and $\vec{e}_{i+1} = (u_{i+1}, u'_{i+1})$, then $u_{i+1} = u'_i$. The length of the path is the number of edges in it. The directed neighborhood of depth ℓ of $\vec{e} = (v, c)$, denoted by $N^{\ell}_{\vec{e}}$, is defined as the induced subgraph containing all edges and nodes on paths $\vec{e}_1, \ldots, \vec{e}_\ell$ starting at node v such that $\vec{e}_1 \neq \vec{e}$. An example of directed neighborhood is given in Figure A.2.

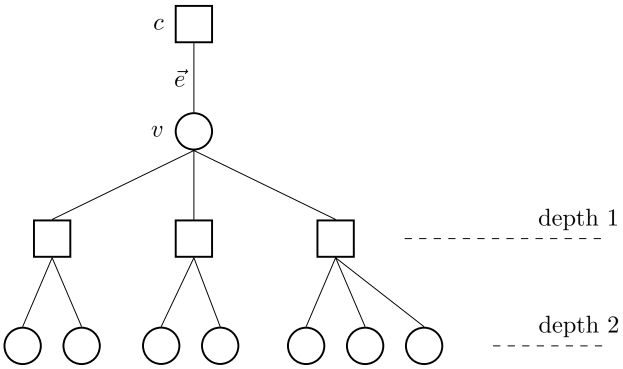


Figure A.2: Illustration of $N^2_{\vec{e}}$, the depth 2 neighborhood of a directed edge $\vec{e} = (v, c)$.

Assume that the depth 2ℓ neighborhood of an edge in the graph is cycle-free. Let p_j denote the probability that an edge is present after j rounds of peeling. Probability that a check node connects to no variable nodes from below (see Figure A.2) is then $\rho(1 - p_j)$. The probability of one check node that the variable node is connected to being not a singleton is then $1 - \rho(1 - p_j)$. Since there are multiple check nodes that the variable node connects to from below, we have all of them being not a singleton to be $\lambda(1 - \rho(1 - p_j))$. Hence, under cycle-free assumption we have

$$p_{j+1} = \lambda(1 - \rho(1 - p_j)), \text{ for } j = 0, 1, \ldots, \ell - 1. \tag{A.9}$$

Substituting in λ and ρ from equations (A.3) and (A.8) in the recurrence (A.9) we get

$$p_{j+1} = \left(1 - e^{-\frac{d}{\eta} p_j}\right)^{d-1}, \text{ for } j = 0, 1, \ldots, \ell - 1. \tag{A.10}$$

This equation tracks the expectation of an edge to remain in the graph after j rounds of peeling. If we are hoping to recover all variable nodes, all edges in the graph need to be removed. This would hold if p_j goes to 0 as the number of iterations j increases. However, this requires a certain relation between η and d. If $p_{j+1} < p_j$ for all $j \geq 0$, then p_j goes to 0 as j increases. Solving for η that satisfies this property given d requires using numerical methods (such as searching using bisection method). One way is to plot the evolution and see if it crosses the $p_{j+1} = p_j$ line. Figure A.3 plots the evolution of p_j given in (A.9) for $d = 3$. As can be seen, $\eta = 1.2218$ just satisfies this property, and anything below this value crosses above the $p_{j+1} = p_j$. The values in the Table A.1 are numerically calculated for different values of d using equation (A.10).

A.2 Convergence to Cycle-Free Case

We have the following lemma which says that the expected number of edges removed from the graph is tracked well by the density evolution equations, and that the performance concentrates tightly around this expected value.

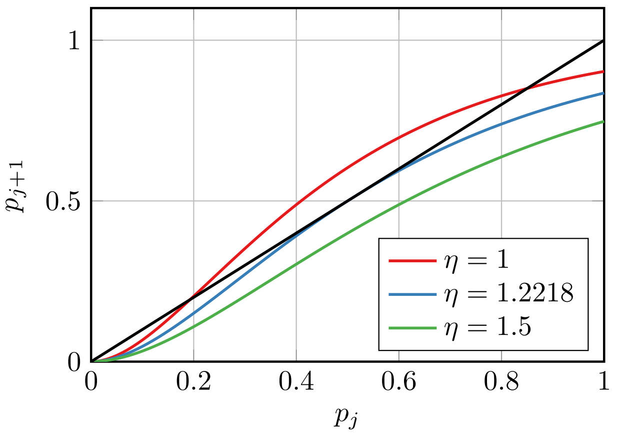


Figure A.3: Evolution of p_j in equation A.10 for different values of $\eta = M/K$ for $d = 3$. We see that when $\eta = 1$, the curve crosses above the $p_{j+1} = p_j$ line. The smallest value of η for which the curve is always below the $p_{j+1} = p_j$ line is $\eta = 1.2218$.

Lemma 8. *Let Z be the total number of edges that are removed after ℓ iterations of the peeling decoder over a randomly chosen graph from $C(K, M, d)$. Further, let p_ℓ be given as in equation* (A.10)*. Then, there exists a constant γ such that for any $\epsilon > 0$ and sufficiently large K we have*

$$\mathbb{E}[Z] < 2Kdp_\ell \tag{A.11}$$

and

$$\Pr\left(|Z - \mathbb{E}[Z]| > kd\epsilon\right) < \exp\left(-\gamma\epsilon^2 k^{1/(2\ell+1)}\right). \tag{A.12}$$

The proof follows the lines of the proofs of Theorem 2 in [74] and Lemma 5 in [70].

