Sparse Graph Codes for Compression, Sensing, and Secrecy

by

Venkat Bala Chandar

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering and Computer Science
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2010

Author .....
Department of Electrical Engineering and Computer Science
May 21, 2010

Certified by .....
Gregory W. Wornell
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by .....
Devavrat Shah
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Terry P. Orlando
Chairman, Department Committee on Graduate Students

Sparse Graph Codes for Compression, Sensing, and Secrecy

by

Venkat Bala Chandar

Submitted to the
Department of Electrical Engineering and Computer Science

May 21, 2010

In Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

Sparse graph codes were first introduced by Gallager over 40 years ago. Over the last two decades, such codes have been the subject of intense research, and capacity-approaching sparse graph codes with low complexity encoding and decoding algorithms have been designed for many channels. Motivated by the success of sparse graph codes for channel coding, we explore the use of sparse graph codes for four other problems related to compression, sensing, and security.

First, we construct locally encodable and decodable source codes for a simple class of sources. Local encodability refers to the property that when the original source data changes slightly, the compression produced by the source code can be updated easily. Local decodability refers to the property that a single source symbol can be recovered without having to decode the entire source block.

Second, we analyze a simple message-passing algorithm for compressed sensing recovery, and show that our algorithm provides a nontrivial ℓ_1/ℓ_1 guarantee. We also show that very sparse matrices and matrices whose entries must be either 0 or 1 have poor performance with respect to the restricted isometry property for the ℓ_2 norm.

Third, we analyze the performance of a special class of sparse graph codes, LDPC codes, for the problem of quantizing a uniformly random bit string under Hamming distortion. We show that LDPC codes can come arbitrarily close to the rate-distortion bound using an optimal quantizer. This is a special case of a general result showing a duality between lossy source coding and channel coding—if we ignore computational complexity, then good channel codes are automatically good lossy source codes. We also prove a lower bound on the average degree of vertices in an LDPC code as a function of the gap to the rate-distortion bound.

Finally, we construct efficient, capacity-achieving codes for the wiretap channel, a model of communication that allows one to provide information-theoretic, rather than computational, security guarantees. Our main results include the introduction of a new security critertion which is an information-theoretic analog of semantic security,

the construction of capacity-achieving codes possessing strong security with nearly linear time encoding and decoding algorithms for any degraded wiretap channel, and the construction of capacity-achieving codes possessing semantic security with linear time encoding and decoding algorithms for erasure wiretap channels.

Our analysis relies on a relatively small set of tools. One tool is density evolution, a powerful method for analyzing the behavior of message-passing algorithms on long, random sparse graph codes. Another concept we use extensively is the notion of an expander graph. Expander graphs have powerful properties that allow us to prove adversarial, rather than probabilistic, guarantees for message-passing algorithms. Expander graphs are also useful in the context of the wiretap channel because they provide a method for constructing randomness extractors. Finally, we use several well-known isoperimetric inequalities (Harper's inequality, Azuma's inequality, and the Gaussian Isoperimetric inequality) in our analysis of the duality between lossy source coding and channel coding.

Thesis Supervisor: Gregory W. Wornell
Title: Professor of Electrical Engineering and Computer Science

Thesis Supervisor: Devavrat Shah
Title: Associate Professor of Electrical Engineering and Computer Science

Acknowledgments

First, I would like to thank my advisors, Greg and Devavrat. Greg has been a wonderful mentor since my days as an M. Eng. student, and I have learned so much from both Greg and Devavrat over the years. It goes without saying that they have taught me many valuable research skills, but they also taught me that if a conference deadline is at 7:00, don't try to upload your paper at 6:59! On a more serious note, they showed me the importance of taking the time to present the results of my research clearly. Indeed, this thesis would probably be incomprehensible but for their efforts to help me improve the clarity of the presentation. As I neared graduation, their career advice and help with the job search have been very useful, and I am thankful that I got to work with two advisors who, in addition to being great research mentors, take so much interest in their students' professional development.

I would also like to thank Piotr, not only for serving on my thesis committee, but for teaching me about compressed sensing in his streaming algorithms class. Chapter 4 started out as my class project for his class, and he originally suggested that I extend my results for binary matrices to general sparse matrices.

Thanks to all of the friends I have made over the last eight years. Since my days here as an undergrad, I have looked forward to grabbing lunch with Ravi and Ajay, and later Lohith and Srikanth, too. I will really miss our weekly trips to India Quality. I would like to thank Da, James, Qing, Yuval, Ying-zong, Maryam, Lane, Vijay, Anthony, Ashish, Charles, Urs, and Aslan for making the Signals, Information, and Algorithms lab such a fun and stimulating place to work. I have shared an office with Charles, Urs, and Da during my time in the lab, and all three were excellent office mates. I enjoyed all of our conversations, whether research related or not. I miss the weekly squash games with Anthony and Vijay, and I'm expecting some tougher competition the next time we play! Even though our work on asynchronous communication didn't make it into my thesis, one of my favorite experiences in grad school was working with Aslan. I would never have guessed that a short conversation one afternoon would turn into a years-long, and eventually trans-atlantic, collaboration,

and I look forward to working with him much more in the future. Also, thanks to Tricia for all of her help over the years, especially for being able to accomodate all of my last-minute requests, be it for conference registration or booking a room for my defense.

Finally, I would like to thank Mom, Dad, and Vijay. I could never have finished this thesis without all of your love and encouragement.

Contents

List of Figures

Chapter 1

Introduction

This thesis explores the application of sparse graph codes to several problems broadly related to compression, sensing, and security. In Section 1.1, we briefly review the basic mathematical model for communication over a noisy channel, as proposed by Shannon in his classic work [107]. Next, in Section 1.2 we discuss sparse graph codes informally, postponing a formal description of sparse graph codes to Chapter 2. Finally, in Section 1.3 we give a high-level overview of the problems considered in this thesis and our main results.

1.1 Channel Coding Review

In [107], Shannon proposed a mathematical model for communication over a noisy channel. Figure 1-1 shows the basic communication model. The message is a string

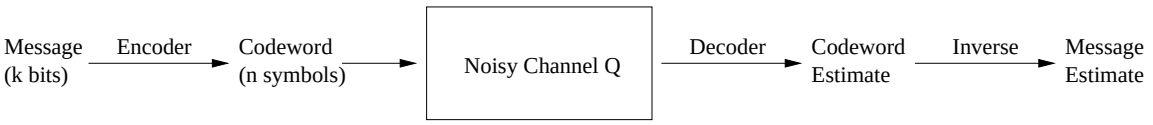

Figure 1-1: Mathematical model for communication. First, a message of k bits (i.e., a string of k 0's and 1's) is encoded into a string of n symbols from the channel input alphabet $\mathcal{X}$. The resulting codeword is sent over the channel Q, producing a string of n symbols from the channel output alphabet $\mathcal{Y}$. The decoder first estimates the codeword, and then applies the inverse of the encoding map to obtain an estimate of the original message of k bits.

of bits, and the first step in communication is to encode the message by mapping it into a codeword composed of symbols from $\mathcal{X}$, the channel input alphabet. Next, the codeword is transmitted over the channel, and the receiver receives a noisy version of the codeword. Formally, the channel is assumed to be a discrete memoryless channel (DMC). Recall that a memoryless channel can be specified by its input alphabet $\mathcal{X}$, output alphabet $\mathcal{Y}$, and transition matrix $Q(y|x)$. Specifically, if a string x^n is fed into the channel, then the channel output is a random variable Y^n distributed according to

$$Q(Y^n = y^n|x^n) = \prod_{i=1}^{n} Q(y_i|x_i).$$

One of the basic results of information theory, the Channel Coding Theorem [28, 30, 107], shows that each memoryless channel has a capacity, that is, a maximum number of bits per channel use that can be transmitted reliably in the limit of long channel inputs.

The receiver must recover the message from Y^n. Conceptually, we split this procedure into two steps. First, the receiver estimates which codeword $c \in \mathcal{X}^n$ was sent over the channel. Then, the receiver inverts the encoding mapping to determine the original message corresponding to codeword c. Most of coding theory focuses on the design of good codes, i.e., designing large sets of codewords that can be reliably distinguished when sent over the channel. The encoding mapping and inverting the encoding mapping are typically an afterthought. One possible justification for this is that from a theoretical standpoint, most codes proposed to date are linear codes, and the encoding and inverse maps can be implemented in polynomial (in the code length) complexity for any code. Designing a code and a decoding algorithm, on the other hand, is much harder. For example, the maximum likelihood (ML) decoding problem for linear codes is NP-hard in general [12], so a lot of research has focused on the design of linear codes possessing additional structure that allows ML or near-ML decoding in linear time at rates close to capacity. As we will see in Section 1.2, sparse graph codes have gone a long way towards solving the code design problem. Of course, if we can construct codes that are decodable in linear time, then the complexity of

the encoding and inverse maps, which is always polynomial, can suddenly become overwhelming compared to the decoding complexity. Fortunately, certain classes of sparse graph codes support not only linear time decoding, but also possess linear time encoding and inverse algorithms.

1.2 Informal Overview of Sparse Graph Codes

Sparse graph codes were first proposed in [45], and have emerged as an attractive solution to the channel coding problem. In this section we informally review some of the major results in the area. A more formal description of the sparse graph codes used in this thesis is given in Chapter 2.

As the name implies, sparse graph codes are error-correcting codes defined in terms of a sparse graph, i.e., a graph with very few edges. Typically, by very few edges we mean that as the graph size grows, i.e., as the codeword length increases, the number of edges grows linearly with the number of vertices in the graph. Sparse graph codes form a subclass of a general class of probabilistic models known as graphical models, which are often represented by structures known as factor graphs. Graphical models have found use not just in the design of error-correcting codes, but also in machine learning, statistical inference, and many other fields. Several good introductions to factor graphs can be found in the papers [1, 69]. However, since we focus on sparse graph codes in this thesis, we will not elaborate on the general theory of factor graphs. Instead, we review results that apply specifically to sparse graph codes in the context of channel coding.

There are many different types of sparse graph codes, corresponding to different ways of interpreting a sparse graph as defining a code. For example, low density generator matrix (LDGM) codes are one class of sparse graph codes. It is well-known that LDGM codes are terrible channel codes (see, for example, Section 2.2.1 for a simple explanation of this fact) in the sense that the block error rate, i.e., the probability that an optimal decoder is able to correctly estimate the codeword sent by the transmitter, does not approach 0 in the limit of long code lengths—in fact,

the probability of error approaches 1. Nevertheless, LDGM codes have some nice properties. For example, the encoding map can be computed in linear time. Also, although the block error rate, even under optimal decoding, approaches 1, LDGM codes can be used to reduce the noise level of a channel. For example, instead of considering decoders that produce an estimate of the transmitted codeword, we can consider decoders that produce an estimate for each bit sent by the transmitter. The corresponding notion of error, known as the bit error rate, is the average fraction of bits that such a decoder estimates incorrectly. It turns out that although LDGM codes have a high error rate in the sense that most of the time the sent message cannot be recovered correctly even by an optimal decoder, carefully designed LDGM codes can achieve low bit error rates. For example, although LDGM codes do not allow the reliable transmission of long blocks of messages, a good LDGM code might be designed so that if a message of 1000 bits is encoded by the LDGM code, then decoded by a practical, but potentially suboptimal, decoder, on average at least 990 of the bits are decoded correctly, i.e., the bit error rate is only .01. Intuitively, it should come as no surprise that the ability of LDGM codes to achieve low bit error rates is quite useful, and several papers have used LDGM codes in clever ways to construct codes for the binary erasure channel (BEC) that can be decoded in linear or nearly linear time, and achieve not only low bit error rates, but also low block error rates [75,76,110,113]. [1] Because of the special structure of LDGM codes, these codes typically allow the encoding and inverse maps to be computed in (nearly) linear time as well.

Although with some effort LDGM codes can be used to design codes achieving low block error rates, in practice it is difficult to achieve very low block error rates when operating near channel capacity by just relying on clever combinations of LDGM codes. It turns out that another class of codes, low density parity check (LDPC) codes, can provide much better performance.

LDPC codes were first proposed by Gallager [45]. In [45], Gallager showed that a special class of LDPC codes called regular LDPC codes can achieve the capacity

[1] See Section 2.1 for a formal definition of the BEC.

of the binary symmetric channel (BSC). [2] Specifically, he showed that if an optimal, i.e., maximum likelihood (ML) , decoder is used, then the performance of a regular LDPC code drawn from a suitable random ensemble is likely to be quite good. In fact, [45] provides a detailed analysis computing the rate at which the block error rate decays as a function of the code length n. In contrast to LDGM codes, the block error rate for LDPC codes under ML decoding approaches 0 exponentially quickly in n, and [45] characterizes the error exponent associated with the block error rate. More recently, [72] analyzes the distribution of codewords for several different ensembles of LDPC codes, including the ensembles proposed by Gallager.

There has also been work attempting to characterize how the number of edges in the graph representing an LDPC code (often called the Tanner graph of the code) scales as we try to approach capacity. For example, a famous result of Gallager [45] says that for any LDPC code, the ratio of the number of edges in the Tanner graph to the number of vertices in the graph must grow in order to approach rates closer to the capacity. However, the growth rate is not very fast, so that at least in principle LDPC codes with relatively low average degree can approach rates very close to the capacity.

Although LDPC codes achieve good performance under ML decoding, it can be shown that in the worst-case, ML decoding of an LDPC code is NP-hard, so ML decoding of LDPC codes is likely to be quite complex. As an attempt to address the decoding problem, Gallager [45] proposed a low complexity message-passing algorithm for decoding LDPC codes. Although analyzing the algorithm proved to be challenging, Gallager [45] was able to rigorously show that his message-passing algorithm could correct a (small) constant fraction of errors, and simulation results suggested that Gallager's message-passing algorithm could correct a sizable fraction of errors.

Despite the results in [45], LDPC codes were all but forgotten for roughly thirty years, until Berrou, Glavieux, and Thitimajshima discovered turbo codes [13]. After the discovery of turbo codes, several groups of researchers rediscovered LDPC codes

[2] See Section 2.1 for a formal definition of the BSC.

[76, 77]. Subsequent work on LDPC codes showed that although regular LDPC codes can correct a sizable fraction of errors under message-passing decoding, the more general class of irregular LDPC codes contains codes that can correct far more errors under message-passing decoding. In fact, it has been verified by simulation that for many channels, carefully optimized irregular LDPC codes can achieve low block error rates under message-passing decoding at rates very close to capacity [23, 99, 100]. In the special case of the BEC, it is possible to rigorously prove that LDPC codes achieve the capacity [76, 91] under message-passing decoding.

We now give a high level description of message-passing algorithms, and a rough sketch of how one might go about analyzing the performance of message-passing decoding. One way to understand message-passing decoding is to consider ML decoding of an LDPC code whose Tanner graph is cycle-free. Although ML decoding is NP-hard in general, when the Tanner graph is cycle-free, a simple dynamic programming algorithm can be used for ML decoding. One can interpret this dynamic programming algorithm as a message-passing algorithm, i.e., an algorithm where vertices in the Tanner graph pass messages to their neighbors. These messages are updated iteratively using local update rules (by local, we mean that the updated message leaving a vertex depends only on the messages coming into that vertex at the previous iteration). The reader is referred to [118] for a more detailed exposition of message-passing algorithms and their application in the general context of graphical models.

For cycle-free graphs, by choosing the update rules properly, one can construct message-passing algorithms that perform ML decoding. Unfortunately, it is known that LDPC codes whose Tanner graph is cycle-free are very bad [39], so good codes must have cycles. However, because the update rules are local, a message-passing algorithm can in principle be applied to an arbitrary graph, i.e., even a graph with cycles. If we choose to run a message-passing algorithm on a graph with cycles, there is no guarantee that the algorithm performs ML decoding—in fact, we cannot even guarantee that the messages converge! Even though the behavior of message-passing algorithms is not well-understood for arbitrary graphs, simulations suggest that when the Tanner graph has cycles, message-passing decoding can be very effective

for carefully designed codes. The key insight needed to explain this phenomenon, which already appears in [45], but which was subsequently refined in [76,100], is that the asymptotic behavior of message-passing algorithms can be analyzed when the Tanner graph has no short cycles, i.e., the Tanner graph has large girth. By definition, message-passing algorithms only depend on local data, so if the Tanner graph has no short cycles, then for the first few iterations, the decoding algorithm does not "know" that it is operating on a graph with cycles. Therefore, an analysis based on trees is sufficient to capture the behavior. Since message-passing is much easier to analyze on trees, this provides a powerful approach to analyzing the performance of message-passing decoding of LDPC code ensembles that have large girth. This approach is discussed in detail in [100], and is called density evolution. Density evolution plays a key role in many of the code constructions in this thesis. For example, the analysis in Appendix A includes a fairly detailed example of the density evolution technique.

Density evolution has been used to design LDPC codes that approach the capacity for several channels [23,99] when message-passing decoding, rather than ML decoding, is used. However, the BEC is the only channel for which it has been proved that LDPC codes with message-passing decoding can achieve capacity. There are many proofs of this result [76, 91], and density evolution is the key ingredient in all of these proofs.

We now discuss the encoding problem for LDPC codes. As noted earlier, a lot of research in coding theory has focused on constructing codes and practical decoding algorithms, and often the complexity of the encoding and inverse maps is neglected. In contrast to LDGM codes, there is no known linear time algorithm for encoding a general LDPC code (LDPC codes are linear codes, so these codes must be encodable in polynomial time). In general, the naive encoding algorithm that applies to any linear code takes $O(n^2)$ time, so this naive algorithm is significantly more complex than the linear time message-passing algorithm for decoding an LDPC code. One approach to designing LDPC codes with low encoding complexity [101] uses message-passing decoding in a clever way to encode LDPC codes. For certain classes of LDPC codes, the encoding method proposed in [101] can reduce the encoding complexity to $O(n)$, and even when it does not, the encoding complexity is usually much smaller than the

complexity of the naive encoding algorithm for arbitrary linear codes. However, from a theoretical standpoint, it is difficult to design LDPC codes that can provably be encoded in linear time using the algorithm from [101] and simultaneously approach capacity under message-passing decoding.

It turns out that one can construct sparse graph codes that share many properties with LDPC codes, but which also have additional structure that makes encoding trivial. Nonsystematic irregular repeat-accumulate (NSIRA) codes [97] are one such class of codes. NSIRA codes are a variation of LDPC codes designed to have very efficient encoding and decoding algorithms for the BEC. NSIRA codes build upon a line of work that started with [32], which proposed the class of repeat-accumulate codes. One motivation for repeat-accumulate codes is that this class of codes has a graphical structure which makes the encoding and inverse maps trivial to compute in linear time. At the same time, the graphical structure looks similar to that of LDPC codes, so potentially density evolution and related tools developed for LDPC codes can be used to find repeat-accumulate codes with practical decoding algorithms. In particular, generalizing repeat-accumulate codes to irregular repeat-accumulate codes in much the same way that regular LDPC codes were generalized to irregular LDPC codes, it was shown in [64] that one can construct irregular repeat-accumulate codes that achieve the capacity of the binary erasure channel under message-passing decoding. NSIRA codes take this work one step further, producing codes for which the encoding and inverse maps are trivial, but which also have the property that they achieve the capacity of the BEC under message-passing decoding with uniformly bounded complexity regardless of the gap to capacity—in particular, the average degree of the vertices in the graphical representation of suitable NSIRA codes remains constant as the code length gets large. [3] Contrast this with LDPC codes, where we mentioned previously that even under ML decoding, the average degree of the vertices in the Tanner graph, and hence the complexity of message-passing decoding, must become unbounded as the rate approach the capacity. (Gallager [45] only proved

[3]By uniformly bounded complexity, we mean that the number of operations per bit required for the encoding, decoding, and inverse operations is bounded by some fixed constant independent of the gap to capacity.

a lower bound for the BSC, but a similar result holds for the BEC as well [106].) Thus, in addition to having nice encoding and decoding properties, NSIRA codes are defined by a graph that can in a fundamental sense be significantly sparser than the Tanner graph of a good LDPC code. In Chapter 5 we will see another example of this phenomenon, i.e., a case where codes defined by one class of graph structures may be fundamentally limited so that the average degree must become unbounded to achieve the appropriate information-theoretic limit, while other graph structures can be used to define codes achieving the same limit with a uniformly bounded average degree.

We close our brief overview of sparse graph codes by quickly reviewing the use of expander graphs in coding theory. The message-passing algorithms described above take advantage of the probabilistic nature of the channel, and density evolution can be used to produce bounds on the bit error rate under message-passing decoding. As mentioned above, density evolution is myopic in the sense that it only relies on the local graph structure. Roughly speaking, density evolution can be used to analyze the behavior of a message-passing algorithm on a graph with n vertices by examining a neighborhood of $o(n)$ vertices around any given vertex—density evolution typically breaks down once the neighborhood contains more than about $\sqrt{n}$ vertices. Therefore, a different analysis tool is required if, for example, we want to prove that the block error rate under message-passing decoding decays exponentially with n, because exponential decay cannot be proved without looking at the global structure of the graph.

Although the performance of message-passing algorithms on arbitrary graphs is difficult to analyze, it turns out that expander graphs are a class of graphs possessing properties that make a global analysis of message-passing algorithms more tractable. For example, in [113] it is proved that LDPC codes defined by Tanner graphs with good expansion properties can provide adversarial error-correction guarantees. In other words, instead of assuming that errors are introduced by a memoryless channel, one can show even if the errors are carefully chosen by an adversary, as long as the number of errors is sufficiently small, no matter how the adversary chooses the errors, a message-passing algorithm can be used to recover the original codeword.

Although the number of (adversarial) errors that such codes can correct is quite low, for theoretical purposes these codes are useful as they allow us to provide bounds on the block error rate. Intuitively, the idea is that we use density evolution to analyze the bit error rate, and if the bit error rate is small, then combining our code with a code constructed using expander graphs provides a deterministic guarantee that the few remaining bit errors are corrected. Thus, we can use a code derived from an expander graph to turn bounds on the bit error rate into bounds on the block error rate. For example, we use this approach in Chapter 6 to construct codes for the wiretap channel. Another example where we use expander graphs is in Chapter 4. Chapter 4 focuses on compressed sensing, which roughly speaking corresponds to constructing very high rate codes. It turns out that at high rates, the number of adversarial errors corrected by codes defined using expander graphs is within a constant factor of the channel capacity. Thus, in the high-rate case, codes defined using expander graphs can be quite useful by themselves, and in Chapter 4 we analyze a message-passing algorithm run only on an expander graph with no additional modifications.

The above overview has barely scratched the surface of known results on sparse graph codes. We refer the reader to the many excellent surveys on LDPC codes, for example [102, 111], and the recent book [103] for a more thorough introduction to the major topics in LDPC codes.

1.3 Thesis Outline

In Sections 1.1 and 1.2 we summarized the emergence of sparse graph codes as a practical alternative to random codes for the channel coding problem. The reader familiar with information theory knows that the random coding technique appears in many other contexts, to name a few, source coding, lossy source coding, broadcast channels, and multiple-access channels. Given that sparse graph codes appear to have low complexity and yet still approach the capacity for a broad class of channels, it is natural to wonder if sparse graph codes can be adapted to replace random codes more generally. In this thesis, we present four examples suggesting that sparse graph

codes do have potential as a low complexity alternative to random codes in contexts beyond their original domain of channel coding.

First, in Chapter 3 we consider using sparse graphs for (lossless) compression. Compression, also known as source coding, is a well-studied problem. Traditionally, the most important performance metrics for source coding are the compression rate and the computational complexity of encoding and decoding the source code. In Chapter 3 we are interested in constructing source codes that possess two additional properties—local encodability and local decodability. To define these properties, consider compression of a block, or equivalently, a vector of data. Local encodability is the property that when a single component of the vector is changed, it is easy to update the compressed output. Local decodability is the property that any component of the vector can be recovered efficiently from the compressed output. For example, it should be possible to recover a single component much more efficiently than by running the decompression algorithm to recover the whole vector. It is easy to think of scenarios where source codes possessing these properties could be useful. For example, imagine that you are writing your thesis, and you want to store the thesis in a compressed format. As your advisor gives you feedback, you might make some edits to the thesis—for example, maybe you revise one page. Since you will make many small edits, you don't want to decompress and recompress your thesis every time you make a change. If you use a locally encodable source code, then your compressed thesis can easily be updated as you edit, eliminating the need for decompression and recompression! As another scenario, imagine that one wants to store a large database in a compressed format. In order for the database to be useful, a user needs to be able to quickly access a single record in the database—thus, a source code used to compress the database must be locally decodable.

For a simple class of sources, we propose a solution based on sparse graph codes that possesses nontrivial local encoding and local decoding properties. In more detail, we construct source codes for sources specified by average and maximum constraints, so the source consists of all vectors of nonnegative integers whose average is at most A, and whose maximum is at most M, where A and M are parameters that can be

set arbitrarily. Our source codes achieve rates close to the information-theoretic limit for these sources in the limit of large A. Our codes are also locally encodable and decodable—the expected running time of our local encoding and decoding algorithms is at most polylogarithmic in N, and for many choices of A, M, and N, the expected running time is constant. We provide some motivation for this source model in Chapter 3, but the construction of locally encodable and decodable source codes for more general classes of sources is an interesting direction for future work.

In Chapter 4, we apply sparse graph codes to the problem of compressed sensing. We define the compressed sensing problem formally in Chapter 4, but roughly speaking, compressed sensing is the problem of recovering an N-dimensional signal x from M linear measurements y. When $M < N$, there is not enough information to determine x from y, i.e., the linear system is underdetermined. The key idea in compressed sensing is that if the signal x is very sparse, i.e., most of the components of x are 0, then even in the case that $M < N$, it is possible to reconstruct x exactly from y. For example, if 99% of the components of x are 0, a typical result in compressed sensing might allow one to conclude that x can be recovered exactly from roughly $.05N$ linear measurements via linear programming (LP), provided that these linear measurements satisfy certain technical conditions.

Of course, many signals encountered in the real-world are not perfectly sparse, but rather are approximately sparse in the sense that a few signal coefficients carry most of the information. There are several methods for measuring approximate sparsity and for providing recovery guarantees for approximately sparse signals. One widely used notion is an ℓ_p/ℓ_q guarantee. Mathematically, an ℓ_p/ℓ_q guarantee is a recovery guarantee of the form

$$||x - \hat{x}||_p \leq C||x - x^k||_q \ ,$$

where x is the original signal, x^k denotes the best k-term approximation of x, i.e., the set of k largest magnitude components of x, and $\hat{x}$ denotes the estimate of x produced by a recovery algorithm. C denotes the factor by which the error is multiplied; obviously, all things being equal, smaller values of C are preferable to larger values.

In Chapter 4, we consider ℓ_1/ℓ_1 and ℓ_2/ℓ_1 guarantees.

In the first half of Chapter 4, we show that linear measurements derived from sparse graph codes can perform well for compressed sensing, and further, that message-passing algorithms can be used as an efficient alternative to LP for recovering the original signal x from the measurements y. More precisely, we show that if the linear measurements correspond to a suitably good expander graph (see Section 2.3 for a definition of expander graphs), then a simple message-passing algorithm can be used to recover sparse signals, and the same algorithm also provides an ℓ_1/ℓ_1 guarantee.

In the second half of Chapter 4, we probe the limitations of sparse graph codes with respect to ℓ_2/ℓ_1 guarantees. Specifically, we show that linear measurements based on binary matrices do not possess good restricted isometry constants with respect to the ℓ_2 norm (see Chapter 4 for a definition of restricted isometry constants). We also show that linear measurements based on very sparse matrices do not have good isometry constants, even if the entries do not have to be binary. To date, the restricted isometry property is the only method known for providing ℓ_2/ℓ_1 guarantees, so our results suggest that either a new proof technique or a different class of linear measurements is needed to develop fast (i.e., linear or near linear time) reconstruction algorithms providing ℓ_2/ℓ_1 recovery guarantees.

In Chapter 5, we consider using sparse graphs codes for lossy source coding, also known as rate-distortion coding. Roughly speaking, lossy source coding is similar to traditional, lossless source coding, i.e., the goal is to represent a source using as few bits as possible. However, unlike the lossless case, in lossy source coding we do not have to reconstruct the source exactly from the compressed version. Rather, we just need to reconstruct the source approximately, where the notion of approximate recovery is formalized through a distortion measure quantifying the difference between the original source and our reconstruction.

While we are unable to provide efficient algorithms for lossy source coding using sparse graph codes, we are able to prove nontrivial guarantees on the performance of sparse graph codes when optimal (and computationally very expensive) algorithms are used. Specifically, we show that if one ignores computational complexity, there

is a strong duality between the lossy source coding problem and the channel coding problem. Our results imply that for a broad class of rate-distortion problems, a good channel code (i.e., a code achieving low probability of error under ML decoding) for an appropriate dual channel automatically achieves low distortion for the original rate-distortion problem. This duality result relies on an interesting connection between isoperimetric inequalities and the rate-distortion problem. Note that this result applies to any code, not just sparse graph codes. Intuitively, one can think of channel coding as a sphere-packing problem, while lossy source coding is a covering problem. Our duality result essentially says that in high dimensions, i.e., for long code lengths, these problems are equivalent in the sense that good sphere-packings must also be good covers.

As a simple corollary of this general result, we show that LDPC codes are optimal for the problem of quantizing a binary symmetric source under Hamming distortion (BSS-HD), i.e., quantizing a uniformly random bit string so that the reconstruction is within Hamming distance D of the original string.[4] Intuitively, this result is obtained by observing that the dual channel for the BSS-HD problem is the BSC, and as we noted in Section 1.2, LDPC codes achieve capacity for the BSC. Therefore, they must also be optimal for the BSS-HD problem.

For the BSS-HD problem, we also prove lower bounds on the sparsity of LDGM codes and LDPC codes. This is the analog for lossy source coding of the fact that the LDGM and LDPC graph structures suffer inherent limitations compared to other classes of codes. We already mentioned this in Section 1.2 in the context of channel coding, and our lower bounds show that the average degree of the vertices in the graphs representing LDGM codes and LDPC codes must become unbounded to approach optimal performance for the BSS-HD problem. In a little more detail, we show that for LDGM codes and LDPC codes, the average degree of the vertices in the associated graphs must grow as $\Omega(\log(\frac{1}{\varepsilon}))$, where ε is the gap between the code rate and the optimal code rate for a given distortion level. This dependence on ε is

[4]By optimal, we mean that LDPC codes can achieve performance arbitrarily close to the rate-distortion function. See Chapter 5 for a formal definition of the rate-distortion function and the BSS-HD problem.

tight to within constant factors, and just as the NSIRA codes have graphical representations that have uniformly bounded average degree no matter how close we want to approach the capacity of the BEC, it turns out that a concatenation of LDGM codes and LDPC codes can have uniformly bounded average degree for the BSS-HD problem.

In Chapter 6, we apply sparse graph codes to the wiretap channel [120]. The wiretap channel is a probabilistic model for one of the most fundamental problems in cryptography, namely, how two parties can communicate information secretly in the presence of an eavesdropper. The wiretap channel is interesting because it provides a way around the classic result of Shannon [108] stating that every information-theoretically secure communication scheme essentially has requirements equivalent to a one-time pad, i.e., communicating k bits secretly requires two parties to already share a secret key of length k. As we explain in more detail in Chapter 6, the wiretap channel model allows one to provide security guarantees almost as strong as the guarantees required under Shannon's definition of security [108], but the inherent limitations the model places on the eavesdropper allow us to construct secure coding schemes that are the analogs of public-key encryption, i.e., secure codes for the wiretap channel do not suffer the drawback of the one-time pad in that the two parties do not need to share a secret key.

Our main results for the wiretap channel include the following. First, we define the information-theoretic analog of semantic security [49], a widely used notion of security in computational cryptography, and show that our analog of semantic security is closely related to, but slightly stronger, than the notion of strong security considered in the wiretap channel literature. Next, we propose a general architecture for constructing codes for wiretap channels. Our architecture separates the code design process into the design of a precode to provide security, and a standard channel code to correct errors introduced by the noisy channel between the two parties trying to communicate. This is attractive because the precode can be designed independently of the channel code, so we have effectively decoupled security and error-correction in the code design process. We give a few example precodes based on randomness

extractors (see Section 2.3 for a definition of randomness extractors) to illustrate the code design process, resulting in a proof that for any degraded wiretap channel, there exist precodes possessing strong security whose encoding and decoding complexities are $O(n \log n \log \log n)$. Finally, for the special case of wiretap channels where the component channels are BECs, we design sparse graph codes possessing semantic security whose encoding and decoding complexities are only $O(n)$.

Chapter 7 summarizes the results in this thesis and suggests avenues for future research.

Chapter 2

Preliminaries

In this chapter, we summarize some basic results used later in the thesis. In Section 2.1, we quickly review some basic notions from coding theory and information theory, such as linear codes and the method of types. Next, in Section 2.2, we define sparse graph codes and describe in detail four classes of sparse graph codes that are used in this thesis. Finally, Section 2.3 defines expander graphs and randomness extractors, and describes several constructions of these objects.

2.1 Brief Review of Information Theory and Coding Theory

We start by reviewing a little bit of information theory. First, we recall the definitions of three standard channel models—the binary erasure channel (BEC), the binary symmetric channel (BSC), and the additive white Gaussian noise (AWGN) channel. These three channel models are examples of memoryless channels, as defined in Chapter 1.

- BEC(e): The binary erasure channel with erasure probability e has input alphabet $\{0, 1\}$ and output alphabet $\{0, 1, *\}$. The transition matrix is

$$Q(y|x) = \begin{cases} 1 - e & \text{for} \quad y = x \\ e & \text{for} \quad y = * \end{cases}.$$

The capacity of the BEC is $1 - e$.

- BSC(p): The input and output alphabet for the binary symmetric channel with flip probability p is $\{0, 1\}$. The transition matrix is given by

$$Q(y|x) = \begin{cases} 1-p & \text{for} \quad y = x \\ p & \text{for} \quad y = 1-x \end{cases}.$$

  The capacity of the BSC is $1 - h_b(p)$, where $h_b(p)$ denotes the binary entropy function, i.e., $h_b(p) \triangleq -p\log(p) - (1-p)\log(1-p)$.

- AWGN: The additive white Gaussian noise (AWGN) channel with power constraint P and noise variance σ^2 has input alphabet and output alphabet $\mathbb{R}$. The channel is defined by $Y = X + N$, where X denotes the channel input and N represents ambient noise distributed so that $N \sim \mathcal{N}(0, \sigma^2)$. That is, the channel output Y is the sum of the input and noise, hence the name additive white Gaussian noise channel. The input power constraint is that the average power of the symbols in a codeword must be at most P, i.e., for all codewords x^n in the codebook, $\frac{1}{n}\sum x_i^2 \leq P$. The capacity of the AWGN channel is $\frac{1}{2}\log(1 + \frac{P}{\sigma^2})$.

One of the main goals of coding theory is to design practical codes achieving the capacity of the above channels, and sparse graph codes have gone a long way towards achieving this goal.

The method of types is a commonly used and powerful analysis tool for discrete memoryless channels. We introduce some basic notation for dealing with types, and summarize a few well-known results on types that we use in Chapters 5 and 6. The type of a sequence $x^n \in \mathcal{X}^n$ is the probability distribution $\hat{P}$ over $\mathcal{X}$ such that

$$\hat{P}(x) = \frac{\text{number of occurrences of } x \text{ in } x^n}{n}, \forall x.$$

Similarly, given two strings x^n and y^n, we define the conditional distribution induced by these sequences as

$$\hat{Q}_{Y|X}(y|x) = \frac{\text{number of occurrences of } (y,x) \text{ in } (y^n, x^n)}{\text{number of occurrences of } y \text{ in } y^n}, \forall x.$$

(Note that if a symbol $y \in \mathcal{Y}$ does not occur at least once in y^n, then the above expression is undefined.) We define $\hat{Q}_{X|Y}$ in a similar fashion. We use several standard results regarding types from the theory of large deviations. These results can be found, for example, in [28].

1. The total number of types associated with strings $x^n \in \mathcal{X}^n$ is at most $(n+1)^{|\mathcal{X}|}$.

2. Sanov's theorem—when X^n is generated i.i.d. according to P_X, the probability that X^n has type $\hat{P}$ is at least $\frac{1}{(n+1)^{|\mathcal{X}|}} e^{-nD(\hat{P}||P_X)}$, where $D(\cdot||\cdot)$ denotes the Kullback-Liebler divergence.

3. Given y^n with type $\hat{P}_Y$ and a conditional distribution $Q_{X|Y}$, the number of strings x^n such that the conditional distribution induced by x^n and y^n equals $Q_{X|Y}$ is at least
$$\frac{e^{nH(X|Y)}}{(n+1)^{|\mathcal{X}||\mathcal{Y}|}},$$
where X and Y are random variables distributed according to $(X,Y) \sim Q_{Y|X}\hat{P}_Y$.

4. When a string y^n with type $\hat{P}_Y$ is the input to a channel with distribution $Q_{X|Y}$, the probability that the conditional distribution $\hat{Q}_{X|Y}$ induced by the channel output and channel input satisfies the property $||\hat{Q}_{X|Y}\hat{P}_Y - Q_{X|Y}\hat{P}_Y||_1 \leq \frac{1}{\log(n)}$ is at least $1 - o(1)$, where the $o(1)$ is with respect to n.

Now, we review some basic terminology from coding theory. A binary code C with blocklength n, rate R, and relative distance δ is simply a subset of $\{0,1\}^n$ of size 2^{Rn}, such that the Hamming distance between any two elements of C is at least δn. (Recall that the Hamming distance is the number of coordinates in which two strings differ). We say that a sequence of codes $\{C_n\}$, indexed by the blocklength n, is asymptotically good if $\lim_{n\to\infty} \delta_n > 0$, where δ_n denotes the relative distance of

C_n. Construction of asymptotically good codes with positive rate is one of the main problems in classical coding theory.

The set $\{0,1\}$ can naturally be identified with the two element field $GF(2)$, and $\{0,1\}^n$ can naturally be viewed as an n-dimensional vector space over $GF(2)$. We say that C is a linear code if C is a linear subspace of $\{0,1\}^n$. In the case of binary codes, this simply means that if c_1 and c_2 are two codewords in C, then $c_1 + c_2$ is also in C. Note that addition refers to addition in the vector space $\{0,1\}^n$, i.e., the sum of two binary strings is the bitwise exclusive-or of the two strings.

Since binary linear codes C correspond to linear subspaces of $\{0,1\}^n$, a natural method to represent a linear code C is to specify a basis for the subspace corresponding to C. Specifically, we can form a generator matrix G for C, i.e., a matrix whose row space is equal to the set of strings in C. Note that the generator matrix representation is far from being unique—any two generator matrices with the same row space represent the same code.

Another way to represent the code C is via the dual subspace. In more detail, given two vectors $u, v \in \{0,1\}^n$, let $u \cdot v$ denote the dot product of the two vectors, i.e.,

$$u \cdot v = \sum_{i=1}^{n} u_i v_i,$$

where addition and multiplication are carried out over the field $GF(2)$. Then, the set $C^\perp = \{v \in \{0,1\}^n : v \cdot u = 0 \; \forall u \in C\}$ is a linear subspace of $\{0,1\}^n$, which we call the dual code to C. It can easily be verified that $C^\perp$ uniquely specifies C, for example, by observing that $(C^\perp)^\perp = C$ for any linear subspace C. Therefore, we can specify C by giving a generator matrix H for the dual code $C^\perp$. Note that an equivalent definition of $C^\perp$ is that $C^\perp$ is the nullspace of a generator matrix G of C. Thus, C is the nullspace of H, and H is called a parity check matrix for C.

2.2 Sparse Graph Codes—Definition and Examples

We can naturally associate a bipartite graph with a matrix with entries in $\{0, 1\}$. For example, let G be an m-by-n generator matrix. Then, we can form a bipartite graph with m vertices called variable nodes, and n vertices called check nodes, as shown in Figure 2-1. We connect the i^{th} variable node to the j^{th} check node if $G_{ij} = 1$, where

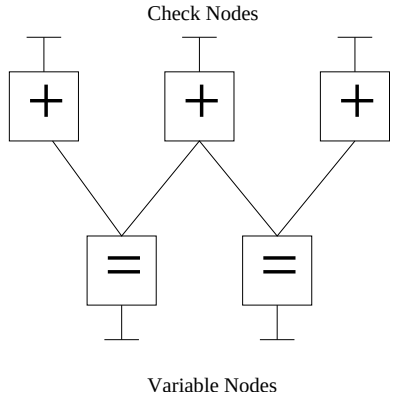

Figure 2-1: Normal graph representation of a generator matrix.

G_{ij} denotes the entry in the i^{th} row and j^{th} column of G. For example, the graph in Figure 2-1 corresponds to the matrix

$$G = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

We can perform a similar procedure for parity check matrices, except that the role of the variable and check nodes is reversed. In more detail, let H be an m-by-n parity check matrix. Then, we form a bipartite graph with n variable nodes and m check nodes, where the i^{th} check node is connected to the j^{th} variable node if $H_{ij} = 1$. Figure 2-2 shows the graph associated with the parity check matrix

$$H = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

Note that this is the same matrix that we used in our generator matrix example, and the only difference between Figures 2-1 and 2-2 is that the variable nodes and the check nodes have been interchanged. This is a special case of an important general principle—if one is given the graph of a code, interchanging the variable nodes with the check nodes gives the graph associated with the dual code.

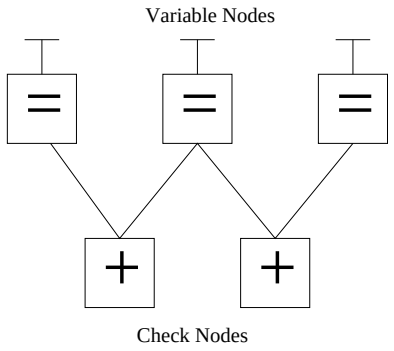

Figure 2-2: Tanner graph representation of a parity check matrix.

In the case of parity check matrices, the graph constructed above is often called the Tanner graph of the code. The use of = to denote variable nodes and + to denote check nodes is meant to be suggestive. For example, consider the Tanner graph shown in Figure 2-2. Imagine that we assign the values 0 and 1 to the edges of the Tanner graph. We say that a variable node is satisfied if the constraint that all values assigned the edges incident to that variable node have the same value, hence the = notation. Similarly, a check node is satisfied if the sum of all the values assigned to the edges incident to the check node is 0, hence the + notation. We say that the value assigned to a variable node is the value assigned to the half-edge leaving the variable node. Then, it is easy to see that the whole Tanner graph is satisfied if and only if the values assigned to the variable nodes form a string in the code C. A similar interpretation is possible for generator matrix codes. This type of graph representation is called a normal graph [42].

Now that we have constructed graphs associated with the generator matrix and parity check matrix of a code, the name sparse graph code should be self-explanatory. A sparse graph code is simply a linear code that is represented by a graph with a small

number of edges. To be explicit, note that a general Tanner graph with n variable nodes and m check nodes could have as many as mn edges. In the context of coding theory, the interesting regime is when m is proportional to n, so this means that the number of edges could grow like n^2. In this thesis, when we refer to a sparse graph code, we usually mean a graph for which the number of edges is proportional to n, i.e., the average degree of the variable nodes is bounded rather than growing with n. Occasionally, however, we may refer to graphs with $O(n \log n)$ edges or even $n^{1+\varepsilon}$ edges for some small ε as sparse graphs since these graphs still have far fewer than n^2 edges.

There are many variations of sparse graph codes, so we briefly discuss a few of the sparse graph codes used in this thesis.

2.2.1 Low Density Generator Matrix (LDGM) Codes

If a code has a generator matrix with a sparse graph representation, i.e., a generator matrix with $O(n)$ 1's, then we call the code a low density generator matrix (LDGM) code. In the context of channel coding, it is easy to convince oneself that LDGM codes are terrible channel codes. To see this, note that even for a BEC, if the channel erases all the check nodes connected to some variable node, then the value of this variable node cannot be determined correctly with probability better than random guessing. By definition, the average degree of the variable nodes in an LDGM code is bounded by a constant d independent of n, so intuitively it is clear that at least one variable node will have all of its neighbors erased, and one can easily turn this into a rigorous proof that the (block) probability of error approaches 1 for LDGM codes.

2.2.2 Low Density Parity Check (LDPC) Codes

If a code has a sparse Tanner graph, then the code is called a low density parity check (LDPC) code. Generally speaking, random ensembles of LDPC codes are considered rather than individual codes. We now describe one of the most common LDPC ensembles, an irregular (λ, ρ)-LDPC ensemble.

A (λ, ρ)-LDPC ensemble is a random code whose Tanner graph is generated as follows. Let λ and ρ be probability distributions over the positive integers, and let λ_i and ρ_i denote the probability assigned to i by λ and ρ, respectively. We construct a random Tanner graph such that λ corresponds to the degree distribution of the variables nodes in the Tanner graph, i.e., for each i, the fraction of variable nodes in the Tanner graph with degree i is λ_i. Similarly, ρ corresponds to the degree distribution of the check nodes in the Tanner graph, i.e., for each i, the fraction of check nodes in the Tanner graph with degree i is ρ_i. Let $E_\lambda = \sum_{i=1}^{\infty} \lambda_i i$ and $E_\rho = \sum_{i=1}^{\infty} \rho_i i$ be the expectations of λ and ρ, respectively. For simplicity, in the following we assume that n is an integer such that $\lambda_i n$ and $\frac{\rho_i E_\lambda}{E_\rho} n$ are integers for all i. We construct a Tanner graph with n variable nodes, $m = \frac{E_\lambda}{E_\rho} n$ check nodes, and $E = E_\lambda n$ edges using the method proposed in [76, 100]. The edges are assigned using the following procedure. We think of each variable node as having a set of sockets. Specifically, we partition the n variable nodes into disjoint sets so that $\lambda_1 n$ nodes have 1 socket, $\lambda_2 n$ nodes have 2 sockets, and so on. Note that because λ is a probability distribution, $\sum_{i=1}^{n} \lambda_i = 1$, so this procedure assigns every variable node to some set. Also, note that the total number of sockets is $\sum_{i=1}^{n} \lambda_i i n = E_\lambda n$. We create sockets for the check nodes similarly. That is, we partition the m check nodes into disjoint sets so that $\rho_1 m$ nodes have 1 socket, $\rho_2 m$ nodes have 2 sockets, and so on. As before, this procedure assigns every check node to some set, and the total number of sockets is again $E_\lambda n$. Thus, we have an equal number of sockets for the variable nodes and the check nodes. Let $v_1, \ldots, v_E$ denote the sockets assigned to variables nodes, let $c_1, \ldots, c_E$ denote the sockets assigned to check nodes, and let π denote a uniformly random permutation of the set $\{1, \ldots, E\}$. Then, the edges of the Tanner graph are formed by adding an edge to the Tanner graph between each pair $(v_i, c_{\pi(i)})$. Figure 2-3 illustrates this procedure for a small example.

Recall from Chapter 1 that unlike LDGM codes, LDPC codes can achieve low block error rates. In Chapter 1 we mentioned two different classes of LDPC codes—regular LDPC codes and irregular LDPC codes. Regular LDPC codes are codes for which λ_i and ρ_i each place all of their probability mass on a single positive integer.

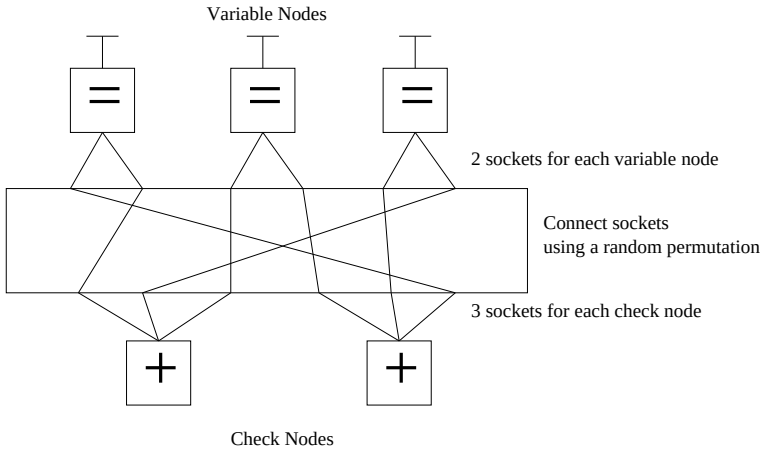

Figure 2-3: Example of a randomly constructed Tanner graph with 6 edges. Each variable node has degree 2, and each check node has degree 3.

Irregular LDPC codes are LDPC codes with general distributions λ and ρ, i.e., the distribution is not constrained to place all the mass on a single integer.

2.2.3 Nonsystematic Irregular Repeat-Accumulate (NSIRA) Codes

As we mentioned in Chapter 1, nonsystematic irregular repeat-accumulate (NSIRA) codes [97] are a variation on LDPC codes designed so that the encoding and inverse maps are trivial to compute. There exist NSIRA codes that achieve capacity for the BEC under message-passing decoding. Furthermore, these codes achieve capacity for the BEC with uniformly bounded complexity regardless of the gap to capacity.

