DISSERTATIONES
INFORMATICAE
UNIVERSITATIS
TARTUENSIS
5

YAUHEN YAKIMENKA

Failure Structures of Message-Passing Algorithms in Erasure Decoding and Compressed Sensing

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

5

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

5

YAUHEN YAKIMENKA

Failure Structures of Message-Passing Algorithms in Erasure Decoding and Compressed Sensing

UNIVERSITY OF TARTU
Press

Institute of Computer Science, Faculty of Science and Technology, University of Tartu, Estonia.

Dissertation has been accepted for the commencement of the degree of Doctor of Philosophy (PhD) in informatics on January 29, 2019 by the Council of the Institute of Computer Science, University of Tartu.

Supervisor

Assoc. Prof.	Vitaly Skachek Institute of Computer Science University of Tartu Tartu, Estonia

Opponents

Prof.	Jens Zumbrägel Faculty of Computer Science and Mathematics University of Passau Passau, Germany
Prof.	Jörg Kliewer Department of Electrical and Computer Engineering New Jersey Institute of Technology Newark, USA

The public defence will take place on March 15, 2019 at 14:15 in Liivi 2-405.

The publication of this dissertation was financed by the Institute of Computer Science, University of Tartu.

ISSN 2613-5906
ISBN 978-9949-77-983-3 (print)
ISBN 978-9949-77-984-0 (PDF)

University of Tartu Press
http://www.tyk.ee/

To the memory of my father
To my mother who will not understand a word
To all my friends who became my family
To my supervisor who became my teacher

ABSTRACT

It was Claude Shannon who started the whole area of information theory back in 1948. His fundamental result was as follows: whatever bad channel you have, there is always a way to send information reliably (i.e. with vanishing probability of error) if you encode large enough blocks of information together. In this thesis, we consider linear codes (which are in fact linear subspaces) over the binary erasure channel (BEC). This channel allows only one kind of error: a bit can be erased. Otherwise the correct value of the bit is received.

In the early 1960s Robert Gallager suggested new linear codes named low-density parity-check (LDPC) codes. They allow for fast iterative (more precisely, message-passing) decoding. However, the performance of short and medium-length codes is suboptimal. On the BEC, it is known that the parity-check matrix used for message-passing decoding can be extended by adjoining redundant rows in order to improve decoding performance. Chapter 2 is dedicated to improvement of upper bounds on the number of these redundant rows (so-called *stopping redundancy*). We improve the best-known bounds and also generalise the concept of stopping redundancy. The chapter also includes extensive numerical experiments to support the theoretical material.

Another problem, known as compressed sensing, started from works of Emmanuel Candès and Terence Tao, and independently David Donoho. It was observed that many important signals can be represented as sparse vectors. The authors suggested to compress such signals on-the-fly, implicitly multiplying them by a measurement matrix. However, the problem of reconstructing the original signal is proven to be NP-hard. Thus, many alternative suboptimal algorithms were suggested. One of them, the interval-passing algorithm (IPA), is the central for the second half of the thesis. More precisely, we ask a question what are the conditions for the algorithm to fail or to succeed. In Chapter 3, we give a complete graph-theoretic criterion of failures. As a case study, we analyse parity-check matrices of array LDPC codes and obtain many results on their failures when used as measurement matrices for the IPA.

In this thesis, we consider failures of both message-passing decoding of LDPC codes and the IPA for compressed sensing. We find many similarities between these two problems and techniques used for their analysis.

CONTENTS

FIGURES

TABLES

NOMENCLATURE AND ABBREVIATIONS

PCM	parity-check matrix
LDPC	low-density parity-check (code)
MP	message-passing (decoding)
MAP	maximum a posteriori (decoding)
ML	maximum-likelihood (decoding)
BEC	binary-erasure channel
IPA	interval-passing algorithm
SRE	standard random ensemble

$\mathbb{F}$	finite (or *Galois*) field
$\mathbb{F}_2$	finite field of size 2
$\mathbb{R}$	field of real numbers
$\mathbb{R}_{\geq 0}$	set of non-negative real numbers
$X^{m \times n}$	set of $m \times n$ matrices with elements from the set X; X is usually a field
$\boldsymbol{x} \cdot \boldsymbol{y}$	scalar product of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$: $$\boldsymbol{x} \cdot \boldsymbol{y} \triangleq \sum_i x_i y_i$$
$[n]$	set of integers $\{1, 2, \ldots, n\}$
$\mathcal{C}$	code
$\mathcal{C}^\perp$	dual code of $\mathcal{C}$
$\mathcal{C}_0^\perp$	set of dual codewords except the all-zero codeword: $$\mathcal{C}_0^\perp = \mathcal{C}^\perp \setminus \{\mathbf{0}\}$$
$\mathfrak{C}$	ensemble of codes
$\mathfrak{S}(n, m)$	standard random ensemble of linear codes with parity-check matrices of size $m \times n$
$\mathfrak{Gal}(n, J, K)$	Gallager ensemble of (J, K)-regular LDPC codes of length n
$\mathbb{P}\left\{\cdot\right\}$	probability measure
$\mathbb{E}\left\{\cdot\right\}$	expected value
$\mathbb{E}_{\mathfrak{C}}\left\{\cdot\right\}$	expected value over ensemble $\mathfrak{C}$
$\mathbb{I}\{\cdot\}$	indicator function

$\Phi(\cdot)$	cumulative distribution function of the standard normal distribution: $$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} \mathrm{d}t$$
$\Phi^{-1}(\cdot)$	inverse of $\Phi(\cdot)$
$S(x, y)$	Stirling number of the second kind (the number of ways to partition a set of x labelled objects into y non-empty unlabelled subsets): $$S(x, y) = \frac{1}{y!} \sum_{j=0}^{y} (-1)^{k-j} \binom{y}{j} j^x$$
$H_{\mathcal{S}}$	matrix formed from columns of the matrix H indexed by the set $\mathcal{S}$
$\|\boldsymbol{x}\|_0$	ℓ_0-norm of vector $\boldsymbol{x}$: $$\|\boldsymbol{x}\|_0 = \sum_i \mathbb{I}\{x_i \neq 0\}$$
$\|\boldsymbol{x}\|_1$	ℓ_1-norm of vector $\boldsymbol{x}$: $$\|\boldsymbol{x}\|_1 = \sum_i \lvert x_i \rvert$$
$\lfloor x \rfloor$	floor function, the greatest integer less than or equal to x
$\lceil x \rceil$	ceiling function, the least integer greater than or equal to x

PREFACE

I started working towards obtaining my PhD degree back in September of 2014, although the first results on stopping redundancy hierarchy were obtained in my master's thesis.

The presented findings are from two different—on the face of it—fields, iterative channel decoding and compressed sensing. The problem about failures of the interval-passing algorithm (IPA) for compressed sensing was suggested to me when I was on my five-month research visit to the University of Bergen. As it turned out, we found many similarities and analogies and often used a similar set of tools in the course of research. In particular, we introduced the concept of *termatiko sets* (from Greek τερματικό 'terminal', 'final') for the IPA which play exactly the same role as *stopping sets* for the iterative decoding over the BEC.

The thesis is written in such a way that a reader with decent undergraduate background in algebra, probability theory and some other widely-known mathematical disciplines will grasp the contents. That is to say, no previous knowledge of information theory, error-correction codes, or compressed sensing is strictly required.

I have got plenty of help over these years. The first one to thank is—without any doubt—my amazing supervisor, Dr Vitaly Skachek. It is him who has introduced me to the world of scientific research. I have learnt from him innumerable skills important for a researcher. He has been constantly giving me freedom to speak out and supporting my ideas. They truly say that the most important component of your doctoral studies is your supervisor.

During the second year, I had a pleasure to spend five months in the University of Bergen thanks to support of the Norwegian-Estonian Research Cooperation Program. On the Norwegian side, my visit was organised by professor Øyvind Ytrehus. Because of his help, my visit was fruitful as I could concentrate purely on research.

While in Bergen, I was closely co-operating with Dr Eirik Rosnes on an everyday basis. Without any doubt, he taught me a lot. That probably was the most efficient time in the course of my PhD studies. In fact, a large part of Chapter 3 is a result of those five months. Moreover, the rest of the chapter is a result of our remote collaboration after my return to Tartu.

I have also enjoyed productive work with my other co-authors, namely, Dr Irina E Bocharova and Dr Boris D Kudryashov. A big part of my current expertise is their merit. I should also acknowledge Alexander Vardy for pointing out a problem of exponential/polynomial growth of stopping redundancy.

My opponents, Professor Jens Zumbrägel and Professor Jörg Kliewer, did amazing job in reviewing this thesis. At first, I was impatient as—in my opinion—they took too long to read it and give their feedback. However, they managed to point out some mistakes in the draft version which I had overlooked. I am genuinely grateful for this.

Very often, the calculations for this thesis were carried out in the High Performance Computing Centre of the University of Tartu. I have never met in person the colleagues working there, but I believe they have done their best so that hardware and software works as intended. I remember only two or three cluster failures during these four years. Luckily, my jobs were not affected.

The Institute of Computer Science and the University of Tartu in general have provided me with a relaxed but inspiring atmosphere which is indispensable for a good research. I would like to extend thanks to all my colleagues there, both current and former.

As they say, money makes the world go round. I am truly grateful for scholarships from Skype and Information Technology Foundation for Education (HITSA), grant EMP133 from the Norwegian-Estonian Research Cooperation Programme, grants PUT405, PRG49, and IUT2-1 from Estonian Research Council, short-term mobility grants from University of Tartu ASTRA project PER ASPERA Doctoral School of Information and Communication Technologies (ICT Doctoral School), as well as support by European Regional Development Fund through the Estonian Centre of Excellence in Computer Science (EXCS).

This thesis would not have been finished without the endless heartening from my friends and family. The latter has unfortunately become smaller in number in the course of last years.

The last but not the least, I would also like to show my appreciation to all Estonian, Norwegian, and European taxpayers whose money were indirectly used to provide me support during these times.

Yauhen Yakimenka
Tartu, January 2019

1. INTRODUCTION

The only excuse for making a useless thing is that one admires it intensely.
—Oscar Wilde, *The Picture of Dorian Gray*

In this chapter, we introduce the required concepts and notation, as well as give an overview of the existing results.

We start with basic definitions and then review some of the standard concepts and facts about channel coding. Next, we discuss main decoding principles and algorithms and introduce the concept that is central for Chapter 2, *stopping redundancy* of a linear code.

After that, we compile some basic facts from the field of compressed sensing in Section 1.3. We look more closely at the interval-passing algorithm (IPA).

We accompany the material with detailed examples.

1.1. Basic definitions

Consider a finite field $\mathbb{F}$ and let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ be a vector[1] of length n with entries from $\mathbb{F}$. A *support* of a vector is the set of indices of non-zero entries in the vector:

$$\operatorname{supp}(\boldsymbol{x}) = \{i : x_i \neq 0\}.$$

The *Hamming weight* of a vector is the cardinality of its support:

$$\mathrm{w}(\boldsymbol{x}) = |\operatorname{supp}(\boldsymbol{x})|.$$

For two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, we define the *Hamming distance* as the number of positions they are different in. In other words,

$$\mathrm{d}(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{w}(\boldsymbol{x} - \boldsymbol{y}).$$

For a positive integer n, we denote $[n] \triangleq \{1, 2, \ldots, n\}$.

Let $H = (h_{ji})$ be an $m \times n$ matrix. We associate with H the bipartite *Tanner graph* $G = (V \cup C, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of nodes corresponding to columns of H, $C = \{c_1, c_2, \ldots, c_m\}$ is a set of nodes corresponding to rows of H, and E is a set of edges between C and V. We will often associate V with $[n]$ and C with $[m]$. There is an edge in E between $c \in C$ and $v \in V$ if and only if $h_{cv} \neq 0$.

We also denote the set of neighbours for each node $v \in V$ and $c \in C$ as follows:

$$\mathcal{N}(v) = \{c \in C : (c, v) \in E\}, \tag{1.1}$$

$$\mathcal{N}(c) = \{v \in V : (c, v) \in E\}. \tag{1.2}$$

Furthermore, if $T \subset V$ or $T \subset C$ and $w \in V \cup C$, then define

$$\mathcal{N}(T) = \bigcup_{t \in T} \mathcal{N}(t) \text{ and } \mathcal{N}_T(w) = \mathcal{N}(w) \cap T.$$

1.2. Stopping redundancy hierarchy

1.2.1. Communication problem

In his groundbreaking paper [47], Shannon suggested separating the general communication problem into *source coding* and *channel coding*. The source encoder converts a source message—which can be a text, multimedia, or other kinds of data—into a stream of symbols from some alphabet. In most of the cases, this alphabet is a field, and in particular the binary finite field $\mathbb{F}_2$, i.e. the symbols are bits. The source encoder also attempts to remove as much redundancy as possible from the original message, e.g. by applying some compression algorithm. At the

[1] Throughout the thesis, we use the terms "vector" and "word" interchangeably.

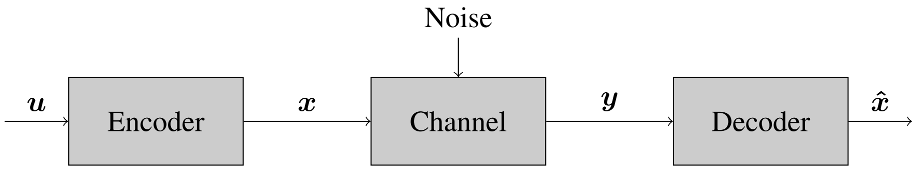


Figure 1. Noisy channel transmission.

next, separate stage, the channel encoder transforms this stream of symbols by judiciously adding redundancy in order to overcome the noise arising from the channel.

In this thesis, we consider only the channel coding problem. That is, we have a sequence of symbols as an input. Fig. 1 schematically describes a general setting of transmission over a noisy channel. Due to noise, the channel output $\boldsymbol{y}$ is in general different from the channel input $\boldsymbol{c}$ but stochastically depends on it. The time is usually *discrete* (and synchronised) and we can denote the channel input and output at time t as x_t and y_t, respectively. The channel is said to be *memoryless*, that is, the output at time t depends only on the input x_t, and the conditional probability distribution $\mathbb{P}\left\{y_t \mid x_t\right\}$ does not change with time. Namely, for mutually independent $x_1, x_2, \ldots, x_T$,

$$\mathbb{P}\left\{y_1, y_2, \ldots, y_T \mid x_1, x_2, \ldots, x_T\right\} = \prod_{t=1}^{T} \mathbb{P}\left\{y_t \mid x_t\right\}.$$

If the output alphabet is continuous, $\mathbb{P}\left\{\cdot \mid \cdot\right\}$ should be understood as probability density function instead. However, in this work, we only consider discrete-output channels unless opposite stated explicitly.

According to Shannon's *channel coding theorem*, for each channel—i.e. for each distribution $\mathbb{P}\left\{y_t \mid x_t\right\}$—there exists a supremum C of achievable rates, $C \in [0, 1]$, called the *capacity* of the channel. More precisely, for each $R < C$, there is a way to encode and decode the input symbols in such a way that the ratio of information in the transmission is R (in other words, the ratio of redundancy is $1 - R$) and decoding error probability vanishes when large enough blocks of data are encoded together.

Elias introduced a model of the *erasure channel* in 1954 as a toy example (cf. [12]). In spite of that, with the expansion of computer networks and, substantially, Internet, this channel attracted much of attention in "real world". It can be seen as a model for the network with packets that can either arrive unchanged or be lost completely—for instance, if time limit exceeded. Besides, many properties and results obtained in an easier way for erasure channel further remain valid in a much broader context—which is rather unforeseen.

The main setting we are interested in is the *binary erasure channel (BEC)*.

Definition 1. The *binary erasure channel (BEC)* with erasure probability p is a discrete memoryless channel with input $x_t \in \mathbb{F}_2$ and output $y_t \in \mathbb{F}_2 \cup \{?\}$ (where

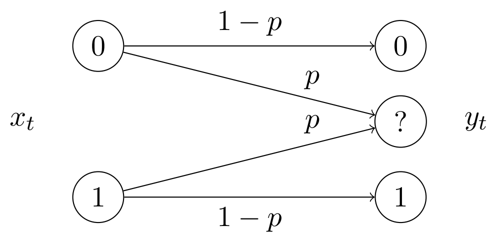


Figure 2. Binary erasure channel.

? denotes erasure) with conditional probability distribution

$$\mathbb{P}\left\{y_t \mid x_t\right\} = \begin{cases} p & \text{for } y_t = ? \text{ and } x_t \in \mathbb{F}_2, \\ 1-p & \text{for } x_t = y_t \in \mathbb{F}_2, \\ 0 & \text{otherwise.} \end{cases}$$

The bits are transmitted over the BEC one by one. Each bit x_t is erased with the probability p and remains unchanged with probability $1-p$, independently of other bits (see Fig. 2). The capacity of the BEC is $1-p$ (cf. [40, Sec. 3.1]).

1.2.2. Codes and ensembles

As it was stated above, it is beneficial to encode data in larger blocks. A *block code* over the finite field $\mathbb{F}$ is defined as any non-empty subset of $\mathbb{F}^n$, the set of length-n vectors with entries from $\mathbb{F}$. However, we restrict ourselves to *linear* codes only and we consider $\mathbb{F}^n$ as a vector space.

Definition 2. Let $\mathbb{F}$ be a finite field. The *linear (block) code* of length n is any (non-degenerate) subspace $\mathcal{C}$ of the vector space $\mathbb{F}^n$.

We interpret elements of $\mathcal{C}$ as *row* vectors and call them *codewords* of $\mathcal{C}$. Dimension k of $\mathcal{C}$ as a vector space is called the *dimension* of the code. From the definition it follows that $|\mathcal{C}| = |\mathbb{F}|^k$. The ratio $R = {}^k\!/_n$ is called the *rate* of the code.

Fix some k codewords from $\mathcal{C}$ that form a basis and write them as rows of a $k \times n$ matrix G. Then G has the rank k and it holds that

$$\mathcal{C} = \left\{ \boldsymbol{x} \in \mathbb{F}^n : \boldsymbol{x} = \boldsymbol{u}G, \boldsymbol{u} \in \mathbb{F}^k \right\}.$$

Such G is called the *generator matrix* as it generates all the codewords when $\boldsymbol{u}$ iterates through $\mathbb{F}^k$. We note that different generator matrices can describe the same code $\mathcal{C}$.

The general setting is the following (cf. Fig. 1). The information one wants to transmit is split into blocks of k symbols and each block $\boldsymbol{u} \in \mathbb{F}^k$ is then mapped by the encoder to a codeword $\boldsymbol{x} = \boldsymbol{u}G$, of length n. Therefore, each codeword intrinsically carries k information symbols and $r \triangleq n-k$ symbols of redundancy. Next, $\boldsymbol{x}$ is sent over the channel. The decoder receives a distorted version of the codeword, $\boldsymbol{y}$, and tries to reconstruct the original codeword. Its estimate of the codeword is usually denoted $\hat{\boldsymbol{x}}$. Since the correspondence between the message $\boldsymbol{u}$ and the codeword $\boldsymbol{x}$ is deterministic and bijective, correct estimate (i.e. $\hat{\boldsymbol{x}} = \boldsymbol{x}$) is considered as the success of decoding.

Particular type of distortions/noise depend on the channel—for example, erasure channel erases some of the symbols:

$$y_i = \begin{cases} x_i, & \text{if } i\text{-th symbol arrives unchanged,} \\ ?, & \text{if } i\text{-th symbol arrives erased.} \end{cases}$$

The *minimum distance* of a code $\mathcal{C}$ is defined as the minimum of distances between non-equal codewords:

$$d = \min\{\mathrm{d}(\boldsymbol{x}_1, \boldsymbol{x}_2) : \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{C}, \boldsymbol{x}_1 \neq \boldsymbol{x}_2\}.$$

It can be easily shown that for linear codes this definition is equivalent to the following:

$$d = \min\{\mathrm{w}(\boldsymbol{x}) : \boldsymbol{x} \in \mathcal{C} \setminus \{\mathbf{0}\}\}.$$

A linear code of length n with dimension k and minimum distance d is denoted as $[n, k, d]$.

Another way to describe a code is via its *parity-check matrix (PCM)*. PCM of a code $\mathcal{C}$ is any matrix H such that the following holds:

$$\boldsymbol{x} \in \mathcal{C} \quad \text{if and only if} \quad H\boldsymbol{x}^{\mathsf{T}} = \mathbf{0}^{\mathsf{T}}\,.$$

In other words, H is any matrix such that $\mathcal{C}$ is its kernel. It follows from the definition that H is $m \times n$ matrix of rank r for some $m \geq r$. We note that a parity-check matrix—and number of its rows—is not uniquely defined for the given code. In fact, it is very common to define a code via its parity-check matrix. In this thesis, this will be a convention.

For the binary case, it is not difficult to see that $\boldsymbol{x}$ is a codeword of $\mathcal{C}$ with the parity-check matrix H if and only if the columns of H indexed by elements of $\mathrm{supp}(\boldsymbol{x})$ sum up to the all-zero column vector.

For the fixed parity-check matrix H of a code $\mathcal{C}$, we often consider the Tanner graph of H and conventionally call it simply the Tanner graph of $\mathcal{C}$. We note that a Tanner graph is not uniquely defined for the code. But of course it is unique for a chosen parity-check matrix H.

Example 3 ($[7, 4, 3]$ Hamming code)**.** Consider as an example the $[7, 4, 3]$ Hamming code. The code is defined by its parity-check matrix:

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

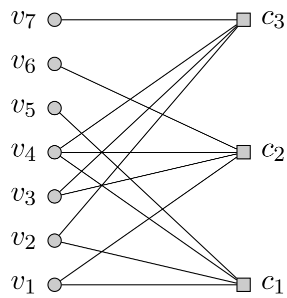


Figure 3. Tanner graph of the $[7, 4, 3]$ Hamming code.

Columns of H are all the binary 3-tuples except the all-zero tuple. The last three columns form the 3×3 unity matrix. Therefore, $\operatorname{rank} H = 3$ and the dimension of the code is $k = n - r = n - \operatorname{rank} H = 7 - 3 = 4$.

Further, let us show why the minimum distance of the code is indeed 3. As it was noted above, each codeword corresponds to the subset of columns in H that sum up to the all-zero column. There is neither the all-zero column nor two equal columns in H. Hence, the minimum distance of the code is at least 3. On the other hand, the first three columns sum up to the all-zero column and therefore $(1, 1, 1, 0, 0, 0, 0)$ is a codeword.

Fig. 3 depicts the Tanner graph corresponding to H. The variable nodes on the left correspond to the columns, and check nodes on the right match the rows of H.

An example of a generator matrix for the Hamming code can be the following:

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

One can easily verify that each row of G is orthogonal to each row of H. $\triangle$

Together with a code $\mathcal{C}$, we consider its *dual code* $\mathcal{C}^\perp$, defined as follows:

$$\mathcal{C}^\perp = \{\boldsymbol{h} \in \mathbb{F}^n : \boldsymbol{h} \cdot \boldsymbol{x} = 0\,, \forall \boldsymbol{x} \in \mathcal{C}\}.$$

That is, the dual code $\mathcal{C}^\perp$ consists of all vectors from $\mathbb{F}^n$ that are orthogonal to *all* codewords of $\mathcal{C}$.

All rows of the generator matrix of $\mathcal{C}$ are codewords of $\mathcal{C}$ and all rows of its parity-check matrix are codewords of $\mathcal{C}^\perp$. It is easy to show that if the parity-check matrix H of $\mathcal{C}$ has exactly r rows (that is, there are no redundant rows), it is then at the same time a generator matrix of $\mathcal{C}^\perp$. The matrix G is always a parity-check matrix of $\mathcal{C}^\perp$.

In what follows, we will consider only binary codes, i.e. codes over the field $\mathbb{F}_2 = \{0, 1\}$ (with operators "+" and "·").

A common method of code analysis is based on code ensembles. In general, an *ensemble* is simply a set of codes together with some probability distribution on this set. A typical approach is to define an ensemble by a uniformly random set of parity-check matrices. In that way, different parity-check matrices can define the same code. However, it is customary to say that one picks a code uniformly at random from an ensemble, while in actual fact, it is a parity-check matrix that is picked uniformly at random. As a result, the probability distribution on the set of codes is not necessarily uniform.

Example 4 (standard random ensemble)**.** The *standard random ensemble* (SRE) $\mathfrak{S}(n, m)$ is defined by means of its $m \times n$ parity-check matrices H, where each entry of H is an independent and identically distributed (i.i.d.) Bernoulli random variable with parameter $1/2$.

There are 2^{mn} different parity-check matrices in the ensemble, and every linear code $\mathcal{C}$ of the length n and the dimension $k \geq n - m$ is present in the ensemble. For $\mathcal{C}$, fix some $(n - k) \times n$ parity-check matrix H_0 of full row rank (i.e. without redundant rows). Then all $m \times n$ parity-check matrices of $\mathcal{C}$ are generated by matrices of coefficients $A \in \mathbb{F}_2^{m \times (n-k)}$ of rank $n - k$:

$$H = AH_0,$$

and there is a bijection between H and A. Therefore, the number of different $m \times n$ parity-check matrices defining $\mathcal{C}$ is equal to the number of binary $m \times (n-k)$ matrices of rank $n - k$ with $m \geq n - k$. The latter is known to be (cf. Lemma 47)

$$\mathcal{M}(m, n - k) = \prod_{i=0}^{n-k-1} (2^m - 2^i).$$

In other words, each linear code of rank $k \geq n - m$ has in $\mathfrak{S}(n, m)$ the probability

$$2^{-mn} \prod_{i=0}^{n-k-1} (2^m - 2^i). \qquad \triangle$$

It is often the case that all parity-check matrices defining ensemble have the same size, and thus the codes have the same length. However, this is not true for a code dimension or rate, as we do not usually guarantee that the rows in a considered parity-check matrix are linearly independent. The ratio ${}^{(n-m)}/_{m}$ is called a *design rate* of a code and the real rate is *at least* the design rate.

In general, arguing about an ensemble can be easier than proving facts about individual codes. And in many cases, a random code from the ensemble behaves similarly to a *typical* code.

1.2.3. Low-density parity-check codes

Low-density parity-check (LDPC) codes were first introduced by Gallager in his groundbreaking thesis [16, 17] but then nearly forgotten for several decades. To

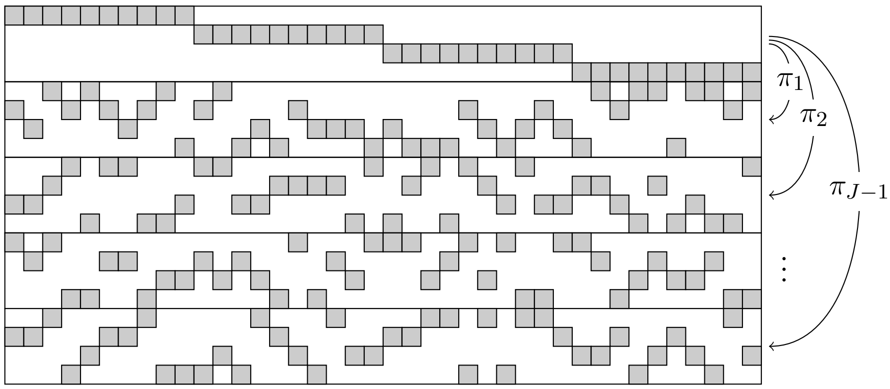


Figure 4. Schematic sketch of a random parity-check matrix from $\mathfrak{Gal}(n, J, K)$. Grey squares denote ones. The column permutations $\pi_1, \pi_2, \ldots, \pi_{J-1}$ are applied to the initial strip.

put it briefly, an LDPC code is a linear code with a *sparse* parity-check matrix (or, equivalently, a sparse Tanner graph). Gallager himself defined *regular* LDPC codes, such that a Tanner graph is both left- and right-regular. In other words, a parity-check matrix of a (J, K)-regular code has J ones in each column and K ones in each row. *Irregular* LDPC codes were introduced in the series of papers [34, 31, 32, 33]. Below we describe three particular kinds of LDPC codes.

The *Gallager ensemble* $\mathfrak{Gal}(n, J, K)$ of (J, K)-regular LDPC codes of length n [16, 17] is defined by parity-check matrices of a special form. An $({}^{nJ}/_{K}) \times n$ parity-check matrix consists of J strips of width $M = n/K$ rows each. In the first strip, the jth row contain K ones in positions $(j-1)K+1, (j-1)K+2, \ldots, jK$ for $j = 1, 2, \ldots, M$. And each of the other strips is a random column permutation π_i, $i = 1, 2, \ldots, J-1$, of the first strip. See Fig. 4 for schematic sketch.

The design rate of each code in the ensemble is $1 - {}^{J}/_{K}$. Yet the rank of a parity-check matrix in $\mathfrak{Gal}(n, J, K)$ cannot be more than

$$r_{\max} = \frac{nJ}{K} - (J-1)$$

due to the presence of redundant rows in *any* such matrix. Therefore, the actual rate of each code in the ensemble is at least

$$1 - \frac{J}{K} + \frac{J-1}{n},$$

although for large values of n the last term is insignificant.

The next ensemble of regular LDPC codes we consider is a special case of [40, Def. 3.15]. We refer to the ensemble as the *Richardson-Urbanke (RU) ensemble*.

For $a \in \{1, 2, ...\}$ denote by a^t the sequence $(a, a, ..., a)$ of t identical symbols a. In order to construct an $m \times n$ parity-check matrix H of an LDPC code from the RU ensemble, one does the following:

- construct the sequence $\boldsymbol{a} = (1^J, 2^J, ..., n^J)$;

- randomly permute $\boldsymbol{a}$ to obtain a sequence $\boldsymbol{b} = (b_1, ..., b_N)$, where $N = Km = Jn$;
- set to one the entries in the first row of H in columns $b_1, ..., b_K$, the entries in the second row of H in columns $b_{K+1}, ..., b_{2K}$, etc. The remaining entries of H are zeroes.

In fact, an LDPC code from the RU ensemble is (J, K)-regular if for given permutations all elements in each of the subsequences $(b_{iK-K+1}, ..., b_{iK})$, $i = 1, 2, \ldots, m$, are different. It is shown in [28] that the fraction of regular codes among the RU LDPC codes is roughly

$$e^{(K-1)(J-1)/2}.$$

In other words, most of the RU codes are irregular. In what follows, we ignore this fact and interpret them as (J, K)-regular codes, and call them "almost regular".

Example 5. Assume we want to generate a $(3, 4)$ (almost) regular parity-check matrix from the RU ensemble of length $n = 36$. We start with constructing the sequence:

$$\begin{aligned} \boldsymbol{a} = (&1,1,1,2,2,2,3,3,3,4,4,4,5,5,5,6,6,6,7,7,7,8,8,8,9,9,9,\\ &10,10,10,11,11,11,12,12,12,13,13,13,14,14,14,15,15,15,\\ &16,16,16,17,17,17,18,18,18,19,19,19,20,20,20,21,21,21,\\ &22,22,22,23,23,23,24,24,24,25,25,25,26,26,26,27,27,27,\\ &28,28,28,29,29,29,30,30,30,31,31,31,32,32,32,33,33,33,\\ &34,34,34,35,35,35,36,36,36). \end{aligned}$$

By applying a random permutation to it, we obtain:

$$\begin{aligned} \boldsymbol{b} = (&28,35,7,5, && 30,\mathbf{23},\mathbf{23},31, && 14,13,20,26, && 7,28,35,8,\\ &11,21,3,14, && 22,34,31,33, && 16,11,27,1, && 16,10,4,31,\\ &17,2,6,18, && 29,6,3,35, && 26,24,33,10, && 27,3,20,9,\\ &13,12,30,9, && 2,17,23,34, && 11,26,\mathbf{15},\mathbf{15}, && 2,29,21,36,\\ &20,5,19,30, && 22,12,27,13, && 33,22,32,29, && 7,34,6,24,\\ &16,14,36,8, && \mathbf{4},\mathbf{4},\mathbf{19},\mathbf{19}, && 12,17,5,21, && \mathbf{1},24,25,\mathbf{1},\\ &25,18,32,8, && 36,28,10,18, && 9,15,32,25). \end{aligned}$$

The numbers in bold repeat in their respective groups of four. The corresponding rows have weights less than four. The resulting parity-check matrix is

$$
\begin{pmatrix}
0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0\\
0&0&1&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&1&0&0\\
1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&1&0&0&0\\
0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&1&1&0&0&0\\
0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0
\end{pmatrix}
$$

and it defines a $[36, 9, 8]$ code. $\triangle$

The *quasi-cyclic (QC) LDPC codes* represent a class of LDPC codes that is intensively used in communication standards. A rate $R = b/c$ QC LDPC code is determined by the $(c-b) \times c$ polynomial parity-check matrix of its parent convolutional code [27]

$$
H(D) = \begin{pmatrix}
h_{11}(D) & h_{12}(D) & \dots & h_{1c}(D) \\
h_{21}(D) & h_{22}(D) & \dots & h_{2c}(D) \\
\vdots & \vdots & \ddots & \vdots \\
h_{(c-b)1}(D) & h_{(c-b)2}(D) & \dots & h_{(c-b)c}(D)
\end{pmatrix}.
$$

Here $h_{ij}(D)$ is either zero or a monomial entry in a formal variable D, that is, $h_{ij}(D) \in \{0, D^{w_{ij}}\}$ with w_{ij} being a non-negative integer, $w_{ij} \le \mu$, and $\mu = \max_{i,j}\{w_{ij}\}$ is called a syndrome memory.

The polynomial matrix $H(D)$ determines an $[Mc, Mb]$ QC LDPC block code using a set of polynomials modulo $D^M - 1$. By tailbiting the parent convolutional code to length $M > \mu$, we obtain the binary parity-check matrix

$$
H_{\mathrm{TB}} = \begin{pmatrix}
H_0 & H_1 & \dots & H_{\mu-1} & H_\mu & O & \dots & O \\
O & H_0 & H_1 & \dots & H_{\mu-1} & H_\mu & \dots & O \\
\vdots & & \ddots & \vdots & \vdots & \vdots & \ddots & \\
H_\mu & O & \dots & O & H_0 & H_1 & \dots & H_{\mu-1} \\
\vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
H_1 & \dots & H_\mu & O & \dots & O & \dots & H_0
\end{pmatrix}
$$

of an equivalent (in the sense of column permutation) *tailbiting* code (see [27, Ch. 2]), where H_i, $i = 0, 1, \dots, \mu$, are binary $(c-b) \times c$ matrices in the series expansion

$$
H(D) = H_0 + H_1 D + \dots + H_\mu D^\mu,
$$

and O is the all-zero matrix of size $(c-b) \times c$. If each column of $H(D)$ contains J non-zero elements, and each row contains K non-zero elements, the QC LDPC block code is (J, K)-regular. It is irregular otherwise.

Another form of an equivalent $[Mc, Mb]$ binary QC LDPC block code can be obtained by replacing the non-zero monomial elements of $H(D)$ by the powers of the circulant $M \times M$ permutation matrix P defined as follows:

$$P = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \end{pmatrix}.$$

The polynomial parity-check matrix $H(D)$ can be interpreted as a $(c-b) \times c$ binary base matrix B labelled by monomials, where the entry in B is one if and only if the corresponding entry of $H(D)$ is non-zero:

$$B = H(D)|_{D=1}.$$

All three matrices B, $H(D)$, and H can be interpreted as bi-adjacency matrices of the corresponding Tanner graphs.

