Diss. ETH No. 14961

Algebraic Coding for Iterative Decoding

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY
ZURICH

for the degree of
Doctor of Technical Sciences

presented by
PASCAL O. VONTOBEL

born 21. August 1972
citizen of Oetwil am See (ZH)
dipl. El.-Ing. ETH

accepted on the recommendation of
Prof. Dr. Hans-Andrea Loeliger, examiner
Prof. Dr. R. Michael Tanner, co-examiner
Prof. Dr. Joachim Rosenthal, co-examiner

2003

To my mother and my brother,
and to the loving memory of my father.

Acknowledgments

Having somebody like Andi Loeliger as a PhD advisor is a privilege. I am very grateful for his constant support, advice, and friendship. Our many common interests made it never difficult to find rewarding discussion topics in information theory, coding theory, and beyond. I hope there will be many more joint hours where old and new subjects can be explored.

It is an honor to work with someone like Michael Tanner who was a pioneer in the field of codes on graphs. Apart from many joint discussions on coding, he, together with his wife Eileen, introduced me to the world of Shakespeare's comedies during a wonderful stay at UC Santa Cruz in the summer of 2000.

I consider working together with Joachim Rosenthal to be a very rewarding experience. Thanks to him, I had two very much enjoyable stays at the University of Notre Dame, a short one in the fall of 2000 and a long one in the summer of 2001. During that time, and already during his visit to EPF Lausanne in 1999/2000, many mathematical subjects could be touched upon and discussed at our joint coffee and tea breaks.

My time as a research and teaching assistant at the Signal and Information Processing Laboratory (ISI) at ETH Zurich actually started when the laboratory was under the guidance of Prof. J. L. Massey and Prof. G. S. Moschytz. Their personality and their inspiring lectures were certainly a main factor in my wish to join that laboratory.

I can only recommend to be a teaching assistant for Amos Lapidoth's information theory lectures. After having already attended Jim Massey's lectures, and having obtained insights from Andi Loeliger, Amos was able to show the beautiful building of information theory in yet another light and from yet another rewarding angle.

I enjoyed very much being a member of ISI and everyone made it special in a specific way. I would like to thank my former office mates Zsolt Kukorelly, Felix Lustenberger, and Dieter Arnold, the former ISI research members Martin Hänggi, Marcel Joho, Gerhard Kramer, Dani Lippuner, Heinz Mathis, Jossy Sayir, Thomas von Hoff, and Peter Wellig, and the present research members Justin Dauwels, Matthias Frey, Qun Gao, Markus Hofbauer, Daniel Hösli, Volker Koch, Sascha Korl, Ralf Kretzschmar, Patrick Merkli, Natalia Miliou, Stefan Moser, and Maja Ostojic. Not having too much to worry if the computers work is a privilege and therefore a warm thanks goes to our system administrator Max Dünki for always providing a stable Unix environment and working out solutions to urgent questions. I would also like to thank the staff members Mrs. Agotai, Mrs. Röösli, Mr. Amatore, Thomas Schärer, and Patrik Strebel for their collaboration. Last but not least, I would like to mention Dieter Arnold with whom I shared many hours of discussions on work-related subjects and other things of life; several joint after-conference trips will remain in my fond memories.

It is a pleasure to acknowledge Sarah Johnson, Steve Weller, and Christine Kelley who took their time to give me various comments about the present thesis.

My deepest thank, however, goes to my family. I thank my parents for their constant support and encouragement throughout all the stages of my education. Merci beaucoup!

Abstract

Since the publication of Shannon's 1948 paper "A Mathematical Theory of Communication" the quest has been on to find practical channel coding schemes that live up to the promises given by Shannon. Traditionally, coding theory focused on finding codes with large minimum distance and then to find an efficient decoding algorithm for such a code. In the realm of iterative decoding the picture is reversed: given an iterative decoding algorithm, one has to look out for codes that are suitable for such an algorithm.

To understand iterative decoding algorithms, it is advantageous to have a basic knowledge of factor graphs and the summary-product algorithm. Therefore, in a first step we show how the detection problems in a data transmission system can be modeled very naturally by factor graphs and solved with the help of the most popular instances of the generic summary-product algorithm, namely the sum-product and the max-product algorithm. We also show how different iteratively decodable channel codes fit very naturally into this picture.

For loop-less factor graphs the summary-product algorithm gives back the desired results; for loopy factor graphs the results are only approximations to the desired ones. As we do not want that the summary-product algorithm becomes prohibitively computationally intense, we have to bound the local state space sizes. Under these circumstances, it seems favorable in our applications at hand to perform a sub-optimal algorithm on a loopy factor graph than to perform an optimal algorithm on a loop-less factor graph. One reason for this phenomenon lies in the fact that under the above restrictions much stronger codes can be achieved when allowing factor graphs with cycles than without cycles.

In order to apply the summary-product algorithm successfully, experimental evidences seem to indicate that it is advisable that the factor graph looks locally tree-like, i.e. that there are no short cycles. This helps the messages of the summary-product algorithm to be as independent as possible. To be able to construct factor graphs of codes having these desirable properties, our next step is therefore to consider constructions of graphs having large girth, i.e. graphs whose length of the shortest cycle is large. We then unify different constructions of graphs that have a large girth and we propose some extensions.

Finally, we come to the heart of our thesis, namely the algebraic construction of codes suitable for iterative decoding. Based on graphs with large girth, we propose various algebraic constructions of regular and irregular low-density parity-check codes and turbo codes. Especially by using more complex subcodes than simple parity-check subcodes and by using bit nodes of different degrees, one can obtain a rich class of codes.

Apart from this main line, we treat several topics that are still within the subject at hand. Namely, we discuss why codes which have a loopless Tanner graph representation cannot be asymptotically good; we give a shorter and more intuitive proof of this fact than available in the literature. We unify different algorithms that can be used to perform the sum-product update rule for an indicator function of a subcode, namely the BCJR algorithm, the one-sweep algorithm, and decoding on the dual code. Finally, we propose some variations of a lower bound first given by Tanner on the minimum distance of codes.

Keywords: Digital data transmission, channel coding, iterative decoding, factor graphs, graphs with large girth, finite geometries, summary-product algorithm, sum-product algorithm, max-product algorithm, low-density parity-check codes, turbo codes, algebraic code constructions.

Kurzfassung

Seit der Publikation von Shannons Artikel “A Mathematical Theory of Communication” im Jahre 1948 wird versucht, praktische Kanalcodierungsverfahren zu finden, die die von Shannon aufgezeigen Versprechen einlösen. Traditionelle Codierungsverfahren zielten darauf hin, Codes mit grosser Minimaldistanz zu konstruieren und nachher für diese Codes effiziente Decodieralgorithmen zu finden. In der Welt der iterativen Decodierung liegt das Problem anders: gegeben sei ein iterativer Decodieralgorithmus, finde Codes, die für diesen Algorithmus geeignet sind.

Um iterative Decodieralgorithmen verstehen zu können, ist es von Vorteil, über ein Basiswissen bezüglich Faktor-Graphen und dem Summier-Produkt-Algorithmus zu verfügen. In einem ersten Schritt werden wir deshalb zeigen, wie die Detektionsprobleme, die bei der digitalen Datenübertragung auftreten, auf natürliche Weise durch Faktor-Graphen dargestellt und mit Hilfe von spezifischen Instanzen des generischen Summier-Produkt-Algorithmus’ gelöst werden können, nämlich dem Summe-Produkt- und dem Max-Produkt-Algorithmus. Des weiteren zeigen wir, wie verschiedene iterativ decodierbare Codes in dieses Bild passen.

Für Faktor-Graphen ohne Schleifen liefert der Summier-Produkt-Algorithmus die gewünschten Resultate; für Faktor-Graphen mit Schleifen jedoch sind die Resultate nur Approximationen der gewünschten Resultate. Da wir vermeiden wollen, dass die Komplexität des Summier-Produkt-Algorithmus’ zu gross wird, verlangen wir, dass die Grösse der lokalen Zustandsräume beschränkt ist. Unter dieser Voraussetzung scheint es in den uns interessierenden Problemkreisen von Vorteil zu sein, einen suboptimalen Algorithmus auf einem Graphen mit Schleifen

anstatt einem optimalen Algorithmus auf einem Graphen ohne Schleifen laufen zu lassen. Ein Grund für dieses Verhalten liegt in der Tatsache, dass unter den obigen Einschränkungen viel stärkere Codes erreicht werden können, wenn Schleifen erlaubt sind als wenn keine Schleifen erlaubt sind.

Experimentelle Resultate deuten darauf hin, dass es zur erfolgreichen Anwendung des Summier-Produkt-Algorithmus' günstig ist, wenn der Faktor-Graph lokal wie ein Baum aussieht, mit anderen Worten, dass er keine kurzen Schleifen besitzt. Dadurch sind die Nachrichten des Summier-Produkt-Algorithmus' so weit unabhängig wie möglich. Damit Konstruktionen von Faktor-Graphen mit diesen gewünschten Eigenschaften möglich werden, betrachten wir in einem nächsten Schritt die Konstruktion von Graphen mit grosser Taillenweite, d.h., Graphen bei denen die Länge der kleinsten Schleife gross ist. Wir vereinheitlichen verschiedene Konstruktionsansätze von Graphen mit grosser Taillenweite und schlagen einige Erweiterungen vor.

Das Herzstück der vorliegenden Arbeit wird dann die algebraische Konstruktion von Codes sein, die für die iterative Decodierung geeignet sind. Basierend auf Graphen mit grosser Taillenweite schlagen wir verschiedene algebraische Konstruktionen von Low-Density-Parity-Check-Codes und von Turbo-Codes vor. Insbesondere die Verwendung von komplexeren Subcodes (verglichen mit einfachen Parity-Check-Subcodes) und von Bitknoten verschiedener Grade liefert eine grosse Klasse von Codes.

Neben diesem Hauptstrang behandeln wir ein paar weitere Gebiete, die zum Themenbereich passen. So erklären wir, warum Codes, die durch schleifenlose Tanner-Graphen repräsentiert werden können, asymptotisch nicht gut sein können; dazu geben wir einen kürzeren und intuitiveren Beweis als in der Literatur verfügbar. Ferner zeigen wir die Gemeinsamkeiten von verschiedenen Algorithmen auf, die für die Durchführung der Summe-Produkt-Aufdatierungsregeln einer Subcode-Indikatorfunktion verwendet werden können; diese Algorithmen sind der BCJR-Algorithmus, der One-Sweep-Algorithmus, und das Decodieren auf dem dualen Code. Schliesslich schlagen wir verschiedene Variationen einer von Tanner eingeführten unteren Schranke für die Minimaldistanz von Codes vor.

Stichworte: Digitale Datenübertragung, Kanalcodierung, iterative Decodierung, Faktor-Graphen, Graphen mit grosser Taillenweite, endliche Geometrien, Summier-Produkt-Algorithmus, Summe-Produkt-Algorithmus, Max-Produkt-Algorithmus, Low-Density-Parity-Check-Codes, Turbo-Codes, algebraische Code-Konstruktionen.

Contents

Chapter 1

Introduction

1.1 Historical Background

Probably any text about the origins of coding and information theory must start with the work published by Shannon in 1948 [105] and so does this chapter. Shannon stated the result that reliable communication is only possible at rates up to channel capacity, meaning that for any desired symbol or block error probability, there exists a channel code and a decoding algorithm that can achieve this symbol or block error probability as long as the rate of the channel code is smaller than the channel capacity. On the other hand, if the rate is larger than capacity, the symbol and the block error probability must be bounded away from zero.

Unfortunately, the proof of the above result is nonconstructive, meaning that it proves only the existence of such a channel code and a decoding algorithm. Therefore, since the appearance of Shannon's theorem, the quest has been on for codes that have a certain structure as e.g. linearity. Moreover, finding practical decoding algorithms matched to these codes is equally important.

The codes and decoding schemes that people have come up can broadly be classified into two classes: "traditional schemes" and "modern schemes", see Fig. 1.1. In "traditional schemes", codes were proposed

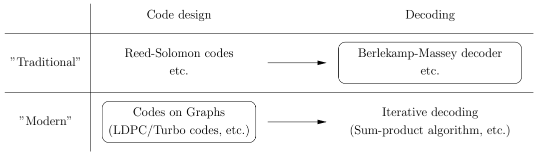

Figure 1.1: Traditional vs. modern channel coding schemes. Not encircled areas correspond to given objects, whereas encircled areas correspond to objects to be found.

that have some desirable properties like large minimum distance (a typical example of such codes are the Reed-Solomon codes [76]). But given a code, it was usually unclear how to decode it efficiently. Often it took quite some time until such a decoding algorithm was found (e.g. the Berlekamp-Massey decoding algorithm for Reed-Solomon codes), if at all. In "modern schemes", the situation is reversed: given an iterative decoding algorithm like the summary-product algorithm, the question is what codes work well together with such an iterative decoding algorithm.

We give now a brief sketch of the historical development of "modern channel coding schemes". At the beginning of the 1960s, Gallager presented so-called low-density parity-check codes (LDPCCs). He also proposed a decoding algorithm whose complexity was unfortunately above the capabilities of the computers at that time, so his work was largely forgotten. In the early 1980s, Tanner extended the work of Gallager to more general codes and algorithms, but also this was ahead of its time. Finally, the publishing of turbo codes in 1993, at a time where computers became fast enough to implement iterative decoding schemes, had a big impact on the channel coding community. Soon thereafter the LDPCCs by Gallager were rediscovered and since then the quest for good LDPCCs, good turbo codes, and good iterative decoding algorithms has been on.[1] A main advantage of the iterative decoding schemes is that they can use soft information that is often provided at the output of a channel, whereas "traditional decoding schemes" in most cases only process hard information.

The well-known convolutional codes (see e.g. [61]) should be classified

[1] In this sense, the above "modern" has to be understood.

somewhat in-between the two classes. Historically, the codes were proposed without being able to give a practical decoding algorithm. Later on, algorithms like sequential decoding and Viterbi decoding were proposed. But Viterbi decoding is equivalent to the max-product algorithm, so convolutional codes can be seen as a possible answer to the question about what codes are suitable for the max-product algorithm.

We would like to point out what some of the differences between "traditional codes" and "modern codes" are.

- A good example of "traditional codes" are the Reed-Solomon codes. Many of their properties follow from the fact that these codes can be presented by generator and parity-check matrices that are Vandermonde matrices in the horizontal and in the vertical direction. Knowing how to compute determinants of square Vandermonde matrices, one can then e.g. derive results about the minimum distance of the code. Generally speaking, in the area of "traditional codes" the usual approach was to select objects and construct codes from them in such a way that minimum distance results could be given.
- For "modern codes" the underlying objects are inherently graphs. It would be desirable to use the same algebraic objects that proved useful for constructing "traditional codes" and hope that they also lead to codes on graphs with a graph structure that is favorable for iterative decoding. But this seems to be rarely the case; therefore, new approaches are required.

  Note that there is a theory called spectral graph theory which relates graph theoretical objects with linear algebra; indeed, some nice properties of graphs can be proved with it. But it seems that for the graphs that show up in coding theory the microscope provided by spectral graph theory gives only a rough picture. Better tools are required to obtain a clearer picture.

1.2 Motivation

Most known results about the performance of LDPCCs and turbo codes are probabilistic results: for given ensembles of codes one can prove certain properties. A randomly generated channel code from such an ensemble can be used, but there is no guarantee that it indeed fulfills the promises given by the ensemble average.

As the title of the present work indicates ("algebraic coding for iterative decoding"), the goal of our project was to find out if one can come up with algebraic constructions of codes (especially linear codes) that are suitable for iterative decoding. I.e., the question was if one can say more about the properties of an algebraically constructed code than for a randomly constructed one and if one can potentially construct codes with superior performance. A further question was if it is possible to find codes that can be described concisely.

1.3 Outline of the Thesis

The second chapter revises the most important notions from coding theory and from graph theory; a very useful graph construction will turn out to be the Cayley graphs. The third chapter aims at giving the reader an introduction to graphical models in general and to factor graphs and the summary-product algorithm in particular; we show how these concepts can be used to solve the decision problems that occur when one wants to transmit data over a noisy channel. The fourth chapter presents different algebraic graph constructions that result in codes that have large girth (i.e. where the length of the shortest cycle is large); some of them will be special Cayley graphs, others will be derived from incidence structures (also known as geometries of rank 2). In the fifth chapter we then show how one can derive different types of algebraically constructed codes that are suitable for iterative decoding.

The first two appendices present some background material on algebra needed to construct the graphs in the fourth chapter. The next three appendices are about results that are within the realm of the topic of this thesis, but not in the line of the main text: we show an alternative proof why codes on factor trees with bounded state space are asymptotically bad, we explain the relationship between three algorithms that implement the sum-product update rules for a function node that is an indicator function node of a subcode, and we present some variations on a minimum distance bound introduced by Tanner. The final appendix contains several tables about graph and code parameters of various graph and code constructions, respectively.

1.4 Contributions of this Thesis

Some of the contributions of this thesis are listed below.

- Different constructions of Cayley graphs with large girth are presented in a unified manner. We expand the range of parameters for which Ramanujan graphs by Lubotzky, Phillips, and Sarnak can be constructed. A construction is proposed that unifies two different Ramanujan graphs and still gives graphs with a girth which is large enough to construct interesting factor graphs representing channel codes.
- We discuss different algebraic methods to construct regular and irregular low-density parity-check codes and show simulation results when using these codes over the binary-input additive white Gaussian noise channel. For some of the codes derived from incidence structures, a lower bound on the minimum distance can be given. Another class of codes we propose is composed of a variety of different little subcodes and bits of different degrees. We also give a construction for turbo codes based on graphs with large girth.
- It is well-known that codes on Tanner trees are asymptotically bad. Whereas the proof given in the literature is mainly combinatorial, we provide a shorter (mainly graph-based) proof of this fact.
- We show how different well-known algorithms for performing the sum-product update rule on a function node representing the indicator function of a subcode can simply be seen as solving certain transformed problems.
- We present some variations on a minimum distance bound that was first proposed by Tanner. Especially, if one knows something about the automorphism group of a code, interesting results can be given.

Chapter 2

Background Material

The aim of this chapter is to review important notions from coding and graph theory which will heavily be used in the subsequent chapters. In the coding part we give standard definitions in the realm of block codes with an emphasis on low-density parity-check codes. In the graph part a main focus will be on Cayley graphs, which will play an important role in the construction of graphs with large girth. These graphs with large girth will then be used in the construction of factor graphs of low-density parity-check codes and turbo codes.

2.1 Coding Theory

Most of the following definitions can be found in any elementary text about coding theory, e.g. [76, 61]. As is standard, we will only use row vectors. Every code will be a block code, i.e., a code of finite length.[1]

Definition 2.1 (Codes)

- **(Block code)** An (n, M) block code $\mathcal{C}$ over $\mathbb{F}_q$ of length n and size M is a subset of size M of the vector space $\mathbb{F}_q^n$. The parameter n

[1]Most of the time, we will use the shorter "code" instead of "block code". Note that in this thesis, when we talk about a code we always mean a channel code, in contrast to a source code or a line code. See also Sec. 3.2.

is called the length of the block code. An element $\mathbf{x}$ of $\mathcal{C}$ is called a codeword.[2]

- **(Linear code)** An $[n, k]$ linear (block) code $\mathcal{C}$ over $\mathbb{F}_q$ is a block code where the set of all codewords forms a k–dimensional subspace (code space) of $\mathbb{F}_q^n$. An $[n, k]$ linear code is therefore also an (n, q^k) code. n and k are called the length and the dimension, respectively, of the code.

□

This thesis deals only with linear codes; the following definitions introduce the relevant notions, especially the notion of an LDPC code.

Definition 2.2 (Linear Codes) Let $\mathcal{C}$ be an $[n, k]$ linear code over $\mathbb{F}_q$.

- **(Generator matrix)** The code $\mathcal{C}$ can be defined by a generator matrix $\mathbf{G}$ whose rows form a basis of the code space; therefore $\mathbf{G}$ must have size $k \times n$. Each codeword $\mathbf{x} \in \mathbb{F}_q^n$ can be written as $\mathbf{x} = \mathbf{u} \cdot \mathbf{G}$ with a suitable $\mathbf{u} \in \mathbb{F}_q^k$. $\mathbf{u}$ consists of the information symbols whereas $\mathbf{x}$ consists of the channel symbols. If all information symbols appear one-to-one somewhere in the channel symbols, the encoding is called systematic.
- **(Parity-check matrix)** Equivalently, $\mathcal{C}$ can be defined by a parity-check matrix $\mathbf{H}$ whose rows span the space orthogonal to the code space. Such a matrix must fulfill $\mathbf{G} \cdot \mathbf{H}^T = \mathbf{0}$, where $\mathbf{G}$ is a any generator matrix of the code. For every codeword $\mathbf{x} \in \mathcal{C}$ it follows that $\mathbf{x} \cdot \mathbf{H}^T = \mathbf{0}$. Note that the rows of $\mathbf{H}$ need not be a basis of the space orthogonal to the code space, they can also form an over-complete basis. Therefore, $\mathbf{H}$ has n columns and at least $n-k$ rows.
- **(Rate)** The rate of the code is defined to be $R \triangleq R(\mathcal{C}) \triangleq k/n$. If $\mathbf{G}$ is a generator matrix, $R \triangleq \#\mathrm{rows}(\mathbf{G})/\#\mathrm{col}(\mathbf{G})$ is the ratio of the number of rows of $\mathbf{G}$ over the number of columns of $\mathbf{G}$. If $\mathbf{H}$ is a parity-check matrix, $R = 1 - \mathrm{rank}(\mathrm{H})/n \geq 1 - \#\mathrm{rows}(\mathbf{H})/\#\mathrm{col}(\mathbf{H})$.
- **(Dual code)** The dual code $\mathcal{C}^\perp$ of $\mathcal{C}$ is the $[n, n-k]$ linear code which consists of all elements of $\mathbb{F}_q^n$ that are orthogonal to all codewords of $\mathcal{C}$.

[2]Sometimes one allows a block code to contain multiple identical entries; but we do not consider this option here any further.

- **(Zero Codeword)** The zero codeword is the vector $\mathbf{0}$. It is always an element of a linear code.
- **(Syndrome)** Let $\mathbf{H}$ be a parity-check matrix. Given a vector $\mathbf{x} \in \mathbb{F}_q^n$, the syndrome is $\mathbf{s}_{\mathbf{H}}(\mathbf{x}) \triangleq \mathbf{x} \cdot \mathbf{H}^T$. It follows that $\mathbf{s}_{\mathbf{H}}(\mathbf{x}) = \mathbf{0}$ if and only if $\mathbf{x} \in \mathcal{C}$.

□

Definition 2.3 (Weight, distance, and spectra) Let $\mathcal{C}$ be an $[n, k]$ linear code over $\mathbb{F}_q$.

- **(Hamming weight)** The Hamming weight $w_{\mathrm{H}}(\mathbf{x})$ of a vector $\mathbf{x} \in \mathbb{F}_q^n$ is the number of non-zero components of $\mathbf{x}$.
- **(Hamming distance)** The Hamming distance $d_{\mathrm{H}}(\mathbf{x}, \mathbf{y})$ between two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ is the Hamming weight of the difference between $\mathbf{x}$ and $\mathbf{y}$, i.e. $d_{\mathrm{H}}(\mathbf{x}, \mathbf{y}) = w_{\mathrm{H}}(\mathbf{x} - \mathbf{y})$. The Hamming distance is a metric.
- **(Distance Spectrum)** The distance spectrum relative to a codeword $\mathbf{x} \in \mathcal{C}$ is the histogram of the Hamming distances (relative to $\mathbf{x}$) of all codewords of $\mathcal{C}$. Because $\mathcal{C}$ is assumed to be linear, the distance spectrum is the same for all codewords.
- **(Weight Spectrum)** The weight spectrum is the histogram of the weight of all codewords. The weight spectrum is equivalent to the distance spectrum relative to the zero codeword. Because $\mathcal{C}$ is assumed to be linear, the weight spectrum is also equivalent to the distance spectrum relative to any codeword.
- **(Minimum Distance)** The minimum distance $d_{\min}$ of $\mathcal{C}$ is the smallest Hamming distance between any two distinct codewords. As $\mathcal{C}$ is assumed to be linear, $d_{\min}$ is equivalent to the smallest Hamming weight of all non-zero codewords. An $[n, k]$ linear block code with minimum distance $d_{\min}$ is said to be an $[n, k, d_{\min}]$ linear block code.

□

Definition 2.4 (LDPC codes) Let $\mathcal{C}$ be an $[n, k]$ linear code over $\mathbb{F}_q$.

- **(LDPC code)** The code $\mathcal{C}$ is called a low-density parity-check (LDPC) code if it has a parity-check matrix which has very few ones per row and per column. (This is of course a fluffy definition.

Usually, when considering LDPC code families for $n \to \infty$, one requires that the number of ones per column and the number of ones per row grow slower than the block length or that these numbers are even bounded. See also [69].)

- **(Partial factor graph of an LDPC code)** A (partial) factor graph/Tanner graph can be associated to an LDPC code. (The exact details will be given in Ch. 3.) For each code symbol position we draw a symbol node, for each line of the parity-check matrix we draw a check node, and there is an edge from a symbol node to a check node if the corresponding entry in the parity-check matrix is nonzero.

- **(Regular LDPC code)** The code $\mathcal{C}$ (defined by a low-density parity-check matrix $\mathbf{H}$) is called a $(w_{\mathrm{col}}, w_{\mathrm{row}})$–regular LDPC code if the Hamming weight of each column of $\mathbf{H}$ equals w_{col} and if the Hamming weight of each row of $\mathbf{H}$ equals w_{row}. The equality $nw_{\mathrm{col}} = mw_{\mathrm{row}}$ relates w_{col} and w_{row}, where m is the number of rows of $\mathbf{H}$. In the partial factor graph representing a regular code, all symbol nodes have the same degree and all check nodes have the same degree.

- **(Irregular LDPC code)** The code $\mathcal{C}$ (defined by a low-density parity-check matrix $\mathbf{H}$) is called an irregular LDPC code when the Hamming weights of the columns vary and/or the Hamming weights of the rows vary. $\mathcal{C}$ is said to be a $(\lambda(x), \rho(x))$-irregular LDPC code if $\lambda(x)$ and $\rho(x)$ are its (edge-oriented) degree sequences. In the partial factor graph representing an irregular code, each symbol and each check node can potentially have a different degree.

- **(Edge-oriented degree sequences)** We call the polynomials $\lambda(x) \triangleq \sum_{i\geq 1} \lambda_i x^{i-1}$ and $\rho(x) \triangleq \sum_{i\geq 1} \rho_i x^{i-1}$ the symbol and the check (edge-oriented) degree sequences [95], respectively. (Note that in both polynomials the exponent of x is by one smaller than the index of the coefficient. This is done in order to simplify the density evolution equations [95, 96].) λ_i is the fraction of edges that are incident on a symbol node with degree i, whereas ρ_i is the fraction of edges that are incident on a check node with degree i. The rate of the code is at least $1 - \int_0^1 \rho(x)\mathrm{d}x / \int_0^1 \lambda(x)\mathrm{d}x$ (see [95]). Note that degree sequences of regular codes have only one nonzero coefficient λ_i and one nonzero coefficient ρ_i.

- **(Vertex-oriented degree sequences)** We call the polynomials $\tilde{\lambda}(x) \triangleq \sum_{i\geq 1} \tilde{\lambda}_i x^{i-1}$ and $\tilde{\rho}(x) \triangleq \sum_{i\geq 1} \tilde{\rho}_i x^{i-1}$ the symbol and the check vertex-oriented degree sequences, respectively. $\tilde{\lambda}_i$ is the fraction of symbol nodes of degree i, whereas $\tilde{\rho}_i$ is the fraction of check nodes of degree i. We have the relations $\tilde{\lambda}_i = (\lambda_i/i)/(\sum_{i'\geq 1}(\lambda_{i'}/i'))$ and $\tilde{\rho}_i = (\rho_i/i)/(\sum_{i'\geq 1}(\rho_{i'}/i'))$.

- **(Maximum symbol node degree, maximum check node degree)** Let $\lambda(x)$ and $\rho(x)$ be the edge-oriented degree sequences of an LDPC code. The largest i such that λ_i is nonzero is the maximum symbol node degree, whereas the largest i such that ρ_i is nonzero is the maximum check node degree.

$\square$

Label	Description	Name
$\mathcal{U}$	a set	universum
$\mathcal{B} \subseteq \mathcal{U}$	$\mathcal{B}$ a subset of $\mathcal{U}$	behavior
$\mathbf{x} \in \mathbf{U}$	element of the universum	outcome
$\mathbf{x} \in \mathcal{B}$	element of the behavior	valid outcome

Table 2.1: Main definitions from behavioral systems theory. Often an outcome is called a configuration and a valid outcome a valid configuration, respectively.

Label	Description	Name
$\mathbb{F}_q^n$	a vector space	vector space
$\mathcal{C} \subseteq \mathbb{F}_q^n$	a subset of $\mathbb{F}_q^n$	code, codebook
$\mathbf{x} \in \mathbb{F}_q^n$	element of the vector space	vector
$\mathbf{x} \in \mathcal{C}$	element of the code	codeword

Table 2.2: The meaning of the definitions in Tab. 2.1 in the context of coding theory.

Remark 2.5 (Behaviors) Codes can be cast in the framework of behaviors (for more about behaviors, see e.g. [92]). Tab. 2.1 gives the relevant definitions from behavior theory [92]: a mathematical model is a pair $(\mathcal{U}, \mathcal{B})$ consisting of a universum $\mathcal{U}$ and a behavior $\mathcal{B}$. In this notation, a code is then a pair $(\mathbb{F}_q^n, \mathcal{C})$. Tab. 2.2 shows line-by-line what the behavioristic notions shown in Tab. 2.1 mean in coding theory. For our purposes, the interesting behaviors will be those that can be expressed by local behaviors. (This issue will be continued in Ch. 3). □

2.2 Graphs

2.2.1 Undirected and Directed Graphs

Most of the following definitions are standard and can be found in any text about graph theory, e.g. [16, 15].

Definition 2.6 (Undirected graph, directed graph) An *undirected* graph $\mathsf{X} = \mathsf{X}(\mathcal{V}, \mathcal{E})$ consists of a vertex set $\mathcal{V}$ and an edge set $\mathcal{E}$ whereby the elements of $\mathcal{E}$ are 2-subsets of $\mathcal{V}$. A *directed* graph $\vec{\mathsf{X}} = \vec{\mathsf{X}}(\mathcal{V}, \vec{\mathcal{E}})$ consists of a vertex set $\mathcal{V}$ and a directed edge set $\vec{\mathcal{E}}$ (sometimes also called arcs) whereby the elements of $\vec{\mathcal{E}}$ are ordered pairs from $\mathcal{V}$. By a graph (without further qualifications) we will always mean an undirected graph. We will not allow self-loops and multiple edges. □

Example 2.7 (Undirected graph) The undirected graph $\mathsf{X} = \mathsf{X}(\mathcal{V}, \mathcal{E})$ in Fig. 2.1 (left) has

$$\begin{aligned}
\mathcal{V} &= \{1, 2, 3, 4, 5, 6, 7, 8, 9\}, \\
\mathcal{E} &= \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{5, 6\}, \{6, 7\}, \{7, 8\}, \\
&\qquad \{8, 1\}, \{2, 9\}, \{5, 9\}, \{7, 9\}\}.
\end{aligned}$$

□

Example 2.8 (Directed graph) The directed graph $\vec{\mathsf{X}} = \mathsf{X}(\mathcal{V}, \vec{\mathcal{E}})$ in Fig. 2.1 (middle) has

$$\begin{aligned}
\mathcal{V} &= \{1, 2, 3, 4, 5, 6, 7\}, \\
\vec{\mathcal{E}} &= \{(1, 3), (3, 2), (3, 4), (5, 4), (6, 4), (5, 6), (6, 5), (6, 7), (1, 7), (7, 1)\}.
\end{aligned}$$

□

We will need some more definitions concerning graphs. We start with some notions concerning the overall structure of a graph.

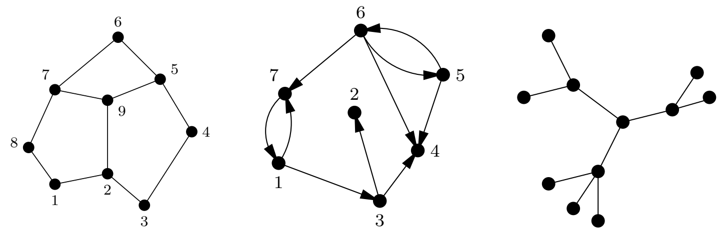

Figure 2.1: Left: undirected graph. Middle: directed graph. Right: (undirected) tree, i.e., (undirected) graph without cycles.

Definition 2.9 Let $\mathsf{X} = \mathsf{X}(\mathcal{V}, \mathcal{E})$ be an undirected graph.

- **(Size)** The size of X is the number of vertices $|\mathcal{V}|$ of X. The size can be finite or infinite.
- **(Components)** A graph consists of different components. Two vertices are in the same component if and only if they can be connected by a walk (see Def. 2.11). We will usually be interested in graphs with one component.
- **(Tree)** A graph containing no cycles is called a tree. Fig. 2.1 (right) gives an example of a tree.
- **(Forest)** If all components of a graph are trees, the graph is called a forest.

□

In a further step, we need definitions about the local structure of a graph.

Definition 2.10 Let $\mathsf{X} = \mathsf{X}(\mathcal{V}, \mathcal{E})$ be an undirected graph.

- **(Distance)** The distance between two vertices of the same component is the length of the shortest walk (see Def. 2.11) connecting them. If the vertices belong to different components, the distance is defined to be infinite.
- **(Degree)** The degree $\deg(v)$ of a vertex v in X is the number of edges incident at vertex v.

- **(Regular graph)** If the degree of all vertices is some constant k, then the graph is a k–regular graph.
- **(Leaf)** A leaf is a vertex of degree 1.
- **(Adjacency)** Two vertices are called *adjacent* if they are connected by an edge.
- **(Adjacency matrix)** The adjacency matrix $\mathbf{A}(\mathsf{X})$ of X is a square matrix where the rows and the columns are indexed by $\mathcal{V}$. The entry in row $v \in \mathcal{V}$ and column $w \in \mathcal{V}$ is 1 if v and w are adjacent, 0 otherwise. Obviously, $\mathbf{A}(\mathsf{X})$ is a symmetric matrix.

□

Cycles and their length will be very important for the construction of codes on graphs; the following definitions introduce the relevant notions.

Definition 2.11 Let $\mathsf{X} = \mathsf{X}(\mathcal{V}, \mathcal{E})$ be an undirected graph.

- **(Walk)** A walk of length k is a sequence of steps along adjacent vertices $v_0, v_1, \ldots, v_k$ such that[3] $v_{i-1} \neq v_{i+1}$ for $1 \leq i \leq k-1$. If all vertices are distinct, a walk is called a path.
- **(Cycle)** A walk $v_0, v_1, \ldots, v_r$ whose vertices are all distinct except that $v_0 = v_r$ is called a cycle. It has r distinct vertices and r distinct edges, and we often speak of an r–cycle, or a cycle of length r.
- **(Eulerian walk)** An Eulerian walk is a walk which uses every edge of the graph exactly once.
- **(Hamiltonian cycle)** An Hamiltonian cycle is a cycle that visits every vertex of the graph exactly once.
- **(Girth)** The girth $g(\mathsf{X})$ is the length of the shortest cycle in X.
- **(Local girth)** The local girth $g(\mathsf{X}, v)$ of a vertex v is the length of the shortest cycle going through v.
- **(Local girth histogram)** The local girth histogram is the histogram of the local girth over all vertices of X.

□

[3] For this last condition we follow the definition of Margulis [82]. Biggs [15] does not have this additional condition for a walk.

Finally, we need some additional quantities that characterize the structure of a graph.

Definition 2.12 Let $\mathsf{X} = \mathsf{X}(\mathcal{V}, \mathcal{E})$ be an undirected graph.

- **(Diameter)** The diameter $\mathrm{diam}(\mathsf{X})$ is the largest distance between any two vertices. The diameter is possibly infinite.
- **(Independence number)** The independence number $i(\mathsf{X})$ is the size of the largest set of vertices such that no pair of them is adjacent.
- **(Chromatic number)** The chromatic number $\chi(\mathsf{X})$ is the minimal number of colors needed to color the vertices such that each pair of adjacent vertices has different colors.
- **(Bipartite-ness)** If X has chromatic number 2, it is a bipartite graph. One way to emphasize the bipartite-ness is to use different vertex shapes for the two vertex classes, as is done e.g. in the case of factor and Tanner graphs (see Ch. 3, e.g. Fig. 3.8).
- **(Eigenvalues, spectrum)** Because the adjacency matrix $\mathbf{A}(\mathsf{X})$ of X is symmetric, it has n real eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$. We order them according to $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. The collection of the eigenvalues is called the spectrum of the graph.[4]
- **(Second-largest eigenvalue)** For a k–regular graph we denote by $\lambda(\mathsf{X})$ the absolute value of the largest eigenvalue (in absolute value) of $\mathbf{A}(\mathsf{X})$ which is distinct from $\pm k$; in other words $\lambda(\mathsf{X})^2$ is the next to largest eigenvalue of $\mathbf{A}(\mathsf{X})^2$. (The reason why the largest eigenvalue is less important is given in Lemma 2.13.)

□

Lemma 2.13 *Let $\mathsf{X}_{n,k} = \mathsf{X}(\mathcal{V}, \mathcal{E})$ be a k–regular graph on n vertices. If $\mathbf{A}(\mathsf{X}_{n,k})$ is its adjacency matrix and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ its eigenvalues then $|\lambda_j| \leq k$. In fact $\lambda_1 = k$, and $\mathsf{X}_{n,k}$ is bipartite if and only if $\lambda_n = -k$. If $\mathsf{X}_{n,k}$ is connected, then $\lambda_1 = k$ and $\lambda_n = -k$ (in the bipartite case) are both simple eigenvalues as is easily verified.*

Proof: See e.g. [16]. For more about the largest eigenvalue and its multiplicity, see also Lemma E.12. □

[4] The part of graph theory that tries to characterize graphs via their spectra is called algebraic graph theory, see e.g. [16].

Definition 2.14 (Ramanujan Graph, see e.g. [65]) A k–regular graph $\mathsf{X}_{n,k}$ of size n is called a *Ramanujan* graph if

$$\lambda(\mathsf{X}_{n,k}) \leq 2\sqrt{k-1}.$$

□

Remark 2.15 (Eigenvalue bound) For a fixed $k \geq 2$, the importance of the number $2 \cdot \sqrt{k-1}$ in the above definition lies in the following lower bound due to Alon and Boppana [3]

$$\liminf_{n\to\infty} \lambda(\mathsf{X}_{n,k}) \geq 2\sqrt{k-1}.$$

But this is only for $n \to \infty$. So, for finite n, a graph might have a $\lambda(\mathsf{X}_{n,k})$ below this bound and will consequently be called a Ramanujan graph.

□

Definition 2.16 (Expansion coefficient) Let $\mathcal{A} \subseteq \mathcal{V}$ be a subset of the vertex set. The boundary $\partial\mathcal{A}$ of $\mathcal{A}$ is defined to be $\partial\mathcal{A} \triangleq \{v \in \mathcal{V} \mid d(v,\mathcal{A}) = 1\}$ where $d(v,\mathcal{A})$ is the distance of a vertex v from the set $\mathcal{A}$ in the graph. A graph is called an expander if it has the property that every $\mathcal{A} \subseteq \mathcal{V}$ has a large boundary. More precisely, an (n,k,c)-expander is a k-regular graph $\mathsf{X}_{n,k} = \mathsf{X}(\mathcal{V},\mathcal{E})$ on n vertices for which every $\mathcal{A} \subseteq \mathcal{V}$ with $|\mathcal{A}| \leq n/2$ satisfies $|\partial\mathcal{A}| \geq c|\mathcal{A}|$; c is called the expansion coefficient. A typical result relating the expansion coefficient and the spectrum of the graph is given below in Lemma 2.17. (Without going into the details we note that there are other reasonable definitions of expansion coefficients available in the literature leading to similar results.) □

Lemma 2.17 (Expansion coefficient) *Let $\mathsf{X}_{n,k} = \mathsf{X}(\mathcal{V},\mathcal{E})$ be a k-regular graph on n vertices. Then $\mathsf{X}_{n,k}$ is an (n,k,c)-expander with expansion coefficient $c = \frac{1}{2}(1 - \lambda(\mathsf{X}_{n,k})/k)$.*

Proof: See e.g. the proof given in the book [116]. □

There are a variety of results along the same line, i.e. which try to quantify the expansion property of a graph in terms of the spectrum. The first such statement was given by Tanner [110]. Papers with similar results include [4] or [52, 53](where the author inter alia discusses what, given the knowledge of the second largest eigenvalue, the best obtainable bounds on the expansion coefficient are) or [107].

Remark 2.18 (Eigenvalues, expansion coefficient, Ramanujan graphs) Lemma 2.17 states that the smaller $\lambda(\mathsf{X})$ is, the larger is the expansion coefficient c that one can guarantee. Therefore, Ramanujan graphs as defined in Def. 2.14 give good expanders because $\lambda(\mathsf{X})$ is below a certain critical bound for such graphs. □

Lemma 2.19 (Diameter vs. number of distinct eigenvalues) *A connected graph $\mathsf{X} = \mathsf{X}(\mathcal{V},\mathcal{E})$ with diameter* $\mathrm{diam}(\mathsf{X})$ *has an adjacency matrix with at least* $\mathrm{diam}(\mathsf{X}) + 1$ *distinct eigenvalues.*

Proof: See e.g. [16]. □

Lemma 2.19 is especially useful for upper bounding the diameter of graphs stemming from finite geometries as the adjacency matrix of such graphs has very often only very few distinct eigenvalues.

Finally, we need some definitions for directed graphs.

Definition 2.20 Let $\vec{\mathsf{X}} = \vec{\mathsf{X}}(\mathcal{V},\vec{\mathcal{E}})$ be a directed graph.

- **(Out-degree)** The out-degree $\mathrm{outdeg}(v)$ of a vertex v in $\vec{\mathsf{X}}$ is the number of edges going out at vertex v.
- **(In-degree)** The in-degree $\mathrm{indeg}(v)$ of a vertex v in $\vec{\mathsf{X}}$ is the number of edges going in at vertex v.

□

2.2.2 Cayley Graphs

A special class of graphs are the Cayley graphs. They can be directed or undirected. To define a Cayley graph we only need a group and a subset of elements from this group. (Some group theory is reviewed in Sec. A.1.)

Definition 2.21 (Directed Cayley graph) Let $\langle\mathcal{G},\circ\rangle$ be a group with element set $\mathcal{G}$ and operation $\circ$. Let a set $\mathcal{S} \subseteq \mathcal{G}$ be a subset of $\mathcal{G}$; the set $\mathcal{S}$ will be called the *generator set*, see also the comment in Prop. 2.26. The directed Cayley graph $\vec{\mathsf{X}}(\mathcal{G},\mathcal{S})$ has

- vertex set $\mathcal{V} = \mathcal{G}$, i.e. for each group element we draw a vertex (for easiness of notation we let the vertex label be the corresponding group element),

- edge set $\vec{\mathcal{E}} = \{(v, v \circ s) \mid v \in \mathcal{V}, s \in \mathcal{S}\}$, i.e., there is a directed edge from vertex v to vertex w if and only if $w = v \circ s$ for some $s \in \mathcal{S}$.

$\square$

Example 2.22 (A directed Cayley graph) Fig. 2.2 (middle) shows a directed Cayley graph based on the integers modulo 8, i.e. $\mathcal{G} = \mathbb{Z}/8\mathbb{Z}$, and the set $\mathcal{S} = \{-1, +1\}$. E.g. there is an edge from 4 to 3 as we have $3 = 4 + (-1)$. (Fig. 2.2 (left) shows only the vertex set.) $\square$

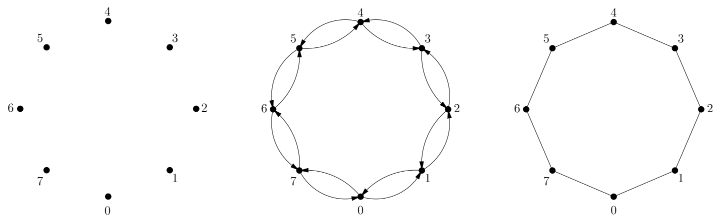

Figure 2.2: Left: for each element of $\mathbb{Z}/8\mathbb{Z}$ we draw a vertex. Middle: directed Cayley graph $\vec{\mathsf{X}}(\mathbb{Z}/8\mathbb{Z}, \{-1, +1\})$. Right: undirected Cayley graph $\mathsf{X}(\mathbb{Z}/8\mathbb{Z}, \{-1, +1\})$.

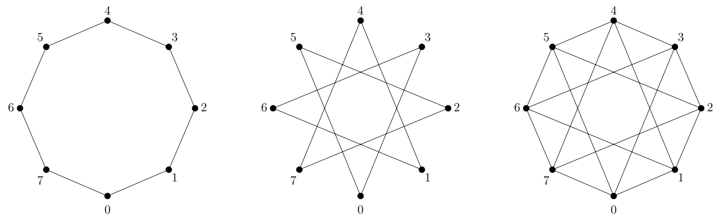

Figure 2.3: Left: undirected Cayley graph $\mathsf{X}(\mathbb{Z}/8\mathbb{Z}, \{-1, +1\})$. Middle: undirected Cayley graph $\mathsf{X}(\mathbb{Z}/8\mathbb{Z}, \{-3, +3\})$. Right: undirected Cayley graph $\mathsf{X}(\mathbb{Z}/8\mathbb{Z}, \{-3, -1, +1, +3\})$.

It is not difficult to see that for a directed Cayley graph $\vec{\mathsf{X}}(\mathcal{G}, \mathcal{S})$ we have $\text{outdeg}(v) = \text{indeg}(v) = |\mathcal{S}|$ for every $v \in \mathcal{V}$. We call the set $\mathcal{S}$ *symmetric* (denoted by $\mathcal{S} = \mathcal{S}^{-1}$) if $s \in \mathcal{S}$ implies $s^{-1} \in \mathcal{S}$. Let $\vec{\mathsf{X}}(\mathcal{G}, \mathcal{S})$ be a directed Cayley graph with symmetric set $\mathcal{S}$: if there is an edge from

v to $w = v \circ s$ (for some $s \in \mathcal{S}$) there must also be an edge from w to $v = w \circ s^{-1}$. Replacing each pair of *directed* edges by a single *undirected* edge we get an undirected Cayley graph as given in the next definition.

Definition 2.23 (Undirected Cayley graph) Let $\langle \mathcal{G}, \circ \rangle$ be a group with element set $\mathcal{G}$ and operation $\circ$. Let the *generator set* $\mathcal{S} \subseteq \mathcal{G}$ be *symmetric*. Then, the undirected Cayley graph $\mathsf{X}(\mathcal{G}, \mathcal{S})$ has

- vertex set $\mathcal{V} = \mathcal{G}$, i.e. for each group element we draw a vertex (for easiness of notation we let the vertex label be the corresponding group element),
- edge set $\mathcal{E} = \{\{v, v \circ s\} \mid v \in \mathcal{V}, s \in \mathcal{S}\}$, i.e., there is an undirected edge from vertex v to vertex w if and only if $w = v \circ s$ and $v = w \circ s^{-1}$ for some $s \in \mathcal{S}$.

$\square$

Example 2.24 (An undirected Cayley graph) Fig. 2.2 (right) shows the undirected Cayley graph based on the integers modulo eight, i.e. $\mathcal{G} = \mathbb{Z}/8\mathbb{Z}$, and the symmetric set $\mathcal{S} = \{-1, +1\}$. This graph can be derived from the directed Cayley graph in Fig. 2.2 (middle) by replacing each pair of directed edges by a single undirected edge.

Fig. 2.3 (middle) shows an undirected Cayley graph based on the same group $\mathcal{G} = \mathbb{Z}/8\mathbb{Z}$ but now with $\mathcal{S} = \{-3, +3\}$. The undirected Cayley graph $\mathsf{X}(\mathbb{Z}/8\mathbb{Z}, \{-3, -1, +1, +3\})$ in Fig. 2.3 (right) can be seen as the "superposition" of the undirected Cayley graphs $\mathsf{X}(\mathbb{Z}/8\mathbb{Z}, \{-1, +1\})$ and $\mathsf{X}(\mathbb{Z}/8\mathbb{Z}, \{-3, +3\})$ in Fig. 2.3 (left and middle). $\square$

Example 2.25 (Another undirected Cayley graph) Fig. 4.4 on page 61 shows an example of an undirected Cayley graph $\mathsf{X}(\mathcal{G}, \mathcal{S})$ based on the group $\mathcal{G}$ and the set $\mathcal{S} = \{\mathbf{A}, \mathbf{B}, \mathbf{A}^{-1}, \mathbf{B}^{-1}\}$, where the group $\mathcal{G}$ is freely generated (see Def. A.3) by $\mathcal{T} = \{\mathbf{A}, \mathbf{B}\}$. $\mathsf{X}(\mathcal{G}, \mathcal{S})$ must be a tree because each cycle would correspond to a non-trivial relation involving the elements of $\mathcal{T}$ and their inverses; but by the freeness of $\mathcal{G}$ this is not possible. The size of the graph must obviously be infinite. $\square$

Properties 2.26 (of Cayley Graphs)

- *It is not difficult to see that for an undirected Cayley graph* $\mathsf{X}(\mathcal{G}, \mathcal{S})$ *we have* $\deg(v) = |\mathcal{S}|$ *for every* $v \in \mathcal{V}$.

- *A symmetric set $\mathcal{S}$ need not have an even number of vertices as the example $\mathcal{S} = \{-1, +1, 4\}$ for the integers modulo 8 shows.*
- *In group theory, a set $\mathcal{S}$ is a generator set of a group $\mathcal{G}$ if all elements of $\mathcal{G}$ can be expressed as words in $\mathcal{S}$. In the context of Cayley graphs we do not require that the generator set $\mathcal{S}$ generates the whole group $\mathcal{G}$. If it does, then the Cayley graph $\mathsf{X}(\mathcal{G}, \mathcal{S})$ has one component; if it does not, then the number of components of the Cayley graph $\mathsf{X}(\mathcal{G}, \mathcal{S})$ is given by the index of the subgroup generated by $\mathcal{S}$ relative to the group $\mathcal{G}$.*
- *The graph $\mathsf{X}(\mathcal{G}, \mathcal{S})$ is* homogeneous (vertex-transitive) *since left multiplication by any element of $\mathcal{G}$ preserves the adjacency relation in this graph.*

Cayley graphs have the nice feature that they can easily be defined and therefore also stored; one only needs to remember the group $\mathcal{G}$ (which can often be defined in a systematic way) and the set $\mathcal{S}$ (which in all our applications will be small).[5] This simplicity in description will also be a crucial facet of the codes to be described in Ch. 5.

To get undirected Cayley graphs (of degree larger than two) with large girth, the underlying group must be *non-abelian*. This follows from the observation that in $\mathsf{X}(\mathcal{G}, \mathcal{S})$ with $\mathcal{G}$ abelian and $s_1, s_2 \in \mathcal{S}$, where $s_1 \neq s_2$, the walk $v, v \circ s_1, v \circ s_1 \circ s_2, v \circ s_1 \circ s_2 \circ s_1^{-1}, v \circ s_1 \circ s_2 \circ s_1^{-1} \circ s_2^{-1} = v$ is a cycle of length four as $s_1 \circ s_2 \circ s_1^{-1} \circ s_2^{-1}$ equals the neutral element of $\mathcal{G}$. Some of the most simply describable non-abelian groups are groups based on two-by-two matrices under matrix multiplication; most constructions to follow will in fact be based on such groups (see App. A).

[5]Note that when the generator set $\mathcal{S}$ generates the whole group $\mathcal{G}$, i.e. when the Cayley graph has only one component, then, strictly speaking, we actually only need to store the generator set $\mathcal{S}$.

Chapter 3

Factor Graphs and the Summary-Product Algorithm

3.1 Introduction and Notation

3.1.1 Introduction

This chapter aims at giving an introduction to factor graphs and the summary-product algorithm as defined in [128] or [58]. Early versions of the summary-product algorithm can be traced back to [38, 109]. Complementary to this introduction, both references [128] and [58] can be highly recommended as tutorial introductions, the paper [58] gives also a historical overview and relations to algorithms in other fields of science.[1]

Factor graphs and the summary-product algorithm can be seen as an instance of a graphical model.

[1]More papers on the subject can also be found in the Feb. 2001 IEEE Transactions on Information Theory issue [47].

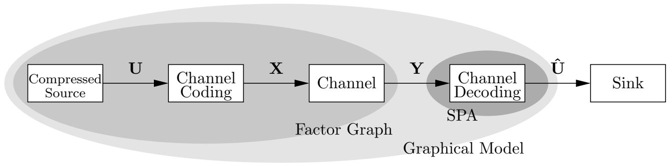

Figure 3.1: A graphical model consists of a graph (here: factor graph) and an algorithm (here: summary-product algorithm, SPA) operating on it. (The notation will be explained later.)

Definition 3.1 (Graphical Model, see e.g. [35]) A graphical model consists of two parts:

- a *graph* (here: factor graph), and
- an *algorithm* which is operating on the graph (here: summary-product algorithm, SPA[2]).

□

We will be interested in representing a digital data transmission system with the help of a factor graph and solving its associated decision problems with the help of the SPA. Without going into the notational details, Fig. 3.1 shows that the factor graph will model the (compressed) source, the channel coding, and the channel, whereas the summary-product algorithm will be the channel decoding algorithm that operates on the factor graph. In formulas, the decision about a symbol based on symbol-wise decoding can be structured as

$$\hat{u}_i^{\text{symbol}}(\mathbf{y}) = \underbrace{\underbrace{\operatorname*{argmax}_{u_i \in \mathcal{U}}}_{\text{Decision taking}} \underbrace{\underbrace{\sum_{\substack{\mathbf{u} \in \mathcal{U}^k,\ \mathbf{x} \in \mathcal{X}^n \\ u_i \text{ fixed}}}}_{\text{Sum}} \underbrace{P_{\mathbf{UXY}}(\mathbf{u}, \mathbf{x}, \mathbf{y})}_{\substack{\text{Factor graph} \\ \text{(written as product)}}}}_{\text{Sum-product algorithm}}}_{\text{Decision about symbol } u_i \text{ based on symbol-wise decoding}},$$

[2]By the summary-product algorithm we will mean the generic algorithm defined over any semi-ring [58]. When the semi-ring is $\langle \mathcal{R}, +, \cdot \rangle$ then we will call it the sum-product algorithm, and when the semi-ring is $\langle \mathcal{R}, \max, \cdot \rangle$ we will call it the max-product algorithm, see also Rem. 3.30. Note that SPA will stand for summary-product algorithm or sum-product algorithm; from the context it should be clear which one is meant.

whereas a decision about a symbol based on block-wise decoding can be structured as

$$\hat{u}_i^{\text{block}}(\mathbf{y}) = \underbrace{\underbrace{\operatorname*{argmax}_{u_i \in \mathcal{U}}}_{\text{Decision taking}} \underbrace{\underbrace{\max_{\substack{\mathbf{u} \in \mathcal{U}^k, \mathbf{x} \in \mathcal{X}^n \\ u_i \text{ fixed}}}}_{\text{Maximum}} \underbrace{P_{\mathbf{UXY}}(\mathbf{u}, \mathbf{x}, \mathbf{y})}_{\substack{\text{Factor graph} \\ \text{(written as product)}}}}_{\text{Max-product algorithm}}}_{\text{Decision about symbol } u_i \text{ based on block-wise decoding}}.$$

(The mathematical details will be explained later in this chapter.) The first decision problem will be solved with the sum-product algorithm, whereas the second decision problem will be solved with the max-product algorithm.

For our purposes we could also use normal graphs introduced by Forney [31]: they turn out to be the natural representation when it comes to Fourier transforms [81] and Legendre transforms [122, 124]. Yet another possibility of a graphical representation is given in [2].

In this chapter we take the following approach:

- Sec. 3.2 introduces the setup of digital data transmission in a classical framework and formulates it as a decision problem.
- Sec. 3.3 gives an introduction to factor graphs.
- Sec. 3.4 introduces the summary-product algorithm and shows how this algorithm can help us to solve the decision problems we are interested in.
- In Sec. 3.5 we reformulate the digital data transmission problem in terms of factor graphs and the summary-product algorithm

3.1.2 Notation

Definition 3.2 (Iverson's convention) Let S be a statement. Using Iverson's convention, we let $[S]$ be equal to 1 if the statement is true and equal to 0 otherwise. □

Definition 3.3 (Chance variable) A chance variable is similar to a random variable except that it assumes values in a general set instead

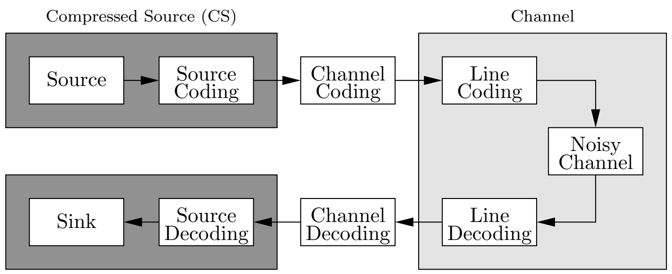

Figure 3.2: Digital data transmission over a noisy channel.

Figure 3.3: Simplified Fig. 3.2.

of the set of reals. Note that notions like expectation do not make sense for chance variables. □

Assumption 3.4 *We assume here that all sets like $\mathcal{U}$, $\mathcal{X}$, and $\mathcal{Y}$ are finite: this allows us to use only probability mass functions (pmfs). It is not difficult to generalize our exposition to the case where e.g. $\mathcal{Y} = \mathbb{R}$: we then have to use also probability densities.*

3.2 Model for Digital Data Transmission over a Noisy Channel

3.2.1 Introduction

Assume that we are interested in transmitting data from a point A to a point B or that we would like to store some information that we would like to retrieve later on.[3] In both cases we desire that the received and the retrieved data is identical to the data transmitted or stored, or, if

[3] For a classical treatment of this subject, see e.g. [93].

errors cannot be avoided, that there are as few errors as possible. As we have mentioned in Ch. 1, it is indeed possible to transmit data with as few errors as desired if one is willing to use (very long) channel codes of rates smaller than capacity.

Fig. 3.2 shows the typical blocks of a model for digital data transmission over a noisy channel:

- **(Source/source coding)** The data we would like to transmit is produced by a source. Its output can be compressed (losslessly or lossy) by a source coding scheme to reduce the amount of data to be transmitted. The output of such a compressed source can be modeled as a binary i.i.d. source, but neither the binary nor the i.i.d. property are crucial for the factor graph setup, as we will see later on. Indeed, factor graphs allow one relatively easily to incorporate a general source model, e.g. the output of a Markov source.
- **(Channel coding)** Here we apply channel coding to the data coming out of the compressed source in order to increase the reliability of the transmission, i.e. to decrease the number of errors at the receiver.
- **(Line coding/noisy channel/line decoding)** In order to send data over a physical medium we apply modulation techniques. Here we use the term "line coding" for modulation and "line decoding" for demodulation. The concatenation of the three blocks line coding, noisy channel, and line decoding is called the channel.
- **(Channel decoding)** Here we try to eliminate transmission errors that might have occurred during transmission.
- **(Source decoding/sink)** Before the data can reach its final destination we uncompress it to its original size.

The block diagram in Fig. 3.2 can be simplified to the block diagram in Fig. 3.3:

- **(Compressed source)** Let $\mathbf{u} = (u_1, u_2, \ldots, u_k)$, where $u_i \in \mathcal{U}$, be the vector of k consecutive source output symbols.
- **(Channel coding)** The channel coding block takes the vector $\mathbf{u}$ of length k and produces a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ of length n where $x_i \in \mathcal{X}$.

$\mathbf{u}$	$\mapsto$	$\mathbf{x} = \mathbf{x}(\mathbf{u})$
$(0,0)$	$\mapsto$	$(0,0,0)$
$(1,0)$	$\mapsto$	$(1,1,0)$
$(0,1)$	$\mapsto$	$(0,1,1)$
$(1,1)$	$\mapsto$	$(1,0,1)$

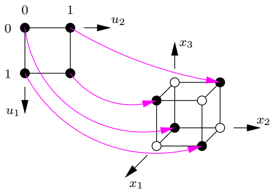

Figure 3.4: Example with $\mathcal{U} = \mathcal{X} = \{0,1\}$, $k = 2$, $n = 3$. Left: Mapping of $\mathbf{u}$ to $\mathbf{x}$ for a binary linear $[3,2,2]$ code. Right: graphical visualization of that mapping.

- **(Channel)** The channel takes as input the vector $\mathbf{x}$ and we observe at the output the vector $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ of length n where $y_n \in \mathcal{Y}$.
- **(Channel decoding)** Based on the observation $\mathbf{y}$ our estimate about $\mathbf{u}$ is $\hat{\mathbf{u}} = (\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_k)$ of length k with $\hat{u}_i \in \mathcal{U}$.[4] Instead of estimating $\mathbf{u}$, we are possibly interested in an estimate $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$ about $\mathbf{x}$, where $\hat{x}_k \in \mathcal{X}$.

We assume that this process is repeated for many blocks and that the chance variables appearing are independent from block to block. (Alternatively, one could consider the case where we let $k \to \infty$ with a fixed ratio k/n.)

3.2.2 Classical Representation of the Different Blocks

We take a two-step approach at discussing and characterizing in depth the various aspects of Fig. 3.3. In a first step, this section treats the various blocks in a way that resembles the approaches taken in classical texts about channel coding. The second step will then be made in Sec. 3.5 where we present the different components in a unified manner.

- **(Stationary memoryless source)** A memoryless source produces i.i.d. chance variables U_i, $i = 1, \ldots, k$, in $\mathcal{U}$. Therefore the joint

[4] Sometimes it makes sense that the alphabet of $\hat{u}_i$ is larger than the alphabet of u_i, but we do not consider this case here.