Formally, NSIRA codes are encoded as follows. Let $m = m_1 \dots m_{Rn} \in \{0,1\}^{Rn}$ be the message. Then, we repeat each bit m_i c_i times, where c_i is a repetition factor that may be different for each bit (this is the irregularity, i.e., the I in NSIRA). Let $x_1 \dots x_{an}$ denote the an bits obtained after this repetition, where $a = \frac{1}{n}\sum_{i=1}^{Rn} c_i$. Then, we randomly permute the bits, and accumulate the result a bits at a time. Formally, let π denote a uniformly distributed permutation over the set $\{1, \dots, n\}$.

Then, the final encoding is

$$y_i = \sum_{j=1}^{ai} x_{\pi(j)} \; ; \qquad 1 \leq i \leq n.$$

The reason these codes are called nonsystematic is that the original message bits m_i are not part of the output.

Figure 2-4 shows the graph corresponding to NSIRA codes. After looking at

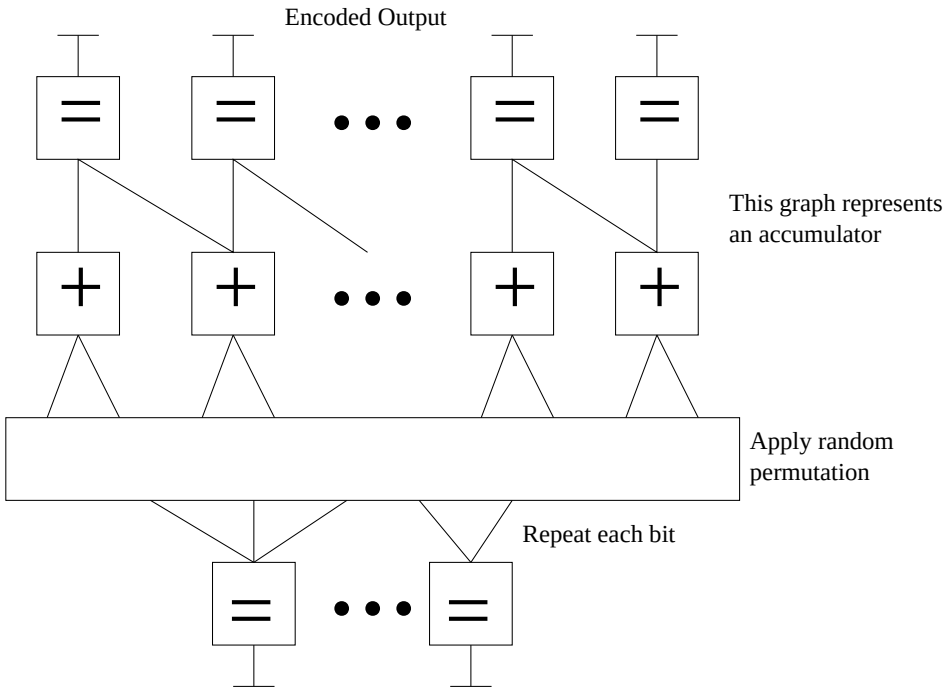

Figure 2-4: Graphical representation of a nonsystematic irregular repeat-accumulate code.

Figure 2-4, one might wonder why NSIRA codes are not just a special class of LDPC codes. NSIRA codes can be interpreted as LDPC codes where some of the variable nodes have been punctured, i.e., the variable nodes corresponding to the original message are not sent over the channel. It is this puncturing that allows NSIRA codes to get around the limitations of LDPC codes and achieve capacity for the BEC with uniformly bounded complexity.

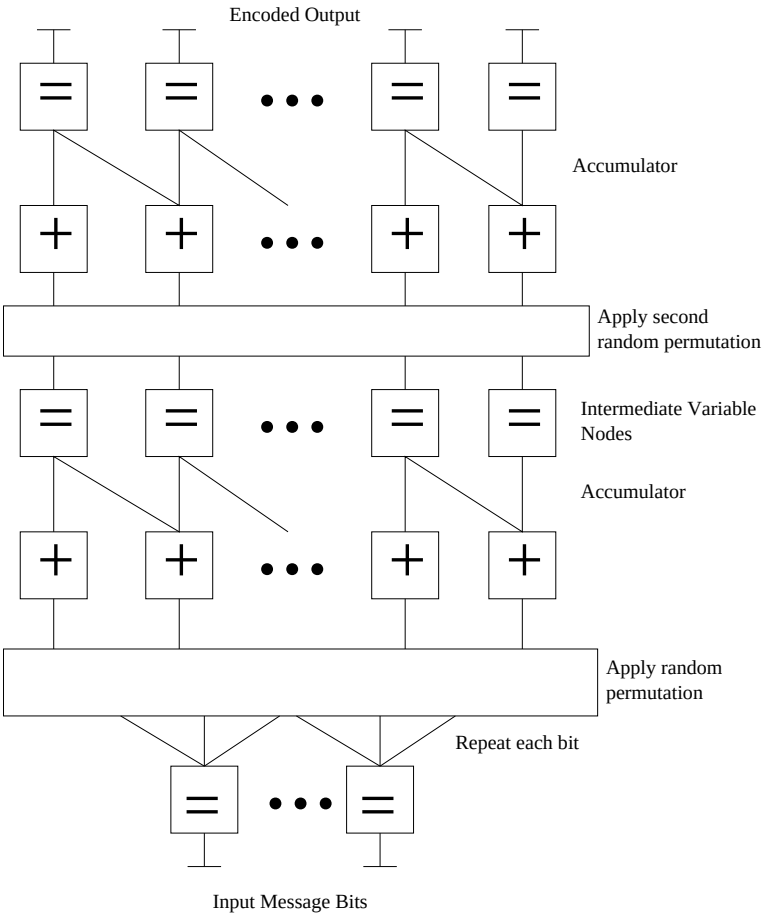

Figure 2-5: Graphical representation of a repeat-accumulate-accumulate code.

2.2.4 Repeat-Accumulate-Accumulate (RAA) Codes

Repeat-accumulate-accumulate (RAA) codes [7] are defined in terms of a repetition factor c and two independent, uniformly random permutations π_1 and π_2. Let $m = m_1 \dots m_{Rn} \in \{0,1\}^{Rn}$ be the message. Then, to encode m, we start by repeating each bit of m c times. Next, we permute these cRn bits according to π_1. Then, pass the permuted bits through an accumulator, i.e., if $x_1, \dots, x_{cRn}$ denote the permuted bits, the output of the accumulator is given by $y_i = \sum_{j=1}^{i} x_j$. To complete the encoding, we repeat this process using π_2, i.e., permute the bits according to π_2 and accumulate the result again. Figure 2-5 shows the graph associated with this encoding process. Note that although RAA codes as originally defined in [7] only include the output of the final accumulator in the encoding, we will find it convenient to include the original message and the output of the intermediate encoding in the encoding as well.

RAA codes were not designed with low-complexity decoding algorithms in mind, but their simple code structure makes them useful in our constructions for the wiretap channel. The key property of these codes is that by setting c appropriately, one can construct asymptotically good RAA codes with positive rate. As we will see in Chapter 6, the fact that RAA codes are asymptotically good but also possess additional structure makes them quite useful in the context of coding for secrecy.

2.3 Expanders and Extractors

There is a huge body of work on constructing expander graphs and the closely related concept of randomness extractors. In this section, we describe just the concepts used in this thesis. For the reader familiar with expander and extractors, this thesis relies on the Lubotzky-Phillips-Sarnak (LPS) construction of Ramanujan graphs [74], as well as Margulis' construction of spectral expanders [78]. Also, we use the zigzag construction of [19], and the Leftover Hash Lemma [59], a well-known method for constructing randomness extractors. A good reference for the reader interested in learning more about expander graphs and their applications is [57].

There are several definitions of expander graphs. The two notions that are relevant to the constructions in this thesis are spectral expansion and vertex expansion. First, we define spectral expansion. Let $\{G_n\}$ be a sequence of d-regular graphs, indexed by the number of vertices (d-regular means that every vertex is incident to d edges). Let $\lambda_1 = d \geq \lambda_2 \ldots \geq \lambda_n$ denote the eigenvalues of the adjacency matrix of G_n, sorted in descending order. We say that the sequence of graphs $\{G_n\}$ is a family of spectral expanders if

$$\limsup_{n\to\infty} \lambda_2 < d,$$

i.e., there is a constant separation between the largest eigenvalue and the second largest eigenvalue. Intuitively, such graphs are well-connected. For example, a random walk on such a graph converges to its stationary distribution quickly. We use spectral expanders indirectly, in the sense that spectral expanders form the basis for constructions of randomness extractors and vertex expanders.

Now, we discuss vertex expansion. Vertex expansion can be defined for general graphs, but in this thesis we are only interested in the bipartite case. Let $G = (X, Y, E)$ be a bipartite graph with left vertex set X, right vertex set Y, and edge set $E \subset X \times Y$. For an arbitrary set of vertices $S \subset X \cup Y$, let $\Gamma(S)$ be the set of neighbors of S, i.e., $\Gamma(S) = \{u \in X \cup Y : \exists v \in S \text{ such that } (u, v) \in E\}$.

Definition 2.3.1 (Vertex Expander)**.** *The bipartite graph $G = (X, Y, E)$ is a (c, d)-regular (K, α)-expander if every vertex in X has degree c, every vertex in Y has degree d, and for every set $S \subset X$ of size at most K, $|\Gamma(S)| \geq \alpha c|S|$.*

Intuitively, $|\Gamma(S)|$ is always at most $c|S|$, so if α is close to 1, then this says that small sets of vertices on the left have nearly disjoint sets of neighbors on the right.

As alluded to in Chapter 1, expander graphs can be used to construct codes with adversarial error-correction guarantees. For example, vertex expanders with $\alpha > .5$ can be used to construct asymptotically good error correcting codes. Specifically, in [113] it is proved that if G is a (K, α)-expander with $\alpha > .5$, and if we view G as the Tanner graph of an LDPC code, the resulting code has minimum distance at least K. Using the probabilistic method, one can prove the existence of $(K, 1-\varepsilon)$-expander graphs with N left vertices and $M = O(K \log(\frac{N}{K}))$ right vertices. Thus, in the regime where $K, N \to \infty$ such that the ratio $\frac{K}{N}$ is kept fixed, we obtain asymptotically good codes. In fact, when $\alpha > .75$, we can say even more. As shown in [113], if G is a (K, α)-expander with $\alpha > .75$, a simple message-passing algorithm can be used to correct any pattern of $\frac{K}{2}$ errors. Recall from Chapter 1 that we can use this kind of result to provide bounds on the block error rate.

Closely related to the notion of expander graphs is the concept of a randomness extractor. Roughly speaking, randomness extractors are functions designed to purify randomness. Formally, we define the min-entropy of a random variable X distributed over some finite alphabet, say $\{1, 2, \ldots, N\}$, as

$$H_0(X) = \max_{i \in \{1,\ldots,N\}} -\log(\Pr[X = i]).$$

We say that a random variable X is a (k, ε)-source if there exists some distribution

with min-entropy k whose statistical (ℓ_1) distance from X is at most ε. Finally, a (k, ε)-randomness extractor Ext is a function mapping two arguments, a source X and a random seed S uniformly distributed over some set $\{1, \ldots, 2^{k'}\}$, into the set $\{1, \ldots, 2^k\}$, satisfying the following property. For all sources X with min-entropy at least k, the random variable $(S, \mathrm{Ext}(X, S))$ is a $(k + k', \varepsilon)$-source, i.e., $(S, \mathrm{Ext}(X, S))$ has statistical distance at most ε from the uniform distribution over $\{1, \ldots, 2^{k+k'}\}$. Note that the above definition is usually referred to in the literature as the definition of a strong randomness extractor. The reason functions satisfying the above definition are called strong extractors is because the output of the function looks close to uniform even when the seed is part of the output. We use randomness extractors in our code constructions for the wiretap channel in Chapter 6.

We now describe in more detail the constructions of expanders and extractors used in Chapters 3, 4, and 6.

2.3.1 The Lubotzky-Phillips-Sarnak (LPS) Construction

In this section, we describe the LPS construction of Ramanujan graphs. The LPS construction takes two parameters, p and q, such that (a) p and q are both prime numbers, (b) p and q are both congruent to 1 mod 4, and (c) $\left(\frac{p}{q}\right) = -1$, i.e., p is not a quadratic residue mod q. The following lemma from [74] shows that the LPS construction produces spectral expander graphs with large girth.

Lemma 2.3.1. *The LPS construction produces a (bipartite) $p+1$-regular graph with $q^3 - q$ vertices, $\lambda_2 \leq 2\sqrt{p}$, and girth at least $4\log_p(q) - \log_p 4$.*

For completeness, we describe the LPS construction below. The LPS construction is simply a $(p + 1)$-regular Cayley graph for the projective linear group of 2 by 2 matrices over the finite field GF_q, i.e., $PGL(2, q)$. To be explicit, a famous theorem of Jacobi (see, for example, [55] for an elementary proof) states that the number of representations of a positive integer n as a sum of 4 squares is $8\sum_d d$, where the sum ranges over all divisors d of n not divisible by 4. Specializing this result to the case of a prime p congruent to 1 mod 4, it follows that there are precisely

$p+1$ solutions (over the integers) to the equation $p = a_0^2 + a_1^2 + a_2^2 + a_3^2$ where a_0 is odd and positive. For example, when $p = 5$, the 6 solutions are $(a_0, a_1, a_2, a_3) = (1, 2, 0, 0), (1, 0, 2, 0), (1, 0, 0, 2), (1, -2, 0, 0), (1, 0, -2, 0)$, and $(1, 0, 0, -2)$. To each of these solutions, we associate an element of $PGL(2, q)$. Specifically, because $q \equiv 1 \mod 4$, there exists an element $x \in GF_q$ satisfying $x^2 = -1$, i.e., -1 is a quadratic residue mod q. So, we associate a solution (a_0, a_1, a_2, a_3) with the matrix

$$\begin{pmatrix} a_0 + xa_1 & a_2 + xa_3 \\ -a_2 + xa_3 & a_0 - xa_1 \end{pmatrix}.$$

Finally, the graph returned by the LPS construction is simply the Cayley graph of $PGL(2, q)$ where the generating set S is the set of $p+1$ matrices associated with the solutions (a_0, a_1, a_2, a_3). Note that this set of matrices is closed under inversion, so the Cayley graph is undirected.

For future reference, we note that the Cayley graph constructed above is bipartite. Specifically, $PGL(2, q)$ contains the subgroup $PSL(2, q)$, the projective special linear group. It is easy to see that $PSL(2, q)$ and its single coset form a bipartition of $PGL(2, q)$—the determinant of any matrix in S is p, which by assumption is not a quadratic residue mod q. Thus, every edge in the Cayley graph must have one endpoint in $PSL(2, q)$ and one endpoint outside $PSL(2, q)$.

2.3.2 The Margulis Construction

This section describes Margulis' construction of expander graphs, which we use in Chapter 6 to construct extractors. The construction produces an 8-regular graph G_n with n^2 vertices, where n can be any positive integer. We think of the vertices as pairs (x, y) of integers modulo n, i.e., x and y are in the range $0, 1, \ldots, n-1$. The 8 neighbors of the vertex (x, y) are $(x + 2y, y), (x, 2x + y), (1 + x + 2y, y), (x, 1 + 2x + y), (x - 2y, y), (x, y - 2x), (y, x - 2y - 1)$, and $(x, y - 2x - 1)$, where all arithmetic is carried out modulo n. The following lemma, from [44], shows that this graph is a spectral expander.

Lemma 2.3.2. *The graph G_n satisfies $\lambda_2 \leq 5\sqrt{2} < 8$ for every positive integer n.*

2.3.3 The Zigzag Product

The zigzag construction of [19] produces a graph using 3 other bipartite graphs. Depending on the properties of the 3 input graphs, this construction can be used to construct either vertex expanders or randomness extractors. For example, in Chapter 3, we use the zigzag construction to prove the existence of vertex expanders with an efficient algorithm to compute the neighbors of a given vertex, and in Chapter 6 we use the zigzag construction to construct randomness extractors. Although we do not state this explicitly in Chapter 4, the zigzag construction can also be used to give explicit linear measurement matrices for our message-passing reconstruction algorithm, albeit with worse parameters than the best expanders known to exist via the probabilistic method.

The following lemma, which is just a special case of Theorem 7.1 from [19], summarizes the most important properties of the zigzag construction for our purposes.

Lemma 2.3.3. *Let Z_1 be the graph produced by the LPS construction (or any suitable spectral expander graph). For any $T, \varepsilon > 0$, there exist suitable constant sized graphs Z_2 and Z_3 such that the output graph Z produced by applying the zigzag construction to input graphs Z_1, Z_2, and Z_3 is a graph with N left vertices, N/T right vertices, and left degree* $\mathrm{poly}(\log(T), 1/\varepsilon)$. *Furthermore, Z is a $(\Omega(\varepsilon N/(TD)), (1-\varepsilon)D)$-bipartite expander.*

For completeness, we describe the zigzag construction below. The following is essentially just a summary of [19], although we make some very minor changes in the statements of the lemmas from [19] to suit our purposes. To explain the construction, it will be useful to view bipartite graphs as functions from the left vertices to the right vertices. Specifically, we associate a bipartite graph with its edge function $E(i, j)$. The edge function [1] gives the j^{th} neighbor of vertex i on the left, i.e., for a D-

[1] Note that the letter E is chosen in [19] as an abbreviation for extractor, not edge. The function E will usually be some kind of randomness extractor, but familiarity with the theory of random-

regular graph with N vertices on the left and M vertices on the right, E goes from $[N] \times [D] \to [M]$, where we use $[N]$ to denote the set $\{1, 2, \ldots, N\}$.

We will be interested in edge functions with special properties, as summarized in the following definitions. Note that these are identical to the definitions in [19], except that we have not taken logarithms. We use the notation U_N to denote the uniform distribution over a set of size N.

Definition 2.3.2. *A function $E : [N] \times [D] \to [M]$ is called an (ε, A) extracting conductor if for any $1 \leq 2^k \leq \frac{M}{A}$ and any source X over $[N]$ such that $H_0(X) \geq k$, the distribution of $E(P, U_D)$ is a $(k + \log(A), \varepsilon)$-source.*

Definition 2.3.3. *A function $E : [N] \times [D] \to [M]$ is called a (K_{max}, ε) lossless conductor if for any $1 \leq 2^k \leq K_{max}$ and any source X over $[N]$ such that $H_0(X) \geq k$, the distribution of $E(P, U_D)$ is a $(k + \log(D), \varepsilon)$-source.*

Definition 2.3.4. *A pair of functions $E : [N] \times [D] \to [M]$ and $C : [N] \times [D] \to [B]$ is called a $(K_{max}, \varepsilon, A)$ buffer conductor if E is an (ε, A) extracting conductor and the pair (E, C), viewed as a function from $[N] \times [D] \to [M] \times [D]$, is a (K_{max}, ε) lossless conductor.*

As a special case of a buffer conductor, note that if (E, C) is a permutation, then (E, C) is automatically an $(N, 0)$ lossless conductor.

Definition 2.3.5. *A pair of functions $E : [N] \times [D] \to [M]$ and $C : [N] \times [D] \to [B]$ is called a (ε, A) permutation conductor if E is an (ε, A) extracting conductor and the pair (E, C), viewed as a function from $[N] \times [D] \to [M] \times [D]$, is a permutation.*

Now, we define the graph, i.e., edge function, produced by the zigzag construction. Let Z_1, Z_2, and Z_3 be input graphs with the following properties. The graph Z_1 has N_1 vertices on the left, each with degree D_1, and $M_1 B_1$ vertices on the right, so the associated edge function is defined from $[N_1] \times [D_1] \to [M_1 B_1]$. Because $[M_1 B_1]$ can naturally be identified with $[M_1] \times [B_1]$, we represent the edge function for Z_1 as

ness extractors is not needed to understand the zigzag construction. Rather, the connection with randomness extractors is useful for analyzing the properties of the construction.

(E_1, C_1), where E_1 goes from $[N_1] \times [D_1] \to [M_1]$ and C_1 goes from $[N_1] \times [D_1] \to [B_1]$. The graph Z_2 has N_2 vertices on the left, each with degree D_2, and $D_1 B_2$ vertices on the right. We represent the edge function for Z_2 as (E_2, C_2), where E_2 goes from $[N_2] \times [D_2] \to [D_1]$ and C_2 goes from $[N_2] \times [D_2] \to [B_2]$. Finally, Z_3 has $B_1 B_2$ vertices on the left, each with degree D_3, and M_3 vertices on the right. We represent the edge function for Z_3 as $E_3 : ([B_1] \times [B_2]) \times [D_3] \to [M_3]$, where, as before, we have used the natural correspondence between the sets $[B_1 B_2]$ and $[B_1] \times [B_2]$.

The output graph Z produced by the zigzag construction has $N_1 N_2$ left vertices, each with degree $D_2 D_3$, and $M_1 M_3$ right vertices. The only thing left to specify is the edge function $E : ([N_1] \times [N_2]) \times ([D_2] \times [D_3]) \to ([M_1] \times [M_3])$. We compute $E((x_1, x_2), (r_2, r_3))$ as follows.

$$\begin{aligned}
(r_1, z_1) &\triangleq (E_2, C_2)(x_2, r_2) \\
(y_1, z_2) &\triangleq (E_1, C_1)(x_1, r_1) \\
y_2 &\triangleq E_3((z_1, z_2), r_3) \\
E((x_1, x_2), (r_2, r_3)) &\triangleq (y_1, y_2).
\end{aligned}$$

The above construction is well-defined whenever the graphs have the appropriate sizes and degrees. However, to produce graphs with good expansion, we must impose additional conditions on Z_1, Z_2, and Z_3. To prove Lemma 2.3.3, we take Z_2 to be an optimal buffer conductor of constant size, and we take Z_3 to be an optimal lossless conductor of constant size. The parameters of such conductors are described by Lemmas 4.2 and 4.3 of [19]. For our application, it is convenient to make Z_2 and Z_3 left- and right-regular as well. It is easily verified that essentially the same performance as in lemmas 4.2 and 4.3 of [19] can be obtained even with the regularity restriction.

2.3.4 The Leftover Hash Lemma

In this section, we describe the Leftover Hash Lemma [59], a simple method for constructing extractors. Since the proof of correctness is short, we include the proof to keep this thesis as self-contained as possible. Roughly speaking, the Leftover Hash Lemma says that a good family of hash functions can be used to construct extractors.

Lemma 2.3.4. *Let $\{h_s\}$ be a family of hash functions mapping $\mathcal{X}$ to $\mathcal{Y}$. Let $|\mathcal{Y}| \leq \varepsilon^2 2^k$, and let $\mathcal{S}$ denote the index set of the family $\{h_s\}$. Assume that for all $x \in \mathcal{X}$, $h_S(x)$ is uniformly distributed over $\mathcal{Y}$, and that for all $x_1 \neq x_2 \in \mathcal{X}$, the collision probability*

$$\Pr[h_S(x_1) = h_S(x_2)] \leq \frac{1}{|\mathcal{Y}|},$$

where S is a random variable uniformly distributed over $\mathcal{S}$. Then, the function $\mathrm{Ext} : \mathcal{X} \times \mathcal{S} \to \mathcal{Y}$ *defined by* $\mathrm{Ext}(X, S) = (h_S(X))$ *is a (k, ε)-extractor.*

Before proving Lemma 2.3.4, we state a preliminary lemma.

Lemma 2.3.5. *Let k be such that 2^k is an integer, and let X be a random variable such that $H_0(X) \leq k$. Then, X can be expressed as a convex combination of random variables X_i, where each X_i is uniformly distributed over some set of size 2^k.*

Proof. Let $\{1, 2, \ldots, n\}$ denote the support of X, and let $p_1, p_2, \ldots, p_n$ denote the probability distribution of X. The condition that $H_0(X) \geq k$ can be expressed by the linear constraints

$$0 \leq p_i \leq 2^{-k} \text{ for } 1 \leq i \leq n,$$
$$\sum_{i=1}^{n} p_i = 1.$$

The lemma is equivalent to the statement that any vertex of the polytope specified by the above constraints has the property that 2^k of the p_i's are 2^{-k}, and the remaining $n - 2^k$ p_i's are 0. This is obvious, because any vertex is defined by the intersection of n linearly independent constraints, and it is easily verified that the intersection of n of the constraints has the property that every p_i is either 0 or 2^{-k}. □

Proof of Lemma 2.3.4. We must show that for any random variable X such that $H_0(X) \geq k$, the distribution of $(S, h_S(X))$ is close to the uniform distribution over $\mathcal{S} \times \mathcal{Y}$. From lemma 2.3.5, it suffices to consider random variables X uniformly distributed over a set of size 2^k, so in the following we assume that X is of this form. For such X, we must show that

$$\sum_{s,y} \left| \Pr[S = s \text{ and } h_S(X) = y] - \frac{1}{|\mathcal{S}||\mathcal{Y}|} \right| \leq \varepsilon.$$

This sum can be rewritten as

$$\sum_{s} \Pr[S = s] \sum_{y} \left| \Pr[h_S(X) = y | S = s] - \frac{1}{|\mathcal{Y}|} \right| \leq \varepsilon.$$

The inner sum is the statistical distance between the output of the hash function h_S and the uniform distribution, conditioned on the event $S = s$, so the sum can be viewed as the expected value of the statistical (ℓ_1) distance $D_1(S)$ between $h_S(X)$ and the uniform distribution, where the expectation is over S. Instead of directly computing a bound on the expectation of $D_1(S)$, we bound $E[D_2(S)^2]$, where $D_2(S)$ denotes the ℓ_2 distance.

We analyze $E[D_2(S)^2]$ using the bound on collision probability. Specifically,

$$\begin{aligned} E[D_2(S)^2] &\leq \frac{1}{2^{2k}} \left(2^k + \sum_{x_1 \neq x_2} \left(\Pr[h_S(x_1) = h_S(x_2)] - \frac{1}{|\mathcal{Y}|} \right) \right) \\ &\leq 2^{-k}. \end{aligned}$$

To complete the proof, note that the Cauchy-Schwartz inequality implies that

$$\begin{aligned} E[D_1(S)] &\leq \sqrt{|\mathcal{Y}|} 2^{-.5k} \\ &\leq \varepsilon, \end{aligned}$$

which proves that $\mathrm{Ext}(X, S) = (h_S(X))$ is a (k, ε)-extractor. □

The following lemma is just a special case of 2.3.4, since the function $\mathrm{Ext}(x, s) = (x \cdot s)^{\lfloor k + 2 \log \varepsilon \rfloor}$ clearly satisfies the hypotheses of Lemma 2.3.4.

Lemma 2.3.6. *(Leftover Hash Lemma) The function* $\mathrm{Ext}(x, s) = (x \cdot s)^{\lfloor k+2\log\varepsilon \rfloor}$ *is a* (k, ε)*-extractor. The superscript notation indicates that the output of the extractor is only the first* $\lfloor k + 2\log\varepsilon \rfloor$ *bits of the product.*

Chapter 3

Locally Encodable and Decodable Source Codes

3.1 Introduction

In this chapter, we consider the problem of constructing locally encodable and decodable source codes. As described in Chapter 1, the problem is to construct good (lossless) source codes satisfying two additional properties—local encodability and local decodability. Recall that local encodability is the property that when a single component of the source is changed, it is easy to update the compressed output. Local decodability is the property that any component of the source can be recovered efficiently from the compressed output. For example, it should possible to recover a single component much more efficiently than by running the decompression algorithm to recover the entire source. It is easy to think of scenarios where source codes possessing these properties could be useful. We mentioned a couple of scenarios in Chapter 1, e.g., updating a thesis and storing a database. As another example, in a high-speed network router one wishes to count the number of packets per flow in order to police the network or identify a malicious flow. Since packets arrive at very high speed, and in a large network we may want to quickly determine the number of packets in an important flow rather than obtain the data for every single flow, we are naturally led to consider the problem of maintaining dynamically changing integers.

Similar problems appear in many other applications that arise in networks, databases, signal processing, etc. In the remainder of this chapter, we focus exclusively on the problem of constructing a locally encodable and decodable source code for a class of integer sources motivated by the high-speed network router scenario, but we emphasize that by no means is this the only interesting class of sources to consider. An important challenge for future work is to construct locally encodable and decodable source codes for a broader class of sources.

In the high-speed network router scenario and related applications, there are two key requirements. First, we need to maintain the collection of integers in a dynamic manner so that any integer can be updated (increment, decrement) and evaluated very quickly. Second, we need the source to be represented as compactly as possible to minimize storage, which is often highly constrained in these scenarios. Therefore, we are interested in source codes capable of storing N integers in a compressed manner while maintaining the ability to read and write any of these N integers in nearly constant time.[1] Since a source code capable of representing N integers can also be viewed as a data structure for storing N integers, in the following we use the terms source code and data structure interchangeably.

3.1.1 Prior work

Efficient storage of a sequence of integers in a compressed manner is a classical problem of *source coding*, with a long history, starting with the pioneering work by Shannon [107]. The work of Lempel and Ziv [122] provided an elegant universal compression scheme. Since that time, there has continued to be important progress on compression algorithms. However, most well-known schemes suffer from two drawbacks. First, these schemes do not support efficient updating of the compressed data. This is because in typical systems, a small change in one value in the input sequence leads to a huge change in the compressed output. Second, such schemes require the

[1] Or, at worst , in poly-logarithmic time (in N). Moreover, we will insist that the space complexity of the algorithms utilized for read and write/update be of smaller order than the space utilized by the data structure.

whole sequence to be decoded from its compressed representation in order to recover even a single element of the input sequence.

A long line of research has addressed the second drawback. For example, Bloom filters [14, 20, 93] are a popular data structure for storing a set in compressed form while allowing membership queries to be answered in constant time. The rank/select problem [24, 50, 51, 63, 86, 87, 92, 98] and dictionary problem [43, 92, 94] in the field of succinct data structures are also examples of problems involving both compression and the ability to efficiently recover a single element of the input, and [95] gives a succinct data structure for arithmetic coding that supports efficient, local decoding, but not local encoding. In summary, this line of research successfully addresses the second drawback but not the first drawback, e.g., changing a single value in the input sequence can still lead to huge changes in the compressed output.

Similarly, several recent papers have examined the first drawback but not the second. In order to be able to update an individual integer efficiently, one must consider compression schemes that possess some kind of *continuity* property, whereby a change in a single value in the input sequence leads to a small change in the compressed representation. In recent work, Varshney et al. [117] analyzed "continuous" source codes from an information-theoretic point of view, but the notion of continuity considered is much stronger than the notion we are interested in, and [117] does not take computational complexity into account. Also, Mossel and Montanari [84] have constructed space efficient "continuous" source codes based on nonlinear, sparse graph codes. However, because of the nonlinearity, it is unclear that the continuity property of the code is sufficient to give a computationally efficient algorithm for updating a single value in the input sequence.

In contrast, linear, sparse graph codes have a natural, computationally efficient algorithm for updates. A large body of work has focussed on such codes in the context of streaming algorithms and compressed sensing. Most of this work essentially involves the design of *sketches* based on linear codes [16, 17, 25, 35, 48, 60–62, 88, 116, 121]. Among these, [25, 48, 60–62, 88, 121] consider sparse linear codes. Existing solutions from this body of work are ill-suited for our purposes for two reasons. First, many of

the decoding algorithms in the literature (LP based or combinatorial) require decoding all integers to read even one integer—notable exceptions include the Count-Min and Count-Sketch algorithms [21, 25], which do support local decoding. Second, most existing algorithms are suboptimal in terms of space utilization when the input is very sparse, e.g., the Count-Min and Count-Sketch algorithms do not achieve compression rates near the information-theoretic limit.

Perhaps the work most closely related to the results in this chapter is that of Lu et al. [73], which develops a space efficient linear code. They introduced a multi-layered linear graph code (which they refer to as a Counter Braid). In terms of graph structure, the codes considered in [73] are essentially identical to Tornado codes [76], so Counter Braids can be viewed as one possible generalization of Tornado codes designed to operate over the integers instead of a finite field. In [73], it is shown that space-efficient Counter Braids exist provided that (a) the graph structure and layers are carefully chosen depending upon the distributional information of inputs, and (b) the decoding algorithm is based on maximum likelihood (ML) decoding, which is in general computationally very expensive. The authors propose a message-passing algorithm to overcome this difficulty and provide an asymptotic analysis to quantify the performance of this algorithm in terms of space utilization. However, the message-passing algorithm may not provide optimal performance in general.

In what follows, we describe the precise problem formulation, our data structure, and the associated algorithms for local encoding and local decoding. Before detailing the construction, though, we provide some perspectives. The starting point for our construction is the set of layered, sparse graphical codes introduced in [73]. In order to obtain the desired result, we have to overcome two non-trivial challenges. The first challenge concerns controlling the space utilized by the auxiliary information in order to achieve overall space efficiency. In the context of sparse graphical codes, this includes information about the adjacency matrix of the graph. Storing the adjacency matrix naively, say with an adjacency list representation, requires $\Omega(N \log N)$ space for an $N \times \Theta(N)$ connected, bipartite graph, which can already be much more than $N \log A$ when A is small compared to N. Therefore, it is necessary to use graphs that

can be stored succinctly. The second challenge concerns the design of local encoding and decoding algorithms. Linear sparse graph codes have a natural local encoding algorithm, but it is not clear how to design a local decoding algorithm. As alluded to in the earlier, most existing decoding algorithms in the literature are designed for decoding the entire vector rather than just a single component. Therefore, we need to develop a novel decoding algorithm.

To address the first challenge, we use random graphs based on appropriate Ramanujan graphs. These graphs have a succinct description (i.e., $O(\sqrt{N}\log N)$ space), efficient 'look up' (i.e., there exists a constant time algorithm that can use the succinct description to determine the neighbors of any given vertex), and a 'locally tree-like' property (i.e., girth at least $\Omega(\log N)$). To address the second challenge, we take advantage of our graphs' large girth to design a local decoding algorithm based on a message-passing algorithm. Intuitively, our algorithm can be viewed as a procedure for obtaining successively refined estimates of a single component of the source based on the local graph structure.

3.2 Problem Formulation

Source Model. The source $\mathbf{x}$ is a vector of N integers, denoted $x_1, \ldots, x_N$, constrained so that $\mathbf{x}$ lies in the set

$$\mathcal{S}(A, M, N) = \{\mathbf{x} \in \mathbb{N}^N : \|\mathbf{x}\|_1 \leq AN, \quad \|\mathbf{x}\|_\infty \leq M\}.$$

Let $M \geq A$ without loss of generality, where the average A and maximum M can be arbitrary functions of N such that $A \leq M \leq NA$. Although our data structure can be designed for any given value of A, our results are most meaningful in the regime in which both N and A (and thus M) are large. We wish to design a data structure that can store any $\mathbf{x} \in \mathcal{S}(A, M, N)$. Note that we do not assume that $\mathbf{x}$ is drawn from a probability distribution, but rather that $\mathbf{x}$ can be any element from $\mathcal{S}$, so potentially the source might even be chosen by an adversary with the hope of

causing our algorithm to fail.

Performance Criteria. We measure the performance of a data structure by considering the following performance metrics:

> *Write/Update.* For any $i, 1 \leq i \leq N$, x_i can be updated (increment, decrement or write) in near constant time. That is, the data structure is locally encodable.
>
> *Read.* For any $i, 1 \leq i \leq N$, x_i can be read in nearly constant time. That is, the data structure is locally decodable.
>
> *Space.* The amount of space utilized is the minimal possible. The space utilized for storing auxiliary information about the data structure (e.g., pointers or random bits) as well as the space utilized by the read and write algorithms should be accounted for.

Properties and Drawbacks of Some Naive Solutions. We briefly discuss a few naive solutions, each with some of these properties but not all, that will help explain the non-triviality of this seemingly simple question. First, if each integer is allocated $\log M$ bits, then one solution is the standard Random Access Memory: it has $O(1)$ read and write complexity. However, the space required is $N \log M$, which can be much larger than the minimal space required (as we'll establish, the minimal space required is essentially $N \log A$). To overcome this poor space utilization, consider the following prefix-free coding solution. Here, each x_i is stored in a serial manner using a prefix-free code, for example the Elias delta code [38]. Such a prefix-free code requires roughly $\log x_i + \log \log x_i$ bits to represent x_i (in the limit of large x_i), so the prefix-free coding solution is essentially optimal in that it utilizes (roughly) $N(\log A + \log \log A)$ bits. However, the read and write/update operations have $\Omega(N)$ complexity, e.g., if the length of x_1's prefix-free representation increases, then we must shift over the representations of $x_2, \ldots, x_N$ to make more room for x_1.

As a third example, consider the case $A = 1/N$ and $M = 1$, i.e., the input x has at most one entry set to 1, and all other entries are 0. For this choice of parameters, we can use the Hamming code to store x. Specifically, let H denote the parity check

matrix of the Hamming code, i.e., the i^{th} column of H is the binary expansion of i. Then, our data structure simply stores the syndrome Hx. The write complexity is $O(1)$.[2] The read complexity is also $O(1)$ in the worst-case, because Hx is nothing but the binary expansion of the position of the 1 in x. Thus, the Hamming code gives a solution that uses optimal space (when N is one less than a power of 2), and read and write both take $O(1)$ time, so the Hamming code gives an example where the data is compressed, and read and write can still be accomplished with low complexity.

Note that without loss of generality, M can be chosen so that $A \leq M \leq AN$, because the maximum must be at least the average, and if the average of a set of nonnegative numbers is at most A, then the maximum value of any number in the set is at most AN. Thus, $A = 1/N$ and $M = 1$ represents the maximum possible gap between the average and maximum. At the other extreme, consider the case where $A = M$. In this case, our first example (the Random Access Memory) uses the optimal amount of space, and read and write have $O(1)$ complexity. Thus, at either extreme of the A/M region, it is possible to perform optimal compression and have low complexity read and write algorithms. In this chapter, we desire solutions that achieve similar performance for a broader range of settings of the parameters N, A, and M.

3.3 Construction

This section describes the construction and corresponding read/write algorithms used to prove our main result. A caricature of the data structure is portrayed in Figure 3-1.

3.3.1 Overall Structure

Let L denote the number of layers. Let $V_0 = \{1, \ldots, N\}$, with $i \in V_0$ corresponding to x_i. Each vertex $i \in V_0$ has a counter of b_0 bits allocated to it. Because of this association, in what follows, we use the terms vertex and counter interchangeably,

[2] We assume that operations on $\log N$-bit integers take constant time.

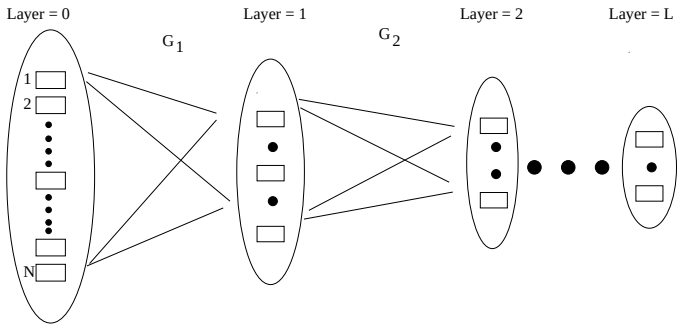

Figure 3-1: An example of the data structure.

e.g., sometimes we call the i^{th} vertex in V_0 the i^{th} counter in V_0, and vice versa. Intuitively, the b_0 bits corresponding to the i^{th} vertex of V_0 are used to store the least significant b_0 bits of x_i, while layers 1 through L are used to store the more significant bits. Since the average is constrained, we should be able to share storage for the more significant bits of the x_i, because many of these more significant bits must be 0 to keep the average below A. To formalize this intuition, let V_ℓ denote the collection of vertices corresponding to layer $\ell, 1 \leq \ell \leq L$, with $|V_\ell| \leq |V_{\ell-1}|$ for $1 \leq \ell \leq L$. Each vertex $i \in V_\ell$ is allocated a counter of b_ℓ bits. The vertices of layers $\ell - 1$ and ℓ, i.e., $V_{\ell-1}$ and V_ℓ, are connected by a bipartite graph $G_\ell = (V_{\ell-1}, V_\ell, E_\ell)$ for $1 \leq \ell \leq L$. We denote the i^{th} counter (stored at the ith vertex) in V_ℓ by (i, ℓ), and use $c(i, \ell)$ to denote the value stored by counter (i, ℓ). The role of G_ℓ is to set up associations between counters in layers $\ell - 1$ and ℓ. Roughly speaking, G_ℓ tells us how to propagate information when a counter overflows. The precise role of G_ℓ will become clear after Section 3.3.3.

For technical reasons, we create another copy of layers $1, \ldots, L$ with vertices V'_ℓ and corresponding graphs G'_ℓ for $1 \leq \ell \leq L$. Each vertex in V'_ℓ has b'_ℓ bits allocated to it, $G'_1 = (V_0, V'_1, E'_1)$, and $G'_\ell = (V'_{\ell-1}, V'_\ell, E'_\ell), 2 \leq \ell \leq L$. We denote the i^{th} counter in V'_ℓ by $(i, \ell)'$, and we use $c(i, \ell)'$ to denote the value stored by counter $(i, \ell)'$.

3.3.2 Setting Parameters

Now we define the parameters $L, b_\ell, b'_\ell, |V_\ell|$, and $|V'_\ell|$ for $0 \leq \ell \leq L$. Later, we describe the graphs G_ℓ and $G'_\ell, 1 \leq \ell \leq L$. Let C and $K \geq 3$ be positive integers, and let

$\varepsilon = 3/K$. Parameters C and ε control the excess space used by the data structure compared to the information-theoretic limit. Set L as

$$L = \left(\log\log\left(\frac{M}{2K^2A^2}\right) - \log\log\left(A\right)\right)^+ + C,$$

where $x^+ = \max\{x, 0\}$. Set $b_0 = \log(K^2A)$, and recall that $|V_0| = N$. Let $|V_1| = \varepsilon N$, and let $b_1 = \log(12KA)$. For $2 \le \ell \le L-1$, $|V_\ell| = 2^{-(\ell-1)}\varepsilon N$, and $b_\ell = 3$. Finally, $|V_L| = 2^{-(L-1)}\varepsilon N$, and $b_L = \log\left(\frac{M}{2K^2A^2}\right)$. For the second series of layers, we set $|V_1'| = \varepsilon N$, and $b_1' = \log(rA)$, where $r = K\text{poly}\left(\log\left(K\right)\right)$. For $2 \le \ell \le L-1$, $|V_\ell'| = \varepsilon^\ell N$, and $b_\ell' = r$. For the last set, $|V_L'| = \varepsilon^L N$, and $b_L' = \log\left(\frac{Mr}{K^2A^2}\right)$.

Description of G_ℓ

Here we describe the graphs $G_\ell, 1 \le \ell \le L$. Recall that we want these graphs to have a succinct description, efficient look up, and large girth. We also want the graphs to 'look random', i.e., the graphs need to be random enough to enable our analysis. With these properties in mind, we start by describing the construction of G_L. We start with a bipartite graph H with the following properties: (a) H has $\sqrt{|V_{L-1}|}$ vertices on the left and $.5\sqrt{|V_{L-1}|}$ vertices on the right; and (b) H is $(3,6)$-regular. The construction of such an H will be described later, but first we use it to define G_L. Let $\tilde{G}$ be a graph consisting of $\sqrt{|V_{L-1}|}$ disjoint copies of H. Formally, $\tilde{G}$ has left vertices $v_0, v_1, \ldots, v_{|V_{L-1}|-1}$ and right vertices $w_0, w_1, \ldots, w_{|V_L|-1}$. Connect $v_0, v_1, \ldots, v_{\sqrt{|V_{L-1}|}-1}$ to $w_0, w_1, \ldots, w_{.5\sqrt{|V_{L-1}|}-1}$ using H. Then, connect $v_{\sqrt{|V_{L-1}|}}, v_{1+\sqrt{|V_{L-1}|}}, \ldots, v_{2\sqrt{|V_{L-1}|}-1}$ to $w_{.5\sqrt{|V_{L-1}|}}, w_{1+.5\sqrt{|V_{L-1}|}}, \ldots, w_{\sqrt{|V_{L-1}|}-1}$ using H, and so on. G_L is constructed to be isomorphic to $\tilde{G}$. Specifically, every left vertex $i \in \{0, \ldots, |V_{L-1}| - 1\}$ of G_L inherits the neighborhood of left vertex $Q(i) \in \{0, \ldots, |V_{L-1}| - 1\}$ of $\tilde{G}$, where $Q(i)$ is defined as follows:

1. Let $F : \{0, \ldots, |V_{L-1}|-1\} \to \sqrt{|V_{L-1}|} \times \sqrt{|V_{L-1}|}$ be such that $F(i) = (r(i), c(i))$, where

$$r(i) = i \mod \sqrt{|V_{L-1}|} \quad \text{and}$$

$$c(i) = \left(i + \left\lfloor \frac{i}{\sqrt{|V_{L-1}|}} \right\rfloor \right) \mod \sqrt{|V_{L-1}|}.$$

2. Let the random map $R : \sqrt{|V_{L-1}|} \times \sqrt{|V_{L-1}|} \to \sqrt{|V_{L-1}|} \times \sqrt{|V_{L-1}|}$ be defined as $R(x, y) = (\pi_1(x), \pi_2(y))$, where π_1, π_2 are two permutations of length $\sqrt{|V_{L-1}|}$ chosen independently and uniformly at random.

3. $Q = F^{-1} \circ R \circ F$.

To gain some intuition for Q, imagine that we arrange the vertices in V_{L-1} in a square $\sqrt{|V_{L-1}|}$-by-$\sqrt{|V_{L-1}|}$ grid. There are many ways of assigning vertices to cells in the grid, and the mapping F has the nice property that the left vertices of each copy of H are mapped so that no two vertices are in the same row or column of the grid. For example, the vertices $v_0, v_1, \ldots, v_{\sqrt{|V_{L-1}|}-1}$ are mapped to the main diagonal of the grid. The random map R corresponds to applying independent random permutations to the rows and columns of the grid. This interpretation of Q as a mapping applying random permutations to the rows and columns of the grid is useful in our analysis of the READ algorithm, specifically in the proof of Lemma A.2.8.

The construction of $G_\ell, 2 \leq \ell \leq L-1$ follows a similar procedure, with a different number of copies of H used to construct the associated $\tilde{G}$ to account for the varying size of V_ℓ (and of course, new randomness to define the corresponding Q). Finally, G_1 is constructed in essentially the same way as G_L, but with a different $(3, K)$-regular base graph H_1. The base graphs H and H_1 are based on the LPS construction of Ramanujan graphs. The details of the construction are given in Appendix 3.8.1.

In summary, the graphs $G_\ell, 1 \leq \ell \leq L$ constructed above have the following properties.

Lemma 3.3.1. *For $1 \leq \ell \leq L$, G_ℓ has the following properties: (a) G_ℓ can be stored using $O(\sqrt{N} \log N)$ space, (b) for any vertex v (in the left or right partition), v's neighbors can be computed in $O(1)$ time, and (c) G_ℓ has girth $\Omega(\log N)$.*

Description of G'_ℓ

We require different properties from the G'_ℓ's than from the G_ℓ's. We still want graphs with succinct descriptions and efficient look up, but instead of large girth, we want the G'_ℓ's to be very good (vertex) expanders, as defined in Section 2.3. The G'_ℓ's can

be chosen deterministically, because our analysis does not rely on any randomness in the G'_ℓ's. To construct G'_ℓ, we use the zigzag construction described in Section 2.3.3. In Appendix 3.8.1, we describe the precise choice of graphs we use in the zigzag construction to guarantee the following properties of the G'_ℓ's.

Lemma 3.3.2. *For sufficiently large K, there exists constants $l = \text{poly}(\log(K))$ and $r = l/\varepsilon$ such that G'_ℓ has the following properties: (a) G'_ℓ can be stored using $O(1)$ space[3], (b) for any vertex v (in the left or right partition), v's neighbors can be computed in $O(1)$ time, and (c) G'_ℓ is an (l, r)-regular $(2/K^2, 2/3)$-expander.*

3.3.3 Preliminary Illustration of Encoding/Decoding Operations

In this section we explain how the previously described data structure can be used to store the source. Specifically, we work out a small example to illustrate the basic steps required by the read and write algorithms. To this end, consider the situation when $N = 4$ and $L = 2$. That is, $V_0 = \{1, 2, 3, 4\}$. Let V_1 and V_2 be such that $|V_1| = 2$ and $|V_2| = 1$. Let $b_0 = 2$, $b_1 = 2$, and $b_2 = 2$, and initially let all counters be set to 0, i.e., initially the data structure stores the source vector $x_1 = x_2 = x_3 = x_4 = 0$. We start by explaining the write algorithm. Say that we want to write $x_3 = 2$, or equivalently, we wish to increment x_3 by 2. We accomplish this by adding 2 to $c(3, 0)$. Since the counters in layer 0 are allocated two bits each, these counters have a capacity of 4, i.e., they can store any value in the set $\{0, 1, 2, 3\}$. Therefore, we add 2 to the current value modulo 4. That is, the value in counter $(3, 0)$ is updated to $0 + 2 \mod 4 = 2$. The resulting 'overflow' or 'carry' is $\lfloor (0 + 2)/4 \rfloor = 0$. Since the overflow is 0, no further operations are required. This is shown in Figure 3-2(a).