Example 6. Consider the following 3×4 polynomial matrix:

$$H(D) = \begin{pmatrix} D^0 & D^0 & D^0 & D^0 \\ D^0 & D^1 & D^4 & D^6 \\ D^0 & D^5 & D^2 & D^3 \end{pmatrix}.$$

Using tailbiting length $M = 9$, we obtain the following $(3, 4)$-regular parity-check matrix:

$$\left(\begin{array}{cccc}
1\,0\,0\,0\,0\,0\,0\,0\,0 & 1\,0\,0\,0\,0\,0\,0\,0\,0 & 1\,0\,0\,0\,0\,0\,0\,0\,0 & 1\,0\,0\,0\,0\,0\,0\,0\,0 \\
0\,1\,0\,0\,0\,0\,0\,0\,0 & 0\,1\,0\,0\,0\,0\,0\,0\,0 & 0\,1\,0\,0\,0\,0\,0\,0\,0 & 0\,1\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,1\,0\,0\,0\,0\,0\,0 & 0\,0\,1\,0\,0\,0\,0\,0\,0 & 0\,0\,1\,0\,0\,0\,0\,0\,0 & 0\,0\,1\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,1\,0\,0\,0\,0\,0 & 0\,0\,0\,1\,0\,0\,0\,0\,0 & 0\,0\,0\,1\,0\,0\,0\,0\,0 & 0\,0\,0\,1\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,1\,0\,0\,0\,0 & 0\,0\,0\,0\,1\,0\,0\,0\,0 & 0\,0\,0\,0\,1\,0\,0\,0\,0 & 0\,0\,0\,0\,1\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,1\,0\,0\,0 & 0\,0\,0\,0\,0\,1\,0\,0\,0 & 0\,0\,0\,0\,0\,1\,0\,0\,0 & 0\,0\,0\,0\,0\,1\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,1\,0\,0 & 0\,0\,0\,0\,0\,0\,1\,0\,0 & 0\,0\,0\,0\,0\,0\,1\,0\,0 & 0\,0\,0\,0\,0\,0\,1\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,1\,0 & 0\,0\,0\,0\,0\,0\,0\,1\,0 & 0\,0\,0\,0\,0\,0\,0\,1\,0 & 0\,0\,0\,0\,0\,0\,0\,1\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,1 & 0\,0\,0\,0\,0\,0\,0\,0\,1 & 0\,0\,0\,0\,0\,0\,0\,0\,1 & 0\,0\,0\,0\,0\,0\,0\,0\,1 \\
\\
1\,0\,0\,0\,0\,0\,0\,0\,0 & 0\,1\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,0\,1\,0\,0\,0\,0 & 0\,0\,0\,0\,0\,0\,1\,0\,0 \\
0\,1\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,1\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,0\,0\,1\,0\,0\,0 & 0\,0\,0\,0\,0\,0\,0\,1\,0 \\
0\,0\,1\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,1\,0\,0\,0\,0\,0 & 0\,0\,0\,0\,0\,0\,1\,0\,0 & 0\,0\,0\,0\,0\,0\,0\,0\,1 \\
0\,0\,0\,1\,0\,0\,0\,0\,0 & 0\,0\,0\,0\,1\,0\,0\,0\,0 & 0\,0\,0\,0\,0\,0\,0\,1\,0 & 1\,0\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,1\,0\,0\,0\,0 & 0\,0\,0\,0\,0\,1\,0\,0\,0 & 0\,0\,0\,0\,0\,0\,0\,0\,1 & 0\,1\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,1\,0\,0\,0 & 0\,0\,0\,0\,0\,0\,1\,0\,0 & 1\,0\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,1\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,1\,0\,0 & 0\,0\,0\,0\,0\,0\,0\,1\,0 & 0\,1\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,1\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,1\,0 & 0\,0\,0\,0\,0\,0\,0\,0\,1 & 0\,0\,1\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,0\,1\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,1 & 1\,0\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,1\,0\,0\,0\,0\,0 & 0\,0\,0\,0\,0\,1\,0\,0\,0 \\
\\
1\,0\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,0\,0\,1\,0\,0\,0 & 0\,0\,1\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,1\,0\,0\,0\,0\,0 \\
0\,1\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,0\,0\,0\,1\,0\,0 & 0\,0\,0\,1\,0\,0\,0\,0\,0 & 0\,0\,0\,0\,1\,0\,0\,0\,0 \\
0\,0\,1\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,0\,0\,0\,0\,1\,0 & 0\,0\,0\,0\,1\,0\,0\,0\,0 & 0\,0\,0\,0\,0\,1\,0\,0\,0 \\
0\,0\,0\,1\,0\,0\,0\,0\,0 & 0\,0\,0\,0\,0\,0\,0\,0\,1 & 0\,0\,0\,0\,0\,1\,0\,0\,0 & 0\,0\,0\,0\,0\,0\,1\,0\,0 \\
0\,0\,0\,0\,1\,0\,0\,0\,0 & 1\,0\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,0\,0\,0\,1\,0\,0 & 0\,0\,0\,0\,0\,0\,0\,1\,0 \\
0\,0\,0\,0\,0\,1\,0\,0\,0 & 0\,1\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,0\,0\,0\,0\,1\,0 & 0\,0\,0\,0\,0\,0\,0\,0\,1 \\
0\,0\,0\,0\,0\,0\,1\,0\,0 & 0\,0\,1\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,0\,0\,0\,0\,0\,1 & 1\,0\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,1\,0 & 0\,0\,0\,1\,0\,0\,0\,0\,0 & 1\,0\,0\,0\,0\,0\,0\,0\,0 & 0\,1\,0\,0\,0\,0\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,1 & 0\,0\,0\,0\,1\,0\,0\,0\,0 & 0\,1\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,1\,0\,0\,0\,0\,0\,0
\end{array}\right)$$

with each block being a power of P. The parity-check matrix defines a $[36, 11, 6]$ code. $\triangle$

1.2.4. Decoding of linear codes

As it was mentioned before, the decoding process is a reconstruction of the original codeword. We start with two most generic decoders, *maximum a posteriori (MAP)* and *maximum-likelihood (ML)*. In fact, these decoders describe only the *objective* of decoding, while particular implementations depend on the channels under consideration.[2]

Consider a discrete memoryless channel with input in $\mathbb{F}$ and output in $\mathcal{Y}$, where $\mathcal{Y}$ is different from $\mathbb{F}$ in general case. The transmitter chooses a codeword $\boldsymbol{x}$ from a code $\mathcal{C}$ with probability $\mathbb{P}\left\{\boldsymbol{x}\right\}$ and sends it over the channel. Let $\boldsymbol{y}$ be an output of the channel and its conditional distribution $\mathbb{P}\left\{\boldsymbol{y} \mid \boldsymbol{x}\right\}$. The MAP decoder chooses an estimate $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}^{\mathrm{MAP}}(\boldsymbol{y})$ that maximises a posteriori probability

$$\mathbb{P}\left\{\hat{\boldsymbol{x}}^{\mathrm{MAP}}(\boldsymbol{y}) = \boldsymbol{x}\right\}.$$

The corresponding probability for the decoder to reconstruct the original codeword incorrectly is

$$\mathbb{P}\left\{\hat{\boldsymbol{x}}^{\mathrm{MAP}}(\boldsymbol{y}) \neq \boldsymbol{x}\right\} = 1 - \mathbb{P}\left\{\hat{\boldsymbol{x}}^{\mathrm{MAP}}(\boldsymbol{y}) = \boldsymbol{x}\right\}.$$

This kind of error is called *block* or *frame* error, as we check only whether the decoder has correctly reconstructed the whole codeword (i.e. block). We expand:

$$\begin{aligned}
\mathbb{P}\left\{\hat{\boldsymbol{x}}^{\mathrm{MAP}}(\boldsymbol{y}) = \boldsymbol{x}\right\} &= \sum_{\boldsymbol{b} \in \mathcal{Y}^n} \mathbb{P}\left\{\boldsymbol{x} = \hat{\boldsymbol{x}}^{\mathrm{MAP}}(\boldsymbol{b}), \boldsymbol{y} = \boldsymbol{b}\right\} \\
&= \sum_{\boldsymbol{b} \in \mathcal{Y}^n} \mathbb{P}\left\{\boldsymbol{y} = \boldsymbol{b}\right\} \mathbb{P}\left\{\boldsymbol{x} = \hat{\boldsymbol{x}}^{\mathrm{MAP}}(\boldsymbol{b}) \mid \boldsymbol{y} = \boldsymbol{b}\right\}.
\end{aligned}$$

Hence, we can do maximisation for each $\boldsymbol{b}$ separately. Moreover, each term $\mathbb{P}\left\{\boldsymbol{y} = \boldsymbol{b}\right\}$ is invariant of choice of function $\hat{\boldsymbol{x}}^{\mathrm{MAP}}(\cdot)$. Therefore, we simplify the optimisation problem to maximisation for a fixed $\boldsymbol{b} \in \mathcal{Y}^n$. In other words,

$$\begin{aligned}
\hat{\boldsymbol{x}}^{\mathrm{MAP}}(\boldsymbol{b}) &= \arg\max_{\boldsymbol{a} \in \mathcal{C}} \mathbb{P}\left\{\boldsymbol{x} = \boldsymbol{a} \mid \boldsymbol{y} = \boldsymbol{b}\right\} \\
&= \arg\max_{\boldsymbol{a} \in \mathcal{C}} \mathbb{P}\left\{\boldsymbol{y} = \boldsymbol{b} \mid \boldsymbol{x} = \boldsymbol{a}\right\} \frac{\mathbb{P}\left\{\boldsymbol{x} = \boldsymbol{a}\right\}}{\mathbb{P}\left\{\boldsymbol{y} = \boldsymbol{b}\right\}} \\
&= \arg\max_{\boldsymbol{a} \in \mathcal{C}} \mathbb{P}\left\{\boldsymbol{y} = \boldsymbol{b} \mid \boldsymbol{x} = \boldsymbol{a}\right\} \mathbb{P}\left\{\boldsymbol{x} = \boldsymbol{a}\right\}.
\end{aligned}$$

This is the MAP decoding rule.

Further, it is often the case that all codewords are equally likely on the channel input:

$$\mathbb{P}\left\{\boldsymbol{x} = \boldsymbol{a}\right\} = \frac{1}{|\mathcal{C}|}.$$

[2]However, one can also use a very straightforward decoding approach: check all the codewords of the code and choose one of them that fits the decoding objective best. Although in all the cases except trivial this approach is dramatically inefficient.

In this situation, we can simplify to the ML decoding rule:

$$\begin{aligned}\hat{\boldsymbol{x}}^{\mathrm{MAP}}(\boldsymbol{b}) &= \arg\max_{\boldsymbol{a}\in\mathcal{C}} \mathbb{P}\left\{\boldsymbol{y}=\boldsymbol{b} \mid \boldsymbol{x}=\boldsymbol{a}\right\} \mathbb{P}\left\{\boldsymbol{x}=\boldsymbol{a}\right\} \\ &= \arg\max_{\boldsymbol{a}\in\mathcal{C}} \mathbb{P}\left\{\boldsymbol{y}=\boldsymbol{b} \mid \boldsymbol{x}=\boldsymbol{a}\right\} = \hat{\boldsymbol{x}}^{\mathrm{ML}}(\boldsymbol{b}).\end{aligned}$$

That is, for uniform distribution of channel input vector $\boldsymbol{x}$, the MAP and ML decoders coincide.

For the BEC, ML decoding is equivalent to solving a system of linear equations. More precisely, assume that we have a code with a parity-check matrix H, and that the received word is $\boldsymbol{y}$. Let the positions of erasures be $\mathcal{E} \subseteq [n]$. Denote by $H_{\mathcal{E}}$ the matrix formed from the columns of H indexed by $\mathcal{E}$, and by $\boldsymbol{y}_{\mathcal{E}}$, the vector formed by the entries of $\boldsymbol{y}$ indexed by $\mathcal{E}$. Denote $\bar{\mathcal{E}} = [n] \setminus \mathcal{E}$ and, similarly, define $H_{\bar{\mathcal{E}}}$ and $\boldsymbol{y}_{\bar{\mathcal{E}}}$. Then the parity-check equations can be written as

$$H_{\mathcal{E}}\boldsymbol{y}_{\mathcal{E}}^{\mathsf{T}} + H_{\bar{\mathcal{E}}}\boldsymbol{y}_{\bar{\mathcal{E}}}^{\mathsf{T}} = \mathbf{0}^{\mathsf{T}},$$

where $\mathbf{0}$ is the all-zero vector of the corresponding length. Since $\boldsymbol{y}_{\bar{\mathcal{E}}}$, $H_{\bar{\mathcal{E}}}$, and $H_{\mathcal{E}}$ are known, we can rewrite the equations in the following form

$$H_{\mathcal{E}}\boldsymbol{y}_{\mathcal{E}}^{\mathsf{T}} = H_{\bar{\mathcal{E}}}\boldsymbol{y}_{\bar{\mathcal{E}}}^{\mathsf{T}}. \tag{1.3}$$

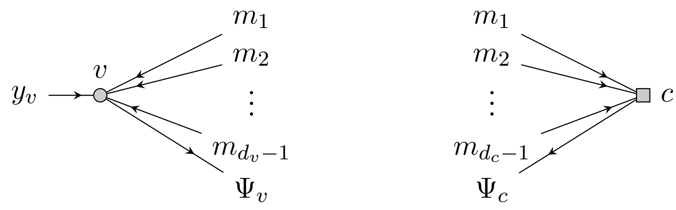
It is a system of linear equations with a vector of unknowns $\boldsymbol{y}_{\mathcal{E}}$ and a matrix of coefficients $H_{\mathcal{E}}$. This system always has at least one solution, the originally transmitted codeword. If this solution is not unique, we say that the ML decoder fails.

It is not difficult to see that the ML decoder fails if and only if $\mathcal{E}$ contains a support of some non-zero codeword $\boldsymbol{c}$. Indeed, the columns indexed by $\mathrm{supp}(\boldsymbol{c})$ sum up to the all-zero column. Therefore, the matrix $H_{\mathcal{E}}$ does not have full column rank, and (1.3) has multiple solutions.

1.2.5. Belief-propagation decoding

The next decoding method is central for this thesis. It is known by the names *iterative*, *message-passing (MP)*, or *belief-propagation (BP)*. However, an iterative algorithm is any algorithm that consists of iterations. Similarly, an MP algorithm is an iterative algorithm that passes messages (e.g. the IPA is a message-passing algorithm, cf. Section 1.3.1). Finally, BP is an MP algorithm with messages being *beliefs* about a value of an incident variable node. It is the narrowest name for this decoding algorithm and therefore we favour it.

The BP decoder can be defined for rather general channels. But to avoid unnecessary intricacy, we formulate the algorithm for a particular case of the BEC, as it is precisely what we need in the thesis. We refer an interested reader to a book [40], which discusses different aspects of BP in depth.



(a) In a variable node (b) In a check node

Figure 5. Message processing in BP decoding.

We next describe the BP decoder on the BEC in detail. Assume that a word $\boldsymbol{x} \in \mathbb{F}_2^n$ is sent and $\boldsymbol{y} \in \{0, 1, ?\}^n$ is received. We remind that due to nature of BEC, $\boldsymbol{x}$ and $\boldsymbol{y}$ agree in non-erased positions. The algorithm operates on the Tanner graph of a code in rounds by exchanging messages between variable and check nodes over the edges. Each message is from $\{0, 1, ?\}$ and it is a local belief about what the value of an incident variable node is. On the BEC, these beliefs are rather polarised; we either know for sure the value of a bit (0 or 1) or both 0 and 1 are equally likely.

In a variable-to-check message round, each variable node sends messages to each of the check nodes it neighbours. In a variable node v, the message sent over the edge e is a function of the bit y_v received from the channel and the incoming messages over all the edges *except* the edge e. If the degree of v is d_v and $m_1, m_2, \ldots, m_{d_v-1} \in \{0, 1, ?\}$ are the incoming messages (see Fig. 5a), the outgoing message is defined as follows:

$$\Psi_v(y_v, m_1, m_2, \ldots, m_{d_v-1}) = \begin{cases} b & \text{if any of } y_v, m_1, \ldots, m_{d_v-1} \text{ equals } b \in \mathbb{F}_2, \\ ? & \text{if } y_v = m_1 = \cdots = m_{d_v-1} = ? \end{cases}$$

That is, if any of the check nodes has recovered the value of x_v (or $y_v = x_v \neq ?$), this value is further propagated to other check nodes (but not directly back to itself).

At the very first iteration of the algorithm, each variable node v simply sends the bit it received from the channel, y_v.

In a check-to-variable round, similar processing happens. However, the nature of parity (sum of all incoming bits should be zero) is exploited. Namely, if the check node c of degree d_c receives messages $m_1, m_2, \ldots, m_{d_c-1} \in \{0, 1, ?\}$ (see Fig. 5b), the message sent over the remaining edge is defined as follows:

$$\Psi_c(m_1, m_2, \ldots, m_{d_c-1}) = \begin{cases} \sum_{i=1}^{d_c-1} m_i & \text{if every } m_i \in \mathbb{F}_2, \\ ? & \text{if any of } m_1, \ldots, m_{d_c-1} \text{ equals } ? \end{cases}$$

Indeed, if all the variable nodes incident to c except one have their values recovered, the value of the remaining incident variable node equals to the sum (over $\mathbb{F}_2$) of the others.

Contrary to the message rules, the current *global* estimate on the value of a variable node is based on the bit received from the channel and *all* the incoming messages. BP decoding stops when either all the bits of the codeword have been recovered, or the algorithm is 'stuck' and no new bits are being recovered.

At first sight, it might seem that using *all* d_v incoming messages might be beneficial (as we use more information already available). However, one can prove that this does not give any additional decoding power. On the other hand, the fact that a new outgoing message uses only *extrinsic* information is crucial for proving many fundamental facts about BP decoding over BEC. Again, we refer an interested reader to [40] for much broader and detailed picture.

A good example is worth a thousand words. Therefore, let us follow a particular instance of BP decoding step by step.

Example 7 ([40, Sec. 3.5])**.** Consider the $[7, 4, 3]$ Hamming code again. We use the Tanner graph from Fig. 3. Assume the word received from the channel is $\boldsymbol{y} = (0, ?, ?, 1, 0, ?, 0)$. Fig. 6 illustrates iterations of BP decoding. The vector $\hat{\boldsymbol{x}}$ indicates the current global estimate of the transmitted word $\boldsymbol{x}$. Note that $\hat{x}_i$ is based on y_i and *all* incoming messages to v_i and it is not used to calculate next messages.

For example, consider the check-to-variable message sent from c_1 to v_2 at iteration 1. It is the sum of the incoming messages 0, 1, and 0 modulo 2, received from v_1, v_4, and v_5, respectively.

After iteration 1, the value $x_2 = 1$ is recovered. This further allows to recover of $x_3 = 0$ after iteration 2. And that consequently leads to recovery of $x_6 = 1$ after iteration 3. Iteration 4 is not in fact needed, as all the bits have already been recovered. We only show it to illustrate what the further messages would be. $\triangle$

The following concept of *stopping sets* was first proposed by Richardson and Urbanke [39] in connection with efficient encoding of LDPC codes. Yet for BP *decoding* over the BEC, they play similar role as codewords for ML decoding in the sense that they are the core reason for a decoding algorithm to fail.

The definition of a stopping set can be given in either terms of a Tanner graph or a parity-check matrix.

Definition 8. A *stopping set* $\mathcal{S}$ in a Tanner graph is a subset of variable nodes such that all check nodes that are connected to $\mathcal{S}$, connected to $\mathcal{S}$ at least twice.

Definition 9. Let H be an $m \times n$ parity-check matrix of a binary linear code $\mathcal{C}$. A set $\mathcal{S} \subseteq [n]$ is called a *stopping set* if $H_{\mathcal{S}}$ contains no row of Hamming weight one.

The following is important for understanding the role of stopping sets for BP decoding over the BEC.

Proposition 10. *If the received word has erasures in positions indexed by a set $\mathcal{E} \subset [n]$ and $\mathcal{E}$ contains as a subset a non-empty stopping set $\mathcal{S}$, then the BP decoder fails.*

By convention, an empty set is also considered as a stopping set. It is important

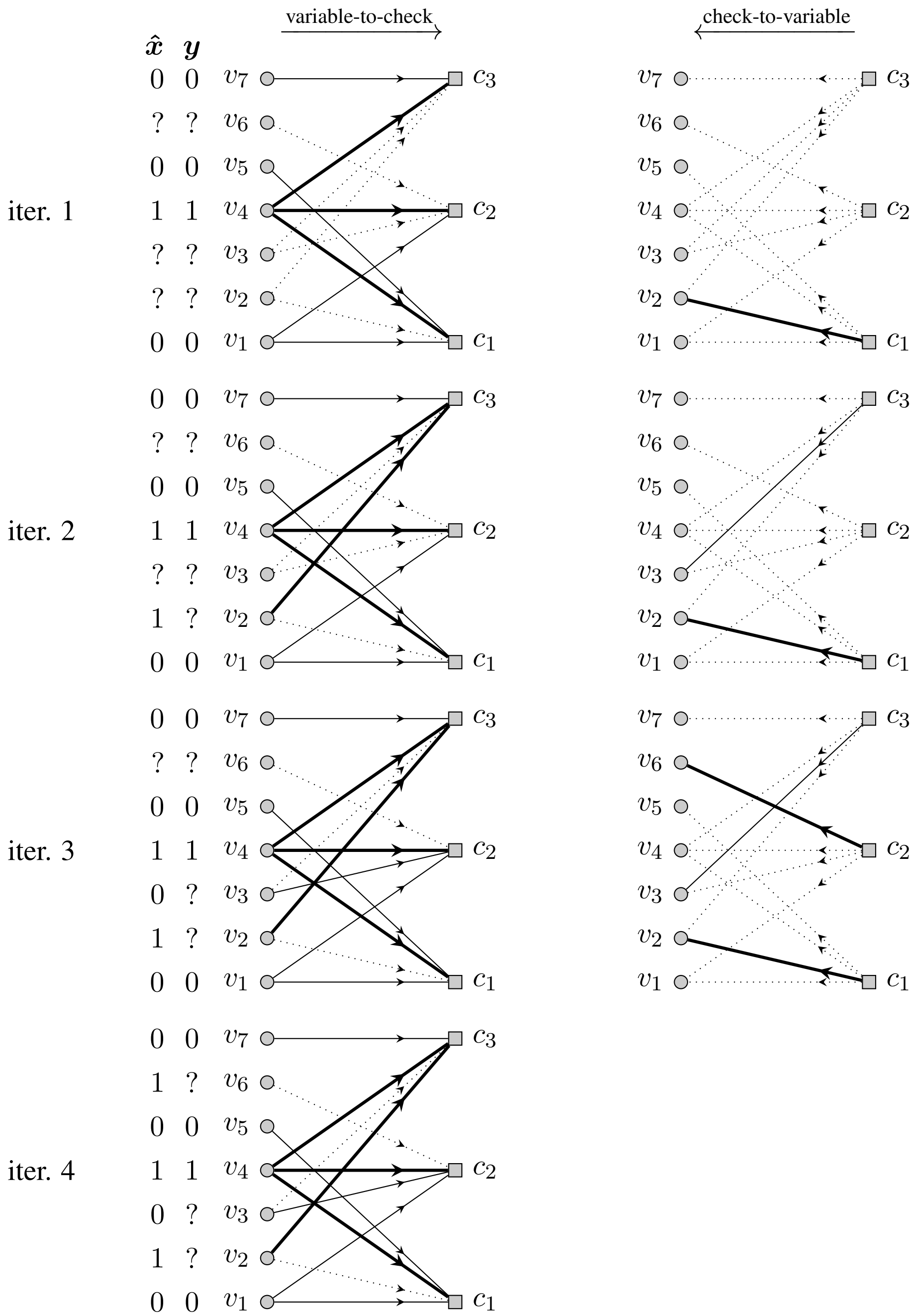


Figure 6. BP decoding of the $[7,4,3]$ Hamming code with the received word $\boldsymbol{y} = (0,?,?,1,0,?,0)$. A dotted arrow indicates a message ?, a thin arrow indicates a message 0, and a thick arrow indicates a message 1. We recover $x_2 = 1$ after the first iteration, $x_3 = 0$ after the second, and $x_6 = 1$ after the third. The recovered codeword is $\boldsymbol{x} = (0,1,0,1,0,1,0)$.

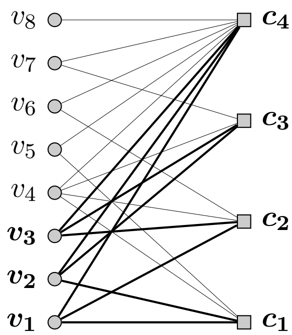


Figure 7. Example of a stopping set $T = \{v_1, v_2, v_3\}$ in the Tanner graph of the $[8, 4, 4]$ extended Hamming code. Each of the neighbouring check nodes c_1, c_2, c_3, c_4 is connected to T at least twice.

to stress that stopping sets are structures in a particular parity-check matrix (or, equivalently, in a particular Tanner graph) and not in the code. We note also that support of every codeword is a stopping set.

Example 11. Consider the parity-check matrix of the $[8, 4, 4]$ *extended* Hamming code:

$$H = \begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{0} & 1 & 1 & 0 & 0 & 0 \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & 1 & 0 & 1 & 0 & 0 \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & 1 & 0 & 0 & 1 & 0 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & 1 & 1 & 1 & 1 & 1 \end{pmatrix}. \tag{1.4}$$

The set of positions $T = \{1, 2, 3\}$ forms a stopping set (the columns are in bold in (1.4)). By exhaustive checking, one can see that this parity-check matrix has in total 125 stopping sets of size up to four, 16 of which are also supports of codewords. $\triangle$

1.2.6. Stopping redundancy

Following terminology of [46], we formulate the next definition.

Definition 12. A binary vector $\boldsymbol{h}$ *covers* a stopping set (or any subset of columns) $\mathcal{S}$ if $\mathrm{supp}(\boldsymbol{h})$ intersects with $\mathcal{S}$ in exactly one position. Consequently, a matrix covers $\mathcal{S}$ if any of its rows cover $\mathcal{S}$.

We note that if $\mathcal{S}$ is a stopping set in a parity-check matrix H and $\boldsymbol{h}$ covers $\mathcal{S}$, then, after adjoining $\boldsymbol{h}$ as a row to H, $\mathcal{S}$ is not a stopping set in the obtained extended matrix. With some abuse of notation, we say that a stopping set $\mathcal{S}$ is *covered* in that extended matrix.[3]

Definition 13. A stopping set $\mathcal{S}$ is *coverable* (by a code $\mathcal{C}$), if there exists a (possibly extended) parity-check matrix of $\mathcal{C}$ that covers $\mathcal{S}$.

[3]That is to say, we will use "a stopping set $\mathcal{S}$ is covered by a matrix" and "$\mathcal{S}$ is not a stopping set in a matrix" interchangeably.

The definition is equivalent to the following statement. If we denote by $H^{(2^r)}$ the parity-check matrix of $\mathcal{C}$ consisting of *all* the dual codewords, then a stopping set $\mathcal{S}$ is coverable by $\mathcal{C}$ if and only if $\mathcal{S}$ is covered by $H^{(2^r)}$.

In order to reduce the failure probability of BP decoding algorithm over the BEC, it was proposed in [46] to add redundant rows, which are exactly the codewords of $\mathcal{C}^\perp$, to a parity-check matrix in such a way that the resulting matrix has no stopping sets of small size. Specifically, we are interested in constructing a parity-check matrix consisting of the minimum number of rows from $\mathcal{C}^\perp$ so that all the stopping sets of size less than d are covered. It was shown in [46] that it is always possible, i.e. all stopping sets of size less than d are coverable.

In this work, we build on the approach in [46], namely we extend a parity-check matrix by choosing codewords from $\mathcal{C}^\perp$ and adjoining them as redundant rows. An extended matrix is constructed so that it does not contain stopping sets of small size. In the sequel, we provide a detailed analysis of the minimum number of additional rows in order to achieve this goal. In what follows, we use the terms "row of a parity-check matrix" and "codeword from $\mathcal{C}^\perp$" interchangeably. We also note that a particular order of rows in a parity-check matrix is not important.

Definition 14 ([46])**.** The size of the smallest stopping set of a parity-check matrix H, denoted by $s(H)$ (or $s_{\min}(H)$), is called the *stopping distance* of the matrix.

It is known that a maximal parity-check matrix $H^{(2^r)}$ consisting of all 2^r codewords of $\mathcal{C}^\perp$ is an *orthogonal array* of strength $d-1$ (cf. [35, Ch. 5, Thm. 8]). This means that for any $\mathcal{S} \subseteq [n]$ of size i, $1 \leq i \leq d-1$, $H_{\mathcal{S}}^{(2^r)}$ contains each i-tuple as its row exactly 2^{r-i} times and, hence, $\mathcal{S}$ is covered by exactly $i \cdot 2^{r-i}$ rows of $H^{(2^r)}$.

Example 15. Consider the parity-check matrix of the $[8,4,4]$ extended Hamming code from (1.4). Fig. 8 shows all codewords of its dual code. In particular, there are six dual codewords (i.e. redundant rows) that cover the stopping set $\{1,2,3\}$. $\triangle$

The following definition was introduced in [46].

Definition 16. The *stopping redundancy* of $\mathcal{C}$, denoted by $\rho(\mathcal{C})$, is the smallest number of rows in any (rank-r) parity-check matrix of $\mathcal{C}$, such that the corresponding stopping distance is d.

It was shown in [46, Thm. 3], that *any* parity-check matrix H of a binary linear code $\mathcal{C}$ with the minimum distance $d \leq 3$ already has $s(H) = d$. In what follows, we are mostly interested in the case $d > 3$.

1.3. Compressed sensing

The reconstruction of a (mathematical) object from a partial set of observations in an efficient and reliable manner is of fundamental importance. Compressed sensing, motivated by the ground-breaking work of Candès and Tao [6, 7], and

$$\begin{array}{cccccccc}
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

Figure 8. Codewords of the code dual to the $[8, 4, 4]$ extended Hamming code. The solid rectangle denote the original parity-check matrix in (1.4). The dotted rectangle is an orthogonal array. Each of six dashed codewords cover the stopping set $\{1, 2, 3\}$.

independently by Donoho [9], is a research area in which the object to be reconstructed is a k-sparse signal vector (there are at most k non-zero entries in the vector) over the real numbers. The partial information provided is a linear transformation of the signal vector, the *measurement vector*, and the objective is to reconstruct the object from a small number of measurements.

Compressed sensing provides a mathematical framework which shows that, under some conditions, signals can be recovered from far fewer measurements than with conventional signal acquisition methods. The main idea in compressed sensing is to exploit the property that most of the interesting signals have an inherent structure or contain redundancy. The compressed sensing problem is described in more details below.

Let $\boldsymbol{x} \in \mathbb{R}^n$ be an n-dimensional k-sparse signal (i.e. it has at most k non-zero entries), and let $A = (a_{ji})$ be an $m \times n$ real measurement matrix. We consider the recovery of $\boldsymbol{x}$ from measurements $\boldsymbol{y}^\mathsf{T} = A\boldsymbol{x}^\mathsf{T} \in \mathbb{R}^m$, where $m < n$ and $k < n$.

The reconstruction problem of compressed sensing is to find the sparsest $\boldsymbol{x}$ (i.e. the one that minimizes the ℓ_0-norm) under the constraint $\boldsymbol{y}^\mathsf{T} = A\boldsymbol{x}^\mathsf{T}$, which in general is an NP-hard problem. Basis pursuit is an algorithm which reconstructs $\boldsymbol{x}$ by minimizing its ℓ_1-norm under the constraint $\boldsymbol{y}^\mathsf{T} = A\boldsymbol{x}^\mathsf{T}$ (see [6]). This is a linear program, and thus it can be solved in polynomial time. The algorithm has a remarkable performance, but its complexity is high, making it impractical for many applications that require fast reconstruction. A fast reconstruction algorithm for non-negative real signals and measurement matrices is the IPA which is described below in Section 1.3.1.

1.3.1. Interval-passing algorithm

Iterative reconstruction algorithms for compressed sensing have received considerable interest recently. See, for instance, [57, 38, 8, 45, 37, 10, 11] and references therein. The IPA for reconstruction of non-negative sparse signals was introduced by Chandar *et al.* in [8] for binary measurement matrices. The algorithm was further generalized to non-negative real measurement matrices in [38].

An improvement to the IPA using the principle of *verification* was proposed recently in [51]. The proposed algorithm performs better than the plain IPA and also better than the plain verification algorithm, first introduced in [45], for measurement matrices equal to parity-check matrices of LDPC codes.

Note that there is a clear connection between the IPA and the iterative message-passing algorithm proposed for counter braids in [30] (see also [42]) in the sense that the algorithm for counter braids is a special case of the IPA (see Section 1.3.1 below). Thus, the results derived in this work apply immediately also to iterative decoding of counter braids as described in [30].

Recall that we want to reconstruct the signal vector $\boldsymbol{x} \in \mathbb{R}^n_{\geq 0}$ from the linear requirement

$$\boldsymbol{y}^\mathsf{T} = A\boldsymbol{x}^\mathsf{T},$$

where both the measurement matrix $A = (a_{ji}) \in \mathbb{R}^{m\times n}_{\geq 0}$ and the measurement vector $\boldsymbol{y} \in \mathbb{R}^m_{\geq 0}$ are known. Together with A, we consider its Tanner graph (cf. Section 1.1). Let V be the set of *variable* nodes corresponding to columns of A, and C the set of *measurement* nodes[4] corresponding to rows of A. As previously, $\mathcal{N}(\cdot)$ denotes the set of neighbours.

The IPA is based on the following idea. Consider one measurement $c \in C$:

$$\sum_{v\in\mathcal{N}(c)} a_{cv}x_v = y_c$$

and express the value for one of the variable nodes, $v \in V$:

$$x_v = \frac{1}{a_{cv}}\left(y_c - \sum_{\substack{v'\in\mathcal{N}(c)\\ v'\neq v}} a_{cv'}x_{v'}\right). \tag{1.5}$$

Assume we have upper bounds $x_{v'} \leq M_{v'}$ for all $v' \in \mathcal{N}(c)\setminus\{v\}$. Then from (1.5) and non-negativity of A, we obtain a lower bound on x_v:

$$x_v \geq \frac{1}{a_{cv}}\left(y_c - \sum_{\substack{v'\in\mathcal{N}(c)\\ v'\neq v}} a_{cv'}M_{v'}\right). \tag{1.6}$$

[4] Note the difference in terminology of Tanner graph from that in the context of linear codes.

In much the same fashion, if we have lower bounds $x_{v'} \geq \mu_{v'}$, we can express an upper bound on x_v:

$$x_v \leq \frac{1}{a_{cv}} \left(y_c - \sum_{\substack{v' \in \mathcal{N}(c) \\ v' \neq v}} a_{cv'} \mu_{v'} \right). \tag{1.7}$$

For each pair $(c, v) \in C \times V$ of connected check and variable nodes (i.e. $a_{cv} > 0$), we obtain a pair of new bounds (1.6) and (1.7) that are based on the previously known bounds.

Briefly, the IPA establishes some initial trivial bounds on the values of x_v, $v \in V$, and further tries to improve these bounds in an iterative manner using (1.6) and (1.7). The hope here is that at some iteration upper and lower bounds coincide thus recovering the true value of (unknown) x_v.

To be more specific, the IPA iteratively sends messages between variable and measurement nodes. Each message contains two real numbers, a *lower bound* and an *upper bound* on the value of the variable node to which it is affiliated. Let $\mu^{(\ell)}_{v \to c}$ (resp. $\mu^{(\ell)}_{c \to v}$) denote the lower bound of the message from variable node v (resp. measurement node c) to measurement node c (resp. variable node v) at iteration ℓ. The corresponding upper bound of the message is denoted by $M^{(\ell)}_{v \to c}$ (resp. $M^{(\ell)}_{c \to v}$). It is a distinct property of the algorithm that at any iteration ℓ, $\mu^{(\ell)}_{v \to c} \leq x_v \leq M^{(\ell)}_{v \to c}$ and $\mu^{(\ell)}_{c \to v} \leq x_v \leq M^{(\ell)}_{c \to v}$, for all $v \in V$ and $c \in \mathcal{N}(v)$. We omit rather straightforward proof of this fact (e.g. by induction).

The detailed steps of the IPA are shown in Algorithm 1 below. The lower bounds are initialised with zeroes, and these values are implied in corresponding initial upper bounds. Iterations continue while there is some progress, i.e. at least one of the bounds for some variable node changes between iterations. Since we expect that the signal $\boldsymbol{x}$ is sparse, the output of the IPA is the lower bounds on the corresponding values of variable nodes, thus gravitating to output zeroes.

From Lines 4, 14, and 15 in Algorithm 1, one can see that both $\mu^{(\ell)}_{v \to c}$ and $M^{(\ell)}_{v \to c}$ are independent of $c \in \mathcal{N}(v)$. Thus, we will often denote $\mu^{(\ell)}_{v \to c}$ by $\mu^{(\ell)}_{v \to \cdot}$ and $M^{(\ell)}_{v \to c}$ by $M^{(\ell)}_{v \to \cdot}$.

Note that in the special case when setting $M^{(0)}_{v \to \cdot} = \infty$ for all $v \in V$, the algorithm reduces to the iterative decoding algorithm outlined in [30] for counter braids. In fact, due to this initialization, only upper bounds need to be computed for odd iterations and only lower bounds for even iterations (for both variables nodes and measurement/counter nodes).

Example 17. Suppose we have the following measurement matrix:

$$A = \begin{pmatrix} 1 & 2 & 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 1 & 3 & 0 \\ 0 & 1 & 0 & 1 & 0 & 3 \\ 0 & 0 & 4 & 0 & 3 & 2 \end{pmatrix}$$

and the signal vector is $\boldsymbol{x} = (1, 8, 3, 0, 0, 0)$. Measurement vector is then $\boldsymbol{y} = \boldsymbol{x}A^\intercal = (20, 3, 8, 12)$. Fig. 9 illustrates iterations of the IPA. $\triangle$

Algorithm 1: Interval-passing algorithm (cf. [38, Alg. 1])

Function $\mathrm{IPA}(\boldsymbol{y}, A)$**:**
  Input: vector of measurements $\boldsymbol{y}$, measurement matrix A
  Output: estimate of the original signal, $\hat{\boldsymbol{x}}$
  forall $v \in V$ **do** `/* initialisation */`
    $\mu_{v\to\cdot}^{(0)} \leftarrow 0$
    $M_{v\to\cdot}^{(0)} \leftarrow \min_{c\in\mathcal{N}(v)} (y_c/a_{cv})$
  $\ell \leftarrow 0$
  repeat `/* iterations */`
    $\ell \leftarrow \ell + 1$
    forall $c \in C$, $v \in \mathcal{N}(c)$ **do**

$$\mu_{c\to v}^{(\ell)} \leftarrow \frac{1}{a_{cv}} \left(y_c - \sum_{\substack{v'\in\mathcal{N}(c)\\ v'\neq v}} a_{cv'} M_{v'\to\cdot}^{(\ell-1)} \right)$$

      if $\mu_{c\to v}^{(\ell)} < 0$ **then** `/* ensure non-negativity */`
        $\mu_{c\to v}^{(\ell)} \leftarrow 0$

$$M_{c\to v}^{(\ell)} \leftarrow \frac{1}{a_{cv}} \left(y_c - \sum_{\substack{v'\in\mathcal{N}(c)\\ v'\neq v}} a_{cv'} \mu_{v'\to\cdot}^{(\ell-1)} \right)$$

    forall $v \in V$ **do**

$$\mu_{v\to\cdot}^{(\ell)} \leftarrow \max_{c\in\mathcal{N}(v)} \mu_{c\to v}^{(\ell)}$$

$$M_{v\to\cdot}^{(\ell)} \leftarrow \min_{c\in\mathcal{N}(v)} M_{c\to v}^{(\ell)}$$

  until $\mu_{v\to\cdot}^{(\ell)} = \mu_{v\to\cdot}^{(\ell-1)}$ **and** $M_{v\to\cdot}^{(\ell)} = M_{v\to\cdot}^{(\ell-1)}$ **forall** $v \in V$
  $\hat{\boldsymbol{x}} = (\hat{x}_v)_{v\in V} \leftarrow (\mu_{v\to\cdot}^{(\ell)})_{v\in V}$ `/* result */`
  return $\hat{\boldsymbol{x}}$

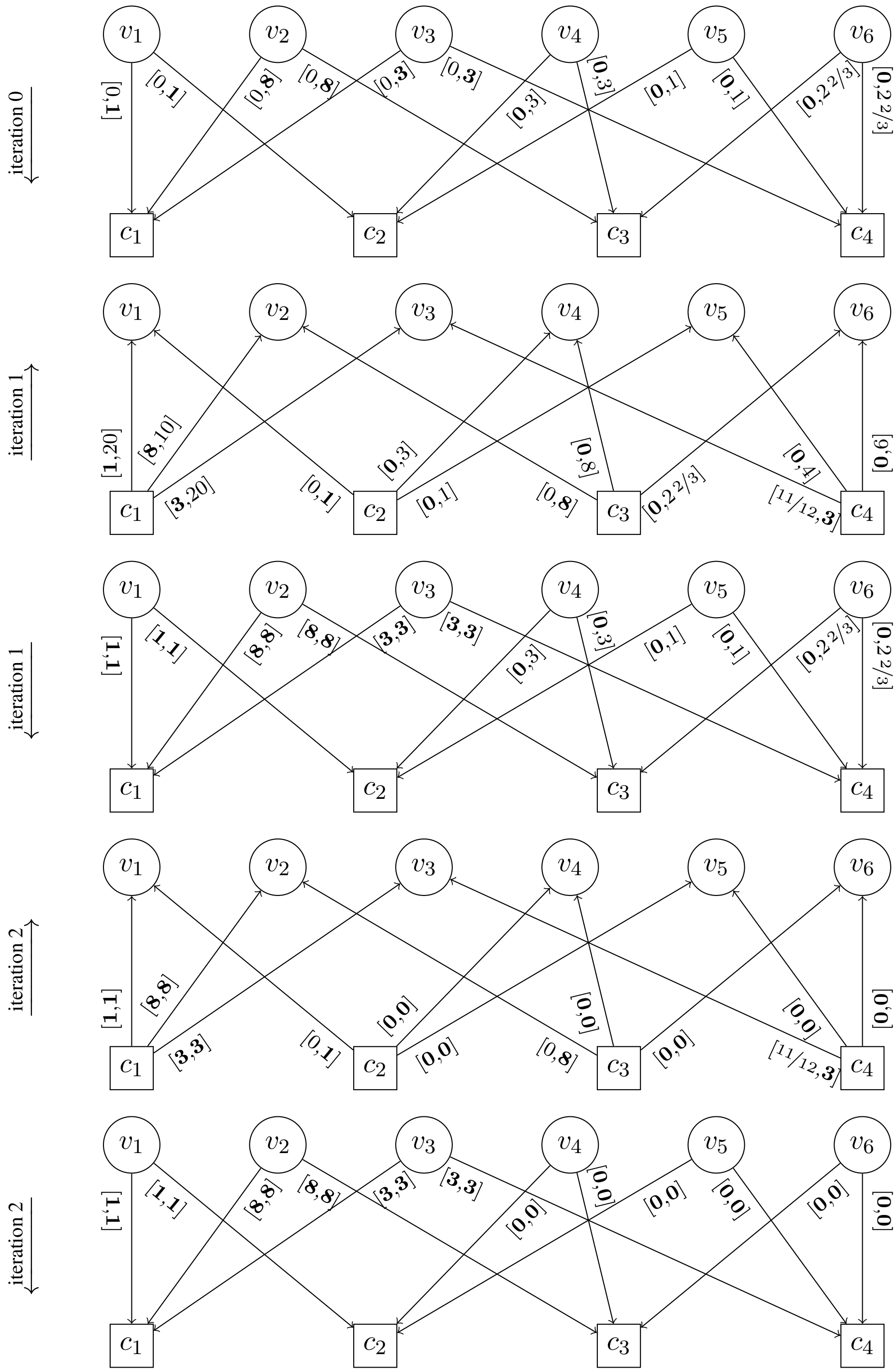


Figure 9. IPA reconstruction example. The original signal vector is $\boldsymbol{x} = (1, 8, 3, 0, 0, 0)$ and the measurement vector is $\boldsymbol{y} = (20, 3, 8, 12)$. Numbers in bold correspond to exact bounds. The last iteration is omitted because the signal has already been reconstructed.