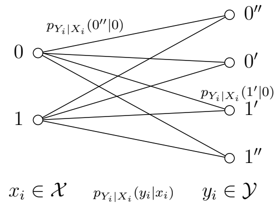

Figure 3.5: Example of a channel diagram. (This is *not* a factor graph.)

pmf of $\mathbf{U}$ is

$$P_{\mathbf{U}}(\mathbf{u}) = \prod_{i=1}^{k} P_{U_i}(u_i).$$

- **(Symmetric source)** A symmetric source is a stationary memoryless source where additionally all symbols are equiprobable, i.e., $P_{U_i}(u_i) = 1/|\mathcal{U}|$ for all $u_i \in \mathcal{U}$. In this case, all sequences are equally probable, i.e., $P_{\mathbf{U}}(\mathbf{u}) = 1/|\mathcal{U}|^k$ for all $\mathbf{u} \in \mathcal{U}^k$. If $|\mathcal{U}| = 2$, the symmetric source is a binary symmetric source (BSS).
- **(Channel coding)** The art of channel coding is to find a "good" function (mapping) from the space $\mathcal{U}^k$ to the space $\mathcal{X}^n$ such that the points in $\mathcal{X}^n$ are as "far apart" as possible. This deterministic mapping can be formulated with the help of the conditional probabilities[5]

$$P_{\mathbf{X}|\mathbf{U}}(\mathbf{x}|\mathbf{u}) = \big[\ \mathbf{x} \text{ is the codeword corresponding to } \mathbf{u}\ \big],$$

  i.e., $P_{\mathbf{X}|\mathbf{U}}(\mathbf{x}|\mathbf{u})$ is a "deterministic" conditional probability.
- **(Linear block codes)** Linear block codes were introduced in Sec. 2.1. They can be represented by a generator matrix $\mathbf{G}$ or a parity-check matrix $\mathbf{H}$.

Example 3.5 (Binary linear block code) Fig. 3.4 shows a small example of a code of length $n = 3$, dimension $k = 2$, redundancy

[5] Here we use Iverson's convention (see Def. 3.2).

$n-k=1$, rate $R=k/n=2/3$, and alphabets $\mathcal{U}=\mathcal{X}=\mathbb{F}_2$ with generator and parity-check matrices

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

where the mapping $\mathbf{u}\mapsto\mathbf{x}(\mathbf{u})=\mathbf{u}\cdot\mathbf{G}$ is given in the table in Fig. 3.4. □

- **(Channel)** A channel is characterized by the channel law $P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$ which is the conditional probability of $\mathbf{y}$ given $\mathbf{x}$.
- **(Memoryless channel)** In the case of a memoryless channel we can write the channel law $P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$ as

$$P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})=\prod_{i=1}^{n}P_{Y_i|X_i}(y_i|x_i).$$

A possible channel can be represented by a diagram as in Fig. 3.5. In this thesis, we consider mainly memoryless channel models like the binary-input additive white Gaussian noise channel (BI-AWGN-C) or the binary symmetric channel (BSC). The input to the BI-AWGNC is ±1, the output is a real number, and the quality of the channel is characterized by the variance of the additive white Gaussian noise. The BSC has as input 0 or 1, as output also 0 and 1, and is characterized by the cross-over probability.

- **(Channel decoding algorithm)** A channel decoding algorithm is a method that, based on the received channel output, gives back a codeword.
- **(Channel decoding)** Channel decoding is the art of finding a "good" and efficient decoding algorithm for a given channel code and a given channel. By "good" we mean that the probability of error should be as small as possible. Note that nearly all algorithms known today that can decode efficiently classical codes can only process the hard information of the channel output.[6] A notable exception to this rule are convolutional codes that can be decoded using the Viterbi algorithm; but this algorithm works only practically if the memory size of the convolutional code is not too large.

[6] E.g. the case of a BI-AWGNC: the hard information is the sign of the channel output, whereas the soft information is the magnitude of the channel output.

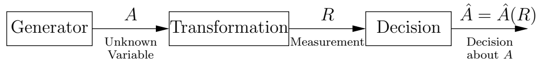

Figure 3.6: Setup for decision problems.

3.2.3 Decision Theory

The standard decision problem is depicted in Fig. 3.6. Let A be a chance variable with alphabet $\mathcal{A}$ where $|\mathcal{A}|<\infty$. Based on a measurement of a chance variable R with alphabet $\mathcal{R}$ we can make a decision $\hat{A}=\hat{A}(R)$ about A. We assume that R is a function of A (and possibly some other chance variables). Note that A and R can be chance variables, vectors, or matrices. This section discusses different possibilities to give a decision $\hat{a}(r)$ based on a measurement $R=r$.[7]

Lemma 3.6 (Maximum a-posteriori) *If we minimize the error probability $P_{\text{error}}=P\big[\hat{A}(R)\neq A\big]$, or equivalently if we maximize the probability of a correct decision $P_{\text{correct}}=P\big[\hat{A}(R)=A\big]=1-P_{\text{error}}$, then we obtain the maximum a-posteriori decision function $\hat{a}_{\text{MAP}}:\mathcal{R}\to\mathcal{A}$ with*[8]

$$\hat{a}_{\text{MAP}}(r)=\operatorname*{argmax}_{a\in\mathcal{A}}\ P_{A|R}(a|r)\overset{(*)}{=}\operatorname*{argmax}_{a\in\mathcal{A}}\ P_{AR}(a,r),$$

where equality $(*)$ follows from $P_{A|R}(a|r)=P_{AR}(a,r)/P_R(r)$.

Remark 3.7 (Maximum likelihood) A related rule is the maximum likelihood decision rule $\hat{a}_{\text{ML}}(r)$ where

$$\hat{a}_{\text{ML}}(r)=\operatorname*{argmax}_{a\in\mathcal{A}}\ P_{R|A}(r|a).$$

It is well known that $\hat{a}_{\text{MAP}}(r)=\hat{a}_{\text{ML}}(r)$ for all $r\in\mathcal{R}$ if $P_A(a)=\text{const.}$ for all $a\in\mathcal{A}$. □

We now discuss the maximum a-posteriori decision rule for several interesting scenarios important in coding theory.

[7] If the set $\mathcal{A}$ were continuous, one would call $\hat{A}(R)$ an estimate.

[8] Strictly speaking, argmax returns a set. If the set contains several elements, then we randomly pick one of these; otherwise, we return the single element.

Example 3.8 (Block-wise decoding) We set

$$\begin{aligned} A &= \mathbf{U} && \text{(vector to be estimated)} \\ R &= \mathbf{Y} && \text{(measurement)} \\ \hat{A} &= \hat{\mathbf{U}}^{\text{block}} && \text{(estimate).} \end{aligned}$$

Minimization of $P_{\text{error}} = P_{\text{block}} = P[\hat{\mathbf{U}} \neq \mathbf{U}]$ leads to the decision rule

$$\hat{\mathbf{u}}^{\text{block}}(\mathbf{y}) = \operatorname*{argmax}_{\mathbf{u} \in \mathcal{U}^k} P_{\mathbf{UY}}(\mathbf{u}, \mathbf{y}) = \operatorname*{argmax}_{\mathbf{u} \in \mathcal{U}^k} \max_{\mathbf{x} \in \mathcal{X}^n} P_{\mathbf{UXY}}(\mathbf{u}, \mathbf{x}, \mathbf{y}).$$

The introduction of the chance vector $\mathbf{X}$ does not change the problem as $\mathbf{x}$ is assumed to be a deterministic function of $\mathbf{U}$ (see Sec.3.2.2). The decision taken for component i ($1 \leq i \leq k$) of $\hat{\mathbf{u}}^{\text{block}}(\mathbf{y})$ can also be written as

$$\hat{u}_i^{\text{block}}(\mathbf{y}) = \operatorname*{argmax}_{u_i \in \mathcal{U}} \max_{\substack{\mathbf{u} \in \mathcal{U}^k \\ u_i \text{ fixed}}} P_{\mathbf{UY}}(\mathbf{u}, \mathbf{y}) = \operatorname*{argmax}_{u_i \in \mathcal{U}} \max_{\substack{\mathbf{u} \in \mathcal{U}^k, \mathbf{x} \in \mathcal{X}^n \\ u_i \text{ fixed}}} P_{\mathbf{UXY}}(\mathbf{u}, \mathbf{x}, \mathbf{y}).$$

Note: The well-known Viterbi-Algorithm (VA) efficiently performs the task of finding the block-wise estimate in the case of convolutional or trellis codes. □

Example 3.9 (Symbol-wise decoding) For each $i = 1, \ldots, k$ we set

$$\begin{aligned} A &= U_i && \text{(variable to be estimated)} \\ R &= \mathbf{Y} && \text{(measurement)} \\ \hat{A} &= \hat{U}_i^{\text{symbol}} && \text{(estimate).} \end{aligned}$$

Minimization of $P_{\text{error}} = P_{\text{symbol}}^{(i)} = P[\hat{U}_i \neq U_i]$ leads to the decision rules

$$\begin{aligned} \hat{u}_i^{\text{symbol}}(\mathbf{y}) &= \operatorname*{argmax}_{u_i \in \mathcal{U}} P_{U_i \mathbf{Y}}(u_i, \mathbf{y}) \\ &= \operatorname*{argmax}_{u_i \in \mathcal{U}} \sum_{u_1} \cdots \sum_{u_{i-1}} \sum_{u_{i+1}} \cdots \sum_{u_k} \sum_{x_1} \cdots \sum_{x_n} P_{\mathbf{UXY}}(\mathbf{u}, \mathbf{x}, \mathbf{y}) \\ &= \operatorname*{argmax}_{u_i \in \mathcal{U}} \sum_{\substack{\mathbf{u} \in \mathcal{U}^k, \ \mathbf{x} \in \mathcal{X}^n \\ u_i \text{ fixed}}} P_{\mathbf{UXY}}(\mathbf{u}, \mathbf{x}, \mathbf{y}) \quad (i = 1, \ldots, k). \end{aligned}$$

□

We note that $P_{\mathbf{UXY}}(\mathbf{u}, \mathbf{x}, \mathbf{y})$ appears in the expressions for block-wise- and symbol-wise-decoding. But one performs different operations to get the block-wise/symbol-wise estimates. Later on, when we will use factor graphs, block-wise decoding will be done with the help of the max-product algorithm whereas symbol-wise decoding will be done with the help of the sum-product algorithm. In the case of loopy factor graphs one still applies the same algorithms but the results might not be the same as with "correct" block- or symbol-wise decoding as defined in Exs. 3.8 and 3.9.

3.3 Factor Graphs

3.3.1 A Casual Introduction to Factor Graphs

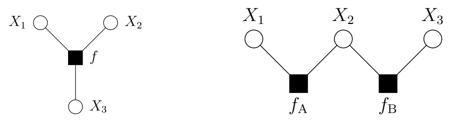

Figure 3.7: Examples of factor graphs.

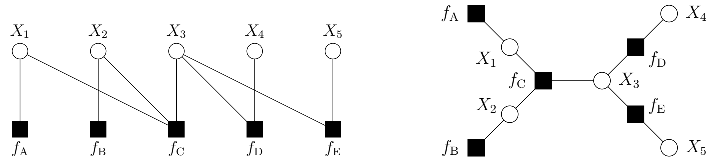

Figure 3.8: Factor graph (twice the same factor graph; the one on the right-hand side has been redrawn so that one sees the structure more clearly).

A factor graph represents a multivariate function. Fig. 3.7 (left) is the factor graph of the (global) function $f(x_1, x_2, x_3)$: (filled) squares represent function nodes and (empty) circles represent variable nodes. There is an edge between a function node and a variable node if and only if the corresponding variable is an argument of the corresponding function. Note that a factor graph is a *bipartite* graph: edges are only allowed between vertices of different types.

The concept of factor graphs becomes interesting as soon as we know more about the structure of the function to be represented, namely about its factorization. Let us assume that the above global function $f(x_1, x_2, x_3)$ factors as

$$f(x_1, x_2, x_3) = f_{\mathrm{A}}(x_1, x_2) \cdot f_{\mathrm{B}}(x_2, x_3).$$

Then we can take advantage of this additional knowledge: the factor graph in Fig. 3.7 (right) represents this factorization. We call f the *global* function and f_{A} and f_{B} the *local* functions.

Let us consider a second example. We assume that the global function $f(x_1, x_2, x_3, x_4, x_5)$ factors as

$$\begin{aligned} f(x_1,&x_2, x_3, x_4, x_5) \\ &= f_{\mathrm{A}}(x_1) \cdot f_{\mathrm{B}}(x_2) \cdot f_{\mathrm{C}}(x_1, x_2, x_3) \cdot f_{\mathrm{D}}(x_3, x_4) \cdot f_{\mathrm{E}}(x_3, x_5), \end{aligned}$$

with local functions f_{A}, f_{B}, f_{C}, f_{D}, and f_{E}. Two different drawings of its factor graph are shown in Figs. 3.8 (left and right). We observe that one and the same function can be represented by graphs with different (but graph-theoretically equivalent) drawings: some are more pleasing than others.

3.3.2 A More Formal Introduction to Factor Graphs

Definition 3.10 (Factor graph) In general, a factor graph representing a function f consists of variable nodes and function nodes and there are the following rules. (We assume the function f to be written as a product of factors.)

- **(Global function)** f is called the global function.
- **(Function nodes/local functions)** There is a function node for every factor. A factor is called a local function. Function nodes are drawn as filled squares.
- **(Variable/symbol nodes)** There is a variable node for every variable. Variable nodes are drawn as empty circles and labeled by a capitalized version of the variable name (see also Footnote 9).

- **(Bipartite-ness)** The function node representing some factor g is connected with the variable node X if and only if x is an argument of g.
- **(Configuration)** A configuration is a particular assignment of values to all variable nodes. Variables in factor graphs are denoted by capital letters, whereas particular assignments of values to variable nodes are denoted by small letters.[9]
- **(Configuration space)** The configuration space Ω is the set of all configurations: it is the domain of the global function f.
- **(Valid configuration)** A configuration $\omega \in \Omega$ will be called valid if $f(\omega) \neq 0$.

□

Properties 3.11 (of a factor graph)

- *In every fixed configuration, every variable has some definite value. We may therefore consider the variable nodes in a factor graph as functions of the configuration ω.*
- *A main application area of factor graphs is in the area of probabilistic models. (In this case, the sample space can usually be identified with the configuration space Ω.)*
- *Related to factor graphs are the normal graphs that were introduced by Forney [31]: in normal graphs the variable nodes are allowed to have only degree one or two.*[10] *This condition is far less restrictive than might appear at first sight.*[11]

More definitions and properties concerning factor graphs can be found in the tutorial paper [58]. See also [128, 31].

[9] This imitates the notation used in probability theory to denote chance/random variables and realizations thereof.

[10] The variable nodes are then not drawn anymore; the variables are directly associated to the edges.

[11] Results e.g. about duality of behaviors can much more elegantly be expressed in normal graphs than in factor graphs. But for our purposes in this thesis, the factor graphs as defined above are entirely sufficient.

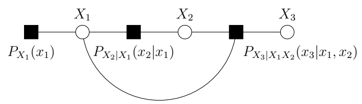

Figure 3.9: Factor graph representations of (3.1) and (3.2), respectively.

Example 3.12 (Markov chain) In probability theory, factorizations of joint probability measures are expressions of independence. E.g., let X_1, X_2, and X_3 be discrete chance variables with joint probability mass function $P_{X_1X_2X_3}(x_1,x_2,x_3)$. Then X_1, X_2, X_3 form a Markov chain if and only $P_{X_1X_2X_3}(x_1,x_2,x_3)$ can be factored as

$$\begin{aligned} P_{X_1X_2X_3}&(x_1,x_2,x_3) \\ &= P_{X_1}(x_1)\cdot P_{X_2|X_1}(x_2|x_1)\cdot P_{X_3|X_1X_2}(x_3|x_1,x_2) \qquad (3.1)\\ &= P_{X_1}(x_1)\cdot P_{X_2|X_1}(x_2|x_1)\cdot P_{X_3|X_2}(x_3|x_2). \qquad (3.2)\end{aligned}$$

This factorization is shown in Fig. 3.9: the left-hand side depicts the factor graph corresponding to (3.1), whereas the right-hand side shows the factor graph corresponding to (3.2).

Upon removal of the variable node X_2 and its adjacent edges, the factor graph in Fig. 3.9 (right) falls into two disconnected components, with X_1 and X_3 in different components, which expresses the conditional independence of X_1 and X_3 given X_2. It is easy to see that this generalizes to any factor graph of a joint pmf: conditioned on the variables in any (variable node) cut set of the graph, the variables in the two resulting components are independent. □

An introduction to factor graphs without showing the summary-product algorithm would be meaningless; therefore, the next section will give an overview over this algorithm.

3.4 The Summary-Product Algorithm

Instead of a general derivation of the summary-product algorithm, we discuss a simple example that contains all the important ingredients

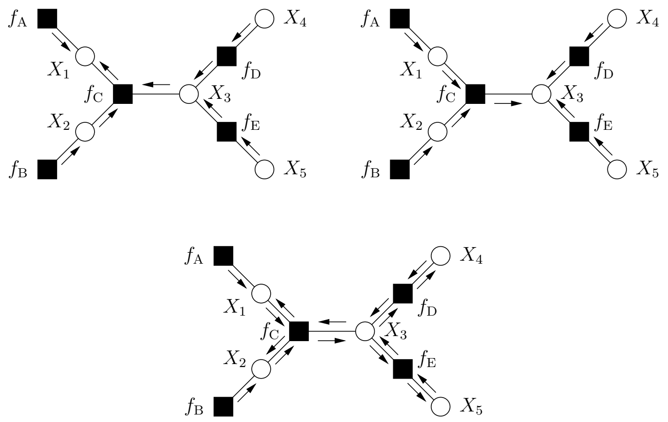

Figure 3.10: Top left: messages necessary for calculating $\tilde{f}_1(x_1)$. Top right: messages necessary for calculating $\tilde{f}_3(x_3)$. Bottom: messages necessary for calculating $\tilde{f}_1(x_1)$, $\tilde{f}_2(x_2)$, $\tilde{f}_3(x_3)$, $\tilde{f}_4(x_4)$, and $\tilde{f}_5(x_5)$.

in which we will be interested in afterwards. Basically, the summary-product algorithm can be defined for any commutative semi-ring with unity element, but here we focus on the standard case (called the sum-product algorithm) where this semi-ring is the real field, the sum is the usual real summation, and the product the usual real product. For the reader who is interested in more details, we recommend [128] or [58].

3.4.1 Factor Trees

First, we consider only factor trees, i.e. factor graphs that are trees; in Secs. 3.4.2 and 3.5.3 we extend our discussion to loopy factor graphs. Let us consider again the factor graph in Fig. 3.8(b) with the global function

$$\begin{aligned} f(x_1,x_2,&x_3,x_4,x_5) \\ &= f_{\mathrm{A}}(x_1)\cdot f_{\mathrm{B}}(x_2)\cdot f_{\mathrm{C}}(x_1,x_2,x_3)\cdot f_{\mathrm{D}}(x_3,x_4)\cdot f_{\mathrm{E}}(x_3,x_5).\end{aligned}$$

Assume that we are interested in calculating the marginal functions[12]

$$\tilde{f}_1(x_1) \triangleq \sum_{x_2,x_3,x_4,x_5} f(x_1,x_2,x_3,x_4,x_5),$$
$$\tilde{f}_2(x_2) \triangleq \sum_{x_1,x_3,x_4,x_5} f(x_1,x_2,x_3,x_4,x_5),$$
$$\tilde{f}_3(x_3) \triangleq \sum_{x_1,x_2,x_4,x_5} f(x_1,x_2,x_3,x_4,x_5),$$
$$\tilde{f}_4(x_4) \triangleq \sum_{x_1,x_2,x_3,x_5} f(x_1,x_2,x_3,x_4,x_5),$$
$$\tilde{f}_5(x_5) \triangleq \sum_{x_1,x_2,x_3,x_4} f(x_1,x_2,x_3,x_4,x_5).$$

The calculations necessary to get $\tilde{f}_1(x_1), \ldots, \tilde{f}_5(x_5)$ will have two features: the structure of the calculation is reflected by the structure of the factor graph and partial results for one marginal function can be used for the calculation of the other marginal functions.

First, we focus on the calculations necessary to get $\tilde{f}_1(x_1)$ and try to give them a structural meaning relative to the corresponding factor graph. The main idea is to "push" the summation signs in (3.3) on page 37 as far right as possible, see (3.4) on page 37.

The objects $\mu_{X_i \to f_j}(x_i)$ or $\mu_{f_j \to X_i}(x_i)$, which can be associated with the edge between the vertices X_i and f_j, are called *messages* and are functions of x_i. The *domain* of such a function is the alphabet of X_i. From (3.4) we see that for each edge in Fig. 3.8 we get a message.

The factor graph leads us to the optimal number of computations needed to obtain the marginal function $\tilde{f}_1(x_1)$, see Fig. 3.10(top left).[13] Similar manipulations can be performed for calculating $\tilde{f}_2(x_2)$, $\tilde{f}_3(x_3)$, $\tilde{f}_4(x_4)$, $\tilde{f}_5(x_5)$ and messages can be introduced as well. See e.g. Fig. 3.10 (top right) for the messages necessary for calculating $\tilde{f}_3(x_3)$. Interestingly, the message associated to a direction along an edge is for all five

[12] The notion of a marginal function obviously varies depending on what the underlying semi-ring of the summary-product algorithm under consideration is.

[13] This statement about optimality refers to the fact that the sum-product algorithm takes optimally advantage of the structure of the factorization of the global function. Simplifications stemming from additional knowledge about the structure of local function nodes can potentially also be used to optimize the algorithm; this is especially apparent when a local function node itself can be expressed by a factor graph that is a tree.

$$\tilde{f}_1(x_1) \triangleq \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} f(x_1,x_2,x_3,x_4,x_5)$$

$$= \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} f_{\mathrm{A}}(x_1)\cdot f_{\mathrm{B}}(x_2)\cdot f_{\mathrm{C}}(x_1,x_2,x_3)\cdot f_{\mathrm{D}}(x_3,x_4)\cdot f_{\mathrm{E}}(x_3,x_5) \quad (3.3)$$

$$= \underbrace{f_{\mathrm{A}}(x_1)}_{\mu_{f_{\mathrm{A}}\to X_1}(x_1)} \cdot \underbrace{\sum_{x_2}\sum_{x_3} f_{\mathrm{C}}(x_1,x_2,x_3)\cdot \underbrace{\underbrace{f_{\mathrm{B}}(x_2)}_{\mu_{f_{\mathrm{B}}\to X_2}(x_2)}}_{\mu_{X_2\to f_{\mathrm{C}}}(x_2)} \cdot \underbrace{\underbrace{\sum_{x_4} f_{\mathrm{D}}(x_3,x_4)\cdot \underbrace{1}_{\mu_{X_4\to f_{\mathrm{D}}}(x_4)}}_{\mu_{f_{\mathrm{D}}\to X_3}(x_3)} \cdot \underbrace{\sum_{x_5} f_{\mathrm{E}}(x_3,x_5)\cdot \underbrace{1}_{\mu_{X_5\to f_{\mathrm{E}}}(x_5)}}_{\mu_{f_{\mathrm{E}}\to X_3}(x_3)}}_{\mu_{X_3\to f_{\mathrm{C}}}(x_3)}}_{\mu_{f_{\mathrm{C}}\to X_1}(x_1)} \quad (3.4)$$

computations the same. It is therefore sufficient to calculate each message exactly once, see Fig. 3.10 (bottom).

For a general factor tree we realize that the above observations still hold: one can define messages that are *sent along edges* and the messages are *processed* (taking products and doing summations) at the vertices. We see that messages can be *"reused"* in the sense that many partial calculations are the same, so it suffices to perform them only once. We get the following update rules for symbol and function nodes, respectively.[14]

Algorithm 3.13 (SPA update rules)

- **(SPA update rule for symbol nodes)** The message from the variable node X to the function node f_4 in Fig. 3.11 is defined to be

$$\mu_{X\to f_4}(x) \triangleq \mu_{f_1\to X}(x)\cdot\mu_{f_2\to X}(x)\cdot\mu_{f_3\to X}(x). \qquad (3.5)$$

- **(SPA update rule for function nodes)** The message from the function node f to the variable node X_4 in Fig. 3.12 is defined to be

$$\mu_{f\to x_4}(x_4) = \sum_{x_1}\sum_{x_2}\sum_{x_3} f(x_1,x_2,x_3,x_4) \cdot \mu_{X_1\to f}(x_1)\cdot\mu_{X_2\to f}(x_2)\cdot\mu_{X_3\to f}(x_3). \qquad (3.6)$$

- **(SPA rule to get marginal function)** To get the final marginal function $\tilde{f}(x)$ for the variable X in Fig. 3.13 one calculates

$$\tilde{f}(x) = \mu_{f_1\to X}(x)\cdot\mu_{f_2\to X}(x)\cdot\mu_{f_3\to X}(x)\cdot\mu_{f_4\to X}(x). \qquad (3.7)$$

□

[14]The rules are shown for specific examples; the general rules can easily be derived from them.

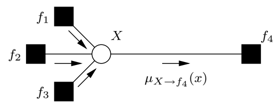

Figure 3.11: Partial factor graph showing the sum-product algorithm update rule for a symbol node. The incoming messages (which are not labeled here) are $\mu_{f_1\to X}$, $\mu_{f_2\to X}$, and $\mu_{f_3\to X}$; the outgoing message is $\mu_{X\to f_4}$.

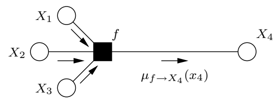

Figure 3.12: Partial factor graph showing the sum-product algorithm update rule for a function node. The incoming messages (which are not labeled here) are $\mu_{X_1\to f}$, $\mu_{X_2\to f}$, and $\mu_{X_3\to f}$; the outgoing message is $\mu_{f\to X_4}$.

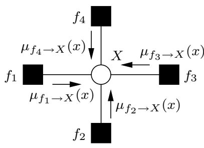

Figure 3.13: Partial factor graph showing the sum-product algorithm rule to get a marginal function. The incoming messages are $\mu_{f_1\to X}$, $\mu_{f_2\to X}$, $\mu_{f_3\to X}$, and $\mu_{f_4\to X}$.

Remark 3.14 (Aspects of the SPA)

- **(Rescaling)** If one is interested in the marginal functions only up to a factor, one can freely rescale the messages by a factor at each instant.[15] The equalities in (3.5), (3.6), and (3.7) turn then into

[15]Because the knowledge of the relative magnitude of the message and marginal function components, respectively, is sufficient in decoding applications, messages can be scaled by a constant. The constant must be *positive* so as not to change the order relationship between the different components.

proportionalities.

- **(Message update scheduling)** A message update scheduling says when one has to calculate what message. For factor trees there is obviously an optimal message update scheduling. For more on this topic, see [128, 58].

□

3.4.2 Loopy Factor Graphs

The summary-product algorithm as given in Alg. 3.13 tells us how to produce locally outgoing messages from incoming messages. Nothing prevents us therefore to apply these rules also to loopy factor graphs according to some message update schedule (after having initialized the messages to be some dummy messages at the beginning). But there will be neither an optimal update schedule as there was for loop-less factor graphs, nor will the calculated marginal functions be the exact marginals, they will be merely approximations thereof. (We will come back to this issues in Sec. 3.5.3.)

3.5 Factor Graphs and the Summary-Product Algorithm in the Context of Digital Data Transmission

This sections shows how a factor graph representing the joint pmf $P_{\mathbf{UXY}}$ of a digital data transmission problem can be derived and how the SPA helps to solve the decision taking problem.

We have seen in Ex. 3.9 that if we are interested in a symbol-wise decision taking, such decisions are done according to the formula

$$\hat{u}_i^{\text{symbol}}(\mathbf{y}) = \underbrace{\underbrace{\operatorname*{argmax}_{u_i\in\mathcal{U}}}_{\text{Decision taking}} \underbrace{\underbrace{\sum_{\substack{\mathbf{u}\in\mathcal{U}^k,\ \mathbf{x}\in\mathcal{X}^n\\ u_i\ \text{fixed}}}}_{\text{Sum}} \underbrace{P_{\mathbf{UXY}}(\mathbf{u},\mathbf{x},\mathbf{y})}_{\substack{\text{Factor graph}\\ \text{(written as product)}}}}_{\text{Sum-product algorithm}}}_{\text{Decision about symbol } u_i \text{ based on symbol-wise decoding}}, \tag{3.8}$$

for $1 \leq i \leq k$. The labeled parts of (3.8) already indicate our three-stage approach:

- **(Factor graph)** The factor graph will represent the joint pmf $P_{\mathbf{UXY}}$ (see Secs 3.5.1 and 3.5.2),
- **(Sum-product algorithm)** The sum-product algorithm will give us the marginals that we need in order to make a symbol-wise decision (in the case of loopy graphs the SPA will give us approximations to the true marginals) (see Sec. 3.5.3).
- **(Decision taking)** A decision can be taken based on the calculated marginals.

3.5.1 An Introductory Example of Factor Graph Modeling

Before starting, we need to define the XOR indicator function.

Definition 3.15 (XOR indicator function) For $v_i \in \mathbb{F}_2$ we define

$$f_{\text{XOR}}(v_1, \ldots, v_m) = [v_1 + \cdots + v_m = 0\,(\text{mod } 2)].$$

□

Example 3.16 (A factor graph that models digital data transmission) We would like to find the factor graph which represents the joint pmf $P_{\mathbf{UXY}}$ of $\mathbf{U}$, $\mathbf{X}$, and $\mathbf{Y}$, where the involved chance variables were defined in Secs. 3.2.1 and 3.2.2. As channel code we will use the $[3, 2, 2]$ binary linear block code from Ex. 3.5. The following steps refer to Fig. 3.14.

- **(Step 1)** The global function of the factor graph which represents the code shall be the joint probability function

$$P_{\mathbf{UXY}}(\mathbf{u},\mathbf{x},\mathbf{y}) = P_{\mathbf{U}}(\mathbf{u}) \cdot P_{\mathbf{X}|\mathbf{U}}(\mathbf{x}|\mathbf{u}) \cdot P_{\mathbf{Y}|\mathbf{UX}}(\mathbf{y}|\mathbf{u},\mathbf{x}).$$

- **(Step 2)** Since the random vectors $\mathbf{U}$, $\mathbf{X}$ and $\mathbf{Y}$ form a *Markov chain*, we have $P_{\mathbf{Y}|\mathbf{UX}}(\mathbf{y}|\mathbf{u},\mathbf{x}) = P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$ and thus

$$P_{\mathbf{UXY}}(\mathbf{u},\mathbf{x},\mathbf{y}) = P_{\mathbf{U}}(\mathbf{u}) \cdot P_{\mathbf{X}|\mathbf{U}}(\mathbf{x}|\mathbf{u}) \cdot P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}).$$

- **(Step 3)** We replace the symbol node representing the vectors $\mathbf{U}$ by the symbol nodes $U_1, \ldots, U_k$. Similarly we replace the symbol nodes corresponding to $\mathbf{X}$ and $\mathbf{Y}$ by $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_n$, respectively.
- **(Step 4)** Assuming that the source is memoryless we have the factorization

$$P_{\mathbf{U}}(\mathbf{u}) = P_{U_1}(u_1) \cdot P_{U_2}(u_2).$$

- **(Step 5)** The "deterministic" conditional probability function representing the encoding can be written as

$$\begin{aligned} &P_{\mathbf{X}|\mathbf{U}}(\mathbf{x}|\mathbf{u}) \\ &\qquad = P_{X_1|\mathbf{U}}(x_1|\mathbf{u}) \cdot P_{X_2|\mathbf{U}X_1}(x_2|\mathbf{u}, x_1) \cdot P_{X_3|\mathbf{U}X_1X_2}(x_3|\mathbf{u}, x_1, x_2) \\ &\qquad = P_{X_1|\mathbf{U}}(x_1|\mathbf{u}) \cdot P_{X_2|\mathbf{U}}(x_2|\mathbf{u}) \cdot P_{X_3|\mathbf{U}}(x_3|\mathbf{u}) \end{aligned}$$

Taking the generator matrix of the binary linear $[3, 2, 2]$ code from Ex. 3.5, we have $x_1 = u_1$, $x_2 = u_1 + u_2$ and $x_3 = u_2$. Then[16],

$$\begin{aligned} P_{X_1|\mathbf{U}}(x_1|\mathbf{u}) &= [x_1 = u_1] &&= [u_1 + x_1 = 0] &&= f_{\mathrm{XOR}}(u_1, x_1), \\ P_{X_2|\mathbf{U}}(x_2|\mathbf{u}) &= [x_2 = u_1 + u_2] &&= [u_1 + u_2 + x_2 = 0] &&= f_{\mathrm{XOR}}(u_1, u_2, x_2), \\ P_{X_3|\mathbf{U}}(x_3|\mathbf{u}) &= [x_3 = u_2] &&= [u_2 + x_3 = 0] &&= f_{\mathrm{XOR}}(u_2, x_3). \end{aligned}$$

- **(Step 6)** Assuming that the channel is memoryless we have

$$P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = P_{Y_1|X_1}(y_1|x_1) \cdot P_{Y_2|X_2}(y_2|x_2) \cdot P_{Y_3|X_3}(y_3|x_3).$$

□

Example 3.17 (Decoding with the help of the sum-product algorithm) We continue Ex. 3.16, in which we derived a factor graph representing the joint pmf $P_{\mathbf{UXY}}(\mathbf{u}, \mathbf{x}, \mathbf{y})$ of all occurring chance variables in that setup (see Fig. 3.14). Note that in that example the factor graph turned out to be a tree.

Given a measurement of the channel output $\mathbf{Y} = \mathbf{y}$ we desire now to get a decision about U_1 and U_2. To be more specific, we would like to have symbol-wise decoding (as presented in Ex. 3.9) of U_1 and U_2, which in our case reads

$$\hat{u}_i = \begin{cases} 0 & \text{if } P_{U_i\mathbf{Y}}(0, \mathbf{y}) > P_{U_i\mathbf{Y}}(1, \mathbf{y}) \\ 1 & \text{if } P_{U_i\mathbf{Y}}(0, \mathbf{y}) < P_{U_i\mathbf{Y}}(1, \mathbf{y}) \end{cases}, \tag{3.9}$$

[16] All statements inside Iverson's brackets are to be read modulo 2.

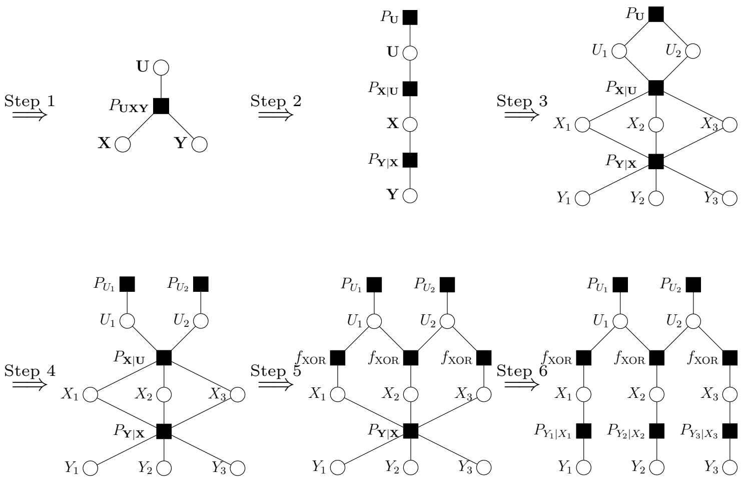

Figure 3.14: Stepwise refining a factor graph by stepwise including known facts about the factorization of the global function.

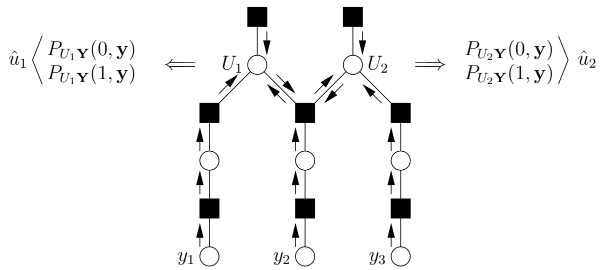

Figure 3.15: Messages necessary for calculating the marginal functions $P_{U_1\mathbf{Y}}(u_1, \mathbf{y})$ and $P_{U_2\mathbf{Y}}(u_2, \mathbf{y})$ for a given measurement $\mathbf{Y} = \mathbf{y}$. These marginal functions can be used to get symbol-wise decisions $\hat{u}_1$ and $\hat{u}_2$ about U_1 and U_2, respectively, based on the measurement $\mathbf{Y} = \mathbf{y}$.

where $i = 1, 2$. (If $P_{U_i \mathbf{Y}}(0, \mathbf{y}) = P_{U_i \mathbf{Y}}(1, \mathbf{y})$ we decide randomly among 0 and 1.) To be able to do these decisions, we therefore need to know the marginal functions

$$\tilde{f}_1(u_1) \triangleq P_{U_1 \mathbf{Y}}(u_1, \mathbf{y}) = \sum_{u_2, x_1, x_2, x_3} P_{\mathbf{UXY}}(\mathbf{u}, \mathbf{x}, \mathbf{y}),$$
$$\tilde{f}_2(u_2) \triangleq P_{U_2 \mathbf{Y}}(u_2, \mathbf{y}) = \sum_{u_1, x_1, x_2, x_3} P_{\mathbf{UXY}}(\mathbf{u}, \mathbf{x}, \mathbf{y}).$$

Knowing these marginal functions up to a positive constant is entirely sufficient for the decisions in (3.9) to be taken (see also Rem. 3.14). The SPA is now a tool to calculate the necessary marginal functions in an efficient way: the necessary messages are drawn in Fig. 3.15. □

Remark 3.18 (Ramifications)

- **(Puncturing)** One way to increase the rate of a code is to puncture a parity symbol. To be specific, assume that we puncture the symbol X_i. In the factor graph representation this means the following: one leaves the variable node X_i as it is, but the function node $P_{Y_i|X_i}(y_i|x_i)$, the variable node Y_i, and the dangling edges are removed.

- **(State vector)** Some codes are easier to describe by introducing a state (chance) vector $\mathbf{S}$ having two properties: it is completely determined by the input vector $\mathbf{U}$, i.e. $P_{\mathbf{S}|\mathbf{U}}(\mathbf{s}|\mathbf{u}) = [\mathbf{s}$ is a state vector corresponding to $\mathbf{u}]$, and its components are not directly transmitted over the channel. In our factor graphs we often draw state symbol nodes by double circles. The new global function is $P_{\mathbf{USXY}}(\mathbf{u}, \mathbf{s}, \mathbf{x}, \mathbf{y})$.

- **(Complexity)** Apparently, the factor graph seems to have become "much more complex" while going from Fig. 3.14 (upper-left) to Fig. 3.14 (bottom-right): but note that the function nodes got much simpler with regard to the dimensionality of their respective arguments. The complexity of performing the SPA on a factor graph will strongly depend on the complexity of the local function nodes, i.e., the simpler the local functions are, the more efficient the SPA can be performed. Of course, on loopy factor graphs there is a fundamental trade-off between the exactness of the desired results and the complexity of performing the SPA. The square-root bound on the state complexity of a tail-biting trellis compared to the state complexity of an ordinary trellis is a nice example of the possible trade-off [129].

□

3.5.2 Additional Examples of Factor Graph Modeling

In the last introductory subsection we showed how a factor graph representing the joint pmf of all occurring chance variables can be found for the data transmission problem. It would be interesting to model different types of sources and different types of channels (see e.g. [5]) within the framework of factor graphs, but this thesis is about channel codes. So, the additional examples we are about to show, will deal only with the factor graph representation of the channel codes. The whole factor graph can then be found as in the previous subsection.

Representing a channel code by a factor graph amounts to finding a partial factor graph for

$$P_{\mathbf{X}|\mathbf{U}}(\mathbf{x}|\mathbf{u}) = \big[\ \mathbf{x} \text{ is the codeword corresponding to } \mathbf{u}\ \big].$$

Such a representation is often non-unique and finding representations that are suitable for iterative decoding is still to be considered an art. The following examples show some possibilities.

Example 3.19 (A first LDPC code) Let a binary linear $[5, 3]$ code[17] have generator matrix $\mathbf{G}$ and parity-check matrix $\mathbf{H}$, where

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

We remind the reader that $\mathbf{x}$ is a codeword if and only if $\mathbf{x} \cdot \mathbf{H}^T = \mathbf{0}$. Let X_1, X_2, and X_5 be the information bits, i.e. we identify $X_1 = U_1$,

[17] Of course, this is a toy example of an LDPC code. But the procedure shown here is exactly the standard procedure for LDPC codes of any size.

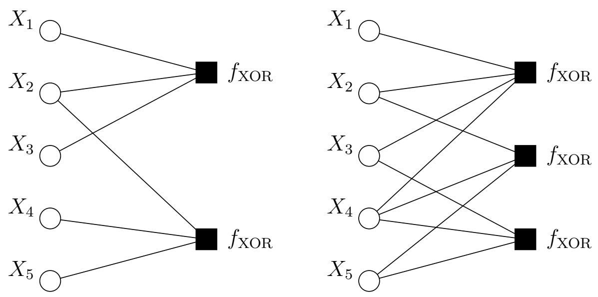

Figure 3.16: Partial factor graph corresponding to an LDPC code in Ex. 3.19 (left) and in Ex. 3.20 (right).

$X_2 = U_2$, and $X_5 = U_3$; then

$$\begin{aligned} P_{\mathbf{X}|\mathbf{U}}(\mathbf{x}|\mathbf{u}) &= \big[\ \mathbf{x} \text{ is the codeword corresponding to } \mathbf{u}\ \big] \\ &= [x_1 + x_2 + x_3 = 0] \cdot [x_2 + x_4 + x_5 = 0] \\ &= f_{\mathrm{XOR}}(x_1, x_2, x_3) \cdot f_{\mathrm{XOR}}(x_2, x_4, x_5). \end{aligned}$$

We get the partial factor graph as in Fig. 3.16 (left). □

Example 3.20 (A second LDPC code) Let a binary linear $[5, 2]$ code have generator matrix $\mathbf{G}$ and parity-check matrix $\mathbf{H}$, where

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

We let X_1 and X_2 be the information bits, i.e. we identify $X_1 = U_1$ and $X_2 = U_2$; then

$$\begin{aligned} P_{\mathbf{X}|\mathbf{U}}(\mathbf{x}|\mathbf{u}) &= \big[\ \mathbf{x} \text{ is the codeword corresponding to } \mathbf{u}\ \big] \\ &= [x_1 + x_2 + x_3 + x_4 = 0] \cdot [x_2 + x_4 + x_5 = 0] \\ &\quad \cdot [x_3 + x_4 + x_5 = 0] \\ &= f_{\mathrm{XOR}}(x_1, x_2, x_3, x_4) \cdot f_{\mathrm{XOR}}(x_2, x_4, x_5) \\ &\quad \cdot f_{\mathrm{XOR}}(x_3, x_4, x_5). \end{aligned}$$

We get the partial factor graph shown in Fig. 3.16 (right). □

Remark 3.21 (Cycles in a factor graph) The factor graph shown in Fig. 3.16, which was derived for the code in Ex. 3.19, is a tree (i.e., it has no loops) whereas the factor graph shown in Fig. 3.16 (right), which was derived for Ex. 3.20, has cycles: e.g. there is a cycle of length 4 formed by the vertices

$$X_2 \text{ — top } f_{\mathrm{XOR}} \text{ — } X_4 \text{ — middle } f_{\mathrm{XOR}} \text{ — } X_2,$$

or a cycle of length 6 formed by the vertices

$$X_2 \text{ — top } f_{\mathrm{XOR}} \text{ — } X_3 \text{ — bottom } f_{\mathrm{XOR}} \text{ — } X_5 \text{ — middle } f_{\mathrm{XOR}} \text{ — } X_2.$$

The second case, namely of having loops in the factor graph representing a code, is the typical case for (good) LDPC codes. The reason is that for LDPC codes which have a cycle-free factor graph representation (as above with simple f_{XOR} function nodes and without hidden states) one can show that their minimum distance is asymptotically bad (see App. C). Therefore, our factor graphs representing LDPC codes will have cycles, but one of our design criteria will be to avoid small cycles as far as possible (see Ch. 5). □

Example 3.22 (LDPC code, in general) The above procedure can be carried out for any (systematically encoded) binary linear block code defined by a parity-check matrix (which does not necessarily have to have full rank). We see that

- For every symbol of the code we get a symbol node.
- For every line of the parity-check matrix we get an f_{XOR}-function node.
- The degree of a symbol node is the number of ones in the column of $\mathbf{H}$ corresponding to the symbol.
- The degree of an f_{XOR}-function node (check node) is the number of ones in the row of $\mathbf{H}$ corresponding to that function node.
- Therefore, in a partial factor graph of a $(w_{\mathrm{col}}, w_{\mathrm{row}})$-regular LDPC code, all symbol nodes have degree w_{col} and all check nodes have degree w_{row}.
- In a partial factor graph of a $(\lambda(x), \rho(x))$-irregular LDPC code, the fraction of symbol nodes of degree i is $\tilde{\lambda}_i$, whereas the fraction of check nodes of degree i is $\tilde{\rho}_i$. (For notation, see Def. 2.4.)

- Extensions to non-binary codes are easily possible. But note that the complexity of the update rules of the summary-product algorithm will increase with larger alphabets.
- It is important to note that the degree of a symbol node can *change* when one is extending the partial factor graph representing a code to the complete factor graph. When talking about degrees of symbol nodes (e.g. when talking about degree sequences) it is *standard* to take the degrees of symbol nodes in the partial factor graph representing only the channel code.

□

Example 3.23 (QCRA LDPC code) The class of quasi-cyclic repeat-accumulate (QCRA) LDPC codes is a family of codes that was discussed by Tanner in [112]. Fig. 3.17 shows a $[44, 22, 8]$ linear code from this class designed by Tanner and the present author, see also Sec. 5.23. □

Example 3.24 (Convolutional code, trellis code) To be specific, we describe a rate–1/2 convolutional code which per time step takes one input bit and produces two channel bits. The encoder is described by

$$\begin{aligned} S_k &= f(S_{k-1}, U_k), \\ X_{k,1} &= g_1(S_{k-1}, U_k) \\ X_{k,2} &= g_2(S_{k-1}, U_k) \end{aligned}$$

The function nodes in the partial factor graph in Fig. 3.18 correspond to the function

$$\begin{aligned} & p_{S_k X_{k,1} X_{k,2} | S_{k-1} U_k}(s_k, x_{k,1}, x_{k,2} | s_{k-1}, u_k) \\ & = p_{S_k | S_{k-1} U_k}(s_k | s_{k-1}, u_k) \cdot p_{X_{k,1} | S_{k-1} U_k}(x_{k,1} | s_{k-1}, u_k) \\ & \qquad \cdot p_{X_{k,2} | S_{k-1} U_k}(x_{k,2} | s_{k-1}, u_k) \\ & = \big[s_k = f(s_{k-1}, u_k) \big] \cdot \big[x_{k,1} = g_1(s_{k-1}, u_k) \big] \cdot \big[x_{k,2} = g_2(s_{k-1}, u_k) \big]. \end{aligned}$$

We prefer to use only one function node and not to split it up into three "smaller" function nodes. By this we avoid the introduction of four-cycles which are generally bad for iterative decoding. From this point of view, Fig. 3.18 is much better as it is actually a tree. □

Example 3.25 (Turbo code) Turbo codes consist of two systematic recursive convolutional codes and an interleaver in-between. Based on

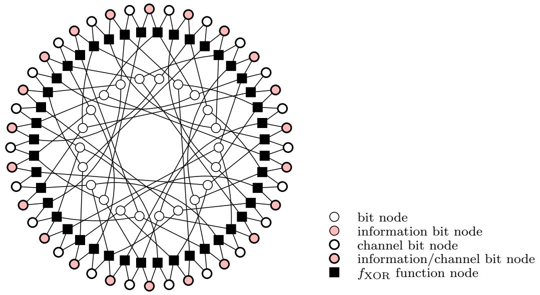

Figure 3.17: Part of factor graph corresponding to a QCRA LDPC code.

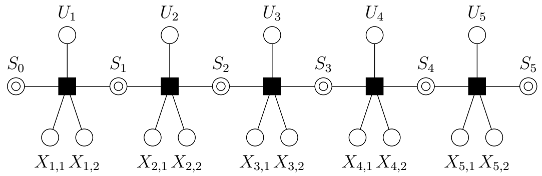

Figure 3.18: Part of factor graph corresponding to a convolutional code.

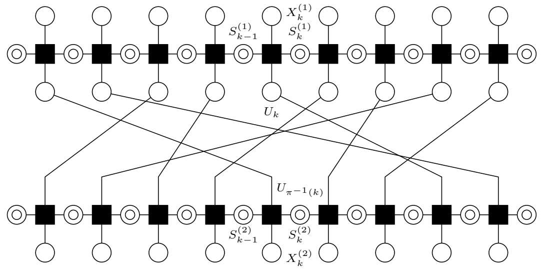

Figure 3.19: Part of factor graph corresponding to a turbo code. (The details of this factor graph are discussed in Sec. 5.7.2.)

the input chance vector $\mathbf{U}$ (which is also transmitted), the state sequences $\mathbf{S}^{(1)}$ and $\mathbf{S}^{(2)}$ and channel bit sequences $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ are produced according to

$$\begin{aligned} S_k^{(1)} &= f_1(S_{k-1}^{(1)}, U_k), & X_k^{(1)} &= g_1(S_{k-1}^{(1)}, U_k), \\ S_k^{(2)} &= f_2(S_{k-1}^{(2)}, U_{\pi^{-1}(k)}), & X_k^{(2)} &= g_2(S_{k-1}^{(2)}, U_{\pi^{-1}(k)}), \end{aligned}$$

where π is a permutation of the input symbols. Additionally, in order to drive the states of both constituent encoders into the zero state, one can use termination methods as e.g. proposed in [26]. If one wants to increase the rate of the code one can use puncturing; usually this is done according to some regular pattern.

From these considerations, the factor graph of a turbo code as shown in Fig. 3.19 can easily be derived along the same lines as for the convolutional code above. □

Remark 3.26 (Different representations) The different code representations shown before can be classified into image representations and kernel representations.

- **(Image representation)** In an image representation the code is described as an image of a transformation rule. Such representations were used in Exs. 3.16, 3.24, and 3.25.

- **(Kernel representation)** In a kernel representation a configuration is a valid configuration (i.e. a codeword) if a certain set of relations is fulfilled. Such representations were used in Exs. 3.19, 3.20, 3.22, and 3.23.

Sometimes it makes sense to represent a code by a factor graph that is partly an image representation, partly a kernel representation. □

Remark 3.27 (Tanner graphs) Already in 1981, Tanner introduced a graphical representation for codes [109]. A Tanner graph can be seen as a partial factor graph representing the function $P_{\mathbf{X}|\mathbf{U}}(\mathbf{x}|\mathbf{u})$, which can only take on the values 0 and 1. □

Remark 3.28 (Behaviors) Rem. 2.5 already put coding theory in the context of behaviors. Factor graphs (i.e. partial factor graphs representing a channel code), and with them Tanner graphs, can also be put in this context. Compared with Tanner graphs, factor graphs can additionally model *latent variables* which we called here the state vector $\mathbf{s}$; $\mathbf{u}$ and $\mathbf{x}$ correspond then to the *manifest variables.* The *full behavior* is the set of all allowed $(\mathbf{u}, \mathbf{x}, \mathbf{s})$ triples whereas the *manifest behavior* is the set of all $(\mathbf{u}, \mathbf{x})$ such that there exists an $\mathbf{s}$ such that the triple $(\mathbf{u}, \mathbf{x}, \mathbf{s})$ is in the full behavior. (Possibly, $\mathbf{u}$ can also be considered as a latent variable.) □

3.5.3 The Summary-Product Algorithm on Loopy Factor Graphs

The example given in Sec. 3.5.1, where a loop-less factor graph resulted, might be misleading in the sense that the factor graphs we will usually be interested in are loopy. Astonishingly, applying the summary-product algorithm on loopy factor graphs works excellently in the context of coding applications. It is clearly a sub-optimal algorithm for decoding as it no longer gives the exact marginals which are necessary for a symbol-wise maximum a-posteriori decision taking. It seems that it is advantageous to perform a sub-optimal algorithm on a loopy factor graph (with a large state space for most of the cut-sets) than to perform an optimal algorithm on a factor tree (with small state space for most cut-sets), see also the observations about complexity stated in Rem. 3.18.

Once a factor graph representing the digital data transmission scheme has been found, decoding of a received channel output vector proceeds according to the scheme shown in Fig. 3.20:

- All messages are initialized.

- Messages are updated according to an update schedule. The schedule may vary from step to step.

- After each step one calculates the marginal functions of the symbols one would like to decode.

- Taking decisions based on the current marginal functions (of all channel bits), one obtains a word.

- One terminates if the obtained word is a codeword or if a prespecified maximum number of steps is attained.

$$\begin{array}{rcl}
\begin{array}{c}\text{SPA messages}\\ \text{after step 0}\end{array} & \xrightarrow{(3.7)} & \begin{array}{c}\text{Marginal functions}\\ \text{after step 0}\end{array} \\
{\scriptstyle \text{one step of SPA}}\atop{\scriptstyle \text{(3.5) and (3.6)}} \Big\downarrow & & \\
\begin{array}{c}\text{SPA messages}\\ \text{after step 1}\end{array} & \xrightarrow{(3.7)} & \begin{array}{c}\text{Marginal functions}\\ \text{after step 1}\end{array} \\
{\scriptstyle \text{one step of SPA}}\atop{\scriptstyle \text{(3.5) and (3.6)}} \Big\downarrow & & \\
\vdots & & \\
{\scriptstyle \text{one step of SPA}}\atop{\scriptstyle \text{(3.5) and (3.6)}} \Big\downarrow & & \\
\begin{array}{c}\text{SPA messages}\\ \text{after step } m\end{array} & \xrightarrow{(3.7)} & \begin{array}{c}\text{Marginal functions}\\ \text{after step } m\end{array}
\end{array}$$

Figure 3.20: Main idea of decoding based on the SPA.

The following update schedules for the partial factor graph representing the channel code are standard.

- **(LDPC codes)** At each step one first updates all variable nodes and then all check nodes.
- **(Turbo codes)** Each step consists of four parts (we refer to the factor graph in Fig. 3.19): first the messages in the top constituent code are updated, then the information symbols in-between, then the messages in the bottom constituent code and finally again the information symbols in-between. Note that both the top and the bottom constituent code are represented by a factor tree: therefore there is an optimal update schedule for updating the messages in these parts of the factor graph (these partial steps essentially correspond to performing the forward-backward algorithm or the BCJR algorithm [11]).

One can think of many other reasonable update schedules. E.g. in [79] the authors present a modified update schedule for LDPC codes which tries to account for the loops that are present in the graph: those vertices whose local girth is smaller are updated less frequently than the vertices with a large local girth.

The degrees of freedom in choosing a sensible update schedule become even larger when the factor graph accounts for the transmission over a channel with memory; see e.g. [44] for update schedules when LDPC codes are used as channel codes in the context of magnetic storage applications.

We finish this section by commenting on various issues.

Remark 3.29 (Sum-product algorithm on loopy graphs) Besides some particular results (as e.g. in the case of factor graphs with one loop [1] or e.g. factor graphs representing jointly Gaussian densities [126, 100]), the sum-product algorithm on loopy factor graphs is still poorly understood. Recently, it was shown that the fixed points of the sum-product algorithm are in a one-to-one relationship with zero-gradient points of a Bethe free energy associated with the underlying problem [131]. This observation has lead to the introduction of generalized belief propagation schemes, see [131, 132] but also [85]. The directions the research will go is quite unclear but exciting. A promising direction seems to be to study pseudo-codewords as was done in [56]. □

Remark 3.30 (Various instances of the summary-product algorithm)

- **(Max-product algorithm)** If we are interested in a block-wise decision taking as defined in Ex. 3.8, the resulting formula reads

$$\hat{u}_i^{\text{block}}(\mathbf{y}) = \underbrace{\underbrace{\operatorname*{argmax}_{u_i\in\mathcal{U}}}_{\text{Decision taking}} \underbrace{\underbrace{\max_{\substack{\mathbf{u}\in\mathcal{U}^k,\mathbf{x}\in\mathcal{X}^n\\ u_i\ \text{fixed}}}}_{\text{Maximum}} \underbrace{P_{\mathbf{UXY}}(\mathbf{u},\mathbf{x},\mathbf{y})}_{\substack{\text{Factor graph}\\ \text{(written as product)}}}}_{\text{Max-product algorithm}}}_{\text{Decision about symbol } u_i \text{ based on block-wise decoding}},$$

  and has exactly the same structure as (3.8), except that the summations are replaced by maximum-operators. In other words, the underlying ring has changed from $\langle \mathcal{R}, +, \cdot \rangle$ to $\langle \mathcal{R}, \max, \cdot \rangle$. We can still take the factor graphs as defined above (as the product part remained the same), but this time the summary-product algorithm is the max-product algorithm. (More information about underlying rings and other related issues can e.g. be found in [128, 58].)

- **(Belief propagation, forward-backward algorithm, BCJR algorithm, Viterbi algorithm)** The sum-product algorithm is essentially the same as belief propagation [91] from the artificial intelligence literature. Note also that performing the sum-product algorithm (with the appropriate message update scheduling) on the factor graph in Fig. 3.18 derived in Ex. 3.24 essentially corresponds to the forward-backward algorithm or BCJR algorithm [11]. On the other hand, performing the max-product algorithm (with the appropriate message update scheduling) on the factor graph in Fig. 3.18 is essentially equivalent to the Viterbi algorithm.

- **(Infinite factor trees)** The summary-product algorithm is nowadays quite well-understood for factor trees, also when they are infinite. Tools as density evolution and calculation of thresholds are standard today (see e.g. [96, 95] but also [38]). Much of the current research efforts go into generalizing these tools and concepts to the case of loopy factor graphs (see e.g [23]).

□

Remark 3.31 (Various aspects of the SPA)

- **(Complexity Issues, simplified SPAs)** The complexity of the SPA is obviously given by the complexity of the different update rules in (3.5), (3.6), and (3.7) whose complexity is essentially given by the alphabet sizes of the involved variable nodes. Of course, the least complexity one gets for binary variable nodes. Various authors have proposed simplified SPA algorithms, i.e. algorithms where the summary-product update rules are approximated by rules that are even simpler to calculate.

- **(Different Representations)** The summary-product algorithm can be converted into other algorithms. E.g. in the case of binary codes the messages are just vectors with two entries. As it is sufficient to know only the ratio of these two entries (likelihood-ratio), one can translate the SPA update rules into likelihood-ratio update rules (or consequently also log-likelihood-ratio update rules). One can also use variations known as the one-sweep algorithm or decoding on the dual code (see App. D).

- **(Continuous alphabets of measured variables)** The SPA was presented here for discrete alphabets. But essentially nothing changes (also in view of complexity) if the measured variables are from a continuous alphabet, e.g. if we had $\mathcal{Y} = \mathbb{R}$ in our examples.

- **(Continuous alphabets)** The SPA can be extended to include variable nodes where the alphabet is $\mathbb{R}$ (or a subset thereof): pmfs get replaced by densities and sums get replaced by integrals. Apart from these changes, everything that was said up to now is also valid for this broader scenario. The problem is though in the update of the messages, i.e., in general the updates are not computable in finite time anymore. To alleviate this there are different possibilities. If the joint pdf is jointly Gaussian then all messages are Gaussian: messages can be represented by the means and the covariances [63, 58]. Other approaches include to allow just a finite set of messages or a family of messages that can easily be parameterized: calculated messages are then mapped to the nearest available message.

- **(Order-one reprocessing)** We shortly discuss a list-decoding method introduced by Fossorier [33]. It uses the log-likelihood estimates of each bit as given by the SPA after each step. The outline of the technique is given in Alg. 3.32 and Fig. 3.21.

  Algorithm 3.32 (Order-one reprocessing) Assume that we decode a binary $[n, k]$ linear code. After each step of the SPA we perform the following steps.

  – We calculate estimates of the log-likelihood ratios of each code-bit (as given by the SPA).
  – Based on these we search for the most reliable independent positions (MRIPs) of code-bits, i.e. the k independent bits with the highest absolute log-likelihood ratio. Of these bits we take hard-decisions.
  – We get $k + 1$ codeword estimates with the following procedure. One of them we get by taking the hard-decision bits in the MRIPs and completing the remaining $n - k$ bits so that the whole word is a codeword. Additional k codeword estimates are obtained by flipping each of the k bits in the MRIPs independently and completing to a codeword[18].

  After having done this for several steps of the SPA we get a list of codeword estimates. We take the one with the largest codeword

[18]Flipping every possible pair of bits in the most reliable basis would be called order-two reprocessing, etc.

$$
\begin{array}{ccccc}
\begin{array}{c}\text{SPA messages}\\ \text{after step } 0\end{array} & \longrightarrow & \begin{array}{c}\text{Marg. funcs.}\\ \text{after step } 0\end{array} & \xrightarrow[\text{reprocessing}]{\text{order-one}} & \begin{array}{c}\text{codeword}\\ \text{estimates } 0\end{array} \\
\downarrow{\scriptstyle \text{one step of SPA}} & & & & \\
\begin{array}{c}\text{SPA messages}\\ \text{after step } 1\end{array} & \longrightarrow & \begin{array}{c}\text{Marg. funcs.}\\ \text{after step } 1\end{array} & \xrightarrow[\text{reprocessing}]{\text{order-one}} & \begin{array}{c}\text{codeword}\\ \text{estimates } 1\end{array} \\
\downarrow{\scriptstyle \text{one step of SPA}} & & & & \\
\vdots & & & & \\
\downarrow{\scriptstyle \text{one step of SPA}} & & & & \\
\begin{array}{c}\text{SPA messages}\\ \text{after step } m\end{array} & \longrightarrow & \begin{array}{c}\text{Marg. funcs.}\\ \text{after step } m\end{array} & \xrightarrow[\text{reprocessing}]{\text{order-one}} & \begin{array}{c}\text{codeword}\\ \text{estimates } m\end{array}
\end{array}
$$

Figure 3.21: Main idea of list decoding based on the SPA and order-one reprocessing.

likelihood ratio. Different termination criteria are discussed in [33].