Now, suppose x_3 is increased further by 3. Then, $c(3, 0)$ is changed to $2 + 3 \mod 4 = 1$, and a 'carry' of $\lfloor (2 + 3)/4 \rfloor = 1$ is added to the counters in layer 1 that are connected to $(3, 0)$ via the graph G_1, i.e., counter $(1, 1)$. Repeating the same

[3]Even if we use a model of computation that requires us to store the multiplication tables for certain finite fields explicitly, the space is $o(N)$.

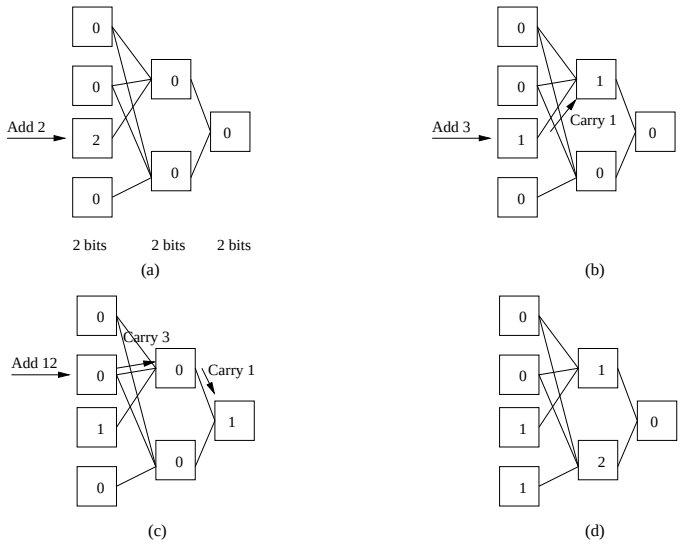

Figure 3-2: (a) Add 2 to x_3, (b) add 3 to x_3, (c) add 12 to x_2, and (d) initial config. for decoding.

process, $c(1,1)$ is changed to $0+1 \mod 4 = 1$. The resulting carry is 0, so no further changes are required. This is shown in Figure 3-2(b). Using a similar approach, writing $x_2 = 12$ will lead to Figure 3-2(c). Note that the same algorithm can be used to decrement any of the x_i, by treating a decrement as an increment by a negative value.

Finally, we give a simplified example illustrating the intuition behind the decoding algorithm. Figure 3-2(d) shows the initial configuration. The decoding algorithm uses the values stored in the counters to compute upper and lower bounds on the value of a given x_i. To see how this can be done for the simple example above, let us compute upper and lower bounds on x_3. First, observe that the counter $(1,2)$ stores 0. Therefore, both counters in V_1 did not overflow. Thus, each counter in V_1 must currently store the sum of the overflows of its neighbors in V_0.

We know $c(3,0)$, so the problem of determining upper and lower bounds on x_3 is equivalent to determining upper and lower bounds on the overflow of counter $(3,0)$. To determine a lower bound, consider the tree obtained by doing a breadth-first search of depth 2 in G_1 starting from $(3,0)$, as shown in Figure 3-3(a). A trivial lower bound for the overflow of the bottom two counters is 0. Therefore, $1-0-0=1$ must be an upper bound on the overflow of $(3,0)$.

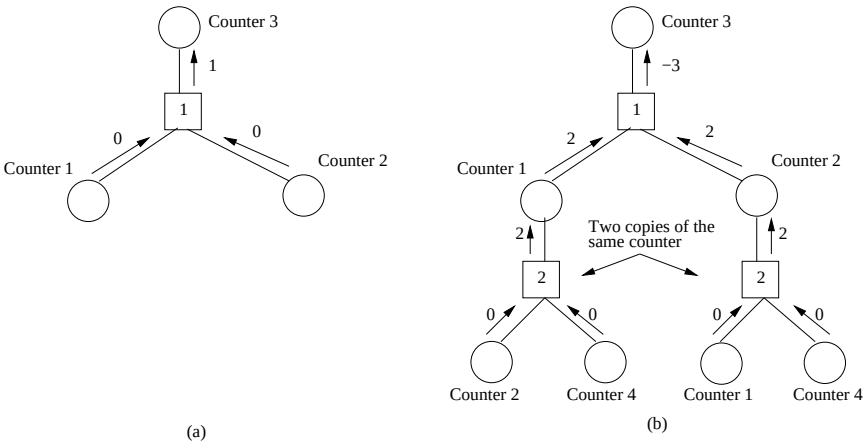

Figure 3-3: (a) Breadth-first search tree of depth 2 rooted at $(3, 0)$, (b) breadth-first search tree of depth 4 rooted at $(3, 0)$.

Next, consider the graph obtained by doing a breadth-first search of depth 4 in G_1 starting from $(3, 0)$. This graph is not a tree, but we can pretend that it is a tree by making copies of vertices that are reached by more than one path, as shown in Figure 3-3(b). As before, 0 is a trivial lower bound for the overflow of the counters at the bottom of the tree. Therefore, $2 - 0 - 0 = 2$ is an upper bound for both counters at depth 2 in the tree. This means that $1 - 2 - 2 = -3$ must be a lower bound on the overflow of $(3, 0)$. Of course, in this case the trivial lower bound of 0 is better than the bound of -3 obtained by this reasoning. Since our upper bound on the overflow is 1 and our lower bound is 0, we haven't recovered the exact value of counter $(3, 0)$'s overflow. However, in general, if this type of reasoning leads to matching upper and lower bounds, then the value of $(3, 0)$'s overflow, and therefore the value of x_3, is recovered exactly. One can view our analysis later in this chapter as showing that if we choose the graphs and counter sizes properly, then it is extremely likely that the reasoning above does give matching upper and lower bounds very quickly, i.e., long before the breadth-first search tree reaches a size comparable to N, and therefore we can use this kind of reasoning to construct an efficient local decoding algorithm.

3.4 WRITE Algorithm

The pseudocode of Algorithm 1 gives a more formal description of the WRITE algorithm illustrated by example in Section 3.3. This algorithm is essentially identical to the algorithm presented in [73].

Initialization - Let $y(v, \ell)$ be the amount that needs to be added to $c(v, \ell)$, with similar notation for the counters $(v, \ell)'$. We set $y(i, 0) = c$. We implicitly initialize $y(v, \ell)$ to 0 for all other counters.

$S_0 = S'_0 = \{i\}$. $S_\ell = S'_\ell = \emptyset$ for $\ell = 2, 3, \ldots, L$.
for $\ell = 0$ to $L - 1$ **do**
  for each $v \in S_\ell$ **do**
    overflow$(v) \leftarrow \lfloor \frac{c(v,\ell)+y(v,\ell)}{2^{b_\ell}} \rfloor$.
    $c(v, \ell) \leftarrow c(v, \ell) + y(v, \ell) \bmod 2^{b_\ell}$.
    if overflow$(v) \neq 0$ and $\ell < L$ **then**
      $S_{\ell+1} \leftarrow S_{\ell+1} \cup \{u : ((v, \ell), (u, \ell + 1)) \in E_{\ell+1}\}$
      for each $u \in \{u : ((v, \ell), (u, \ell + 1)) \in E_{\ell+1}\}$ **do**
        $y(u, \ell + 1) \leftarrow y(u, \ell + 1)+$ overflow(v)
      end for
    end if
  end for
  for each $v \in S'_\ell$ **do**
    overflow$(v) \leftarrow \lfloor \frac{c(v,\ell)'+y(v,\ell)'}{2^{b'_\ell}} \rfloor$.
    $c(v, \ell)' \leftarrow c(v, \ell)' + y(v, \ell)' \bmod 2^{b'_\ell}$.
    if overflow$(v) \neq 0$ and $\ell < L$ **then**
      $S'_{\ell+1} \leftarrow S'_{\ell+1} \cup \{u : ((v, \ell)', (u, \ell + 1)') \in E'_{\ell+1}\}$
      for each $u \in \{u : ((v, \ell)', (u, \ell + 1)') \in E'_{\ell+1}\}$ **do**
        $y(u, \ell + 1)' \leftarrow y(u, \ell + 1)'+$ overflow(v)
      end for
    end if
  end for
end for

Algorithm 1: WRITE Algorithm

Lemma 3.4.1. *WRITE has worst-case running time at most*

$$2^{O\left(1+\left(\log\log\left(\frac{M}{2K^2A^2}\right)-\log\log(A)\right)^+\right)} = \text{poly}\left(\max\left(\log\left(\frac{M}{2A^2}\right), 1\right)\right),$$

and uses at most

$$2^{O\left(1+\left(\log\log\left(\frac{M}{2K^2A^2}\right)-\log\log(A)\right)^+\right)} = \text{poly}\left(\max\left(\log\left(\frac{M}{A^2}\right), 1\right)\right)$$

space.

3.5 READ Algorithm

In what follows, we provide a detailed description of READ, the decoding algorithm that recovers the value x_i for any given i. READ has two parts, which we call WHP and BAILOUT. WHP decodes a single input with high probability, and is based on the message-passing algorithm proposed by [73]. However, to achieve the local decoding property, we carefully design an incremental version of the algorithm, so that WHP has a low expected running time. In the rare event that WHP fails to recover x_i, we run BAILOUT, which is slower but never fails.

3.5.1 WHP Algorithm

The idea behind WHP is that to recover x_i, we only need to use local information, i.e., instead of looking at all counters, we only use the values of counters that are close (in terms of graph distance) to counter i in layer 0. As illustrated in Section 3.3, local information can be used to compute upper and lower bounds on x_i. Following this insight, we propose an algorithm capable of computing better upper and lower bounds as more local information (with respect to the graphical code) is utilized. This algorithm can then be run incrementally, using more and more information until it manages to decode (i.e., obtains matching upper and lower bounds).

A formal description of WHP is as follows. First, a subroutine is stated that can recover the overflow of a single counter in an arbitrary layer of the data structure, given additional information not directly stored in our structure, namely the values in the next layer assuming the next layer has infinite sized counters. This subroutine is used by the 'WHP inner loop' that utilizes more and more local information incre-

mentally as discussed above. The values that would have been stored if the counter size was infinite are not directly available for any layer $\ell < L$, but for the last layer L and our choice of parameters these values are precisely the values already stored by our data structure, i.e., the last layer never overflows as long as $\mathbf{x} \in \mathcal{S}(A, M, N)$. Therefore, using this subroutine, one can recursively obtain the necessary values for layers $L-1, \ldots, 0$. The 'WHP outer loop' achieves this by selecting appropriate parameters in the 'WHP inner loop'.

WHP is based on the decoding algorithm introduced by Lu et. al. [73]. Our contribution is an analysis of a modified version of their algorithm. Specifically, the subroutine given in Section 3.5.1 can be viewed as a modified version of the algorithm from [73] that is designed to run incrementally and locally, i.e., since we are only interested in a single x_i, our algorithm only needs to examine a small subset of the counters stored in the data structure. Our analysis shows that this local version of the algorithm can be much more efficient than the original algorithm, i.e., most of the time WHP can determine the value of x_i by examining far fewer than N counters.

WHP Subroutine

For any counter (u, ℓ), the goal is to recover the value that it would have stored if $b_\ell = \infty$. To this end, suppose the values that would have been stored in the counters in $V_{\ell+1}$ if $b_{\ell+1} = \infty$ are already known. As explained before, this will be satisfied recursively, starting with $\ell + 1 = L$. The subroutine starts by computing the breadth-first search neighborhood of (u, ℓ) in $G_{\ell+1}$ of depth t, for an even t. Hence the counters at depth t of this breadth-first search, or computation, tree belong to V_ℓ (the root is at depth 0). This neighborhood is indeed a tree provided that G_ℓ does not contain a cycle of length $\leq 2t$. We will always select t so that this holds. This computation tree contains all the counters that are used by the subroutine to determine the value that (u, ℓ) would have had if $b_\ell = \infty$.

Once the computation tree has been computed, the overflow of the root (u, ℓ) can be determined using the upper bound-lower bound method described by example in Section 3.3. In the following more formal description, we imagine that the edges of

the computation tree are oriented to point towards the root, so that it makes sense to talk about incoming and outgoing edges. Let $C(v, \ell)$ denote the set of all children of counter (v, ℓ), and let $P(v, \ell)$ denote the parent of counter (v, ℓ), with similar notation for counters in $V_{\ell+1}$. Also, for a counter $(v, \ell+1) \in V_{\ell+1}$, let c_v denote the value that this counter would have had if $b_{\ell+1} = \infty$, which we recall was assumed to be available. (Note: for simplicity, we leave out the $\ell + 1$ in the definition of c_v because c_v is only defined for counters in $V_{\ell+1}$, hence the $\ell + 1$ is redundant.)

1: **Initialization** - for all counters (v, ℓ) at depth t, set the value of the outgoing edge so that $m_{(v,\ell)\to P(v,\ell)} = 0$. (Lower bound)
2: Next, we compute the value for edges leaving the counters at depth $t-1$, i.e., the counters just above the leaves. For counter $(v, \ell+1)$ at depth $t-1$, the outgoing value is computed as $m_{(v,\ell+1)\to P(v,\ell+1)} = c_v - \sum_{(w,\ell)\in C(v,\ell+1)} m_{(w,\ell)\to(v,\ell+1)}$. (Upper bound)
3: The value on an edge leaving a counter (v, ℓ) at depth $t-2$ is given by

$$m_{(v,\ell)\to P(v,\ell)} = \min_{(w,\ell+1)\in C(v,\ell)} m_{(w,\ell+1)\to(v,\ell)}.$$

(Upper bound)
4: The value on an edge leaving a counter $(v, \ell+1)$ at depth $t-3$ is computed the same way as in step 1, i.e., $m_{(v,\ell+1)\to P(v,\ell+1)} = c_v - \sum_{(w,\ell)\in C(v,\ell+1)} m_{(w,\ell)\to(v,\ell+1)}$. (Lower bound)
5: The value on an edge leaving a counter (v, ℓ) at depth $t-4$ is given by

$$m_{(v,\ell)\to P(v,\ell)} = \max\left(\max_{(w,\ell+1)\in C(v,\ell)} m_{(w,\ell+1)\to(v,\ell)}, 0\right).$$

(Lower bound)
6: We work our way up the tree by repeating steps 2-5, until we reach the root.
7: **Termination** - When the procedure finishes, we get a value $c_{u,t}$ for the outgoing value from the root counter (u, ℓ) for a depth t computation tree. To complete the subroutine, we repeat steps 1-6 with a computation tree of depth $t+2$, getting a value $c_{u,t+2}$. If $c_{u,t} = c_{u,t+2}$, i.e., we have matching upper and lower bounds, then the subroutine outputs $c_{u,t}$ as the value of (u, ℓ)'s overflow. If $c_{u,t} \neq c_{u,t+2}$, then the subroutine declares a failure.

Algorithm 2: WHP Subroutine

In the pseudocode of Algorithm 2, $m_{(v,\ell)\to(w,\ell+1)}$ denotes the number, or 'message', associated by the subroutine with the edge pointing from counter (v, ℓ) to counter $(w, \ell+1)$, with similar notation for edges pointing from a counter in $V_{\ell+1}$ to a counter

in V_ℓ. This number is either an upper bound or a lower bound on the overflow of whichever endpoint of the edge is in V_ℓ. To aid intuition, we indicate in each step of the pseudocode whether a lower bound or an upper bound is being computed. We emphasize that although we assumed that the input to the subroutine included values for all of the counters in $V_{\ell+1}$, it is clear that the subroutine above only requires values for the counters in $V_{\ell+1}$ that also belong to the appropriate computation tree.

We conclude this section by stating two properties of the subroutine above. The first is Lemma 2 from [73], which formalizes the intuition that the subroutine computes lower and upper bounds.

Lemma 3.5.1. *Consider running the WHP subroutine on a computation tree of depth t, where we assume that t is even. For a counter (v, ℓ) at depth $t' \equiv t \bmod 4$, the value on the outgoing edge from (v, ℓ) to $P(v, \ell)$ is a lower bound on (v, ℓ)'s overflow. For a counter (v, ℓ) at depth $t' \equiv t + 2 \bmod 4$, the value on the outgoing edge from (v, ℓ) to $P(v, \ell)$ is an upper bound on (v, ℓ)'s overflow.*

As a corollary, we get the following lemma.

Lemma 3.5.2. *Assume that the WHP subroutine does not fail. Then, the value returned by the subroutine is the correct value of (u, ℓ)'s overflow.*

WHP Inner Loop

Using repeated calls to the subroutine described above, the WHP inner loop determines the value of x_i.

The WHP inner loop has a parameter for each layer, which we denote by $t_\ell^{(1)}$. The algorithm proceeds as follows. To recover x_i, we start by building the computation tree of depth $t_1^{(1)} + 2$ in G_1, i.e., the computation tree formed by a breadth-first search of the graph G_1 rooted at $(i, 0)$. Next, for each counter in this computation tree that also belongs to V_1, i.e., the counters at odd depth in the computation tree, we form a new computation tree of depth $t_2^{(1)} + 2$ in G_2. This gives us a set of computation trees in G_2. For each computation tree, the counters at odd depth belong to V_3, so we repeat the process and build a computation tree of depth $t_3^{(1)} + 2$ in G_3 for each of

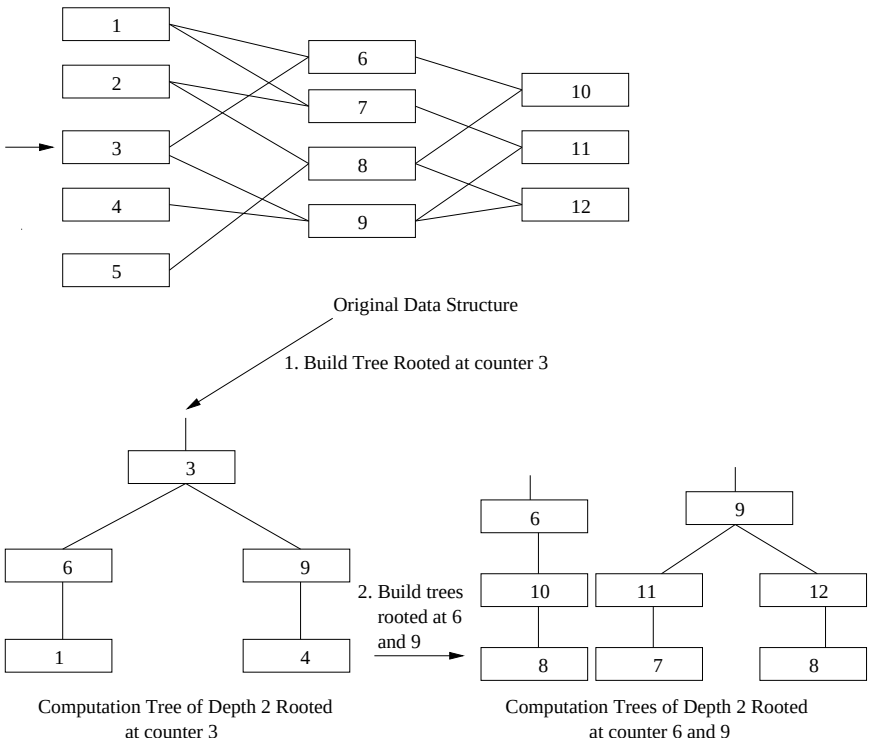

Figure 3-4: Example of forming multiple computation trees.

these counters. We repeat this process until we reach the last layer. Figure 3-4 gives an example of this process.

Now, we work backwards from the last layer (layer L). Each computation tree in G_L has depth $t_L^{(1)} + 2$. We run the subroutine described above on each of these computation trees. When we run the subroutine on a particular computation tree, we set the input c_v for each counter (v, L) in V_L that also belongs to the computation tree to be $c(v, L)$. Intuitively, this means that we are assuming that no counter in V_L overflows. As mentioned earlier, for any $\mathbf{x} \in \mathcal{S}(A, M, N)$, this will be the case.

Assuming that none of the subroutine calls fail, we have computed the overflow for all the counters that appear in computation trees for G_{L-1}. Let $(v, L-1)$ be a counter in V_{L-1} that also belongs to a computation tree in G_{L-1}. We have computed $(v, L-1)$'s overflow, which we denote by overflow$(v, L-1)$. To compute the value that $(v, L-1)$ would have stored if $b_{L-1} = \infty$, we simply compute $c_v = c(v, L-1) + 2^{b_{L-1}}$ overflow$(v, L-1)$. Once we have computed these values, we can run the subroutine on the computation trees in G_{L-1}. Then, we compute the appropriate values for

counters in V_{L-2} using the formula $c_v = c(v, L-2) + 2^{b_{L-2}}\text{overflow}(v, L-2)$, and so on until either the subroutine fails or we successfully run the subroutine on all the computation trees. Assuming that the subroutine finishes successfully on all of the computation trees, the final subroutine call gives us the overflow of $(i, 0)$. Then, $x_i = c(i, 0) + 2^{b_0}\text{overflow}(i, 0)$. Thus, if none of the subroutine calls fail, we successfully recover x_i.

WHP Outer Loop

The WHP outer loop repeats the WHP inner loop several times. Specifically, the WHP outer loop starts by running the WHP inner loop. If none of the subroutine calls fail, then we recover x_i and we are done. Otherwise, we run the WHP inner loop again, but with a new set of parameters $t_1^{(2)}, t_2^{(2)}, \ldots, t_L^{(2)}$. If some subroutine call fails, we run the WHP inner loop again with a new set of parameters $t_1^{(3)}, t_2^{(3)}, \ldots, t_L^{(3)}$, and so on up to some maximum number M of repetitions. If after M repetitions we still fail to recover x_i, we declare a failure. For our purposes, we set the parameters as follows. To specify $t_\ell^{(p)}$, we introduce the auxiliary numbers $\delta^{(p)}$ and $n_\ell^{(p)}$. For $1 \le p \le \log(N)/(\log(L))^2, 1 \le \ell \le L$, we define $\delta^{(p)}, n_\ell^{(p)}$, and $t_\ell^{(p)}$ as follows:

$$\begin{aligned}
\delta^{(p)} &= e^{-L^{p+1}} \\
n_\ell^{(p)} &= e^{\ell\left(c_1 + \frac{\log(c)}{\log(d)} \log\log\left(\frac{L}{\delta^{(p)}}\right)\right)} \\
t_\ell^{(p)} &= \left\lceil t^* + \frac{1}{\log d} \log\left(\frac{1}{a}\log\left(\frac{n_\ell^{(p)} L}{\delta^{(p)}}\right)\right)\right\rceil,
\end{aligned}$$

where a, c, c_1, d, and t^* are some fixed constants. Finally, we set $M = \log(N)/(\log(L)^2)$. This completes the specification of all the parameters. The following lemma summarizes the performance of WHP.

Lemma 3.5.3. *WHP has expected running time*

$$2^{O\left(\left(1+\left(\log\log\left(\frac{M}{2K^2A^2}\right)-\log\log(A)\right)^+\right)\log\left(2+\left(\log\log\left(\frac{M}{2K^2A^2}\right)-\log\log(A)\right)^+\right)\right)} = \text{quasipoly}\left(\max\left(\log\left(\frac{M}{A^2}\right),1\right)\right),$$

and uses

$$2^{O\left(\left(1+\left(\log\log\left(\frac{M}{2K^2A^2}\right)-\log\log(A)\right)^+\right)\log\left(2+\left(\log\log\left(\frac{M}{2K^2A^2}\right)-\log\log(A)\right)^+\right)\right)} = \text{quasipoly}\left(\max\left(\log\left(\frac{M}{A^2}\right),1\right)\right)$$

space. The probability that WHP fails is $o\left(1/N^4\right)$.

3.5.2 BAILOUT Algorithm

BAILOUT is an algorithm that recovers the source $\mathbf{x}$ with 0 probability of failure. It is similar in spirit to WHP, and only uses the values of counters in $V_0, V_1', \ldots, V_L'$. Like WHP, BAILOUT recovers the source starting with the last layer and working back to the first layer. The main difference is that because the entire source is recovered, BAILOUT operates on the entire graph instead of just a local neighborhood. As with WHP, we describe BAILOUT by first describing a subroutine, and then showing how the subroutine can be applied recursively to recover the input.

BAILOUT Subroutine

In this section, we give an algorithm that solves the following problem. We are given the values that all of the counters in V_ℓ' would have stored if $b_\ell' = \infty$. The goal is to recover the values that all counters in $V_{\ell-1}'$ would have stored if $b_{\ell-1}' = \infty$. The algorithm we use to solve this problem is essentially the same as the upper bound-lower bound method described in Section 3.5.1, except that we modify the algorithm to run on the entire graph instead of just a computation tree. Algorithm 3 gives a

pseudocode description of the algorithm.

1: **Initialization** - For all edges $((u,\ell-1)',(v,\ell)') \in E'_\ell$, set $m^{(0)}_{(u,\ell-1)'\to(v,\ell)'} = 0$. (Lower bound)

2: Next, we compute the values in the other direction. Specifically, for all edges $((u,\ell-1)',(v,\ell)') \in E'_\ell$, set

$$m^{(0)}_{(v,\ell)'\to(u,\ell-1)'} = c_v - \sum_{(w,\ell-1)'\in N(v,\ell)'-(u,\ell-1)'} m^{(0)}_{(w,\ell-1)'\to(v,\ell)'}.$$

(Upper bound)

3: We update the values from $V'_{\ell-1}$ to V'_ℓ using the formula

$$m^{(1)}_{(u,\ell-1)'\to(v,\ell)'} = \min_{(w,\ell)'\in N(u,\ell-1)'} m^{(0)}_{(w,\ell)'\to(u,\ell-1)'}.$$

(Upper bound)

4: We update the values from V'_ℓ to $V'_{\ell-1}$ using the formula

$$m^{(1)}_{(v,\ell)'\to(u,\ell-1)'} = c_v - \sum_{(w,\ell-1)'\in N(v,\ell)'-(u,\ell-1)'} m^{(1)}_{(w,\ell-1)'\to(v,\ell)'}.$$

(Lower bound)

5: We update the values from $V'_{\ell-1}$ to V'_ℓ using the formula

$$m^{(2)}_{(u,\ell-1)'\to(v,\ell)'} = \max\left(0, \max_{(w,\ell)'\in N(u,\ell-1)'} m^{(1)}_{(w,\ell)'\to(u,\ell-1)'}\right).$$

(Lower bound)

6: **Termination** - Repeat steps 2-5 until the values converge. By convergence, we mean that all the values computed at step 5 stay the same if we repeat steps 2-5 again. When the messages have converged, the outgoing messages leaving any counter $(u,\ell-1)' \in V'_{\ell-1}$ are all the same, and the value of this message is the value that we return as overflow$(u,\ell-1)'$.

Algorithm 3: BAILOUT Subroutine

The algorithm maintains two numbers for each edge $((u,\ell-1)',(v,\ell)') \in E'_\ell$, corresponding to a number for each direction. Because the algorithm is iterative, we denote these numbers by $m^{(t)}_{(u,\ell-1)'\to(v,\ell)'}$ and $m^{(t)}_{(v,\ell)'\to(u,\ell-1)'}$, where t denotes the iteration number. Also, for an edge $((u,\ell-1)',(v,\ell)')$, we use $N(u,\ell-1)'$ to denote the set of vertices adjacent to $(u,\ell-1)'$, and $N(u,\ell-1)'-(v,\ell)'$ to denote the set of vertices other than $(v,\ell)'$ that are adjacent to $(u,\ell-1)'$. As in Section 3.5.1, to aid intuition we indicate whether the numbers being computed represent upper or lower

bounds.

BAILOUT Loop

Now, we show how the BAILOUT subroutine can be used to determine the value of x_i. As with WHP, we work backwards from layer L. Specifically, we run the BAILOUT subroutine to recover the overflows of all counters in V'_{L-1}. When we run the subroutine on V'_{L-1}, we set the input c_v for each counter $(v, L)'$ to be $c(v, L)'$. Once the subroutine finishes, we have computed the overflow for all the counters in V'_{L-1}. To compute the value that $(v, L-1)'$ would have stored if $b'_{L-1} = \infty$, we compute $c_v = c(v, L-1)' + 2^{b'_{L-1}}$ overflow$(v, L-1)$. Once we have computed these values, we can run the BAILOUT subroutine on V'_{L-2}, and so on until we have computed the overflow for all counters in V_0. Then, $x_i = c(i, 0)' + 2^{b'_0}$ overflow$(i, 0)$.

The following lemma states that the BAILOUT algorithm works as desired.

Lemma 3.5.4. *The BAILOUT algorithm recovers the overflow for all counters in V_0 correctly.*

For technical reasons, we cannot use a naive implementation of the BAILOUT subroutine. Specifically, in the first layer, the numbers stored on the edges can be at least as large as $\frac{M}{K^2A}$. Thus, a naive implementation that stores the numbers associated with each edge using $\log(\frac{M}{K^2A})$ space will use at least $3N \log(\frac{M}{K^2A})$ space for just the first layer, while we want the workspace used by the algorithm to be much less than $N \log A$. In Appendix A.1, we show how to modify the BAILOUT algorithm so that it uses a small amount of space, albeit at the cost of an increased running time.

Lemma 3.5.5. *The modified BAILOUT algorithm has running time $O(N^4)$ and uses at most $4N + O(\varepsilon \log(1/\varepsilon)N)$ space.*

Before moving on, we note that the BAILOUT subroutine can be interpreted as an algorithm for compressed sensing. As alluded to in Chapter 1 and described formally in Chapter 4, one of the basic problems in compressed sensing is recovering a

signal with very few nonzero entries from a small set of linear measurements. Lemma 3.8.7 can be viewed as saying that the BAILOUT subroutine is capable of recovering nonnegative signals with at most αn nonzero components when the measurement matrix is a suitably good expander graph. Note that there is nothing special about the choice of sparsity αn—for any k and n, we noted in Chapter 2 that suitable expanders with $O(k \log(n/k))$ right vertices exist, so the BAILOUT subroutine can be used to recover nonnegative signals with at most k nonzero components from $O(k \log(n/k))$ measurements. In Chapter 4 we explore the connection between the BAILOUT subroutine and compressed sensing further, and show that the BAILOUT subroutine can provide an ℓ_1/ℓ_1 reconstruction guarantee.

3.6 Space

We conclude this section with the following two lemmas, which prove that for large A, the data structure uses close to the optimal amount of space.

Lemma 3.6.1. *With the choice of parameters given in Section 3.3.2, the generic data structure uses*

$$\left(1 + 2\varepsilon + \frac{3\varepsilon}{2^C}\right) N \log(A) + O\left(N \log\left(\frac{1}{\varepsilon}\right)\right)$$

space.

Lemma 3.6.2. *Any data structure that can recover the input with probability* 1 *must use space at least* $N \log A$. [4] *Also, there exists a data structure using space at most*

$$\left\lceil N\left(\log(A) + \frac{1+A}{A\ln(2)}\right)\right\rceil$$

which is capable of recovering the input with probability 1*, albeit with a computationally very expensive recovery algorithm.*

[4] If we allow a small probability of failure, then the lower bound is reduced, but it can be verified that the loss is negligible.

3.7 Main Result

Finally, we can state the main result of this chapter, Theorem 3.7.1.

Theorem 3.7.1. *With the choice of parameters given in Section 3.3.2, the generic data structure has the following properties:*

1. *Write—There exists an algorithm $WRITE(i, c)$ that increments x_i by (a possibly negative) c with worst-case running time*

$$2^{O\left(1+\left(\log\log\left(\frac{M}{2K^2A^2}\right)-\log\log(A)\right)^+\right)} = \text{poly}\left(\max\left(\log\left(\frac{M}{A^2}\right), 1\right)\right).$$

2. *Read—The exists a (randomized) algorithm $READ(i)$ that returns x_i. The expected running time of $READ(i)$ is*

$$2^{O\left(\left(1+\left(\log\log\left(\frac{M}{2K^2A^2}\right)-\log\log(A)\right)^+\right)\log\left(2+\left(\log\log\left(\frac{M}{2K^2A^2}\right)-\log\log(A)\right)^+\right)\right)} = \text{quasipoly}\left(\max\left(\log\left(\frac{M}{A^2}\right), 1\right)\right).$$

3. *Space—The data structure uses*

$$\left(1 + 2\varepsilon + \frac{3\varepsilon}{2^C}\right) N \log(A) + O\left(N \log\left(\frac{1}{\varepsilon}\right)\right)$$

space.

Proof. Immediate from Lemmas 3.4.1, 3.5.3, 3.5.5, and 3.6.1. □

Remarks:

1. Theorem 3.7.1 shows that our data structure is optimal in terms of space utilization, as Lemma 3.6.1 shows that the space required is at least $N \log A$; encoding/decoding can take poly($\log N$) time at worst, e.g., when $A = O(1)$ and $M = N^\alpha$ for some $\alpha \in (0, 1)$, but in more balanced situations, e.g. when $M = N^\alpha$ and $A = N^{\alpha/2}$, the encoding/decoding time can be $O(1)$.

2. For the running time bounds in parts $1, 2$, and 3 of Theorem 3.7.1, the constant hidden inside the big-O may depend on C and ε. However, the constant hidden inside the big-O for the space bound in part 4 of Theorem 3.7.1, does not depend on C or ε.

3. For READ, we emphasize that the probabilities are over the random choices of the data structure. We are not assuming any probability distribution on the input, and the expected running time bound holds even if the input is generated adversarially.

4. Lemma 3.6.2 shows that any data structure capable of recovering the input must use at least $N \log A$ space. Thus, in the limit of large N and A, our data structure uses essentially the optimum amount of space. Looking at the expression given in part 3 of Theorem 3.7.1, we see that in the limit of large A, C should be chosen as large as possible, and ε should be proportional to $1/\log A$, in order to minimize the space. Of course, choosing small values of ε and large values of C increases the complexity of READ and WRITE exponentially, so in practice one might be forced to use more space, i.e., choose a larger value of ε and/or a smaller value of C, so that READ and WRITE have reasonably low complexity.

5. We emphasize that when we calculate the space used by our data structure, we are not just counting the space used by counters. The space bound in Theorem 3.7.1 includes the total space needed to implement the structure, i.e., the space bound includes the space needed to store the graphs and the work space used by READ and WRITE.

3.8 Analysis

To prove Theorem 3.7.1, we must analyze the space used by our data structure, and the complexity of the READ and WRITE algorithms. First, in Section 3.8.1 we prove Lemmas 3.3.1 and 3.3.2, showing that the graphs used in our data structure have all

the desired properties. Next, we analyze the space used by our data structure in Section 3.8.2. Section 3.8.3 analyzes the complexity of WRITE by proving Lemma 3.4.1, and Section 3.8.4 analyzes the complexity of READ by proving Lemmas 3.5.3 and 3.5.4. Because the analysis of READ is the most involved part of the proof of Theorem 3.7.1, we relegate some of the technical details to Appendices A.1 and A.2. Note that in all proofs, when measuring complexity, we make an assumption similar to the random access memory (RAM) model of computation. Specifically, we assume that basic arithmetic operations on $O(\log A + \log N)$-bit integers take $O(1)$ time.

3.8.1 Basic Properties of our Graphs

In this section, we prove Lemma 3.3.1 and Lemma 3.3.2.

Proof of Lemma 3.3.1

First, we explain how to construct the base graphs H and H_1. Then, we prove Lemma 3.3.1.

To construct H, we start with the graph produced by the LPS construction with $p = 5$ and some q. This gives us a 6-regular graph. As noted in Chapter 2, this graph is actually bipartite. The only reason that we cannot use the graph produced by the LPS construction directly as H is that the 2 partite sets have the same size. To get a $(3, 6)$-regular graph from the 6-regular bipartite Cayley graph, we need to double the size of the left partite set. To do this, we split each left vertex, i.e., each vertex corresponding to an element in $PSL(2, q)$, into two vertices, each with degree 3. For example, let v be a vertex in $PSL(2, q)$. It is connected to $s_1v, s_2v, \ldots, s_6v$. We split v into v_1 and v_2, where v_1 is connected to s_1v, s_2v, and s_3v, while v_2 is connected to s_4v, s_5v, and s_6v. Repeating this for all vertices in $PSL(2, q)$ gives us a $(3, 6)$-regular graph. This is the graph that use as the base graph H. It is obvious that this splitting process does not reduce the girth, so H also has girth $\Omega(\log_p(q))$.

To construct H_1, we use a similar procedure, but we start with the graph produced by the LPS construction with $p = K - 1$ and some q. This is a K-regular bipartite

graph. Instead of splitting each vertex in $PSL(2,q)$ into two copies, we split each vertex into $K/3$ copies. Each copy has degree 3 as above, i.e., we split the generating set S into $K/3$ sets of size 3 and assign neighbors accordingly. This gives us a $(3,K)$-regular graph with large girth, and we use this graph as H_1.

Note that the above constructions of H and H_1 assume that $K-1$ is a prime number congruent to 5 mod 12, and that $\sqrt{2^{-(L-2)}\varepsilon N}$ and $6\sqrt{N}/K$ are both of the form q^3-q for some prime $q \equiv 1 \bmod 4$ such that either $\left(\frac{5}{q}\right) = -1$ or $\left(\frac{K-1}{q}\right) = -1$ so that the LPS construction gives a graph with the correct number of vertices. These assumptions can easily be removed. Also, we observe that the $(3,6)$-regular graph H constructed above is essentially the same as the graph constructed in [104]. The $(3,K)$-regular graph H_1 can be viewed as a natural generalization of the construction in [104] to higher degrees.

Proof of Lemma 3.3.1. The base graph H can be stored explicitly in adjacency list form using $o(N)$ space. The two permutations π_R and π_C and their inverses can be stored using $o(N)$ space. With this information, it is obvious that we can compute the neighbors of any vertex using $O(1)$ memory lookups and computations involving $O(\log N)$-bit numbers.

Regarding the girth, it is clear that the girth of G_ℓ is at least the girth of H. From Lemma 2.3.1, we know that the girth of H is $\Omega(\log N)$. □

The reader may have noticed that we do not actually have to store H explicitly. The description of the LPS construction given in Section 2.3.1 shows that the neighbors of a vertex in H are given by an explicit formula, and hence we don't need to store the neighbors in an adjacency list. However, this savings does not affect the asymptotic space since the majority of the space used by our data structure is in the counters.

Proof of Lemma 3.3.2

We use the zigzag product, as described in Section 2.3.3, to construct the expander graphs needed in our analysis of the BAILOUT subroutine. As per Theorem 7.1

of [19], we take Z_1 to be the rotation map of the Ramanujan graph produced by the LPS construction. This is a permutation conductor, and from the description of the LPS construction given in Section 2.3.1, it is clear that the edge function for this permutation conductor can be computed in $O(1)$ time. We take Z_2 to be an optimal buffer conductor of constant size, and we take Z_3 to be an optimal lossless conductor of constant size. The parameters of such conductors are described by Lemmas 4.2 and 4.3 of [19]. We choose Z_2 and Z_3 to be both left- and right-regular—as noted in Section 2.3.3, this does not significantly affect the achievable performance.

Proof of Lemma 3.3.2. We store the small graphs Z_2 and Z_3 explicitly in adjacency list form, which requires $O(1)$ space. As noted above, we can compute the neighbors of a vertex in the graph produced by the LPS construction without storing the graph. Therefore, it follows directly from the definition of the zigzag product (as given in Section 2.3.3) that we can compute the neighbors in the zigzag product in $O(1)$ time. In a little more detail, from the definition of the edge function, it is clear that given any vertex on the left, the neighbors on the right can be computed in constant time, because the edge functions for Z_1, Z_2 and Z_3 can all be computed in constant time. Also, given a vertex on the right of Z, we can compute its neighbors on the left by inverting the edge function for Z. Clearly the edge functions for Z_2 and Z_3 can be inverted in constant time, and from the description of the LPS construction given in Section 2.3.1, it is clear that the edge function for Z_1 can also be inverted in constant time. Therefore, we can invert the edge function for Z in constant time as well. Finally, it is clear that Z_1 is left and right regular, and it is easily verified that if the input graphs are left and right regular, then the output graph Z produced by the zigzag product is also left and right regular. The expansion property is a direct consequence of Lemma 2.3.3. □

3.8.2 Analysis of Space Requirements

Proof of Lemma 3.6.1. Since the workspace used by the READ and WRITE algorithms is bounded by Lemmas 3.4.1, 3.5.3, and 3.5.5, we can complete the proof of

Lemma 3.6.1 by adding up the space used by the counters and the graphs. Lemmas 3.3.1 and 3.3.2 show that the space used by the graphs is $o(N)$. Thus, we just have to count the space used by counters. In the following analysis, for simplicity we ignore ceilings. Because there are only $O(N)$ counters in the entire date structure, this means that we only have to add $O(N)$ to the bound obtained by ignoring ceilings.

In V_0, each counter uses $\log(K^2A)$ bits each. The counters in V_1 use $\log(12AK)$ bits each. Thus, the space used by the first layer is $N\log(K^2A) + \varepsilon N\log(12AK)$. V_2 through V_{L-1} use 3 bits per counter, and since the number of counters decreases geometrically, this give us a total space of $O(N)$. V_L uses $\log(1 + \frac{M}{2K^2A^2})$ bits per counter, and there are $2^{-(L-1)}\varepsilon N$ counters in this layer. Thus, V_L uses at most $2^{-(C-1)}\varepsilon N\log(A) + O(N)$ space. Similarly, V_1' contributes $\varepsilon N\log(Ar)$ space, $V_2', \ldots, V_{L-1}'$ contribute a total of $O(\varepsilon N\log(r))$ space, and V_L' contributes at most $\varepsilon 2^{-C}N\log(A) + O(N)$ space. Adding together all these contributions, we see that the total space used by counters is $\left(1 + 2\varepsilon + 3\varepsilon 2^{-C}\right) N\log(A) + O\left(N\log(K)\right)$. □

Proof of Lemma 3.6.2. To prove the first part of the lemma, observe that the number of possible inputs satisfying the constraints is at least A^N. To see this, note that if each x_i is chosen from the set $\{0, 1, \ldots, A-1\}$, then the input does satisfy all the constraints because $\sum x_i$ is clearly less than AN, and the maximum is less than M. Thus, there are at least A^N inputs satisfying the constraints, which means that a lower bound on the space used by any data structure is $N\log A$.

For the second part of the lemma, note that an optimal data structure only requires space at most $\lceil\log(\text{number of inputs})\rceil$, because with this much space every possible input can be encoded to a different state of the data structure, and hence the mapping from inputs to states is invertible. To get an upper bound on the number of inputs satisfying the constraints, we neglect the constraint $\max x_i < M$. Then, the number of possible inputs is given by the binomial coefficient $\binom{(A+1)N-1}{N}$. It is well-known

(cf. [28]) that

$$\begin{aligned}\log\left(\binom{(A+1)N-1}{N}\right) &< N\left((A+1)\log\left(A+1\right)\right)\\ &\quad -NA\log\left(A\right),\end{aligned}$$

and applying the inequality $\ln(1+x) < x$, we see that

$$\begin{aligned}\log\left(\binom{(A+1)N-1}{N}\right) &< N\left((A+1)\left(\log\left(A\right)+\frac{1}{A}\right)\right)-NA\log\left(A\right)\\ &= N\left(\log\left(A\right)+\frac{1+A}{A\ln\left(2\right)}\right).\end{aligned}$$

□

3.8.3 Analysis of WRITE

In this section, we analyze the WRITE algorithm.

Proof of Lemma 3.4.1. We can analyze the complexity of WRITE by counting the number of counters that the algorithm looks at in the course of processing a query. Each counter we examine requires us to perform a constant number of basic arithmetic operations, and by assumption these operations take constant time. So, within a constant factor, the complexity is upper bounded by the number of counters that the algorithm examines. To upper bound the number of examined counters, note that in the worst case, every counter we modify (except in V_L and V'_L) overflows. If this happens, then we must update $3^{\ell-1}K$ counters in V_ℓ and l^ℓ counters in V'_ℓ. Summing up the number of counters examined in all the layers, we see that the complexity of WRITE is $O(l^L)$. Since $L = (\log\log(\frac{M}{2K^2A^2}) - \log\log(A))^+ + C$, the proof is complete. □

3.8.4 Analysis of READ

In this section we prove Lemmas 3.5.3 and 3.5.4.

WHP Analysis

To prove Lemma 3.5.3, we must prove that the algorithm actually works, i.e., that x_i is recovered with high probability, and that the algorithm has a low expected running time. We begin with proofs of some preliminary results needed to establish Lemma 3.5.3. First, we summarize the discussion in Sections 3.5.1 and 3.5.1 with the following lemma.

Lemma 3.8.1. *Assume that WHP does not fail. Then, the value returned by WHP is the correct value of x_i.*

Proof. Lemma A.2.1 (proved in Section A.2.1) shows that the counters in V_L never overflow. Thus, when the WHP subroutine is first called in the last layer, the c_v values are correct. From Lemma 3.5.2, it follows that this property is maintained as an invariant, i.e., provided that the subroutine calls are successful, the c_v values are always correct as we move back towards the first layer. This implies that x_i is recovered correctly. □

Lemma 3.8.1 says that WHP never makes an undetected error. Thus, we can just focus on the probability of failure.

Lemma 3.8.2. *Consider the p^{th} repetition of the WHP inner loop, and assume that the WHP subroutine has run successfully on all computation trees in $G_{\ell+2}, \ldots, G_L$. Then, the probability that the WHP subroutine fails for any computation tree in $G_{\ell+1}$ is at most $\delta^{(p)}/L$.*

The proof of Lemma 3.8.2 is more technical than the other proofs in this chapter, so we relegate it to Appendix A.2. Lemma 3.8.2 allows us to bound the probability that WHP fails.

Lemma 3.8.3. *WHP fails with probability at most $\delta^{(M)}$.*

Proof. Let A_ℓ^p be the event that any subroutine call fails in layer ℓ during the p^{th} repetition. Lemma 3.8.2 states that $P(A_\ell^p) \leq \delta^{(p)}/L$. Now, the probability that the p^{th} repetition fails can be upper bounded using the union bound. Specifically,

$P(p^{th}$ repetition fails$) = P(\cup_\ell A_\ell^p) \leq \sum_\ell P(A_\ell^p) \leq L\delta^{(p)}/L = \delta^{(p)}$. The probability that WHP fails is the probability that the last repetition fails, which we have shown is at most $\delta^{(M)}$. □

Now, we bound the expected running time of WHP. First, we state another preliminary lemma.

Lemma 3.8.4. *The running time of the p^{th} repetition is $O(n_L^{(p)})$.*

Proof. The proof of Lemma 3.8.4 is essentially the same as the proof of Lemma 3.4.1. Specifically, WHP performs a constant number of basic arithmetic operations for each counter examined. Thus, we can bound the complexity of the p^{th} repetition by the number of counters that WHP examines during the p^{th} repetition. Let n_ℓ^* denote the number of counters in V_ℓ that are examined during the p^{th} repetition. The following formula gives a recursion for n_ℓ^*:

$$n_\ell^* \leq n_{\ell-1}^* c^{t_{\ell-1}^{(p)}},$$

where c is some constant related to the graph G_ℓ. Specifically, the size of the computation tree in each layer grows exponentially in the depth of the tree, and c represents how quickly the tree grows. For example, for a $(3, 6)$-regular graph, one valid choice of c is $c = 40$.

It is easily verified that the definition of $n_\ell^{(p)}$ given in Section 3.4 satisfies the above recursion, and therefore the number of counters in V_ℓ that are examined during the p^{th} repetition is at most $n_\ell^{(p)}$. Also, it clear from the definition that $n_\ell^{(p)}$ grows rapidly as a function of ℓ. In particular, $n_\ell^{(p)} \geq 2n_{\ell-1}^{(p)}$. Thus, the total number of counters examined is at most $\sum_\ell n_\ell^{(p)} = O(n_L^{(p)})$. □

Lemma 3.8.4 lets us bound the expected running time of WHP.