2. STOPPING REDUNDANCY HIERARCHY BEYOND THE MINIMUM DISTANCE

Even things that are true can be proved.
—Oscar Wilde, *The Picture of Dorian Gray*

The main focus of this chapter is obtaining upper bounds on the minimum amount of rows in a (redundant) parity-check matrix of a fixed code, under some condition on presence of stopping sets in a parity-check matrix.

First, we present existing bounds on the stopping redundancy, as well as our modifications to these bounds. The latter give the tightest bounds for general codes, to the best of our knowledge. Next, we examine stopping redundancy hierarchy, which is a generalisation of the stopping redundancy concept. We also briefly analyse the choice of initial matrix which is important for our methods. Further, we suggest how to approach the ML decoding performance by using redundant parity-check matrices. As the presented bounds strongly depend on the knowledge of stopping sets spectra—which is often an intractable problem—we suggest different approaches to tackle this. The chapter concludes with extensive numerical results, for both particular codes and ensemble-average values.

The contents of this chapter are based on [54] and its further extension in [55]. Results in Section 2.3.3 are from [3].

2.1. Upper bounds on stopping redundancy

In this section, we present several upper bounds on the stopping redundancy of a linear code. Specifically, Section 2.1.1 provides an overview of existing results and the proof of a bound modified from the best one known. In Section 2.1.2, we generalise the concept of stopping redundancy by introducing stopping redundancy hierarchy. Section 2.1.3 is devoted to the discussion of ways to choose the initial rows in a parity-check matrix, which is important for the presented bound.

2.1.1. Upper bounds for general codes

In [46], Schwartz and Vardy presented an upper bound on the stopping redundancy of a general binary linear $[n, k, d]$ code $\mathcal{C}$:

$$\rho(\mathcal{C}) \leq \binom{r}{1} + \binom{r}{2} + \cdots + \binom{r}{d-2}. \tag{2.1}$$

This bound is constructive. More precisely, the authors adjoin all linear combinations of up to $d-2$ rows from the original parity-check matrix and prove that the resulting matrix has the stopping distance d.

The other related works are [50, 23, 22, 24, 25, 13, 19, 21], which present other constructive upper bounds—for general linear codes, for some specific families or for particular codes.

On the other hand, *probabilistic* arguments gave a rise to better bounds [22, 25, 19, 20, 54], yet these bounds are non-constructive. The main probabilistic technique in this thesis dates back to the work of Han and Siegel [19]. They established the following bound:

$$\rho(\mathcal{C}) \leq \min\{t \in \mathbb{N} : \mathcal{E}_{n,d}(t) < 1\} + (r - d + 1), \tag{2.2}$$

where

$$\mathcal{E}_{n,d}(t) \triangleq \sum_{i=1}^{d-1} \binom{n}{i} \left(1 - \frac{i}{2^i}\right)^t .$$

Briefly, $\mathcal{E}_{n,d}(t)$ is the average number of stopping sets of size at most $d-1$ in a parity-check matrix formed from t dual codewords chosen randomly with repetition from $\mathcal{C}^\perp$. Therefore, if for some t we have $\mathcal{E}_{n,d}(t) < 1$, there is a realisation (i.e. choice of t dual codewords) when the obtained parity-check matrix has no stopping sets of size less than d. The term $(r - d + 1)$ is added to guarantee the correct rank of the obtained parity-check matrix.

The bound (2.2) has been improved by Han, Siegel, and Vardy in [20] by calculating probabilities in a more precise fashion and introducing one more stage of the probabilistic construction algorithm. At that stage, new rows are chosen one by one. We further refined this bound in [54] by carefully selecting the first non-random rows. This gives the smallest known values for most codes (to the best of our knowledge). A slightly modified version of the bound is presented in Theorem 19, which is the main result of this chapter.

Before proceeding, we prove the following technical result, which will be used further.

Lemma 18. *For any integers $i, j, r \geq 1$, and $j < 2^r$, define*

$$\pi(r,i,j) \triangleq 1 - \frac{i \cdot 2^{r-i}}{2^r - j}\,.$$

Then, for any integer $r \leq r'$, and $i \leq i'$, we have $\pi(r,i,j) \leq \pi(r',i,j)$ and $\pi(r,i,j) \leq \pi(r,i',j)$. In other words, $\pi(r,i,j)$ is monotonically non-decreasing in integer variables r and i.

Proof. The statement of the lemma follows easily if we rewrite:

$$\pi(r,i,j) = 1 - \frac{i}{2^i} \cdot \frac{1}{1 - j \cdot 2^{-r}}. \qquad \square$$

Below we present a bound modified from [54, Thm. 1]. More precisely, we drop the burdensome requirement

$$(r-1)(d-1) \leq 2^{d-1} \tag{2.3}$$

thus making the bound applicable to all the binary linear codes. On the other hand, we need to add the rank deficiency term Δ to ensure that the constructed parity-check matrix has the required rank. However, for medium and long codes, this term is negligible in comparison with the stopping redundancy.

Theorem 19. *For an $[n,k,d]$ linear binary code $\mathcal{C}$ let $H^{(\tau)}$ be any $\tau \times n$ matrix consisting of τ different codewords of the dual code $\mathcal{C}^{\perp}$ and let u_i denote the number of stopping sets of size i, $i = 1, 2, \ldots, d-1$, in $H^{(\tau)}$. For $t = 0, 1, \cdots, 2^r - \tau$, we introduce the following notations:*

$$\mathcal{D}_t = \sum_{i=1}^{d-1} u_i \prod_{j=\tau+1}^{\tau+t} \pi(r,i,j)\,,$$

$$P_{t,0} = \lfloor \mathcal{D}_t \rfloor\,,$$

$$P_{t,j} = \Big\lfloor \pi(r, d-1, \tau+t+j) P_{t,j-1} \Big\rfloor\,, \quad j = 1, 2, \ldots$$

$$\Delta = r - \max\{\operatorname{rank} H^{(\tau)}, d-1\}\,,$$

and let κ_t be the smallest integer such that $P_{t,\kappa_t} = 0$. Then

$$\rho \leq \tau + \min_{0 \leq t < 2^r - \tau} \{t + \kappa_t\} + \Delta\,. \tag{2.4}$$

Proof. We prove the theorem in two steps. First, we show the existence of a $(\tau + t) \times n$ matrix with a number of stopping sets less or equal to $P_{t,0}$. Second, we show that this number further decreases when we add carefully selected rows one by one. Finally, after adding a sufficient number of rows, we obtain a matrix with no stopping sets of size less than d.

Step 1. By orthogonal array property, for any subset of columns $\mathcal{S} \subseteq [n]$ of size i, $i = 1, 2, \ldots, d-1$, there are exactly $i \cdot 2^{r-i}$ codewords in $\mathcal{C}_0^{\perp}$, that cover $\mathcal{S}$. If $\mathcal{S}$ is not covered by $H^{(\tau)}$, none of these $i \cdot 2^{r-i}$ codewords is present among the rows of $H^{(\tau)}$.

Fix a stopping set $\mathcal{S}$ in $H^{(\tau)}$. Next, draw t codewords from the set $\mathcal{C}_0^{\perp} \setminus \{\text{rows of } H^{(\tau)}\}$ at random without repetition. There are

$$\binom{2^r - \tau - 1}{t}$$

ways to do this, provided the order of selection does not matter. On the other hand, in the same set $\mathcal{C}_0^{\perp} \setminus \{\text{rows of } H^{(\tau)}\}$, there are $(2^r - \tau - 1) - i \cdot 2^{r-i}$ codewords that do *not* cover $\mathcal{S}$ and there are

$$\binom{(2^r - \tau - 1) - i \cdot 2^{r-i}}{t}$$

ways to draw t codewords out of them. Therefore, if we draw t codewords from the set $\mathcal{C}_0^{\perp} \setminus \{\text{rows of } H^{(\tau)}\}$ at random without repetition, the probability not to cover $\mathcal{S}$ by any one of them is

$$\binom{(2^r - \tau - 1) - i \cdot 2^{r-i}}{t} \Big/ \binom{2^r - \tau - 1}{t} = \prod_{j=\tau+1}^{\tau+t} \pi(r, i, j)\,.$$

This holds for each $\mathcal{S}$ that was not originally covered by $H^{(\tau)}$. Since the numbers of the stopping sets of sizes $1, 2, \ldots, d-1$ are $u_1, u_2, \ldots, u_{d-1}$, respectively, the average[1] number of the stopping sets of size less than d that are left after adjoining t random rows to $H^{(\tau)}$ is

$$\sum_{i=1}^{d-1} u_i \prod_{j=\tau+1}^{\tau+t} \pi(r, i, j) \triangleq \mathcal{D}_t\,.$$

Furthermore, since the above expression is an expected value of an integer random variable, there exists its realisation (i.e. choice of t rows), such that the number of stopping sets left is not more than $\lfloor \mathcal{D}_t \rfloor \triangleq P_{t,0}$. Fix these t rows and further assume that we have a $(\tau + t) \times n$ matrix $H^{(\tau+t)}$ with not more than $P_{t,0}$ stopping sets of size less than d.

[1] Averaging is by the choice of t rows.

Step 2. Adjoin to $H^{(\tau+t)}$ a random codeword from $\mathcal{C}_0^{\perp} \setminus \{\text{rows of } H^{(\tau+t)}\}$. If some stopping set $\mathcal{S}$ of size i, $1 \le i \le d-1$, has not been covered by $H^{(\tau+t)}$ yet, there are exactly $i \cdot 2^{r-i}$ codewords in $\mathcal{C}_0^{\perp} \setminus \{\text{rows of } H^{(\tau+t)}\}$ that cover $\mathcal{S}$ and, thus, the probability that $\mathcal{S}$ stays non-covered after adjoining this new row is

$$1 - \frac{i \cdot 2^{r-i}}{2^r - (\tau + t + j)} = \pi(r, i, \tau + t + 1) \overset{\text{Lemma 18}}{\le} \pi(r, d-1, \tau + t + 1).$$

This holds for any stopping set $\mathcal{S}$ of size i. Then, there exists a codeword in $\mathcal{C}_0^{\perp} \setminus \{\text{rows of } H^{(\tau+t)}\}$ such that after adjoining it as a row to $H^{(\tau+t)}$, the number of non-covered stopping sets becomes less or equal to

$$\left\lfloor \pi(r, d-1, \tau + t + 1) P_{t,0} \right\rfloor \triangleq P_{t,1}.$$

To this end, we fix this new row and further assume that we have a $(\tau+t+1) \times n$ matrix $H^{(\tau+t+1)}$ with the number of the stopping sets of size smaller than d less or equal to $P_{t,1}$. After that, we iteratively repeat Step 2. We stop when the number of non-covered stopping sets is equal to zero.

Finally, we need to ensure that the rank of the resulting matrix is indeed r. We already know that it is not less than $\operatorname{rank} H^{(\tau)}$. On the other hand, since we covered all the stopping sets of size less than d, the rank is at least $d-1$. Hence it is enough to add Δ additional rows to ensure the correct rank of the parity-check matrix. □

Note. The expression in (2.4) is monotonically non-decreasing in u_i. Often, the exact values of u_i are difficult to find and in that case upper bounds are used instead.

Note. By applying Lemma 18 to the expressions for $\mathcal{D}_t$ and $P_{t,j}$, we obtain that (2.4) is also monotonically non-decreasing in r. Sometimes, a parity-check matrix is redundant[2] and the number of rows m is larger than r. It might be more convenient to use m instead of r and the bound (2.4) still holds.

To give a flavour of differences between the existing bounds on stopping redundancy, we calculate the bounds (2.1) in [46], (2.2) in [19], the bound in [20, Thm. 7], the bound in [54, Thm. 1], and the bound in Theorem 19. The two last bounds are calculated in two modes. First, we use $\tau = 1$ and $H^{(\tau)}$ consists of the first row of the parity-check matrix of the corresponding code. Next, we use whole parity-check matrices of the codes as $H^{(\tau)}$ (in Table 1, m denotes the number of rows in a parity-check matrix used).

We calculate the aforementioned bounds for the following codes:

- the $[24, 12, 8]$ extended Golay self-dual code (cf. Section 2.3.1);
- the $[48, 24, 12]$ extended quadratic residue (QR) self-dual code (cf. [35, Sec. 16]);

[2]For instance, recall Gallager (J, K)-regular codes (cf. Section 1.2.3).

Table 1. Comparison of upper bounds on the stopping redundancy of different codes.

	$[24, 12, 8]$ Golay	$[48, 24, 12]$ QR	$[155, 64, 20]$ Tanner
(2.1)	2509	4 540 385	$6.2 \cdot 10^{18}$
(2.2)	232	4440	1 526 972
[20, Thm. 7]	182	3564	1 260 673
[54, Thm. 1], $\tau = 1$	180	3538	1 247 888
Theorem 19, $\tau = 1$	185	3562	1 247 960
[54, Thm. 1], $\tau = m$	168	2543	2573
Theorem 19, $\tau = m$	168	2543	2573

- the $(3, 5)$-regular $[155, 64, 20]$ Tanner code in [49].

Table 1 presents numerical results. The original bound by Schwartz and Vardy (2.1) is the only constructive bound here, but it is by far the worst. Note that the bound in Theorem 19 is only slightly worse than [54, Thm. 1] but it is applicable to any code. Often, a code that do not satisfy (2.3) has its stopping distance equal to the minimum distance. Yet the new bound is useful for calculation of the stopping redundancy hierarchy (see Section 2.1.2).

The bounds in [54, Thm. 1] and Theorem 19 with $\tau = m$ give the tightest results. However, they require knowledge of the stopping set spectrum of a parity-check matrix. For the Golay and the QR codes, we calculate their spectra by exhaustive brute-force checking. For the Tanner code, we use the spectrum obtained in [43, Tab. 1]. For longer codes, calculating a stopping sets spectrum can be infeasible even for the method in [43] and similar works. We suggest a way to overcome this obstacle in Section 2.2.3.

2.1.2. Stopping redundancy hierarchy

In Definition 16, it is required that the stopping distance of a parity-check matrix is exactly d. However, a more general requirement can be imposed. Thus, in [21], it was required that the parity-check matrix does not contain stopping sets of size up to ℓ, for some $\ell < d$. This can be achieved by adjoining a smaller number of rows to a parity-check matrix.

The following definition is according to [21, Def. 2.4].

Definition 20. The ℓ-th *stopping redundancy* of $\mathcal{C}$, $1 \leq \ell \leq d-1$, is the smallest non-negative integer $\rho_\ell(\mathcal{C})$ such that there exists a (possibly redundant) parity-check matrix of $\mathcal{C}$ with $\rho_\ell(\mathcal{C})$ rows and the stopping distance $\ell + 1$ (equivalently, with no stopping sets of size less than or equal to ℓ). The ordered set of integers $(\rho_1(\mathcal{C}), \rho_2(\mathcal{C}), \ldots, \rho_{d-1}(\mathcal{C}))$ is called the *stopping redundancy hierarchy* of $\mathcal{C}$.

From Definition 20, we have that $\rho(\mathcal{C}) = \rho_{d-1}(\mathcal{C})$.

Note. For $\ell \leq d-1$, an upper bound on the ℓ-th stopping redundancy can be formulated as in Theorem 19, where d is replaced by $\ell + 1$. We omit the details.

It is important to notice that stopping sets of size d or larger can also cause failures of the BP decoder on the BEC (see, for example, [50]). Thus, in order to approach the ML performance with the BP decoder, we should also cover stopping sets of size d or larger, at least those that, if erased, can be still decoded by the ML decoder. In fact, we show in Section 2.2.1 that it is always possible to achieve ML decoding performance by adjoining sufficiently large number of redundant rows. We will generalise Definition 20 accordingly (see Definition 24).

2.1.3. Choice of initial matrix

Theorem 19 does not suggest how one should choose the initial $\tau \times n$ matrix. In general, it is a difficult question, as it strongly depends on the particular code. Below, we propose some simple heuristics.

Fix $\tau = 1$. Then, Lemma 46 in Appendix A gives two values for a weight w of the row of $H^{(\tau)}$, one of which is guaranteed to cover the maximum number of stopping sets of size not more than ℓ:

$$w_{\text{opt}} \in \left\{ \left\lfloor \frac{n+1}{\ell} \right\rfloor, \left\lceil \frac{n}{\ell} \right\rceil \right\}.$$

However, a codeword of such weight does not necessarily exist in $\mathcal{C}^{\perp}$. Hence one needs to consider the closest alternatives. After a dual codeword of weight w is fixed, the number of stopping sets of size less than d in $H^{(\tau)}$ is expressed as

$$u_i = \binom{n}{i} - w\binom{n-w}{i-1},$$

and these values can be further used with the bound in Theorem 19.

The situation becomes more complicated for $\tau = 2$, as in that case the optimal choice depends not only on the weights of the first two rows of $H^{(\tau)}$, but also on the size of the intersection of their supports. For simplicity, we can take two different rows of the same weight and obtain the corresponding estimate on the number of stopping sets. More precisely, if $\tau = 2$, $H^{(\tau)}$ consists of two dual codewords $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ of weight w each with an intersection of supports of size $|\operatorname{supp}(\boldsymbol{h}_1) \cap \operatorname{supp}(\boldsymbol{h}_2)| = \delta$, then the total number of stopping sets of size less than d in $H^{(\tau)}$ equals (cf. [54, Cor. 2])

$$u_i = \binom{n}{i} - 2w\binom{n-w}{i-1} + \delta\binom{n-2w+\delta}{i-1} + (w-\delta)^2\binom{n-2w+\delta}{i-2}.$$

We can generalize this approach for $\tau > 2$ rows in $H^{(\tau)}$ by using the principle of inclusion-exclusion. However, this leads to explosion of terms in the formula for u_i. We do not continue in that direction.

2.2. Achieving maximum-likelihood performance

The ML decoder provides the best decoding error performance for a variety of memoryless channels. As it was mentioned in Section 1.2.4, for the BEC, ML decoding is equivalent to solving a system of linear equations.

The reason for the difference in performance of the ML and the BP decoders is existence of (coverable) stopping sets in a parity-check matrix used for BP decoding. In the following sections, we aim at making BP decoding performance closer to that of ML performance.

2.2.1. ML-decodable stopping sets

In Section 2.1, we analysed techniques for removal of all stopping sets of size up to $d-1$. However, as it has been mentioned above, in order to approach the ML performance with BP decoding, one should aim at covering stopping sets of size equal to or larger than d too. This can be achieved by adjoining redundant rows to a parity-check matrix. The following two lemmas will be instrumental in the analysis that follows.

As before, let H be a parity-check matrix of a code $\mathcal{C}$. By $H^{(2^r)}$ we denote the matrix whose rows are all 2^r codewords of $\mathcal{C}^\perp$, and $H_{\mathcal{E}}^{(2^r)}$ denotes the matrix formed by columns of $H^{(2^r)}$ indexed by $\mathcal{E}$.

Lemma 21. *The following statements are equivalent:*

1. *columns of $H_{\mathcal{E}}$ are linearly dependent;*
2. *there exists a non-zero codeword* $\boldsymbol{c}$*, such that* $\mathrm{supp}(\boldsymbol{c}) \subseteq \mathcal{E}$*;*
3. *if all positions in $\mathcal{E}$ have been erased then the ML decoder fails.*

Proof. First, we show that 1) and 2) are equivalent. A set of columns of $H_{\mathcal{E}}$ is linearly dependent if and only if it has a non-empty subset of columns which sums up to an all-zero column. This subset of columns corresponds to a support of a non-zero codeword $\boldsymbol{c}$. Hence 1) and 2) are equivalent indeed.

Next, we show that 2) and 3) are equivalent. If a support of a non-zero codeword $\boldsymbol{c}$ has been erased, decoding fails due to the fact that both $\boldsymbol{c}$ and all-zero codeword are two different solutions to the linear system (1.3). Vice versa, if after erasing positions in $\mathcal{E}$ there are at least two solutions of (1.3), these two solutions are correct codewords of $\mathcal{C}$ with their supports differing on some subset of $\mathcal{E}$ only. Sum of these codewords is another codeword $\boldsymbol{c}$ with $\mathrm{supp}(\boldsymbol{c}) \subseteq \mathcal{E}$. □

Next, consider the case when the columns of $H_{\mathcal{E}}$ are linearly independent.

Lemma 22. *The following statements are equivalent:*

1. *columns of $H_{\mathcal{E}}$ are linearly independent;*
2. *$H_{\mathcal{E}}^{(2^r)}$ is an orthogonal array of strength $|\mathcal{E}|$.*

And if any of them holds then

3. *$\mathcal{E}$ is not a stopping set in $H^{(2^r)}$.*

Proof. Both statements 1) and 3) follow from 2) in a straightforward manner.

We prove next that 2) follows from 1). First of all, if there are redundant rows in H, we can ignore them and assume that $m = r$. Owing to the fact that columns of $H_{\mathcal{E}}$ are linearly independent, there exist $|\mathcal{E}|$ rows in $H_{\mathcal{E}}$ that form a full-rank square matrix. Then, each of the remaining $r - |\mathcal{E}|$ rows of $H_{\mathcal{E}}$ can be represented as a linear combination of these $|\mathcal{E}|$ rows. Without loss of generality assume that

$$H_{\mathcal{E}} = \left(\frac{B}{TB} \right) ,$$

where B is an $|\mathcal{E}| \times |\mathcal{E}|$ full-rank matrix, and T is a $(r - |\mathcal{E}|) \times |\mathcal{E}|$ matrix of coefficients.

Each row of $H_{\mathcal{E}}^{(2^r)}$ is bijectively mapped onto r coefficients of linear combination $\boldsymbol{\alpha} = (\boldsymbol{\alpha}' \mid \boldsymbol{\alpha}'')$, where $\boldsymbol{\alpha}' \in \mathbb{F}_2^{|\mathcal{E}|}$, and $\boldsymbol{\alpha}'' \in \mathbb{F}_2^{r-|\mathcal{E}|}$, as follows:

$$\boldsymbol{\alpha} \left(\frac{B}{TB} \right) = \boldsymbol{\alpha}' B + \boldsymbol{\alpha}'' TB = (\boldsymbol{\alpha}' + \boldsymbol{\alpha}'' T)B \, .$$

Fix the vector $\boldsymbol{\alpha}''$ (and therefore the vector $\boldsymbol{\alpha}'' T$ of size $|\mathcal{E}|$ is fixed). Then, the transformation

$$\boldsymbol{\alpha}' \mapsto \boldsymbol{\alpha}' + \boldsymbol{\alpha}'' T$$

is a bijection of $\mathbb{F}_2^{|\mathcal{E}|}$. Since B is a full-rank matrix, the transformation

$$\boldsymbol{\alpha}' \mapsto (\boldsymbol{\alpha}' + \boldsymbol{\alpha}'' T)B$$

is a bijection too. Hence, for a fixed $\boldsymbol{\alpha}''$, if we iterate over all $\boldsymbol{\alpha}'$, each of the rows in $\mathbb{F}_2^{|\mathcal{E}|}$ is generated exactly once.

This holds for each of $2^{r-|\mathcal{E}|}$ possible choices for $\boldsymbol{\alpha}''$. Hence, each vector of $\mathbb{F}_2^{|\mathcal{E}|}$ appears as a row in $H_{\mathcal{E}}^{(2^r)}$ exactly $2^{r-|\mathcal{E}|}$ times. Thus, $H_{\mathcal{E}}^{(2^r)}$ is an orthogonal array of strength $|\mathcal{E}|$. □

We can summarise the results of Lemmas 21 and 22. Assume that $\mathcal{S}$ is a stopping set in a parity-check matrix of a code $\mathcal{C}$ and, during transmission of a codeword, the positions indexed by $\mathcal{S}$ have been erased. We have two cases. If the columns of $H_{\mathcal{S}}$ are linearly independent (and therefore there is no codeword $\boldsymbol{c} \in \mathcal{C}$ with $\operatorname{supp}(\boldsymbol{c}) \subseteq \mathcal{S}$), then the ML decoder can decode this erasure pattern. Also, $\mathcal{S}$ is a coverable stopping set and there exists a parity-check matrix (possibly, with redundant rows) that allows the BP decoder to decode this erasure pattern. Alternatively, if the columns of $H_{\mathcal{S}}$ are linearly dependent (and therefore there exists a codeword $\boldsymbol{c} \in \mathcal{C}$ with $\operatorname{supp}(\boldsymbol{c}) \subseteq \mathcal{S}$), the ML decoder fails and, therefore, the BP decoder fails too.

This leads us to the following definition.

Definition 23. Let H be a parity-check matrix (of rank r) of a code $\mathcal{C}$. We say that a stopping set $\mathcal{S}$ in H is *ML-decodable* (with respect to $\mathcal{C}$) if columns of $H_{\mathcal{S}}$ are linearly independent.

A stopping set $\mathcal{S}$ is ML-decodable if and only if no codeword $\boldsymbol{c} \in \mathcal{C}$ has $\mathrm{supp}(\boldsymbol{c}) \subseteq \mathcal{S}$. Note that this definition is independent of a particular parity-check matrix of the code $\mathcal{C}$, as the columns indexed by $\mathcal{S}$ are linearly-independent in any parity-check matrix (of rank r) of $\mathcal{C}$. Obviously, each ML-decodable stopping set is coverable.

We can now generalise Definition 20.

Definition 24. The ℓ-th *stopping redundancy* of $\mathcal{C}$, $1 \leq \ell \leq r$, is the smallest non-negative integer $\rho_\ell(\mathcal{C})$ such that there exists a (possibly redundant) parity-check matrix of $\mathcal{C}$ with $\rho_\ell(\mathcal{C})$ rows and no ML-decodable stopping sets of size $1, 2, \ldots, \ell$. The ordered set of integers $(\rho_1(\mathcal{C}), \rho_2(\mathcal{C}), \ldots, \rho_r(\mathcal{C}))$ is called the *stopping redundancy hierarchy* of $\mathcal{C}$.

The difference from Definition 20 (and, equivalently, [21, Def. 2.4]) is that, in Definition 24, ℓ can be as large as r (while in [21], $\ell \leq d-1$, which is a more limiting condition). Additionally, in Definition 24, only ML-decodable stopping sets are eliminated. However, all the stopping sets of size $\ell \leq d-1$ are of full column rank, and therefore Definition 24 contains [21, Def. 2.4] as a special case.

As we see, ML-decodable stopping sets are exactly those stopping sets that, if erased, can be decoded by the ML decoder (that is why their name). On the other hand, all of them are coverable. Therefore, our techniques for calculating probability of being covered in the proof of Theorem 19 are still valid. In the sequel, we re-formulate the upper bound.

We note that the r-th stopping redundancy $\rho_r(\mathcal{C})$ of $\mathcal{C}$ is the smallest number of rows in a parity-check matrix of $\mathcal{C}$ such that the BP decoder achieves the ML decoding performance, as no erasure pattern of size more than r can be decoded even by the ML decoder.

Definition 25. We call $\rho_r(\mathcal{C})$ a *maximum-likelihood (ML) stopping redundancy* of $\mathcal{C}$.

Next, we formulate an upper bound on the ℓ-th stopping redundancy, as defined in Definition 24, for $\ell \leq r$, $r = n-k$.

Theorem 26. *For an $[n, k, d]$ linear code $\mathcal{C}$ let $H^{(\tau)}$ be any $\tau \times n$ matrix consisting of τ different non-zero codewords of the dual code $\mathcal{C}^\perp$ and let u_i denote the number of not covered ML-decodable stopping sets of size i, $i = 1, 2, \ldots, \ell$ ($\ell \leq r$), in $H^{(\tau)}$. Then the ℓ-th stopping redundancy is*

$$\rho_\ell(\mathcal{C}) \leq \Xi_\ell^{(I)}(u_1, u_2, \ldots, u_\ell) \triangleq \tau + \min_{0 \leq t < 2^r - \tau} \{t + \kappa_t\} + \Delta\,,$$

where

$$\mathcal{D}_t = \sum_{i=1}^{\ell} u_i \prod_{j=\tau+1}^{\tau+t} \pi(r, i, j)\,,$$

$$P_{t,0} = \lfloor \mathcal{D}_t \rfloor\,,$$

$$P_{t,j} = \Big\lfloor \pi(r, \ell, \tau + t + j) P_{t,j-1} \Big\rfloor\,, \quad j = 1, 2, \ldots$$

$$\Delta = r - \max\left\{\operatorname{rank} H^{(\tau)}, \ell\right\} ,$$

and κ_t is the smallest j such that $P_{t,j} = 0$.

We remark that the difference between the statements of Theorem 26 and of Theorem 19 is that the value $d - 1$ is replaced by ℓ.

Proof. The proof follows the lines of that in Theorem 19 with the only difference that now for each ML-decodable stopping set $\mathcal{S}$, the corresponding matrix $H_{\mathcal{S}}^{(2^r)}$ contains all the tuples of size $|\mathcal{S}|$ equal number of times, as it was shown above.

Next, we analyze the rank deficiency. Let us denote by $H^{(\tau+t+\kappa_t)}$ the parity-check matrix we obtain by adding $t+\kappa_t$ rows to $H^{(\tau)}$ analogously to the procedure in the proof of Theorem 19. Note that if there is a stopping set $\mathcal{S}$ in $H^{(\tau+t+\kappa_t)}$ of size $|\mathcal{S}| \leq \ell$, then it is not ML-decodable and, consequently, there is a codeword $\boldsymbol{c} \in \mathcal{C}$ with $\operatorname{supp}(\boldsymbol{c}) \subseteq \mathcal{S}$.

Now, recall that $H^{(2^r)}$ is of rank $r \geq \ell$. Thus, there is a subset $\mathcal{I} \subseteq [n]$ of size $|\mathcal{I}| = \ell$ so that the columns of $H_{\mathcal{I}}^{(2^r)}$ are linearly independent. In particular, this means that there is no codeword $\boldsymbol{c} \in \mathcal{C}$ with $\operatorname{supp}(\boldsymbol{c}) \subseteq \mathcal{I}$. Consider $H_{\mathcal{I}}^{(\tau+t+\kappa_t)}$. If its columns are linearly independent then $\operatorname{rank} H^{(\tau+t+\kappa_t)} \geq \ell$.

Assume now to the contrary, that columns of $H_{\mathcal{I}}^{(\tau+t+\kappa_t)}$ are linearly dependent. This means there is a subset of columns $\mathcal{S} \subseteq \mathcal{I}$ that sum up to the all-zero column. Hence, the Hamming weight of each row of $H_{\mathcal{S}}^{(\tau+t+\kappa_t)}$ is even and $\mathcal{S}$ is a stopping set in $H^{(\tau+t+\kappa_t)}$. As it was mentioned above, this means there is a codeword $\boldsymbol{c} \in \mathcal{C}$ with $\operatorname{supp}(\boldsymbol{c}) \subseteq \mathcal{S} \subseteq \mathcal{I}$. This is a contradiction, and thus columns of $H_{\mathcal{I}}^{(\tau+t+\kappa_t)}$ are linearly independent. This in turn means that $\operatorname{rank} H^{(\tau+t+\kappa_t)} \geq \ell$.

On the other hand, $\operatorname{rank} H^{(\tau+t+\kappa_t)} \geq \operatorname{rank} H^{(\tau)}$. Therefore, it is enough to add Δ additional redundant rows to $H^{(\tau+t+\kappa_t)}$ to ensure the resulting rank to be r, as required for a parity-check matrix of $\mathcal{C}$. □

Corollary 27. *There exists an extended parity-check matrix with no more than $\Xi_r^{(I)}(u_1, u_2, \ldots, u_r)$ rows, such that the BP decoder with this matrix fails if and only if the ML decoder fails. It follows that the decoding error probability of these two decoders is equal.*

Computing the number u_i of ML-decodable stopping sets of size i—or even finding the corresponding upper bound—might be a difficult task for general codes, except for trivial cases. In what follows, we suggest two approaches:

- ensemble-average approach (see Section 2.2.2);
- finding estimates on u_i numerically (see Section 2.2.3).

2.2.2. Exact ensemble-average maximum-likelihood stopping redundancy

In order to apply the upper bounds on the stopping redundancy to ensemble-average values, we formulate a weaker bound inspired by [20].

Theorem 28. *Assume that $\mathcal{C}$ is a linear $[n,k]$-code and H is a parity-check matrix consisting of m different rows being codewords of the dual code $\mathcal{C}^\perp$, such that there are u_i ML-decodable stopping sets of size $i = 1, 2, \ldots, \ell$ ($\ell \le r$), in H. Then the ℓ-th stopping redundancy is bounded from above as follows:*

$$\rho_\ell(\mathcal{C}) \le \Xi_\ell^{(II)}(u_1, u_2, \ldots, u_\ell)$$
$$\triangleq m + \min_{0 \le t < 2^m - m} \left\{ t + \sum_{i=1}^{\ell} u_i \prod_{j=m+1}^{m+t} \pi(m,i,j) \right\}.$$

Proof. Analogous to Step 1 in Theorem 19, we again choose t codewords from $\mathcal{C}_0^\perp \setminus \{\text{rows of } H\}$ uniformly at random without repetitions and adjoin them to H. The average number of not covered ML-decodable stopping sets in this extended matrix becomes equal to

$$\sum_{i=1}^{\ell} u_i \prod_{j=m+1}^{m+t} \pi(r,i,j).$$

For each of these stopping sets, we add at most one row from $\mathcal{C}_0^\perp$ to cover it, and thus the total number of rows in the parity-check matrix becomes

$$m + t + \sum_{i=1}^{\ell} u_i \prod_{j=m+1}^{m+t} \pi(r,i,j) \;\le\; m + t + \sum_{i=1}^{\ell} u_i \prod_{j=m+1}^{m+t} \pi(m,i,j)\,.$$

By minimizing this expression over the choice of t, we obtain the required upper bound. We note that minimizing over t up to $2^m - m$ is just a matter of further convenience, as the true minimum value is obtained for $t < 2^r - m \le 2^m - m$. $\square$

Now we can formulate the ensemble-average result.

Corollary 29. *Consider an ensemble $\mathfrak{C}$ of codes, where the probability distribution of the codes is determined by the probability distribution on $m \times n$ parity-check matrices. Moreover, assume that the parity-check matrix H of rank $r = n - k$ corresponding to the $[n,k]$ code $\mathcal{C} \in \mathfrak{C}$ has $u_i^{(H)}$ ML-decodable stopping sets of size i, where $i = 1, 2, \cdots, \ell$. Denote the ensemble-average number of such stopping sets:*

$$\bar{u}_i = \mathbb{E}_{\mathfrak{C}} \left\{ u_i^{(H)} \right\}.$$

Then, the average ℓ-th stopping redundancy over the ensemble $\mathfrak{C}$ is bounded from above as follows:

$$\mathbb{E}_{\mathfrak{C}} \left\{ \rho_\ell(\mathcal{C}) \right\} \le \Xi_\ell^{(II)}(\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_\ell).$$

Proof. First, we observe that Theorem 28 yields an upper bound on $\rho_\ell(\mathcal{C})$ for every integer $0 \le t < 2^m - m$:

$$\rho_\ell(\mathcal{C}) \le m + t + \sum_{i=1}^{\ell} u_i \prod_{j=m+1}^{m+t} \pi(m,i,j).$$

Then, $\Xi_\ell^{(II)}$ is a minimum of these upper bounds over the values of t.

Fix some integer $0 \le t < 2^m - m$ and take the average over $\mathfrak{C}$:

$$\mathbb{E}_{\mathfrak{C}} \{\rho_\ell(\mathcal{C})\} \le m + t + \sum_{i=1}^{\ell} \bar{u}_i \prod_{j=m+1}^{m+t} \pi(m,i,j).$$

As it holds for each t, it should also hold for their minimum:

$$\mathbb{E}_{\mathfrak{C}} \{\rho_\ell(\mathcal{C})\} \le m + \min_{0 \le t < 2^r - m} \left\{ t + \sum_{i=1}^{\ell} \bar{u}_i \prod_{j=m+1}^{m+t} \pi(m,i,j) \right\}. \qquad \square$$

2.2.3. Statistical estimation of the number of ML-decodable stopping sets

In this section, we aim at finding statistical estimates on the number of ML-decodable stopping sets and further use them in the upper bounds on the stopping redundancy hierarchy.

Lemma 30. *Consider a parity-check matrix H of an $[n,k]$-code $\mathcal{C}$. For $1 \le i \le r$, fix a number N_i and generate N_i random subsets of $[n]$ uniformly at random (with repetitions), namely $\mathcal{S}_1^{(i)}, \mathcal{S}_2^{(i)}, \ldots, \mathcal{S}_{N_i}^{(i)}$, each subset consisting of i elements. For $j = 1, 2, \ldots, N_i$, we define the following events:*

$$x_j^{(i)} = \begin{cases} 1, & \text{if } \mathcal{S}_j^{(i)} \text{ is an ML-decodable stopping set in } H, \\ 0, & \text{otherwise.} \end{cases}$$

If u_i is a number of ML-decodable stopping sets of size i in H, and ε_i is some small fixed number then[3]

$$\mathbb{P}\{u_i < \hat{u}_i\} = 1 - \varepsilon_i,$$

where

$$\hat{u}_i = \binom{n}{i} \left(\tilde{x}^{(i)} + \kappa \sqrt{\frac{\hat{V}}{N_i} + \frac{\gamma_1 \hat{V} + \gamma_2}{N_i^2}} \right), \tag{2.5}$$

[3] We note that in fact this probability is approximate but it becomes exact for $N_i \to \infty$. We refer interested reader to [5] and the references therein for more details.

$$\kappa = \Phi^{-1}(1-\varepsilon_i), \qquad \eta = \kappa^2/3 + 1/6, \tag{2.6}$$

$$\bar{x}^{(i)} = \frac{\sum_{j=1}^{N_i} x_j^{(i)}}{N_i}, \qquad \tilde{x}^{(i)} = \frac{N_i \bar{x}^{(i)} + \eta}{N_i + 2\eta}, \tag{2.7}$$

$$\gamma_1 = -\frac{13}{18}\kappa^2 - \frac{17}{18}, \qquad \gamma_2 = \frac{\kappa^2}{18} + \frac{7}{36}, \tag{2.8}$$

$$\hat{V} = \bar{x}^{(i)}(1 - \bar{x}^{(i)}) \,. \tag{2.9}$$

Proof. Random variables $\left\{x_j^{(i)}\right\}$ are independent and identically distributed according to the Bernoulli distribution with success probability

$$\theta_i = \frac{u_i}{\binom{n}{i}} \,.$$

Here θ_i is unknown because u_i is unknown.