□

Usually one needs less iterations compared to the “pure” SPA. One simple explanation why this algorithm works well is that the MRIPs change after each step so that a reasonably large set of codewords (in which hopefully the correct one lies) is examined.

The complexity of the algorithm is essentially given by the complexity of completing the codewords. For this, one needs to put the generator matrix into row-reduced echelon form according to the MRIPs; the complexity of this operation is roughly proportional to the cube of the code length.[19] The complexity of flipping single bits in the MRIPs and completing the codeword is proportional to the square of the code length. (The complexity of flipping pairs in the MRIPs and completing the codeword is proportional to the cube of the code length.)

□

[19] Instead of doing the re-encoding by performing a Gaussian elimination one could try using techniques as presented in [97].

Chapter 4

Graphs with Large Girth

This chapter focuses on the construction of graphs with large girth (the girth is the length of the shortest cycle, see Def. 2.11). The most important constructions that we will encounter are the following:

- A Cayley graph (for a definition, see Sec. 2.2.2) construction given by Margulis [82]: this was the first explicit graph construction in the literature having a girth that growths proportionally to the logarithm of the number of vertices.

- A Cayley graph construction defined by Lubotzky, Phillips, and Sarnak [65] (and at the same time also by Margulis [84]) of so-called Ramanujan graphs: they have many extremal properties; among them, the girth is above the Erdős-Sachs bound. We also propose an extended construction using such graphs.

- Graphs from finite geometries, especially finite generalized polygons: finite generalized polygons are extremal in the sense that for a given diameter they have the largest possible girth, namely twice the diameter.

Note that proofs are given in Sec. 4.5.

4.1 An Introduction to the Construction of Cayley Graphs with Large Girth

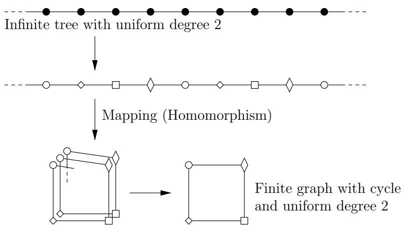

Figure 4.1: From an infinite two-regular tree to a finite two-regular graph with a cycle. Vertices of the same color (i.e. shape) get mapped onto the same vertex.

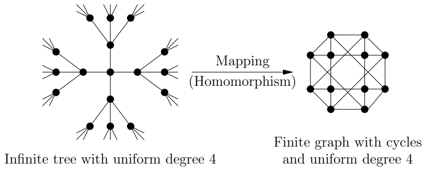

Figure 4.2: From an infinite four-regular tree to a finite four-regular graph with cycles.

This section gives a preview of the main ideas that lead to the construction of Cayley graphs with large girth in Secs. 4.2 and 4.3.

Assume that we would like to construct a two-regular graph with large girth. One way to start would be as in Fig. 4.1 (top) by a two-regular infinite tree, color the vertices in a "regular" pattern (the coloring is done here by using different vertex shapes), lying vertices with the same color on top of each other (see Fig. 4.1 (bottom left)) and upon identifying vertices and edges that lie on top of each other one gets the finite two-regular graph with a cycle in Fig. 4.1 (bottom right). This "coloring" and identifying is of course nothing else than defining a homomorphism. The same principle can be applied to get from an infinite four-regular tree to a finite four-regular graph with cycles, as shown in Fig. 4.2. The proofs about the length of cycles (and the girth) in the finite graphs will then be based on properties of the vertex-labeling of the tree.

4.2 First Construction of a Graph with Large Girth by Margulis

$$\begin{tikzcd}
\underset{\text{(matrix ring)}}{\mathcal{M}_2(\mathbb{Z})} \\
\underset{\text{(modular group)}}{\mathrm{SL}_2(\mathbb{Z})} \arrow[u] \\
\underset{\text{(free subgroup)}}{\mathcal{G}} \arrow[u] \arrow[r, "\text{Cayley graph}"] \arrow[d, "\phi"'] & \underset{\text{(infinite tree)}}{\mathsf{X}(\mathcal{G},\mathcal{S})} \arrow[d, "\phi"'] \\
\underset{\text{(factor group)}}{\mathcal{G}_q} \arrow[r, "\text{Cayley graph}"] & \underset{\text{(finite graph with cycles)}}{\mathsf{X}_q^{\mathrm{Mar}} = \mathsf{X}(\mathcal{G}_q,\mathcal{S}_q)}
\end{tikzcd}$$

Figure 4.3: Rings, groups, and Cayley graphs leading to the graph construction by Margulis. (The different objects and their relationships are defined in the text.)

Margulis [82] parameterized his Cayley graph family by the letter p; but instead of p, we will use the letter q.[1] The breakthrough by Margulis was that he was able to give a lower bound on the girth of his Cayley graphs. The construction of Margulis [82] can be condensed to the diagram shown in Fig. 4.3:

- We start with a ring, namely the ring $\mathcal{M}_2(\mathbb{Z})$ of 2×2-matrices over $\mathbb{Z}$ (see Sec. A.3). We also need a norm: one takes the matrix norm induced from the L_2 vector norm (see Sec. A.5).

[1] The motivation behind this is to use a notation that is similar to the notation in Sec. 4.3, where q is related to the graph size.

- The modular group[2] $\mathrm{SL}_2(\mathbb{Z})$ is the multiplicative group of invertible elements of $\mathcal{M}_2(\mathbb{Z})$.
- $\mathcal{G}$ is a free subgroup (with generator set $\mathcal{S}$) of $\mathrm{SL}_2(\mathbb{Z})$. As the set $\mathcal{S}$ is chosen such that it generates freely the whole group $\mathcal{G}$, the Cayley graph $\mathsf{X}(\mathcal{G}, \mathcal{S})$ is an infinite tree of regular degree 4.
- A homomorphism ϕ maps the infinite group $\mathcal{G}$ onto the finite group $\mathcal{G}_q \triangleq \mathrm{SL}_2(\mathbb{F}_q)$. Accordingly, ϕ maps the set $\mathcal{S}$ to the set $\mathcal{S}_q \triangleq \phi(\mathcal{S})$. As the group $\mathcal{G}_q$ is finite, the Cayley graph $\mathsf{X}_q^{\mathrm{Mar}} \triangleq \mathsf{X}(\mathcal{G}_q, \mathcal{S}_q)$ is a finite graph of regular degree 4 and must have loops.

The following subsections give the explicit definitions of the algebraic objects. The climax will then be the subsection about the proof of the lower bound on the girth.

4.2.1 The Free Subgroup $\mathcal{G}$ of the Modular Group $\mathrm{SL}_2(\mathbb{Z})$

Let us define two matrices from $\mathrm{SL}_2(\mathbb{Z})$, namely

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}, \qquad \mathbf{B} = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}.$$

The matrices $\mathbf{A}$ and $\mathbf{B}$ are special in the sense, that there are no non-trivial multiplicative relations between them, i.e., any two different reduced words over $\{\mathbf{A}, \mathbf{B}\}$ define different elements of $\mathrm{SL}_2(\mathbb{Z})$ (see e.g. Exercise 13(h) of Ch. 2.3 of [77]). Therefore any element of $\mathrm{SL}_2(\mathbb{Z})$ that is a word in $\{\mathbf{A}, \mathbf{B}\}$ has a *strictly reduced*[3] *unique factorization*, e.g.

$$\begin{pmatrix} -23 & 36 \\ -16 & 25 \end{pmatrix} = \mathbf{A} \cdot \mathbf{B}^{-1} \cdot \mathbf{A} \cdot \mathbf{A} \cdot \mathbf{B} \cdot \mathbf{A}^{-1}. \tag{4.1}$$

Compare this with the *unique factorization* of integers in prime factors:

$$12 = 2 \cdot 2 \cdot 3 = 2 \cdot 3 \cdot 2 = 3 \cdot 2 \cdot 2. \tag{4.2}$$

[2]Margulis [82] does not emphasize this, but for the girth proof it will turn out to be important that the modular group can be embedded in a ring where a norm is defined.

[3]By a reduced factorization we mean that we do not allow $\mathbf{A}$ and $\mathbf{A}^{-1}$ or $\mathbf{B}$ and $\mathbf{B}^{-1}$ to be neighbors in the factorization as such pairs can be replaced by the identity matrix.

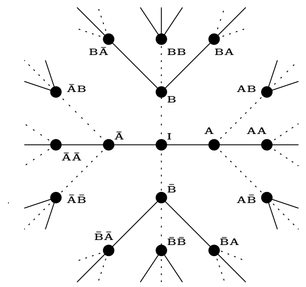

Figure 4.4: Undirected Cayley graph $\mathsf{X}(\mathcal{G}, \mathcal{S})$ with $\mathcal{S} = \{\mathbf{A}, \mathbf{B}, \mathbf{A}^{-1}, \mathbf{B}^{-1}\}$. The solid edges correspond to the generators $\mathbf{A}$ and $\mathbf{A}^{-1}$, whereas the dotted edges correspond to the generators $\mathbf{B}$ and $\mathbf{B}^{-1}$, respectively. (Here we have set $\bar{\mathbf{A}} \triangleq \mathbf{A}^{-1}$ and $\bar{\mathbf{B}} \triangleq \mathbf{B}^{-1}$.)

Whereas in (4.2) the factors are unique but not the order, the factors *and* the order are unique in in (4.1).

Let now the subgroup $\mathcal{G}$ of $\mathrm{SL}_2(\mathbb{Z})$ be generated by the set $\mathcal{S} = \{\mathbf{A}, \mathbf{B}, \mathbf{A}^{-1}, \mathbf{B}^{-1}\}$; it is not difficult to see that this subgroup relation is strict, e.g. the matrix $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ is not in $\mathcal{G}$. From our discussion it is clear that $\mathcal{G}$ is freely generated by $\mathbf{A}$ and $\mathbf{B}$ and that therefore each element of $\mathcal{G}$ has a unique reduced factorization. Moreover, the Cayley graph $\mathsf{X}(\mathcal{G}, \mathcal{S})$ in Fig. 4.4 is an infinite tree of degree four (see also the discussion in Ex. 2.25).

4.2.2 The Mapping ϕ and the Cayley Graph $\mathsf{X}_q^{\mathrm{Mar}}$

Let q be an odd prime and let $\mathcal{G}_q \triangleq \mathrm{SL}_2(\mathbb{Z}/q\mathbb{Z})$. We consider the surjective homomorphism

$$\begin{aligned} &\phi : \mathrm{SL}_2(\mathbb{Z}) \to \mathcal{G}_q \\ &\begin{pmatrix} a & b \\ c & d \end{pmatrix} \mapsto \begin{pmatrix} a \,(\mathrm{mod}\ q) & b \,(\mathrm{mod}\ q) \\ c \,(\mathrm{mod}\ q) & d \,(\mathrm{mod}\ q) \end{pmatrix}. \end{aligned}$$

Let us set $\mathbf{A}_q \triangleq \phi(\mathbf{A})$ and $\mathbf{B}_q \triangleq \phi(\mathbf{B})$. Obviously, $\mathbf{A}_q^{-1} = \phi(\mathbf{A}^{-1})$ and $\mathbf{B}_q^{-1} = \phi(\mathbf{B}^{-1})$. We also set $\mathcal{S}_q \triangleq \{\mathbf{A}_q, \mathbf{B}_q, \mathbf{A}_q^{-1}, \mathbf{B}_q^{-1}\}$. We let $\mathsf{X}_q^{\mathrm{Mar}}$ be the Cayley graph $\mathsf{X}(\mathcal{G}_q, \mathcal{S}_q)$; as $\mathbf{A}_q$ and $\mathbf{B}_q$ generate the whole group $\mathcal{G}_q$, the graph has one component, and because the group $\mathcal{G}_q$ is finite, the graph contains loops.

4.2.3 A Lower Bound on the Girth of $\mathsf{X}_q^{\mathrm{Mar}}$

Theorem 4.1 *The girth of the graph $\mathsf{X}(\mathcal{G}_q, \mathcal{S}_q)$ (as defined above) has the lower bound*

$$g\big(\mathsf{X}(\mathcal{G}_q, \mathcal{S}_q)\big) \geq 2\log_\alpha(q/2) - 1,$$

and asymptotically the lower bound

$$\begin{aligned} g\big(\mathsf{X}(\mathcal{G}_q, \mathcal{S}_q)\big) &\geq \frac{2}{3} \cdot \log_\alpha(n) - 2 \cdot \log_\alpha(2) \\ &= \big(0.831\ldots + o(1)\big) \log_{4-1}(n). \end{aligned}$$

(See also Fig. 4.7.)

Proof: See Sec. 4.5.1. We recommend to the reader to have a look at the proof. □

Remark 4.2 (Girth and non-trivial relations) To a cycle of length ℓ in a Cayley graph $\mathsf{X}(\mathcal{G}, \mathcal{S})$ we can associate a non-trivial relation of the form $s_1 \cdots s_\ell = e$, where $s_i \in \mathcal{S}$ and where e is the neutral element of $\mathcal{G}$. Therefore, a Cayley graph $\mathsf{X}(\mathcal{G}, \mathcal{S})$ has girth g if and only if there are no non-trivial relations among elements of $\mathcal{S}$ of length smaller than g but there is at least one non-trivial relation among elements of $\mathcal{S}$ of length g. □

4.2.4 Another Family of Cayley Graphs by Margulis

In [84], Margulis defined another family of graphs with large girth. If some condition about the chosen parameters is fulfilled, these graphs can be presented as Cayley graph. They turn out to be equivalent to the graphs in [65] which we will discussed in Sec. 4.3.

The way Margulis presents this construction, these graphs can be seen as a variation on his construction that we just have shown. The key features of the changes are

- the ring $\mathcal{M}_2(\mathbb{Z})$ is replaced by the ring of two-by-two matrices over the q–adic numbers [55],
- the L_2 matrix norm is replaced by some metric based on (non-archimedean) valuations,
- the triangle inequality is replaced by the ultrametric inequality.

Note that this construction by Margulis heavily uses concepts about trees introduced earlier by Serre [103, 104].

4.3 The Graph Construction by Lubotzky, Phillips, and Sarnak

$$\begin{tikzcd}
\substack{\mathbb{H}(\mathbb{Z}) \\ \text{(quaternion ring)}} & \\
\substack{\Lambda'(2) \\ \text{(quaternion semi-group)}} \arrow[u] & \\
\substack{\Lambda(2) \\ \text{(free subgroup)}} \arrow[u] \arrow[r, "\text{Cayley graph}"] \arrow[d, "\phi"'] & \substack{\mathsf{X}(\Lambda(2), \mathcal{S}) \\ \text{(infinite tree)}} \arrow[d, "\phi"'] \\
\substack{\phi(\Lambda(2)) \\ \text{(factor group)}} \arrow[r, "\text{Cayley graph}"] & \substack{\mathsf{X}_{q,p}^{\mathrm{LPS}} = \mathsf{X}(\phi(\Lambda(2)), \phi(\mathcal{S})) \\ \text{(finite graph with cycles)}}
\end{tikzcd}$$

Figure 4.5: Rings, groups, and Cayley graphs used in the construction of $\mathsf{X}_{q,p}^{\mathrm{LPS}}$. (The algebraic objects and their relationships are defined in the text.)

In this section we discuss the graphs $\mathsf{X}_{q,p}^{\mathrm{LPS}}$ which were introduced by Lubotzky, Phillips, and Sarnak [65].[4] The graphs $\mathsf{X}_{q,p}^{\mathrm{LPS}}$ are Cayley graphs and the idea for their construction is analogous to the one given in Sec. 4.2. We also propose a graph construction whose building blocks are $\mathsf{X}_{q,p}^{\mathrm{LPS}}$ graphs.

4.3.1 Construction of $\mathsf{X}_{q,p}^{\mathrm{LPS}}$

In [65], the graphs $\mathsf{X}_{q,p}^{\mathrm{LPS}}$ were defined for two odd distinct primes q and p where $q = 1 \pmod 4$ and $p = 1 \pmod 4$.[5] In [101] it is alluded that the construction is possible for any distinct odd primes q and p, but we are not aware of an explicit construction given in the literature. But using the extended definitions as we gave them in App. B a construction can be given for the unrestricted case.

As already mentioned, the graphs $\mathsf{X}_{q,p}^{\mathrm{LPS}}$ are Cayley graphs and the idea for their construction is analogous to the one given in Sec. 4.2. We would like to comment on some points of the construction (see Fig. 4.5, compare with Fig. 4.3).

- The ring of integral quaternions $\mathbb{H}(\mathbb{Z})$ is the underlying ring (see App. B). The norm that is used is the quaternion norm.
- $\Lambda'(2)$ is a multiplicative semi-group (with basis $\mathcal{F}_p$) of the multiplicative group of the ring $\mathbb{H}(\mathbb{Z})$ (see Defs. B.4 and B.20).
- From $\Lambda'(2)$ we derive $\Lambda(2)$, which is a free group. (See Defs. B.21 and B.4; Sec. B.2, and especially Lemma B.22, establish the freeness property.)
- From the freeness of $\Lambda(2)$ follows that the Cayley graph $\mathsf{X}(\Lambda(2), S)$ with $\mathcal{S} = \{[\alpha] \mid \alpha \in \mathcal{F}_p\}$ must be an infinite tree. By Lemma B.5 its regular degree is $|\mathcal{S}| = |\mathcal{F}_p| = p + 1$.

[4] Compared to the notation in [65], we made the following changes: first, the quantities q and p are in the lower index and not in the upper index; second, we have exchanged the order of q and p so that the first lower index reminds us of the graph size whereas the second lower index reminds us of the degree of the graph. This is akin to $\mathsf{X}(\mathcal{G}, \mathcal{S})$, where the first argument says something about the size whereas the second something about the degree.

[5] In order to guarantee that the resulting graphs have regular degree $p + 1$, one additionally needs the easily fulfillable technical condition $q > \sqrt{p}$.

- The homomorphism ϕ is the one given in Def. B.29. By Lemma B.30 it maps $\Lambda(2)$ onto $\mathrm{PGL}_2(\mathbb{F}_q)$ if $\left(\frac{p}{q}\right) = -1$, and onto $\mathrm{PSL}_2(\mathbb{F}_q)$ if $\left(\frac{p}{q}\right) = +1$. Note that the fields $\mathbb{Z}/q\mathbb{Z}$ and $\mathbb{F}_q$ are isomorphic because q is an odd prime.
- The new Cayley graph $\mathsf{X}_{q,p}^{\mathrm{LPS}} \triangleq \mathsf{X}(\phi(\Lambda(2)), \phi(\mathcal{S}))$ is a regular graph of degree $p+1$, and as both $\mathrm{PGL}_2(\mathbb{F}_q)$ and $\mathrm{PSL}_2(\mathbb{F}_q)$ are finite-sized groups, the new Cayley graphs are finite and contain cycles.
- We see that instead of the ring $\mathcal{M}_2(\mathbb{Z})$ (as in the first construction by Margulis) we now calculate in the ring $\mathbb{H}(\mathbb{Z})$. But finally the graphs are presented as Cayley graphs over some matrix groups by using some isomorphisms from the quaternions to matrix algebras.

Although the mathematics behind the graphs $\mathsf{X}_{q,p}^{\mathrm{LPS}}$ is heavy, it is finally not too difficult to work with these graphs.

Example 4.3 **($\mathsf{X}_{q,p}^{\mathrm{LPS}}$ graph with $q = 17$ and $p = 5$)** Let us choose $q = 17$ and $p = 5$. Note that $q = 1 \pmod 4$ and $p = 1 \pmod 4$.

- As 5 is a quadratic non-residue modulo 17, the Legendre symbol $\left(\frac{p}{q}\right)$ equals -1. Therefore $\phi(\Lambda(2))$ equals $\mathrm{PGL}_2(17)$; a possible presentation of this group is given in Sec. A.4.3.
- To find the set of generators $\phi(\mathcal{S})$ we must first determine the set $\mathcal{F}_5$ of size $p + 1 = 6$ according to Def. B.4:

  $$\mathcal{F}_5 = \left\{1 \pm 2\mathbf{i} + 0\mathbf{j} + 0\mathbf{k}, 1 + 0\mathbf{i} \pm 2\mathbf{j} + 0\mathbf{k}, 1 + 0\mathbf{i} + 0\mathbf{j} \pm 2\mathbf{k}\right\}.$$

  Under the mapping ϕ as in Def. B.29 (in Th. B.27 we choose $u = 4$) we obtain the set $\phi(\mathcal{S})$ having $p + 1 = 6$ elements:

  $$\phi(\mathcal{S}) = \left\{\begin{pmatrix} 1 \pm 8 & 0 \\ 0 & 1 \mp 8 \end{pmatrix}, \begin{pmatrix} 1 & \pm 2 \\ \mp 2 & 1 \end{pmatrix}, \begin{pmatrix} 1 & \pm 8 \\ \pm 8 & 1 \end{pmatrix}\right\}.$$

  Note that the determinant of $\phi(s)$ is $p \pmod q$, i.e. $5 \pmod{17}$ for all $s \in \mathcal{S}$. Looking up in Table F.1, we see that $\mathsf{X}_{17,5}^{\mathrm{LPS}}$ has 4896 vertices, girth 8, and diameter 9.

□

Remark 4.4 (Morgenstern graph construction) Morgenstern [86] has given a construction of Ramanujan graphs with the same parameters q and p as above, but now p can be any prime power and q must be a power of p. The basic idea is still the same but the mathematics behind gets even more complicated (algebraic function fields, etc.). □

Remark 4.5 (Mansour et al. construction) Mansour et al. [78] have recently proposed some extensions to the LPS construction. By studying the case where q is not necessarily a prime power, they were able to construct graphs for an extended set of parameters q and p. □

4.3.2 Properties of $\mathsf{X}_{q,p}^{\mathrm{LPS}}$

The graphs $\mathsf{X}_{q,p}^{\mathrm{LPS}}$ have the properties stated in the next theorem; see also Tab. F.1 for a list of graph parameters.

Theorem 4.6 (Lubotzky, Phillips, Sarnak [65]) *Let q and p be any distinct odd primes with $q > \sqrt{p}$. The graph $\mathsf{X}_{q,p}^{\mathrm{LPS}}$ has the following properties.*

1. $\left(\frac{p}{q}\right) = -1$; *$\mathsf{X}_{q,p}^{\mathrm{LPS}}$ has order $n = |\mathsf{X}_{q,p}^{\mathrm{LPS}}| = q(q^2-1)$ and is bipartite.*
   - *Results about the girth $g(\mathsf{X}_{q,p}^{\mathrm{LPS}})$:*
     - *Upper bound: $g(\mathsf{X}_{q,p}^{\mathrm{LPS}}) \leq 4\log_p(q) + \log_p(4) + 2$.*
     - *Exact result: let $x(q) \triangleq \lceil 2\log_p(q)\rceil$. If $p^{x(q)} - q^2$ can be written as $4^a(8b+7)$ with non-negative integers a and b, then $g(\mathsf{X}_{q,p}^{\mathrm{LPS}}) = 2\lceil 2\log_p(q) + \log_p(2)\rceil$, otherwise $g(\mathsf{X}_{q,p}^{\mathrm{LPS}}) = 2x(q)$.*
     - *Lower bound: $g(\mathsf{X}_{q,p}^{\mathrm{LPS}}) \geq 4\log_p(q) - \log_p(4)$.*
   - $\mathrm{diam}(\mathsf{X}_{q,p}^{\mathrm{LPS}}) \leq 2\log_p(n) + 2\log_p(2) + 1$.
   - *It is Ramanujan (see Def. 2.14), therefore a good expander.*
2. $\left(\frac{p}{q}\right) = +1$; *$\mathsf{X}_{q,p}^{\mathrm{LPS}}$ is of order $n = |\mathsf{X}_{q,p}^{\mathrm{LPS}}| = q(q^2-1)/2$ and is not bipartite,*
   - $g(\mathsf{X}_{q,p}^{\mathrm{LPS}}) \geq 2\log_p(q)$,
   - $\mathrm{diam}(\mathsf{X}_{q,p}^{\mathrm{LPS}}) \leq 2\log_p(n) + 2\log_p(2) + 1$,
   - $i(\mathsf{X}_{q,p}^{\mathrm{LPS}}) \leq (2\sqrt{p}/(p+1))n$,
   - $\chi(\mathsf{X}_{q,p}^{\mathrm{LPS}}) \geq (p+1)/(2\sqrt{p})$.
   - *It is Ramanujan (see Def. 2.14), therefore a good expander.*

Proof: The proof for $q = 1 \,(\mathrm{mod}\ 4)$ and $p = 1 \,(\mathrm{mod}\ 4)$ is given in [65]. But using the extended definitions as we gave them in App. B, the statements hold for any distinct odd primes q and p. The upper bound on the girth and the exact result about the girth in the first case stem from [17]. See also [84] (where Margulis' m has to be set to $2q$) and [83]. □

Remark 4.7 (Girth results) Note that in the case $\left(\frac{p}{q}\right) = -1$ the lower and upper bound on the girth establish the value of $g(\mathsf{X}_{q,p}^{\mathrm{LPS}})$ almost exactly, since $g(\mathsf{X}_{q,p}^{\mathrm{LPS}})$ must be an even integer lying in the interval $[4\log_p(q) - \log_p(4), 4\log_p(q) + \log_p(4) + 2]$, whose length is $2 + \log_p(16)$. For $p \geq 5$ this interval contains at most two even numbers, and in most cases only one, thus determining $g(\mathsf{X}_{q,p}^{\mathrm{LPS}})$ exactly.

Let us also mention that the lower bound results on the girth for the above non-bipartite graphs can be doubled when considering the bipartite double cover of the corresponding graphs, see [117]. □

4.3.3 The Extended Construction

Let us propose the following construction which combines different $\mathsf{X}_{q,p}^{\mathrm{LPS}}$ graphs.

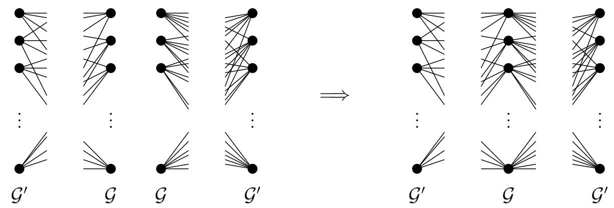

Figure 4.6: Concatenation of two LPS Ramanujan graphs (shown here for $p_1 = 3$ and $p_2 = 5$). Left: $\mathsf{X}_{q,p_1}^{\mathrm{LPS}}$ of degree $p_1 + 1 = 4$ and $\mathsf{X}_{q,p_2}^{\mathrm{LPS}}$ of degree $p_2 + 1 = 6$. Right: the joint graph where the vertices from $\mathcal{G}$ of $\mathsf{X}_{q,p_1}^{\mathrm{LPS}}$ and $\mathsf{X}_{q,p_2}^{\mathrm{LPS}}$ have been identified.

Definition 4.8 (Extended graphs $\mathsf{X}^{\mathrm{LPS}}_{q,p_1,p_2}$)

- Fix an odd prime q. Choose two distinct odd primes p_1 and p_2 that are also distinct from q and such that $\left(\frac{p_1}{q}\right) = \left(\frac{p_2}{q}\right) = -1$, $q > \sqrt{p_1}$, and $q > \sqrt{p_2}$. (We just give the construction for this case. More general constructions can be given.)

- Let $\mathcal{G} \triangleq \mathrm{PSL}_2(\mathbb{F}_q)$ and $\mathcal{G}'$ be the coset of $\mathrm{PSL}_2(\mathbb{F}_q)$ in $\mathrm{PGL}_2(\mathbb{F}_q)$. Because the graphs $\mathsf{X}^{\mathrm{LPS}}_{q,p_1}$ and $\mathsf{X}^{\mathrm{LPS}}_{q,p_2}$ are both bipartite they can be drawn as shown in Fig. 4.6).

- Now we identify the vertices from $\mathcal{G}$ of $\mathsf{X}^{\mathrm{LPS}}_{q,p_1}$ and $\mathsf{X}^{\mathrm{LPS}}_{q,p_2}$: the result is the desired graph $\mathsf{X}^{\mathrm{LPS}}_{q,p_1,p_2}$. Obviously, a third of the vertices has degree $p_1 + 1$, a third of the vertices has degree $p_2 + 1$, and a third of the vertices has degree $p_1 + p_2 + 2$.

□

Theorem 4.9 *As in Def. 4.8, let $\mathsf{X}^{\mathrm{LPS}}_{q,p_1}$ and $\mathsf{X}^{\mathrm{LPS}}_{q,p_2}$ be two LPS Ramanujan graphs with $p_1 \neq p_2$ and $\left(\frac{p_1}{q}\right) = \left(\frac{p_2}{q}\right) = -1$. The joint graph $\mathsf{X}^{\mathrm{LPS}}_{q,p_1,p_2}$ as constructed in Fig. 4.6 has the following girth:*

$$g(\mathsf{X}^{\mathrm{LPS}}_{q,p_1,p_2}) = \min\left(8, g(\mathsf{X}^{\mathrm{LPS}}_{q,p_1}), g(\mathsf{X}^{\mathrm{LPS}}_{q,p_2})\right).$$

Proof: See Sec. 4.5.2. □

Remark 4.10 (Graph modification) When one does not take all edges of $\mathsf{X}^{\mathrm{LPS}}_{q,p_1}$ and $\mathsf{X}^{\mathrm{LPS}}_{q,p_2}$ in the construction of $\mathsf{X}^{\mathrm{LPS}}_{q,p_1,p_2}$ (e.g. by not taking all generators), it is easily possible for the modified joint graph to have larger girth. □

4.3.4 Girth Benchmark

Fig. 4.7 captures different results concerning the girth of various graph constructions. We assume the graphs X to be r-regular and of size $n(\mathsf{X})$.

- **(Lower bound)** 0 is trivially a lower bound on the girth.

- **(Upper bound)** The stated upper bound follows from simple counting arguments.

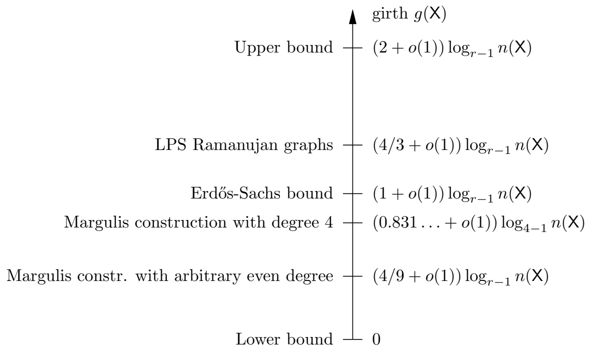

Figure 4.7: Girth benchmark. (See the comments in the text.)

- **(Erdős-Sachs bound)** Erdős and Sachs [28] gave a non-constructive proof that graphs exist whose girth is larger than $\log_{r-1} n(\mathsf{X})$. As mentioned by Margulis [84], random constructions have a girth that lies asymptotically at the Erdős-Sachs bound.

- **(Margulis construction with degree 4)** This is the construction shown in Sec. 4.2.

- **(Margulis constr. with arbitrary even degree)** This is a construction by Margulis [82] not discussed in this thesis.

- **(LPS Ramanujan graphs)** These are the Ramanujan graphs by Lubotzky, Phillips, and Sarnak discussed in Sec. 4.3 for the case where $\left(\frac{p}{q}\right) = -1$.

Remark 4.11 (Girth bounds vs. minimum distance bounds in coding) There is a striking similarity between these girth results and some minimum distance results known from coding theory. The girth upper bound seems to correspond to the Hamming bound [76], the Erdős-Sachs bound seems to correspond to the Gilbert-Varshamov bound [76], and the LPS Ramanujan graphs whose girth is above the Erdős-Sachs bound seem to correspond to some algebraic geometric codes whose minimum distance is above the Gilbert-Varshamov bound. □

4.4 Finite Generalized Polygons

After the constructions of Cayley graphs in the last sections we now turn to graphs derived from finite geometries with a special emphasis on finite generalized polygons. Finite generalized polygons are extremal in the sense that the girth is the largest for a given diameter, namely twice the diameter. A finite generalized n-gon has by definition girth $2n$ and diameter n.

4.4.1 Point-Line Incidence Structures

We start by giving the definition of an incidence structure; Fig. 4.8 shows the general setting and Fig. 4.9 gives an example.

Definition 4.12 (Incidence structure) Let $\Gamma = \Gamma(\mathcal{P}, \mathcal{L}, \mathtt{I})$ be an incidence structure (or incidence plane as in [30], sometimes also called a geometry of rank 2).

- **(Point set)** $\mathcal{P}$ is the point set.
- **(Line set)** $\mathcal{L}$ is the line set.
- **(Incidence relation)** $\mathtt{I} \subseteq \mathcal{P} \times \mathcal{L}$ is the incidence relation. (Throughout we tacitly assume that $\mathcal{P} \cap \mathcal{L} = \emptyset$.)
- **(Incidence)** A point $p \in \mathcal{P}$ is incident on (or lies on) a line $L \in \mathcal{L}$ if and only if $(p, L) \in \mathtt{I}$, which we denote by $p \,\mathtt{I}\, L$.

□

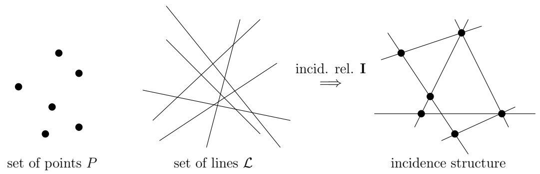

Figure 4.8: An incidence structure consists of points, lines and an incidence relation.

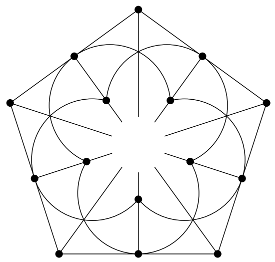

Figure 4.9: Finite generalized quadrangle $W(2)$ of order $(s, t) = (2, 2)$; all lines through the center are to be continued. (From [12].)

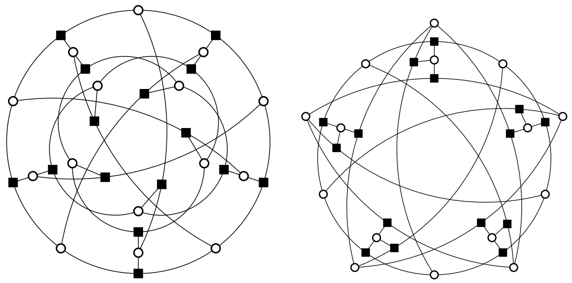

Figure 4.10: Point-line graph associated to the incidence structure in Fig. 4.9. (The left- and right-hand side graphs are essentially the same, the arrangement of the vertices and edges is varied.)

Definition 4.13 (The Point-line graph associated to an incidence structure) To an incidence structure $\Gamma = \Gamma(\mathcal{P}, \mathcal{L}, \mathtt{I})$ we can associate a point-line graph $\mathsf{X} = (\mathcal{V}, \mathcal{E})$ with the vertex set $\mathcal{V} = \mathcal{P} \cup \mathcal{L}$, and there is an edge connecting two vertices if and only if there is an incidence relation between these two vertices. Such graphs are by definition bipartite. □

Example 4.14 (The Point-line graph associated to an incidence structure) The point-line graph associated to the finite generalized quadrangle in Fig. 4.9 is given in Fig. 4.10: the points are drawn as empty circles and the lines as filled squares to highlight the fact that such a graph is always bipartite. □

Definition 4.15 (Matrix associated to an incidence structure) To an incidence structure Γ we can also associate a matrix, namely the incidence matrix with entries 0's and 1's: if the lines correspond to the rows and the points to the columns, respectively, then we shall call this matrix $\mathbf{I}(\Gamma)$. □

Definition 4.16 (Dual incidence structure) Obviously, to every incidence structure, a dual incidence structure can be defined in which points become lines and vice versa. □

4.4.2 Definition, Existence, Properties, and Examples of Finite Generalized Polygons

Definition 4.17 (Weak generalized n-gon) A weak generalized n-gon is a geometry $\Gamma = \Gamma(\mathcal{P}, \mathcal{L}, \mathtt{I})$ such that the following two axioms are satisfied:

- Γ contains no ordinary k-gon (as a subgeometry), for $2 \leq k < n$.
- Any two elements $x, y \in \mathcal{P} \cup \mathcal{L}$ are contained in some ordinary n-gon (again as a subgeometry) in Γ, a so-called apartment.

(Note that also other collections of axioms would be possible.) Instead of 3-gons, 4-gons, 5-gons, 6-gons, 8-gons, and n-gons we will also speak of triangles, quadrangles, pentagons, hexagons, octagons, and polygons. We will use the following abbreviations: FGT (finite generalized triangle), FGQ (finite generalized quadrangle), FGH (finite generalized hexagon), FGO (finite generalized octagon). FGP is used for "finite generalized polygon" and *not* for "finite generalized pentagon". □

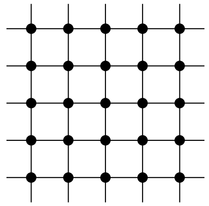

Figure 4.11: Incidence structure of a grid, which is a weak finite generalized quadrangle.

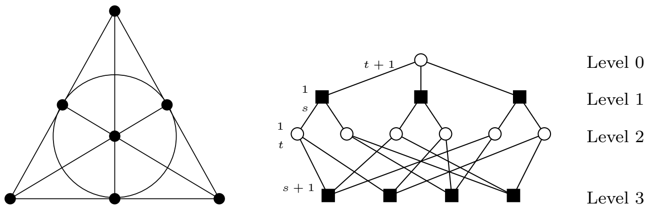

Figure 4.12: Projective plane with seven points and seven lines. Left: points and lines. Right: derived point-line graph.

Definition 4.18 (Generalized n-gon) A generalized n-gon is a weak generalized n-gon Γ which satisfies also the following axiom.

- Γ is *thick*, i.e. through each point there are *at least three* lines and *at least three* points lie on each line.

□

Definition 4.19 If

- on *each line* there are $s+1$ points and
- through *each point* there are $t+1$ lines,

we say that the FGP has *order* (s, t). □

Therefore, an FGP of order (s, t) is thick (see Def. 4.18) if $s > 1$ and $t > 1$. It is called "thin" if $s = 1$ and/or $t = 1$. Note that an ordinary

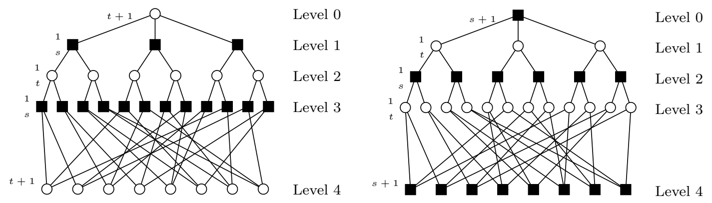

Figure 4.13: Point-line graph of an finite generalized quadrangle with parameters $(s, t) = (2, 2)$. Left: a point is at the top. Right: a line is a the top.

polygon has $(s, t) = (1, 1)$.

We now give different examples of FGPs.

Example 4.20 (Grid as a weak FGQ) A so-called grid is a weak finite generalized quadrangle where $s = 1$ or $t = 1$, see e.g. Fig. 4.11. Codes derived from grids are known as product codes (see e.g. Fig. 3 in [109]). □

Example 4.21 (Finite projective planes) Finite projective planes are incidence structures where

- two distinct points lie on a common line,
- two line cross in exactly one point.

Codes based on finite projective planes have already been studied a long time ago (interestingly, some of them turn out to be cyclic). At that time, the codes were decoded by the one-step majority logic decoding algorithm (see e.g. [61]). Recently, these codes were recognized to be suited for iterative decoding (see e.g. [67, 57] but also the somewhat earlier work in [54, 87]). □

Example 4.22 (Finite Generalized Quadrangle of order $(s, t) = (2, 2)$) An FGQ of order $(s, t) = (2, 2)$ is shown in Fig. 4.9. Point-line graphs of FGQs of order $(s, t) = (2, 2)$ are shown in Figs. 4.10 and 4.13. □

Feit and Higman [30] have given conditions on the existence of FGPs: they found that interesting structures are only possible for n up to 8.

Theorem 4.23 (Existence of FGPs) *Let Γ be a finite generalized n–gon with order (s, t). Let f be the square-free part of st.*

- *If Γ is not an ordinary polygon, then n must be in $\{2, 3, 4, 6, 8, 12\}$,*
- *If $s > 1$ and $t > 1$ (i.e. Γ is a thick FGP) then n must be in $\{2, 3, 4, 6, 8\}$.*
- *If $n = 3$ or $n = 6$ then $f = 1$.*
- *If $n = 8$ then $f = 2$.*
- *If n is odd, then $s = t$.*

Proof: (The proof in [30] is very elegant; we give here an outline of it.) Fix a positive integer n. The proof looks at the matrix $\mathbf{L} = \mathbf{I}(\Gamma)^T \mathbf{I}(\Gamma)$ ($\mathbf{I}(\Gamma)$ is the incidence matrix, lines are rows, points are columns) of a hypothetical finite generalized n-gon Γ. It is then possible to study the multiplicities of the eigenvalues of $\mathbf{L}$ by the technique shown in Lemma A.13: if some of these multiplicities are not fractions (and therefore not integers), such a finite generalized n-gon cannot exist. □

Lemma 4.24 (Number of points and lines in an FGP) *An FGT of order (s, t) has $|\mathcal{P}| = st + s + 1$ points and $|\mathcal{L}| = st + t + 1$ lines. An FGQ of order (s, t) has $|\mathcal{P}| = (s + 1)(st + 1)$ points and $|\mathcal{L}| = (t + 1)(st + 1)$ lines. Similar formulas can be given for FGHs and FGOs (see e.g. [120]).*

Proof: See Sec. 4.5.3. □

4.4.3 Explicit Constructions of Finite Generalized Quadrangles

In order to give the reader a flavor of how FGQs can be constructed, we discuss an FGQ called $W(q)$. Before we can do so, we need the definition of a projective space.

Definition 4.25 (Projective space) The type of projective spaces (or projective geometries) of n dimensions we will use can be embedded in an affine space of $n + 1$ dimensions.

The points of the projective space (called proj. points) are the one-dimensional subspaces of the affine space. The lines of the projective space (called proj. lines) are the two-dimensional subspaces of the affine space. A proj. point lies on a proj. line if and only if the corresponding one-dimensional affine subspace is a subspace of the corresponding two-dimensional affine subspace.

To be specific, the representation will be as follows. Let $\mathbf{x}$ be an $(n+1)$-dimensional vector representing a point in an $(n+1)$-dimensional affine space. The subspace through $\mathbf{x}$ will be denoted by $\langle \mathbf{x} \rangle$: this will also be the label of the corresponding projective point. Note that $\langle \mathbf{x} \rangle = \langle a \cdot \mathbf{x} \rangle$ for any nonzero a. Two proj. points $\langle \mathbf{x} \rangle$ and $\langle \mathbf{y} \rangle$ determine a projective line if $\langle \mathbf{x} \rangle \neq \langle \mathbf{y} \rangle$, i.e. if $\mathbf{x}$ and $\mathbf{y}$ do not lie in the same affine subspace. □

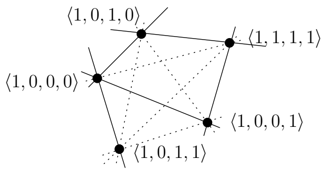

Figure 4.14: Part of a three-dimensional finite projective space over $\mathbb{F}_2$. Proj. points are the vertices denoted by $\langle x_0, x_1, x_2, x_3 \rangle$ and the (solid or dotted) lines are the proj. lines. Let $\langle \mathbf{x} \rangle$ and $\langle \mathbf{y} \rangle$ be two projective points: if $f(\mathbf{x}, \mathbf{y}) = 0$, then the points $\langle \mathbf{x} \rangle$ and $\langle \mathbf{y} \rangle$ are connected by a solid line, otherwise by a dotted line. The solid lines are therefore the lines of the FGQ $W(2)$ in Ex. 4.26.

The next example gives the details of a typical construction of an FGQ.

Example 4.26 (The FGQ $W(q)$ based on a symplectic polarity) We construct an FGQ called $W(q)$ of order $(s,t) = (q,q)$ where q is a prime power.

Let us first give a *short* (but complete!) description: the points are the points of $\mathrm{PG}(3,q)$, the lines are the totally isotropic lines with respect to a symplectic polarity, and incidence is the one inherited from $\mathrm{PG}(3,q)$.

Let us give now a more *detailed* description. To define an FGQ we have to say what the set of points, the set of lines, and the incidence relation are.

- **(Underlying projective space)** We consider an $n = 3$ dimensional projective space which can be embedded in an $n+1 = 4$ dimensional affine space. Proj. points will be of the form $\langle \mathbf{x} \rangle$ where $\mathbf{x} = (x_0, x_1, x_2, x_3) \in \mathbb{F}_q^n$ and $\mathbf{x} \neq \mathbf{0}$.
- **(Symplectic polarity)** We introduce the following canonical[6] symplectic bilinear form (symplectic polarity):

$$f(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} & +1 & & \\ -1 & & & \\ & & & +1 \\ & & -1 & \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}.$$

- **(Point set)** The set of points of the FGQ equals the set of all proj. points in this 3-dimensional projective space.
- **(Line set)** The set of lines of the FGQ consists of all proj. lines which fulfill a certain condition: a proj. line defined by two distinct proj. points $\langle \mathbf{x} \rangle$ and $\langle \mathbf{y} \rangle$ is a line of the FGQ if and only if $f(\mathbf{x}, \mathbf{y}) = 0$. (Note that this condition is well-defined.) Proj. lines fulfilling this condition are called totally isotropic lines.
- **(Incidence)** Then, a point of the FGQ is on a line of the FGQ if and only if the corresponding proj. point lies on the corresponding proj. line (i.e. the incidence is the one inherited from the projective space).

For $q = 2$, we show parts of the underlying projective space in Fig. 4.14 and indicate what proj. lines are also lines of the FGQ.

The incidence structure defined above contains quadrangles: e.g. the quadrangle given by the sequence of proj. points $\langle 1,0,0,0 \rangle$, $\langle 1,0,0,1 \rangle$, $\langle 1,1,1,1 \rangle$, $\langle 1,0,1,0 \rangle$, $\langle 1,0,0,0 \rangle$. But as shown in Sec. 4.5.4, it does not contain any triangles. The complete proof showing that $W(q)$ is an FGQ can be found e.g. in [89] or [120]. For $q = 2$, the FGQ $W(q)$ is shown in Fig. 4.9. □

[6] Taking any other symplectic polarity gives an isomorphic FGQ.

Table F.2 lists different FGQs. The description of explicit constructions can be found e.g. in [89] (where possible isomorphisms and dualities between the classes are mentioned) or [90] (this paper gives an explicit construction for each at that time known (i.e. in 1990) isomorphism class of FGQs.)

Before finishing this section, let us mention that constructions of finite generalized *hexagons* (FGHs) of orders (q, q) and (q, q^3) related to the groups $G_2(q)$ and ${}^3D_4(q)$ can be found in [22] for q even and in [6] for q odd. Note that the point-line graphs of FGHs and FGOs can be drawn similarly to FGTs and FGQs in Figs. 4.12 (right) and 4.13, respectively: but FGHs would have levels 0 to 6 and FGOs would have levels 0 to 8.

4.4.4 The Relation of Finite Generalized Polygons to Partial Geometries

Johnson and Weller looked at the suitability of the class of partial geometries [51] to derive LDPC codes from them. In this subsection we would like to show what the connections between FGPs and partial geometries are.

Partial geometries are a large class of combinatorial structures that include Steiner triple systems [71], Kirkman triple systems [49, 50],oval designs [127], FGQs, some of the finite geometries of [54, 87, 67, 57] (the references are to papers in iterative decoding that used such constructions). Tab. F.7 lists some of the possible partial geometries. Bose [19] gave the following definition of a partial geometry.

Definition 4.27 (Partial geometry) A partial geometry is an incidence structure $\Gamma = \Gamma(\mathcal{P}, \mathcal{L}, \mathtt{I})$ that fulfills the following axioms. Let s, t, and α be some positive integers.

- Each point p is incident with $t+1$ lines and each line L is incident with $s+1$ points.
- Any two lines have at most one point in common.
- For any non-incident point-line pair (p, L) the number of lines incident with p and intersecting L equals α.

□

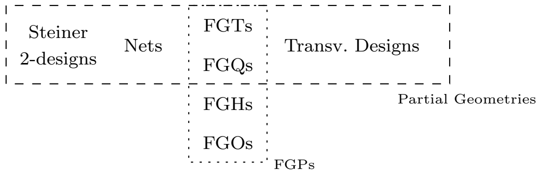

Figure 4.15: FGTs and FGQs are at the intersection of the class of finite generalized polygons (FGPs) and partial geometries. Steiner 2-designs are also known as balanced incomplete block designs (BIBDs) with $\lambda = 1$. (The ordering within the class of partial geometries was chosen arbitrarily.)

Lemma 4.28 (Number of points and lines in a partial geometry) *A partial geometry of order (s, t) and parameter α has $|\mathcal{P}| = (s+1)(st+\alpha)/\alpha$ points and $|\mathcal{L}| = (t+1)(st+\alpha)/\alpha$ lines.*

Proof: Follows from counting arguments as in Lemma 4.24. □

Remark 4.29 (FGPs and partial geometries) From Defs. 4.17 and 4.27 it follows that FGTs and FGQs belong to the class of partial geometries, but FGHs and FGOs do not. Note that partial geometries have girth six if $\alpha > 1$. The only exception are the FGQs (partial geometries with $\alpha = 1$) which have girth eight. □

Remark 4.30 (Resolvability) Some of the partial geometries have the interesting feature that they are resolvable: this means that one can define a new incidence structure with all original points but only a subset of the lines such that all points in the new incidence structure lie on the same number of lines. (This property can be used when deriving different regular LDPC codes from a single partial geometry). □

4.5 Proofs

4.5.1 Proof of Theorem 4.1

We will look at a pair of walks in the Cayley graph $\mathsf{X}(\mathcal{G}, \mathcal{S})$. Because of the homogeneity property of Cayley graphs (see Prop. 2.26), we can without loss of generality assume that these walks start at the vertex with label $\mathbb{1}$ (i.e. the neutral element of the corresponding multiplicative matrix groups). We will also study the corresponding pair of walks in the Cayley graph $\mathsf{X}_q^{\mathrm{Mar}} = \mathsf{X}(\mathcal{G}_q, \mathcal{S}_q)$; also here we can without loss of generality assume that they start at the same vertex $\phi(\mathbb{1}) = \mathbb{1}$.

- **(Pair of walks in the Cayley graph $\mathsf{X}(\mathcal{G}, \mathcal{S})$)** Consider the two different walks (see Fig. 4.16 (left))

$$\begin{aligned} W &= (w_0, w_1, \ldots, w_r), \\ W' &= (w'_0, w'_1, \ldots, w'_{r'}) \end{aligned}$$

  of lengths $r \geq 1$ and $r' \geq 1$, respectively, with $w_0 = w'_0 = \mathbb{1}$. We have $w_i = w_{i-1}u_i$ and $w'_i = w'_{i-1}u'_i$ for some $u_i, u'_i \in \mathcal{S}$. Because $\mathsf{X}(\mathcal{G}, \mathcal{S})$ is a tree, we have $u_1 \cdots u_r \neq u'_1 \cdots u'_{r'}$, and because of the definition of a walk (Def. 2.11), the words $u_1 \cdots u_r$ and $u'_1 \cdots u'_{r'}$ are reduced, and therefore $w_r \neq w'_{r'}$. But in order to continue, we assume that the mappings of w_r and $w'_{r'}$, respectively, are equal, i.e., $\phi(w_r) = \phi(w'_{r'})$.

- **(Pair of walks in the Cayley graph $\mathsf{X}_q^{\mathrm{Mar}} = \mathsf{X}(\mathcal{G}_q, \mathcal{S}_q)$)** The second pair of walks is the first pair of walks under the mapping ϕ (see Fig. 4.16 (right)):

$$\begin{aligned} \tilde{W} &= \phi(W) = (\tilde{w}_0, \tilde{w}_1, \ldots, \tilde{w}_r), \\ \tilde{W}' &= \phi(W') = (\tilde{w}'_0, \tilde{w}'_1, \ldots, \tilde{w}'_{r'}) \end{aligned}$$

  of lengths $r \geq 1$ and $r' \geq 1$, respectively, with $\tilde{w}_0 = \tilde{w}'_0 = \mathbb{1}$. Because of the assumption $\phi(w_r) = \phi(w'_{r'})$, the end points of the second pair of walks must be equal, i.e., $\tilde{w}_r = \tilde{w}'_{r'}$. Moreover, we have $\tilde{w}_i = \tilde{w}_{i-1}\tilde{u}_i$ and $\tilde{w}'_i = \tilde{w}'_{i-1}\tilde{u}'_i$ for $\tilde{u}_i = \phi(u_i) \in \mathcal{S}_q$ and $\tilde{u}'_i = \phi(u'_i) \in \mathcal{S}_q$, and so $\tilde{u}_1 \cdots \tilde{u}_r = \tilde{u}'_1 \cdots \tilde{u}'_{r'}$.

We therefore have

$$\begin{aligned} u_1 \cdots u_r - u'_1 \cdots u'_{r'} &\neq \mathbf{0}, && (4.3) \\ \tilde{u}_1 \cdots \tilde{u}_r - \tilde{u}'_1 \cdots \tilde{u}'_{r'} &= \mathbf{0}. && (4.4) \end{aligned}$$

The idea now is to give an upper and a lower bound on the expression $||\mathbf{M}||_2$ (where $||\mathbf{M}||_2$ is the matrix norm induced from the L_2 vector norm,

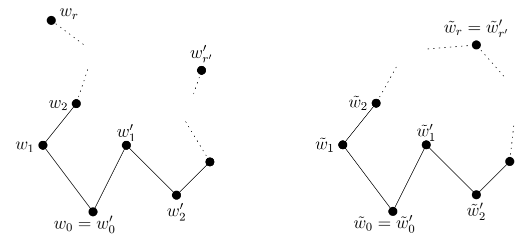

Figure 4.16: Left: Pair of walks in the Cayley graph $\mathsf{X}(\mathcal{G}, \mathcal{S})$. Right: Pair of walks in the Cayley graph $\mathsf{X}_q^{\mathrm{Mar}} = \mathsf{X}(\mathcal{G}_q, \mathcal{S}_q)$).

see definitions in Sec. A.5) of the matrix $\mathbf{M} \triangleq u_1 \cdots u_r - u'_1 \cdots u'_{r'}$ in the ring $\mathcal{M}_2(\mathbb{Z})$.[7] For the matrices in the set $\mathcal{S}$ it turns out that the norms are all the same, namely

$$||\mathbf{A}||_2 = ||\mathbf{B}||_2 = ||\mathbf{A}^{-1}||_2 = ||\mathbf{B}^{-1}||_2 = 1 + \sqrt{2} = 2.4142\ldots \triangleq \alpha.$$

- **(Upper bound on $||\mathbf{M}||_2$)** An upper bound on $||\mathbf{M}||_2$ can be derived as follows

$$\begin{aligned} ||u_1 \cdots u_r - u'_1 \cdots u'_{r'}||_2 &\overset{(*)}{\leq} ||u_1 \cdots u_r||_2 + ||u'_1 \cdots u'_{r'}||_2 \\ &\overset{(**)}{\leq} ||u_1||_2 \cdots ||u_r||_2 + ||u'_1||_2 \cdots ||u'_{r'}||_2 \\ &\overset{(***)}{=} \alpha^r + \alpha^{r'} \leq 2\alpha^{\max(r,r')}, \end{aligned}$$

  where inequality $(*)$ follows from (A.2), inequality $(**)$ from (A.3), and equality $(***)$ from the fact that $u_i, u'_i \in \mathcal{S}$.

- **(Lower bound on $||\mathbf{M}||_2$)** To obtain a lower bound on $||\mathbf{M}||_2$, we note that (using the homomorphism property of ϕ and (4.4))

$$\begin{aligned} \phi\big(u_1 \cdots u_r - u'_1 \cdots u'_{r'}\big) &= \phi(u_1) \cdots \phi(u_r) - \phi(u'_1) \cdots \phi(u'_{r'}) \\ &= \tilde{u}_1 \cdots \tilde{u}_r - \tilde{u}'_1 \cdots \tilde{u}'_{r'} = \mathbf{0}, \end{aligned}$$

  from which we conclude that all entries of $\mathbf{M}$ must be multiples of q, at least one of which is nonzero (because of (4.3)). Therefore

[7] See the comment in Footnote 2 on page 60.

$||\mathbf{M}||_2 = q \cdot ||\mathbf{M}'||_2$, where $\mathbf{M}'$ is a matrix with integer entries, at least one of which is nonzero. It can easily be seen from the definition of the matrix norm in Def. A.11 that $||\mathbf{M}'||_2 \geq 1$, from which we get $||\mathbf{M}||_2 \geq q$.

Combining the upper and the lower bound on $||\mathbf{M}||_2$ we get $q \leq ||\mathbf{M}_2|| \leq 2\alpha^{\max(r,r')}$ or $\max(r, r') \geq \log_\alpha(q/2)$. Given a cycle of even length in $\mathsf{X}(\mathcal{G}_q, \mathcal{S}_q)$, we can choose the walks such that $r = r'$, so that the length $r + r' = 2r$ of the cycle is at least $2\log_\alpha(q/2)$. Given a cycle of odd length in $\mathsf{X}(\mathcal{G}_q, \mathcal{S}_q)$, we can choose the walks such that $r = r' + 1$, so that the length $r + r' = 2r - 1$ of the cycle is at least $2\log_\alpha(q/2) - 1$. Upon remarking that $|\mathcal{G}_q| = |\operatorname{SL}_2(\mathbb{Z})| = q(q^2 - 1)$ we get the theorem. □

4.5.2 Proof of Theorem 4.9

Every cycle that involves only edges in the left part of the joint graph $\mathsf{X}^{\mathrm{LPS}}_{q,p_1,p_2}$ is a cycle in the original graph $\mathsf{X}^{\mathrm{LPS}}_{q,p_1}$; therefore $g(\mathsf{X}^{\mathrm{LPS}}_{q,p_1})$ is an upper bound on $g(\mathsf{X}^{\mathrm{LPS}}_{q,p_1,p_2})$. Equivalently, every cycle that involves only edges in the right part of the joint graph $\mathsf{X}^{\mathrm{LPS}}_{q,p_1,p_2}$ is a cycle in the original graph $\mathsf{X}^{\mathrm{LPS}}_{q,p_2}$; therefore $g(\mathsf{X}^{\mathrm{LPS}}_{q,p_2})$ is an upper bound on $g(\mathsf{X}^{\mathrm{LPS}}_{q,p_1,p_2})$. It only remains to analyze the cycles that involve edges from both parts.

Let α_i's be quaternions from the set $\mathcal{F}_{p_1}$ (therefore they have norm $N(\alpha_i) = p_1$) and let β_i's be quaternions from the set $\mathcal{F}_{p_2}$ (therefore they have norm $N(\beta_i) = p_2$).

It suffices to analyze cycles that start and end in vertex $\mathbb{1}$ of $\mathcal{G}$. There are two types of cycles that can occur: the first are the ones that are generated by the mapping ϕ. But using similar arguments as in [65], the length of such cycles must be at least as large as the minimum of $g(\mathsf{X}^{\mathrm{LPS}}_{q,p_1})$ and $g(\mathsf{X}^{\mathrm{LPS}}_{q,p_2})$. The second type of cycles are cycles that are already present in the underlying graph (note that now the underlying graph, i.e. the graph before the mapping ϕ, can also have cycles). In quaternion language (i.e. after properly generalizing $\Lambda(2)$ from Def. B.21), we have to look for cycles that start at quaternion 1 and end at a quaternion of the form $p_1^{m_1} p_2^{m_2}$, $m_1 \in \mathbb{Z}$, $m_2 \in \mathbb{Z}$, i.e. a quaternion that is in the same congruence class as the quaternion 1).

- A cycle of length four involving both parts would typically visit the quaternions $1, \alpha_1, \alpha_1\alpha_2, \alpha_1\alpha_2\beta_1, \alpha_1\alpha_2\beta_1\beta_2$ for some $\alpha_i \in \mathcal{F}_{p_1}$ and $\beta_i \in \mathcal{F}_{p_2}$. If this is a cycle of length four, $\alpha_1\alpha_2\beta_1\beta_2$ must equal p_1p_2. But from Lemma B.18 it follows that $\alpha_2 = \overline{\alpha_1}$ and $\beta_2 = \overline{\beta_1}$ and therefore that the above walk is not a proper walk.
- A cycle of length six involving both parts would typically be like $1, \ldots, \alpha_1\alpha_2\beta_1\beta_2\beta_3\beta_4$ for some $\alpha_i \in \mathcal{F}_{p_1}$ and $\beta_i \in \mathcal{F}_{p_2}$. But from Lemma B.19 it follows that $\alpha_2 = \overline{\alpha_1}$ and therefore that the above walk is not a proper walk. Using a similar argument as for the previous walk, a walk of the form $1, \ldots, \alpha_1\alpha_2\alpha_3\alpha_4\beta_1\beta_2$ can also be shown to be not a proper walk. The non-existence of other types of walks of length six is shown like this: if there exists e.g. a walk of the form $1, \ldots, \alpha_1\alpha_2\beta_1\beta_2\alpha_3\alpha_4$ then there must also exist a walk of the form $1, \ldots, \alpha_3\alpha_4\alpha_1\alpha_2\beta_1\beta_2$; but such a walk cannot exist as shown above.
- A cycle of length eight exists. This follows from the fact that the quaternion $p_1^2p_2^2$ can be factorized as $\alpha_1\alpha_2\beta_1\beta_2\alpha_3\alpha_4\beta_3\beta_4$ with $\alpha_2 \neq \overline{\alpha_1}$, $\beta_2 \neq \overline{\beta_1}$, $\alpha_4 \neq \overline{\alpha_3}$, and $\beta_4 \neq \overline{\beta_3}$.

□

4.5.3 Proof of Lemma 4.24

For counting the number of points of an FGT of order (s, t) we use Fig. 4.12 (right): at level 0 there is 1 point and at level 2 there are $(t+1) \cdot s$ points, which makes a total of $1 + (t+1) \cdot s = st + s + 1$ points. The counting of numbers of lines of an FGT is similar.