Proof. Let Z_e for $e \in \{1, \ldots, dK\}$ be the indicator that eth edge remains on the graph after ℓ iterations of peeling. The number of edges remaining in the graph after ℓ rounds of peeling is then

$$Z = \sum_{e=1}^{dK} Z_e. \tag{A.13}$$

By symmetry, we have $\mathbb{E}[Z] = dK\mathbb{E}[Z_1]$. We bound $\mathbb{E}[Z_1]$ as,

$$\begin{aligned}\mathbb{E}[Z_1] &= \mathbb{E}\left[Z_1 | N_{\vec{e}_1}^{2\ell} \text{ is cycle-free}\right] \Pr\left(N_{\vec{e}_1}^{2\ell} \text{ is cycle-free}\right) \\ &\quad + \mathbb{E}\left[Z_1 | N_{\vec{e}_1}^{2\ell} \text{ is not cycle-free}\right] \Pr\left(N_{\vec{e}_1}^{2\ell} \text{ is not cycle-free}\right) \\ &\le \mathbb{E}\left[Z_1 | N_{\vec{e}_1}^{2\ell} \text{ is cycle-free}\right] + \Pr\left(N_{\vec{e}_1}^{2\ell} \text{ is not cycle-free}\right).\end{aligned} \tag{A.14}$$

In Lemma 10, we show that $\Pr\left(N_{\vec{e}_1}^{2\ell} \text{ is not cycle-free}\right) \leq O(\log^{2\ell}(K)/K)$. We also have, from the definition of density evolution equations $\mathbb{E}\left[Z_1 | N_{\vec{e}_1}^{2\ell} \text{ is cycle-free}\right] = p_\ell$. Thus, for sufficiently large K we have

$$\mathbb{E}\left[Z\right] < 2Kdp_\ell,$$

which proves equation A.11.

Next, we show the concentration of Z around its mean. The number of remaining edges is given by (A.13). Note that the summands are not independent with each other. Therefore, we cannot use concentration of independent random variables. Thus, we resort to a martingale-based argument similar to [74]. Suppose that we expose the whole set of Kd edges of the graph one at a time. We let

$$Y_i = \mathbb{E}\left[Z | Z_1, \ldots, Z_i\right], \text{ for } i \in \{0, 1, \ldots, Kd\}. \tag{A.15}$$

By definition, $Y_0, Y_1, \ldots, Y_{Kd}$ is a Doob's martingale process, where $Y_0 = \mathbb{E}\left[Z\right]$ and $Y_{Kd} = Z$. To use Azuma's inequality to show concentration, we require that $|Y_{i+1} - Y_i| \leq \delta_i$ for some $\delta_i > 0$. Then, Azuma's inequality would state that

$$\Pr\left(|Z - \mathbb{E}\left[Z\right]| > t\right) \leq 2\exp\left(\frac{-t^2}{2\sum_{i=1}^{dK} \delta_i^2}\right). \tag{A.16}$$

In [74], the authors have shown that for regular bipartite graphs with variable node degree d_v and check node degree d_c we have

$$\delta_i < 8(d_v d_c)^\ell, \tag{A.17}$$

where ℓ is the number of iterations. However, since the check node degree is not regular in our case, we need further analysis.

Let D_c be the degree of check node c; because D_c follows a Poisson distribution with a fixed mean of $1/\eta$, we have

$$\Pr\left(D_c > \kappa_1 K^{2/(4\ell+1)}\right) \overset{(a)}{\leq} \left(\frac{e}{\eta\kappa_1 K^{2/(4\ell+1)}}\right)^{\kappa_1 K^{2/(4\ell+1)}} \leq \kappa_2 \exp\bigl(-\kappa_3 K^{2/(4\ell+1)}\bigr), \tag{A.18}$$

for some κ_2 and κ_3, where (a) follows from Claim 1 below. Let $\bar{B}$ be the event that there is a check node with degree $> \kappa_1 K^{2/(4\ell+1)}$. Then, by union bound over all ηK check nodes, we have

$$\Pr\left(\bar{B}\right) \leq O\bigl(K\exp\bigl(-\kappa_3 K^{2/(4\ell+1)}\bigr)\bigr). \tag{A.19}$$

Under the complement of $\bar{B}$, substituting the upper bound on the check node degree in equation (A.17), we have

$$\delta_i^2 = O\left(K^{4\ell/(4\ell+1)}\right). \tag{A.20}$$

Substituting this into Azuma's inequality, we find that

$$\begin{aligned}\Pr\left(|Z-\mathbb{E}\left[Z\right]|>Kd\epsilon\right) &\leq \Pr\left(|Z-\mathbb{E}\left[Z\right]|>dK\epsilon|B\right)+\Pr\left(\bar{B}\right)\\ &\leq 2\exp\left(\frac{-K^2d^2\epsilon^2}{2\sum_{i=1}^{dK}\delta_i^2}\right)+O\left(K\exp\bigl(-\kappa_3K^{2/(4\ell+1)}\bigr)\right) \qquad \text{(A.21)}\\ &\leq \exp\bigl(-\gamma\epsilon^2K^{1/(4\ell+1)}\bigr),\end{aligned}$$

for some γ, which proves equation (A.12). □

The following lemma bounds the size of the set of variable nodes and the set of check nodes in the neighborhood $N_{\vec{e}}^{2\ell}$ with high probability. The sizes of these sets will be used to bound the probability of $N_{\vec{e}}^{2\ell}$ not being cycle-free.