We further apply the $1-\varepsilon_i$ upper limit second-order corrected one-sided confidence interval constructed in [5, (10)] and based on Edgeworth expansion. In our notation, it states that

$$\mathbb{P}\left\{\theta_i < \tilde{x}^{(i)} + \kappa\sqrt{\frac{\hat{V}}{N_i} + \frac{\gamma_1 \hat{V} + \gamma_2}{N_i^2}}\right\} = 1 - \varepsilon_i. \tag{2.10}$$

From this we obtain the required result. □

This estimate can be used in conjunction with the upper bounds in Theorem 26 and Theorem 28. More specifically, we fix $N_1, N_2, \ldots, N_\ell$, and $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_\ell$, and then we obtain that

$$\rho_\ell(\mathcal{C}) \le \Xi_\ell^{(I)}\left(\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_\ell\right),$$

which holds with probability

$$\prod_{i=1}^{\ell}(1-\varepsilon_i) \,.$$

Furthermore, this approach can be extended to estimating the ensemble-average ℓ-th stopping redundancy, $\mathbb{E}_{\mathfrak{C}}\left\{\rho_\ell(\mathcal{C})\right\}$, as follows.

Lemma 31. *In the settings of Corollary 29, for $1 \le i \le m$, fix a number N_i and generate N_i random pairs $\left(H_j^{(i)}, \mathcal{S}_j^{(i)}\right)$, $j = 1, 2, \ldots, N_i$, where $H_j^{(i)}$ is a parity-check matrix of a code from $\mathfrak{C}$, and $S_j^{(i)}$ is a random subset of $[n]$ consisting of i elements, $H_j^{(i)}$ and $\mathcal{S}_j^{(i)}$ being independent.*

For $j = 1, 2, \ldots, N_i$, we define the following events:

$$y_j^{(i)} = \begin{cases} 1, & \text{if } \mathcal{S}_j^{(i)} \text{ is an ML-decodable stopping set in } H_j^{(i)}, \\ 0, & \text{otherwise.} \end{cases}$$

For a fixed small ε_i,

$$\mathbb{P}\left\{\bar{u}_i < \hat{\hat{u}}_i\right\} = 1 - \varepsilon_i,$$

where $\hat{\hat{u}}_i$ is defined similar to $\hat{u}_i$ in (2.5) *to* (2.9) *with $x_j^{(i)}$, $\bar{x}^{(i)}$ and $\tilde{x}^{(i)}$ replaced by $y_j^{(i)}$, $\bar{y}^{(i)}$ and $\tilde{y}^{(i)}$, respectively.*

Proof. Analogous to the proof of Lemma 30. □

If we fix $N_1, N_2, \ldots, N_\ell$ and $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_\ell$, we obtain that

$$\mathbb{E}_{\mathfrak{C}}\left\{\rho_\ell(\mathcal{C})\right\} \leq \Xi_\ell^{(II)}(\hat{\hat{u}}_1, \hat{\hat{u}}_2, \ldots, \hat{\hat{u}}_\ell) \tag{2.11}$$

with probability $\prod_{i=1}^{\ell}(1 - \varepsilon_i)$.

2.2.4. Case study: standard random ensemble

In this section, we demonstrate application of the aforementioned bounds to the standard random ensemble $\mathfrak{S}(n, m)$ defined in Example 4.

As it is shown in Appendix B, for $i \leq m$, the number of full-rank $m \times i$ matrices with no rows of weight one is equal to $\mathcal{N}(m, i)$ defined in (B.1). Fix some subset of columns $\mathcal{S}$ of size i, and choose a random parity-check matrix H from the ensemble $\mathfrak{S}(n, m)$. The probability that there is an ML-decodable (but not covered) stopping set in the columns indicated by $\mathcal{S}$, is as follows:

$$\frac{\mathcal{N}(m, i)}{2^{mi}} . \tag{2.12}$$

We used here the fact that $H_{\mathcal{S}}$, the submatrix of H consisting of columns indexed by $\mathcal{S}$, is equal to every $m \times i$ matrix equiprobably. Therefore, the average number of not covered ML-decodable stopping sets of size i in H is

$$\bar{u}_i = \mathbb{E}_{\mathfrak{S}(n,m)}\left\{u_i^{(H)}\right\} = \binom{n}{i}\frac{\mathcal{N}(m, i)}{2^{mi}} .$$

Next, we can apply Corollary 29 to obtain the upper bound on the ensemble-average ℓ-th stopping redundancy.

We illustrate the behaviour of the obtained bound in Fig. 10. It can be observed empirically that the bound grows exponentially. We remark that the presented values of the upper bound on the maximal stopping redundancy (Fig. 10, Table 5, Fig. 15) in some cases can take on very large values. In this work, we only show consistency of the obtained numerical results and the theoretical bounds. However, our experiments with short to moderate length codes [3, 4] show that decoding with redundant parity-check matrices can be a practical near-ML decoding technique in some cases.

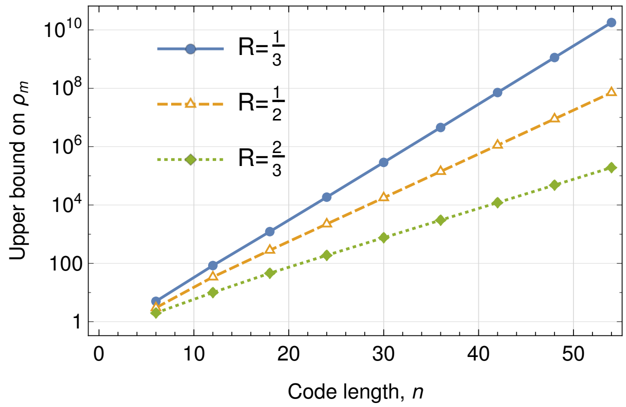


Figure 10. Upper bounds on $\mathfrak{S}(n,m)$-average m-th stopping redundancy ($m = (1-R)n$).

2.3. Numerical results

2.3.1. $[24, 12, 8]$ extended Golay code

Consider the $[24, 12, 8]$ extended Golay code. We use the systematic double-circulant parity-check matrix H given in [35, p. 65] as a means to define the code (see Table 2). The matrix has the stopping distance $s(H) = 4$.

Due to the small size of the parity-check matrix, we are able to calculate the values $u_1, u_2, \ldots, u_{12}$ by exhaustive checking of all the subsets of $\{1, 2, \ldots, 24\}$ of size up to 12. We use these values to calculate the upper bounds in Theorems 26 and 28.

Next, we generate $N_i = 1000$ $(1 \le i \le 12)$ random subsets of $\{1, 2, \cdots, 24\}$ and register the events according to Lemma 30. The following sequence of frequencies of ML-decodable stopping sets (as defined in Lemma 30) was obtained:

$$\left\{\bar{x}^{(i)}\right\}_{i=1}^{12} = \{0, 0, 0, 0.01, 0.039, 0.122, 0.219, 0.345, \\ 0.487, 0.621, 0.652, 0.463\}\,.$$

We repeat the experiments with a different value $N_i = 10^6$ $(1 \le i \le 12)$, and obtain the following sequence of frequencies:

$$\left\{\bar{x}^{(i)}\right\}_{i=1}^{12} = \{0, 0, 0, 0.010314, 0.042985, 0.109956, 0.214436, \\ 0.350958, 0.496478, 0.616122, 0.635654, 0.440123\}\,.$$

By setting $\varepsilon_i = 0.001$ for all i (therefore, $\prod_{i=1}^{12}(1-\varepsilon_i) = 0.988066$), we employ both sets of values in Lemma 30 and, further, in Theorems 26 and 28. The results are presented in Table 3. We observe consistency between the theoretical and the empirical results presented therein.

Table 2. Systematic double-circulant parity-check matrix of the $[24, 12, 8]$ extended Golay code. Dots denote zero entries.

$$
\begin{pmatrix}
1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot \\
1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & 1 \\
1 & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot \\
1 & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot & 1 & 1 & 1 & \cdot & \cdot \\
1 & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot & 1 & 1 & 1 & \cdot \\
1 & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot & 1 & 1 & 1 \\
1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot & 1 & 1 \\
1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot & 1 \\
1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot \\
1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 \\
1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{pmatrix}
$$

Table 3. Stopping redundancy hierarchies of the $[24, 12, 8]$ extended Golay code.

	ML-decodable stopping sets		Theorem 26		Theorem 28	
Exact u_i	u_1	0	ρ_1	12	ρ_1	12
	u_2	0	ρ_2	12	ρ_2	12
	u_3	0	ρ_3	12	ρ_3	12
	u_4	110	ρ_4	25	ρ_4	27
	u_5	1837	ρ_5	49	ρ_5	51
	u_6	14 795	ρ_6	91	ρ_6	95
	u_7	74 349	ρ_7	168	ρ_7	174
	u_8	257 796	ρ_8	304	ρ_8	316
	u_9	649 275	ρ_9	540	ρ_9	560
	u_{10}	1 206 755	ρ_{10}	927	ρ_{10}	960
	u_{11}	1 585 794	ρ_{11}	1507	ρ_{11}	1558
	u_{12}	1 189 574	ρ_{12}	2241	ρ_{12}	2309
Estimates $\hat{u}_i$ $(N_i = 10^3)$	$\hat{u}_1$	0	ρ_1	12	ρ_1	12
	$\hat{u}_2$	1	ρ_2	13	ρ_2	13
	$\hat{u}_3$	12	ρ_3	17	ρ_3	17
	$\hat{u}_4$	247	ρ_4	28	ρ_4	30
	$\hat{u}_5$	2596	ρ_5	51	ρ_5	53
	$\hat{u}_6$	21 061	ρ_6	94	ρ_6	98
	$\hat{u}_7$	90 406	ρ_7	171	ρ_7	178
	$\hat{u}_8$	288 582	ρ_8	307	ρ_8	319
	$\hat{u}_9$	700 573	ρ_9	544	ρ_9	564
	$\hat{u}_{10}$	1 309 119	ρ_{10}	933	ρ_{10}	967
	$\hat{u}_{11}$	1 740 882	ρ_{11}	1519	ρ_{11}	1570
	$\hat{u}_{12}$	1 384 130	ρ_{12}	2265	ρ_{12}	2333
Estimates $\hat{u}_i$ $(N_i = 10^6)$	$\hat{u}_1$	0	ρ_1	12	ρ_1	12
	$\hat{u}_2$	0	ρ_2	12	ρ_2	12
	$\hat{u}_3$	0	ρ_3	12	ρ_3	12
	$\hat{u}_4$	112	ρ_4	25	ρ_4	27
	$\hat{u}_5$	1853	ρ_5	49	ρ_5	51
	$\hat{u}_6$	14 930	ρ_6	91	ρ_6	95
	$\hat{u}_7$	74 656	ρ_7	168	ρ_7	174
	$\hat{u}_8$	259 204	ρ_8	304	ρ_8	316
	$\hat{u}_9$	651 167	ρ_9	540	ρ_9	561
	$\hat{u}_{10}$	1 211 318	ρ_{10}	927	ρ_{10}	961
	$\hat{u}_{11}$	1 590 393	ρ_{11}	1508	ρ_{11}	1559
	$\hat{u}_{12}$	1 194 310	ρ_{12}	2241	ρ_{12}	2310

2.3.2. Greedy heuristics for a redundant parity-check matrix

In [46], the authors suggest a greedy (lexicographic) algorithm to search for redundant rows in order to remove all stopping sets of size up to 7. The algorithm requires the full list of stopping sets, as well as the full list of dual codewords. We note that this straightforward approach is applicable to the Golay code due to its short length.

Based on the ideas discussed in Section 2.2, we can apply the algorithm akin to that of Schwartz and Vardy beyond the code minimum distance. In that case, the algorithm works with the full list of *ML-decodable* stopping sets of the code. We now describe the algorithm in more detail.

Fix ℓ, $4 \le \ell \le 12$, and generate the list

$$\mathcal{L} = \{\mathcal{S} \subseteq [n] : |\mathcal{S}| \le \ell,\, \operatorname{rank} H_{\mathcal{S}} = |\mathcal{S}|\}\,,$$

i.e. the list of ML-decodable stopping sets of size up to ℓ (including) (with respect to the Golay code) in an "empty" parity-check matrix (before putting any rows). Next, we iteratively construct a parity-check matrix. At each iteration, we find one of the 4095 non-zero dual codewords[4] with the highest score. The score is of heuristic nature and for a dual codeword $\boldsymbol{h}$ it is calculated as follows:

$$\operatorname{score}(\boldsymbol{h}) = \sum_{\mathcal{S}\in\mathcal{L}} |\mathcal{S}| \cdot \mathbb{I}\{\boldsymbol{h} \text{ covers } \mathcal{S}\}.$$

The row $\boldsymbol{h}^*$ with the maximum score is added to the matrix we build, and the stopping sets covered by $\boldsymbol{h}^*$ are removed from $\mathcal{L}$. Iterations continue until $\mathcal{L}$ is empty. As we have only ML-decodable stopping sets in $\mathcal{L}$ (all of them are coverable), the algorithm stops before we add all the 4095 rows. To this end, we verify that the obtained parity-check matrix has rank 12.

A small difference with [46] in the proposed approach is a random choice of $\boldsymbol{h}^*$ when several dual codewords have the same score. In that case, we run the algorithm several times and choose the matrix with the least number of rows. Fig. 11 illustrates the number of rows in the best obtained matrices for $\ell = 4, 5, \dots, 12$. We further refer to these matrices as $H^{(12)}$, $H^{(16)}$, $H^{(23)}$, $H^{(34)}$, $H^{(54)}$, $H^{(86)}$, $H^{(139)}$, $H^{(232)}$, and $H^{(370)}$, according to the number of rows they have.

Table 4 shows the numbers of undecodable patterns for the aforementioned extended parity-check matrices. The notation Ψ is used to denote the number of such patterns in a parity-check matrix. Note that the number of such patterns for the BP decoder with $H^{(370)}$ is *exactly* the same as for the ML decoder. This is in accordance with the discussion in Section 2.2.

Further, let $\Psi(w)$ be a number of erasure patterns of weight w, $0 \le w \le n$, in a code of length n, that cannot be decoded by some decoding method over the

[4] Recall that the $[24, 12, 8]$ extended Golay code is self-dual.

Table 4. Number of undecodable erasure patterns for different parity-check matrices of the $[24, 12, 8]$ extended Golay code.

	w										
	0–3	4	5	6	7	8	9	10	11	12	≥ 13
Total patterns		10 626	42 504	134 596	346 104	735 471	1 307 504	1 961 256	2 496 144	2 704 156	$\binom{24}{w}$
Ψ_H	0	110	2277	19 723	100 397	343 035	844 459	1 568 875	2 274 130	2 637 506	$\binom{23}{w}$
$\Psi_{H^{(34)}}$	0	0	0	0	0	3598	82 138	585 157	1 717 082	2 556 402	$\binom{24}{w}$
$\Psi_{H^{(54)}}$	0	0	0	0	0	759	16 424	195 190	1 027 002	2 242 956	$\binom{24}{w}$
$\Psi_{H^{(86)}}$	0	0	0	0	0	759	12 144	98 822	570 567	1 774 724	$\binom{24}{w}$
$\Psi_{H^{(139)}}$	0	0	0	0	0	759	12 144	91 080	437 744	1 438 874	$\binom{24}{w}$
$\Psi_{H^{(232)}}$	0	0	0	0	0	759	12 144	91 080	425 040	1 324 074	$\binom{24}{w}$
$\Psi_{H^{(370)}} = \Psi_{ML}$	0	0	0	0	0	759	12 144	91 080	425 040	1 313 116	$\binom{24}{w}$

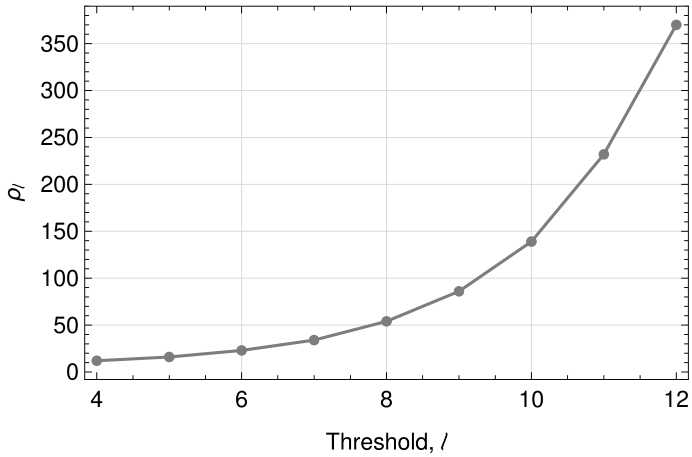


Figure 11. Upper bound on the stopping redundancy hierarchy of the $[24, 12, 8]$ extended Golay code obtained by greedy search.

BEC. Then, the frame error rate (also known as the block error rate) is a function of the bit erasure probability p, as follows:

$$\mathrm{FER}(p) = \sum_{w=0}^{n} \Psi(w) p^w (1-p)^{n-w} .$$

Based on the number of undecodable erasure patterns, we plot the performance curves in Fig. 12. We note that plots for $H^{(54)}$ and larger matrices are almost visually indistinguishable from the plot for $H^{(370)}$.

2.3.3. $[48, 24]$ low-density parity-check codes

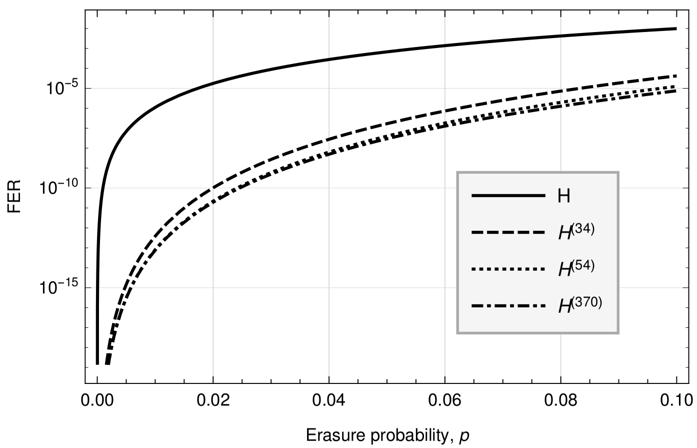
In this section, we consider four different LDPC codes of length 48 and dimension 24 (see Table 6).

$[48, 24]$-spBL denotes the best (linear) $[48, 24]$ code with a sparse parity-check matrix with the lowest possible correlation between its rows. Its minimum distance is 12.

$(4, 8)$-RU is a code chosen from 100 000 randomly-generated codes from RU ensemble. The code was chosen based on minimum distance, $d_{\min}$, and the first non-zero weight spectrum coefficient, $A_{d_{\min}}$ (i.e. the number of codewords of weight $d_{\min}$).

$(3, 6)$-QC is a QC LDPC code obtained by optimisation of lifting degrees for a constructed base matrix in order to guarantee the best possible minimum distance under a given restriction on the girth value of the Tanner graph of the code.

Finally, $(3, 6)$-NB denotes a binary image of non-binary code constructed by the standard two-stage procedure. It consists of labelling a proper binary base parity-check matrix by random non-zero elements of the extension of the binary field $\mathbb{F}_2$. Here, we select a parity-check matrix of a binary LDPC code from the RU ensemble.



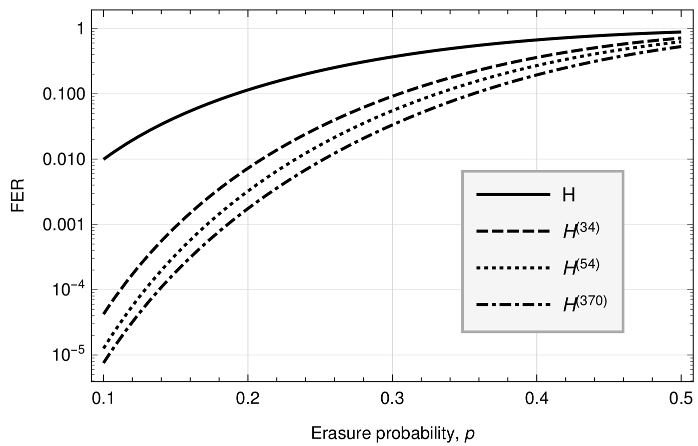


Figure 12. Frame error rates for different parity-check matrices of the $[24, 12, 8]$ extended Golay code, obtained by the randomized greedy algorithm. There are no ML-decodable stopping sets of size up to 3, 7, 8, and 12, for H, $H^{(34)}$, $H^{(54)}$, and $H^{(370)}$, respectively.

Table 5. ML stopping redundancies average over $\mathfrak{S}(n,m)$. Estimates hold with probability 95%.

n	$R = 1/3$			$R = 1/2$			$R = 2/3$		
	ρ_m	$\hat{\rho}_m$	$\varepsilon^{(m)}$, %	ρ_m	$\hat{\rho}_m$	$\varepsilon^{(m)}$, %	ρ_m	$\hat{\rho}_m$	$\varepsilon^{(m)}$, %
6	6	6	1.27	3	3	1.7	2	2	2.53
12	84.99	85	0.64	34.75	34.77	0.85	10.55	10.55	1.27
18	1223.92	1224.18	0.43	281.32	281.37	0.57	46.11	46.12	0.85
24	18 557	18 557.6	0.32	2234.5	2234.82	0.43	189.07	189.08	0.64
30	288 386	288 422	0.26	17 715.6	17 717.1	0.34	758.87	758.9	0.51
36	$4.5288 \cdot 10^6$	$4.5301 \cdot 10^6$	0.21	140 636	140 645	0.28	3027.58	3027.7	0.43
42	$7.1464 \cdot 10^7$	$7.1467 \cdot 10^7$	0.18	$1.1180 \cdot 10^6$	$1.1181 \cdot 10^6$	0.24	12 064.5	12 065.1	0.37
48	$1.1308 \cdot 10^9$	$1.1310 \cdot 10^9$	0.16	$8.8982 \cdot 10^6$	$8.8987 \cdot 10^6$	0.21	48 084	48 085.4	0.32
54	$1.7926 \cdot 10^{10}$	$1.7928 \cdot 10^{10}$	0.14	$7.0879 \cdot 10^7$	$7.0883 \cdot 10^7$	0.19	191 731	191 734	0.28

Table 6. Codes from Section 2.3.3.

code	(J,K)	$d_{\min}$	$A_{d_{\min}}$	d_{stop}	d_{dual}	$\rho_{d_{\min}}, \rho_{d_{\min}+1}, \rho_{d_{\min}+2}$	ρ_r
$[48,24]$-spBL	$(6,12)$	12	17 296	4	12	6240, 12 151, 23 468	13 761 585
$(4,8)$-RU	$(4,8)$	7	1	5	5	83, 175, 380	12 549 204
$(3,6)$-QC	$(3,6)$	7	8	7	5	58, 130, 274	9 876 964
$(3,6)$-NB	$(3,6)$	8	7	4	7	355, 751, 1551	13 819 276

We simulate the BP and ML decoding over the BEC channel for the four LDPC codes whose parameters are presented in Table 6. In Fig. 13, the FER performance of the BP and ML decoding over the BEC is compared. It is easy to see that the best BP decoding performance (and at the same time the worse ML decoding performance) is shown by the QC LDPC code with the most sparse parity-check matrix. We remark that $[48, 24]$-spBL, as expected, has the best ML decoding performance. Its BP decoding performance is worse than that of the selected LDPC codes except for the binary image of non-binary LDPC code.

Fig. 14 shows the BP decoding performance over the BEC of the codes $(3, 6)$-QC and $[48, 24]$-spBL from Table 6 when their parity-check matrices are extended. We call the corresponding decoding technique *redundant parity check (RPC)* decoding. The number next to "RPC" in Fig. 14 indicates the number of redundant rows that was added. The best convergence of FER performance of BP decoding to that of ML decoding is demonstrated by the QC LDPC code, while the best linear code has the slowest convergence. We observe that the obtained simulation results are consistent with the estimates on the stopping redundancy hierarchy given in Table 6.

2.3.4. Standard random ensemble

In this section, we apply the results of Lemma 31 to the standard random ensemble $\mathfrak{S}(n, m)$ (cf. Example 4). We calculate estimates on $\mathbb{E}_{\mathfrak{S}(n,m)}\left\{\rho_\ell(\mathcal{C})\right\}$ for different n and $m = (1 - R)n$ for design code rates $R \in \{1/3, 1/2, 2/3\}$. For each pair (n, m) and each size $i = 1, 2, \ldots, m$, we generate $N = 10^7$ pairs $\left(H^{(i)}, \mathcal{S}^{(i)}\right)$ and register the frequencies of $\mathcal{S}^{(i)}$ being an ML-decodable stopping set in $H^{(i)}$.

Based on the frequencies, we obtain estimates $\hat{\bar{u}}_i$ on the ensemble-average sizes $\bar{u}_i$. For each size of the stopping sets i, we use $\varepsilon_i = 1 - 0.95^{1/m}$, which gives a confidence of 95% that the estimates on $\bar{u}_i$ hold.

After that, we apply Corollary 29 in order to obtain bounds on $\mathbb{E}_{\mathfrak{C}}\left\{\rho_m(\mathcal{C})\right\}$ for selected values of m. These bounds are denoted by $\hat{\rho}_m$. Table 5 presents the resulting values. They are compared to the values $\Xi_\ell^{(II)}(\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_\ell)$ (obtained analytically, and denoted by ρ_m). We observe that the numerical results are a very good approximation to the theoretical values.

2.3.5. Gallager ensemble

We repeat the experiments of the previous subsection on the Gallager ensemble $\mathfrak{Gal}(n, J, K)$ (cf. Section 1.2.3) for different values of (J, K) and different lengths n. As it was mentioned earlier, the rank of each parity-check matrix in the ensemble is at most $r_{\max} = nJ/K - (J - 1)$. Therefore, the ML decoding performance is achieved when all the ML-decodable stopping sets of size up to $r_{\max}$ are covered. Fig. 15 demonstrates the values of the ML stopping redundancy, $\rho_{r_{\max}}$, for different lengths and different choices of J and K. We observe three clusters of plots according to the design rates of the codes.

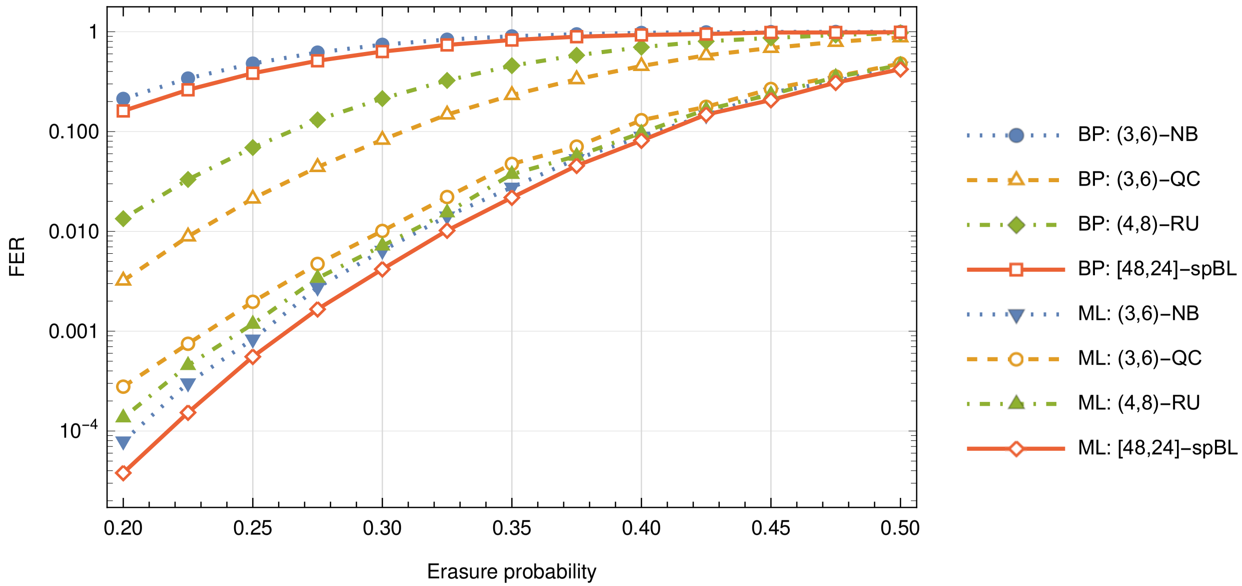


Figure 13. Comparison of FER performance of BP decoding over the BEC for $[48, 24]$ LDPC codes.

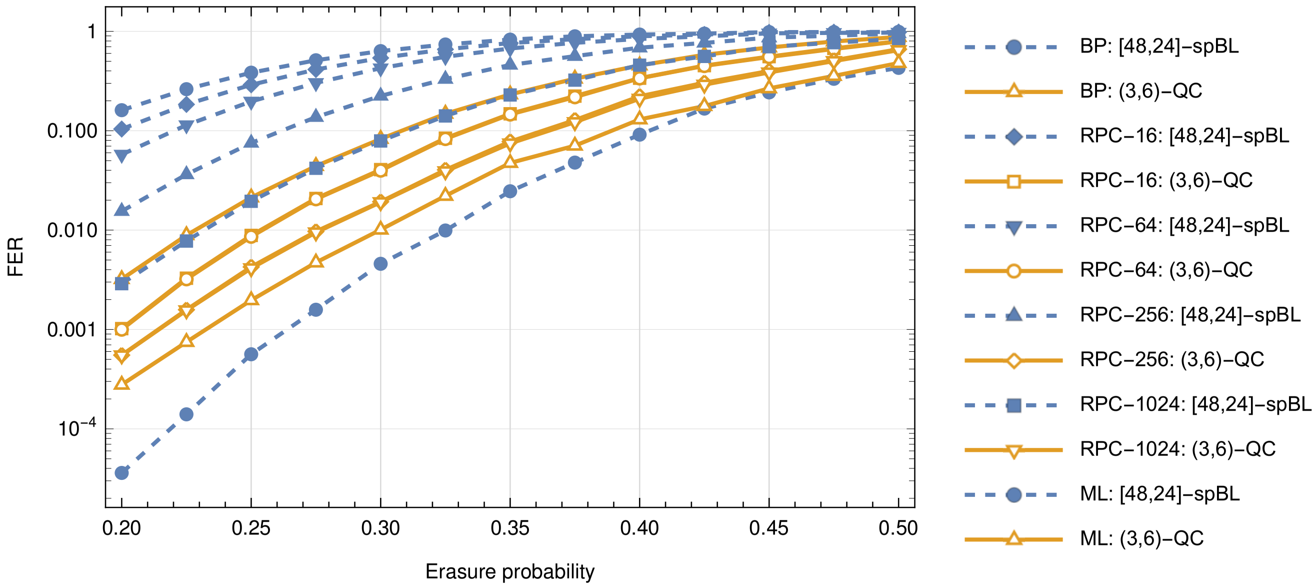


Figure 14. FER performance of BP, RPC, and ML decoding over the BEC for $[48, 24]$-spBL and $(3, 6)$-QC codes.

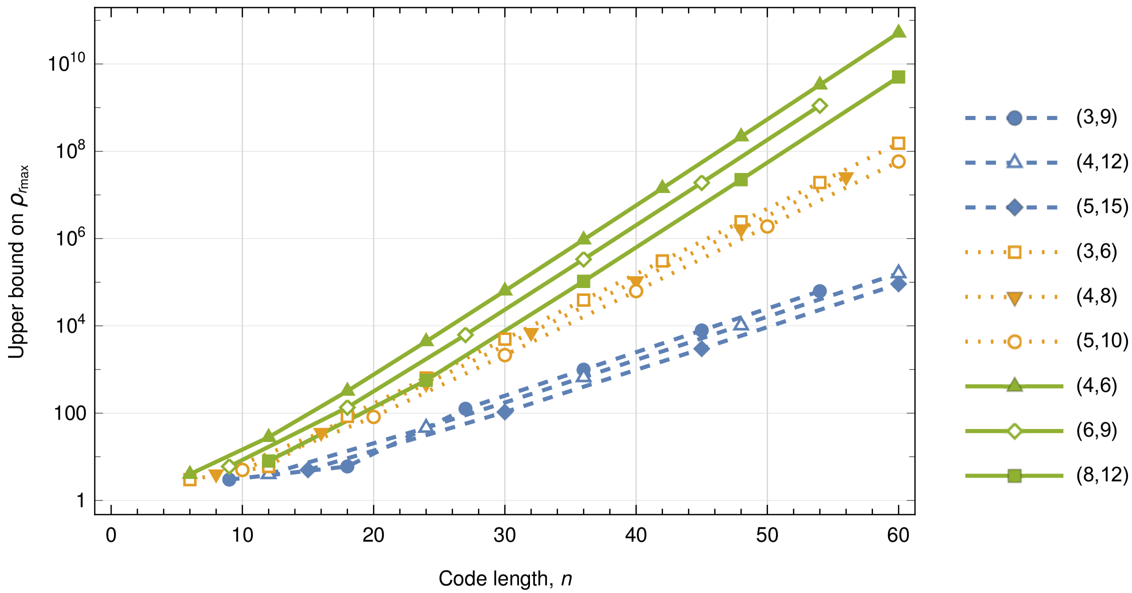


Figure 15. Upper bounds on $\mathfrak{Gal}(n, J, K)$-average $r_{\max}$-th stopping redundancy.

3. FAILURE ANALYSIS OF THE INTERVAL-PASSING ALGORITHM FOR COMPRESSED SENSING

Ambition is the last refuge of the failure.
—Oscar Wilde, *Phrases and Philosophies for the Use of the Young*

This chapter explores failures of the interval-passing algorithm. We start with a basic simplification, reducing the problem to the case of zeroes and ones (but the *binary* elements of $\mathbb{F}_2$). This reduction allows for introduction of *termatiko sets*, which are combinatorial structures in a Tanner graph corresponding to a measurement matrix. The concept of termatiko sets is central to this chapter because they are the solely cause of the IPA failures to correctly reconstruct the original signal. The size of the smallest termatiko set in a measurement matrix, *termatiko distance*, plays an important role in reconstruction abilities of the matrix under the IPA.

We formulate a criterion of the IPA failure, suggest some heuristics to find termatiko sets of a matrix, and examine some ideas how to improve reconstruction performance of the IPA, e.g. by increasing the termatiko distance of a particular matrix. After that, we study in great detail column-regular measurement matrices. In particular, we obtain some upper bounds and exact results on termatiko distance of array LDPC codes. The chapter concludes with a comprehensive set of numerical results.

The contents of this chapter are based on original conference publication [52] which was further extended to [53].

3.1. Failing sets of the interval-passing algorithm

In this section, we present several results related to failures of the IPA. In particular, in Section 3.1.1, we show that the IPA fails to recover $\boldsymbol{x}$ from $\boldsymbol{y}$ if and only if it fails to recover a corresponding binary vector of the same support, and also that only positions of non-zero values in the matrix A are of importance for success of recovery (see Lemma 32 below). Based on Lemma 32, we introduce the concept of termatiko sets in Section 3.1.2 and give a complete (graph-theoretic) description of the failing sets of the IPA in Section 3.1.3. In Section 3.1.4, a counter-example to [38, Thm. 2] is provided. Finally, two heuristic approaches to locate small-size termatiko sets from a list of stopping sets are outlined in Section 3.1.5.

3.1.1. Signal support recovery

Consider two related problems IPA($\boldsymbol{y}$, A) and IPA($\boldsymbol{s}$, B), where $\boldsymbol{s}^\mathsf{T} = B\boldsymbol{z}^\mathsf{T}$ and $\boldsymbol{z} \in \{0,1\}^n$ has support $\operatorname{supp}(\boldsymbol{z}) = \operatorname{supp}(\boldsymbol{x})$, i.e. $\boldsymbol{x}$ and $\boldsymbol{z}$ have the same support. The binary matrix B contains ones exactly in the positions where A has non-zero values. We show below in Lemma 32 that these two problems behave identically, namely that they recover exactly the same positions of $\boldsymbol{x}$ and $\boldsymbol{z}$. However, note that this is true if the *identical* algorithm (Algorithm 1) is applied to both problems, i.e. the binary nature of $\boldsymbol{z}$ is not exploited.

Lemma 32. *Let* $A = (a_{ji}) \in \mathbb{R}_{\geq 0}^{m\times n}$, $\boldsymbol{x} \in \mathbb{R}_{\geq 0}^{n}$, $B = (b_{ji}) \in \{0,1\}^{m\times n}$, *and* $\boldsymbol{z} \in \{0,1\}^n$, *where* $\operatorname{supp}(\boldsymbol{z}) = \operatorname{supp}(\boldsymbol{x})$ *and*

$$b_{ji} = \begin{cases} 0\,, & \text{if } a_{ji} = 0\,,\\ 1\,, & \text{otherwise}\,. \end{cases}$$

Further, denote $\boldsymbol{y}^\mathsf{T} = A\boldsymbol{x}^\mathsf{T}$, $\boldsymbol{s}^\mathsf{T} = B\boldsymbol{z}^\mathsf{T}$, $\hat{\boldsymbol{x}} = \mathrm{IPA}(\boldsymbol{y}, A)$, *and* $\hat{\boldsymbol{z}} = \mathrm{IPA}(\boldsymbol{s}, B)$. *Then, for all* $v \in V$,

$$\hat{x}_v = x_v \quad \text{if and only if} \quad \hat{z}_v = z_v\,.$$

Proof. Define subsets of V in which either the lower or the upper bound of a variable-to-measurement message, at a given iteration ℓ, is equal to x_v or z_v as follows:

$$\gamma_x^{(\ell)} = \left\{v \in V : \mu_{v\to\cdot}^{(\ell)} = x_v\right\}, \quad \Gamma_x^{(\ell)} = \left\{v \in V : M_{v\to\cdot}^{(\ell)} = x_v\right\},$$
$$\gamma_z^{(\ell)} = \left\{v \in V : \lambda_{v\to\cdot}^{(\ell)} = z_v\right\}, \quad \Gamma_z^{(\ell)} = \left\{v \in V : \Lambda_{v\to\cdot}^{(\ell)} = z_v\right\},$$

where $\lambda_{v\to\cdot}^{(\ell)}$ and $\Lambda_{v\to\cdot}^{(\ell)}$ denote, respectively, the lower and the upper bound of a message from variable node v to any measurement node $c \in \mathcal{N}(v)$ at iteration ℓ for IPA($\boldsymbol{s}$, B) (analogously to $\mu_{v\to\cdot}^{(\ell)}$ and $M_{v\to\cdot}^{(\ell)}$ for IPA($\boldsymbol{y}$, A)).