For counting the points of an FGQ we take Fig. 4.13 (right): on level 1 there are $s+1$ points and on level 3 there are $(s+1) \cdot t \cdot s$ points, making a total of $s+1+(s+1) \cdot t \cdot s = s^2t+st+s+1 = (st+1)(s+1)$ points. For counting of the number of lines of an FGQ one can take Fig. 4.13 (left). □

4.5.4 Proof to Example 4.26

We want to show that in the incidence structure defined in Ex. 4.26 there are no triangles. The proof is by contradiction. So, assume that there are three different proj. points $\langle \mathbf{x} \rangle$, $\langle \mathbf{y} \rangle$, and $\langle \mathbf{z} \rangle$ such that the three pairs $\{\langle \mathbf{x} \rangle, \langle \mathbf{y} \rangle\}$, $\{\langle \mathbf{x} \rangle, \langle \mathbf{z} \rangle\}$, and $\{\langle \mathbf{y} \rangle, \langle \mathbf{z} \rangle\}$ each lie on a line of the incidence structure.

Without loss of generality we can assume $\mathbf{x} = (1,0,0,0)$. From the above assumptions we must have $f(\mathbf{x},\mathbf{y}) = 0$ from which $y_1 = 0$ follows. On the other hand, we must also have $f(\mathbf{x},\mathbf{z}) = 0$ from which $z_1 = 0$ follows. Furthermore, we must have $f(\mathbf{y},\mathbf{z}) = 0$, which is equivalent to $y_2 z_3 - y_3 z_2 = 0$. But now we can write

$$(y_0 z_2 - y_2 z_0) \cdot (1,0,0,0) = z_2 \cdot (y_0, 0, y_2, y_3) - y_2 \cdot (z_0, 0, z_2, z_3),$$

i.e. the proj. point $\langle \mathbf{x} \rangle$ lies on the line trough the pair $\{\langle \mathbf{y} \rangle, \langle \mathbf{z} \rangle\}$. This means that the triple $\{\langle \mathbf{x} \rangle, \langle \mathbf{y} \rangle, \langle \mathbf{z} \rangle\}$ must be collinear. This confirms the non-existence of triangles. $\square$

Chapter 5

Algebraic Constructions of Codes for Iterative Decoding

5.1 Introduction and Historical Background

In the preceding chapters we have prepared the stage on which we will now algebraically construct codes that are suitable for iterative decoding. We will construct (binary) low-density parity-check codes in various flavors along with some (binary) turbo codes. The constructions will mostly be of an algebraic nature, sometimes with a random component.

Definition 5.1 (Types of low-density parity-check codes) We use the abbreviation LDPCC for "low-density parity-check code". Furthermore, we will distinguish two large classes of LDPCCs.

- **(SS-LDPCCs)** Let strict-sense low-density parity-check codes (SS-LDPCCs) be LDPCCs where all check nodes are simple parity-checks (i.e. in the partial factor graph representing such an SS-LDPCC, the function nodes will all be f_{XOR} function nodes). *Regular* SS-LDPCCs have factor graphs where all symbol nodes have uniform degree and all check nodes have uniform degree, whereas *irregular* SS-LDPCCs do not have to satisfy these restrictions.

- **(WS-LDPCCs)** By a wide-sense low-density parity-check code (WS-LDPCC) we will mean an LDPCC that has more complex subcodes (i.e. in the partial factor graph representing such a WS-LDPCC, the function nodes are indicator function nodes of any subcodes). *Regular* WS-LDPCCs have uniform symbol degrees and use only one subcode, whereas *irregular* WS-LDPCCs do not have to satisfy these restrictions.

□

5.1.1 Historical Background and Previous Algebraic Constructions

We list some milestones in the development of codes suitable for iterative decoding.

- In the early sixties of the last century, Gallager [37, 38] defined the class of regular SS-LDPCCs.
- In 1981, Tanner [109] generalized this idea to WS-LDPCCs. He also gave some algebraic constructions of SS-LDPCCs and WS-LDPCCs.
- The paper that really started the field was written in 1993 by Berrou, Glavieux, and Thitimajshima [14] about their turbo codes.
- MacKay and Neal [72, 73] rediscovered the SS-LDPCCs, see also [69].

Most constructions of LDPCCs in the literature before we started our project were of a random nature, but there have also been algebraic constructions.

- Tanner [109] gave some algebraic constructions of SS-LDPCCs and WS-LDPCCs in 1981.
- Margulis [82] presented some SS-LDPCCs based on graphs with large girth in 1982. Five years later he presented SS-LDPCCs based on the Ramanujan graphs defined by himself and at the same time by Lubotzky, Phillips, and Sarnak, see [83].

- Karplus and Krit [54] (advised by Tanner) used a well-known class of codes (namely codes derived from finite projective geometries) for iterative decoding, see e.g. [61]. Later, Lucas *et al.* [67] and Kou *et al.* [57] also analyzed the suitability of this class of codes for iterative decoding, see also the related codes in [87].
- At the same time as our paper [99], which discussed an SS-LDPCC construction based on LPS Ramanujan graphs, Lafferty and Rockmore [59] presented some nearly regular WS-LDPCCs based on the LPS Ramanujan graphs [65]. (As a note on the side, whereas the Lafferty/Rockmore construction used only two nearly identical subcodes, our construction in [123] presented irregular WS-LDPCCs where the subcodes were taken from a broad range and some bit nodes had large degree.) Let us also mention that already in 1997, Tillich and Zémor [117] looked at codes derived from Ramanujan graphs; but their focus was on cycle codes derived from such graphs and their performance under maximum-likelihood decoding when transmitting over a BSC.
- Algebraic SS-LDPCCs based on linear congruences were derived by Bond, Hui, and Schmidt [18]. Some follow-up work was performed by O'Sullivan, Greferath, and Smarandache [88].
- Algebraic constructions of SS-LDPCCs where the parity-check matrix is composed of cyclically shifted identity matrices were presented in [115].

Our construction of codes will mainly be based on graphs with large girth. There have been other approaches that also tried to optimize the girth.

- Mao and Banihashemi [80] randomly generated a set of codes; they experimentally found out that the codes in this set which have a large average local girth have a lower bit-error rate than the average code from that set.
- Hu, Eleftheriou, and Arnold [45, 46] presented some random constructions of regular and irregular SS-LDPCCs where they optimized the girth; also here the results were superior compared to non-optimized codes.

5.1.2 Construction Guidelines for LDPCCs

As we mentioned in Sec. 3.5.3, the sum-product algorithm is still poorly understood when it operates on loopy factor graphs. For constructing codes, we will therefore formulate some desirable properties derived from general experiences with iterative decoding. By the "factor graph" we mean in the following the partial factor graph representing the code, see Sec. 3.5.

- **(No small cycles)** It is desirable that the factor graph has no small cycles so as to assure as long as possible the independence of the messages [58]. Therefore, it is desirable that the girth of the factor graph is as large as possible, or that the local girth histogram (see Def. 2.11) is good [80]. (Note that for bipartite graphs the girth is always an even number.)
- **(Small diameter)** The factor graph should have a small diameter so as to allow the information to flow quickly from one part of the factor graph to another.
- **(Good expander)** The resulting factor graph should be a good expander [107], see also the corresponding definition Def. 2.16 for regular graphs. This means that for some $\alpha > 0$ and some $\varepsilon > 0$, if one starts from any subset $\mathcal{A}$ of the symbol nodes whose size satisfies $|\mathcal{A}| \leq \alpha \cdot n$, then the set $\partial\mathcal{A}$ of function nodes connected to the vertices of $\mathcal{A}$ should satisfy

  $$|\partial\mathcal{A}| \geq \varepsilon|\mathcal{A}|,$$

  where ε is the 'expansion factor' (which should preferable be large). We require a similar property when starting from subsets of function nodes.
- **(Density evolution)** Density evolution on infinite factor trees is nowadays well understood. Results from density evolution can partially be carried over to loopy factor graphs of medium to long size codes. The factor graph should therefore have locally a structure that is similar to the one suggested by density evolution, see e.g. [95] where degree sequences are given (Def. 2.4).
- **(Minimum distance)** One lesson that has been learned from iterative decoding compared to classical coding and decoding theory is that the minimum distance is not the one and only truth about a code. In the waterfall region the performance of the code depends mostly on the structure of the factor graph (density evolution) and is nearly independent of the minimum distance. But a low minimum distance leads to an error floor at high signal-to-noise ratios. Therefore, it is still desirable that the code has a decent minimum distance.
- **(Stopping sets)** When codewords from a given code are sent over the binary erasure channel, stopping sets of the code determine the iterative decoding performance [23]. It is therefore desirable that a code has no small stopping sets. (How far these stopping sets determine the performance of a code that is sent e.g. over a BI-AWGNC remains to be seen.)
- **(Pseudo-codewords)** Pseudo-codewords were defined in by Wiberg in [128] and later work on the topic includes [32, 36]. There is strong evidence [56] that pseudo-codewords play the crucial role for understanding iterative decoding. (Note that each stopping set is also a pseudo-codeword [56].) Obviously, one should try to maximize the minimum pseudo-weight; we leave this problem open for future research.
- **(Encoding complexity)** The encoding complexity of a code should be low.
- **(Description size, code length, code rate)** It is desirable to have families of codes whose members can be compactly described. Moreover, a family of codes should allow to construct codes of various lengths and rates.

The listed items are of course somehow conflicting. Low encoding complexity can be achieved if the block length is chosen small but this will not result in a large minimum distance nor in a large girth. Upper bounds for the minimum distance in terms of the rate and block length are well known.

5.1.3 Outline of this Chapter

The codes that will be treated in the main part of this chapter have been classified according to regular SS-LDPCCs, irregular SS-LDPCCs, regular WS-LDPCCs, irregular WS-LDPCCs, LDPCCS with state symbols, and turbo codes. (Of course, we could have taken a different approach by classifying according to the underlying graph.) After showing some simulation results we will discuss lower bounds on the minimum distance and different encoding possibilities. The proofs are given at the end of the chapter. Note that we discuss only binary codes; constructions of codes over larger alphabets (see e.g. [70]) are possible along the same lines but additionally to defining the position of the nonzero entries of the parity-check matrix one has to define the value of the nonzero entries.

5.2 Construction of Regular SS-LDPCCs

5.2.1 The Construction by Margulis

Already in 1982, Margulis [82] constructed $(3,6)$-regular LDPCCs. His construction was based on the graphs that he introduced at the beginning of the same paper, but his approach is not at all limited to these graphs; indeed, as long as the graph is a Cayley graph, his approach can be used. His construction guarantees that every walk in the factor graph can be mapped back to a walk in the underlying graph, especially every cycle gets mapped back to a cycle. Therefore, the girth of the factor graph of the code is not smaller than the girth of the underlying graph. The guidelines in Sec. 5.1.2 suggest that the girth of the factor graph should be large; this can be guaranteed by starting with an underlying graph having large girth.

Construction 5.2 (Margulis's construction idea) Margulis [82] proposed the following (designed) rate-1/2-construction of a $(d/2, d)$–regular binary SS-LDPCC based on an d–regular undirected Cayley $\mathsf{X}(\mathcal{G},\mathcal{S})$ (and its directed Cayley graph version $\vec{\mathsf{X}}(\mathcal{G},\mathcal{S})$) with group $\mathcal{G}$ and symmetric set $\mathcal{S} \subseteq \mathcal{G}$ as in Fig. 5.1 (left), where d is an even positive integer. Therefore, $|\mathcal{S}| = d$ and we let $\mathcal{S} \triangleq \{s_1, s_1^{-1}, s_2, s_2^{-1}, \ldots, s_{d/2}, s_{d/2}^{-1}\}$. We now split this set into two disjoint sets according to

$$\mathcal{S} \triangleq \mathcal{S}' \cup \mathcal{S}'' \quad \text{with} \quad \begin{cases} \mathcal{S}' \triangleq \{s_1, s_2, \ldots, s_{d/2}\} \\ \mathcal{S}'' \triangleq \{s_1^{-1}, s_2^{-1}, \ldots, s_{d/2}^{-1}\} \end{cases} .$$

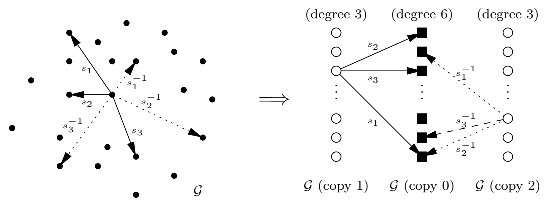

Figure 5.1: Illustration of Constr. 5.2 for $d = 6$. Left: Directed Cayley graph $\vec{\mathsf{X}}(\mathcal{G},\mathcal{S})$ with $\mathcal{S} = \{s_1, s_2, s_3\} \cup \{s_1^{-1}, s_2^{-1}, s_3^{-1}\}$. Right: Factor graph of a $(3,6)$–regular LDPCC before replacing directed edges by undirected edges (see also Footnote 1).

The (partial) factor graph representing the LDPCC is constructed as follows:

- Draw three copies of the vertices of $\mathsf{X}(\mathcal{G},\mathcal{S})$ as in Fig. 5.1 (right): the vertices of two copies are bit nodes, the vertices of one copy are f_{XOR} function nodes.
- Each bit node on the left-hand side is connected by a directed edge to an f_{XOR} function node using a generator from the set $\mathcal{S}'$.
- Each bit node on the right-hand side is connected by a directed edge to an f_{XOR} function node using a generator from the set $\mathcal{S}''$.
- Produce undirected edges by removing the arrows from the directed edges.[1]

$\square$

Properties 5.3 (of Constr. 5.2) *Some properties of the factor graphs obtained in Constr. 5.2.*

- *If the Cayley graph $\mathsf{X}(\mathcal{G},\mathcal{S})$ is bipartite then the factor graph consists of two isomorphic components. For coding purposes we can take either one.*

[1] A different approach that would be similar to the transition from a directed to an undirected Cayley graph as in Ex. 2.24 is the following: to each directed edge we draw a directed edge in the reverse direction associated to the inverse generator; then we replace each pair of directed edges by a single undirected edge.

- *The factor graph has $2 \cdot |\mathcal{G}|$ bit nodes and $|\mathcal{G}|$ check nodes. If the underlying graph is bipartite (see first property), then the factor graph has two isomorphic components with $|\mathcal{G}|$ bit nodes and $|\mathcal{G}|/2$ check nodes each.*

- *From the last remark it follows that the resulting code has designed rate $1/2$. The rate is higher if the resulting parity-check matrix has not full rank.*

- *The factor graph corresponds to a $(d/2, d)$-regular LDPC code.*

- *If we fix some starting vertex in the underlying undirected Cayley graph then we can map back any walk in the factor graph. Especially, every cycle gets mapped to a cycle in the underlying graph. Therefore, the girth of the factor graph is at least as large as the girth of the underlying graph.*

Proof: The statements are obvious from the construction. □

Example 5.4 (Margulis's construction) In his paper [82], Margulis essentially gave the above construction via an example. He constructed a $(3, 6)$-regular SS-LDPCC where his underlying graph was a modification of the Cayley graph described at the beginning of the same paper. □

Example 5.5 (Margulis's construction with $\mathsf{X}^{\mathrm{LPS}}_{q,p}$ graphs) In [98, 99] we presented a $(3, 6)$-regular SS-LDPCC where the underlying graph is a Ramanujan graph as proposed by Lubotzky, Phillips, and Sarnak (see Sec. 4.3)[2]. From the graphs $\mathsf{X}^{\mathrm{LPS}}_{q,p}$, where q and p are some distinct odd primes, we get

- If $\left(\frac{p}{q}\right) = -1$, the underlying graph is bipartite. We get two isomorphic $((p + 1)/2, p + 1)$-regular SS-LDPCCs of length $q(q^2 - 1)$ and designed rate $1/2$.

- If $\left(\frac{p}{q}\right) = +1$, the underlying graph is not bipartite. The resulting code is a $((p + 1)/2, p + 1)$-regular SS-LDPCC of length $q(q^2 - 1)$ and designed rate $1/2$. (Note that compared to the former case, the factor graph here has only one component but the underlying group is only half as large.)

[2] After the publication of our papers, we realized that Margulis himself had proposed such a construction in [83].

Taking e.g. $q = 17$, $p = 5$, we get a $(3, 6)$-regular SS-LDPCC of length 4896 (note that $\left(\frac{5}{17}\right) = -1$). The girth guaranteed by the underlying graph is 8 (see Tab. F.1), but it turns out that the factor graph has girth 12, which is almost the optimal girth for a $(3, 6)$-regular SS-LDPCC of that length (the optimal would girth be at most 14)[3]. The resulting parity-check matrix has 2448 rows and 4896 columns, but the rank is 2422 (i.e. the rank loss is 26), leading to a rate $(4896 - 2422)/4896 = 0.5053$, which is roughly 1% above the designed rate $1/2$. For more comments on this construction, we refer the reader to Sec. 5.8.2. □

Lemma 5.6 (Girth upper bound) *This lemma provides an upper bound on the girth for regular SS-LDPCCs. So, consider a $(w_{\mathrm{col}}, w_{\mathrm{row}})$-regular SS-LDPCC of length n having a parity-check matrix with m rows such that $n = m w_{\mathrm{row}}/w_{\mathrm{col}}$. Let $\alpha \triangleq (w_{\mathrm{col}} - 1)(w_{\mathrm{row}} - 1)$. In the case $g = 2 \pmod 4$ one has necessarily the inequality*

$$g \leq 4 \log_\alpha(m) + 2 = 4 \log_\alpha(n) + 4 \log_\alpha\left(\frac{w_{\mathrm{col}}}{w_{\mathrm{row}}}\right) + 2,$$

whereas in the case $g = 0 \pmod 4$ one has necessarily the inequality

$$g \leq 4 \log_\alpha(m) - 4 \log_\alpha(w_{\mathrm{col}}) + 4 = 4 \log_\alpha(n) - 4 \log_\alpha(w_{\mathrm{row}}) + 4.$$

Proof: See Sec. 5.11.1. □

Remark 5.7 (Generalizations) There are the following obvious generalizations of Constr. 5.2.

- The set $\mathcal{S}$ can be split in any desired way into two disjoint sets $\mathcal{S}'$ and $\mathcal{S}''$. If $|\mathcal{S}'| = |\mathcal{S}''|$, the resulting code is a $(|\mathcal{S}|/2, |\mathcal{S}|)$–regular SS-LDPCC. If the sets $\mathcal{S}'$ and $\mathcal{S}''$ are of unequal size, then half the bit nodes will have degree $|\mathcal{S}'|$ and half the bit nodes will have degree $|\mathcal{S}''|$ whereas all check nodes will have degree $|\mathcal{S}|$.

- Irregular SS-LDPCCs can also be constructed by modifying Margulis's approach, see Sec. 5.3.

□

[3] If the girth were 16 we could start at a check node and all the vertices reached after 7 steps would have to be different. In this way we could predict that there are $6 + 60 + 600 + 6000$ different symbol vertices when there are actually only 4896. So the girth cannot be more than 14 for these parameters.

5.2.2 LDPCCs from Finite Incidence Structures

We show how finite incidence structures can be used to derive regular SS-LDPCCs and other codes.

Construction 5.8 (Deriving a factor graph from an incidence structure) There are many ways to get a binary code from an incidence structure; we want to focus on three of them leading to code classes $\mathcal{C}^{(1)}$, $\mathcal{C}^{(2)}$, and $\mathcal{C}^{(3)}$. For all three classes we first replace the incidence structure (having points and lines) by its point-line graph (see Def. 4.13) as shown on the left-hand side of Fig. 5.2.

- **(Class $\mathcal{C}^{(1)}$)** Factor graphs of codes in class $\mathcal{C}^{(1)}$ (see Fig. 5.2) are derived by replacing point and line vertices in the point-line graph by bit nodes and f_{XOR} function nodes, respectively. (The drawing of point-line graphs was chosen such that pictorially there is no difference between the point-line graph and the factor graph of a code from class $\mathcal{C}^{(1)}$.) We will discuss such codes in this section.

  It is easy to see that the parity-check matrix of a code from $\mathcal{C}^{(1)}$ must have the structure shown in Fig. 5.4, where we identified a bit with the point where it came from and a check with the line where it came from. Note that the girth and diameter of the factor graph are the same as the girth and diameter of the point-line graph, respectively.

  Without going into the details we note that one can also obtain a code from class $\mathcal{C}^{(1)}$ by starting with the dual of an incidence structure (see Def. 4.16); the columns of the parity-check matrix then correspond to the lines and the rows of the parity-check matrix to the points.

- **(Class $\mathcal{C}^{(2)}$)** Factor graphs of codes in class $\mathcal{C}^{(2)}$ (see Fig. 5.2) are derived by replacing point and line vertices in the point-line graph by subcode function nodes and by replacing edges in the point-line graph by bit nodes (of degree two) with two attached edges. If the subcode function nodes are all f_{XOR} function nodes then the resulting code is known in the coding literature as a cycle code; as the minimum distance is given by the length of the shortest cycle divided by two (i.e. the girth divided by two), these codes do not have a good minimum distance. One therefore usually takes indicator function nodes of more complex subcodes to get a better minimum distance. Such codes are discussed in Sec. 5.4.

- **(Class $\mathcal{C}^{(3)}$)** Factor graphs of codes in class $\mathcal{C}^{(3)}$ are basically derived as codes in $\mathcal{C}^{(1)}$ but with a pre-processing step. As shown in Fig. 5.3, the following procedure is applied to each point in the point-line incidence structure: a point is replaced by multiple points and the lines through the original point are divided up (randomly or according to some rule) among the new points. (This construction was e.g. proposed in [57].) Similarly, one can split a line into multiple lines. But most of the underlying structure gets destroyed by this approach; we do not consider this method any further in this chapter.

□

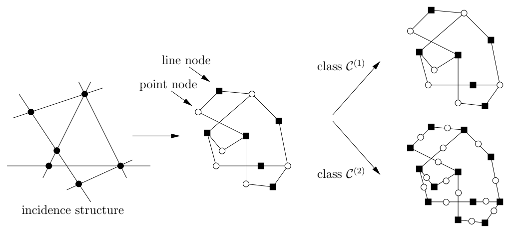

Figure 5.2: From an incidence structure to a binary code (classes $\mathcal{C}^{(1)}$ and $\mathcal{C}^{(2)}$).

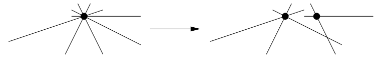

Figure 5.3: From an incidence structure to a binary code (first step for class $\mathcal{C}^{(3)}$).

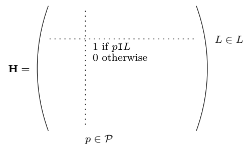

Figure 5.4: Form of a parity-check matrix of a code of class $\mathcal{C}^{(1)}$.

Example 5.9 (From an FGT to a factor graph) Assume to have an FGT of order (s,t). Then the codes in class $\mathcal{C}^{(1)}$ derived from that FGT will have bit nodes of degree $t+1$, checks nodes of degree $s+1$, and a code length of $st+s+1$, see Lemma 4.24.

Projective planes are an example of FGTs where $s=t=q$ for some prime power q (see Ex. 4.21). The resulting codes were e.g. studied in [61]. They are also known as difference-set cyclic codes [76] or one-step majority logic decodable codes because they are decodable using the one-step majority logic decoding algorithm [61]. Recently, they have been rediscovered to be suitable for iterative decoding [67, 57]. But already in 1991, Karplus and Krit [54] together with Tanner had considered these codes for iterative decoding, see also the related codes in [87]. For $q=2^m$, $m \geq 1$, the rate is

$$R = 1 - \frac{3^m+1}{2^{2m}+2^m+1},$$

and the minimum distance is $d_{\min} = q+2 = 2^m+2$. For $q \to \infty$ (q a power of 2) the rate tends to 1(!) Interestingly, it turns out that these codes are cyclic codes. (For more about the minimum distance, see also Th. 5.32.)

A drawback of this class is that the degree of the bit and check nodes grows roughly proportionally to the square root of the block length. So these codes are to be considered practical only up to a block length of roughly 1000. Apparently, the updates of the messages during decoding have to be done with a high-enough precision to get the most out of the medium-length to long codes [67]. □

Example 5.10 (From an FGQ to a factor graph) Assume to have an FGQ of order (s,t). Then the code in the class $\mathcal{C}^{(1)}$ derived from that FGQ will have bit nodes of degree $t+1$, checks nodes of degree $s+1$, a code length of $(s+1)(st+1)$, and a parity-check matrix of size $(t+1)(st+1) \times (s+1)(st+1)$. (This follows from Lemma 4.24.) Tab. F.2 gives an overview over different FGQ constructions, Tabs. F.12 and F.13 give the general minimum distance lower bounds according to Sec. 5.9, whereas Tabs. F.3, F.4, F.5, F.6, F.9, F.10, F.11 list various codes derived from specific FGQ constructions along with their parameters.

Special cases of such codes derived from FGQs have been considered by Bagchi, Sastry, Brouwer, and others some ten years ago (see, e.g. [8]).

Whereas they used traditional decoding methods, in [125] we investigated the suitability of these codes for iterative decoding.

E.g. if the underlying incidence structure is the FGQ $W(q)$ (see Ex. 4.26), then $s=q$, $t=q$ for some prime power q and the length of the code is $n=(q+1)(q^2+1)=q^3+q^2+q+1$. (For $q=2$ the resulting factor graph is shown in Fig. 4.10 (left and right), as it is pictorially equivalent to the point-line graph.) Following [7], the rate of the code for q odd is[4]

$$R = \frac{1}{2} \cdot \frac{q}{q+1},$$

so in the limit $q \to \infty$ it is $1/2$, whereas for $q=2^m$ [102]

$$R = 1 - \frac{\left(1 + \left(\frac{1+\sqrt{17}}{2}\right)^{2m} + \left(\frac{1-\sqrt{17}}{2}\right)^{2m}\right)}{(2^m+1)\cdot(2^{2m}+1)},$$

and so for $q \to \infty$ the rate tends to 1(!)[5] So these code are represented by parity-check matrices that are highly redundant (in the sense that a subset of the rows would be sufficient to define the code). For minimum distance lower bounds, see Th. 5.34. □

Remark 5.11 (Redundancy) As can be seen from the above results, most of the codes derived from FGPs are special in that they have highly redundant parity-check matrices, i.e. a large fraction of the rows of the parity-check matrices are linearly dependent. Note that when generating an SS-LDPCC randomly, the probability of a random LDPC code having a highly redundant parity-check matrix is vanishingly small.

For analog (iterative) decoding [64] it has been recognized that codes described by highly redundant parity-check equations can lead to fault tolerance, and one can even use cheaper/smaller transistors (also with quadratic instead of exponential characteristic) with minor degradations in performance [68]. For these reasons, the redundancy of equations defined by FGPs could prove a valuable attribute. □

[4] This result holds more generally for all FGQs where all points on some line L are regular, see [7].

[5] This is similar to the SS-LDPCCs derived from finite projective geometries with $q=2^m$.

λ_2	λ_3	λ_4	λ_{11}	ρ_7	ρ_8
0.23882	0.29515	0.03261	0.43342	0.43011	0.56989

$\tilde{\lambda}_2$	$\tilde{\lambda}_3$	$\tilde{\lambda}_4$	$\tilde{\lambda}_{11}$	$\tilde{\rho}_7$	$\tilde{\rho}_8$
0.4500	0.3708	0.0307	0.1485	0.4631	0.5369

Table 5.1: Optimized bit and check degree sequences for maximal bit degree 11 and designed rate 1/2 taken from Table 1 in [95]. λ_i and ρ_i are the edge-oriented proportions whereas $\tilde{\lambda}_i$ and $\tilde{\rho}_i$ are the vertex-oriented proportions.

5.3 Construction of Irregular SS-LDPCCs

In the second half of the last decade, it was recognized that irregular SS-LDPCCs with a sensitive choice of bit and check degrees can improve the performance quite markedly (see, e.g., [66, 95]); the tool of density evolution for calculating performance thresholds proved to be very valuable. There are two possibilities to describe ensembles of irregular LDPCCs, either using edge-oriented degree sequences or vertex-oriented degree sequences, see Def. 2.4. Whereas $\lambda(x)$ and $\rho(x)$ are more convenient for the density evolution analysis, the coefficients of $\tilde{\lambda}(x)$ and $\tilde{\rho}(x)$ are what we need for our code design purposes. Tab. 5.1 shows optimized degree sequences e.g. for irregular SS-LDPCCs with maximal bit degree 11 and designed rate 1/2 as presented in [95].

In this section we present a construction of irregular SS-LDPCCs where we aim at combining the following ideas:

- LDPCC construction by Margulis (1982), as shown in Constr. 5.2.
- LPS Ramanujan graphs, see Sec. 4.3.
- Incorporating irregular degree sequences as proposed by Richardson, Shokrollahi, and Urbanke [95], see also Def. 2.4.
- Easy encodability as proposed by MacKay, Wilson, and Davey (see e.g. [75] or [97]). Interestingly, there is a trade-off between easy encodability and good minimum distance, see e.g. [24].
- The codes should have a simple description (see e.g. the codes given in the construction by Margulis in Constr. 5.2 or the QCRA LDPCCs by Tanner [112]).

We discuss now the construction of irregular SS-LDPCCs as proposed by us in [99]. The basic idea is to start as in Constr. 5.2 but to tweak the construction so as to get irregular degrees.

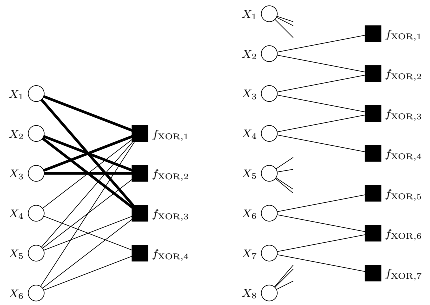

Figure 5.5: Left: partial factor graph of an irregular SS-LDPCC with a short cycle involving only bit nodes of degree two. (Note the highlighted edges.) Right: possible solution to avoid short cycles involving only bit nodes of degree two.

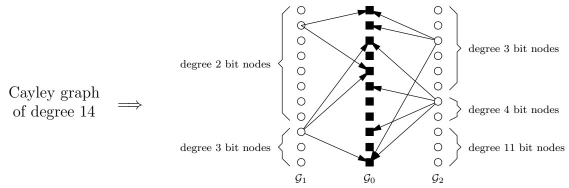

Figure 5.6: From a Cayley graph of degree 14 to an irregular SS-LDPCC with maximal bit degree 11.

Example 5.12 (Irregular SS-LDPCC) We describe a specific construction, namely of a rate-1/2 code of length 4896 with maximum bit degree 11. It is not too difficult to derive other codes following the same principles. Tab. 5.1 lists the optimized bit and check degree polynomials $\lambda(x)$ and $\rho(x)$ for a maximal bit degree of 11 as given in [95].[6] The same

[6]These degree sequences were optimized for infinite trees. As our code size is large enough, these degree sequences are a good starting point.

table lists also the polynomials $\tilde{\lambda}(x)$ and $\tilde{\rho}(x)$ which give the relevant proportions for our construction. Note that $\tilde{\lambda}_2 = 0.45$, i.e., nearly half the bit nodes have degree 2. One needs to take special care of this fact because of the following reason.

Fig. 5.5 (left) shows a (partial) factor graph of an irregular SS-LDPCC with three bit nodes having degree two. There is a cycle, namely

$$X_1 — f_{\mathrm{XOR},1} — X_3 — f_{\mathrm{XOR},2} — X_2 — f_{\mathrm{XOR},3} — X_1$$

(highlighted in Fig. 5.5 (left)), where the involved bit nodes have only degree two. It is not difficult to see that setting these bits to one and the other bits to zero is a configuration that fulfills all checks, i.e., it is a codeword. We learn that a cycle of length L involving only bit nodes of degree two leads to a codeword of weight $L/2$. To avoid low-weight codewords, our goal should therefore be to either avoid at all such cycles or to allow such cycles only if they have large length.

It is indeed possible to avoid such cycles as long as $\tilde{\lambda}_2 < 0.50$ by assigning the edges of the degree–2 bit nodes in a fashion similar to the one in Fig. 5.5 (right), which in parity-check matrix notation corresponds to a submatrix of the form

$$\begin{pmatrix} 1 & & & & \\ 1 & 1 & & & \\ & 1 & 1 & & \\ & & 1 & 1 & \\ & & & \ddots & \ddots \end{pmatrix}. \tag{5.1}$$

To construct the irregular SS-LDPCC we proceed as follows.

- Select a Cayley graph as base object. Here we take $\mathsf{X}^{q,p}$ with $q = 17$ and $p = 13$ (see also Tab. F.1): the underlying group is $\mathrm{PSL}_2(\mathbb{F}_q)$ (because $\left(\frac{p}{q}\right) = +1$) and $|\mathcal{S}| = 13+1 = 14$ which we partition into a set $\mathcal{S}'$ of size 3 and a set $\mathcal{S}''$ of size 11.
- We draw the bit and check nodes as shown in Fig. 5.6, where we select $\mathcal{G}_0 = \mathcal{G}_1 = \mathcal{G}_2 = \mathrm{PSL}_2(\mathbb{F}_q)$.
- First we construct the right part. For each vertex in $\mathcal{G}_2$ we have to select the subset of generators of $\mathcal{S}''$ which connect this vertex to the vertices in $\mathcal{G}_0$. The sizes of each of these subsets is given by the desired bit node degrees.

  The degrees of the check nodes (which according to Table 5.1 should all be nearly equal) can be controlled by choosing the above subsets in a suitable way; a simple solution is to take a greedy algorithm that always selects the subsets which minimizes the variance of the current check-node degrees during the build-up process.
- Secondly, we construct the left part. It essentially consists of finding chains of edges where the bits have degree two as alluded to in Fig. 5.6 (right). For this we can use the generators in the set $\mathcal{S}'$ but also generators of the set $\mathcal{S}''$ if there will be no two (directed) edges with the same generator incident in the same check node. Finally, one can choose the generators of the bit nodes having degree three. Here one possibility is to choose generators that minimize the variance of the check-node degrees.
- Finally we get undirected edges by removing the arrows of the directed edges. (For an alternative approach, see Footnote 1 on page 91.)

For the same reason as in the original Margulis construction (see Constr. 5.2), this construction guarantees that the girth of the factor graph is not less than the girth of the underlying (undirected) Cayley graph. □

Remark 5.13 (Variations) There are several variations to the above example which we would like to discuss.

- **(Description complexity)** For describing the resulting code, one needs to store the generators used for every bit node. The amount of data is not much less than what one has to save for a randomly generated parity-check matrix. To reduce the description complexity one can do the following: one puts the bit and check nodes into different partitions according to some subgroup and its cosets (suggestions for suitable subgroups of $\mathrm{PSL}_2(\mathbb{F}_q)$ are given in Sec. A.4.5, but any other subgroup works equally well). The generators are then not anymore chosen for each bit node separately, but the same generators are taken for all bit nodes in a coset. The crucial observation is this: let $\mathcal{H}$ be a subgroup of $\mathcal{G}$ and let $\mathcal{H} \cdot g$ be a coset of $\mathcal{H}$ for some $g \in \mathcal{G}$. If $s \in \mathcal{S}$ is some generator then all

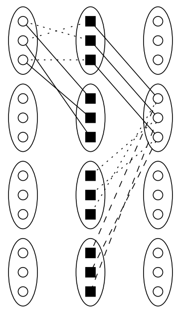

Figure 5.7: If the same generator is used for all bit nodes in a certain coset then the bit nodes in that coset will be connected to check nodes that are all in some coset. (Note that the lines of equal type correspond to equal generators.)

bit nodes in the coset $\mathcal{H} \cdot g$ will be connected to some check node in the set

$$(\mathcal{H} \cdot g) \cdot s = \mathcal{H} \cdot (g \cdot s) = \mathcal{H} \cdot g',$$

for some $g' \in \mathcal{G}$. But this set is again a coset! Therefore bit nodes in a coset get connected to check nodes that are all in the same coset. (Fig. 5.7 tries to give an idea what happens when the coset size is three.) We get the following trade-off: the less cosets one has, the smaller is the description complexity but also the freedom of choice of selecting generators that fulfill all the requirements gets smaller. A reasonable choice in Ex. 5.12 was to take the subgroup and its cosets according to the first subgroup proposed in Sec. A.4.5. So each copy $\mathcal{G}_0$, $\mathcal{G}_1$, and $\mathcal{G}_2$ gets partitioned into $17 + 1 = 18$ cosets. We therefore just have to save the generators for $2 \cdot 18 = 36$ bit node cosets. This is of course much less than for a randomly chosen irregular LDPCC.

- **(Structure of the parity-check matrix)** Doing the selection of generators according to cosets implies that the parity-check matrix

has a structure like

$$\mathbf{H} = \left(\begin{array}{ccccc|ccccc} \mathbf{P} & - & - & - & - & \mathbf{P} & \mathbf{P} & \mathbf{P} & \mathbf{P} & \mathbf{P} \\ \mathbf{P} & \mathbf{P} & - & - & \mathbf{P} & - & \mathbf{P} & \mathbf{P} & - & \mathbf{P} \\ - & \mathbf{P} & \mathbf{P} & - & \mathbf{P} & - & - & - & \mathbf{P} & \mathbf{P} \\ - & - & \mathbf{P} & \mathbf{P} & - & \mathbf{P} & - & \mathbf{P} & \mathbf{P} & \mathbf{P} \\ - & - & - & \mathbf{P} & \mathbf{P} & \mathbf{P} & \mathbf{P} & \mathbf{P} & \mathbf{P} & - \end{array} \right). \tag{5.2}$$

Here $\mathbf{P}$ represents (possibly different) permutation matrices[7] (or possibly the sum of several permutation matrices). The special structure of the left-hand side of the parity-check matrix in (5.2) stems from the avoidance of small cycles involving only bits of degree two, see also (5.1).

Notice also the similarity in the form of this parity-check matrix to other constructions, e.g. in [115, 112] or in Constr. 5.23 where the parity-check matrices are built up of cyclically shifted identity matrices (which are permutation matrices).

- **(Encoding complexity)** As alluded to in (5.2), the left-hand side of the graph and therefore the left-hand side of the parity-check matrix may be chosen so as to get near-upper triangular form. Such a structure leads to a reduced encoding complexity compared to a code with a non-structured parity-check matrix, see also the approaches in [75]. There is of course a trade-off, as such a structure may potentially also cause codewords of smaller weight. For more on encoding, see also the comments in Sec. 5.10.

- **(Several Cayley graphs)** In Sec. 4.3.3 we presented a construction where two LPS Ramanujan graphs were combined and we could still say something about the girth. Actually, the graph presented in Fig. 4.6 can directly be turned into a factor graph by defining the left-hand side and the right-hand side vertices to be bit nodes and the middle vertices to be check nodes. Additionally, one has the option of using only a subset of the generators. It is basically also possible to take more than two Cayley graphs in order to obtain a larger graph: taking such a graph as a starting point one can construct regular and irregular SS-LDPCCs.

- **(Code length and code rate)** Fix some subgroup of the underlying Cayley graph group. When one eliminates all bit nodes of one

[7]A permutation matrix is by definition a square matrix with entries “0” and “1” having exactly one “1” per row and one “1” per column.

or several cosets and/or all check nodes of one or several cosets, one can get different code lengths and/or different code rates. It may happen though that one is slightly limited in the choice of generators that connect bit nodes that have not been removed and check nodes that have not been removed.

□

5.4 Construction of Regular WS-LDPCCs

An often used method for constructing regular WS-LDPCCs was already proposed by Tanner [109].

Construction 5.14 (From a regular graph to a regular WS-LDPCC) Take any d_0-regular graph and fix a subcode of length $n_1 = d_0$. The factor graph is obtained by the following steps (see also Fig. 5.9).

- Every vertex is replaced by the indicator function of the same subcode.
- Every edge is replaced by a bit node (of degree 2) and two edges attached to it.

The remaining degrees of freedom are how the subcodes are placed, i.e., what subcode bit is associated with what (outgoing) edge. Note that to avoid a small minimum distance, the guidelines in Constr. 5.8 should be observed.

Such codes can be decoded using the standard sum-product algorithm; for performing the message update rules for the subcode indicator functions, one of the algorithms discussed in App. D may be used. □

Properties 5.15 *Assume that the underlying d_0-regular graph has n_0 vertices and girth g_0, and that the subcode has length $n_1 = d_0$ and rate R_1. The factor graphs constructed in Constr. 5.14 have*

- *$n \triangleq d_0 \cdot n_0/2$ bit nodes,*
- *$m \triangleq n_0$ check nodes,*
- *rate $R \geq 1 - 2 \cdot (1 - R_1)$,*

- *girth $g = 2g_0$.*

Proof: See Sec. 5.11.2. □

Remark 5.16 (Subcodes) Given a degree d_0, there are usually not that many choices of different non-isomorphic subcodes of length $n_1 = d_0$. To get an overall rate that is not too small, the subcodes must be of relatively high rate, e.g. for an overall designed rate of 1/2, the subcodes must be of rate 3/4.

For $d_0 = 14$, we e.g. have the following subcodes.

- A $[14, 13, 2]$ subcode (which is a simple parity-check subcode) with parity-check matrix
$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$
- A $[14, 12, 2]$ subcode with parity-check matrix (note that there are also different possibilities)
$$H = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$
- A $[14, 11, 2]$ subcode (which can be seen as a punctured $[15, 11, 3]$ binary Hamming code) with parity-check matrix (note that there are also different possibilities)
$$H = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}.$$
- A $[14, 10, 3]$ subcode (which can be considered as a shortened binary $[15, 11, 3]$ Hamming code) with parity-check matrix (note that there are also different possibilities)
$$H = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}.$$

□

Example 5.17 (WS-LDPCCs by Lafferty and Rockmore) Lafferty and Rockmore presented in [59] codes based on Constr. 5.14. As underlying graph they took a $\mathsf{X}_{q,p}^{\mathrm{LPS}}$ LPS Ramanujan graph (see Sec. 4.3) where the uniform degree is $d_0 = p + 1$. More specifically, they used the examples $q = 17$, $p = 13$ and $q = 29$, $p = 13$. Because $\left(\frac{13}{17}\right) = +1$ and $\left(\frac{13}{29}\right) = +1$, in the first case the underlying graph has $|\operatorname{PSL}_2(17)| = 2448$ vertices of degree 14, whereas in the second case the underlying graph has $|\operatorname{PSL}_2(29)| = 12180$ vertices of degree 14. The resulting codes have a block length of 17136 and 85260, respectively. Taking all subcodes to be $[14, 11, 2]$ codes, the overall designed rate is $4/7$, whereas taking all subcodes to be $[14, 10, 3]$ codes, the overall designed rate is $3/7$. To get overall designed rate $1/2$, Lafferty and Rockmore chose half the subcodes to be $[14, 11, 2]$ codes and half of the subcodes to be $[14, 10, 3]$ codes. The subcodes were placed randomly. Decoding performance results using the standard sum-product algorithm are shown in [59]. □

5.5 Construction of Irregular WS-LDPCCs

Constr. 5.14 can in a very natural way be generalized to a construction producing irregular WS-LDPCCs in the sense that one takes different types of subcodes instead of just taking only one type of subcode. This was actually already done in Ex. 5.17, where the main motivation was to get a rate-$1/2$ code. But one can go much further as was shown in [123]: by optimizing the percentages of the different subcodes being used and also having bit nodes of different degrees, it is possible to get code ensembles which have a better threshold. This is very much in the spirit of the transition from regular to irregular SS-LDPCCs: there the degree sequences are varied in order to optimize the density evolution and to get better thresholds.

A second motivation for the code construction method presented in this section was the fact that most graph families constructed by mathematicians are regular (i.e. the degrees of all vertices are the same) or bi-regular. Instead of changing the graph by extending/pruning it like in Ex. 5.12, we want to preserve the underlying graph as far as possible.

Construction 5.18 (Irregular WS-LDPCCs from replacements)

- We start with a bipartite graph of regular degree d_0.

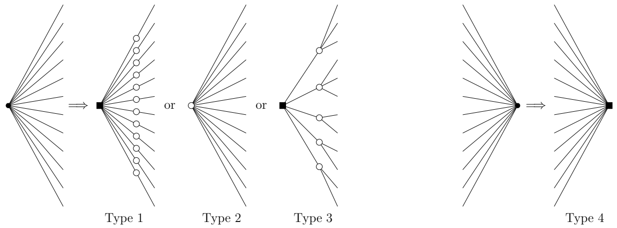

Figure 5.8: Construction of WS-LDPCCs by replacement. Left: different types of replacements for a vertex (and its adjacent edges) on the left-hand side of a graph. Right: replacement for a vertex on the right-hand side of a graph.

- Each vertex on the left-hand side (plus its adjacent outgoing edges) is replaced by a factor graph of type 1, 2, or 3 or any similar replacement, see Fig. 5.8 (left) for $d_0 = 12$.
- Each vertex on the right-hand side (plus its adjacent outgoing edges) is replaced by a factor graph of type 4 or any similar replacement, see Fig. 5.8 (right) for $d_0 = 12$.
- To each function node we have to select and place a subcode.

This type of construction can be extended to any bipartite or non-bipartite graph. The condition of regular degree is not crucial at all. □

We illustrate the above construction by two examples.

Example 5.19 (Irregular WS-LDPCC) We start with a bipartite graph of regular degree d_0, see Fig. 5.9 (left). For drawing purposes we choose degree 3 on both sides but in practice one usually takes graphs with larger degrees, e.g. $d_0 = 12$. By using only replacements of type 1 on the left-hand side and of type 4 on the right-hand side, we arrive at the factor graph in Fig. 5.9 (right). The function nodes can basically be any indicator function of any subcode, but one should avoid the following pitfalls.

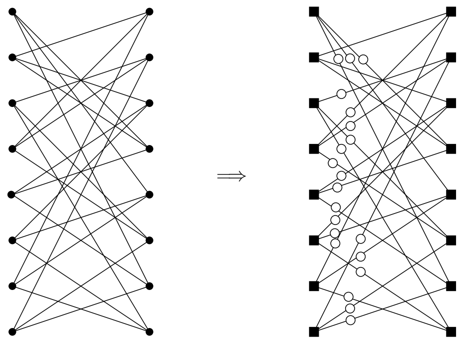

Figure 5.9: Construction of a (regular/irregular) WS-LDPCC by replacements (see Ex. 5.19 for details).

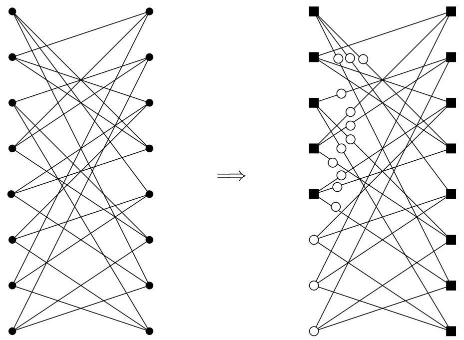

Figure 5.10: Construction of an irregular WS-LDPCC by replacements (see Ex. 5.20 for details).

- If all subcodes are simple parity-check subcodes then one runs into the problem of low-weight codewords already mentioned in Constr. 5.8.
- The same problem can occur when using subcodes that have minimum distance 2.

One possibility to circumvent this nuisance is by taking e.g. only subcodes with minimum distance at least 3 for either the left-hand or the right-hand side. □

Example 5.20 (Irregular WS-LDPCC with different bit node degrees) We start with a bipartite graph of regular degree d_0, see Fig. 5.10 (left). For drawing purposes we choose degree 3 on both sides but in practice one usually takes graphs with larger degrees, e.g. $d_0 = 12$. Besides replacements of type 1, we now also allow replacements of type 2 on the left-hand side: typically a factor graph as in Fig. 5.10 (right) results. The same issues concerning the avoidance of low-weight codewords as in Ex. 5.19 apply here; choosing e.g. the subcodes on the left-hand side to have minimum distance at least 3 basically solves the problem.

□

Remark 5.21 (Issues concerning Constr. 5.18)

- **(Placing subcodes)** Constr. 5.18 can in principle be applied to any bipartite graph, be it randomly generated or algebraically constructed. But if e.g. the graph is a Cayley graph then one can do the labeling of the bit positions of the subcodes according to the generators. Potentially, one can systematically exploit the additional knowledge available to avoid the problem of low-weight codewords.
- **(Girth, diameter, expansion)** Clearly, graph parameters like the girth, the diameter, and the expansion constant of the original graph and of the resulting factor graph are tightly related. If e.g. only type-1 replacements are used on the left-hand side, the girth and the diameter are doubled and the expansion constant of the factor graph can be expressed in term of the expansion constant of the original graph. For other constructions based on extending/pruning graphs (as e.g. Ex. 5.12) it is generally only possible to give such results for a subset of these parameters.

Note that there are sizes and degrees for the underlying graph where a Ramanujan graph of girth at least 6 exist, but for the same size and degree it is virtually impossible to randomly construct an underlying graph having girth better than 4. (This is especially the case when the degree is large.)

- **(Optimization, density evolution)** Similarly to the optimization of the degree sequences for SS-LDPCCs one can now optimize the fractions of different subcodes being used and the fraction of high-degree bits. This allows to "shape" the density evolution in a favorable way.

  Density evolution for irregular SS-LDPCCs is relatively simple because the processing needed at bit nodes can be done by convolutions; the same applies to the f_{XOR} function nodes after a certain transformation. For more complex subcodes this simplification is unfortunately not possible anymore: unless one uses a very coarse quantization of the messages, the memory and time requirements are huge (see also the problems with density evolution for turbo codes [94]). One can work around this problem by estimating the densities of the outgoing messages by using the sum-product algorithm on the subcode and simulating the input messages according to the densities of the incoming messages. This is clearly a trade-off between doing exact density evolution and pure simulations.

- **(Decoding)** Decoding is done using the standard sum-product algorithm (SPA, see e.g. [58]) with updating alternatingly the bit and the subcode nodes. Updating messages at a subcode node by the SPA amounts to applying the BCJR algorithm, the one-sweep algorithm, or decoding on the dual code, see App. D.

□

Remark 5.22 (Strengths of Constr. 5.18) One of the strengths of Constr. 5.18 is that for a given underlying graph, a large range of code length and code rates can be achieved: by varying the proportions of the different subcodes being used and the proportion of the high-bit nodes, one can get the desired parameters. Moreover, one can try to improve the threshold of the ensemble as far as the restrictions on length and rate allow. □

5.6 LDPCCs with State Bits

We briefly discuss an example of an LDPCC that has also state (or hidden) bits.[8]

Construction 5.23 (QCRA LDPCCs) Quasi-cyclic repeat-accumulate (QCRA) LDPCCs were studied by Tanner [112] and represent a straight-forward modification of repeat-accumulate codes introduced by McEliece *et al.* A typical QCRA LDPCC has a parity-check matrix of the form

$$\mathbf{H} \triangleq \left(\begin{array}{cc|cccc} \mathbb{1}^{(a_1)} & — & \mathbb{1}^{(0)} & — & — & \mathbb{1}^{(1)} \\ \mathbb{1}^{(a_2)} & \mathbb{1}^{(a_4)} & \mathbb{1}^{(1)} & \mathbb{1}^{(0)} & — & — \\ — & \mathbb{1}^{(a_5)} & — & \mathbb{1}^{(1)} & \mathbb{1}^{(0)} & — \\ \mathbb{1}^{(a_3)} & \mathbb{1}^{(a_6)} & — & — & \mathbb{1}^{(1)} & \mathbb{1}^{(0)} \end{array}\right),$$

where $\mathbb{1}^{(0)} \triangleq \mathbb{1}$ is the $m \times m$–identity matrix and $\mathbb{1}^{(a)}$ is the $m \times m$–identity matrix shifted cyclically to the left a times, and — is an $m \times m$–zero matrix. E.g. $\mathbb{1}^{(1)}$ has ones at positions $(1, m)$, $(2, 1)$, $(3, 2)$, $(4, 3)$, …, $(m, m-1)$. The parity-check matrix has size $4m \times 6m$, where the columns to the left of the vertical separation line correspond to the $2m$ state (or hidden) bits and the columns to the right of the vertical separation line correspond to the $4m$ code bits that are sent over the channel. As can easily be seen, the designed rate is 1/2. □

Example 5.24 (QCRA LDPCC) In Constr. 5.23 the degrees of freedom lie in choosing the integers m and a_i, $i = 1, \ldots, 6$. In an explicit construction[9], we have chosen $m = 11$, which resulted in a code with length 44 and designed rate 1/2. $a_1, \ldots, a_6$ were selected among all possible choices so as to get the largest possible minimum distance and girth. A solution yielding a minimum distance of 8 and a girth of 10 is $a_1 = 1$, $a_2 = 0$, $a_3 = 1$, $a_4 = 9$, $a_5 = 0$, and $a_6 = 4$. Fig. 3.17 on page 49 shows the resulting factor graph, where the first 22 columns of $\mathbf{H}$ correspond to the inner bits whereas the last 44 columns of $\mathbf{H}$ correspond

[8] See also the recent paper [130], where it was looked at this type of codes and where it was tried to relate generalized belief propagation for usual LDPCCs to belief propagation for LDPCCs with state bits.

[9] This unpublished code was jointly designed by R. M. Tanner and the present author (June/July 1999).

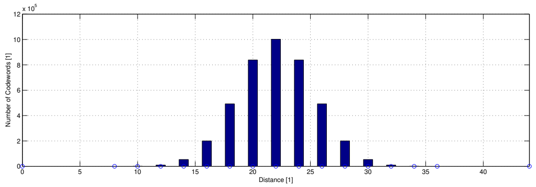

Figure 5.11: Distance spectrum of the $[44, 22]$ code presented in Ex. 5.24. The horizontal and vertical axes are both linear; there is a circle on the horizontal axis if and only if there is at least one codeword of the corresponding weight, e.g. there are no codewords of odd weight.

to the outer bits. So, the first 22 columns essentially describe the interleaver, whereas the last 44 columns describe the "repeat-accumulate" part. Fig. 5.11 shows the weight spectrum (see Def. 2.3) of this $[44, 22]$ code. Of course, one can also consider transmitting all 66 bits, resulting in a designed rate-1/3 code. □

5.7 Turbo Codes

We discuss how interleavers for parallel concatenated turbo codes with good minimum distance can be derived from graphs having large girth.

5.7.1 Introduction and Motivation

In [20] it is shown for parallel concatenated turbo codes that the minimum distance has an upper bound which is proportional to the logarithm of the interleaver length and therefore also proportional to the logarithm of the block length (see [20] for the exact expressions). In this statement it is tacitly assumed that we consider a class of turbo codes where the component convolutional codes are fixed and only the interleaver changes with the interleaver length. The idea of the proof is essentially that certain non-zero low-weight codewords can easily be shown to exist and one can give an upper bound on their weight. Our approach in this

section is to construct interleavers which try to avoid all these low-weight codewords as far as possible. We will derive the interleavers from graphs which have large girth.

For other algebraic constructions of interleavers based on other principles, see e.g. [108] or [113] and the references therein. See also [13] for some extensions of the results of [20], and see [119] for some random constructions that optimize the length of certain cycles.

Our approach is the following: in Sec. 5.7.2 we introduce different graphs representing parallel concatenated turbo codes, Sec. 5.7.3 discusses which low-weight codewords should be avoided, and Sec. 5.7.4 presents different approaches for constructing interleavers.

5.7.2 Different Graphs representing Turbo Codes

In this section we discuss different representations of parallel concatenated turbo codes with the help of graphs. Parallel concatenated turbo codes consist of two systematic recursive convolutional codes (RCCs) whereby the input bits of the second RCC are the permuted input bits of the first RCC. A first possible representation is by factor graphs [128, 58] as given in Fig. 5.12 (see also Ex. 3.25). The empty simple circles represent input and output bits, the empty double circles represent states and the filled squares represent the constraints (trellis sections). The upper part represents the first RCC, the lower part represents the second RCC, and in-between is the interleaver which permutes the input bits. A typical constraint node (trellis section) of the first RCC has at time index k the state $s^{(1)}_{k-1}$ on the left-hand side, the state $s^{(1)}_{k}$ on the right-hand side, the information bit u_k as input, and the parity bit $x^{(1)}_k$ as output. A typical trellis section of the second RCC has at time index k the state $s^{(2)}_{k-1}$ on the left-hand side, the state $s^{(2)}_{k}$ on the right-hand side, the information bit $u_{\pi^{-1}(k)}$ as input, and the parity bit $x^{(2)}_k$ as output; the function $\pi(.)$ represents the permutation of the input bits of the first RCC to the input bits of the second RCC. We assume to have no puncturing and so at time k one transmits the three bits u_k, $x^{(1)}_k$, and $x^{(2)}_k$, which results in a designed rate-1/3 code. One can use termination of the turbo code as e.g. proposed in [26].

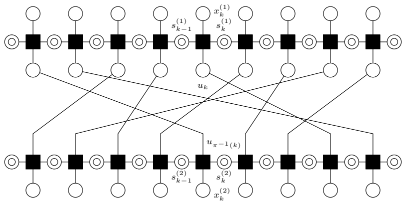

Figure 5.12: Factor graph of a parallel concatenated turbo code with interleaver length 9.

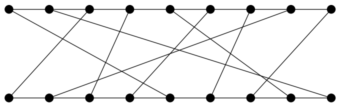

Figure 5.13: Interleaver graph (IG) of a parallel concatenated turbo code with interleaver length 9.

For guaranteeing that the summary-product algorithm works well on a specific factor graph, it is advisable that there are no small cycles so that the factor graph looks locally tree-like in order that the messages are as independent as possible [128]. To see better what cycles are involved in the factor graph of Fig. 5.12 we omit all nodes having degree 1 and 2. (Note that completing the factor graph by adding the function nodes necessary to represent transmission over a memoryless channel does not create new loops.) In this way, we obtain a graph which looks like the one in Fig. 5.13; we call such a graph an interleaver graph (IG). Afterwards, we will see that having no small cycles helps not only the summary-product algorithm but it also helps to avoid low-weight codewords.

5.7.3 Low-Weight Codewords

Assume that both RCCs have memory size ν and that the denominator polynomials (which must not necessarily be the same) are primitive polynomials. Let $P = 2^\nu - 1$ and let us focus on only one of these RCCs. One can show (see e.g. [20]) that if there are two "1"s at a distance $L \cdot P$

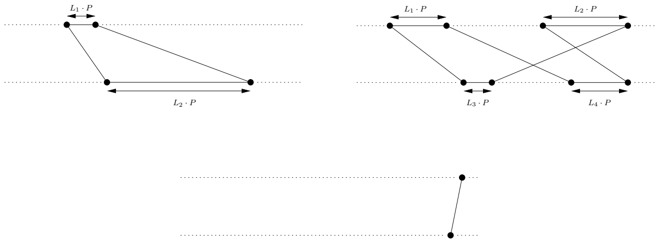

Figure 5.14: Top left: pattern in an interleaver graph with 2 input bits being "1" leading to a low-weight codeword if $L_1 + L_2$ is small. Top right: pattern in an interleaver graph with 4 input bits being "1" leading to a low-weight codeword if $L_1 + L_2 + L_3 + L_4$ is small. Bottom: pattern with one input bit being "1" leading to a low-weight codeword. (P is the period length of the component convolutional codes, see text for more explanations.)

apart at the input ($L \in \mathbb{N}$), only finitely many "1"s are produced at the output: the weight is not larger than $\alpha \cdot L + \beta$, where α and β are constants depending only on the convolutional encoder. Therefore one should avoid interleaver graphs as depicted in Fig. 5.14 (top left) having connections such that $L_1 + L_2$ is small as this produces codewords of weight not larger than $2 + \alpha_{\mathrm{C1}} L_1 + \alpha_{\mathrm{C2}} L_2 + \beta_{\mathrm{C1}} + \beta_{\mathrm{C2}}$. ($\alpha_{\mathrm{C1}}$, β_{C1}, α_{C2}, and β_{C2} are the α's and β's of the first and second RCC, resp.) In the same manner, situations as in Fig. 5.14 (top right) should be avoided where $L_1 + L_2 + L_3 + L_4$ is small as this produces codewords of weight not larger than $4 + \alpha_{\mathrm{C1}}(L_1 + L_2) + \alpha_{\mathrm{C2}}(L_3 + L_4) + \beta_{\mathrm{C1}} + \beta_{\mathrm{C2}}$. Of course, this can easily be generalized to longer cycles. Whereas situations as in Fig. 5.14 (top left) are handled by spread (S-random) interleaver designs as in [27] or related design techniques like [60], they cannot handle situations like in Fig. 5.14 (top right).

One sees that the codewords of minimum weight are upper bounded by a constant plus a linear function of the shortest length of certain special cycles. In [20] a graph similar to our interleaver graph is derived where one sees these special cycles explicitly. Roughly, as the girth of a graph (having degrees larger than two) is upper bounded by a constant

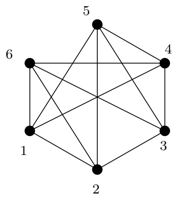

Figure 5.15: An example graph with uniform degree 4.

times the logarithm of the number of vertices, one finally gets the result that the minimum distance can at best only grow proportionally to the logarithm of the interleaver length.

A different situation that should be avoided is depicted in Fig. 5.14 (bottom): a low-weight codeword (generated by a single “1” among the input bits) is produced because there is an edge connecting a vertex near the very end of the upper chain and a vertex near the very end of the lower chain. (We call the subgraph consisting of the vertices and horizontal edges of the upper part of the IG the upper chain; the subgraph consisting of the vertices and horizontal edges of the lower part is called the lower chain.)

5.7.4 Construction of Interleaver Graphs

As the weight of low-weight codewords is related to the girth of the IG (see Sec. 5.7.3), we will try to construct IGs whose girth is large. The idea is to derive from the LPS Ramanujan graphs (RGs) $\mathsf{X}^{\mathrm{LPS}}_{q,p}$ described in Sec. 4.3[10] an IG as e.g. shown in Fig. 5.13 where we are able to give a lower bound on the girth. A possible way to do this is as follows.