Lemma 3.8.5.

$$E[\textit{running time of WHP}] = 2^{O(L\log(L))}.$$

Proof. Let T_p be the running time of the p^{th} repetition, and let p_p be the probability that the first p repetitions fail. Then, the expected running time is given by

$$E[\text{running time}] \;=\; T_1 + T_2 p_1 + T_3 p_2 + \ldots + T_M p_{M-1}.$$

Now, Lemma 3.8.4 says that $T_p = O(n_L^{(p)})$. Also, from the proof of lemma 3.8.3, it is clear that $p_p \leq \delta^{(p)}$. Thus, the expected running time is upper bounded by

$$E[\text{running time}] = O\left(n_L^{(1)} + n_L^{(2)}\delta^{(1)} + \ldots + n_L^{(M)}\delta^{(M-1)}\right).$$

Substituting the definitions of $\delta^{(p)}$ and $n_\ell^{(p)}$ given in Section 3.5.1 into the above formula, we see that the expected running time is at most

$$e^{O(L\log(L))} + \sum_{i=2}^{M} e^{-L^i} e^{O((i+1)L\log(L))}.$$

Clearly all terms inside the sum are less than 1, so the running time is dominated by the $e^{O(L\log(L))}$ term. □

Proof of Lemma 3.5.3. This lemma is a corollary of Lemmas 3.8.3 and 3.8.5. The running time bound is obtained by substituting the value of L into the expression given in Lemma 3.8.5. It is easily verified that $\delta^{(M)} = o(1/N^4)$. □

BAILOUT Analysis

Now, we prove Lemma 3.5.4. Lemma 3.5.4 follows easily from four preliminary lemmas. First, the following lemma is the analog of Lemma 3.5.1 for the BAILOUT subroutine. The proof follows from the same arguments used to prove Lemma 3.5.1, so we omit the proof.

Lemma 3.8.6. *Let $(u, \ell-1)'$ be a counter in $V'_{\ell-1}$. When the BAILOUT subroutine is run on G'_ℓ, the numbers $m^{(2t)}_{(u,\ell-1)'\to(v,\ell)'}$ are monotonically nondecreasing lower bounds*

on the overflow of counter $(u, \ell-1)'$*, and the numbers* $m^{(2t+1)}_{(u,\ell-1)'\to(v,\ell)'}$ *are monotonically nonincreasing upper bounds on the overflow of counter* $(u, \ell-1)'$.

The next lemma shows that the BAILOUT subroutine recovers overflows properly in a single layer, provided that most of the overflows are 0.

Lemma 3.8.7. *Let* G *be a* (c, d)*-regular bipartite* $(2\alpha n, 1/2 + \varepsilon)$*-expander, for some* $\varepsilon > 0$*, with* n *vertices on the left and* m *vertices on the right. When the BAILOUT subroutine is run on* G*, the overflow is computed correctly for all vertices provided that the number of left vertices with nonzero overflow is at most* αn.

Proof. Define the set W_{2t} to be the set of vertices whose lower bounds are wrong after $2t$ iterations of the BAILOUT subroutine. Initially, $|W_0| \leq \alpha n$. To start our analysis, note that Lemma 3.8.6 implies that the BAILOUT subroutine must terminate, since the lower bounds are monotonically nondecreasing integers upper bounded by M. Also, if $t > s$, then $W_{2t} \subseteq W_{2s}$. Thus, to prove the lemma, it suffices to show that if $0 < |W_{2t}| \leq \alpha n$, then $|W_{2t+2}| < |W_{2t}|$, because this implies that for sufficiently large t, W_{2t} must be empty.

Now we prove that if $0 < |W_{2t}| \leq \alpha n$, then $|W_{2t+2}| < |W_{2t}|$. To prove this, we just have to show that some vertex in W_{2t} is not in W_{2t+2}. This is where we use the expansion of G. Let S be the set of vertices adjacent to some vertex in W_{2t}. Then, $|S| \leq c|W_{2t}|$. Now, consider the set T of vertices such that all of their neighbors are in S—T includes W_{2t} by definition. We know that $|T| < 2|W_{2t}|$, because $|W_{2t}| \leq \alpha n$ and G is a good expander. We claim that T contains all vertices whose upper bounds at time $2t + 1$ are wrong. To see this, note that an upper bound leaving a vertex u on the left at time $2t + 1$ is correct as long as one of its neighbors v is not in S. This is because if v is not in S, then all the incoming lower bounds to v are correct, so the upper bound computed by v for u's overflow must also be correct.

To complete the proof, note that $|T| < 2\alpha n$, so the expansion property guarantees that there is a vertex $w \in T$ with a unique neighbor, i.e., there exists a $w \in T$ and an $x \in S$ such that w is connected to x, and x is connected to no other vertex in T—for a more detailed derivation of this fact, see Section 4.5.1. By construction, this

w must belong to W_{2t}. Also, because x is a unique neighbor, x's neighbors other than w all lie outside of T, and thus have correct upper bounds. Therefore, w computes the correct lower at the next iteration. This proves that $w \in W_{2t}$ but $w \notin W_{2t+2}$, which proves the lemma. □

The next lemma proves that the initial c_v values in V'_L are correct.

Lemma 3.8.8. *No counter in V'_L can overflow.*

Proof. We prove that the largest possible value that needs to be stored in a counter in V'_L is at most $\frac{Mr}{K^2A^2}$. Because $2^{b'_L} > \frac{Mr}{K^2A^2}$, none of these counters overflow.

To verify the upper bound, consider the first layer. By assumption, the inputs have maximum value M. Therefore, the overflow is at most $\left\lfloor \frac{M}{K^2A} \right\rfloor$. Now, each counter in V'_1 computes the sum of the overflows associated with its r neighbors in V_0. Let s denote this sum. Then, the overflow is given by $\left\lfloor \frac{s}{Ar} \right\rfloor$. Thus, the overflow is bounded by $\frac{r(\frac{M}{K^2A})}{Ar} \leq \frac{M}{K^2A^2}$.

Now, each counter in V'_2 takes the sum of the overflows of its r neighbors in V'_1, and each of these overflows is at most $\frac{M}{K^2A^2}$. Thus, the overflow for counters in V'_2 is bounded by

$$\frac{r\frac{M}{K^4A^2}}{r} < \frac{M}{K^2A^2}.$$

Continuing down to the last layer, we see that in the last layer the counters in V'_{L-1} have an overflow of at most $\frac{M}{K^2A^2}$. Since each counter in V'_L is connected to r counters in V'_{L-1}, the largest possible value that a counter in V'_L needs to store is at most $\frac{Mr}{K^2A^2}$. □

The next lemma bounds the number of overflowing counters in $V'_0, \ldots, V'_{L-1}$.

Lemma 3.8.9. *The fraction of counters that overflow in V'_ℓ is at most $1/K^2$.*

Proof. Recall that the input must satisfy $\sum x_i < AN$. Thus, at most a $1/K^2$ fraction of the inputs have values that are greater than or equal to K^2A, which proves the claim for V_0. Because G'_1 is (l, r)-regular, the average value associated with counters on the right is at most r/K^2, and therefore at most $1/K^2$ of the counters in V'_1 are

greater than or equal to r, let alone Ar. Repeating this analysis for the future layers, we see that at every layer at most $1/K^2$ of the counters overflow. □

Proof of Lemma 3.5.4. The proof is straightforward given Lemmas 3.8.7, 3.8.8, and 3.8.9. As usual, we work backwards from the last layer. From Lemma 3.8.8, we know that the c_v values assigned to counters in V'_L are correct. From Lemma 3.8.9, we know that at most a $1/K^2$ of the counters overflow in V_{L-1}. Also, G'_L is an (l, r)-regular $(2/K^2, 2/3)$-expander, so Lemma 3.8.7 implies that the overflows are recovered properly. Therefore, the BAILOUT algorithm maintains as an invariant the property that the c_v values are correct at each layer. Thus, when the algorithm terminates, all inputs have been recovered correctly. □

3.9 Conclusion

Our main contribution is a locally encodable and decodable data structure for storing N nonnegative integers satisfying average and maximum constraints. We view this as a step towards developing general locally encodable and decodable source codes.

There are many directions for future work. One direction is to explore whether a variant of our data structure can be made practical. Our main result, Theorem 3.7.1, only provides asymptotic performance guarantees. For moderate values of N, e.g., $N < 1,000,000$, significant optimization issues remain to develop a practical data structure using space near the information-theoretic limit. For example, our choices for the number of layers, the number of counters in each layer, and the number of bits allocated to the counters in each layer are sufficient to prove asymptotic statements, but all of these parameters will have to be finely tuned to achieve good performance for moderate N. Also, it is quite likely that irregular graphs can achieve better performance than the regular graphs considered in this chapter. An important question to answer in this context is how to construct irregular graphs that are suitable for local encoding and decoding, i.e., irregular graphs with a compact description that still allows efficient computation of the neighbors of any given vertex. Although it may be difficult to prove rigorous results for irregular graphs, since constructing ir-

regular graphs with large girth is an open problem, it may be easy to show through simulation that any reasonably random construction of irregular graphs achieves good performance.

In a more theoretical direction, the data structure considered in this chapter shows that for certain settings of the parameters N, A, and M, i.e., for certain classes of sources, there exist source codes achieving rates close to the information-theoretic limit that still possess constant time or nearly constant time algorithms for local encoding and local decoding. This suggests the following two avenues for exploration. First, are there other classes of sources for which there exist (nearly) optimal locally encodable and locally decodable source codes? For example, do there exist locally encodable and decodable source codes for every discrete memoryless source? Before answering this question, one would need to determine what the correct notion of local encodability is for a probabilistic source. To see why this is not completely trivial, note that implicit in the concept of local encoding is the fact that the source gets updated over time, and for many natural update models, even if the initial source distribution is i.i.d. P for some distribution P, after several updates the source may no longer be distributed i.i.d. P. One possibility for addressing this issue is to generalize the average and maximum constraints considered in this chapter to arbitrary constraints on the type of the source, e.g., a Bernoulli source might be modelled as a binary source constrained so that after every update, the fraction of 1's in the source is always less than p.

Coming at the problem from the other direction, are there nontrivial lower bounds on the complexity of local encoding and local decoding in terms of the gap between the space achieved by a particular source code and the information-theoretic limit? In the context of succinct data structures, [95] shows how to construct a locally decodable version of arithmetic codes. For local encoding, one might hope to obtain lower bounds via a metric embedding argument. In more detail, one could try to show that any function mapping the source into a small number of bits must map two source realizations that are close to each other, e.g., that differ in only one component, to bit strings that are far apart, e.g., differ in half of the coordinates. If

one had such a lower bound, one could conclude that even if auxiliary computation is free, just updating the compressed version of the source is complex because half of the coordinates need to be changed. Unfortunately, the results of [84] shows that there exist very sparse nonlinear graph codes that approach the information-theoretic limit for Bernoulli sources. Thus, if there are any nontrivial lower bounds for locally encodable source codes for Bernoulli sources, then such bounds must rely on the computational model and not just a metric embedding argument. Thus, it seems likely that if there are nontrivial lower bounds on the complexity of locally encodable and decodable source codes, proving such bounds may have to rely on the fact that the code is simultaneously locally encodable and locally decodable, since the above results suggest that one can construct codes possessing either property individually.

Chapter 4

Sparse Graph Codes for Compressed Sensing

In this chapter, we explore the use of sparse graph codes for compressed sensing. In Section 4.1, we formally define the compressed sensing problem considered in this thesis and summarize some of the prior work. Then, we present some positive results regarding the use of sparse graph codes for compressed sensing. Specifically, we analyze a simple message-passing algorithm for the compressed sensing of nonnegative signals. We show that our message-passing algorithm can not only recover sparse signals, but can also provide a nontrivial ℓ_1/ℓ_1 guarantee—see Section 4.4 for a formal statement of our results.

In the second half of this chapter (Section 4.6), we prove that two classes of matrices—binary matrices (matrices whose entries are either 0 or 1) and very sparse matrices—do not behave well with respect to the restricted isometry property (RIP) for the ℓ_2 norm. We define the RIP and give some background on its importance in Section 4.6, but the reader familiar with compressed sensing knows that the RIP is the main tool used in known constructions of algorithms providing ℓ_2/ℓ_1 guarantees, and that ℓ_2/ℓ_1 guarantees are desirable because they are stronger than ℓ_1/ℓ_1 guarantees. Thus, our results suggest that either sparse graph codes are inherently not capable of providing ℓ_2/ℓ_1 guarantees, or a new proof technique is needed to prove such strong recovery guarantees for sparse graph codes.

4.1 Background on Compressed Sensing

The basic compressed sensing problem is to estimate a vector $x \in \mathbb{R}^n$ from a set of linear measurements $y = Ax$, where $y \in \mathbb{R}^m$ and A is a known m-by-n matrix. As briefly discussed in Chapter 1, the key idea in compressed sensing is that if the signal x is constrained to be sparse or approximately sparse, then it is possible to recover x even when $m << n$. More precisely, one of the basic results in compressed sensing is that there exist matrices A with only $m = O(k \log(n/k))$ rows such that for all k-sparse x, i.e., all x with at most k nonzero components, we can recover x exactly from Ax. Furthermore, [35] and [16, 17] observed that recovery can be accomplished in polynomial time via linear programming (LP), provided that the measurement matrix A satisfies certain technical conditions.

When defined over a finite field, the problem of recovering a sparse vector from linear measurements, e.g., recovery of a sparse vector from linear measurements over $GF(2)$, is essentially the fundamental problem in the design of binary linear error-correcting codes. In essence, compressed sensing and coding theory both aim to design a matrix A and an estimation or decoding algorithm that allows for faithful recovery of x from y when x is constrained to be sparse.

Given the close connection between the goals of compressed sensing and error-correcting codes, it is natural to ask if the message-passing algorithms that have been so successfully applied to the decoding of error-correcting codes can also be applied to compressed sensing. In the first half of this chapter, we take a step in this direction by analyzing the performance of a message-passing recovery algorithm when the matrix A corresponds to the adjacency matrix of a bipartite graph with good expansion properties. Results of a similar flavor are well-known in the context of coding theory, but are only beginning to be explored in the context of compressed sensing. We summarize some of the related work in this direction below.

Compressed sensing was first put forward in [35] and [16, 17], and as mentioned previously, these works proved that linear programming (LP) can be used to find the sparsest solution to $y = Ax$ under certain conditions on the measurement matrix

A. Since then, many reconstruction algorithms have been proposed [11, 25, 26, 47, 48, 61, 62, 90, 116, 121]—see, e.g., [62] for a summary of various combinations of measurement matrices and algorithms, and their associated performance characteristics. Most existing combinations fall into two broad classes. In the first class, inspired by high-dimensional geometry, the measurement matrix A is typically dense (almost all entries are nonzero), and the recovery algorithms are based on linear or convex optimization. The second class consists of combinatorial algorithms operating on sparse measurement matrices (A typically has only $O(n)$ nonzero entries). Examples include the algorithms of [11, 62, 121]. In particular, Algorithm 1 from [121] is essentially identical to the Sipser-Spielman message-passing algorithm [113]. The algorithm we consider in this chapter also falls into the second class, and is a minor variant of the algorithm proposed in [73]. [1] Very recent work on the use of a message-passing algorithm to identify compressed sensing thresholds appears in [36, 37]. Relative to our analysis, [36] and [37] are more general in that arbitrary (i.e., even dense) matrices A are considered. However, [36, 37] restrict attention to a probabilistic analysis, while we perform an adversarial analysis, and thus we are able to provide deterministic reconstruction guarantees for arbitrary (nonnegative) x.

On the coding theory side, we briefly recall some of the results alluded to in Chapter 1. Specifically, Spielman [113] proved that when the Tanner graph of an LDPC code has sufficiently high expansion, a simple bit-flipping algorithm can correct a constant fraction of errors, even if the errors are chosen by an adversary. In [15], this result was extended to show that a broad class of message-passing algorithms, including common algorithms such as the so-called "Gallager A" and "B" algorithms, also correct a constant fraction of (adversarial) errors when there is sufficient expansion. Finally, [40] suggested decoding LDPC codes via LP, and [41] proved that this LP decoder can correct a constant fraction of (adversarial) errors when there is sufficient expansion. We show that similar techniques can be used to analyze the performance of the message-passing algorithm proposed considered in this chapter.

[1] For the reader who has already read Chapter 3, the algorithm is just the BAILOUT subroutine of Section 3.5.2.

4.2 Problem Formulation

In Sections 4.2, 4.3, 4.4, and 4.5, we describe and analyze a message-passing algorithm for the compressed sensing of nonnegative signals. As our problem model, we seek to estimate a vector $x \in \mathbb{R}_+^n$ of interest from observations of the form $y = Ax \in \mathbb{R}_+^m$, where $A = [A_{ij}] \in \{0,1\}^{m \times n}$ is a known measurement matrix ($\mathbb{R}_+$ denotes the set of nonnegative real numbers). Associated with A is the following bipartite graph $G = (X, Y, E)$. We define $X = \{1, \ldots, n\}$, $Y = \{1, \ldots, m\}$, and $E = \{(i,j) \in X \times Y : A_{ij} = 1\}$. Next, associated with vertex $i \in X$ is x_i, the ith component of x, and with vertex $j \in Y$ is y_j, the jth component of y. Further, we define

$$N_x(i) = \{j \in Y : (i,j) \in E\}, \quad \text{for all } i \in X,$$
$$N_y(j) = \{i \in X : (i,j) \in E\}, \quad \text{for all } j \in Y.$$

Note that the degrees of $i \in X$, $j \in Y$ are $|N_x(i)|$, $|N_y(j)|$, respectively. The structure in A is specified via constraints on the associated graph G. Recall the definition of a (c,d)-regular (k,α)-expander given in Chapter 2.

Definition 4.2.1 (Expander graph)**.** *A given bipartite graph $G = (X, Y, E)$ is a (c,d)-regular (k,α)-expander if every vertex $i \in X$ has degree c, every vertex $j \in Y$ has degree d, and for all subsets $S \subset X$ such that $|S| \leq k$, we have $|\Gamma(S)| \geq \alpha c|S|$, where $\Gamma(S) \triangleq \cup_{i \in S} N_x(i)$.*

4.3 An Iterative Recovery Algorithm

The message-passing algorithm for iteratively recovering x is identical to the BAILOUT subroutine defined in Section 3.5.2. For convenience, we describe the algorithm again below. The algorithm maintains two numbers for each edge $(i,j) \in E$, corresponding to a message in each direction. Let $t \geq 0$ denote the iteration number, and let $m_{i \to j}^{(t)}$ and $m_{j \to i}^{(t)}$ denote the two messages along edge $(i,j) \in E$ in the t^{th} iteration. The principle behind the algorithm is to alternately determine lower and upper bounds

on x. Specifically, $m_{i\to j}^{2t}$ and $m_{j\to i}^{2t+1}$ are lower bounds on x_i for all $t \geq 0$; $m_{i\to j}^{2t+1}$ and $m_{j\to i}^{2t}$ are upper bounds on x_i for all $t \geq 0$. Also, these lower (respectively, upper) bounds are monotonically increasing (respectively, decreasing). That is,

$$m_{i\to j}^{0} \leq m_{i\to j}^{2} \leq \ldots; \quad m_{i\to j}^{1} \geq m_{i\to j}^{3} \geq \ldots.$$

Formally, the algorithm is given by the following pseudocode.

1: Initialization ($t = 0$): for all $(i, j) \in E$, set $m_{i\to j}^{0} = 0$.
2: Iterate for $t = 1, 2, \ldots$:

a) $t = t + 1$, update messages for all $(i, j) \in E$ via

$$m_{j\to i}^{2t-2} = y_j - \sum_{k\in N_y(j)\setminus i} m_{k\to j}^{2t-2} \tag{4.1}$$

$$m_{i\to j}^{2t-1} = \min_{l\in N_x(i)} \left(m_{l\to i}^{2t-2}\right) \tag{4.2}$$

$$m_{j\to i}^{2t-1} = y_j - \sum_{k\in N_y(j)\setminus i} m_{k\to j}^{2t-1} \tag{4.3}$$

$$m_{i\to j}^{2t} = \max\left[0, \max_{l\in N_x(i)} \left(m_{l\to i}^{2t-1}\right)\right] \tag{4.4}$$

b) Estimate x_i via $\hat{x}_i^s = m_{i\to j}^s$ for $s = 2t-1, 2t$.

3: Stop when converged (assuming it does).

Algorithm 4: Recovery Algorithm

4.4 Main Results

Our results on the performance of the message-passing algorithm are summarized in the following two theorems. The first establishes that the algorithm is able to recover sparse signals exactly.

Theorem 4.4.1. *Let G be a (c, d)-regular $(\lfloor \frac{2}{1+2\varepsilon}k + 1 \rfloor, \frac{1}{2} + \varepsilon)$-expander. Then, as long as $k \geq \|x\|_0 = |\{i \in X : x_i \neq 0\}|$, the estimate produced by the algorithm satisfies $\hat{x}^t = x$, for all $t \geq T = O(\log_{1+\varepsilon}(k))$.*

The second theorem establishes that the algorithm can approximately recover x that are not exactly k-sparse, i.e., the algorithm can provide an ℓ_1/ℓ_1 recovery guarantee.

Theorem 4.4.2. *Let G be a (c,d)-regular $(\lfloor \frac{k}{2\varepsilon}+1 \rfloor, \frac{1}{2}+\varepsilon)$-expander. Let $x^{(k)}$ denote the best k-sparse approximation to x, i.e.,*

$$x^{(k)} = \min_{z \in R_+^n : \|z\|_0 \leq k} \|x - z\|_1.$$

Then, for all even $t \geq T = O(\log_{1+\varepsilon}(k) \log_{1+\varepsilon}(cd))$,

$$\|x - \hat{x}^t\|_1 \leq \left(1 + \frac{d}{2\varepsilon}\right) \|x - x^{(k)}\|_1.$$

Discussion: As a sample choice of parameters, we mentioned in Chapter 2 that there exist (c,d)-regular $(k, > .5)$-expanders with $c = O(\log(\frac{n}{k}))$ and $d = O(\frac{n}{k})$. With such expanders, Theorem 4.4.1 implies that the message-passing algorithm can recover any k-sparse x from $O(k \log(\frac{n}{k}))$ measurements in time $O(n \log(\frac{n}{k}) \log(k))$. Compared to the Sipser-Spielman algorithm [113], the message-passing algorithm requires less expansion (0.5 vs. 0.75), but the Sipser-Spielman algorithm works for arbitrary (i.e., not just nonnegative) vectors x. In [67], it is shown that recovery of nonnegative x is possible from far less expansion, but the recovery algorithm proposed in [67] is significantly slower. In particular, they are only able to prove a running time bound of $O(nk^2)$.

Similarly, with appropriate expanders, Theorem 4.4.2 implies that the message-passing algorithm can be implemented to use $O(k \log(\frac{n}{k}))$ measurements, and the factor multiplying the error in the ℓ_1/ℓ_1 guarantee is $O(\frac{n}{k})$. The algorithm can be implemented sequentially in $O(n(\log(\frac{n}{k}))^2 \log(k))$ time, or in parallel $O(\frac{n}{k} \log(k) \log(\frac{n}{k}))$ time using $O(n)$ processors. In particular, when $k = \Theta(n)$—a regime typically of interest in information-theoretic analysis—the algorithm provides a constant factor ℓ_1/ℓ_1 guarantee with $O(n \log(n))$ running time. In this regime, our algorithm is faster than almost all existing algorithms, e.g., [47, 48, 90]; the only exception is [62], which

is faster, and stronger, in that the multiplier in the ℓ_1/ℓ_1 guarantee is only $(1+\varepsilon)$ for the algorithm of [62], rather than $O(\frac{n}{k})$. However, relative to the algorithm of [62], ours has the advantage of working with a smaller expansion factor (albeit at the cost of requiring expansion from larger sets), and is easily parallelizable. In addition, we believe that this message-passing algorithm may be applicable in more general settings, e.g., providing guarantees for recovering "random" vectors when the graph is not an expander, but possesses other properties, such as large girth.

4.5 Analysis

4.5.1 Analysis of Sparse Recovery

(Note: The proof of Theorem 4.4.1 is essentially the same as the proof of Lemma 3.8.7, but because we use Theorem 4.4.1 as the starting point for our proof of Theorem 4.4.2, we repeat the proof below.)

We start by recalling a certain monotonicity property of the messages. For each $i \in X$, the messages $m_{i\to j}^{2t}$ are monotonically nondecreasing lower bounds on x_i, and the messages $m_{i\to j}^{2t+1}$ are monotonically nonincreasing upper bounds on x_i. This can be easily verified by induction. Given this monotonicity property, clearly the messages at even and odd times have limits: if these messages are equal after a finite number of iterations, then the algorithm recovers x. We establish that this is indeed the case under the assumptions of Theorem 4.4.1.

To this end, define $W_{2t} = \{i \in X : x_i > m_{i\to.}^{2t}\}$, i.e., W_{2t} is the set of vertices whose lower bounds are incorrect after $2t$ iterations. Clearly, $|W_0| \leq k$ since $\|x\|_0 \leq k$, and the lower bounds to x_i, $m_{i\to.}^{2t}$, are nonnegative for all t. The monotonicity property of the lower bounds implies that for $0 \leq s < t$, $W_{2t} \subseteq W_{2s}$. Therefore, it is sufficient to establish that $|W_{2t+2}| < (1-2\varepsilon)|W_{2t}|$ if $0 < |W_{2t}| \leq k$; this implies that after $O(\log(k))$ iterations, W_{2t} must be empty.

Suppose $0 < |W_{2t}| \leq k$. Since $W_{2t+2} \subset W_{2t}$, it suffices to show that at least a 2ε fraction of the vertices in W_{2t} are not in W_{2t+2} . We prove this by using the expansion

properties of G. Let $V = \Gamma(W_{2t}) \subset Y$ be the set of all neighbors of W_{2t}. Let $T \subset X$ be $\{i \in X : N_x(i) \subset V\}$. Since G is (c, d)-regular, $|V| \leq c|W_{2t}|$. Also, by definition $W_{2t} \subset T$. We state three important properties of T:

P1. $|T| < 2|W_{2t}|/(1 + 2\varepsilon)$. Suppose not. Then, consider any $T' \subset T$ with $|T'| = \lfloor 2|W_{2t}|/(1 + 2\varepsilon) + 1 \rfloor$. We reach a contradiction as follows:

$$c|W_{2t}| \geq \ |V| \ \geq |\Gamma(T')| \geq |T'|(1 + 2\varepsilon)\frac{c}{2} > |W_{2t}|c.$$

P2. Let $U = \{i \in X : m_{i\to\cdot}^{2t+1} > x_i\}$. Then, $U \subset T$. This is because $m_{i\to\cdot}^{2t+1} = x_i$ if there exists $j \in N_x(i)$ such that $j \notin V$. To see this, note that for such a j, all $k \in N_y(j) \setminus i$ are not in W_{2t}, and hence $x_k = m_{k\to j}^{2t}$ for all these k, so $y_j - \sum_{k \in N_y(j)\setminus i} = x_i$.

P3. Let $T^1 = \{i \in T : \exists j \in V \text{ s.t. } N_y(j) \cap T = \{i\}\}$. Then, $|T^1| \geq 2\varepsilon|T|$. To see this, let $A = |\{j \in V : |N_y(j) \cap T| = 1\}|$, and let $B = |V| - A$. Then, number of edges between T and V is at least $2B + A$, and since G is (c, d)-regular, the number of edges between T and V is at most $c|T|$. Therefore, $A + 2B \leq c|T|$. Now, by [P1], $|T| < 2k/(1 + 2\varepsilon)$, so $|\Gamma(T)| \ \geq c|T|(1 + 2\varepsilon)/2$. Therefore, $A + B \geq c|T|(1 + 2\varepsilon)/2$, whence $A \geq 2\varepsilon c|T|$.

To complete the proof, note that $T^1 \subset W_{2t}$, and $|T^1| \geq 2\varepsilon|T|$ by [P3]. For each $i \in T^1$, let $j(i) \in V$ be its unique neighbor in the definition of T^1, i.e., $N_y(j(i)) \cap T = \{i\}$. Then, [P2] implies that for all $k \in N_y(j(i)) \setminus i$, we have $m_{k\to j(i)}^{2t+1} = x_k$. Therefore, $m_{j(i)\to i}^{2t+1} = x_i$, so $m_{i\to\cdot}^{2t+2} = x_i$. Thus, $i \notin W_{2t+2}$, i.e., $T^1 \subset W_{2t} \setminus W_{2t+2}$, completing the proof of Theorem 4.4.1.

4.5.2 Analysis of ℓ_1/ℓ_1 Recovery

This section establishes Theorem 4.4.2 in two steps. First, using techniques similar to those used to prove Theorem 4.4.1, we obtain a very weak bound on the reconstruction error. Next, we improve this weak bound, by showing that when the error is large, it must be reduced significantly in the next iteration. This yields the desired result.

Given $x \in \mathbb{R}^n_+$, let $x^{(k)}$ denote the best k-term approximation to x. Let $X_+ = \{i \in X : x_i^{(k)} \neq 0\}$, and let $X_0 = X \setminus X_+$. For an arbitrary $S \subset X$, let $e^t(S) = \sum_{i \in S}(x_i - \hat{x}_i^t)$ at the end of iteration t; recall that $\hat{x}^t$ is the algorithm's estimate after t iterations. Note that $\hat{x}_i^{2s} \leq x_i \leq \hat{x}_i^{2s+1}$, so $e^t(S) \geq 0$ for even t, and $e^t(S) \leq 0$ for odd t.

Now, we state the first step of the proof, i.e., the weak bound on reconstruction error.

Lemma 4.5.1. *Let G be a (c,d)-regular $(\lfloor \frac{2k}{1+2\varepsilon} + 1 \rfloor, \frac{1}{2} + \varepsilon)$-expander, for some $\varepsilon > 0$. Then, after $t = O(\log k)$ iterations,*

$$\|x - \hat{x}^t\|_1 \leq O\left((cd)^{O(\log(k))} \log(k)\right) \|x - x^{(k)}\|_1.$$

Proof. We use ideas similar to those used in the proof of Theorem 4.4.1. Let $V = \Gamma(X_+)$ be the set of neighbors of X_+, and let $S' = \{i \in X : N_x(i) \subset V\}$. Also, define sets $S_\ell, \ell \geq 0$ as follows:

$$S_0 = X \setminus S', \quad S_1 = \{i \in S' : \exists j \in V \text{ s.t. } N_y(j) \cap S' = \{i\}\},$$

and for $\ell \geq 2$,

$$S_\ell = \{i \in S' : \exists j \in V \text{ s.t. } N_y(j) \cap (S' \setminus \cup_{\ell' < \ell} S_{\ell'}) = \{i\}\}.$$

We note that by arguments similar to those used to establish property [P3], it follows that $|S_\ell| \geq 2\varepsilon |S' \setminus \cup_{\ell' < \ell} S_{\ell'}|$. Also, [P1] implies that $|S'| \leq \frac{2k}{1+2\varepsilon}$. Therefore, S_ℓ is empty for $\ell \geq O(\log k)$.

Adapting arguments used in the proof of Theorem 4.4.1, we bound $e^{2\ell}(S_\ell)$ for $\ell \geq 0$. First, by definition $S_0 \subset X_0$, so $e^0(S_0) \leq \|x - x^{(k)}\|_1$. Now, consider $e^2(S_1)$. By definition, each vertex $i \in S_1$ has a unique neighbor j, i.e., a neighbor j such that $N_y(j) \setminus i \subset S_0$. Therefore,

$$x_i - \hat{x}_i^2 \leq \sum_{i' \in N_y(j) \setminus i} (\hat{x}_{i'}^1 - x_{i'}).$$

Each $i' \in S_0$, so for each i' we have a neighbor $j' \notin V$, i.e., $N_y(j') \subset X_0$. Therefore,

$$\hat{x}_{i'}^1 - x_{i'} \leq \sum_{i'' \in N_y(j') \setminus i'} (x_{i''} - \hat{x}_{i''}^0),$$

where all $i'' \in X_0$. Thus,

$$x_i - \hat{x}_i^2 \leq \sum_{i' \in N_y(j) \setminus i} \sum_{i'' \in N_y(j') \setminus i'} (x_{i''} - \hat{x}_{i''}^0),$$

and summing over all $i \in S_1$, we obtain

$$e^2(S_1) \leq \sum_{i \in S_1} \sum_{i' \in N_y(j) \setminus i} \sum_{i'' \in N_y(j') \setminus i'} (x_{i''} - \hat{x}_{i''}^0).$$

Now, we bound the number of times a particular vertex $i'' \in S_0$ can appear on the right-hand side of the above inequality. i'' can only occur in sums corresponding to a vertex $i \in S_1$ such that there exists a walk of length 4 between i'' and i in G. Therefore, i'' can occur in at most $(cd)^2$ terms; hence, $e^2(S_1) \leq (cd)^2 e^0(S_0)$. Similarly, we can bound $e^{2\ell}(S_\ell)$ for $\ell > 1$ by induction. Assume that for all $\ell' < \ell$, $e^{2\ell'}(S_{\ell'}) \leq (cd)^{2\ell'} \|x - x^{(k)}\|_1$. For each vertex $i \in S_\ell$, there exists a unique neighbor j, i.e., j satisfies $N_y(j) \setminus i \subset \cup_{\ell' < \ell} S_\ell$. Thus, $x_i - \hat{x}_i^{2\ell} \leq \sum_{i' \in N_y(j) \setminus i} (\hat{x}_{i'}^{2\ell-1} - x_{i'})$. As before, each i' has a unique neighbor j', and summing over $i \in S_\ell$, we obtain

$$e^{2\ell}(S_\ell) \leq \sum_{i \in S_\ell} \sum_{i' \in N_y(j) \setminus i} \sum_{i'' \in N_y(j') \setminus i'} (x_{i''} - \hat{x}_{i''}^{2\ell-2}),$$

where all $i'' \in \cup_{\ell' < \ell-1} S_\ell$. Again, each i'' can only occur $(cd)^2$ times, so we conclude that

$$\begin{aligned} e^{2\ell}(S_\ell) &\leq (cd)^2 \sum_{\ell'=0}^{\ell-2} e^{2\ell-2}(S_{\ell'}) \leq (cd)^2 \sum_{\ell'=0}^{\ell-2} e^{2\ell'}(S_{\ell'}) \\ &\leq (cd)^2 \sum_{\ell'=0}^{\ell-2} (cd)^{2\ell'} \|x - x^{(k)}\|_1 \leq (cd)^{2\ell'} \|x - x^{(k)}\|_1, \end{aligned}$$

where the second inequality is true because of the monotonicity property of the lower bounds. Thus, we have shown inductively that $e^{2\ell}(S_\ell) \leq (cd)^{2\ell}\|x - x^{(k)}\|_1$ for all ℓ. Since there are at most $O(\log k)$ nonempty sets S_ℓ, it follows that after $t = O(\log k)$ iterations,

$$\|x-\hat{x}^t\|_1 \leq \sum_\ell e^{2\ell}(S_\ell) \leq O\left((cd)^{O(\log(k))}\log(k)\right)\|x-x^{(k)}\|_1.$$

□

On one hand, Lemma 4.5.1 gives a weak bound on the reconstruction error, as the multiplier is poly(n). On the other hand, it provides good starting point for us to boost it to obtain a better bound by using the second step described next. To that end, we first state a definition and lemma adapted from [41].

Definition 4.5.1. *Given a (c,d)-regular bipartite graph $G = (X, Y, E)$ and a subset $S \subset X$, let $B(S) = \{i \in X \setminus S : N_x(i) \cap \Gamma(S) > \frac{c}{2}\}$. For a given constant $\delta > 0$, we say that S has a δ-matching if there exists a set $M \subset E$ such that: (a) $\forall j \in Y$, at most one edge of M is incident to j; (b) $\forall i \in S \cup B(S)$, at least δc edges of M are incident to i.*

Lemma 4.5.2. *Let $G = (X, Y, E)$ be a (c,d)-regular $(\lfloor \frac{k}{2\varepsilon} + 1 \rfloor, \frac{1}{2} + \varepsilon)$-expander, for some $\varepsilon > 0$. Then, every $S \subset X$ of size at most k has a $\frac{1}{2} + \varepsilon$-matching.*

To keep this chapter self-contained, we include the proof of Lemma 4.5.2. The following proof is essentially identical to Proposition 4 and Lemma 5 in [41].

Proof. We construct a $\frac{1}{2} + \varepsilon$-matching by analyzing the following max-flow problem. Consider the subgraph of G induced by the set of left vertices $U = S \cup B(S)$ and right vertices $V = \Gamma(S \cup B(S))$. We assign a capacity of 1 to every edge in this subgraph, and direct these edges from U to V. Finally, we add a source s with an edge of capacity $(\frac{1}{2} + \varepsilon)c$ pointing to each vertex in U, and a sink t with an incoming edge of capacity 1 from every vertex in V. If the maximum $s - t$ flow in this graph is $(\frac{1}{2} + \varepsilon)c|U|$, then we have constructed a $\frac{1}{2} + \varepsilon$-matching. To see this, recall that if the

capacities are integral, then the maximum flow can always be chosen to be integral, and the edges between U and V with nonzero flow values in an integral maximum flow form a $\frac{1}{2}+\varepsilon$-matching.

To complete the proof, we show that the minimum $s-t$ cut in the max-flow problem constructed above is $(\frac{1}{2}+\varepsilon)c|U|$. To see this, consider an arbitrary $s-t$ cut $s \cup A \cup B$, where $A \subset U$ and $B \subset V$. The capacity of this cut is $(\frac{1}{2}+\varepsilon)c(|U| - |A|) + |B| + C$, where C is the number of edges between A and $V - B$. Assume, for sake of contradiction, that $\Gamma(A) \not\subset B$. Then, from the above formula it follows that we can produce a cut of at most the same value by replacing B by $B \cup \Gamma(A)$. In more detail, imagine that we move the vertices in $\Gamma(A) \cap V - B$ to B one by one, i.e., we move these vertices from the t-side of the cut to the s-side of the cut. Each time we move such a vertex, $|B|$ goes up by 1, and C decreases by at least 1 because by definition we are moving vertices in $\Gamma(A)$. Therefore, without loss of generality we can assume that $\Gamma(A) \subset B$. Now, an argument similar to that used to prove P1 shows that $|A| \leq \lfloor \frac{|S|}{2\varepsilon} \rfloor$: $|S| \leq k$, so if $|A| \geq \lfloor \frac{|S|}{2\varepsilon} + 1 \rfloor$, then there exists a set of size $k' = \lfloor \frac{|S|}{2\varepsilon} + 1 \rfloor$ with at most $c|S| + \frac{c}{2}(k' - |S|) < (\frac{1}{2}+\varepsilon)ck'$ neighbors, contradicting the $(\lfloor \frac{k}{2\varepsilon} + 1 \rfloor, \frac{1}{2}+\varepsilon)$-expansion of G. Therefore, $|\Gamma(A)| \geq (\frac{1}{2}+\varepsilon)c|A|$, so the min-cut has capacity at least $(\frac{1}{2}+\varepsilon)c(|U| - |A|) + (\frac{1}{2}+\varepsilon)c|A| = (\frac{1}{2}+\varepsilon)c|U|$. □

We use δ-matchings to prove that the reconstruction error decays by a constant factor in each iteration.

Lemma 4.5.3. *Let G be a (c,d)-regular $(\lfloor \frac{k}{2\varepsilon} + 1 \rfloor, \frac{1}{2}+\varepsilon)$-expander, for some $\varepsilon > 0$. Then,*

$$e^{2t+2}(X_+) \leq \frac{1-2\varepsilon}{1+2\varepsilon} e^{2t}(X_+) + \frac{2d}{1+2\varepsilon} e^{2t}(X_0).$$

In our proof of Lemma 4.5.3, we make use of the following lemma establishing a simple invariance satisfied by the message-passing algorithm. Since this invariance was used earlier in the proof of Lemma 4.5.1, a proof is omitted.

Lemma 4.5.4. *For any $i \in X$, construct a set S as follows. First, choose a vertex $j \in N_x(i)$. Next, for each $i' \in N_y(j) \setminus i$, choose a vertex $w(i') \in N_x(i')$ (note that*

these choices can be arbitrary). Finally, define S as $\cup_{i' \in N_y(j) \setminus i} N_y(w(i')) \setminus i'$. Then, no matter how j and $w(i')$ are chosen,

$$x_i - \hat{x}_i^{(2t+2)} \leq \sum_{i'' \in S} (x_{i''} - \hat{x}_{i''}^{(2t)}).$$

Proof of Lemma 4.5.3. Lemma 4.5.2 guarantees the existence of a $\frac{1}{2} + \varepsilon$-matching, say M, for the set X_+ of (at most) k vertices in X. We use this $\frac{1}{2} + \varepsilon$-matching to produce a set of inequalities of the form given in Lemma 4.5.4. By adding these inequalities, we prove Lemma 4.5.3.

For each $i \in X_+$, let $M(i)$ be the set of neighbors of i in the $\frac{1}{2} + \varepsilon$-matching. We construct an inequality, or equivalently, a set S, for each member of $M(i)$. We construct the sets S sequentially as follows. Fix i and $j \in M(i)$. For each $i' \in N_y(j) \setminus i$, we must choose a neighbor $w(i')$. If $i' \in X_+$ or $i' \in B(X_+)$, set $w(i')$ to be any vertex in $M(i')$ that has not been chosen as $w(i')$ for some previously constructed set. If $i' \in X \setminus (X_+ \cup B(X_+))$, choose $w(i')$ to be any element of $N_x(i') \setminus \Gamma(X_+)$ that has not been chosen as $w(i')$ for some previously constructed set. Although it may not be immediately apparent, we will see that this process is well-defined, i.e., i' will always be able to choose a neighbor $w(i')$ that has not been used previously. First, however, we complete the proof assuming that the process is well-defined.

To that end, we establish Lemma 4.5.3 by adding together all the inequalities associated with the sets S constructed above. First, consider the left-hand side of this sum. The only terms that appear are $x_i - \hat{x}_i^{(2t+2)}$, where $i \in X_+$, and each of these appears at least $(\frac{1}{2} + \varepsilon)c$ times since $|M(i)| \geq (\frac{1}{2} + \varepsilon)c$ for all such i. On the right-hand side, we must count how many times each term $x_i - \hat{x}_i^{(2t)}$ appears in some inequality, i.e., how many times vertex i appears in the second level of some set S. We break the analysis up into two cases. First, assume that $i \in X_+$. Then, $x_i - \hat{x}_i^{(2t)}$ can appear in the second level of a set S only if some vertex in $N_x(i)$ was chosen as $w(i')$ for some $i' \neq i$ when we were defining S. This is only possible for $i' \in X_+ \cup B(X_+)$. To bound the contribution due to such i', note that the vertices in $M(i)$ can never be chosen as $w(i')$ for $i' \neq i$, and that every vertex in $\cup_{i \in X_+} M(i)$ is chosen at most

once. Therefore, $x_i - \hat{x}_i^{(2t)}$ appears at most $(\frac{1}{2} - \varepsilon)c$ times. To bound the number of appearances of $x_i - \hat{x}_i^{(2t)}$ for $i \notin X_+$, note that any vertex can appear in some set S at most cd times. To see this, note that any vertex in Y can appear as $w(i')$ for a set S at most d times, once for each of its neighbors, because a single vertex in X never chooses the same neighbor as its $w(i')$ more than once. The bound then follows since each vertex in X has degree c. Hence,

$$\left(\frac{1}{2} + \varepsilon\right) c\, e^{2t+2}(X_+) \leq \left(\frac{1}{2} - \varepsilon\right) c\, e^{2t}(X_+) + c\, d\, e^{2t}(X_0),$$

or equivalently,

$$e^{2t+2}(X_+) \leq \frac{1 - 2\varepsilon}{1 + 2\varepsilon} e^{2t}(X_+) + \frac{2d}{1 + 2\varepsilon} e^{2t}(X_0).$$

We complete the proof by showing that the process for constructing the sets S is well-defined. The analysis above implicitly establishes this already. First, note that every $i' \in X_+ \cup B(X_+)$ has at least $(\frac{1}{2} + \varepsilon)c$ distinct neighbors that can be chosen as $w(i')$, and by definition every $i' \in X \setminus (X_+ \cup B(X_+))$ has at least $\frac{c}{2}$ distinct neighbors that can be chosen as $w(i')$. Therefore, in order to prove that the construction procedure for the sets S is well-defined, it suffices to show that every vertex can appear as an i', i.e., in the first level of some S, at most $\frac{c}{2}$ times. For $i' \in X_+ \cup B(X_+)$, at least $(\frac{1}{2} + \varepsilon)c$ of i''s neighbors are in the $\frac{1}{2} + \varepsilon$-matching, so any such i' appears at most $(\frac{1}{2} - \varepsilon)c$ times. For $i' \in X \setminus (X_+ \cup B(X_+))$, by definition $N_x(i') \cap \Gamma(X_+) \leq \frac{c}{2}$, so any such i' appears at most $\frac{c}{2}$ times. □

Completing the proof of Lemma 4.4.2. We combine lemmas 4.5.1 and 4.5.3. First, from Lemma 4.5.1, after $t = O(\log(k))$ iterations, the error satisfies the bound

$$\|x - \hat{x}^t\|_1 \leq O((cd)^{O(\log(k))} \log(k)) \|x - x^{(k)}\|_1.$$

Lemma 4.5.3 implies that after an additional $O(\log(k) \log(cd))$ iterations, the error satisfies

$$\|x - \hat{x}^{t+O(\log(k)\log(cd))}\|_1 \leq \left(1 + \frac{d}{2\varepsilon}\right) \|x - x^{(k)}\|_1.$$

To see this, apply the inequality

$$e^{2t+2}(X_+) \leq \frac{1-2\varepsilon}{1+2\varepsilon} e^{2t}(X_+) + \frac{2d}{1+2\varepsilon} e^{2t}(X_0)$$

repeatedly, and note that $e^{2t}(X_0)$ is monotonically nonincreasing as a function of t, so $e^{2t}(X_0) < e^0(X_0)$. □

4.6 Negative Results for Sparse Graph Codes

In this section, we define the RIP and give some background to explain its importance in the context of compressed sensing. Sections 4.6.2 and 4.6.3 prove that binary and very sparse matrices have bad performance with respect to the RIP.

4.6.1 Background on the Restricted Isometry Property (RIP)

As mentioned in Section 4.1, [35] and [16, 17] showed that when the measurement matrix A satisfies certain technical conditions, LP can be used to recover sparse signals. In particular, [18] introduced the RIP_p as a useful tool for proving this result.

Definition 4.6.1. *Let A be an $m \times n$ matrix. Then, A satisfies the $(k, D) - RIP_p$ if there exists $c > 0$ such that for all k-sparse $x \in \mathbb{R}^n$,*

$$c||Ax||_p \leq ||x||_p \leq cD||Ax||_p.$$

Here $||\cdot||_p$ denotes the ℓ_p norm. c is a scaling constant that we include for convenience in what follows.

In the following, we suppress the subscript p when the results being reviewed are valid for any choice of p. The RIP is useful because it can be shown that if a matrix A satisfies the RIP, then LP can be used recover a k-sparse signal x from Ax. We note that there are other properties besides the RIP that can be used to prove that a matrix can be used for compressed sensing (cf. [66]). However, one nice aspect of

the RIP is that the RIP allows one to prove bounds on the reconstruction error in the case that x is only approximately sparse and/or the measurements are corrupted by noise, e.g., the RIP_2 can be used to prove ℓ_2/ℓ_1 reconstruction guarantees.

As mentioned in the introduction to this chapter, ℓ_2/ℓ_1 guarantees are stronger than ℓ_1/ℓ_1 guarantees, so constructing matrices satisfying the RIP_2 is a natural problem to consider. It is well-known that there exist matrices satisfying the RIP_2 with $O(k \log (n/k))$ rows, and this is within a constant factor of a lower bound which states that $\Omega(k \log (n/k))$ rows are necessary (see, for example, [5]). However, the proof that such matrices exist uses the probabilistic method, and an explicit construction of such a matrix remains elusive.

The currently known explicit constructions ([89], [26], [31]) use many more rows than the best (randomly constructed) matrices. Specifically, these constructions requires $\Omega(k^2)$ rows. One would hope that these constructions could be improved to get explicit matrices with close to $O(k \log (n/k))$ rows. For example, the results of [9, 10] show that explicit matrices can be constructed that satisfy the RIP_1. Specifically, there exist (non-explicit) matrices with only $O(k \log (n/k))$ rows which satisfy the RIP_1. In addition, there are explicit matrices with $O(k2^{(\log \log n)^{O(1)}})$ rows that satisfy the RIP_1. Another nice property of these matrices is that they are sparse, in the sense that most of the entries are 0. Sparsity is desirable because it can make algorithms for computing Ax and recovering x from Ax more efficient [9, 10].

In Section 4.6.2, we show that for the RIP_2, a fundamentally different approach is needed. In particular, the previously mentioned explicit constructions all use binary matrices, i.e., matrices whose entries are 0 or 1. We show that binary matrices require substantially more than $O(k \log (n/k))$ rows to satisfy the RIP_2. In Section 4.6.3, we show that even if one relaxes the binary constraint, matrices that satisfy the RIP_2 must have significantly more nonzero entries that matrices satisfying the RIP_1.

4.6.2 Binary Matrices are Bad With Respect to the RIP_2

In this section we prove that binary matrices cannot achieve good performance with respect to the RIP_2. First, we make a couple of definitions that will be used in what

follows. Given a binary matrix A, let f be the minimum fraction of 1's in any column of A, i.e., fm is the minimum number of ones in a column of A. Also, let r be the maximum number of 1's in any row of A. By permuting the rows and columns of A, we may assume that $A_{1,i} = 1$ for $1 \leq i \leq r$, i.e., that the first row of A has a 1 in the first r entries. In the following proofs, we assume that this reordering has already been done.