Lemma 9. *Let C_ℓ be the number of check nodes and V_ℓ be the number of variable nodes in the neighborhood $N_{\vec{e}}^{2\ell}$. Then*

$$\Pr\left(V_\ell>\kappa_1\log^\ell K\right)\leq O(1/K), \qquad \text{(A.22)}$$

$$\Pr\left(C_\ell>\kappa_1\log^\ell K\right)\leq O(1/K). \qquad \text{(A.23)}$$

Proof. Let $\alpha_i := \Pr\left(V_i>\kappa_1\log^i K\right)$. We have

$$\alpha_i\leq\alpha_{i-1}+\Pr\left(V_i>\kappa_1\log^i K|V_{i-1}<\kappa_1\log^{i-1}K\right). \qquad \text{(A.24)}$$

Given $V_{i-1}<\kappa_1\log^{i-1}K$, we have $C_i<\kappa_2\log^{i-1}K$ at depth i for some $\kappa_2>0$ since left degree is bounded by d. Hence, we can bound the second term as

$$\Pr\left(V_i>\kappa_1\log^i K|V_{i-1}<\kappa_1\log^{i-1}K\right)\leq\Pr\left(V_i>\kappa_1\log^i K|C_i<\kappa_2\log^{i-1}K\right). \qquad \text{(A.25)}$$

Let the number of check nodes at exactly depth i be $\breve{C}_i$, and D_j be the degrees of each of these check nodes. We have

$$\Pr\left(V_i>\kappa_1\log^i K|C_i<\kappa_2\log^{i-1}K\right)\leq\Pr\left(\sum_{j=1}^{\breve{C}_i}D_j\geq\kappa_3\log^i K\right) \qquad \text{(A.26)}$$

for some $\kappa_3>0$. Since the check node degrees are distributed as Poisson random variables with rate $1/\eta$, and sum of Poisson random variables are also distributed as Poisson, and $\breve{C}_i\leq C_i\leq\kappa_2\log^{i-1}K$,

$$\Pr\left(\sum_{j=1}^{\breve{C}_i}D_j\geq\kappa_3\log^i K\right)\overset{(a)}{\leq}\left(\frac{e\breve{C}_i/\eta}{\kappa_3\log^i K}\right)^{\kappa_3\log^i K}\leq\left(\frac{\kappa_4}{\log K}\right)^{\kappa_3\log^i K}\leq\frac{\kappa_5}{K}, \qquad \text{(A.27)}$$

for some $\kappa_4 > 0$ and $\kappa_5 > 0$, where (a) follows from Claim 1. Therefore, we have

$$\alpha_i \leq \alpha_{i-1} + \frac{\kappa_5}{K}, \tag{A.28}$$

and thus V_ℓ satisfies

$$\Pr\left(V_\ell > \kappa_1 \log^\ell K\right) \leq O(1/K), \tag{A.29}$$

for each fixed ℓ. Similar analysis can be performed to prove the bound on C_ℓ. □

The following lemma bounds the probability of $N_{\vec{e}}^{2\ell}$ not being cycle-free.

Lemma 10. *Consider a randomly selected edge $\vec{e}$ in a graph G chosen from $C(K, M, d)$. We have*

$$\Pr\left(N_{\vec{e}}^{2\ell} \textit{ is not cycle-free}\right) \leq O\left(\frac{(\log K)^{2\ell}}{K}\right). \tag{A.30}$$

Proof. Let C_i be the number of check nodes, and V_i be the number of variable nodes present in $N_{\vec{e}}^{2i}$. We bound the probability as

$$\begin{aligned}\Pr\left(N_{\vec{e}_1}^{2\ell} \text{ is not cycle-free}\right) \leq & \Pr\left(V_\ell > \kappa_1 \log^\ell K\right) + \Pr\left(C_\ell > \kappa_1 \log^\ell K\right) \\ & + \Pr\left(N_{\vec{e}_1}^{2\ell} \text{ is not cycle-free} | V_\ell \leq \kappa_1 \log^\ell K, C_\ell \leq \kappa_1 \log^\ell K\right).\end{aligned} \tag{A.31}$$

From Lemma 9, we have a probabilistic bound on the number of check nodes and variable nodes in $N_{\vec{e}}^{2\ell}$, so the first two terms are $O(1/K)$.