In a first step we start with an RG with arbitrary odd prime $q \neq 3$ and $p = 3$, i.e., the graph has uniform degree $3 + 1 = 4$. We take an arbitrary Eulerian walk (EW) in it.[11] As the RG has uniform degree

[10] Our proposed constructions can start with any graph having large girth and regular degree four. If one is interested in asymptotic existence results, the class of LPS Ramanujan graphs is especially useful.

[11] For the definition of EWs, see Def. 2.11. Note that EWs can be constructed in

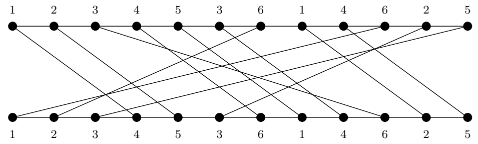

Figure 5.16: A first IG derived from the graph in Fig. 5.15.

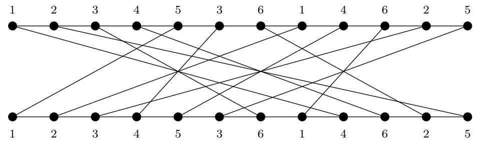

Figure 5.17: A second IG derived from the graph in Fig. 5.15.

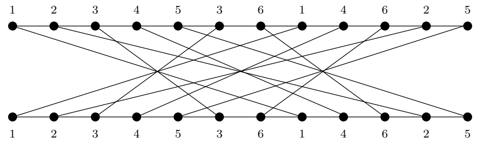

Figure 5.18: A third IG derived from the graph in Fig. 5.15.

4, each vertex gets visited twice. E.g., Fig. 5.15 shows a graph with 6 vertices and uniform degree 4 (this is not an RG as defined above, this graph was chosen for demonstration purposes only). A possible EW visits the vertices $1-2-3-4-5-3-6-1-4-6-2-5-(1)$.

In a second step we define the labeling of the upper chain of the IG. We take an upper chain that has twice the number of vertices of the RG and the labeling is done according to the EW chosen before. (Actually, we get a long closed chain, but we cut it at an arbitrary edge.) The lower chain is labeled in the same manner. Continuing the example we started in Fig. 5.15 we get the upper and lower chain as in Fig. 5.16.

In a third step we finally define which vertex in the upper chain is connected to which vertex in the lower chain. E.g. for the first vertex of the upper chain with label "1", this can be done in the following way: determine the right neighbor of the second "1" in the lower chain, which in this case is "4". The first vertex of the upper chain with label "1" is finally connected to the other occurrence of "4" in the lower chain. The other vertices are connected in the same manner and we obtain the IG as shown if Fig. 5.16.

It is not difficult to see that every walk in the IG can be mapped back to a walk in the original graph, especially every cycle can be mapped back to a closed walk. Therefore, the girth of the interleaver graph is at least as large as in the original graph and we can use the same lower bound on the girth as for the RG. If, as in our case, the original graph is an RG with parameters q and $p = 3$, then the girth grows proportional to the logarithm of q or equivalently proportional to the logarithm of the interleaver length. Finally, also the resulting minimum distance will grow proportionally to the logarithm of the interleaver length, although the coefficient in front of the logarithm will be quite weak.

A different possibility is to connect the first vertex with label "1" in the upper chain to the right neighbor of the second "1" in the lower chain, i.e. to the vertex with label "4" that lies between the vertices with labels "1" and "6". The other connections are done in a completely analogue manner and we obtain the IG as shown in Fig. 5.17. Again, the girth can be shown to be at least as good as in the original RG.

Of course, instead to connecting to the right neighbor in the last

time proportional to the number of edges in the graph.

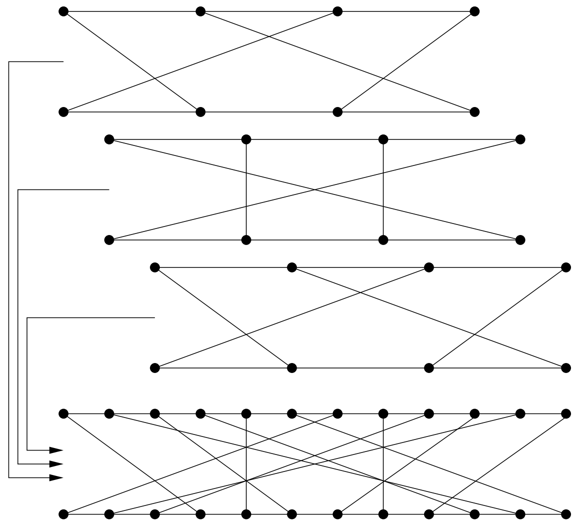

Figure 5.19: Out of P small IGs of length 4 a large IG of length 12 is constructed (here $P = 3$).

construction one can also connect to the left neighbor and still make the same conclusions. In fact, one can also connect the first vertex with label "1" in the upper chain with the second vertex with label "1" in the lower chain and do the other connections analogously (see Fig. 5.18).

As mentioned, the problem in the construction just presented is though in the small constant in front of the logarithm. It is advisable to have a good girth but it is also equally important to avoid certain patterns leading to codewords having low weight as shown in Sec. 5.7.3. One can resolve this problem with the following construction. For a given P (where P is defined in Sec. 5.7.3), one constructs in the first step P (small) IGs of the same size using one of the above constructions. In the next step one combines them as shown in Fig. 5.19 to a new (large) IG. It is not difficult to see that the weight of every low-weight codeword of the form as discussed in Sec. 5.7.3 is a function of the girth of the small IGs and for this girth we can give a lower bound by construction.

Additionally, we can shift the lower chain relatively to the upper chain to avoid unlucky connections at the end of the IG as pointed out at the end of Sec. 5.7.3 (Fig. 5.14(bottom)).

As a specific construction we consider the case $q = 5$, $p = 3$. As p is a quadratic non-residue modulo q, the RG has $q(q^2 - 1) = 120$ vertices of degree 4. An EW of such an RG must be of length $2 \cdot 120 = 240$ and therefore the small IGs have upper and lower chain lengths of 240 (we used different possibilities to get from the EW to the small IG). As component RCCs we choose systematic RCCs of memory size $\nu = 3$ with transfer functions $n_1(D)/d_1(D)$ and $n_2(D)/d_2(D)$, respectively, where $n_1(D) = n_2(D) = 1 + D + D^2 + D^3$, $d_1(D) = 1 + D + D^3$ and $d_2(D) = 1 + D^2 + D^3$; therefore $P = 7$. The large IG of length $7 \cdot 240 = 1680$ consists of 7 copies of the small IG (each of them individually shifted cyclically by a certain offset). As we do no puncturing of the code, we get a designed rate of 1/3. We use termination of the turbo code, so the final code has length $3 \cdot (1680 + 3) = 5049$ and 1680 information bits. Using the algorithms as proposed in [40] to compute the minimum distance we obtained a minimum distance of 30. This compares favorably with results shown in [39] and [40]: Table 2 of [39] shows that a random interleaver construction of turbo codes with the same interleaver length and memory size $\nu = 3$ ($P = 7$) gives an average minimum distance of roughly 14 and a best minimum distance of 22. Table IV and Fig. 7 of [40] show that random interleaver construction of turbo codes with interleaver length 1280 and memory size $\nu = 4$ ($P = 15$) gives an average minimum distance of 19.7 and a best minimum distance of 30. In Table II of [40] the CCSDS rate-1/3 code with interleaver length 1784 and $\nu = 4$ ($P = 15$) is shown to have minimum distance 32. (Note that our construction has only $\nu = 3$ and $P = 7$.)

Remark 5.25 (Variations, open problems) We would like to comment on different points.

- Hamiltonian cycle: instead of an EW one can take a Hamiltonian cycle.[12] In this case, instead of starting with a graph with uniform degree four one can of course also start with a graph with uniform degree three. (Unless the graph construction is such that one knows what a possible Hamiltonian cycle is, finding a Hamiltonian cycle in a graph — if there is one at all — is usually a non-trivial problem.)

[12] For the definition of a Hamiltonian cycle, see Def. 2.11.

- Interleaver length: the algorithms presented in [40] work in reasonable time for small and medium interleaver lengths. The minimum distance of turbo codes with random interleavers of these sizes can therefore be checked by these algorithms. This is not anymore possible for long interleaver lengths; there lies a potential advantage of our construction over random constructions.
- The presented method works especially well for medium sized to long sized interleavers. An important step will be to find good possibilities to derive IGs with more possible choices of interleaver length.
- Description size: if the large IG consists of several copies of the same small IG, one needs only to save the interleaver of the small IG and the offsets (therefore one needs roughly P times less space to save the interleaver).
- One might also study different versions of how to combine several small IGs to a big IG.
- In [62] so-called unifilar turbo codes were discussed. It is possible to derive interleavers for such turbo codes in similar ways as proposed in this section for the derivation of IGs for classical turbo codes.

□

5.8 Simulation of Channel Codes Under Iterative Decoding

5.8.1 The Setup

This section describes simulation results of a selected subset of the codes that were presented in this chapter. Our simulations were based on the model shown in Fig. 3.3: we mainly considered memoryless channels like the binary-input additive white Gaussian noise channel (BI-AWGNC) under the assumption that synchronization on the symbol and block level is guaranteed. Of course, as mentioned in Ch. 3, other channel models would also fit into the iterative decoding concept. Indeed, one of the strength of factor graphs and the summary-product algorithm is that very general scenarios can be treated in a unified manner. But such

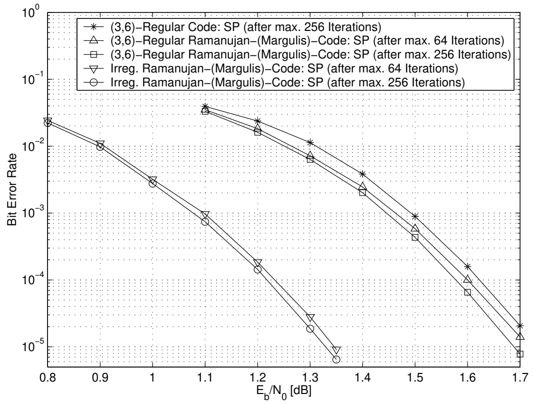

Figure 5.20: Bit-error rate simulation results for different codes derived from Cayley graphs with large girth using Margulis's (modified) construction idea when transmitting over the BI-AWGNC (see also text).

more complex channel models were not at the heart of this thesis, so we will stick with the BI-AWGNC.

We used the sum-product algorithm (indicated by "SP" in the plot legends) with a message update schedule as discussed in Sec. 3.5.3: the decoding was stopped when either the hard-decision of the channel bits gave a valid codeword or if a maximum number m of iterations was reached (indicated in the plot legends by "after max. m iterations").

5.8.2 Simulation of Codes based on Margulis's (Modified) Construction Idea

Fig. 5.20 shows bit error rate curves for the following binary codes of length 4896 and designed rate 1/2:

- a randomly generated $(3,6)$-regular LDPCC where four-cycles have been eliminated (its actual rate is 1/2),

- a $(3,6)$-regular LDPCC as discussed in Constr. 5.2 based on a $\mathsf{X}^{\mathrm{LPS}}_{q,p}$ graph with $q = 17$ and $p = 5$ (its actual rate is 0.5053),
- an irregular LDPCC as given in Ex. 5.12 with maximum bit node degree 11 (its actual rate is 1/2).

It can be seen that the second code is slightly better than the first code: the main reason for this is that the rate of the second code is slightly above the designed rate and the horizontal axis is E_{b}/N_0, i.e. the information bit energy divided by the single-side noise spectral density.[13] So, algebraic constructions can offer here a slight advantage, as such a rank loss of the parity-check matrix can only come from an algebraic construction. (Note that randomly constructed parity-check matrices have usually a very small rank loss.)

The third code, which is an irregular SS-LDPCC offers the performance advantages that can be expected from a code having this irregularity (see e.g. [95]).

Remark 5.26 (Weaknesses) Already in [98] we pointed out that there is an error floor for the codes originally proposed by Margulis [82]. Recently, MacKay and Postol [74] pointed out that in that case the error floor does not stem from low-weight codewords but from what they call near-codewords[14] where the sum-product apparently gets stuck. They also point out that the above $(3,6)$ code based on the graph $\mathsf{X}^{\mathrm{LPS}}_{q,p}$ with $q = 17$ and $p = 13$ has a codeword of weight 24 causing an error floor of the *block* error rate somewhat below 10^{-6} (corresponding to a *bit* error rate somewhat below 10^{-7}) [74]. (Note that for this last code, a tree-oriented minimum distance bound (see Def. 5.31) yields a minimum distance lower bound of 14.)

These weaknesses are indeed not satisfactory. We suspect that a possible weakness could stem from the fact that the parity-check matrix can be written as $\mathbf{H} = [\mathbf{H}_1|\mathbf{H}_2]$, where $\mathbf{H}_1$ and $\mathbf{H}_2$ have both regular column and row weight 3, respectively. If $\mathbf{H}_1$ and $\mathbf{H}_2$ were circulant matrices, a possible generator matrix would be $\mathbf{G} = [\mathbf{H}_2^T|\mathbf{H}_1^T]$ because circulant matrices commute; obviously, the minimum distance of such a code would be at most 6. $\mathbf{H}_1$ and $\mathbf{H}_2$ are not circulant matrices in

[13] This is the standard way of plotting the bit-error rate for codes having different rates.

[14] Near-codewords are low-weight vectors with a low-weight syndrome.

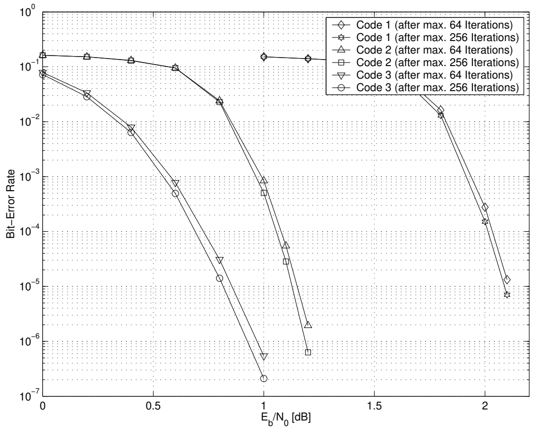

Figure 5.21: Bit-error rate simulation results of three different rate-1/3 codes transmitted over the BI-AWGNC (see also text).

the Margulis construction based on Ramanujan graphs, but they can be considered as circulant matrices over a non-abelian group. Of course such circulant matrices do not commute anymore but it is to be suspected that in a certain sense they nearly commute.

We propose the following approach that hopefully avoids these weaknesses: the matrices $\mathbf{H}_1$ and $\mathbf{H}_2$ should have less in common, this can e.g. be achieved by combining different Ramanujan graphs as was done in Sec. 4.3.3. This approach seems indeed to work. We started with an $\mathsf{X}^{\mathrm{LPS}}_{17,3,5}$ graph where we did not change the left-hand side but took only 3 out of 5 generators from the right-hand side in order to construct a $(3,6)$-regular SS-LDPCC. This matrix has no rank loss anymore so we lost the slight advantage of the rank loss of 26 lines that we had observed above. But simulations indicate that there is no error floor where we had one before. □

5.8.3 Simulation of Irregular WS-LDPCCs

In Fig. 5.21 we show the simulation results of three different codes.

- Code 1 is a randomly generated $(4,6)$-regular rate-1/3 LDPCC with parameters $[13104, 4368]$ where cycles of length four have been eliminated.
- Code 2 is a rate-1/3 code with the parameters $[13104, 4368]$ based on a 12-regular (bipartite) LPS Ramanujan graph with $q = 13$ and $p = 11$ where as subcodes we have uniformly taken $[12, 8, 3]$ subcodes (this code is similar in spirit to one presented in [59], see also Sec. 5.17).
- Code 3 is a rate-1/3 code with parameters $[7098, 2366]$ based on the same (bipartite) LPS Ramanujan graph with $q = 13$ and $p = 11$. But now, we used different subcodes: in the left half, 50% of the vertices are type-1 replacements having a $[12, 8, 3]$ subcode and 50% are type-2 replacements having a bit-node of degree 12; in the right half we use type-4 replacements where 10% have a $[12, 8, 3]$ subcode, 40% have a $[12, 9, 2]$ subcode, 23% have a $[12, 10, 2]$ subcode, and 27% have a $[12, 11, 2]$ subcode (i.e., a simple XOR).[15]

Observe that Code 3 is by 46% shorter than the Codes 1 and 2(!) At the moment, the subcodes have been placed randomly, but future work will aim at placing the subcodes in a sensible algebraic fashion. Let us mention that the threshold given by channel capacity for rate-1/3 codes transmitted over the BI-AWGNC is at -0.49 dB and that the threshold of the $(4,6)$-regular LDPC code ensemble lies at 1.67 dB (see [96]).

5.8.4 Simulation of Codes from Finite Geometries

First we consider a rate-7/16 $[400, 175, 16]$ binary code from class $\mathcal{C}^{(1)}$ which is based on an FGQ of order (q, q) with $q = 7$ and has bit and check degree $q + 1 = 8$, see Constr 5.8 and Ex. 5.10.

The bit error rate can be seen in Fig. 5.22: we see that increasing the maximal number of iterations steadily increases the performance (in the

[15] Certainly, with more time and effort, one can find better proportions.

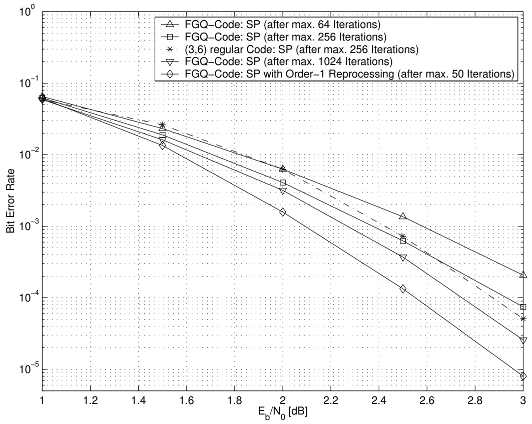

Figure 5.22: Bit error rate for a code of class $\mathcal{C}^{(1)}$ when transmitting over the BI-AWGNC (see also text).

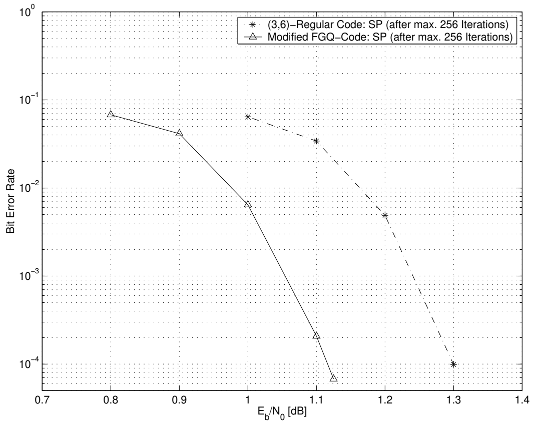

Figure 5.23: Bit error rate for a code of class $\mathcal{C}^{(2)}$ and a $(3,6)$-regular SS-LDPCC when transmitting over the BI-AWGNC (see also text).

range shown). Additionally, Fig. 5.22 shows the bit-error rate curve when using the sum-product algorithm and order-1 reprocessing as discussed in Rem. 3.31: quite a significant performance gain can be achieved, even with a smaller number of maximal number of iterations. For comparison purposes we also plotted the bit error rate for a randomly generated rate-$1/2$ $[400, 200]$ linear $(3, 6)$-regular code.

In general, it seems to be that in the realm of (very) high-rate codes, codes from class $\mathcal{C}^{(1)}$ that are derived from finite geometries offer a performance advantage over randomly constructed codes, see e.g. [67, 51]. Partly, this is because 4-cycles are very difficult to avoid for randomly constructed codes of very high rate.

As a second example we show simulation results of a code of class $\mathcal{C}^{(2)}$ derived from an FGQ, see Constr. 5.8. Fig. 5.23 shows a length 33320 code of rate $1/2$ based on a 14-regular graph with girth 8 stemming from the FGQ $W(q)$ with $q = 13$. As subcodes we took $[14, 10, 3]$ and $[14, 11, 2]$ codes, which were placed randomly with the following restriction: in order to avoid low-weight codewords we put the $[14, 10, 3]$ codes at the vertices representing points in the point-line graph and the $[14, 11, 2]$ codes at the vertices representing lines in the point-line graph. We compare this code with a $(3, 6)$-regular SS-LDPCC of the same length and rate; the performance gains are in the same range as the performance gains for a similar construction based on Ramanujan graphs reported in [59] (see also Sec. 5.17).

5.9 Minimum Distance

Definition 5.27 (Notation) Let $\mathbf{H}$ be a parity-check matrix and let $\mu_1 > \mu_2 > \cdots > \mu_s$ be the ordered eigenvalues of $\mathbf{L} \triangleq \mathbf{H}^T\mathbf{H}$. When $\mathbf{H}$ has uniform row weight we call this weight w_{row} and when it has uniform column weight we call this weight w_{col}. $\square$

The following two lower bounds on the minimum distance for binary codes which have a parity-check matrix that has constant column weight and constant row weight, respectively, were proposed by Tanner [114]. (More about the first bound can also be found in App. E.)

Theorem 5.28 (Bit-oriented minimum distance lower bound) *Let a code $\mathcal{C}$ of length n have the parity-check matrix $\mathbf{H}$ which has uniform column weight w_{col} and uniform row weight w_{row}. If the multiplicity of the largest eigenvalue μ_1 is of $\mathbf{L}$ one, then*

$$d_{\min} \geq n \cdot \frac{2w_{\mathrm{col}} - \mu_2}{\mu_1 - \mu_2}.$$

Proof: See [114]. □

Theorem 5.29 (Parity-oriented minimum distance lower bound) *Let a code $\mathcal{C}$ of length n have the parity-check matrix $\mathbf{H}$ which has uniform column weight w_{col} and uniform row weight w_{row}. If the multiplicity of the largest eigenvalue μ_1 of $\mathbf{L}$ is one, then*

$$d_{\min} \geq n \cdot \frac{2}{w_{\mathrm{row}}} \cdot \frac{2w_{\mathrm{col}} + w_{\mathrm{row}} - 2 - \mu_2}{\mu_1 - \mu_2}.$$

Proof: See [114]. □

Remark 5.30 (Eigenvalues of $\mathbf{H}^T\mathbf{H}$ and of $\mathbf{H}\mathbf{H}^T$) Note that the non-zero eigenvalues of $\mathbf{H}^T\mathbf{H}$ and $\mathbf{H}\mathbf{H}^T$ and their multiplicities are the same [114]. For the above two bounds it is therefore irrelevant if we define $\mathbf{L} \triangleq \mathbf{H}^T\mathbf{H}$ or $\mathbf{L} \triangleq \mathbf{H}\mathbf{H}^T$ in Def. 5.27. □

Additionally, sometimes the following approach gives a simply calculatable lower bound on the minimum distance.

Definition 5.31 (Tree-oriented minimum distance lower bound) For each bit position i in the code we define a $d_{\min}(i)$ which is given by the following procedure: set this bit to "1" and see how many bits at least (in a worst-case scenario) must also be "1" so that all constraints are fulfilled except for some constraints which correspond to check nodes that are far away from the bit node i in the factor graph of the code. A lower bound on $d_{\min}$ is of course given by the smallest $d_{\min}(i)$. Note that for a bit-node-transitive code all $d_{\min}(i)$ are the same. □

To be specific, we discuss the three bounds for a code derived from a projective plane.

Theorem 5.32 (Lower bounds on the minimum distance of codes from projective planes) *Let a class $\mathcal{C}^{(1)}$ code derived from a projective plane Γ of order q (i.e. from an FGT of order $(s,t) = (q,q)$) be given by the parity-check matrix $\mathbf{H} = \mathbf{I}(\Gamma)$. Then the bit-oriented bound (BB), the parity-oriented bound (PB) and the tree bound (TB) are*

$$d_{\min}^{\mathrm{BB}} = q+2, \quad d_{\min}^{\mathrm{PB}} = 2\frac{2q+1}{q+1}, \quad d_{\min}^{\mathrm{TB}} = q+2,$$

i.e. $d_{\min} \geq \max(d_{\min}^{\mathrm{BB}}, d_{\min}^{\mathrm{PB}}, d_{\min}^{\mathrm{TB}}) = q+2$.

Proof: See Sec. 5.11.3. □

Remark 5.33 (about Th. 5.32) We consider the same codes as in Th. 5.32. In [57], the code parameters are listed for $q = 2^s$: one gets $n = 2^{2s} + 2^s + 1$, $k = n - 3^s - 1$, $d_{\min} = 2^s + 2$. So the bit-oriented bound actually gives the correct result, whereas the check-oriented bound is weak as it approaches the value 4 in the limit $q \to \infty$. Note that the code has a rate approaching 1 for $q \to \infty$ despite the fact that the matrix $\mathbf{H}$ is a square matrix (therefore it must have high rank loss). □

Theorem 5.34 (Codes from FGQs) *Let a class $\mathcal{C}^{(1)}$ code derived from an FGQ Γ of order (s,t) be given by the parity-check matrix $\mathbf{H} = \mathbf{I}(\Gamma)$. Then the bit-oriented bound (BB), the parity-oriented bound (PB) and the tree bound (TB) are*

$$\begin{aligned} d_{\min}^{\mathrm{BB}} &= (s+1)(t-s+2), \\ d_{\min}^{\mathrm{PB}} &= 2(t+1), \\ d_{\min}^{\mathrm{TB}} &= 2(t+1), \end{aligned}$$

i.e. $d_{\min} \geq \max(d_{\min}^{\mathrm{BB}}, d_{\min}^{\mathrm{PB}}, d_{\min}^{\mathrm{TB}})$. *See also Tabs. F.12 and F.13.*

Proof: See Sec. 5.11.4. These bounds were presented in our paper [125]. □

Remark 5.35 (about Th. 5.34) We consider the same codes as in Th. 5.34. The (PB) and the (TB) are always equal; if $s = t$, then all three bound are equal. Tabs. F.12 and F.13 give results for FGQs of different order (note that sometimes the number of checks is larger than the number of bits, but all these codes have non-vanishing rate). The above minimum distance lower bounds are in accordance with Bagchi and Sastry [9] where they show that the minimum weight is at least $2(t^{n/2}-1)/(t-1)$ for regular finite generalized n-gons with $n = 4, 6, 8$ (for the definition of regularity see [9]). Codewords achieving this minimal bound correspond to thin FGPs of order $(1, t)$. In the case of FGQs, a codeword with weight $2(t+1)$ defines a (thin) subquadrangle of order $(1, t)$. □

Theorem 5.36 (Codes from partial geometries) *For codes from class $\mathcal{C}^{(1)}$ derived from partial geometries, one can apply the bounds in Ths. 5.28 and 5.29 to get the minimum distance lower bounds shown in Tab. F.8.*

Proof: The proof is similar to the proof of Th. 5.34. These bounds were first presented by Johnson and Weller [51]. □

5.10 Codeword Generation

For turbo codes it is straightforward to produce codewords as turbo codes are defined by an image representation. Encoding of LDPCCs, which are defined by kernel representation, is usually not that easy. In this section, we would like to discuss various codeword generation methods.

- The conceptually simplest method is to construct a generator matrix $\mathbf{G}$ from the parity-check matrix $\mathbf{H}$. Although $\mathbf{H}$ is sparse, $\mathbf{G}$ will in general not be sparse at all. Encoding through $\mathbf{x} = \mathbf{u} \cdot \mathbf{G}$ has consequently a complexity on the order of the squared block length.

- Consider an iterative procedure for decoding codes transmitted over a binary erasure channel (BEC). If there are not too many erasures, such a decoder is able to infer the whole codeword from a set of known positions. But this is exactly the same as encoding: the unerased bits correspond to the information bits whereas the erased bits to the parity bits to be found. This method was analyzed e.g. in [97], where they were able to relate the threshold of the code ensemble when transmitted over a BEC to the encoding complexity. It turns out, that the number of needed operations is $n + \gamma^2$ where n is the block length and γ is the so-called gap, i.e. the difference between capacity of the BEC and the threshold of the code ensemble under consideration. Therefore, codes whose threshold is nearer to the capacity of the BEC are easier to encode.

- If the code is known to be cyclic (e.g. as for some codes derived from projective planes, see Ex. 5.9) or quasi-cyclic, this additional knowledge can be used for simplified encoding (see e.g. [111], especially Sec. IX). In general if one knows a (fast) Fourier transform for the automorphism group (or one of its subgroups) one can use that for speeding-up the encoding.

We now discuss several specific encoding examples.

Example 5.37 (SS-LDPCC from the FGQ $W(2)$) We consider an SS-LDPCC derived from the FGQ $W(q)$ for $q = 2$, see Constr. 5.8. We want to use the techniques introduced in [111] (from where we adopt also the notation): for this, we need to know if the automorphism group of $W(q)$ has a cyclic subgroup. Indeed, as stated in [10], $W(q)$ has the automorphism group $\Gamma\,\mathrm{Sp}(4, q)$, where $\mathrm{Sp}(4, q)$ has the a cyclic subgroup of order $q^2 + 1$.

For $q = 2$, we get a cyclic subgroup of order $q^2 + 1 = 5$. The parity-check matrix can now be re-grouped such that all bits belonging to the same orbit are next to each other and so that checks belonging to the same orbit are next to each other. We get (where $\mathbb{1}^{(s)}$ is a 5×5 identity matrix shifted left cyclically s times and $-$ is a zero matrix of appropriate size)

$$\mathbf{H} = \begin{pmatrix} \mathbb{1}^{(2)} & \mathbb{1}^{(0)} & \mathbb{1}^{(3)} \\ \mathbb{1}^{(1)} & \mathbb{1}^{(0)} + \mathbb{1}^{(3)} & - \\ \mathbb{1}^{(4)} & - & \mathbb{1}^{(2)} + \mathbb{1}^{(3)} \end{pmatrix}.$$

We want to transform this matrix by multiplication with an invertible matrix from the left. Let $\mathbb{F}_{2^4}$ be the finite field with 2^4 elements and let ξ be a primitive element in $\mathbb{F}_{2^4}$. Let $\alpha \triangleq \xi^3$, so the order of α is 5. Let $\mathbf{T} \triangleq \mathrm{diag}(\mathbf{F}, \mathbf{F}, \mathbf{F})$ be a block-diagonal matrix where $\mathbf{F}$ is a discrete Fourier transform matrix of size 5×5 based on α, i.e. $[\mathbf{F}]_{ij} = \alpha^{(i-1)(j-1)}$ (indexing from 1 to 5). We get $\tilde{\mathbf{H}} = \mathbf{T} \cdot \mathbf{H}$ and the row space of $\tilde{\mathbf{H}}$ is spanned by $\mathbf{\Lambda}_0 \otimes (1, 1, 1, 1, 1)$ and $\mathbf{\Lambda}_1 \otimes (1, \alpha, \alpha^2, \alpha^3, \alpha^4)$ and its conjugates where

$$\mathbf{\Lambda}_0 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & - & - \\ 1 & - & - \end{pmatrix}, \qquad \mathbf{\Lambda}_1 = \begin{pmatrix} \alpha^2 & \alpha^0 & \alpha^3 \\ \alpha^1 & \alpha^0 + \alpha^3 & - \\ \alpha^4 & - & \alpha^2 + \alpha^3 \end{pmatrix}.$$

Both $\mathbf{\Lambda}_0$ and $\mathbf{\Lambda}_1$ have rank 2; following [111], $\mathbf{H}$ therefore has rank $\mathrm{rank}(\mathbf{\Lambda}_0) \cdot 1 + 4 \cdot \mathrm{rank}(\mathbf{\Lambda}_1) \cdot 1 = 10$. Let $\mathbf{\Gamma}_{-i}$ span the space orthogonal to the one spanned by $\mathbf{\Lambda}_i$, $i = 0, \ldots, 4$. The row space of the generator matrix is then spanned by the row space of the matrices $\mathbf{\Gamma}_0 \otimes (1,1,1,1,1)$ and $\mathbf{\Gamma}_{-1} \otimes (1, \alpha^{-1}, \alpha^{-2}, \alpha^{-3}, \alpha^{-4})$ and its conjugates. We get

$$\mathbf{\Gamma}_0 = \begin{pmatrix} - & 1 & 1 \end{pmatrix}, \qquad \mathbf{\Gamma}_{-1} = \begin{pmatrix} \alpha^2 + \alpha^4 & 1 & \alpha^1 + \alpha^2 + \alpha^3 \end{pmatrix}.$$

Both $\mathbf{\Gamma}_0$ and $\mathbf{\Gamma}_{-1}$ have rank $3 - 2 = 1$, so following [111] the total rank of the generator matrix is $\mathrm{rank}(\mathbf{\Gamma}_0) \cdot 1 + 4 \cdot \mathrm{rank}(\mathbf{\Gamma}_{-1}) \cdot 1 = 5$ (which of course is equivalent to $15 - 10$). A possible choice for the generator matrix is

$$\mathbf{G} = \begin{pmatrix} \mathbb{1}^{(2)} + \mathbb{1}^{(4)} & \mathbb{1}^{(0)} & \mathbb{1}^{(1)} + \mathbb{1}^{(2)} + \mathbb{1}^{(3)} \end{pmatrix}.$$

□

Example 5.38 (SS-LDPCC from the FGQ $W(5)$) This case can be treated in the same style as the case $q = 2$ shown in Ex. 5.37. This time the automorphism group of the code has a cyclic subgroup of order $q^2 + 1 = 26 = 2 \cdot 13$, so there is also a cyclic subgroup of order 13 and we take a generator of such a subgroup. □

Example 5.39 (Regular WS-LDPCCs and optimized irregular WS-LDPCC) To estimate the encoding complexity of SS-LDPCCs one can e.g. use the results of Richardson and Urbanke [97] which show that the complexity is on the order of $n + g^2$ where n is the block length and γ is the so-called gap. By their simplest procedure (Cor. 1), the gap can then be estimated to be $\gamma = (1 - R - \alpha^*)n$, where R is the rate and α^* is the threshold of the code ensemble being transmitted over the *binary erasure channel (BEC)*. This procedure can be extended to hold also for WS-LDPCCs.

- For the ensemble where Code 1 in Sec. 5.8.3 stems from we obtain $\alpha^* = 0.506$ and $\gamma = 0.1606n$;
- for the ensemble where Code 2 in Sec. 5.8.3 stems from we obtain $\alpha^* = 0.584$ and $\gamma = 0.0827n$;
- finally, for the ensemble where Code 3 in Sec. 5.8.3 stems from we obtain $\alpha^* = 0.636$ and $\gamma = 0.0307n$.

The calculation of α^* for irregular SS-LDPCCs is done by specifying the degree polynomial $\lambda(x)$ belonging to the bit nodes and the degree polynomial $\rho(x)$ belonging to the check nodes. The update formula for calculating the average proportion of error messages after each iteration can then immediately be stated and α^* can be calculated (see, e.g., [106] and references therein). For ensembles of codes where Code 2 and 3 stem from, a similar update formula can be derived. Interesting is the fact that the pseudo-$\rho(x)$ that would correspond to more complex subcodes can contain positive *and* negative coefficients (the sum of them is still 1), whereas $\rho(x)$ has only non-negative coefficients when all subcodes are simple XORs. We conjecture that this large choice of freedom might open interesting ways of constructing capacity-achieving ensembles over the binary erasure channel. □

5.11 Proofs

5.11.1 Proof of Lemma 5.6

In the case $g = 2 \pmod 4$, we start at a check node and count the number of check nodes reached after a distance at most $\frac{g}{2} - 1 = 2 \cdot \left(\frac{g-2}{4}\right)$ steps. This results in the inequality

$$1 + w_{\mathrm{row}}(w_{\mathrm{col}} - 1) + w_{\mathrm{row}}(w_{\mathrm{col}} - 1)\alpha + \cdots + w_{\mathrm{row}}(w_{\mathrm{col}} - 1)\alpha^{\frac{g-2}{4} - 1} \leq m,$$

where α was given in the lemma statement. From this we deduce the inequality

$$1 + \alpha + \cdots + \alpha^{\frac{g-2}{4}} = \frac{\alpha^{\frac{g-2}{4}+1} - 1}{\alpha - 1} \leq m.$$

The result readily follows. In the case $g = 0 \pmod 4$, we start at a symbol node and count the number of check nodes reached after a distance at most $\frac{g}{2} - 1 = 2 \cdot \frac{g}{4} - 1$ steps; we omit the details. □

5.11.2 Proof of Properties 5.15

The number of bit and check nodes follows easily from the construction. For calculating the rate we make the following observations. The subcodes are $[n_1, R_1 n_1]$ binary subcodes with redundancy $(1-R_1)n_1$, where $n_1 = d_0$. For the overall code we get from each of the n_0 subcodes $(1-R_1)n_1$ linear constraints leading to a total redundancy of $r = (1-R_1)n_0 n_1$. The code rate is then at least

$$1 - \frac{r}{n} = 1 - \frac{(1-R_1)n_0 n_1}{n} = 1 - \frac{(1-R_1)n_0 n_1}{n_0 n_1/2} = 1 - 2(1-R_1).$$

The girth statement trivially follows from the girth of the underlying graph. □

5.11.3 Proof of Theorem 5.32

To apply the bit-oriented and the parity-oriented bound to codes defined by $\mathbf{H} = \mathbf{I}(\Gamma)$ we have to calculate the eigenvalues of $\mathbf{H}^T\mathbf{H}$, see the following lemma.

Lemma 5.40 *Let* $\mathbf{H}$ *be given as in Th. 5.32.* $\mathbf{L} \triangleq \mathbf{H}^T\mathbf{H}$ *has the two distinct eigenvalues* μ_i *with the multiplicities* ν_i*:*

$$\begin{aligned} \mu_1 &= q^2 + 2q + 1, & \nu_1 &= 1, \\ \mu_2 &= q, & \nu_2 &= q^2 + q. \end{aligned}$$

Proof: (of Lemma 5.40) Let $\mathbf{L}$ of size $n \times n$, where $n = q^2 + q + 1$, have the columns and rows indexed by the points of the projective plane. Associate to row i the point p_i and to column j the point p_j. If $i \neq j$, i.e. $p_i \neq p_j$, then $[\mathbf{L}]_{ij}$ corresponds to the number of lines through both points p_i and p_j, here $[\mathbf{L}]_{ij} = 1$. If point $i = j$, i.e. $p_i = p_j$, then $[\mathbf{L}]_{ij}$ corresponds to the number of lines through p_i, here $[\mathbf{L}]_{ij} = q + 1$. We therefore have $\mathbf{L} = q\mathbb{1} + \mathbf{J}$, where $\mathbf{J}$ is the all-ones matrix. As $\mathbf{L}$ is a circulant matrix, its eigenvalues are given by the discrete Fourier transform of its first row. We get the eigenvalue $\mu_1 = q + n = q^2 + 2q + 1$ with multiplicity 1 and the eigenvalue $\mu_2 = \cdots = \mu_n = q$ with multiplicity $q^2 + q$. □

Proof: (of Theorem 5.32) $\mathbf{H}$ has dimensions $n \times n$, where $n = q^2 + q + 1$ and $w_{\mathrm{col}} = t + 1 = q + 1$ and $w_{\mathrm{row}} = s + 1 = q + 1$. Using the results from Lemma 5.40, we get by applying the bit-oriented bound from Th. 5.28

$$d_{\min} \geq n \cdot \frac{2w_{\mathrm{col}} - \mu_2}{\mu_1 - \mu_2} = (q^2 + q + 1) \cdot \frac{2(q+1) - q}{q^2 + 2q + 1 - q} = q + 2.$$

Similarly, by applying the parity-oriented bound from Th. 5.29 we get

$$\begin{aligned} d_{\min} &\geq n \cdot \frac{2}{w_{\mathrm{row}}} \cdot \frac{2w_{\mathrm{col}} + w_{\mathrm{row}} - 2 - \mu_2}{\mu_1 - \mu_2} \\ &= (q^2 + q + 1) \cdot \frac{2}{(q+1)} \cdot \frac{2(q+1) + (q+1) - 2 - q}{q^2 + 2q + 1 - q} = 2\frac{2q+1}{q+1}. \end{aligned}$$

The tree bound is obtained by considering Fig. 4.12 (right). □

5.11.4 Proof of Theorem 5.34

To apply the bit-oriented and the parity-oriented bound to codes defined by $\mathbf{H} = \mathbf{I}(\Gamma)$ we have to calculate the eigenvalues of $\mathbf{H}^T\mathbf{H}$, see the following lemma.

Lemma 5.41 *The matrix* $\mathbf{L} \triangleq \mathbf{H}^T\mathbf{H}$ *has the three distinct eigenvalues* μ_i *with the multiplicities* ν_i*:*

$$\begin{aligned} \mu_1 &= (s+1)(t+1), & \nu_1 &= 1, \\ \mu_2 &= s + t, & \nu_2 &= \frac{st(s+1)(t+1)}{(s+t)}, \\ \mu_3 &= 0, & \nu_3 &= \frac{s^2(st+1)}{(s+t)}. \end{aligned}$$

Note that derivations of the eigenvalues and their multiplicities for different matrices associated with generalized quadrangles were also given in [89] or [9].

Proof: (of Lemma 5.41) Let $\mathbf{L} = \mathbf{H}^T\mathbf{H}$. Associate to row i of $\mathbf{L}$ the point p_i and to column j the point p_j, respectively.

- If $i = j$, i.e., $p_i = p_j$, then $[\mathbf{L}]_{ij} = t + 1$, $[\mathbf{L}^2]_{ij} = (t+1)(s+t+1)$ and $[\mathbf{L}^3]_{ij} = ((t+1)^2 + s(t+1))(t+1) + s(t+1)^2 + s(s+t)(t+1)$.

- If p_i and p_j are adjacent then $[\mathbf{L}]_{ij} = 1$, $[\mathbf{L}^2]_{ij} = s + 2t + 1$ and $[\mathbf{L}^3]_{ij} = (s + 2t + 1)(t + 1) + s(t + 1) + (s + t)^2 + st(t + 1)$.
- Finally, if p_i and p_j are not adjacent then $[\mathbf{L}]_{ij} = 0$, $[\mathbf{L}^2]_{ij} = t + 1$ and $[\mathbf{L}^3]_{ij} = (t + 1)^2 + (t + 1)(s + t) + s(t + 1)^2$.

From this, one can derive that $\mathbf{L}$ is a root of the equation $X^3 + a_2X^2 + a_1X + a_0 = 0$ with $a_2 = -st-2s-2t-1$, $a_1 = s^2t+st^2+s^2+t^2+2st+s+t$, $a_0 = 0$. Therefore the minimal polynomial of $\mathbf{L}$ is $f(X) = X^2 + a_2X^2 + a_1X + a_0 = (X - \mu_1)(X - \mu_2)(X - \mu_3)$, with μ_1, μ_2, and μ_3 given in the statement of the Lemma. So, this matrix must have the eigenvalues μ_1, μ_2, and μ_3 which are different in the interesting cases ($s \geq 1$, $t \geq 1$). We get the multiplicities of these eigenvalues with the help of Lemma A.13. □

Proof: (Long version of the proof of Theorem 5.34) $\mathbf{H}$ is of size $m \times n$ with $m = (t+1)(st+1)$, $n = (s+1)(st+1)$, column weight $w_{\text{col}} = t+1$, and row weight $w_{\text{row}} = s + 1$. Using the eigenvalues of $\mathbf{H}^T\mathbf{H}$ given in Lemma 5.41, we can apply Ths. 5.28 and 5.29 to get

$$d_{\min}^{\text{BB}} = (s+1)(st+1) \cdot \frac{2(t+1) - (s+t)}{(s+1)(t+1) - (s+t)} = (s+1)(t-s+2),$$

$$\begin{aligned} d_{\min}^{\text{PB}} &= (s+1)(st+1) \cdot \frac{2}{(s+1)} \cdot \frac{2(t+1) + (s+1) - 2 - (s+t)}{(s+1)(t+1) - (s+t)} \\ &= 2(t+1). \end{aligned}$$

To calculate the tree-oriented bound we can choose any bit to start with. So we choose the topmost bit node at level 0 in Figure 4.13 (left). To fulfill all the parity checks at level 1, at least $t + 1$ bits at level 2 must be one. Finally, to fulfill all the checks at level 3 there must be at least $(t + 1) \cdot t/(t + 1) = t$ bits at level 4 which are one. This gives a total of $1 + (t + 1) + t = 2(t + 1)$.

(Short version of the proof of Th. 5.34) For the (BB) and the (PB) one uses 5.28 and 5.29 in conjunction with the eigenvalues given in Lemma A.13. For the (TB) one can choose any starting bit as the code is bit-transitive, e.g. the bit at level 0 in Figure 4.10 (left) and derive constraints on the minimum number of bits that are one at levels 2 and 4 such that the constraints at level 1 and 3 are fulfilled. □

Chapter 6

Concluding Remarks

6.1 Conclusions

We have shown how factor graphs and the summary-product algorithm can be used to solve the decision problems associated with digital data transmission problems. They offer a unifying framework where different sources, channel codes, and channel models can be combined.

In this thesis we especially focused on the channel coding part. Unfortunately, not every code can be decoded efficiently using the summary-product algorithm: in order that the algorithm performs well, the code must be representable by a factor graph which should adhere to certain rules. We tried to identify such rules and then to construct channel codes accordingly; our main goal was to produce factor graphs having large girth in order to guarantee that the messages are as long as possible independent. We focused on algebraic constructions: such constructions can be advantageous because there are algebraic constructions of graphs that have several extremal properties (which with very high probability cannot be achieved by random constructions) and they can potentially lead to factor graphs having a simple description. Moreover, for certain classes of codes one can give minimum distance lower bounds. A disadvantage of algebraic constructions is that the algebraically constructed graphs, where the codes are derived from, do not exist for all desirable sizes and parameters. One way out of this dilemma is given by codes that also use subcodes that are different from simple parity-check codes: by employing different types of subcodes and also bit nodes of different degrees, a large range of code parameters can be achieved while at the same time the ensemble iterative-decoding threshold might be improved. At the moment, the subcodes have been placed randomly, but future work will aim at placing the subcodes in a sensible algebraic fashion.

6.2 Open Problems

For random ensembles of codes suitable for iterative decoding, one can usually give average results about the minimum distance and other code parameters, but for a specific code out of such an ensemble this is not anymore possible. One key advantage of algebraic constructions of codes that are suitable for iterative decoding is that potentially one can prove something about the minimum distance of the code. But such results are very sparse. Indeed, to the best of our knowledge, the only known practical codes where a tight minimum distance lower bound (or the exact minimum distance) can be given, are codes from finite geometries, but all these codes have a minimum distance that grows slower than the block length. But many ensembles of randomly constructed codes promise a linear growth of the minimum distance.

Nowadays the problem of finding codes that can be decoded iteratively and whose decoding performance is near capacity for bit-error rates of 10^{-5} can be considered as solved from an engineering point of view. There are many random and algebraic constructions and also many iterative decoding algorithms that achieve this goal. (This is so, because these bit-error rates can be simulated.) But there still remains a considerable amount of open problems.

- To find a family of simply describable codes that operate near to channel capacity.
- To find good families of codes of small block length (perhaps with a modified iterative algorithm).
- To find good families of codes of very high rate.

- To be able to say at what SNR a code achieves a bit-error/block-error rate of 10^{-15} with a certain iterative decoding algorithm.

One can hope that in the future somebody will come up with the "Reed-Solomon codes" of iterative decoding. By this we mean a code family that has

- a simple description,
- a large range of possible lengths and possible rates,
- a good minimum distance,
- provable performance results: one can say at what SNR a bit/block error rate of 10^{-15} is achieved with a certain iterative decoding algorithm.

Appendix A

Matrix Groups and Rings

After the repetition of some basic facts from group theory and after reviewing the Legendre symbol we will give the definition of the matrix ring $\mathcal{M}_2(\mathcal{R})$ and of the four matrix groups $\mathrm{GL}_2(\mathbb{F}_q)$, $\mathrm{SL}_2(\mathbb{F}_q)$, $\mathrm{PGL}_2(\mathbb{F}_q)$, and $\mathrm{PSL}_2(\mathbb{F}_q)$. Some subgroups of $\mathrm{PSL}_2(\mathbb{F}_q)$ are given which are useful for our coding purposes. We also list some results concerning matrix norms and multiplicities of eigenvalues for matrices over the field $\mathbb{C}$.

A.1 Definition of a Group

The following definitions are standard in group theory, see e.g. [34, 77].

Definition A.1 (Group) A group $\langle \mathcal{G}, \circ \rangle$ consist of a set $\mathcal{G}$ and a binary operation $\circ$, where

- **(Closure)** For all $g_1, g_2 \in \mathcal{G}$ we have $g_1 \circ g_2 \in \mathcal{G}$.
- **(Associativity)** For all $g_1, g_2, g_3 \in \mathcal{G}$ we have $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$.
- **(Identity, neutral element)** There is an element $e \in \mathcal{G}$ such that $e \circ g = g \circ e = g$ for all $g \in \mathcal{G}$.

- **(Inverse element)** For each $g \in \mathcal{G}$ there is a $g' \in \mathcal{G}$ such that $g \circ g' = g' \circ g = e$. Instead of g' we write g^{-1}.

The size $|\mathcal{G}|$ is called the order of a group; it may be finite or infinite. □

Definition A.2 (Word) If $\mathcal{T}$ is a subset of the group $\langle \mathcal{G}, \circ \rangle$ then a word W over $\mathcal{T}$ is a finite sequence $w_1 \circ \cdots \circ w_n$ such that for each i, $1 \leq i \leq n$, either w_i or w_i^{-1} belongs to $\mathcal{T}$. The word W is *reduced* if and only if $w_{i+1} \neq w_i^{-1}$ for all $i = 1, \ldots, n-1$. □

Definition A.3 (Free group) We call a group $\langle \mathcal{G}, \circ \rangle$ freely generated by a set $\mathcal{T}$ if every element $g \in \mathcal{G}$ can be written in a unique way as a reduced word in $\mathcal{T}$. It follows that there are no non-trivial relations of the form $w_1 \circ \cdots \circ w_n = e$ for $n \geq 1$ such that for each i, $1 \leq i \leq n$, either w_i or w_i^{-1} belongs to $\mathcal{T}$. □

Definition A.4 (Multiplicative group of a ring) If $\mathcal{R} = \langle \mathcal{R}, +, \cdot \rangle$ is a ring, then $\mathcal{R}^* = \langle \mathcal{R}^*, \cdot \rangle$ is its multiplicative group, i.e., the group of all invertible elements of $\mathcal{R}$ under multiplication. In the case where $\mathcal{R}$ is a field, we have $\mathcal{R}^* = \mathcal{R} \setminus \{0\}$. □

A.2 Legendre Symbol

Besides its general importance, the Legendre symbol will be used to define the matrix group $\mathrm{PSL}_2(q)$.

Definition A.5 (Quadratic residues and quadratic nonresidues) An element of a group is a *QR (quadratic residue)* if it can be written as the square of an element of the same group. Otherwise, the element is called a *QNR (quadratic non-residue).* □

Example A.6 (Quadratic residues and quadratic nonresidues) If $q = 17$, the QRs of the multiplicative group $\mathbb{F}_q^*$ of $\mathbb{F}_q$ are:

$$1, 4, 9, 16, 8, 2, 15, 13.$$

The QNRs are:

$$3, 5, 6, 7, 10, 11, 12, 14.$$

□

Definition A.7 (Legendre symbol) The Legendre symbol for an integer m and and a prime n is defined to be

$$\left(\frac{m}{n}\right) \triangleq \begin{cases} 0 & \text{if } n|m \\ +1 & \text{if } m \text{ is a QR modulo } n \\ -1 & \text{if } m \text{ is a QNR modulo } n \end{cases} .$$

□

Theorem A.8 (Gauss) *Let p_1 and p_2 be two distinct prime numbers different from 2. Then*

$$\left(\frac{p_1}{p_2}\right) = \left(\frac{p_2}{p_1}\right) \cdot (-1)^{[(p_1-1)/2][(p_2-1)/2]}.$$

Therefore, if both p_1 or p_2 are equal 3 modulo 4, then $\left(\frac{p_1}{p_2}\right) = -\left(\frac{p_2}{p_1}\right)$, otherwise $\left(\frac{p_1}{p_2}\right) = \left(\frac{p_2}{p_1}\right)$.

A.3 Matrix Rings

Definition A.9 (Matrix ring) Let $\mathcal{M}_2(\mathcal{R})$ be the ring (under addition and multiplication) of 2×2–matrices with entries in the ring $\mathcal{R}$. □

$$\begin{tikzcd}
\mathrm{GL}_2(\mathbb{F}_q) \arrow[r, "(1)"] \arrow[d, "(2)"'] & \mathrm{SL}_2(\mathbb{F}_q) \arrow[d, "(3)"'] \\
\mathrm{PGL}_2(\mathbb{F}_q) \arrow[r, "(4)"] & \mathrm{PSL}_2(\mathbb{F}_q)
\end{tikzcd}$$

Figure A.1: The relations between the groups $\mathrm{GL}_2(\mathbb{F}_q)$, $\mathrm{SL}_2(\mathbb{F}_q)$, $\mathrm{PGL}_2(\mathbb{F}_q)$, $\mathrm{PSL}_2(\mathbb{F}_q)$ (see text for explanations).

Type of matrix	Constraints	# Possibilities
$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$	$a \neq 0$ arbitrary, b, c arbitrary and $d \neq a^{-1}bc$	$(q-1)q^2(q-1)$
$\begin{pmatrix} 0 & b \\ c & d \end{pmatrix}$	$b \neq 0$, $c \neq 0$ arbitrary, d arbitrary	$(q-1)^2 q$
total		$(q^2-1)(q-1)q$

Table A.1: A possible listing of the elements of $\mathrm{GL}_2(\mathbb{F}_q)$.

Type of matrix	Constraints	# Possibilities
$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$	$a \neq 0$ arbitrary, b, c arbitrary and $d = a^{-1}(1+bc)$	$(q-1)q^2$
$\begin{pmatrix} 0 & b \\ c & d \end{pmatrix}$	$b \neq 0$ arbitrary, d arbitrary and $c = -b^{-1}$	$(q-1) \cdot q$
total		$q(q^2-1)$

Table A.2: A possible listing of the elements of $\mathrm{SL}_2(\mathbb{F}_q)$.

Type of matrix	Constraints	# Possibilities
$\begin{pmatrix} 1 & b \\ c & d \end{pmatrix}$	b, c arbitrary and $d \neq 0$ arbitrary	$q^2(q-1)$
$\begin{pmatrix} 0 & 1 \\ c & d \end{pmatrix}$	d arbitrary and $c \neq 0$ arbitrary	$q(q-1)$
total		$q(q^2-1)$

Table A.3: A possible listing of the elements of $\mathrm{PGL}_2(\mathbb{F}_q)$.

Type of matrix	Constraints	# Possibilities
$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$	$a = 1, \ldots, \frac{q-1}{2}$ arbitrary, b, c arbitrary and $d = a^{-1}(1+bc)$	$\frac{q-1}{2} \cdot q^2$
$\begin{pmatrix} 0 & b \\ c & d \end{pmatrix}$	$b = 1, \ldots, \frac{q-1}{2}$ arbitrary, d arbitrary and $c = -b^{-1}$	$\frac{q-1}{2} \cdot q$
total		$\frac{q(q^2-1)}{2}$

Table A.4: A possible listing of the elements of $\mathrm{PSL}_2(\mathbb{F}_q)$ (when seen as a factor group of $\mathrm{SL}_2(\mathbb{F}_q)$). Here we assume q to be an odd prime such that $\mathbb{F}_q$ is isomorphic to $\mathbb{Z}/q\mathbb{Z}$.

A.4 Matrix Groups

In this section we list various matrix groups over finite fields where the group operation is the usual matrix multiplication. Basically one could consider $n \times n$–matrices for any positive n but we will focus on the case of 2×2–matrices, i.e, $\mathrm{GL}_2(\mathbb{F}_q)$, $\mathrm{SL}_2(\mathbb{F}_q)$, $\mathrm{PGL}_2(\mathbb{F}_q)$, $\mathrm{PSL}_2(\mathbb{F}_q)$. In the case of $\mathrm{PSL}_2(\mathbb{F}_q)$ we impose the additional constraint that q is an odd prime.[1]

The relations between the four different groups can be summarized with the diagram in Fig. A.1. The transitions (1), (2), (3), and (4) will be discussed in the subsequent sections.

A.4.1 The Group $\mathrm{GL}_2(\mathbb{F}_q)$

The general linear group $\mathrm{GL}_2(\mathbb{F}_q)$ consists of all invertible 2×2 matrices over $\mathbb{F}_q$. The group size is

$$|\mathrm{GL}_2(\mathbb{F}_q)| = (q^2-1) \cdot (q^2-q) = (q^2-1)(q-1)q.$$

The matrices can be listed as shown in Tab. A.1.

A.4.2 The Group $\mathrm{SL}_2(\mathbb{F}_q)$

The special linear group $\mathrm{SL}_2(\mathbb{F}_q)$ consists of all invertible 2×2 matrices over $\mathbb{F}_q$ which have determinant 1. There are

$$|\mathrm{SL}_2(\mathbb{F}_q)| = \frac{(q^2-1) \cdot (q^2-q)}{(q-1)} = q(q^2-1)$$

such matrices. They can be listed as shown in Tab. A.2. Transition (1) in the diagram in Fig. A.1 means that $\mathrm{SL}_2(\mathbb{F}_q)$ is the subgroup consisting of matrices of $\mathrm{GL}_2(\mathbb{F}_q)$ having determinant 1. Note that this group is closed under multiplication as from $\det(\mathbf{A}) = 1$ and $\det(\mathbf{B}) = 1$ it follows that $\det(\mathbf{A} \cdot \mathbf{B}) = \det(\mathbf{A}) \cdot \det(\mathbf{B}) = 1 \cdot 1 = 1$. It is also closed under inversion as from $\det(\mathbf{A}) = 1$ it follows that $\det(\mathbf{A}^{-1}) = (\det(\mathbf{A}))^{-1} = 1^{-1} = 1$.

[1] Basically, $\mathrm{PSL}_2(\mathbb{F}_q)$ can be defined for any even or odd prime power q, but we will not need these groups.

A.4.3 The Group $\mathrm{PGL}_2(\mathbb{F}_q)$

The projective general linear group $\mathrm{PGL}_2(\mathbb{F}_q)$ consists of all invertible 2×2 matrices over $\mathbb{F}_q$, modulo the normal subgroup of all non-zero multiples of the identity matrix. There are

$$|\mathrm{PGL}_2(\mathbb{F}_q)| = \frac{(q^2-1)\cdot(q^2-q)}{(q-1)} = q(q^2-1)$$

such matrices. They can be listed as shown in Tab. A.3. Transition (2) in the diagram in Fig. A.1 can be seen as taking a factor group: $\mathrm{PGL}_2(\mathbb{F}_q) \cong \mathrm{GL}_2(\mathbb{F}_q)/\{a\mathbf{I} \,|\, a \in \mathbb{F}_q^*\}$. Group elements should therefore be written as $\mathbf{A}\cdot\{a\mathbf{I} \,|\, a \in \mathbb{F}_q^*\}$, but for easiness of notation we will always let one element be a representative of the whole set, e.g. let $c \in \mathbb{F}_q^*$, then $c \cdot \mathbf{A}$ is a representative of $\mathbf{A}\cdot\{a\mathbf{I} \,|\, a \in \mathbb{F}_q^*\}$.

Let now q be an odd prime such that $\mathbb{F}_q \cong \mathbb{Z}/q\mathbb{Z}$. For the matrices of size 2×2, i.e. the size of matrices we are interested in, the Legendre symbol $\left(\frac{\det(\mathbf{A})}{q}\right)$ is well defined. This follows from the fact that

$$\left(\frac{\det(c\cdot\mathbf{A})}{q}\right) = \left(\frac{c^2\cdot\det(\mathbf{A})}{q}\right) = \left(\frac{\det(\mathbf{A})}{q}\right)$$

for any $c \in \mathbb{F}_q^*$.

A.4.4 The Group $\mathrm{PSL}_2(\mathbb{F}_q)$

In this section, we assume q to be an odd prime, i.e. $\mathbb{F}_q \cong \mathbb{Z}/q\mathbb{Z}$. There are two ways to define the projective special linear group $\mathrm{PSL}_2(\mathbb{F}_q)$, which lead to two isomorphic groups.

- Transition (3) in the diagram in Fig. A.1 corresponds to taking a factor group: $\mathrm{PSL}_2(\mathbb{F}_q) \cong \mathrm{SL}_2(\mathbb{F}_q)/\{+\mathbf{I}, -\mathbf{I}\}$.

- Transition (4) in the diagram in Fig. A.1 corresponds to defining $\mathrm{PSL}_2(\mathbb{F}_q)$ as the subgroup of those matrices $\mathbf{A}$ of $\mathrm{PGL}_2(\mathbb{F}_q)$ whose determinant is a square modulo q, i.e., $\left(\frac{\det(\mathbf{A})}{q}\right) = +1$. This is well defined because of the following two reasons.
  - The expression $\left(\frac{\det(\mathbf{A})}{q}\right)$ is well defined for any $\mathbf{A} \in \mathrm{PGL}_2(\mathbb{F}_q)$ as we have seen in Sec. A.4.3.
  - For $\mathbf{A} \in \mathrm{PGL}_2(\mathbb{F}_q)$ with $\left(\frac{\det(\mathbf{A})}{q}\right) = +1$ and $\mathbf{B} \in \mathrm{PGL}_2(\mathbb{F}_q)$ with $\left(\frac{\det(\mathbf{B})}{q}\right) = +1$ we have

$$\begin{aligned}\left(\frac{\det(\mathbf{A}\cdot\mathbf{B})}{q}\right) &= \left(\frac{\det(\mathbf{A})\cdot\det(\mathbf{B})}{q}\right) = \left(\frac{\det(\mathbf{A})}{q}\right)\cdot\left(\frac{\det(\mathbf{B})}{q}\right)\\ &= (+1)\cdot(+1) = +1.\end{aligned}$$

  $\mathrm{PSL}_2(\mathbb{F}_q)$ is therefore an index-2 subgroup of $\mathrm{PGL}_2(\mathbb{F}_q)$. (Because every index-2 subgroup is a normal subgroup, $\mathrm{PSL}_2(\mathbb{F}_q)$ is also a normal subgroup of $\mathrm{PGL}_2(\mathbb{F}_q)$.)