The following theorem shows that binary matrices require substantially more rows than optimal matrices satisfying the RIP_2.

Theorem 4.6.1. *Let A be an $m \times n$ binary matrix satisfying the $(k, D)-RIP_2$. Then, $m \geq \min\left\{\frac{k^2}{D^4}, \frac{n}{D^2}\right\}$.*

Before proving Theorem 4.6.1, we need the following lemma.

Lemma 4.6.1. *Let A be an $m \times n$ binary matrix satisfying the $(k, D) - RIP_2$. Then, $f \leq \frac{D^2}{k}$.*

Proof. To start, we square the inequalities defining the $(k, D) - \text{RIP}_2$ to obtain

$$c^2||Ax||_2^2 \leq ||x||_2^2 \leq c^2 D^2 ||Ax||_2^2.$$

We can bound c by considering the 1-sparse vector $x = e_i$, where e_i is a standard basis vector with a 1 in a coordinate corresponding to a column of A with fm 1's. Then, the inequalities above give $c^2 fm \leq 1 \leq c^2 D^2 fm$. We will only need the second inequality, which we rewrite as

$$\frac{1}{c^2} \leq D^2 fm. \tag{4.5}$$

Now, consider the vector x with 1's in the first k positions and 0's elsewhere, i.e., $x = \sum_{i=1}^{k} e_i$. Let $w(i)$ denote the number of 1's in row i and in the first k columns of A. From the definition of f, the first k columns of A contain at least fmk 1's, so $\sum w(i) \geq fmk$. Applying the Cauchy-Schwartz inequality, we obtain

$$||Ax||_2^2 = \sum_{i=1}^{m} w(i)^2 \geq \frac{\left(\sum_{i=1}^{m} w(i)\right)^2}{m} \geq (fk)^2 m.$$

But

$$||Ax||_2^2 \leq \frac{||x||_2^2}{c^2} \leq D^2 fmk,$$

so putting the two inequalities together gives $(fk)^2 m \leq D^2 fmk$. Cancelling out fmk from both sides gives $f \leq \frac{D^2}{k}$. □

Now, we prove Theorem 4.6.1.

Proof. Lemma 4.6.1 gives us a bound on f that will be useful when $r > k$. We can obtain a second bound on f from the obvious inequality $fmn \leq$ number of 1's in $A \leq rm$. Thus, $f \leq \frac{r}{n}$. As we will see, this bound is useful when $r \leq k$.

For notational convenience, let $s = \min\{r, k\}$. Consider the vector x with 1's in the first s positions and 0's elsewhere, i.e., $x = \sum_{i=1}^{s} e_i$. For this choice of x, we see that $||Ax||_2^2 \geq s^2$, because the first entry of Ax is s. Note that because $s \leq k$, the inequalities defining the $(k, D) - \text{RIP}_2$ apply to x. Thus, we can apply equation 4.5 to get

$$s^2 \leq ||Ax||_2^2 \leq \frac{||x||_2^2}{c^2} \leq D^2 fms.$$

We now plug in our two bounds on f. If $r > k$, then $s = k$, and we use the bound from Lemma 4.6.1. This gives

$$k^2 \leq D^2 \frac{D^2}{k} mk,$$

so $m \geq \frac{k^2}{D^4}$. Similarly, if $r \leq k$, then $s = r$, so using our second bound on f gives

$$r^2 \leq D^2 \frac{r}{n} mr.$$

Thus, in this case $m \geq \frac{n}{D^2}$. Putting the two bounds together, we see that $m \geq \min\{\frac{k^2}{D^4}, \frac{n}{D^2}\}$, completing the proof. □

Note that our lower bounds on m are essentially tight for a large range of parameters. Specifically, assume that $D = O(1)$ and that $k = \Theta(n^\alpha)$ for some constant $0 < \alpha < .5$. Then, the matrices constructed in [31] achieve $m = O((\frac{k}{\alpha})^2)$, so these matrices are within a constant factor of our lower bound.

4.6.3 Very Sparse Matrices are Bad with Respect to the RIP_2

Using an argument similar to that in the previous section, we can show that sparse matrices (with arbitrary nonzero entries) do not achieve good performance with respect to the RIP_2. Define the density ρ of A as

$$\rho = \frac{\text{number of nonzero entries in } A}{n},$$

i.e., ρ is the average number of nonzero entries per column. We have the following result.

Theorem 4.6.2. *Let A be an $m \times n$ matrix that satisfies the $(k, D) - RIP_2$. Assume that A has density ρ. Then, $\rho \geq \min\left\{\frac{n}{4D^2m}, \frac{k}{2D^2}\right\}$.*

Proof. Because there are no restrictions on the entries in A, we can rescale A, and thus without loss of generality we may assume that the c appearing in the definition of the RIP_2 is 1. Now, by Markov's inequality, if the density is ρ, then at most $\frac{n}{2}$ columns of A have more than 2ρ nonzero entries. If we discard these columns, the modified matrix A' still satisfies the $(k, D) - \text{RIP}_2$, and has at least $\frac{n}{2}$ columns. All columns of A' have at most 2ρ ones. Now, by considering the definition of the RIP_2 when x is a standard basis vector, we see that the norm of every column of $A' \geq 1$, and therefore every column of A' has an element with absolute value $\geq \frac{1}{\sqrt{2\rho}}$.

Now, associate each column of A' with the row of the largest element (in absolute value) in the column, breaking ties arbitrarily. Then, by the pigeonhole principle, some row has at least $\frac{n}{2m}$ columns associated with it. By appropriately permuting the rows and columns, without loss of generality we may assume that columns $1, \ldots, \frac{n}{2m}$ are associated with the first row. Consider the vector $x = \sum_{i=1}^{\frac{n}{2m}} \text{sgn}(A_{1,i})e_i$, where $\text{sgn}(A_{1,i})$ denotes the sign of the i^{th} entry in the first row of A, i.e., $\text{sgn}(A_{1,i}) = 1$ if the i^{th} entry is positive, and -1 otherwise. Clearly $||x||_2^2 = \frac{n}{2m}$, but $A'x$ has first entry at least $\frac{n}{2m\sqrt{2\rho}}$. Thus, if $\frac{n}{2m} \leq k$, then x is k-sparse, and we have

$$\frac{n^2}{8\rho m^2} \leq ||A'x||_2^2 \leq D^2\frac{n}{2m}.$$

Rearranging this inequality, we see that $\rho \geq \frac{n}{4D^2m}$.

On the other hand, if $\frac{n}{2m} > k$, we can consider the k-sparse vector $x = \sum_{i=1}^{k} \text{sgn}(A_{1,i})e_i$. This gives us

$$\frac{k^2}{2\rho} \leq ||A'x||_2^2 \leq D^2k,$$

so $\rho \geq \frac{k}{2D^2}$. □

Theorem 4.6.2 gives us a lower bound on the density for matrices that are (asymptotically) optimal with respect to the RIP_2. Namely, Theorem 4.6.2 implies that to have $m = O(k \log \frac{n}{k})$, ρ must be at least $\min\{\Omega(\frac{n}{4D^2k \log \frac{n}{k}}), \frac{k}{2D^2}\}$. For example, in the case that $D = O(1)$ and $k = n^\alpha$ for some $0 < \alpha < 1$, we see that matrices satisfying the RIP_2 with $m = O(k \log \frac{n}{k})$ must have density at least $\Omega(n^{\min\{\alpha, 1-\alpha\}-o(1)})$. This is in contrast with the case for the RIP_1, where the results of [9, 10] imply that there exist (non-explicit) matrices with density $O(\log \frac{n}{k})$ and $O(k \log(\frac{n}{k}))$ rows that satisfy the RIP_1.

4.7 Conclusion

In the first part of this chapter, we performed an adversarial analysis of a simple message-passing algorithm for recovering a vector $x \in R_+^n$ from measurements $y = Ax \in R_+^m$, where $A \in \{0, 1\}^{m \times n}$. We showed that when x has at most k nonzero entries and A corresponds to a bipartite graph with expansion factor greater than 0.5, the message-passing algorithm recovers x exactly. For approximate recovery, we proved that when A corresponds to a bipartite graph with expansion factor greater than 0.5, the message-passing algorithm produces an estimate $\hat{x}$ satisfying an ℓ_1/ℓ_1 recovery guarantee of the form $\|x - \hat{x}\|_1 \leq O(\frac{n}{k})\|x - x^{(k)}\|_1$, where $x^{(k)}$ is the best k-sparse approximation of x. Choosing appropriate expanders, this implies that the message-passing algorithm can recover k-sparse x from $O(k \log(\frac{n}{k}))$ measurements in time $O(n \log(\frac{n}{k}) \log(k))$, and can recover arbitrary x with an ℓ_1/ℓ_1 guarantee from $O(k \log(\frac{n}{k}))$ in time $O(n(\log(\frac{n}{k}))^2 \log(k))$. As noted in Section 4.4, these results are better than almost all algorithms in the literature, with the exception of [62]. Viewed

more broadly, our results can be interpreted as a further step toward formally connecting the theory of message-passing algorithms with that of compressed sensing, which we anticipate being of growing importance to further advances in the field.

In the second part of this chapter, we showed that sparse graph codes are inherently limited with respect to the RIP_2. Finding an explicit construction of matrices satisfying the RIP_2 remains an interesting open problem. We showed that existing constructions have essentially been pushed as far as possible, in the sense that our bound on the number of rows required by binary matrices is within a constant factor of the construction in [31]. We also proved that matrices satisfying the RIP_2 must have significantly more nonzero entries than matrices satisfying the RIP_1. This suggests that from the point of view of developing efficient algorithms, either a new class of codes, or a new analysis technique, is required to design efficient (linear or nearly linear time) algorithms providing stronger reconstruction guarantees, e.g., ℓ_2/ℓ_1 guarantees.

Chapter 5

Lossy Source Coding

This chapter presents several results on lossy source coding. In the first half of this chapter (Sections 5.2 and 5.3), we show that for a broad class of rate-distortion problems, LDPC codes can achieve rates arbitrarily close to the rate-distortion function. In addition, our proof technique shows that if one ignores computational complexity, there is a strong connection between channel coding and rate-distortion coding. Specifically, we show that for a broad class of rate-distortion problems, a channel code achieving low probability of error (under ML decoding) for the reverse channel [28] automatically achieves low distortion (using an optimal quantizer) for the original rate-distortion problem. In particular, as the code rate approaches the capacity of the reverse channel, the distortion approaches the rate-distortion bound.

In retrospect, this result is not surprising—many authors, for example [28], have observed that there is a connection between the standard random coding arguments used for channel coding and lossy source coding. However, the fact that the random coding solutions are related does not mean that a given good channel code can be used for lossy source coding. Phrased differently, just knowing that on average, randomly chosen codes are good for both channel coding and lossy source coding does not allow one to conclude that every good channel code is a good lossy source code. In fact, recent work on lossy source coding suggests that lossy source coding and channel coding are not as closely related as one would hope. For example, [80] suggests that although LDPC codes are much better than LDGM codes for channel coding, LDGM

codes may be fundamentally better than LDPC codes for lossy source coding. Our results do not rule out the possibility that LDPC codes are worse than LDGM codes when complexity is taken into consideration, but they do show that the connection between the lossy source coding and channel coding problems is stronger than just a relationship between random codes—ignoring complexity, any given good channel code must be a good lossy source code.

In the second half of the chapter (Sections Section 5.4 and Section 5.5), we investigate different sparse graph structures to see how the choice of graph structure impacts the tradeoff between graph sparsity, i.e., the number of edges in the graphical representation of the code, and the best achievable rate-distortion performance. For the problem of compressing a binary symmetric source under Hamming distortion (BSS-HD) (see Section 5.1 for a precise definition of this problem), we provide lower bounds on the number of edges in the graph representing an LDGM code or an LDPC code as a function of the rate-distortion performance. More precisely, we show that even if an optimal (and potentially computationally expensive) quantization algorithm is used, the average degree of the vertices in the graph representing an LDGM code or an LDPC code most grow like $\Omega(\log(\frac{1}{\varepsilon}))$ in order for the code to come within ε of the rate-distortion bound. Contrast this with a compound code formed by concatenating LDGM and LDPC codes. It has been shown [79] that using an optimal quantization algorithm, the compound construction can produce codes achieving performance arbitrarily close to the rate-distortion bound for the BSS-HD problem [79], and the average degree remains uniformly bounded regardless of the gap ε to the rate-distortion bound.

We emphasize that the lower bounds on sparsity for LDGM codes and LDPC codes should not be viewed as a strong argument against using these codes. The logarithmic dependence of the average degree on the gap to the rate-distortion bound is identical to the famous lower bound from [45] on the performance of LDPC codes for the BSC as a function of the average check node degree. Despite this lower bound, LDPC codes remain one of the best practical solutions to many channel coding problems, and our lower bounds by no means preclude either LDGM or LDPC codes

from being practically useful for lossy source coding. However, our lower bounds represent a first step towards extending some results from channel coding to the context of lossy source coding. Specifically, in the context of channel coding for the BEC, LDGM codes cannot achieve low block error rates, and [106] shows that the average degree of the vertices in good LDPC codes must grow logarithmically as a function of the gap to capacity. On the other hand, the compound construction formed by concatenating LDGM codes and LDPC codes can achieve rates arbitrarily close to channel capacity (under ML decoding) for any memoryless binary input output-symmetric channel [58]. Furthermore, several classes of compound codes have been found that achieve capacity for the BEC under message-passing decoding with uniformly bounded complexity [58, 96]. In summary, for the BEC, these results show that compound codes offer fundamental advantages over either LDGM codes or LDPC codes in the limit of long blocklengths and very small gaps to capacity.

Our lower bounds show that it is possible that compound codes offer a similar advantage in the context of lossy source coding. Of course, to formalize the analogy between lossy source coding and the BEC one would have to develop an efficient quantization algorithm for compound codes, and as mentioned previously, our lower bounds do not rule out the possibility of efficient quantization algorithms for LDGM codes and LDPC codes, just as the lower bound from [45] did not rule out efficient algorithms for decoding LDPC codes over the BSC. All one can conclude from our lower bounds is that if a message-passing algorithm, or similar algorithm whose complexity is proportional to the number of edges in the graph, is used for quantization, then compound codes can potentially achieve performance near the rate-distortion bound with lower complexity than LDGM codes or LDPC codes.

5.1 Rate-Distortion Problem Formulation

In this section, we review the rate-distortion problem [109] and define some notation. In the standard rate-distortion problem formulation, $\mathcal{X}$ and $\mathcal{Y}$ denote finite source and reconstruction alphabets, respectively, and P_X denotes a probability distribution

over $\mathcal{X}$. Let $d : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ denote the per-symbol distortion function, and let M denote the maximum value of d over $\mathcal{X} \times \mathcal{Y}$. We consider a quantizing an i.i.d. source X^n where each symbol is distributed according to P_X. Given two strings $x^n \in \mathcal{X}^n$ and $y^n \in \mathcal{Y}^n$, define the distortion as the sum of the per-letter distortions, i.e., $d(x^n, y^n) = \sum_{i=1}^n d(x_i, y_i)$. The goal is to represent X^n using as few bits as possible, subject to the constraint that the expected distortion between X^n and the reconstruction $Y^n(X^n)$ is at most Dn, or equivalently, the expected average distortion per source symbol is at most D. We define the rate-distortion function $R(D)$ as the minimum rate of any code capable of quantizing X^n to within expected average distortion D. As shown in [109],

$$R(D) = \min_{Q_{Y|X}: E_{P_X Q_{Y|X}}[d(X,Y)] \leq D} I(X;Y),$$

where X is a random variable distributed according to P_X, Y is a $\mathcal{Y}$-valued random variable, $Q_{Y|X}$ denotes the conditional distribution of Y given X, and $E_{P_X Q_{Y|X}}$ denotes expectation with respect to the distribution where the pair $X, Y \sim P_X Q_{Y|X}$, i.e., $p(x, y) = P_X(x) Q_{Y|X}(y|x)$ for all pairs $x \in \mathcal{X}, y \in \mathcal{Y}$.

Here are three examples of rate-distortion problems considered in this chapter.

- Binary Erasure Quantization: The binary erasure quantization (BEQ) problem has $\mathcal{X} = \{0, 1, *\}$ and $\mathcal{Y} = \{0, 1\}$. The distortion function is given by

$$d(x, y) = \begin{cases} 0 & \text{for} \quad x \in \{0, 1\} \text{ and } x = y \\ 1 & \text{for} \quad x \in \{0, 1\} \text{ and } x \neq y \\ 0 & \text{for} \quad x = * \end{cases}.$$

  The source distribution P_X is specified in terms of the erasure probability e. Given $e \in [0, 1]$, $P_X(*) = e$ and $P_X(0) = P_X(1) = \frac{1-e}{2}$. The associated rate-distortion function at 0 distortion is $R(0) = 1 - e$.

- Quantization Of A Binary Symmetric Source Under Hamming Distortion: The problem of quantizing of a binary symmetric source under Hamming distortion

(BSS-HD) is the rate-distortion problem where $\mathcal{X} = \mathcal{Y} = \{0, 1\}$, $P_X(x) = .5$ for both $x = 0$ and $x = 1$, and

$$d(x, y) = \begin{cases} 0 & \text{for} \quad x = y \\ 1 & \text{for} \quad x \neq y \end{cases}.$$

The associated rate-distortion function is $R(D) = 1 - h_b(D)$, where $h_b(D) = -D\log(D) - (1 - D)\log(1 - D)$ denotes the binary entropy function.

- Quantization of a Gaussian Source Under Mean-Square Distortion: Quantization of a Gaussian source under mean-square distortion is the rate-distortion problem where $\mathcal{X} = \mathcal{Y} = \mathcal{R}$, $X \sim \mathcal{N}(0, 1)$, and $d(x, y) = (x - y)^2$. The associated rate-distortion function is $R(D) = \frac{1}{2}\log(\frac{1}{D})$. Note that this problem does not satisfy our definition of a rate-distortion problem because $\mathcal{X}$ and $\mathcal{Y}$ are not finite, but as we will see in Section 5.3, our proof techniques can easily be extended to this case.

5.2 LDPC Codes are Good for BEQ

To start our study of lossy source coding, we reexamine the BEQ problem, which was first considered in [80]. The following lemma, from [80], gives a useful relationship between good codes for the BEC and good codes for BEQ.

Lemma 5.2.1. *A linear code C with block length n can recover from a particular erasure sequence (under ML decoding) if and only if the dual code $C^\perp$ can quantize the dual sequence, i.e., the sequence where all the erased symbols have been turned into unerased symbols and vice versa. Also, if C can recover from an erasure sequence using message-passing decoding, then $C^\perp$ can quantize the dual sequence using a dualized form of message-passing decoding. (For the details of how to dualize message-passing to work for BEQ, see [80].)*

Based on Lemma 5.2.1, one can show that LDGM codes that are duals of capacity-achieving LDPC codes for the BEC achieve the rate-distortion function for BEQ (at 0

distortion). On the other hand, in [80] the authors prove that low complexity LDPC codes are not effective for this problem. Specifically, they prove that for BEQ, if an LDPC code is capable of quantizing the source so that the distortion is exactly 0 with high probability, then the check degree must be at least $\Omega(\log n)$. This result would suggest that LDPC codes are bad quantizers, because even for the simple distortion in BEQ, the check degree must grow at least logarithmically to achieve good performance.

As we will demonstrate, there is a simple way around the lower bound proved in [80]. The bound in [80] is reminiscent of the $\Omega(\log n)$ lower bound on the check degree for LT codes [75] on the BEC. By combining LT codes with a precode, Raptor codes [110] are able to beat the lower bound for LT codes and achieve constant complexity encoding and decoding per bit. So, let us consider the dual of a Raptor code. Figure 5-1 shows the graph representing a Raptor code, and Figure 5-2 shows the graph representing the dual of a Raptor code.

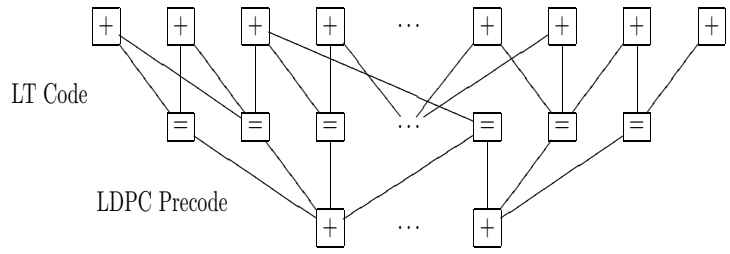

Figure 5-1: Raptor code—this code performs well on the BEC.

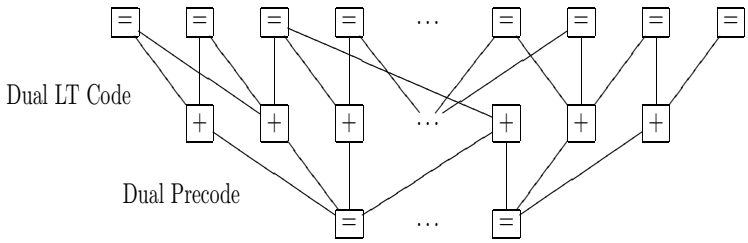

Figure 5-2: Dual Raptor code—this code performs well for BEQ.

Figure 5-2 warrants some explanation. If we define G to be a generator matrix for the LT part of a Raptor code, and H to be a parity check matrix for the precode, then

a Raptor code is the set $\{c|\exists x \text{ s.t. } xG = c, Hx^T = 0\}$. Therefore, the dual is the set $\{y|\exists z \text{ s.t. } Gy^T + (zH)^T = 0\}$. To see this, note that for any vectors c and y satisfying these constraints, $cy^T = xGy^T = x(zH)^T = xH^Tz^T = (zHx^T)^T = 0$. Thus, the set of y's defined by our constraint must be contained in the dual code. Assuming that G and H have full rank, it is easy to see that the dimension of our set of y's is the maximum possible value, which shows that the constraint exactly specifies the dual. Translating the constraint into normal graph notation [42], computing Gy^T can be accomplished by dualizing the LT part of the Raptor code, i.e., switch the variable and check nodes as in the top part of Figure 5-2. [1] To compute $(zH)^T$ we just dualize the precode, i.e., switch the variable and check nodes as in the bottom part of Figure 5-2. Thus, the constraint $Gy^T + (zH)^T = 0$ is exactly captured by the graph in Figure 5-2. Because the dual constraint only asks for some z to exist, to interpret Figure 5-2 properly, the codeword is placed on the top set of variable nodes, and the bottom variables nodes are all erased, i.e., we can set them arbitrarily in order to satisfy the check nodes.

Because Raptor codes are capacity-achieving for the BEC, it follows from Lemma 5.2.1 that the duals of Raptor codes achieving the rate-distortion function for BEQ. Now, imagine redrawing Figure 5-2 with all the variable nodes on the top, i.e., move the variables nodes corresponding to the dual precode to the top of the figure. Then, we see that a dual Raptor code is really just an LDPC code. The important point is that the variable nodes corresponding to the precode are always erased, i.e., a dual Raptor code is an LDPC code where a few variable nodes are always erased. This is analogous to the situation in channel coding. Intuitively, the reason that LT codes must have $\Omega(\log n)$ average degree to achieve low error probability is that otherwise there are always a few variable nodes that are never connected to any checks. The LDPC precode of a Raptor code allows a Raptor code to beat the $\Omega(\log n)$ lower bound because the precode covers every variable node, so there are no longer any

[1] As noted in Chapter 2, for LDGM and LDPC codes drawn using Forney's normal graph notation [42], the dual code can be obtained by swapping the variables nodes with the check nodes. More generally, swapping the variables nodes with the check nodes allows us to compute Gy^T if we do not view the check nodes as constraints, but instead think of them as producing outputs.

isolated variable nodes. In the context of BEQ, the key idea in the proof of the $\Omega(\log n)$ lower bound from [80] on the check degree for LDPC codes is that if the degree is too small, then some check nodes will not be connected to any erased variable nodes, and with probability .5 such check nodes will be unsatisfiable. The dual of the precode allows dual Raptor codes to beat this lower bound because by padding the input with a small number of erased variable nodes, every check node is now guaranteed to be connected to at least one erased variable node. Put differently, just as the LDPC precode in a Raptor code "fixes" the few variable nodes we could not recover by decoding the LT code, the LDGM dual precode of a dual Raptor code fixes the few parity checks we could not satisfy using the dual LT code. Thus, the simple modification of reserving a set of variables nodes in an LDPC code to always be erased, i.e., padding the input with a few erasures, allows LDPC codes to achieve optimal performance for BEQ. This suggests that we should not give up on LDPC codes so quickly, and as we show in the next section, it turns out that LDPC codes can perform quite well for the BSS-HD problem, at least when we assume that an optimal (and potentially computationally complex) quantization algorithm is used.

5.3 LDPC Codes are Good for Hamming Distortion

In this section, we prove that LDPC codes can achieve points arbitrarily close to the rate-distortion bound for the BSS-HD problem. We actually prove a more general statement which roughly says that good channel codes are also good lossy source codes. The proof is based on two simple ideas. First, if a channel code allows reliable transmission, then intuitively it is clear that the typicality regions associated to different codewords cannot overlap significantly. The second idea is to take advantage of the concentration of measure phenomenon in high dimensional spaces. Specifically, we can use isoperimetric inequalities to show that for any code, the distortion is tightly concentrated around its expectation. We note that this approach has been

pursued in [30]. Our results are essentially identical to the results derived in [30], but we state the results in more generality, and we simplify the presentation by using Azuma's inequality instead of the Blowing Up Lemma of [30].

Now, we proceed with the formal argument. The usual proof that rates above $R(D)$ are achievable, as given in [109] or [28], uses a random coding argument. For our purposes, it is useful to instead prove achievability in a manner similar to that of [30]. Roughly speaking, this achievability proof picks a certain channel (corresponding to the minimizing Q in the formula for $R(D)$), and shows that a code coming very close to the capacity of this channel is also a good code for the original rate-distortion problem. The following theorem summarizes our formal result.

Theorem 5.3.1. *Let Q^* denote the minimizing conditional distribution in the formula for $R(D)$. Define the reverse channel $Q' : \mathcal{Y} \to \mathcal{X}$ so that when $X, Y \sim P_X Q^*$, Q' is the conditional distribution of X given Y. Let C be a code for the reverse channel Q' with rate R that has average error probability at most .5. Then, when an optimal quantization algorithm is used (i.e., an algorithm that always encodes the source to the codeword minimizing the distortion), the code C achieves expected average distortion at most $D + 2\sqrt{R(D) - R + f(R)} + o(1)$, where $f(R)$ is a continuous function of R satisfying $\lim_{R \to R(D)} f(R) = 0$, and the $o(1)$ is with respect to n. In the case that the reverse channel is symmetric, we can choose $f(R) = R(D) - R$.*

Before proving Theorem 5.3.1, we note that the theorem easily implies that LDPC codes are good for the BSS-HD problem.

Theorem 5.3.2. *For every $0 \leq \delta \leq .5$ and every $\varepsilon > 0$, there exist LDPC codes of rate $1 - h_b(\delta) + \varepsilon$ achieving expected average distortion δ for the BSS-HD problem. The maximum degree of any node in the graphs of these codes is $O(\log(\frac{1}{\varepsilon}))$.*

Proof. It is well-known that the reverse-channel for the BSS-HD problem with distortion constraint $D = \delta$ is simply BSC(δ). It is also well-known [45, 77] that LDPC codes can achieve rates arbitrarily close to the capacity of BSC(δ) with vanishing probability of error under ML decoding. In particular, Mackay's ensemble [77] gives a family of LDPC codes with rate $1 - h_b(\delta) - \varepsilon$ that achieves probability of error at

most .5 (in fact, the probability of error decays exponentially with the blocklength n), and the maximum degree of any variable node in the Tanner graphs associated with this family of codes is $O(\log(\frac{1}{\varepsilon}))$. (We can also construct codes such that the maximum degree of any node, variable or check, is at most $O(\log(\frac{1}{\varepsilon}))$.) Although [77] does not explicitly state the above bound on degree, the bound follows immediately from the equation right after equation 42 in [77]. Applying Theorem 5.3.1, noting that BSC(δ) is symmetric, so that we may take $f(R) = R(D) - R$, it follows that this family of codes achieves expected average distortion at most $\delta + 2\sqrt{2\varepsilon} + o(1)$, completing the proof. □

Now, we proceed with the proof of Theorem 5.3.1. As alluded to above, this theorem is essentially identical to Thm. 2.3 of [30]. The main difference is that we state the theorem in a form that makes it explicitly clear that the theorem provides a nontrivial bound on the distortion obtained from codes that are not quite capacity-achieving, while the statement in [30] may leave the reader with the impression that the distortion can only be bounded for codes that achieve the capacity of the reverse channel. To prove Theorem 5.3.1, we need the following lemma, which is just a special case of Theorem 7.4.2 from [3], which in turn is a special case of Azuma's inequality.

Lemma 5.3.1. *Let $P_1, \ldots, P_n$ be probability distributions over $\mathcal{X}$, and let $P = \prod_{i=1}^{n} P_i$ be the product probability distribution over $\{0,1\}^n$, i.e., P generates each source symbol independently. Consider a set $S \subset \mathcal{X}^n$, and define*

$$S_\varepsilon = \{x^n \in \mathcal{X}^n : \exists x'^n \in S \text{ such that } x^n \text{ and } x'^n \text{ differ in at most } \varepsilon n \text{ coordinates}\}.$$

If $P(S) \geq e^{-\frac{\varepsilon^2 n}{2}}$, then $P(S_{2\varepsilon}) \geq 1 - e^{-\frac{\varepsilon^2 n}{2}}$.

This lemma captures the isoperimetric inequality/concentration of measure that we need to prove Theorem 5.3.1. In [30], the equivalent lemma is the Blowing Up Lemma. Our lemma is much simpler to prove, and although our lemma does not give the optimal isoperimetric constants that one could obtain by applying, for example, Harper's isoperimetric inequality [56], our slightly cruder estimates are sufficient to prove Theorem 5.3.1.

Proof of Theorem 5.3.1. We prove Theorem 5.3.1 by showing that the typical sets for the codewords in C are approximately disjoint from each other. Then, we use the union of these typical sets as the set S in Lemma 5.3.1 to obtain the Theorem.

Formally, since C has average probability of error at most .5 when used on the reverse channel Q', it follows that for at least one quarter of the codewords $c \in C$, the conditional probability of error given that c is sent is at most .75. We classify these codewords according to their types (see Chapter 2 for a quick review of some basic results in the method of types). Let C' be the code consisting of only those codewords in the type that contains the maximum number of codewords, and let P_Y denote the type of the codewords in C'. Then, C' clearly achieves average probability of error at most .75. Therefore, there exists a channel $\tilde{Q}$ such that $||\tilde{Q}P_Y - Q'P_Y||_1 \leq \frac{1}{\log n}$, and so that conditioned on the event that the empirical channel $\hat{Q}_{X|Y} = \tilde{Q}$, the average probability of error is at most .8. Note that without loss of generality, we may assume that the decoding rule is deterministic, i.e., we can always choose a deterministic ML decoding rule. Performing one more level of expurgation, we see that for at least one tenth of the codewords $c \in C'$, the conditional probability of error given that c is sent and that the empirical channel is $\tilde{Q}$ is at most .9, and that this guarantee is achieved by a deterministic decoding rule. Let C'' be this set of codewords. Then, we have shown that all codewords in C'' have the same type, and for every codeword $c \in C''$, the probability of error given that c is sent and that the empirical channel is $\tilde{Q}$ is at most .9. Also, because the total number of types is at most $(n+1)^{|\mathcal{Y}|}$, we see that $|C''| \geq \frac{|C|}{40(n+1)^{|\mathcal{Y}|}}$.

Let $\hat{P}$ be the type induced by $\tilde{Q}$ and P_Y, i.e.,

$$\hat{P}(x) = \sum_{y \in \mathcal{Y}} P_Y(y)\tilde{Q}(x|y).$$

Fix a codeword $c \in C''$, and let T denote the size of the set $x^n \in \mathcal{X}^n$ such that the conditional distribution $\hat{Q}_{X|Y}$ induced by x^n and c is equal to $\tilde{Q}$. Note that by symmetry, it does not matter which codeword c is chosen since the size of this set only depends on the type, and all codewords in C'' have type P_Y. Now, because

the decoding rule for C'' is deterministic, we conclude that for each $c \in C''$, there is a decoding region $D(c)$ of size at least $.1T$ such that the decoder declares c when the decoder's input is in $D(c)$, and the decoding regions $D(c)$ and $D(c')$ are disjoint whenever $c \neq c'$. Therefore, we conclude that $|\cup_{c\in C''} D(c)| \geq .1T|C''|$. We apply Lemma 5.3.1 with $S = \cup_{c\in C''} D(c)$. Since the source is i.i.d., we choose all the P_i's to be P_X, so that the product distribution P is the distribution of the i.i.d. source X^n. Then, Sanov's theorem implies that

$$P(S) \geq \frac{.1T|C''|}{(n+1)^{|\mathcal{X}||\mathcal{Y}|}} e^{-(D(\hat{P}||P_X)+H(\hat{X}))n},$$

where $H(\cdot)$ denotes entropy and $\hat{X}$ denotes a random variable distributed according to $\hat{P}$. From [28] we know that $T \geq \frac{e^{H(\hat{X}|\hat{Y})}}{(n+1)^{|\mathcal{X}||\mathcal{Y}|}}$, where $\hat{Y}$ is a random variable distributed according to P_Y, and the joint distribution of $\hat{X}$ and $\hat{Y}$ is $\tilde{Q}P_Y$. Thus,

$$P(S) \geq \frac{1}{400(n+1)^{3|\mathcal{X}||\mathcal{Y}|}} e^{(H(\hat{X}|\hat{Y})+R-D(\hat{P}||P_X)-H(\hat{X}))n}.$$

Let $\alpha = -(H(\hat{X}|\hat{Y}) + R - D(\hat{P}||P_X) - H(\hat{X}))$. Then, Lemma 5.3.1 implies that

$$P(S_{2\sqrt{\alpha}+o(1)}) \geq 1 - e^{-\alpha n}.$$

To complete the proof, we must bound α and the expected distortion. First, we bound α. Since $\alpha = I(\hat{X};\hat{Y}) - R + D(\hat{P}||P_X)$, we bound $I(\hat{X};\hat{Y}) - R$ and $D(\hat{P}||P_X)$ separately. Recall two basic facts about the rate-distortion problem. First, the capacity of the reverse channel Q' is the same as the capacity of the channel Q^*. Second, for any DMC, the capacity-achieving output distribution is always unique, and the capacity-achieving output distribution for Q' is P_X. The first fact implies that $I(\hat{X};\hat{Y}) \leq R(D)$, so $I(\hat{X};\hat{Y}) - R \leq R(D) - R$. The second fact allows us to bound $D(\hat{P}||P_X)$. Specifically, let

$$f(R) = \sup_{P'_Y : I(X';Y') \geq R} D(P'_X||P_X),$$

where P'_Y denotes a probability distribution over $\mathcal{Y}$, Y' denotes a random variable distributed according to P'_Y, X' denotes a random variable distributed so that $(X', Y') \sim \tilde{Q}P'_Y$, and P'_X denotes the marginal distribution of X'. The second fact implies that the function f is continuous and $\lim_{R \to R(D)} f(R) = 0$, and the converse to the coding theorem [30] implies that $f + o(1)$ is an upper bound on $D(\hat{P}||P_X)$. In the case of a symmetric reverse channel, we know that the capacity-achieving output distribution is uniform, and that $H(\hat{X}|\hat{Y})$ does not depend on the distribution of $\hat{Y}$. Therefore, in this case, to achieve rate $R = R(D) - g$, we must have $H(\hat{X}) \geq \log(|X|) - g - o(1)$. Thus, $D(X'||X) = \log(|X|) - H(X') \leq g$, which proves that we can take $f(R) = R(D) - R$ in the case of a symmetric reverse channel. In any case, we have shown that $\alpha \leq f(R) + R(D) - R + o(1)$.

Now, we bound the expected distortion. By definition, $E[d(X, Y)] \leq D$, so $E_{\tilde{Q}}[d(X, Y)] \leq D + \frac{1}{\log n}$. Therefore, whenever $x^n \in S$, the distortion is at most $D + \frac{M}{\log n}$, and since the distortion measure is additive and bounded by M, it follows that for all $x^n \in S_{2\sqrt{\alpha}+o(1)}$, the distortion is at most $D + 2M\sqrt{\alpha} + o(1)$. Since the distortion is always at most Mn, we conclude that the expected distortion is at most $D + 2M\sqrt{\alpha} + o(1) + Mne^{-\alpha n}$. □

Comments:

1. The authors of [82] analyze the performance of MacKay's ensemble for BSS-HD. One might hope for a more direct proof that LDPC codes are good for quantization by strengthening their approach. We briefly explain why their approach is not sufficient to prove that LDPC codes with sublogarithmic degree can come close to the rate-distortion bound. The authors of [82] provide a fairly complicated proof, but the same result can be derived using Chebyshev's inequality and bounding the covariance term. However, even with this method, choosing codes from MacKay's ensemble still gives a logarithmic degree bound. Intuitively, the reason MacKay's ensemble does not give a better bound is that it takes $n \ln n$ steps before a random walk on a hypercube has mixed well enough for the covariance term to be small.

A more fundamental limitation of the proof in [82] is that this proof is based on trying to lower bound the probability that all (or almost all) possible syndromes can be written using only a few columns of the parity check matrix. This approach makes it impossible to prove any bound of better than logarithmic degree for MacKay's ensemble. Specifically, if we want to be able to write all syndromes, the matrix we look at must have full rank. However, one can easily show that if we use a constant (or sublogarithmic) degree in MacKay's ensemble, then with high probability the matrix will have a row where every entry is 0. Thus, with high probability the matrix does not have full rank. The probability of every row having at least one entry set to 1 does not become large until the degree becomes logarithmic, which explains why the analysis in [82] only produces a logarithmic degree bound.

2. The authors of [82] observe that for dense parity check matrices, convergence to the optimal distortion is exponential, i.e., the probability that the realized distortion is larger than the expected distortion by any fixed constant decays exponentially with the blocklength. Lemma 5.3.1 shows that this exponential convergence is actually a property of any code. However, in the context of rate-distortion, we can actually ask for something stronger. Specifically, there exist codes that come close to the rate-distortion function for quantization under a Hamming distortion in the worst-case, i.e., the covering radius can be made close to the optimal distortion. To see this, say we have a code with expected distortion δ. From Lemma 5.3.1, with probability at most $\frac{1}{n}$ the observed distortion is greater than $\delta + \sqrt{2n\ln(n)}$. So, to create a worst-case quantizer, repeatedly try to quantize a given source by adding independent, uniformly chosen strings to the given source. A simple union bound argument shows that after n trials, with high probability the worst-case distortion is at most $\delta + \sqrt{2n\ln(n)}$. This only increases the rate by $\frac{\log n}{n}$, so we see that asymptotically we achieve the same distortion and rate as before, but now with a worst-case guarantee.[2] If we

[2] As noted in the proof of Theorem 5.4.1, we can assume without loss of generality that n is as large because we can always repeat a code without changing its performance, so we can get into the asymptotic regime where $\frac{\log n}{n}$ becomes negligible.

restrict ourselves to linear codes, the arguments in [6] show that we can append a few extra columns (or rows for LDGM codes) so that the resulting linear code works well in the worst-case. The basic idea is that by adding $O(\log n)$ suitably chosen columns, we can "fix" the original code to cover the small fraction of words that previously would have been quantized with high distortion. The drawback of this method is that the resulting code may no longer be sparse, i.e., the matrix may now have $O(n \log n)$ nonzero entries instead of $O(n)$ entries.

We close this section by observing that Theorem 5.3.1 can be extended to the problem of quantizing a Gaussian source under mean-square distortion. The argument is essentially the same, except that instead of Lemma 5.3.1, we apply the Gaussian isoperimetric inequality [114].

Lemma 5.3.2 (Gaussian Isoperimetric Inequality)**.** *Let X^n be distributed i.i.d. $\mathcal{N}(0,1)$, and for an arbitrary (Lebesgue measurable) set $S \subset R^n$, let S_ε denote the set of points whose Euclidean distance to S is at most $\varepsilon\sqrt{n}$. Then, for all positive δ and ε,*

$$\operatorname*{argmin}_{S:\Pr[X^n \in S] \geq \delta} \Pr[X^n \in S_\varepsilon]$$

is a halfspace chosen so that $\Pr[X^n \in S] = \delta$.

Theorem 5.3.3. *Let X^n be a source whose components are i.i.d. $\mathcal{N}(0,1)$. Let C be a code of rate R for the additive white Gaussian noise (AWGN) channel with noise variance D that has average error probability at most .5. Then, when an optimal quantization algorithm is used (i.e., an algorithm that always encodes the source to the codeword minimizing the distortion), the code C achieves expected average mean square distortion per symbol of at most $D + 2\sqrt{R(D) - R} + o(1)$.*

Proof. The proof proceeds along similar lines to the proof of Theorem 5.3.1, so we will be brief. Because a Gaussian source is continuous, we cannot use types the way we did for discrete memoryless sources. However, because the reverse channel for a Gaussian source under mean-square distortion is the AWGN channel, we can take advantage of spherical symmetry. In particular, we replace the notion of type with

the power of the codewords, and then quantize the power into a suitable number of bins. This allows us to define decoding regions $D(c)$ as before, and we obtain a similar formula for α. Since the Gaussian channel is symmetric, the same method of bounding $f(R)$ applies. Now, from Lemma 5.3.2, we see that if the probability of a set S under the Gaussian measure is $e^{-\alpha n}$, then the probability of $S_{\sqrt{8\alpha}}$ is at least $1 - e^{-\alpha n}$, so we obtain the required bound on the distortion. □

5.4 A Lower Bound on the Degree of LDGM Codes

Now, we move on to the problem of proving lower bounds. In this section, we show that LDGM codes cannot come arbitrarily close to the rate-distortion bound for the BSS-HD problem unless the average variable node degree becomes unbounded. Formally, we prove the following theorem.

Theorem 5.4.1. *Let d be the average degree of the variable nodes in some LDGM code. Let R be the rate of the code, and let δ be the expected distortion. Define $\varepsilon = \delta - \delta_{opt}$, where δ_{opt} is the optimal distortion for rate R, i.e., $R + h_b(\delta_{opt}) = 1$. Then, $d = \Omega(\log \frac{1}{\varepsilon})$.*

Note that Theorem 5.4.1 applies to any code, not codes drawn from some specific random ensemble. Before proving Theorem 5.4.1, we state the following lemma, which is closely related to Lemma 5.3.1, and is also just a special case of Theorem 7.4.2 from [3].

Lemma 5.4.1. *Let $P_1, \ldots, P_n$ be probability distributions over $\mathcal{X}$, and let $P = \prod_{i=1}^{n} P_i$ be the product probability distribution over $\mathcal{X}^n$, i.e., P generates each source symbol independently. Let $C \subset \mathcal{Y}^n$ be a code, and let D be the distortion when X^n is quantized to the codeword in C that minimizes the distortion. Then, $\Pr[D - E[D] > \varepsilon M] \leq e^{-\frac{\varepsilon^2 n}{2}}$.*

Thus, for any code, the distortion is tightly concentrated.

Proof of Theorem 5.4.1. This is a simple application of Lemma 5.4.1. Consider a code with rate R and average variable node degree d. We prove Theorem 5.4.1 by

proving a lower bound on ε in terms of d. Let D be a random variable denoting the realized distortion. From Lemma 5.4.1, we know that $\Pr[D > \delta_{opt} + 2\varepsilon] < e^{-\frac{\varepsilon^2 n}{2}}$. Thus, since the source is uniform, at least $2^n - 2^{(1-\frac{\log(e)\varepsilon^2}{2})n} \doteq 2^n$ words have distortion less that $\delta_{opt} + 2\varepsilon$.

Let ε_1 be a parameter that we set later. If we flip $\frac{\varepsilon_1 n}{2d}$ information bits of the code, and these bits correspond to variable nodes with degree $\leq 2d$, then the triangle inequality implies that the distortion increases by at most $\varepsilon_1 n$. Note that if the average variable node degree of the code is d, then it follows that at least half of the variable nodes have degree at most $2d$, so we do have quite a bit of freedom in our choice of which $\varepsilon_1 n$ bits to flip. We take advantage of this freedom by counting the number of (codeword, source) pairs such that the relative distance between the codeword and the source is $\leq \delta_{opt} + 2\varepsilon + \varepsilon_1$ in two different ways.

First, from the point of view of codewords, the number of such pairs is clearly $2^{Rn}\mathrm{Vol}(n, (\delta_{opt}+2\varepsilon+\varepsilon_1)n)$, where $\mathrm{Vol}(n, d)$ denotes the number of words in a Hamming ball of radius d in an n-dimensional space. We already observed that almost every source word has a codeword within distance $\delta_{opt}+2\varepsilon$. Therefore, using our observation about flipping information bits, the number of such pairs is at least $2^n\mathrm{Vol}(\frac{Rn}{2}, \frac{\varepsilon_1 n}{2d})$. Now, we may assume without loss of generality that n is as large as we like. This is because if we are given a code C with block length n_0, we can construct a code with block length Kn_0 with the same rate, average check degree, and expected distortion, where K is any integer greater than 1. This code is simply the repetition of C K times, i.e., the code quantizes a source of length Kn_0 by splitting the source into K blocks of length n_0 and applying the code C separately to each block. Therefore, we consider the limit as $n \to \infty$, so that the exponent becomes the important term in each of the two ways of counting (codeword, source) pairs. Specifically,

$$2^{Rn}\mathrm{Vol}\left(n, \left(\delta_{opt} + 2\varepsilon + \varepsilon_1\right) n\right) \doteq 2^{(R+h_b(\delta_{opt}+2\varepsilon+\varepsilon_1))n}$$

and

$$2^n\mathrm{Vol}\left(\frac{Rn}{2}, \frac{\varepsilon_1 n}{2d}\right) \doteq 2^{(1+.5Rh_b(\frac{\varepsilon_1}{Rd}))n}.$$

Therefore, for sufficiently large n, the condition

$$2^{Rn}\mathrm{Vol}\left(n,\left(\delta_{opt}+2\varepsilon+\varepsilon_1\right)n\right)\geq 2^n\mathrm{Vol}\left(\frac{Rn}{2},\frac{\varepsilon_1 n}{2d}\right)$$

implies that $R+h_b(\delta_{opt}+2\varepsilon+\varepsilon_1)\geq 1+.5Rh_b(\frac{\varepsilon_1}{Rd})$, provided that $\delta_{opt}+2\varepsilon+\varepsilon_1\leq .5$.

To complete the proof, observe that $h_b()$ is a concave function, so $h_b(\delta_{opt}+2\varepsilon+\varepsilon_1)\leq h_b(\delta_{opt})+h_b'(\delta_{opt})(2\varepsilon+\varepsilon_1)$. This means that $h_b'(\delta_{opt})(2\varepsilon+\varepsilon_1)\geq .5Rh_b(\frac{\varepsilon_1}{Rd})$, i.e., $\varepsilon\geq\frac{1}{2}(\frac{.5Rh_b(\frac{\varepsilon_1}{Rd})}{h_b'(\delta_{opt})}-\varepsilon_1)$. Now, we optimize over ε_1. Taking the derivative with respect to ε_1, we find that

$$\frac{d}{d\varepsilon}\left(\frac{.5Rh_b(\frac{\varepsilon_1}{Rd})}{h_b'(\delta_{opt})}-\varepsilon_1\right) \;=\; \frac{.5}{dh_b'(\delta_{opt})}\log\left(\frac{Rd}{\varepsilon_1}-1\right)-1.$$

Setting this expression to 0, we find that the optimal value is $\varepsilon_1^*=\frac{Rd}{1+2^{2dh_b'(\delta_{opt})}}$. Assume for the moment that this value of ε_1^* satisfies the inequality $\delta_{opt}+2\varepsilon+\varepsilon_1^*\leq .5$, so that ε_1^* is a valid value of ε_1. Then, substituting in ε_1^*, we find that

$$\begin{aligned}
\varepsilon &\geq \frac{1}{2}\left(\frac{.5Rh_b(\frac{\varepsilon_1^*}{Rd})}{h_b'(\delta_{opt})}-\varepsilon_1^*\right)\\
&= \frac{1}{2}\left(\frac{.5R}{h_b'(\delta_{opt})}\frac{1}{1+2^{2dh_b'(\delta_{opt})}}\log\left(1+2^{2dh_b'(\delta_{opt})}\right)\right.\\
&\quad \left.+\frac{.5R}{h_b'(\delta_{opt})}\frac{2^{2dh_b'(\delta_{opt})}}{1+2^{2dh_b'(\delta_{opt})}}\log\left(\frac{1+2^{2dh_b'(\delta_{opt})}}{2^{2dh_b'(\delta_{opt})}}\right)-\frac{Rd}{1+2^{2dh_b'(\delta_{opt})}}\right)\\
&\geq \frac{.5R}{2h_b'(\delta_{opt})}\frac{2^{2dh_b'(\delta_{opt})}}{1+2^{2dh_b'(\delta_{opt})}}\log\left(1+2^{-2dh_b'(\delta_{opt})}\right)\\
&\geq \frac{R}{4h_b'(\delta_{opt})}\frac{\left(1-2^{-1-2dh_b'(\delta_{opt})}\right)}{1+2^{2dh_b'(\delta_{opt})}}\\
&\geq \frac{R}{8h_b'(\delta_{opt})}\frac{1}{1+2^{2dh_b'(\delta_{opt})}}.
\end{aligned}$$

To obtain the second to last inequality, we have used the inequalities $x\ln(1+\frac{1}{x})\geq 1-\frac{1}{2x}$ and $\log(e)>1$. In the last inequality, we have used the fact that $h_b'(\delta_{opt})$ is nonnegative, so the first term in the numerator is at least $\frac{1}{2}$. If we invert the last

inequality, we see that $d \geq \frac{\log\left(\frac{R}{8h_b'(\delta_{opt})\varepsilon}-1\right)}{2h_b'(\delta_{opt})}$, proving that d must be $\Omega(\log\left(\frac{1}{\varepsilon}\right))$.