Next, We focus on the last term. Assume depth $2i$, where $i < \ell$, is cycle-free. Also, assume that t more edges from the leaf variable nodes in $N_{\vec{e}}^{2i}$ to the check nodes are revealed without creating a loop. Then, the probability that the next revealed edge from a leaf variable node to a check node does not create a loop is

$$1 - \frac{C_\ell}{\eta K}. \tag{A.32}$$

Therefore, given that $N_{\vec{e}}^{2i}$ is cycle-free, the probability that $N_{\vec{e}}^{2i+1}$ is cycle-free is lower bounded by

$$\left(1 - \frac{C_\ell}{\eta K}\right)^{C_{i+1} - C_i}. \tag{A.33}$$

By similar reasoning, given that $N_{\vec{e}}^{2i+1}$ is cycle-free, the probability that $N_{\vec{e}}^{2(i+1)}$ is cycle free is lower bounded by

$$\left(1 - \frac{V_\ell}{K}\right)^{V_{i+1} - V_i}. \tag{A.34}$$

Therefore, the probability that $N_{\vec{e}}^{2\ell}$ being not-cycle-free is bounded by

$$1-\left(1-\frac{C_\ell}{\eta K}\right)^{C_\ell}\left(1-\frac{V_\ell}{K}\right)^{V_\ell} \leq \left(\frac{V_\ell^2}{K}+\frac{C_\ell^2}{\eta K}\right) \leq O\left(\frac{\log^{2\ell}}{K}\right), \tag{A.35}$$

where the last inequality follows from the conditioning $V_\ell = O\big(\log^\ell K\big)$ and $C_\ell = O\big(\log^\ell K\big)$. Combining all the terms in (A.31) concludes the proof. □

The following claim provides a bound on the probability of tail events of Poisson random variables which we used in the proofs of Lemma 8 and Lemma 9.

Claim 1. *If a random variable D has a Poisson distribution with mean μ, then*

$$\Pr\left(D \geq d\right) \leq \left(\frac{e\mu}{d}\right)^d \tag{A.36}$$

for all $d > \mu$.

Proof. The moment generating function of D is $\mathbb{E}\left[e^{\theta D}\right] = e^{\mu(e^\theta - 1)}$ for all $\theta \in \mathbb{R}$. We have

$$\begin{aligned}\Pr\left(D \geq d\right) &= \Pr\left(e^{\theta D} \geq e^{\theta d}\right)\\ &\leq e^{\mu(e^\theta-1)-\theta d}.\end{aligned} \tag{A.37}$$

Minimizer of this over $\theta > 0$ is $\theta^* = \log(d/\mu)$, which is greater than 0 for $d > \mu$. Substituting in the equation we get

$$\begin{aligned}e^{\mu(e^{\theta^*}-1)-\theta^* d} &= e^{d-\mu}\left(\frac{\mu}{d}\right)^d\\ &\leq \left(\frac{e\mu}{d}\right)^d,\end{aligned} \tag{A.38}$$

which proves the claim. □

A.3 Expander Graph Property and Complete Recovery of Variable Nodes

Definition 8. Let G be a bipartite graph with K left nodes and M right nodes. We call G an (α, β) expander, if for all subsets $\mathcal{S}$ of left nodes of size at most αK we have $|N(S)| > \beta\,|\mathcal{S}|$, where $N(\mathcal{S})$ is the neighborhood of $\mathcal{S}$.

The following lemma shows that if the graph is an appropriate expander, then the peeling continues to recover all variable nodes given that it has recovered a sufficiently large fraction of them.

Lemma 11. *Consider a graph from the ensemble $C(K, M, d)$ that is an $(\alpha, d/2)$ expander for some $\alpha > 0$. If the peeling-decoder over this graph succeeds in decoding all but at most αK variable nodes, then it decodes all the variable nodes.*

Proof. Let $\mathcal{S}$ be the set of the variable nodes that the peeling decoder fails to decode. We have $\mathcal{S} \leq \alpha K$ and $|N(\mathcal{S})| > d|\mathcal{S}|/2$ by the assumption of the lemma. Note that the peeling decoder fails to decode the set $\mathcal{S}$ if and only if there are no more singleton check nodes in $N(\mathcal{S})$. Assume all check nodes in $N(\mathcal{S})$ to be a multiton, the number of edges connecting to the check nodes in the set $N(\mathcal{S})$ have to be at least $2|N(\mathcal{S})| > d|\mathcal{S}|$. This is a contradiction since there are only $d|\mathcal{S}|$ edges going from set $\mathcal{S}$ to $N(\mathcal{S})$ by construction. Hence, there must be at least one singleton check node to continue the peeling process. □

The following lemma shows that graphs from the ensemble $C(N, K, d)$ is an expander with high probability for $d \geq 3$.