There are

$$|\mathrm{PSL}_2(\mathbb{F}_q)| = \frac{(q^2-1)\cdot(q^2-q)}{(q-1)\cdot 2} = \frac{1}{2}q(q^2-1)$$

matrices in $\mathrm{PSL}_2(\mathbb{F}_q)$. When considering $\mathrm{PSL}_2(\mathbb{F}_q)$ as a factor group of $\mathrm{SL}_2(\mathbb{F}_q)$, then these matrices can be listed as shown in Tab. A.4. To be precise, the matrices $\mathbf{A}$ are representatives for the sets $\mathbf{A}\cdot\{+\mathbf{I}, -\mathbf{I}\}$. On the other hand, when considering $\mathrm{PSL}_2(\mathbb{F}_q)$ as a subgroup of $\mathrm{PSL}_2(\mathbb{F}_q)$ a listing is also possible and — similarly to Sec. A.4.3 — the matrices $\mathbf{A}$ are representatives for the sets $\mathbf{A}\cdot\{a\mathbf{I} \,|\, a \in \mathbb{F}_q^*\}$.

A.4.5 Subgroups of $\mathrm{PSL}_2(\mathbb{F}_q)$ and Their Cosets

Except for $\mathrm{PSL}_2(\mathbb{F}_2)$ (which is isomorphic to the symmetric group S_3) and $\mathrm{PSL}_2(\mathbb{F}_3)$ (which is isomorphic to the alternating group A_4), the group $\mathrm{PSL}_n(\mathbb{F}_q)$ is simple for every $n \geq 2$ and prime power q, i.e. it has no normal subgroups.

In the following we consider only *right* cosets, i.e. if $\mathcal{U}$ is a subgroup of a group $\mathcal{G}$, then cosets are of the form $\mathcal{U}\cdot g$ for some element $g \in \mathcal{G}$.

Let $\mathrm{PSL}_2(\mathbb{F}_q)$ be considered as a subgroup of $\mathrm{PGL}_2(\mathbb{F}_q)$. To motivate the subgroups of $\mathrm{PSL}_2(\mathbb{F}_q)$ that we are about to present, we make the following observations.

- Let $\mathbf{A} \triangleq \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{PSL}_2(\mathbb{F}_q)$. The ratio a/b of the entries a and b of the matrix lies in $\mathbb{P}^1_{\mathbb{F}_q} \triangleq \mathbb{F}_q \cup \{\infty\}$. (Note that the ratio a/b is a well-defined quantity.)

- Similarly, the well-defined ratio c/d lies in $\mathbb{P}^1_{\mathbb{F}_q}$.

- Not every pair of ratios a/b and c/d is possible: it is not difficult to see that out of $(q+1)^2$ only $q(q+1)$ are possible. (The remaining $q+1$ cases are not possible as the determinant would be zero.)

- From the calculation

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \underbrace{\begin{pmatrix} a_1 & b_1 \\ c_1 & d_1 \end{pmatrix}}_{\triangleq \mathbf{A}_1} = \begin{pmatrix} a' & b' \\ c' & d' \end{pmatrix},$$

  we conclude that the fraction $a'/b' \in \mathbb{P}^1_{\mathbb{F}_q}$ is

$$a'/b' = \frac{aa_1 + bc_1}{ab_1 + bd_1} = \frac{a_1\,(a/b) + c_1}{b_1\,(a/b) + d_1} = f_{\mathbf{A}_1}\,(a/b)\,,$$

  and that the fraction $c'/d' \in \mathbb{P}^1_{\mathbb{F}_q}$ is

$$c'/d' = \frac{ca_1 + dc_1}{cb_1 + dd_1} = \frac{a_1\,(c/d) + c_1}{b_1\,(c/d) + d_1} = f_{\mathbf{A}_1}\,(c/d)\,.$$

  We see that the ratio a'/b' is a function only of the entries of $\mathbf{A}_1$ and the ratio a/b, and the ratio c'/d' is a function only of the entries of $\mathbf{A}_1$ and the ratio c/d.

Now we are in a position to give the definition of two subgroups of $\mathrm{PSL}_2(\mathbb{F}_q)$ suitable for our coding purposes. (The choice was based on the simplicity of the description; there are many other subgroups that would be suitable too.)

- **(A first subgroup of** $\mathrm{PSL}_2(\mathbb{F}_q)$**)** The first subgroup is given by

$$\mathcal{U}_1 \triangleq \left\{ \begin{pmatrix} x & 0 \\ y & x^{-1} \end{pmatrix} \middle|\, x \in \mathbb{F}_q^*, y \in \mathbb{F}_q \right\}.$$

  (One readily sees that $\mathbb{1} \in \mathcal{U}_1$ and that $\mathcal{U}_1$ is closed under multiplication.) We have[2] $|\mathcal{U}_1| = q(q-1)/2$. Therefore we have $|\mathrm{PSL}_2(\mathbb{F}_q)|/|\mathcal{U}_1| = q+1$ cosets. We conclude that we have a total of $q+1$ cosets of size $q(q-1)/2$. It is not difficult to see that matrices that are in the same coset have the same ratio of the elements in the first row.

- **(A second subgroup of** $\mathrm{PSL}_2(\mathbb{F}_q)$**)** The second subgroup is given by

$$\mathcal{U}_2 \triangleq \left\{ \begin{pmatrix} x & 0 \\ 0 & x^{-1} \end{pmatrix} \middle|\, x \in \mathbb{F}_q^* \right\}.$$

  (One readily sees that $\mathbb{1} \in \mathcal{U}_2$ and that $\mathcal{U}_2$ is closed under multiplication.) We have[3] $|\mathcal{U}_1| = (q-1)/2$. Therefore we have $|\mathrm{PSL}_2(\mathbb{F}_q)|/|\mathcal{U}_1| = q(q+1)$ cosets. We conclude that we have a total of $q(q+1)$ cosets of size $(q-1)/2$. It is not difficult to see that matrices that are in the same coset have the same ratio of the elements in the first row and the same ratio of the elements in the second row.

A.5 Vector and Matrix Norms

In this section we consider vectors and matrices over the field $\mathbb{C}$. These standard definitions and properties can be found e.g. in [42].

Definition A.10 (L_2 vector norm) The L_2 vector norm of a vector $\mathbf{x}$ of length n over $\mathbb{C}$ is

$$||\mathbf{x}||_2 \triangleq \sqrt{|x_1|^2 + \cdots + |x_n|^2}.$$

□

Definition A.11 (L_2 matrix norm) The L_2 matrix norm (induced by the L_2 vector norm) of a square matrix $\mathbf{M}$ is then defined to be

$$||\mathbf{M}||_2 \triangleq \sup_{||\mathbf{x}||_2 = 1} ||\mathbf{M}\mathbf{x}||_2 = \sqrt{\lambda_{\max}(\mathbf{M}^H\mathbf{M})},$$

where $\lambda_{\max}(\mathbf{M}^H\mathbf{M})$ is the largest eigenvalue of $\mathbf{M}^H\mathbf{M}$. □

[2] Note that (x, y) and $(-x, -y)$ lead to the same group element in $\mathrm{PSL}_2(\mathbb{F}_q)$.
[3] Note that x and $-x$ lead to the same group element in $\mathrm{PSL}_2(\mathbb{F}_q)$.

Lemma A.12 (Properties of the L_2 matrix norm) *It can be shown that for square matrices* $\mathbf{M}$ *and* $\mathbf{N}$ *one has*

$$||\mathbf{M}||_2 = \sqrt{\lambda_{\max}(\mathbf{M}^H\mathbf{M})}, \tag{A.1}$$

$$||\mathbf{M}+\mathbf{N}||_2 \le ||\mathbf{M}||_2 + ||\mathbf{N}||_2 \quad \textit{(triangle inequality)}, \tag{A.2}$$

$$||\mathbf{M}\cdot\mathbf{N}||_2 \le ||\mathbf{M}||_2 \cdot ||\mathbf{N}||_2. \tag{A.3}$$

A.6 Multiplicities of Eigenvalues

In this section the matrices are over the field $\mathbb{C}$. The following lemma is e.g. given in [30]; this lemma is the main tool in that paper.

Lemma A.13 *Let* $\mathbf{M}$ *be a diagonalizable matrix of size* $n \times n$ *with* ℓ *different eigenvalues* $\lambda_1, \ldots, \lambda_\ell$ *with (algebraic and geometric) multiplicities* $\nu_1, \ldots, \nu_\ell$. *Then the minimal polynomial of* $\mathbf{M}$ *is* $f(X) = \prod_{i=1}^{\ell}(X - \lambda_i)$, *i.e.* $f(\mathbf{M}) = \mathbf{0}$. *Furthermore, the multiplicities are* $\nu_i = \mathrm{trace}(f_i(\mathbf{M}))/f_i(\lambda_i)$ *where* $f_i(X) = f(X)/(X - \lambda_i)$ *for* $i = 1, \ldots, \ell$.

Proof: We omit this proof. □

Appendix B

Quaternion Groups, Rings, and Fields

We define and discuss different rings and fields that are related to quaternions. Sec. B.2 focuses on so-called integral quaternions, whereas Sec. B.3 treats isomorphisms between quaternion rings (groups) and matrix rings (groups). Some results are extensions of definitions used in [65] in order to get a larger class of Ramanujan graphs. Note that in order to distinguish the usual primes from the quaternion primes (that appear only in this appendix) we shall call them here the rational primes, see also Def. B.9.

B.1 Definition of Quaternions

Definition B.1 (Quaternion field, see e.g. [25])

- The (skew) field of quaternions $\langle \mathbb{H}(\mathbb{C}), +, \cdot \rangle$ over $\mathbb{C}$ is formed of all quaternions[1]

$$\mathbb{H}(\mathbb{C}) \triangleq \left\{\alpha = \alpha_0 + \alpha_1\mathbf{i} + \alpha_2\mathbf{j} + \alpha_3\mathbf{k} \,|\, \alpha_i \in \mathbb{C}\right\},$$

[1] The notation $\mathbb{H}(\cdot)$ honors their discoverer Hamilton.

where the units $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ satisfy the relations

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1,\ \mathbf{ij} = \mathbf{k} = -\mathbf{ji},\ \mathbf{jk} = \mathbf{i} = -\mathbf{kj},\ \mathbf{ki} = \mathbf{j} = -\mathbf{ik}.$$

- The conjugate of α is

$$\overline{\alpha} \triangleq \alpha_0 - \alpha_1\mathbf{i} - \alpha_2\mathbf{j} - \alpha_3\mathbf{k},$$

- The norm $N(\alpha)$ of α is

$$N(\alpha) \triangleq \alpha\overline{\alpha} = \overline{\alpha}\alpha = \alpha_0^2 + \alpha_1^2 + \alpha_2^2 + \alpha_3^2.$$

□

Lemma B.2 *Let α and β be two quaternions, then*

$$N(\alpha\beta) = N(\alpha) \cdot N(\beta).$$

Furthermore, the associative law holds, i.e. $(\alpha\beta)\gamma = \alpha(\beta\gamma)$ for three quaternions α, β, and γ. It follows that the (multiplicative) inverse α^{-1} of a quaternion α is

$$\alpha^{-1} = \big(N(\alpha)\big)^{-1} \cdot \overline{\alpha}.$$

Definition B.3 (Integral quaternions, see e.g. [25]) We call a quaternion an *integral quaternion* if all its coordinates are integers, i.e.

$$\mathbb{H}(\mathbb{Z}) = \big\{\alpha = \alpha_0 + \alpha_1\mathbf{i} + \alpha_2\mathbf{j} + \alpha_3\mathbf{k} \,|\, \alpha_i \in \mathbb{Z}\big\}.$$

We call such a quaternion *odd* if its norm is odd. An integral quaternion whose norm is unity is called a *unit*. There are only eight units, namely ± 1, $\pm\mathbf{i}$, $\pm\mathbf{j}$, $\pm\mathbf{k}$. A quaternion is said to be *associated* with its products by the eight units. Moreover, two integral quaternions α and β are equal modulo an integer m if and only if $\alpha_i = \beta_i \,(\mathrm{mod}\ m)$, for $i = 0, 1, 2, 3$ and we write $\alpha = \beta \,(\mathrm{mod}\ m)$. □

Note that $\langle \mathbb{H}(\mathbb{C}), +, \cdot\rangle$ is a non-commutative field (i.e. a skew field), whereas $\langle \mathbb{H}(\mathbb{Z}), +, \cdot\rangle$ is a non-commutative ring.

We introduce now some special families of integral quaternions[2], namely those having norm p, where p is an odd (rational) prime.[3]

[2] The families for $p = 1\,(\mathrm{mod}\ 4)$ were given in [65], we extend the definitions here so that also $p = 3\,(\mathrm{mod}\ 4)$ is allowed.

[3] For the meaning of a rational prime, see the introductory paragraph and Def. B.9.

Definition B.4 (Some special families of integral quaternions) If p is an odd (rational) prime then let

$$\mathcal{F}_p' \triangleq \big\{\alpha \in \mathbb{H}(\mathbb{Z}) \,|\, N(\alpha) = \alpha_0^2 + \alpha_1^2 + \alpha_2^2 + \alpha_3^2 = p\big\}.$$

- If $p = 1\,(\mathrm{mod}\ 4)$ then let

$$\mathcal{F}_p \triangleq \left\{\alpha \in \mathbb{H}(\mathbb{Z}) \,\middle|\, \begin{array}{c} N(\alpha) = \alpha_0^2 + \alpha_1^2 + \alpha_2^2 + \alpha_3^2 = p, \\ \alpha_0 > 0 \text{ is odd}, \alpha_1, \alpha_2, \alpha_3 \text{ are even} \end{array}\right\}.$$

- If $p = 3\,(\mathrm{mod}\ 4)$ then let

$$\mathcal{F}_p \triangleq \left\{\alpha \in \mathbb{H}(\mathbb{Z}) \,\middle|\, \begin{array}{c} N(\alpha) = \alpha_0^2 + \alpha_1^2 + \alpha_2^2 + \alpha_3^2 = p, \\ \alpha_0 > 0, \alpha_1, \alpha_2 \text{ are odd}, \alpha_3 \text{ is even} \end{array}\right\}.$$

□

The set $\mathcal{F}_p'$ can be considered as the set of prime quaternions of $\mathbb{H}(\mathbb{Z})$ which have norm p.

Lemma B.5 *Let p be an odd (rational) prime as in Def. B.4. Then,*

$$|\mathcal{F}_p'| = 8(p+1), \qquad |\mathcal{F}_p| = (p+1),$$
$$\mathcal{F}_p' = \big\{\varepsilon\alpha \,|\, \varepsilon \in \{\pm 1, \pm\mathbf{i}, \pm\mathbf{j}, \pm\mathbf{k}\}, \alpha \in \mathcal{F}_p\big\}.$$

Moreover, every element of $\mathcal{F}_p'$ can be written in a unique way as a product of an unit and an element of $\mathcal{F}_p$.

Proof: To find the size of the set $\mathcal{F}_p'$ we have to count the solutions of the diophantine equation $\alpha_0^2 + \alpha_1^2 + \alpha_2^2 + \alpha_3^2 = p$. A theorem by Jacobi states that the diophantine equation $\alpha_0^2 + \alpha_1^2 + \alpha_2^2 + \alpha_3^2 = n$ has $8\sum_{d|n, 4\nmid d} d$ solutions. Setting $n = p$ gives the desired result about $|\mathcal{F}_p'|$.

Let $p = 1\,(\mathrm{mod}\ 4)$. By considering the equation $\alpha_0^2 + \alpha_1^2 + \alpha_2^2 + \alpha_3^2 = p$ modulo 4, one sees that exactly one out of the α_i's must be odd. From this observation it is not difficult to see that every element in $\mathcal{F}_p'$ can be written in an unique way as the product of an unit and an element from $\mathcal{F}_p$. As there are eight units, $|\mathcal{F}_p| = |\mathcal{F}_p'|/8 = p + 1$.

Let $p = 3\,(\mathrm{mod}\ 4)$. By considering the equation $\alpha_0^2 + \alpha_1^2 + \alpha_2^2 + \alpha_3^2 = p$ modulo 4, one sees that exactly one out of the α_i's must be even. The rest is as in the case $p = 1\,(\mathrm{mod}\ 4)$. □

B.2 Properties of Quaternions

The paper [25] by Dickson establishes that for odd quaternions one has a theory of greatest common divisors and the usual factorization (on the left and right). In this section, we list some results from [25]; the culmination will be the establishment of Lemma B.22.

Theorem B.6 *Any integral quaternion whose norm is even can be expressed in one and but one way in the form $2^e\pi\gamma$, where π is one of the six quaternions*

$$1, \quad 1+\mathbf{i}, \quad 1+\mathbf{j}, \quad 1+\mathbf{k}, \quad (1+\mathbf{i})(1+\mathbf{j}), \quad (1+\mathbf{i})(1+\mathbf{k}),$$

while γ is an odd quaternion, and $e > 0$ if $\pi = 1$.

Proof: See [25]. □

In the next theorem, it is important that β is an *odd* quaternion.

Theorem B.7 *If α is any integral quaternion and β is any* odd *quaternion, we can find integral quaternions $\xi^{(1)}$, $\gamma^{(1)}$, $\xi^{(2)}$, and $\gamma^{(2)}$ such that*

$$\alpha = \xi^{(1)}\beta + \gamma^{(1)}, \quad N(\gamma^{(1)}) < N(\beta),$$
$$\alpha = \beta\xi^{(2)} + \gamma^{(2)}, \quad N(\gamma^{(2)}) < N(\beta).$$

Proof: See [25]. □

Theorem B.8 *Any two integral quaternions α and β, at least one of which is odd, have a right-hand greatest common divisor $\delta^{(1)}$ which is uniquely determined up to a unit factor. Also*

$$\delta^{(1)} = \alpha^{(1)}\alpha + \beta^{(1)}\beta,$$

where $\alpha^{(1)}$ and $\beta^{(1)}$ are integral quaternions. Similarly, there is a left-hand greatest common divisor, unique up to unit factor, for which

$$\delta^{(2)} = \alpha\alpha^{(2)} + \beta\beta^{(2)},$$

where $\alpha^{(2)}$ and $\beta^{(2)}$ are integral quaternions.

Proof: See [25]. □

Definition B.9 (Prime quaternion) An integral quaternion, not a unit, is called *prime* if it admits only such representations as a product of two integral quaternions in which one of the two factors is a unit. Note that the primes in $\mathbb{Z}$ are called *rational primes* in the following. □

Theorem B.10 *Every rational prime p is a product of two integral quaternions neither of which is a unit, so that p is not a prime quaternion.*

Proof: See [25]. □

Theorem B.11 *If $N(\pi)$ is a (rational) prime number, π is a prime quaternion and conversely.*

Proof: See [25]. □

Theorem B.12 *Every rational prime p is a product of two conjugate prime quaternions, and all prime quaternions arise as factors of rational primes.*

Proof: See [25]. □

Corollary B.13 *Every positive integer is a sum of four integral squares.*

Theorem B.14 *If ζ is any odd quaternion and $N(\zeta) = pqr\ldots$, where $p, q, r, \ldots$ are the (equal or distinct) prime factors of $N(\zeta)$ arranged in an arbitrary, but definite, order, then $\zeta = \alpha\beta\gamma\ldots$, where $\alpha, \beta, \gamma, \ldots$, are prime quaternions whose norms are $p, q, r, \ldots$, respectively. This decomposition is unique apart from the association of unit factors.*

Proof: See [25], but see also Remark B.15. □

Remark B.15 (on Th. B.14) We think that Th. B.14 (taken from [25]) is wrong as stated above. A contradictory example is $17 = (4+\mathbf{i})(4-\mathbf{i}) = (3+2\mathbf{i}+2\mathbf{j})(3-2\mathbf{i}-2\mathbf{j})$, but $4+\mathbf{i}$ is obviously neither a unit times $3+2\mathbf{i}+2\mathbf{j}$ nor a unit times $3-2\mathbf{i}-2\mathbf{j}$. In our opinion, uniqueness can only be shown when considering reduced expressions, see e.g. Th. B.16. □

Theorem B.16 *Let p be a odd (rational) prime. Every $\alpha \in \mathbb{H}(\mathbb{Z})$ with $N(\alpha) = p^k$ can be expressed uniquely in the form*

$$\alpha = \varepsilon \cdot p^r \cdot R_m\left(\mathcal{F}_p\right),$$

where ε is a unit, $2r + m = k$, and R_m is a reduced word of length m consisting of elements of $\mathcal{F}_p$, where $\mathcal{F}_p$ was defined in Def. B.4.

Proof: See [41] or [65]. □

Corollary B.17 *Let α be an integral quaternion with $\alpha = 1\,(\text{mod } 2)$ and $N(\alpha) = p^k$ then*

$$\alpha = \pm p^r R_m\left(\mathcal{F}_p\right),$$

with $2r + m = k$, and this representation is unique.

We can use Dickson's proof idea for Th. B.14 to prove Lemmas B.18 and B.19; in these cases this proof technique works.

Lemma B.18 *Let p_1 and p_2 be two distinct odd primes. Factoring the quaternion p_1p_2 as $\alpha_1 \cdot \alpha_2 \cdot \beta_1 \cdot \beta_2$, where $\alpha_i \in \mathcal{F}_{p_1}$ and $\beta_i \in \mathcal{F}_{p_2}$, we must have $\alpha_2 = \overline{\alpha_1}$ and $\beta_2 = \overline{\beta_1}$.*

Proof: Let α_1 be any element of $\mathcal{F}_{p_1}$; obviously, α_1 divides the quaternion p_1p_2 from the left. Let α_2 be the left-hand greatest common divisor of $\alpha_1^{-1}p_1p_2 = \alpha_2\beta_1\beta_2$ and p_1; therefore $N(\alpha_2) = p_1$. $\alpha_2 \triangleq \overline{\alpha_1}$ fulfills the requirement that $N(\alpha_2) = p_1$ and by the uniqueness of the greatest common divisor (up to a unit factor) this is the only solution fulfilling $\alpha_2 \in \mathcal{F}_{p_1}$. Continuing, one shows that for any $\beta_1 \in \mathcal{F}_{p_2}$ one has $\beta_2 = \overline{\beta_1}$. □

Lemma B.19 *Let p_1 and p_2 be two distinct odd primes. Factoring the quaternion $p_1p_2^2$ as $\alpha_1 \cdot \alpha_2 \cdot \beta_1 \cdot \beta_2 \cdot \beta_3 \cdot \beta_4$, where $\alpha_i \in \mathcal{F}_{p_1}$ and $\beta_i \in \mathcal{F}_{p_2}$, we must have $\alpha_2 = \overline{\alpha_1}$.*

Proof: The proof of $\alpha_2 = \overline{\alpha_1}$ proceeds along the same line as the proof to Lemma B.18. □

Definition B.20 (The semi-group $\Lambda'(2)$) Let p be an odd (rational) prime.[4]

- If $p = 1\,(\text{mod } 4)$, let

$$\Lambda'(2) \triangleq \{\alpha \in \mathbb{H}(\mathbb{Z}) \mid \alpha = 1\,(\text{mod } 2), N(\alpha) = p^\nu, \nu \in \mathbb{Z}\}.$$

- If $p = 3\,(\text{mod } 4)$, let

$$\Lambda'(2) \triangleq \left\{ \alpha \in \mathbb{H}(\mathbb{Z}) \;\middle|\; \begin{array}{l} \alpha = 1\,(\text{mod } 2), N(\alpha) = p^\nu, \text{even } \nu \in \mathbb{Z} \\ \alpha = 1 + \mathbf{i} + \mathbf{j}\,(\text{mod } 2), N(\alpha) = p^\nu, \text{odd } \nu \in \mathbb{Z} \end{array} \right\}.$$

[4] The case $p = 1\,(\text{mod } 4)$ was given in [65]. We extend it to the case $p = 3\,(\text{mod } 4)$.

It is not difficult to see that $\Lambda'(2)$ is closed under multiplication. □

Definition B.21 (The group $\Lambda(2)$) Let $\Lambda(2) \triangleq \{[\alpha] \mid \alpha \in \Lambda'(2)\}$, where we identify two elements $[\alpha]$ and $[\beta]$ whenever $\pm p^{\nu_1}\alpha = p^{\nu_2}\beta$, $\nu_1, \nu_2 \in \mathbb{Z}$. $\Lambda(2)$ is therefore a group where $[\alpha][\beta] = [\alpha\beta]$ and $[\alpha][\overline{\alpha}] = [1]$. □

Lemma B.22 *The group $\Lambda(2)$ is freely generated by $\{[\alpha] \mid \alpha \in \mathcal{F}_p\}$.*

Proof: This result is an application of the unique representation of a specific set of integral quaternions in Cor. B.17 and the definition of the group $\Lambda(2)$ in Def. B.21. □

B.3 Isomorphisms

The theorems in this section could be stated more generally, but they are sufficient for our needs in the main text.

Assumption B.23 *Throughout this section we assume q to be an odd (rational) prime such that we have the field isomorphism*

$$\mathbb{F}_q \cong \mathbb{Z}/q\mathbb{Z}.$$

We need a lemma already known to Euler.

Lemma B.24 *For any (rational) prime p, the integer -1 can be written as the sum of two squares modulo p.*

Proof: The case $p = 2$ is trivially true; therefore we can assume p to be an odd (rational) prime. We have to find integers x and y such that $x^2 + y^2 = -1\,(\text{mod } p)$. If -1 is a QR modulo p, we may take $y = 0$. Henceforth, let -1 be a QNR modulo p. Let a be the first QR of p among the terms $p-1, p-2, p-3, \ldots$. Then $a + 1 = b$ is a QNR. Hence there exist integers x and y for which $a = x^2\,(\text{mod } p)$ and $-b = y^2\,(\text{mod } p)$. $x^2 + y^2 = -1\,(\text{mod } p)$ is then equivalent to $a - b = -1\,(\text{mod } p)$. □

Now we discuss isomorphisms that were stated in [65] for the case $q = 1\,(\text{mod } 4)$. We mention also the case $q = 3\,(\text{mod } 4)$, which is certainly known in the corresponding algebra literature.

Theorem B.25 *Let $\mathcal{M}_2(\mathbb{Z}/q\mathbb{Z})$ be the ring (under addition and multiplication) of 2×2–matrices with entries in $\mathbb{Z}/q\mathbb{Z}$ where q is an odd prime.*

- *If -1 is a QR modulo q, then the mapping*

$$\begin{aligned}\tilde{\varphi}:\mathbb{H}(\mathbb{Z}/q\mathbb{Z}) &\to \mathcal{M}_2(\mathbb{Z}/q\mathbb{Z})\\ \alpha_0+\alpha_1\mathbf{i}+\alpha_2\mathbf{j}+\alpha_3\mathbf{k} &\mapsto \begin{pmatrix}\alpha_0+\alpha_1 u & \alpha_2+\alpha_3 u\\ -\alpha_2+\alpha_3 u & \alpha_0-\alpha_1 u\end{pmatrix},\end{aligned}$$

  where u must be chosen such that $u^2=-1\,(\mathrm{mod}\ q)$, is a ring isomorphism.

- *If -1 is a QNR modulo q, then the mapping*

$$\begin{aligned}\tilde{\varphi}:\mathbb{H}(\mathbb{Z}/q\mathbb{Z}) &\to \mathcal{M}_2(\mathbb{Z}/q\mathbb{Z})\\ \alpha_0+\alpha_1\mathbf{i}+\alpha_2\mathbf{j}+\alpha_3\mathbf{k} &\mapsto \begin{pmatrix}\alpha_0+\alpha_1 u+\alpha_3 v & \alpha_2+\alpha_3 u-\alpha_1 v\\ -\alpha_2+\alpha_3 u-\alpha_1 v & \alpha_0-\alpha_1 u-\alpha_3 v\end{pmatrix},\end{aligned}$$

  where u and v must be chosen such that $u^2+v^2=-1\,(\mathrm{mod}\ q)$, is a ring isomorphism.

Note that the mapping in the first case is a special case of the mapping in the second case by choosing $v=0$.

Proof: If -1 is a QR then there exists by definition an u such that $u^2=-1\,(\mathrm{mod}\ q)$, whereas if -1 is a QNR then there exist u and v such that $u^2+v^2=-1\,(\mathrm{mod}\ q)$ by Lemma B.24. To show that both mappings are ring homomorphisms one shows that indeed $\tilde{\varphi}(\alpha+\beta)=\tilde{\varphi}(\alpha)+\tilde{\varphi}(\beta)$ and that $\tilde{\varphi}(\alpha\beta)=\tilde{\varphi}(\alpha)\tilde{\varphi}(\beta)$. To show that $\tilde{\varphi}$ is an isomorphism one shows the existence of $\tilde{\varphi}^{-1}$, which is easily done. □

Corollary B.26 *Using the isomorphisms of Th. B.25, we have*

$$\begin{aligned}N(\alpha) &= \det\big(\tilde{\varphi}(\alpha)\big)\ \ (\mathrm{mod}\ q),\\ \tilde{\varphi}(\overline{\alpha}) &= \mathrm{adj}\big(\tilde{\varphi}(\alpha)\big)\ \ (\textit{in } \mathcal{M}_2(\mathbb{Z}/q\mathbb{Z})),\end{aligned}$$

for any $\alpha\in\mathbb{H}(\mathbb{Z})$, where $\mathrm{adj}(\mathbf{M})$ is the adjungated matrix of $\mathbf{M}$, i.e., $\mathrm{adj}(\mathbf{M})=\det(\mathbf{M})\cdot\mathbf{M}^{-1}$.

Proof: Calculating $\det(\tilde{\varphi}(\alpha))$ shows that this is indeed equal modulo q to the norm of a quaternion which by definition is $N(\alpha)=\alpha_0^2+\alpha_1^2+\alpha_2^2+\alpha_3^2$. The second assertion follows equally easily. □

Theorem B.27 *Let q be an odd (rational) prime and let $\mathcal{Z}\le\mathbb{H}(\mathbb{Z}/q\mathbb{Z})^*$ be the central subgroup[5] of $\mathbb{H}(\mathbb{Z}/q\mathbb{Z})^*$, i.e. $\mathcal{Z}=(\mathbb{Z}/q\mathbb{Z})^*$.*

- *If -1 is a QR modulo q, then the mapping*

$$\begin{aligned}\varphi:\mathbb{H}(\mathbb{Z}/q\mathbb{Z})^*/\mathcal{Z} &\to \mathrm{PGL}_2(\mathbb{Z}/q\mathbb{Z})\\ \alpha_0+\alpha_1\mathbf{i}+\alpha_2\mathbf{j}+\alpha_3\mathbf{k} &\mapsto \begin{pmatrix}\alpha_0+\alpha_1 u & \alpha_2+\alpha_3 u\\ -\alpha_2+\alpha_3 u & \alpha_0-\alpha_1 u\end{pmatrix},\end{aligned}$$

  where u must be so chosen such that $u^2=-1\,(\mathrm{mod}\ q)$, is an isomorphism.

- *If -1 is a QNR modulo q, then the mapping*

$$\begin{aligned}\varphi:\mathbb{H}(\mathbb{Z}/q\mathbb{Z})^*/\mathcal{Z} &\to \mathrm{PGL}_2(\mathbb{Z}/q\mathbb{Z})\\ \alpha_0+\alpha_1\mathbf{i}+\alpha_2\mathbf{j}+\alpha_3\mathbf{k} &\mapsto \begin{pmatrix}\alpha_0+\alpha_1 u+\alpha_3 v & \alpha_2+\alpha_3 u-\alpha_1 v\\ -\alpha_2+\alpha_3 u-\alpha_1 v & \alpha_0-\alpha_1 u-\alpha_3 v\end{pmatrix},\end{aligned}$$

  where u and v must be chosen such that $u^2+v^2=-1\,(\mathrm{mod}\ q)$, is an isomorphism.

Note that the mapping in the first case is a special case of the mapping in the second case by choosing $v=0$.

Proof: Follows easily from Th. B.25. □

Corollary B.28 *Using the isomorphisms of Th. B.27 and the result of Cor B.26, we have for an odd (rational) prime q*

$$\begin{aligned}\left(\frac{N(\alpha)}{q}\right) &= \left(\frac{\det\big(\varphi(\alpha)\big)}{q}\right),\\ \varphi(\overline{\alpha}) &= \mathrm{adj}\big(\varphi(\alpha)\big)\quad (\textit{in } \mathrm{PGL}_2(\mathbb{Z}/q\mathbb{Z})),\end{aligned}$$

for any $\alpha\in\mathbb{H}(\mathbb{Z})$, where $\mathrm{adj}(\mathbf{M})$ is the adjungated matrix of $\mathbf{M}$, i.e., $\mathrm{adj}(\mathbf{M})=\det(\mathbf{M})\cdot\mathbf{M}^{-1}$.

Proof: Follows from Th. B.27 and Cor. B.26. □

[5] We remind the reader that the central subgroup of a group is the largest subgroup such that every element of this subgroup commutes with every element of the group.

Definition B.29 Let the homomorphism

$$\phi : \Lambda(2) \to \mathrm{PGL}_2(\mathbb{Z}/q\mathbb{Z})$$

be the concatenation of the homomorphism

$$\begin{aligned} \Lambda(2) &\to \mathbb{H}(\mathbb{Z}/q\mathbb{Z})^*/\mathcal{Z} \\ \alpha &\mapsto (\alpha \mod q) \end{aligned}$$

and the isomorphism $\varphi : \mathbb{H}(\mathbb{Z}/q\mathbb{Z})^*/\mathcal{Z} \to \mathrm{PGL}_2(\mathbb{Z}/q\mathbb{Z})$ given in Th. B.27. □

Lemma B.30 (Image of ϕ) *The homomorphism ϕ of Def. B.29 has the image*

$$\mathrm{image}(\phi) = \begin{cases} \mathrm{PGL}_2(\mathbb{Z}/q\mathbb{Z}) & \text{if } \left(\frac{p}{q}\right) = -1, \\ \mathrm{PSL}_2(\mathbb{Z}/q\mathbb{Z}) & \text{if } \left(\frac{p}{q}\right) = +1. \end{cases}$$

Proof: The proof for q and p equal to 1 modulo 4 is given in the proof to Prop. 3.3 in [65]. For the other cases, the proof of Prop. 3.3 in [65] can be extended; the details are omitted here. □

Appendix C

About Codes on Trees

C.1 Introduction

Etzion, Trachtenberg, and Vardy gave in [29] upper bounds on the minimum distance $d_{\min}$ of linear codes that can be represented by *cycle-free* Tanner graphs. Note that in this chapter we will always assume that the check nodes are simple parity-check nodes.

Definition C.1 We say that a linear code can be represented by a cycle-free Tanner graph if the Tanner graph associated to a possible parity-check matrix of the code has no cycles. (Equivalently, that code can be represented by a factor graph having no state nodes and no cycles.) □

Theorem C.2 (see [29]) *Assume a linear $[n, k, d_{\min}]$ code with rate $R = k/n$ that can be represented by a cycle-free Tanner graph. Then,*

$$d_{\min} \leq \begin{cases} \left\lfloor \frac{n}{k+1} \right\rfloor + \left\lfloor \frac{n+1}{k+1} \right\rfloor < \frac{2}{R} & (R < 1/2) \\ 2 & (R \geq 1/2) \end{cases}.$$

The aim of this appendix is to give a simpler derivation of this result. In [29] the derivations were made using combinatorial considerations about the parity-check matrix $\mathbf{H}$ whereas our proof is based on graph-theoretic considerations. Notice also that their proof is based on induction, whereas our proof is not.

The case $R \geq 1/2$ is anyway not too difficult to prove and for $R < 1/2$ it is actually much easier to prove a slightly weaker version of the bound given in Th. C.2.

Corollary C.3 *Under the same assumptions as in Th. C.2 a slightly weaker bound for $R < 1/2$ is*

$$d_{\min} \leq \left\lfloor \frac{2n}{k+1} \right\rfloor < \frac{2}{R}. \tag{C.1}$$

After the introduction of some notation in Sec. C.2, Sec. C.3 gives the necessary steps to prove Th. C.2. Sec. C.4 gives some additional results, whereas Sec. C.5 collects the proofs to all statements. Only a fraction of the steps is necessary to prove Cor. C.3 (see Th. C.15 and Remarks C.9 and C.16).

C.2 Notation

For the notation used in this appendix we refer to Table C.1.

- All codes that we will consider in this appendix will be *binary*; an extension to non-binary codes is straightforward.
- By a *vertex* in a Tanner graph we either mean a variable node or a check node; furthermore, edges are always between a variable and a check node.
- Generally, *check nodes* in Tanner graphs can be any indicator function but in this chapter – as we have already mentioned at the beginning – we assume to have only the indicator function corresponding to the addition modulo two, see also f_{XOR} in Def. 3.15.
- A *leaf* of a graph is a vertex which has degree 1; a vertex without any neighbors is not a leaf (by definition).
- There is a *one-to-one correspondence* between parity-check matrices and Tanner graphs. To be able to switch back and forth between

Description	Name
Linear code of length n, dimension k, and minimum distance $d_{\min}$	$[n, k, d_{\min}]$
Graph X	X
Set of vertices of a graph X	$V\mathsf{X}$
Set of edges of a graph X	$E\mathsf{X}$
Set of (binary) variable nodes of a graph X	$B\mathsf{X}$
Set of check nodes of a graph X	$C\mathsf{X}$
Number of components of a graph X	$\operatorname{Comp}\mathsf{X}$
Degree of a vertex v (= number of adjacent vertices)	$\deg(v)$
Remainder of an integer a when divided by an integer b	$R_b(a)$

Table C.1: Notation

a code and its representation by a Tanner graph, we will implicitly fix a parity-check matrix representing that code.

C.3 New Path

C.3.1 General Remarks

We make the following observations:

- By codes that can be represented by cycle-free Tanner graphs we mean codes that have a Tanner graph that is a *forest*, i.e. a collection of trees.
- As the *minimum distance* of a code whose graph consists of different components is the minimum of the minimum distances of each component, we will focus on analyzing only codes that can be represented by trees having at least two variable nodes (see Lemma C.17).
- A linear $[n, k, d_{\min}]$ code has redundancy $r \triangleq n - k$.

- If a check node is a leaf then the adjacent variable node is idle, i.e., always zero. As the bound holds for the code where this symbol node is removed (i.e. where the corresponding symbol position in the code is punctured), it is not difficult to see that the bound holds also for the longer code having this idle symbol. Thus we can safely assume that all *leaves are variable nodes*.

- With the above assumption (namely that there are no check nodes which are leaves) we can state that redundant check equations imply cycles in the Tanner graph. A graph having no cycles can therefore have *no redundant check equations*. A linear $[n, k, d_{\min}]$ code having the tree T as Tanner graph has $|B\mathsf{T}| = n$, $|C\mathsf{T}| = r = n-k$, and $|V\mathsf{T}| = |B\mathsf{T}| + |C\mathsf{T}| = n + r = 2n - k$.

- A Tanner graph T which is a tree fulfills $|E\mathsf{T}| = |V\mathsf{T}| - 1$.

- A *valid configuration* is an assignment of values ("0" or "1") to the variable nodes such that all checks are fulfilled; there is a *one-to-one correspondence* between codewords and valid configurations.

Lemma C.4 gives a lower bound on the number of leaves of a Tanner tree; Lemma C.11 will be an extension of this result.

Lemma C.4 *Let $\mathcal{L}$ be the set of leaves (by assumption only variable nodes) of a Tanner tree T corresponding to a code with dimension k and length $n \geq 2$. Independent of the rate of the code, the number of leaves is larger than the dimension of the code, i.e. $|\mathcal{L}| > k$ or equivalently $|\mathcal{L}| \geq k + 1$.*

Proof: See Sec. C.5.1. □

C.3.2 Codes with $R \geq 1/2$

Theorem C.5 (Second Part of Theorem C.2) *A linear $[n, k, d_{\min}]$ code with $R = k/n \geq 1/2$ whose Tanner graph is a tree has $d_{\min} \leq 2$.*

Proof: See Sec. C.5.2. □

C.3.3 Codes with $R < 1/2$

Let T be a Tanner tree of a linear $[n, k, d_{\min}]$ code. The following partitioning of the Tanner graph will be essential to prove Theorem C.14.

Algorithm C.6 (Partitioning of the Tanner Graph) We do this algorithm until all leaves are used up.

- Select a leaf that has not been visited until now.
- To such a leaf we associate all variable nodes that can be reached by passing only check nodes with valency two. All these variable nodes, together with the check nodes and all edges between these variable nodes, form a tree called a "small tree".

□

A graph similar to the one in Figure C.1 results.

Properties C.7 (of the resulting graph of Algorithm C.6)

- *Let the shells 2 and 3 contain the small trees whereby in shell 2 there are only variable nodes of small trees that are adjacent to a check node not in the small tree. Shell 3 contains n_1 variable and r_1 check nodes, respectively (by construction, $n_1 = r_1$).*

- *Let shell 1 contain all check nodes that are adjacent to a small tree. Let the core contain all remaining variable and check nodes.*

- *There are r_2 check nodes in shell 1, let them be numbered from 1 to r_2. Let N_j be the number of adjacent small trees to check node j (therefore, the number of variable nodes in shell 2 is $n_2 = \sum_{j=1}^{r_2} N_j$).*

- *By construction, all the small trees are disjunct and are labeled $\mathsf{T}_i^{(j)}$, where $j \in \{1, \ldots, r_2\}$ is the number of the adjacent check node in shell 1, and where i is a number in the range from 1 to N_j. (Of course, no small tree has the same label.) Let the number of variable nodes in the small tree $\mathsf{T}_i^{(j)}$ be $t_i^{(j)} \triangleq \left|B\mathsf{T}_i^{(j)}\right|$.*

- *Let the tree consisting of all variable and check nodes and edges in the core, shell 1, and shell 2 be called the "inner tree" T' (note that* *there is a slight overlapping of the inner tree with the small trees). As this tree results from the tree* T *by removing* n_1 *variable and* $r_1 = n_1$ *check nodes, the corresponding code has length* $n - n_1$*, redundancy* $r - r_1$*, dimension* $(n - n_1) - (r - r_1) = n - r = k$ *and minimum distance* 2*, i.e. has parameters* $[n - n_1, k, 2]$*.*

- *For a valid configuration, all the values of the variable nodes in one part* $\mathsf{T}_i^{(j)}$ *are the same; this follows from the fact that they are all connected by parity-checks of degree two.*

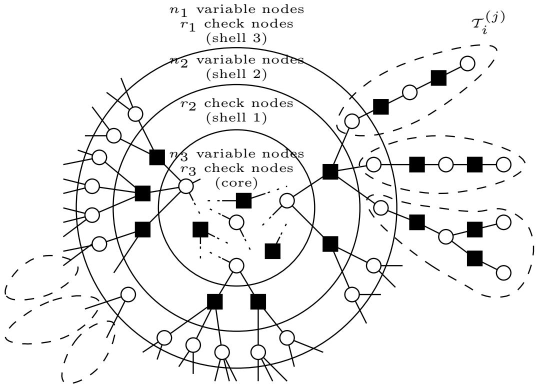

Figure C.1: A code with $R \leq 1/2$ whose Tanner graph is a tree and its partitioning according to Alg. C.6. Circles are variable nodes and squares are check nodes. Small trees are enclosed by dashed lines.

Lemma C.8 *Consider a code whose Tanner graph is a tree and where at least one check node of this tree has degree larger than 2. The minimum distance* $d_{\min}$ *of such a code is upper bounded by* $\min_{(i,i',j):\ i \neq i'} t_i^{(j)} + t_{i'}^{(j)}$*.*

Proof: See Sec. C.5.3. □

Remark C.9 (Weaker Result) If one is only interested in the weaker result in (C.1), one can go directly to Th. C.15 on page 168. □

Lemma C.10 *Trees fulfilling conditions (C.2) and (C.3), i.e.,*

$$0 \leq \left(\max_i t_i^{(j)}\right) - \left(\min_i t_i^{(j)}\right) \leq 1 \quad (\text{for any } 1 \leq j \leq r_2), \tag{C.2}$$

$$0 \leq \left(\max_{(i,j)} t_i^{(j)}\right) - \left(\min_{(i,j)} t_i^{(j)}\right) \leq 1 \quad \begin{pmatrix}\text{where max and min are} \\ \text{over all legal pairs } (i,j)\end{pmatrix}, \tag{C.3}$$

have the largest attainable minimum distance with the given values for n*,* k*,* n_1*,* r_1*, etc.*

Proof: See Sec. C.5.4. □

As mentioned before, Lemma C.11 is a generalization of Lemma C.4, which gives a lower bound on the number of leaves of a Tanner tree.

Lemma C.11 *Let* $\mathcal{L}$ *be the set of leaves and* $\mathcal{I}$ *be the set of variable nodes which are non-leaves (i.e. interior nodes) of a Tanner tree* X *of a code with dimension* k *and length* $n \geq 2$*, i.e.,* $\mathcal{I} = B\mathsf{X} \setminus \mathcal{L}$*. We must have* $|\mathcal{L}| = k + 1 + \Delta$ *and* $|\mathcal{I}| = |C\mathsf{X}| - \Delta - 1$ *with* $\Delta = \sum_{v \in \mathcal{I}} (\deg(v) - 2) \geq 0$*.*

Proof: See Sec. C.5.5. □

Lemma C.12 *For the inner graph* T'*, which corresponds to a code with parameters* $[n - n_1 = n_2 + n_3, k, 2]$*, we must have* $n_2 = k + 1 + \Delta$ *and* $n_3 \geq r_2 - \Delta - 1$ *with* $\Delta \geq 0$*.*

Proof: See Sec. C.5.6. □

Lemma C.13 *Let* a *and* b *be positive integers and let* $a' \triangleq R_b(a)$*. If* a *is written as* b *integer summands which are differing by at most one, we get*

$$a = \underbrace{\left\lceil \frac{a}{b} \right\rceil + \cdots + \left\lceil \frac{a}{b} \right\rceil}_{a' \text{ summands}} + \underbrace{\left\lfloor \frac{a}{b} \right\rfloor + \cdots + \left\lfloor \frac{a}{b} \right\rfloor}_{b - a' \text{ summands}}.$$

Proof: See Sec. C.5.7. □

Theorem C.14 (First Part of Theorem C.2) *A linear* $[n, k, d_{\min}]$ *code with* $R = k/n < 1/2$ *whose Tanner graph is a tree fulfills*

$$d_{\min} \leq \left\lfloor \frac{n}{k+1} \right\rfloor + \left\lfloor \frac{n+1}{k+1} \right\rfloor < \frac{2}{R}. \tag{C.4}$$

Proof: See Sec. C.5.8. But before reading this proof, we encourage the reader to first study the simpler Th. C.15 and its proof. □

Theorem C.15 (Weak form of Theorem C.14) *A linear* $[n, k, d_{\min}]$ *code with* $R = k/n < 1/2$ *whose Tanner graph is a tree fulfills*

$$d_{\min} \leq \left\lfloor \frac{2n}{k+1} \right\rfloor < \frac{2}{R}.$$

Proof: See Sec. C.5.9. □

Remark C.16 (Weaker vs. stronger result) Note that for the proof of Th. C.15 we only needed the development up to Lemma C.8 on page 166. Observe that the difference between the weaker and the stronger upper bounds is 0 or 1. □

Lemma C.17 *The results of Ths. C.5 and C.14 hold also for forests.*

Proof: See Sec. C.5.10. □

C.4 Additional Results

C.4.1 Number of Check Nodes of Degree Two

The following lemma tries to convey some more intuition about what is happening for codes on trees.

Lemma C.18 *Let* X *be a Tanner tree (we are still assuming that no check node is a leaf) and let* $\mathcal{C}'$ *be the set of parity-check nodes with degree two and let* $\mathcal{C}'' \triangleq C\mathsf{X} \setminus \mathcal{C}'$ *be the parity-check nodes with degree larger than two. Independent of the rate of the code, the number of parity-check nodes of degree two fulfills* $|\mathcal{C}'| \geq 2r - n + 1 = n - 2k + 1$. *Especially for codes with* $R = k/n \leq 1/2$ *this bound is non-trivial.*

Proof: Omitted. □

Remark C.19 (Interpretation of Lemma C.18) For rates smaller than 1/2 we clearly have check-nodes with valency two, i.e., they are connected only to two different variable nodes. So these two variable nodes always have the same value in a valid configuration. Codes with this property are generally not too good. □

C.4.2 Asymptotically Good and Asymptotically Bad Codes

Definition C.20 A sequence of codes $\mathcal{C}_1, \mathcal{C}_2, \ldots$ over GF(q), for q fixed, where $\mathcal{C}_i$ is an $[n_i, k_i, d_i]$ code with rate $R_i = k_i/n_i$ and $n_{i+1} > n_i$, is called an *infinite family* $\{\mathcal{C}_i\}$ *of codes.* A subfamily of codes is a subsequence $\{\mathcal{C}_{i_\ell}\}$ of $\{\mathcal{C}_i\}$ with $i_{\ell+1} > i_\ell$. □

Definition C.21 If an infinite family $\{\mathcal{C}_i\}$ of codes has an infinite subfamily $\{\mathcal{C}_{i_\ell}\}$ which fulfills

$$\lim_{\ell\to\infty} R_{i_\ell} = \lim_{\ell\to\infty} \frac{k_{i_\ell}}{n_{i_\ell}} = c' \quad \text{and} \quad \lim_{\ell\to\infty} \frac{d_{i_\ell}}{n_{i_\ell}} = c''$$

with $c' > 0$ and $c'' > 0$, the infinite family of codes is said to be *asymptotically good.* Else it is called *asymptotically bad.* □

Theorem C.22 (see [29]) *Infinite families of codes which have cycle-free Tanner graphs are asymptotically bad.*

Proof: Omitted. □

C.4.3 Graphs with Loops and the Cycle Rank

Shortening a code at position j is a process done in two steps:

- All codewords of $\mathcal{C}$ having a zero at position j are selected.
- Code bit j is taken away from all the remaining codewords.

Lemma C.23 *Let a code* $\mathcal{C}$ *with parameters* $[n, k, d_{\min}]$ *be shortened at position* j*. The resulting code* $\mathcal{C}'$ *has parameters* $[n', k', d'_{\min}]$ *with* $n' = n - 1$, $k' \geq k - 1$, *and* $d'_{\min} \geq d_{\min}$.

Proof: Omitted. □

Definition C.24 The *cycle rank* c of a graph is the number of independent cycles of a graph X and can be calculated as $c = |E\mathsf{X}| - |V\mathsf{X}| + \operatorname{Comp}\mathsf{X}$, where $\operatorname{Comp}\mathsf{X}$ is the number of components of the graph. □

The graph interpretation of shortening a code at position j is the following: we have to take away the variable node corresponding to code bit j and all its incident edges. Doing so, the cycle rank of the graph cannot increase.

Theorem C.25 (see [29]) *For an infinite family of codes to be asymptotically good, the cycle rank of the corresponding Tanner graphs must grow proportionally to the code length.*

Proof: Omitted. □

C.5 Proofs

C.5.1 Proof of Lemma C.4

As the code has dimension k, it has 2^k codewords and therefore the Tanner graph has 2^k valid configurations. But not all of the possible $2^{|\mathcal{L}|}$ choices for values of leaves lead to valid configurations, therefore $2^{|\mathcal{L}|} > 2^k$ and finally $|\mathcal{L}| > k$, or equivalently $|\mathcal{L}| \geq k+1$. □

C.5.2 Proof of Theorem C.5

As $n = 1$ is trivial, we assume $n \geq 2$. $R = k/n \geq 1/2$ means $k \geq n/2$ and $r = n - k \leq n/2$. Let $\mathcal{L}$ be the set of leaves and using Lemma C.4 we get the following chain of inequalities

$$|\mathcal{L}| \geq k+1 > k \geq \frac{n}{2} \geq r,$$

which means that there are more leaves than parity-check nodes. Following the pigeon-hole principle, there are at least two leaves which are connected to the same parity-check node, say b_j and $b_{j'}$. But a configuration having ones at nodes b_j and $b_{j'}$ and zeros everywhere else is clearly a valid configuration and corresponds to a codeword having weight 2; therefore $d_{\min} \leq 2$. □

C.5.3 Proof of Lemma C.8

By construction, there is certainly a $j \in \{1, \ldots, r_2\}$ such that $N_j \geq 2$. Assume (i, i', j) minimizes the expression given in the statement. The configuration having ones at all variable nodes of $\mathsf{T}_i^{(j)}$ and $\mathsf{T}_{i'}^{(j)}$ and zeros everywhere else clearly is a valid configuration corresponding to a codeword with weight $t_i^{(j)} + t_{i'}^{(j)}$. □

C.5.4 Proof of Lemma C.10

Assume that condition (C.2) is not fulfilled for a certain j and let $i' = \arg\max_i t_i^{(j)}$ and $i'' = \arg\min_i t_i^{(j)}$. The graph having the small tree sizes $t_{i'}^{(j)} - 1$ and $t_{i''}^{(j)} + 1$ instead of $t_{i'}^{(j)}$ and $t_{i''}^{(j)}$, respectively, cannot have a smaller minimum distance.

Assume that condition (C.2) is fulfilled but condition (C.3) is not and let $(i', j') = \arg\max_{(i,j)} t_i^{(j)}$ and $(i'', j'') = \arg\min_{(i,j)} t_i^{(j)}$ (note that we must have $j' \neq j''$). The graph having the small tree sizes $t_{i'}^{(j')} - 1$ and $t_{i''}^{(j'')} + 1$ instead of $t_{i'}^{(j')}$ and $t_{i''}^{(j'')}$, respectively, cannot have a smaller minimum distance and the fulfillment of condition (C.2) is not hurt. □

C.5.5 Proof of Lemma C.11

Note that for $v \in \mathcal{L}$ we have $\deg(v) = 1$ by definition, whereas for $v \in \mathcal{I}$ we have $\deg(v) \geq 2$. On the one hand $\sum_{v \in B\mathsf{X}} \deg(v) = |E\mathsf{X}|$ and on the other hand

$$\begin{aligned}\sum_{v \in B\mathsf{X}} \deg(v) &= \left(\sum_{v \in \mathcal{L}} \underbrace{\deg(v)}_{=1}\right) + \underbrace{\left(\sum_{v \in \mathcal{I}} \underbrace{(\deg(v) - 2)}_{\geq 0}\right)}_{\triangleq \Delta \geq 0} + \left(\sum_{v \in \mathcal{I}} 2\right) \\ &= |\mathcal{L}| + \Delta + 2|\mathcal{I}|,\end{aligned}$$

and so $|E\mathsf{X}| = |\mathcal{L}| + \Delta + 2|\mathcal{I}|$. Using $|E\mathsf{X}| = |V\mathsf{X}| - 1 = |\mathcal{L}| + |\mathcal{I}| + |C\mathsf{X}| - 1$ we obtain $|\mathcal{L}| + |\mathcal{I}| + |C\mathsf{X}| - 1 = |\mathcal{L}| + \Delta + 2|\mathcal{I}|$ and finally $|\mathcal{I}| = |C\mathsf{X}| - \Delta - 1$. From $|C\mathsf{X}| = |B\mathsf{X}| - k = |\mathcal{L}| + |\mathcal{I}| - k$ follows $|\mathcal{L}| = k + 1 + \Delta$. □

C.5.6 Proof of Lemma C.12

We use Lemma C.11 and identify X with the inner tree T'. (We remind the reader that the code defined by the tree T' has dimension k, see Prop. C.7.) Then $|\mathcal{L}| = n_2$, $|\mathcal{I}| = n_3$, $|C\mathsf{X}| = r_2 + r_3$, and $r_3 \geq 0$, and so $n_2 = k + 1 + \Delta$ and $n_3 = r_2 + r_3 - \Delta - 1 \geq r_2 - \Delta - 1$. □

C.5.7 Proof of Lemma C.13

If $a' = 0$ then $\lceil a/b \rceil = a/b = \lfloor a/b \rfloor$ and the result follows. If $1 \leq a' \leq b-1$ then $\lceil a/b \rceil = \lfloor a/b \rfloor + 1$ and the result follows easily. □

C.5.8 Proof of Theorem C.14

We perform Alg. C.6 as defined at the beginning of Sec. C.3.3 and introduce the variables n_1, n_2, n_3, r_1, r_2, r_3, $\mathsf{T}_i^{(j)}$, $t_i^{(j)}$, etc., as in Figure C.1 and let additionally $m \triangleq n + \Delta = m'' n_2 + m'$, where $m' \triangleq R_{n_2}(m)$ and $m'' \triangleq \lfloor m/n_2 \rfloor$. We assume that the small trees of the code graph fulfill (C.2) and (C.3) of Lemma C.10 and that n_2, n_3, r_2, and Δ are constrained by the result given in Lemma C.12. Let $a \triangleq \sum_{j=1}^{r_2} \sum_{i=1}^{N_j} t_i^{(j)} = n - n_3 = n_1 + n_2$, $b \triangleq \sum_{j=1}^{r_2} \sum_{i=1}^{N_j} 1 = \sum_{j=1}^{r_2} N_j = n_2$, and $a' \triangleq R_b(a) = R_{n_2}(n - n_3)$. Important in the derivation of this theorem is to have strong bounds on a, b, and a' in order to use Lemma C.13 together with Lemmas C.8 and C.10 as effectively as possible. We have to consider different cases.

(Case 1) $k = 1$. Then T is "partitioned" into one part. So we have $d_{\min} = n$ and $d_{\min}$ fulfills the following equality (proving is done by looking at the cases n even and n odd)

$$d_{\min} = n = \left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{n+1}{2} \right\rfloor = \left\lfloor \frac{n}{k+1} \right\rfloor + \left\lfloor \frac{n+1}{k+1} \right\rfloor .$$

(Case 2) Our partitioning consists of at least two small trees $\mathsf{T}_i^{(j)}$ and we have

$$a = n - n_3 = n + \Delta - n_3 - \Delta = m - n_3 - \Delta = m'' n_2 + (m' - n_3 - \Delta).$$

Among other things, $a \leq m$ and $b = n_2$, thus $\lfloor a/b \rfloor \leq \lfloor m/n_2 \rfloor$.

(Case 2.1) $m' - n_3 - \Delta < 0$. So $\lceil a/b \rceil = \lceil (m'' n_2 + m' - n_3 - \Delta)/n_2 \rceil \leq \lceil m'' n_2 / n_2 \rceil = m'' = \lfloor m/n_2 \rfloor$. Then by Lemmas C.8 and C.10

$$d_{\min} \leq \left\lfloor \frac{a}{b} \right\rfloor + \left\lceil \frac{a}{b} \right\rceil \leq \left\lfloor \frac{m}{n_2} \right\rfloor + \left\lfloor \frac{m}{n_2} \right\rfloor \leq \left\lfloor \frac{m}{n_2} \right\rfloor + \left\lfloor \frac{m+1}{n_2} \right\rfloor . \qquad \text{(C.5)}$$

(Case 2.2) $0 \leq m' - n_3 - \Delta$. By this and, as by definition, $m' \leq n_2 - 1$, we have $a' = m' - n_3 - \Delta$.

(Case 2.2.1) $m' = n_2 - 1$. Therefore n_2 divides $m + 1$. But $a \leq m$ and $b = n_2$, so that $\lceil a/b \rceil \leq \lceil m/n_2 \rceil = \lceil (m+1)/n_2 \rceil = (m+1)/n_2 = \lfloor (m+1)/n_2 \rfloor$. By Lemmas C.8 and C.10

$$d_{\min} \leq \left\lfloor \frac{a}{b} \right\rfloor + \left\lceil \frac{a}{b} \right\rceil \leq \left\lfloor \frac{m}{n_2} \right\rfloor + \left\lfloor \frac{m+1}{n_2} \right\rfloor . \qquad \text{(C.6)}$$

(Case 2.2.2) Otherwise, $0 \leq m' \leq n_2 - 2$. So $a' = m' - n_3 - \Delta \leq n_2 - n_3 - \Delta - 2$. But by Lemma C.11, $n_3 \geq r_2 - \Delta - 1$ such that $a' \leq n_2 - r_2 - 1$. $b - a' \geq r_2 + 1$ of the $t_i^{(j)}$ must have value $\lfloor a/b \rfloor$. By the pigeon-hole principle, as there are only r_2 different j's, two of these $t_i^{(j)}$ must have the same j. By Lemmas C.8 and C.10

$$d_{\min} \leq 2 \left\lfloor \frac{a}{b} \right\rfloor \leq 2 \left\lfloor \frac{m}{n_2} \right\rfloor \leq \left\lfloor \frac{m}{n_2} \right\rfloor + \left\lfloor \frac{m+1}{n_2} \right\rfloor . \qquad \text{(C.7)}$$

In (C.5), (C.6), and (C.7) we have the same upper bound for the minimum distance which can be upper bounded by using $n \geq k + 1$ (by assumption on the rate of the code), $n_2 = k + 1 + \Delta$ (by Lemma C.11), and $m = n + \Delta$ (by definition)

$$\begin{aligned} d_{\min} &\leq \left\lfloor \frac{m}{n_2} \right\rfloor + \left\lfloor \frac{m+1}{n_2} \right\rfloor = \left\lfloor \frac{n+\Delta}{k+1+\Delta} \right\rfloor + \left\lfloor \frac{n+1+\Delta}{k+1+\Delta} \right\rfloor \\ &= \left\lfloor \frac{n\left(1+\frac{\Delta}{n}\right)}{(k+1)\left(1+\frac{\Delta}{k+1}\right)} \right\rfloor + \left\lfloor \frac{(n+1)\left(1+\frac{\Delta}{n+1}\right)}{(k+1)\left(1+\frac{\Delta}{k+1}\right)} \right\rfloor \\ &\leq \left\lfloor \frac{n}{k+1} \right\rfloor + \left\lfloor \frac{n+1}{k+1} \right\rfloor . \end{aligned}$$

This concludes the proof of the left inequality in (C.4). Finally, we show the trueness of the right inequality in (C.4). From $k \leq n$ and $R = k/n$ we get

$$\left\lfloor \frac{n}{k+1} \right\rfloor + \left\lfloor \frac{n+1}{k+1} \right\rfloor \leq \frac{n}{k+1} + \frac{n+1}{k+1} < \frac{n}{k} + \frac{n\left(1+\frac{1}{n}\right)}{k\left(1+\frac{1}{k}\right)} \leq \frac{n}{k} + \frac{n}{k} = \frac{2}{R}.$$

□

C.5.9 Proof of Theorem C.15

To each upper index j there are at least two small trees $\mathsf{T}_i^{(j)}$. Let the small trees be numbered in the following way: $t_i^{(j)}$ is non-decreasing in i for a fixed j and $t_1^{(j)} + t_2^{(j)}$ is non-decreasing in j. (From this follows that $t_i^{(j)} \geq (t_1^{(j)} + t_2^{(j)})/2$ for all j and $3 \leq i \leq N_j$.) The small trees are non-overlapping and so the sum of all $t_i^{(j)}$'s can be upper bounded by the code length n; note that we have a total of $n_2 = \sum_{j=1}^{r_2} N_j$ small trees. By applying Lemma C.4 to the inner tree T' defined on page 166 we have $n_2 \geq k+1$ and so we get

$$n \geq \sum_{j=1}^{r_2} \sum_{i=1}^{N_j} t_i^{(j)} \geq n_2 \cdot \frac{t_1^{(1)} + t_2^{(1)}}{2} \geq (k+1) \cdot \frac{t_1^{(1)} + t_2^{(1)}}{2},$$

and finally by Lemma C.8

$$d_{\min} \leq t_1^{(1)} + t_2^{(1)} \leq \left\lfloor \frac{2n}{k+1} \right\rfloor \leq \frac{2n}{k+1} < \frac{2n}{k} = \frac{2}{R}.$$

□

C.5.10 Proof of Lemma C.17

If one of the trees has no check node, it consists only of one variable node; the minimum distance of the whole code is then not larger than 1 and so the bounds hold also for the forest. Therefore, for the following we assume that each tree has at least one check node and two variable nodes.