To prove the lemma, it is enough to show that at each iteration ℓ, $\gamma_x^{(\ell)} = \gamma_z^{(\ell)}$ and $\Gamma_x^{(\ell)} = \Gamma_z^{(\ell)}$. We demonstrate this by induction on ℓ.

Base case.

$$
\begin{aligned}
\gamma_x^{(0)} &= \{v \in V : x_v = 0\} = \{v \in V : z_v = 0\} = \gamma_z^{(0)}\,, \\
\Gamma_x^{(0)} &= \{v \in V : \exists c \in \mathcal{N}(v) \text{ s.t. } y_c = a_{cv} x_v\} \\
&= \{v \in V : \exists c \in \mathcal{N}(v) \text{ s.t. } s_c = z_v\} = \Gamma_z^{(0)}\,.
\end{aligned}
$$

Inductive step. Consider iteration $\ell \geq 1$. First note that all $v \in V$ with $x_v = 0$ (and hence $z_v = 0$) belong to both $\gamma_x^{(\ell)}$ and $\gamma_z^{(\ell)}$.

If $x_v > 0$ (and hence $z_v = 1$) then from Line 14 of Algorithm 1 and the definition of $\gamma_x^{(\ell)}$, we have $v \in \gamma_x^{(\ell)}$ if and only if there exists $c \in \mathcal{N}(v)$ such that $\mu_{c\to v}^{(\ell)} = x_v$. More precisely:

$$
\begin{aligned}
a_{cv} x_v &= y_c - \sum_{\substack{v' \in \mathcal{N}(c)\\ v' \neq v}} a_{cv'} M_{v'\to\cdot}^{(\ell-1)} \\
&= a_{cv} x_v + \sum_{\substack{v' \in \mathcal{N}(c)\\ v' \neq v}} a_{cv'} \left(x_{v'} - M_{v'\to\cdot}^{(\ell-1)}\right) \leq a_{cv} x_v\,.
\end{aligned}
$$

Equality holds if and only if $M_{v'\to\cdot}^{(\ell-1)} = x_{v'}$ for all $v' \in \mathcal{N}(c) \setminus \{v\}$ or, in our notation, $\mathcal{N}(c) \setminus \{v\} \subset \Gamma_x^{(\ell-1)}$. However, from the inductive assumption, $\Gamma_z^{(\ell-1)} = \Gamma_x^{(\ell-1)}$ and hence $\Lambda_{v'\to\cdot}^{(\ell-1)} = z_{v'}$ for all $v' \in \mathcal{N}(c)\setminus\{v\}$. This is equivalent to $\lambda_{c\to v}^{(\ell)} = z_v$ and thus $v \in \gamma_z^{(\ell)}$. Therefore, for all $v \in V$, v either belongs to both $\gamma_x^{(\ell)}$ and $\gamma_z^{(\ell)}$, or to none of them.

Analogously, we can show that $\Gamma_x^{(\ell)} = \Gamma_z^{(\ell)}$. Details are omitted for brevity. □

Lemma 32 gives a powerful tool for analysis of IPA performance. Instead of considering $A \in \mathbb{R}_{\geq 0}^{m\times n}$ and $\boldsymbol{x} \in \mathbb{R}_{\geq 0}^{n}$ we need to work only with binary A and $\boldsymbol{x}$ (although all operations are still performed over $\mathbb{R}$). Because of that, we assume that A is binary in the rest of the paper.

Example 33. Recall Example 17. The corresponding binary matrix is

$$
B = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}
$$

and the corresponding signal vector is $\boldsymbol{z} = (1, 1, 1, 0, 0, 0)$. Then the measurement vector is $\boldsymbol{s} = \boldsymbol{z}B^{\intercal} = (3, 1, 1, 1)$. Fig. 16 illustrates iterations of the IPA. See the similarities with Example 17.

△

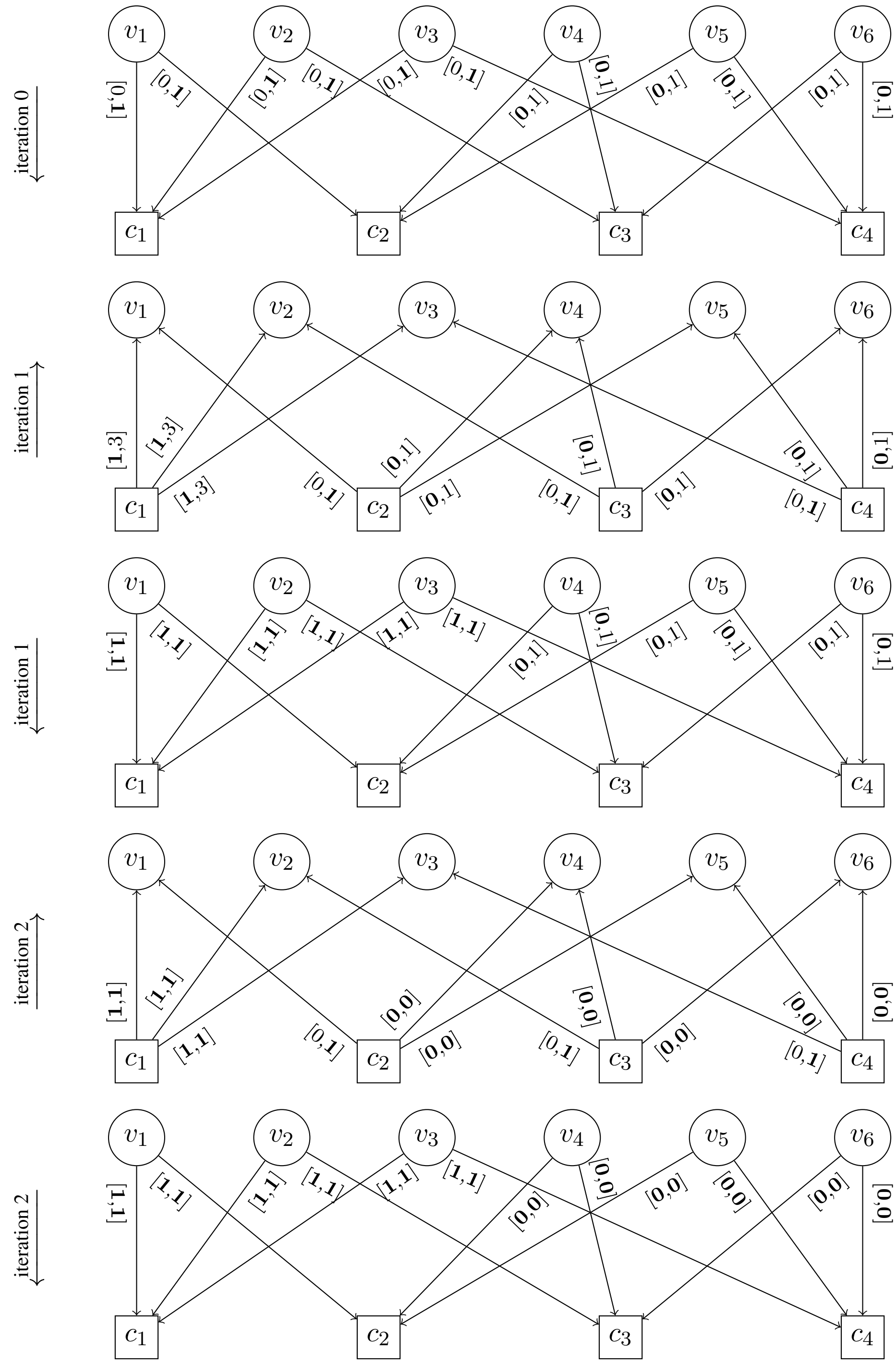


Figure 16. Example of IPA reconstruction with a 0/1 matrix. The original signal vector is $\boldsymbol{z} = (1, 1, 1, 0, 0, 0)$ and the measurement vector is $\boldsymbol{s} = (3, 1, 1, 1)$. Numbers in bold correspond to exact bounds. The last iteration is omitted because the signal has already been reconstructed. Compare the reconstruction process with Fig. 9.

3.1.2. Termatiko sets

We have shown in the previous section that IPA failures depend only on values being zero/non-zero. Therefore, for a particular measurement matrix A, we can speak about failure sets—analogous to stopping sets for BP decoding over the BEC. From Line 17 of Algorithm 1, we see that outputting zeroes is “default” behaviour of IPA, i.e. a zero will be output if the IPA has not advanced with a particular position reconstruction.[1]

We define termatiko sets through the complete failures of the IPA, i.e. when no non-zero positions have been reconstructed and the output is the all-zero vector.

Definition 34. We call $T \subset V$ a *termatiko set* if and only if $\mathrm{IPA}(\boldsymbol{x}_T A^\intercal, A) = \mathbf{0}$, where $\boldsymbol{x}_T$ is a binary vector with support $\mathrm{supp}(\boldsymbol{x}_T) = T$.

From Lemma 32, it follows that the IPA completely fails to recover $\boldsymbol{x} \in \mathbb{R}^n_{\geq 0}$ if and only if $\mathrm{supp}(\boldsymbol{x}) = T$, where T is a non-empty termatiko set.

Definition 34 is, in some sense, indirect. The following theorem gives a criterion of a termatiko set, in terms of a Tanner graph of a measurement matrix.

Theorem 35. *Let T be a subset of the variable nodes set V. We denote by $N = \mathcal{N}(T)$ the set of measurement nodes connected to T and by S, the other variable nodes connected only to N, as follows:*

$$S = \{v \in V \setminus T \,:\, \mathcal{N}_N(v) = \mathcal{N}(v)\} \,.$$

Then, T is a termatiko set if and only if for each $c \in N$ one of the following two conditions holds (cf. Figs. 18 and 19):

- *c is connected to S (this implies $S \neq \varnothing$);*
- *c is not connected to S and*

$$\Big|\big\{v \in \mathcal{N}_T(c) \,:\, \forall c' \in \mathcal{N}(v)\,,\, |\mathcal{N}_T(c')| \geq 2\big\}\Big| \geq 2\,.$$

Proof. Consider the problem $\mathrm{IPA}(\boldsymbol{x}_T A^\intercal, A)$, where $\boldsymbol{x}_T$ is a binary vector with support $\mathrm{supp}(\boldsymbol{x}_T) = T$ and T satisfies the conditions of the theorem.

We first note that measurement nodes in $C \setminus N$ have value zero and hence all variable nodes connected to them (i.e. $v \in V \setminus (T \cup S)$) are recovered as zeroes at the initialisation step of Algorithm 1. As a consequence, they can be safely pruned and w.l.o.g. we can assume that $C = N$ and $V = T \cup S$.

We show by induction that for all $v \in T \cup S$ at each iteration $\ell \geq 0$ it holds that $\mu^{(\ell)}_{v\to\cdot} = 0$ and $M^{(\ell)}_{v\to\cdot} \geq 1$. Moreover, each measurement node $c \in N$ that is not connected to S has at least two different neighbours $v_1, v_2 \in T$ with $M^{(\ell)}_{v_1\to\cdot} \geq 2$ and $M^{(\ell)}_{v_2\to\cdot} \geq 2$.

We will use the fact that

$$x_v = \begin{cases} 1\,, & \text{if } v \in T\,, \\ 0\,, & \text{if } v \in S\,. \end{cases}$$

[1] Yet it can also be the case that the true value at the position is zero indeed.

Also we note that $y_c = |\mathcal{N}_T(c)|$ for all $c \in N$.

Base case. For $\ell = 0$ we immediately obtain from Algorithm 1 that $\mu^{(0)}_{v\to\cdot} = 0$ and, as each $c \in N$ has at least one non-zero neighbour, $M^{(0)}_{v\to\cdot} \geq 1$. In addition, consider $c \in N$ that is not connected to S. It has at least two different neighbours $v_1, v_2 \in T$, each connected only to measurement nodes with not less than two neighbours in T. Therefore, $M_{v_1\to\cdot} \geq 2$ and $M_{v_2\to\cdot} \geq 2$.

Inductive step. Consider $\ell \geq 1$. For all $c \in N$ and all $v \in \mathcal{N}(c)$,

$$M^{(\ell)}_{c\to v} = y_c - \sum_{v' \in \mathcal{N}(c), v' \neq v} \mu^{(\ell-1)}_{v'\to\cdot} = y_c\,.$$

Hence, upper bounds are exactly the same as for $\ell = 0$ and the same inequalities hold for them.

In order to find lower bounds, we consider two cases for $c \in N$. If c is connected to S, then

$$y_c - \sum_{\substack{v' \in \mathcal{N}(c)\\ v' \neq v}} M^{(\ell-1)}_{v'\to\cdot} \leq (|\mathcal{N}(c)| - 1) - \sum_{\substack{v' \in \mathcal{N}(c)\\ v' \neq v}} 1 = 0$$

and therefore $\mu^{(\ell)}_{c\to v} = 0$. If c is connected to T only, then

$$y_c - \sum_{\substack{v' \in \mathcal{N}(c)\\ v' \neq v}} M^{(\ell-1)}_{v'\to\cdot} \leq |\mathcal{N}_T(c)| - \left(1 + \sum_{\substack{v' \in \mathcal{N}_T(c)\\ v' \neq v}} 1\right) = 0$$

and again $\mu^{(\ell)}_{c\to v} = 0$. Here, the extra 1 inside the parenthesis indicates the fact that for at least one v' we have $M^{(\ell-1)}_{v'\to\cdot} \geq 2$. Thus, at each iteration of the IPA for each $v \in V$ the lower bound is equal to zero, and the algorithm will return $\hat{\boldsymbol{x}} = \mathbf{0}$.

We have demonstrated that if T satisfies the conditions of the theorem, it is a termatiko set. What remains to be proven is that if T does *not* satisfy the conditions of the theorem, the IPA can recover at least some of the non-zero values.

Assume that there exists $c^* \in N$ connected to T only (i.e. $\mathcal{N}_T(c^*) = \mathcal{N}(c^*)$) and such that

$$\left| \left\{ v \in \mathcal{N}_T(c^*) \,:\, \forall c' \in \mathcal{N}(v)\,,\ |\mathcal{N}_T\left(c'\right)| \geq 2 \right\} \right| \leq 1\,.$$

If this set has one element, denote it by v^*. If it is empty, let v^* be any element of $\mathcal{N}_T(c^*)$.

A special case when $|\mathcal{N}_T(c^*)| = 1$ is trivial. Otherwise, for any $v \in \mathcal{N}_T(c^*) \setminus \{v^*\}$, there exists $c'_v \in \mathcal{N}(v)$ such that $|\mathcal{N}_T(c'_v)| \leq 1$, which in truth means that $\mathcal{N}_T(c'_v) = \{v\}$.

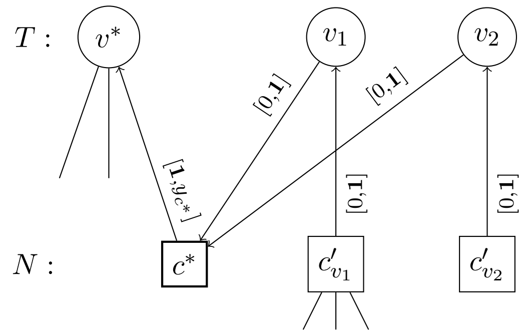


Figure 17. Exact bounds propagation in a non-termatiko set. Here $[\mu, M]$ denotes sending a lower bound of μ and an upper bound of M in the direction given by the corresponding arrow. Numbers in bold are the exact bounds.

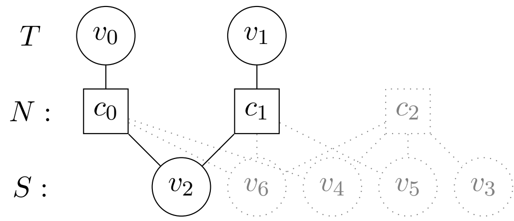


Figure 18. Example of a termatiko set T with all measurement nodes in N connected to both T and S (cf. Theorem 35). The rest of the Tanner graph is drawn dotted.

Hence, at the initialization step of the IPA, for all $v \in \mathcal{N}_T(c^*) \setminus \{v^*\}$ we will have $\mu^{(0)}_{v\to\cdot} = 0$ and $M^{(0)}_{v\to\cdot} = 1$. Therefore, at iteration $\ell = 1$:

$$\mu^{(1)}_{c^*\to v^*} \leftarrow y_{c^*} - \sum_{\substack{v \in \mathcal{N}_T(c^*) \\ v \neq v^*}} M^{(0)}_{v\to\cdot} = y_{c^*} - \sum_{\substack{v \in \mathcal{N}_T(c^*) \\ v \neq v^*}} 1 = 1\,.$$

Thus, the IPA will output 1 for position $v^* \in T$, which means that T is not a termatiko set. See Fig. 17 for illustration. $\square$

Theorem 35 gives a precise graph-theoretic description of termatiko sets. In fact, it defines two important subclasses of termatiko sets; stopping sets and sets

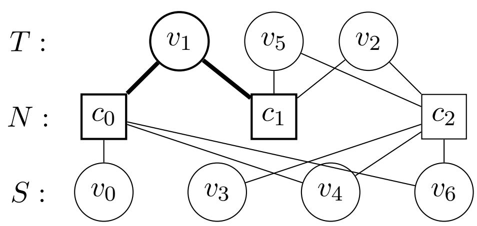


Figure 19. Example of a termatiko set T with a measurement node c_1 connected to T only (cf. Theorem 35). Highlighted is the connection to a measurement node c_0, which is connected to T only once.

with all $c \in N$ connected to both T and S (these classes have non-empty intersection). Also, it is worth noting that $T \cup S$ is a stopping set. Thus, a termatiko set is always a subset of some stopping set.

Definition 36. The size of the smallest non-empty termatiko set in a measurement matrix A is called the *termatiko distance* and denoted by $h_{\min}(A)$.

3.1.3. General failing sets

In Section 3.1.2, we defined termatiko sets as supports of binary vectors that avert the IPA from recovering *any* of the ones. However, the algorithm can fail *partially*, i.e. recover only some of the positions of ones.

Before proceeding further, we prove the following lemma.

Lemma 37. *Let $\boldsymbol{x}, \boldsymbol{x}' \in \{0,1\}^n$ be two vectors such that $\operatorname{supp}(\boldsymbol{x}) \subset \operatorname{supp}(\boldsymbol{x}')$ and denote $D = \operatorname{supp}(\boldsymbol{x}') \setminus \operatorname{supp}(\boldsymbol{x})$. Let $\mu^{(\ell)}$ and $M^{(\ell)}$ be respectively the lower and the upper bounds at the ℓ-th step of Algorithm 1 on input $(\boldsymbol{x}A^\mathsf{T}, A)$. Also, let $\lambda^{(\ell)}$ and $\Lambda^{(\ell)}$ be respectively the lower and the upper bounds at the ℓ-th step of Algorithm 1 on input $(\boldsymbol{x}'A^\mathsf{T}, A)$. Then, the following holds:*

$$\begin{aligned}
&\lambda^{(\ell)}_{v\to\cdot} \leq \mu^{(\ell)}_{v\to\cdot} &&\leq M^{(\ell)}_{v\to\cdot} &&\leq \Lambda^{(\ell)}_{v\to\cdot}\,, \quad \forall v \notin D\,,\\
&\lambda^{(\ell)}_{v\to\cdot} \leq \mu^{(\ell)}_{v\to\cdot} + 1 &&\leq M^{(\ell)}_{v\to\cdot} + 1 &&\leq \Lambda^{(\ell)}_{v\to\cdot}\,, \quad \forall v \in D\,.
\end{aligned}$$

Proof. Denote $\boldsymbol{y} = \boldsymbol{x}A^\mathsf{T}$ and $\boldsymbol{y}' = \boldsymbol{x}'A^\mathsf{T}$. Obviously, for any $c \in C$, $y'_c = y_c + |\mathcal{N}(c) \cap D| \geq y_c$. In particular, for any $c \in \mathcal{N}(D)$, $y'_c \geq y_c + 1$, and for all $c \notin \mathcal{N}(D)$, $y'_c = y_c$.

We prove the lemma by induction.

Base case. Obviously, $\lambda^{(0)}_{v\to\cdot} = \mu^{(0)}_{v\to\cdot} = 0$ for all $v \in V$. For $v \in D$, $c \in \mathcal{N}(v)$ implies $c \in \mathcal{N}(D)$ and hence $\Lambda^{(0)}_{v\to\cdot} \geq \min_{c\in\mathcal{N}(v)}(y_c + 1) = M^{(0)}_{v\to\cdot} + 1$. Analogously, for $v \notin D$, $\Lambda^{(0)}_{v\to\cdot} \geq M_{v\to\cdot}$.

Inductive step. Consider step $\ell \geq 1$. From Line 9 of Algorithm 1 we have:

$$\begin{aligned}
\lambda^{(\ell)}_{c\to v} &= y'_c - \sum_{\substack{v'\in\mathcal{N}(c)\\ v'\neq v}} \Lambda^{(\ell-1)}_{v'\to\cdot}\\
&= y_c + |\mathcal{N}(c) \cap D| - \sum_{\substack{v'\in\mathcal{N}(c)\cap D\\ v'\neq v}} \Lambda^{(\ell-1)}_{v'\to\cdot} - \sum_{\substack{v'\in\mathcal{N}(c)\setminus D\\ v'\neq v}} \Lambda^{(\ell-1)}_{v'\to\cdot}\\
&\leq y_c + |\mathcal{N}(c) \cap D| - \sum_{\substack{v'\in\mathcal{N}(c)\cap D\\ v'\neq v}} \left(M^{(\ell-1)}_{v'\to\cdot} + 1\right) - \sum_{\substack{v'\in\mathcal{N}(c)\setminus D\\ v'\neq v}} M^{(\ell-1)}_{v'\to\cdot}\\
&= \begin{cases} \mu^{(\ell)}_{c\to v}\,, & v \notin D\,,\\ \mu^{(\ell)}_{c\to v} + 1\,, & v \in D\,. \end{cases}
\end{aligned}$$

One can show in a similar manner that $\Lambda^{(\ell)}_{c\to v} \geq M^{(\ell)}_{c\to v} + 1$ for $v \in D$ and $\Lambda^{(\ell)}_{c\to v} \geq M^{(\ell)}_{c\to v}$ for $v \notin D$.

Finally, from Lines 14 and 15 of Algorithm 1 we obtain

$$\lambda^{(\ell)}_{v\to\cdot} = \max_{c'\in\mathcal{N}(v)} \lambda^{(\ell)}_{c'\to v} \le \max_{c'\in\mathcal{N}(v)} \mu^{(\ell)}_{c'\to v} = \mu^{(\ell)}_{v\to\cdot}, \text{ for } v\notin D,$$
$$\lambda^{(\ell)}_{v\to\cdot} = \max_{c'\in\mathcal{N}(v)} \lambda^{(\ell)}_{c'\to v} \le \mu^{(\ell)}_{v\to\cdot} + 1\,, \text{ for } v\in D,$$
$$\Lambda^{(\ell)}_{v\to\cdot} = \max_{c'\in\mathcal{N}(v)} \Lambda^{(\ell)}_{c'\to v} \ge M^{(\ell)}_{v\to\cdot}\,, \text{ for } v\notin D,$$
$$\Lambda^{(\ell)}_{v\to\cdot} = \max_{c'\in\mathcal{N}(v)} \Lambda^{(\ell)}_{c'\to v} \ge M^{(\ell)}_{v\to\cdot} + 1\,, \text{ for } v\in D. \qquad \square$$

The next theorem presents a connection between (partial) failures of the IPA and termatiko sets. In particular, it shows that the IPA fails on a signal in $\mathbb{R}^n_{\ge 0}$ if and only if its support contains a non-empty termatiko set.

Theorem 38. *The IPA fails to recover a non-negative real signal $\boldsymbol{x}' \in \mathbb{R}^n_{\ge 0}$ (i.e. $\mathrm{IPA}(\boldsymbol{x}'A^\mathsf{T}, A) \neq \boldsymbol{x}'$) if and only if the support of $\boldsymbol{x}'$ contains a non-empty termatiko set.*

Proof. According to Lemma 32, without loss of generality we can assume that $\boldsymbol{x}' \in \{0,1\}^n$ is a binary signal and A is a matrix with its entries being 0 or 1.

Assume T is a non-empty termatiko set such that $T \subset \mathrm{supp}(\boldsymbol{x}')$. We also consider a binary $\boldsymbol{x} \in \{0,1\}^n$ with $\mathrm{supp}(\boldsymbol{x}) = T$. Since T is a termatiko set, lower bounds on variable nodes in T will be zeroes on each step of $\mathrm{IPA}(\boldsymbol{x}A^\mathsf{T}, A)$. Further application of Lemma 37 to $\boldsymbol{x}$ and $\boldsymbol{x}'$ shows that lower bounds on variable nodes in T will be zeroes also on each step of $\mathrm{IPA}(\boldsymbol{x}'A^\mathsf{T}, A)$ and therefore these positions will be incorrectly recovered as zeroes.

To prove the converse direction, assume that $\hat{\boldsymbol{x}}' = \mathrm{IPA}(\boldsymbol{x}'A^\mathsf{T}, A) \neq \boldsymbol{x}'$. Since zero entries are always correctly recovered by the IPA, the only mistakes are ones being incorrectly recovered as zeroes. Let us define T as the corresponding positions:

$$T = \{v \in V \,:\, x'_v = 1 \text{ and } \hat{x}'_v = 0\}.$$

And let us define the vector $\boldsymbol{x} \in \{0,1\}^n$ as follows:

$$x_v = \begin{cases} 1, & \text{if } v \in T,\\ 0, & \text{otherwise.}\end{cases}$$

Obviously, $\mathrm{supp}(\boldsymbol{x}) = T \subseteq \mathrm{supp}(\boldsymbol{x}')$. Moreover, $\mathrm{IPA}(\boldsymbol{x}A^\mathsf{T}, \boldsymbol{x}) = \boldsymbol{0}$. Therefore, by the definition, T is a termatiko set. $\square$

3.1.4. Counterexample to [38, Thm. 2]

In [38, Thm. 2], a condition for full recovery of $\boldsymbol{x}$ is stated. However, we show that the theorem in not completely correct. We repeat the statement here (with a slightly adapted notation).

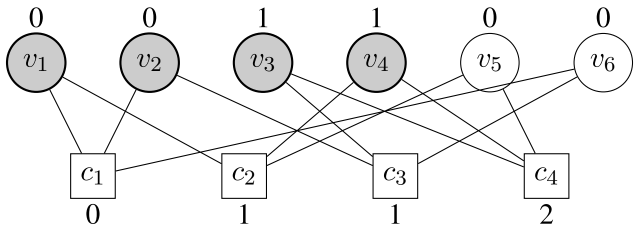


Figure 20. Counter-example to [38, Thm. 2]. The set of variable nodes is $V = \{v_1, \ldots, v_6\}$ (circles) and the set of measurement nodes is $C = \{c_1, \ldots, c_4\}$ (squares). The integer attached to a node is its corresponding value (x_{v_i} for variable node v_i and y_{c_i} for measurement node c_i). $V_S = \{v_1, v_2, v_3, v_4\} \subset V$ (shaded in grey) is the minimal stopping set and c_1 is a zero-valued ($y_{c_1} = 0$) measurement node connected to V_S. Note that v_5 is not in V_S, but exactly because of it, the IPA cannot correctly recover v_4.

Theorem 39 ([38, Thm. 2], incorrect)**.** *Let $A \in \mathbb{R}^{m \times n}$ be a binary measurement matrix and $V_S = \{v_1, v_2, \ldots, v_k\}$ be a subset of variable nodes forming a minimal stopping set.*[2] *Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n_{\geq 0}$ be a signal with at most $k-2$ non-zero values, i.e. $\|\boldsymbol{x}\|_0 \leq k-2$, such that the set of non-zero variables is a subset of V_S. Then, the IPA can recover $\boldsymbol{x}$ if there exists at least one zero measurement node among the neighbours of V_S.*

Fig. 20 illustrates a counterexample to the theorem. Note that the Tanner graph in Fig. 20 is $(2, 3)$-regular (only regular Tanner graphs with variable node degree at least two were considered in [38]) and satisfies the conditions of [38, Thm. 2]. In particular, there are at most $|V_S| - 2 = 2$ non-zero-valued variable nodes which are both in V_S (V_S is a minimal stopping set contained in V); and there is at least one zero-valued measurement node among the neighbours of V_S. However, it can be readily seen that the IPA outputs $\hat{\boldsymbol{x}} = (0, 0, 1, 0, 0, 0)$, i.e. it recovers only one non-zero variable node (v_4 and v_5 are both connected to c_2 and c_4 and thus indistinguishable; hence, the IPA will definitely fail).

We believe that the main problematic issue in the proof given in [38, Thm. 2] is that variable nodes outside of the minimal stopping set V_S are not considered. Despite the fact that such nodes will be recovered as zeroes (because of the specific implementation of the IPA, see Line 17 of Algorithm 1), during iterations they still can "disturb" the values inside of the stopping set.

Finally, we remark that since the statement of [38, Thm. 2] is used in the proof of [38, Thm. 3], the latter should be further verified.

3.1.5. Heuristics to find small-size termatiko sets

It has been shown above that stopping sets may contain termatiko sets as proper subsets (and every stopping set is a termatiko set itself). Thus, one way to locate termatiko sets is to first enumerate all stopping sets of size at most τ (for a given binary measurement matrix and threshold τ) and then look for subsets that are

[2] A stopping set is *minimal* if it does not contain a smaller stopping set.

termatiko sets. For a given binary measurement matrix A, small-size stopping sets can be identified using the algorithm from [44, 43].

Next, we present another heuristic approach that targets the subclass of termatiko sets mentioned in Section 3.1.2, namely the case when all $c \in N$ are connected to both T and S. This symmetry leads to the observation that both T and S are termatiko sets. Therefore, we can try to split a stopping set into two disjoint termatiko sets, T and S. We call stopping sets that allow such a split *splittable*.

Consider a stopping set $D \subset V$. Our goal is to split the variable nodes from D into two disjoint sets T and S such that $D = T \cup S$ and each $c \in N = \mathcal{N}(D)$ is connected to both T and S. The heuristic greedy Algorithm 2 tries to find such a split by painting (green or red) the variable nodes in D. The whole algorithm is based on the following idea. If there is $c \in N$ such that all its neighbours in D except exactly one have already been painted to the same colour, then the remaining node should be painted the colour opposite to other neighbours of c. In the algorithm, the colour of variable node $v \in D$ is denoted by col_v. It starts with a random node, paints it green (Line 5), and puts it into a working set Q of "freshly-painted" nodes. Further, at each iteration, it takes a random variable node v from Q and constructs the set of variable nodes *Opp*. A node $u \in D$ is included in *Opp* if it is not coloured and also connected via some c to v and all the neighbours of c in D except u have the same colour (Line 13). By our heuristic assumption, we paint all the variable nodes in *Opp* the colour opposite to the colour of v (Line 14). Further, all the elements of *Opp* are added to Q for further processing (Line 15). If at some point Q becomes empty but not all variable nodes from D have been painted yet, the algorithm has nothing better to do than just randomly guess a colour of some variable node that has not been painted yet (Lines 17 to 19). Algorithm 2 finishes when Q becomes empty and all the variable nodes from D have been painted. After that, in Line 20, the algorithm verifies the obtained solution for correctness, i.e. whether each $c \in N$ is connected both to T and to S. In turn, it follows that both T and S are termatiko sets. If so, the algorithm returns the pair (T, S). Otherwise it returns FAIL.

We remark that by changing the randomized steps of Algorithm 2 into branching steps, one can get an exhaustive search algorithm that outputs all the splits (T, S) with the stated property (each $c \in N$ is connected to both T and S).

3.2. Column-regular measurement matrices

In this section, we present results for column a-regular measurement matrices, i.e. matrices with a non-zero entries in each column. The first result is a lower bound on the termatiko distance $h_{\min}$.

Theorem 40. *The termatiko distance of a column a-regular measurement matrix with no cycles of length 4 is at least a.*

Proof. Assume to the contrary that we have a termatiko set $T = \{v_1, v_2, \ldots, v_t\}$

Algorithm 2: SPLIT algorithm

Function $\mathrm{SPLIT}(D)$**:**

  Input: subset of variable nodes $D \subset V$

  Output: split of D into two termatiko sets, or FAIL

  $N \leftarrow \mathcal{N}(D)$ `/* initialisation */`

  $col_v \leftarrow ?$ **forall** $v \in V$

  $v \xleftarrow{\text{rnd}} D$

  $col_v \leftarrow$ GREEN

  $Q \leftarrow \{v\}$

  while $Q \neq \varnothing$ **do** `/* iterations */`

    $v \xleftarrow{\text{rnd pop}} Q$

    if $col_v =$ GREEN **then**

      $OppCol \leftarrow$ RED

    else

      $OppCol \leftarrow$ GREEN

    $Opp \leftarrow \{u \in D \,:\, col_u = ?$ and $\exists c \in \mathcal{N}(u) \cap \mathcal{N}(v)$ s.t. $\forall v' \in \mathcal{N}_D(c) \setminus \{u\}, col_{v'} = col_v\}$

    $col_u \leftarrow OppCol$ **forall** $u \in Opp$

    $Q \leftarrow Q \cup Opp$

    if $Q = \varnothing$ **and** $\{u \in D \,:\, col_u = ?\} \neq \varnothing$ **then**

      $v \xleftarrow{\text{rnd}} \{u \in D \,:\, col_u = ?\}$

      $col_v \xleftarrow{\text{rnd}} \{\text{GREEN}, \text{RED}\}$ `/* random guess */`

      $Q \leftarrow \{v\}$

  if $\exists c \in N$ s.t. $|\{col_v \,:\, v \in \mathcal{N}_D(c)\}| = 1$ **then** `/* correct? */`

    return FAIL

  else

    $T \leftarrow \{v \in D \,:\, col_v = \text{GREEN}\}$ `/* result */`

    $S \leftarrow \{v \in D \,:\, col_v = \text{RED}\}$

    return (T, S)

of size $t \leq a-1$. Define N and S as in Theorem 35.

First assume that $S \neq \varnothing$. Take any $u \in S$. Also split N into t non-intersecting subsets $N_1, \ldots, N_t$ such that $N = N_1 \cup N_2 \cup \cdots \cup N_t$, where

$$\begin{aligned}
N_1 &= \mathcal{N}(v_1)\,, \\
N_2 &= \mathcal{N}(v_2) \setminus N_1 = \mathcal{N}(v_2) \setminus \mathcal{N}(v_1)\,, \\
N_3 &= \mathcal{N}(v_3) \setminus N_2 = \mathcal{N}(v_3) \setminus (\mathcal{N}(v_2) \cup \mathcal{N}(v_1))\,, \\
&\ldots \\
N_t &= \mathcal{N}(v_t) \setminus N_{t-1} = \mathcal{N}(v_t) \setminus (\mathcal{N}(v_{t-1}) \cup \mathcal{N}(v_{t-2}) \cup \cdots \cup \mathcal{N}(v_1))\,.
\end{aligned}$$

As the measurement matrix has no cycles of length 4, no two variable nodes can share more than one measurement node. In particular, u cannot share more than one measurement node with any of $v_1, v_2, \ldots, v_t$. Therefore, u is connected not more than once to each of the sets $N_1, N_2, \ldots, N_t$, and thus $|\mathcal{N}(u)| \leq t \leq a-1$, which contradicts the fact that the degree of each variable node is a. It follows that $S = \varnothing$.

Since $S = \varnothing$, each measurement node in N should be connected to T at least twice. Furthermore, since the degree of each variable node is a, we have $|N| \leq \frac{at}{2}$. On the other hand, by definition, $|\mathcal{N}(v_j)| = a$ and $\mathcal{N}(v_j)$ shares not more than one element with each of $\mathcal{N}(v_{j-1}), \mathcal{N}(v_{j-2}), \ldots, \mathcal{N}(v_1)$. Therefore,

$$|N_j| = \Big| \mathcal{N}(v_j) \setminus (\mathcal{N}(v_{j-1}) \cup \mathcal{N}(v_{j-2}) \cup \cdots \cup \mathcal{N}(v_1)) \Big| \geq a - j + 1\,,$$

and we obtain

$$|N| = \left| \bigcup_{j=1}^{t} N_j \right| \geq at - \frac{t(t-1)}{2}\,.$$

It follows that

$$at - \frac{t(t-1)}{2} \leq |N| \leq \frac{at}{2}\,,$$

from which we get that $t \geq a+1$. This is a contradiction, since we have assumed $t \leq a-1$. □

As each stopping set is a termatiko set and each codeword support is a stopping set, we have that $h_{\min} \leq s_{\min} \leq d_{\min}$. Hence, the following result can be seen a corollary of Theorem 40.

Corollary 41. *For a column a-regular parity-check matrix, $d_{\min} \geq s_{\min} \geq a$.*

3.2.1. Measurement matrices from array low-density parity-check codes

A particular case of column a-regular measurement matrices are the parity-check matrices of array LDPC codes [14]. For a prime $q > 2$ and an integer $a < q$ the array LDPC code $\mathcal{C}(q,a)$ has length q^2 and can be defined by the parity-check matrix

$$H(q,a) = \begin{pmatrix} I & I & I & \cdots & I \\ I & P & P^2 & \cdots & P^{q-1} \\ I & P^2 & P^4 & \cdots & P^{2(q-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I & P^{a-1} & P^{2(a-1)} & \cdots & P^{(a-1)(q-1)} \end{pmatrix},$$

where I is the $q \times q$ identity matrix and P is a $q \times q$ permutation matrix defined by[3]

$$P = \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}.$$

It is easy to see that $\mathcal{C}(q,a)$ is an (a,q)-regular code of dimension $q^2 - qa + a - 1$. Its minimum distance will be denoted by $d(q,a)$ and stopping distance by $h(q,a)$.

In [56], a new representation of $H(q,a)$ was introduced. In particular, since each column of the parity-check matrix $H(q,a)$ has a blocks and each block is a permutation of $(1,0,0,\ldots,0)^\mathsf{T}$, we can represent each column as a length-a column vector of elements from $\mathbb{F}_q$, the field of integers modulo q. More precisely, $i \in \mathbb{F}_q$ is bijectively mapped to a vector

$$\left(\overbrace{0,\ldots,0}^{i},1,\overbrace{0,\ldots,0}^{q-i-1}\right)^\mathsf{T},$$

and any column in $H(q,a)$ is of the form

$$(i, i+j, i+2j, \ldots, i+(a-1)j)^\mathsf{T} \pmod q \tag{3.1}$$

for some $i,j \in \mathbb{F}_q$. Note that in (3.1) the field elements i and j are considered as integers and the operations (addition and multiplication) are standard integer operations, while $(\bmod\ q)$ denotes integer reduction modulo q. In the following, with some abuse of notation, a field element from $\mathbb{F}_q$ and its integer representation in the range $\{0,1,\ldots,q-1\}$ are used interchangeably. Furthermore, addition, subtraction, and multiplication might be either standard integer addition, subtraction, and multiplication, or denote field operations. However, this will be clear from the context. Also, note that since there are q^2 distinct columns in $H(q,a)$, any pair $(i,j) \in \mathbb{F}_q^2$ specifies a valid column. Therefore, the columns of $H(q,a)$ (or variable nodes V) can be identified with pairs $(i,j) \in \mathbb{F}_q^2$.