Lemma 12. *Consider a graph G randomly selected from the ensemble $C(N, K, d)$ with $d \geq 3$. Then G is an $(\alpha, d/2)$ expander with probability at least $1 - O(1/K)$ for sufficiently small but constant α.*

Proof. Consider $\mathcal{S}$ of variable nodes in a random graph G from $C(K, N, d)$ for $d \geq 3$. Let $N(\mathcal{S})$ be its neighbors, and let $E_{\mathcal{S}}$ denote the event $|N(\mathcal{S})| \leq d|\mathcal{S}|/2$. We have

$$\begin{aligned}\Pr(E_{\mathcal{S}}) &< \binom{M}{d|\mathcal{S}|/2}\left(\frac{d|\mathcal{S}|}{2M}\right)^{d|\mathcal{S}|/2} \\ &\leq \left(\frac{d|\mathcal{S}|e}{2N}\right)^{d|\mathcal{S}|/2}.\end{aligned} \tag{A.39}$$

Let E_v be the probability that a subset of size v of variable nodes in G having neighborhood of size less than or equal to $\frac{dv}{2}$. By union bound, we have

$$\begin{aligned}\Pr(E_v) &< \binom{K}{v}\left(\frac{dve}{2N}\right)^{dv/2} \\ &< \left(\frac{v}{K}\right)^{(d/2-1)v} c^v \leq \left(\frac{vc^2}{K}\right)^{v/2},\end{aligned} \tag{A.40}$$

where $c = e(de/(2\eta))^{d/2}$ is some constant for fixed d. Union bounding over all possible values v up to $\alpha^* K$, where we choose $\alpha^* < 1/(2c^2)$ gives

$$\sum_{v=2}^{\alpha^* K} \Pr(E_v) \leq \sum_{v=2}^{\alpha^* K}\left(\frac{vc^2}{K}\right)^{v/2} = O(1/K), \tag{A.41}$$

which concludes the proof. □

Bibliography

[1] N. Abramson, “Cascade decoding of cyclic product codes,” *IEEE Transactions on Communication Technology*, vol. 16, no. 3, pp. 398–402, Jun. 1968.

[2] A. Aghazadeh, O. Ocal, and K. Ramchandran, “CRISPRLand: Interpretable large-scale inference of DNA repair landscape based on a spectral approach,” in *28th Conference on Intelligent Systems for Molecular Biology (ISMB)*, Jul. 2020, (forthcoming, accepted).

[3] F. Allen, L. Crepaldi, C. Alsinet, A. J. Strong, V. Kleshchevnikov, P. De Angeli, P. Páleníková, A. Khodak, V. Kiselev, M. Kosicki, *et al.*, “Predicting the mutations generated by repair of Cas9-induced double-strand breaks,” *Nature Biotechnology*, vol. 37, no. 1, p. 64, 2019.

[4] T. Baharav, K. Lee, O. Ocal, and K. Ramchandran, “Straggler-proofing massive-scale distributed matrix multiplication with d-dimensional product codes,” in *IEEE Int. Symp. on Inf. Theory*, Jun. 2018, pp. 1993–1997.

[5] T. Baharav and K. Ramchandran. (Jan. 2018). Leaving randomness to nature: d-dimensional product codes through the lens of Generalized-LDPC codes, [Online]. Available: `https://people.eecs.berkeley.edu/~kannanr/assets/high-dim-prod-codes.pdf`.

[6] M. Bayati and A. Montanari, “The dynamics of message passing on dense graphs, with applications to compressed sensing,” *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 764–785, Jan. 2011.

[7] A. Beck and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” *SIAM J. Imaging Sci.*, vol. 2, no. 1, pp. 183–202, 2009.

[8] R. E. Blahut, *Algebraic Codes for Data Transmission*. Cambridge University Press, 2003.

[9] R. C. Bose and D. K. Ray-Chaudhuri, “On a class of error correcting binary group codes,” *Information and control*, vol. 3, no. 1, pp. 68–79, 1960.

[10] Y. Bresler, “Spectrum-blind sampling and compressive sensing for continuous-index signals,” *Inf. Theory and Appl. Workshop, 2008*, pp. 547–554, 2008.

[11] K. Chandrasekher, O. Ocal, and K. Ramchandran, “Density evolution on a class of smeared random graphs,” in *IEEE Int. Symp. on Inf. Theory*, Jun. 2017, pp. 2915–2919.

[12] ——, “Density evolution on a class of smeared random graphs: A theoretical framework for fast MRI,” *CoRR*, vol. abs/1705.02453, 2017. arXiv: `1705.02453`.

[13] G. C. Clark and J. B. Cain, *Error-Correction Coding for Digital Communications*, 1st. Springer Publishing Company, Incorporated, 2013.

[14] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–297, 1965.

[15] G. Dasarathy, P. Shah, B. N. Bhaskar, and R. D. Nowak, "Sketching sparse matrices, covariances, and graphs via tensor products," *IEEE Trans. Inf. Theory*, vol. 61, no. 3, pp. 1373–1388, 2015.

[16] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, pp. 74–80, 2013.

[17] F. Didier, "Efficient erasure decoding of Reed-Solomon codes," *CoRR*, vol. abs/0901.1886, 2009. arXiv: `0901.1886`.

[18] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[19] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proc. Natl. Acad. Sci. USA*, vol. 106, no. 45, pp. 18 914–18 919, Sep. 2009.

[20] ——, "Message passing algorithms for compressed sensing: I. motivation and construction," in *IEEE Inf. Theory Workshop on Inf. Theory*, IEEE, Jan. 2010, pp. 1–5.