Finally, we must address the case that ε_1^* is invalid, i.e., $\delta_{opt} + 2\varepsilon + \varepsilon_1^* \geq .5$. For $\varepsilon < .25(.5 - \delta_{opt})$, this implies that $\varepsilon_1^* \geq .25 - .5\delta_{opt}$. Now, there exists a constant d_0 depending only on δ_{opt} such that for $d > d_0$, $\varepsilon_1^* < .25 - .5\delta_{opt}$, so we will be done if we can show that if there exists a code with average degree d achieving distortion $\delta_{opt}+\varepsilon$, then there exists a code with average degree larger than d_0 that achieves at most the same distortion. But this is clear because we can always add, say $2d_0$ variable nodes that are connected to every single check node, and hence make the average degree greater than d_0. Adding variable nodes clearly cannot increase the distortion, and applying the repetition trick used previously, we see that we can make the additional rate $\frac{2d_0}{n}$ as small as we want, so the proof is complete. □

Discussion: This bound is tight in the sense that $\log(\frac{1}{\varepsilon})$ is the correct scaling of the average degree as a function of the gap to the rate-distortion bound. In more detail, one can show that there exists LDGM codes within ε of the rate distortion bound and with maximum variable node degree $d = O(\log(\frac{1}{\varepsilon}))$. One way to see this is to consider a dualized form of MacKay's ensemble. [3] Alternatively, we can use the simpler Poisson ensemble considered in [79]. This ensemble will have a few nodes with large (logarithmic) degree, but a simple argument combining Lemma 5.3.1 with an expurgation argument where we delete the few variable nodes with large degree shows that if $d = O(\log(\frac{1}{\varepsilon}))$, then the gap can be reduced to ε.

In another sense, the bound in Theorem 5.4.1 is quite loose. The bound says $\varepsilon = \Omega(\frac{1}{2^{\alpha d}})$ for some constant α, while the random ensembles described above have a much larger value of ε. (Note : we can optimize the bound without using the simplification of approximating h_b by a first order Taylor series expansion. However, this does not improve the bound significantly.)

[3] By "dualized" form of MacKay's ensemble, we mean an ensemble that selects each row of the generator matrix independently using a random walk. This is not quite the dual of MacKay's ensemble, since the dual would select each column independently. However, selecting rows independently is more convenient since this guarantees a bounded maximum variable degree. The proof that the dualized form of MacKay's ensemble works is similar to [79], but involves quite a bit of algebra.

5.5 A Lower Bound on the Degree of LDPC Codes

To complete the picture, we now show a similar lower bound for LDPC codes.

Theorem 5.5.1. *Let d be the average degree of the check nodes in some LDPC code. Let $R > 0$ be the rate of the code, and let δ be the expected distortion. Define $\varepsilon = \delta - \delta_{opt}$, where δ_{opt} is the optimal distortion for rate R, i.e., $R + h_b(\delta_{opt}) = 1$. Then, $d = \Omega(\log(\frac{1}{\varepsilon}))$.*

As with LDGM codes, in some sense this bound is tight, since Theorem 5.3.2 shows that MacKay's ensemble achieves a gap of ε to the rate-distortion bound with maximum variable node degree bounded by $O(\log(\frac{1}{\varepsilon}))$.[4]

Before going into the details, we summarize the main ideas of the proof of Theorem 5.5.1. Our proof is similar to the well-known proof from [45] that LDPC codes cannot come arbitrarily close to the capacity of the BSC with uniformly bounded degree. Specifically, let H denote the parity check matrix of the code under consideration, and let w be a random vector whose components are generated i.i.d. with probability δ of being 1 and probability $1-\delta$ of being 0. Then, assuming that the check nodes have bounded degrees, the syndrome Hw does not look close to uniform—in particular, the syndrome has expected (normalized) Hamming weight less than .5. Next, we apply Azuma's inequality to bound deviations from the mean, and this allows us to count the number of syndromes that can be realized by inputs with normalized Hamming weight δ. In particular, the number of syndromes is substantially smaller than the number of syndromes we would need for an optimal rate-distortion code, and it is this gap in the number of syndromes that allows us to prove that the average degree of the check nodes must grow as $\Omega(\log(\frac{1}{\varepsilon}))$.

Now we fill in the details to complete the proof sketch above. We start with two simple lemmas.

Lemma 5.5.1. *Let H be an $m \times n$ parity check matrix for some code such that the average Hamming weight of the rows of H, i.e., the average check degree of the code,*

[4] As for LDGM codes, we can also delete the few check nodes with large degree to get codes with maximum variable node and check node degree bounded by $O(\log(\frac{1}{\varepsilon}))$. Note that deleting check nodes cannot increase the distortion.

is d. Let $w \in \{0,1\}^n$ *be a vector whose coordinates are generated i.i.d. with probability* δ *of being* 1 *and probability* $1-\delta$ *of being* 0*. Then,*

$$E[|Hw|_H|] < m\frac{1-(1-2\delta)^d}{2}.$$

Proof. Let w_i be the Hamming weight of the i^{th} row of the parity check matrix. The expected Hamming weight is

$$\sum_{i=1}^{m} \frac{1-(1-2\delta)^{w_i}}{2}.$$

Applying Jensen's inequality, we see that this sum is upper bounded by $m\frac{1-(1-2\delta)^d}{2}$.

□

Note that for a random linear code, the expected Hamming weight would be $\frac{m}{2}$, so we see that the expected Hamming weight is noticeably smaller for a code with low average check degree. This is the key fact in our proof.

Our second preliminary lemma, Lemma 5.5.2, is like Lemmas 5.3.1 and 5.4.1 in that it is also a special case of Theorem 7.4.2 from [3].

Lemma 5.5.2. *Let* H *be an* $m \times n$ *parity check matrix for some code, and assume that every column of* H *contains at most* c 1*'s. Let* $w \in \{0,1\}^n$ *be a vector whose coordinates are generated i.i.d. with probability* δ *of being* 1 *and probability* $1-\delta$ *of being* 0*. Then,*

$$\Pr[||Hw|_H - E[|Hw|_H]| > \varepsilon] \leq 2e^{-\frac{\varepsilon^2 n}{2c^2}}.$$

Now, we combine Lemmas 5.5.1 and 5.5.2.

Lemma 5.5.3. *Let* H *be an* $m \times n$ *parity check matrix for some code such that the average Hamming weight of the rows of* H *is* d*. Let* $w \in \{0,1\}^n$ *be a vector whose coordinates are generated i.i.d. with probability* δ *of being* 1 *and probability* $1-\delta$ *of being* 0*. Then, for any* $0 < \varepsilon, f < 1$*, there exists a set* $S_{f,\varepsilon} \subset \{0,1\}^m$ *with size at*

most $2^{fn}\,Vol(m(\frac{1-(1-2\delta)^d}{2}+\varepsilon))$ *such that*

$$\Pr[Hw \notin S_{f,\varepsilon}] \leq 2e^{-\frac{\varepsilon^2 nf^2}{2(1-f)d^2(1-R)^2}}.$$

Proof. We may assume without loss of generality that the parity check matrix has been permuted so that the columns are sorted in descending order of Hamming weight. Let the columns in sorted order be $c_1, \ldots, c_n$. The first fn columns $c_1, \ldots, c_{fn}$ induce at most 2^{fn} possible syndromes, i.e., there are at most 2^{fn} syndromes formed by taking linear combinations of only the first fn columns. We denote these syndromes by $s_1, \ldots, s_{2^{fn}}$.

We define the set $S_{f,\varepsilon}$ as follows. Let $B(x,d)$ to be the Hamming ball of radius d around x. Then, we define

$$S_{f,\varepsilon} = \bigcup_{i=1}^{2^{fn}} B\left(s_i, m\left(\frac{1-(1-2\delta)^d}{2}+\varepsilon\right)\right).$$

Note that $S_{f\varepsilon}$ contains at most $2^{fn}\mathrm{Vol}(m, m(\frac{1-(1-2\delta)^d}{2}+\varepsilon))$ strings, as required.

To complete the proof, we show that with high probability, the syndrome lands inside the set S. Observe that for the syndrome to land outside of the set S, it is necessary that the syndrome induced by the columns $c_{1+fn}, \ldots, c_n$ has Hamming weight at least $m(\frac{1-(1-2\delta)^d}{2}+\varepsilon)$, because otherwise the syndrome of the whole word will be close to the s_i corresponding to the induced syndrome for the first fn columns. Thus, it suffices to bound the probability that the syndrome induced by the columns $c_{1+fn}, \ldots, c_n$ has Hamming weight at least $m(\frac{1-(1-2\delta)^d}{2}+\varepsilon)$. This probability can be bounded easily by applying Lemmas 5.5.1 and 5.5.2 to the submatrix H' of H induced by columns $c_{1+fn}, \ldots, c_n$. First, since we have thrown away some columns, it is clear that the average weight of the rows of H' is at most the average weight of rows in H. Lemma 5.5.1 implies that $E[|H'w|_H] \leq nh_b(\delta_{opt})\frac{1-(1-2\delta)^d}{2}$. Next, observe that every column of H' has weight at most $\frac{d(1-R)}{f}$. To see this, note that average weight of the rows of H is d, so the average weight of the columns of H is $d(1-R)$. Thus, at most a fraction f of the columns of H can have weight larger that $\frac{d(1-R)}{f}$, which means

that columns $c_{1+fn}, \ldots, c_n$ have weight at most $\frac{d(1-R)}{f}$.

Because the column weight is bounded, we can apply Lemma 5.5.2. Lemma 5.5.2 implies that

$$\Pr[H'w|_H - E[|H'w|_H] > \lambda] \leq 2e^{-\frac{\lambda^2 f^2}{2(1-f)nd^2(1-R)^2}}.$$

Combining this with our bound on $E[|H'w|_H$, we see that

$$\Pr[Hw \notin S] \leq 2e^{-\frac{\varepsilon^2 n f^2}{2(1-f)d^2(1-R)^2}}. \quad \square$$

We use Lemma 5.5.3 to count the number of syndromes induced by vectors with Hamming weight at most δn. First, we recall an elementary bound on binomial coefficients.

Lemma 5.5.4. *Let w be generated i.i.d. with probability δ of being 1. Assume that δn is an integer. Then, $\Pr[|w|_H = \delta n] = \sqrt{\frac{1}{2\pi\delta(1-\delta)n}}(1 + o(1))$.*

Proof. Apply Stirling's approximation $n! = n^n e^{-n}\sqrt{2\pi n}(1 + o(1))$ to the expression

$$\binom{n}{\delta n}(\delta n)^{\delta n}((1-\delta)n)^{(1-\delta)n} = \frac{n!}{(\delta n)!((1-\delta)n)!}(\delta n)^{\delta n}((1-\delta)n)^{(1-\delta)n}.$$

$\square$

Lemma 5.5.5. *Let H be an $m \times n$ parity check matrix for some code such that the average Hamming weight of the rows of H is d, and let f, ε, and $S_{f,\varepsilon}$ be as in Lemma 5.5.3. Then, the number of syndromes induced by vectors $w \in \{0,1\}^n$ with weight $\leq \delta n$ that are outside $S_{f,\varepsilon}$ is at most*

$$(1 + \delta n)2^{h_b(\delta)n}\sqrt{2\pi\delta(1-\delta)n}e^{-\frac{\varepsilon^2 n f^2}{2(1-f)d^2(1-R)^2}}.$$

Proof. Consider the distribution on w analyzed in Lemma 5.5.3, i.e., the distribution where each bit of w is generated i.i.d. with probability δ of being 1. Lemma 5.5.4 says that

$$\Pr[|w|_H = \delta n] = \sqrt{\frac{1}{2\pi\delta(1-\delta)n}}(1 + o(1)).$$

Combining this with Lemma 5.5.3, we see that

$$\Pr[Hw \notin S_{f,\varepsilon} | |w|_H = \delta n] \leq 2\sqrt{2\pi\delta(1-\delta)n}e^{-\frac{\varepsilon^2 n f^2}{2(1-f)d^2(1-R)^2}}.$$

Now, since each bit of w is generated i.i.d., it is obvious that conditioned on the event $\{|w|_H = \delta n\}$, w is distributed uniformly over all strings with weight δn. Therefore, the probability above is just the fraction of inputs with weight δn whose induced syndromes are outside the set $S_{f,\varepsilon}$. Using the inequality $\binom{n}{\delta n} \leq 2^{h_b(\delta)n}$, we see that the number of syndromes outside of S that can be represented by inputs with weight exactly δn is at most

$$2^{h_b(\delta)n}\sqrt{2\pi\delta(1-\delta)n}e^{-\frac{\varepsilon^2 n f^2}{2(1-f)d^2(1-R)^2}}.$$

To complete the proof, note that we can repeat the analysis for all weights less than δn as well. Specifically, Lemma 5.5.3 holds for all δ, and the bound above is monotonically increasing in δ for $\delta < .5$. Thus, we see that the number of syndromes outside of S that are representable by w such that $|w|_H \leq \delta n$ is

$$(1+\delta n)2^{h_b(\delta)n}\sqrt{2\pi\delta(1-\delta)n}e^{-\frac{\varepsilon^2 n f^2}{2(1-f)d^2(1-R)^2}}.$$

□

Proof of Theorem 5.5.1. To prove Theorem 5.5.1, we essentially just have to set f and ε appropriately in Lemma 5.5.5. As in the proof of Theorem 5.4.1, note that we can assume without loss of generality that n is as large as we want because repetition preserves the rate, average degree, and expected distortion. Thus, the important part of the bounds in Lemmas 5.5.3 and 5.5.5 is the exponent.

From Stirling's approximation and Lemma 5.5.3, we know that size of $S_{f,\varepsilon}$ is at most $n2^{fn+nh_b(\frac{1-(1-2\delta)^d}{2}+\frac{\varepsilon}{1-R})}$. The exponent of this expression is

$$fn + nh_b\left(\frac{1-(1-2\delta)^d}{2} + \frac{\varepsilon}{1-R}\right).$$

The exponent for the expression in Lemma 5.5.5 is

$$n\left(h_b(\delta) - \frac{\varepsilon^2 f^2}{2(1-f)d^2(1-R)^2}\right).$$

If both of these exponents are less than $n(1-R)$, then it follows that for sufficiently large n, a negligible fraction of the syndromes are represented by inputs with weight at most δn. Thus, with probability going to 1 the (normalized) distortion is bigger than δ, so the expected distortion is also bigger than δ.

Now, to make the first exponent sufficiently small, we need to choose ε small enough so that the balls we draw around each s_i have normalized radius less than .5. We choose $\varepsilon = (1-R)\frac{(1-2\delta)^d}{4}$. We set $f = \frac{1}{8}(\frac{(1-2\delta)^d}{4})^2$. Then, using the inequality $h_b(\frac{1}{2} - x) \leq 1 - 4x^2$,[5] we see that the first exponent is in fact less than $n(1-R)$. To make the second exponent sufficiently small, we need

$$h_b(\delta) - \frac{\varepsilon^2 f^2}{2(1-f)d^2(1-R)^2} < (1-R).$$

Since $h_b(\cdot)$ is concave, it follows that $h_b(\delta) < 1 - R + h_b'(\delta_{opt})(\delta - \delta_{opt})$, so it suffices to choose δ such that

$$h_b'(\delta_{opt})(\delta - \delta_{opt}) - \frac{\varepsilon^2 f^2}{2(1-f)d^2(1-R)^2} < 0.$$

Using our expressions for ε and f, it is easily verified that

$$\delta < \delta_{opt} + \frac{(1-2\delta_{opt})^{6d}}{2^{19}d^2(h_b'(\delta_{opt}) + \frac{6d}{1-2\delta_{opt}})}$$

is a valid choice for δ. Thus, the expected distortion for any code is lower bounded by the quantity on the right hand side of the last inequality. Inverting this inequality, we see that $d = \Omega(\log(\frac{1}{\varepsilon}))$, completing the proof. □

[5]This inequality is easily derived using Taylor series.

5.6 Conclusion

In the first part of this chapter (Sections 5.2 and 5.3), we proved that LDPC codes can be used for certain rate-distortion problems. First, we proved that a slight modification allows one to construct LDPC codes that are good for BEQ. Next, we improved the results of [82] by showing that if an optimal encoding algorithm is used, then LDPC codes can come arbitrarily close to the rate-distortion function for the BSS-HD problem. Our proof also shows that for many rate-distortion problems, any channel code that works well for the reverse channel will be good for the associated rate-distortion problem, provided that an optimal quantization algorithm is used. The key ingredient in the proof is an appropriate isoperimetric inequality. For our purposes, Azuma's inequality was sufficient for discrete sources, while the Gaussian isoperimetric inequality was used for the case of quantizing a Gaussian source under mean-square distortion. These isoperimetric inequalities allow us to show that in high dimensions, good sphere-packings, i.e., good channel codes, are also good coverings, i.e., good lossy source codes.

In the second half of this chapter (Sections 5.4 and 5.5), we proved lower bounds on the average degree for LDGM and LDPC codes, and our lower bounds are within a constant factor of the upper bounds we computed in the first half of the chapter. In particular, our results imply that the compound construction from [79] achieves lower average degree than LDGM codes or LDPC codes can achieve individually. Thus, it seems that combining LDGM codes and LDPC codes could potentially lead to more efficient algorithms for the BSS-HD problem. For the special case of the BEC, this is already known to be the case [58, 96].

Probably the most important problem related to this chapter is developing efficient algorithms for quantization. Given the slow rate of growth for our bounds on average degree, i.e., $d = \Theta(\log(\frac{1}{\varepsilon}))$, even algorithms for LDGM or LDPC codes could be very useful. One could argue that from a practical point of view that the BSS-HD problem is essentially solved. For example, convolutional codes with a relatively short constraint length already come quite close to the rate-distortion bound. Even

short random codes converge to the rate-distortion bound quickly. However, from a theoretical point of view, developing algorithms for BSS-HD with complexity per bit that provably scales subexponentially as a function of $\frac{1}{\varepsilon}$, where ε denotes the gap to the rate-distortion bound remains an open problem, although there has been some recent progress [68].

Chapter 6

Code Constructions for Wiretap Channels

In this chapter, we consider code constructions for secure communication. Security was first considered from an information-theoretic perspective in the classic paper by Shannon [108]. One of the key results in [108] is that from an information-theoretic perspective, the one-time pad is an optimal strategy for transmitting a secret message from a transmitter to a receiver. From a practical standpoint, this result poses a major problem, because it means that in order for a transmitter to send n bits to a receiver securely, the transmitter and receiver must already share a key of n (uniformly random) bits that is kept secret from the eavesdropper. Thus, the results from [108] essentially say that to transmit a message of n bits securely over a channel in the presence of an eavesdropper, the transmitter and receiver must meet in private and agree to a secret key of n bits, which means that the transmitter and receiver might as well exchange the message while they meet in private and not use the channel at all.

In many introductory treatments of cryptography, this result is used as motivation for weakening the notion of security. Specifically, instead of requiring secure transmission against an arbitrary eavesdropper, one can place the constraint that the eavesdropper has limited computational power. Exploring this model turns out to be very fruitful. Assuming that certain widely believed (but unproven) claims in

computational complexity are true, e.g., the existence of one-way functions and trapdoor functions, it is possible to implement many useful cryptographic protocols such as public key cryptography, pseudorandom generators, signatures, etc. Of course, these schemes are only secure against computationally bounded eavesdroppers, and the security proofs rely on unproven assertions in computational complexity theory.

As an alternative to placing computational constraints on the eavesdropper, Wyner introduced the wiretap channel as a new way of constraining the eavesdropper [120]. In the wiretap channel, we assume that communication occurs over a broadcast channel. Specifically, the transmitter can send information over a (potentially noisy) channel to the intended receiver. The constraint on the eavesdropper is that the eavesdropper only gets to see a noisy version of what the transmitter sends to the intended receiver, i.e., although there may be a noisy channel between the transmitter and the intended receiver, there is also a noisy channel between the transmitter and the eavesdropper. The motivation for such a model is that at the physical layer, bits are sent over noisy channels, so rather than abstract away the channel noise by assuming that channel coding has been used to create a clean channel, we may be able to take advantage of the inherently noisy nature of the channel to accomplish secure transmission. In [120], Wyner analyzed some simple channel models and showed that when the channel between the transmitter and the eavesdropper is "noisier" than the channel between the transmitter and the intended receiver, it is possible to construct coding strategies that allow the transmitter and intended receiver to exchange messages at a positive rate (measure in terms of bits/channel use) in such a way that the eavesdropper learns very little information about the sent message. The notion of security proposed by Wyner is rather weak, however. In [83] and [29] a stronger notion of security was proposed, and it was shown in [83] that the maximum achievable rate for this stronger notion of security is identical to the maximum achievable rate for Wyner's weaker notion of security, so there is no rate loss associated with using the stronger notion of security. In Section 6.2, we describe the wiretap channel and the associated security criteria more formally.

In recent years, there has been renewed interest in the wiretap channel, as well as

several other models of information-theoretic security. For example, in the context of biometrics and secret-key generation, [81] presents some methods for securely extracting a shared secret when an eavesdropper has side information about the key. This line of work also generalizes the notion of security by considering various notions of security, some weaker and some stronger in the sense of giving the eavesdropper stronger attack capabilities, e.g., [33, 34]. Also, several code constructions have been proposed for specific wiretap channel models. For example, [105, 115] construct codes for a wiretap channel where the channel between the transmitter and the intended receiver is noiseless and the channel between the transmitter and the eavesdropper is a BEC, or a binary input additive white Gaussian noise channel. However, these papers only consider the weak notion of security proposed by Wyner, and the complexity of the proposed encoding algorithms is potentially $O(n^2)$, as it depends on the encoding algorithm for LDPC codes proposed in [101], and it is unclear that appropriate degree distributions can be designed to so that the resulting codes can be encoded in subquadratic time by the algorithm from [101] while still providing even weak security guarantees. Other than in the special case of the BEC, the proposed codes do not achieve the maximum possible rate for secure communication.

6.1 Summary of Results

In this chapter, we start by describing the wiretap channel and the associated security criteria considered in [120] and [83]. Then, we introduce a new notion of security which is slightly stronger that the notion considered in [83]. We call this new notion of security semantic security, because of the close connection between our notion and the notion of semantic security [49] widely used in computational cryptography. Next, we introduce a new channel model that we call the constant-capacity compound wiretap channel. This channel model is a special case of the compound wiretap channel introduced in [71]. Roughly speaking, the constant-capacity compound wiretap channel attempts to address one drawback of the standard wiretap channel, namely that the standard wiretap channel assumes that the transmitter has

complete knowledge of the channel between himself and the eavesdropper. In practical situations, it seems unlikely that the transmitter can obtain this information, and the constant-capacity compound wiretap channel attempts to address this concern by assuming that the transmitter only knows the capacity (or at least an upper bound on the capacity) of the channel connecting himself to the eavesdropper. We define the constant-capacity compound wiretap channel more formally in Section 6.2. We believe that finding coding strategies that provide security under weaker constraints on the channel connecting the transmitter to the eavesdropper than those imposed by the constant-capacity compound wiretap channel is an important open problem that could have strong implications for the use of codes providing information-theoretic security guarantees in practice.

After introducing the new notion of security and the constant-capacity compound wiretap channel, we provide a simple reduction from the wiretap and constant-capacity compound wiretap channel problems to the well-studied problem of channel coding. Specifically, in Section 6.3, we show how to achieve the secrecy capacity of a wiretap or constant-capacity compound wiretap channel by separately designing a precode with good security properties and a good channel code for the channel between the transmitter and the intended receiver. This reduction uses randomness extractors [19] (see Chapter 2 for a quick review of randomness extractors). We note that extractors were also used in [83], but we apply them in a different manner. Our reduction is useful because it means that the design of codes for the wiretap or constant-capacity compound wiretap channel can be separated into the design of a good channel code and a precode with good secrecy properties, and the channel code and the precode can be designed completely independently of each other. Our reduction is able to provide strong security in the sense of [83], but unfortunately it does not provide semantic security. Using some simple constructions of randomness extractors, we show the existence of coding schemes possessing strong security for the wiretap and constant-capacity compound wiretap channels achieving rates up to the secrecy capacity of the (degraded) wiretap channel and up to one-half of the secrecy

capacity of the constant-capacity compound wiretap channel. [1] The encoding and decoding complexity of these schemes is only $O(n \log(n) \log\log(n))$, where n denotes the block length of the code.

In Section 6.4, we construct a coding scheme that achieves rates up to the secrecy capacity and possesses semantic security for the special case of a noiseless channel to the intended receiver and a BEC to the eavesdropper, i.e., the same channel considered in [115]. Our codes have encoding and decoding complexity $O(n)$, and importantly, the complexity is uniformly bounded regardless of how close we wish to come the secrecy capacity. Finally, in Section 6.5, we construct a coding scheme that achieves rates up to the secrecy capacity and possesses semantic security for the case of BECs to the intended receiver and the eavesdropper. Our codes have encoding and decoding complexity $O(\frac{n}{\varepsilon}(\log(\frac{1}{\varepsilon}))^2)$, where ε is a measure of the gap to the secrecy capacity (see Section 6.5 for a formal statement of the result).

6.2 Formal Model and Discussion of Various Notions of Security

The wiretap channel model introduced by [120] is essentially identical to the standard broadcast channel, but the goals of communication are very different. Figure 6-1 depicts a wiretap channel. There is a discrete memoryless channel (DMC) Q_1 with input alphabet $\mathcal{X}$ and output alphabet $\mathcal{Y}$ connecting the transmitter to the intended receiver, and another DMC Q_2 with input alphabet $\mathcal{X}$ and output alphabet $\mathcal{Z}$ connecting the transmitter to the eavesdropper. Recall the definition of a DMC given in Chapter 1—if x^n is the input to the wiretap channel, then the channel outputs Y^n and Z^n at the intended receiver and the eavesdropper are random variables distributed

[1] A degraded wiretap channel is a special case of a wiretap channel satisfying a technical condition that roughly states that the channel between the transmitter and the intended receiver is less noisy than the channel between the transmitter and the eavesdropper. See Section 6.2 for a precise definition.

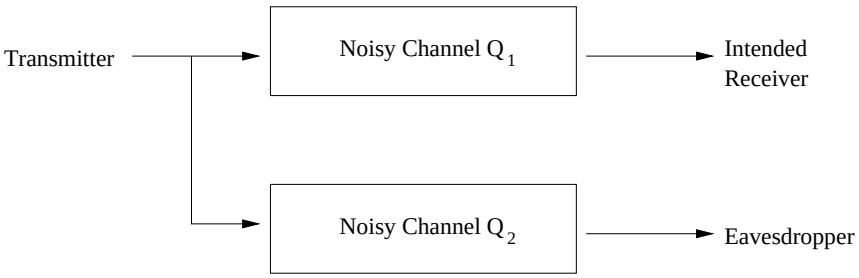

Figure 6-1: Wiretap channel. Communication is carried over a broadcast channel composed of a channel Q_1 connecting the transmitter to the intended receiver, and a channel Q_2 connecting the transmitter to the eavesdropper.

according to

$$\begin{aligned} P(Y^n = y^n|x^n) &= \prod_{i=1}^{n} Q_1(y_i|x_i) \\ P(Z^n = z^n|x^n) &= \prod_{i=1}^{n} Q_2(z_i|x_i). \end{aligned}$$

Note that the joint distribution of Y^n and Z^n is irrelevant for our purposes.

In addition to the standard wiretap channel, we define the constant-capacity compound wiretap channel as follows. The constant-capacity compound wiretap channel is specified by two parameters—a channel Q_1 between the transmitter and the intended receiver, and a capacity constraint C_2. In the constant-capacity compound wiretap channel, the eavesdropper is allowed to choose the channel Q_2 to be any DMC with capacity at most C_2. Note that the eavesdropper is allowed to make this choice in an adversarial manner even after seeing the coding scheme that the transmitter and intended receiver have agreed upon, so we must design a coding scheme that is secure no matter which channel the eavesdropper chooses.

As usual in information theory, we consider codes that map messages to codewords consisting of n elements from $\mathcal{X}$. Formally, we define a coding scheme to be a sequence of codes $\{M_n, E_n, D_n\}$, where M_n denotes the number of messages that the n^{th} code can transmit, E_n denotes the encoding function, which maps the M_n messages to elements of $\mathcal{X}^n$, and D_n denotes the decoding function used by the intended

receiver. Note that E_n can be randomized, but we do not allow the transmitter and the intended receiver to share common randomness. Intuitively, it is clear that allowing randomness in the encoding procedure makes achieving security easier, since a randomized encoding function should increase the eavesdropper's uncertainty about which message the transmitter sent. One can allow randomized decoders D_n as well, but it is easy to see that this will not help, e.g., we can always choose a deterministic MAP decoder.

There are three important performance metrics associated with a coding scheme $\{M_n, E_n, D_n\}$. The first is the rate of the coding scheme, defined as

$$R = \liminf_{n\to\infty} \frac{\log M_n}{n}.$$

(Note that we could equally well have chosen lim sup—this will not change any of our results, and in any case, in the sequel we always construct coding schemes for which the lim inf in the definition of the rate is actually a limit, i.e., the sequence converges.) The second performance metric is probability of error. Formally, let M denote the sent message, so that M is a random variable uniformly distributed over the set $\{1, 2, \ldots, M_n\}$. Let Y^n denote the output of the channel Q_1 when M is sent, i.e., Y^n is the output when $E_n(M)$ is the channel input. Then, we define the probability of error in the standard way, i.e.,

$$P_{e,n} = \Pr[D_n(Y^n) \neq M],$$

where the subscript n denotes the probability of error for the n^{th} code in the coding scheme.

The final performance metric of interest is the security guarantee of the scheme. As discussed in the introduction, we consider three notions of security. Let Z^n denote the output of the channel Q_2 when M is sent, i.e., Z^n is the output received by the eavesdropper when $E_n(M)$ is sent. The first notion of security is weak security, the definition given by Wyner.

Definition 6.2.1. *We say that a coding scheme possesses weak security if*

$$\lim_{n\to\infty} \frac{I(M;Z^n)}{n} = 0.$$

The motivation for such a definition is that if one accepts that mutual information $I(M;Z^n)$ captures how useful Z^n is in determining M, then the normalized mutual information $\frac{I(M;Z^n)}{n}$ represents the rate of information leakage to the receiver. Compared to other notions of security widely used in the cryptographic community, however, this notion of security is unacceptable, which is why we refer to this notion as weak security. As an example, note that a scheme for which the eavesdropper is always able to learn the first $\sqrt{n}$ bits of M can still satisfy the definition of weak security, while in many cryptographic applications we want to guarantee that learning even 1 bit of M is difficult. This motivates our next definition.

Definition 6.2.2. *We say that a coding scheme possesses strong security if there exists an $\varepsilon > 0$ such that*

$$I(M;Z^n) \leq 2^{-\varepsilon n}.$$

Strong security was first defined in [83]. Note that strong security at least eliminates the problem posed by our previous example—a coding scheme that allows the eavesdropper to reliably learn the first $\sqrt{n}$ bits of the message will certainly not have $I(M;Z^n)$ decay exponentially with n.

Our final notion of security, semantic security, is a very slight strengthening of strong security. We define semantic security so that it is the analog for the wiretap channel of semantic security as defined by [49] for computationally secure cryptosystems. Before stating the definition, we briefly recall the motivation given by [49] for their definition of semantic security. At an intuitive level, we want our definition of security to capture the idea that the eavesdropper learns no information after observing the ciphertext, or in our case, the outputs of the channel Q_2. The weak and strong notions of security are an attempt to capture this fact, but instead of directly moving to mutual information as a measure of the information leakage, the approach taken by [49] is to give an operational definition of security. This is similar to the

standard channel coding problem in information theory. In [107], Shannon defined an operational notion of capacity, and then proved that the operational definition of capacity happens to coincide with the mutual information. As we will see in a moment, the operational definition of security embodied in the definition of semantic security is closely related to the definition of strong security given above, i.e., mutual information will appear as a reasonable measure of information leakage, but by giving an operational definition we will see that a small modification of strong security can be interpreted as giving a strong operational definition of security.

We now state the definition of semantic security. As in [49], we capture the idea that the eavesdropper learns no information from the channel output by saying that if the eavesdropper is capable of computing some function of the sent message after observing the output of the channel Q_2, then the eavesdropper is capable of computing the same function without ever looking that the output of the channel Q_2. The key extension of semantic security beyond the previous notion of strong security is that we consider the case that the eavesdropper may have some side information about the sent message. We model this side information by assuming that the eavesdropper's prior on the message may not be uniform.

Definition 6.2.3. *Let $\{M_n, E_n, D_n\}$ denote a coding scheme. Given a set $\mathcal{S}$, a function $f_n : M_n \to \mathcal{S}$, and a probability distribution P_n over $\{1, \ldots, M_n\}$, let $p_g = \max_{y \in \mathcal{Y}} \Pr[f_n(M) = y]$, where M is a random variable distributed according to P_n.*

We say that the coding scheme $\{M_n, E_n, D_n\}$ possesses semantic security if there exists an $\varepsilon > 0$ such that for all sufficiently large n, all probability distributions P_n over M_n, all functions f_n, and all functions $A_n : \mathcal{Z}^n \to \mathcal{S}$,

$$\Pr[A_n(Z^n) = f_n(M)] \leq p_g + 2^{-\varepsilon n},$$

where M is a message distributed according to P_n and Z^n denotes the output of Q_2.

Note that p_g represents the maximum probability that the eavesdropper can guess $f_n(M)$ correctly without any knowledge except the probability distribution P_n and the function f_n. Thus, the definition above formally captures the idea that regardless of

any side information (modelled by the probability distribution P_n), the eavesdropper cannot compute any function of the sent message after seeing the channel output unless he could compute the function without looking at the channel output at all. However, this definition appears quite unwieldy. Fortunately, as shown in [49], the definition can be simplified as follows.

Proposition 6.2.1. *The following definition is equivalent to Definition 6.2.3. A coding scheme possesses semantic security if and only if there exists an $\varepsilon > 0$ such that for all sufficiently large n, all pairs of messages $m_1, m_2 \in \{1, \ldots, M_n\}$, and all functions $A_n : \mathcal{Z}^n \to \{m_1, m_2\}$,*

$$\Pr[A_n(Z^n) = M] \leq \frac{1}{2} + 2^{-\varepsilon n},$$

where M denotes the random variable that is equally likely to take on the values m_1 and m_2, and Z^n denotes the channel output when M is transmitted.

The proof of equivalence in [49] is complicated by the fact that they are dealing with the computational setting. In our situation, where arbitrary functions A_n are allowed, the proof of equivalence is simpler.

Proof. The definition given in Proposition 6.2.1 cannot be stronger than Definition 6.2.3, because the side information can always be chosen so that P_n places probability .5 on messages m_1 and m_2. Thus, the definition give in Proposition 6.2.1 is just a special case of Definition 6.2.3.

In the other direction, note that the best algorithm A_n for the definition given in Proposition 6.2.1 is the maximum a posteriori (MAP) estimate of the message given Z^n, and since the two messages are equally likely, the probability that this estimate is correct is simply $\frac{1}{2} + \frac{||P(Z^n|m_1) - P(Z^n|m_2)||_1}{4}$, where $P(Z^n|m_i)$ denotes the probability distribution of the eavesdropper's channel output given that message m_i is sent, and $||\cdot||_1$ denotes the ℓ_1-norm. Thus, the definition given in Proposition 6.2.1 implies that the ℓ_1-distance between any two conditional distributions $P(Z^n|m_1)$ and $P(Z^n|m_2)$ is exponentially small.

We complete the proof with a simple hybrid argument. Let P_n and f_n denote the side information and function to guess, respectively. Let M denote the message that the transmitter sends to the intended receiver, and let Z^n denote the corresponding channel output observed by the eavesdropper. Imagine that instead of $E_n(M)$, the input to channel Q_2 is actually $E_n(M')$, where M' distributed according to P_n, and M' is independent of M. Let Z'^n denote the corresponding channel outputs. If the eavesdropper is given access to Z'^n instead of Z^n, clearly the probability that the eavesdropper guesses $f_n(M)$ correctly is at most p_g, since Z'^n is independent of M. To complete the proof, observe that the distribution of (M, Z^n) is very close to the distribution of (M, Z'^n). Specifically, the definition given in Proposition 6.2.1 implies that $||P(Z^n|M = m) - P(Z'^n)||_1 \leq 2^{2-\varepsilon n}$ for $m = m_1$ and $m = m_2$. Therefore, for any algorithm A_n, $\Pr[A_n(Z^n) = M] \leq \Pr[A_n(Z'^n) = M] + 2^{2-\varepsilon n}$. □

Proposition 6.2.1 starts to make the connection with strong security clearer. As a simple corollary, we obtain the following lemma.

Lemma 6.2.1. *Any coding scheme possessing semantic security also possesses strong security. Conversely, given any coding scheme possessing strong security, performing a negligible amount of expurgation produces a coding scheme of the same rate possessing semantic security.*

Proof. First, we show that semantic security implies strong security. Note that

$$I(M; Z^n) = \sum_m \frac{1}{M_n} D(P_{Z^n|m} || P_{Z^n}),$$

where P_{Z^n} denotes the distribution of Z^n when the sent message M is chosen uniformly from the set $\{1, \ldots, M_n\}$, and $P_{Z^n|m}$ denotes the distribution of Z^n conditioned on the event that $M = m$. Semantic security implies that for some $\varepsilon > 0$, $||P_{Z^n|m} - P_{Z^n}||_1 \leq 2^{-\varepsilon n}$ for all messages m. From the ℓ_1-bound on entropy [28], it follows that $D(P_{Z^n|m}||P_{Z^n}) \leq ||P_{Z^n|m} - P_{Z^n}||_1$, so $I(M; Z^n) \leq 2^{-\varepsilon n}$, showing that semantic security implies strong security.

In the other direction, assume that the coding scheme $\{M_n, E_n, D_n\}$ possesses

strong security. Pinsker's inequality [28] implies that

$$\sum_m \frac{1}{2}||P_{Z^n|m} - P_{Z^n}||_1^2 \leq 2^{-\varepsilon n},$$

so by Markov's inequality, at least half of the messages m have the property $||P_{Z^n|m} - P_{Z^n}||_1^2 \leq 2^{2-\varepsilon n}$. Therefore, if we expurgate the coding scheme by only including this half of the messages, we end up with a scheme such that for any messages m_1, m_2, $||P_{Z^n|m_1} - P_{Z^n|m_2}||_1 \leq 2^{2-.5\varepsilon n}$. Thus, the expurgated coding scheme possesses semantic security, and the rate loss is only 1 bit, i.e., $\frac{1}{n}$, so the expurgated scheme has the same rate. □

Before proceeding, we make a couple of remarks. First, the proof of Lemma 6.2.1 makes it clear that there is yet another equivalent definition of semantic security that may perhaps looks more natural to information theorists. That is, we can also define semantic security as the condition that $I(\tilde{M}; Z^n) \leq e^{-\varepsilon n}$ for all $\tilde{M}$, where $\tilde{M}$ denotes a random variable that can have any distribution over the message set, not just the uniform distribution, and Z^n denotes the channel output received by the eavesdropper when $\tilde{M}$ is the sent message. Second, note that although the proof of Lemma 6.2.1 shows that we can turn a code possessing strong security into a code possessing semantic security by expurgating a vanishing fraction of the messages, the expurgation procedure may not be computationally efficient. An interesting open problem is to provide a computationally efficient reduction from strong security to semantic security.

Finally, we can define the secrecy capacity of a wiretap channel.

Definition 6.2.4. *The secrecy capacity of a wiretap channel is the supremum of the rate computed with respect to coding schemes that (a) satisfy* $\lim_{n\to\infty} P_{e,n} = 0$*, and (b) possess semantic security.*

Lemma 6.2.1 shows that in the above definition, the secrecy capacity is unchanged if we replace semantic security with strong security. As mentioned in the introduction, [83] shows that if we replace strong security with weak security the secrecy capacity

is unchanged as well, so the secrecy capacity for weak security is identical to the secrecy capacity for semantic security. We recall the following formula from [29] for the secrecy capacity of degraded wiretap channels (degraded channels are a special case of channels where Q_1 is less noisy than Q_2 in the sense of the following lemma).

Lemma 6.2.2. *We say that the channel Q_1 is less noisy than Q_2 if $I(U;Y) \geq I(U;Z)$ for every distribution $P_U P_{X|U} P_{Y,Z|X}$, where U is any auxiliary random variable such that U is independent of Y and Z conditioned on X. The capacity of a wiretap channel where Q_1 is less noisy than Q_2 is given by*

$$\max_{P_X} I(X;Y) - I(X;Z),$$

where the maximization is over all distributions P_X over $\mathcal{X}$.

From the above formula, it is clear that the secrecy capacity of the constant-capacity compound wiretap channel is at most $C_1 - C_2$, where C_1 denotes the capacity of the channel connecting the transmitter to the intended receiver. As we will see in the next section, the rate $C_1 - C_2$ is achievable, so we define the capacity of the constant-capacity compound wiretap channel as $C_1 - C_2$.

Since there is no rate loss associated with using a stronger definition of security, from the point of view of code design, we should try to design codes meeting the strongest definition of security. This is the goal of the rest of this chapter. In the next section, we prove that one can construct a (possibly computationally complex) precode to provide strong security for either the wiretap channel or the constant-capacity compound wiretap channel, and this precode can be designed independently of the channel code used to guarantee reliable communication between the transmitter and the intended receiver. We illustrate the utility of this approach by providing two concrete precodes with moderate computational complexity. In Sections 6.4 and 6.5, we construct low complexity coding schemes possessing semantic security for the special case where the channels Q_1 and Q_2 are both BECs.

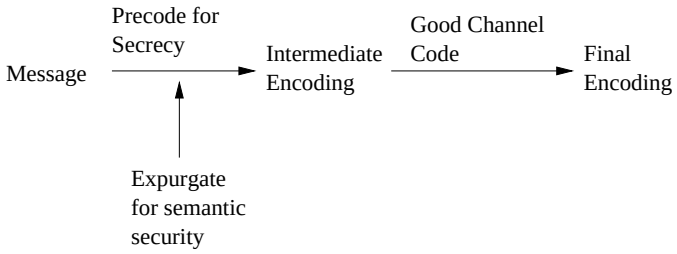

Figure 6-2: Proposed coding strategy. We split the code construction problem into the design of a precode with good security properties and a good channel code for the channel Q_1 connecting the transmitter to the intended receiver. A good precode can provide strong security, and a (non-constructive) expurgation scheme can provide semantic security.

6.3 General Approach for Strong Security Using "Invertible" Extractors

In this section, we start by proving a general reduction that splits code design for the wiretap channel and the constant-capacity compound wiretap channel into the separate problems of designing a channel code and a precode providing secrecy. Then, we provide some concrete examples to illustrate the kind of performance one can achieve with this approach.

Figure 6-2 shows the coding strategy we propose for the wiretap and constant-capacity compound wiretap channels. To encode a message, first we use a precode based on a randomness extractor, followed by a good channel code for the channel between the transmitter and the intended receiver. Before presenting a formal analysis of this coding strategy, we provide some intuition for why this strategy should provide security. As mentioned in Chapter 2, randomness extractors are functions that purify randomness, i.e., a randomness extractor takes as input a source of randomness with min-entropy k and produces k bits that are approximately uniformly distributed. Now, to communicate a message m securely, imagine that we start by computing a random preimage of m under a randomness extractor. Assuming that the extractor has suitable properties, when m is a random k bit message, and we choose a random preimage, we end up with a uniformly distributed k'-bit message, where $k' > k$ is a parameter we can tune based on the rate we are trying to achieve. To complete the

encoding, imagine that we encode this k'-bit preimage with a good channel code for Q_1. The intended receiver can decode the code, hence recover the k'-bit preimage, and compute m reliably. Now, because the eavesdropper's channel Q_2 is noisier than Q_1, intuitively we should be able to arrange things so that the eavesdropper's conditional distribution over the k'-bit preimage given the channel output Z^n has min-entropy k. Then, the extraction properties of the precode imply that the eavesdropper's conditional distribution over the k-bit message given Z^n is approximately uniform, i.e., the eavesdropper essentially learns nothing about the message m.

We formalize the intuition above with the following theorem.

Theorem 6.3.1. *Let Q_1 and Q_2 denote two discrete memoryless channels (with the same input alphabet), and let C_2 denote the capacity of Q_2. Given a channel code C for Q_1 with rate R and an $(R - C_2 - \varepsilon, 2^{-\varepsilon n})$-extractor* Ext *with the property that* $\mathrm{Ext}(X, S)$ *is uniformly distributed when X and S are uniformly and independently distributed, we can encode a message m of rate $R - C_2 - \varepsilon$ by computing a random preimage p of m under* Ext*. Then, encoding the seed s and the preimage p with C produces a code possessing strong security.*

Proof. First, we analyze the intended receiver. By assumption, the channel code C achieves low probability of error, so the intended receiver can decode the channel input, i.e., the intended receiver can decode both p and s. Therefore, the intended receiver can recover the message by computing $\mathrm{Ext}(p, s)$.

Now, we must show that most of the time, the eavesdropper learns very little after observing the channel output. As mentioned above, the rough idea is to show that for the eavesdropper, the conditional distribution of the channel input given the channel output is exponentially close (in terms of statistical, i.e., ℓ_1, distance) to a distribution with min-entropy $(R - C_2 - \varepsilon)n$. Then, the definition of (strong) extractor implies that the conditional distribution of the message is essentially uniform. We fill in the details below. (Note: for notational convenience, in the following proof we use ε to refer to any constant that can be made arbitrarily small. For example, using this notation, $\varepsilon + \varepsilon = \varepsilon$.)

First, we must show that the conditional distribution of the channel input given the channel output is close to a distribution with large min-entropy. For simplicity, we assume that the eavesdropper has access to the seed s—clearly this can only help, and as we will see, the definition of a strong extractor implies that giving the eavesdropper the seed does not help him much anyway. Thus, we are only interested in the distribution of the channel input corresponding to p. We prove that with high probability (over the choice of m, s, and the channel noise), the conditional distribution of the channel input given the channel output is a $((R - C_2 - \varepsilon)n, 2^{-\varepsilon n})$-source. To prove this, we group the codewords according to their type. Assume that the sent codeword's type has at least $2^{(R-\varepsilon)n}$ codewords. Note that this happens with probability at least $1 - \text{poly}(n)2^{-\varepsilon n}$ because there are only polynomially many types. Next, assume that the channel noise is typical. Specifically, assume that the joint type $\hat{P}_{XZ}$ between the sent codeword and the channel output satisfies $||\hat{P}_{XZ} - \hat{P}_X Q_2||_1 \leq \varepsilon$. where $\hat{P}_X$ denotes the type of the sent codeword. Again, it is well-known that such typical sets have probability at least $1 - 2^{-\varepsilon n}$. We show that when the sent codeword and the channel noise satisfy the above two assumptions, the conditional distribution of the channel input given the channel output is a $((R - C_2 - \varepsilon)n, 2^{-\varepsilon n})$-source.