[3]Note that here P shifts the elements in the direction opposite to that defined in Section 1.2.3. However, the results are equivalent up to column reordering.

Further, as rows of the matrix can be split into a blocks of q rows each, it is convenient to identify rows of $H(q, a)$ (or measurement nodes C) with pairs in $\mathbb{Z}_a \times \mathbb{F}_q$, so that the j-th row ($1 \leq j \leq aq$) is identified (or indexed) by[4]

$$\langle \lfloor (j-1)/q \rfloor, (j-1) \pmod{q} \rangle \ .$$

In other words, row 1 is indexed by $\langle 0, 0 \rangle$, row 2 by $\langle 0, 1 \rangle$, up to row q which is indexed by $\langle 0, q-1 \rangle$, row $q+1$ by $\langle 1, 0 \rangle$, and so on. With this notation, variable node $(i, j) \in V = \mathbb{F}_q^2$ is connected to measurement nodes $\{\langle 0, i \rangle, \langle 1, i+j \rangle, \langle 2, i+2j \rangle, \ldots, \langle q-1, i+(q-1)j \rangle\} = \{\langle s, i+sj \rangle \mid s \in \mathbb{Z}_a\}$.

For $s \in \mathbb{Z}_a$, we call the q consecutive rows $\langle s, 0 \rangle$, $\langle s, 1 \rangle$, $\ldots$, $\langle s, q-1 \rangle$ (or corresponding measurement nodes) the *s-th strip*. We will extensively use the fact that every variable node has exactly one neighbouring measurement node in each of the strips.

Define the permutations $\varphi : \mathbb{F}_q^2 \mapsto \mathbb{F}_q^2$ and $\psi : \mathbb{Z}_a \times \mathbb{F}_q \mapsto \mathbb{Z}_a \times \mathbb{F}_q$, with parameters $\alpha \in \mathbb{F}_q \setminus \{0\}$, $\beta_1, \beta_2 \in \mathbb{F}_q$, by[5]

$$\begin{aligned} \varphi(i, j) &= (\alpha i + \beta_1, \alpha j + \beta_2) \, , \\ \psi(s, t) &= \langle s, \alpha t + (\beta_1 + s\beta_2) \rangle \ . \end{aligned}$$

It is well-known (cf. [56, Lem. 2]) that $\mathcal{C}(q, a)$ is invariant under the doubly transitive group of "affine" permutations defined above. In other words, such a pair of transformations is an automorphism on the Tanner graph of an array LDPC code, i.e. $\langle s, t \rangle \in \mathcal{N}((i, j))$ if and only if $\psi(s, t) \in \mathcal{N}(\varphi(i, j))$ for all choices of α, β_1, β_2. In particular, $T = \{v_1, v_2, \ldots, v_{|T|}\}$ is a termatiko set if and only if $\{\varphi(v_1), \varphi(v_2), \ldots, \varphi(v_{|T|})\}$ is a termatiko set. The number of choices for α, β_1, β_2 is $q^2(q-1)$ and this is the number of different automorphisms of this particular type, one of them being the identity (when $\alpha = 1$, $\beta_1 = \beta_2 = 0$). Furthermore, it is also well-known that there are no cycles of length 4 in Tanner graph corresponding to the parity-check matrix of an array LDPC code [14].

In the following, the *support matrix* of a subset of variable nodes, $U \subset V$, will be the submatrix of $H(q, a)$ consisting of the columns indexed by U. Furthermore, the *support matrix of a codeword* is the support matrix of the support of the codeword. We will mostly write the support matrix in a compact form using the representation in (3.1), i.e. as an $a \times |U|$ matrix over $\mathbb{F}_q$. For example, the support matrix of the subset $\{(i_1, j_1), (i_2, j_2), (i_3, j_3)\} \subset V$ of three variable nodes

[4] $\mathbb{Z}_a$ denotes the ring of integers modulo a, and we use angular brackets for measurement nodes to clearly differentiate between C and V.

[5] $\varphi(i, j)$ and $\psi(s, t)$ are shorthand notations for $\varphi((i, j))$ and $\psi(\langle s, t \rangle)$, respectively.

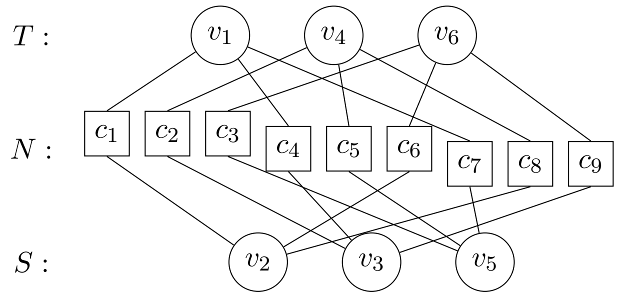


Figure 21. Termatiko set of size 3 in $H(q,3)$. Measurement nodes $c_1, c_2, \dots, c_9$ are grouped according to being in the first, the second, and the third strips in $H(q,3)$.

is written as[6]

$$\begin{bmatrix} i_1 & i_2 & i_3 \\ i_1+j_1 & i_2+j_2 & i_3+j_3 \\ i_1+2j_1 & i_2+2j_2 & i_3+2j_3 \\ \cdots & \cdots & \cdots \\ i_1+(a-1)j_1 & i_2+(a-1)j_2 & i_3+(a-1)j_3 \end{bmatrix}.$$

3.2.2. Termatiko distance multiplicity of $H(q,3)$

Consider the array LDPC code $\mathcal{C}(q,3)$. It is $(3,q)$-reqular and each column of its parity-check matrix $H(q,3)$ can be represented by the vector $(i, i+j, i+2j)^\intercal \in \mathbb{F}_q^3$, from which it follows that if $v \in V$ is connected to $c_1 = \langle 0, s_1 \rangle$, $c_2 = \langle 1, s_2 \rangle$, and $c_3 = \langle 2, s_3 \rangle$, then $2s_2 = s_1 + s_3$ (i.e. s_1, s_2, s_3 form an arithmetic progression).

Theorem 42. *There are $q^2(q-1)(q-2)/3$ termatiko sets of minimum size 3 in $H(q,3)$ for any $q \geq 5$ and their support matrices have (up to automorphisms) one of the forms*

$$\begin{bmatrix} 0 & 2 & -2-2j \\ 0 & 2+j & 1 \\ 0 & 2+2j & 4+2j \end{bmatrix} \quad or \quad \begin{bmatrix} 0 & 2 & 4+2j \\ 0 & 2+j & 1+j \\ 0 & 2+2j & -2 \end{bmatrix},$$

for any $j \in \mathbb{F}_q \setminus \{q-1, q-2\}$.

Proof. See Appendix C. □

We remark that this formula is similar to the formula for the number of weight-6 codewords in $\mathcal{C}(q,3)$ provided in [29, Thm. 2]. In fact, the number of termatiko sets of size 3 is twice the number of codewords of weight 6. Fig. 21 provides an illustration of a termatiko set of size 3 in $H(q,3)$.

[6] Recall that we associate V with $\mathbb{F}_q^2$.

3.2.3. Upper bounds on the termatiko distance of $H(q, a)$

It follows from Theorem 40 that the termatiko distance of $H(q, a)$ is $h_{\min} \geq a$, and from Theorem 42 it follows that the bound is indeed tight for $a = 3$. In this subsection, we derive upper bounds on the termatiko distance when $4 \leq a \leq 7$. The approach is inspired by the following observation.

It was shown in [36] that $d(q, 3) = 6$, and in [56] the authors derived the explicit support matrix

$$\begin{bmatrix} \mathbf{0} & 0 & 2i-2j & \boldsymbol{2i-2j} & -2i & \boldsymbol{-2i} \\ \mathbf{0} & -2i+j & 0 & \boldsymbol{-i} & -i & \boldsymbol{-2i+j} \\ \mathbf{0} & -4i+2j & -2i+2j & \boldsymbol{-4i+2j} & 0 & \boldsymbol{-2i+2j} \end{bmatrix}$$

(up to equivalence under the aforementioned automorphisms) for codewords of weight 6, where $i \in \mathbb{F}_q \setminus \{0\}$ and $j \in \mathbb{F}_q$ with $j \neq i, 2i$. It is worth noting that the columns 1, 4, and 6 (marked in bold) of the support matrix above form a termatiko set. The same is true for the columns 2, 3, and 5. Hence, the support of each minimum-weight codeword in $H(q, 3)$ can be split into two size-3 termatiko sets.

Deriving upper bounds on the minimum distance of array LDPC codes has attracted some attention, and tight bounds have been derived for $4 \leq a \leq 7$ in [48, 41]. In these works, explicit support matrices of codewords have been tabulated. A further exploration of these support matrices shows that a half-and-half split into two termatiko sets is possible; the connected measurement nodes are connected to both termatiko sets. We can now successfully apply Algorithm 2 to some known cases.

Matrix $H(q, 3)$. By applying Algorithm 2 to the aforementioned support matrix we obtain the (correct) split

$$\left[\begin{array}{ccc|ccc} 0 & 2i-2j & -2i & 0 & 2i-2j & -2i \\ 0 & -i & j-2i & j-2i & 0 & -i \\ 0 & 2j-4i & 2j-2i & 2j-4i & 2j-2i & 0 \end{array}\right], \tag{3.2}$$

where the vertical line indicates the actual split. Note that the columns are reordered so that both the first three and the last three form termatiko sets.

If we set $i = -1$, then we obtain the first general form in Theorem 42 (with columns reordered) in the left part of (3.2). To get the second termatiko set in (3.2), we also set $i = -1$ but then also apply an automorphism with $\alpha = 1$, $\beta_1 = 0$, $\beta_2 = -2 - j$, and substitute $j \mapsto -3 - j$. The resulting support matrix is of the second general form from Theorem 42 (with columns reordered).

Matrix $H(q, 4)$. In [48, Fig. 3], the authors present the support matrix of a weight-10 codeword for $H(q, 4)$ for $q > 7$. Since $\alpha = 12$ is co-prime with any prime $q > 4$, each entry in the matrix in [48] can be multiplied by $\alpha = 12$, which is equivalent to applying a doubly transitive automorphism. The resulting matrix becomes

$$\begin{bmatrix} 0 & 0 & -12 & -24 & -6 & -6 & -24 & -12 & -30 & -30 \\ 0 & 3 & 0 & -12 & -4 & 3 & -13 & -4 & -13 & -12 \\ 0 & 6 & 12 & 0 & -2 & 12 & -2 & 4 & 4 & 6 \\ 0 & 9 & 24 & 12 & 0 & 21 & 9 & 12 & 21 & 24 \end{bmatrix}.$$

Application of Algorithm 2 gives the split indicated in Table 7 where the columns have been re-ordered. For $q = 11$, we exhaustively checked all 4-subsets of $\mathbb{F}_q^2$ and did not find any termatiko sets among them, therefore $h(11, 4) = 5$. For the special cases $H(5, 4)$ and $H(7, 4)$, weight-8 codeword support matrices were presented in [56, Thm. 7 and 8]. These can be split too, and the results of the splits are shown in Table 7.

Matrix $H(q, 5)$. In [48, Fig. 4], an explicit support matrix of weight-12 codewords from $H(q, 5)$ is presented[7] for $q \neq 11$. Multiplying each entry of the matrix by $\alpha = 6$, which is co-prime with $q > 5$, and applying Algorithm 2 to the resulting matrix results in a half-and-half split (see Table 7). For $q = 7$, we verified exhaustively that the bound is tight, i.e. $h(7, 5) = 6$. Furthermore, for $q = 11$, there exists a weight-10 codeword and the result of its split is shown in Table 7.

Matrix $H(q, 6)$. In [41, (13)], the authors presented a support matrix of codewords of weight 20 for $H(q, 6)$. We multiply its entries by $\alpha = 2$ and apply Algorithm 2 to the resulting matrix. The algorithm succeeds to create a half-and-half split and the result is presented in Table 8. The authors prove in [41] that there are no repetitive columns in the matrix for $q > 11$. For the special cases $H(7, 6)$ and $H(11, 6)$, they provide particular support matrices which we are also able to split half-and-half with Algorithm 2 (see Table 8).

Matrix $H(q, 7)$. In [41, (17)], the authors present a support matrix for codewords of weight 24 for $H(q, 7)$. We multiply its entries by $\alpha = 4$ and successfully split it using Algorithm 2 (see Table 8).

Matrix $H(q, a > 7)$. From the previous subsections it appears that the termatiko distance is a half of the minimum distance for array LDPC codes. However, proving this in general might be difficult as not all codewords can be split half-and-half. For instance, for $q = 7$ and $a = 4$ we have found a (minimal) codeword of weight 20 that cannot be split into two termatiko sets each of size 10 (proved by exhaustive search) . The support matrix of the codeword is

$$\begin{bmatrix} 2 & 3 & 4 & 1 & 2 & 3 & 5 & 6 & 0 & 1 & 2 & 5 & 6 & 5 & 4 & 5 & 5 & 0 & 2 & 5 \\ 2 & 3 & 4 & 2 & 3 & 4 & 6 & 0 & 2 & 3 & 4 & 0 & 1 & 1 & 1 & 2 & 3 & 6 & 1 & 4 \\ 2 & 3 & 4 & 3 & 4 & 5 & 0 & 1 & 4 & 5 & 6 & 2 & 3 & 4 & 5 & 6 & 1 & 5 & 0 & 3 \\ 2 & 3 & 4 & 4 & 5 & 6 & 1 & 2 & 6 & 0 & 1 & 4 & 5 & 0 & 2 & 3 & 6 & 4 & 6 & 2 \end{bmatrix}.$$

We gather the results for the termatiko distances of array LDPC codes in Table 9. We additionally put results for measurement matrices $H(5, 5)$ and $H(7, 7)$, although usually $a < q$ is required for array LDPC codes.[8] The exact termatiko distances for these two cases are obtained by splitting small-size stopping sets

[7]It seems the authors did not verify that the columns of the support matrix are different. However, for $q = 11$, two columns are identical. Therefore, we treat $H(11, 5)$ as a special case.

[8]Having $a = q$ still gives array LDPC codes of strictly positive rate since $H(q, a)$ has redundant rows.

using Algorithm 2. This procedure produces termatiko sets of size 5 and 7, respectively. From Theorem 40 it follows that these values give the exact termatiko distance in these two cases. Alternatively, for $a = 5$, one can remove the 5-th and the last column from the matrix in Table 7 (they are identical for $q = 5$) and obtain a valid codeword support matrix of a weight-10 codeword that is also splittable in two termatiko sets of size 5.

3.2.4. Decreasing termatiko distance by adjoining redundant rows to a measurement matrix

As it was discussed in Chapter 2, for BP decoding over the BEC one can add redundant rows to a parity-check matrix in order to decrease the number of stopping sets [46]. This is also the case for relaxed linear programming decoding of binary linear codes on any symmetric channel [15]. In this section, we aim to improve the recovery performance of the IPA by adding redundant rows to a measurement matrix, inspired by success on the BEC. However, there is one fundamental difference in the sense that the real linear combinations that are added to the measurement matrix should contain non-negative entries only. Furthermore, we would like to stress that redundant rows that we add to the measurement matrix are not used to provide new measurements, but rather used in the recovery process, which means that also measurements need to be linearly combined at the receiver. Thus, this procedure does not make the compression rate of the scheme worse, but rather potentially improve the recovery performance.

The following lemma shows that adding redundant rows to the measurement matrix does not harm IPA reconstruction performance, namely that it does not create new termatiko sets.

Lemma 43. *Adding redundant measurements does not create new termatiko sets.*

Proof. Let the original measurement matrix be denoted by A. Its extended version with non-negative redundant rows is denoted by A'. The matrix A' is constructed such that the first rows of A' are exactly the rows of A and the remaining rows are real-valued linear combinations of the rows of A with non-negative entries.[9]

Denote also the Tanner graph corresponding to A' by $(V' \cup C', E')$, and let $\mathcal{N}'$, $\mathcal{N}'_T$ be the notation for neighbours in this Tanner graph (analogously to (1.1)). Consider some signal vector $\boldsymbol{x}$ and two problems, IPA$(\boldsymbol{y}, A)$ and IPA$(\boldsymbol{y}', A)$, where $\boldsymbol{y} = \boldsymbol{x}\boldsymbol{A}^{\mathsf{T}}$ and $\boldsymbol{y}' = \boldsymbol{x}A'^{\mathsf{T}}$.

The set of variable nodes is the same, i.e. $V = V'$, but the set of measurement nodes is now a superset of the original set, i.e. $C \subset C'$. The same is true for the set of edges, $E \subset E'$. It holds for all $v \in V$ that $\mathcal{N}'(v) = \mathcal{N}'_C(v) \cup \mathcal{N}'_{C' \setminus C}(v) = \mathcal{N}(v) \cup \mathcal{N}'_{C' \setminus C}(v)$. For all $c \in C$, we have $\mathcal{N}'(c) = \mathcal{N}(c)$. This in turn means that $y_c = y'_c$ for $c \in C$.

[9] Non-negativity of matrix entries is important for the correctness of the IPA.

Table 7. Codeword support matrices split into termatiko sets. Vertical lines illustrate how to split the codewords into pairs of distinct termatiko sets each of half the size.

Matrix	Codeword weight	Codeword support matrix split
$H(q,4)$ $q \geq 11$	10	$\left[\begin{array}{ccccc\|ccccc} 0 & -6 & -24 & -12 & -30 & 0 & -12 & -24 & -6 & -30 \\ 0 & 3 & -13 & -4 & -12 & 3 & 0 & -12 & -4 & -13 \\ 0 & 12 & -2 & 4 & 6 & 6 & 12 & 0 & -2 & 4 \\ 0 & 21 & 9 & 12 & 24 & 9 & 24 & 12 & 0 & 21 \end{array}\right]$
$H(5,4)$ $z \in \mathbb{F}_5 \setminus \{0\}$ $k \in \{0, 2z\}$	8	$\left[\begin{array}{cccc\|cccc} 0 & 3k+3z & 2k+4z & 2z & 0 & 3k+3z & 2k+4z & 2z \\ 0 & 3z & k+4z & k+2z & k+4z & 0 & k+2z & 3z \\ 0 & 2k+3z & 4z & 2k+2z & 2k+3z & 2k+2z & 0 & 4z \\ 0 & 4k+3z & 4k+4z & 3k+2z & 3k+2z & 4k+4z & 4k+3z & 0 \end{array}\right]$
$H(7,4)$ $z \in \mathbb{F}_7 \setminus \{0\}$ $k \in \{0, 2z, 4z, 6z\}$	8	$\left[\begin{array}{cccc\|cccc} 0 & 2k+5z & 2k+z & 4z & 0 & 2k+5z & 2k+z & 4z \\ 0 & k+2z & 5z & k+4z & k+2z & 0 & k+4z & 5z \\ 0 & 6z & 5k+2z & 2k+4z & 2k+4z & 5k+2z & 0 & 6z \\ 0 & 6k+3z & 3k+6z & 3k+4z & 3k+6z & 3k+4z & 6k+3z & 0 \end{array}\right]$
$H(q,5)$ $q \neq 11$	12	$\left[\begin{array}{cccccc\|cccccc} 0 & -4 & -18 & -22 & -6 & -16 & 0 & -6 & -22 & -18 & -4 & -16 \\ 0 & 1 & -8 & -12 & -3 & -11 & 1 & 0 & -11 & -12 & -3 & -8 \\ 0 & 6 & 2 & -2 & 0 & -6 & 2 & 6 & 0 & -6 & -2 & 0 \\ 0 & 11 & 12 & 8 & 3 & -1 & 3 & 12 & 11 & 0 & -1 & 8 \\ 0 & 16 & 22 & 18 & 6 & 4 & 4 & 18 & 22 & 6 & 0 & 16 \end{array}\right]$
$H(11,5)$	10	$\left[\begin{array}{ccccc\|ccccc} 0 & 5 & 4 & 7 & 6 & 7 & 4 & 0 & 5 & 6 \\ 1 & 0 & 10 & 8 & 3 & 1 & 3 & 10 & 8 & 0 \\ 2 & 6 & 5 & 9 & 0 & 6 & 2 & 9 & 0 & 5 \\ 3 & 1 & 0 & 10 & 8 & 0 & 1 & 8 & 3 & 10 \\ 4 & 7 & 6 & 0 & 5 & 5 & 0 & 7 & 6 & 4 \end{array}\right]$

Table 8. Codeword support matrices split into termatiko sets (continued). Vertical lines illustrate how to split the codewords into pairs of distinct termatiko sets each of half the size.

Matrix	Codeword weight	Codeword support matrix split
$H(7,6)$	12	$\left[\begin{array}{cccccc\|cccccc} 0&3&6&2&5&4&2&6&5&4&0&3\\ 0&6&5&4&3&1&3&0&6&5&1&4\\ 0&2&4&6&1&5&4&1&0&6&2&5\\ 0&5&3&1&6&2&5&2&1&0&3&6\\ 0&1&2&3&4&6&6&3&2&1&4&0\\ 0&4&1&5&2&3&0&4&3&2&5&1 \end{array}\right]$
$H(11,6)$	16	$\left[\begin{array}{cccccccc\|cccccccc} 0&10&1&5&7&6&6&0&6&10&5&1&0&7&0&6\\ 0&4&7&10&2&6&9&8&7&0&8&9&10&6&2&4\\ 0&9&2&4&8&6&1&5&8&1&0&6&9&5&4&2\\ 0&3&8&9&3&6&4&2&9&2&3&3&8&4&6&0\\ 0&8&3&3&9&6&7&10&10&3&6&0&7&3&8&9\\ 0&2&9&8&4&6&10&7&0&4&9&8&6&2&10&7 \end{array}\right]$
$H(q,6)$ $q > 11$	20	$\left[\begin{array}{cccccccccc\|cccccccccc} 0&-22&-2&-20&10&-8&12&-10&-32&22&-10&-2&10&-32&22&-20&0&-8&-22&12\\ 0&-16&8&-8&9&-7&17&1&-15&16&-8&0&16&-16&17&-15&9&1&-7&8\\ 0&-10&18&4&8&-6&22&12&2&10&-6&2&22&0&12&-10&18&10&8&4\\ 0&-4&28&16&7&-5&27&23&19&4&-4&4&28&16&7&-5&27&19&23&0\\ 0&2&38&28&6&-4&32&34&36&-2&-2&6&34&32&2&0&36&28&38&-4\\ 0&8&48&40&5&-3&37&45&53&-8&0&8&40&48&-3&5&45&37&53&-8 \end{array}\right]$
$H(q,7)$	24	$\left[\begin{array}{cccccccccccc\|cccccccccccc} 0&-18&-14&-20&-8&-4&8&2&6&-12&10&-22&6&0&-4&-22&8&-20&10&-12&-8&-18&2&-14\\ 0&-14&-10&-12&-7&1&6&4&8&-6&9&-15&6&4&0&-14&9&-15&8&-10&-6&-12&1&-7\\ 0&-10&-6&-4&-6&6&4&6&10&0&8&-8&6&8&4&-6&10&-10&6&-8&-4&-6&0&0\\ 0&-6&-2&4&-5&11&2&8&12&6&7&-1&6&12&8&2&11&-5&4&-6&-2&0&-1&7\\ 0&-2&2&12&-4&16&0&10&14&12&6&6&6&16&12&10&12&0&2&-4&0&6&-2&14\\ 0&2&6&20&-3&21&-2&12&16&18&5&13&6&20&16&18&13&5&0&-2&2&12&-3&21\\ 0&6&10&28&-2&26&-4&14&18&24&4&20&6&24&20&26&14&10&-2&0&4&18&-4&28 \end{array}\right]$

Table 9. Termatiko distances of array LDPC code matrices $H(q,a)$.

	$a=3$	$a=4$	$a=5$	$a=6$	$a=7$
$q=5$	3	4	5	–	–
$q=7$	3	4	6	6	7
$q=11$	3	5	5	6..8	7..12
$q\geq 13$	3	4 or 5	5 or 6	6..10	7..12

Let μ' and M' (with corresponding indices) be bounds in the iterations of $\mathrm{IPA}(y', A')$. In order to prove the statement of the lemma, it is enough to show that for all iterations $\ell \geq 0$, $\mu'^{(\ell)}_{v\to\cdot} \geq \mu^{(\ell)}_{v\to\cdot}$ and $M'^{(\ell)}_{v\to\cdot} \leq M^{(\ell)}_{v\to\cdot}$. In other words, we show that the intervals $[\mu', M']$ are at least as tight as $[\mu, M]$. We show this by induction on ℓ.

Base Case. $\mu'^{(0)}_{v\to\cdot} = 0 = \mu^{(0)}_{v\to\cdot}$ and

$$M'^{(0)}_{v\to\cdot} = \min_{c\in\mathcal{N}'(v)} (y'_c/a'_{cv}) \leq \min_{c\in\mathcal{N}(v)} (y'_c/a'_{cv}) = \min_{c\in\mathcal{N}(v)} (y_c/a_{cv}) = M^{(0)}_{v\to\cdot}\,.$$

Inductive Step. Consider iteration $\ell \geq 1$. At each step ℓ of the IPA and for all $c \in C$ and $v \in \mathcal{N}'(c) = \mathcal{N}(c)$, we have

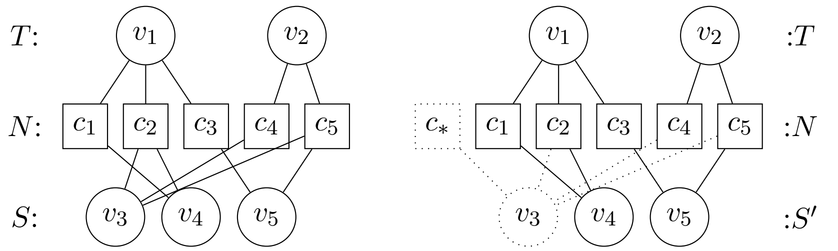
$$\begin{aligned}\mu'^{(\ell)}_{c\to v} &= \frac{1}{a'_{cv}}\left(y'_c - \sum_{\substack{v'\in\mathcal{N}'(c)\\ v'\neq v}} a'_{cv'} M'^{(\ell-1)}_{v'\to\cdot}\right) = \frac{1}{a_{cv}}\left(y_c - \sum_{\substack{v'\in\mathcal{N}(c)\\ v'\neq v}} a_{cv'} M'^{(\ell-1)}_{v'\to\cdot}\right)\\ &\geq \frac{1}{a_{cv}}\left(y_c - \sum_{\substack{v'\in\mathcal{N}(c)\\ v'\neq v}} a_{cv'} M^{(\ell-1)}_{v'\to\cdot}\right) = \mu^{(\ell)}_{c\to v}\,.\end{aligned}$$

In the same manner, we have that for all $c \in C$, $M'^{(\ell)}_{c\to v} \leq M^{(\ell)}_{c\to v}$. We further apply these inequalities to Lines 14 and 15 in Algorithm 1 and, using properties of the functions $\min(\cdot)$ and $\max(\cdot)$, we obtain the desired result. $\square$

From Lemma 43 it follows that adding redundant rows to the measurement matrix cannot harm the IPA. The following example shows that adding such rows can indeed improve the performance of the IPA by removing termatiko sets.

Example 44. Consider the binary measurement matrix

$$A = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix} \end{array}.$$



(a) $T = \{v_1, v_2\}$, $S = \{v_3, v_4, v_5\}$, and $N = \{c_1, c_2, c_3, c_4, c_5\}$ are the sets defined in Theorem 35. T is a termatiko set.

(b) $S' = \{v_4, v_5\} \neq S$ because of the new measurement node c_*. T is not a termatiko set any more.

Figure 22. Adding a redundant measurement c_* corresponding to the difference of rows c_2 and c_1 of the matrix in Example 44.

The corresponding Tanner graph is shown in Fig. 22a. Note that the set $\{v_1, v_2\}$ is a termatiko set for this matrix. However, if we add a redundant row c_* equal to the difference of rows c_2 and c_1, $\{v_1, v_2\}$ is not a termatiko set for the extended matrix[10]

$$A' = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_* \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array},$$

since c_4 violates conditions in Theorem 35:

- c_4 is not connected to S', and
- $\mathcal{N}_T(c_4) = \{v_2\}$, $\mathcal{N}(v_2) = \{c_4, c_5\}$, and each of c_4, c_5 is connected to T only once; therefore

$$\left|\{v \in \mathcal{N}_T(c_4) \,:\, \forall c' \in \mathcal{N}(v), |\mathcal{N}_T(c')| \geq 2\}\right| = 0\,.$$

Fig. 22b illustrates the differences. △

The following question arises: which redundant rows do we need to add in order to remove the largest number of harmful small-size termatiko sets. We propose the following heuristic approach. First, fix some list of small-size termatiko sets for the original measurement matrix A and generate a pool of redundant rows which (hopefully) help to remove at least one termatiko set from the list as follows.

Consider a termatiko set T from the list and its corresponding set S. A redundant row $\boldsymbol{r} = (r_1, r_2, \ldots, r_n)$ for the measurement matrix A can be uniquely

[10] Recall that operations are performed over $\mathbb{R}$.

defined by coefficients $\alpha_1, \alpha_2, \ldots, \alpha_m \in \mathbb{R}$ in a linear combination

$$r_v = \sum_{c \in C} a_{cv} \alpha_c.$$

However, since in the real calculations floating-point numbers are effectively rational numbers, by multiplying all α's by some common multiplier of their denominators, we can make them all integer, and they still produce a redundant row $\boldsymbol{r}$ with the same support. Therefore, with no loss of generality, we assume that α's are integers. If original matrix A has integer entries, the resulting extended matrix has integer entries as well, which allows for a faster IPA in applications where the signal $\boldsymbol{x}$ is integer.

There are two types of redundant rows that will be collected in the pool. The first type "breaks" the termatiko set T for sure. It has one non-zero entry in the positions in T and zeroes in entries indexed by S. The other entries of $\boldsymbol{r}$ can be chosen arbitrarily. More precisely, for a fixed $v_0 \in T$ we solve the (integer) linear programming problem

$$\begin{aligned} \text{minimize} \quad & \sum_{v \in V \setminus \{T \cup S\}} r_v = \sum_{v \in V \setminus \{T \cup S\}} \sum_{c \in C} a_{cv} \alpha_c \\ \text{s.t.} \quad & r_v \begin{cases} \geq 0\,, & v \notin T \cup S\,, \\ = 0\,, & v \in T \cup S \setminus \{v_0\}\,, \\ \geq 1\,, & v = v_0\,, \end{cases} \end{aligned}$$

where $\alpha_1, \alpha_2, \ldots, \alpha_m$ are integer variables. Minimization here is not essential and is used to obtain smaller coefficients in a redundant row. In fact, for any feasible solution, the corresponding redundant row eliminates the termatiko set T. A redundant row can potentially be obtained for each $v_0 \in T$. As a final remark, relaxing the α's to be real numbers turns the program into a standard linear program that can be solved using the simplex method. However, as noted above, having integers (of moderate size) in the measurement matrix has some potential benefits. Thus, when the size of the program is not too large and can be solved using a standard solver in a reasonable time (which is the case in our examples), we keep the integer constraint on the α's.

Redundant rows of the second type do not necessarily "break" T, but they have good chances for doing that. The basic idea is to make variable nodes in S to not satisfy Theorem 35. Hence, they are not included in S for the extended matrix. Hopefully, this eliminates T as a termatiko set for the extended matrix. Note that having several non-zero entries in positions in S is better, since all of them disappear from S (and we do not add new ones to S). This have a greater chance of removing T. The corresponding (integer) linear program is

$$r_v \begin{cases} \geq 0\,, & v \notin T\,, \\ \leq 1000\,, & v \notin T\,, \\ = 0\,, & v \in T\,, \end{cases}$$

$$\sum_{v \in S} r_v \geq 10|S|,$$

where the constants 10 and 1000 are chosen rather arbitrarily; 10 is used in order to make non-zero entries in $\boldsymbol{r}_S$ more likely, and the upper bounds of 1000 make sure the entries in $\boldsymbol{r}$ are of limited size. Note that no objective function is specified, since any feasible solution will do. For each termatiko set T, this approach produces at most one redundant row.

Finally, after constructing the pool of redundant rows as described above, we start adjoining them to the matrix A one by one in a greedy manner as follows. Let the list of termatiko sets be denoted by LIST and the pool of redundant rows by POOL. For each row $\boldsymbol{r} \in \mathsf{POOL}$, we calculate the score

$$\mathrm{score}(\boldsymbol{r}) = \sum_{T \in \mathsf{RMV}(\mathsf{LIST}, \boldsymbol{r})} |T| \,,$$

where $\mathsf{RMV}(\mathsf{LIST}, \boldsymbol{r})$ is a subset of LIST consisting of termatiko sets that are not termatiko sets after adjoining row $\boldsymbol{r}$ to the current measurement matrix. The row $\boldsymbol{r}^*$ with the maximum score is adjoined to the measurement matrix, the termatiko sets in $\mathsf{RMV}(\mathsf{LIST}, \boldsymbol{r})$ are removed from LIST, and the scores are re-calculated for the updated LIST and measurement matrix. The procedure is continued until LIST is empty or all scores are zero (which means that no additional termatiko sets can be removed).

3.3. Numerical results

In this section, we present numerical results for different specific measurement matrices and for ensembles of measurement matrices, as well as simulation results of the IPA performance.

3.3.1. Termatiko distance estimates of specific matrices

For all considered matrices, by using the algorithm in [43, 44], we first find all stopping sets of size less than some threshold . Then, we exhaustively search for termatiko sets as subsets of these stopping sets, as it is explained in Section 3.1.5. The results are tabulated in Table 10 for five different measurement matrices, denoted by $A^{(1)}$, $A^{(2)}$, $A^{(3)}$, $A^{(4)}$, and $A^{(5)}$, respectively. Due to the heuristic nature of the approach, the estimated termatiko distance is a true upper bound on the actual termatiko distance, while the estimated multiplicities are true lower bounds on the actual multiplicities.

Measurement matrix $A^{(1)}$ is the 33×121 parity-check matrix $H(11, 3)$ of the array-based LDPC code $\mathcal{C}(11, 3)$ of column-weight 3 and row-weight 11 described in Section 3.2.1, $A^{(2)}$ is the parity-check matrix of the $[155, 64, 20]$ Tanner code from [49] (cf. Section 2.1.1), $A^{(3)}$ is taken from the IEEE802.16e standard [26] (it is the parity-check matrix of a rate-$3/4$, length-1824 LDPC code; using base model matrix A and the alternative construction, see [44, (1)]), $A^{(4)}$ is a 276×552 parity-check matrix of an irregular LDPC code, while $A^{(5)}$ is a 159×265 parity-check matrix of a $(3, 5)$-regular LDPC code built from arrays of permutation matrices from Latin squares.

For the matrix $A^{(1)}$, we have also compared the results with an exact enumeration of all termatiko sets of size at most 5. When considering all stopping sets of size at most 11, the heuristic approach finds the exact multiplicities for sizes 3 and 4, but it underestimates the number of termatiko sets of size 5 by about 7.5% (the missing ones are the subsets of the stopping sets of size 12 to 14). This indicates that higher order terms (for all tabulated matrices) are most likely strict lower bounds on the exact multiplicities.

As it can be seen from Table 10, for all matrices except for $A^{(3)}$, the estimated termatiko distance is about half the stopping distance. The smallest-size termatiko sets all correspond to termatiko sets with all measurement nodes in N connected to both T and S (cf. Theorem 35).

3.3.2. Termatiko distance estimates of protograph-based matrix ensembles

Consider the protograph-based $(3, 6)$-regular LDPC code ensemble defined by the *protomatrix* $H = (3, 3)$. We randomly generate 200 parity-check matrices from this ensemble using a lifting factor of 100 (the two non-zero entries in the protomatrix are replaced by 100×100 binary matrices of row weight 3 in which all right-shifts of the first row—picked at random—occur in some order).

For each lifted matrix, we first find all stopping sets of size at most 16 by using the algorithm in [43, 44]. Then, the termatiko distance is estimated for each matrix as explained above. The results are depicted in Fig. 23 as a function of the code index (the blue curve shows the minimum distance $d_{\min}$, the red curve shows the minimum size of a non-codeword stopping set, denoted by $\tilde{s}_{\min}$, while the green curve shows the estimated termatiko distance $\hat{h}_{\min}$). The average $d_{\min}$, $s_{\min}$, and $\hat{h}_{\min}$ (over the 200 matrices) are 6.84, 5.92, and 3.90, respectively.[11] We repeat a similar experiment using a lifting factor of 200, and average $d_{\min}$, $s_{\min}$, and $\hat{h}_{\min}$ (again over 200 randomly generated matrices) become 9.21, 7.75, and 5.80, respectively.

Next, we repeat the same calculations for 200 randomly generated parity-check matrices from the protograph-based $(4, 8)$-regular LDPC code ensemble. For each parity-check matrix, we consider all stopping sets of size up to 14. For some matrices, the minimum distances of the corresponding codes are larger than 14, thus we calculate them separately. Fig. 24 presents the results of the calculations. The average $d_{\min}$, $s_{\min}$, and $\hat{h}_{\min}$ are 12.53, 9.75, and 8.41, respectively.

[11] Note that here the second average value is of stopping distances, and not the sizes of the smallest non-codeword stopping sets.

Table 10. Estimated termatiko set size spectra (initial part) of the measurement matrices from Section 3.3, where $\hat{h}_{\min}$ denotes the estimated termatiko distance. $\mathfrak{T}_1$ corresponds to termatiko sets with all measurement nodes in N connected to both T and S, and $\mathfrak{T}_2$ corresponds to all the remaining termatiko sets. Also shown are the exact stopping distances and stopping set size spectra (initial part). Entries in bold are exact values. For $A^{(1)}$, the heuristic approach gives a multiplicity of $5\,875\,518$ for size 5, while the exact number is $6\,318\,378$ (an underestimation of about 7.5%).