[21] S. Dutta, V. R. Cadambe, and P. Grover, "Coded convolution for parallel and distributed computing within a deadline," *CoRR*, vol. abs/1705.03875, 2017. arXiv: `1705.03875`.

[22] Ö. Eğecioğlu, E. Gallopoulos, and Ç. K. Koç, "A parallel method for fast and practical high-order Newton interpolation," *BIT*, vol. 30, no. 2, pp. 268–288, Jun. 1990.

[23] P. Elias, "Error-free coding," *Transactions of the IRE Professional Group on Information Theory*, vol. 4, no. 4, pp. 29–37, Sep. 1954.

[24] P. Feng and Y. Bresler, "Spectrum-blind minimum-rate sampling and reconstruction of multiband signals," in *IEEE Int. Conf. on Acoust., Speech, and Signal Process.*, vol. 3, Atlanta: IEEE, May 1996, 1688–1691 vol. 3.

[25] M. P. Fitz, "Further results in the fast estimation of a single frequency," *IEEE Transactions on Communications*, vol. 42, no. 234, pp. 862–864, 1994.

[26] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory,*, vol. 8, no. 1, pp. 21–28, 1962.

[27] A. Gilbert and P. Indyk, "Sparse recovery using sparse matrices," *Proc. IEEE*, vol. 98, no. 6, pp. 937–947, Jun. 2010.

[28] M. J. E. Golay, "Notes on digital coding," *Proceedings of the Institute of Radio Engineers*, vol. 37, pp. 657–657, Jun. 1949.

[29] A. Goldsmith, *Wireless Communications*. Cambridge University Press, 2005.

[30] C. Häger, H. D. Pfister, F. Brännström, *et al.*, “Density evolution for deterministic generalized product codes on the binary erasure channel,” *arXiv preprint arXiv:1512.00433*, Dec. 2015.

[31] A. Hocquenghem, “Codes correcteurs d’erreurs,” *Chiffres*, vol. 2, no. 2, pp. 147–56, 1959.

[32] W. Hoeffding, “Probability inequalities for sums of bounded random variables,” in *The Collected Works of Wassily Hoeffding*, Springer, 1994, pp. 409–426.

[33] A. Hormati and M. Vetterli, “Annihilating filter-based decoding in the compressed sensing framework,” in *Wavelets XII*, ser. Proc. SPIE, Aug. 2007, pp. 670 121–670 121.

[34] K. Ireland and M. Rosen, *A Classical Introduction to Modern Number Theory*. Springer-Verlag New York, 1990, vol. 84.

[35] N. T. Janakiraman, A. Vem, K. R. Narayanan, and J.-F. Chamberland, “Sub-string/pattern matching in sub-linear time using a sparse Fourier transform approach,” *arXiv preprint arXiv:1704.07852*, 2017.

[36] J. Johnson and M. Puschel, “In search of the optimal Walsh-Hadamard transform,” in *IEEE Int. Conf. on Acoust., Speech, and Signal Process.*, vol. 6, 2000, 3347–3350 vol.6.

[37] E. Jonas, Q. Pu, S. Venkataraman, I. Stoica, and B. Recht, “Occupy the cloud: Distributed computing for the 99%,” in *Proceedings of the 2017 Symposium on Cloud Computing*, ser. SoCC ’17, Santa Clara, California: Association for Computing Machinery, 2017, pp. 445–451.

[38] S. Kay, “A fast and accurate single frequency estimator,” *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, pp. 1987–1990, 1989.

[39] M. Kocaoglu, K. Shanmugam, A. G. Dimakis, and A. Klivans, “Sparse polynomial learning and graph sketching,” in *Advances in Neural Information Processing Systems*, 2014, pp. 3122–3130.

[40] S. Kudekar, T. J. Richardson, and R. L. Urbanke, “Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC,” *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 803–834, Jan. 2011.

[41] ——, “Spatially coupled ensembles universally achieve capacity under belief propagation,” *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 7761–7813, Jan. 2013.

[42] E. Kushilevitz and Y. Mansour, “Learning decision trees using the Fourier spectrum,” *SIAM J. Comput.*, vol. 22, no. 6, pp. 1331–1348, 1993.

[43] H. J. Landau, “Necessary density conditions for sampling and interpolation of certain entire functions,” *Acta Math.*, vol. 117, pp. 37–52, 1 Jul. 1967.

[44] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, “Speeding up distributed machine learning using codes,” *IEEE Int. Symp. on Inf. Theory*, vol. 2016-Augus, pp. 1143–1147, 2016. eprint: `arXiv:1512.02673v1`.

[45] K. Lee, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, “Coded computation for multicore setups,” in *IEEE Int. Symp. on Inf. Theory*, Jun. 2017, pp. 2413–2417.

[46] K. Lee, C. Suh, and K. Ramchandran, “High-dimensional coded matrix multiplication,” *IEEE Int. Symp. on Inf. Theory*, pp. 2418–2422, 2017.

[47] M. Lee and M. Kaveh, “Fast Hadamard transform based on a simple matrix factorization,” *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 34, no. 6, pp. 1666–1667, Dec. 1986.