We can model the situation with a bipartite graph. The left vertices correspond to the codewords with type $\hat{P}_X$, and the right vertices correspond to all channel outputs with the type $\hat{P}_Z$, where $\hat{P}_X$ and $\hat{P}_Z$ are the marginal distributions induced by $\hat{P}_{XZ}$, i.e., $\hat{P}_X$ is the type of the sent codeword and $\hat{P}_Z$ is the type of the eavesdropper's channel outputs. We connect two vertices if they induce the joint type $\hat{P}_{XZ}$. Thus, we have $2^{(R-\varepsilon)n}$ vertices on the left, and every vertex on the left has the same degree. This degree is at least $2^{n(H(Z|X)-\varepsilon)}$, where $X, Z \sim \hat{P}_X Q_2$. There are $2^{n(H(Z)+\varepsilon)}$ vertices on the right, so we see that the average degree of a vertex on the right is $2^{n(R+H(Z|X)-H(Z)-\varepsilon)} \geq 2^{n(R-C_2-\varepsilon)}$. Note that by the symmetry of a discrete memoryless channel, when we observe a particular channel output, the conditional distribution of the channel input is uniform over the left vertices adjacent to the right vertex corresponding to the observed channel output. Therefore, we simply need to analyze the probability that the observed channel output has degree exponentially

smaller than the average degree. It is obvious that this happens with exponentially small probability. In more detail, every edge of the graph is equally likely, so we just need to bound the fraction of edges that are incident to right vertices with degree less than $2^{-\varepsilon n}$ times the average degree, and clearly the fraction of such edges is at most $2^{-\varepsilon n}$. Therefore, we have shown that with probability $1 - 2^{-\varepsilon n}$, the conditional distribution of the channel input given the channel output is an $((R - C_2 - \varepsilon)n, 2^{-\varepsilon n})$-source.

To complete the proof, note that the definition of a strong extractor implies that even with knowledge of the seed, with (exponentially) high probability the output of the extractor is close to uniform. □

We emphasize that the above coding scheme also works for the constant-capacity compound wiretap channel, because the coding scheme does not require any knowledge of Q_2 other than the capacity (so that the rate of the extractor can be set appropriately). Also, note that same arguments work for the AWGN channel and for stationary ergodic channels, i.e., channels with a typical set, and that we can use the same arguments to achieve the secrecy capacity for a given degraded wiretap channel by shaping the channel code appropriately.

6.3.1 Precodes Based on the Leftover Hash Lemma

We now present a simple, but quite effective, precode construction based on the Leftover Hash Lemma [59]. Recall from Chapter 2 that the Leftover Hash Lemma provides a very simple construction of extractors. The extractor takes as input a string of n bits, and uses a seed of n bits. We can naturally view an n-bit binary string as an element of $GF(2^n)$. Then, the extractor is simply $\mathrm{Ext}(x, s) = x \cdot s$, where $\cdot$ denotes multiplication over $GF(2^n)$. We recall Lemma 2.3.6 from Section 2.3.4, restated below for convenience.

Lemma 6.3.1. *(Leftover Hash Lemma) The function* $\mathrm{Ext}(x, s) = (x \cdot s)^{\lfloor k + 2\log\varepsilon \rfloor}$ *is a* (k, ε)*-extractor. The superscript notation indicates that the output of the extractor is only the first* $\lfloor k + 2\log\varepsilon \rfloor$ *bits of the product.*

(Note: The domain of x is the set of nonzero elements of $GF(2^n)$, while the domain of s is all elements of $GF(2^n)$. With these domains, it is easily seen that when X and S are uniform, the output of the extractor is uniform, so Theorem 6.3.1 applies.)

Using this extractor construction in Theorem 6.3.1, we get a simple precode that can be used for the wiretap or constant-capacity compound wiretap channel. The complexity of encoding and decoding the precode is quite low. Specifically, encoding requires us to select a random seed, invert it, and perform a single multiplication with a random padding of the message. To decode, we just multiply the channel input with the seed. Developing efficient algorithms for arithmetic over finite fields is a well-studied problem, and although naive algorithms for multiplication and inversion might take $\Omega(n^2)$ time, using recent improvements to some well-known algorithms for finite field arithmetic [46], the complexity of multiplication and inversion can be reduced to $O(n \log(n) \log\log(n))$.[2]

The main drawback of this construction is the seed length. For the constant-capacity compound wiretap channel, since the seed is as long as the input message, this means that we achieve a rate of $\frac{C_1 - C_2}{2}$ instead of $C_1 - C_2$. On the other hand, for the scenarios usually considered in the literature, i.e., where we are only trying to construct a secure code against one possible eavesdropping channel, the seed can be chosen once and does not need to be transmitted. Phrased differently, the seed plays the role of a random ensemble of codes, and Theorem 6.3.1 and Lemma 2.3.6 imply that with exponentially high probability over the choice of the seed, the resulting (deterministic) precode is very good for any particular eavesdropping channel with capacity at most C_2. Thus, for the standard wiretap channel where we only need to be secure against a single eavesdropper channel, this construction achieves rate $C_1 - C_2$, and satisfies the definition of strong security. By shaping the channel

[2]There is a minor technicality here. We assume that n is of the form $(p-1)p^d$ for some integer d and some prime p such that 2 is a generator of Z_p^* and $2^{p-1} \not\equiv 1 \bmod p^2$. The complexity of the algorithm from [46] depends on p. In particular, the stated running times bounds hide a factor depending on p in the big-O constant. For our purposes, it suffices to note that 3 is a valid choice of p, so this means that for any n, we can find an n' in the range $n \le n' \le 3n$ of the appropriate form with $p = 3$. Thus, the condition that n be of the form $(p-1)p^d$ is not a severe restriction. In fact, since many small primes satisfy these conditions, e.g., 5 and 11 are also valid choices of p, we can find an n' in a much smaller range than $[n, 3n]$, but we do not pursue this further.

code appropriately, we can achieve any rate up to $\max_{P_X} I(X;Y) - I(X;Z)$ with this extractor, i.e., we can achieve the capacity of any degraded wiretap channel. To summarize, we see that for the usual wiretap channel, the interesting remaining questions are whether we can construct codes that achieve capacity with a linear running time and/or codes that possess semantic security. For the special case where Q_1 and Q_2 are both BECs, we address these questions in the next two sections by giving schemes that achieve semantic security with linear running time, but even for the case of a noiseless channel to the intended receiver and a BSC to the adversary, the problem of designing a linear time precode that possesses strong or semantic security remains open.

For the constant-capacity compound wiretap channel, constructing codes that can achieve the capacity is also an interesting challenge. Based on the previous discussion, it should be clear that we could get closer to the capacity if we started with an extractor with better seed length. In particular, it is well-known in the extractor literature that a seed length of εn is sufficient to guarantee $2^{-\varepsilon n}$ statistical distance to a uniform source (see, for example, [19]). In fact, relatively recently (over the last decade), many extractors have been constructed that achieve much better performance than the simple hashing-based construction above–in particular, the performance is nearly optimal in terms of the seed length. For example, the recent algebraic construction [54] produces extractors with parameters sufficient to achieve rates arbitrarily close to the capacity of the constant-capacity compound wiretap channel. However, existing constructions of extractors, although polynomial time, are not very efficient. Furthermore, the extractor constructions in the literature do not concern themselves with the problem of computing a random preimage, which is crucial in our setting. For example, it is not even clear that a random preimage for the construction in [54] can be found in polynomial time, let alone near-linear time.

The zigzag product was not designed for the case of exponentially small statistical distance that we have in mind, but we can use the zigzag construction to construct a precode providing security somewhere between weak and strong security. In more detail, if we use the Margulis expanders (described, for example, in Section 2.3.2)

instead of the LPS construction in the zigzag construction, we can construct an extractor such that evaluating the extractor and computing a random preimage can both be accomplished in $O(n)$ time—the key advantage of the Margulis construction over the LPS construction is that the edge function can be computed using only addition, as opposed to addition and multiplication, which is why we can eliminate the logarithmic factors associated with finite field multiplication that would be present if we used the LPS construction instead. This extractor only achieves a constant gap to the uniform distribution, i.e., instead of exponentially decaying mutual information, we can only produce extractors so that for any constant $\delta > 0$, $I(M; Z^n) \leq \delta$. Thus, there remains room for significant improvement for the constant-capacity compound wiretap channel. Also, the question of how to construct codes for the constant-capacity compound wiretap channel possessing semantic security remains open.

6.4 Semantic Security for the Noiseless/BEC case

In this section, we construct codes with linear time encoding and decoding algorithms that possess semantic security and achieve the capacity of a wiretap channel with a noiseless channel to the intended receiver and a BEC with erasure rate e_e to the eavesdropper. The codes we construct also have the property that the complexity is uniformly bounded with respect to the gap from capacity. The code construction we use is based on the nonsystematic irregular repeat-accumulate (NSIRA) codes described in Section 2.2.3. Specifically, by taking the dual codes of NSIRA codes, we get a suitable precode for the wiretap channel. We will explain why the dual codes are useful in a moment, but first we show that NSIRA codes can be combined with expander graphs to produce efficiently decodable codes achieving the capacity of the BEC with exponentially decaying probability of error. Note that since the channel to the intended receiver is noiseless, we do not need to use a channel code after the precode.

The key fact we use about NSIRA codes is that these codes can achieve very good performance on the BEC. Specifically, it is shown in [97] that NSIRA codes

can achieve capacity on the BEC for all erasure probabilities $p \in [0, .95]$, and it is conjectured that these codes achieve capacity for all $p \in [0, 1)$. Furthermore, these codes achieve capacity with uniformly bounded complexity regardless of the gap to capacity.

For future reference, we note that it will be useful to modify the NSIRA codes by adding a few extra parity checks. Specifically, we add εn parity checks with appropriate edges connecting these parity checks to the Rn information bits so that the resulting graph between the information bits and the εn new parity checks has good expansion properties. The reason for doing this is that the proof in [97] that NSIRA codes can achieve capacity for the BEC uses the density evolution technique. As alluded to in Chapter 1, this technique proves that the bit error probability approaches 0 in the limit of long codes and a large enough number of message-passing iterations, but density evolution does not directly provide bounds on the block error rate. By adding new parity checks and connections that form a graph with good expansion properties, we can construct a code that achieves rates arbitrarily close to the capacity, but with the further property that the block error rate decays exponentially as a function of n. Formally, we have the following lemma.

Lemma 6.4.1. *There exist codes that achieve rates arbitrarily close to capacity for the BEC with exponentially decaying block error probability, and these codes can be decoded in linear time with uniformly bounded complexity.*

Before proving Lemma 6.4.1, we need the following simple result about the existence of certain expander graphs.

Lemma 6.4.2. *For every sufficiently large n and every $\varepsilon > 0$, there exists a $\delta > 0$ and a left-regular bipartite graph G with n left vertices, εn right vertices, and left degree 10, so that G is also a $(\delta n, .8)$-expander.*

(Note: in the lemma and proof, for simplicity we assume that εn is an integer)

Proof. We use the probabilistic method. Consider a random graph formed by having each of the n left vertices randomly select (with replacement) 10 neighbors on the

right. We show that for a suitably small δ, every subset S of at most δn vertices on the left has at least $8|S|$ neighbors on the right. First, consider a fixed set S. The union bound implies that the probability that S has fewer than $8|S|$ neighbors is at most

$$\binom{\varepsilon n}{8|S|}\left(\frac{8|S|}{\varepsilon n}\right)^{10|S|}.$$

Taking a union bound over all sets of size at most δn, we see that the probability that any subset of size $\leq \delta n$ has too few neighbors is at most

$$\begin{aligned}
&\sum_{s=1}^{\delta n}\binom{n}{s}\binom{\varepsilon n}{8s}\left(\frac{8s}{\varepsilon n}\right)^{10s} \\
&\leq \sum_{s=1}^{\delta n}\left(\frac{ne}{s}\right)^{s}\left(\frac{\varepsilon n}{8s}\right)^{8s}\left(\frac{8s}{\varepsilon n}\right)^{10s} \\
&= \sum_{s=1}^{\delta n}\left(\frac{64es}{\varepsilon^2 n}\right)^{s} \\
&< 1
\end{aligned}$$

for sufficiently large n, provided that $\delta < \frac{\varepsilon^2}{64e^2}$. □

Proof of Lemma 6.4.1. The codes we consider are the NSIRA codes with degree distribution as specified in [97], along with the expander modification suggested above. Specifically, we add εn parity check nodes and connect them to the Rn information bits so that the graph induced by the information bits and the εn extra parity check nodes is a $(\delta n, .8)$-expander such that every information bit has degree 10. We know that such an expander exists by Lemma 6.4.2. Now, we use the natural decoder, i.e., first we decode the NSIRA code pretending that the extra parity checks are not present. Once the decoder finishes, we attempt to decode (using the standard message-passing algorithm) any remaining erased information bits by using the εn extra parity checks. (In other words, the idea of "ignoring" the parity checks is purely for conceptual reasons. The decoder we consider is equivalent to simply running the message-passing decoder on the whole graph.) It is well-known that for a $(k, \geq .5)$-expander, every subset of size at most k contains a vertex with a unique neighbor.

In the language of coding theory, all stopping sets have size greater than k. Since we chose the connections so that the resulting graph has expansion factor $.8 > .5$, we see that the only way the decoder can fail is if more than δn information bits are erased after the first stage of decoding.

It is known [100] that the bit error rate of BP is tightly concentrated. That is, although density evolution only provides a guarantee on the bit error rate, density evolution does show that the probability that the bit error rate deviates from the expected bit error rate by a constant is exponentially small. The density evolution analysis from [97] shows that NSIRA codes achieve the capacity, which means that after a suitably large number of BP iterations, the expected bit error rate is less than $.5\delta$, and therefore the probability that more than δn bits are erased after some suitable number of iterations is exponentially small. Since the BP decoder on the BEC has the property than once information bits are recovered, they stay recovered, we don't have to worry about information bits becoming erased in later iterations, i.e., when the BP decoder finishes, fewer than δn bits are erased with exponentially high probability. The decoding complexity is proportional to the number of edges in the graph. For the expander, this is at most $10Rn$ regardless of how small we make ε. As shown in [97], the number of edges in the NSIRA code also does not grow as a function of the gap to capacity. Therefore, the proof is complete. □

Comment: Lemma 6.4.1 shows that we can construct codes with uniformly bounded decoding complexity that also have exponentially decaying block error rate. Using the expanders of Lemma 6.4.2 in the construction of Spielman codes [113], we can construct codes that have uniformly bounded encoding and decoding complexity, and exponentially decaying block error rate.

It turns out that the dual codes of the modified NSIRA codes constructed above are very good precodes for the noiseless/BEC wiretap channel. For the reader familiar with the binary erasure quantization (BEQ) problem (defined in Chapter 5), the intuition for taking the dual code is that good codes for the BEQ problem provide semantic security for the noiseless/BEC wiretap channel, and as mentioned in Chapter 5, good codes for the BEQ problem can be constructed by taking the duals of good

codes for the BEC. To keep this chapter self-contained, we do not explicitly refer to the BEQ problem in our analysis. Instead, we first work out the structure of the duals of the modified NSIRA codes constructed above, and then explain how the dual codes can be used as precodes. After explaining how the dual codes can used as precodes, we analyze the properties that a precode must satisfy to possess semantic security—the reader familiar with the BEQ problem will recognize that the properties guaranteeing semantic security are equivalent to the properties that make a precode good for the BEQ problem, but no knowledge of the BEQ problem is needed to understand our analysis.

For simplicity, we start by describing the dual code of the standard NSIRA code, i.e., without the extra expander graph. Recall from Chapter 2 that the dual of a code can be computed by switching the variable nodes and the check nodes. Figure 6-3 shows the normal graph of the dual of an NSIRA code. Imagine that the bottom k

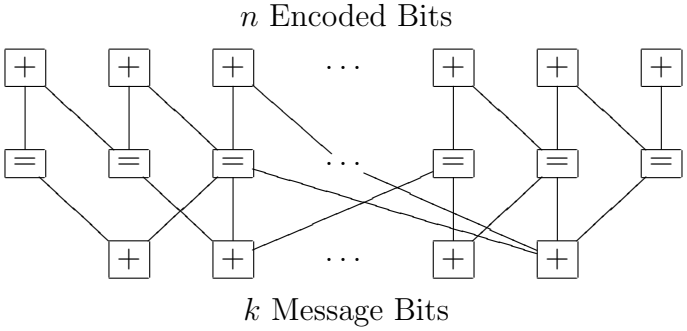

Figure 6-3: Dual NSIRA code.

check nodes do not need to be set to 0, but rather these bottom nodes will be used to encode a message. Specifically, we encode a k-bit message m as follows. First, assign values to the variable nodes so that these values are uniformly distributed over all values satisfying the constraint that the bottom k parity checks are equal to m. At this stage, it may not be entirely clear that this can be done efficiently, but we will address this concern in a moment. Once the variable nodes have been assigned these values, we compute the values of the top n parity checks. The values of these checks comprise the length n encoding of m.

With this strategy, observe that the intended receiver can easily recover m. Specifi-

cally, the intended receiver has a noiseless channel, so the intended receiver can recover the variables node values by simply accumulating the n symbols of the encoding in $O(n)$ time. Then, because the bottom part of the graph has $O(n)$ edges, the receiver can compute the values of the bottom k parity checks in $O(n)$ time, and the values of these parity checks are equal to the sent message m. The eavesdropper, however, will most likely learn absolutely nothing about the message, i.e., most of the time the posterior distribution of m given the eavesdropper's channel output is uniform over all possible messages. To see this, first observe that by construction, the prior on the channel input is uniform over all strings of length n. Therefore, the eavesdropper's channel output fixes some bits, but the erased bits are uniformly distributed. Now, for a fixed message m', the eavesdropper determines the probability that m' is the sent message by counting the number of assignments to the variable nodes that (a) match the unerased bits received by the channel, and (b) result in m' being the value of the k bottom parity checks. This number is normalized by the number of possible assignments to the variable nodes that match the unerased channel bits to get the posterior distribution. Now, basic linear algebra implies that if the matrix induced by the variable nodes, the unerased channel bits, and the k bottom parity checks has full rank, then for every m', the posterior is the same, i.e., the posterior distribution is uniform. Since row rank equals column rank, we see that this matrix has full rank if and only if the dual of the dual NSIRA code, i.e., the original NSIRA code, can recover the message when a BEC erases the set of channel bits that are received unerased by the dual NSIRA code. Formally, [80] proves the following lemma (note: this lemma is identical to Lemma 5.2.1, but we restate it below for convenience).

Lemma 6.4.3. *A linear code C with block length n can recover from a particular erasure sequence (under ML decoding) if and only if the dual code $C^{\perp}$ has full rank when restricted to the dual erasure sequence, i.e, the sequence where all the erased symbols have been turned into unerased symbols and vice versa. Also, if C can recover from an erasure sequence using message-passing decoding, then we can perform the encoding process described above for $C^{\perp}$ using a dualized form of message-passing.*

Comment: The dualized form of message-passing decoding allows us to perform encoding in linear time. For the details of how to dualize message-passing decoding, we refer the reader to [80].

Thus, we find that whenever the NSIRA code decodes a block correctly, our dual precode secures the message perfectly, i.e., the eavesdropper learns nothing about the message. Unfortunately, as we noted previously, the results on NSIRA codes rely on the density evolution technique, and therefore the results in [97] only show that the bit error probability, as opposed to the block error probability, is small. It is precisely for this reason that we introduced the modified NSIRA codes above. Lemma 6.4.1 shows that the modified codes achieve exponentially small block error rate, so by suitably modifying the strategy above, we are able to achieve semantic security.

Figure 6-4 shows the normal graph of the dual of a modified NSIRA code. It is still

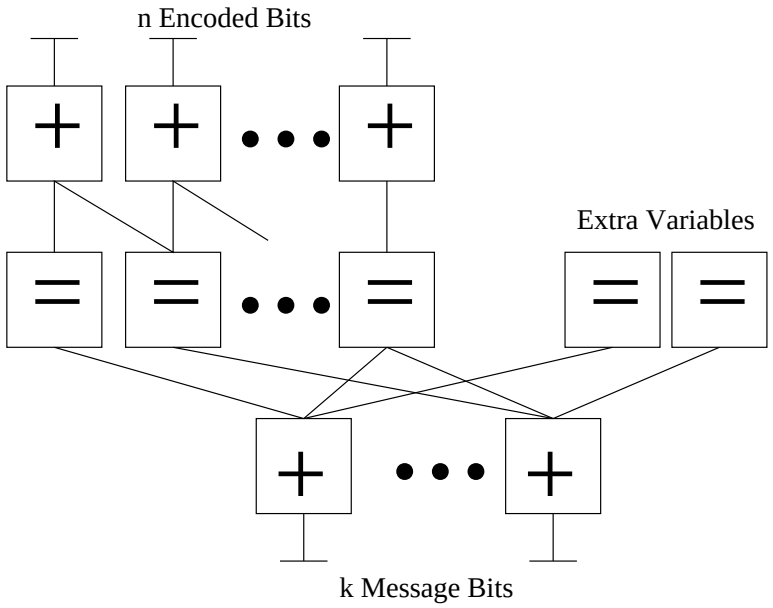

Figure 6-4: Dual modified NSIRA code.

possible to encode a message m efficiently by using a dualized form of message-passing decoding. However, note that decoding is impossible, because the extra parity checks in the modified NSIRA code become extra variables in the dual modified NSIRA code, and the top n parity checks give us no information about these variable nodes. Therefore, we need a way to communicate these extra variable nodes to the intended

receiver while keeping them secret from the eavesdropper. But this is exactly the problem we started with, so a natural strategy is to recursively apply the strategy outlined above until the number of variable nodes that needs to be communicated is small, say $O(\sqrt{n})$, at which point we can use a brute-force algorithm and a random linear code to communicate the variable nodes. We consider this approach later in this section, but for now we present another way of communicating the extra variable nodes.

The key idea is that the number of extra variable nodes is very small, i.e., there are only εn such nodes to communicate. Therefore, we can be very inefficient in terms of the rate at which we communicate these nodes. Based on the previous analysis, we see that we can guarantee that the key remains secret if we encode the extra variable nodes with a code that is the dual of a code with exponentially small error probability on the BEC, but now we no longer need to use a capacity-achieving code. Of course, the code still needs to have a suitable structure so that encoding and decoding can be done in a computationally efficient manner. As we now show, the dual codes of the repeat-accumulate-accumulate (RAA) codes proposed in [7] have all the required properties. Recall from Chapter 2 that RAA codes with suitable parameters are asymptotically good, i.e., the distance of these codes grows as δn for some constant $\delta > 0$.

As noted in Chapter 2, we define RAA codes so that the encoding includes the information bits and the outputs of both accumulators—this is slightly different from the definition of RAA codes in [7], as [7] only includes the output of the second accumulator. However, clearly our encoding is asymptotically good if the encoding from [7] is asymptotically good. We use the dual of an RAA code as our precode, but with one extra modification. Specifically, we attach a parity check of degree d to each code bit of the dual RAA code. We add this essentially so that we can amplify the erasure rate seen by the eavesdropper. The following lemma summarizes the important properties of dual RAA codes with the parity check modification.

Lemma 6.4.4. *The dual RAA codes constructed above secure the message perfectly with exponentially high probability. The encoding and decoding complexity is* $O(n)$.

(Note: whether the encoding /decoding complexity is uniformly bounded or not is irrelevant, because the encoding complexity and decoding complexity are both going to be measured with respect to εn anyway.)

Proof. To encode a message m, we start by assigning uniformly random values to the variable nodes. This induces values for the check nodes. We take the exclusive-or of the parity checks corresponding to the information bits of the RAA code with m to get the actual values used in the encoding. Finally, for each encoded bit, we select a uniformly random string of length d whose parity is equal to the parity check. Clearly this whole process is linear time.

The intended receiver can easily decode the message. Specifically, first the receiver computes the parities of the groups of d received symbols to recover the values used in the encoding. Then, the receiver inverts the two accumulators, and uses the recovered variable node values to compute the remaining parity checks. Taking the exclusive-or of these parity checks with the received values produces m, and this process clearly takes linear time.

Now, we analyze the probability that the eavesdropper's posterior distribution is not uniform. First, note that for each group of d symbols, if even a single symbol is erased, then the posterior distribution on the value of the associated bit of the encoding is uniform. Thus, we have essentially amplified the erasure probability for each bit of the encoding from e_e to $1-(1-e_e)^d$. As before, to prove security we need to show that the matrix induced by the variable nodes and unerased bits received by the eavesdropper has full rank, but since RAA codes have distance δn, by definition this matrix has full rank whenever the eavesdropper receives fewer than δn unerased bits. Thus, the probability that the encoding is not perfectly secure is at most the probability that a channel that erases each bit with probability $1-(1-e_e)^d$ erases fewer than $(1-\delta)n$ bits. This is exponentially small provided that $1-(1-e_e)^d > 1-\delta$, i.e., $d > \frac{\log(\delta)}{\log(1-e_e)}$. □

Finally, we can state the main result of this section.

Theorem 6.4.1. *The combination of a dual RAA code with the parity check modification and a modified NSIRA code achieves the secrecy capacity of the noiseless/BEC wiretap channel (with erasure probability at least .05) and possesses semantic security. Furthermore, encoding and decoding can be accomplished with uniformly bounded complexity, i.e., encoding and decoding take $O(n)$ where the constant hidden inside the big-O notation does not depend on the gap to capacity.*

Proof. We have done most of the work already. The encoder encodes the modified NSIRA code as described above, and then encodes the extra εn variable nodes using the dual RAA code with the parity check modification. The analysis of the intended receiver is trivial. The intended receiver first decodes the εn extra variable nodes using the decoding procedure for dual RAA codes with the parity check modification. Then, the intended receiver decodes the modified NSIRA code by first determining the values of the n variable nodes via accumulation, and then determining the message by computing the values of the remaining parity checks, which is possible because the values of the εn extra variable nodes are known.

To analyze the eavesdropper, first note that Lemma 6.4.4 says that the extra variable nodes are perfectly secure with exponentially high probability. The modified NSIRA code is analyzed in the same way that we analyzed NSIRA codes, except that now the probability that the matrix induced by the eavesdropper's unerased received bits, potential message bits, and the variable nodes does not have full rank is exponentially small. Therefore, the probability that the overall coding scheme does not perfectly secure the message is exponentially small. Finally, note that our analysis of security does not depend on which message m was sent, i.e., the rank of the associated matrix is only a function of which bits the channel erases, not which message was sent, so we get semantic security for free. □

6.4.1 Explicit Constructions

Theorem 6.4.1 shows that there exist codes with computationally efficient encoding and decoding algorithms that possess semantic security for the special case of a noise-

less channel to the intended receiver and a BEC to the eavesdropper. However, our construction is based on the analysis of a random ensemble of codes, so we cannot prove that a particular code has the desired properties. It would be nice to have an explicit construction of codes achieving the secrecy capacity with uniformly bounded complexity that also possess semantic security, or at least an efficient method to verify whether a particular code from a random ensemble possesses the desired properties. We note that this is closely related to the notion of deterministic extraction from a bit-fixing source [22]. In more detail, the coding scheme constructed above achieves secrecy essentially by constructing an extractor for the special class of sources known as bit-fixing sources, i.e., sources where some bits are uniform and other bits are fixed to prescribed values. In the computer science literature, the class of bit-fixing sources has been considered in the context of extractors because unlike the case of arbitrary (k, ε)-sources, where it is obvious that random seed is necessary, for the case of bit-fixing sources one can show via the probabilistic method that there exist deterministic functions capable of extracting randomness from an arbitrary bit-fixing source. Explicit constructions capable of extracting a small amount of randomness are given in [65]. In our case, we have made the problem easier because we only need to be able to extract randomness from almost all bit-fixing sources—for an exponentially small fraction of bit-fixing sources, it is okay if we fail to extract randomness. On the other hand, we proved the existence of extractors that come equipped with extremely efficient algorithms to perform extraction and compute a random preimage, and we are able to extract almost all the randomness in the source. This is reminiscent of the difference between the bounded errors model versus the BSC in channel coding—in Shannon's BSC model, we do not need to be able to correct all error patterns, just the overwhelming majority of them.

To really compare our results with the results in deterministic extraction from bit-fixing sources, it is not fair to use random ensembles, since the whole point of the deterministic extraction problem is to provide a deterministic function that is guaranteed to work. In other words, although we have proved that codes chosen from a random ensemble work with very high probability, it would be nice to be able to

construct a particular code and prove that this fixed code has the desired security properties. For the codes constructed above, verifying the properties of the NSIRA code can be done easily for very large n by checking that the graph has large girth, and then analyzing the appropriate subtrees of small depth with density evolution. We can construct suitable expanders explicitly via the zigzag product [19]. In fact, because we only require unique-neighbor expansion, we can use even more explicit expanders [2]. However, explicitly constructing or verifying that an RAA code has distance at least δn is an open problem, although some progress was made in [53]. More generally, using our strategy, to have an easily verifiable construction of a good precode, we need to explicitly construct (or efficiently verify) an asymptotically good code such that the dual code can easily be encoded and such that the dual code supports an efficient inversion procedure (so that the intended receiver can decode efficiently). Although we have not solved this problem, we can make some progress by applying the recursive strategy suggested earlier. Once the number of extra variable nodes is reduced to $n^{\frac{1}{3}}$, we no longer have to worry about the efficiency of encoding or inversion, since these procedures can be carried out in $O(n^2)$ and $O(n^3)$ time, respectively, for any linear code. Constructing asymptotically good codes explicitly is simple, since we can use the same expander constructions as before. Thus, we get an explicit code, but this code does not quite achieve semantic security since the probability that the eavesdropper learns some information only decays as $e^{-n^{\frac{1}{3}}}$, which vanishes, but not exponentially quickly.

6.5 Semantic Security for the BEC/BEC case

In this section, we construct codes with linear encoding and decoding complexity that possess semantic security and achieve the capacity of a wiretap channel where the channel between the transmitter and the receiver is BEC(e_1) and the channel between the transmitter and the eavesdropper is BEC(e_2). Unlike the previous section, the codes we construct do not have uniformly bounded complexity—the complexity of the precode is now $O(\frac{n}{\varepsilon}(\log(\frac{1}{\varepsilon}))^2)$. As in the setup of Theorem 6.3.1, we do not make

any assumptions on the rate R channel code used for the BEC (other than the fact that the code is linear), i.e., we construct an ensemble of precodes such that for any linear channel code, with high probability a randomly chosen member of the ensemble is a good precode for the given channel code.

Copying the approach from the previous section, we would like to construct an extractor for the appropriate source. Assuming that a linear code is used for the channel code, it follows that the eavesdropper learns that the channel input is uniformly distributed over some affine subspace of $GF(2)^{Rn}$. Thus, instead of constructing an extractor for bit-fixing sources, we must construct an extractor for arbitrary affine subspaces. It turns out that we can accomplish this by introducing some more randomness into the construction. At a high level, the construction contains many of the same basic ingredients as the previous case. We start by showing how to communicate near the secrecy capacity assuming that the intended receiver already has εn secret bits, i.e., the analog of a modified NSIRA code. Then, we show a highly suboptimal (in terms of rate) scheme for transmitting the εn secret bits, i.e., the analog of the dual RAA codes with the parity check modification.

To construct a capacity-approaching precode assuming a secret key, we proceed as follows. Construct a matrix

$$M = \left[\begin{array}{c|c} I & H \end{array} \right].$$

Here I denotes the $n(e_2 - e_1 - \varepsilon)$-dimensional identity matrix, and H is an $n(e_2 - 1 + R - \varepsilon)$ by $n(R - e_2 + e_1 + \varepsilon)$ matrix distributed so that each entry is i.i.d. with probability $\frac{c}{n}$ of being 1, where c is a parameter that we set later. In addition to M, form an $n(e_2 - e_1 - \varepsilon)$ by $n\varepsilon'$ matrix N such that every set of εn rows of N is linearly independent. One way of accomplishing this by taking N to be the adjacency matrix of an $(\varepsilon n, > .5)$-expander graph with $n(e_2 - e_1 - \varepsilon)$ left vertices and $n\varepsilon' = O(n\varepsilon \log(\frac{1}{\varepsilon}))$ right vertices. Intuitively, N corresponds to the small key that needs be transmitted secretly, but at a potentially suboptimal rate. Formally, the encoding process proceeds as follows. Given an $n(e_2 - 1 + R - \varepsilon)$-bit message m , first we pick a random secret

key s of length $n\varepsilon'$, and a random string x of length $n(R - e_2 + e_1 - \varepsilon)$. Then, we compute

$$t = Hx + Ns + m.$$

The input to the channel code is the Rn-bit string $[t, x]$. Note that $m = M[t, x] + Ns$, so if the intended receiver can learn the secret key s by some other means, then the message m can be recovered easily. As we will see, for suitable choices of c, ε, and ε', the probability that the eavesdropper's posterior distribution on the message is not uniform is exponentially small.

Now, we explain how to transmit s. We start with the dual RAA codes, but we use a generalization of the parity check modification. Specifically, we apply a random permutation to a dual RAA code with relative distance at least .001 and rate $\frac{1}{K}$ such that $K(h_b(.001) - .0001) > 2$. Then, we encode with a channel code capable of recovering from a .001 fraction of erasures. Finally, we encode each bit of the output using a constant size code to transform the given wiretap channel into a channel where the transmitter and the receiver are connected by BEC($\leq .0005$) and the transmitter and the adversary are connected by BEC($\geq .99995$).

Theorem 6.5.1. *The code construction above can achieve rates arbitrarily close to the secrecy capacity for the BEC/BEC wiretap channel, and possesses semantic security. Encoding and decoding take $O(\frac{n}{\varepsilon}(\log \frac{1}{\varepsilon})^2)$ time, where ε denotes the gap to capacity.*

Proof. The complexity of encoding and decoding the main code, specified by M and N, is $O(cn + n \log(\frac{1}{\varepsilon}))$, assuming that N is chosen as an optimal expander graph. The complexity of encoding and decoding the dual RAA code used to transmit the secret key is $O(n\varepsilon' \log(\frac{1}{\varepsilon}))$. Thus, the complexity of encoding and decoding is $O(n(c + \log(\frac{1}{\varepsilon})))$.

To analyze the security of the scheme, we show that regardless of the (linear) channel code used to correct errors on the BEC connecting the transmitter to the intended receiver, the precode described above provides semantic security. As in the case of a noiseless channel between the transmitter and the intended receiver, our analysis essentially boils down to proving that certain matrices have full rank with

high probability. Formally, assume that the channel connecting the transmitter and the eavesdropper erases at least $n(e_2 - .5\varepsilon)$ symbols. Note that the probability that this does not happen decays exponentially with n. Because the code is linear, the eavesdropper learns $n(1 - e_2 + .5\varepsilon)$ linear combinations of the Rn-bit string $[t, x]$. We show that regardless of the choice of the $n(1 - e_2 + .5\varepsilon)$ linear combinations, the matrices M and N protect the message m perfectly with high probability. Formally, let S denote a basis for any $n(1 - e_2 + .5\varepsilon)$ dimensional subspace of $GF(2)^{Rn}$. We show that

$$T = \begin{bmatrix} S & 0 \\ M & N \end{bmatrix}$$

has full rank, i.e., rank $n(R - .5\varepsilon)$ with high probability. To see this, compute the expected number of strings y such that $yT = 0$. We represent y as $y = [y_1, y_2]$, where y_1 is a string of length $n(1 - e_2 + .5\varepsilon)$ and y_2 is a string of length $n(e_2 - 1 + R - \varepsilon)$, so that $yT = y_1S + y_2M$. Note that S is a basis, so for each y_2, there is at most one y_1 such that $y_1S + y_2M = 0$. However, we show that when y_2 has $w > \varepsilon n$ 1's, the probability that there exists any y_1 such that $y_1S + y_2M = 0$ is small. To see this, observe that for such a y_2, y_2M is distributed as a string of Rn independent bits, where each bit has probability

$$\frac{1 + (1 - 2\frac{c}{n})^w}{2}$$

of being 0. Thus, for $c = \frac{1}{\varepsilon}\log(\frac{1}{\varepsilon})$, the probability of any string is at most $(\frac{1+\varepsilon^2}{2})^{Rn}$. Thus, the expected number of strings y_2 with at least εn 1's for which there exists a y_1 such that $y_1S + y_2M = 0$ is at most

$$\begin{aligned} 2^{n(e_2-1+R-\varepsilon)}2^{n(1-e_2+.5\varepsilon)}\left(\frac{1+\varepsilon^2}{2}\right)^{Rn} &\leq 2^{-.5\varepsilon n+\log(e)\varepsilon^2 Rn} \\ &\leq 2^{-.25\varepsilon n} \end{aligned}$$

for sufficiently small ε, e.g., $\varepsilon < .25\ln(2)$. Therefore, by Markov's inequality, the

probability that there exists any string y such that $yT = 0$ and such that the associated y_2 has more than εn 1's is exponentially small. On the other hand, if y_2 has at most εn 1's, then by the choice of N it is impossible to have $yT = 0$, so it follows that T has full rank with high probability. Therefore, with all but exponentially small probability, m is perfectly secured provided that s is perfectly secured.

To complete the proof, we must show that s is perfectly secured with all but exponentially small probability. Let S be a subspace as above, but now of dimension $.0001K\varepsilon' n$. Let C denote the code formed by taking the set of encodings of the all 0-string. Then, $C^\perp$ is an RAA code, except that we have applied a random permutation. Proving that s is communicated secrectly boils down to showing that the matrix

$$T = \begin{bmatrix} S \\ C^\perp \end{bmatrix}$$

has full rank, where with a slight abuse of notation, $C^\perp$ in definition of T denotes a basis for the code $C^\perp$. To show that $yT = 0$ has no nonzero solutions, we must show that no codeword in $C^\perp$ lies in S. The key point is that the RAA code is asymptotically good. Thus, if c is a codeword in $C^\perp$, then c has at least $.001K\varepsilon' n$ 1's, and at most $.999K\varepsilon' n$ 1's. [3] Therefore, after the random permutation, the probability that c lies in S is at most $\text{poly}(n)2^{.0001K\varepsilon' n}2^{-Kh_b(.001)\varepsilon' n}$, so the probability that any codeword lies inside S is at most $\text{poly}(n)2^{n\varepsilon'(1-K(h_b(.001)-.0001))}$, completing the proof of secrecy.

Finally, notice that the gap to capacity is $O(\varepsilon') = O(\varepsilon \log(\frac{1}{\varepsilon}))$, and the encoding and decoding complexities are $O(\frac{n}{\varepsilon'}(\log(\frac{1}{\varepsilon'}))^2)$, which completes the proof of the theorem. □

6.5.1 Explicit Constructions

Theorem 6.5.1 shows that there exist codes with computationally efficient encoding and decoding algorithms that possess semantic security for the special case of erasure

[3] To obtain the upper bound on the number of 1's, we slightly modify an RAA code by appending 0's to every codeword.

channels to the intended receiver and the eavesdropper. As in the case of Theorem 6.4.1, we can ask for an explicit construction of a code with the desired properties. This is closely related to the notion of deterministic extraction from affine sources [4], i.e., sources that are uniform over some affine subspace. In the computer science literature, this has been considered for large alphabets, where a Reed-Solomon like construction provides some nontrivial guarantees. Also, the weaker notion of an affine disperser, which is a function that only produces one bit of output, with the property that the function is not constant on any affine source, has been considered in [8]. As before, we have made the problem easier because we only need to be able to extract from almost all affine sources—for an exponentially small fraction of affine sources, it is okay if we fail to extract randomness. On the other hand, we have once again proved the existence of extractors that come equipped with extremely efficient algorithms to perform extraction and compute a random preimage, and we are able to extract almost all the randomness in the source. Of course, the new construction uses much more randomness than in the case of Theorem 6.4.1, so it is no longer so simple to construct an explicit, or easily verifiable, code. We leave the construction of explicit codes for the case of BECs to the intended receiver and the eavesdropper as an open problem.

6.6 Conclusion

In this chapter, we proposed a new channel model called the constant-capacity compound wiretap channel, and we proposed semantic security as a slight strengthening of the notion of strong security from [83]. We also proposed several different code constructions based on the principle that secrecy can be guaranteed using a precode, so channel codes can be treated as a black box designed independently of the precode. In particular, we showed a simple construction that achieves the capacity of any degraded wiretap channel, for example the Gaussian wiretap channel, possesses strong security, and has $O(n \log(n) \log\log(n))$ encoding and decoding complexity. For the constant-capacity compound wiretap channel, we provided a code construc-

tion that achieves half of the secrecy capacity, possesses strong security, and has $O(n \log(n) \log\log(n))$ encoding and decoding complexity. We also constructed codes possessing semantic security for two simple cases of the wiretap channel. First, for the case of a noiseless channel to the intended receiver and a BEC to the eavesdropper, we constructed codes that achieve the secrecy capacity, possess semantic security, and can be encoded and decoded with uniformly bounded complexity. For the case of BECs to both the intended receiver and the eavesdropper, we constructed codes that achieve the secrecy capacity, possess semantic security, and can be encoded and decoded with $O(\frac{n}{\varepsilon}(\log(\frac{1}{\varepsilon}))^2)$ complexity, where ε denotes the gap to capacity.

There are several open problems related to results in this chapter. As mentioned in Sections 6.4.1 and 6.5.1, constructing explicit codes with the same performance as our random ensembles could be interesting. In particular, constructing explicit RAA codes that are asymptotically good is an interesting challenge for future work. Also, constructing secrecy-capacity achieving codes with linear time encoding and decoding algorithms that possess semantic security for more general channels remains open. For example, even in the simple case of a noiseless channel to the intended receiver and a BSC to the eavesdropper, it is not known how to construct such codes. Finally, it would be interesting to explore other methods of generalizing the wiretap channel. We have proposed the constant-capacity compound wiretap channel in this chapter as an attempt to make the constraint that the eavesdropper is somehow fundamentally limited by physics in terms of how well he can receive the transmitter's signal more plausible, but coming up with more general models is particularly important assuming that one wants to pursue the wiretap channel as an actually practical model, rather than just an interesting theoretical toy problem. As one example of why such generalizations are important, we note that assuming certain conjectures in computational complexity theory are true, e.g., the existence of trapdoor permutations, computationally secure transmission is possible at the usual capacity of the channel between the transmitter and the intended receiver, and this remains the case even if the channel between the transmitter and the eavesdropper is noiseless. Thus, we pay a large penalty by using information-theoretic security (unless the channel be-

tween the transmitter and the eavesdropper has very low capacity), so unless a strong argument can be made for the underlying assumptions of the wiretap channel model, it is unlikely that one can justify using the information-theoretic security guarantee over a computational security guarantee.

Chapter 7

Conclusion

This thesis has explored the application of sparse graph codes to four different problems, showing that sparse graph codes have potential as a low complexity alternative to random codes in contexts beyond channel coding. We briefly summarize the main results of Chapters 3, 4, 5, and 6. Then, we describe possible directions for future work.

7.1 Summary of Results

First, in Chapter 3 we considered using sparse graphs codes to construct locally encodable and decodable source codes. For a simple class of sources, namely vectors of nonnegative integers subject to average and maximum constraints, we proposed a solution based on sparse graph codes that possesses nontrivial local encoding and local decoding properties, while still using space close to the information-theoretic limit.

In Chapter 4, we applied sparse graph codes to the problem of compressed sensing. In the first half of Chapter 4, we showed that linear measurements derived from sparse graph codes, specifically expander graphs, can perform well for compressed sensing, and further, that message-passing algorithms can be used as an efficient alternative to LP for recovering the original signal x from the measurements y. In particular, we showed that a simple message-passing algorithm can be used to recover sparse

signals, and the same algorithm also provides an ℓ_1/ℓ_1 guarantee. In the second half of Chapter 4, we showed that linear measurements based on binary matrices or on very sparse matrices do not behave well with respect to the RIP_2, suggesting that either a new proof technique or a different class of linear measurements is needed to develop fast (i.e., linear or near linear time) reconstruction algorithms providing ℓ_2/ℓ_1 recovery guarantees.

In Chapter 5, we considered using sparse graphs codes for lossy source coding. While we could not develop efficient algorithms for lossy source coding using sparse graph codes, we were able to prove nontrivial guarantees on the performance of sparse graph codes when optimal (and computationally very expensive) algorithms are used. Specifically, we show that if one ignores computational complexity, there is a strong duality between the lossy source coding problem and the channel coding problem—for many rate-distortion problems, a good channel code (i.e., a code achieving low probability of error under ML decoding) for an appropriate dual channel automatically achieves low distortion for the original rate-distortion problem. This duality result was proved using isoperimetric inequalities, and interestingly this result applies to any code, not just sparse graph codes. Roughly speaking, our result says that in high dimensions, i.e., for long blocklengths, good sphere-packings are automatically good covers.

In the second half of Chapter 5, we analyzed the interplay between graph structure and sparsity for the BSS-HD problem. We proved lower bounds on the sparsity of LDGM codes and LDPC codes that are tight to within constant factors. As explained in Chapter 5, this is the analog in lossy source coding of a phenomenon that has already been observed for channel coding, where a compound construction based on LDGM codes and LDPC codes can fundamentally achieve lower complexity (uniformly bounded complexity for the BEC) than either LDGM codes or LDPC codes individually.

Finally, in Chapter 6 we applied sparse graph codes to the wiretap channel [120]. First, we defined the information-theoretic analog of semantic security [49], and clarified the relationship between our analog of semantic security and the notion of strong

security considered in the wiretap channel literature. Then, we proposed a general architecture for constructing codes for wiretap channels, allowing the separate design of a precode with good security properties and a good channel code. We used this approach to show that for any degraded wiretap channel, there exist precodes possessing strong security whose encoding and decoding complexities are $O(n \log n \log \log n)$. For the special case of wiretap channels where the component channels are BECs, we designed sparse graph codes possessing semantic security whose encoding and decoding complexities are only $O(n)$.

7.2 Future Work

There are many interesting avenues for future research. We briefly review some of the open problems suggested in Chapters 3, 4, 5, and 6.

For Chapter 3, an interesting direction for future work is to consider methods from the data structures literature. Specifically, techniques from that field may provide lower bounds showing that local encoding and local decoding cannot be simultaneously achieved with low complexity. On the positive side, techniques from the data structures literature may also suggest techniques for improving our code constructions. More broadly, extending our results to the design of locally encodable and decodable source codes for a larger class of sources would be very interesting.

We view the results in Chapter 4 as a step towards formally connecting the theory of message-passing algorithms with that of compressed sensing. An interesting research direction would be to extend the connection between message-passing and compressed sensing further. For example, it would be interesting to design message-passing algorithms that provide ℓ_2/ℓ_1 reconstruction guarantees. We note that this problem was also suggested in [52] in the context of explicitly constructing good measurement matrices, but even proving results for randomly constructed matrices would be interesting.

Finding an explicit construction of matrices satisfying the RIP_2 remains an interesting open problem. We have shown that existing constructions have essentially been

pushed as far as possible, in the sense that our bound on the number of rows required by binary matrices is within a constant factor of the construction in [31]. From the point of view of developing efficient algorithms, it would be interesting to explore new class of codes, or develop a new analysis technique besides RIP_2, to design more efficient algorithms providing ℓ_2/ℓ_1 guarantees.

The main open problem related to Chapter 5 is to design a provably efficient algorithm for the BSS-HD problem. As noted in Chapter 5, even developing algorithms for the BSS-HD problem with complexity per bit that provably scales subexponentially as a function of $\frac{1}{\varepsilon}$, where ε denotes the gap to the rate-distortion bound, is still an open problem.

There are several open problems related to results in Chapter 6. As mentioned in Sections 6.4.1 and 6.5.1, constructing explicit codes with the same performance as our random ensembles could be interesting. In particular, constructing explicit RAA codes that are asymptotically good is an interesting challenge. Also, constructing secrecy-capacity achieving codes with linear time encoding and decoding algorithms that possess semantic security for more general channels remains open. For example, even in the simple case of a noiseless channel to the intended receiver and a BSC to the eavesdropper, it is not known how to construct such codes. Finally, it would be interesting to explore other methods of generalizing the wiretap channel. We proposed the constant-capacity compound wiretap channel as an attempt to make the constraint that the eavesdropper is somehow fundamentally limited by physics in terms of how well he can receive the transmitter's signal slightly more palatable, but coming up with more general models is particularly important assuming that one wants to pursue the wiretap channel as an actually practical model, rather than just an interesting theoretical toy problem.

In a broader context, there are many other problems in information theory where random codes have been used, and it would be interesting to construct sparse graph codes for these problems. The examples considered in this thesis showed mainly positive results, i.e., for the problems we considered sparse graph codes could replace random codes with essentially no loss in performance, but sparse graph codes could

lower the complexity dramatically. It would be interesting to see if there are problems where there is a fundamental performance loss incurred by using sparse graph codes. For example, the results on RIP_2 in Chapter 4 are in this vein, but because RIP_2 is not a necessary condition for ℓ_2/ℓ_1 reconstruction guarantees, these result does not quite furnish an example where sparse graph codes are provably substantially worse than random codes.

Appendix A

Additional Proofs for Chapter 3

This appendix contains a proof of lemma 3.8.2, including a detailed example of the density evolution technique, and a proof of Lemma 3.5.5, that the BAILOUT algorithm can be modified to run in a small amount of space.

A.1 Analysis of modified BAILOUT algorithm

In this section, we describe the modified BAILOUT algorithm and prove Lemma 3.5.5.

The modified BAILOUT algorithm is very similar to the original BAILOUT algorithm. The reason that the BAILOUT algorithm cannot be implemented in the natural way is that if we store the numbers associated with each edge naively, then the space will be to large. So, the modified BAILOUT algorithm stores the messages in compressed form. Of course, storing the messages in compressed form requires some modifications of the BAILOUT algorithm.