Measurement matrix	$\hat{h}_{\min}$	Initial estimated termatiko set size spectrum	$s_{\min}$	Initial stopping set size spectrum
$A^{(1)}$	$\mathbf{3}$	$\mathfrak{T}_1$: $(\mathbf{3630}, \mathbf{93775}, \mathbf{6318378}, 48548225, 71709440, 36514170, 7969060, 856801, 41745)$ $\mathfrak{T}_2$: $(0, 0, 0, 410190, 18610405, 71153445, 86844725, 58849681, 28430160)$	$\mathbf{6}$	$(\mathbf{1815}, \mathbf{605}, \mathbf{45375}, \mathbf{131890}, \mathbf{3550382}, \mathbf{28471905})$
$A^{(2)}$	9	$\mathfrak{T}_1$: $(465, 3906, 12555, 8835, 0, 0, \ldots)$ $\mathfrak{T}_2$: $(0, 0, 0, 1860, 5115, 10695, 2325, 5580, 2325, 6045$ $10850, 22103, 39990, 106175)$	$\mathbf{18}$	$(\mathbf{465}, \mathbf{2015}, \mathbf{9548}, \mathbf{23715}, \mathbf{106175})$
$A^{(3)}$	8	$\mathfrak{T}_1$: $(228, 0, 0, \ldots)$ $\mathfrak{T}_2$: $(0, 76, 0, 76, 684, 532, 152, 532, 1520)$	$\mathbf{9}$	$(\mathbf{76}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{76}, \mathbf{76}, \mathbf{304}, \mathbf{1520})$
$A^{(4)}$	8	$\mathfrak{T}_1$: $(184, 598, 1242, 391, 0, 0)$ $\mathfrak{T}_2$: $(0, 0, 0, 69, 23, 0, 23, 46, 161, 391, 1012, 2300, 5796)$	$\mathbf{15}$	$(\mathbf{46}, \mathbf{161}, \mathbf{391}, \mathbf{897}, \mathbf{2093}, \mathbf{5796})$
$A^{(5)}$	7	$\mathfrak{T}_1$: $(106, 0, 0, 53, 901, 3233, 954, 53, 0, 0, \ldots)$ $\mathfrak{T}_2$: $(0, 0, 0, 0, 0, 0, 106, 265, 106, 636, 689, 477, 583, 371, 1325, 2915, 5830, 9964)$	$\mathbf{14}$	$(\mathbf{53}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{53}, \mathbf{106}, \mathbf{583}, \mathbf{1484}, \mathbf{3922}, \mathbf{9964})$

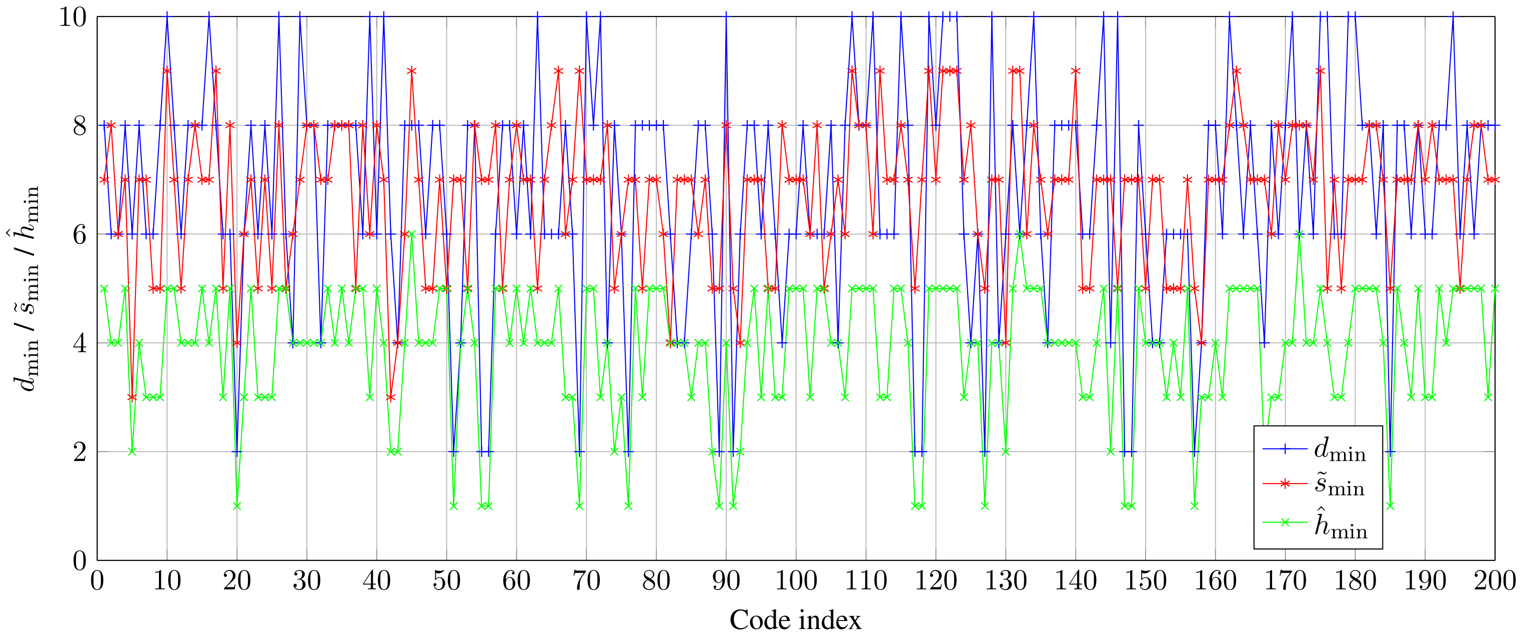


Figure 23. Minimum distance $d_{\min}$, minimum size of a non-codeword stopping set $\tilde{s}_{\min}$, and estimated termatiko distance $\hat{h}_{\min}$ versus code index for randomly generated binary measurement matrices from a protograph-based $(3, 6)$-regular LDPC code ensemble.

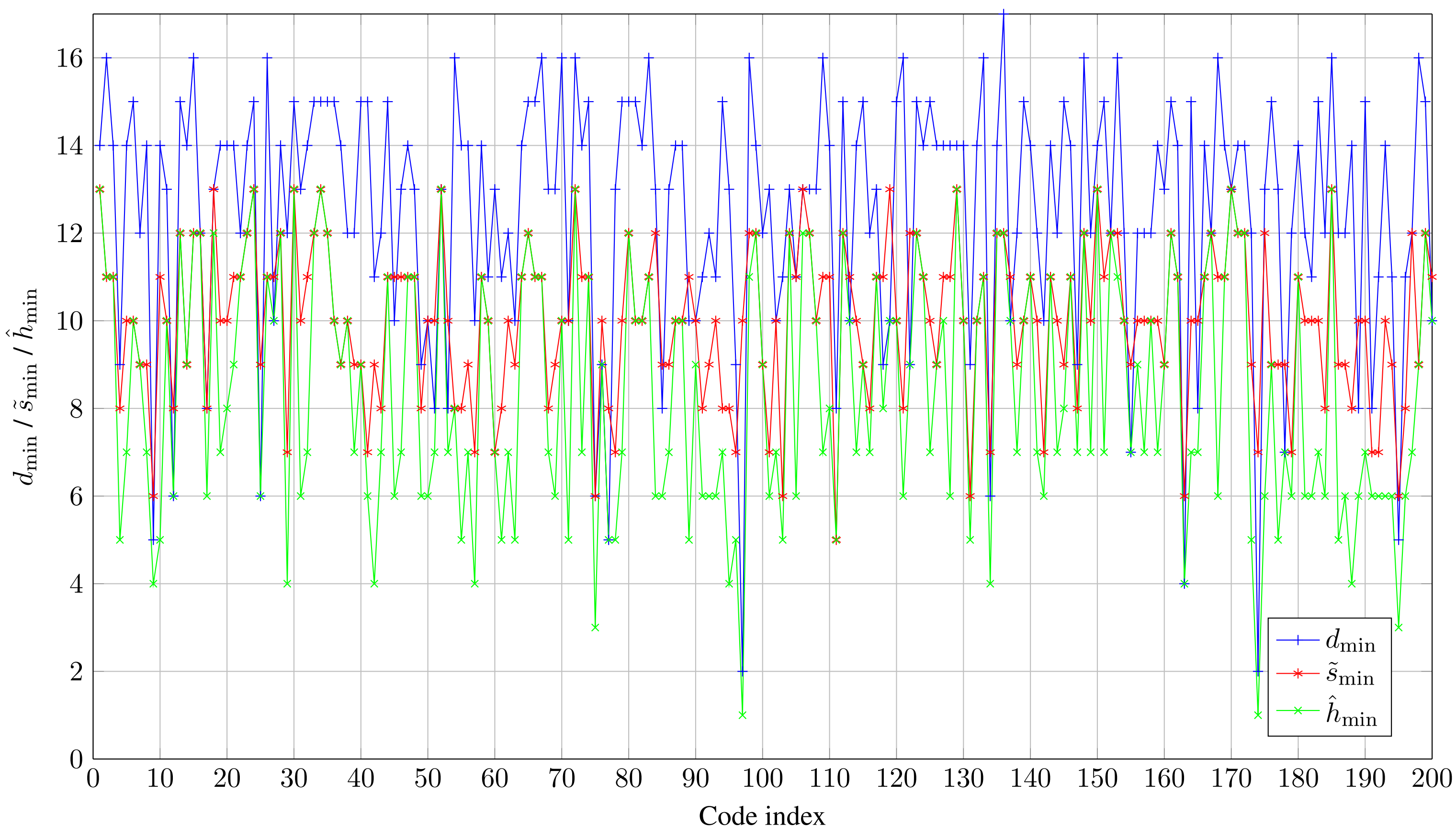


Figure 24. Minimum distance $d_{\min}$, minimum size of a non-codeword stopping set $\tilde{s}_{\min}$, and estimated termatiko distance $\hat{h}_{\min}$ versus code index for randomly generated binary measurement matrices from a protograph-based $(4, 8)$-regular LDPC code ensemble.

3.3.3. Performance of SPLIT algorithm

In order to see how Algorithm 2 performs, we apply it to the stopping sets of size at most 14 for the protograph-based matrices described in Section 3.3.2 (both $(3,6)$ and $(4,8)$-regular).

Table 11 shows the average number of stopping sets of size w, $w = 1, 2, \ldots, 14$, for the 200 randomly generated $(3,6)$-regular matrices (the numbers are exact). It also presents the fraction of the matrices that have stopping sets of size w. In particular, all the 200 matrices have stopping sets of size $w = 13$ and $w = 14$. For a fixed w, we also consider the total multiset of all stopping sets from all the matrices together and calculate the fraction of them that are splittable in their corresponding matrix. The last column of Table 11 displays these numbers. Next, we build the total multiset of all splittable stopping sets from all the matrices together and repeatedly run Algorithm 2 to estimate the average success probability across the multiset. The resulting frequencies are depicted in Fig. 25. The aforementioned calculations are repeated for the 200 randomly generated $(4,8)$-regular matrices. The results are presented in Table 12 and Fig. 26.

3.3.4. Adding redundant rows

To illustrate the efficiency of the heuristic algorithm from Section 3.2.4 in removing small-size termatiko sets, we choose three out of the 200 $(3,6)$-regular matrices (with a lifting factor of 100) in Section 3.3.2 as example matrices. More precisely, the matrices with indices 20, 72, and 172, denoted by $A_{\mathsf{PG}}^{(20)}$, $A_{\mathsf{PG}}^{(72)}$, and $A_{\mathsf{PG}}^{(172)}$, respectively, are selected. These matrices are chosen to demonstrate different behaviour patterns.

For all three matrices, we apply the algorithm from Section 3.2.4 in order to remove termatiko sets by adding redundant rows. The algorithm adds 30 redundant rows to $A_{\mathsf{PG}}^{(20)}$, 55 rows to $A_{\mathsf{PG}}^{(72)}$, and 68 rows to $A_{\mathsf{PG}}^{(172)}$. Due to computational limitations, we are able to tackle only a limited number of termatiko sets. $A_{\mathsf{PG}}^{(20)}$ originally had the highest number of termatiko sets, and because of that we only process all termatiko sets of size up to 5 (including). For $A_{\mathsf{PG}}^{(72)}$, we process all termatiko sets of size up to 7, and for $A_{\mathsf{PG}}^{(172)}$, sizes up to 8. Accordingly, we occasionally denote the extended matrices by $A_{\mathsf{EPG}^{(5)}}^{(20)}$, $A_{\mathsf{EPG}^{(7)}}^{(72)}$, and $A_{\mathsf{EPG}^{(8)}}^{(172)}$. The numbers of termatiko sets decrease for all matrices, however, for $A_{\mathsf{PG}}^{(72)}$ and $A_{\mathsf{PG}}^{(172)}$ we are also able to increase their termatiko distances. Table 13 shows the estimated termatiko set size spectra (initial part) for the original and extended matrices.

In order to see how changes in the termatiko set size spectra influence performance under the IPA, we perform simulations to estimate the frame-error rate, i.e. the probability of failure to recover an original signal correctly for different values of its Hamming weight w. The results are presented in Fig. 28a. We remark that the performance of the IPA and its comparison with other algorithms for efficient

Table 11. Stopping sets (including codewords) distribution over 200 randomly generated matrices from the protograph-based $(3, 6)$-regular LDPC code ensemble. The numbers are exact.

w	average number of size-w stopping sets	fraction of codes having size-w stopping sets	fraction of size-w stopping sets allowing a (T, S)-split
1	0.000	0.000	-
2	0.080	0.075	1.000
3	0.010	0.010	0.000
4	0.150	0.125	0.267
5	0.320	0.215	0.094
6	1.350	0.485	0.222
7	5.365	0.690	0.070
8	10.860	0.925	0.174
9	33.695	0.995	0.083
10	105.935	1.000	0.099
11	298.085	1.000	0.079
12	953.220	1.000	0.082
13	3029.230	1.000	0.070
14	9887.395	1.000	0.076

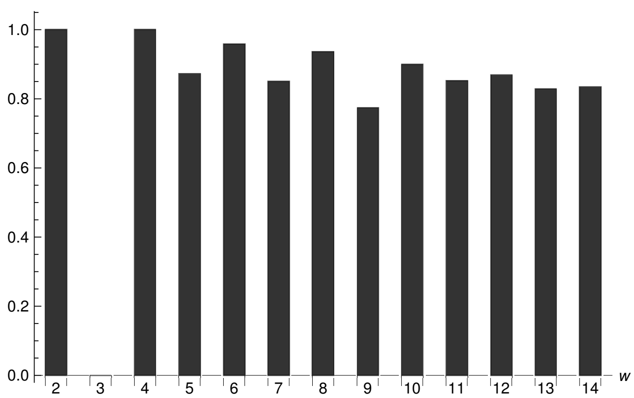


Figure 25. Average success rate of Algorithm 2 on stopping sets that allow a (T, S)-split for the 200 randomly generated matrices from the protograph-based $(3, 6)$-regular LDPC code ensemble. Note that there are no splittable stopping sets of size $w = 3$.

Table 12. Stopping sets (including codewords) distribution over 200 randomly generated matrices from the protograph-based $(4, 8)$-regular LDPC code ensemble. The numbers are exact.

w	average number of size-w stopping sets	fraction of codes having size-w stopping sets	fraction of size-w stopping sets allowing a (T, S)-split
1	0.000	0.000	-
2	0.010	0.010	1.000
3	0.000	0.000	-
4	0.125	0.005	0.000
5	0.210	0.020	0.000
6	0.295	0.045	0.051
7	0.185	0.085	0.243
8	3.415	0.190	0.013
9	4.720	0.335	0.010
10	20.525	0.545	0.014
11	70.705	0.720	0.012
12	305.780	0.910	0.029
13	827.665	1.000	0.064
14	2219.780	1.000	0.128

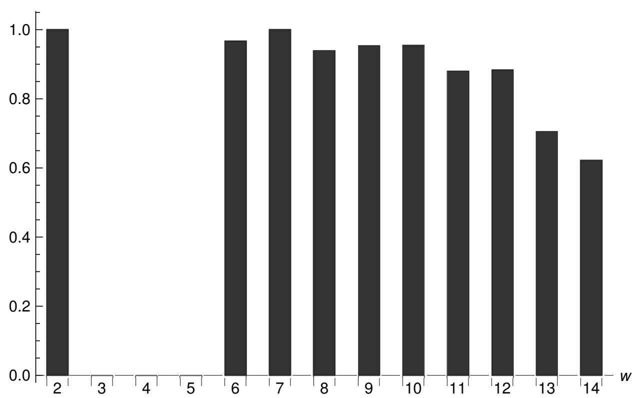


Figure 26. Average success rate of Algorithm 2 on stopping sets that allow a (T, S)-split for the 200 randomly generated matrices from the protograph-based $(4, 8)$-regular LDPC code ensemble. Note that there are no splittable stopping sets of sizes $w = 3, 4, 5$.

Table 13. Estimated termatiko set size spectra (initial part) for three protograph-based matrices from Fig. 23 before and after adding redundant rows. Numbers in angle brackets stand for termatiko distance $h_{\min}$, size of the smallest non-codeword stopping set $\tilde{s}_{\min}$, and minimum distance $d_{\min}$, respectively, for the original non-extended measurement matrices. Numbers in bold are exact. We tried to "remove" termatiko sets of size up to ℓ (including).

	$A_{\mathsf{PG}}^{(20)}\langle 1,4,2\rangle$		$A_{\mathsf{PG}}^{(72)}\langle 3,7,10\rangle$		$A_{\mathsf{PG}}^{(172)}\langle 6,8,6\rangle$	
w	original ($\ell=0$)	extended ($\ell=5$)	original ($\ell=0$)	extended ($\ell=7$)	original ($\ell=0$)	extended ($\ell=8$)
1	**2**	**2**	**0**	**0**	**0**	**0**
2	**4**	**1**	**0**	**0**	**0**	**0**
3	**11**	**0**	**1**	**0**	**0**	**0**
4	**82**	**0**	**3**	**0**	**0**	**0**
5	**837**	**16**	**19**	**2**	**0**	**0**
6	**7860**	**265**	**83**	**0**	**23**	**0**
7	84 059	5214	794	0	263	0
8	670 146	61 519	5204	98	1780	5
9	1 885 358	182 366	6904	109	2134	10
10	2 859 840	182 366	4806	68	1295	9
11	3 371 631	306 240	5124	18	1538	8
12	3 489 631	324 033	6717	35	2225	17
13	3 177 444	351 216	10 483	151	3819	36

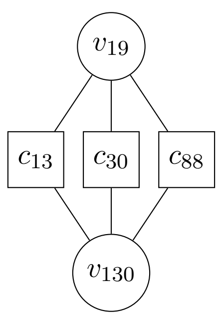
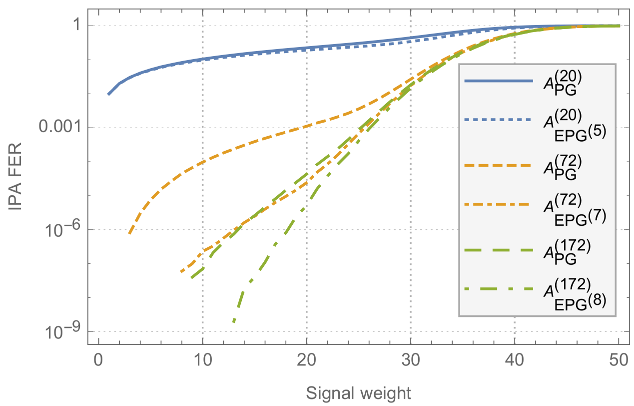
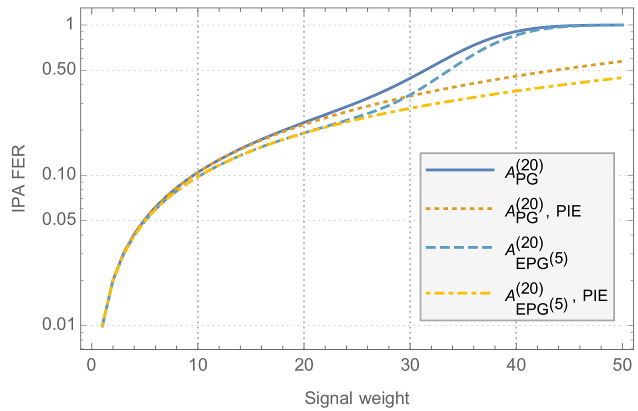


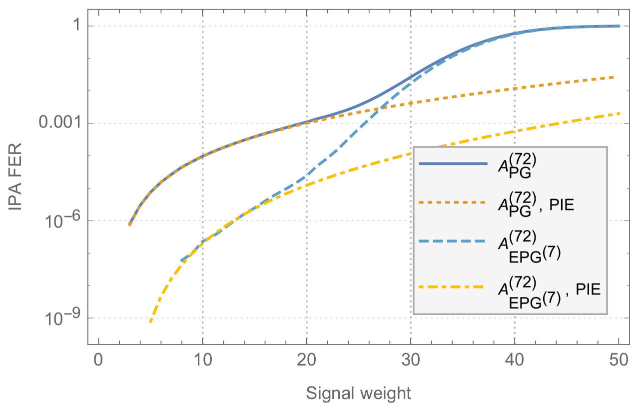
Figure 27. $\{v_{19}\}$ and $\{v_{130}\}$ are both size-1 termatiko sets in $A_{\mathsf{PG}}^{(20)}$.



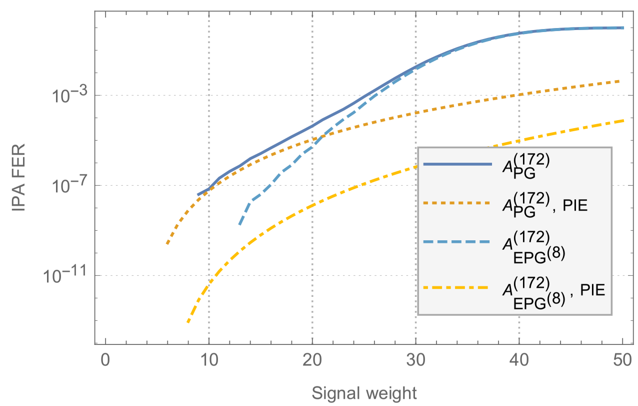
(a) All three matrices, both original and extended



(b) $A_{\text{PG}}^{(20)}$



(c) $A_{\text{PG}}^{(72)}$



(d) $A_{\text{PG}}^{(172)}$

Figure 28. FER performance of the IPA versus the weight of a signal vector for several protograph-based measurement matrices.

reconstruction of sparse signals have been investigated in [38] (see Figs. 4 and 8). We refer an interested reader to that work.

To better understand the curves, we also add lower bounds based on the principle of inclusion-exclusion. The following is a well-known result (see, e.g. [2, Ch. 1]).

Lemma 45 (principle of inclusion-exclusion (PIE)). *Assume that $\mathcal{A}_1$, $\mathcal{A}_2$, …, $\mathcal{A}_M$ are some arbitrary events. Then*

$$\mathbb{P}\left\{\bigcup_{i=1}^{M} \mathcal{A}_i\right\} = \sum_{k=1}^{M} (-1)^{k-1} \left(\sum_{\substack{I \subset [M] \\ |I|=k}} \mathbb{P}\left\{\bigcap_{i\in I} \mathcal{A}_i\right\} \right).$$

More precisely, we take into consideration only the 30–50 smallest termatiko sets of a matrix. Then we build a theoretical curve as if the matrix would contain only these termatiko sets. Hence, reconstruction fails if and only if the support of a signal contains any of these 30–50 termatiko sets as a subset.

Assume that the termatiko sets of the matrix are $T_1, T_2, \ldots$, and let $\mathcal{A}_i$ denote the event that a weight-w subset of $[n]$ chosen uniformly at random is a superset of T_i. We remark that if $T_i \subset T_j$, then $\mathcal{A}_i \supset \mathcal{A}_j$ and $\mathcal{A}_i \cup \mathcal{A}_j = \mathcal{A}_i$. Therefore, if we include T_i into the list of consideration, then there is no point to also include T_j. This pre-filtering can save computation time, as many termatiko sets are in fact subsets of other termatiko sets. Next, we consider only M termatiko sets which we denote by $T_1, T_2, \ldots, T_M$. Note that it is not required that the chosen termatiko sets are the M smallest; any M termatiko sets can be chosen and the result below will still be a correct lower bound. However, in the simulations, we take the M smallest ones, for some integer $M > 0$. This is because we are particularly interested in a negative effect of the smallest termatiko sets.

With the aforementioned notation, the true FER is lower-bounded as

$$\begin{aligned}
\mathrm{FER}(w) = \mathbb{P}\left\{\bigcup_{i} \mathcal{A}_i\right\} \geq \mathbb{P}\left\{\bigcup_{i=1}^{M} \mathcal{A}_i\right\} &\overset{\text{PIE}}{=} \sum_{k=1}^{M} (-1)^{k-1} \left(\sum_{\substack{I \subset [M] \\ |I|=k}} \mathbb{P}\left\{\bigcap_{i\in I} \mathcal{A}_i\right\} \right) \\
&= \frac{1}{\binom{n}{w}} \sum_{k=1}^{M} (-1)^{k-1} \left(\sum_{\substack{I \subset [M] \\ |I|=k}} \binom{n - \left|\bigcup_{i\in I} T_i\right|}{w - \left|\bigcup_{i\in I} T_i\right|} \right).
\end{aligned}$$

If the number of terms in the sum becomes too large, then we can use the truncated lower bound

$$\mathrm{FER}(w) \geq \frac{1}{\binom{n}{w}} \sum_{k=1}^{2L} (-1)^{k-1} \left(\sum_{\substack{I \subset [M] \\ |I|=k}} \binom{n - \left|\bigcup_{i \in I} T_i\right|}{w - \left|\bigcup_{i \in I} T_i\right|} \right)$$

for some $2L < M$ (the so-called Bonferroni inequality). This truncated expression becomes equal to the full inclusion-exclusion formula for weight w if $\left|\bigcup_{i \in I} T_i\right| > w$ for all $I \subset [M]$, $|I| > 2L$. This simple fact allows for faster calculation of better FER lower bounds for sparse signals. The FER curves together with the lower bounds are depicted in Figs. 28b to 28d.

The three matrices $A_{\mathsf{PG}}^{(20)}$, $A_{\mathsf{PG}}^{(72)}$, and $A_{\mathsf{PG}}^{(172)}$ represent different behaviour after adding redundant rows. $A_{\mathsf{PG}}^{(20)}$ is intrinsically bad and cannot be fixed as illustrated in Fig. 27. In particular, since both $\{v_{19}\}$ and $\{v_{130}\}$ are connected only to $\{c_{13}, c_{30}, c_{88}\}$, their values cannot be recovered. The reason being that if $v_{19} = \alpha$, $v_{130} = \beta$, and $\alpha + \beta > 0$, each of c_{13}, c_{30}, c_{88} keeps only the sum $\alpha + \beta$, and there are infinitely many solutions for α and β. It is worth noting that this is not a failure of the IPA, since, strictly speaking, the information has been lost in the compression process (even an optimal recovery algorithm would fail here).

On the other hand, both $A_{\mathsf{EPG}^{(7)}}^{(72)}$ and $A_{\mathsf{EPG}^{(8)}}^{(172)}$ increase termatiko distance (compared to $A_{\mathsf{PG}}^{(72)}$ and $A_{\mathsf{PG}}^{(172)}$, respectively), and show a significant improvement in the sparse region which shows the importance of designing measurement matrices with a high termatiko distance.

4. CONCLUSION

The good ended happily, and the bad unhappily. That is what Fiction means.
—Oscar Wilde, *The Importance of Being Earnest*

In this thesis, we studied the failure events of two iterative message-passing algorithms, namely the belief-propagation for LDPC decoding over the binary erasure channel and the interval passing algorithm for compressed sensing. Despite the fact that the algorithms appeared in rather different study domains, we were able to find many similarities in both their nature and the research methods we used.

In particular, for the case of the belief propagation decoder, we improved existing bounds on the stopping redundancy hierarchy of linear codes. We also generalised the concept to the case of stopping sets having size more than the minimum distance of a code. This gave a partial answer to the question how to achieve maximum-likelihood decoding performance with the belief propagation decoder.

For the interval-passing algorithm, we formulated and proved the precise criterion for the algorithm to fail. For that, we introduced termatiko sets as the core failure structures of the algorithm. We also suggested some heuristic methods to improve reconstruction performance of the interval-passing algorithm by methods borrowed from the belief propagation decoder. Besides that, we presented extensive numerical experiments, in particular, for measurement matrices from the array LDPC codes.

There are still many open questions left. One of the main problems of stopping redundancy hierarchy is whether it is possible to construct a family of linear codes with its stopping redundancy growing polynomially with the length of a code. We conjecture that for a rather general family of codes, stopping redundancy grows exponentially.

As to the interval-passing algorithm, we think that it is possible to improve its reconstruction abilities by judiciously choosing measurement matrices. While we suggest one target characteristic in the search for good measurement matrices—high termatiko distance—it is of interest to construct explicit matrices. The first step in this direction have been already done, see for example [18].

Appendix A. OPTIMAL PARITY-CHECK MATRIX ROW WEIGHT

In this appendix, we aim to find a weight w of a row in a parity-check matrix, which covers the maximum number of stopping sets of size up to ℓ, provided that n is fixed. It is easy to see that any row of length n and weight w covers exactly

$$w \sum_{i=1}^{\ell} \binom{n-w}{i-1}$$

stopping sets of weight up to ℓ. Lemma 46 provides an answer to that maximization question.

Lemma 46. *Fix two positive integers n and $2 \le \ell \le n$ and define a discrete function $F : \{1, 2, \ldots, n-\ell+1\} \to \mathbb{N}$ in the following way:*

$$F(w) = F_{n,\ell}(w) = w \sum_{i=0}^{\ell-1} \binom{n-w}{i} .$$

Then

$$\arg\max_{w} F(w) \in \left\{ \left\lfloor \frac{n+1}{\ell} \right\rfloor, \left\lceil \frac{n}{\ell} \right\rceil \right\} .$$

Proof. First of all, it is easy to see that

$$\left\lfloor \frac{n+1}{\ell} \right\rfloor = \left\lceil \frac{n}{\ell} \right\rceil \quad \text{or} \quad \left\lfloor \frac{n+1}{\ell} \right\rfloor + 1 = \left\lceil \frac{n}{\ell} \right\rceil .$$

Then, to prove the statement of the lemma, it is sufficient to show that $F(w)$ increases for $w < \left\lfloor \frac{n+1}{\ell} \right\rfloor$ and decreases for $w \ge \left\lceil \frac{n}{\ell} \right\rceil$.

Consider a finite difference:

$$\Delta F(w) = F(w+1) - F(w) .$$

It can be expanded as follows:

$$\begin{aligned}
\Delta F(w) &= F(w+1) - F(w) \\
&= (w+1) \sum_{i=0}^{\ell-1} \binom{n-w-1}{i} - w \sum_{i=0}^{\ell-1} \binom{n-w}{i} \\
&= (w+1) \sum_{i=0}^{\ell-1} \binom{n-w-1}{i} - w \sum_{i=0}^{\ell-1} \left(\binom{n-w-1}{i} + \binom{n-w-1}{i-1} \right) \\
&= \sum_{i=0}^{\ell-1} \binom{n-w-1}{i} - w \sum_{i=0}^{\ell-1} \binom{n-w-1}{i-1} .
\end{aligned}$$

We have:

$$\begin{aligned}\Delta F(w) &= \sum_{i=0}^{\ell-1}\left(\binom{n-w-1}{i} - w\binom{n-w-1}{i-1}\right)\\ &= \sum_{i=0}^{\ell-1}\frac{(n-w-1)!}{i!(n-w-i)!}\left(n-i-w(i+1)\right)\ .\end{aligned}$$

If we require that

$$w \le \frac{n-\ell+1}{\ell}\,,$$

then it follows also that

$$w < \frac{n-i}{i+1} \quad \text{for all } i < \ell-1\,;$$

hence, each of the terms $(n-i-w(i+1))$ is positive for $i<\ell-1$ and $(n-\ell+1-w\ell)\ge 0$. Therefore,

$$F(1) < F(2) < \cdots < F\left(\left\lfloor\frac{n+1}{\ell}-1\right\rfloor\right) < F\left(\left\lfloor\frac{n+1}{\ell}\right\rfloor\right)\ .$$

On the other hand, we can write:

$$\begin{aligned}\Delta F(w) &= \sum_{i=0}^{\ell-1}\binom{n-w-1}{i} - w\sum_{i=0}^{\ell-1}\binom{n-w-1}{i-1}\\ &= \sum_{i=0}^{\ell-1}\binom{n-w-1}{i} - w\sum_{i=0}^{\ell-2}\binom{n-w-1}{i}\\ &= \binom{n-w-1}{\ell-1} + (1-w)\sum_{i=0}^{\ell-2}\binom{n-w-1}{i}\,.\end{aligned}$$

And, if $w>1$, we have:

$$\begin{aligned}\Delta F(w) &< \binom{n-w-1}{\ell-1} + (1-w)\binom{n-w-1}{\ell-2}\\ &= \frac{(n-w-1)!}{(\ell-1)!(n-\ell-w+1)!}(n-w\ell)\,.\end{aligned}$$

If we further require $w \ge \frac{n}{\ell}$, then $\Delta F(w) < 0$ and

$$F\left(\left\lceil\frac{n}{\ell}\right\rceil\right) > F\left(\left\lceil\frac{n}{\ell}\right\rceil+1\right) > \cdots > F(n-\ell+1). \qquad \square$$

Appendix B. FULL-RANK BINARY MATRICES WITH NO ROWS OF HAMMING WEIGHT ONE

In this appendix, we compute the number of full-rank binary matrices with no rows of weight one. The results in this appendix are based on [1].

Lemma 47. *Let $m \geq i$ and denote by $\mathcal{M}(m,i)$ the number of full-rank binary $m \times i$ matrices. Then*

$$\mathcal{M}(m,i) = \prod_{t=0}^{i-1} (2^m - 2^t).$$

Proof. As $m \geq i$, all columns in such matrices are linearly independent. We have $2^m - 1$ choices for the first column (any non-zero vector in $\mathbb{F}_2^m$), $2^m - 2$ choices for the second column (any vector in $\mathbb{F}_2^m$ except the all-zero vector and the first column), $2^m - 2^2$ choices for the third column (any vector in $\mathbb{F}_2^m$ except for the vectors in the subspace spanned by the first two columns), etc. Altogether, we have

$$\mathcal{M}(m,i) = (2^m - 1)(2^m - 2)\cdots(2^m - 2^{i-1}) = \prod_{t=0}^{i-1}(2^m - 2^t). \qquad \square$$

Lemma 48. *Let $m \geq i$ and denote by $\mathcal{N}(m,i)$ the number of full-rank binary $m \times i$ matrices with no rows of Hamming weight one. Then*

$$\begin{aligned} \mathcal{N}(m,i) = \sum_{k=0}^{i} \binom{i}{k} \cdot k! \sum_{p=0}^{m} (-1)^{m-p} \cdot \binom{m}{p} \\ \cdot 2^{kp} \cdot S(m-p,k) \prod_{t=0}^{i-k-1} \left(2^p - 2^t\right), \end{aligned} \tag{B.1}$$

where $S(x,y)$ is a Stirling number of the second kind:

$$S(x,y) \triangleq \frac{1}{y!} \sum_{j=0}^{y} (-1)^{y-j} \binom{y}{j} j^x.$$

Proof. Using the result of Lemma 47, the number of full-rank $m \times i$ matrices with exactly z zero rows can be obtained by using the inclusion-exclusion principle, as follows:

$$\binom{m}{z} \sum_{p=0}^{m-z} (-1)^{m-z-p} \binom{m-z}{p} \prod_{t=0}^{i-1} (2^p - 2^t)\,. \tag{B.2}$$

Now, let us consider the requirement not to have rows of weight one. We use the inclusion-exclusion principle.

Let P_ι ($\iota = 1, 2, \ldots, i$) be the property that there is a row with a single 1 at ι'th coordinate. Suppose that an $m \times i$ matrix satisfies properties with indices from a set $R \subseteq [i]$ with $|R| = k$. Then the set of row indices is partitioned as

$$[m] = J \sqcup \bar{J}\,,$$

where J consists of indices corresponding to rows with a single 1 at a coordinate from R, and $\bar{J} = [m] \setminus J$. Let $|J| = j$ (we have $j \geq k$).

To enumerate possible submatrices, whose rows are indexed by J and columns by $[i]$, we notice that their columns essentially define an ordered partition of their rows into k non-empty sets. Hence, the number of such submatrices equals to $k!\,S(j,k)$.

The number of submatrices whose rows and columns are indexed by $\bar{J}$ and $\bar{R}$, respectively, with exactly z zero rows can be calculated from (B.2). They can be extended to all submatrices with rows indexed by $\bar{J}$ in $(2^k - k)^z (2^k)^{m-j-z}$ ways because each zero row can be extended by anything except of k-vectors of weight 1 (as we already collected them in rows J), and others can be extended by anything.

Putting all together, we have

$$\begin{aligned}
\mathcal{N}(m,i) &= \sum_{k=0}^{i} (-1)^k \binom{i}{k} \sum_{j=k}^{m} \binom{m}{j} k! S(j,k) \\
&\quad \cdot \sum_{z=0}^{m-j} \binom{m-j}{z} (2^k - k)^z (2^k)^{m-j-z} \\
&\quad \cdot \sum_{p=0}^{m-j-z} (-1)^{m-j-z-p} \binom{m-j-z}{p} \prod_{t=0}^{i-k-1} (2^p - 2^t) \\
&= \sum_{k=0}^{i} (-1)^k \binom{i}{k} \sum_{j=k}^{m} \binom{m}{j} k! S(j,k) \\
&\quad \cdot \sum_{p=0}^{m-j} (-1)^{m-j-p} \binom{m-j}{p} 2^{kp} k^{m-j-p} \prod_{t=0}^{i-k-1} (2^p - 2^t)
\end{aligned}$$

Here, we understand $0^0 = 1$. For instance, $k^{m-j-p} = 1$ for the case $k = m - j - p = 0$. Further, we expand $S(j,k)$ according to the definition and get:

$$\begin{aligned}
\mathcal{N}(m,i) &= \sum_{k=0}^{i} \binom{i}{k} \sum_{j=k}^{m} \binom{m}{j} \sum_{\ell=0}^{k} (-1)^\ell \binom{k}{\ell} \ell^j \\
&\quad \cdot \sum_{p=0}^{m-j} (-1)^{m-j-p} \binom{m-j}{p} 2^{kp} k^{m-j-p} \prod_{t=0}^{i-k-1} (2^p - 2^t)
\end{aligned}$$

$$= \sum_{k=0}^{i} \binom{i}{k} \sum_{\ell=0}^{k} (-1)^{\ell} \binom{k}{\ell} \sum_{j=k}^{m} \sum_{p=0}^{m-j} \binom{m}{j} \ell^{j}$$

$$\cdot (-1)^{m-j-p} \binom{m-j}{p} 2^{kp} k^{m-j-p} \prod_{t=0}^{i-k-1} (2^p - 2^t)$$

$$= \sum_{k=0}^{i} \binom{i}{k} \sum_{\ell=0}^{k} (-1)^{k-\ell} \binom{k}{\ell} \sum_{j=k}^{m} \sum_{p=0}^{m-j} \binom{m}{j} \ell^{j}$$

$$\cdot (-1)^{m-j-p+k} \binom{m-j}{p} 2^{kp} k^{m-j-p} \prod_{t=0}^{i-k-1} (2^p - 2^t)$$

We continue simplifications of the formula:

$$\mathcal{N}(m,i) = \sum_{k=0}^{i} \binom{i}{k} \sum_{\ell=0}^{k} (-1)^{k-\ell} \binom{k}{\ell}$$

$$\cdot \sum_{p=0}^{m} \binom{m}{p} 2^{kp} (-\ell)^{m-p} \prod_{t=0}^{i-k-1} (2^p - 2^t)$$

$$= \sum_{k=0}^{i} \binom{i}{k} k! \sum_{p=0}^{m} (-1)^{m-p} \binom{m}{p} 2^{kp} S(m-p,k) \prod_{t=0}^{i-k-1} (2^p - 2^t) \, .$$

□

We note that for the medium and large values of m and i, the ratio of the number of full-rank binary $m \times i$ matrices without rows of weight one to the number of all full-rank binary matrices is quite close to 1, and hence the relative error becomes close to 0. For example, for $m = 50$ and $i = 30$ we have

$$\frac{\mathcal{M}(50,30) - \mathcal{N}(50,30)}{\mathcal{N}(50,30)} \approx 1.40 \cdot 10^{-6}.$$

Since obviously $\mathcal{M}(m,i) \geq \mathcal{N}(m,i)$, the former is a correct upper bound, which is rather tight for the medium and large values of m and i. For practical purposes, calculating and analysing $\mathcal{M}(m,i)$ is much easier than $\mathcal{N}(m,i)$.