[48] R. T. Leenay, A. Aghazadeh, J. Hiatt, D. Tse, T. L. Roth, R. Apathy, E. Shifrut, J. F. Hultquist, N. Krogan, Z. Wu, *et al.*, “Large dataset enables prediction of repair after CRISPR–Cas9 editing in primary t cells,” *Nature Biotechnology*, vol. 37, no. 9, pp. 1034–1037, 2019.

[49] X. Li, J. K. Bradley, S. Pawar, and K. Ramchandran, “The SPRIGHT algorithm for robust sparse Hadamard transforms,” in *IEEE Int. Symp. on Inf. Theory*, IEEE, 2014, pp. 1857–1861.

[50] X. Li and K. Ramchandran, “An active learning framework using sparse-graph codes for sparse polynomials and graph sketching,” in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 2170–2178.

[51] X. Li, J. K. Bradley, S. Pawar, and K. Ramchandran, “SPRIGHT: A fast and robust framework for sparse Walsh-Hadamard transform,” *arXiv.org*, Aug. 2015. arXiv: `1508.06336v1 [cs.IT]`.

[52] X. Li, S. Pawar, and K. Ramchandran, “Sub-linear time support recovery for compressed sensing using sparse-graph codes,” *arXiv*, Dec. 2014. arXiv: `1412.7646v1 [cs.IT]`.

[53] Y. M. Lu and M. N. Do, “A theory for sampling signals from a union of subspaces,” *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2334–2345, Jun. 2008.

[54] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, “Compressed sensing MRI,” *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 72–82, Mar. 2008.

[55] M. D. Macleod, “Fast nearly ML estimation of the parameters of real or complex single tones or resolved multiple tones,” *IEEE Trans. Signal Process.*, vol. 46, pp. 141–148, 1998.

[56] A. Maleki, L. Anitori, Z. Yang, and R. G. Baraniuk, “Asymptotic analysis of complex LASSO via complex approximate message passing (CAMP),” *IEEE Trans. Inf. Theory*, vol. 59, no. 7, pp. 4290–4308, Mar. 2013.

[57] M. Mishali, Y. C. Eldar, O. Dounaevsky, and E. Shoshan, “Xampling: Analog to digital at sub-Nyquist rates,” *IET circuits*, vol. 5, pp. 8–20, 1 Jan. 2011.

[58] M. Mishali and Y. C. Eldar, “From theory to practice: Sub-Nyquist sampling of sparse wideband analog signals,” *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 2, pp. 375–391, Apr. 2010.

[59] E. D. Nelson and M. L. Fredman, “Hadamard spectroscopy,” *JOSA*, vol. 60, no. 12, pp. 1664–1669, 1970.

[60] D. Nishimura, *Principles of Magnetic Resonance Imaging*. Stanford University, 2010.

[61] O. Ocal, S. Kadhe, and K. Ramchandran, "Low-degree pseudo-Boolean function recovery using codes," in *IEEE Int. Symp. on Inf. Theory*, Jul. 2019, pp. 1207–1211.

[62] O. Ocal, X. Li, and K. Ramchandran, "A sparse-graph-coded filter bank approach to minimum-rate spectrum-blind sampling," in *IEEE Int. Conf. on Acoust., Speech, and Signal Process.*, Mar. 2016, pp. 6365–6369.

[63] F. Ong, S. Pawar, and K. Ramchandran, "Fast and efficient sparse 2D discrete Fourier transform using sparse-graph codes," *arXiv preprint arXiv:1509.05849*, Sep. 2015.

[64] J. Otwinowski, D. M. McCandlish, and J. B. Plotkin, "Inferring the shape of global epistasis," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 115, no. 32, p. 201 804 015, 2018.

[65] M. van Overbeek, D. Capurso, M. M. Carter, M. S. Thompson, E. Frias, C. Russ, J. S. Reece-Hoyes, C. Nye, S. Gradia, B. Vidal, *et al.*, "DNA repair profiling reveals nonrandom outcomes at Cas9-mediated breaks," *Molecular Cell*, vol. 63, no. 4, pp. 633–646, 2016.

[66] C. H. Papadimitriou, "Optimality of the fast Fourier transform," *J. ACM*, vol. 26, no. 1, pp. 95–102, Jan. 1979.

[67] S. Pawar and K. Ramchandran, "Computing a k-sparse n-length discrete Fourier transform using at most $4k$ samples and $O(k \log k)$ complexity," in *IEEE Int. Symp. on Inf. Theory*, IEEE, Istanbul, Jul. 2013, pp. 464–468.

[68] ——, "A robust R-FFAST framework for computing a k-sparse n-length DFT in $O(k \log n)$ sample complexity using sparse-graph codes," in *IEEE Int. Symp. on Inf. Theory*, IEEE, 2014, pp. 1852–1856.

[69] ——, "R-FFAST: A robust sub-linear time algorithm for computing a sparse DFT," *IEEE Trans. Inf. Theory*, vol. 64, pp. 451–466, 2017.

[70] ——, "FFAST: An algorithm for computing an exactly k-sparse DFT in $O(k \log k)$ time," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 429–450, 2018.

[71] R. Pedarsani, K. Lee, and K. Ramchandran, "Sparse covariance estimation based on sparse-graph codes," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sep. 2015, pp. 612–619.