Assume that the forest consists of two trees; without loss of generality, we can assume that the first tree has a minimum distance that is not larger than the minimum distance of the second tree. By introducing a new edge which connects a *check node* of the first tree with a *variable node* of the second tree, we get a new large tree out of the old forest. Note that neither the length nor the dimension of the code changed but the minimum distance cannot increase. (We have only changed a check equation; but as there are still no cycles there are no redundant check equations and the dimension stays constant.) As the bounds in Theorems C.5 and C.14 hold for the new tree, they also hold for the old forest. Forests with more than two trees are treated by doing the above procedure recursively several times. □

Appendix D

The BCJR Algorithm, the One-Sweep Algorithm, and Decoding on the Dual Code

D.1 Introduction and Motivation

Assume to have a partial factor graph as in Fig. D.1, where the function node represents the indicator function of a linear subcode $\mathcal{C}$. During an update of this function node we compute the outgoing messages $\mu_{f \to X_i}(x_i)$ based on the incoming messages $\mu_{X_i \to f}(x_i)$ using the sum-product algorithm[1] (i.e. the summary-product algorithm over the ring $\langle \mathcal{R}, +, \cdot \rangle$). If the subcode is represented by a trellis, one efficient way to solve this update task is by using the BCJR algorithm [11], or equivalently, the forward-backward algorithm.

But there are alternatives. In this appendix we would like to dis-

[1]Note that we are interested in calculating the exact outgoing messages based on the incoming messages; we are not interested in approximations to the sum-product algorithm.

cuss the relationship between three algorithm that perform this update, namely the BCJR algorithm, the one-sweep algorithm by Johansson and Zigangirov [48], and decoding on the dual code [43] (see also [31]). In contrast to the BCJR algorithm, which needs a forward and a backward sweep, the one-sweep algorithm only needs a slightly more complex forward sweep with some simple processing at the end.[2] Decoding on the dual code can be an interesting alternative when the subcode is of high-rate whence the dual subcode is of low-rate which results in less operations to be performed.

To motivate the upcoming results we briefly look at a different problem. Often it is useful to solve a transformed problem and then to transform the solution back to the desired space. E.g., one way of solving a linear differential equation is by finding the solution directly in the time domain. This is not alway easy. As shown in Fig. D.2, a more elegant way is to use the Laplace transform to get an algebraic equation, solve this equation in the frequency domain and then transform the solution back to the time domain.

In a similar fashion, as shown in Fig. D.3, the one-sweep algorithm and decoding on the dual code can each be considered as solving a transformed problem. The diagram in Fig. D.4 then summarizes the relationships between the solution vectors of the three algorithms that will be presented. We conjecture that showing the common underlying pattern and the connections between these algorithms might perhaps inspire new transforms that lead to new efficient algorithms.

Before starting with the exact description of the algorithms, we would like to make several comments.

- In the original formulation of the one-sweep algorithm in [48], the subcode was the whole code and the algorithm was expressed in terms of a-priori, conditional, and a-posteriori probabilities. In the process of the derivation of the results in this appendix, we give a more general version of the one-sweep algorithm that uses the more "abstract" concept of messages (especially we do not need anymore a distinction if a-priori probabilities are given or not). In our opinion, this more abstract viewpoint makes the derivation somewhat simpler.

[2]The one-sweep algorithm is non-local in the sense that the mentioned processing at the end is non-local in the factor graph shown in Fig. D.8.

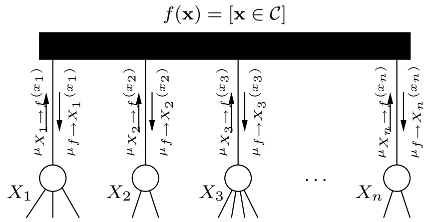

Figure D.1: Partial factor graph where the function node $f(\mathbf{x}) = [\mathbf{x} \in \mathcal{C}]$ represents the subcode $\mathcal{C}$.

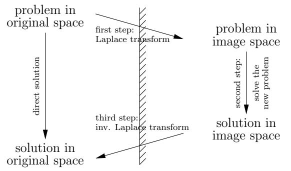

Figure D.2: Solving a linear differential equation with the help of the Laplace transform.

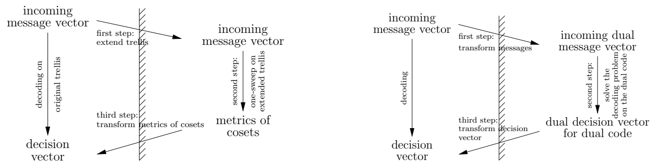

Figure D.3: Transformed versions of message passing in a subcode function node. Left: one-sweep algorithm. Right: decoding on the dual code.

- We will discuss the case where the symbol nodes X_i are binary. Generalizations to non-binary fields are easily possible. In generalizing, one has to be careful with respect to the following fact: the binary DFT and inverse DFT are (up to scaling) equivalent. This is not anymore the case for other fields. We tried to use therefore the matrices $\mathbf{D} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ and $\mathbf{D}^{-1} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^{-1}$ only when the DFT and the inverse DFT, respectively, are involved.
- The summary-product algorithm in general can be formulated for any semi-ring with identity element. But note that the one-sweep algorithm and decoding on the dual code are especially tailored for the case where the summary-product algorithm is taken to be the sum-product algorithm (over the ring of real (or complex) numbers).
- In general, it is not easy to say which algorithm is preferable over the other in terms of complexity. Besides the facts mentioned above, we think that a fair comparison is only possible if one really has a specific example at hand that has to be implemented in a specific software/hardware environment.

Definition D.1 We will use the following notations and definitions.

- Let $\mathcal{I} = \{1, 2, \ldots, n\}$ be the set of integers from 1 to n.
- Let $\mathcal{F} = \{0, 1\}$ be the binary field; as we are only considering the binary case, $x_i \in \mathcal{F}$ (for all $i \in \mathcal{I}$).
- Let $\mathbf{e}_i$ be a row vector of length n filled with zeros except for a "1" at position i. Let $\mathcal{C}_i \triangleq \mathcal{C} + \mathbf{e}_i$ ($i \in \mathcal{I}$) be cosets of the code $\mathcal{C}$.
- To simplify notation, we will use for all $i \in \mathcal{I}$ the abbreviations

$$\mu_i(x_i) \triangleq \mu_{X_i \to f}(x_i) \quad \text{and} \quad \mu_i^{\perp}(x_i) \triangleq \mu^{\perp}_{X_i \to f}(x_i).$$

  for the incoming messages and incoming dual messages (dual messages are defined below).
- We let the DFT matrix be

$$\mathbf{D} \triangleq \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

- For all $i \in \mathcal{I}$, let

$$\begin{pmatrix} \mu_i^{\perp}(0) \\ \mu_i^{\perp}(1) \end{pmatrix} \triangleq \mathbf{D} \cdot \begin{pmatrix} \mu_i(0) \\ \mu_i(1) \end{pmatrix} \tag{D.1}$$

  be the dual incoming messages which are related to the usual incoming messages through the DFT.
- For all $i \in \mathcal{I}$, let $\Lambda_i \triangleq \mu_i(0)/\mu_i(1)$ and $\Lambda_i^{\perp} \triangleq \mu_i^{\perp}(0)/\mu_i^{\perp}(1)$ be the likelihood ratio and the "dual likelihood ratio", respectively. We will also need the transformation matrices

$$\mathbf{T}_i \triangleq \begin{pmatrix} 1 & 1 \\ \Lambda_i^{-1} & \Lambda_i \end{pmatrix}, \quad \text{and} \quad \mathbf{T}_i^{\perp} \triangleq \begin{pmatrix} 1 & 1 \\ (\Lambda_i^{\perp})^{-1} & \Lambda_i^{\perp} \end{pmatrix}.$$

- The final goal of all our efforts in this appendix is to calculate in an efficient manner the outgoing messages, i.e. to compute for all $i \in \mathcal{I}$

$$\mu_{f \to X_i}(0) \triangleq \sum_{\substack{\mathbf{x} \in \mathcal{F}^n \\ x_i = 0}} f(\mathbf{x}) \prod_{\substack{\ell \in \mathcal{I} \\ \ell \neq i}} \mu_\ell(x_\ell) = \sum_{\substack{\mathbf{x} \in \mathcal{C} \\ x_i = 0}} \prod_{\substack{\ell \in \mathcal{I} \\ \ell \neq i}} \mu_\ell(x_\ell),$$

$$\mu_{f \to X_i}(1) \triangleq \sum_{\substack{\mathbf{x} \in \mathcal{F}^n \\ x_i = 1}} f(\mathbf{x}) \prod_{\substack{\ell \in \mathcal{I} \\ \ell \neq i}} \mu_\ell(x_\ell) = \sum_{\substack{\mathbf{x} \in \mathcal{C} \\ x_i = 1}} \prod_{\substack{\ell \in \mathcal{I} \\ \ell \neq i}} \mu_\ell(x_\ell).$$

- For $i \in \mathcal{I}$ we define

$$\eta_i(0) \triangleq \mu_{X_i \to f}(0) \cdot \mu_{f \to X_i}(0) = \mu_i(0) \cdot \mu_{f \to X_i}(0) = \sum_{\substack{\mathbf{x} \in \mathcal{C} \\ x_i = 0}} \prod_{\ell=1}^{n} \mu_\ell(x_\ell),$$

$$\eta_i(1) \triangleq \mu_{X_i \to f}(1) \cdot \mu_{f \to X_i}(1) = \mu_i(1) \cdot \mu_{f \to X_i}(1) = \sum_{\substack{\mathbf{x} \in \mathcal{C} \\ x_i = 1}} \prod_{\ell=1}^{n} \mu_\ell(x_\ell).$$

  (Note that if we had to make a decision about X_i according to the sum-product algorithm, one would base it on $\eta_i(0)$ and $\eta_i(1)$.)
- Vectors (e.g. codewords) that are related to codes will be written as row vectors whereas message vectors will be written as column vectors.
- Often, one is only interested in the message vectors up to scaling (see also Rem. 3.14 on page 39). The same comments applies also to the vectors related to messages defined in this appendix.

Corollary D.2 *Instead of calculating $\mu_{f\to X_i}(x_i)$ directly, we can first calculate $\eta_i(x_i)$ and then get the desired outgoing messages using the formula $\mu_{f\to X_i}(x_i) = \eta_i(x_i)/\mu_i(x_i)$. In matrix notation,*

$$\begin{pmatrix}\mu_{f\to X_i}(0)\\ \mu_{f\to X_i}(1)\end{pmatrix} = \begin{pmatrix}1/\mu_i(0) & 0\\ 0 & 1/\mu_i(1)\end{pmatrix}\cdot\begin{pmatrix}\eta_i(0)\\ \eta_i(1)\end{pmatrix}.$$

D.2 Relations Between the Different Algorithms

$$\begin{tikzcd}
\underset{\text{decoding on code}}{\begin{pmatrix}\eta_i(0)\\ \eta_i(1)\end{pmatrix}} \arrow[r, "\mathbf{T}_i"] \arrow[d, "\mathrm{DFT}"'] & \underset{\text{one-sweep algorithm on code}}{\begin{pmatrix}\chi\\ \chi_i\end{pmatrix}} \\
\underset{\text{one-sweep algorithm on dual code}}{\begin{pmatrix}\chi^{\perp}\\ \chi_i^{\perp}\end{pmatrix}} & \underset{\text{decoding on dual code}}{\begin{pmatrix}\eta_i^{\perp(0)}\\ \eta_i^{\perp(1)}\end{pmatrix}} \arrow[l, "\mathbf{T}_i^{\perp}"] \arrow[u, "\mathrm{DFT}"']
\end{tikzcd}$$

Figure D.4: Diagram highlighting the relations between the three different algorithms (for each $i \in \mathcal{I}$).

The diagram in Fig. D.4 summarizes the relationships between the solution vectors of the three algorithms that will be presented. In Def. D.1 we have already given the definition of the vector $(\eta_i(0), \eta_i(1))^T$, which is shown in the upper left-hand corner of the diagram in Fig. D.4; its relation to the outgoing messages was given in Cor. D.2. The one-sweep algorithm and decoding on the dual code correspond to calculating *linear transformations* of the vectors $(\eta_i(0), \eta_i(1))^T$ for all $i \in \mathcal{I}$.

The following sections (Secs. D.4, D.5, and D.6) will discuss step-by-step what the different entries in the diagram in Fig. D.4 mean. But before we can start discussing the various algorithms, we need to know how to construct a trellis for a given block code (Sec. D.3).

D.3 Construction of Trellises

An optimal method for constructing a trellis for an $[n, k]$ code given by a parity-check matrix

$$\mathbf{H} \triangleq \begin{bmatrix}\mathbf{h}_1^T & \mathbf{h}_2^T & \cdots & \mathbf{h}_n^T\end{bmatrix}$$

is given in [11][3]; it is also the one used in [48]. Let r be the number of rows of $\mathbf{H}$. For simplicity in exposition, we will make in the following the reasonable assumption that $\mathbf{h}_i^T \neq \mathbf{0}^T$ for all $i \in \mathcal{I}$.

Definition D.3 Let $i \in \mathcal{I}$.

- If $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is a row vector of length n, we let $\mathbf{x}_{[1:i]} \triangleq (x_1, x_2, \ldots, x_i)$.
- Let $\mathbf{H}_{[1:i]} \triangleq \left[\mathbf{h}_1^T, \mathbf{h}_2^T, \ldots, \mathbf{h}_i^T\right]$ be a submatrix of $\mathbf{H}$.
- Let $\mathbf{s}_0 \triangleq \mathbf{0}$ and let the row vectors $\mathbf{s}_i = \mathbf{x}_{[1:i]} \cdot \mathbf{H}_{[1:i]}^T$ of length r be the so-called partial syndromes.
- In a trellis each path from left to right represents a codeword and all paths should start and end in the same state (usually the zero state). Moreover, for each codeword there is such a path.

$\square$

Algorithm D.4 (to construct a trellis for a block code) Given a block code, a trellis is now drawn according to the following steps.

- Draw a grid of nodes with 2^r rows and $n + 1$ columns and do the labeling at the bottom and on the left-hand side of the grid as shown in Fig. D.5. The vertical labels are the different possible partial syndromes, i.e. all binary vectors of length r. The horizontal positions are labeled from 0 to n whereas the label of code-bit x_i is in-between position $i - 1$ and i, $i \in \mathcal{I}$.
- The paths through the trellis are constructed in the following way. For each $\mathbf{x} \in \mathcal{C}$ one calculates the partial syndromes $\mathbf{s}_i$ for all $i \in \{0, 1, \ldots, n\}$; the path of $\mathbf{x}$ through the trellis is then the unique path that goes through state $\mathbf{s}_i$ at position i for all $i \in \{0, 1, \ldots, n\}$.

[3] By optimal we mean optimal in various senses, see e.g. [121]. For other orderings of the time axis there might be realizations with less states.

If the path between position $i-1$ and position i connects two different partial syndromes the label is "1", else it is "0". Note that $\mathbf{s}_0 = \mathbf{0}$ and $\mathbf{s}_n = \mathbf{0}$ for all $\mathbf{x} \in \mathcal{C}$, so all the paths start and end in the same state (as it should be by definition of a trellis). $\square$

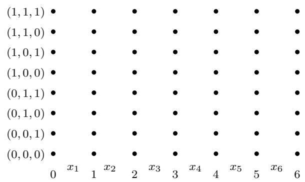

Figure D.5: Grid for a trellis of a code having a parity-check matrix with $n = 6$ columns and $r = 3$ rows.

Example D.5 (First trellis construction example) As a first example we consider the binary linear $[6, 3, 2]$ code with the parity-check matrix

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

Performing Alg. D.4 for the given code results in the trellis shown in Fig. D.6 (left). E.g., for the codeword $\mathbf{x} = (1, 1, 0, 1, 1, 1)$ we get the partial syndromes

$$\mathbf{s}_0 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}, \ \mathbf{s}_1 = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}, \ \mathbf{s}_2 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}, \ \mathbf{s}_3 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix},$$
$$\mathbf{s}_4 = \begin{pmatrix} 1 & 1 & 0 \end{pmatrix}, \ \mathbf{s}_5 = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}, \ \mathbf{s}_6 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}.$$

$\square$

Example D.6 (Second trellis construction example) As a second example we consider the binary $[7, 4, 3]$ Hamming code given by the parity-check matrix

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

Performing Alg. D.4 we get the trellis shown in Fig. D.7 (left). $\square$

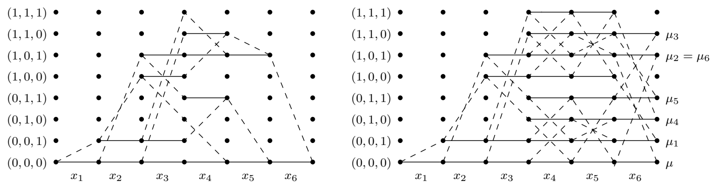

Figure D.6: Trellises for Ex. D.5 (Solid line: "0", dashed line: "1").

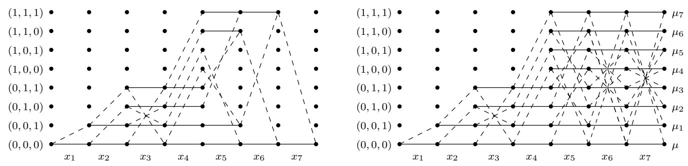

Figure D.7: Trellises for Ex. D.6 (Solid line: "0", dashed line: "1").

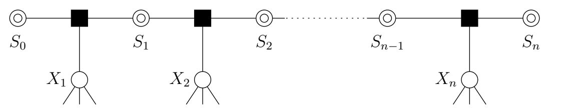

Figure D.8: Detailed version of the partial factor graph shown in Fig. D.1. The sum-product algorithm on this factor graph corresponds to the BCJR algorithm (or forward-backward algorithm).

D.4 The BCJR Algorithm

Without going too much into the details, the message-passing formulation of the BCJR algorithm (or forward-backward algorithm) does the calculations according to the branches in the trellises as defined in Sec. D.3. From a higher-level viewpoint this corresponds to the sum-product algorithm on the partial factor graph shown in Fig. D.8, where the function nodes represent indicator functions of the trellis sections.

D.5 The One-Sweep Algorithm

Definition D.7 We define

$$\chi \triangleq \sum_{\mathbf{x}\in\mathcal{C}} \prod_{\ell=1}^{n} \mu_\ell(x_\ell), \quad \chi_i \triangleq \sum_{\mathbf{x}\in\mathcal{C}_i} \prod_{\ell=1}^{n} \mu_\ell(x_\ell) \quad (\text{for } i \in \mathcal{I}).$$

□

Theorem D.8 *Let $i \in \mathcal{I}$. Then, $\eta_i(0)$, $\eta_i(1)$, χ, and χ_i are related by*

$$\mathbf{T}_i \cdot \begin{pmatrix} \eta_i(0) \\ \eta_i(1) \end{pmatrix} = \begin{pmatrix} \chi \\ \chi_i \end{pmatrix}, \tag{D.2}$$

where the $\mathbf{T}_i$ matrices were defined in Def. D.1. The outgoing messages are then proportional to

$$\begin{pmatrix} \mu_{f\to X_i}(0) \\ \mu_{f\to X_i}(1) \end{pmatrix} \propto \begin{pmatrix} \frac{1}{\mu_i(0)} \cdot \left(\frac{\mu_i(0)}{\mu_i(1)} - \frac{\chi_i}{\chi} \right) \\ \frac{1}{\mu_i(1)} \cdot \left(\frac{\chi_i}{\chi} - \frac{\mu_i(1)}{\mu_i(0)} \right) \end{pmatrix} \propto \begin{pmatrix} \mu_i(0) & -\mu_i(1) \\ -\mu_i(1) & \mu_i(0) \end{pmatrix} \cdot \begin{pmatrix} \chi \\ \chi_i \end{pmatrix}. \tag{D.3}$$

Proof: For a proof, see Sec. D.7.1. □

Th. D.8 states that if we can calculate efficiently the vectors χ and χ_i $(i \in \mathcal{I})$, then we can also calculate efficiently the vectors $(\eta_i(0), \eta_i(1))^T$ and $(\mu_{f\to X_i}(0), \mu_{f\to X_i}(1))^T$. That χ and χ_i $(i \in \mathcal{I})$ can indeed be calculated efficiently is shown in the rest of this section. Note that (D.2) is represented by the upper line in the diagram of Fig. D.4, whereas the same relation for the dual code is given in the lower line of the same diagram.

We extend the above trellis construction in Alg. D.4 in the following way: we not only draw all paths for $\mathbf{x} \in \mathcal{C}$, but also the paths associated to all $\mathbf{x} \in \mathcal{C}_i$ for all $i \in \mathcal{I}$. By this extension we get for the codes in Exs. D.5 and D.6 the extended trellises in Fig. D.6 (right) and Fig. D.7 (right), respectively. The labels μ_i on the right-hand side are attached to the level where all codewords of a coset end.[4] The label μ is attached to the level $\mathbf{0}$ where all paths of the codewords of $\mathcal{C}$ end. Note that not all possible final syndromes are associated with a μ or μ_i; but there are some which are connected to several μ_i's, namely exactly when there are several identical columns in the parity-check matrix.

[4] Note that $\mathbf{s}_n = \mathbf{x} \cdot \mathbf{H}^T = \mathbf{e}_i \cdot \mathbf{H}^T = \mathbf{h}_i$ is the same for all vectors $\mathbf{x}$ in the same coset $\mathcal{C}_i$.

Remark D.9 (Special property of Ex. D.6) The parity-check matrix in Ex. D.6 is special in the sense that all possible columns except the zero column appear exactly once. Thus to every final syndrome we can associate exactly a $\mu, \mu_1, \ldots, \mu_n$. (Note, $n = 2^{n-k} - 1$, where k is the dimension and $n-k$ the redundancy of the code, respectively.) □

Algorithm D.10 (to calculate χ and χ_i $(i \in \mathcal{I})$)

- Set

  $$\chi(\mathbf{s}, 0) \triangleq \begin{cases} 1 & (\mathbf{s} = \mathbf{0}), \\ 0 & (\mathbf{s} \neq \mathbf{0}), \end{cases}.$$

  for all binary vectors $\mathbf{s}$ of length r.

- Given $\mu_i(x_i) = \mu_{X_i\to f}(x_i)$, perform the following calculations for all i from 1 to n for all binary vectors $\mathbf{s}$ of length r.

  $$\chi(\mathbf{s}, i) \triangleq \chi(\mathbf{s}, i-1)\mu_i(0) + \chi(\mathbf{s} - \mathbf{h}_i, i-1)\mu_i(1).$$

  (More specifically, this has only to be done for the states $\mathbf{s}$ at position i of the extended trellis if there is at least one branch incident from the left.)

- Finally, we set

  $$\chi \triangleq \chi(\mathbf{0}, n) \quad \text{and} \quad \chi_i \triangleq \chi(\mathbf{h}_i, n) \quad (\text{for all } i \in \mathcal{I}).$$

□

Lemma D.11 *Algorithm D.10 calculates the quantities $\chi, \chi_1, \ldots, \chi_n$ as defined in Def. D.7.*

Proof: We leave it to the reader to check the validity of Algorithm D.10. □

Remark D.12 (Relation to the BCJR algorithm) The above algorithm is of course akin to the forward recursion of the BCJR algorithm, but it is now performed on the extended trellis instead of on the original trellis. □

D.6 Decoding on the Dual Code

If $\mathcal{C}$ is an $[n, k]$ code, let $\mathcal{C}^\perp$ be its dual code with parameters $[n, n-k]$. Let $\mathcal{F} = \{0, 1\}$ be the binary field. So $\mathcal{F}^n$ is the set of all binary sequences of length n. Most of the results that are needed in this section about the Walsh-Hadamard transform are also needed to derive the MacWilliams identities; these results can therefore also be found in Ch. 5 of [76].

Definition D.13 (The Walsh-Hadamard Transform (WHT)) Let g be a function $g : \mathcal{F}^n \to \mathbb{C}$. Let $\mathbf{x} \cdot \tilde{\mathbf{x}}$ be the scalar product of the vectors $\mathbf{x}$ and $\tilde{\mathbf{x}}$ calculated modulo 2. Then $g^\perp : \mathcal{F}^n \to \mathbb{C}$, where

$$g^\perp(\tilde{\mathbf{x}}) = \sum_{\mathbf{x} \in \mathcal{F}^n} g(\mathbf{x})(-1)^{\mathbf{x}\cdot\tilde{\mathbf{x}}},$$

is called the Walsh-Hadamard transform of $g(.)$. Note that this can also be interpreted as the n-dimensional discrete Fourier transform (DFT) of length 2. A WHT of dimension 1 shall be called a DFT. □

Remember for the next definition that the DFT of $(\mu_\ell(0), \mu_\ell(1))^T$ is called $(\mu_\ell^\perp(0), \mu_\ell^\perp(1))^T$ (see Def. D.1).

Definition D.14 Analogous to Defs. D.1 and D.7 for the code $\mathcal{C}$ we define for the dual code $\mathcal{C}^\perp$ for each $i \in \mathcal{I}$

$$\eta_i^{\perp(0)} \triangleq \sum_{\substack{\mathbf{x} \in \mathcal{C}^\perp \\ x_i = 0}} \prod_{\ell=1}^{n} \mu_\ell^\perp(x_\ell), \quad \eta_i^{\perp(1)} \triangleq \sum_{\substack{\mathbf{x} \in \mathcal{C}^\perp \\ x_i = 1}} \prod_{\ell=1}^{n} \mu_\ell^\perp(x_\ell),$$

$$\chi^\perp \triangleq \sum_{\mathbf{x} \in \mathcal{C}^\perp} \prod_{\ell=1}^{n} \mu_\ell^\perp(x_\ell), \qquad \chi_i^\perp \triangleq \sum_{\mathbf{x} \in \mathcal{C}_i^\perp} \prod_{\ell=1}^{n} \mu_\ell^\perp(x_\ell).$$

□

Lemma D.15 *Let $\mathcal{C}$ be a binary linear code, $\mathcal{C}^\perp$ be its dual code, and let $\mathbf{x} \in \mathcal{F}^n$. Then,*

$$\frac{1}{|\mathcal{C}^\perp|} \cdot \sum_{\tilde{\mathbf{x}} \in \mathcal{C}^\perp} (-1)^{\mathbf{x}\cdot\tilde{\mathbf{x}}} = [\mathbf{x} \in \mathcal{C}].$$

Proof: For a proof, see Sec. D.7.2. □

Lemma D.16 (Poisson Summation Formula) *Let $\mathcal{C}$ be a binary linear code. Then,*

$$\sum_{\mathbf{x} \in \mathcal{C}} g(\mathbf{x}) = \frac{1}{|\mathcal{C}^\perp|} \sum_{\tilde{\mathbf{x}} \in \mathcal{C}^\perp} g^\perp(\tilde{\mathbf{x}}).$$

Proof: For a proof, see Sec. D.7.3. □

Lemma D.17 ("Independent" Product in Time-Domain corresponds to "Independent" Product in Frequency Domain) *We assume that $g(\mathbf{x})$ can be written for any $\mathbf{x} \in \mathcal{F}^n$ as a product in the following way*

$$g(\mathbf{x}) \triangleq \prod_{\ell \in \mathcal{I}} g_\ell(x_\ell), \tag{D.4}$$

then its WHT $g^\perp(\tilde{\mathbf{x}})$ can also be written for any $\tilde{\mathbf{x}} \in \mathcal{F}^n$ as a product

$$g^\perp(\tilde{\mathbf{x}}) = \prod_{\ell \in \mathcal{I}} g_\ell^\perp(\tilde{x}_\ell),$$

with the DFT $g_\ell^\perp(\tilde{x}_\ell) \triangleq \sum_{x_\ell} g_\ell(x_\ell)(-1)^{x_\ell \tilde{x}_\ell} = g_\ell(0) + (-1)^{\tilde{x}_\ell} g_\ell(1)$.

Proof: For a proof, see Sec. D.7.4. □

Theorem D.18

$$\mathbf{D} \cdot \begin{pmatrix} \eta_i(0) \\ \eta_i(1) \end{pmatrix} = \frac{1}{|\mathcal{C}^\perp|} \begin{pmatrix} \chi^\perp \\ \chi_i^\perp \end{pmatrix}.$$

Proof: For a proof, see Sec. D.7.5. □

The relationship given in Th. D.18 is depicted on the left-hand side of the diagram in Fig. D.4 whereas its dual formulation is depicted on the right-hand side of the same diagram.

Instead of the above formulation, we can also relate the outgoing messages of the code to the outgoing messages of the dual code as given in the next theorem.

Theorem D.19

$$\begin{pmatrix} \mu_{f \to X_i}(0) \\ \mu_{f \to X_i}(1) \end{pmatrix} = \frac{1}{|\mathcal{C}^\perp|} \cdot \mathbf{D} \cdot \begin{pmatrix} \mu_{f \to X_i}^\perp(0) \\ \mu_{f \to X_i}^\perp(1) \end{pmatrix} \tag{D.5}$$

where

$$\mu^{\perp}_{f\to X_i}(0) \triangleq \sum_{\substack{\tilde{\mathbf{x}}\in\mathcal{C}^{\perp}\\ \tilde{x}_i=0}} \prod_{\substack{\ell\in\mathcal{I}\\ \ell\neq i}} \mu^{\perp}_{\ell}(\tilde{x}_\ell), \qquad \mu^{\perp}_{f\to X_i}(1) \triangleq \sum_{\substack{\tilde{\mathbf{x}}\in\mathcal{C}^{\perp}\\ \tilde{x}_i=1}} \prod_{\substack{\ell\in\mathcal{I}\\ \ell\neq i}} \mu^{\perp}_{\ell}(\tilde{x}_\ell).$$

Proof: For a proof, see Sec. D.7.6, and for an alternative proof, see Sec. D.7.7. □

Note that the message vectors $(\mu^{\perp}_{f\to X_i}(0), \mu^{\perp}_{f\to X_i}(1))^T$ (for $i\in\mathcal{I}$) can be calculated using the BCJR algorithm on the trellis of the dual code using the dual input messages $(\mu^{\perp}_i(0), \mu^{\perp}_i(1))^T$. From a higher viewpoint this is equivalent to using the sum-product algorithm on the dual factor graph (see also [31]). Note also that in (D.5) we use – contrary to intuition – a DFT (and not an inverse DFT) to go from the dual domain back to the primal domain.

D.7 Proofs

D.7.1 Proof of Theorem D.8

We first prove (D.2). We note that

$$\eta_i(0)+\eta_i(1) = \sum_{\substack{\mathbf{x}\in\mathcal{C}\\ x_i=0}} \prod_{\ell=1}^{n} \mu_\ell(x_\ell) + \sum_{\substack{\mathbf{x}\in\mathcal{C}\\ x_i=1}} \prod_{\ell=1}^{n} \mu_\ell(x_\ell) = \sum_{\mathbf{x}\in\mathcal{C}} \prod_{\ell=1}^{n} \mu_\ell(x_\ell) = \chi.$$

This gives the first line of the matrix equation in (D.2). There are many bijective mappings between the code $\mathcal{C}$ and the coset $\mathcal{C}_i$; one of them assigns to $\mathbf{x}\in\mathcal{C}$ the element $\mathbf{x}'\triangleq \mathbf{x}+\mathbf{e}_i\in\mathcal{C}_i$. For any $\mathbf{x}\in\mathcal{F}^n$, $\mathbf{x}'\triangleq\mathbf{x}+\mathbf{e}_i$ with $x_i=0$ (i.e. $x'_i=1$) follows

$$\frac{\prod_{\ell=1}^{n}\mu_\ell(x'_\ell)}{\prod_{\ell=1}^{n}\mu_\ell(x_\ell)} = \frac{\mu_i(x'_i)}{\mu_i(x_i)} = \frac{\mu_i(1)}{\mu_i(0)} = \Lambda_i^{-1}. \tag{D.6}$$

Similarly, for any $\mathbf{x}\in\mathcal{F}^n$, $\mathbf{x}'\triangleq\mathbf{x}+\mathbf{e}_i$ with $x_i=1$ (i.e. $x'_i=0$) we get

$$\frac{\prod_{\ell=1}^{n}\mu_\ell(x'_\ell)}{\prod_{\ell=1}^{n}\mu_\ell(x_\ell)} = \frac{\mu_i(x'_i)}{\mu_i(x_i)} = \frac{\mu_i(0)}{\mu_i(1)} = \Lambda_i. \tag{D.7}$$

Using the definitions of $\eta_i(0)$ and $\eta_i(1)$ and applying the results of (D.6) and (D.7), one gets

$$\begin{aligned}
\Lambda_i^{-1}\cdot\eta_i(0)+\Lambda_i\cdot\eta_i(1) &= \Lambda_i^{-1}\cdot \sum_{\substack{\mathbf{x}\in\mathcal{C}\\ x_i=0}} \prod_{\ell=1}^{n}\mu_\ell(x_\ell) + \Lambda_i\cdot \sum_{\substack{\mathbf{x}\in\mathcal{C}\\ x_i=1}} \prod_{\ell=1}^{n}\mu_\ell(x_\ell)\\
&= \sum_{\substack{\mathbf{x}\in\mathcal{C}\\ x_i=0\\ \mathbf{x}'=\mathbf{x}+\mathbf{e}_i}} \prod_{\ell=1}^{n}\mu_\ell(x'_\ell) + \sum_{\substack{\mathbf{x}\in\mathcal{C}\\ x_i=1\\ \mathbf{x}'=\mathbf{x}+\mathbf{e}_i}} \prod_{\ell=1}^{n}\mu_\ell(x'_\ell)\\
&= \sum_{\substack{\mathbf{x}\in\mathcal{C}\\ \mathbf{x}'=\mathbf{x}+\mathbf{e}_i}} \prod_{\ell=1}^{n}\mu_\ell(x'_\ell) = \sum_{\mathbf{x}'\in\mathcal{C}_i} \prod_{\ell=1}^{n}\mu_\ell(x'_\ell) = \chi_i.
\end{aligned}$$

But this equation corresponds exactly to the second line of the matrix equation in (D.2). Solving (D.2) for $(\eta_i(0),\eta_i(1))^T$, we get

$$\begin{pmatrix}\eta_i(0)\\ \eta_i(1)\end{pmatrix} = \frac{1}{\Lambda_i-\Lambda_i^{-1}} \begin{pmatrix}\Lambda_i & -1\\ -\Lambda_i^{-1} & 1\end{pmatrix}\cdot\begin{pmatrix}\chi\\ \chi_i\end{pmatrix} \propto \begin{pmatrix}\frac{\mu_i(0)}{\mu_i(1)}-\frac{\chi_i}{\chi}\\ \frac{\chi_i}{\chi}-\frac{\mu_i(1)}{\mu_i(0)}\end{pmatrix}.$$

(D.3) follows upon combining this result with Cor. D.2. □

D.7.2 Proof of Lemma D.15

If $\mathbf{x}\in\mathcal{C}$ then $\mathbf{x}\cdot\tilde{\mathbf{x}}=0$ (mod 2) and so $(-1)^{\mathbf{x}\cdot\tilde{\mathbf{x}}}=1$ for any $\tilde{\mathbf{x}}\in\mathcal{C}^{\perp}$ and therefore the sum is $|\mathcal{C}^{\perp}|$. If $\mathbf{x}\notin\mathcal{C}$ then there exists a $\tilde{\mathbf{x}}'\in\mathcal{C}^{\perp}$ such that $\mathbf{x}\cdot\tilde{\mathbf{x}}'=1$ (mod 2). Let $S\triangleq\sum_{\tilde{\mathbf{x}}\in\mathcal{C}^{\perp}}(-1)^{\mathbf{x}\cdot\tilde{\mathbf{x}}}$. But then

$$\begin{aligned}
S &= \sum_{\tilde{\mathbf{x}}\in\mathcal{C}^{\perp}} (-1)^{\mathbf{x}\cdot\tilde{\mathbf{x}}} \overset{(*)}{=} \sum_{\tilde{\mathbf{x}}:\,\tilde{\mathbf{x}}+\tilde{\mathbf{x}}'\in\mathcal{C}^{\perp}} (-1)^{\mathbf{x}\cdot(\tilde{\mathbf{x}}+\tilde{\mathbf{x}}')} \overset{(**)}{=} \sum_{\tilde{\mathbf{x}}\in\mathcal{C}^{\perp}} (-1)^{\mathbf{x}\cdot(\tilde{\mathbf{x}}+\tilde{\mathbf{x}}')}\\
&= (-1)^{\mathbf{x}\cdot\tilde{\mathbf{x}}'}\cdot\sum_{\tilde{\mathbf{x}}\in\mathcal{C}^{\perp}}(-1)^{\mathbf{x}\cdot\tilde{\mathbf{x}}} = -S,
\end{aligned}$$

implies that $S=0$, where equality $(*)$ follows from replacing $\tilde{\mathbf{x}}$ by $\tilde{\mathbf{x}}+\tilde{\mathbf{x}}'$ and equality $(**)$ follows from the fact that $\tilde{\mathbf{x}}'\in\mathcal{C}^{\perp}$. □

D.7.3 Proof of Lemma D.16

By the definition of the WHT and Lemma D.15,

$$\begin{aligned}\frac{1}{|\mathcal{C}^\perp|}\sum_{\tilde{\mathbf{x}}\in\mathcal{C}^\perp} g^\perp(\tilde{\mathbf{x}}) &= \frac{1}{|\mathcal{C}^\perp|}\sum_{\tilde{\mathbf{x}}\in\mathcal{C}^\perp}\sum_{\mathbf{x}\in\mathcal{F}^n} g(\mathbf{x})(-1)^{\mathbf{x}\cdot\tilde{\mathbf{x}}}\\ &= \sum_{\mathbf{x}\in\mathcal{F}^n} g(\mathbf{x})\frac{1}{|\mathcal{C}^\perp|}\sum_{\tilde{\mathbf{x}}\in\mathcal{C}^\perp}(-1)^{\mathbf{x}\cdot\tilde{\mathbf{x}}} = \sum_{\mathbf{x}\in\mathcal{F}^n} g(\mathbf{x})\cdot[\mathbf{x}\in\mathcal{C}]\\ &= \sum_{\mathbf{x}\in\mathcal{C}} g(\mathbf{x}).\end{aligned}$$

□

D.7.4 Proof of Lemma D.17

Before proving Lemma D.17, we need the auxiliary Lemma D.20. To get a feeling for the content of the this lemma, we suggest to have a look at Ex. D.21.

Lemma D.20 *Let* $b_{i,j}$, $i = 1,\ldots,N_1$, $j = 1,\ldots,N_2$. *Then*

$$\sum_{\mathbf{j}\in\{1,\ldots,N_2\}^{N_1}}\prod_{i=1}^{N_1} b_{i,j_i} = \prod_{i=1}^{N_1}\sum_{j=1}^{N_2} b_{i,j}.$$

Proof: The proof is by induction on N_1. □

Example D.21 (to Lemma D.20) Let $N_1 = 3$ and $N_2 = 2$. Then Lemma D.20 reads

$$\begin{aligned}&b_{1,1}b_{2,1}b_{3,1} + b_{1,1}b_{2,1}b_{3,2} + b_{1,1}b_{2,2}b_{3,1} + b_{1,1}b_{2,2}b_{3,2} +\\ &b_{1,2}b_{2,1}b_{3,1} + b_{1,2}b_{2,1}b_{3,2} + b_{1,2}b_{2,2}b_{3,1} + b_{1,2}b_{2,2}b_{3,2}\\ &\quad = (b_{1,1}+b_{1,2})\cdot(b_{2,1}+b_{2,2})\cdot(b_{3,1}+b_{3,2}).\end{aligned}$$

□

Proof: (Proof of Lemma D.17)

$$\begin{aligned}g^\perp(\tilde{\mathbf{x}}) = \sum_{\mathbf{x}\in\mathcal{F}^n} g(\mathbf{x})(-1)^{\mathbf{x}\cdot\tilde{\mathbf{x}}} &\overset{(*)}{=} \sum_{\mathbf{x}\in\mathcal{F}^n}\prod_{\ell\in\mathcal{I}}\underbrace{g_\ell(x_\ell)(-1)^{x_\ell\tilde{x}_\ell}}_{\text{corresponds to } b_{\ell,x_\ell}}\\ &\overset{(**)}{=} \prod_{\ell\in\mathcal{I}}\sum_{x\in\mathcal{F}} g_\ell(x)(-1)^{x\tilde{x}_\ell} \overset{(***)}{=} \prod_{\ell\in\mathcal{I}} g_\ell^\perp(\tilde{x}_\ell),\end{aligned}$$

where step $(*)$ follows from (D.4) and the step $(**)$ from Lemma D.20 and step $(***)$ from the definitions in the Lemma statement. □

D.7.5 Proof of Theorem D.18

Fix an $i \in \mathcal{I}$. Consider $g(x_\ell) \triangleq \mu_\ell(x_\ell)$ such that $g(\mathbf{x}) = \prod_{\ell=1}^n \mu_\ell(x_\ell)$. We have $g^\perp(\tilde{\mathbf{x}}) = \prod_{\ell=1}^n \mu_\ell^\perp(\tilde{x}_\ell)$ by Lemma D.17. Then,

$$\begin{aligned}\eta_i(0)+\eta_i(1) &= \sum_{\substack{\mathbf{x}\in\mathcal{C}\\ x_i=0}}\prod_{\ell=1}^n \mu_\ell(x_\ell) + \sum_{\substack{\mathbf{x}\in\mathcal{C}\\ x_i=1}}\prod_{\ell=1}^n \mu_\ell(x_\ell) = \sum_{\mathbf{x}\in\mathcal{C}}\prod_{\ell=1}^n \mu_\ell(x_\ell)\\ &= \sum_{\mathbf{x}\in\mathcal{C}} g(\mathbf{x}) \overset{(*)}{=} \frac{1}{|\mathcal{C}^\perp|}\sum_{\tilde{\mathbf{x}}\in\mathcal{C}^\perp} g^\perp(\tilde{\mathbf{x}}) = \frac{1}{|\mathcal{C}^\perp|}\sum_{\tilde{\mathbf{x}}\in\mathcal{C}^\perp}\prod_{\ell=1}^n \mu_\ell^\perp(\tilde{x}_\ell)\\ &= \frac{\chi^\perp}{|\mathcal{C}^\perp|},\end{aligned}$$

where equality $(*)$ follows from Lemma D.16

Fix some $i \in \mathcal{I}$. Let $g_i(x_i) \triangleq \mu_i(x_i)(-1)^{x_i}$, from which $g_i^\perp(\tilde{x}_i) = \mu_i^\perp(\tilde{x}_i + 1)$. For $\ell \neq i$, let $g_\ell(x_\ell) \triangleq \mu_\ell(x_\ell)$, then $g_\ell^\perp(\tilde{x}_\ell) = \mu_\ell^\perp(\tilde{x}_\ell)$. $g(\mathbf{x}) = \prod_{\ell=1}^n g_\ell(x_\ell)$ has the WHT $g^\perp(\tilde{\mathbf{x}}) = \prod_{\ell=1}^n g_\ell^\perp(\tilde{x}_\ell)$ by Lemma D.17.

Then,

$$\begin{aligned}
\eta_i(0) - \eta_i(1) &= \sum_{\substack{\mathbf{x}\in\mathcal{C}\\ x_i=0}} \prod_{\ell=1}^{n} \mu_\ell(x_\ell) - \sum_{\substack{\mathbf{x}\in\mathcal{C}\\ x_i=1}} \prod_{\ell=1}^{n} \mu_\ell(x_\ell)\\
&= \sum_{\substack{\mathbf{x}\in\mathcal{C}\\ x_i=0}} \prod_{\ell=1}^{n} \mu_\ell(x_\ell) + \sum_{\substack{\mathbf{x}\in\mathcal{C}\\ x_i=1}} (-1)^{x_i} \prod_{\ell=1}^{n} \mu_\ell(x_\ell)\\
&= \sum_{\mathbf{x}\in\mathcal{C}} (-1)^{x_i} \prod_{\ell=1}^{n} \mu_\ell(x_\ell) = \sum_{\mathbf{x}\in\mathcal{C}} g(\mathbf{x}) \overset{(*)}{=} \frac{1}{|\mathcal{C}^\perp|} \sum_{\tilde{\mathbf{x}}\in\mathcal{C}^\perp} g^\perp(\tilde{\mathbf{x}})\\
&= \frac{1}{|\mathcal{C}^\perp|} \sum_{\substack{\tilde{\mathbf{x}}\in\mathcal{C}^\perp\\ \tilde{\mathbf{x}}'=\tilde{\mathbf{x}}+\mathbf{e}_i}} \prod_{\ell\in\mathcal{I}} \mu_\ell^\perp(\tilde{x}'_\ell) = \frac{1}{|\mathcal{C}^\perp|} \sum_{\tilde{\mathbf{x}}'\in\mathcal{C}_i^\perp} \prod_{\ell\in\mathcal{I}} \mu_\ell^\perp(\tilde{x}'_\ell) = \frac{\chi_i^\perp}{|\mathcal{C}^\perp|},
\end{aligned}$$

where equality $(*)$ follows from Lemma D.16. □

D.7.6 Proof of Theorem D.19

Without caring about proportionality factors, this result can be obtained by observing

$$\begin{pmatrix} a & -b \\ -b & a \end{pmatrix} \propto \mathbf{D} \cdot \begin{pmatrix} 1/(a+b) & 0 \\ 0 & 1/(a-b) \end{pmatrix} \cdot \mathbf{D}^{-1}$$

while combining Th. D.8, the dual of Th. D.18, and Eq. (D.1). □

D.7.7 Alternative Proof of Theorem D.19

For some $i \in \mathcal{I}$, let $g_i(x_i) \triangleq 1$ for all $x_i \in \mathcal{F}$, from which $g_i^\perp(\tilde{x}_i) = 2 \cdot [\tilde{x}_i = 0]$ for all $\tilde{x}_i \in \mathcal{F}$. For $\ell \neq i$, let $g_\ell(x_\ell) \triangleq \mu_\ell(x_\ell)$ for all $x_\ell \in \mathcal{F}$,

then $g_\ell^\perp(\tilde{x}_\ell) = \mu_\ell^\perp(\tilde{x}_\ell)$ for all $\tilde{x}_\ell \in \mathcal{F}$ and

$$\begin{aligned}
&\mu_{f\to X_i}(0) + \mu_{f\to X_i}(1)\\
&\quad= \sum_{\substack{\mathbf{x}\in\mathcal{C}\\ x_i=0}} \prod_{\substack{\ell=1\\ \ell\neq i}}^{n} \mu_\ell(x_\ell) + \sum_{\substack{\mathbf{x}\in\mathcal{C}\\ x_i=1}} \prod_{\substack{\ell=1\\ \ell\neq i}}^{n} \mu_\ell(x_\ell) = \sum_{\mathbf{x}\in\mathcal{C}} \prod_{\substack{\ell=1\\ \ell\neq i}}^{n} \mu_\ell(x_\ell)\\
&\quad= \sum_{\mathbf{x}\in\mathcal{C}} g(\mathbf{x}) \overset{(*)}{=} \frac{1}{|\mathcal{C}^\perp|} \sum_{\tilde{\mathbf{x}}\in\mathcal{C}^\perp} g^\perp(\tilde{\mathbf{x}}) = \frac{1}{|\mathcal{C}^\perp|} \sum_{\tilde{\mathbf{x}}\in\mathcal{C}^\perp} 2[\tilde{x}_i = 0] \prod_{\substack{\ell=1\\ \ell\neq i}}^{n} \mu_\ell^\perp(\tilde{x}_\ell)\\
&\quad= \frac{2}{|\mathcal{C}^\perp|} \sum_{\substack{\tilde{\mathbf{x}}\in\mathcal{C}^\perp\\ \tilde{x}_i=0}} \prod_{\substack{\ell=1\\ \ell\neq i}}^{n} \mu_\ell^\perp(\tilde{x}_\ell) = \frac{2}{|\mathcal{C}^\perp|} \mu_{f\to X_i}^\perp(0),
\end{aligned}$$

where equality $(*)$ follows from Lemma D.16.

For some $i \in \mathcal{I}$, let $g_i(x_i) \triangleq (-1)^{x_i}$ for all $x_i \in \mathcal{F}$, from which $g_i^\perp(\tilde{x}_i) = 2 \cdot [\tilde{x}_i = 1]$ for all $\tilde{x}_i \in \mathcal{F}$. For $\ell \neq i$, let $g_\ell(x_\ell) \triangleq \mu_\ell(x_\ell)$ for all $x_\ell \in \mathcal{F}$, then $g_\ell^\perp(\tilde{x}_\ell) = \mu_\ell^\perp(\tilde{x}_\ell)$ for all $\tilde{x}_\ell \in \mathcal{F}$ and

$$\begin{aligned}
&\mu_{f\to X_i}(0) - \mu_{f\to X_i}(1)\\
&\quad= \sum_{\substack{\mathbf{x}\in\mathcal{C}\\ x_i=0}} \prod_{\substack{\ell=1\\ \ell\neq i}}^{n} \mu_\ell(x_\ell) - \sum_{\substack{\mathbf{x}\in\mathcal{C}\\ x_i=1}} \prod_{\substack{\ell=1\\ \ell\neq i}}^{n} \mu_\ell(x_\ell) = \sum_{\mathbf{x}\in\mathcal{C}} (-1)^{x_i} \prod_{\substack{\ell=1\\ \ell\neq i}}^{n} \mu_\ell(x_\ell)\\
&\quad= \sum_{\mathbf{x}\in\mathcal{C}} g(\mathbf{x}) \overset{(*)}{=} \frac{1}{|\mathcal{C}^\perp|} \sum_{\tilde{\mathbf{x}}\in\mathcal{C}^\perp} g^\perp(\tilde{\mathbf{x}}) = \frac{1}{|\mathcal{C}^\perp|} \sum_{\tilde{\mathbf{x}}\in\mathcal{C}^\perp} 2[\tilde{x}_i = 1] \prod_{\substack{\ell=1\\ \ell\neq i}}^{n} \mu_\ell^\perp(\tilde{x}_\ell)\\
&\quad= \frac{2}{|\mathcal{C}^\perp|} \sum_{\substack{\tilde{\mathbf{x}}\in\mathcal{C}^\perp\\ \tilde{x}_i=1}} \prod_{\substack{\ell=1\\ \ell\neq i}}^{n} \mu_\ell^\perp(\tilde{x}_\ell) = \frac{2}{|\mathcal{C}^\perp|} \mu_{f\to X_i}^\perp(1),
\end{aligned}$$

where equality $(*)$ follows from Lemma D.16. Solving for the vector $(\mu_{f\to X_i}(0), \mu_{f\to X_i}(1))^T$ yields the desired result. □

Appendix E

Variations on a Minimum Distance Bound by Tanner

E.1 Introduction and Notation

In [114], Tanner presented a lower bound on the minimum distance of a binary code. Tanner derived that bound (which is restated here in Th. E.2) having regular LDPC codes in mind, but the bound can be used for any binary code fulfilling certain restrictions.

In this appendix we are looking at variations on this bound; we will especially focus on codes with a certain automorphism group. Our main result is that a code which has a two-transitive automorphism group fulfills $(d_{\min}-1)(d_{\min}^{\perp}-1) \geq n-1$, where $d_{\min}$ and $d_{\min}^{\perp}$ are the minimum distance of the code and of its dual code, respectively (see Th. E.9, but also Rem. E.10).

We will consider binary $[n, k, d_{\min}]$ codes, i.e. codes of length n, dimension k, minimum distance $d_{\min}$ and rate $R = k/n$. Let a specific parity-check matrix $\mathbf{H}$ of this code have dimension $r \times n$ with $r \geq n-k$. As we are looking at binary codes, the entries of $\mathbf{H}$ are 0 and 1 from $\mathbb{F}_2$,

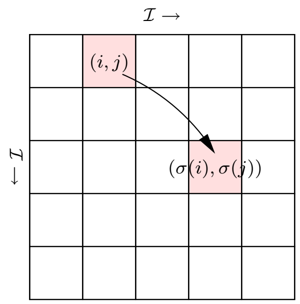

Figure E.1: Illustration of the action of σ on $\mathcal{I} \times \mathcal{I}$ (here: $\sigma(0) = 2$, $\sigma(1) = 3, \ldots$).

respectively. But in this appendix we will assume that the matrices and vectors have *real entries*, so the 0's and 1's in the parity-check matrix are considered to be real numbers.

We define the set $\mathcal{I} \triangleq \{0, \ldots, n-1\}$ with cardinality $|\mathcal{I}| = n$ which indexes the columns of $\mathbf{H}$ and the set $\mathcal{R} \triangleq \{0, \ldots, r-1\}$ of cardinality $|\mathcal{R}| = r$ which indexes the rows of $\mathbf{H}$. Let $\mathcal{A}$ be the automorphism group of the code $\mathcal{C}$; note that the dual code $\mathcal{C}^{\perp}$ has the same automorphism group. A permutation $\sigma \in \mathcal{A}$ is a bijective mapping (permutation) from $\mathcal{I}$ to $\mathcal{I}$; such a mapping induces a bijective mapping of $\mathcal{I} \times \mathcal{I}$ to $\mathcal{I} \times \mathcal{I}$ given by $\sigma((i,j)) = \big(\sigma(i), \sigma(j)\big)$ (for an illustration, see Figure E.1). Let $\mathbf{L}$ be the matrix $\mathbf{L} \triangleq \mathbf{H}^T \cdot \mathbf{H}$ of size $n \times n$, $\mathbb{1}$ be the identity matrix and $\mathbf{J}$ be the matrix consisting only of ones. $[\mathbf{H}]_{i,j}$ is the entry of $\mathbf{H}$ in row i and column j, whereas $[\mathbf{x}]_i$ is the i-th entry of a vector $\mathbf{x}$. In general, row indices of matrices are taken modulo the number of rows of the matrix and column indices of matrices are taken modulo the number of columns of the matrix.

E.2 Minimum Distance Bound for Binary Codes

Definition E.1 (Notation) Let $\mathbf{H}$ be a parity-check matrix and let

$\mu_1 > \mu_2 > \cdots > \mu_s$ be the ordered eigenvalues of $\mathbf{L} \triangleq \mathbf{H}^T\mathbf{H}$. Note that the non-zero eigenvalues of $\mathbf{H}^T\mathbf{H}$ and $\mathbf{H}\mathbf{H}^T$ and their multiplicities are the same [114]. When $\mathbf{H}$ has uniform row weight we call this weight w_{row} and when it has uniform column weight we call this weight w_{col}. □

We will use the following theorem which gives a lower bound on the minimum distance of a binary code [114].

Theorem E.2 (Bit-oriented lower bound on the minimum distance) *Let a binary code $\mathcal{C}$ of length n have the parity-check matrix $\mathbf{H}$ which has uniform column weight w_{col} and uniform row weight w_{row}. If the multiplicity of the largest eigenvalue μ_1 of $\mathbf{L}$ is one*[1]*, then*

$$d_{\min} \geq n \cdot \frac{2w_{\mathrm{col}} - \mu_2}{\mu_1 - \mu_2}.$$

Proof: See Sec. E.7.1. □

Corollary E.3 *Let $\mathbf{H}$ fulfill the conditions of Theorem E.2. If $\mathbf{L}$ has the form $\mathbf{L} = (a-b) \cdot \mathbb{1} + b \cdot \mathbf{J}$ (the matrix $\mathbf{L}$ thus has the value a in the main diagonal and the value b everywhere else) with $a \geq b > 0$, then*

$$d_{\min} \geq \frac{a}{b} + 1.$$

Proof: See Sec. E.7.2. □

E.3 Invariance Conditions of L

In this section we are considering binary codes with automorphism group $\mathcal{A}$. We assume that the parity-check matrix consists of *all* the codewords in the dual code which have the same weight $w = w_{\mathrm{row}}$. We assume that they span the whole dual code.[2] Under these circumstances there is for each $\sigma \in \mathcal{A}$ an induced permutation $\rho_\sigma : \mathcal{R} \rightarrow \mathcal{R}$ of the rows, i.e. $[\mathbf{H}]_{i,j} = [\mathbf{H}]_{\rho_\sigma(i), \sigma(j)}$ (see Figure E.2 for an illustration).

[1] See the comments in Sec. E.6.

[2] If this condition is not satisfied, our lower bounds are still true. This is because if $\mathcal{C}^{(1)} \subseteq \mathcal{C}^{(2)}$ or equivalently if $\mathcal{C}^{(1)\perp} \supseteq \mathcal{C}^{(2)\perp}$ then $d_{\min}^{(1)} \geq d_{\min}^{(2)}$. A lower bound for $d_{\min}^{(2)}$ is therefore also a lower bound for $d_{\min}^{(1)}$.

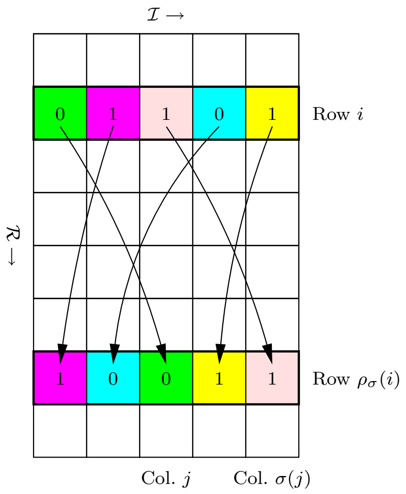

Figure E.2: Illustration of the of the action of (ρ_σ, σ) on the matrix $\mathbf{H}$ (here: $\sigma(0) = 2$, $\sigma(1) = 0$, $\sigma(2) = 4$, $\sigma(3) = 1$, $\sigma(4) = 3$).

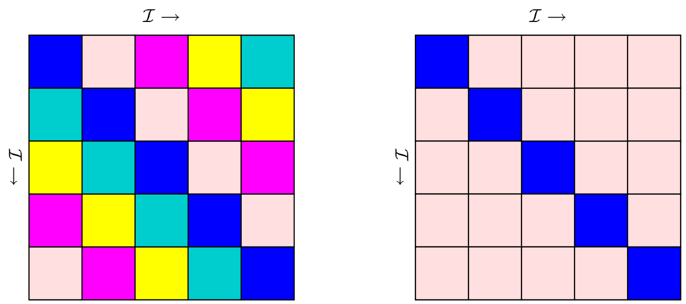

Figure E.3: The orbits of $\mathbf{L}$ in the case of the automorphism group $\mathcal{A}$ having a cyclic subgroup of order n (left) and having a two-transitive automorphism group (right).

Lemma E.4 *Within each orbit given by the action of $\mathcal{A}$ on $\mathcal{I} \times \mathcal{I}$, the entries of $\mathbf{L}$ are equal.*

Proof: See Sec. E.7.3. □

Corollary E.5 *If the automorphism group is one-transitive on the set $\mathcal{I}$ then the column weight of $\mathbf{H}$ is uniform.*

Proof: See Sec. E.7.4. □

Corollary E.6 *Let $\mathcal{G}$ be a subgroup of the automorphism group $\mathcal{A}$ of a code and assume that $\mathcal{G}$ is cyclic of order n. Then the column weight of $\mathbf{H}$ is uniform and there exists a labeling of the columns of $\mathbf{H}$ such that $\mathbf{L}$ is circulant. (See also Figure E.3 (left).)*

Proof: See Sec. E.7.5. □

E.4 Codes which are Two-Transitive on the Bits

In this section we are considering binary codes that are two-transitive on the bits. We assume that the rows of the parity-check matrix are built up by all codewords in the dual code which have the same weight $w = w_{\text{row}}$. (We assume that they span the whole dual code. See also the comment in Footnote 2.)

Corollary E.7 *Assume that the automorphism group $\mathcal{A}$ is two-transitive on $\mathcal{I}$. Then $\mathbf{H}$ has uniform column weight and $\mathbf{L}$ has the special form $\mathbf{L} = (a - b) \cdot \mathbb{1} + b \cdot \mathbf{J}$ for certain integers a and b. (The matrix $\mathbf{L}$ thus has the value a in the main diagonal and the value b everywhere else.) (See also Figure E.3 (right).)*

Proof: See Sec. E.7.6. □

We now try to express the values a and b with the help of the parameters of $\mathbf{H}$.