Appendix C. PROOF OF THEOREM 42

To prove Theorem 42, we need the following lemma.

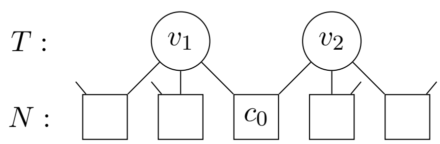
Lemma 49. *Assume $T = \{v_1, v_2, v_3\}$ is a termatiko set of size 3 in $H(q, 3)$. Define N and S analogously to Theorem 35. Then, $S \neq \varnothing$, and for each $c \in N$, it holds that $|\mathcal{N}_T(c)| = 1$ and $|\mathcal{N}_S(c)| > 0$.*

Proof. Assume first that some $c_0 \in N$ is not connected to S (including the case $S = \varnothing$). Then, from Theorem 35, c_0 is connected to T at least twice (w.l.o.g. let v_1 and v_2 be these two variable nodes) and for any $c \in \mathcal{N}(v_1) \cup \mathcal{N}(v_2)$ (including $c = c_0$) it holds that $|\mathcal{N}_T(c)| \geq 2$. See Fig. 29a for illustration. As any two variable nodes share not more than one measurement node, we have $\mathcal{N}(v_1) \cap \mathcal{N}(v_2) = \{c_0\}$. Therefore, since $|\mathcal{N}(v_1)| = |\mathcal{N}(v_2)| = 3$, we have $|\mathcal{N}(v_1) \cup \mathcal{N}(v_2)| = 5$. Now, count number of edges between T and N. On the one hand, it is $|\mathcal{N}(v_1)| + |\mathcal{N}(v_2)| + |\mathcal{N}(v_3)| = 3 + 3 + 3 = 9$. On the other hand, it is not less than

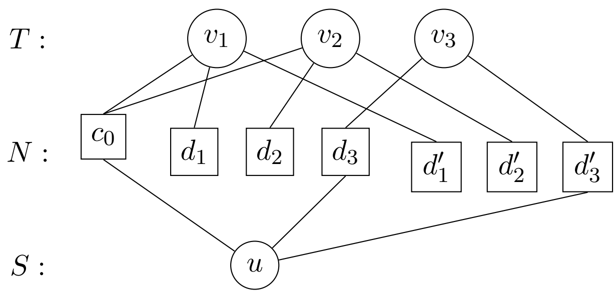
$$\sum_{c \in \mathcal{N}(v_1) \cup \mathcal{N}(v_2)} |\mathcal{N}_T(c)| \geq 2\,|\mathcal{N}(v_1) \cup \mathcal{N}(v_2)| = 10\,.$$

This contradiction shows that $S \neq \varnothing$ and that each $c \in N$ is connected to both T and S.

Next, we prove that each $c \in N$ is connected to T only once. Again, assume to the contrary that some $c_0 \in N$ is connected to T at least twice, w.l.o.g. to



(a) Scenario under the assumption that there exists a measurement node $c_0 \in N$ not connected to S



(b) Scenario under the assumption that there exists a measurement node $c_0 \in N$ connected to T twice. The measurement nodes are grouped according to the three different strips as $\{c_0\}$, $\{d_1, d_2, d_2\}$, and $\{d'_1, d'_2, d'_3\}$

Figure 29. Illustration for Lemma 49.

v_1 and v_2, and let $u \in S$ be connected to c_0 (as we have just shown, such u exists). Recall that $\mathcal{N}(u) \subset N$ by definition of S from Theorem 35. Since v_1 and v_2 are both connected to c_0, they do not share any other measurement node. Additionally, recall that each variable node is connected to three measurement nodes, each from a different strip. Hence, v_1 and v_2 are connected to different measurement nodes $d_1, d_2 \in N$ in another strip (different from the strip of c_0), and also to two different measurement nodes $d_1', d_2' \in N$ in the remaining strip. See Fig. 29b for illustration. Now, u cannot be connected to any of d_1, d_2, d_1', d_2' as it already shares one measurement node with each of v_1 and v_2. Therefore, there exists a measurement node $d_3 \in \mathcal{N}(u)$ in the same strip that contains d_1 and d_2. However, d_3 should be also connected to T. Thus, the only possibility left is that d_3 is connected to v_3. The same argument can be used for the strip that contains d_1' and d_2'; it contains a node d_3', and d_3' is connected to both u and v_3. We have a contradiction, as u and v_3 share two different measurement nodes (meaning that there should exist a cycle of length 4 in the corresponding Tanner graph). Therefore, every $c \in N$ is connected to T exactly once. □

From Lemma 49 it follows that $|N| = 9$ and that v_1, v_2, v_3 do not share any measurement nodes. Next, we turn to the proof of Theorem 42.

Proof. From Theorem 40 we know that $h_{\min} \geq 3$; thus, we only need to prove the multiplicity result. Assume we have a termatiko set $T = \{v_1, v_2, v_2\}$, and denote $\mathcal{N}(v_1) = \{c_{11}, c_{21}, c_{31}\}$, where c_{11}, c_{21}, c_{31} belong to the first, the second, and the third strips, respectively. Analogously, denote $\mathcal{N}(v_2) = \{c_{12}, c_{22}, c_{32}\}$ and $\mathcal{N}(v_3) = \{c_{13}, c_{23}, c_{33}\}$. As shown above, $|N| = |\{c_{11}, \ldots, c_{33}\}| = 9$ (all these measurement nodes are different). As usual, we define the set S as in Theorem 35.

In order not to share any two (or more) measurement nodes with any of v_1, v_2, v_3, each $u \in S$ should be connected to $c_{1\pi_1}$, $c_{2\pi_2}$, and $c_{3\pi_3}$, where $\boldsymbol{\pi} = \boldsymbol{\pi}^{(u)} = (\pi_1, \pi_2, \pi_3)$ is some permutation of $\{1, 2, 3\}$. Thus, we will denote candidates for the set S as $u_{\pi_1\pi_2\pi_3}$. In other words, $\mathcal{N}(u_{\pi_1\pi_2\pi_3}) = \{c_{1\pi_1}, c_{2\pi_2}, c_{3\pi_3}\}$, from which it follows that there are 6 candidates for S and $|S| \leq 6$. Turn to Fig. 30 for illustration.

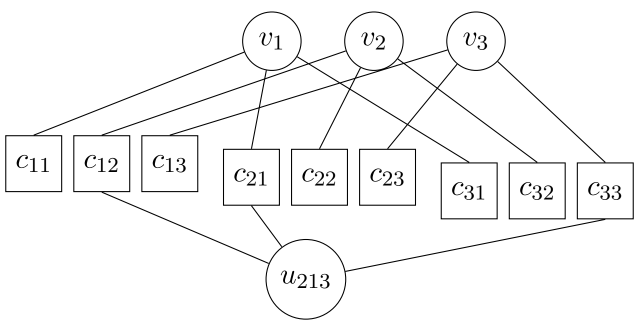


Figure 30. Illustration for the proof of Theorem 42 for $\boldsymbol{\pi} = (2, 1, 3)$ and hence u_{213}. Vertices $c_{11}, c_{12}, \ldots, c_{33}$ are grouped according to the three different strips.

As each $c_{xy} \in N$ (for all $x, y \in \{1, 2, 3\}$) should be connected to S, S should include some $u_{\boldsymbol{\pi}}$ with $\pi_x = y$, for all choices of x and y. For example, c_{11} should be connected to S, and thus either u_{123} or u_{132} (or both) should be present in S.

By applying the corresponding automorphism, we can set $v_1 = (0, 0)$ and $v_2 = (2, j)$ for some $j \in \mathbb{F}_q$.[1] With this notation, the support matrix of T becomes

$$\begin{bmatrix} 0 & 2 & \cdot \\ 0 & 2+j & \cdot \\ 0 & 2+2j & \cdot \end{bmatrix},$$

where the dots stand for entries which are currently unknown.

For the remainder of the proof, we exhaustively check all cases and sub-cases, based on the assumption that some $u_{\pi_1\pi_2\pi_3} \in S$. As we noted before, since c_{11} should be connected to S, either u_{123} or u_{132} (or both) should be in S.

1. First, assume that $u_{123} \in S$, which means that c_{11}, c_{22}, and c_{33} are connected to the same variable node (u_{123}), and thus the corresponding values in the support matrix will form an arithmetic progression. More precisely, the values $\{0, 2+j, \cdot\}$ should form an arithmetic progression, and we immediately obtain the support matrix

$$\begin{bmatrix} 0 & 2 & \cdot \\ 0 & 2+j & \cdot \\ 0 & 2+2j & 4+2j \end{bmatrix}.$$

   Further, c_{12} should also be connected to S, and thus either u_{213} or u_{231} (or both) should be in S.

   - By assuming that $u_{213} \in S$, we obtain that c_{12}, c_{21}, and c_{33} are connected to the same variable node $u_{213} \in S$. Hence, the values $\{2, 0, 4+2j\}$ should form an arithmetic progression. From this we get that $4 + 2j = -2$ and then $j = -3$. The updated support matrix is

$$\begin{bmatrix} 0 & 2 & \cdot \\ 0 & -1 & \cdot \\ 0 & -4 & -2 \end{bmatrix}.$$

   - By assuming that $u_{231} \in S$, we have that $\{2, \cdot, 0\}$ form an arithmetic progression and then we can replace "$\cdot$" by 1. However, the values in the column of any support matrix should also form an arithmetic progression. Hence, the support matrix becomes

$$\begin{bmatrix} 0 & 2 & -2-2j \\ 0 & 2+j & 1 \\ 0 & 2+2j & 4+2j \end{bmatrix}.$$

[1] Note that we choose the integer 2 to make further numbers look "prettier", although any non-zero value from $\mathbb{F}_q$ would work here.

Other sub-cases are omitted for brevity.

2. On the other hand, if we assume $u_{132} \in S$, then the values corresponding to c_{11}, c_{23}, and c_{32} (i.e. $\{0, \cdot, 2+2j\}$) form an arithmetic progression. From this we immediately obtain the updated support matrix

$$\begin{bmatrix} 0 & 2 & \cdot \\ 0 & 2+j & 1+j \\ 0 & 2+2j & \cdot \end{bmatrix} .$$

We again omit further sub-cases for brevity.

The different cases can be represented as nodes in a search tree (see Fig. 31). Note that the branches in the tree are not mutually exclusive, but they cover all cases. This means that the same termatiko set can be obtained more than once. The two cases marked in bold in Fig. 31 are general cases. Moreover, by setting $j = 0$ or $j = -3$, we can obtain other particular cases (these relations are shown by dotted arrows). Note that branching stops at these general cases, as even these general forms already ensure that $\{v_1, v_2, v_3\}$ is a valid termatiko set. Other branches need to go one level deeper. Since the set of equations

$$\begin{cases} -2-2j & = 4+2j\,, \\ 1 & = 1+j\,, \\ 4+2j & = -2 \end{cases}$$

do not have a solution for $q \geq 5$, these two general cases do not intersect.

Nonetheless, we still need to check that the three columns are different in each of these two cases. The corresponding requirement for the first bold case is

$$\begin{cases} 0 & \neq 2+j, \\ 0 & \neq 2+2j, \\ 0 & \neq -2-2j, \\ 0 & \neq 4+2j, \\ 2 & \neq -2-2j, \\ 2+j & \neq 1, \end{cases} \quad \Leftrightarrow \quad \begin{cases} j & \neq -2, \\ j & \neq -1. \end{cases}$$

For the second bold case we obtain the condition

$$\begin{cases} 0 & \neq 2+j\,, \\ 0 & \neq 2+2j\,, \\ 0 & \neq 4+2j\,, \\ 0 & \neq 1+j\,, \\ 2 & \neq 4+2j\,, \\ 2+2j & \neq -2\,, \end{cases} \quad \Leftrightarrow \quad \begin{cases} j & \neq -2\,, \\ j & \neq -1\,. \end{cases}$$

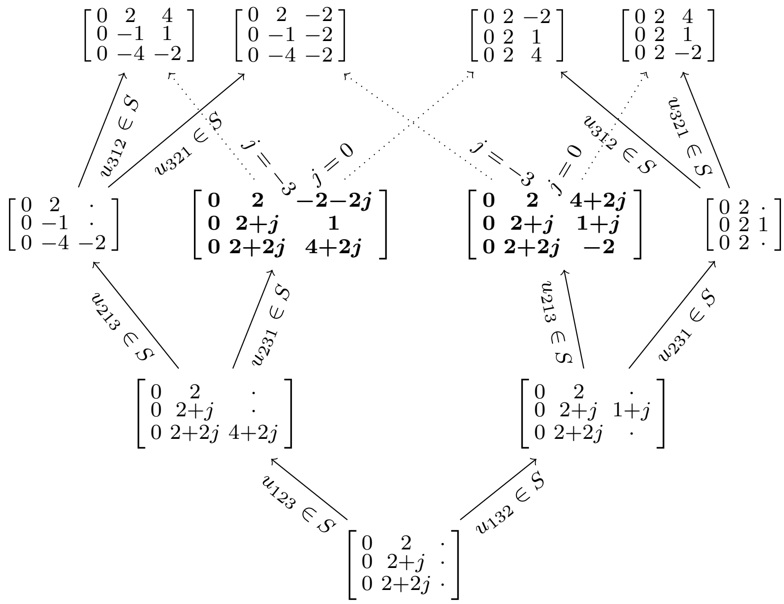


Figure 31. Different cases for the proof of Theorem 42. Dotted arrows show special cases for particular values of the variable j.

Therefore, in total there are $q-2$ choices for j in each of the cases. This means that there are exactly $2(q-2)$ termatiko sets with fixed $v_1 = (0,0)$ and $v_2 = (2,\cdot)$. Any other termatiko set of size 3 in $H(q,3)$ can be obtained by applying an automorphism (there are $q^2(q-1)$ such automorphisms). However, in this manner, we count each termatiko set $3! = 6$ times. Thus, the total number of distinct size-3 termatiko sets in $H(q,3)$ is $q^2(q-1)(q-2)/3$. $\square$

BIBLIOGRAPHY

[1] M. Alekseyev. Number of full-rank binary matrices with no rows of weight 1. MathOverflow, 2017. `https://mathoverflow.net/q/264835` (version: 2017-03-20).

[2] D. P. Bertsekas and J. N. Tsitsiklis. *Introduction to probability*, volume 1. Athena Scientific Belmont, MA, 2002.

[3] I. E. Bocharova, B. D. Kudryashov, V. Skachek, and Y. Yakimenka. Distance properties of short LDPC codes and their impact on the BP, ML and near-ML decoding performance. In *Int. Castle Meeting Coding Theory Applications*, pages 48–61. Springer, 2017.

[4] I. E. Bocharova, B. D. Kudryashov, V. Skachek, and Y. Yakimenka. Improved redundant parity-check based BP decoding of LDPC codes. In *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, pages 1161–1165, 2018.

[5] T. T. Cai. One-sided confidence intervals in discrete distributions. *J. Stat. Planning Inference*, 131(1):63–88, Apr. 2005.

[6] E. J. Candes and T. Tao. Decoding by linear programming. *IEEE Trans. Inf. Theory*, 51(12):4203–4215, Dec. 2005.

[7] E. J. Candes and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. Inf. Theory*, 52(12):5406–5425, Dec. 2006.

[8] V. Chandar, D. Shah, and G. W. Wornell. A simple message-passing algorithm for compressed sensing. In *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, pages 1968–1972, Austin, TX, June 2010.

[9] D. L. Donoho. Compressed sensing. *IEEE Trans. Inf. Theory*, 52(4):1289–1306, Apr. 2006.

[10] D. L. Donoho, A. Javanmard, and A. Montanari. Information-theoretically optimal compressed sensing via spatial coupling and approximate message passing. *IEEE Trans. Inf. Theory*, 59(11):7434–7464, Nov. 2013.

[11] D. L. Donoho, A. Maleki, and A. Montanari. Message-passing algorithms for compressed sensing. *Proc. Nat. Acad. Sci.*, 106(45):18914–18919, Sept. 2009.

[12] P. Elias. Error-free coding. *Trans. IRE Professional Group Inf. Theory (TIT)*, 4:29–37, 1954.

[13] T. Etzion. On the stopping redundancy of Reed-Muller codes. *IEEE Trans. Inf. Theory*, 52(11):4867–4879, Nov. 2006.

[14] J. L. Fan. Array codes as low-density parity-check codes. In *Proc. 2nd Int. Symp. Turbo Codes & Rel. Topics*, pages 543–546, Brest, France, Sept. 2000.

[15] J. Feldman, M. J. Wainwright, and D. R. Karger. Using linear programming to decode binary linear codes. *IEEE Trans. Inf. Theory*, 51(3):954–972, Mar. 2005.

[16] R. G. Gallager. Low-density parity-check codes. *IRE Trans. Inf. Theory*, 8(1):21–28, 1962.

[17] R. G. Gallager. Low-density parity-check codes, 1963. PhD thesis.

[18] S. Habib and J. Kliewer. Algebraic optimization of binary spatially coupled measurement matrices for interval passing. *arXiv preprint arXiv:1809.05647*, 2018.

[19] J. Han and P. H. Siegel. Improved upper bounds on stopping redundancy. *IEEE Trans. Inf. Theory*, 53(1):90–104, Jan. 2007.

[20] J. Han, P. H. Siegel, and A. Vardy. Improved probabilistic bounds on stopping redundancy. *IEEE Trans. Inf. Theory*, 54(4):1749–1753, Apr. 2008.

[21] T. Hehn, O. Milenkovic, S. Laendner, and J. B. Huber. Permutation decoding and the stopping redundancy hierarchy of cyclic and extended cyclic codes. *IEEE Trans. Inf. Theory*, 54(12):5308–5331, Dec. 2008.

[22] H. D. Hollmann and L. M. Tolhuizen. Generating parity check equations for bounded-distance iterative erasure decoding. In *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, pages 514–517, July 2006.

[23] H. D. Hollmann and L. M. Tolhuizen. Generating parity check equations for bounded-distance iterative erasure decoding of even weight codes. In *Proc. 27th Symp. Inf. Theory Benelux*, pages 17–24, June 2006.

[24] H. D. Hollmann and L. M. Tolhuizen. Generic erasure correcting sets: Bounds and constructions. *J. Comb. Theory, Ser. A*, 113(8):1746–1759, Nov. 2006.

[25] H. D. Hollmann and L. M. Tolhuizen. On parity-check collections for iterative erasure decoding that correct all correctable erasure patterns of a given size. *IEEE Trans. Inf. Theory*, 53(2):823–828, Feb. 2007.

[26] *IEEE Standard for Local and metropolitan area networks, Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems, Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1*, Feb. 2006. IEEE Std 802.16e-2005 and IEEE Std 802.16-2004/Cor1-2005.

[27] R. Johannesson and K. S. Zigangirov. *Fundamentals of convolutional coding*, volume 15. John Wiley & Sons, 2015.

[28] S. Litsyn and V. Shevelev. On ensembles of low-density parity-check codes: asymptotic distance distributions. *IEEE Trans. Inf. Theory*, 48(4):887–908, 2002.

[29] H. Liu, L. Ma, and J. Chen. On the number of minimum stopping sets and minimum codewords of array LDPC codes. *IEEE Commun. Lett.*, 14(7):670–672, July 2010.

[30] Y. Lu, A. Montanari, B. Prabhakar, S. Dharmapurikar, and A. Kabbani. Counter braids: A novel counter architecture for per-flow measurement. In *Proc. Int. Conf. Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 121–132, Annapolis, MD, June 2008. ACM.

[31] M. G. Luby, M. Mitzenmacher, M. A. Shokrollah, and D. A. Spielman. Analysis of low density codes and improved designs using irregular graphs. In *Proc. 30th Annu. ACM Symp. Theory of Computing*, pages 249–258. ACM, 1998.

[32] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Efficient erasure correcting codes. *IEEE Trans. Inf. Theory*, 47(2):569–584, 2001.

[33] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Improved low-density parity-check codes using irregular graphs. *IEEE Trans. Inf. Theory*, 47(2):585–598, 2001.

[34] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann. Practical loss-resilient codes. In *Proc. 29th Annu. ACM Symp. Theory of Computing*, pages 150–159. ACM, 1997.

[35] F. J. MacWilliams and N. J. A. Sloane. *The theory of error-correcting codes*. Elsevier, 1977.

[36] T. Mittelholzer. Efficient encoding and minimum distance bounds of Reed-Solomon-type array codes. In *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, page 282, Lausanne, Switzerland, June/July 2002.

[37] H. V. Pham, W. Dai, and O. Milenkovic. Sublinear compressive sensing reconstruction via belief propagation decoding. In *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, pages 674–678, Seoul, Korea, June/July 2009.

[38] V. Ravanmehr, L. Danjean, B. Vasic, and D. Declercq. Interval-passing algorithm for non-negative measurement matrices: Performance and reconstruction analysis. *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, 2(3):424–432, Sept. 2012.

[39] T. Richardson and R. Urbanke. Efficient encoding of low-density parity-check codes. *IEEE Trans. Inf. Theory*, 47(2):638–656, Feb. 2001.

[40] T. Richardson and R. Urbanke. *Modern coding theory*. Cambridge university press, 2008.

[41] E. Rosnes, M. A. Ambroze, and M. Tomlinson. On the minimum/stopping distance of array low-density parity-check codes. *IEEE Trans. Inf. Theory*, 60(9):5204–5214, Sept. 2014.

[42] E. Rosnes and A. G. i Amat. Asymptotic analysis and spatial coupling of counter braids. *IEEE Trans. Inf. Theory*, pages 1–1, 2018.

[43] E. Rosnes and Ø. Ytrehus. An efficient algorithm to find all small-size stopping sets of low-density parity-check matrices. *IEEE Trans. Inf. Theory*, 55(9):4167–4178, Sept. 2009.

[44] E. Rosnes, Ø. Ytrehus, M. A. Ambroze, and M. Tomlinson. Addendum to “An efficient algorithm to find all small-size stopping sets of low-density parity-check matrices”. *IEEE Trans. Inf. Theory*, 58(1):164–171, Jan. 2012.

[45] S. Sarvotham, D. Baron, and R. G. Baraniuk. Sudocodes – fast measurement and reconstruction of sparse signals. In *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, pages 2804–2808, Seattle, WA, USA, July 2006.

[46] M. Schwartz and A. Vardy. On the stopping distance and the stopping redundancy of codes. *IEEE Trans. Inf. Theory*, 52(3):922–932, Mar. 2006.

[47] C. E. Shannon. A mathematical theory of communication. *Bell System Tech. J.*, 27(3):379–423, 623–656, July 1948.

[48] K. Sugiyama and Y. Kaji. On the minimum weight of simple full-length array LDPC codes. *IEICE Trans. Fundamentals*, E91-A(6):1502–1508, June 2008.

[49] R. M. Tanner, D. Sridhara, and T. Fuja. A class of group-structured LDPC codes. In *Proc. Int. Symp. Commun. Theory and Appl. (ISCTA)*, Ambleside, England, July 2001.

[50] J. H. Weber and K. A. Abdel-Ghaffar. Stopping set analysis for Hamming codes. In *Proc. IEEE Inf. Theory Workshop (ITW)*, pages 244–247, Aug. 2005.

[51] X. Wu and Z. Yang. Verification-based interval-passing algorithm for compressed sensing. *IEEE Signal Process. Lett.*, 20(10):933–936, Oct. 2013.

[52] Y. Yakimenka and E. Rosnes. On failing sets of the interval-passing algorithm for compressed sensing. In *Proc. 54th Annu. Allerton Conf. Commun., Control, Computing (Allerton)*, pages 306–311, Sept. 2016.

[53] Y. Yakimenka and E. Rosnes. Failure analysis of the interval-passing algorithm for compressed sensing. *arXiv preprint arXiv:1806.05110*, 2018.

[54] Y. Yakimenka and V. Skachek. Refined upper bounds on stopping redundancy of binary linear codes. In *Proc. IEEE Inf. Theory Workshop (ITW)*, pages 1–5, Apr. 2015.

[55] Y. Yakimenka, V. Skachek, I. E. Bocharova, and B. D. Kudryashov. Stopping redundancy hierarchy beyond the minimum distance. *IEEE Trans. Inf. Theory*, pages 1–1, 2018.

[56] K. Yang and T. Helleseth. On the minimum distance of array codes as LDPC codes. *IEEE Trans. Inf. Theory*, 49(12):3268–3271, Dec. 2003.

[57] F. Zhang and H. D. Pfister. Verification decoding of high-rate LDPC codes with applications in compressed sensing. *IEEE Trans. Inf. Theory*, 58(8):5042–5058, Aug. 2012.

INDEX

SUMMARY IN ESTONIAN

Sõnumivahetusalgoritmide tõrgete struktuurid kustutuste dekodeerimises ja hõredas signaalihõives

Esitatud tulemused on näiliselt kahest erinevast valdkonnast, nimelt käsitleme iteratiivse kanali dekodeerimise ja hõreda signaalihõive (ingl k. *compressed sensing*) meetodeid. Intervallivahetusalgoritmi (ingl k. *interval-passing algorithm, IPA*) tõrkeid hõrendatud signaalihõives soovitati mul uurida viiekuulise uurimiskülastuse jooksul Bergeni Ülikoolis. Leidsime palju sarnasusi nendes uurimisvaldkondades kasutatavate meetodite ja uurimisvahendite vahel. Me esitasime IPA jaoks termatiko hulgad (kreeka k. τερματικό ehk lõplik), mis käituvad täpselt samamoodi kui peatavad hulgad sõnumivahetusdekodeerimise korral üle kahendkustutuskanali (ingl k. *binary erasure channel, BEC*).

Shannon pani informatsiooniteooria uurimisele aluse juba 1948. aastal, jõudes järeldusele, et ükskõik kui halva kanali korral on alati võimalik informatsiooni veakindlalt edastada, kodeerides andmeid piisavalt suurel hulgal. Me käsitleme lineaarseid kodeerimise meetodeid kahendkustutuskanali kontekstis. Sellise kanali puhul infoühik kas jõuab veatult kohale või kustub, kusjuures info kustumine on vastuvõtjale tuvastatav.

1960ndatel pakkus Gallager välja lineaarsed hõredad paarsuskontrolli koodid (ingl k. *low-density parity-check, LDPC*), mis võimaldasid kiiret sõnumivahetusdekodeerimist. Lühikese ja keskmise pikkusega koodide puhul ei ole aga LDPC koodide jõudlus optimaalne. Kahendkustutuskanali korral on teada, et sõnumivahetusdekodeerimiseks kasutatavat paarsuskontrollimaatriksit saab laiendada ridadele liiasuse lisamisega. Käesoleva töö teine peatükk käsitleb lisatavate ridade arvu ülemise tõkke täiustamist. Me parandasime seni parimat ülemist tõket ning üldistasime nende kontseptsiooni. Antud peatükk hõlmab lisaks teoreetilisele materjalile ka hulgaliselt arvutuslikke katseid, mis teooriat kinnitavad.

Teine eelmainitud uurimisvaldkondadest, hõre signaalihõive, sai alguse Candèse ja Tao, ning eraldiseisvalt Donoho, töödest. Mitmeid olulisi signaale saab esitada hõredate vektoritena ja nemad pakkusid välja vastuvõetud signaalide jooksvalt hõrendamise, korrutades neid kaudselt läbi mõõtemaatriksiga. Sellisel juhul esialgse signaali taastamine on aga NP keerukusklassi kuuluv probleem. Keerukusest tulenevalt on välja töötatud lihtsamaid alternatiivseid meetodeid, milledest ühte, intervallivahetusalgoritmi, käsitleb käesoleva töö teine pool. Kolmandas peatükis me uurime, millistel juhtudel antud algoritm annab tõrke. Me kirjeldame täieliku graafiteoreetilise kriteeriumi, mille korral tõrked esinevad. Juhtumiuuringuna vaatlesime paarsuskontrollimaatrikseid LDPC koodides ja saime palju tulemusi tõrgete kohta, kasutades neid mõõtemaatriksitena IPAs.

Me uurisime sõnumivahetusalgoritme kustutuste dekodeerimises ja hõredas signaalihõives. See tõi nende algoritmide vahel esile mitmed sarnasused ja võimaldab ühtlustada uurimisvahendeid nende analüüsiks.

CURRICULUM VITAE

Personal data

Name:	Yauhen Yakimenka
Date of birth:	April 24th, 1986
Place of birth:	Baranavichy, Belarus
Citizenship:	Republic of Belarus
Languages:	Belarusian, Russian, and English
Contact:	jjauhien@gmail.com

Education

2014–	University of Tartu PhD candidate in Computer Science
2012–2014	Tallinn University of Technology & University of Tartu MSc in Engineering
2009–2011	Belarusian State University of Informatics and Radioelectronics MSc in Engineering
2003–2008	Belarusian State University BSc in Computer Science

Scientific work

Main fields of interest:

- Error-correcting codes
- Low-density parity-check codes
- Iterative methods of decoding
- Compressed sensing

ELULOOKIRJELDUS

Isikuandmed

Nimi:	Yauhen Yakimenka
Sünniaeg:	24. aprill 1986
Sünnikoht:	Baranavitšõ, Valgevene
Kodakondsus:	Valgevene Vabariik
Keeled:	Valgevene, vene ja inglise
Kontakt:	jjauhien@gmail.com

Haridus

2014–	Tartu Ülikool informaatika doktorant
2012–2014	Tallinna Tehnikaülikool ja Tartu Ülikool tehnikateaduste magister
2009–2011	Valgevene Riiklik Informaatika ja Raadioelektroonika Ülikool tehnikateaduste magister
2003–2008	Valgevene Riiklik Ülikool informaatika bakalaureus

Teadustegevus

Peamised uurimisvaldkonnad:

- veaparanduskoodid;
- hõredad paarsuskontrolli koodid;
- iteratiivsed dekodeerimise meetodid;
- hõre signaalihõive.

LIST OF ORIGINAL PUBLICATIONS

The following publications of the author served as a basis for this thesis.

1. Y. Yakimenka and V. Skachek. Refined upper bounds on stopping redundancy of binary linear codes. In *Proc. IEEE Inf. Theory Workshop (ITW)*, pages 1–5, Apr. 2015.
2. Y. Yakimenka and E. Rosnes. On failing sets of the interval-passing algorithm for compressed sensing. In *Proc. 54th Annu. Allerton Conf. Commun., Control, Computing (Allerton)*, pages 306–311, Sept. 2016.
3. I. E. Bocharova, B. D. Kudryashov, V. Skachek, and Y. Yakimenka. Distance properties of short LDPC codes and their impact on the BP, ML and near-ML decoding performance. In *Int. Castle Meeting Coding Theory Applications*, pages 48–61. Springer, 2017.
4. Y. Yakimenka, V. Skachek, I. E. Bocharova, and B. D. Kudryashov. Stopping redundancy hierarchy beyond the minimum distance. *IEEE Trans. Inf. Theory*, early access (accepted for publication).
5. Y. Yakimenka and E. Rosnes. Failure analysis of the interval-passing algorithm for compressed sensing. *arXiv preprint arXiv:1806.05110*, 2018. (Submitted for publication in IEEE Transactions on Information Theory)

Besides this, the author co-authored during his PhD studies the following papers that are not included in this thesis.

1. I. E. Bocharova, B. D. Kurdyashov, V. Skachek, and Y. Yakimenka. Low complexity algorithm approaching the ML decoding of binary LDPC codes. In *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, pages 2704–2708, 2016.
2. I. E. Bocharova, B. D. Kudryashov, V. Skachek, and Y. Yakimenka. Average spectra for ensembles of LDPC codes and applications. In *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, pages 361–365, 2017.
3. I. E. Bocharova, B. D. Kudryashov, V. Skachek, and Y. Yakimenka. Improved redundant parity-check based BP decoding of LDPC codes. In *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, pages 1161–1165, 2018.
4. I. E. Bocharova, B. D. Kudryashov, V. Skachek, and Y. Yakimenka. BP-LED decoding algorithm for LDPC codes over AWGN channels. *IEEE Trans. Inf. Theory*, early access (accepted for publication).

DISSERTATIONES INFORMATICAE PREVIOUSLY PUBLISHED IN DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS

19. **Helger Lipmaa.** Secure and efficient time-stamping systems. Tartu, 1999, 56 p.
22. **Kaili Müürisep.** Eesti keele arvutigrammatika: süntaks. Tartu, 2000, 107 lk.
23. **Varmo Vene.** Categorical programming with inductive and coinductive types. Tartu, 2000, 116 p.
24. **Olga Sokratova.** Ω-rings, their flat and projective acts with some applications. Tartu, 2000, 120 p.
27. **Tiina Puolakainen.** Eesti keele arvutigrammatika: morfoloogiline ühestamine. Tartu, 2001, 138 lk.
29. **Jan Villemson.** Size-efficient interval time stamps. Tartu, 2002, 82 p.
45. **Kristo Heero.** Path planning and learning strategies for mobile robots in dynamic partially unknown environments. Tartu 2006, 123 p.
49. **Härmel Nestra.** Iteratively defined transfinite trace semantics and program slicing with respect to them. Tartu 2006, 116 p.
53. **Marina Issakova.** Solving of linear equations, linear inequalities and systems of linear equations in interactive learning environment. Tartu 2007, 170 p.
55. **Kaarel Kaljurand.** Attempto controlled English as a Semantic Web language. Tartu 2007, 162 p.
56. **Mart Anton.** Mechanical modeling of IPMC actuators at large deformations. Tartu 2008, 123 p.
59. **Reimo Palm.** Numerical Comparison of Regularization Algorithms for Solving Ill-Posed Problems. Tartu 2010, 105 p.
61. **Jüri Reimand.** Functional analysis of gene lists, networks and regulatory systems. Tartu 2010, 153 p.
62. **Ahti Peder.** Superpositional Graphs and Finding the Description of Structure by Counting Method. Tartu 2010, 87 p.
64. **Vesal Vojdani.** Static Data Race Analysis of Heap-Manipulating C Programs. Tartu 2010, 137 p.
66. **Mark Fišel.** Optimizing Statistical Machine Translation via Input Modification. Tartu 2011, 104 p.
67. **Margus Niitsoo**. Black-box Oracle Separation Techniques with Applications in Time-stamping. Tartu 2011, 174 p.
71. **Siim Karus.** Maintainability of XML Transformations. Tartu 2011, 142 p.
72. **Margus Treumuth.** A Framework for Asynchronous Dialogue Systems: Concepts, Issues and Design Aspects. Tartu 2011, 95 p.
73. **Dmitri Lepp.** Solving simplification problems in the domain of exponents, monomials and polynomials in interactive learning environment T-algebra. Tartu 2011, 202 p.

74. **Meelis Kull.** Statistical enrichment analysis in algorithms for studying gene regulation. Tartu 2011, 151 p.
77. **Bingsheng Zhang.** Efficient cryptographic protocols for secure and private remote databases. Tartu 2011, 206 p.
78. **Reina Uba.** Merging business process models. Tartu 2011, 166 p.
79. **Uuno Puus.** Structural performance as a success factor in software development projects – Estonian experience. Tartu 2012, 106 p.
81. **Georg Singer.** Web search engines and complex information needs. Tartu 2012, 218 p.
83. **Dan Bogdanov.** Sharemind: programmable secure computations with practical applications. Tartu 2013, 191 p.
84. **Jevgeni Kabanov.** Towards a more productive Java EE ecosystem. Tartu 2013, 151 p.
87. **Margus Freudenthal.** Simpl: A toolkit for Domain-Specific Language development in enterprise information systems. Tartu, 2013, 151 p.
90. **Raivo Kolde.** Methods for re-using public gene expression data. Tartu, 2014, 121 p.
91. **Vladimir Šor.** Statistical Approach for Memory Leak Detection in Java Applications. Tartu, 2014, 155 p.
92. **Naved Ahmed.** Deriving Security Requirements from Business Process Models. Tartu, 2014, 171 p.
94. **Liina Kamm.** Privacy-preserving statistical analysis using secure multiparty computation. Tartu, 2015, 201 p.
100. **Abel Armas Cervantes.** Diagnosing Behavioral Differences between Business Process Models. Tartu, 2015, 193 p.
101. **Fredrik Milani.** On Sub-Processes, Process Variation and their Interplay: An Integrated Divide-and-Conquer Method for Modeling Business Processes with Variation. Tartu, 2015, 164 p.
102. **Huber Raul Flores Macario.** Service-Oriented and Evidence-aware Mobile Cloud Computing. Tartu, 2015, 163 p.
103. **Tauno Metsalu.** Statistical analysis of multivariate data in bioinformatics. Tartu, 2016, 197 p.
104. **Riivo Talviste.** Applying Secure Multi-party Computation in Practice. Tartu, 2016, 144 p.
108. **Siim Orasmaa.** Explorations of the Problem of Broad-coverage and General Domain Event Analysis: The Estonian Experience. Tartu, 2016, 186 p.
109. **Prastudy Mungkas Fauzi.** Efficient Non-interactive Zero-knowledge Protocols in the CRS Model. Tartu, 2017, 193 p.
110. **Pelle Jakovits.** Adapting Scientific Computing Algorithms to Distributed Computing Frameworks. Tartu, 2017, 168 p.
111. **Anna Leontjeva.** Using Generative Models to Combine Static and Sequential Features for Classification. Tartu, 2017, 167 p.
112. **Mozhgan Pourmoradnasseri.** Some Problems Related to Extensions of Polytopes. Tartu, 2017, 168 p.

113. **Jaak Randmets.** Programming Languages for Secure Multi-party Computation Application Development. Tartu, 2017, 172 p.
114. **Alisa Pankova.** Efficient Multiparty Computation Secure against Covert and Active Adversaries. Tartu, 2017, 316 p.
116. **Toomas Saarsen.** On the Structure and Use of Process Models and Their Interplay. Tartu, 2017, 123 p.
121. **Kristjan Korjus.** Analyzing EEG Data and Improving Data Partitioning for Machine Learning Algorithms. Tartu, 2017, 106 p.
122. **Eno Tõnisson.** Differences between Expected Answers and the Answers Offered by Computer Algebra Systems to School Mathematics Equations. Tartu, 2017, 195 p.

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

1. **Abdullah Makkeh**. Applications of Optimization in Some Complex Systems. Tartu 2018, 179 p.
2. **Riivo Kikas.** Analysis of Issue and Dependency Management in Open-Source Software Projects. Tartu 2018, 115 p.
3. **Ehsan Ebrahimi.** Post-Quantum Security in the Presence of Superposition Queries. Tartu 2018, 200 p.
4. **Ilya Verenich.** Explainable Predictive Monitoring of Temporal Measures of Business Processes. Tartu 2019, 151 p.