[72] W. K. Pratt, J. Kane, and H. C. Andrews, "Hadamard transform image coding," *Proc. IEEE*, vol. 57, no. 1, pp. 58–68, Jan. 1969.

[73] D. M. Rankin and T. A. Gulliver, "Single parity check product codes," *IEEE Transactions on Communications*, vol. 49, no. 8, pp. 1354–1362, 2001.

[74] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[75] ——, "Multi-edge type LDPC codes," in *Workshop honoring Prof. Bob McEliece on his 60th birthday, California Institute of Technology, Pasadena, California*, 2002, pp. 24–25.

[76] ——, *Modern Coding Theory*. Cambridge University Press, 2008.

[77] Z. R. Sailer and M. J. Harms, “Detecting high-order epistasis in nonlinear genotype-phenotype maps,” *Genetics*, vol. 205, no. 3, pp. 1079–1088, 2017.

[78] R. Scheibler, S. Haghighatshoar, and M. Vetterli, “A fast Hadamard transform for signals with sublinear sparsity in the transform domain,” *IEEE Trans. on Inf. Theory*, vol. 61, no. 4, pp. 2115–2132, 2015.

[79] C. E. Shannon, “A mathematical theory of communication,” *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[80] ——, “Communication in the presence of noise,” *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[81] M. W. Shen, M. Arbab, J. Y. Hsu, D. Worstell, S. J. Culbertson, O. Krabbe, C. A. Cassa, D. R. Liu, D. K. Gifford, and R. I. Sherwood, “Predictable and precise template-free CRISPR editing of pathogenic variants,” *Nature*, vol. 563, no. 7733, p. 646, 2018.

[82] D. Slepian, “On bandwidth,” *Proceedings of the IEEE*, vol. 64, no. 3, pp. 292–300, 1976.

[83] D. Slepian and H. O. Pollak, “Prolate spheroidal wave functions, Fourier analysis and uncertainty - i,” *Bell System Technical Journal*, vol. 40, no. 1, pp. 43–63, 1961.

[84] J. Tan, Y. Ma, and D. Baron, “Compressive imaging via approximate message passing with image denoising,” *IEEE Trans. Sig. Process.*, vol. 63, no. 8, pp. 2085–2092, Mar. 2015.

[85] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, “Gradient coding: Avoiding stragglers in distributed learning,” in *ICML*, 2017, pp. 3368–3376.

[86] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[87] J. A. Tropp, “Greed is good: Algorithmic results for sparse approximation,” *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[88] M. Unser, “Sampling–50 years after Shannon,” *Proc. IEEE*, vol. 88, no. 4, pp. 569–587, Apr. 2000.

[89] R. Varshamov, “Estimate of the number of signals in error correcting codes,” *Docklady Akad. Nauk, SSSR*, vol. 117, pp. 739–741, 1957.

[90] M. Vetterli and J. Kovačevic, *Wavelets and Subband Coding*. Prentice-Hall, Inc., 1995.

[91] M. Vetterli, P. Marziliano, and T. Blu, “Sampling signals with finite rate of innovation,” *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 1417–1428, Jun. 2002.

[92] M. J. Wainwright, *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*, ser. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2019.

[93] A. Wang, W. Bae, J. Han, S. Bailey, O. Ocal, P. Rigge, Z. Wang, K. Ramchandran, E. Alon, and B. Nikolić, "A real-time, 1.89-GHz bandwidth, 175-kHz resolution sparse spectral analysis RISC-V SoC in 16-nm FinFET," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 7, pp. 1993–2008, Jul. 2019.

[94] A. Wang, W. Bae, J. Han, S. Bailey, P. Rigge, O. Ocal, Z. Wang, K. Ramchandran, E. Alon, and B. Nikolić, "A real-time, analog/digital co-designed 1.89-GHz bandwidth, 175-kHz resolution sparse spectral analysis RISC-V SoC in 16-nm FinFET," in *ESSCIRC 2018 - IEEE 44th European Solid State Circuits Conference (ESSCIRC)*, Sep. 2018, pp. 322–325.

[95] T. Wimalajeewa, Y. C. Eldar, and P. K. Varshney, "Recovery of sparse matrices via matrix sketching," *arXiv preprint arXiv:1311.2448*, 2013.

[96] L. Xiao, Y. Dong, S. Pawar, R. Pedarsani, and K. Ramchandran, "Sub-linear time support recovery for compressed sensing using sparse-graph codes," *IEEE Trans. Inf. Theory*, vol. 65, pp. 6580–6619, 2019.

[97] Y. Yang, P. Grover, and S. Kar, "Coding method for parallel iterative linear solver," *CoRR*, vol. abs/1706.00163, 2017. arXiv: `1706.00163`.

[98] D. Yin, R. Pedarsani, X. Li, and K. Ramchandran, "Compressed sensing using sparse-graph codes for the continuous-alphabet setting," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, 2016, pp. 758–765.

[99] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded Fourier transform," *arXiv preprint arXiv:1710.06471*, 2017.

[100] ——, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," in *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., 2017, pp. 4403–4413.