To give a more formal description of the modified BAILOUT algorithm, we first describe the compression scheme that will be used. The compression scheme is just a simple prefix-free code, similar to the naive prefix-free coding solution described in Section 3.2. Given an integer x, the prefix-free encoding is computed as follows. First, we compute l, the length of the binary expansion of x. Then, we compute l', the length of the binary expansion of l. Now, the prefix-free encoding of x starts with

1: **Initialization** - Set $l_v = m_v = 0$ for all left vertices v.
2: Update m_v by computing

$$m_v = \max_{w \in N(v)} \left(c_w - \sum_{x \in N(w)-v} \min_{y \in N(x)} \left(c_y - \sum_{z \in N(y)-x} l_z \right) \right)^+ .$$

3: Set $l_v = m_v$ for all left vertices v.
4: **Termination** - Repeat steps 1-2 until the values converge. By convergence, we mean that $l_v = m_v$ for all left vertices before step 2 is run, i.e., step 2 has no effect.
5: **return** overflow$(v) = l_v$ for all vertices.

Algorithm 5: Modified BAILOUT Subroutine

l' 1's, followed by a 0. Next, we append the binary expansion of l. Finally, we append the binary expansion of x. [1]

Now, the modified BAILOUT algorithm stores a number for each left vertex, instead of for each edge. Intuitively, this makes sense because the BAILOUT subroutine has the property that all numbers associated with edges in the direction leaving a left vertex v are the same, and clearly storing the same information multiple times is redundant.

The modified BAILOUT algorithm stores the numbers associated with left vertices in compressed form by concatenating the prefix-free encodings described above. Algorithm 5 gives a pseudocode description of the modified BAILOUT subroutine. For simplicity, we store two numbers l_v and m_v for each left vertex, corresponding to upper and lower bounds. In the pseudocode of Algorithm 5, we leave out the subscripts $\ell - 1$ and ℓ since these should be clear from Section 3.5.2.

The overall modified BAILOUT algorithm calls this subroutine recursively, in an identical way to the BAILOUT algorithm, except for one difference. The c_v values used in all layers except the last layer are also stored in a compressed form using the prefix-free encoding. Note that because the l_v, m_v, and c_v values are always stored in compressed form, reading or updating a single l_v or m_v can be as computationally expensive as reading through the entire compressed representation.

[1] This scheme is very similar to Elias delta coding [38], and Elias delta coding would also work for our application, but our method is slightly easier to describe.

Proof of Lemma 3.5.5. Correctness is clear because the modified BAILOUT subroutine performs an equivalent computation to the original BAILOUT subroutine. To analyze the running time, observe that the BAILOUT subroutine is called L times (once for each layer $0 \leq \ell \leq L-1$), and the proof of Lemma 3.5.4 shows that we require at most N iterations for a single subroutine call to converge. In each iteration, we update $O(N)$ values. To update a single value, we need to read $O(1)$ values stored in compressed form. Let U be an upper bound on the length of any of the compressed strings representing l_v, m_v, and c_v. Then, reading or updating a single value takes $O(U)$ time. Therefore, the total running time is at most $O(N^2LU)$ time.

The workspace needed is the space to store l_v, m_v, and c_v in each layer. [2] Note that the space can be recycled. Specifically, as we work our way back towards the first layer, the initialization destroys the l_v, m_v, and c_v values from the previous subroutine call. Therefore, we just need to analyze the maximum space needed for one subroutine call. The space is $O(U)$, so this means we just need to produce an upper bound on U.

We complete the proof by showing that $U = 4N + O(\varepsilon \log(1/\varepsilon)N)$. First, note that for positive x, our prefix-free encoding uses at most $4 + \log x + 2\log\log(2x)$ bits to represent the integer x, and our prefix-free encoding uses 4 bits to encode $x = 0$. Now, consider layer 0. We know that l_v and m_v are always at most overflow(v), so the compressed representation of these numbers has length at most $\sum_{i=1}^{N} 4 + \log(\text{overflow}(x_i)) + 2\log\log(2\text{overflow}(x_i))$, where it is understood that for those terms where the overflow is 0, we take the value of the logarithm to be 0 rather than $-\infty$. The functions $\log x$ and $\log\log(2x)$ are both concave, $\sum \text{overflow}(x_i) \leq N/K^2$, and $\text{overflow}(x_i) > 0$ for at most N/K^2 inputs. Therefore, the maximum of $\sum_{i=1}^{N} 4 + \log(\text{overflow}(x_i)) + 2\log\log(2\text{overflow}(x_i))$ is achieved when all the nonzero overflows are equal, giving us an upper bound of $4N + O(N/K^2)$. The same technique can be used to bound the length of the compressed representation of the c_v values in the first layer. The analysis can also be repeated for the higher layers, using Lemma

[2] There is also workspace associated with performing arithmetic, e.g., subtraction, but this is $O(\log(AN))$, and hence negligible for our purposes.

3.8.9. However, because the size of the layers shrinks geometrically, not surprisingly the largest upper bound comes from the first layer. Thus, $U = 4N + O(\varepsilon \log(1/\varepsilon)N)$, completing the proof. □

A.2 Completing the Analysis of WHP

In this section, we complete the analysis of WHP by proving Lemma 3.8.2. The proof involves several steps. First, we show that no counters in V_L can ever overflow. Thus, even though b_L is finite, the counters in b_L already contain the values that they would have stored if $b_L = \infty$. Next, we focus on a single computation tree, and analyze how the failure probability decays as a function of the depth of the tree. We show that the failure probability decays doubly exponentially as a function of the depth. This proof is done in two steps. First, we introduce a new probability distribution on the counter values. This distribution is different than the probability distribution induced by our data structure. For the new distribution, we are able to modify an argument from [70] to prove doubly exponential decay. In the second step, we show that the new distribution is in some sense close to the distribution induced by our data structure, and in particular doubly exponential decay under the first distribution implies doubly exponential decay under the second distribution. Finally, we use a union bound over computation trees to bound the probability of failure for each layer ℓ.

A.2.1 Bounds on the Number of Overflowing Counters

In this section we prove two lemmas bounding how many counters can overflow. These lemmas are the analog of Lemmas 3.8.8 and 3.8.9 for the V_ℓ's, and the proofs are identical.

Lemma A.2.1. *No counter in V_L can overflow.*

Proof. We prove that the largest possible value that needs to be stored in a counter in V_L is at most $\frac{M}{2K^2A^2}$. Because $2^{b_L} > \frac{M}{2K^2A^2}$, none of these counters overflow.

To verify the upper bound, consider the first layer. By assumption, the inputs have maximum value M. Therefore, the overflow is at most $\left\lfloor \frac{M}{K^2 A} \right\rfloor$. Each counter in V_1 computes the sum of the overflows associated with its K neighbors in V_0. Let s denote this sum. Then, the overflow is given by $\left\lfloor \frac{s}{12KA} \right\rfloor$. Thus, the overflow is bounded by $\frac{K(\frac{M}{K^2 A})}{12KA} = \frac{M}{12K^2A^2}$.

Now, each counter in V_2 takes the sum of the overflows of its 6 neighbors in V_1, and each of these overflows is at most $\frac{M}{12K^2A^2}$. Thus, the overflow for counters in V_2 is bounded by

$$\frac{6\frac{M}{12K^2A^2}}{8} < \frac{M}{12K^2A^2}.$$

Continuing down to the last layer, we see that in the last layer the counters in V_L have an overflow of at most $\frac{M}{12K^2A^2}$. Since each counter in V_{L+1} is connected to 6 counters in V_L, the largest possible value that a counter in V_L needs to store is at most $\frac{M}{2K^2A^2}$. □

Now, we show that even for $V_0, \ldots, V_{L-1}$, the fraction of counters that overflow must be small.

Lemma A.2.2. *The fraction of counters in V_0 that overflow is at most $1/K^2$. Also, for all $\ell > 0$, the fraction of counters in V_ℓ that overflow is at most $1/12$.*

Proof. Recall that the input must satisfy $\sum x_i < AN$. Thus, at most a $1/K^2$ fraction of the inputs can have values that are greater than or equal to K^2A, which proves the first part of the lemma. Because G_1 is $(3, K)$-regular, the average value associated with counters on the right is at most KA, and therefore at most $1/12$ of the counters in V_1 are greater than or equal to $12KA$. Repeating this analysis for the future layers, we see that at every layer at most $1/12$ of the counters overflow. □

A.2.2 Failure Probability for a Single Computation Tree, Part 1

In this section we introduce a probability distribution over the counter values, and prove that for this distribution the failure probability decays doubly exponentially.

This will be useful later when we prove Lemma 3.8.2.

First, we describe the probability distribution considered in this section. Pick an arbitrary ℓ. We want to analyze a computation tree in G_ℓ, so we define a probability distribution over the counters in $V_{\ell-1}$. For each counter in $V_{\ell-1}$, we independently assign it an overflow as follows. With probability q, the overflow is 0, i.e., there is no overflow. With probability $1-q$, the overflow is nonzero. We do not specify how to assign a specific nonzero value to the overflow because, as we shall see, for the purposes of analysis it is sufficient to know whether the overflow is 0 or nonzero. In other words, for nonzero overflow, the precise value of the overflow does not affect whether the WHP subroutine is successful or not. Regardless of how the nonzero overflows are assigned, we assume that the input to the WHP subroutine has the correct c_v values for the counters in V_ℓ, i.e., once the overflows in $V_{\ell-1}$ have been assigned, we pretend that $b_\ell = \infty$, so that each counter in V_ℓ stores the sum of the overflows of its neighbors in $V_{\ell-1}$.

Now that the probability distribution is set, we can analyze the probability of failure for the WHP subroutine under this probability distribution. Specifically, consider a counter $(v, \ell-1)$ in $V_{\ell-1}$, and imagine running the WHP subroutine (omitting step 7, the termination step) on computation trees rooted at $(v, \ell-1)$ of increasing depths. Let p_t be the probability that after running the subroutine on the computation tree rooted at $(v, \ell-1)$ of depth t, the message outgoing from $(v, \ell-1)$ (either a lower or upper bound depending on the depth of the tree mod 4) is not equal to $(v, \ell-1)$'s overflow. Also, let $\lambda(x) = x^2$, and let $\rho(x) = x^5$ if $\ell > 1$. If $\ell = 1$, let $\rho(x) = x^{K-1}$.

Lemma A.2.3. *There exists a constant $\tau > 0$ so that $p_{4t+2} = \lambda(1-\rho(1-p_{4t}))$ and $p_{4t+4} = q\lambda(1-\rho(1-p_{4t+2}))$ for $t < \tau \log N$.*

Remarks: As promised in Chapter 1, Lemma A.2.3 is an application of density evolution to our construction. Also, Lemma A.2.3 is implicit in Theorem 2 of [73].

Proof. To prove the formulas above, we consider steps 1 through 5 of the WHP subroutine separately. For ease of exposition, we start by considering a computation tree of depth 4.

First, consider the value on an edge outgoing from a leaf of the computation tree, i.e., the outgoing value from a counter at depth 4. Recall that these messages are all 0, and thus, $p_0 = q$, because by assumption each counter overflows with probability q.

Next, consider the value on an edge from a counter (u, ℓ) at depth 3 to its parent $P(u, \ell)$. We want to compute the probability that this value is correct, i.e., the value is equal to the overflow of $P(u, \ell)$. Recall that this value is computed by subtracting from c_u the sum of incoming values to (u, ℓ) from $C(u, \ell)$. Thus, the outgoing value from (u, ℓ) is correct if and only if the incoming values are all correct. For our probability model, recall that every counter in $V_{\ell-1}$ sets its overflow independently, and therefore the messages coming into (u, ℓ) are independent. Each value is correct with probability $1 - p_0$, which means that for the definition of $\rho(x)$ given above, the probability that the value leaving u is correct is $\rho(1 - p_0)$.

Now, consider the probability that the value on an edge leaving a counter $(u, \ell-1)$ at depth 2 is correct, i.e., the value is equal to $(u, \ell - 1)$'s overflow. Recall that we take the minimum of the values on the incoming edges. From Lemma 3.5.1, it follows that this procedure produces the correct value if and only if at least one edge coming into $(u, \ell - 1)$ has the correct value. Because every counter in V_ℓ sets its overflow independently, the incoming values to $(u, \ell - 1)$ are independent. For each incoming value, the probability of being wrong is $1 - \rho(1 - p_0)$. Thus, for the definition of λ given above, the probability that all incoming values are wrong is $\lambda(1 - \rho(1 - p_0))$, i.e., $p_2 = \lambda(1 - \rho(1 - p_0))$.

Next, we must consider the value on an edge leaving a counter (u, ℓ) at depth 1. Because the value is computed in the same way as the outgoing value from a counter at depth 3, the analysis is identical. Therefore, the probability that the outgoing value is wrong is simply $\rho(1 - p_2)$.

Finally, consider the outgoing value from the root. This value is computed by taking the maximum of the values on all incoming edges to the root, and 0. For this value to be wrong, observe that two things must happen. First, from Lemma 3.5.1, we see that for the outgoing value to be wrong every value coming into the root must

be wrong. However, from Lemma 3.5.1 we know that even if every incoming edge has the wrong value, if the root has zero overflow, then the outgoing message is still correct. This is because we take the maximum of not just the incoming values, but also 0. The values coming into the root are independent of the root, and each of these values is wrong with probability $1-\rho(1-p_2)$. Thus, the probability that all incoming values are wrong is $\lambda(1-\rho(1-p_2))$. Because the incoming values are independent of the root, the overall probability that the outgoing value from the root is wrong is $p_4 = q\lambda(1-\rho(1-p_2))$.

The preceding analysis proves the lemma for $t = 0$. However, it is clear that nothing in the analysis is specific to the $t = 0$ case, i.e., these recursions remain valid for going from p_4 to p_6 and p_6 to p_8, and so on. Thus, we have proved the lemma, except for one minor technical detail which we now describe.

The above derivation crucially relies on the fact that the computation tree is a tree in the graph-theoretic sense—if the computation tree contains a cycle, then the values on edges coming into a counter may not be independent. In order to guarantee that the computation tree is a tree, $2(4t+4)$ must be less than the girth of the graph G_ℓ. By Lemma 3.3.1 we know that the girth of G_ℓ is $\Omega(\log N)$, and hence a suitable $\tau > 0$ exists. □

As an aside, in the following analysis we implicitly assume that $t \ll \log N$. Looking at the values $t_\ell^{(p)}$ defined in Section 3.5.1, it is clear that all of the $t_\ell^{(p)}$'s are $o(\log N)$, so for our purposes the assumption $t \ll \log N$ is not a major restriction.

Lemma A.2.3 gives us a method for bounding the probability of failure. Specifically, recall that the WHP subroutine (with step 7 included) fails if the computation trees of depth t and depth $t+2$ give different outgoing values for the root. For this to happen, at least one of these two computation trees must give the wrong outgoing value. Applying the union bound, we see that the probability that at least one of the outgoing messages is wrong is at most $p_t + p_{t+2}$. Thus, our goal is to show that p_t decays doubly exponentially as a function of t.

We split the proof that p_t decays doubly exponentially into two steps. First, we show that $p_t \to 0$ as $t \to \infty$. Then, we modify an argument from [70] to prove that

if $p_t \to 0$ as $t \to \infty$, then p_t decays doubly exponentially.

Lemma A.2.4. *For $\ell > 1$, if $q = 1/12$, then $p_t \to 0$ as $t \to \infty$. For $\ell = 1$, if $q = 1/K^2$, then $p_t \to 0$ as $t \to \infty$.*

Proof. Let $f(x) = \lambda(1 - \rho(1 - \lambda(1 - \rho(1 - x))))$. From Lemma A.2.3, we know that $p_{4t+4} = qf(p_{4t})$. Also, note that $p_0 = q$.

Observe that f is a continuous function, f is nonnegative on the interval $[0, q]$, and $f(0) = 0$. Assume that $qf(x) < x$ for all $x \in (0, q]$. Then, clearly the recursion above satisfies the property $p_t \to 0$ as $t \to \infty$. Thus, to complete the proof we just have to show that $qf(x) < x$ for all $x \in (0, q]$. Specifically, we must show that

$$\frac{1}{12}(1 - (1 - (1 - (1 - x)^5)^2)^5)^2 < x$$

for $0 < x \leq \frac{1}{12}$, and that

$$\frac{1}{K^2}(1 - (1 - (1 - (1 - x)^{K-1})^2)^{K-1})^2 < x$$

for $0 < x \leq \frac{1}{K^2}$.

The second inequality can easily be proved using Taylor series. Specifically, for any $0 < x < 1$, Taylor's theorem implies that $(1 - x)^{K-1} > 1 - (K - 1)x$. [3] Thus, $1 - (1 - x)^{K-1} < (K - 1)x$, so $(1 - (1 - x)^{K-1})^2 < (K - 1)^2x^2$. Therefore, $(1 - (1 - (1 - (1 - x)^{K-1})^2)^{K-1})^2 < (K - 1)^2((K - 1)^2x^2)^2 < K^6x^4$. Thus, it suffices to show that $K^4x^4 < x$ for $0 < x \leq 1/K^2$. But $K^4x^4 < x$ for all $0 < x < 1/K^{4/3}$, so we are done.

Applying the analysis above with $K = 6$ gives the bound

$$\frac{1}{5\sqrt{5}}(1 - (1 - (1 - (1 - x)^5)^2)^5)^2 < x$$

for $0 < x \leq \sqrt{5}/25$, and $1/12 < \sqrt{5}/25$, completing the proof. □

Now that we know that $p_t \to 0$, we prove that the decay is doubly exponential.

[3] Recall that we assume that $K > 2$.

First, we summarize the results of Section V-A of [70] in the following lemma.

Lemma A.2.5. *Let d_l, d_r be two positive integers, and assume that $d_l \geq 3$. Let $\lambda(x) = x^{d_l - 1}$, and let $\rho(x) = x^{d_r - 1}$. Consider the recursion $y_{s+1} = q\lambda(1 - \rho(1 - y_s))$, where $y_0 = q$. If q is chosen so that $y_s \to 0$ as $s \to \infty$, then there exist constants $a > 0, d > 1, s^*$ such that $y_s \leq e^{-ad^{s-s^*}}$.*

We use Lemma A.2.5 to prove that p_t decays doubly exponentially.

Lemma A.2.6. *If q is chosen so that $p_t \to 0$ as $t \to \infty$, then there exist constants $a > 0, d > 1, t^*$ such that $p_t \leq e^{-ad^{t-t^*}}$.*

Proof. The only difference between the recursion taking p_t to p_{t+4} and the recursion considered in Lemma A.2.5 is that we apply the map $x \to \lambda(1 - \rho(1 - x))$ twice instead of just once. By expanding $\lambda(1 - \rho(1 - x))$ in a Taylor series around $x = 0$, i.e., expanding out the polynomial, it is clear that $\lambda(1 - \rho(1 - x)) = O(x^2)$ for x near 0, and thus $\lambda(1 - \rho(1 - x)) < x$ for all x in some neighborhood of 0, say $[0, \alpha]$. Also, note that $\lambda(1 - \rho(1 - x))$ is monotonically increasing over the interval $[0, 1]$.

Now, by assumption there exists some t^* such that $p_t^* < \alpha$. For all future iterations, we have

$$p_{t+1} \leq p\lambda(1 - \rho(1 - p_t)),$$

i.e., we can eliminate one instance of the map $x \to \lambda(1 - \rho(1 - x))$ because we know that the map only makes things smaller for $x < \alpha$. But once we eliminate one instance of the map $x \to \lambda(1 - \rho(1 - x))$, we are left with exactly the same recurrence as in Lemma A.2.5.

To apply Lemma A.2.5, we need to verify that the value of q is such that the recursion $y_{s+1} = q\lambda(1 - \rho(1 - y_s))$, $y_0 = q$ has the property that $y_s \to 0$ as $s \to \infty$. By assumption, $p_t \to 0$ as $t \to \infty$. Looking at the recursions given in Lemma A.2.3, we see that because $\lambda(1 - \rho(1 - x))$ is monotonically increasing for $x \geq 0$, $y_s < p_{2s}$, and thus $y_s \to 0$. Thus, Lemma A.2.5 does apply, and we can conclude that p_t decays doubly exponentially once $p_t < \alpha$. □

Remark: As we observed above, $\lambda(1 - \rho(1 - x))$ is monotonically increasing on the interval $[0, 1]$. Therefore, if we view p_t as a function of q, then p_t is monotonically increasing, i.e., for all t, p_t is larger for larger q. Thus, although we only explicitly proved doubly exponential decay for the values $q = 1/12$ and $q = 1/K^2$, it follows that we have also proved doubly exponential decay for all values smaller than $1/12$ and smaller than $1/K^2$.

A.2.3 Failure Probability for a Single Computation Tree, Part 2

Now, we prove that for the probability distribution induced by our data structure, the failure probability is doubly exponential. This represents the next step after the analysis presented in Section A.2.2 towards proving Lemma 3.8.2.

To prove doubly exponential decay, we prove that in some sense, our choice of random graphs induces a distribution that is "close" to the i.i.d. distribution considered in the previous section. We do this in two steps. First, we show that the failure probability when a fraction q of the counters are chosen uniformly at random to overflow is not much larger than the failure probability when the counters overflow i.i.d. with probability q. Then, we show that the probability of failure under the distribution induced by our data structure is not much larger than the probability of failure in the uniform model. Together with Lemma A.2.6, this implies that we get doubly exponential decay.

Lemma A.2.7. *Under the same assumptions as in Lemma A.2.6, for any $0 \leq q \leq 1$, the probability of failure when we select a fraction q of the counters in $V_{\ell-1}$ to overflow uniformly at random is at most $\sqrt{e}$ times the probability of failure when the set of counters that overflow is generated by selecting each counter independently with probability q.*

Proof. To show that the failure probability behaves essentially the same under the two different models, we will show that the input to the subroutine has approximately the same distribution under either model. Specifically, given a computation tree, let

a configuration be an assignment of either "overflow" or "don't overflow" to each counter at even depth in the computation tree. Then, the failure probability is the sum of the probabilities of configurations that cause the WHP subroutine to fail.

We show that for each configuration, the probability of observing this configuration under the model where we select a fraction q of the counters to overflow uniformly is at most $\sqrt{e}$ times as much as the probability when the counters overflow independently with probability q. Note that our proof assumes that the local neighborhood is small—in particular, our proof only works under the assumption that the computation tree has at most $\sqrt{|V_{\ell-1}|}$ vertices, i.e., the number of iterations in every layer must be substantially smaller than $\log N$. As noted previously, this is always the case for our choice of parameters, so this assumption is valid.

Given the assumption that the computation tree is small, the proof is straightforward. Define N_1 to be the number of counters that overflow in a particular configuration, and define N_2 to be the number of counters that do not overflow. Then, for the independent overflow model, the probability of this configuration is $q^{N_1}(1-q)^{N_2}$. For the uniform model, the probability is

$$\cdot \frac{\begin{array}{c} q|V_{\ell-1}|\,(q|V_{\ell-1}|-1)\,(q|V_{\ell-1}|-2)\ldots(q|V_{\ell-1}|-N_1+1) \\ (1-q)\,|V_{\ell-1}|\,((1-q)\,|V_{\ell-1}|-1)\ldots((1-q)\,|V_{\ell-1}|-N_2+1) \end{array}}{|V_{\ell-1}|\,(|V_{\ell-1}|-1)\ldots(|V_{\ell-1}|-N_1-N_2+1)} \cdot^{4}$$

Now, we want to show that this second expression is not much bigger than $q^{N_1}(1-q)^{N_2}$. This is simple. The first N_1 terms are all $\leq q$, so we can upper bound the first N_1 terms by q^{N_1}. For the remaining N_2 terms, we rewrite each fraction as

$$(1-q)\left(1+\frac{i+N_1-\frac{i}{1-q}}{|V_{\ell-1}|-i-N_1}\right) \leq (1-q)\,e^{\frac{i+N_1-\frac{i}{1-q}}{|V_{\ell-1}|-i-N_1}},$$

where i ranges from 0 to N_2-1. To complete the proof, we just need to upper bound

$$e^{\sum_{i=0}^{N_2-1}\frac{i+N_1-\frac{i}{1-q}}{|V_{\ell-1}|-i-N_1}}.$$

[4]This formula can of course be written more compactly using binomial coefficients, but expanding it out this way makes the rest of the proof obvious.

This is where we use the assumption that the computation tree is small, i.e., that $N_1 + N_2 \leq \sqrt{|V_{\ell-1}|}$. Because of this assumption, we know that the denominator of every fraction is at least $.5|V_{\ell-1}|$, provided that $|V_{\ell-1}| \geq 4$. Now, if $q = 0$ or $q = 1$, both models are identical. Thus, we may assume that $0 < q < 1$. This means that $i - \frac{i}{1-q} \leq 0$ for nonnegative i. Thus, we may bound the sum in the exponent by $\sum 2N_1/|V_{\ell-1}| = 2N_1N_2/|V_{\ell-1}|$. But $N_1N_2 < .25|V_{\ell-1}|$ because $N_1 + N_2 \leq \sqrt{|V_{\ell-1}|}$, so the exponent is at most .5. This proves that for all configurations, the ratio between the probabilities under the two models is at most $\sqrt{e}$. Thus, the failure probability when the overflowing counters are selected uniformly is at most $\sqrt{e}$ times as much as the failure probability when the counters overflow independently. □

Lemma A.2.8. *The failure probability for the distribution induced by our data structure is at most twice the failure probability for the uniform case.*

Proof. The proof is similar to the proof of Lemma A.2.7. Specifically, we prove that the input distribution induced by our data structure is close to the distribution for the uniform case. To do this, note that in terms of the grid interpretation of Q given in Section 3.3.2, the uniform case corresponds to applying a uniformly random permutation to the left vertices of $\tilde{G}$, instead of a permutation to the rows and columns separately. Fix a computation tree, and consider the distribution of where the computation tree gets mapped under a uniformly random permutation versus the row-column permutation. The key observation is that because of our choice of the grid layout in Section 3.3.2, i.e., our choice of F, the left vertices of every computation tree all lie in different rows and columns. Therefore, the row-column permutation distribution is identical to the uniformly random distribution, conditioned on the event that the uniformly random permutation maps all left vertices in the computation tree to different rows and columns.

To finish the proof, note that the probability that a uniformly random permutation does not map a set of size S to all different rows and columns is at most $\binom{S}{2}3/\sqrt{2N}$, which is less than .5 for $S < .5N^{.25}$. For our choice of $t_\ell^{(p)}$, the computation tree is always smaller that $.5N^{.25}$, so the probability that a uniformly random

permutation maps the computation tree to all different rows and columns is at least .5, and therefore the failure probability conditioned on this event is at most twice the unconditional failure probability. □

Lemma A.2.9. *The failure probability for the distribution induced by our data structure is at most* $e^{-ad^{t-t^*}}$.

Proof. We prove the lemma by combining Lemmas A.2.7 and A.2.8. To start our analysis, consider the event that exactly k counters in $V_{\ell-1}$ overflow. Conditioned on this event, Lemma A.2.7 says that the failure probability is at most $\sqrt{e}$ times the failure probability when the counters overflow independently with probability $k/|V_{\ell-1}|$. Lemma A.2.2 says that $k/|V_{\ell-1}| \leq 1/12$ for $\ell > 0$, and $k/|V_0| \leq 1/K^2$. Also, we noted above that for the independent overflow model, the failure probability is monotonically increasing as a function of the overflow probability, i.e., monotonically increasing as a function of $k/|V_{\ell-1}|$. So, we see that for all k, the failure probability conditioned on k counters overflowing is at most $\sqrt{e}$ times the failure probability when the counters overflow independently with probability $1/12$ (if $\ell > 0$) or $1/K^2$ (if $\ell = 0$).

Combining the above discussion with Lemma A.2.8, we see that the failure probability is bounded by $4\sqrt{e}e^{-ad^{t-t^*}}$ for all k. [5] To complete the proof, observe that the probability of failure can be expressed as

$$\begin{aligned}
\Pr[\text{failure}] &= \sum_k \Pr[\text{exactly } k \text{ counters overflow}] \\
&\quad \cdot \Pr[\text{failure}|\text{exactly } k \text{ counters overflow}] \\
&\leq \sum_k 4\sqrt{e}e^{-ad^{t-t^*}} \Pr[\text{exactly } k \text{ counters overflow}] \\
&\leq 4\sqrt{e}e^{-ad^{t-t^*}}.
\end{aligned}$$

Note that the $4\sqrt{e}$ factor can be absorbed into a, d, and t^*. □

[5] One factor of 2 comes from Lemma A.2.8. The other factor of 2 comes from the fact we use a union bound, i.e., as mentioned previously, we must add $p_t + p_{t+2}$.

A.2.4 Completing the Proof of Lemma 3.8.2

Finally, we prove Lemma 3.8.2.

Proof of Lemma 3.8.2. From Lemma A.2.9, we know that for a single computation tree, the failure probability is at most $e^{-ad^{t-t^*}}$. From the proof of Lemma 3.8.4, we know that the number of computation trees in G_ℓ during the p^{th} repetition is at most $n_\ell^{(p)}$. The union bound implies that the probability that any subroutine call fails in G_ℓ is at most $n_\ell^{(p)} e^{-ad^{t-t^*}}$. Substituting in the definitions given Section 3.5.1, we see that $n_\ell^{(p)} e^{-ad^{t-t^*}} \leq \delta^{(p)}/L$. □

Bibliography

[1] S. M. Aji and R. J. McEliece, "The Generalized Distributive Law," *IEEE Trans. Inform. Theory*, vol. 46, pp. 325-343, March 2000.

[2] Noga Alon and Michael Capalbo, "Explicit Unique-Neighbor Expanders," in *Proc. FOCS*, 2002.

[3] N. Alon and J. Spencer, *The Probabilistic Method*, Wiley-Interscience, 2000.

[4] B. Barak, G. Kindler, R. Shaltiel, B. Sudakov, and A. Wigderson, "Simulating independence: new constructions of condensers, ramsey graphs, dispersers, and extractors," in in *Proc. STOC*, pp. 1–10, 2005.

[5] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A simple proof of the restricted isometry property for random matrices," *Constructive Approximation*, vol. 28, no. 3, pp. 253–263, Dec. 2008.

[6] A. Barg, "Complexity Issues in Coding Theory," in book: *Handbook of Coding Theory*, vol. 1, Elsevier Science, 1998

[7] L. Bazzi, M. Mahdian, and D. A. Spielman,"The Minimum Distance of Turbo-Like Codes," *IEEE Trans. Inform. Theory*, vol. 55, pp. 6–15, January 2009.

[8] E. Ben-Sasson and S. Kopparty, "Affine dispersers from subspace polynomials," in *Proc. STOC*, pp. 65–74, 2009.

[9] R. Berinde, A. Gilbert, P. Indyk, H. Karloff, and M. Strauss, "Combining geometry and combinatorics: a unified approach to sparse signal recovery," 2008.

[10] R. Berinde and P. Indyk, "Sparse recovery using sparse random matrices," MIT CSAIL Technical Report, available at http://hdl.handle.net/1721.1/40089.

[11] R. Berinde and P. Indyk, "Sequential sparse matching pursuit," in *Proc. Allerton Conf.*, Monticello, IL, 2009.

[12] E. R. Berlekamp, R. J. McEliece, and H. C. A. van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Trans. Inform. Theory*, vol. 24, pp. 384–386, 1978.

[13] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes," in *Proceedings of the Int. Conf. on Communications*, Geneva, Switzerland, May 1993.

[14] Burton H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[15] D. Burshtein and G. Miller, "Expander graph arguments for message passing algorithms," *IEEE Trans. Inform. Theory*, vol. 47, pp. 782–790, Feb. 2001.

[16] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inform. Theory*, vol. 52, pp. 489–509, Feb. 2006.

[17] E. J. Candes, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Commun. Pure, Appl. Math.*, vol. 59, pp. 1208–1223, Aug. 2006.

[18] E. Candès and T. Tao, "Decoding by linear programming," *IEEE Trans. Inform. Theory*, vol. 51, no. 12, pp. 4203–4215, December 2005.

[19] M. Capalbo, O. Reingold, S. Vadhan, and A. Wigderson, "Randomness conductors and constant degree lossless expanders," in *Proc. STOC*, Montréal, QC, Canada, May 2002.

[20] Larry Carter, Robert Floyd, John Gill, George Markowsky, and Mark Wegman, "Exact and approximate membership testers," in *Proc. STOC* , pp. 59–65, 1978.

[21] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," ICALP, 2002.

[22] B. Chor, J. Friedman, O. Goldreich, J. Hastad, S. Rudich, and R. Smolensky, "The bit extraction problem or t-resilient functions," in *Proc. FOCS*, pp. 396-407, 1985.

[23] S.-Y. Chung, G. D. Forney, Jr., T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Communications Letters*, vol. 5, no. 2, Feb. 2001.

[24] David R. Clark and J. Ian Munro, "Efficient suffix trees on secondary storage," in Proc. 7th ACM/SIAM Symposium on Discrete Algorithms, pp. 383–391, 1996.

[25] G. Cormode and S. Muthukrishnan, "Improved data stream summaries: The count-min sketch and its applications," Latin, 2004.

[26] G. Cormode and S. Muthukrishnan, "Combinatorial algorithms for compressed sensing," in *Proc. CISS*, Princeton, NJ, Mar. 2006.

[27] T. M. Cover, "Enumerative source encoding," *IEEE Trans. Inform. Theory*, vol. 19, no. 1, pp. 73-77, 1973.

[28] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[29] I. Csiszàr and J. Körner, "Broadcast channels with confidential messages," IEEE Trans. Inform. Theory, vol. 24, no. 3, pp. 339–348, 1978.

[30] I. Csiszàr and J. Körner, *Information Theory: Coding Theorems for Discrete Memoryless Systems*, Academic Press, New York, 1981.

[31] R. DeVore," Deterministic Constructions Of Compressed Sensing Matrices," *Preprint*, 2007.

[32] D. Divsalar, H. Jin, and R. J. McEliece, "Coding theorems for 'turbo-like' codes," in *Proc. Allerton Conf.*, Monticello, IL, 1998.

[33] Y. Dodis and A. Smith, "Correcting Errors Without Leaking Partial Information," in *Proc. STOC*, May 2005.

[34] Y. Dodis and D. Wichs, "Non-malleable Extractors and Symmetric Key Cryptography from Weak Secrets," in *Proc. STOC*, May 2009

[35] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inform. Theory*, vol. 52, pp. 1289–1306, Apr. 2006.

[36] D. Donoho, A. Maleki and A. Montanari, "Message passing algorithms for compressed sensing: I. Motivation and construction," in *Proc. IEEE ITW,* Cairo, Egypt, Jan. 2010.

[37] D. Donoho, A. Maleki and A. Montanari, "Message passing algorithms for compressed sensing: II. Analysis and validation," in *Proc. IEEE ITW,* Cairo, Egypt, Jan. 2010.

[38] P. Elias, "Universal codeword sets and representations of the integers," *IEEE Trans. Inform. Theory*, vol. 21, no. 2, pp. 194–203, Mar. 1975.

[39] T. Etzion, A. Trachtenberg, and A. Vardy "Which codes have cycle-free Tanner graphs?," *IEEE Trans. Inform. Theory*, vol. 45, no. 6, pp. 2173–2181, September 1999.

[40] J. Feldman, "Decoding Error-Correcting Codes via Linear Programming," Ph. D. thesis, MIT, Cambridge, MA, Sep. 2003.

[41] J. Feldman, T. Malkin, R. A. Servedio, C. Stein, and M. J. Wainwright, "LP decoding corrects a constant fraction of errors," *CORC Tech. Rep. TR-2003-08*, Columbia Univ., New York, Dec. 2003.

[42] G.D. Forney, "Codes on Graphs: Normal Realizations," *IEEE Trans. Inform. Theory*, vol. 47, pp. 520-548, February 2001.

[43] Michael L. Fredman, Janos Komlos, and Endre Szemeredi, "Storing a sparse table with 0(1) worst case access time," *Journal of the ACM*, vol. 31, no. 3, pp. 538–544, 1984. See also FOCS 82.

[44] O. Gabber and Z. Galil, "Explicit constructions of linear-sized superconcentrators," *Journal Comput. System Sci.*, vol. 22, no. 3, pp. 407-420, 1981. Special issue dedicated to Michael Machtey.

[45] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[46] S. B. Gashkov and I. S. Sergeev, "The complexity and depth of Boolean circuits for multiplication and inversion in some fields $GF(2^n)$," Moscow University Mathematics Bulletin, Vol. 64, No. 4, pp. 139-143, August 2009.

[47] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin, "Algorithmic linear dimension reduction in the L1 norm for sparse vectors," in *Proc. Allerton Conf.*, Monticello, IL, 2006.

[48] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin, "One sketch for all: fast algorithms for compressed sensing," in *Proc. ACM STOC*, pp. 237–246, 2007.

[49] S. Goldwasser and S. Micali, "Probabilistic encryption & how to play mental poker keeping secret all partial information," in *Proc. ACM STOC*, 1982.

[50] Alexander Golynski, Roberto Grossi, Ankur Gupta, Rajeev Raman, and S. Srinivasa Rao, "On the size of succinct indices," in *Proc. 15th European Symposium on Algorithms*, pp. 371382, 2007.

[51] Alexander Golynski, Rajeev Raman, and S. Srinivasa Rao, "On the redundancy of succinct data structures," in *Proc. 11th Scandinavian Workshop on Algorithm Theory*, 2008.

[52] V. Guruswami, J. Lee, and A. Wigderson, "Euclidean sections with sublinear randomness and error-correction over the reals," in *Proc. RANDOM*, 2008.

[53] V. Guruswami and W. Machmouchi, "Explicit interleavers for a Repeat Accumulate Accumulate (RAA) code construction," in *Proc. ISIT*, 2008.

[54] V. Gurusami, C. Umans and S. Vadhan, "Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes," *Journal of the ACM*, vol. 56, no. 4, 2009.

[55] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. USA: Oxford University Press, 1979.

[56] L. H. Harper, "Optimal Numberings and Isoperimetric Problems on Graphs," *J. Comb. Theory*, vol. 1, no. 3, pp. 385–393, 1966.

[57] S. Hoory, N. Linial, and A. Wigderson, "Expander Graphs and their Applications," *Bulletin (New Series) of the American Mathematical Society*, vol. 43, no. 4, pp. 439-561, Oct. 2006.

[58] C. Hsu and A. Anastasopoulos, "Capacity-Achieving Codes with Bounded Graphical Complexity on Noisy Channels," in *Proc. Allerton Conf.*, Monticello, IL, 2005.

[59] R. Impagliazzo, L. Levin and M. Luby, "Pseudo-random generation from one-way functions," in *Proc. STOC*, 1989, pp. 12–24.

[60] P. Indyk. Sketching, streaming and sublinear-space algorithms. Graduate course notes, available at http://stellar.mit.edu/S/course/6/fa07/6.895/, 2007.

[61] P. Indyk, "Explicit constructions for compressed sensing of sparse signals," in *Proc. SODA*, San Francisco, CA, 2008.

[62] P. Indyk and M. Ruzic, "Practical near-optimal sparse recovery in the L1 norm," in *Proc. Allerton Conf.*, Monticello, IL, 2008.

[63] Guy Jacobson, "Space-efficient static trees and graphs," in *Proc. FOCS*, pp. 549–554, 1989.

[64] H. Jin, A. Khandekar, and R. J. McEliece, "Irregular repeat-accumlate codes," in *Proc. Second International Conference on Turbo Codes and Related Topics*, pp. 1–8, Brest, France, September 2000.

[65] J. Kamp and D. Zuckerman, "Deterministic extractors for bit-fixing sources and exposure-resilient cryptography," in *Proc. FOCS*, pp. 92–101, 2003.

[66] B. Kashin and V. Temlyakov, "A remark on compressed sensing," *Preprint*, 2007.

[67] M. A. Khajehnejad, A. G. Dimakis, W. Xu, and B. Hassibi, "Sparse Recovery of Positive Signals with Minimal Expansion," available at http://arxiv.org/abs/0902.4045.

[68] S. B. Korada, "Polar codes for channel and source coding," Ph.D. dissertation, EPFL, Lausanne, Switzerland, July 2009.

[69] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Trans. Inform. Theory*, vol. 47, pp. 498-519, February 2001.

[70] M. Lentmaier, D. V. Truhachev, K. Sh. Zigangirov, and D. J. Costello, Jr., "An Analysis of the Block Error Probability Performance of Iterative Decoding," *IEEE Trans. Inform. Theory*, vol. 51, no. 11, pp. 3834–3855, Nov. 2005.

[71] Y. Liang, G. Kramer, H. V. Poor, and S. Shamai (Shitz), "Compound Wiretap Channels," EURASIP Journal on Wireless Communications and Networking, vol. 2009, Article ID 142374, 12 pages, 2009.

[72] S. Litsyn and V. Shevelev, "On Ensembles of Low-Density Parity-Check Codes: Asymptotic Distance Distributions," *IEEE Trans. Inform. Theory*, vol. 48, no. 4, pp. 887–908, April 2002.

[73] Y. Lu, A. Montanari, B. Prabhakar, S. Dharmapurikar and A. Kabbani, "Counter braids: a novel counter architecture for per-flow measurement," in *Proc. ACM SIGMETRICS/Performance*, June 2008.

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

[75] M. Luby, "LT-codes," in *Proc. FOCS*, 2002.

[76] M. G. Luby, M. Mitzenmacher, M. Amin Shokrollahi, and D. A. Spielman, "Efficient Erasure Correcting Codes," *IEEE Trans. Inform. Theory*, vol. 47, No. 2, pp. 569-584, Feb. 2001.

[77] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399–431, Mar. 1999. See also the correction: *IEEE Trans. Inform. Theory*, vol. 47, pp. 2101, July, 2001.

[78] G. A. Margulis, "Explicit construction of concentrators," Problemy Peredachi Informatsii 9 (4) (1973) 71-80 (English translation Problems of Information Transmission, Plenum, New York (1975)).

[79] E. Martinian and M. Wainwright, "Low Density Codes Achieve the Rate-Distortion Bound," dcc, pp. 153-162, Data Compression Conference (DCC'06), 2006.

[80] E. Martinian and J.S. Yedidia, "Iterative Quantization Using Codes on Graphs," in *Proc. Allerton Conf.*, Monticello, IL, 2003.

[81] E. Martinian, S. Yekhanin, and J. S. Yedidia, "Secure Biometrics Via Syndromes," in *Proc. Allerton Conf.*, Monticello, IL, 2005.

[82] Y. Matsunaga and H. Yamamoto, "A Coding Theorem for Lossy Data Compression by LDPC Codes," *IEEE Trans. Inform. Theory*, vol. 49, pp. 2225-2229, September 2003.

[83] U. Maurer and S. Wolf, "Information-Theoretic Key Agreement: From Weak to Strong Secrecy for Free," *Lecture Notes in Computer Science*, pp. 351-368, Springer-Verlag, 2000.

[84] A. Montanari and E. Mossel, "Smooth compression, Gallager bound and Nonlinear sparse-graph codes," in *Proc. ISIT*, 2008.

[85] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 1995.

[86] J. Ian Munro, "Tables," in *Proc. 16th Conf. Foundations of Soft., Tech., and Th. Comp. Sci.*, pp. 37–40, 1996.

[87] J. Ian Munro, Venkatesh Raman, and S. Srinivasa Rao, "Space efficient suffix trees," *Journal of Algorithms*, vol. 39, no. 2, pp. 205–222, 2001. See also FSTTCS98.

[88] S. Muthukrishnan, "Data streams: Algorithms and applications," Available at http://athos.rutgers.edu/ muthu/stream-1-1.ps, 2003.

[89] S. Muthukrishnan, "Some algorithmic problems and results in compressed sensing," in *Allerton Conference*, 2006.

[90] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Appl. Comp. Harmonic Anal.*, vol. 26, pp. 301–321, 2009.

[91] P. Oswald and A. Shokrollahi, "Capacity achieving sequences for the erasure channel" *IEEE Trans. Inform. Theory*, vol. 48, no. 12, pp. 3017–3028, December 2002.

[92] Rasmus Pagh, "Low redundancy in static dictionaries with constant query time," *SIAM Journal on Computing*, 31(2):353–363, 2001. See also ICALP 99.

[93] Anna Pagh, Rasmus Pagh, and S. Srinivasa Rao, "An optimal Bloom filter replacement," in *Proc. SODA*, pp. 823–829, 2005.

[94] Rasmus Pagh and Flemming Friche Rodler, "Cuckoo hashing," *J. Algorithms*, 51(2):122–144, 2004. See also ESA 01.

[95] M. Patrascu, "Succincter," in *Proc. FOCS*, pp.305-313, 2008.

[96] H.D. Pfister and I. Sason, "Accumulate-Repeat-Accumulate Codes: Systematic Codes Achieving the Binary Erasure Channel Capacity with Bounded Complexity," in *Proc. Allerton Conf.*, Monticello, IL, 2005.

[97] H.D. Pfister, I. Sason, and R. Urbanke. "Capacity-Achieving Ensembles for the Binary Erasure Channel With Bounded Complexity," *IEEE Trans. on Inform. Theory*, Vol 51 (7), pp. 2352-2379, July 2005.

[98] Rajeev Raman, Venkatesh Raman, and S. Srinivasa Rao, "Succinct indexable dictionaries with applications to encoding k-ary trees and multisets," in *Proc. SODA*, pp. 233–242, 2002.

[99] T. J. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity- approaching low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, Feb. 2001.

[100] T. J. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, pp. 599-618, February 2001.

[101] T. J. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 638–656, February 2001.

[102] T. J. Richardson and R. Urbanke, "An introduction to the analysis of iterative coding systems," Institute for Mathematics and Its Applications, vol. 123, 2001.

[103] T. J. Richardson and R. Urbanke, *Modern Coding Theory*, Cambridge University Press, 2008.

[104] J. Rosenthal and P.O. Vontobel, "Constructions of LDPC codes using Ramanujan graphs and ideas from Margulis," in *Proc. Allerton Conf.*, Monticello, IL, 2000.

[105] L. Ruoheng, L. Yingbin, H. V. Poor, and P. Spasojevic,"Secure Nested Codes for Type II Wiretap Channels", in *Proc. IEEE ITW,* Tahoe City, CA, Sept. 2007.

[106] I. Sason and R. Urbanke, "Parity-check density versus performance of binary linear block codes over memoryless symmetric channels," *IEEE Trans. Inform. Theory*, vol. 49, no. 7, pp. 1611–1635, July 2003.

[107] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379-423 and 623-656, July and October, 1948.

[108] C. E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, pp. 656–715, Oct. 1949.

[109] C. E. Shannon, "Coding theorems for a discrete source with a fidelity criterion," IRE Nat. Conv. Rec, 1959.

[110] A. Shokrollahi, "Raptor Codes," available at http://www.inference.phy.cam.ac.uk/MacKay/dfountain/RaptorPaper.pdf.

[111] A. Shokrollahi,"LDPC Codes: An Introduction," 2003.

[112] M. Sipser and D. A. Spielman, "Expander codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1710–1722, Nov. 1996.

[113] D. Spielman, "Linear-time encodable and decodable error-correcting codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1723-1731, November 1996.

[114] V. N. Sudakov, B. S. Cirelson [Tsirelson], Extremal properties of half-spaces for spherically invariant measures, (Russian) Problems in the theory of probability distributions, II, Zap. Nauchn. Sem. Leningrad. Otdel. Mat. Inst. Steklov. (LOMI) 41 (1974), 14–24, 165

[115] A. Thangaraj, S. Dihidar, A. R. Calderbank, S. W. McLaughlin, and J-M. Merolla, "Applications of LDPC Codes to the Wiretap Channel," *IEEE Trans. Inform. Theory*, vol. 53, pp. 2933-2945, August 2007.

[116] J. A. Tropp, "Greed is good: algorithmic results for sparse approximation," *IEEE Trans. Inform. Theory*, vol. 50, pp. 2231–2242, Oct. 2004.

[117] L. Varshney, J. Kusuma, and V. K. Goyal, "Malleable Coding: Compressed Palimpsests," preprint available at http://arxiv.org/abs/0806.4722.

[118] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Foundations and Trends in Machine Learning*, vol. 1, pp. 1-305.

[119] M. J. Wainwright and E. Maneva, "Lossy source encoding via message-passing and decimation over generalized codewords of LDGM codes," *Proc. ISIT*, 2005

[120] A. D. Wyner, "The Wire-Tap Channel," *Bell System Technical Journal*, vol. 54, no. 8, pp. 1355–1387, Oct. 1975.

[121] W. Xu and B. Hassibi, "Efficient compressive sensing with determinstic guarantees using expander graphs," in *Proc. IEEE ITW*, Lake Tahoe, CA, 2007.

[122] Jacob Ziv and Abraham Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Trans. Inform. Theory*, vol. 23, no. 3, pp.337-343, May 1977.