Lemma E.8 *The integers a and b in Cor. E.7 fulfill*

$$a = w_{\text{col}}, \quad b = \frac{w_{\text{row}} - 1}{n - 1} w_{\text{col}}, \quad \frac{a}{b} = \frac{n - 1}{w_{\text{row}} - 1}.$$

(Note that in the fraction a/b the values r and w_{col} do not *appear. r and w_{col} are generally "unknown" in the sense that they can only be determined by finding out how many rows* **H** *has.)*

Proof: See Sec. E.7.7. □

Theorem E.9 *If the automorphism group $\mathcal{A}$ of a code is two-transitive, then*

$$(d_{\min} - 1) \cdot (w_{\mathrm{row}} - 1) \geq (n - 1).$$

If $w = w_{\mathrm{row}} = d_{\min}^{\perp}$ then

$$(d_{\min} - 1) \cdot (d_{\min}^{\perp} - 1) \geq (n - 1),$$

and if the code is self-dual then

$$d_{\min} \geq \sqrt{n - 1} + 1,$$

Proof: See Sec. E.7.8. See also Rem. E.10. □

Remark E.10 (Elimination of variables) To be able to get such a general result, it was important that r (the number of rows of $\mathbf{H}$) and w_{col} (the number of ones per column of $\mathbf{H}$) do not appear in the expression for a/b in the main Lemma E.8. If r and w_{col} are known one could possibly find better numerical bounds. □

E.4.1 Application of the Bound to Extended Quadratic Residue Codes

The automorphism group $\mathcal{A}$ of extended quadratic residue codes $\hat{\mathcal{Q}}$ of length n (where $4|n$ and n is a prime plus one) is the group $\mathrm{PSL}_2(\mathbb{F}_{n-1})$ which is known to be two-transitive (see [76], p. 490–491). They are self-dual codes and therefore by Theorem E.9

$$d_{\min} \geq \sqrt{n - 1} + 1.$$

One can potentially derive slightly tighter bounds by modifying the proof of Theorem E.2. The problem is then that in the fraction a/b the "unknowns" r and w_{col} do not disappear anymore, they generally have to be determined otherwise. If they are known, one can possibly get better lower bounds.

We compare this with the "usual" bounds [76]. From a bound in [76] we get $d_{\min} \geq \sqrt{n-1} + \varepsilon$, and from a slightly tighter bound we obtain $d_{\min} \geq \sqrt{n - 7/4} + 1/2 + \varepsilon$, where $\varepsilon = 1$ if the extended code has a minimum distance which is by one larger than the not extended code and $\varepsilon = 0$ if the extended code has a minimum distance which equals that of the not extended code (see [76], p. 483).

E.5 Extensions

Assume that $\mathbf{L} \leq (a-b)\mathbb{1}+b\mathbf{J}$, where this inequality has to be understood in the following way: each entry of the matrix on the left-hand side must be smaller equal to the corresponding matrix element on the right-hand side. Then one can prove that the bound $d_{\min} \geq a/b + 1$ still applies. Interesting are of course the cases where the ration a/b is significant or where one can say something in general. (The case where a subgroup of the automorphism group is two-transitive is kind of the best one can hope as there the ratio a/b is the largest for specific n and w_{row}.)

E.6 About the Multiplicity of the Largest Eigenvalue of a Matrix Associated to a Parity-Check Matrix

The purpose of this section is to comment on a technical point of Th. E.2. So, let us assume that $\mathbf{H}$ has uniform row weight $w_{\mathrm{row}}^{(\mathbf{H})}$ and uniform column weight $w_{\mathrm{col}}^{(\mathbf{H})}$.

Lemma E.11 *The matrix $\mathbf{L} \triangleq \mathbf{H}^T \cdot \mathbf{H}$ has uniform row weight $w_{\mathrm{row}}^{(\mathbf{L})}$ and uniform column weight $w_{\mathrm{col}}^{(\mathbf{L})}$, whereby $w_{\mathrm{row}}^{(\mathbf{L})} = w_{\mathrm{col}}^{(\mathbf{L})} = w_{\mathrm{row}}^{(\mathbf{H})} \cdot w_{\mathrm{col}}^{(\mathbf{H})}$.*

Proof: See Sec. E.7.9. □

Obviously, $\mathbf{L}$ is a symmetric matrix having non-negative integer entries. The matrix $\mathbf{L}$ can be interpreted as the adjacency matrix of a graph having n vertices. The entry $[\mathbf{L}]_{i,j}$ means that there are $[\mathbf{L}]_{i,j}$ edges between node i and node j for $i \neq j$ and the entry $[\mathbf{L}]_{i,i}$ means that node i has $[\mathbf{L}]_{i,i}$ self-loops. If $\mathbf{H}$ has uniform row and column degree then this graph has uniform degree $w_{\text{row}}^{(\mathbf{L})} = w_{\text{col}}^{(\mathbf{L})} = w_{\text{row}}^{(\mathbf{H})} \cdot w_{\text{col}}^{(\mathbf{H})}$ by Lemma E.11. Note that the number of components of this graph equals the number of components of the factor graph associated to the parity-check matrix $\mathbf{H}$.

The following is a fact from spectral graph theory.

Lemma E.12 *Let $\mathbf{A}$ be the adjacency matrix of a graph with uniform degree w. All eigenvalues are real, the largest eigenvalue is w and the multiplicity of the largest eigenvalue equals the number of components of the graph. Eigenvectors with eigenvalue w have the property that the entries corresponding to vertices in the same component of the graph have the same value.*

Proof: See e.g. Ch. 3 in [16]. □

E.7 Proofs

E.7.1 Proof of Theorem E.2

See [114]; the importance of the multiplicity of the largest eigenvalue being 1 is because we need to know the corresponding eigenvector for the proof.

Equality holds when two conditions are fulfilled: firstly, there must be a codeword $\mathbf{x}$ of minimum weight such that the vector $\mathbf{H} \cdot \mathbf{x}^T$ consists only of zeros and twos, and secondly, there must be only two different eigenvalues, i.e. one (largest) eigenvalue μ_1 of multiplicity one and an eigenvalue μ_2 of multiplicity $n-1$.[3] □

[3] The second condition (which is not a necessary condition) can be replaced by the less stringent condition that the projection of the different codewords having minimum distance onto the space spanned by the third to the n–th eigenvector must be zero. This happens e.g. in the case of finite generalized quadrangles, see Sec. 4.4.

E.7.2 Proof of Corollary E.3

We express the eigenvalues μ_1 and μ_2 of $\mathbf{L}$ in terms of a and b. As the matrix $\mathbf{L}$ is circulant, the eigenvalues are the values of the discrete Fourier transform of the first row. Transforming the vector $[a, b, \ldots, b] = [a-b, 0, \ldots, 0] + [b, b, \ldots, b]$ we get $[a-b, a-b, \ldots, a-b] + [nb, 0, \ldots, 0] = [a+(n-1)b, a-b, \ldots, a-b]$. Therefore there are only two different eigenvalues, namely $\mu_1 = a + (n-1)b$ with multiplicity 1 and $\mu_2 = a - b$ with multiplicity $n-1$. With the result

$$a = [\mathbf{L}]_{0,0} = \sum_{\ell \in \mathcal{R}} [\mathbf{H}^T]_{0,\ell} [\mathbf{H}]_{\ell,0} = \sum_{\ell \in \mathcal{R}} \left([\mathbf{H}]_{\ell,0}\right)^2 = \sum_{\ell \in \mathcal{R}} [\mathbf{H}]_{\ell,0} = w_{\text{col}},$$

we finally can restate the bit-oriented bound on the minimum distance in Theorem E.2 as

$$d_{\min} \geq n \cdot \frac{2w_{\text{col}} - \mu_2}{\mu_1 - \mu_2} = n \cdot \frac{2a - (a-b)}{a + (n-1)b - (a-b)} = n \cdot \frac{a+b}{nb} = \frac{a}{b} + 1.$$

This result can also be derived more directly by modifying the proof of Theorem E.2. □

E.7.3 Proof of Lemma E.4

We consider any automorphism $\sigma \in \mathcal{A}$ with induced permutation of the rows ρ_σ:

$$\begin{aligned}
[\mathbf{L}]_{i,j} &= \sum_{\ell \in \mathcal{R}} [\mathbf{H}^T]_{i,\ell} [\mathbf{H}]_{\ell,j} = \sum_{\ell \in \mathcal{R}} [\mathbf{H}]_{\ell,i} [\mathbf{H}]_{\ell,j} = \sum_{\ell \in \mathcal{R}} [\mathbf{H}]_{\rho_\sigma(\ell),\sigma(i)} [\mathbf{H}]_{\rho_\sigma(\ell),\sigma(j)} \\
&\overset{(*)}{=} \sum_{\ell' \in \mathcal{R}} [\mathbf{H}]_{\ell',\sigma(i)} [\mathbf{H}]_{\ell',\sigma(j)} = \sum_{\ell' \in \mathcal{R}} [\mathbf{H}^T]_{\sigma(i),\ell'} [\mathbf{H}]_{\ell',\sigma(j)} = [\mathbf{L}]_{\sigma(i),\sigma(j)},
\end{aligned}$$

where equality $(*)$ follows from the fact that ρ_σ is a bijection, so we can do a summation variable substitution $\ell' = \rho_\sigma(\ell)$. It follows that all elements of $\mathbf{L}$ corresponding to the entries with indices $\{(\sigma(i), \sigma(j)) \mid \sigma \in \mathcal{A}\}$ must have the same value. But these sets of indices are exactly the orbits given by the action of $\mathcal{A}$ on $\mathcal{I} \times \mathcal{I}$. □

E.7.4 Proof of Corollary E.5

As can be seen from a generalization of a calculation in the proof of Corollary E.3, the weight of column i of $\mathbf{H}$ equals $[\mathbf{L}]_{i,i}$. But as the automorphism group is one-transitive, the set $\{(i,i) \mid i \in \mathcal{I}\}$ is an orbit and thus all $[\mathbf{L}]_{i,i}$, $i \in \mathcal{I}$, are equal by Lemma E.4. $\square$

E.7.5 Proof of Corollary E.6

$\mathcal{G}$ is clearly one-transitive on $\mathcal{I}$ and thus the column weight of $\mathbf{H}$ is uniform by Cor. E.5. For a specific $\sigma \in \mathcal{G}$ of order n the bits of the code can be labeled in such a way that $\sigma(i) = i + 1 \,(\mathrm{mod}\ n)$. The orbits of $\mathcal{G}$ acting on $\mathcal{I} \times \mathcal{I}$ are $\{(i,j) \mid j - i = \text{const.} \,(\mathrm{mod}\ n)\}$. It follows that $[\mathbf{L}]_{i,j} = [\mathbf{L}]_{i+h,j+h}$ for $h \in \mathcal{I}$, which is the condition a matrix must fulfill to be circulant. $\square$

E.7.6 Proof of Corollary E.7

$\mathcal{A}$ is two-transitive and thus also one-transitive from which the uniform column weight of $\mathbf{H}$ follows by Corollary E.5. The fact that $\mathcal{A}$ is two-transitive on $\mathcal{I}$ is equivalent with the fact that the action of $\mathcal{A}$ on $\mathcal{I} \times \mathcal{I}$ has two orbits, namely $\{(i,i) \mid i \in \mathcal{I}\}$ and $\{(i,j) \mid i,j \in \mathcal{I}, i \neq j\}$. It follows that the matrix $\mathbf{L}$ is constant on the diagonal and constant everywhere else. $\square$

E.7.7 Proof of Lemma E.8

To determine a we can calculate the value of any entry in the diagonal of $\mathbf{L}$.

$$a = [\mathbf{L}]_{0,0} = \sum_{\ell \in \mathcal{R}} [\mathbf{H}^T]_{0,\ell} [\mathbf{H}]_{\ell,0} = \sum_{\ell \in \mathcal{R}} \left([\mathbf{H}]_{\ell,0}\right)^2 = \sum_{\ell \in \mathcal{R}} [\mathbf{H}]_{\ell,0} = w_{\mathrm{col}}.$$

To determine b we calculate the sum of all elements of $\mathbf{L}$ in two different ways. On the one hand,

$$\sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} [\mathbf{L}]_{i,j} = |\mathcal{I}| \cdot a + (|\mathcal{I}|^2 - |\mathcal{I}|) \cdot b = n \cdot a + (n^2 - n) \cdot b.$$

On the other hand, let $\mathcal{S}_\ell = \{i \,|\, [\mathbf{H}]_{\ell,i} = 1\}$ be the index set of the positions where $\mathbf{H}$ is 1 in line ℓ and we get

$$\begin{aligned}
\sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} [\mathbf{L}]_{i,j} &= \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} \sum_{\ell \in \mathcal{R}} [\mathbf{H}^T]_{i,\ell} [\mathbf{H}]_{\ell,j} = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} \sum_{\ell \in \mathcal{R}} [\mathbf{H}]_{\ell,i} [\mathbf{H}]_{\ell,j} \\
&= \sum_{\ell \in \mathcal{R}} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} [\mathbf{H}]_{\ell,i} [\mathbf{H}]_{\ell,j} = \sum_{\ell \in \mathcal{R}} \sum_{i \in \mathcal{S}_\ell} \sum_{j \in \mathcal{S}_\ell} 1 \cdot 1 = \sum_{\ell \in \mathcal{R}} |\mathcal{S}_\ell|^2 \\
&= \sum_{\ell \in \mathcal{R}} w_{\mathrm{row}}^2 = |\mathcal{R}| \cdot w_{\mathrm{row}}^2 = r \cdot w_{\mathrm{row}}^2,
\end{aligned}$$

where we have used $|\mathcal{S}_\ell| = w_{\mathrm{row}}$ for each ℓ. Equating the two expressions, we get $na + (n^2 - n)b = r w_{\mathrm{row}}^2$. With the help of $n \cdot w_{\mathrm{col}} = r \cdot w_{\mathrm{row}}$ (which follows from counting the entries of $\mathbf{H}$ in two different ways) and $a = w_{\mathrm{col}}$ this can be simplified to $n w_{\mathrm{col}} + (n^2 - n)b = (n w_{\mathrm{col}} / w_{\mathrm{row}}) \cdot w_{\mathrm{row}}^2$ or equivalently $w_{\mathrm{col}} + (n-1)b = w_{\mathrm{col}} w_{\mathrm{row}}$. Finally, we get $b = w_{\mathrm{col}}(w_{\mathrm{row}} - 1)/(n-1)$. $\square$

E.7.8 Proof of Theorem E.9

Corollary E.7 says that $\mathbf{L} = (a - b) \cdot \mathbb{1} + b \cdot \mathbf{J}$ with ratio a/b given in Lemma E.8: $a/b = (n-1)/(w_{\mathrm{row}} - 1)$. Inserting this in Corollary E.3 we finally get

$$d_{\min} \geq \frac{a}{b} + 1 = \frac{n-1}{w_{\mathrm{row}} - 1} + 1,$$

or equivalently $(d_{\min} - 1) \cdot (w_{\mathrm{row}} - 1) \geq (n - 1)$. $\square$

E.7.9 Proof of Lemma E.11

For row $i \in \mathcal{I}$ we have

$$\begin{aligned}
\sum_{j \in \mathcal{I}} [\mathbf{L}]_{i,j} &= \sum_{j \in \mathcal{I}} \sum_{\ell \in \mathcal{R}} [\mathbf{H}^T]_{i,\ell} [\mathbf{H}]_{\ell,j} = \sum_{j \in \mathcal{I}} \sum_{\ell \in \mathcal{R}} [\mathbf{H}]_{\ell,i} [\mathbf{H}]_{\ell,j} \\
&= \sum_{\ell \in \mathcal{R}} [\mathbf{H}]_{\ell,i} \underbrace{\sum_{j \in \mathcal{I}} [\mathbf{H}]_{\ell,j}}_{= w_{\mathrm{row}}^{(\mathbf{H})}} = w_{\mathrm{row}}^{(\mathbf{H})} \cdot \underbrace{\sum_{\ell \in \mathcal{R}} [\mathbf{H}]_{\ell,i}}_{= w_{\mathrm{col}}^{(\mathbf{H})}} = w_{\mathrm{row}}^{(\mathbf{H})} \cdot w_{\mathrm{col}}^{(\mathbf{H})}.
\end{aligned}$$

As this result is independent of i, we get uniform row weight. The uniform column weight follows easily from the fact that the matrix $\mathbf{L}$ is symmetric. $\square$

Appendix F

Tables

This appendix lists several tables used throughout the text.

q	p	$\left(\frac{p}{q}\right)$	size	GLB	girth	GUB	diam.	DUB
5	3	−1	120	4.60	6	9.12	6	10.98
7	3	−1	336	5.82	8	10.35	6	12.85
7	5	−1	336	3.97	6	7.70	6	9.09
11	3	+1	660	4.37	5		8	14.08
11	5	+1	660	2.98	6		6	9.93
11	7	−1	1320	4.22	6	7.64	6	9.09
13	3	+1	1092	4.67	7		9	15.00
13	5	−1	2184	5.51	8	9.24	7	11.41
13	7	−1	2184	4.56	6	7.98	6	9.62
13	11	−1	2184	3.70	6	6.86	6	7.99
17	3	−1	4896	9.05	12	13.58	11	17.73
17	5	−1	4896	6.18	8	9.90	9	12.42
17	7	−1	4896	5.11	6	8.54	7	10.44
17	11	−1	4896	4.15	6	7.30	6	8.66
17	13	+1	2448	2.21	6		4	7.62
19	3	−1	6840	9.46	12	13.98	11	18.34
19	5	+1	3420	3.66	5		7	11.97
19	7	+1	3420	3.03	5		6	10.08
19	11	+1	3420	2.46	5		6	8.37
19	13	−1	6840	4.05	6	7.13	6	8.43
19	17	+1	3420	2.08	3		4	7.23
23	3	+1	6072	5.71	9		12	18.12
23	5	−1	12144	6.93	8	10.65	8	13.55
23	7	−1	12144	5.73	8	9.16	7	11.38
23	11	−1	12144	4.65	6	7.81	7	9.42
23	13	+1	6072	2.44	3		5	8.33
23	17	−1	12144	3.94	6	6.92	6	8.13
23	19	−1	12144	3.79	6	6.73	5	7.86

Table F.1: Girth and diameter of selected $\mathsf{X}^{\mathrm{LPS}}_{q,p}$ graphs. ("GLB" means "girth lower bound", "GUB" means "girth upper bound", and "DUB" means "diameter upper bound".)

Name	s	t	Description
$Q(3,q)$	q	1	based on a nonsingular quadratic (classical)
$Q(4,q)$	q	q	based on a nonsingular quadratic (classical)
$Q(5,q)$	q	q^2	based on a nonsingular quadratic (classical)
$H(3,q)$	q^2	q	based on a nonsingular hermitian variety (classical)
$H(4,q)$	q^2	q^3	based on a nonsingular hermitian variety (classical)
$W(q)$	q	q	based on a symplectic polarity (classical)
$T_2(\mathbf{O})$	q	q	uses an oval $\mathbf{O}$ (non-classical)
$T_3(\mathbf{O})$	q	q^2	uses an ovoid $\mathbf{O}$ (non-classical)
$\mathrm{AS}(q)$	$q-1$	$q+1$	uses the FGQ $W(q)$ (non-classical)
$T_2^*(\mathbf{O})$	$q-1$	$q+1$	uses a hyperoval $\mathbf{O}$ (non-classical)
$T_2(\mathbf{O}_n)$	q	q	uses a hyperoval with nucleus on it (non-classical)
$T_2(\mathbf{O}_{m,n})$	$q+1$	$q-1$	uses a hyperoval with two special points on it (non-classical)
	q^2	q	many different constructions based on families of subgroups of certain groups (classical and non-classical)

Table F.2: Possible Constructions of FGQs of order (s,t) (according to [90]). q is always a prime power. "Classical" are the FGQs which were introduce by Tits in [118]. (As shown in [21], these are the FGQs that may be embedded in $\mathrm{PG}(d,q)$ for $d = 3,4,5$.)

q	n	k	$R = k/n$	$d_{\min}$
2	15	5	1/3	6
3	40	15	3/8	8
4	85	35	7/17	10
5	156	65	5/12	12
7	400	175	7/16	
8	585	287	287/585	
9	820	369	9/20	
11	1464	671	11/24	
13	2380	1105	13/28	
16	4369	2479	2479/4369	
17	5220	2465	17/36	

Table F.3: Class $\mathcal{C}^{(1)}$ codes from the FGQ $W(q)$ of order $(s,t) = (q,q)$. (Construction II.3 in [90].)

q	n	k	$R = k/n$	$d_{\min}$	$d_{\min}^{\mathrm{BB}}$	$d_{\min}^{\mathrm{PB}}$	$d_{\min}^{\mathrm{TB}}$
2	45	24	8/15	6	0	6	6
3	280	189	27/40		(void)	8	8
4	1105	844	844/1105			10	10

Table F.4: Class $\mathcal{C}^{(1)}$ codes from an FGQ of order $(s,t) = (q^2,q)$ which is the dual FGQ of $Q(5,q)$. (Construction II.1 in [90].)

q	n	k	$R = k/n$	$d_{\min}$	$d_{\min}^{\mathrm{BB}}$	$d_{\min}^{\mathrm{PB}}$	$d_{\min}^{\mathrm{TB}}$
2	16	9	9/16	4	4	4	4
3	45	24	8/15	6	6	6	6
4	96	50	25/48		8	8	8
5	175	90	18/35		10	10	10
7	441	224	32/63		14	14	14
8	640	341	341/640		16	16	16
9	891	450	50/99		18	18	18
11	1573	792	72/143		22	22	22
13	2535	1274	98/195		26	26	26

Table F.5: Class $\mathcal{C}^{(1)}$ codes from an FGQ of order $(s,t) = (q+1, q-1)$ which is the dual FGQ of AS(q). (Construction IV in [90].)

q	n	k	$R = k/n$	$d_{\min}$	$d_{\min}^{\mathrm{BB}}$	$d_{\min}^{\mathrm{PB}}$	$d_{\min}^{\mathrm{TB}}$
2	8	1	1/8	8	8	8	8
3	27	6	2/9	12	12	10	10
4	64	18	9/32	16	16	12	12
5	125	40	8/25		20	14	14
7	343	126	18/49		28	18	18
8	512	213	213/512		32	20	20
9	729	288	32/81		36	22	22
11	1331	550	50/21		44	26	26
13	2197	936	72/169		52	30	30

Table F.6: Class $\mathcal{C}^{(1)}$ codes from the FGQ AS(q) of order $(s,t) = (q-1, q+1)$. (Construction IV in [90].)

Class of partial geometry	α	$\lvert\mathcal{P}\rvert$	$\lvert\mathcal{L}\rvert$
Steiner 2-design	$s+1$	$st+s+1$	$\frac{(t+1)(st+s+1)}{s+1}$
Net	t	$(s+1)^2$	$(s+1)(t+1)$
Transversal design	s	$(s+1)(t+1)$	$(t+1)^2$
FGQ	1	$(s+1)(st+1)$	$(t+1)(st+1)$
Proper partial geometry	$1<\alpha<\min\{s,t\}$	$\frac{(s+1)(st+\alpha)}{\alpha}$	$\frac{(t+1)(st+\alpha)}{\alpha}$

Table F.7: Different partial geometries. Steiner 2-designs are also known as balanced incomplete block designs (BIBDs) with $\lambda = 1$.

Class of partial geometry	α	Minimum distance of code from class
Steiner 2-design	$s+1$	$d_{\min} \geq \max\left\{t+2, \frac{2(t+s+1)}{(s+1)}\right\}$
Net	t	$d_{\min} \geq \max\left\{\frac{(s+1)(2t-s+1)}{t}, 4\right\}$
Transversal design	s	$d_{\min} \geq \max\left\{\frac{(s+1)(t+1)}{s}, \frac{2(s+t)}{s}\right\}$
FGQ	1	$d_{\min} \geq \max\{(s+1)(t+2-s), 2(t+1)\}$
Proper partial geometry	$1<\alpha$, $\alpha<\min\{s,t\}$	$d_{\min} \geq \max\left\{\frac{(s+1)(t+1-s+\alpha)}{\alpha}, \frac{2(t+\alpha)}{\alpha}\right\}$

Table F.8: Lower bounds on the minimal distance of codes from different classes of partial geometries (from [51], but note that there the codes are derived from the dual partial geometries). Steiner 2-designs are also known as balanced incomplete block designs (BIBDs) with $\lambda = 1$.

q	n	k	$R=k/n$	$d_{\min}$	$d_{\min}^{\mathrm{BB}}$	$d_{\min}^{\mathrm{PB}}$	$d_{\min}^{\mathrm{TB}}$	comments
2	27	6	2/9	12	12	10	10	best code has $d_{\min} \leq 12$
3	112	21	3/16	32	32	18	18	best known has $d_{\min} \leq 38$, best poss. is $d_{\min} \leq 44$

Table F.9: Class $\mathcal{C}^{(1)}$ codes from the FGQ $Q(5,q)$ of order $(s,t)=(q,q^2)$. (Construction II.1 in [90].)

q	n	k	$R=k/n$	$d_{\min}$	$d_{\min}^{\mathrm{BB}}$	$d_{\min}^{\mathrm{PB}}$	$d_{\min}^{\mathrm{TB}}$	comments
2	16	9	9/16	4	0	4	4	best code has $d_{\min} \leq 4$
4	96	50	25/48	8	0	8	8	(best code has $14 \leq d_{\min} \leq 21$)
8	640	341	341/640		0	16	16	

Table F.10: Class $\mathcal{C}^{(1)}$ codes from the FGQ of order $(s,t)=(q+1,q-1)$ which is the dual FGQ of $T_2^*(\mathbf{O})$. (Construction V.I in [90].)

q	n	k	$R = k/n$	d_{min}	$d_{\text{min}}^{\text{BB}}$	$d_{\text{min}}^{\text{PB}}$	$d_{\text{min}}^{\text{TB}}$	comments
2	8	1	1/8	8	8	8	8	best code has $d_{\text{min}} = 8$
4	64	18	9/32	16	16	12	12	best code has $d_{\text{min}} = 22$, such code exists
8	512	213	213/512		32	20	20	

Table F.11: Class $\mathcal{C}^{(1)}$ codes from the FGQ $T_2^*(\mathbf{O})$ of order $(s,t) = (q-1, q+1)$. (Construction V.I in [90].)

s	t	n	r	$d_{\text{min}}^{\text{BB}}$	$d_{\text{min}}^{\text{PB}} = d_{\text{min}}^{\text{TB}}$
q	q	q^3+q^2+q+1	q^3+q^2+q+1	$2(q+1)$	$2(q+1)$
q^2	q	$q^5+q^3+q^2+1$	q^4+q^3+q+1	(void)	$2(q+1)$
q	q^2	q^4+q^3+q+1	$q^5+q^3+q^2+1$	$(q+1)(q^2-q+2)$	$2(q^2+1)$
q^3	q^2	$q^8+q^5+q^3+1$	$q^7+q^5+q^2+1$	(void)	$2(q^2+1)$
q^2	q^3	$q^7+q^5+q^2+1$	$q^8+q^5+q^3+1$	$(q^2+1)(q^3-q^2+2)$	$2(q^3+1)$
$q+1$	$q-1$	q^3+2q	q^3	(void)	$2q$
$q-1$	$q+1$	q^3	q^3+2q	$4q$	$2(q+2)$

Table F.12: Lower bounds on the minimum distance as given by the (BB), (PB) and (TB). q is always a prime power.

s	t	n	r	$d_{\text{min}}^{\text{BB}}/n$	$d_{\text{min}}^{\text{PB}}/n = d_{\text{min}}^{\text{TB}}/n$
q	q	q^3	q^3	$2/q^2$	$2/q^2$
q^2	q	q^5	q^4	(void)	$2/q^4$
q	q^2	q^4	q^5	$1/q$	$2/q^2$
q^3	q^2	q^8	q^7	(void)	$2/q^6$
q^2	q^3	q^7	q^8	$1/q^2$	$2/q^4$
$q+1$	$q-1$	q^3	q^3	(void)	$2/q^2$
$q-1$	$q+1$	q^3	q^3	$4/q^2$	$2/q^2$

Table F.13: Asymptotic properties of the results in Tab. F.12. The minimum distance as given by the (BB), (PB) and (TB). q is always a prime power.

Abbreviations

AWGN	additive white Gaussian noise
AWGNC	additive white Gaussian noise channel
BCJR	Bahl, Cocke, Jelinek, and Raviv
BEC	binary erasure channel
BER	bit error rate
BI-AWGNC	binary-input additive white Gaussian noise channel
BIBD	balanced incomplete block design
BSC	binary symmetric channel
DFT	discrete Fourier transform
EW	Eulerian walk
FGH	finite generalized hexagon
FGO	finite generalized octagon
FGP	finite generalized polygon
FGQ	finite generalized quadrangle
FGT	finite generalized triangle
IG	interleaver graph
LDPC	low-density parity-check
LDPCC	low-density parity-check code
LPS	Lubotzky, Phillips, and Sarnak
MAP	maximum a-posteriori
ML	maximum likelihood
MPA	max-product algorithm
MRIP	most reliable independent positions
pdf	probability density function
pmf	probability mass function
QCRA	quasi-cyclic repeat-accumulate
QCRAC	quasi-cyclic repeat-accumulate code
RCC	recursive convolutional code

SPA	summary-product algorithm, sum-product algorithm
SS-LDPCC	strict-sense low-density parity-check codes
WS-LDPCC	wide-sense low-density parity-check codes

List of Symbols

Elementary

$\triangleq$	Definition
$\lvert \cdot \rvert$	Cardinality of a set
$\in$	Set membership
$\subset$	Proper subset
$\subseteq$	Subset
$\propto$	Proportional to
$\lceil \cdot \rceil$	The smallest integer not smaller than
$\lfloor \cdot \rfloor$	The largest integer not larger than
$\mathcal{S}$	Set
$\vec{\mathcal{S}}$	Directed set (used for directed graphs)
image	Image operator
mod	Modulo operator
$\left(\frac{a}{b}\right)$	Legendre symbol of a and b
$\square$	End of proof, example, definition, remark, algorithm, or construction

Algebra

$\mathbb{N}$	Natural numbers
$\mathbb{Z}$	Ring of integers
$\mathbb{Z}/m\mathbb{Z}$	Ring of integers modulo m
$\mathbb{Q}$	Field of rationals
$\mathbb{R}$	Field of reals
$\mathbb{C}$	Field of complex numbers
$\mathbb{F}_q$	Finite field with q elements
$\mathbb{H}(\mathcal{K})$	(Skew) Field of quaternions over a field $\mathcal{K}$
$\mathbb{H}(\mathbb{Z})$	Ring of integral quaternions
$\mathcal{G}$	Group
W	Word
$\mathcal{R}$	Ring
$\mathcal{R}^*$	(Multiplicative) group of invertible elements of ring $\mathcal{R}$
$\mathcal{K}$	Field
$\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$	Imaginary units in quaternion field
$N(\alpha)$	Norm of quaternion α
$\overline{\alpha}$	Conjugate of quaternion α
$\mathcal{F}_p$	Special quaternion family
$\mathcal{F}'_p$	Special quaternion family
Sp	Symplectic group
$R_b(a)$	Remainder of an integer a when divided by an integer b
$R_m(\mathcal{G})$	reduced word of length m consisting of elements of some group $\mathcal{G}$

Linear Algebra

$\mathbf{M}$	Matrix
$[\mathbf{M}]_{ij}$	Entry of matrix $\mathbf{M}$ in row i and column j
$\mathbf{v}$	Vector
$[\mathbf{v}]_i$	Entry of vector $\mathbf{v}$ at position i
$\mathbf{M}^T$	Transpose of matrix $\mathbf{M}$
$\mathbf{M}^H$	Hermitian transpose of matrix $\mathbf{M}$
$\mathcal{M}_2(\mathbb{Z})$	Ring of 2×2 matrices over $\mathbb{Z}$
$\mathrm{GL}_2(\mathcal{K})$	General linear group of 2×2-matrices over a field $\mathcal{K}$
$\mathrm{SL}_2(\mathcal{K})$	Special linear group of 2×2-matrices over a field $\mathcal{K}$
$\mathrm{PGL}_2(\mathcal{K})$	Projective general linear group of 2×2-matrices over a field $\mathcal{K}$
$\mathrm{PSL}_2(\mathcal{K})$	Projective special linear group of 2×2-matrices over a field $\mathcal{K}$
$\mathrm{diag}(\cdot)$	Diagonal matrix
trace	Trace operator
$\mathbb{1}$	Identity matrix
$\mathbb{1}^{(s)}$	Cyclically shifted identity matrix
$\lvert\lvert\mathbf{v}\rvert\rvert_2$	L_2 norm of vector $\mathbf{v}$
$\lvert\lvert\mathbf{M}\rvert\rvert_2$	matrix norm of $\mathbf{M}$ induced by the L_2 vector norm
$\otimes$	Kronecker product of two matrices

Geometry

$\mathbb{P}^1_{\mathcal{K}}$	Projective line over the field $\mathcal{K}$
PG	Projective geometry
AG	Affine geometry
Γ	Incidence structure
$\Gamma(\mathcal{P}, \mathcal{L}, \mathtt{I})$	Incidence structure with point set $\mathcal{P}$, line set $\mathcal{L}$, and incidence relation $\mathtt{I}$
$\mathcal{P}$	Point set
$\mathcal{L}$	Line set
$\mathtt{I}$	Incidence relation
$\mathbf{I}(\Gamma)$	Incidence matrix of an incidence structure Γ
$\mathbf{O}$	Oval

Coding

$\mathcal{C}$	Code
$\mathcal{C}^{\perp}$	Dual code of $\mathcal{C}$
$\mathbf{G}$	Generator matrix of a linear block code
$\mathbf{H}$	Parity-check matrix of a linear block code
$\mathcal{C}^{(1)}$	Code class 1
$\mathcal{C}^{(2)}$	Code class 2
$\mathcal{C}^{(3)}$	Code class 3
w_{H}	Hamming weight
d_{H}	Hamming distance
$d_{\min}$	Minimum distance
$d_{\min}^{\mathrm{BB}}$	Bit-oriented lower bound on the minimum distance
$d_{\min}^{\mathrm{PB}}$	Parity-oriented lower bound on the minimum distance
$d_{\min}^{\mathrm{TB}}$	Tree-oriented lower bound on the minimum distance
w_{col}	Column weight
w_{row}	Row weight
μ	Message
$\mu^{\perp}$	Dual message
η	(Used in Ch. D)
$\eta^{\perp}$	(Used in Ch. D)
χ	(Used in Ch. D)
$\chi^{\perp}$	(Used in Ch. D)
Λ	Likelihood ratio
$\Lambda^{\perp}$	Dual likelihood ratio
$\hat{u}_{\mathrm{MAP}}$	Maximum a-posteriori (MAP) estimate
$\hat{u}_{\mathrm{ML}}$	Maximum likelihood (ML) estimate
f_{XOR}	Indicator function of XOR function node

Graphs

X	(Undirected) graph
$\mathsf{X}(\mathcal{V}, \mathcal{E})$	(Undirected) graph with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$
$\vec{\mathsf{X}}$	Directed graph
$\vec{\mathsf{X}}(\mathcal{V}, \vec{\mathcal{E}})$	Directed graph with vertex set $\mathcal{V}$ and directed edge set $\vec{\mathcal{E}}$
$\mathsf{X}(\mathcal{G}, \mathcal{S})$	Cayley graph X over the group $\mathcal{G}$ and generator set $\mathcal{S}$
$\vec{\mathsf{X}}(\mathcal{G}, \mathcal{S})$	Directed Cayley graph $\vec{\mathsf{X}}$ over the group $\mathcal{G}$ and generator set $\mathcal{S}$
$\mathsf{X}_q^{\mathrm{Mar}}$	Graph defined by Margulis with parameter q
$\mathsf{X}_{q,p}^{\mathrm{LPS}}$	Ramanujan graph with parameters q and p by Lubotzky, Phillips, and Sarnak
$\mathsf{X}_{q,p_1,p_2}^{\mathrm{LPS}}$	Extended graph (based on LPS graphs) with parameters q, p_1, and p_2
$\mathcal{V}$	Vertex set
$\mathcal{E}$	Edge set
$\vec{\mathcal{E}}$	Directed edge set
T	Tree
F	Forest
$\mathbf{A}(\mathsf{X})$	Adjacency matrix of a graph X
$V\mathsf{X}$	Set of vertices of a graph X
$E\mathsf{X}$	Set of edges of a graph X
$B\mathsf{X}$	Set of (binary) variable nodes of a graph X
$C\mathsf{X}$	Set of check nodes of a graph X
$\mathrm{Comp}\,\mathsf{X}$	Number of components of a graph X
$\deg(v)$	Degree of a vertex v
$\mathrm{outdeg}(v)$	Out-degree of a vertex v
$\mathrm{indeg}(v)$	In-degree of a vertex v

Bibliography

[1] S. M. Aji, G. B. Horn, and R. J. McEliece, "Iterative decoding on graphs with a single cycle," in *Proc. IEEE Intern. Symp. on Inform. Theory*, MIT, Cambridge, MA, USA, Aug. 16-21 1998, p. 276.

[2] S. M. Aji and R. J. McEliece, "The generalized distributive law," *IEEE Trans. on Inform. Theory*, vol. 46, no. 2, pp. 325–343, 2000.

[3] N. Alon, "Eigenvalues and expanders," *Combinatorica*, vol. 6, no. 2, pp. 83–96, 1986, theory of computing (Singer Island, Fla., 1984).

[4] N. Alon and V. D. Milman, "λ_1, isoperimetric inequalities for graphs, and superconcentrators," *J. Combin. Theory Ser. B*, vol. 38, no. 1, pp. 73–88, 1985.

[5] D. Arnold, A. Kavčić, R. Kötter, H.-A. Loeliger, and P. O. Vontobel, "The binary jitter channel: a new model for magnetic recording," in *Proc. IEEE Intern. Symp. on Inform. Theory*, Sorrento, Italy, June 25–30 2000, p. 433.

[6] L. Bader and G. Lunardon, "Generalized hexagons and polar spaces," *Discrete Math.*, vol. 208/209, pp. 13–22, 1999, combinatorics (Assisi, 1996).

[7] B. Bagchi, A. E. Brouwer, and H. A. Wilbrink, "Notes on binary codes related to the $\mathrm{O}(5, q)$ generalized quadrangle for odd q," *Geom. Dedicata*, vol. 39, no. 3, pp. 339–355, 1991.

[8] B. Bagchi and N. S. N. Sastry, "Even order inversive planes, generalized quadrangles and codes," *Geom. Dedicata*, vol. 22, no. 2, pp. 137–147, 1987.

[9] ——, "Codes associated with generalized polygons," *Geom. Dedicata*, vol. 27, no. 1, pp. 1–8, 1988.

[10] ——, "One-step completely orthogonalizable codes from generalized quadrangles," *Inform. and Comput.*, vol. 77, no. 2, pp. 123–130, 1988.

[11] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. on Inform. Theory*, vol. IT–20, pp. 284–287, Mar. 1974.

[12] L. M. Batten, *Combinatorics of Finite Geometries*, 2nd ed. Cambridge: Cambridge University Press, 1997.

[13] L. Bazzi, M. Mahdian, S. Mitter, and D. Spielman, "The minimum distance of turbo-like codes," *preprint*, 2002.

[14] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes (1)," in *Proc. IEEE Int. Conf. Communications*, Geneva, Switzerland, May 1993, pp. 1064–1070.

[15] N. L. Biggs, *Discrete Mathematics*, 2nd ed. New York: The Clarendon Press and Oxford University Press, 1989.

[16] ——, *Algebraic Graph Theory*, 2nd ed. Cambridge: Cambridge University Press, 1993.

[17] N. L. Biggs and A. G. Boshier, "Note on the girth of Ramanujan graphs," *J. Combin. Theory Ser. B*, vol. 49, no. 2, pp. 190–194, 1990.

[18] J. Bond, S. Hui, and H. Schmidt, "Linear-congruence constructions of low-density parity-check codes," in *Codes, Systems, and Graphical Models (Minneapolis, MN, 1999)*, B. Marcus and J. Rosenthal, Eds. Springer Verlag, New York, Inc., 2001, pp. 83–100.

[19] R. C. Bose, "Strongly regular graphs, partial geometries and partially balanced designs," *Pacific J. Math.*, vol. 13, pp. 389–419, 1963.

[20] M. Breiling, "A logarithmic upper bound on the minimum distance of turbo codes," *submitted to IEEE Trans. Inform. Theory, available online under* `http://www.lnt.de/~breiling`, 2001.

[21] F. Buekenhout and C. Lefèvre, "Generalized quadrangles in projective spaces," *Arch. Math. (Basel)*, vol. 25, pp. 540–552, 1974.

[22] A. M. Cohen and B. N. Cooperstein, "Generalized hexagons of even order," *Discrete Math.*, vol. 106/107, pp. 139–146, 1992, a collection of contributions in honour of Jack van Lint.

[23] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. on Inform. Theory*, vol. 48, no. 6, pp. 1570–1579, 2002.

[24] C. Di, R. Urbanke, and T. Richardson, "Weight distributions: how deviant can you be?" in *Proc. IEEE Intern. Symp. on Inform. Theory*, Washington, D.C., USA, June 24-29 2001, p. 50.

[25] L. E. Dickson, "Arithmetic of quaternions," *Proc. London Math. Soc.*, vol. 20, no. 2, pp. 225–232, 1920.

[26] D. Divsalar and F. Pollara, "Turbo codes for deep-space communications," *JPL, TDA Progress Report*, vol. 42-120, Feb. 1995.

[27] S. Dolinar and D. Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations," *JPL, TDA Progress Report*, vol. 42-122, Aug. 1995.

[28] P. Erdős and H. Sachs, "Reguläre Graphen gegebener Taillenweite mit minimaler Knotenzahl," *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe*, vol. 12, pp. 251–257, 1963.

[29] T. Etzion, A. Trachtenberg, and A. Vardy, "Which codes have cycle-free Tanner graphs," *IEEE Trans. on Inform. Theory*, vol. IT–45, pp. 2173–2183, Sept. 1999.

[30] W. Feit and G. Higman, "The nonexistence of certain generalized polygons," *J. Algebra*, vol. 1, pp. 114–131, 1964.

[31] G. D. Forney, Jr., "Codes on graphs: normal realizations," *IEEE Trans. on Inform. Theory*, vol. 47, no. 2, pp. 520–548, 2001.

[32] G. D. Forney, Jr., R. Koetter, F. Kschischang, and A. Reznik, "On the effective weights of pseudocodewords for codes defined on graphs with cycles," in *Codes, Systems, and Graphical Models (Minneapolis, MN, 1999)*, ser. IMA Vol. Math. Appl., B. Marcus and J. Rosenthal, Eds. Springer Verlag, New York, Inc., 2001, vol. 123, pp. 101–112.

[33] M. P. C. Fossorier, "Iterative reliability-based decoding of low-density parity-check codes," *IEEE J. Sel. Areas Comm.*, vol. JSAC-19, pp. 908–917, May 2001.

[34] J. B. Fraleigh, *A First Course in Abstract Algebra.* Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills, Ont., 1967.

[35] B. J. Frey, *Graphical Models for Machine Learning and Digital Communication.* Cambridge, MA, USA: MIT Press, 1998.

[36] B. J. Frey, R. Koetter, and A. Vardy, "Signal-space characterization of iterative decoding," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 766–781, 2001.

[37] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inform. Theory*, vol. 8, pp. 21–28, Jan. 1962.

[38] ——, *Low-Density Parity-Check Codes.* M.I.T. Press, Cambridge, MA, 1963, available online under `http://justice.mit.edu/people/gallager.html`.

[39] R. Garello, F. Chiaraluce, P. Pierleoni, M. Scaloni, and S. Benedetto, "On error floor and free distance of turbo codes," in *Proc. IEEE Int. Conf. Communications*, vol. 1, Helsinki, Finland, June 11-14 2001, pp. 45–49.

[40] R. Garello, P. Pierleoni, and S. Benedetto, "Computing the free distance of turbo codes and serially concatenated codes with interleavers: algorithms and applications," *IEEE Trans. on Comm.*, vol. COMM-19, no. 5, pp. 800–812, May 2001, programs available under `http://www.tlc.ee.unian.it/ricerca/turbo/freedistance.html`.

[41] L. Gerritzen and M. van der Put, *Schottky Groups and Mumford Curves*. Berlin: Springer, 1980.

[42] G. H. Golub and C. F. van Loan, *Matrix Computations*. The Johns Hopkins University Press, 1996.

[43] C. R. P. Hartmann and L. D. Rudolph, "An optimum symbol-by-symbol decoding rule for linear codes," *IEEE Trans. on Inform. Theory*, Sept. 1976.

[44] D. Hösli and E. Svensson, *Low-Density Parity-Check Codes for Magnetic Recording*. Diploma Project, ETH Zurich, 2000.

[45] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Progressive edge-growth Tanner graphs," in *Proc. IEEE GLOBECOM*, San Antonio, TX, USA, Nov. 2001, pp. 995 –1001.

[46] ——, "Irregular progressive edge-growth (PEG) Tanner graphs," in *Proc. IEEE Intern. Symp. on Inform. Theory*, Lausanne, Switzerland, June 30–July 5 2002, p. 480.

[47] *Special Issue on 'Codes on graphs and iterative decoding'*, ser. IEEE Trans. on Inform. Theory, vol. IT–47. Institute of Electrical and Electronics Engineers, Feb. 2001.

[48] T. Johansson and K. Zigangirov, "A simple one-sweep algorithm for optimal APP symbol decoding of linear block codes," *IEEE Trans. on Inform. Theory*, vol. IT–44, pp. 3124–3129, Nov. 1998.

[49] S. J. Johnson and S. R. Weller, "Construction of low-density parity-check codes from Kirkman triple systems," in *Proc. IEEE Inform. Theory Workshop*, Cairns, Australia, Sep. 2-7 2001, pp. 90–92.

[50] ——, "Construction of low-density parity-check codes from Kirkman triple systems," in *Proc. IEEE GLOBECOM*, San Antonio, TX, USA, Nov. 2001, pp. 970–974.

[51] ——, "Codes for iterative decoding from partial geometries," in *Proc. IEEE Intern. Symp. on Inform. Theory*, Lausanne, Switzerland, June 30–July 5 2002, p. 310.

[52] N. Kahale, "On the second eigenvalue and linear expansion of regular graphs," in *Expanding graphs (Princeton, NJ, 1992)*, ser. DIMACS Ser. Discrete Math. Theoret. Comput. Sci. Providence, RI: Amer. Math. Soc., 1993, vol. 10, pp. 49–62.

[53] ——, "Eigenvalues and expansion of regular graphs," *J. Assoc. Comput. Mach.*, vol. 42, no. 5, pp. 1091–1106, 1995.

[54] K. Karplus and H. Krit, "A semi-systolic decoder for the PDSC-73 error-correcting code," *Discrete Applied Mathematics, North-Holland*, vol. 33, pp. 109–128, 1991.

[55] N. Koblitz, *p-adic Numbers, p-adic Analysis, and Zeta-Functions*, 2nd ed., ser. Graduate Texts in Mathematics. New York: Springer-Verlag, 1984, vol. 58.

[56] R. Koetter and P. O. Vontobel, "Graph-covers and iterative decoding of finite-length codes," in *3rd Intern. Conf. on Turbo Codes and Related Topics*, Brest, France, Sept. 1–5 2003.

[57] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," *IEEE Trans. on Inform. Theory*, vol. IT–47, pp. 2711–2736, Nov. 2001.

[58] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. on Inform. Theory*, vol. IT–47, no. 2, pp. 498–519, 2001.

[59] J. Lafferty and D. Rockmore, "Codes and iterative decoding on algebraic expander graphs," in *Proc. of ISITA 2000*, Hawaii, USA, 2000.

[60] D. Le Ruyet and H. V. Thien, "Design of cycle optimized interleavers for turbo codes," in *2nd Intern. Conf. on Turbo Codes and Related Topics*, Brest, France, Sept. 4–7 2000, pp. 335–338.

[61] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1983.

[62] H.-A. Loeliger, "New turbo-like codes," in *Proc. IEEE Intern. Symp. on Inform. Theory*, Ulm, Germany, June 29-July 4 1997, p. 109.

[63] ——, "Least squares and Kalman filtering on Forney graphs," in *Codes, Graphs, and Systems*, R. E. Blahut and R. Koetter, Eds. Kluwer, 2002, festschrift in honor of G. D. Forney, Jr.

[64] H.-A. Loeliger, M. Helfenstein, F. Lustenberger, and F. Tarköy, "Probability propagation and decoding in analog VLSI," in *Proc. IEEE Intern. Symp. on Inform. Theory*, MIT, Cambridge, MA, USA, Aug. 16-21 1998, p. 146.

[65] A. Lubotzky, R. Phillips, and P. Sarnak, "Ramanujan graphs," *Combinatorica*, vol. 8, no. 3, pp. 261–277, 1988.

[66] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs and belief propagation," in *Proc. IEEE Intern. Symp. on Inform. Theory*, MIT, Cambridge, MA, USA, Aug. 16-21 1998, p. 117.

[67] R. Lucas, M. Fossorier, Y. Kou, and S. Lin, "Iterative decoding of one-step majority logic decodable codes based on belief propagation," *IEEE Trans. on Comm.*, vol. COMM-48, pp. 931–937, June 2000.

[68] F. Lustenberger, "On the design of analog VLSI iterative decoders," Ph.D. dissertation, Swiss Federal Institute of Technology, Zurich, October 2000.

[69] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. on Inform. Theory*, vol. IT–45, no. 2, pp. 399–431, 1999.

[70] D. J. C. MacKay and M. C. Davey, "Low-density parity check codes over GF(q)," *IEEE Comm. Letters*, vol. 2, no. 6, pp. 165–167, 1998.

[71] ——, "Evaluation of Gallager codes for short block length and high rate applications," in *Codes, Systems, and Graphical Models (Minneapolis, MN, 1999)*, B. Marcus and J. Rosenthal, Eds. Springer Verlag, New York, Inc., 2001, pp. 113–130.

[72] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, no. 18, p. 1645, 29 Aug. 1996.

[73] ——, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, no. 6, pp. 457–458, 13 Mar. 1997.

[74] D. J. C. MacKay and M. S. Postol, "Weaknesses of Margulis and Ramanujan-Margulis low-density parity-check codes," *preprint*, 2002, available online under `http://www.inference.phy.cam.ac.uk/mackay/CodesRegular.html`.

[75] D. J. C. MacKay, S. T. Wilson, and M. C. Davey, "Comparison of constructions of irregular Gallager codes," *IEEE Trans. on Comm.*, vol. COMM–47, no. 10, pp. 1449–1454, 1999.

[76] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. New York: North-Holland, 1977.

[77] W. Magnus, A. Karrass, and D. Solitar, *Combinatorial Group Theory*, revised ed. New York: Dover Publications Inc., 1976, presentations of groups in terms of generators and relations.

[78] M. M. Mansour, M. M. Mansour, and N. R. Shanbhag, "On the architecture-aware structure of LDPC codes from generalized Ramanujan graphs and their decoder architectures," in *Proc. of Conference on Information Sciences and Systems*, Johns Hopkins University, Baltimore, MD, USA, Mar. 12–14 2003.

[79] Y. Mao and A. H. Banihashemi, "Decoding low-density parity-check codes with probabilistic scheduling," *IEEE Comm. Letters*, vol. 5, no. 10, pp. 414–416, 2001.

[80] ——, "A heuristic search for good low-density parity-check codes at short block lengths," in *Proc. 2001 IEEE Int. Conf. on Communications*, vol. 1, Helsinki, Finland, June 11–14 2001, pp. 41–44.

[81] Y. Mao and F. R. Kschischang, "On factor graphs and the Fourier transform," in *Proc. IEEE Intern. Symp. on Inform. Theory*, Washington, D.C., USA, June 24-29 2001, p. 224.

[82] G. A. Margulis, "Explicit constructions of graphs without short cycles and low density codes," *Combinatorica*, vol. 2, no. 1, pp. 71–78, 1982.

[83] ——, "Some new constructions of low-density parity-check codes," in *3rd. Intern. Seminar on Information Theory, Convolution Codes and Multi-User Communication*, Sochi, 1987, pp. 275–279.

[84] ——, "Explicit group-theoretic constructions of combinatorial schemes and their applications in the construction of expanders and concentrators," *Problemy Peredachi Informatsii*, vol. 24, no. 1, pp. 51–60, 1988.

[85] R. J. McEliece and M. Yildirim, "Belief propagation of partially ordered sets," in *Mathematical Systems Theory in Biology, Communication, Computation, and Finance, IMA Volumes in Math. & Appl.*, D. Gilliam and J. Rosenthal, Eds. Springer Verlag, 2003.

[86] M. Morgenstern, "Existence and explicit constructions of $q+1$ regular Ramanujan graphs for every prime power q," *J. Combin. Theory Ser. B*, vol. 62, no. 1, pp. 44–62, 1994.

[87] J. E. M. Nilsson and R. Kötter, "Iterative Decoding of Product Code Constructions," in *Proc. ISITA 94*, Sydney, Australia, Nov. 1994, pp. 1059–1064.

[88] M. E. O'Sullivan, M. Greferath, and R. Smarandache, "Construction of LDPC codes from affine permutation matrices," in *Proc. 40th Allerton Conf. on Communications, Control, and Computing*, Allerton House, Monticello, Illinois, USA, October 2–4 2002.

[89] S. E. Payne and J. A. Thas, *Finite Generalized Quadrangles*. Boston, Mass.: Pitman (Advanced Publishing Program), 1984.

[90] S. E. Payne, "A census of finite generalized quadrangles," in *Finite Geometries, Buildings, and Related Topics (Pingree Park, CO, 1988)*. New York: Oxford Univ. Press, 1990, pp. 29–36.

[91] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, ser. The Morgan Kaufmann Series in Representation and Reasoning. San Mateo, CA: Morgan Kaufmann, 1988.

[92] J. Polderman and J. Willems, *Introduction to Mathematical Systems Theory*. Springer-Verlag New York, Inc., 1998.

[93] J. G. Proakis, *Digital Communications*, 3rd ed. McGraw–Hill, 1995.

[94] T. Richardson and R. Urbanke, "Thresholds for turbo codes," in *Proc. IEEE Intern. Symp. on Inform. Theory*, Sorrento, Italy, June 25–30 2000, p. 317.

[95] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. on Inform. Theory*, vol. 47, no. 2, pp. 619–637, 2001.

[96] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. on Inform. Theory*, vol. 47, no. 2, pp. 599–618, 2001.

[97] ——, "Efficient encoding of low-density parity-check codes," *IEEE Trans. on Inform. Theory*, vol. 47, no. 2, pp. 638–656, 2001.

[98] J. Rosenthal and P. O. Vontobel, "Constructions of LDPC codes using Ramanujan graphs and ideas from Margulis," in *Proc. of the 38th Allerton Conference on Communication, Control, and Computing*, Allerton House, Monticello, Illinois, USA, Oct. 4–6 2000, pp. 248–257.

[99] ——, "Construction of regular and irregular LDPC codes using Ramanujan graphs and ideas from Margulis," in *Proc. IEEE Intern. Symp. on Inform. Theory*, Washington, D.C., USA, June 24–29 2001, p. 4.

[100] P. Rusmevichientong and B. Van Roy, "An analysis of belief propagation on the turbo decoding graph with Gaussian densities," *IEEE Trans. on Inform. Theory*, vol. IT–47, no. 2, pp. 745–765, 2001.

[101] P. Sarnak, *Some Applications of Modular Forms*. Cambridge: Cambridge University Press, 1990.

[102] N. S. N. Sastry and P. Sin, "The code of a regular generalized quadrangle of even order," in *Group Representations: Cohomology, Group Actions and Topology (Seattle, WA, 1996)*. Providence, RI: Amer. Math. Soc., 1998, pp. 485–496.

[103] J.-P. Serre, *Arbres, amalgames,* SL_2. Paris: Société Mathématique de France, 1977, avec un sommaire anglais, Rédigé avec la collaboration de Hyman Bass, Astérisque, No. 46.

[104] ——, *Trees*. Berlin: Springer-Verlag, 1980, translated from the French by John Stillwell.

[105] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 623–656, July/Oct. 1948.

[106] M. A. Shokrollahi, "Capacity-achieving sequences," in *Codes, Systems, and Graphical Models (Minneapolis, MN, 1999)*, B. Marcus and J. Rosenthal, Eds. Springer Verlag, New York, Inc., 2001, pp. 153–166.

[107] M. Sipser and D. Spielman, "Expander codes," *IEEE Trans. on Inform. Theory*, vol. 42, pp. 1710–1722, Nov. 1996.

[108] O. Y. Takeshita and D. J. Costello, "New deterministic interleaver designs for turbo codes," *IEEE Trans. on Inform. Theory*, vol. IT–46, no. 6, pp. 1988–2006, Sept. 2000.

[109] R. M. Tanner, "A recursive approach to low-complexity codes," *IEEE Trans. on Inform. Theory*, vol. IT–27, pp. 533–547, Sept. 1981.

[110] ——, "Explicit concentrators from generalized N-gons," *SIAM J. Algebraic Discrete Methods*, vol. 5, no. 3, pp. 287–293, 1984.

[111] ——, "A transform theory for a class of group-invariant codes," *IEEE Trans. on Inform. Theory*, vol. IT–34, pp. 752–775, July 1988.

[112] ——, "On quasi-cyclic repeat-accumulate codes," in *Proc. of the 37th Allerton Conference on Communication, Control, and Computing*, Allerton House, Monticello, Illinois, USA, Sep. 22-24 1999, pp. 249–259.

[113] ——, "Toward an algebraic theory for turbo codes," in *2nd Intern. Conf. on Turbo Codes and Related Topics*, Brest, France, Sept. 4–7 2000, pp. 17–25.

[114] ——, "Minimum-distance bounds by graph analysis," *IEEE Trans. on Inform. Theory*, vol. IT–47, no. 2, pp. 808–821, 2001.

[115] R. M. Tanner, D. Sridhara, and T. Fuja, "A class of group-structured LDPC codes," in *Proc. of ICSTA 2001*, Ambleside, England, 2001.

[116] A. A. Terras, *Fourier Analysis on Finite Groups and Applications*, ser. London Mathematical Society Student Texts. Cambridge: Cambridge University Press, 1999, vol. 43.

[117] J.-P. Tillich and G. Zémor, "Optimal cycle codes constructed from Ramanujan graphs," *SIAM J. Discrete Math.*, vol. 10, no. 3, pp. 447–459, 1997.

[118] J. Tits, "Sur la trialité et certains groupes qui s'en déduisent," *Inst. Hautes Etudes Sci. Publ. Math.*, vol. 2, pp. 14–60, 1959.

[119] D. Truhachev, M. Lentmaier, O. Wintzell, and K. S. Zigangirov, "On the minimum distance of turbo codes," in *Proc. IEEE Intern. Symp. on Inform. Theory*, Lausanne, Switzerland, June 30–July 5 2002, p. 84.

[120] H. van Maldeghem, *Generalized Polygons*. Basel: Birkhäuser Verlag, 1998.

[121] A. Vardy and F. R. Kschischang, "Proof of a conjecture of McEliece regarding the expansion index of the minimal trellis," *IEEE Trans. on Inform. Theory*, vol. IT–42, pp. 2027–2034, Nov. 1996.

[122] P. O. Vontobel, *Kalman Filters, Factor Graphs, and Electrical Networks*. Post-Diploma Project, ETH Zurich, 2002, available online under `http://www.isi.ee.ethz.ch/publications`.

[123] P. O. Vontobel and H.-A. Loeliger, “Irregular codes from regular graphs,” in *Proc. IEEE Intern. Symp. on Inform. Theory*, Lausanne, Switzerland, June 30–July 5 2002, p. 284.

[124] ——, “On factor graphs and electrical networks,” in *Mathematical Systems Theory in Biology, Communication, Computation, and Finance, IMA Volumes in Math. & Appl.*, D. Gilliam and J. Rosenthal, Eds. Springer Verlag, 2003.

[125] P. O. Vontobel and R. M. Tanner, “Construction of codes based on finite generalized quadrangles for iterative decoding,” in *Proc. IEEE Intern. Symp. on Inform. Theory*, Washington, D.C., USA, June 24–29 2001, p. 223.

[126] Y. Weiss and W. T. Freeman, “On the optimality of the max-product belief propagation algorithm in arbitrary graphs,” *IEEE Trans. on Inform. Theory*, vol. IT–47, no. 2, pp. 736–744, 2001.

[127] S. R. Weller and S. J. Johnson, “Iterative decoding of codes from oval designs,” in *Proc. Defense Applications of Signal Proc. (DASP) 2001*, 2001.

[128] N. Wiberg, “Codes and decoding on general graphs,” Ph.D. dissertation, Linköping University, Sweden, 1996.

[129] N. Wiberg, H.-A. Loeliger, and R. Kötter, “Codes and iterative decoding on general graphs,” *Europ. Trans. on Telecomm.*, vol. 6, pp. 513–525, Sept./Oct. 1995.

[130] J. Yedidia, J. Chen, and M. P. C. Fossorier, “Generating code representations suitable for belief propagation decoding,” in *Proc. 40th Allerton Conf. on Communications, Control, and Computing*, Allerton House, Monticello, Illinois, USA, October 2–4 2002.

[131] J. S. Yedidia, W. T. Freeman, and Y. Weiss, “Bethe free energy, Kikuchi approximations, and belief algorithms,” *preprint, available under* `http://www.merl.com`, 2001.

[132] ——, “Constructing free energy approximations and generalized belief propagation algorithms,” *preprint, available under* `http://www.merl.com`, Aug. 2002.

Index

D

E

About the Author

Pascal O. Vontobel was born in Zurich, Switzerland, on August 21, 1972. After visiting primary school in Greifensee, Switzerland, he attended the Gymnasium in Zurich-Oerlikon, Switzerland, where he obtained a Matura Typus C. In 1992 he joined the Swiss Federal Institute of Technology Zurich (ETH) to study electrical engineering; in 1997 he graduated with a Diploma degree in electrical engineering. Afterwards, he started as a research and teaching assistant at the Signal and Information Processing Laboratory (ISI) at ETH, where he got his post-diploma degree in Information Technology [122] in 2002 and his Ph.D. in 2003. He is currently a postdoctoral research associate at the Coordinated Science Laboratory at the University of Illinois at Urbana-Champaign, IL, USA.